

# SOCIAL NETWORKS

Prof. Sudarshan Iyengar  
Computer Science and Engineering  
IIT Ropar



# INDEX

S. No	Topic	Page No.
	<b><i>Week 1</i></b>	
1	Introduction	1
2	Answer to the puzzle	12
3	Introduction to Python-1	20
4	Introduction to Python-2	44
5	Introduction to Networkx-1	73
6	Introduction to Networkx-2	84
7	Social Networks: The Challenge	130
8	Google Page Rank	136
9	Searching in a Network	139
10	Link Prediction	142
11	The Contagions	145
12	Importance of Acquaintances	147
13	Marketing on Social Networks	149
	<b><i>Week 2</i></b>	
14	Introduction to Datasets	151
15	Ingredients Network	158
16	Synonymy Network	162
17	Web Graph	165
18	Social Network Datasets	168
19	Datasets: Different Formats	171
20	Datasets : How to Download?	185
21	Datasets: Analysing Using Networkx	199
22	Datasets: Analysing Using Gephi	227
23	Introduction : Emergence of Connectedness	240
24	Advanced Material : Emergence of Connectedness	245
25	Programming Illustration : Emergence of Connectedness	253
26	Summary to Datasets	274
	<b><i>Week 3</i></b>	
27	Introduction	276
28	Granovetter's Strength of weak ties	279
29	Triads, clustering coefficient and neighborhood overlap	283
30	Structure of weak ties, bridges, and local bridges	293

31	Validation of Granovetter's experiment using cell phone data	300
32	Embeddedness	304
33	Structural Holes	310
34	Social Capital	313
35	Finding Communities in a graph (Brute Force Method)	321
36	Community Detection Using Girvan Newman Algorithm	337
37	Visualising Communities using Gephi	345
38	Tie Strength, Social Media and Passive Engagement	359
39	Betweenness Measures and Graph Partitioning	367
40	Strong and Weak Relationship - Summary	376

### ***Week 4***

41	Introduction to Homophily - Should you watch your company ?	383
42	Selection and Social Influence	392
43	Interplay between Selection and Social Influence	402
44	Homophily - Definition and measurement	417
45	Foci Closure and Membership Closure	431
46	Introduction to Fatman Evolutionary model	435
47	Fatman Evolutionary Model- The Base Code (Adding people)	439
48	Fatman Evolutionary Model- The Base Code (Adding Social Foci)	454
49	Fatman Evolutionary Model- Implementing Homophily	470
50	Quantifying the Effect of Triadic Closure	480
51	Fatman Evolutionary Model- Implementing Closures	484
52	Fatman Evolutionary Model- Implementing Social Influence	500
53	Fatman Evolutionary Model- Storing and analyzing longitudinal data	512

### ***Week 5***

54	Spatial Segregation: An Introduction	524
55	Spatial Segregation: Simulation of the Schelling Model	531
56	Spatial Segregation: Conclusion	536
57	Schelling Model Implementation-1(Introduction)	538
58	Schelling Model Implementation-2 (Base Code)	543
59	Schelling Model Implementation-3 (Visualization and Getting a list of boundary and internal nodes)	557
60	Schelling Model Implementation-4 (Getting a list of unsatisfied nodes)	572
61	Schelling Model Implementation-5 (Shifting the unsatisfied nodes and visualizing the final graph)	580
62	Chapter - 5 Positive And Negative Relationships (Introduction)	587
63	Structural Balance	589

64	Enemy'S Enemy Is A Friend	597
65	Characterizing The Structure Of Balanced Networks	605
66	Balance Theorem	610
67	Proof Of Balance Theorem	616
68	Introduction to positive and negative edges	621
69	Outline of implemantation	627
70	Creating graph, displaying it and counting unstable triangles	634
71	Moving a network from an unstable to stable state	646
72	Forming two coalitions	667
73	Forming two coalitions contd	678
74	Visualizing coalitions and the evolution	682

## ***Week 6***

75	The Web Graph	688
76	Collecting the Web Graph	694
77	Equal Coin Distribution	703
78	Random Coin Dropping	709
79	Google Page Ranking Using Web Graph	715
80	Implementing PageRank Using Points Distribution Method-1	718
81	Implementing PageRank Using Points Distribution Method-2	723
82	Implementing PageRank Using Points Distribution Method-3	734
83	Implementing PageRank Using Points Distribution Method-4	747
84	Implementing PageRank Using Random Walk Method -1	754
85	Implementing PageRank Using Random Walk Method -2	757
86	DegreeRank versus PageRank	766

## ***Week 7***

87	We Follow	771
88	Why do we Follow?	777
89	Diffusion in Networks	784
90	Modeling Diffusion	787
91	Modeling Diffusion (continued)	793
92	Impact of Communities on Diffusion	798
93	Cascade and Clusters	809
94	Knowledge, Thresholds and the Collective Action	817
95	An Introduction to the Programming Screencast (Coding 4 major ideas)	824
96	The Base Code	830
97	Coding the First Big Idea - Increasing the Payoff	842

98	Coding the Second Big Idea - Key People	858
99	Coding the Third Big Idea- Impact of Communities on Cascades	864
100	Coding the Fourth Big Idea - Cascades and Clusters	879

### ***Week 8***

101	Introduction to Hubs and Authorities (A Story)	891
102	Principle of Repeated Improvement (A story)	895
103	Principle of Repeated Improvement (An example)	899
104	Hubs and Authorities	903
105	PageRank Revisited - An example	909
106	PageRank Revisited - Convergence in the Example	913
107	PageRank Revisited - Conservation and Convergence	917
108	PageRank, conservation and convergence - Another example	919
109	Matrix Multiplication (Pre-requisite 1)	923
110	Convergence in Repeated Matrix Multiplication (Pre-requisite 1)	930
111	Addition of Two Vectors (Pre-requisite 2)	937
112	Convergence in Repeated Matrix Multiplication- The Details	943
113	PageRank as a Matrix Operation	952
114	PageRank Explained	956

### ***Week 9***

115	Introduction to Powerlaw	962
116	Why do Normal Distributions Appear?	969
117	Power Law emerges in WWW graphs	974
118	Detecting the Presence of Powerlaw	981
119	Rich Get Richer Phenomenon	986
120	Summary So Far	993
121	Implementing Rich-getting-richer Phenomenon (Barabasi-Albert Model)-1	996
122	Implementing Rich-getting-richer Phenomenon (Barabasi-Albert Model)-2	1002
123	Implementing a Random Graph (Erdos- Renyi Model)-1	1028
124	Implementing a Random Graph (Erdos- Renyi Model)-2	1031
125	Forced Versus Random Removal of Nodes (Attack Survivability)	1041

### ***Week 10***

126	Rich Get Richer - A Possible Reason	1054
127	Rich Get Richer - The Long Tail	1063
128	Epidemics- An Introduction	1071
129	Introduction to epidemics (contd..)	1079
130	Simple Branching Process for Modeling Epidemics	1082

131	Simple Branching Process for Modeling Epidemics (contd..)	1090
132	Basic Reproductive Number	1095
133	Modeling epidemics on complex networks	1101
134	SIR and SIS spreading models	1106
135	Comparison between SIR and SIS spreading models	1120
136	Basic Reproductive Number Revisited for Complex Networks	1126
137	Percolation model	1131
138	Analysis of basic reproductive number in branching model (The problem statement)	1137
139	Analyzing basic reproductive number 2	1141
140	Analyzing basic reproductive number 3	1148
141	Analyzing basic reproductive number 4	1152
142	Analyzing basic reproductive number 5	1156

### ***Week 11***

143	Small World Effect - An Introduction	1160
144	Milgram's Experiment	1168
145	The Reason	1174
146	The Generative Model	1180
147	Decentralized Search - I	1183
148	Decentralized Search - II	1188
149	Decentralized Search - III	1196

### ***Week 12***

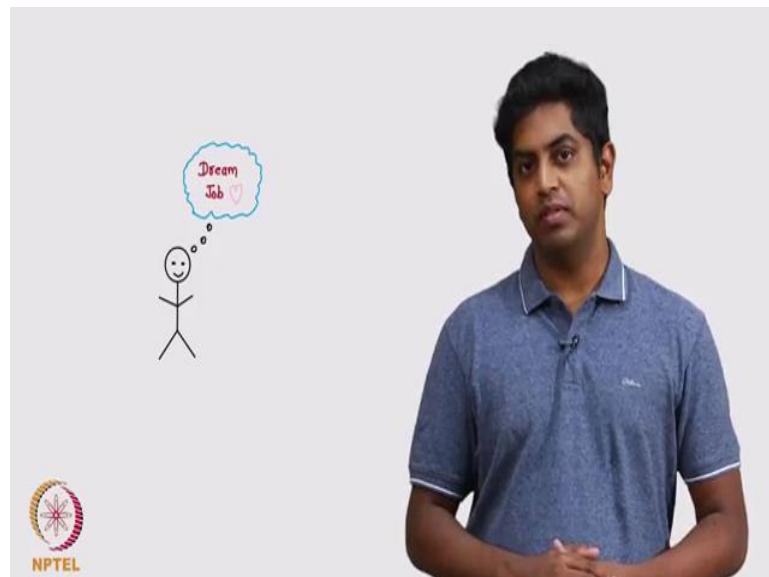
150	Programming illustration- Small world networks : Introduction	1209
151	Base code	1218
152	Making homophily based edges	1221
153	Adding weak ties	1234
154	Plotting change in diameter	1238
155	Programming illustration- Myopic Search : Introduction	1241
156	Myopic Search	1247
157	Myopic Search comparision to optimal search	1263
158	Time Taken by Myopic Search	1269
159	PseudoCores : Introduction	1281
160	How to be Viral	1286
161	Who are the right key nodes?	1290
162	finding the right key nodes (the core)	1292
163	Coding K-Shell Decomposition	1303
164	Coding cascading Model	1315

165	Coding the importance of core nodes in cascading	1330
166	Pseudo core	1343

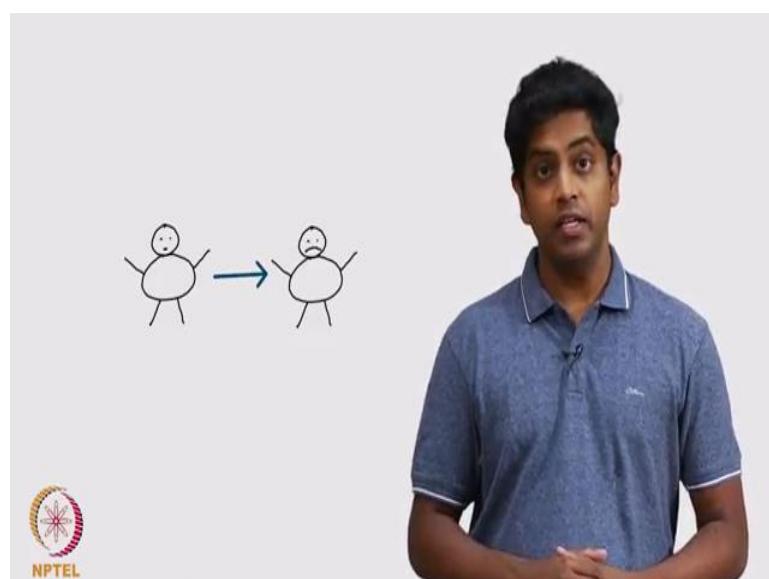
**Social Networks**  
**Prof S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture - 01**  
**Introduction to Social Networks**  
**Introduction**

(Refer Slide Time: 00:16)

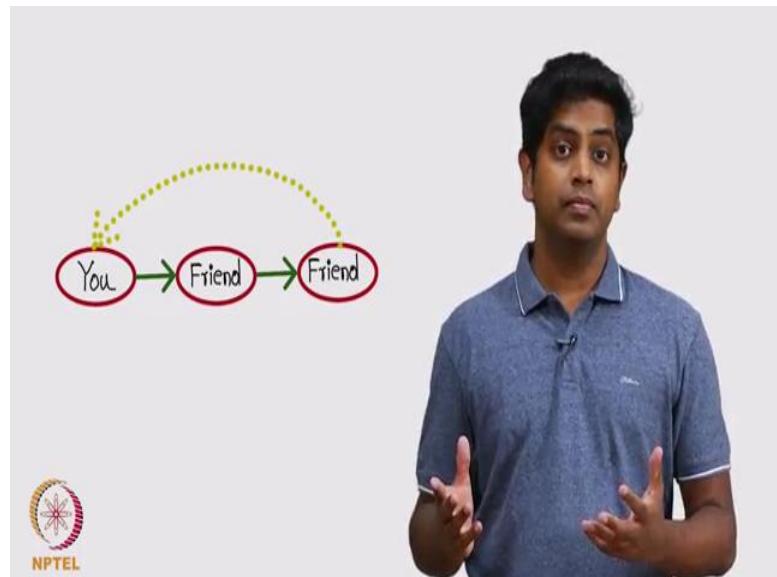


(Refer Slide Time: 00:20)



How do you find your dream job is obesity contagious? Do you think a friend's friend whom you do not know has any influence at all in your life?

(Refer Slide Time: 00:22)



(Refer Slide Time: 00:30)



And what has these questions to do with let say how google works? What are the commonalities across these questions? Are there any commonalities in the first place? Well yes, that is what makes the subject called social networks. So, we will be studying all these questions and more throughout the course without any further ado let us start off

with a nice question. So, we are going to watch a video clip right now, I am going to come back and then analyze what just happened in the video clip.

Hey.

Hey Ahmed, where were you? Class is about to start.

Leave it, do you even know Vardan and Simran are dating and for the Heaven's sake this is just the second day of our college.

How did you come to know about that?

You do not know? Everyone knows about it.

How is that even possible? Anamika told me about it yesterday and that too personally and I just told Harita about it.

Harita! Who is Harita? I do not know about her.

As far as I know she does not talk to many people here. It has been just one day since we have met and I am still wondering; how did you come to know about that?

Of course, not many of us know each other and we have met each other yesterday only and that too only few of us interacted.

I see some signs going around here.

Let me revise it. So, we are the bunch of people who have not met each other before and then met each other yesterday only and only few of us interacted and still.

And yet everyone knows about the news, Anamika told me personally.

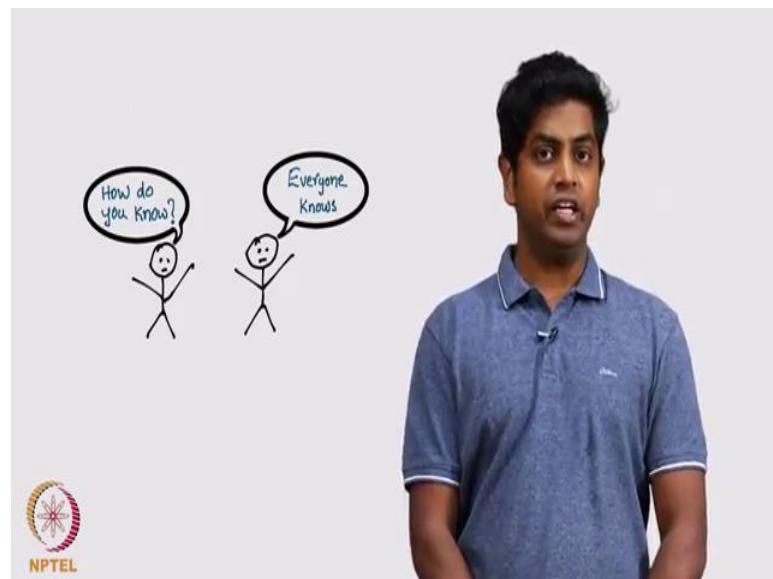
Indeed only few of us know each other still we are so connected.

(Refer Slide Time: 02:26)



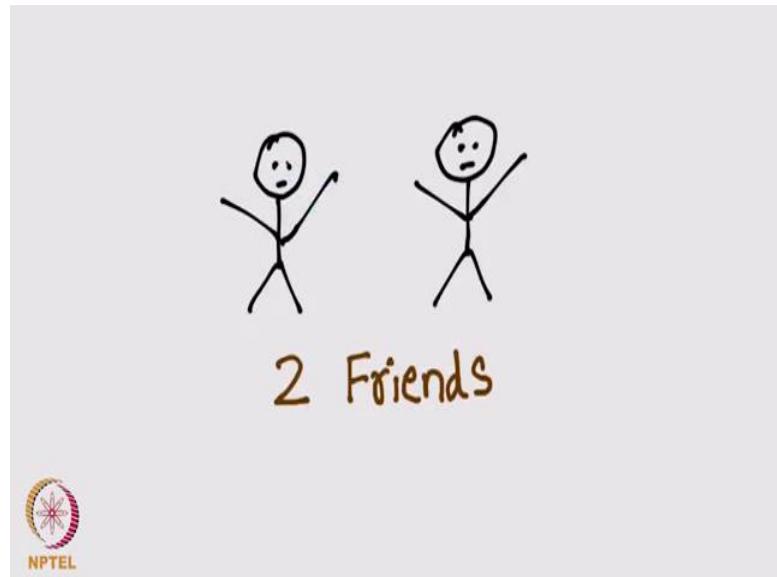
Did you see what just happened in the video clip, there were these 2 friends who newly joined this college and they are talking about a piece of gossip.

(Refer Slide Time: 02:37)



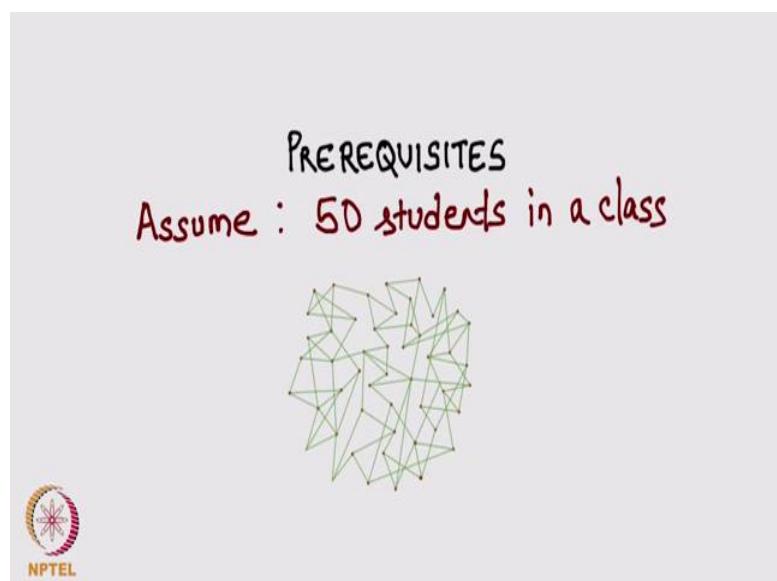
And a person is asking this question; how come you know of this while I thought nobody knows of this except me and in fact, that the person says the entire class knows of this what is so intriguing about this I actually find it intriguing for the following reason.

(Refer Slide Time: 03:11)



They are just less than one week into the class in the college and for everybody to know this piece of gossip they all must be friends with each other, is it not, without being friends with each other, who will come and tell each other about all these things? They may not even be Facebook friends this early, right. It has been less than a week since they have joined the college what is happening here, let us analyze.

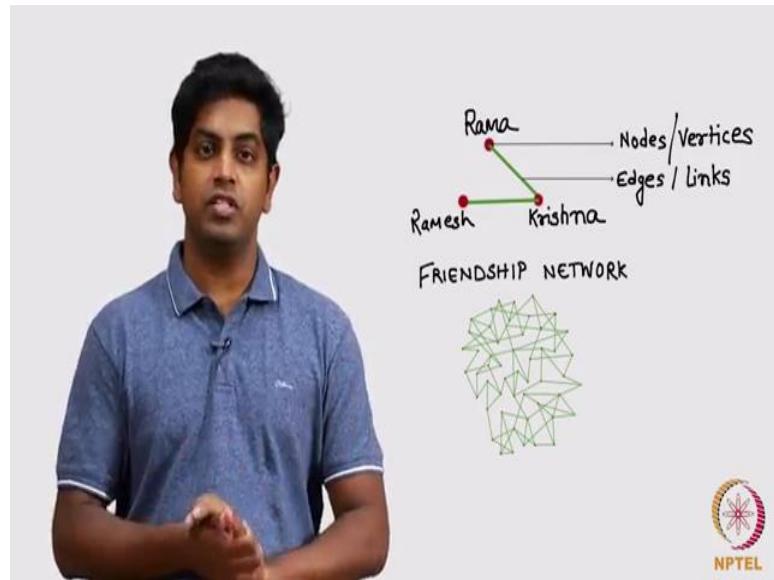
(Refer Slide Time: 03:30)



We need to develop a few pre requisites before we can answer this question. So, let me go slowly, the classroom let us say has some 50 people, 50 people may not be friends

with each other because as I told you it is just the first few days of the class, let me try modeling this these are the 50 friends let me use dots to denote these friends and the friendships between them let me draw a line to denote the friendship by this I mean.

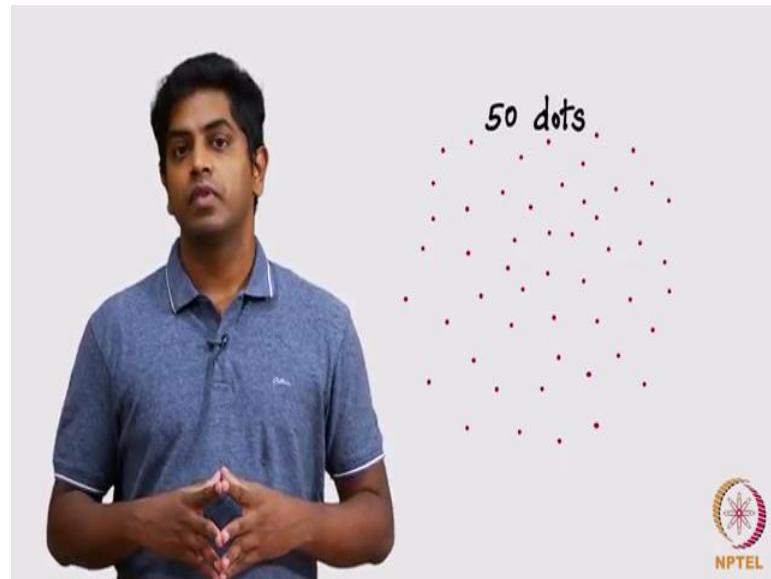
(Refer Slide Time: 04:09)



Assume Rama and Krishna are part of this classroom and they are friends with each other I put a line between them and Krishna and Ramesh are friends with each other I put this line between them and, but Rama and Ramesh are not friends. So, I do not put line between them and I develop the friendship network I call this friendship network.

It might look something like this some points denoting people and lines denoting friends. There are different ways in which people call this dots are called vertices or nodes in the subject and the lines are called edges or links. So, what we will be doing is we will we are going to use the jargon vertices or nodes for dots, edges or links for lines from now onwards throughout the course that is with the nomenclature.

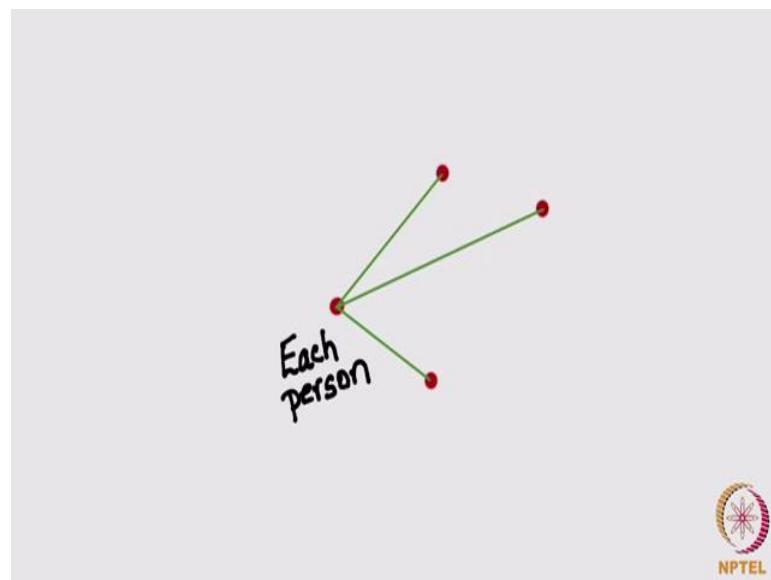
(Refer Slide Time: 05:11)



And now let me do a small experiment and observe what is happening here.

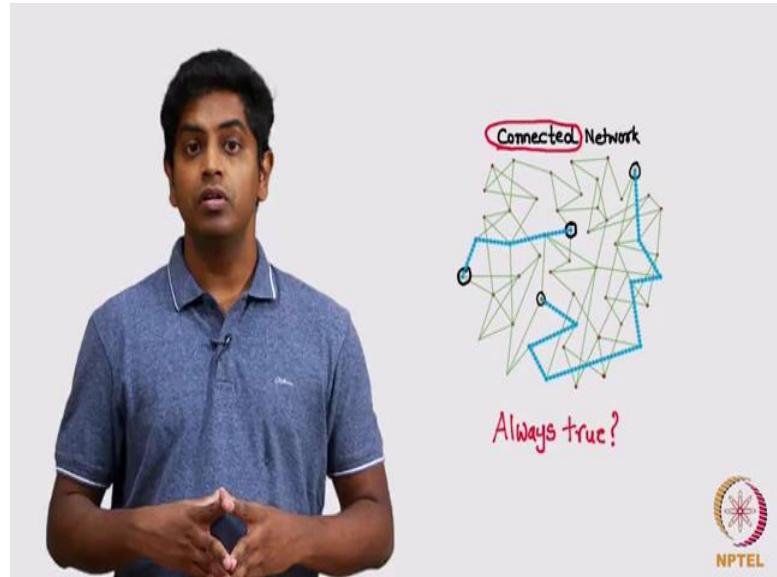
I will take 50 dots representing friends rather people in the class what I will do is let me take a guess first week of the program first week of the college and they are from different places from the country and it has only been first week.

(Refer Slide Time: 05:45)



So, let us say each person would have managed to have 3 friends on an average. So, what I will do I will take a person here and try to randomly pick 3 people and declare him as friends with these 3 people, I do it for every single person look at what is happening.

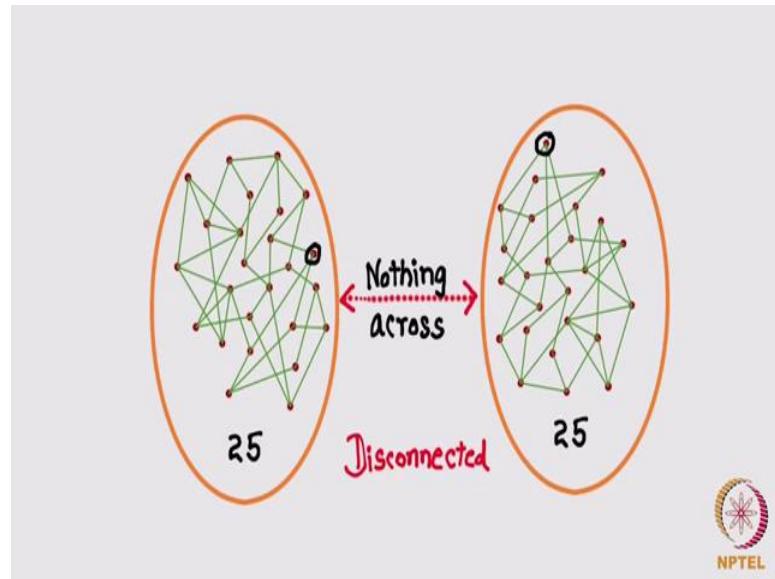
(Refer Slide Time: 05:58)



The final graph I am going to call it as structure a graph the final graph might look like this what is surprising about this graph? The surprising fact is that while a person can actually make all 50 people as his friends the entire class as his friends 49 to be precise excluding him. Let say he just makes 3 people as friends why because it is the beginning of the course and what do I observe here in this graph what is startling for me is that this graph this network is connected what do I mean by connected.

You see take any 2 people here in this graph there is a path that connects these 2 people this is strange is this always true not really.

(Refer Slide Time: 07:00)



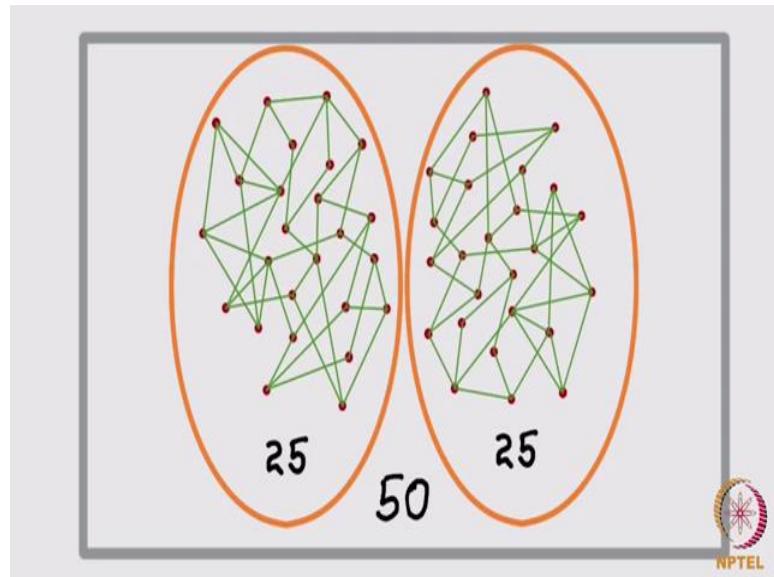
Observe this particular example where there are 50 people, the 20 people, this side 25 people that side they have some friendships within, but there is no friendships across this might also happen in such a case I do not call this graph connected.

(Refer Slide Time: 07:32)



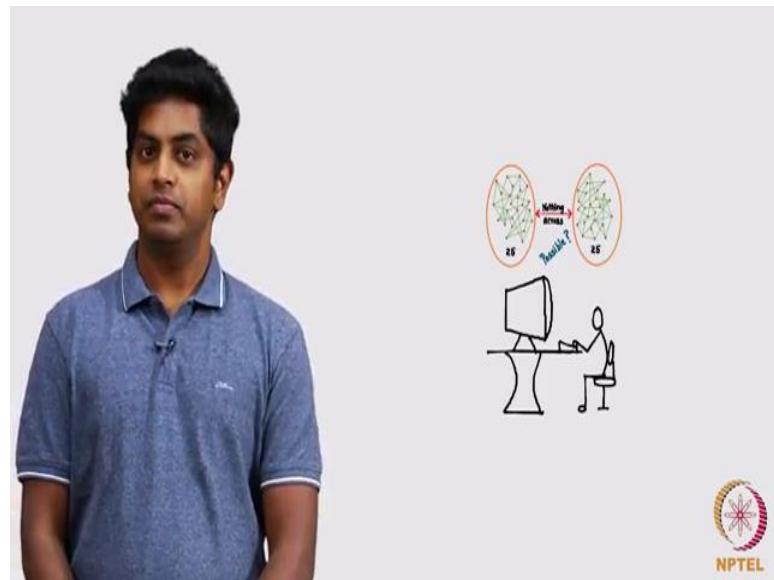
Why because I can take a person from this end and a person from that end there is not path connecting these 2 people, but then look at my previous graph on an average if I make 3 friends per person by picking these 3 people uniformly at random I observe that the graphs gets connected is this always true.

(Refer Slide Time: 07:47)



So, let us look at this network 25 people this side, 25 people that side as I told you and this side there is a network of connections and that side there is a network of connections remember here is a classroom of 50 people and there is a bifurcation of 25 this side, 25 that side.

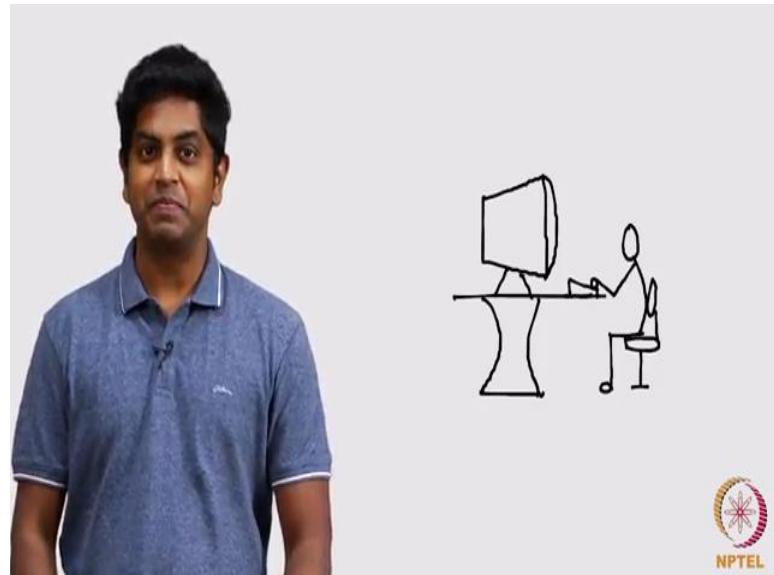
(Refer Slide Time: 08:11)



What do I mean by bifurcation? I mean if you look at the friendships between these people 25 people have friendships within them, 25 people have friendships within them

that side and a point to note is there is no friendship across my question was is this even possible.

(Refer Slide Time: 08:46)

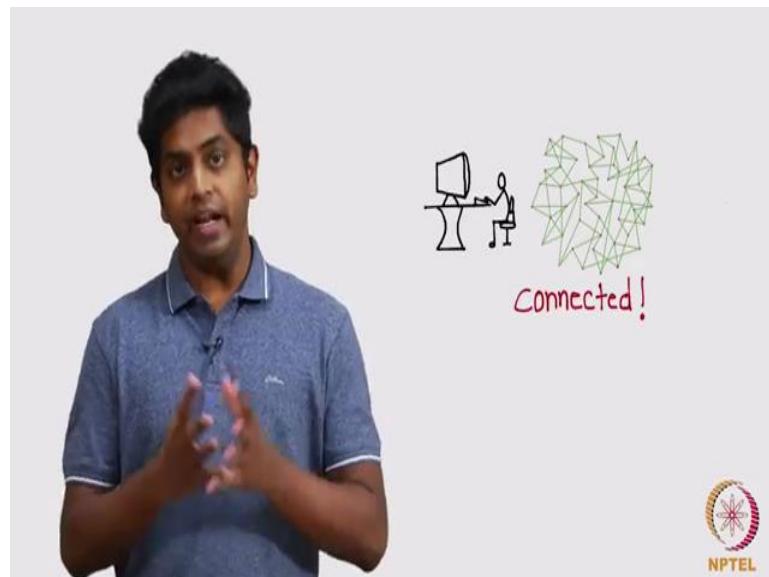


So, let me do one thing, I will now take a break, write a piece of program and get back to you people and tell you my observation of course, I will not show you the program, I will go do the program and come back and tell you the output of the program.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

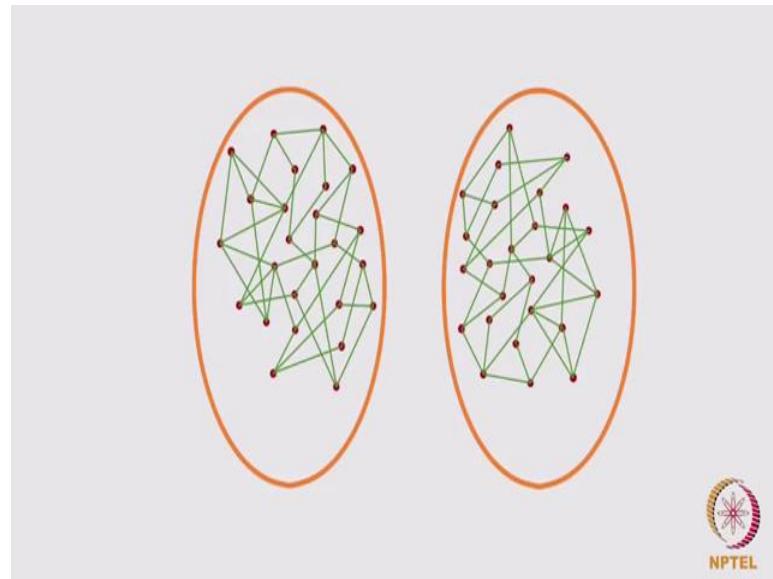
**Lecture - 02**  
**Introduction to Social Networks**  
**Answer to the puzzle**

(Refer Slide Time: 00:05)



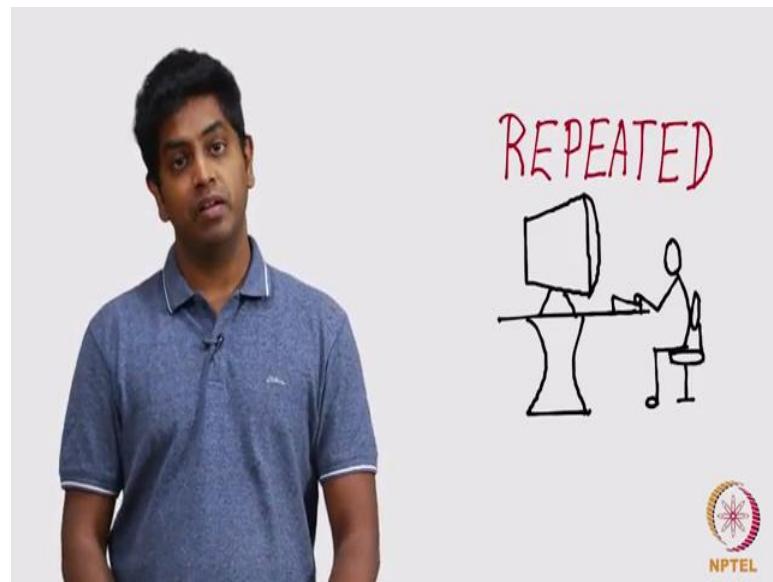
Fine, I am back. So, I did this piece of programming, what did I do? I took 50 dots, for each dot I randomly chose 3 people.

(Refer Slide Time: 00:21)



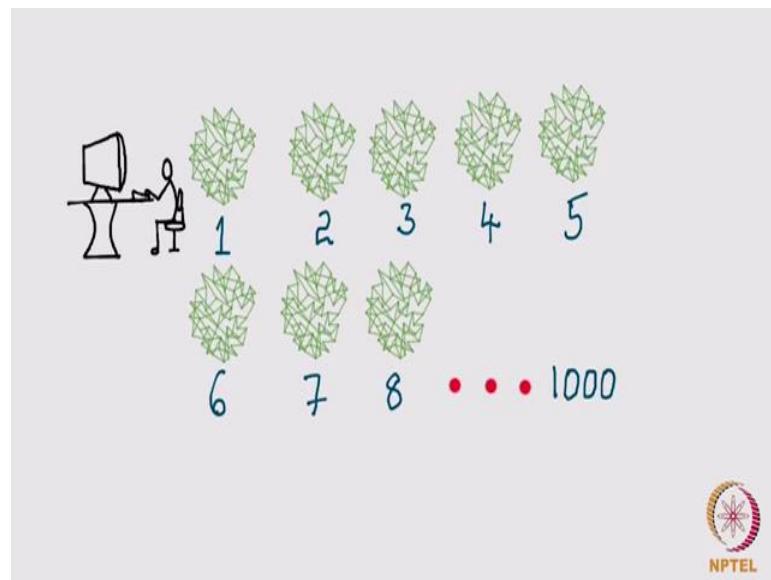
Whatever I told you just a while back I did I choose 3 people uniformly at random per person and made them friends. I did this for everyone and I saw the resultant network, it was connected by that I mean I did not see 2 components like this.

(Refer Slide Time: 00:45)



I did not see this at all, I only saw this. So, what did I do? I repeated my experiment, what do I mean by repeating the experiment?

(Refer Slide Time: 00:49)



Anyway I take a person and make 3 friends uniformly at random for every person, it turned out to be connected for the first time, I repeated this experiment for the second time, I saw it was connected, third time, connected, fourth connected, fifth, sixth, seventh, eighth, I repeated this a 1000 times my program always said it was connected.

(Refer Slide Time: 01:28)



Now, there are 2 possibilities possibility one is something is wrong with my programming I probably I am a bad programmer there is a error in the code bug in the code or there is something happening here. So, I am sure you people are confident with

the instructor that he knows how to write a piece of code. So, probably it is not the first one it is indeed the second one what is happening here let us recollect the question.

If you take a network of let say 50 people with each person having just a few friends the network turns out to be connected unbelievable whys is this true think for a minute.

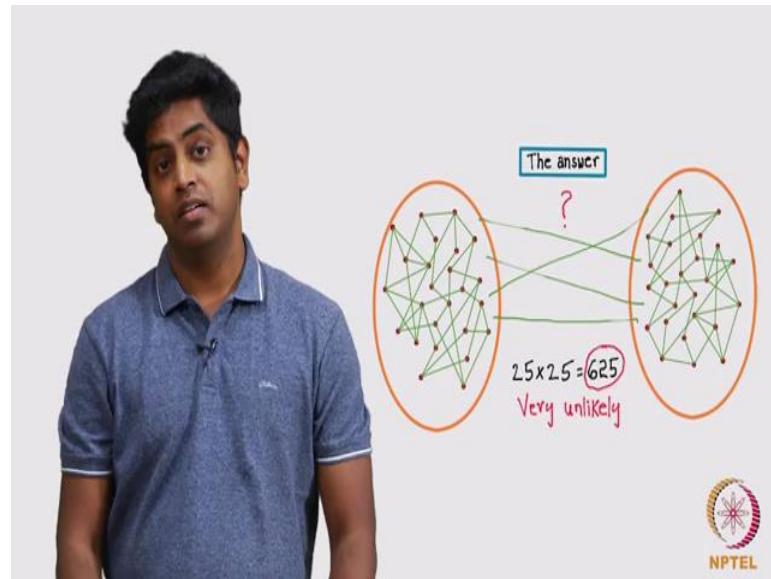
(Refer Slide Time: 02:07)



Well I can tell you the answer, the answer is pretty simple let us get back to this 25; 25 example.

The graph can have this kind of 2 partitions a class with 50 people can have such two partitions or maybe even three partitions or four partitions and so on.

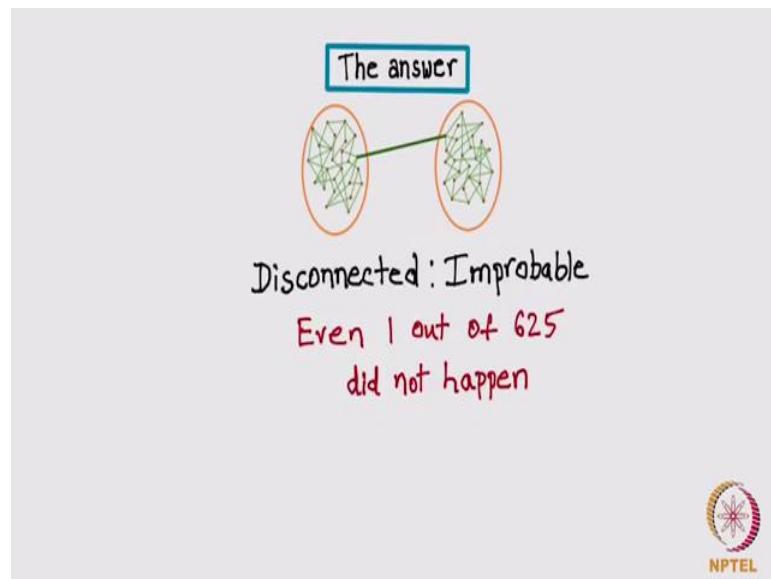
(Refer Slide Time: 02:34)



I think that is impossible why because 2 partitions if it exists 25, 25, 20, 30, whatever two partitions do you see what are the possible friendships between these 2 partitions 25, this side each one has the possibility of making 25 different friends and he chose not to fine maybe he does not like these 25 people.

Another person from this group had 25 possible friendships to make he chose not to so on, so on and so on. So, what do you observe there were roughly 25 into 25 possible friendships here that did not happen how likely is this.

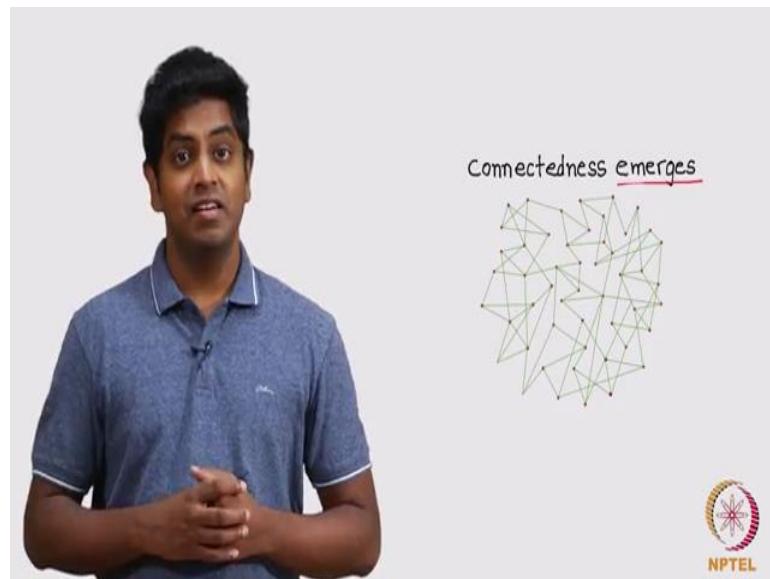
(Refer Slide Time: 03:23)



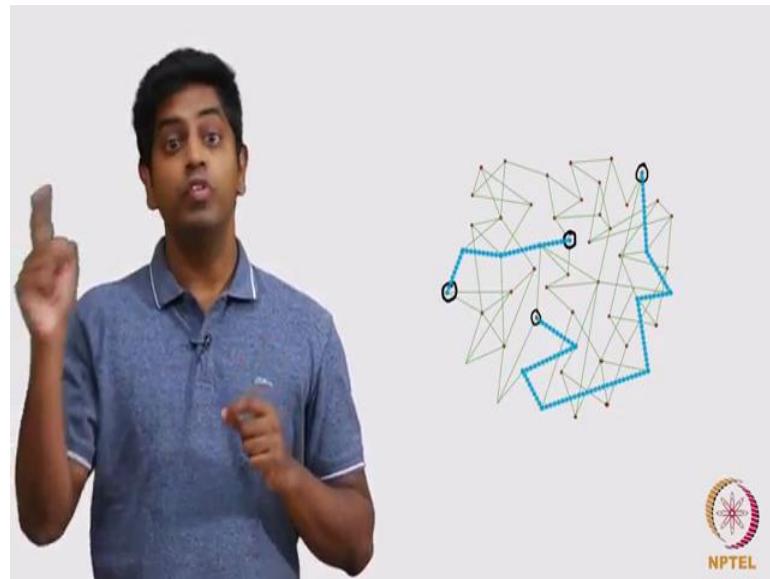
Intuitively very unlikely by that I mean even if one person on this side had friends, let us say a friend on that side over the graph would not look like this it would look like this with one connection, it would not have 2 clusters, 2 clusters would become one cluster out of 625 possible friendships even if one friendship got executed it will not have led to this bifurcation which means we all now we have developed this intuition that bifurcation is improbable.

Although possible with a very small probability I do not worry if you do not know about probability you just need to have this intuition that there are 625 possibilities out of which even one possibility did not get executed that is a very improbable event correct.

(Refer Slide Time: 04:29)

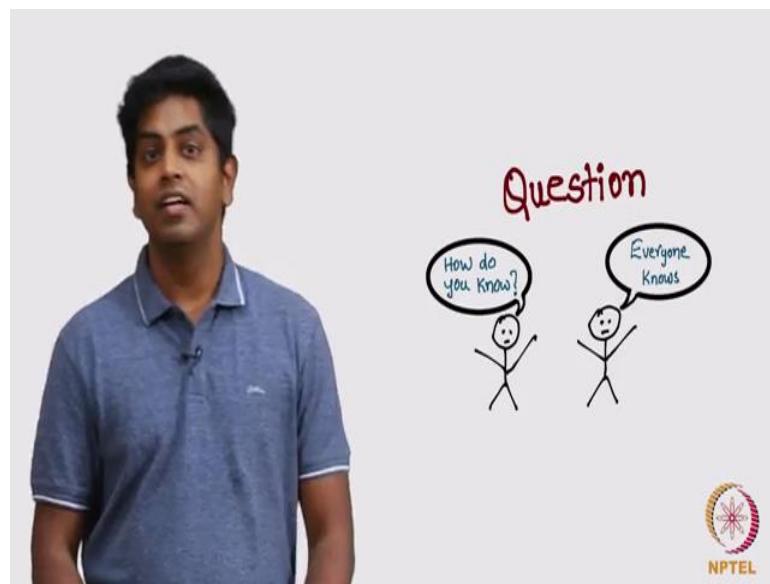


(Refer Slide Time: 04:38)



And that is the reason why in a classroom with 50 people there emerges what is called connectedness by that we mean any 2 people have a path between them which means let us get back to the question that we asked.

(Refer Slide Time: 04:43)



Remember the clip, 2 friends talking to each other and they were wondering how come everyone knows this a piece of juicy gossip that is because even with a fact that the class is only one week old, they are just into the college still with each person having 2 to 3 friends, the graph becomes connected and a piece of gossip. Such as this that that are

friends are discussing can spread like wild fire and if it starts from one node it can reach to all nodes simply because it is connected and not disconnected.

**Social Networks**  
**Prof S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

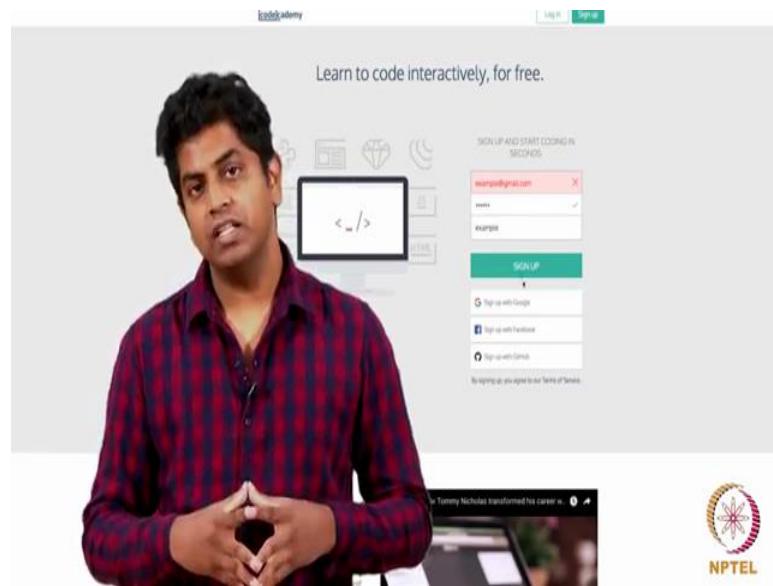
**Lecture - 03**  
**Introduction to Social Networks**  
**Introduction to Phyton-1**

Hi, in this module, we will be introducing you to python. Python is a very powerful programming language yet very simple to understand, very easy to use and it is an open source package. It is a scripting language which has a whole lot of APIs online and it is very easy for you to install these APIs and work with python this is one major reason why I thought you will use python for our network analysis related work social network analysis related work.

So, in this session I will be showing you some quick ways of learning python; quick things that you must know about python and you can get started. It is not required that you undergo a complete course in python and then start using python 3-4 things and you will and you are on you can start using python.

So, my personal recommendation, my personal favorite is this code academy dot com where you can go create a user name then start learning python, it has very nice exercise unless you solve the exercise, you cannot go further and these exercises are in increasing difficulty.

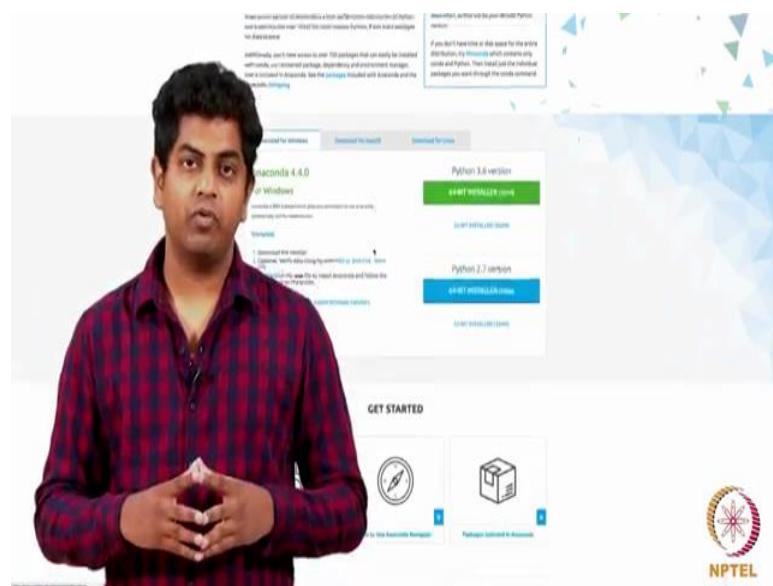
(Refer Slide Time: 01:00)



I strongly suggest that you finish a few lessons there and get yourself familiarized with python.

And or you can just look at what we do on the screen cast which I am going to show you right now and learn python parallelly. Now a word on installing python, if you are using one 2 the process is pretty straight forward on MAC again it is again very easy to install.

(Refer Slide Time: 01:41)



But if you are using windows, I would suggest that you download what is called anaconda which is a big package which includes all the scientific programming

requirements using python and once installed I think you are all done, but one o one 2 one MAC it is a very straight forward process.

For our demonstrations we are using a MAC machine and this is the terminal that I have opened on a MAC machine. So, what I do is I type ipython. So, this ipython is interactive python which opens a shell like this where I can easily type the code.

(Refer Slide Time: 02:15)

```
Type "copyright", "credits" or "license" for more information.  
IPython 5.2.0 -- An enhanced Interactive Python.  
?          -> Introduction and overview of IPython's features.  
%quickref -> Quick reference.  
help       -> Python's own help system.  
object?   -> Details about 'object', use 'object??' for extra details.  
  
In [1]: a=2  
In [2]: b=5  
In [3]: c='sudarshan'  
In [4]: a,b=b,a  
In [5]: print a  
5  
In [6]: print b  
2  
In [7]: a,b=b,a  
In [8]: print a  
2  

```



A note on IPython it is very easily installable on o 1 to 1 MAC, but on windows as I told you need anaconda in which ipython is a built-in package.

In ipython let me try showing you a few things  $a = 2$ , this is basically python with a interactive shell and then let say  $b = 5$ . So, unlike  $c$  you do not have any declarations that are required for variables you just say  $a$  equals to  $b$  equals 5 and it assigns  $a$  to 2 and  $b$  to 5 and if I say  $c$  equals let say Sudarshan. Now  $c$  will be string type variable automatically and look at the ease with which I can swap to variables  $a$  comma  $b$  is equal to  $b$  comma  $a$  and then if I print  $a$  and print  $b$  the values are swapped, let me swap them once again back to normal if I print  $a$  it will be 2 if I print  $b$  it will be 5.

We will now see how one can use for loop while loop and if loop in python. It is pretty straight forward I will still give you an illustration I would suggest that you try getting your hands on with these things so that you get familiarized with it, so a for loop works like this.

(Refer Slide Time: 03:41)

```
In [11]: for i in range(10):
...:     print i
...:
0
1
2
3
4
5
6
7
8
9

In [12]: k=10

In [13]: while (k<=20):
...:     print k
...:     k=k+1
```



Say for *i* in range 10 I say print *I*, it is simply a prints *i* from 0 to 9, this is how a for loop works, a while loop also works in a similar way if I say *k* equals let us say 10 and I say while *k* is less than or equal to 20 do print *k* and then increment *k* just starts *k* with 10 and goes on till 20.

(Refer Slide Time: 04:13)

```
...:     print k
...:     k=k+1
...:

10
11
12
13
14
15
16
17
18
19
20

In [14]: print k
21

In [15]: if (k>20):
...:     print "The value of k is greater than 20"
...:
The value of k is greater than 20

In [16]: if (k>25):
...:     print "something something"
...: else:
...:     print "the statement is false"
...:
the statement is false

In [17]:
```



So, the value of *k* now is 21, what I will do is I will illustrate if loop now using this *k* value, if *k* is greater than 20 which by the way is the case *k*'s value seems to be 21, I will say print the value of *k* is greater than 20.

So, this is true. So, it displays the value of k is greater than 20. So, if I say if k is greater than 25 then print something something I am illustrating the usage of else here I will say else, print the statement is incorrect is false let say is false let say the statement is false right this is how if else loop works. So, we now just now say how for loop works while loop works and if loop works; so just that you people play around with it so that you get familiarized with it.

A big advantage of using a scripting language like python is it comes with a whole lot of library functions I am now going to show you one such library function which will be using a whole lot throughout our case.

(Refer Slide Time: 05:41)

```
In [18]: import random
In [19]: random.randrange(1,10)
Out[19]: 4
In [20]: random.randrange(1,10)
Out[20]: 5
In [21]: random.randrange(1,10)
Out[21]: 6
In [22]: random.randrange(1,10)
Out[22]: 5
In [23]: random.randrange(1,10)
Out[23]: 1
In [24]: random.randrange(1,10)
Out[24]: 2
In [25]: random.randrange(1,10)
Out[25]: 7
In [26]: random.randrange(1,10)
Out[26]: 9
In [27]: random.randrange(1,4)
Out[27]: 2
In [28]:
```



Let me use a library function called random this is how you input if you load the library function to your memory, you say import space random and then in this function you can call in this library you can call a particular function let say rand range one comma 10 what does this do this simply gives you a random number from 1 to 10, let me execute this command once again.

(Refer Slide Time: 06:39)

```
In [28]: random.randrange(1,4)
Out[28]: 1

In [29]: random.randrange(1,4)
Out[29]: 2

In [30]: random.randrange(1,4)
Out[30]: 2

In [31]: random.randrange(1,4)
Out[31]: 2

In [32]: random.randrange(1,4)
Out[32]: 2

In [33]: random.randrange(1,4)
Out[33]: 1

In [34]: random.randrange(1,4)
Out[34]: 3

In [35]: random.randrange(1,4)
Out[35]: 2

In [36]: random.randrange(1,4)
Out[36]: 3

In [37]: random.randrange(1,4)
Out[37]: 3

In [38]: 
```



You press the up arrow in ipython shell and the entire things comes press enter, you will get another random number from the range 1 to 10 again you see I am getting random numbers from 1 to 10.

(Refer Slide Time: 06:44)

```
In [38]: random.randrange(1,4)
Out[38]: 1

In [39]: random.randrange(1,4)
Out[39]: 2

In [40]: random.randrange(1,4)
Out[40]: 2

In [41]: random.randrange(1,4)
Out[41]: 3

In [42]: random.randrange(1,4)
Out[42]: 3

In [43]: random.randrange(1,4)
Out[43]: 2

In [44]: random.randrange(1,4)
Out[44]: 2

In [45]: random.randrange(1,4)
Out[45]: 2

In [46]: random.randrange(1,4)
Out[46]: 1

In [47]: random.randrange(1,5)
Out[47]: 4

In [48]: 
```



Let me now change this instead of 10, I will make it 4, this will give you random numbers from 1 to 4, 1, 2, 3 and so on.

(Refer Slide Time: 06:53)

```
In [46]: random.randrange(1,4)
Out[46]: 1

In [47]: random.randrange(1,5)
Out[47]: 4

In [48]: random.randrange(1,5)
Out[48]: 1

In [49]: random.randrange(1,5)
Out[49]: 2

In [50]: random.randrange(1,5)
Out[50]: 1

In [51]: random.randrange(1,5)
Out[51]: 4

In [52]: random.randrange(1,5)
Out[52]: 2

In [53]: random.randrange(1,5)
Out[53]: 4

In [54]: random.randrange(1,5)
Out[54]: 4

In [55]: random.randrange(1,5)
Out[55]: 3

In [56]: random.randrange?¶
```



Yeah, you will get 4 2 as you keep trying or maybe it is only till yeah it is only till. So, you will not get 5 you will only get the number just before. So, random dot rand; rand range I do not know what it does. So, what I do is random dot rand range question mark.

(Refer Slide Time: 07:11)

```
Out[56]: 3

In [56]: random.randrange?
Signature: random.randrange(start, stop=None, step=1, _int=<type 'int'>, _maxwidth=9007199254740992L)
Docstring:
Choose a random item from range(start, stop[, step]).

This fixes the problem with randint() which includes the
endpoint; in Python this is usually not what you want.
File:      /usr/local/Cellar/python/2.7.13/Frameworks/Python.framework/Versions/2.7/lib/python2.7/random.py
Type:     instancemethod

In [57]: random.randrange(1,100)
Out[57]: 38

In [58]: random.randrange(1,100)
Out[58]: 97

In [59]: random.randrange(1,100)
Out[59]: 21

In [60]: random.randrange(1,100)
Out[60]: 6

In [61]: random.randrange(1,100)
Out[61]: 79

In [62]: random.random()
Out[62]: 0.1938257517766626

In [63]: random.ra?¶
```



This will tell me everything of what it does as you can see choose a random item from the range start till stop the stop is mostly one before the numbers specified here.

So, if I say random rand range one comma hundred it will give me a random number from 1 to 100, in case I say random dot random once again this is another library

function which gives me a random number between 1 0 to 1 see it gave me a random number between 0 to one let me type that once again.

(Refer Slide Time: 07:49)

```
In [63]: random.random()
Out[63]: 0.16229121181178352

In [64]: random.random()
Out[64]: 0.4509053996339152

In [65]: random.random()
Out[65]: 0.5293564813083772

In [66]: random.random()
Out[66]: 0.6486588462492465

In [67]: random.random()
Out[67]: 0.28535819271306043

In [68]: random.random()
Out[68]: 0.553645513203006

In [69]: random.random()
Out[69]: 0.008503050393502853

In [70]: random.random()
Out[70]: 0.160527690736800528

In [71]: random.random()
Out[71]: 0.87985455264565833

In [72]: random.random()
Out[72]: 0.2600508367747828

In [73]:
```



Another random number between 0 to 1, another random number between 0 to 1 so on and so forth. So, basically when I am doing this I am pressing my up arrow and pressing enter.

Up arrow just repeats the previous command all right if you want to see what all you can do with the random library function.

(Refer Slide Time: 08:02)

```
In [78]: random.randint(1,5)
Out[78]: 2

In [79]: random.randint(1,5)
Out[79]: 3

In [80]: random.randint(1,5)
Out[80]: 4

In [81]: random.randint(1,5)
Out[81]: 2

In [82]: random.randint(1,5)
Out[82]: 4

In [83]: random.randint(1,5)
Out[83]: 4

In [84]: random.randint(1,5)
Out[84]: 4

In [85]: random.randint(1,5)
Out[85]: 3

In [86]: random.randint(1,5)
Out[86]: 5

In [87]: random.randint(1,5)
Out[87]: 1

In [88]:
```



Just put a dot after random and press tab you will get all possibilities and you can use the arrow mark to go around and see what does what and there is a whole lot of functions written here.

Let us look at yet another function random rand int. So, what does it do put a question mark you will get to know that it returns a random integer in the range a comma b including both end points. So, let us try that random rand int 1 comma 5 gives you 2 so on and so forth. Somehow strangely 5 is not coming, yeah 5 came right now. So, it is between 1 to 5, but rand range is different as you saw before.

(Refer Slide Time: 08:58)

```
In [2]: L=[]  
In [3]: L.append(2)  
In [4]: print L  
[2]  
In [5]: L.append(10)  
In [6]: print L  
[2, 10]  
In [7]: L.append(100)  
In [8]: print L  
[2, 10, 100]  
In [9]: L.append(7)  
In [10]:
```



We will now look at what are lists in python, lists in python as I said are very similar to arrays and we will see how to use them. So, initially what I will do right now is I will declare lists called L, this is how you declare it L equals open bracket close bracket and then you say L append you put some number 2 print L, L will have that number 2, right, Now let me say L append 10, now to the existing list L, the number 10 gets appended if you say print L, we will get 2 comma 10 so on, I keep adding and then if I print you will see 2, 10, 100 and let us say I will append 7 print L, 7 gets appended so on and so forth.

(Refer Slide Time: 09:49)

```
In [10]: print L  
[2, 10, 100, 7]  
  
In [11]: L.append(77)  
  
In [12]: L.append(14)  
  
In [13]: print L  
[2, 10, 100, 7, 77, 14]  
  
In [14]: L.sort()  
  
In [15]: print L  
[2, 7, 10, 14, 77, 100]  
  
In [16]: L.reverse()  
  
In [17]: print L  
[100, 77, 14, 10, 7, 2]  
  
In [18]:
```



Best part is you can append whatever you want including character string; character string and whatever you want any data type now look at his let me add more numbers to this 77 L append 14 and then print L you have these numbers.

Sorting this is very easy simply say L dot sort and it will sort L as you see L is now sorted if you want to reverse L you simply say L dot reverse see what happens to this see it got reversed descending order.

(Refer Slide Time: 10:37)

```
In [17]: print L  
[100, 77, 14, 10, 7, 2]  
  
In [18]: L.remove?  
Docstring:  
L.remove(value) -- remove first occurrence of value.  
Raises ValueError if the value is not present.  
Type: builtin_function_or_method  
  
In [19]: L.remove(77)  
  
In [20]: print L  
[100, 14, 10, 7, 2]  
  
In [21]: import random  
  
In [22]: for i in range(100):  
...:     L.append(random.random())  
...:  
  
In [23]: print
```



What I can you do with this sort and reverse is what I told you what all can you do simply put a L dot and then press tab key and you will see all possibilities append count extend index insert pop remove. So, if you want to know what remove does type remove and put a question mark and it will tell you remove first occurrence of the value which means if I say L remove 77 it will remove 77 from the list as you can see 77 is here in this list let see if it removes or not I have removed 77 print L yes 77 is removed now.

So, this is about list let me type something non trivial and show you what all you can do with the list we use random function rights. So, let us import random library function and then I say for I in range hundred L append random.

(Refer Slide Time: 11:39)

```
...:     L.append(random.random())
...:
In [23]: print L
[100, 14, 10, 7, 2, 0.685339805366092, 0.125410889316928, 0.7842506934157
13, 0.05014644374059718, 0.8658827151241583, 0.3202467685987337, 0.849887
7583692054, 0.5804410422581844, 0.9540315746081519, 0.9788101296484397, 0
.1329822144851911, 0.848302426347616, 0.09291767291572695, 0.247053062469
55128, 0.33483156531997227, 0.35592939842000815, 0.15710792844710963, 0.4
2730292354472943, 0.5282987519218565, 0.09059016584353952, 0.066155871805
60356, 0.14721621972221832, 0.2226049164442757, 0.5413415984599751, 0.130
4913335642356, 0.19876030769458308, 0.8590282219070453, 0.798714152662500
8, 0.24865491139255336, 0.7459386748170106, 0.06846366618862954, 0.812757
415529382, 0.31415912344242247, 0.19261165481557074, 0.9287215875214563,
0.7658444903182783, 0.7633341081855114, 0.4731736376751251, 0.32668111784
128384, 0.003873879507955258, 0.5662014343335001, 0.7087667353595755, 0.9
329704065004019, 0.9721336218793493, 0.018124606845428604, 0.846376578428
8965, 0.45703182530416087, 0.5814387051879943, 0.2240540887064466, 0.4435
6453843749843, 0.39601072779884894, 0.6708173254167811, 0.523846645877839
2, 0.4000649769590148, 0.38577129924534825, 0.5648017599825583, 0.682869
945206266, 0.8691258963065251, 0.9445264376006239, 0.5745938067741007, 0.]
```

So, which means it appends 100 random numbers to L and L you see a lot of random numbers that apart from the initial 114 10 7 2 that we appended, we now have 100 random numbers appended.

(Refer Slide Time: 11:58)

```
In [25]: L=[]  
In [26]: print L  
[]  
In [27]: for i in range(40):  
...:     L.append(random.randint(1,365))  
...:  
In [28]: print L  
[210, 250, 94, 201, 187, 259, 158, 323, 236, 185, 100, 75, 139, 46, 198,  
192, 145, 357, 256, 39, 241, 222, 22, 209, 101, 173, 353, 316, 339, 10, 2  
5, 359, 89, 348, 175, 70, 111, 230, 189, 267]  
In [29]: L.sort()  
In [30]: print L
```



So let us do a small exercise let me clear the screen this is the command list for clearing the screen what does this do else I will reinitialize L. So, that whatever was present when L is lost, now I will do for I in range 40 L append random rand int 1 comma 365, what am I doing here? I am picking data of birth of a person let say if he was born on February first then you assign the number 32 to him if he was born on January fifth, you assign the number 5 to him. So, based on which day he was born you assign a number from 1 to 365 to that person pick forty people and assign this date of birth randomly to each person.

The point I am trying to make is I am I have now created a list L which comprises of 40 different birthdays dd, mm, day, month, only not the year, 40 different birthdays of people, let me see how it looks like print L has 40 different numbers from 1 to 365 let me sort this let me print this.

(Refer Slide Time: 13:16)

```
In [30]: print L
[10, 22, 25, 39, 46, 70, 75, 89, 94, 100, 101, 111, 139, 145, 158, 173, 1
75, 185, 187, 189, 192, 198, 201, 209, 210, 222, 230, 236, 241, 250, 256,
259, 267, 316, 323, 339, 348, 353, 357, 359]

In [31]: L=[]

In [32]: for i in range(60):
...:     L.append(random.randint(1,365))
...:

In [33]: print L
[249, 279, 351, 214, 354, 207, 3, 142, 278, 364, 37, 268, 336, 136, 72, 2
11, 274, 337, 337, 358, 61, 343, 15, 330, 188, 271, 297, 318, 317, 363, 3
31, 202, 12, 147, 152, 139, 225, 74, 132, 103, 76, 153, 140, 261, 191, 25
3, 143, 85, 143, 327, 179, 305, 33, 338, 119, 39, 274, 300, 226, 224]

In [34]: L.sort()

In [35]: pri
```



So, you see there are all these numbers and there are no repetitions I do not see any repetition here correct there are no repetitions perfect.

Now, what if I were to do the same thing like say I reinitialize for I in range 60 I say L append random rand int 1 comma 365 print l. So, I will again sort this.

(Refer Slide Time: 13:59)

```
In [35]: print L
[3, 12, 15, 33, 37, 39, 61, 72, 74, 76, 85, 103, 119, 132, 136, 139, 140,
142, 143, 143, 147, 152, 153, 179, 188, 191, 202, 207, 211, 214, 224, 22
5, 226, 249, 253, 261, 268, 271, 274, 274, 278, 279, 297, 300, 305, 317,
318, 327, 330, 331, 336, 337, 337, 338, 343, 351, 354, 358, 363, 364]

In [36]: L=[]

In [37]: for i in range(60):
...:     L.append(random.randint(1,365))
...:

In [38]: print L
[287, 232, 271, 173, 74, 83, 239, 150, 147, 199, 351, 206, 281, 315, 301,
154, 178, 246, 307, 103, 197, 152, 18, 75, 65, 53, 318, 285, 148, 160, 2
19, 91, 301, 218, 311, 140, 338, 168, 30, 89, 344, 298, 353, 178, 128, 36
3, 173, 206, 311, 241, 233, 164, 187, 287, 276, 249, 306, 307, 297, 303]

In [39]: L.sort()

In [40]: print L
```



So, that I can check for repetitions if an I will see if there is anything repeated 3, 12, 15, 33, let us see if there is any repetition I do not see a repetition yeah I see a repetition look at this 143 is picked twice, 153 and so on is there any other repetition I do not see any

other repetition 274 is another repetition correct and then any other repetition that you can spot let me check 3; 3, 7 is another repetition there are 3 repetitions. So, fine there were 3 repetitions, but is it coincidental. So, what do I do?

Let me repeat this experiment L equals initialize once again for say for I in range 60 L append random rand int 1 comma 365 then you print L it is; obviously, not sorted. So, you sort it and then print L you will see for 40.

(Refer Slide Time: 15:00)

```
19, 91, 301, 218, 311, 140, 338, 168, 30, 89, 344, 298, 353, 178, 128, 36
3, 173, 206, 311, 241, 233, 164, 187, 287, 276, 249, 306, 307, 297, 303]

In [39]: L.sort()

In [40]: print L
[18, 30, 53, 65, 74, 75, 83, 89, 91, 103, 128, 140, 147, 148, 150, 152, 1
54, 160, 164, 168, 173, 173, 178, 178, 187, 197, 199, 206, 206, 218, 219,
232, 233, 239, 241, 246, 249, 271, 276, 281, 285, 287, 287, 297, 298, 30
1, 301, 303, 306, 307, 307, 311, 311, 315, 318, 338, 344, 351, 353, 363]

In [41]: print L
[18, 30, 53, 65, 74, 75, 83, 89, 91, 103, 128, 140, 147, 148, 150, 152, 1
54, 160, 164, 168, 173, 173, 178, 178, 187, 197, 199, 206, 206, 218, 219,
232, 233, 239, 241, 246, 249, 271, 276, 281, 285, 287, 287, 297, 298, 30
1, 301, 303, 306, 307, 307, 311, 311, 315, 318, 338, 344, 351, 353, 363]

In [42]: len(L)
Out[42]: 60

In [43]:
```



Picking there was no repetition for 60 there was I am repeating the experiment to see if it is true that you will always get repetitions there is a repetition once again once again 173, 173, 178, 178, second repetition right and then 206, 206, another repetition any other repetition let see 287, 287, third repetition, 301, 301, 4 repetitions in this case 301 and 301, excellent, 307, 307, 5 repetitions 6 repetitions, 311 and so on, correct.

So, what did we just observe we observed that by just taking list L with how many entries? Let us say the; I just printed L before let me again print it. So, there were len of L tells you the length of L number of elements in L. So, there are 60 elements because of a for loop it is 60 elements and if you pick randomly some 60 people you will have couple of them sharing their birthdays this is always true almost always true with a high probability when you pick 60 people from randomly and ask them for their birthdays you will have 2 people with the same birthday, what does that mean?

You go to a classroom randomly with some 60 students and ask them for their birthdays you will see 2 people with the same birthday in that class walk in to any classroom, this is almost always true, this is called the birthday paradox this is not a command by the way in python I am just writing it to illustrate it. So, a birthday paradox you can just google for it for more information a very interesting fact that if you pick some 40 to 60 people 40, it is rare, 60 definitely if you pick definitely you will observe 2 people with the same birthday we just took a code and observed it right now like.

That is with less as you would have seen the course homepage the prerequisites for this course is simply a first course in programming I suppose you all have studied programming in one form or the other one of the most powerful facility that you get in programming is your ability to write functions and python has a very easy way in which one can write functions we are going to illustrate that right now.

(Refer Slide Time: 17:41)

```
In [109]: def sumup():
...:     a=random.random()
...:     b=random.random()
...:     return a+b
...:

In [110]: sumup()
Out[110]: 1.1412489051246113

In [111]: sumup()
Out[111]: 1.6141913467879667

In [112]: sumup()
Out[112]: 0.5679386404761081

In [113]: sumup()
Out[113]: 0.7877822673532973

In [114]: sumup()
```



Now, demonstrate how we write functions in python it is a pretty straight forward process let me try writing a function on the shell itself first and then I will show how to write functions in a file. So, it is done the following way you say d e f define and then you will say sumup and then a colon open bracket close bracket and a colon and then you say let say a equals random which means assign a random number from 0 to one to a and then b equals same random assign a random number to b and then I return a plus b enter. So, what does it do when I invoke sumup it just shows me the answer please note that the

answer is more than one here that is because a and b are 2 random numbers between 0 to 1, there is some can beyond one let me invoke sumup once again what will happen picks 2 random numbers between 0 to 1 add them and then display it so on, perfect.

(Refer Slide Time: 18:58)

```
In [111]: 1.6141913467879667
Out[111]: 1.6141913467879667

In [112]: sumup()
Out[112]: 0.5679386404761081
Out[112]: 0.5679386404761081

In [113]: sumup()
Out[113]: 0.7877822673532973
Out[113]: 0.7877822673532973

In [114]: sumup()
Out[114]: 1.0739263058866935
Out[114]: 1.0739263058866935

In [115]: sumup()
Out[115]: 1.0152992931590417
Out[115]: 1.0152992931590417

In [116]: sumup()
Out[116]: 0.6714803710641724
Out[116]: 0.6714803710641724

In [117]: sumup()
Out[117]: 0.9074029800479098
Out[117]: 0.9074029800479098

In [118]: !vi sudarshan.p
```



So, what you can do is whatever command whatever function definition you wrote right now you can actually open a file and then type it as well let me do that I am opening my favorite notepad I mean favorite editor which is vi editor. So, how do I invoke it I use exclamation vi space let say file name say sudarshan dot py.

(Refer Slide Time: 19:30)



This opens a new file you can use your own favorite text editor I use my I use vi editor note that exclamation stands for the here.

(Refer Slide Time: 19:41)

```
In [118]: !vi sudarshan.py
```

```
In [119]: !vi sudarshan.py
```

```
In [120]: !cat sudarshan.py
import random
def sum3rand():
    a=random.random()
    b=random.random()
    c=random.random()
    return a+b+c
```

```
In [121]: import sudarshan
```

```
In [122]: sudarshan.sum3rand()
Out[122]: 2.4726133609954433
```

```
In [123]: 
```



When you say exclamation it stands for outside command and then editor name and I type my file name you can even go outside this terminal and then open the text editor and then type it there also no problem.

So, let me define something here define sum 3 rand which is sum in 3 random numbers how do I do it I should first import random without that I cannot use the random function here in a file a equals random random b equals random random c equals random random and then I return a plus b plus c and then I come out its now saved I will display the contents of sudarshan dot py using the cat command of Linux it will just display the contents of the file you see that this much is there please note I have put import random here otherwise this will not work this was not the case with the defined thing that I wrote here I straight away used it that is because I had imported random already perfect.

Now, what do I do how do I invoke the sum 3 rand that is written inside a file by name sudarshan dot py that is pretty simple. So, what I do is I say I use the same import command when I say import sudarshan the sudarshan library whatever is contained in sudarshan gets loaded basically when I say import sudarshan import random happens and a function sum 3 rand gets defined with a following statements and now I can invoke it

see how I can invoke it I say simply say sudarshan dot sum 3 rand and that is it there you are random number between 0 to 1 added thrice.

(Refer Slide Time: 21:54)

```
Out[122]: 2.4726133609954433
In [123]: sudarshan.sum3rand()
Out[123]: 1.7922210310752487
In [124]: sudarshan.sum3rand()
Out[124]: 1.1484977931402813
In [125]: sudarshan.sum3rand()
Out[125]: 2.4071124341205135
In [126]: sudarshan.sum3rand()
Out[126]: 0.6917199631406367
In [127]: sudarshan.sum3rand()
Out[127]: 1.3254880812441912
In [128]: sudarshan.sum3rand()
Out[128]: 1.4466980344632545
In [129]: sudarshan.sum3rand()
```



Let me type it once again sudarshan dot sum 3 rand again gives you the same where sum 3 rand gives you some random number between 0 to 3 it is not really random because its picking a random number from 0 to 1 adding it thrice, this is actually not uniformly at random if you know probability you will know that, but that is not our concern here we are just observing that it is generating a random number thrice and adding it and then displaying it.

(Refer Slide Time: 22:18)

```
Out[129]: 1.768291238275506  
In [130]: sudarshan.sum3rand()  
Out[130]: 1.6856580370369079  
In [131]: sudarshan.sum3rand()  
Out[131]: 0.7293873918240366  
In [132]: sudarshan.sum3rand()  
Out[132]: 1.7983099756881937  
In [133]: sudarshan.sum3rand()  
Out[133]: 1.4387419167035427  
In [134]: sudarshan.sum3rand()  
Out[134]: 1.0149422968898556  
In [135]: sudarshan.sum3rand()  
Out[135]: 1.9272455647807818  
In [136]: sudarshan.sum3rand()
```



The point is you see what is happening here sudarshan is the name of the file and then in that I am invoking sum 3 rand and I am displaying it.

(Refer Slide Time: 22:41)

```
1 import random  
2 def sum3rand():  
3     a=random.random()  
4     b=random.random()  
5     c=random.random()  
6     return a+b+c  
7  
8 def sumkrand(k):  
9     ans=0  
10    for i in range(k):  
11        ans=ans+random.random()  
12    return ans  
13  
14  
15  
~  
~  
~  
~  
:
```



Now, let me further edit sudarshan dot py, how do I do that vi sudarshan dot py and let me define another function here define sum k rand what does that mean add k random numbers now how do I do that I take k as a parameter here and then I say for i in range k you have had some programming experience that is the pre requisite for this course you know what I am doing? I will say answer is equals 0 and then I will add to answer plus

random, random that is it let me go inside I must return this, I am sorry, I came out without returning it. So, I say return ans.

So, what have I done here k is the parameter here as we can see and then I have initialized the answer to 0 and then I am running a for loop k number of times that is what this means for i in range k means assign value 0 one to k up to k minus one to I whatever is assigned to I, it does not matter to me here I; all I want is I want to execute this loop k times this particular thing k times. So, answer equals answer plus random dot random simply picks random number from 0 to one and adds into the existing answer and assigns the new value to answer and finally, I come out of the; for loop and the return answer. So, what does this do? So, let me see let me save this and then come out.

(Refer Slide Time: 24:09)

```
In [136]: sudarshan.sum3rand()
Out[136]: 1.1629215697772501

In [137]: !vi sudarshan.py

In [138]: !vi sudarshan.py

In [139]: import sudarshan

In [140]: !vi sudarshan.py

In [141]: reload(sudarshan)
Out[141]: <module 'sudarshan' from 'sudarshan.py'>

In [142]: sudarshan.sumkrand(10)
Out[142]: 4.575307276426816

In [143]: sudarshan.sumkrand(10)
Out[143]: 5.2256366698481544

In [144]: sudarshan.sumkrand(100)
```



Now, please note import sudarshan will actually not work what you should do is whenever you have opened a file editor and you have added something new and then saved and come out you should not use import you should use reload sudarshan. So, sudarshan is reloaded in to the memory with the new updates a new function that I wrote just now and then you can open the new file what is that sum k rand and then I say ten. So, what should this output 10 random numbers sum if I say hundred instead of 10 you see hundred instead of 10 it will add hundred times random numbers from 0 to 1.

(Refer Slide Time: 24:57)

```
In [147]: sudarshan.sumkrand(100)
Out[147]: 52.737544589525804

In [148]: sudarshan.sumkrand(2)
Out[148]: 1.0681717586350272

In [149]: sudarshan.sumkrand(1)
Out[149]: 0.8563130656430317

In [150]: sudarshan.sumkrand(1)
Out[150]: 0.03881258471291482

In [151]: sudarshan.sumkrand(1)
Out[151]: 0.6739417080580347

In [152]: sudarshan.sumkrand(1)
Out[152]: 0.5814914963040022

In [153]: !vi sudarshan.py

In [154]:
```



And then output the answer yeah, perfect, if I just say 2 if I just say one it is as good as a random function correct. So, what did we just now learn you learnt the following?

(Refer Slide Time: 25:22)

```
Out[154]: 2.572630972476484

In [155]: sudarshan.sumkrand(5)
Out[155]: 2.563732366891008

In [156]: !vi sudarshan.py

In [157]: !vi sudarshan.py

In [158]: reload(sudarshan)
Out[158]: <module 'sudarshan' from 'sudarshan.py'>

In [159]: sudarshan.sum3rand?
Signature: sudarshan.sum3rand()
Docstring:
This function takes no input, but outputs the sum of 3
random numbers picked between 0 and 1
File:      ~/nptel/sudarshan.py
Type:     function

In [160]:
```



You can create a file and then write down your functions there and then invoke it from outside how do you invoke just say sudarshan dot followed by the name of the file sum 2 sum 3 rand is what we declared without only a parameter it generated a random number by picking a random function thrice and we also saw we also wrote a function by name

by sum k rand with an input parameter let say 5 which took 5 random numbers added and then displayed.

(Refer Slide Time: 25:56)

```
1 import random
2 def sum3rand():
3     '''This function takes no input, but outputs the sum of 3
4     random numbers picked between 0 and 1'''
5     a=random.random()
6     b=random.random()
7     c=random.random()
8     return a+b+c
9
10 def sumkrand(k):
11     '''This function takes k as input and adds random numbers
12     between 0 and 1, k times and then outputs the answer'''
13     ans=0
14     for i in range(k):
15         ans=ans+random.random()
16     return ans
17
18
19
"sdarshan.py" 19L, 443C written
```



One important tip open the editor an important tip you can come here and then write your favorite help one liner you can say this function takes no input, but outputs the sum of 3 random numbers picked between 0 and one and then close it I will see what happens let me show you I am going out saved you just saw what I did I opened the editors sudarshan dot py and I have added this thing here what did I do 3 single quotes followed by some random text and then I close the 3 single quotes and I came out there is some change to the file. So, I should say reload sudarshan.

Now, when I when I want to execute some command in sudarshan I type sudarshan and then I put a dot and then tab is a very important key when you press a tab here it will show you all the functions that are available in sudarshan random is available because random is imported there right sum 3 rand sum k rand is available if you put sum 3 rand which by the way is a function that I just defined and then put a question mark guess what happens.

(Refer Slide Time: 27:20)

```
In [160]: vi sudarshan.py
  File "<ipython-input-160-eddb619d54fd>", line 1
    vi sudarshan.py
      ^
SyntaxError: invalid syntax

In [161]: !vi sudarshan.py

In [162]: reload(sudarshan)
Out[162]: <module 'sudarshan' from 'sudarshan.py'>

In [163]: sudarshan.sumkrand?
Signature: sudarshan.sumkrand(k)
Docstring:
This function takes k as input and adds random numbers
between 0 and 1, k times and then outputs the answer
File:      ~/nptel/sudarshan.py
Type:     function

In [164]: !vi sudarshan.py
```



It shows you your help file that you just now wrote you see this; this function takes no input the output the sum of 3 random numbers picked between 0 and one who wrote this we wrote this just now correct.

Let me write as similar help file for the other command too. So, what do I do vi sudarshan dot py and then I am sorry I forgot to put an exclamation vi sudarshan dot py without exclamation the outside commands do not work vi is an outside command I come here what does this do let me just state that here this function takes k as input and adds random numbers between 0; between 0 random numbers between 0 and 1 k times and then outputs the answer towards the single code save you see what I did I just wrote a help function help one line for sum k rand function and then I go out and then I reload my sudarshan file now I say sudarshan sum k rand and then question mark it shows me the herewith whatever you have typed comes here.

Now, why is this even used this is used because when you write hundreds of functions. So, here we wrote only 2 functions what if I write hundred functions here in this file and I would not know what function is doing what.

(Refer Slide Time: 29:22)

```
In [163]: sudarshan.sumkrand?  
Signature: sudarshan.sumkrand(k)  
Docstring:  
This function takes k as input and adds random numbers  
between 0 and 1, k times and then outputs the answer  
File:      ~/nptel/sudarshan.py  
Type:      function  
  
In [164]: !vi sudarshan.py  
  
In [165]: !vi sudarshan.py  
  
In [166]: import sudarshan  
  
In [167]: reload(sudarshan)  
Out[167]: <module 'sudarshan' from 'sudarshan.pyc'>  
  
In [168]: sudarshan.sum3rand()  
Out[168]: 2.333830823621529  
  
In [169]:
```



So, it is important for us to write down what a function is doing in the form of help file that with the introduction to functions and you know how to define a function on the ipython terminal you also know how to define how to how to create a file edit the file and then type the functions there and then come out of it and then invoke it by saying. Firstly, import sudarshan and then second time when you edit something and then you want to reload it you may say reload sudarshan note reload is used when you edit the file and reload means update the contents of sudarshan, I have just now changed it and then what you do is you say sudarshan dot sum whatever is the name of the file name of the function and it executes that function; however big; however, complicated it executes the function and shows you the answer that is with functions.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture – 04**  
**Introduction to Social Networks**  
**Introduction to Phyton-2**

(Refer Slide Time: 00:08)

```
anamika@anamika-Inspiron-5423:~$ ipython
Python 2.7.6 (default, Oct 26 2016, 20:30:19)
Type "copyright", "credits" or "license" for more information.

IPython 1.2.1 -- An enhanced Interactive Python.
?           -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help        -> Python's own help system.
object?    -> Details about 'object', use 'object??' for extra details.

In [1]: d = {'annie':25, 'yoyo':35, 'sid':65}

In [2]: d
Out[2]: {'annie': 25, 'sid': 65, 'yoyo': 35}

In [3]: 
```



Hi everyone, in one of the previous videos you would have seen how we can make use of list data structure and we saw various functions that we can use with list. In this video we are going to look at yet another data structure that python provides and that is dictionary. So, we are going to see how we can create a dictionary and we are going to look at various functions that we can use with it. Before we get started; let me tell you in which situations, it is ideal to use a dictionary data structure.

So, dictionary is used when we have to store elements that have some sort of attribute attached with them; that is we have some elements and we also have to store some sort of property along with every element. In other words, in dictionary every element has two things; that is key and value, so there will a key value pair for every element, where these values stores the attribute of the key. Another thing to note is that dictionary is stored in memory as hash tables, which makes it faster than as compared to other data structures.

So, if you have some million entries and on one side you store them in the form of list and on the other side you store them in the form of dictionary, accessing these elements through the dictionary will always be faster. This is because accessing elements in dictionary does not go in sequential manner. So, let us get started and I believe you people might be comfortable with ipython, so I will continue using that.

Now, let me create an example dictionary here; assume my aim is to store name the few people and along with that I also want to store the age of these people. So, here I have this age which is an attribute that is attached to every person. So, here I have two things the key is the name of the people and value is their age, so this is the sort of information that I want to store; let me create an example dictionary here. So, I am going to add a few people's names and their age, let me add few elements here; so I add another element and I add another element alright. So, this is how we create a dictionary, so you note here that every element has two things; so this is the name which is the key here and this is the age which is the value here. So, this is the key value pair and we separate them with comma and we add another element and so on and the elements are kept inside these basis; so you can print this dictionary.

An interesting thing to note here is that these values might not necessarily be integers. So, rather they can be strings, they can be floats, they can be another list as well, they can be another dictionary as well. So, this is sort of flexibility that dictionary will provide us while its operation. Now, let me see and let me tell you how we can access the elements of a dictionary, so it is important that whenever we access we should have the key. Now, let me show you usage which does not work with dictionary; let me try accessing the first element by tracking this d0.

(Refer Slide Time: 03:32)

```
IPython 1.2.1 -- An enhanced Interactive Python.
?          -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help       -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: d = {'annie':25, 'yoyo':35, 'sid':65}

In [2]: d
Out[2]: {'annie': 25, 'sid': 65, 'yoyo': 35}

In [3]: d[0]
-----
KeyError                                     Traceback (most recent call last)
<ipython-input-3-17371c6688f6> in <module>()
----> 1 d[0]

KeyError: 0

In [4]:
```



Now, this is not going to work; now this is because there are elements in dictionary are not associated with index unlike the case of a list. For example, in list you can access the elements using their index; however, it does not happen in dictionary; this is because there is not association of any index with the elements. The only association that takes place in a dictionary is between the key and the value. So, just in case you want to access this element Annie, you would have to access it through the key.

(Refer Slide Time: 04:04)

```
%quickref -> Quick reference.
help     -> Python's own help system.
object?  -> Details about 'object', use 'object??' for extra details.

In [1]: d = {'annie':25, 'yoyo':35, 'sid':65}

In [2]: d
Out[2]: {'annie': 25, 'sid': 65, 'yoyo': 35}

In [3]: d[0]
-----
KeyError                                     Traceback (most recent call last)
<ipython-input-3-17371c6688f6> in <module>()
----> 1 d[0]

KeyError: 0

In [4]: d['annie']
Out[4]: 25

In [5]:
```



So, I will write the key here and I will be getting the value that is 25, so this is how we access the elements of a dictionary. Now, why is this sort of thing happening, so d[0] as I showed you is not working, this is because the order of elements in dictionary is not important. Basically, we cannot control the order in which the elements are displayed; let me print dictionary d.

(Refer Slide Time: 04:30)

```
KeyError Traceback (most recent call last)
<ipython-input-3-17371c6688f6> in <module>()
      1 d[0]

KeyError: 0

In [4]: d['annie']
Out[4]: 25

In [5]: d[0]
-----
KeyError Traceback (most recent call last)
<ipython-input-5-17371c6688f6> in <module>()
      1 d[0]

KeyError: 0

In [6]: d
Out[6]: {'annie': 25, 'sid': 65, 'yoyo': 35}

In [7]:
```



This is what I wanted to show you, we had added first Annie and then we had added yoyo and then we had added sid. However, when we are printing this dictionary; we are getting a different order. Now this is because it is up to the dictionary how it is storing the elements in the memory, so they are not like; they are not stored in the same sequence in which we add the elements.

Now, let me show you how we can update the elements in a dictionary; for example, I want to change the value of one of these elements.

(Refer Slide Time: 05:00)

```
KeyError: 0
In [6]: d
Out[6]: {'annie': 25, 'sid': 65, 'yoyo': 35}

In [7]: d['sid'] = 75

In [8]: d
Out[8]: {'annie': 25, 'sid': 75, 'yoyo': 35}

In [9]: d['john'] = 15

In [10]: d
Out[10]: {'annie': 25, 'john': 15, 'sid': 75, 'yoyo': 35}

In [11]: del d['john']

In [12]: d
Out[12]: {'annie': 25, 'sid': 75, 'yoyo': 35}

In [13]: clear
```



So, assume I want to change the value of this element to be this and then I print the dictionary, so you see the value has been updated and now let us see how we can add more elements into this dictionary. So, let me add one more person here say john and I want its age to be some this number and now let me print this, so this element has been added into the dictionary.

Now point to be noted here is that; the syntax for adding an element like this and the syntax for updating some value in the dictionary are the same. So, just in case the element is not presented to be added and in case the element is present, its value will be updated; so it is that simple. Now, let me show you how we can delete entries from a dictionary, so the keyword for that is `del`. Now, assume I want to remove this element john out of this dictionary, so I will access it using the key again as I told you. So, I will write this and this element should have been removed; let me check I will print this, the element has been removed.

Similarly, if you want to remove all the elements from the dictionary; you can also do that by using a simple function that is `d.clear`. So, what this function do, it will remove all the elements from the dictionary, but the dictionary will stay, so you can add more elements later. I will not use it because I do want this dictionary as if now and in case you want to remove the entire dictionary from the memory, you have another keyword;

you can just write del d, d is the name of the dictionary and the dictionary will be removed, so again I will not execute it.

(Refer Slide Time: 06:44)

```
In [14]: d.has_key('annie')
Out[14]: True

In [15]: d.keys()
Out[15]: ['yoyo', 'annie', 'sid']

In [16]: d.values()
Out[16]: [35, 25, 75]

In [17]: d.items()
Out[17]: [('yoyo', 35), ('annie', 25), ('sid', 75)]

In [18]: for i in d.items():
...:     print i
...:
('yoyo', 35)
('annie', 25)
('sid', 75)

In [19]: for i in d.items():
...:     print i[0], 'has age: ', i[1]
```



Now, let me tell you a few functions that might be useful while we are doing the analysis of social networks. One of the functions that is quite used is; to check whether an element is present in the dictionary or not. So, assume I want to check whether the dictionary has an element Annie or not, so I will write d.has key; so this is an important function its quite frequently used. So, since I do not; I cannot see the entire dictionary, I can just execute this function and see whether Annie is present in it or not, so its gave me true which means the element is present.

If I want to access all the keys from the dictionary, I can just write d.keys and it will give me a list. So, this is basically a list of all the key values from all the elements, similarly if we want to access all the values; we have this function; d.values, again it gave me a list, which is having all the ages of all the people.

Now, there is one more function which is again quite frequently used and that is items. So, in case we want to access key as well as value for every element; we can make use of this function d.items. So, you see here again we got a list, but the element of the list are basically tuples, where every tuple has two things; first is key and second is value. Also, if we want to access these elements one by one; we can use a for loop, let me show you

that. So, what I want is; I want to access the individual elements out of these tuples that we are getting from these items function.

So, I will write for `i in d.items`, so this `d.items` is going to return me a list of tuples and I am just going to print them; let us see. So what we did? We accessed these elements one by one that is all. Now, if you want to access key separately and value separately; you can use the index of these elements. For example, here I can write `i[0]`, so `i[0]` will point to the key here and `i[1]` will point to the value here, so I can write `i[0]` has age for example, `i[1]`.

(Refer Slide Time: 09:05)

```
In [16]: d.values()
Out[16]: [35, 25, 75]

In [17]: d.items()
Out[17]: [('yoyo', 35), ('annie', 25), ('sid', 75)]

In [18]: for i in d.items():
....:     print i
....:
('yoyo', 35)
('annie', 25)
('sid', 75)

In [19]: for i in d.items():
....:     print i[0], 'has age: ', i[1]
....:
yoyo has age: 35
annie has age: 25
sid has age: 75

In [20]:
```



So, what we are doing here is; we are accessing the individual elements of the tuple by the index, so that is always possible.

(Refer Slide Time: 09:13)

```
In [22]: d1 = {x:x**2 for x in range(11)}  
In [23]: d1  
Out[23]: {0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81,  
10: 100}  
  
In [24]: d1 = {x:x**2 for x in range(11) if x%2 == 0}  
In [25]: d1  
Out[25]: {0: 0, 2: 4, 4: 16, 6: 36, 8: 64, 10: 100}  
  
In [26]: d1.has_key(3)  
Out[26]: False  
  
In [27]:
```



Now, let me show you how we can quickly create a dictionary, where the elements follow some sort of pattern. So, I am going to make use of for loop for creation of a dictionary, so let me create this dictionary; where the elements should be of this sort and I will use for loop for  $x$  in range say 11; I am sorry in 11. So, this dictionary is created, so the elements in the dictionary are of this sort, where key is  $x$  and the value is  $x$  square. So, this operator you might be knowing in python, it is exponentiation operation and I am using for loop; where the  $x$  is going from, so this range returns a list going from 0 to 10, so 11 is excluded.

Let us print this dictionary and see what it contains, so we have 0, 1, 2, 3 up to 10 as keys and the squares as the values. So, this is how we can quickly create a dictionary; we can also play around with it. For example, I want only even numbers; so I can add another condition here, if  $x \bmod 2$  is equal to 0 that is all. So, let me see what the dictionary contains, its now contains only even numbers are the values and sorry even numbers as the keys and squares as the values.

We can also check the function that we just learnt; has key, so if I write even.has key; if I write 3 here, it should return false because there is not odd number as the key here. So, I think that was our pretty much basic introduction to dictionaries.

Plotting is an integral part of any sort of analysis, since in this course we are going to analyze social networks; we might need to plotters us to better visualize them. Therefore,

in this video we are going to see how we can make use of the package called matplotlib, which is provided by python for plotting purpose.

(Refer Slide Time: 11:15)

```
anamika@anamika-Inspiron-5423:~$ ipython
Python 2.7.6 (default, Oct 26 2016, 20:30:19)
Type "copyright", "credits" or "license" for more information.

IPython 1.2.1 -- An enhanced Interactive Python.
?           -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help        -> Python's own help system.
object?    -> Details about 'object', use 'object??' for extra details.

In [1]: import matplotlib.pyplot as plt

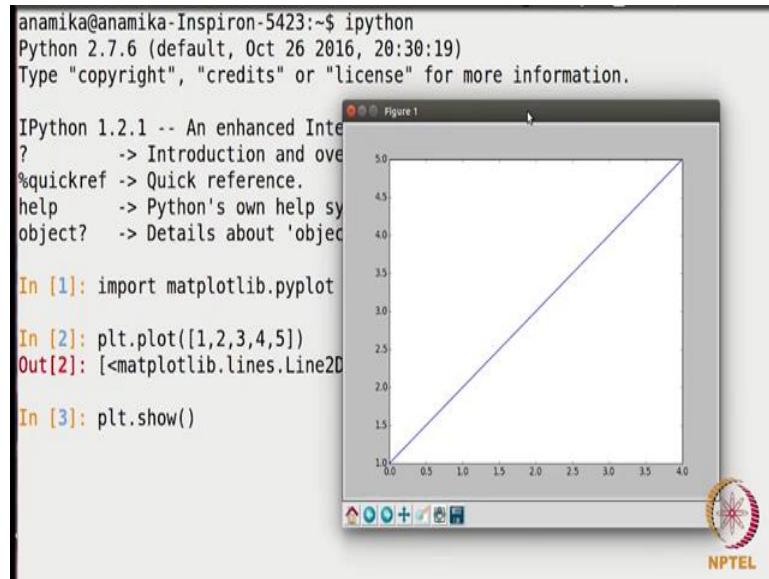
In [2]: plt.plot([1,2,3,4,5])
Out[2]: [
```



So, let us get started, so you might be to import this package first of all. So, the package name is matplotlib, inside matplotlib we have pyplot which contains all the function for plotting. I am going to give it a short name so that it is easy to use, now this means we are going to use plt instead of matplotlib.pyplot now on; so that will make things easier.

This package has a lot of functions for plotting and to customize our plots as well. The very basic function that this function has is called plot, so we are going to first use that function plot. So, I am going to write plt.plot, after that I will be giving various parameters to this plot function. Now again this plot function has various versions, to start with we are going to look at the basic version of this plot function which accepts a single list as a parameter. So, what I am going to do is; I am going to pass a list having some 5 elements here alright, so this is a very basic usage of this plot function. In order to see the plot we need to use a function that is called show, so I am going to write plt.show.

(Refer Slide Time: 12:39)



The screenshot shows a terminal window and a Jupyter Notebook cell. The terminal window displays the Python 2.7.6 welcome message. The Jupyter Notebook cell contains three lines of code: importing matplotlib.pyplot, plotting a list of values [1, 2, 3, 4, 5], and showing the plot. To the right, a plot window titled 'Figure 1' shows a single blue line segment connecting the points (0, 1) and (4, 5). The x-axis ranges from 0.0 to 4.0 with increments of 0.5. The y-axis ranges from 1.0 to 5.0 with increments of 0.5. The plot has no title or axis labels.

When I do that, I get this plot in the other window and as you can see this is very basic plot with no customization, there is no title of the plot, there is not title on the x and x and y axis and you see the x axis is starting from 0 and the y axis is starting from 1. So, let me tell you the reason for it; we called plot function with single parameter with which was only one list. Now when we do that, the values at the past in the single list they are taken as y values, which means this 1, 2, 3, 4, 5 are taken as the y values of the plot and since we have not given any values for the x axis, they are automatically taken up by matplotlib.

Now, since the index in python starts from 0; the x values also start from 0 automatically, which means x is equal to 0 is assigned to y value of 1, x is equal to 1 is assigned to y value of 2 and so on. So, you can see in this plot; x is equal to 0 has y (Refer Time: 13:44) 1, x is equal to 1 has y (Refer Time: 13:46) 2, x is equal to 2 has y value of 3 and so on. So, of course we can always change that, but when we provide only one parameter through this plot function this is how it happens. But we can always customize things; we can make this plot specific to our requirements, so we are going to see how we can do that using other functions. Now, I am going to close this window and I am back on the command window.

(Refer Slide Time: 14:17)

```
anamika@anamika-Inspiron-5423:~$ ipython
Python 2.7.6 (default, Oct 26 2016, 20:30:19)
Type "copyright", "credits" or "license" for more information.

IPython 1.2.1 -- An enhanced Interactive Python.
?          -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help       -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: import matplotlib.pyplot as plt

In [2]: plt.plot([1,2,3,4,5])
Out[2]: [
```



Now, let us look at some versions of this plot function; let us use your plot function where we are passing both x and y values. So, what we are going to do is; we will be passing two lists as parameter, the first list will be the values for the x axis and the second list will be the values for the y axis. So, I am going to pass say 1, 2, 3, 4 and 5 as x values and for y values, let me just pass the squares for example, so this is what I am passing here; so I am explicitly passing both the x and y values alright. Now to see the plot, I will write plt.show and I see this plot.

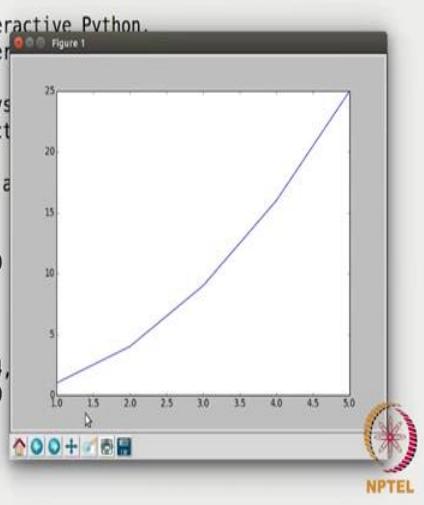
(Refer Slide Time: 14:56)

```
Python 2.7.6 (default, Oct 26 2016, 20:30:19)
Type "copyright", "credits" or "license" for more information.

IPython 1.2.1 -- An enhanced Interactive Python.
?          -> Introduction and over Figure 1
%quickref -> Quick reference.
help       -> Python's own help sys
object?   -> Details about 'object'

In [1]: import matplotlib.pyplot a

In [2]: plt.plot([1,2,3,4,5])
Out[2]: [
```



So, here the values; the corresponding to 1, we have 1 corresponding to 2, we have 4 and so on. Now, I am closing this window; I am going to show you some more versions of plot function.

(Refer Slide Time: 15:15)

```
?      -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help    -> Python's own help system.
object? -> Details about 'object', use 'object??' for extra details.

In [1]: import matplotlib.pyplot as plt

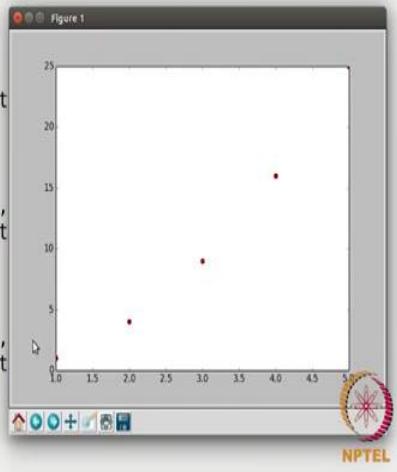
In [2]: plt.plot([1,2,3,4,5])
Out[2]: [
```



Let us get back here, so by default we were getting a blue line here as the plot; we can always change it, we can change the colour, we can change appearance of the plot. For example, by default we were getting a line; we can always change it to some points. Let me give you an example here, so I want the plot to be red in colour and say I want the dots; the points and not the line. So, I will write r; o here; let us see what we get, so every time you call this plot function, you have to write plt.show as well; in order to see the plot.

(Refer Slide Time: 15:50)

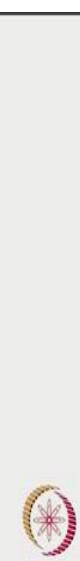
```
%quickref -> Quick reference.  
help      -> Python's own help system.  
object?   -> Details about 'object', use 'object??' for extra details.  
  
In [1]: import matplotlib.pyplot as plt  
  
In [2]: plt.plot([1,2,3,4,5])  
Out[2]: []  
  
In [3]: plt.show()  
  
In [4]: plt.plot([1,2,3,4,5],[1,4,9,16,25])  
Out[4]: []  
  
In [5]: plt.show()  
  
In [6]: plt.plot([1,2,3,4,5],[1,4,9,16,25])  
Out[6]: []  
  
In [7]: plt.show()
```



Now, here you see instead of the line I am now getting the points and that too red in colour.

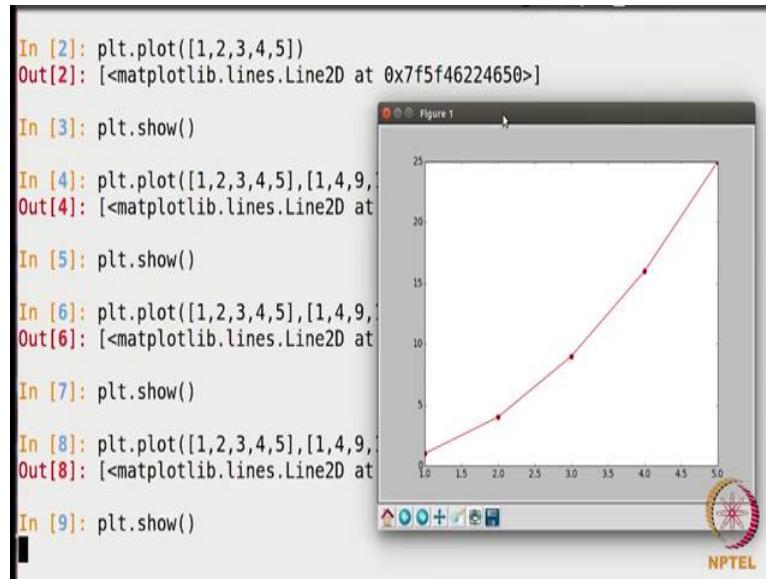
(Refer Slide Time: 15:57)

```
In [1]: import matplotlib.pyplot as plt  
  
In [2]: plt.plot([1,2,3,4,5])  
Out[2]: []  
  
In [3]: plt.show()  
  
In [4]: plt.plot([1,2,3,4,5],[1,4,9,16,25])  
Out[4]: []  
  
In [5]: plt.show()  
  
In [6]: plt.plot([1,2,3,4,5],[1,4,9,16,25],'ro')  
Out[6]: []  
  
In [7]: plt.show()  
  
In [8]: plt.plot([1,2,3,4,5],[1,4,9,16,25],'ro-')  
Out[8]: []  
  
In [9]: plt.show()
```



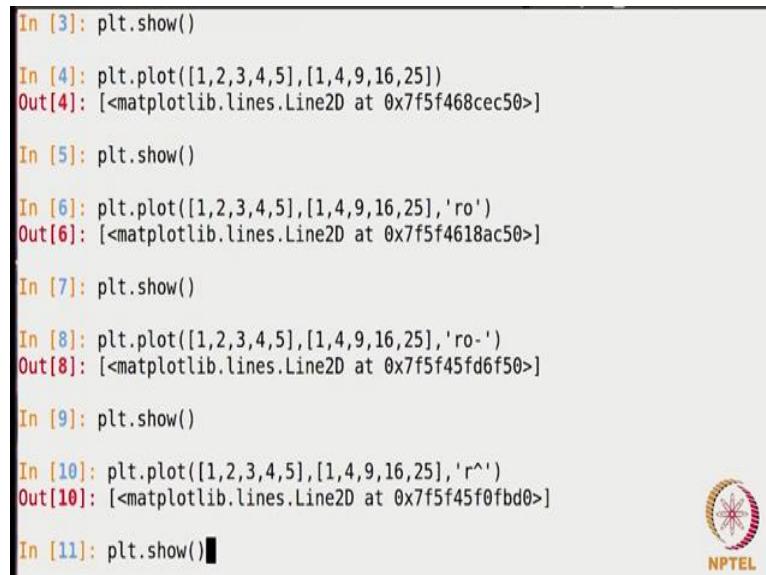
And in case you want the line as well, so you can put a dash here enter and then plt.show, now you get the points as well as line.

(Refer Slide Time: 16:05)



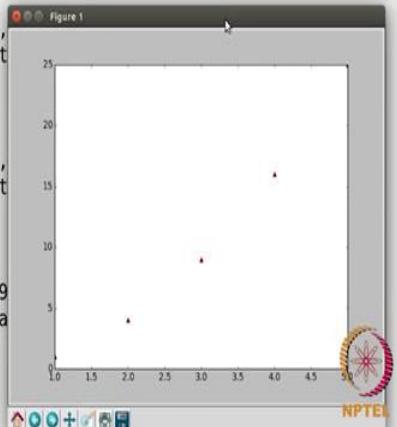
So, these are some of the versions; you can even change the other things, let me show you some more markers here so that you can play around with them.

(Refer Slide Time: 16:23)



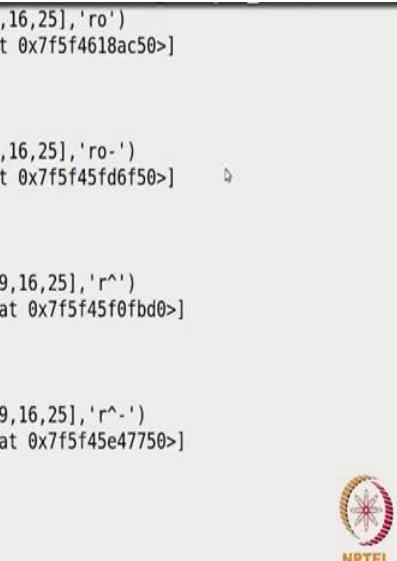
For example, if you want blue triangles; I will just write this symbol here and then I will write plt.show.

(Refer Slide Time: 16:29)

```
In [4]: plt.plot([1,2,3,4,5],[1,4,9,16,25])
Out[4]: [
```

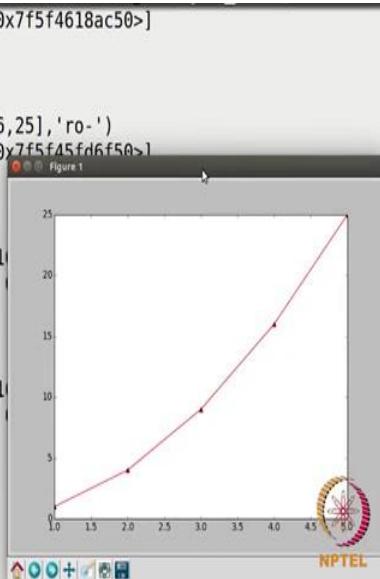
Now, you see you are getting the triangles here.

(Refer Slide Time: 16:33)

```
In [6]: plt.plot([1,2,3,4,5],[1,4,9,16,25], 'ro')
Out[6]: [
```

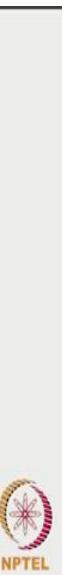
(Refer Slide Time: 16:39)

```
Out[6]: []
In [7]: plt.show()

In [8]: plt.plot([1,2,3,4,5],[1,4,9,16,25],'ro-')
Out[8]: [
```

And if you want triangles as well as lines you can again; you can do that, so you are getting both the things.

(Refer Slide Time: 16:42)

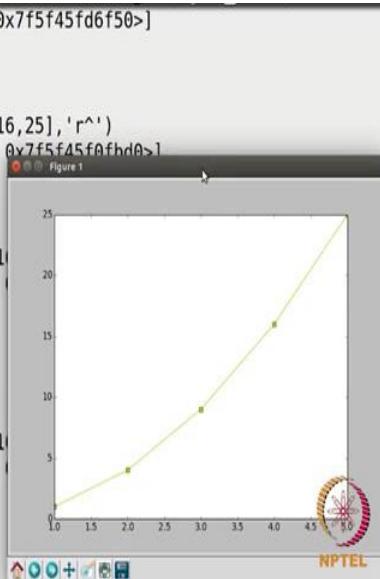
```
In [8]: plt.plot([1,2,3,4,5],[1,4,9,16,25],'ro-')
Out[8]: [
```

And if you want the squares, so here you can write s and say I want yellow squares and then I will write plt.show.

(Refer Slide Time: 16:51)

```
Out[8]: []

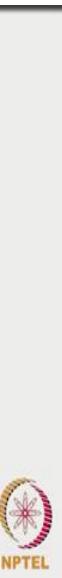
In [9]: plt.show()

In [10]: plt.plot([1,2,3,4,5],[1,4,9,16,25],'r^')
Out[10]: [
```

So, we getting squares here and the line is also there.

(Refer Slide Time: 16:58)

```
In [13]: plt.show()

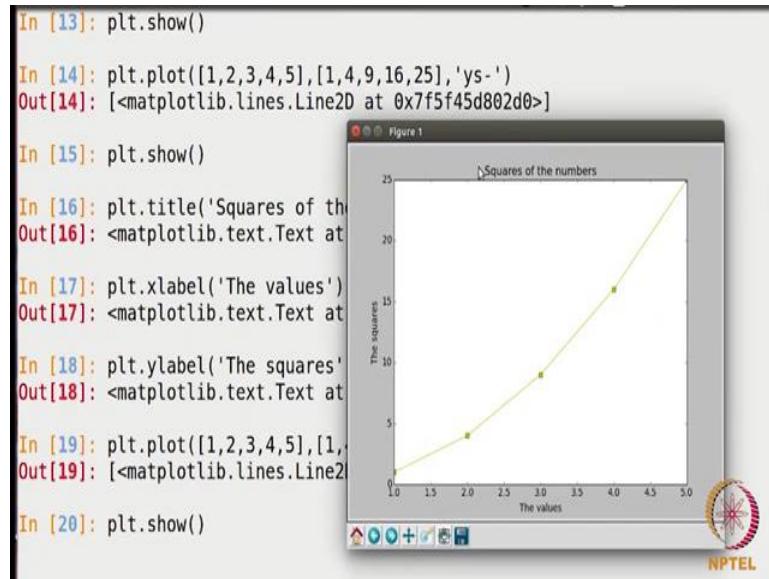
In [14]: plt.plot([1,2,3,4,5],[1,4,9,16,25],'ys-')
Out[14]: [
```

So, there you can basically look at the documentation and you will see there are various and you can just play around with them and use them as per your requirement. As you saw that, there was no title for the plot, there was no title on the x and y axis. So, let us try doing that; the function that we will be using for doing that, there is title function. So, we will write plt.title and we can keep the title of the plot here. Let me give something like squares of the numbers and I also want to give title for the x axis. The function that

we will use for that is x label, so I will write plt.x label; say the values. I also want to give a label for the y axis, so I will write plt.l label and may be the squares.

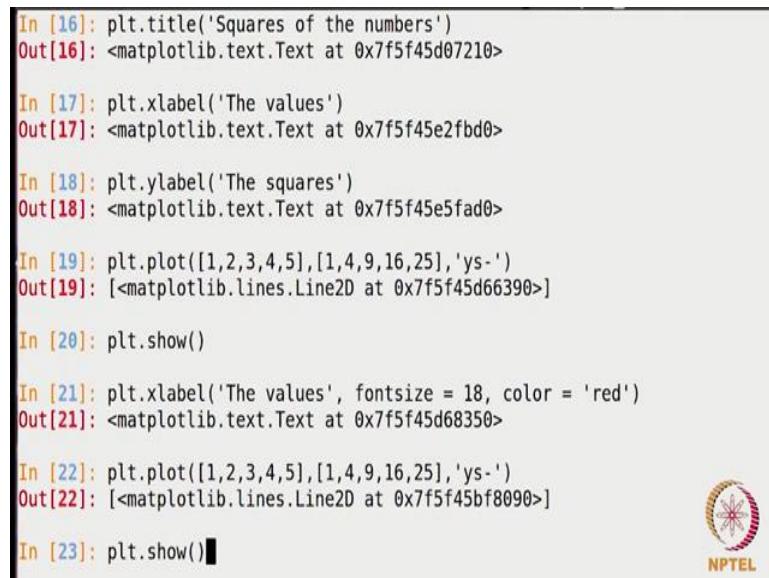
Now, again I will call this plt.plot and then I will call plt.show.

(Refer Slide Time: 18:06)



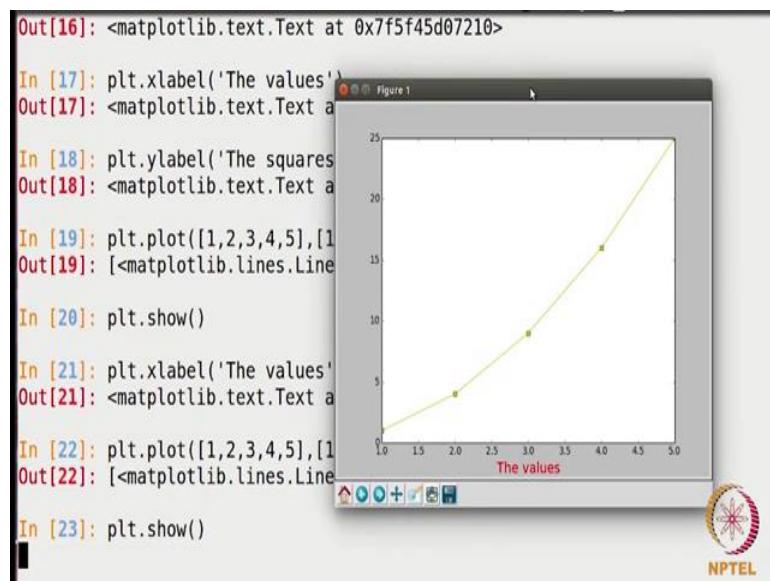
Now, you see we have got this; the title of the plot, we have the title of the x and y axis as well.

(Refer Slide Time: 18:18)



Let me also show you that you can also change the colour and the font size of these labels as well, as per your requirement. So, let me give you one example here; I want to change the x label to be say red in colour. So, we already made use this function rate x label plt.x label sorry, so we had only passed the label here. So, I am going to pass few more parameters here to customize it, so I am going to change the font size say 18 and I also want to change the colour; so I will write colour is equal to say red. Now label is changed, so I will call the plot command again and then I will show it.

(Refer Slide Time: 19:11)



So, you see the x label has changed, the colour has changed, the font has changed; so you can always; you know keep it the way you want it. Another thing that I want you to notice here is that; the values on x axis is starting from 1 to 5 and the values on the y axis is starting from 0 to 25 and if you see the values that we had passed are; the x axis values are from 1 to 5 and y values are from 1 to 25. So, it has automatically chosen the values where to start from and where to end; where we can always change it, we can always customize it. So, let me show you how we can do that.

(Refer Slide Time: 19:52)

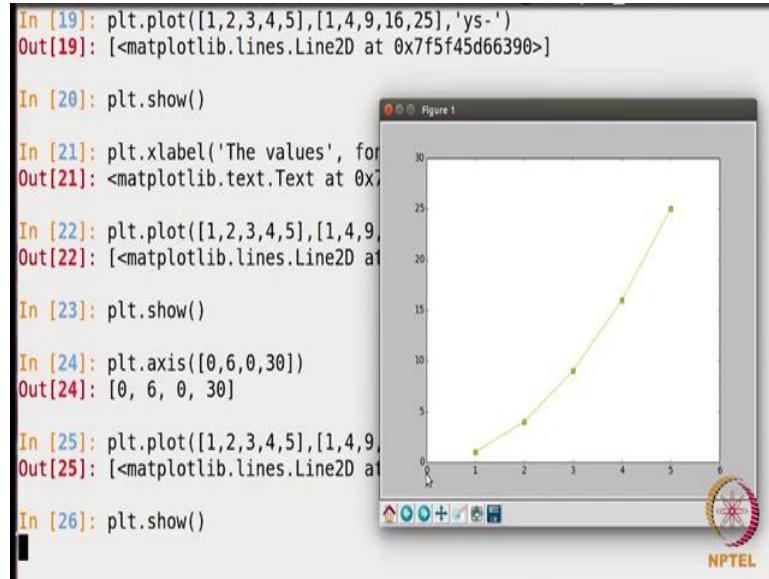
```
In [19]: plt.plot([1,2,3,4,5],[1,4,9,16,25],'ys-')
Out[19]: [
```



The function that we use to change the starting and ending of the axis is called plt.axis itself. So, I will write plt.axis; now this function accepts one parameter which is a list having four values, so the first value is the starting of x axis, the second value is the ending of x axis, the third value is the starting of y axis and the fourth value is the ending of y axis.

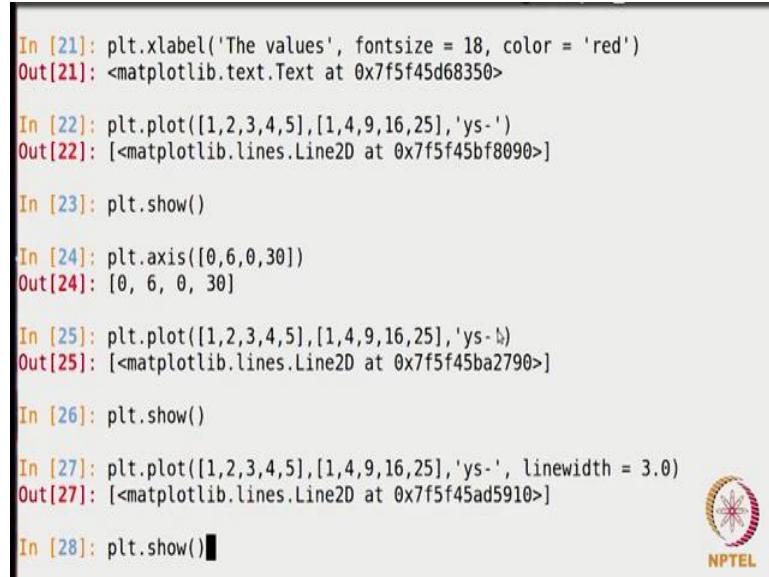
So, here you see our values; x values are starting from 1 to 5, so let me in order to show you its usage; I am going to give it 0, the starting of x axis and the ending of x axis; I am just like that going to give 6; just to show you how it works. And now I have to give the starting y axis, I am going to give 0 here and the ending of y axis; you see it is up to 25. So, let me just give it 30 so that we see how it works alright, so again I will call plt.plot and then I will call plt.show.

(Refer Slide Time: 20:56)



Now, you see here; x values are now starting 0 to 6 and y values are now starting from 0 to; are now going from 0 to 30 alright. So, you can this is always in control you can change it the way you want it.

(Refer Slide Time: 21:15)

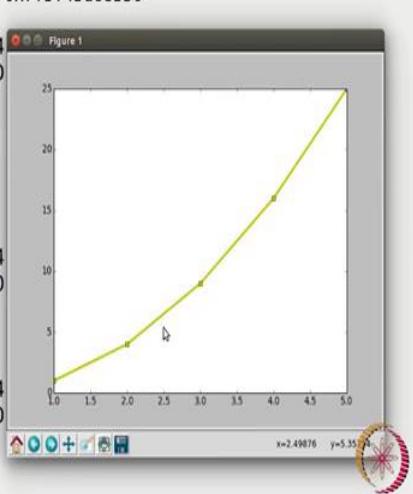


Let me now show you couple of more features of this plot function; for example, you got a very thin line as a plot. If you want to increase the thickness of this line you can do that, let me show you how to do that. So, plt.plot; let me reuse the same function, so here we were passing some three parameters, I am going to add one more parameter here for

the thickness of the line. So, the keyword is line width here, so line width is equal to say 3.0; by default it is 1.0, so I give this line width and then I will call plt.show.

(Refer Slide Time: 21:48)

```
In [21]: plt.xlabel('The values', fontsize = 18, color = 'red')
Out[21]: <matplotlib.text.Text at 0x7f5f45d68350>

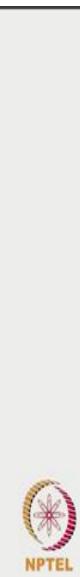
In [22]: plt.plot([1,2,3,4,5],[1,4,9,16,25])
Out[22]: [
```

Now, you see the thickness of this line has increased, so you can always change that it is all about decorating your plot, so that it is better visualizing.

(Refer Slide Time: 22:02)

```
In [23]: plt.show()

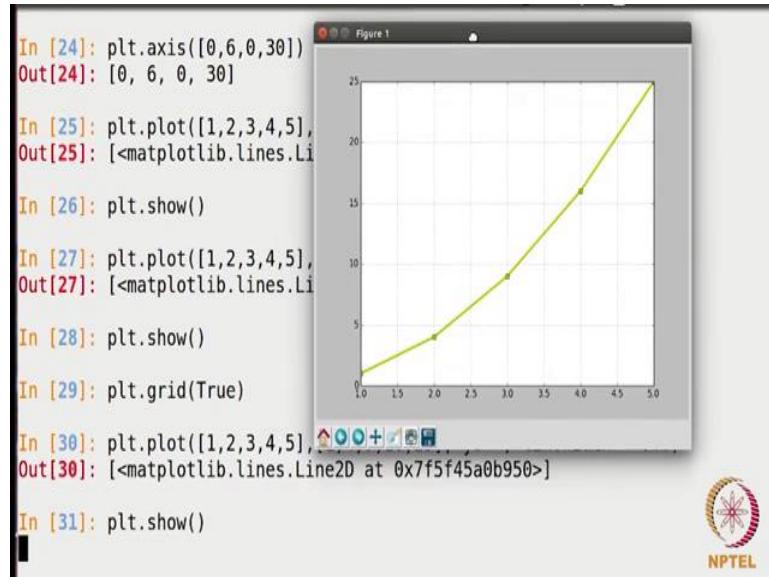
In [24]: plt.axis([0,6,0,30])
Out[24]: [0, 6, 0, 30]

In [25]: plt.plot([1,2,3,4,5],[1,4,9,16,25],'ys-')
Out[25]: [
```

Another thing that is sometimes needed is a grid that you want to display in your plot sometimes. So, by default it is turned off, but you can always turn it on by passing a true parameter there, so the function that we use there is grid. So, I will write plt.grid and I

will pass true here, which is by default false. So, once I do that and I call this plot function and then I will call the show function.

(Refer Slide Time: 22:30)



You see there is a grid here, so that is there in the background you can always turn on and off as per your requirement. You can also save this plot if you want; there are various functions, there are various features here that you can make use of ok.

(Refer Slide Time: 22:51)



Let us get back here, as of now we were showing only one plot inside the plotting region. But that is not necessary; we can always display multiple plots inside the same plotting

region. Let me show you an example for that, let me create the x values by using a loop for example, that will also show you how you can use a loop to create a list, if you do not know already.

So, I right i for i in range; 21, if I write this; I will get the values from 0 to 20; excluding 21. So, basically I will be getting 21 values and let me keep 20 here, so I got the x values and then I want to plot this x along with its squares. So, I will write x and this is exponential operator and I want it red in colours, maybe in line. Now in case I want to also display the x cube as well in the same plotting reason, I can do that. So, let me show you how to do that, so I will write x cube here and say I want that blue in colour, blue lines and assume I want to display x to the 4 as well in the same plotting region. So, I will write x to the 4 here and say I want that yellow in colour; I will do this.

But there is an important point to be noted here, which I want you to know. When I run this command, it should show me an error; let me see here, it is showing an error. Now what is the reason for that? The reason is that x is a list, this is a list and then we are applying this exponentiation operator on a list, which is not allowed in python. So, what we can do here is that; we can make use of a package which is called numpy, which allows such sort of operation. So, if we if we convert this list x into a numpy array; then in that case such operations are allowed. So, the crux is that this kind of operation is not allowed on lists, but it is allowed on numpy arrays. So, let us do one thing we will make use of this numpy package.

(Refer Slide Time: 25:10)

```
In [33]: plt.plot(x, x**2, 'r-', x, x**3, 'b-', x, x**4, 'y-')
-----
TypeError                                 Traceback (most recent call last)
<ipython-input-33-724b24d2d8a2> in <module>()
      1 plt.plot(x, x**2, 'r-', x, x**3, 'b-', x, x**4, 'y-')

TypeError: unsupported operand type(s) for ** or pow(): 'list' and 'int'

In [34]: import numpy

In [35]: x = numpy.array(x)

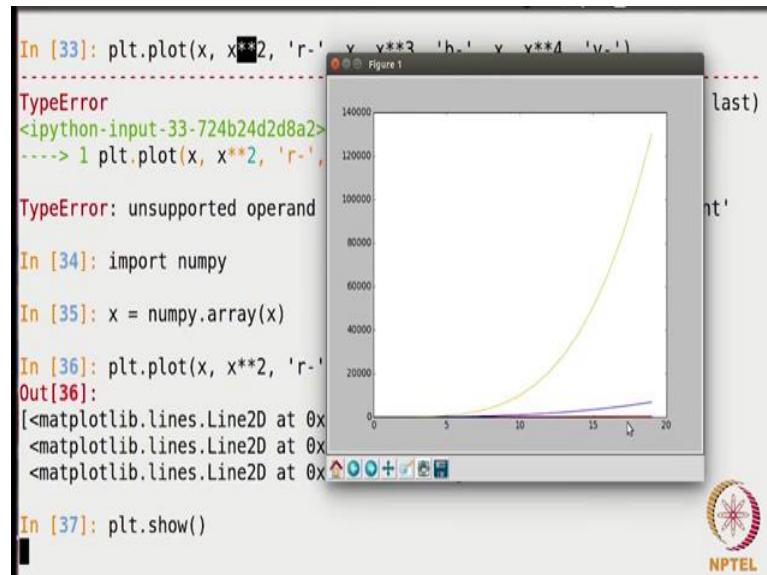
In [36]: plt.plot(x, x**2, 'r-', x, x**3, 'b-', x, x**4, 'y-')
Out[36]:
[<matplotlib.lines.Line2D at 0x7f5f45937090>,
 <matplotlib.lines.Line2D at 0x7f5f45937310>,
 <matplotlib.lines.Line2D at 0x7f5f459379d0>]

In [37]: plt.show()
```



Let me put that; now I will basically convert this  $x$  into a numpy array. So, I will write `numpy.array`, this is a function, so I am passing  $x$  as a list and I am getting it as a numpy array. Now after that, if I called this plot; it is now giving me an error and it is working fine, now we can do `plt.show`.

(Refer Slide Time: 25:41)



So, here you see I am getting all the 3 plots here; yellow, blue and red; I am getting all the plots here. Now since this due to these dimensions of this  $y$  axis, you are not really

able see this red line, but you can always do that as I told you, you can customize, you can change the starting and ending of y axis; so you can play around with that alright.

(Refer Slide Time: 26:05)

```
In [35]: x = numpy.array(x)
In [36]: plt.plot(x, x**2, 'r-', x, x**3, 'b-', x, x**4, 'y-')
Out[36]:
[<matplotlib.lines.Line2D at 0x7f5f45937090>,
 <matplotlib.lines.Line2D at 0x7f5f45937310>,
 <matplotlib.lines.Line2D at 0x7f5f459379d0>]

In [37]: plt.show()

In [38]: import random

In [39]: x = [random.randint(1,100) for i in range(100)]
In [40]: y = [random.randint(1,100) for i in range(100)]

In [41]: plt.scatter(x,y)
Out[41]: <matplotlib.collections.PathCollection at 0x7f5f4584fc50>
In [42]: plt.show()
```

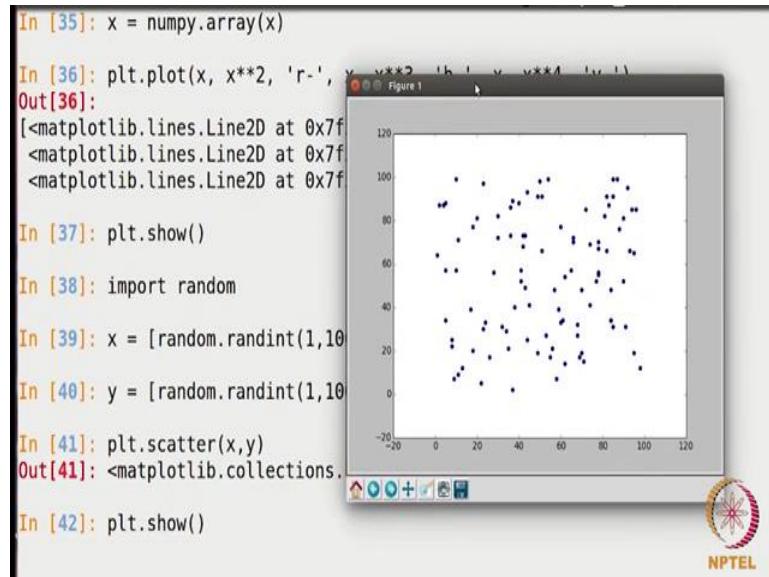


As of now we have seen only a simple plot function, we have seen a simple plot in form of lines or points, but let me tell you that we can also have a different kinds of plots using this matplotlib package; for example, we can have scatter plot, we can have bared plot, we can have heat max as well. So, there are various kinds of plots that we can draw; now I will be showing you some two, three examples just to give an idea.

I will show you how to draw scatter plot, so in order to draw the plot let me create a list randomly also that will show you how you can make use of a random function. Before that I need to import the package, so I will import random package and then I am going to create a list which will be having a 100 random values. So, how do we do that? I will make use of the function randint from this package random and I want the values to be between 1 and 100 and I want 100 such value, so I will write for i in range 100.

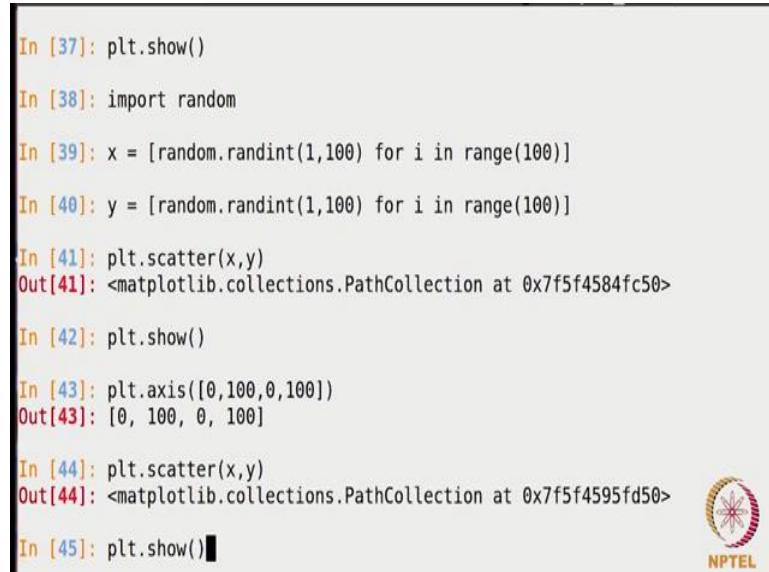
So, I will getting 0 to 99, I will be getting 100 values here alright, so let me also have y values randomly. So, basically I am having a 100 random values as x values and 100 random values as y values. Now, let me draw the scattered plot; the function is scatter itself. So, I will plt.scatter(x, y) and in order to see the plot, we have to write plt.show.

(Refer Slide Time: 27:49)



So, here we get the plot; now again the kinds of operation that we applied in the simple plot function, like we changed the axis, we changed the colour, we changed the titles and everything that also we can do here. For example, here you see by default; the x axis starting from minus 20 and same goes with the y axis which is not looking so good.

(Refer Slide Time: 28:16)



Let me change that, so what I will do is plt.axis, I am going to pass a list with four values; starting of x axis to be 0 and ending to be 100, starting of y to be 0 and ending to

be 100. Now, I will call this plot function again, sorry scattered plt.scatter and then plt.show.

(Refer Slide Time: 28:41)

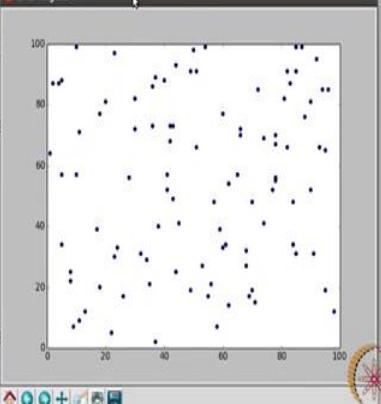
```
In [37]: plt.show()

In [38]: import random

In [39]: x = [random.randint(1,100)
In [40]: y = [random.randint(1,100)
In [41]: plt.scatter(x,y)
Out[41]: <matplotlib.collections.PathCollection at 0x7f5f4584fc50>
In [42]: plt.show()

In [43]: plt.axis([0,100,0,100])
Out[43]: [0, 100, 0, 100]

In [44]: plt.scatter(x,y)
Out[44]: <matplotlib.collections.PathCollection at 0x7f5f4595fd50>
In [45]: plt.show()
```



A scatter plot titled 'Figure 1' showing 100 random points plotted against a 100x100 grid. The x-axis and y-axis both range from 0 to 100. The points are small black dots scattered across the plot area.

Now, you see x and y are starting from 0 now and it is looking quite better, you can clear on, we can change the colours of these dots alright, let us get back here.

(Refer Slide Time: 28:52)

```
In [39]: x = [random.randint(1,100) for i in range(100)]
In [40]: y = [random.randint(1,100) for i in range(100)]
In [41]: plt.scatter(x,y)
Out[41]: <matplotlib.collections.PathCollection at 0x7f5f4584fc50>
In [42]: plt.show()

In [43]: plt.axis([0,100,0,100])
Out[43]: [0, 100, 0, 100]

In [44]: plt.scatter(x,y)
Out[44]: <matplotlib.collections.PathCollection at 0x7f5f4595fd50>
In [45]: plt.show()

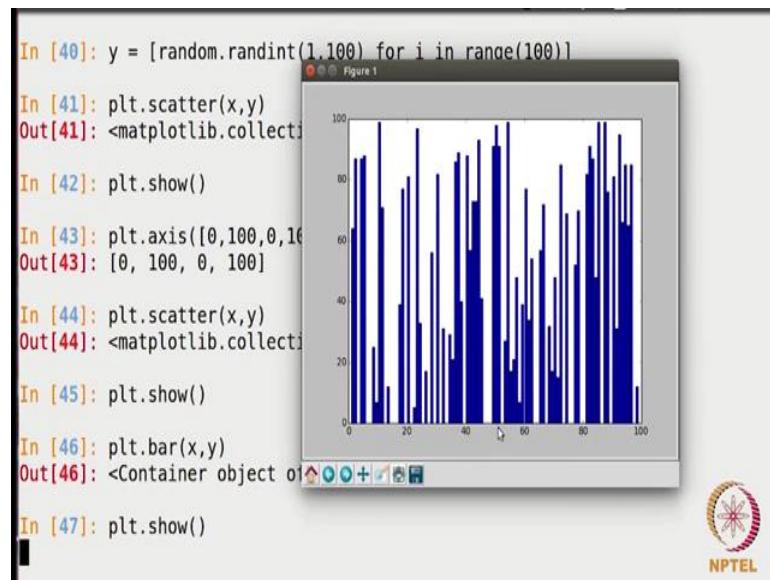
In [46]: plt.bar(x,y)
Out[46]: <Container object of 100 artists>
In [47]: plt.show()
```



A scatter plot showing 100 random points plotted against a 100x100 grid. The x-axis and y-axis both range from 0 to 100. The points are small black dots scattered across the plot area.

Now, let me show you another kind of plot that you can have using this package. If you want to bar chart or a few data, so you can use a function bar; plt.bar, I am going use the same x and y values here again, so that is all; plt.show.

(Refer Slide Time: 29:12)



We get a bar chart here, so by default we always get a blue colour, so axis is fine here 0 to 100 because we had set it already and so you can again clear on with customizing this plot as per your requirements. So, these were just few examples of the kind of flexibility that this package provides you. You can look into the documentation and look for more functions in as per your data as per your requirement.

Thank you.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture – 05**  
**Introduction to Social Networks**  
**Introduction to Networkx-1**

(Refer Slide Time: 00:05)

```
In [13]: import networkx  
In [14]: G=networkx.Graph()  
In [15]: G.add_node(1)  
In [16]: G.add_node(2)  
In [17]: G.add_node(3)  
In [18]: G.add_node(4)  
In [19]: G.add_node(5)  
In [20]: G.nodes()  
Out[20]: [1, 2, 3, 4, 5]  
In [21]: G.add_node(6)  
In [22]: G.nodes()  
Out[22]: [1, 2, 3, 4, 5, 6]  
In [23]:
```



I will now be introducing this graph package called networkx, with the help of which we can do a whole lot of graph analysis. I have pre-installed a networkx api on python, you may have to Google and understand how to install it on your computer. In case you find any difficulty, please let us know and we will help you out. Write to us on our email address and I will personally help you out.

So, we first say import networkx and the library function networkx gets loaded and then I say G = networkx.graph and then there is a graph that gets created and G gets assigned to it and now I say G add node 1 which means vertex 1 gets added. Similarly, I say add node 2 and so on, I will add some 5 nodes to it, 3, add node 4, add node 5 and when I say G.nodes, it will show me all the nodes that I have just now added, I will add one more node; let us say 6 and then I say G nodes, it will print all the nodes that are just now added.

(Refer Slide Time: 01:31)

```
In [21]: G.add_node(6)
In [22]: G.nodes()
Out[22]: [1, 2, 3, 4, 5, 6]
In [23]: G.add_edge(1,2)
In [24]: G.add_edge(1,3)
In [25]: G.add_edge(4,6)
In [26]: G.add_edge(5,4)
In [27]: G.add(edge(2,3
...: )
-----
AttributeError                                 Traceback (most recent call last)
<ipython-input-27-864277492ee1> in <module>()
----> 1 G.add(edge(2,3
      2 )

AttributeError: 'Graph' object has no attribute 'add'
In [28]:
```



Now, I can start adding the edges; how do I add the edges? G.add\_edge 1 , 2; G.add\_edge 1 , 3; G.add\_edge 4 , 6; G.add\_edge 5 , 4; G.add\_edge 2 , 3 and so on, I have added I did not close the bracket.

(Refer Slide Time: 02:00)

```
In [28]: G.add_edge(5,4)
In [29]: G.add(edge(2,3
...: )
-----
AttributeError                                 Traceback (most recent call last)
<ipython-input-27-864277492ee1> in <module>()
----> 1 G.add(edge(2,3
      2 )

AttributeError: 'Graph' object has no attribute 'add'
In [29]: G.edges()
Out[29]: [(1, 2), (1, 3), (2, 3), (4, 5), (4, 6)]
In [30]: G.add_edge(2,6)
In [31]: G.edges()
Out[31]: [(1, 2), (1, 3), (2, 3), (2, 6), (4, 5), (4, 6)]
In [32]: G=networkx.Graph()
```



I have added all the let me do it once again it 3, 1 error; G add underscore edge, there is a mistake there, 2 , 3 and that is it. So, I have added a few edges; I can see all the edges by giving this ,nd G.edges, you see the 5 edges that I just now added is here. Let me add more edge to show you that it is actually getting included, I will add the edge 2 , 6 which

is missing here; add edge 2 , 6 and I will display edges, one more edge gets added; 2 , 6 got added here as you can see alright.

Now, how do I see this addition; now before that you saw that I said G equals networkx.graph.

(Refer Slide Time: 02:55)

```
In [26]: G.add_edge(5,4)
In [27]: G.add(edge(2,3)
...)

-----
AttributeError                                     Traceback (most recent call last)
<ipython-input-27-864277492ee1> in <module>()
      1 G.add(edge(2,3)
      2 )

AttributeError: 'Graph' object has no attribute 'add'

In [28]: G.add_edge(2,3)

In [29]: G.edges()
Out[29]: [(1, 2), (1, 3), (2, 3), (4, 5), (4, 6)]

In [30]: G.add_edge(2,6)

In [31]: G.edges()
Out[31]: [(1, 2), (1, 3), (2, 3), (2, 6), (4, 5), (4, 6)]

In [32]: H=networkx.Graph()
```



If I were to define another graph namely H; should say H = networkx.graph given that tell me using this networkx often, there is one way to create an alias like thing; a macro like thing in c, if you remember.

(Refer Slide Time: 03:10)

```
In [27]: G.add(edge(2,3)
...)

-----
AttributeError                                     Traceback (most recent call last)
<ipython-input-27-864277492ee1> in <module>()
      1 G.add(edge(2,3)
      2 )

AttributeError: 'Graph' object has no attribute 'add'

In [28]: G.add_edge(2,3)

In [29]: G.edges()
Out[29]: [(1, 2), (1, 3), (2, 3), (4, 5), (4, 6)]

In [30]: G.add_edge(2,6)

In [31]: G.edges()
Out[31]: [(1, 2), (1, 3), (2, 3), (2, 6), (4, 5), (4, 6)]

In [32]: import networkx as nx

In [33]:
```



So, what I do is; I will say when I am importing, I will say import networkx and as nx; short form for network.

(Refer Slide Time: 03:17)

```
In [29]: G.edges()
Out[29]: [(1, 2), (1, 3), (2, 3), (4, 5), (4, 6)]

In [30]: G.add_edge(2,6)

In [31]: G.edges()
Out[31]: [(1, 2), (1, 3), (2, 3), (2, 6), (4, 5), (4, 6)]

In [32]: import networkx as nx

In [33]: H=nx.Graph()

In [34]: G.nodes()
Out[34]: [1, 2, 3, 4, 5, 6]

In [35]: G.edges()
Out[35]: [(1, 2), (1, 3), (2, 3), (2, 6), (4, 5), (4, 6)]

In [36]: import matplotlib.pyplot as plt

In [37]: nx.draw(G)

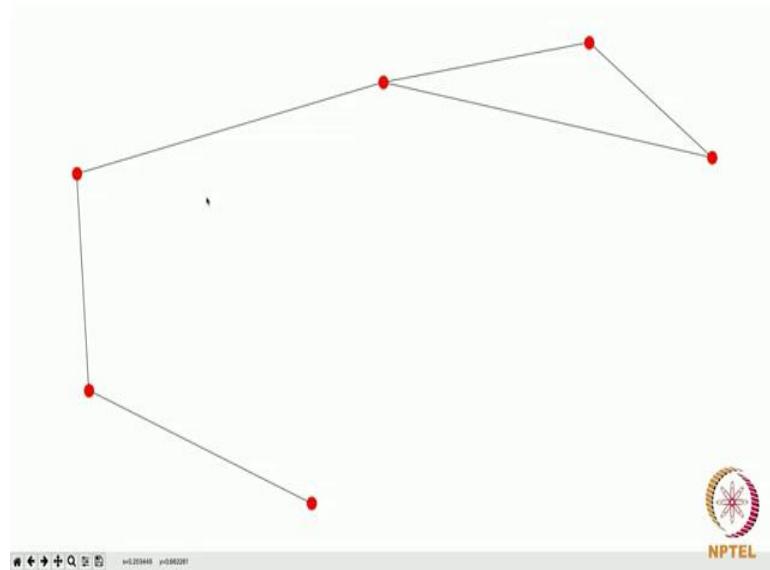
In [38]: plt.show()
```



Now, I can simply say H equals nx graph; instead of networkx graph, I can say nx.graph. Now a graph gets created, anyway I just have to illustrate this usage of import networkx as nx, this is a standard way to do it. If you Google online for code using network, the first line will be import networkx as nx, I did not want you people to get confused; so I thought I will clarify it here. Going back, I created a graph G with the following nodes and the following edges. I want to see how the graph looks like, there is a package for it and I should import that package which helps me visualize the graph; it is called matplotlib, in that I only want this particular sub library of called pyplot and I am going to import that and to avoid that this big ,nd in invoking functions within this, I will say as plt.

Now, I should visualize this graph; this is the way to visualize, it is slightly complicated but you will remember it given that you will be doing it very often. You will simply say; networkx.draw; what you want to draw? You want to draw the graph G; networkx G and then you say plt.show.

(Refer Slide Time: 04:50)



You see the graph that we just now created and we do not know which node is what.

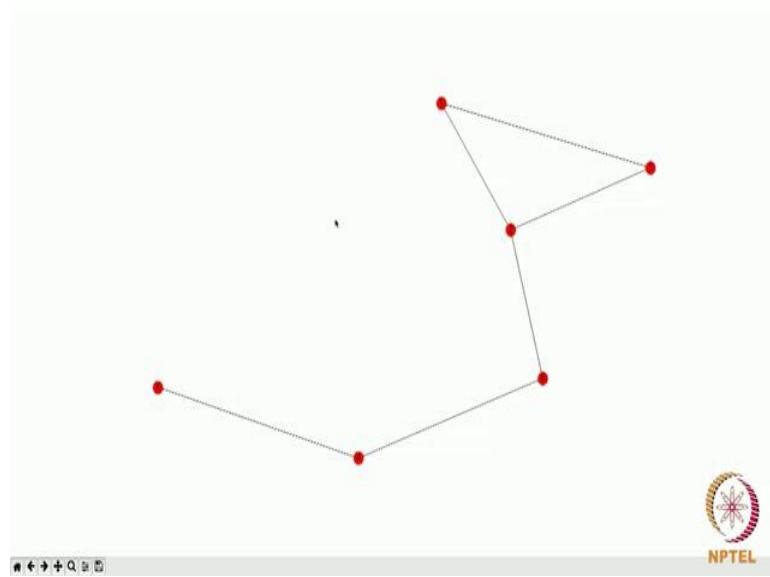
(Refer Slide Time: 04:57)

```
In [25]: nx.draw(G,with_labels=1)
In [26]: plt.show()
```



So, what will do is to display labels I just issue the same ,nd nx draw G and I say with labels equals true. If I include this, you will get; you will see what you will get, you will also get the vertex labels and now I issue the ,nd plt show.

(Refer Slide Time: 05:14)



And you will see that look at this, the graph that I input they are with labels right now, they were without labels in the beginning as you saw.

(Refer Slide Time: 05:24)

```
In [8]: Z=nx.complete_graph(10)
In [9]: Z.nodes()
Out[9]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
In [10]: print Z.edges()
[(0, 1), (0, 2), (0, 3), (0, 4), (0, 5), (0, 6), (0, 7), (0, 8), (0, 9), (1, 2), (1, 3),
(1, 4), (1, 5), (1, 6), (1, 7), (1, 8), (1, 9), (2, 3), (2, 4), (2, 5), (2, 6), (2,
7), (2, 8), (2, 9), (3, 4), (3, 5), (3, 6), (3, 7), (3, 8), (3, 9), (4, 5), (4, 6),
(4, 7), (4, 8), (4, 9), (5, 6), (5, 7), (5, 8), (5, 9), (6, 7), (6, 8), (6, 9), (7, 8),
(7, 9), (8, 9)]
In [11]: Z.edges()
```

Networkx has a lot of built in functions, I will show you one such built in function; let me say Z is equal to nx complete graph 10, this means it generates a complete graph on 10 vertices and assigns it to the variables Z. Let us see what happens, I say enter now a graph on 10 vertices is created, complete graph means it puts all possible edges between these 10 nodes a little bit of thinking will tell you that all possible friendships between 10

people is actually 45 alright. You can do the computation; you can take it as a homework and do it is a very straightforward question.

So, let me look at the number of nodes in Z; what are all the nodes in Z of course, there are 10 nodes here and the edges; print Z edges or simply Z edges, both of them will be output the edges; here are the edges.

(Refer Slide Time: 06:28)

```
(2, 8),  
(2, 9),  
(3, 4),  
(3, 5),  
(3, 6),  
(3, 7),  
(3, 8),  
(3, 9),  
(4, 5),  
(4, 6),  
(4, 7),  
(4, 8),  
(4, 9),  
(5, 6),  
(5, 7),  
(5, 8),  
(5, 9),  
(6, 7),  
(6, 8),  
(6, 9),  
(7, 8),  
(7, 9),  
(8, 9)]
```

In [12]: `print Z.edges()`



NPTEL

If I simply say that Z.edges, it will put it in a line like this; that is why I said print Z.edges; it will neatly show it like this.

(Refer Slide Time: 06:33)

```
(3, 9),  
(4, 5),  
(4, 6),  
(4, 7),  
(4, 8),  
(4, 9),  
(5, 6),  
(5, 7),  
(5, 8),  
(5, 9),  
(6, 7),  
(6, 8),  
(6, 9),  
(7, 8),  
(7, 9),  
(8, 9)]
```

In [12]: `print Z.edges()`

```
[(0, 1), (0, 2), (0, 3), (0, 4), (0, 5), (0, 6), (0, 7), (0, 8), (0, 9), (1, 2), (1, 3),  
(1, 4), (1, 5), (1, 6), (1, 7), (1, 8), (1, 9), (2, 3), (2, 4), (2, 5), (2, 6), (2,  
7), (2, 8), (2, 9), (3, 4), (3, 5), (3, 6), (3, 7), (3, 8), (3, 9), (4, 5), (4, 6), (4,  
7), (4, 8), (4, 9), (5, 6), (5, 7), (5, 8), (5, 9), (6, 7), (6, 8), (6, 9), (7, 8),  
(7, 9), (8, 9)]
```

In [13]: `Z.order()`



NPTEL

There are 45 edges here, how do I say it is 45? There is another ,nd called Z.order which stands for the number of nodes in the graph.

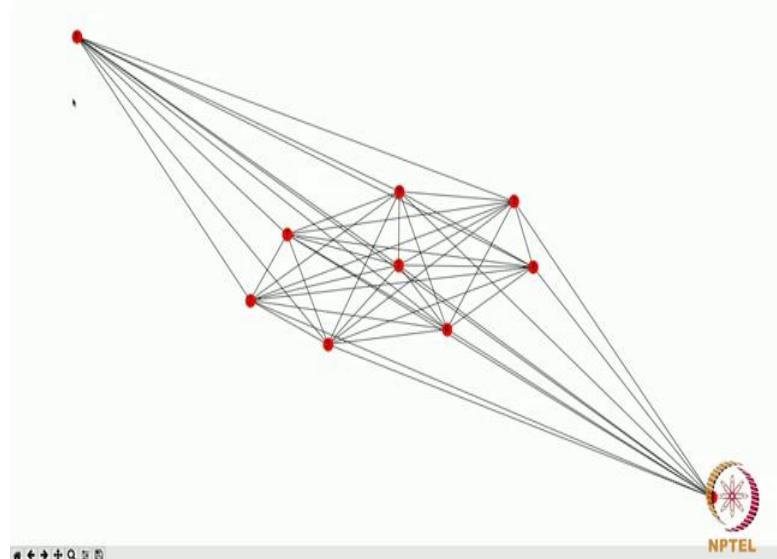
(Refer Slide Time: 06:42)

```
(5, 8),  
(5, 9),  
(6, 7),  
(6, 8),  
(6, 9),  
(7, 8),  
(7, 9),  
(8, 9)]  
  
In [12]: print Z.edges()  
[(0, 1), (0, 2), (0, 3), (0, 4), (0, 5), (0, 6), (0, 7), (0, 8), (0, 9), (1, 2), (1, 3)  
, (1, 4), (1, 5), (1, 6), (1, 7), (1, 8), (1, 9), (2, 3), (2, 4), (2, 5), (2, 6), (2,  
, 7), (2, 8), (2, 9), (3, 4), (3, 5), (3, 6), (3, 7), (3, 8), (3, 9), (4, 5), (4, 6),  
(4, 7), (4, 8), (4, 9), (5, 6), (5, 7), (5, 8), (5, 9), (6, 7), (6, 8), (6, 9), (7, 8),  
(7, 9), (8, 9)]  
  
In [13]: Z.order()  
Out[13]: 10  
  
In [14]: Z.size()  
Out[14]: 45  
  
In [15]: nx.draw(Z,with_labels=1)  
  
In [16]: plt.show()
```



And Z.size which tells you the number of edges in the graph, there are 45 edges. Now, let me visualize this graph Z; how do I visualize it, nx draw Z; I want it with labels, so I say with labels equals 1 and then I say plt show.

(Refer Slide Time: 07:02)



It will show me the graph and here is my complete graph on 10 vertices, as you can see every node is connected to every other node; that is what you mean by a complete graph

correct. Now, let us try seeing a complete graph on 100 vertices, let us see how it looks like; how does a complete graph on 100 vertices look like?

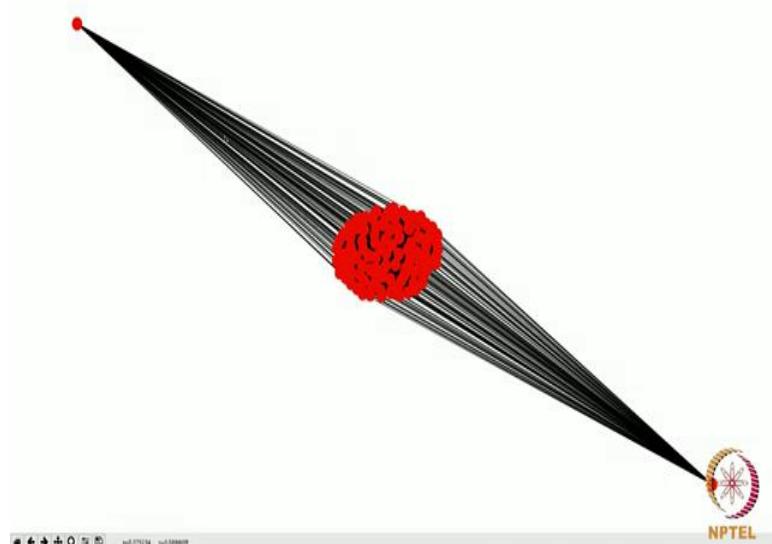
(Refer Slide Time: 07:22)

```
In [5]: H=nx.complete_graph(100)
In [6]: nx.draw(H)
/usr/local/lib/python2.7/site-packages/networkx/drawing/nx_pylab.py:126: MatplotlibDep
recationWarning: pyplot.hold is deprecated.
    Future behavior will be consistent with the long-time default:
    plot commands add elements without first clearing the
    Axes and/or Figure.
b = plt.ishold()
/usr/local/lib/python2.7/site-packages/networkx/drawing/nx_pylab.py:138: MatplotlibDep
recationWarning: pyplot.hold is deprecated.
    Future behavior will be consistent with the long-time default:
    plot commands add elements without first clearing the
    Axes and/or Figure.
plt.hold(b)
/usr/local/lib/python2.7/site-packages/matplotlib/__init__.py:917: UserWarning: axes.h
old is deprecated. Please remove it from your matplotlibrc and/or style files.
    warnings.warn(self.msg_depr_set % key)
/usr/local/lib/python2.7/site-packages/matplotlib/rcsetup.py:152: UserWarning: axes.h
old is deprecated, will be removed in 3.0
    warnings.warn("axes.hold is deprecated, will be removed in 3.0")
In [7]: plt.show()
```



I will say H equals nx, complete graph on 100 vertices then I will draw this H; I will not label it labeling will make it look clumsy, the typical warning message I told ignore this; plt show.

(Refer Slide Time: 07:51)



So, this is a complete graph on 100 vertices, so there are 100 vertices and all possible edges. You can ask me, what is this vertex and this vertex? These are all visualization

styles that are in built into this matplotlib. You can in fact change this visualization to; there are ways to do that. But as of now, all I want to show you people is that you can see that there are 100 vertices and they are connected to each other and this is how we visualize it.

(Refer Slide Time: 08:18)

```
In [4]: G=nx.gnp_random_graph(20,0.5)
```

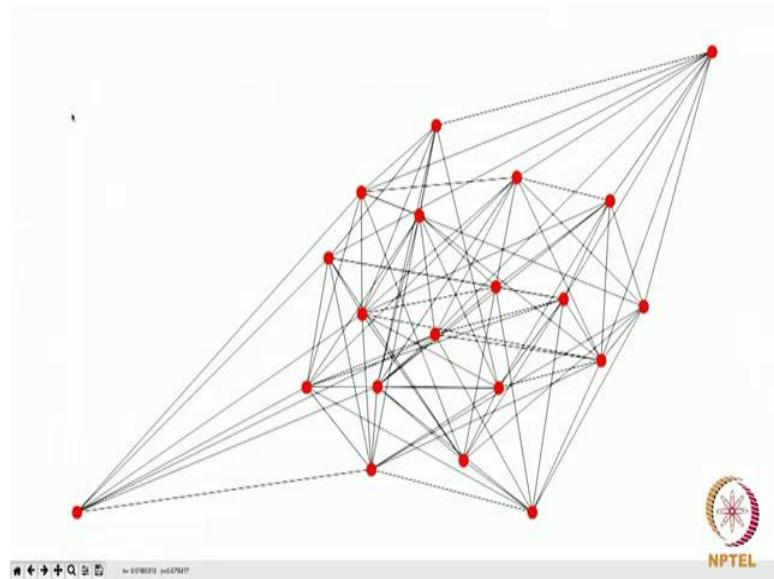
```
In [5]: nx.draw(G)
```



Networkx actually has a lot of built in functions, I will show you one such built in function which you may want to use more often. So, look at this I am going to generate a graph let us say  $G$ ; which is equal to, the ,nd is called gnp random graph; it is an  $nx$ ; gnp random graph on 20 vertices and I say 0.5 here, What does this mean? This means, I am generating a graph on 20 vertices and edges I am putting them with probability 0.5, which means for every edge; I put the edge with probability 0.5, I do not put the edge with probability 0.5; by that I mean, I toss a coin; if I get a head I put that edge, if I get a tail I do not put that edge.

How does the graph look like? A nice way to explain this is there are 20 people and there are friendships forming between them, they decide whether 2 people should become friends or not by tossing a coin; this is such an experiment that is what you mean by gnp graph. More details, I will be covering in my lecture do not worry much if you did not understand what I said here; all I am trying to do is generate a graph  $G$  with 20 nodes with some random edges; how does this graph look like? Lets us draw and see this plt show.

(Refer Slide Time: 09:50)

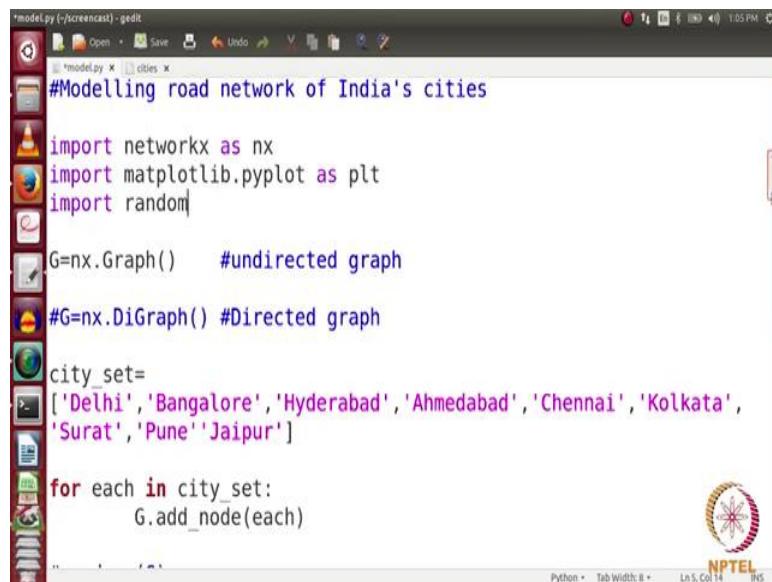


This is how the graph looks like on 20 nodes, where edges are put uniformly at random. Unlike the previous case, this is not a complete graph as you can see; for example, there is no edge between this vertex and this vertex, this is not a complete graph; this is just a graph on 20 nodes with random edges.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture – 06**  
**Introduction to Social Networks**  
**Introduction to Networkx-2**

(Refer Slide Time: 00:08)



```
#Modelling road network of India's cities

import networkx as nx
import matplotlib.pyplot as plt
import random

G=nx.Graph()      #undirected graph

#G=nx.DiGraph() #Directed graph

city_set=
['Delhi','Bangalore','Hyderabad','Ahmedabad','Chennai','Kolkata',
'Surat','Pune','Jaipur']

for each in city_set:
    G.add_node(each)
```

Hello everybody, in the screen cast we are going to do a very interesting programming assignment, what we want to tell you through the screen cast is how do you pick a real world problem and model it with the help of python and networkx. So, we will be using both python as well as networkx in the screen cast. So, that you can learn how to put both of them together and use the real world problem which we will be picking is the network of cities.

So, we have a lot of cities in India. So, we represent every city as a node and then we have edges between these cities which are the roads connecting these cities. So, we will be assuming that between 2 cities there is one road though there can be many. So, we pick one and then there might be a travelling time associated with each of the road. So, you can visualize this as a network where nodes are the cities and edges are the roads connecting them some roads might be good some roads might be bad some roads can be congested some roads can be free and so on which affect their travelling time if you

might if you would have heard of the travelling salesman problem you would be very easily able to imagine it, but its otherwise also its easy to imagine. So, you have a network where there are cities and there are roads connecting the cities. So, we will be making this network with the help of python and networkx will be visualizing it and we will be looking at various properties of this network.

So, let us get started how do we do it? So, what we are going to do is we are going to model the road network of India let say India's cities, let us say we are going to model the road network of cities in India as I have already told you. So, what is the graph here? So, let me first save this file of this folder screen cast here let say I save it with the name cities.Mumbai for better event let say model.py. So, I have a model.py here. So, let us see how do we model the road network? So, let me first make a graph. So, for making a graph I need the networkx package which needs to be imported.

So, we import networkx as nx. So, now, we can use it and we are going to take the graph s undirected. So, we are going to assume that if there is a road between let say Delhi to Mumbai. So, assume there is a road between Delhi and Mumbai. So, it will take you equal number e it will take you equal time to go from Delhi to Mumbai and from Mumbai to Delhi. So, hence we can have one edge and the graph is undirected. So, here we are going to make an undirected graph G equals to nx.graph. So, when you write G equals to nx.graph it automatically creates an undirected graph for you.

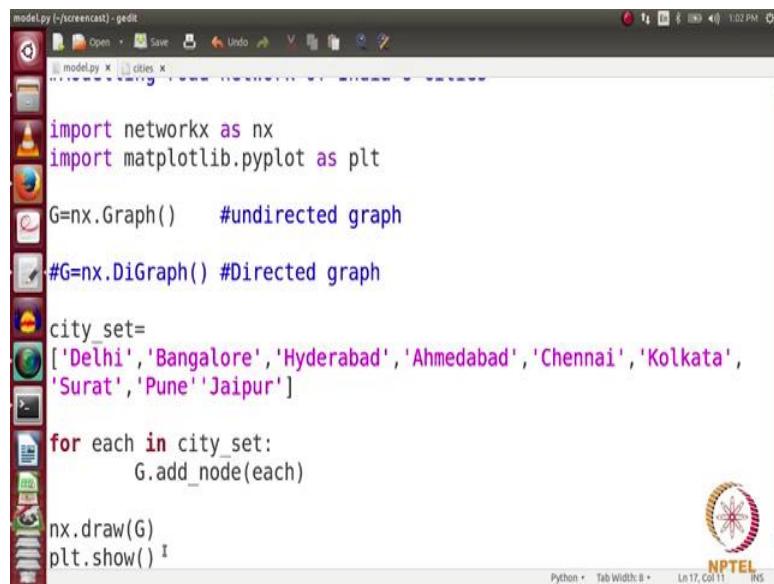
So, let me tell you there if the possibility of making directed graphs as well. So, a simple code by which you can make a directed graph is nx.digraph. So, if you use this command and next of digraph there will be a directed network created. So, it will be a directed graph here we are working with undirected graph. So, just come into this statement here. So, we have made a graph G equals to nx.graph here now I have a graph what I want to do is I want to add a nodes to this graph and every node should represent a city.

So, let us have an array of city here let say city set which is the set of cities where from where we will be choosing the cities. So, I have a file here just I looked at Wikipedia and looked at some of the cities let us pick some cities form here let us say, let us pick all these cities control c and let say I will put all these cities here. So, this city may insert strings. So, one thing well we are working with strings you will have to put these inputs.

So, I will put all of the inputs. So, I have a set of cities here. So, this is the set of cities now we want to create nodes in this graph which should be the cities.

So, what we can do is. So, we have a list here we have to create cities and the values of the cities from should be from this list what we can simply do is for each in city set. So, if you would have gone through the module of using libs in python you would know that each is an iterator which will go through each of the elements of these lists cities set one by one. So, for each in city set what we want to do is we want to add a node this network. So, we do G.add\_node which is the command for adding node and each. So, we have these nodes and getting the network now I want to visualize this network we know that the function which is used for visualization nx.draw and for visualizing we need to import one more module here draw import matplotlib.pyplot as plt.

(Refer Slide Time: 06:46)



The screenshot shows a terminal window titled "model.py (~/screencast) - gedit". The code inside the terminal is as follows:

```
import networkx as nx
import matplotlib.pyplot as plt

G=nx.Graph()      #undirected graph

#G=nx.DiGraph() #Directed graph

city_set=
['Delhi','Bangalore','Hyderabad','Ahmedabad','Chennai','Kolkata',
'Surat','Pune','Jaipur']

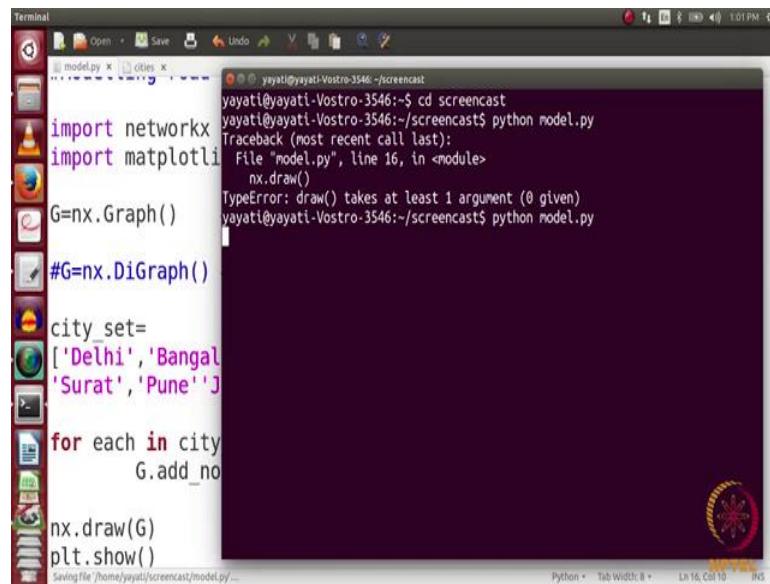
for each in city_set:
    G.add_node(each)

nx.draw(G)
plt.show()
```

The code imports networkx and matplotlib.pyplot, creates an undirected graph G, defines a list of cities, and adds each city to the graph. It then draws the graph and displays it using plt.show().

So, what we can do here is nx.draw and then plt.show. So, let us run this and let us see how does it work? So, I open up terminal window here.

(Refer Slide Time: 07:00)

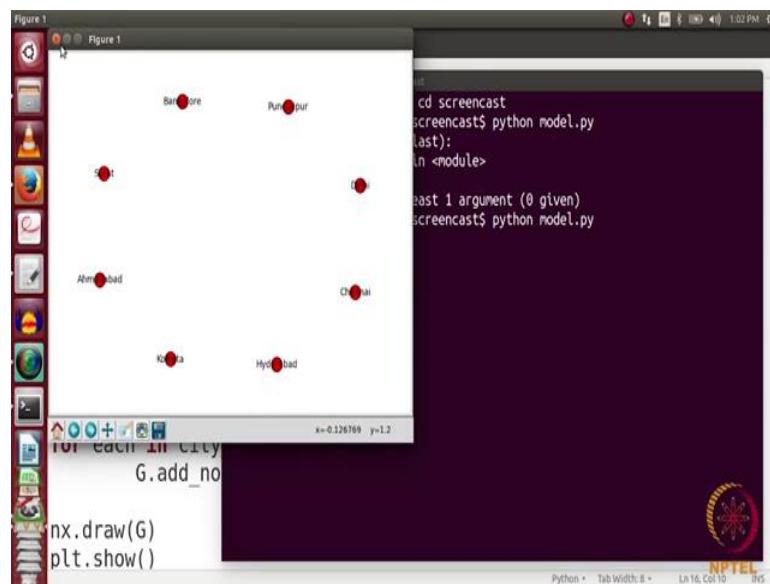


A screenshot of a Linux desktop environment. On the left, there's a file manager window titled 'model.py' showing Python code. On the right, there's a terminal window with the following text:

```
yayati@yayati-Vostro-3546:~$ cd screencast
yayati@yayati-Vostro-3546:~/screencast$ python model.py
Traceback (most recent call last):
  File "model.py", line 16, in <module>
    nx.draw()
TypeError: draw() takes at least 1 argument (0 given)
yayati@yayati-Vostro-3546:~/screencast$
```

So, I will mainly we will be in this terminal window here. So, we are in the home. So, we will first go to this folder, whose name was screen cast and then we run this python; python model.py some problem. So, it says that the draw function takes at least 1 argument. So, when we are nx.draw we need to pass the graph here which is G and then run it.

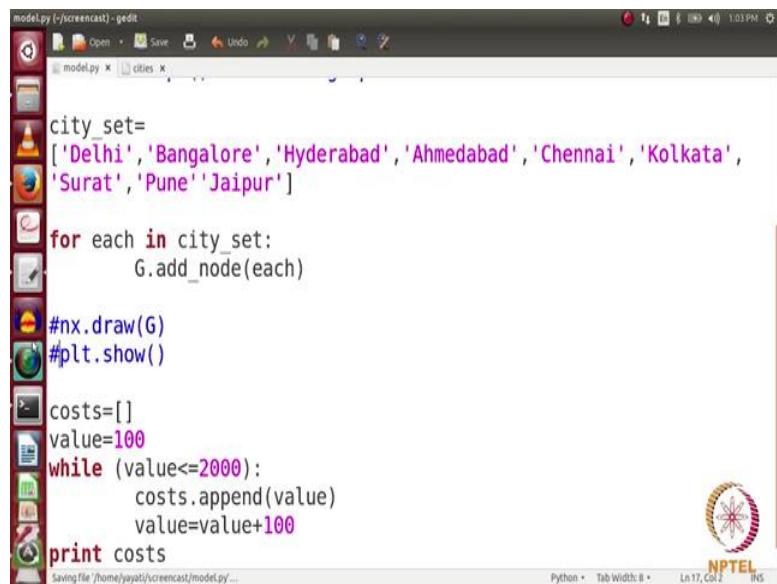
(Refer Slide Time: 07:35)



So you can see here; that we have a network created where each node is one city in this network.

So, we added 1, 2, 3, 4, 5, 6, 7, 8 cities, we have added in this network perfect. So, now, we have a graph in which there are cities, there are 8 cities and next n is to add edges between these cities of roads between these cities and every road should have a have travelling cost associated with it. So, this travelling cost we represent it by the weight of an edge. So, we have to add edges here. So, how do we add edges it is the question. So, we are going to add these edges randomly for now how do we do that.

(Refer Slide Time: 08:15)



```
model.py (~/-/screencast) - gedit
city_set=
['Delhi', 'Bangalore', 'Hyderabad', 'Ahmedabad', 'Chennai', 'Kolkata',
'Surat', 'Pune', 'Jaipur']

for each in city_set:
    G.add_node(each)

#nx.draw(G)
#plt.show()

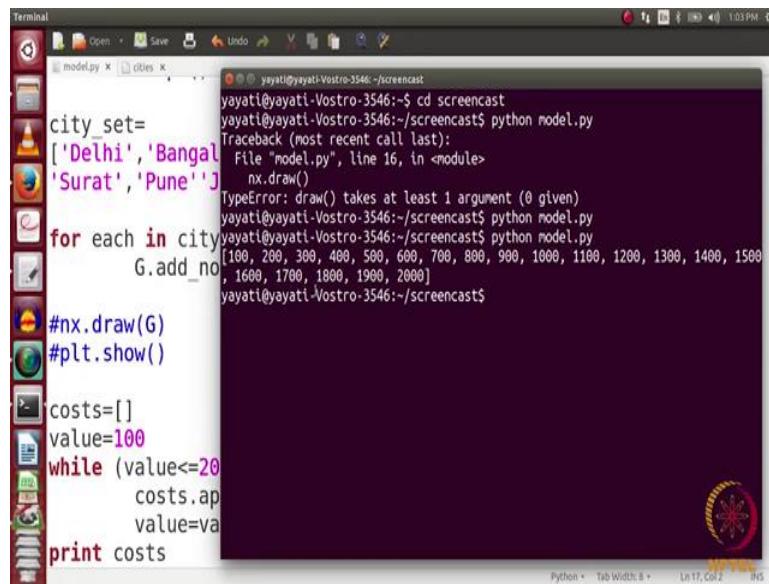
costs=[]
value=100
while (value<=2000):
    costs.append(value)
    value=value+100
print costs

Saving file /home/yayati/screencast/model.py...
Python Tab Width: 8 Ln 17, Col 2 In5
```

So, first of all we have a let say array costs what we are going to do is we are going to add some random edges in this network and each edge will have a random value random weight random travelling cost which will be picked from this lift cost. So, we want to first of all populate this list costs so that we can pick values form here. So, what do we do? Let us say I will add some 20 values here let say starting from 100. So, what I will be doing while value is less than or equals to 2000 costs.append and I will do here value and then value equals to value plus 1. So, it will iterate like 100 sorry it will iterate like 100, 200, 300, 400, it will keep up entering the value and it will go up to 2000.

Let us see what this array looks like and we will just comment these 2 things.

(Refer Slide Time: 09:34)



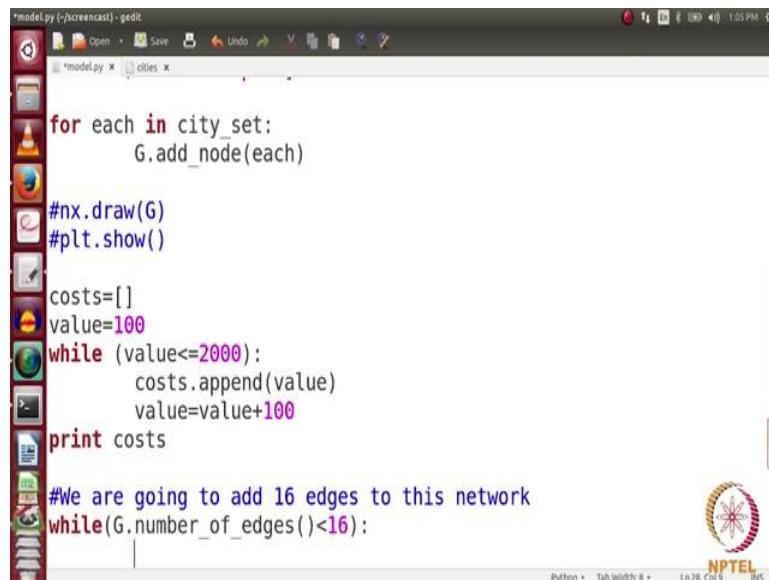
```
Terminal
modelpy X cities X
yayati@yayati-Vostro-3546:~/screencast
city_set=
['Delhi', 'Bangal',
'Surat', 'Pune']
for each in city_set:
    G.add_node(each)
#nx.draw(G)
#plt.show()

costs=[]
value=100
while (value<=2000):
    costs.append(value)
    value=value+100
print costs

yayati@yayati-Vostro-3546:~/screencast$ python model.py
[100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500,
1600, 1700, 1800, 1900, 2000]
yayati@yayati-Vostro-3546:~/screencast$
```

So, this is how our array will look like it has values from 100 to 2000 these are array costs. So, now, we are going to add edges in this network assume that we want to add. So, there are 8 cities. So, for now let us add some 16 edges. So, we are going to add 16 edges through this network.

(Refer Slide Time: 09:56)



```
*modelpy (~)/screencast - gedit
modelpy X cities X
for each in city_set:
    G.add_node(each)

#nx.draw(G)
#plt.show()

costs=[]
value=100
while (value<=2000):
    costs.append(value)
    value=value+100
print costs

#We are going to add 16 edges to this network
while(G.number_of_edges()<16):
```

So, count is in English. So, we are going to add 16 edges to this network. So, till when your port should run while. So, for looking at the number of edges you have this code G.number of edges is less than 16.

So, this code will keep running till the number of edges in your network become 16 what you are going to do is we have choose 2 nodes randomly. So, for choosing a node randomly we note we need to import the function random. So, we import the function random here import random so that we can use it.

(Refer Slide Time: 10:58)

```

model.py (~/screencast) · gedit
costs=[]
value=100
while (value<=2000):
    costs.append(value)
    value=value+100
print costs

#We are going to add 16 edges to this network
while(G.number_of_edges()<16):
    c1=random.choice(G.nodes())
    c2=random.choice(G.nodes())
    if c1!=c2 and G.has_edge(c1,c2)==0:
        w=random.choice(costs)
        G.add_edge(c1,c2,weight=w)

nx.draw(G)
plt.show()

```

We come down. So, we choose city one equals to. So, if you want to choose a value from a list randomly. So, we have random.choice and we have function G.nodes. So, G.nodes give you a set which has already nodes in these networks. So, we are going to choose one of these nodes randomly that is their city 1 and then we choose city 2 which is again random.choice and again G.nodes.

Now, one thing to note here is these 2 cities should not be same. So, only if c 1 is not equals to c 2, we are going to move next what we are going to do is we are going to choose a weight for the edge. So, let say weight which is the travelling cost now this is nothing, but random.choice again a random function and it will pick a random value from our array costs also. So, the one condition for creating an edges both of these cities should not be equal and the second condition is there should not be an edge which is already present between these 2 nodes.

So, for that we have function and G.has underscore edge it tells you whether there is an edge between 2 nodes c 1 comma c 2. So, has edge c 1 comma c 2 should be equal to 0 there should be no edge between them. So, in that case we will choose a weight here w

equals to random.choice costs and then we will add an edge in this network G.add edge and we add edge from c 1 comma c 2 and the weight of this edge is w which was a random number. So, this code keeps running this loop keeps running till we have 16 edges in the network.

Again I want to see these networks. So, I use nx.draw G and then plt.show.

(Refer Slide Time: 13:06)

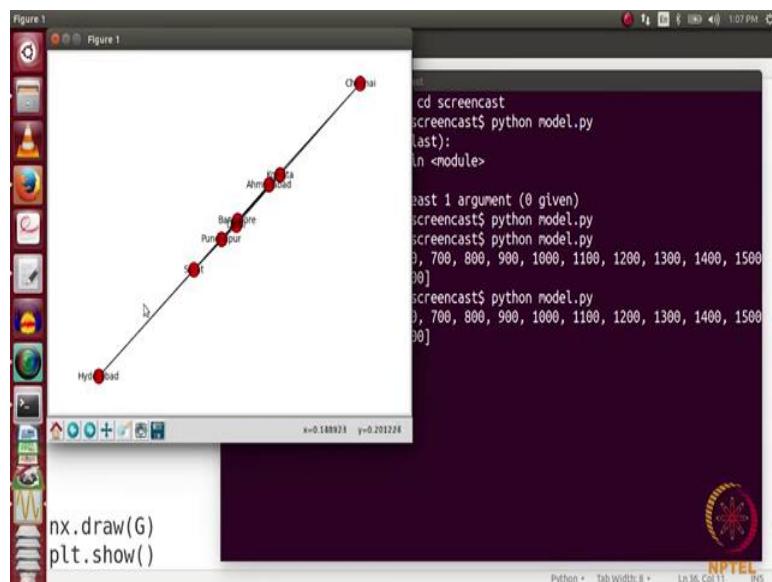
```

Terminal
model.py x | cities x
yayati@yayati-Vostro-3546:~/screencast
yayati@yayati-Vostro-3546:~/screencast$ cd screencast
yayati@yayati-Vostro-3546:~/screencast$ python model.py
Traceback (most recent call last):
  File "model.py", line 16, in <module>
    nx.draw()
TypeError: draw() takes at least 1 argument (0 given)
yayati@yayati-Vostro-3546:~/screencast$ python model.py
yayati@yayati-Vostro-3546:~/screencast$ python model.py
[100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500
 , 1600, 1700, 1800, 1900, 2000]
#We are going to
yayati@yayati-Vostro-3546:~/screencast$ python model.py
while(G.number_o
      _ , 1600, 1700, 1800, 1900, 2000]
      cl=random
      c2=random
      if cl!=c
nx.draw(G)
plt.show()

```

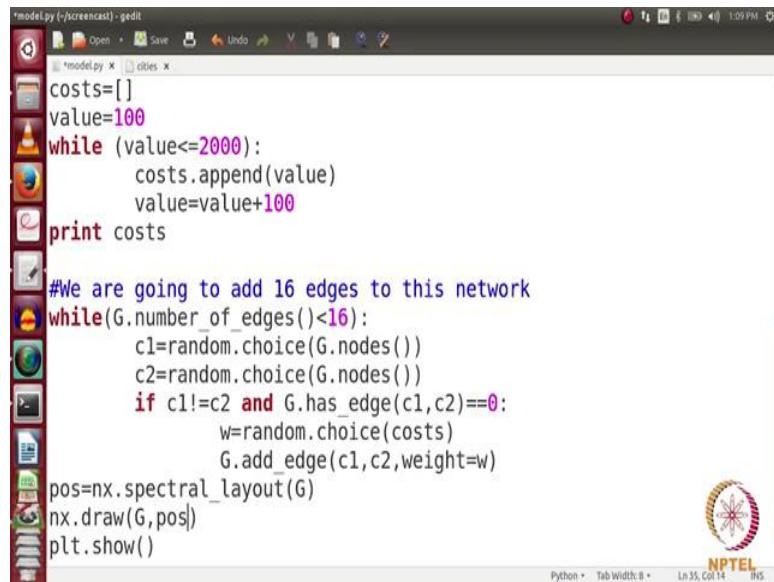
Let us execute this in c.

(Refer Slide Time: 13:09)



So, you see here. So, the edges are not very clear some of the edges are overlapping here. So, what we can do for this is we can change the layout of this graph. So, one of the layout that we can use is the spectral layout. So, for using this spectral layout we have a command.

(Refer Slide Time: 13:40)

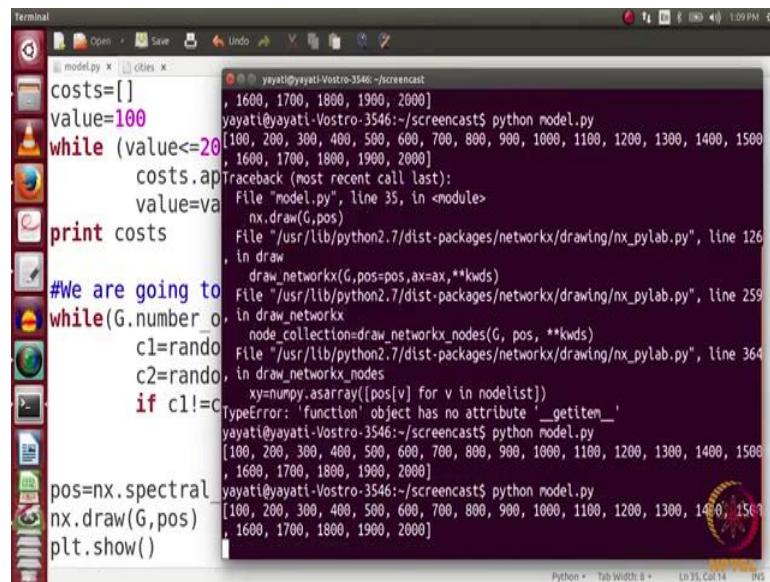


```
model.py (~/screen cast) - gedit
costs=[]
value=100
while (value<=2000):
    costs.append(value)
    value=value+100
print costs

#We are going to add 16 edges to this network
while(G.number_of_edges()<16):
    c1=random.choice(G.nodes())
    c2=random.choice(G.nodes())
    if c1!=c2 and G.has_edge(c1,c2)==0:
        w=random.choice(costs)
        G.add_edge(c1,c2,weight=w)
pos=nx.spectral_layout(G)
nx.draw(G,pos)
plt.show()
```

So, let us say we want to choose a layout pos equals to nx.draw underscore spectral (Refer Time: 13:52) this p o s here (Refer Time: 13:54). So, the command here is nx.and we want a spectral layout. So, we using spectral underscore layout we will pass a graph G here and while drawing it we use nx.draw pos.

(Refer Slide Time: 14:20)



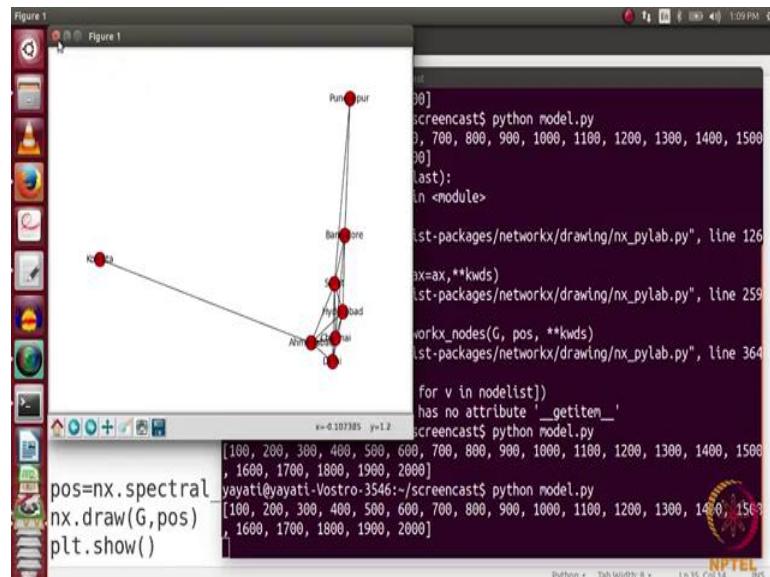
```
Terminal
model.py x cities x
yayati@yayati-Vostro-3546:~/screencast
costs=[]
value=100
while (value<=20
    costs.append(value)
    value+=1
print costs
#We are going to
while(G.number_of_nodes()<20):
    c1=random.randint(0,19)
    c2=random.randint(0,19)
    if c1!=c2:
        G.add_edge(c1,c2)
pos=nx.spectral_layout(G)
nx.draw(G,pos)
plt.show()

yayati@yayati-Vostro-3546:~/screencast$ python model.py
[100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500
, 1600, 1700, 1800, 1900, 2000]
yayati@yayati-Vostro-3546:~/screencast$ python model.py
[100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500
, 1600, 1700, 1800, 1900, 2000]
yayati@yayati-Vostro-3546:~/screencast$ python model.py
[100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500
, 1600, 1700, 1800, 1900, 2000]
yayati@yayati-Vostro-3546:~/screencast$ python model.py
[100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500
, 1600, 1700, 1800, 1900, 2000]

Python+ Tab Width: 8+ Ln 35, Col 14 185
```

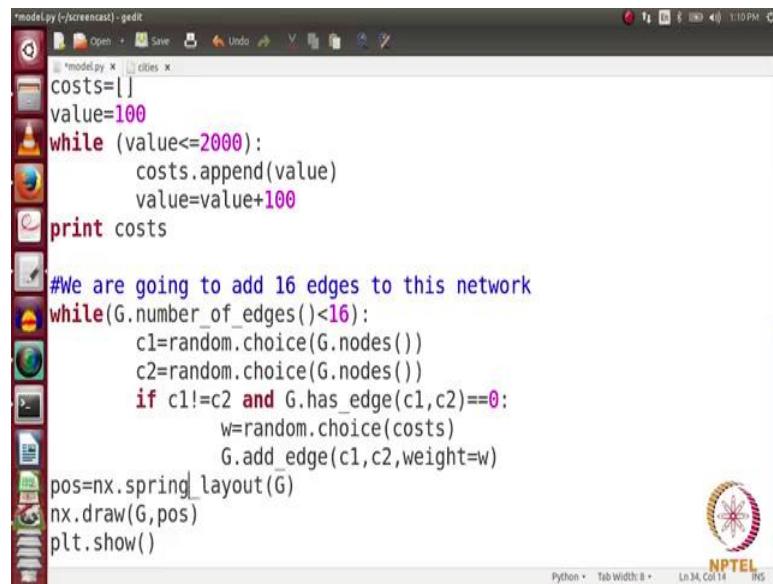
So, let us now visualize these graph python model.py.

(Refer Slide Time: 14:22)



So, when we see it looks better than before ok we can also use some other layouts.

(Refer Slide Time: 14:35)

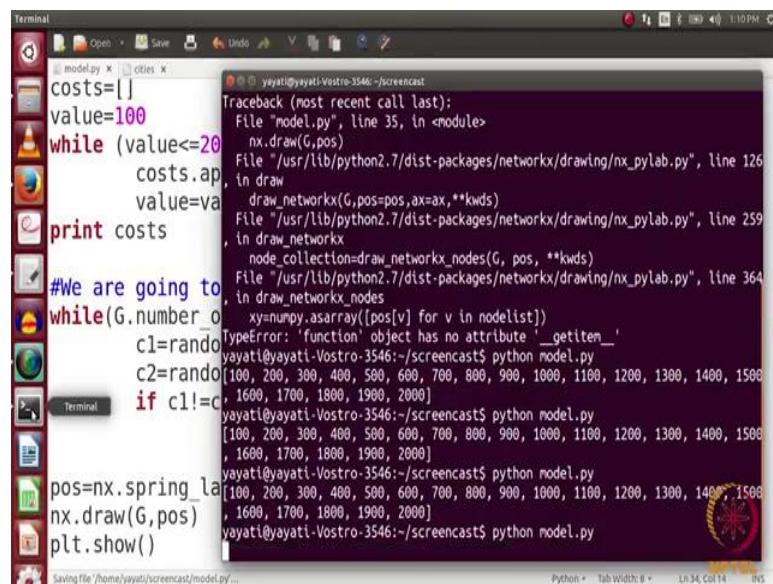


```
model.py (~/screencast) - gedit
costs=[]
value=100
while (value<=2000):
    costs.append(value)
    value=value+100
print costs

#We are going to add 16 edges to this network
while(G.number_of_edges()<16):
    c1=random.choice(G.nodes())
    c2=random.choice(G.nodes())
    if c1!=c2 and G.has_edge(c1,c2)==0:
        w=random.choice(costs)
        G.add_edge(c1,c2,weight=w)
pos=nx.spring_layout(G)
nx.draw(G,pos)
plt.show()
```

Let us say; let us try to use a spring layout look very nice and circular layout might look better here.

(Refer Slide Time: 14:37)

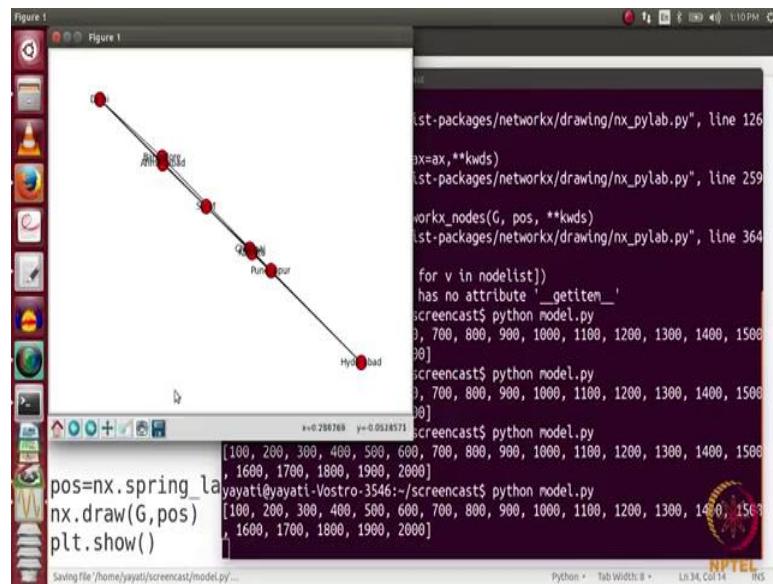


```
Terminal
model.py (~/screencast)
costs=[]
value=100
while (value<=2000):
    costs.append(value)
    value=value+100
print costs

#We are going to add 16 edges to this network
while(G.number_of_edges()<16):
    c1=random.choice(G.nodes())
    c2=random.choice(G.nodes())
    if c1!=c2 and G.has_edge(c1,c2)==0:
        w=random.choice(costs)
        G.add_edge(c1,c2,weight=w)
pos=nx.spring_layout(G)
nx.draw(G,pos)
plt.show()
```

Traceback (most recent call last):  
File "model.py", line 35, in <module>  
 nx.draw(G,pos)  
File "/usr/lib/python2.7/dist-packages/networkx/drawing/nx\_pylab.py", line 126  
 in draw  
 draw\_networkx(G, pos=pos, ax=ax, \*\*kwds)  
File "/usr/lib/python2.7/dist-packages/networkx/drawing/nx\_pylab.py", line 259  
 in draw\_networks  
 node\_collection=draw\_networkx\_nodes(G, pos, \*\*kwds)  
File "/usr/lib/python2.7/dist-packages/networkx/drawing/nx\_pylab.py", line 364  
 in draw\_networkx\_nodes  
 xy=numpy.asarray([pos[v] for v in nodelist])  
TypeError: 'function' object has no attribute '\_getitem\_'  
yayati@yayati-Vostro-3546:~/screencast\$ python model.py  
[100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500  
, 1600, 1700, 1800, 1900, 2000]  
yayati@yayati-Vostro-3546:~/screencast\$ python model.py  
[100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500  
, 1600, 1700, 1800, 1900, 2000]  
yayati@yayati-Vostro-3546:~/screencast\$ python model.py  
[100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500  
, 1600, 1700, 1800, 1900, 2000]  
yayati@yayati-Vostro-3546:~/screencast\$ python model.py

(Refer Slide Time: 14:37)



(Refer Slide Time: 14:43)

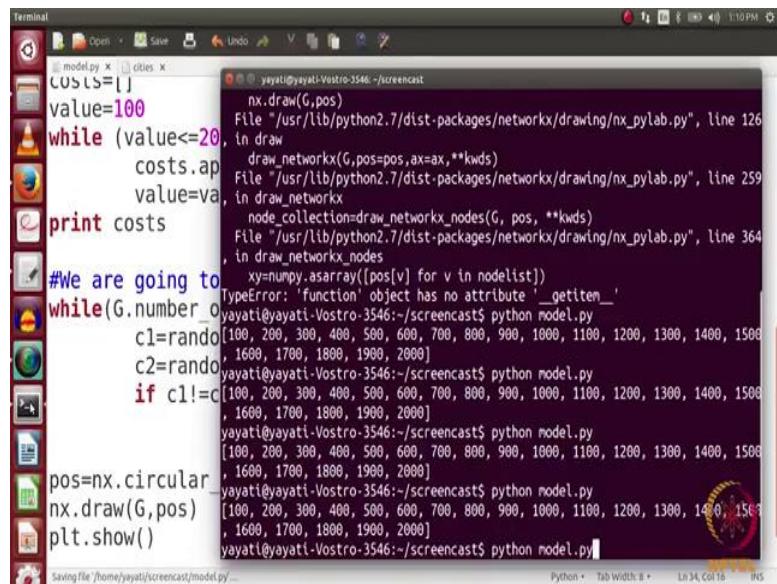
The figure shows a Linux desktop environment with a code editor window titled "model.py" in the foreground. The code in the editor generates a network graph with 16 nodes and 16 edges. It uses NetworkX's circular layout and draws the graph. The code editor interface includes tabs for "Open", "Save", "Undo", and "Redo".

```
costs=[]
value=100
while (value<=2000):
    costs.append(value)
    value=value+100
print costs

#We are going to add 16 edges to this network
while(G.number_of_edges()<16):
    c1=random.choice(G.nodes())
    c2=random.choice(G.nodes())
    if c1!=c2 and G.has_edge(c1,c2)==0:
        w=random.choice(costs)
        G.add_edge(c1,c2,weight=w)

pos=nx.circular_layout(G)
nx.draw(G,pos)
plt.show()
```

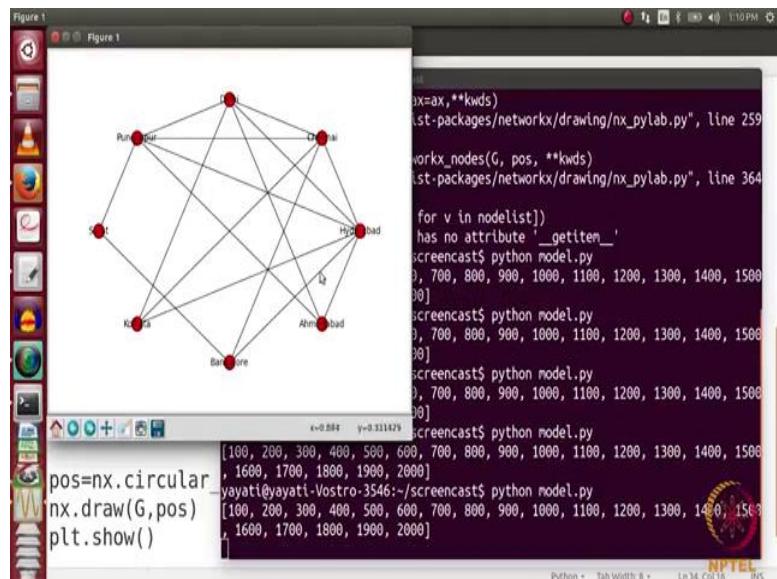
(Refer Slide Time: 14:45)



```
COSTS=1
value=100
while (value<=20,
    costs.append(value)
    value+=1
print costs
#We are going to
while(G.number_of_nodes()<8):
    cl=rando([100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500
              , 1600, 1700, 1800, 1900, 2000])
    c2=rando([100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500
              , 1600, 1700, 1800, 1900, 2000])
    if cl!=c2:
        pos=nx.circular_layout(G)
        nx.draw(G,pos)
        plt.show()
Saving file /home/yayati/screencast/model.py...  
yayati@yayati-Vostro-3546:~/screencast$ python model.py  
[100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500
              , 1600, 1700, 1800, 1900, 2000]  
yayati@yayati-Vostro-3546:~/screencast$ python model.py  
[100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500
              , 1600, 1700, 1800, 1900, 2000]  
yayati@yayati-Vostro-3546:~/screencast$ python model.py  
[100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500
              , 1600, 1700, 1800, 1900, 2000]  
yayati@yayati-Vostro-3546:~/screencast$ python model.py  
[100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500
              , 1600, 1700, 1800, 1900, 2000]  
yayati@yayati-Vostro-3546:~/screencast$ python model.py  
[100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500
              , 1600, 1700, 1800, 1900, 2000]
```

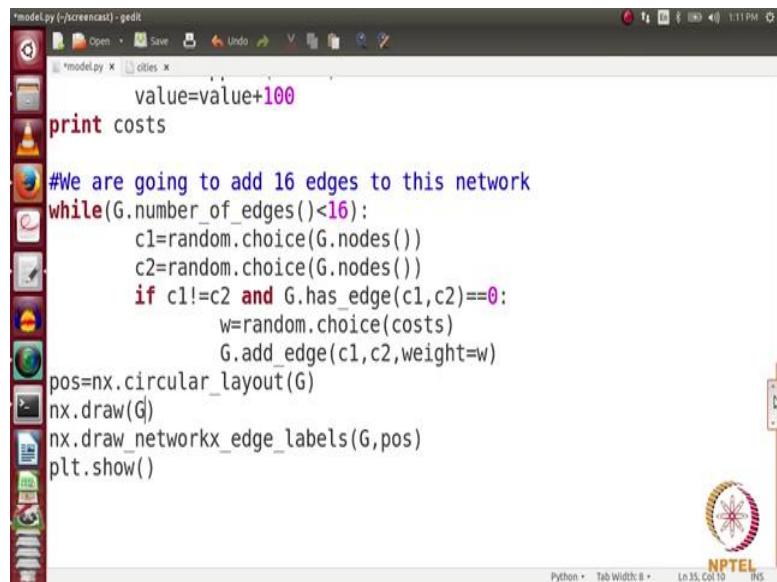
Let us try the circular layout. So, this one looks perfect. So, we have drawn this graph in the circular layout.

(Refer Slide Time: 14:46)



And it looks nice here. So, there are 8 cities and they have 16 edges and each of this edge has some edge weight associated with it. So, you might want to look at this edge weights as well. So, for labeling these edges with their edge weights, so here we have drawn the networkx and let us draw their edge weights are also here.

(Refer Slide Time: 15:11)

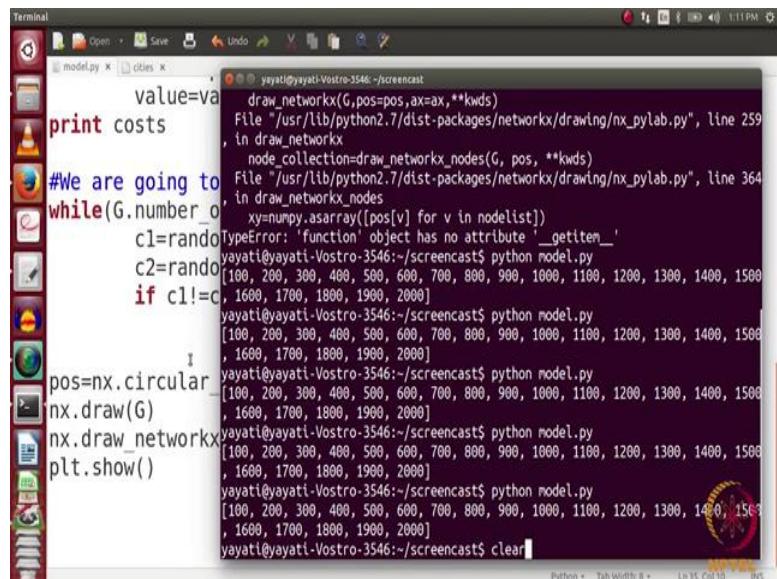


```
model.py (~/screencast) - gedit
value=value+100
print costs

#We are going to add 16 edges to this network
while(G.number_of_edges()<16):
    c1=random.choice(G.nodes())
    c2=random.choice(G.nodes())
    if c1!=c2 and G.has_edge(c1,c2)==0:
        w=random.choice(costs)
        G.add_edge(c1,c2,weight=w)
pos=nx.circular_layout(G)
nx.draw(G)
nx.draw_networkx_edge_labels(G,pos)
plt.show()
```

So, we have this command nx.draw underscore networkx underscore edge underscore labels and then we pass here G comma p o s.

(Refer Slide Time: 15:32)



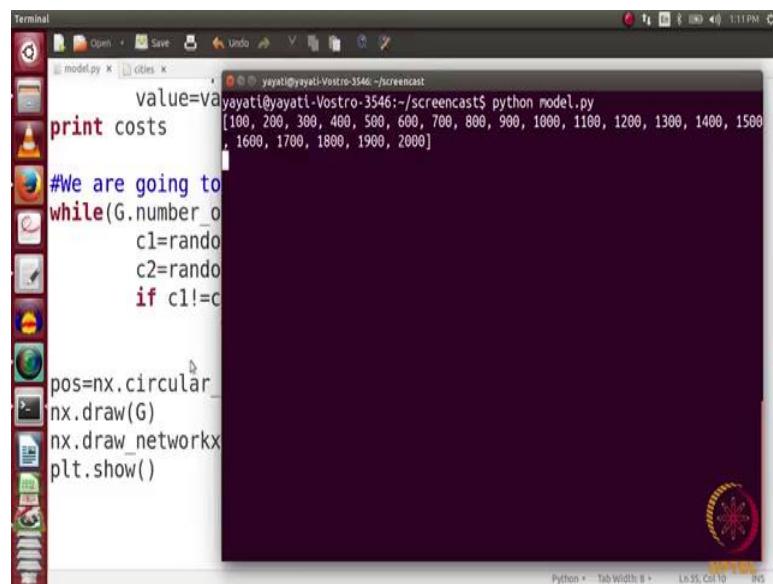
```
Terminal
model.py (~/screencast)
value=value+100
print costs

#We are going to add 16 edges to this network
while(G.number_of_edges()<16):
    c1=random.choice(G.nodes())
    c2=random.choice(G.nodes())
    if c1!=c2 and G.has_edge(c1,c2)==0:
        w=random.choice(costs)
        G.add_edge(c1,c2,weight=w)
pos=nx.circular_layout(G)
nx.draw(G)
nx.draw_networkx_edge_labels(G,pos)
plt.show()

yayati@yayati-Vostro-3546:~/screencast$ python model.py
[100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500
, 1600, 1700, 1800, 1900, 2000]
yayati@yayati-Vostro-3546:~/screencast$ python model.py
[100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500
, 1600, 1700, 1800, 1900, 2000]
yayati@yayati-Vostro-3546:~/screencast$ python model.py
[100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500
, 1600, 1700, 1800, 1900, 2000]
yayati@yayati-Vostro-3546:~/screencast$ python model.py
[100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500
, 1600, 1700, 1800, 1900, 2000]
yayati@yayati-Vostro-3546:~/screencast$ python model.py
[100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500
, 1600, 1700, 1800, 1900, 2000]
yayati@yayati-Vostro-3546:~/screencast$ clear
```

Let us see how does it work and then let me just clear it.

(Refer Slide Time: 15:36)

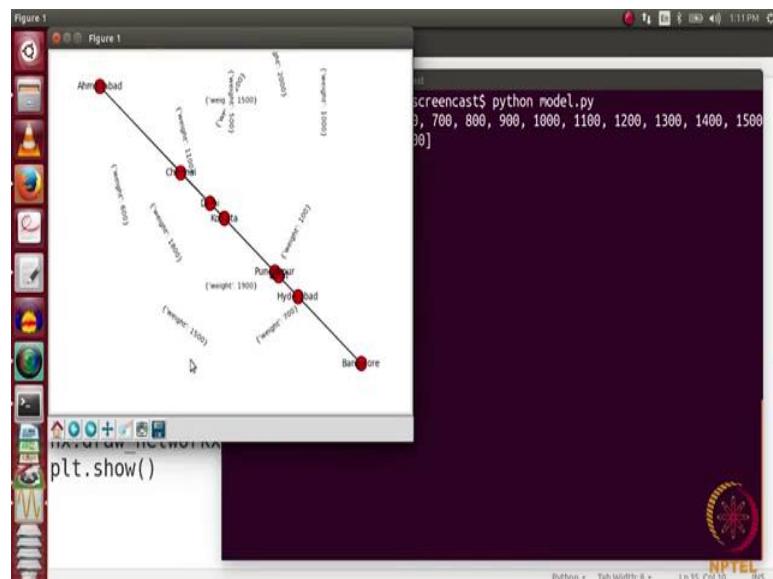


A screenshot of a Linux desktop environment. In the foreground, a terminal window titled 'Terminal' is open, showing Python code for generating a network graph. The code defines a function 'model' that creates a graph 'G' with nodes and edges. It uses a while loop to add edges between randomly selected nodes until a certain number of edges are reached. The graph is then drawn using NetworkX and Matplotlib. The terminal window also shows the command 'python model.py' being run and the resulting edge weights.

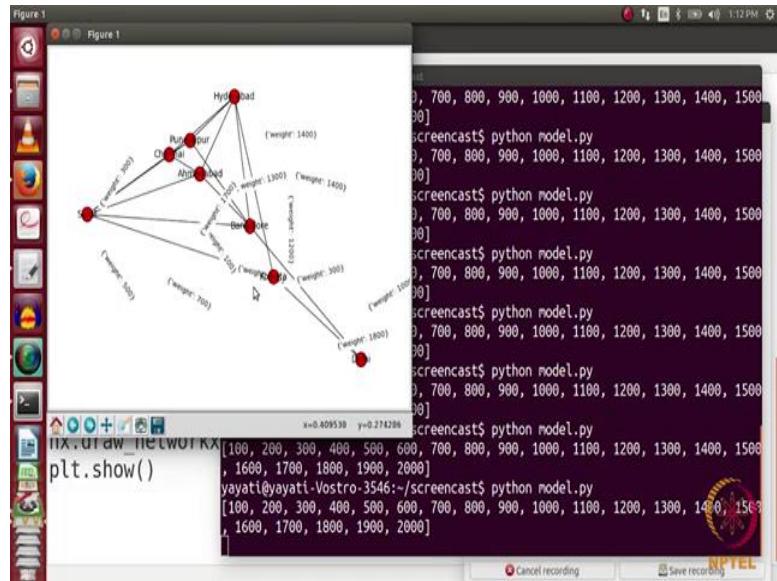
```
value=values
print costs
#We are going to
while(G.number_of_edges() < value):
    c1=random.randint(0,19)
    c2=random.randint(0,19)
    if c1!=c2:
        G.add_edge(c1,c2,weight=costs[c1][c2])
pos=nx.circular_layout(G)
nx.draw(G)
nx.draw_networkx_labels(G)
plt.show()
```

And again we run python model.py.

(Refer Slide Time: 15:37)



(Refer Slide Time: 15:51)

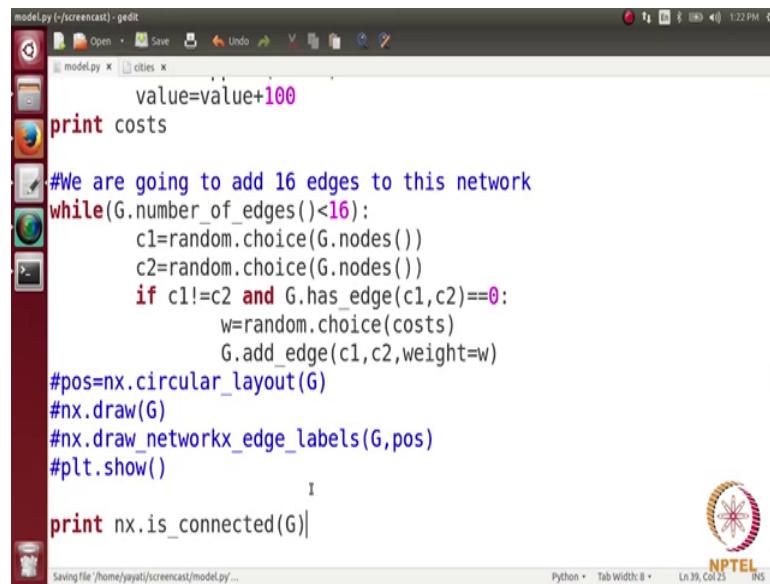


So, you see here, it has to be tried some number of times and when you draw it in a circular layout it turns out to be a pretty nice configuration here no its not very clear, but yeah still you can see the edge weights here. So, now, we have a network which has some 8 nodes and 16 edges and we have visualize this network as well we have drawn these edges randomly. So, we have drawn 16 edges randomly between random pair of nodes.

So, we have now created the network with the cities and the roads what we are next interested in his looking at the paths between these cities and the cost of travelling across that paths as well. So, first let us look at the path between these cities we want to look at the path length between every 2 possible pair of cities and by path length what I mean here is now the some of the weights of the path just to see whether I can reach from city a to city b or not. So, first of all let us that whether even this graph is connected or not because of the graph is connected it means that I can reach from every city to every other city and so on.

So, let us look at whether this graph is connected or not.

(Refer Slide Time: 17:14)



```
model.py (~/screencast) - gedit
value=value+100
print costs

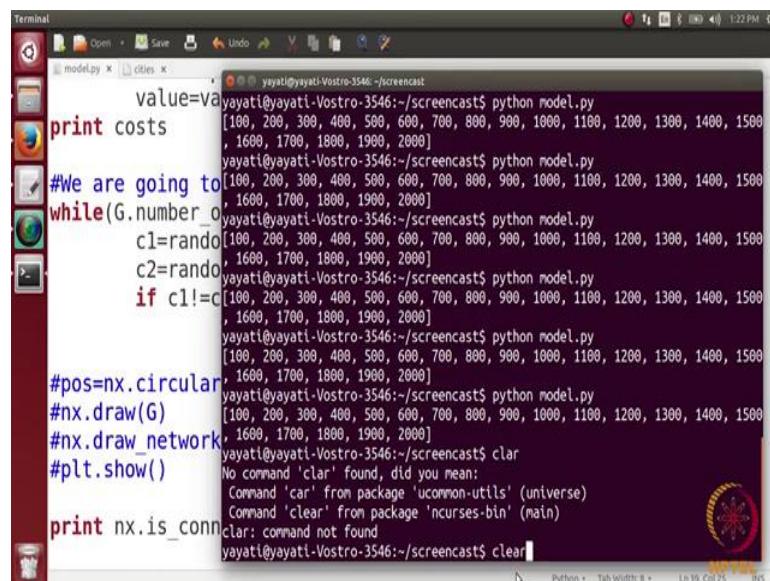
#We are going to add 16 edges to this network
while(G.number_of_edges()<16):
    c1=random.choice(G.nodes())
    c2=random.choice(G.nodes())
    if c1!=c2 and G.has_edge(c1,c2)==0:
        w=random.choice(costs)
        G.add_edge(c1,c2,weight=w)
#pos=nx.circular_layout(G)
#nx.draw(G)
#nx.draw_networkx_edge_labels(G,pos)
#plt.show()

print nx.is_connected(G)

Saving file '/home/yayati/screencast/model.py...' Python Tab Width: 8 Ln 39, Col 25 IIS
```

So, we will just print, so we have a command `nx.is_connected G` which tells us whether this graph is connected or not.

(Refer Slide Time: 17:38)



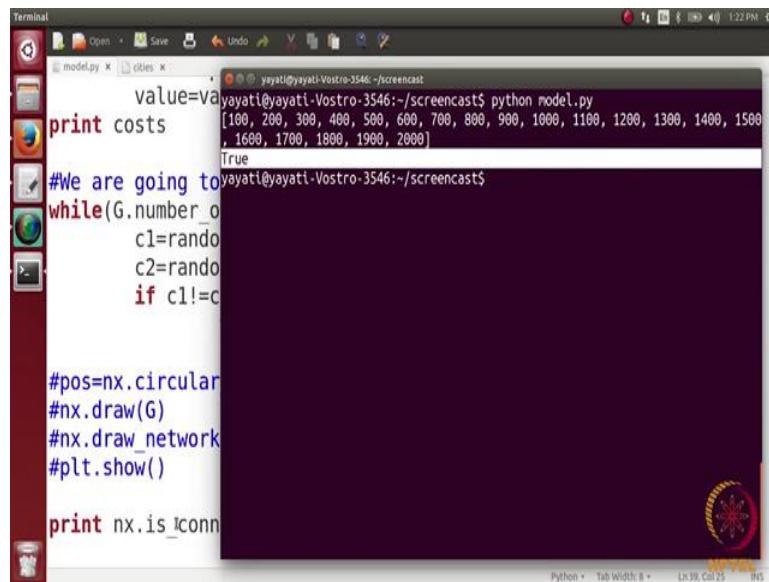
```
Terminal
model.py (~/screencast)
value=value+100
print costs
#We are going to add 16 edges to this network
while(G.number_of_edges()<16):
    c1=random.choice(G.nodes())
    c2=random.choice(G.nodes())
    if c1!=c2 and G.has_edge(c1,c2)==0:
        w=random.choice(costs)
        G.add_edge(c1,c2,weight=w)
#pos=nx.circular_layout(G)
#nx.draw(G)
#nx.draw_networkx_edge_labels(G,pos)
#plt.show()

print nx.is_connected(G)

yayati@yayati-Vostro-3546:~/screencast$ python model.py
[100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500
, 1600, 1700, 1800, 1900, 2000]
yayati@yayati-Vostro-3546:~/screencast$ python model.py
[100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500
, 1600, 1700, 1800, 1900, 2000]
yayati@yayati-Vostro-3546:~/screencast$ python model.py
[100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500
, 1600, 1700, 1800, 1900, 2000]
yayati@yayati-Vostro-3546:~/screencast$ python model.py
[100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500
, 1600, 1700, 1800, 1900, 2000]
yayati@yayati-Vostro-3546:~/screencast$ python model.py
[100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500
, 1600, 1700, 1800, 1900, 2000]
yayati@yayati-Vostro-3546:~/screencast$ python model.py
[100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500
, 1600, 1700, 1800, 1900, 2000]
yayati@yayati-Vostro-3546:~/screencast$ python model.py
[100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500
, 1600, 1700, 1800, 1900, 2000]
yayati@yayati-Vostro-3546:~/screencast$ clear
No command 'clar' found, did you mean:
  Command 'car' from package 'ucommon-utils' (universe)
  Command 'clear' from package 'ncurses-bin' (main)
clar: command not found
yayati@yayati-Vostro-3546:~/screencast$ clear
```

So, let us see, just clear it.

(Refer Slide Time: 17:40)



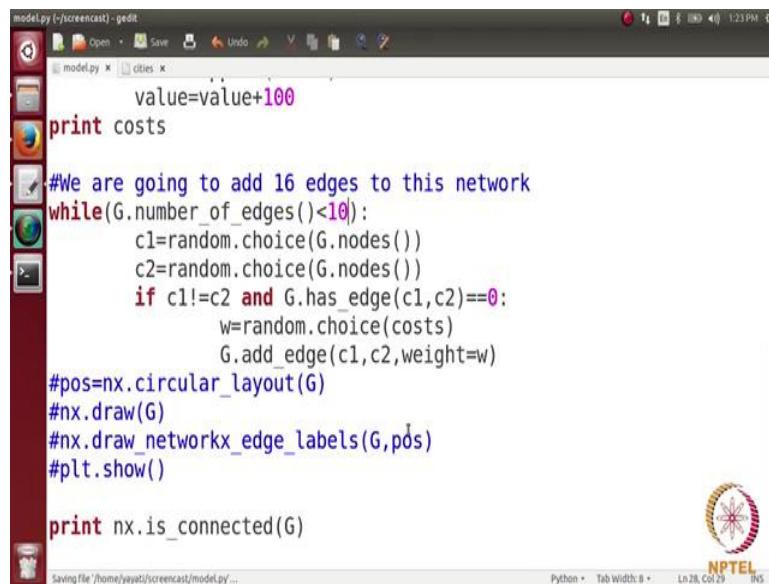
```
Terminal
modelpy x cities x yayati@yayati-Vostro-3546:~/screencast$ python model.py
value=value+100
print costs
[100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500
, 1600, 1700, 1800, 1900, 2000]
True
#We are going to add edges to this network
while(G.number_of_edges()<10):
    c1=random.choice(G.nodes())
    c2=random.choice(G.nodes())
    if c1!=c2 and G.has_edge(c1,c2)==0:
        w=random.choice(costs)
        G.add_edge(c1,c2,weight=w)
#pos=nx.circular_layout(G)
#nx.draw(G)
#nx.draw_networkx_edge_labels(G,pos)
#plt.show()

print nx.is_connected(G)

Python Tab Width: 8 Ln 39, Col 25 IHS
```

See here at the graph is connected it means that there is path between every 2 pair of nodes. So, this, so, what we want is we want to delete some edges let us say. So, that we do not have some paths between some pair of cities.

(Refer Slide Time: 18:07)



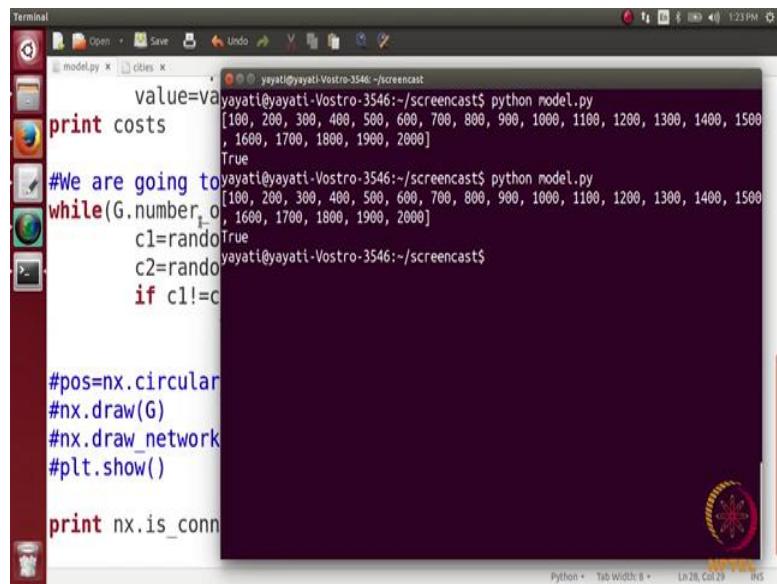
```
modelpy (~screencast) - gedit
modelpy x cities x yayati@yayati-Vostro-3546:~/screencast$ python model.py
value=value+100
print costs
#We are going to add 16 edges to this network
while(G.number_of_edges()<10):
    c1=random.choice(G.nodes())
    c2=random.choice(G.nodes())
    if c1!=c2 and G.has_edge(c1,c2)==0:
        w=random.choice(costs)
        G.add_edge(c1,c2,weight=w)
#pos=nx.circular_layout(G)
#nx.draw(G)
#nx.draw_networkx_edge_labels(G,pos)
#plt.show()

print nx.is_connected(G)

Saving file /home/yayati/screencast/model.py...
Python Tab Width: 8 Ln 28, Col 29 IHS
```

So, what we can do for that is if we can do for doing that is let us reduce the number of edges here from 16 to let say that we will just have 10 edges in this network and let see what happens in that case.

(Refer Slide Time: 18:20)



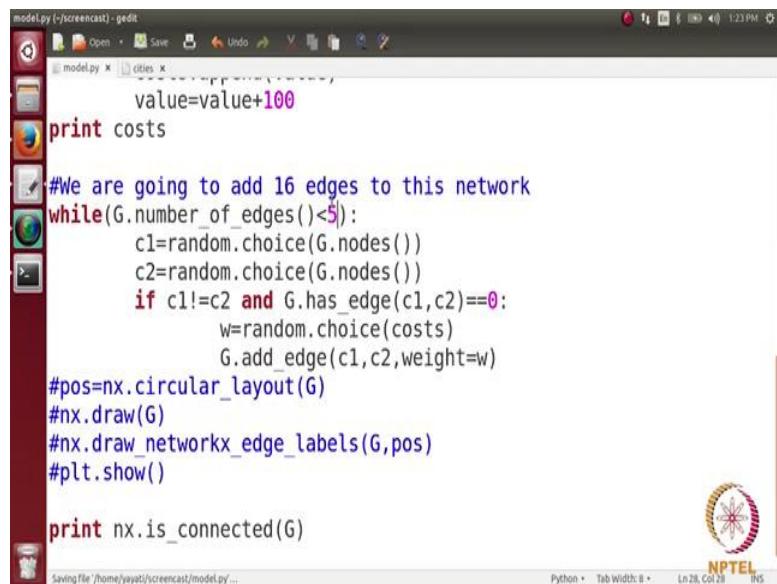
```
Terminal
modelpy x cities x yayati@yayati-Vostro-3546:~/screencast$ python model.py
value=value+100
print costs
#We are going to add 16 edges to this network
while(G.number_of_edges()<5):
    c1=random.choice(G.nodes())
    c2=random.choice(G.nodes())
    if c1!=c2 and G.has_edge(c1,c2)==0:
        w=random.choice(costs)
        G.add_edge(c1,c2,weight=w)
#pos=nx.circular_layout(G)
#nx.draw(G)
#nx.draw_networkx_edge_labels(G,pos)
#plt.show()

print nx.is_connected(G)

Python Tab Width: 8 Ln 28, Col 28 Th5
```

So, we have done edges again the graph is connected. So, it is actually interesting you see it is a network on 8 nodes and we have put just 10 edges even then this turning out be connected.

(Refer Slide Time: 18:34)



```
modelpy (-/screencast) - gedit
modelpy x cities x yayati@yayati-Vostro-3546:~/screencast$ python model.py
value=value+100
print costs
#We are going to add 16 edges to this network
while(G.number_of_edges()<5):
    c1=random.choice(G.nodes())
    c2=random.choice(G.nodes())
    if c1!=c2 and G.has_edge(c1,c2)==0:
        w=random.choice(costs)
        G.add_edge(c1,c2,weight=w)
#pos=nx.circular_layout(G)
#nx.draw(G)
#nx.draw_networkx_edge_labels(G,pos)
#plt.show()

print nx.is_connected(G)

Saving file /home/yayati/screencast/model.py...
Python Tab Width: 8 Ln 28, Col 28 Th5
```

Let us reduce it to 5, let say and then when we put the number of edges be 5, it turns out that the network is not connected. So, it is false network is not connected.

(Refer Slide Time: 18:41)

```
Terminal yayati@yayati-Vostro-3546:~/screencast$ python model.py
value=yayati@yayati-Vostro-3546:~/screencast$ python model.py
print costs
[100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500
, 1600, 1700, 1800, 1900, 2000]
True
#We are going to
while(G.number_o
    cl=random
    c2=random
    if cl!=c
        False
yayati@yayati-Vostro-3546:~/screencast$ python model.py
[100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500
, 1600, 1700, 1800, 1900, 2000]
yayati@yayati-Vostro-3546:~/screencast$ python model.py
#pos=nx.circular
#nx.draw(G)
#nx.draw_network
#plt.show()

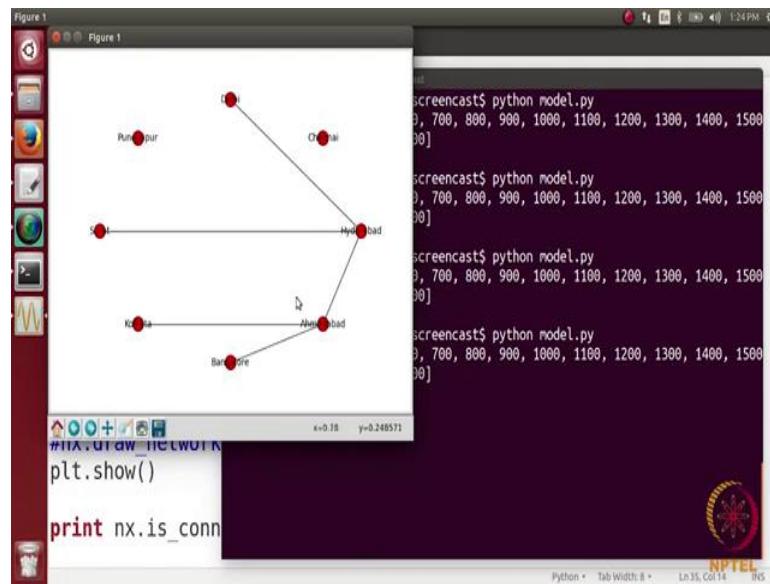
print nx.is_conn
```

So, let us look at this network as well.

(Refer Slide Time: 18:54)

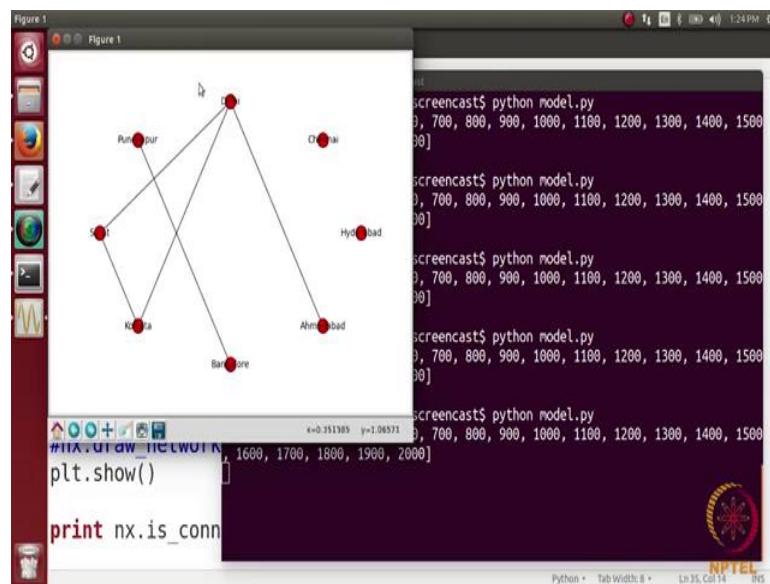
```
Terminal Open Save Undo ⌘ D 12:24 PM  
model.py x cities x yayati@yayati-Vostro-3546:~/screencast$ python model.py  
value=yayati@yayati-Vostro-3546:~/screencast$ python model.py  
print costs  
[100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500  
, 1600, 1700, 1800, 1900, 2000]  
True  
#We are going to  
while(G.number_o  
    c1=random  
    c2=random  
    if c1!=c  
        False  
yayati@yayati-Vostro-3546:~/screencast$ python model.py  
[100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500  
, 1600, 1700, 1800, 1900, 2000]  
pos=nx.circular  
nx.draw(G,pos)  
#nx.draw_network  
plt.show()  
  
print nx.is_conn  
Saving file /home/yayati/screencast/model.py ...  
Python • Tab Width: 8 • Ln 35, Col 14 IN5
```

(Refer Slide Time: 18:58)



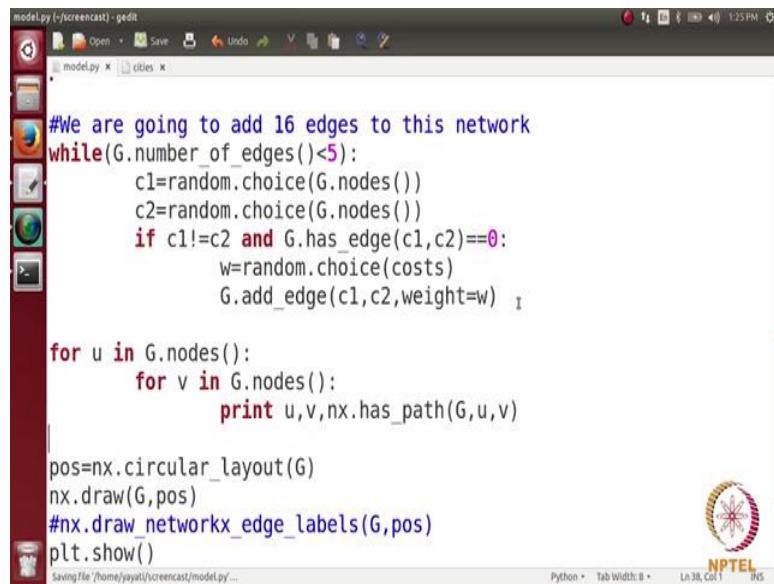
Now, you see the network here right. So, this there is this one city which is not these are which are not connected at all to any of these cities.

(Refer Slide Time: 19:06)



So, let us take one more example. So, it can be like this and so whom, so what is now we want to see assume that we want to see path between which cities path exists and between which cities path does not exist.

(Refer Slide Time: 19:23)



```
#We are going to add 16 edges to this network
while(G.number_of_edges()<5):
    c1=random.choice(G.nodes())
    c2=random.choice(G.nodes())
    if c1!=c2 and G.has_edge(c1,c2)==0:
        w=random.choice(costs)
        G.add_edge(c1,c2,weight=w)

for u in G.nodes():
    for v in G.nodes():
        print u,v,nx.has_path(G,u,v)

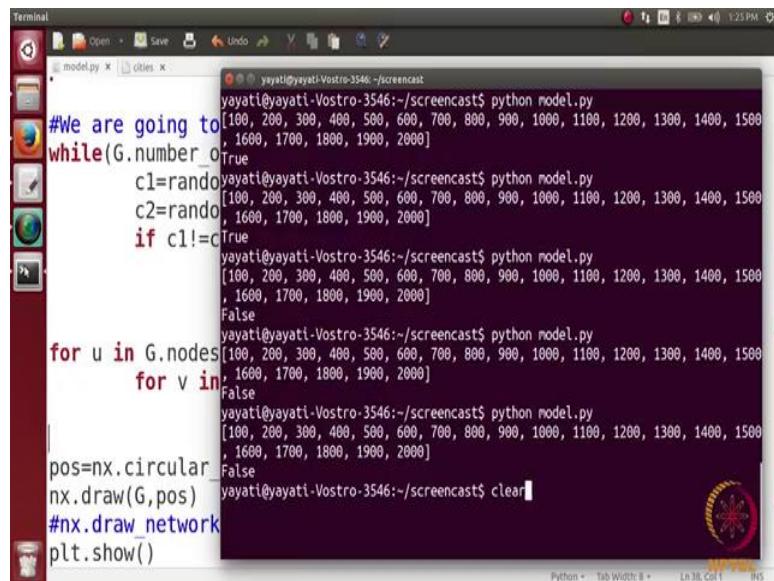
pos=nx.circular_layout(G)
nx.draw(G,pos)
#nx.draw_networkx_edge_labels(G,pos)
plt.show()
```

Saving file /home/yayati/screencast/model.py... Python Tab Width: 8 Ln 38, Col 1 NPTEL IIT5

So, what we do for that is for each possible pair of cities we see that whether there is a path between them or not. So, we have a loop here for let us say  $u$  in  $G.\text{nodes}$  and then for  $v$  in  $G.\text{nodes}$  what do you do is print  $u$   $v$  comma and then we have a function here  $\text{nx}.\text{has\_underscore\_path}$  which tells us whether there is a path between these cities or not.

So, we pass here it  $G$   $u$  comma  $v$  right. So, what I will do is I will put this code before drawing the networks so that we can see both the things simultaneously and let see now just in.

(Refer Slide Time: 20:13)



```
#We are going to add 16 edges to this network
while(G.number_of_edges()<5):
    c1=random.choice(G.nodes())
    c2=random.choice(G.nodes())
    if c1!=c2 and G.has_edge(c1,c2)==0:
        w=random.choice(costs)
        G.add_edge(c1,c2,weight=w)

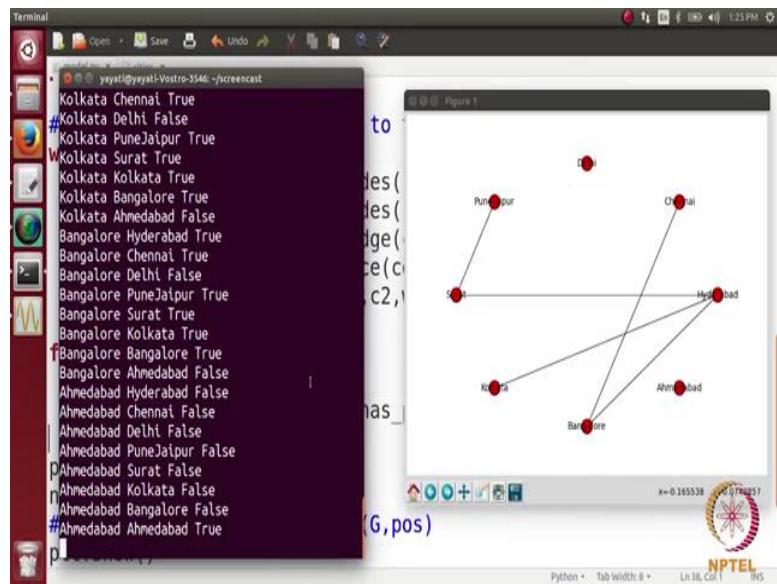
for u in G.nodes():
    for v in G.nodes():
        print u,v,nx.has_path(G,u,v)

pos=nx.circular_layout(G)
nx.draw(G,pos)
#nx.draw_networkx_edge_labels(G,pos)
plt.show()
```

```
yayati@yayati-Vostro-3546:~/screencast$ python model.py
[100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500
, 1600, 1700, 1800, 1900, 2000]
yayati@yayati-Vostro-3546:~/screencast$ python model.py
[100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500
, 1600, 1700, 1800, 1900, 2000]
False
yayati@yayati-Vostro-3546:~/screencast$ python model.py
[100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500
, 1600, 1700, 1800, 1900, 2000]
False
yayati@yayati-Vostro-3546:~/screencast$ python model.py
[100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500
, 1600, 1700, 1800, 1900, 2000]
False
yayati@yayati-Vostro-3546:~/screencast$ python model.py
[100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500
, 1600, 1700, 1800, 1900, 2000]
False
yayati@yayati-Vostro-3546:~/screencast$ clear
```

Python Tab Width: 8 Ln 38, Col 1 NPTEL IIT5

(Refer Slide Time: 20:18)



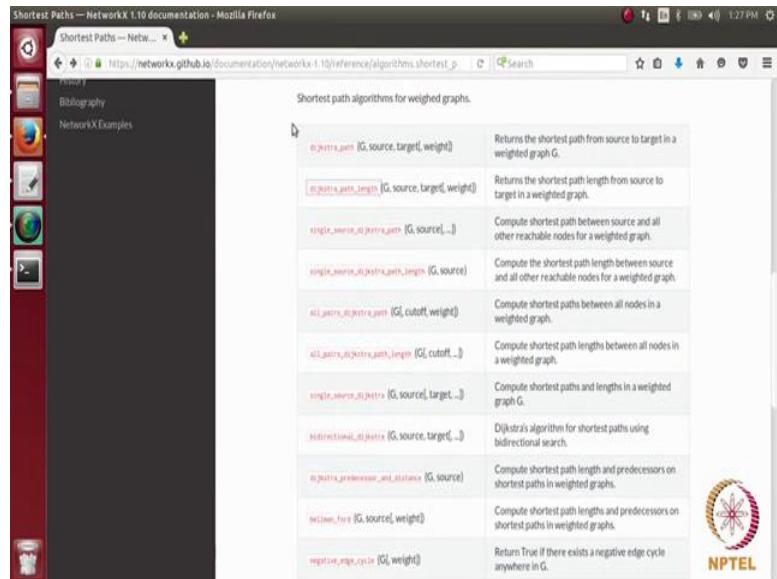
So, you see here. So, this is our network and what is. So, perfect. So, this is our network here and these are the paths. So, it tells you for every possible pair of cities. So, Hyderabad to Hyderabad is; obviously, path is there Hyderabad to Chennai there is a path Hyderabad to Delhi there is no path. So, you see here from let say Hyderabad to Pune and we look from let say Hyderabad to Pune I am very sorry we have mixed within Pune and Jaipur. So, let us say its Pune.

So, from Hyderabad to Pune we looked at there is a path. So, you can go from Hyderabad to Surat and from Surat you can go to Pune. So, this is a fictitious network if the connections might not be really this way it just a fictitious. So, do not look at it in real world then how come Hyderabad is connecting to Surat and then to Pune and so on. So, there is a path from this node Hyderabad to a node here. So, we have listed it for every 2 pair of nodes. So, you can look at; then from Chennai we can go to Hyderabad to let say Bangalore. So, again you see here if you look the connection between let say Chennai and Hyderabad which is somewhere here yeah. So, it turns out to be true and from Delhi you see from Delhi you cannot go anywhere according to these networks. So, every value from Delhi turns out to be false and so on, just close it, stand here.

Now, next what we want to look at is what are the shortest paths in this network. So, we have some really (Refer Time: 22:25) functions in networkx. So, which are the basic

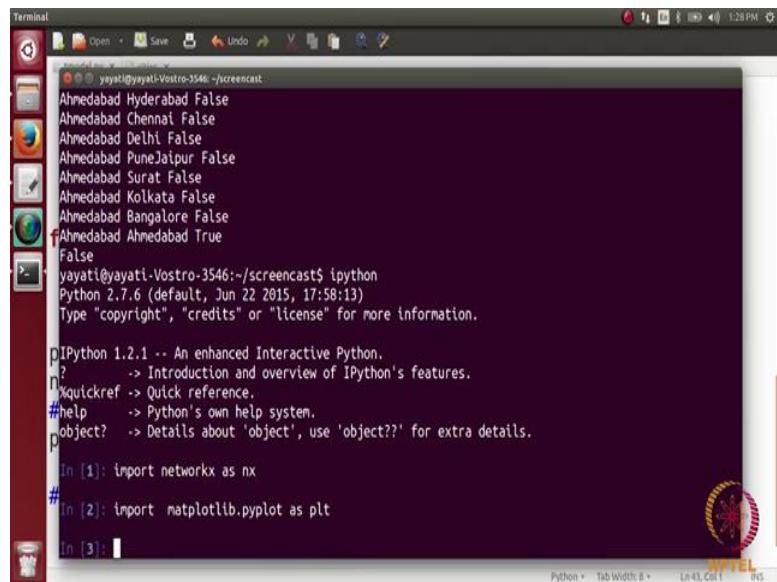
standard algorithms implemented in networkx with the help of which you can see the path length of the travelling cost between 2 cities.

(Refer Slide Time: 22:41)



So, let me quickly show you some of these. So, these are here. So, you can see that these are the functions. So, we have one function discussed standards discussed standards cold path which tells you the shortest paths from source to target and then underscore path length which. So, given 2 nodes it tells you the length of that shortest path then single source discussed a path where you give one source node and it gives you the all the shortest paths and so on. So, we will we are going to use some of these functions and see how do they work in our network? So, for working with this network what will be going to do is let us now switch a little bit to the interactive ipython.

(Refer Slide Time: 23:30)



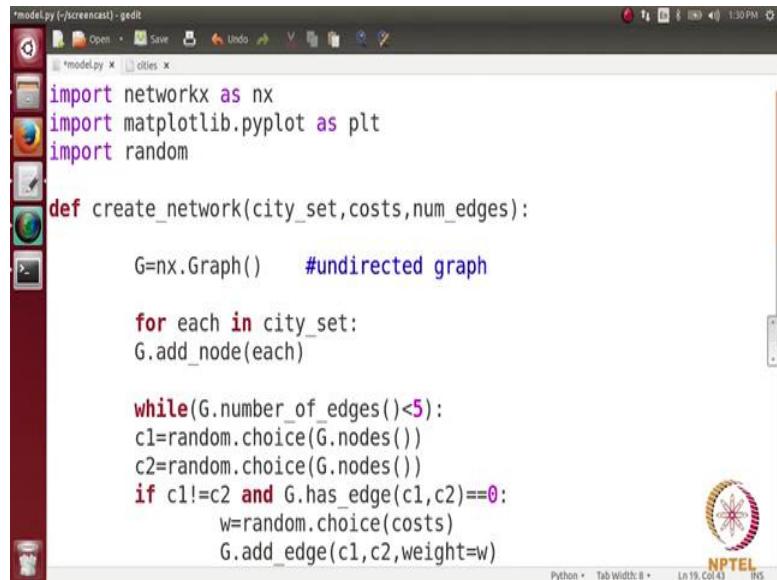
```
yayati@yayati-Vostro-3546:~/screencast
Ahmedabad Hyderabad False
Ahmedabad Chennai False
Ahmedabad Delhi False
Ahmedabad PuneJaipur False
Ahmedabad Surat False
Ahmedabad Kolkata False
Ahmedabad Bangalore False
Ahmedabad Ahmedabad True
False
yayati@yayati-Vostro-3546:~/screencast$ ipython
Python 2.7.6 (default, Jun 22 2015, 17:58:13)
Type "copyright", "credits" or "license" for more information.

IPython 1.2.1 -- An enhanced Interactive Python.
?           --> Introduction and overview of IPython's features.
%quickref --> Quick reference.
#help      --> Python's own help system.
object?    --> Details about 'object', use 'object??' for extra details.

In [1]: import networkx as nx
#
In [2]: import matplotlib.pyplot as plt
In [3]:
```

So, what we are going to do is we are going to turn to ipython and then we are going to import network ipython then we are going to import networkx as nx then import matplotlib.py pyplot as plt perfect. So, we are going to define some functions here. So, let us con let us make this code a little bit of deny stand make some functions from this.

(Refer Slide Time: 23:59)



```
*modelpy (~)/screencast - gedit
*modelpy x cities x
import networkx as nx
import matplotlib.pyplot as plt
import random

def create_network(city_set,costs,num_edges):
    G=nx.Graph()      #undirected graph

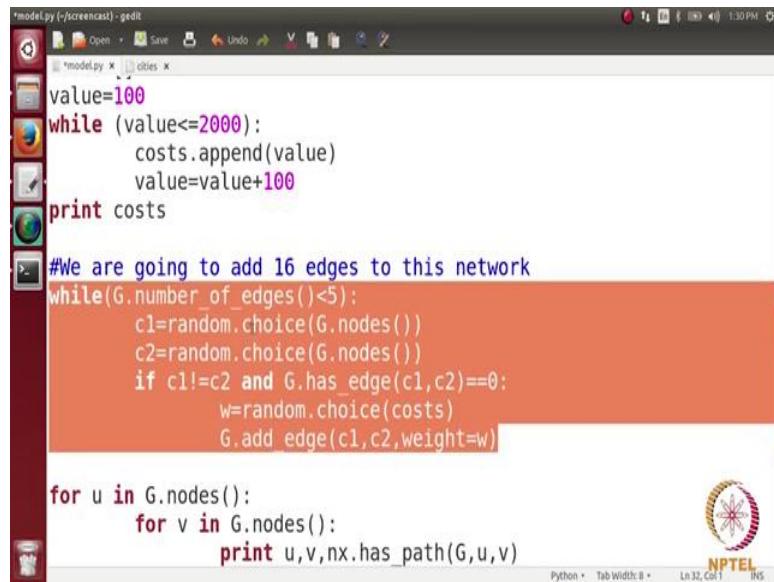
    for each in city_set:
        G.add_node(each)

    while(G.number_of_edges()<5):
        c1=random.choice(G.nodes())
        c2=random.choice(G.nodes())
        if c1!=c2 and G.has_edge(c1,c2)==0:
            w=random.choice(costs)
            G.add_edge(c1,c2,weight=w)
```

So, let me define a function fine create network and let say this function takes us input the set of cities the array costs and let say number of edges which you want to create.

So, what all we will have this inside this function is this term will come here. So, G equals to nx.graph and then for each in city set we are going to add an edge. So, this particular code comes here for each in city set we are going to add a node and then what else.

(Refer Slide Time: 24:54)



The screenshot shows a terminal window titled "model.py (~/screencast) - gedit". The code is written in Python and generates a network graph. It starts by defining a variable "value=100" and then enters a while loop where it adds edges to a graph "G". The loop continues until "G.number\_of\_edges()<5". Inside the loop, two random nodes "c1" and "c2" are chosen from "G.nodes()". If "c1" and "c2" are not the same and there is no edge between them, a random weight "w" is chosen from the "costs" list and an edge is added with weight "w". After the loop, the code prints all edges and paths between nodes using nested loops over "G.nodes()".

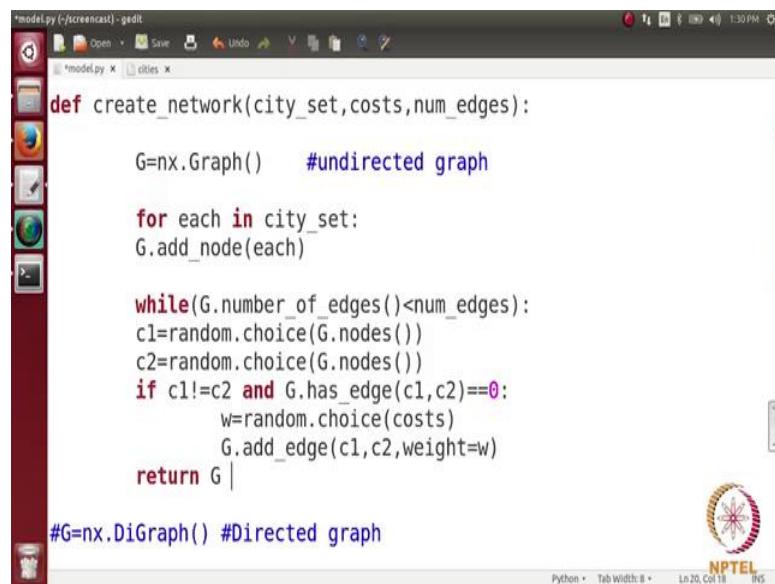
```
value=100
while (value<=2000):
    costs.append(value)
    value=value+100
print costs

#We are going to add 16 edges to this network
while(G.number_of_edges()<5):
    c1=random.choice(G.nodes())
    c2=random.choice(G.nodes())
    if c1!=c2 and G.has_edge(c1,c2)==0:
        w=random.choice(costs)
        G.add_edge(c1,c2,weight=w)

for u in G.nodes():
    for v in G.nodes():
        print u,v,nx.has_path(G,u,v)
```

So, this entire code will be shifted there. So, this is when we are we were adding the, we were adding 5 nodes in this networks. So, will basically change this code first of all let us paste it here. So, why 0 number of edges is less than; so we are adding num edges number of edges. So, let me make it num edges here and then of right and this will return you this graph G which you can then visualize.

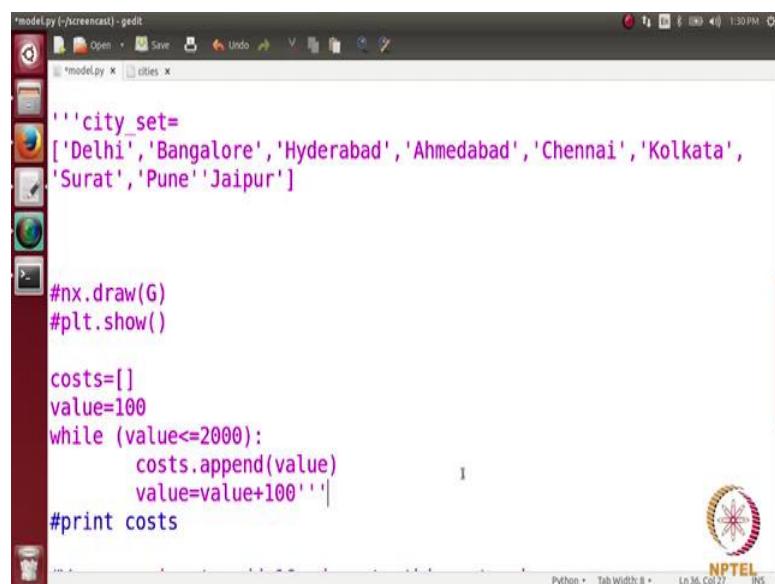
(Refer Slide Time: 25:31)



```
*model.py (~/screencast) - gedit
  Open Save Undo
  model.py cities
def create_network(city_set,costs,num_edges):
    G=nx.Graph()      #undirected graph
    for each in city_set:
        G.add_node(each)
    while(G.number_of_edges()<num_edges):
        c1=random.choice(G.nodes())
        c2=random.choice(G.nodes())
        if c1!=c2 and G.has_edge(c1,c2)==0:
            w=random.choice(costs)
            G.add_edge(c1,c2,weight=w)
    return G
#G=nx.DiGraph() #Directed graph
```

The screenshot shows a terminal window titled "model.py (~/screencast) - gedit". It contains Python code for generating a network graph. The code defines a function "create\_network" that takes three parameters: "city\_set", "costs", and "num\_edges". It creates an undirected graph "G" using nx.Graph(). It then enters a loop where it randomly chooses two nodes from the city set. If the nodes have not yet been connected by an edge and the edge does not already exist, it adds the edge with a random weight from the "costs" list. The loop continues until the number of edges in the graph reaches the specified "num\_edges". Finally, it returns the graph "G". A comment "#G=nx.DiGraph() #Directed graph" is present at the bottom of the function definition.

(Refer Slide Time: 25:48)



```
'''city_set=
['Delhi','Bangalore','Hyderabad','Ahmedabad','Chennai','Kolkata',
'Surat','Pune','Jaipur']

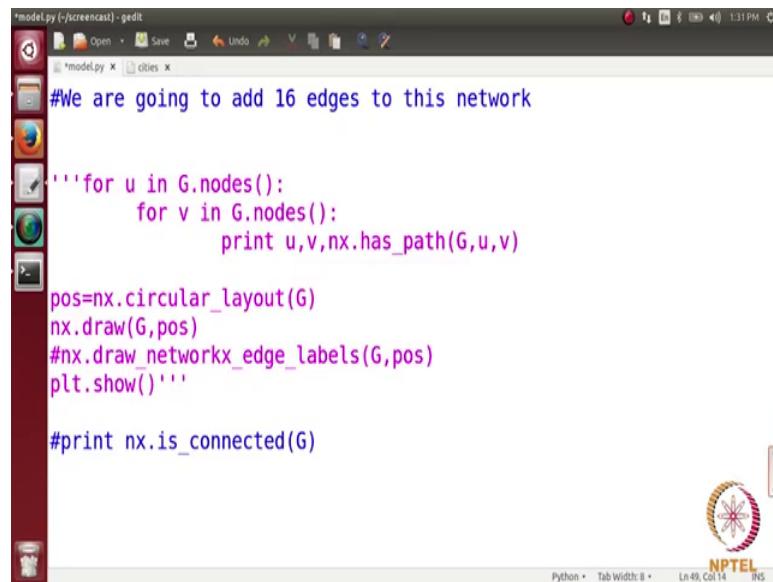
#nx.draw(G)
#plt.show()

costs=[]
value=100
while (value<=2000):
    costs.append(value)
    value=value+100'''
```

The screenshot shows a terminal window titled "model.py (~/screencast) - gedit". It contains Python code for generating a network graph. The code starts by defining a list of cities: "Delhi", "Bangalore", "Hyderabad", "Ahmedabad", "Chennai", "Kolkata", "Surat", "Pune", and "Jaipur". It then includes a commented-out section that would draw the graph and show it using nx.draw and plt.show(). Below that, it initializes an empty list "costs" and sets "value" to 100. It enters a while loop that continues until "value" reaches or exceeds 2000. In each iteration of the loop, it appends the current value to the "costs" list and then increments "value" by 100. A comment "#print costs" is present at the end of the loop.

Next comment rest of are go further mean while write commented rest of the codes. So, everything is most and mostly commented here.

(Refer Slide Time: 25:57)



```
#We are going to add 16 edges to this network

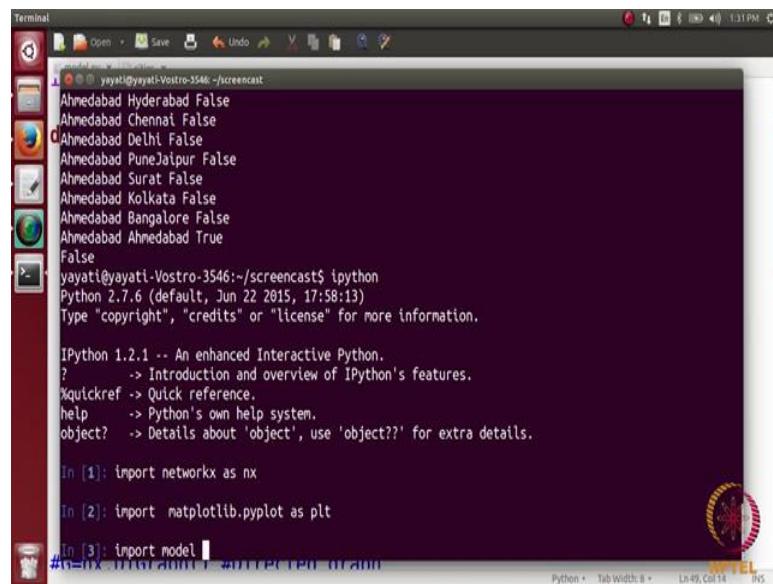
'''for u in G.nodes():
    for v in G.nodes():
        print u,v,nx.has_path(G,u,v)

pos=nx.circular_layout(G)
nx.draw(G,pos)
#nx.draw_networkx_edge_labels(G,pos)
plt.show()'''

#print nx.is_connected(G)
```

Everything is commented except for this function here. So, this function is simply what it is doing taking a city set taking array costs taking the number of edges and creating the graph.

(Refer Slide Time: 26:18)



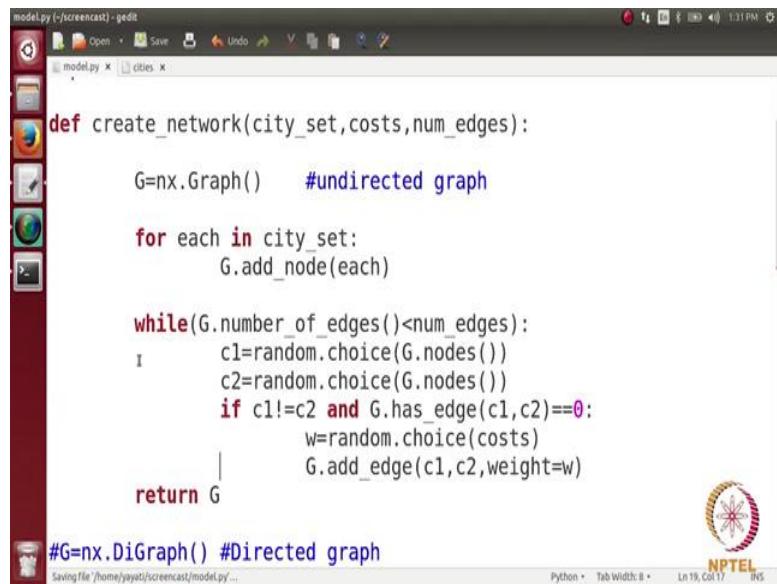
```
Ahmedabad Hyderabad False
Ahmedabad Chennai False
Ahmedabad Delhi False
Ahmedabad Pune Jaipur False
Ahmedabad Surat False
Ahmedabad Kolkata False
Ahmedabad Bangalore False
Ahmedabad Ahmedabad True
False
yayati@yayati-Vostro-3546:~/screencast$ ipython
Python 2.7.6 (default, Jun 22 2015, 17:58:13)
Type "copyright", "credits" or "license" for more information.

IPython 1.2.1 -- An enhanced Interactive Python.
?           -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help       -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: import networkx as nx
In [2]: import matplotlib.pyplot as plt
In [3]: import model
```

So, how do we do it here? So, we are going to import this model here. So, when added it line 12. So, this is the 4 loop.

(Refer Slide Time: 26:30)

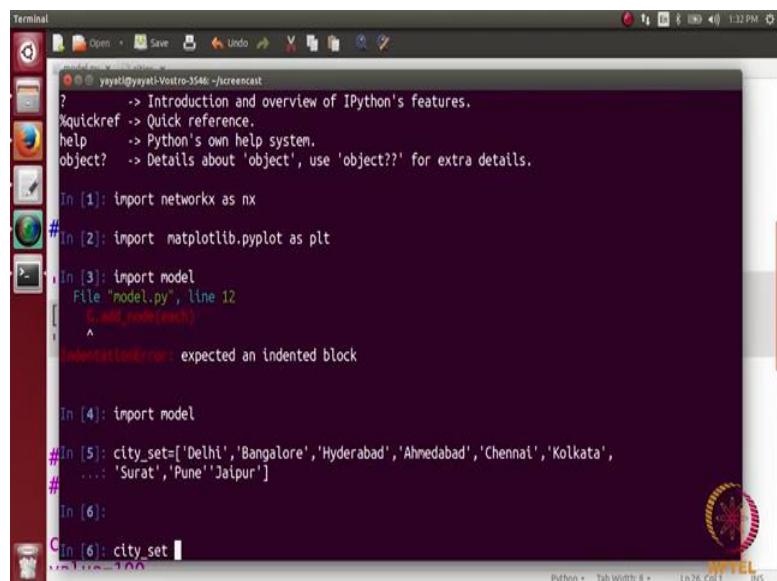


```
model.py (~/screen cast) - gedit
Saving file /home/yayati/screen cast/model.py...
def create_network(city_set,costs,num_edges):
    G=nx.Graph()      #undirected graph
    for each in city_set:
        G.add_node(each)
    while(G.number_of_edges()<num_edges):
        c1=random.choice(G.nodes())
        c2=random.choice(G.nodes())
        if c1!=c2 and G.has_edge(c1,c2)==0:
            w=random.choice(costs)
            G.add_edge(c1,c2,weight=w)
    return G
#G=nx.DiGraph() #Directed graph
```

The screenshot shows a Gedit text editor window with Python code. The code defines a function `create_network` that creates an undirected graph from a set of cities. It uses NetworkX's `nx.Graph()` class. The function iterates until the graph has the specified number of edges, adding edges between random cities with random weights from a given list of costs. A comment at the bottom indicates an alternative for a directed graph.

So, here you have to always take care of the indentation and let us import model and we have imported a.

(Refer Slide Time: 26:40)



```
Terminal
yayati@yayati-Vostro-3546:~/screen cast
?          -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help       -> Python's own help system.
object?    -> Details about 'object', use 'object??' for extra details.

In [1]: import networkx as nx
In [2]: import matplotlib.pyplot as plt
In [3]: import model
File "model.py", line 12
    G.add_node(each)
    ^
IndentationError: expected an indented block

In [4]: import model
In [5]: city_set=['Delhi','Bangalore','Hyderabad','Ahmedabad','Chennai','Kolkata',
...: 'Surat','Pune','Jaipur']
#
In [6]:
C In [6]: city_set
```

The screenshot shows a Jupyter Notebook terminal window. It displays the Python code from the previous slide. In cell [3], there is an indentation error because the line `G.add_node(each)` is not indented under the loop. The error message `IndentationError: expected an indented block` is shown. The code continues with imports for `networkx`, `matplotlib.pyplot`, and `model`, and defines a variable `city_set` containing a list of city names.

So, to call this function we should have some parameters which are the city set the costs and the number of edges. So, let us define these parameters here. So, I will just copy paste the set of cities here. I have a array here for the city set we can also see this array here.

(Refer Slide Time: 27:03)

```
In [4]: import model
# In [5]: city_set=['Delhi', 'Bangalore', 'Hyderabad', 'Ahmedabad', 'Chennai', 'Kolkata',
... 'Surat', 'Pune', 'Jaipur']
# In [6]: costs=[]
# In [7]: costs.append(city_set)
In [8]: value=100
```

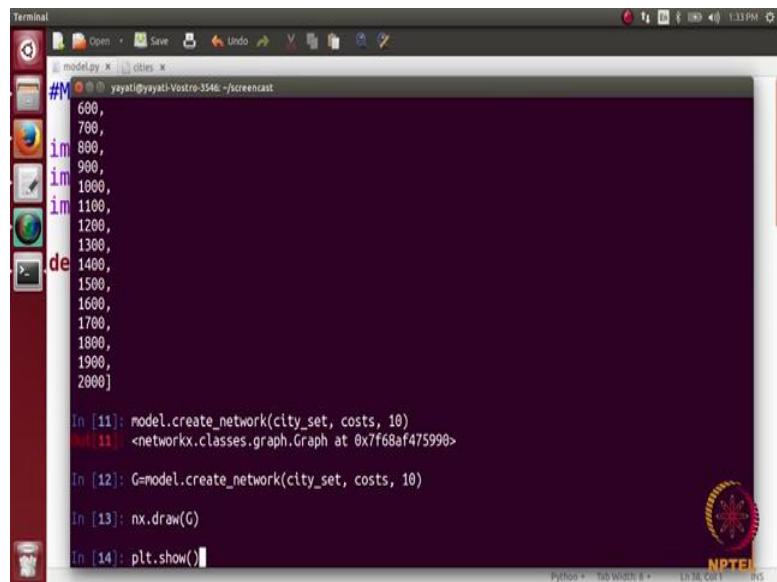
So, we have this city set and then we want these costs to be here. So, in will just take this code yes paste it here.

(Refer Slide Time: 27:16)

```
In [5]: city_set=['Delhi', 'Bangalore', 'Hyderabad', 'Ahmedabad', 'Chennai', 'Kolkata',
... 'Surat', 'Pune', 'Jaipur']
# In [6]: costs=[]
# In [7]: costs.append(city_set)
In [8]: value=100
# In [9]: while (value<=2000):
...     costs.append(value)
...     value=value+100
```

Let us look at the array cost.

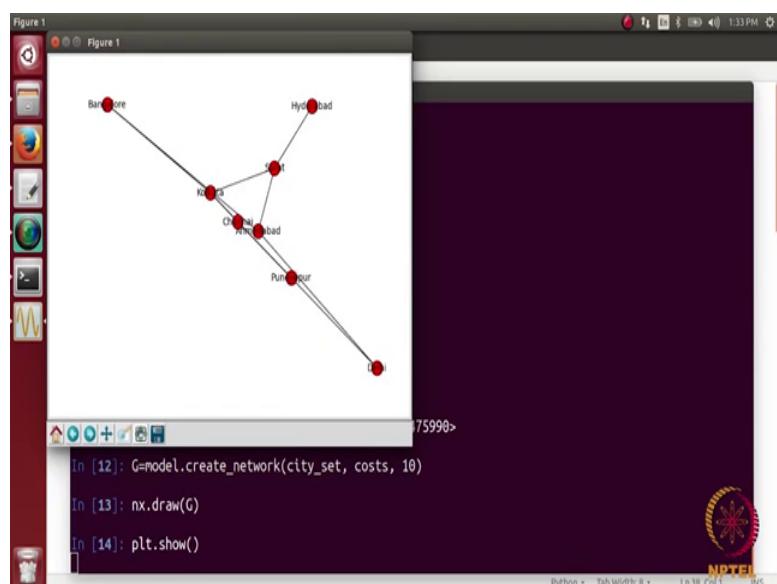
(Refer Slide Time: 27:20)



```
#M 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500, 1600, 1700, 1800, 1900, 2000]
In [11]: model.create_network(city_set, costs, 10)
Out[11]: <networkx.classes.graph.Graph at 0x7f68af475990>
In [12]: G=model.create_network(city_set, costs, 10)
In [13]: nx.draw(G)
In [14]: plt.show()
```

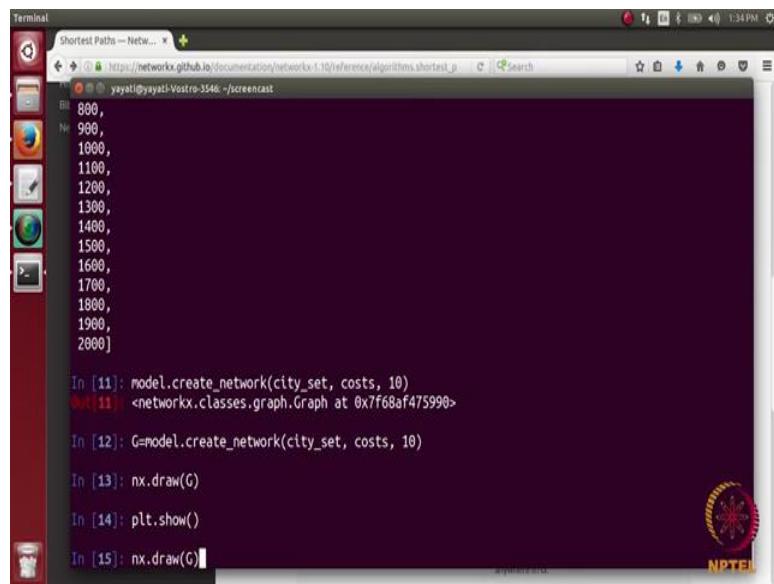
So, we have this array costs here and let say I am going to add some 10 edges. So, let us add 10 edges. So, we are going to call this function and the function name is create underscore network. So, what we are going to do is model.create underscore network and here we are going to pass the 3 parameters. So, what are a parameters one is the cost city set young. So, we are going to pass here city underscore set and then say costs and then we are going to at 10 edges in this network and. So, we have took collect this network in a variable let say this variable is g. So, G equals to this. So, let us try looking at this graph, so let say nx.draw G and then plt.show.

(Refer Slide Time: 28:24)



So, we can see this network here and it is connected. So, we have a graph here and every edge has a weight associated with it here. So, now, what we want to see is we want to see the implementation of the shortest path functions. So, let us one by one take the shortest path functions and see how do they work? So, let us look at the first one here. So, if you give it the graph and the source and the target it will return you the shortest path let us try.

(Refer Slide Time: 29:00)



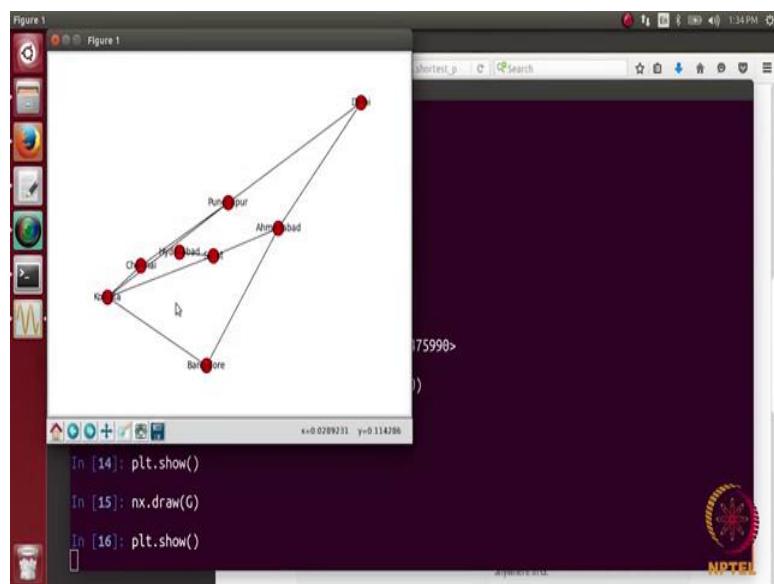
```
Terminal
Shortest Paths — NetworkX 1.10 documentation
yayati@yayati-Vostro-3546:~/screencast
```

```
In [10]: city_set = set([("Kolkata", 0), ("Mumbai", 1), ("Delhi", 2), ("Chennai", 3), ("Bengaluru", 4), ("Hyderabad", 5), ("Pune", 6), ("Jaipur", 7), ("Kochi", 8), ("Bhopal", 9)])
costs = [
    (0, 1, 1000),
    (0, 2, 1200),
    (0, 3, 1500),
    (0, 4, 1800),
    (0, 5, 2000),
    (1, 2, 1100),
    (1, 3, 1400),
    (1, 4, 1700),
    (1, 5, 1900),
    (2, 3, 1300),
    (2, 4, 1600),
    (2, 5, 1800),
    (3, 4, 1200),
    (3, 5, 1500),
    (4, 5, 1100),
    (4, 6, 1300),
    (5, 6, 1200),
    (5, 7, 1400),
    (6, 7, 1100),
    (6, 8, 1300),
    (7, 8, 1200),
    (7, 9, 1500),
    (8, 9, 1400)
]
In [11]: model.create_network(city_set, costs, 10)
Out[11]: <networkx.classes.graph.Graph at 0x7f68af475990>
```

```
In [12]: G=model.create_network(city_set, costs, 10)
In [13]: nx.draw(G)
In [14]: plt.show()
In [15]: nx.draw(G)
```

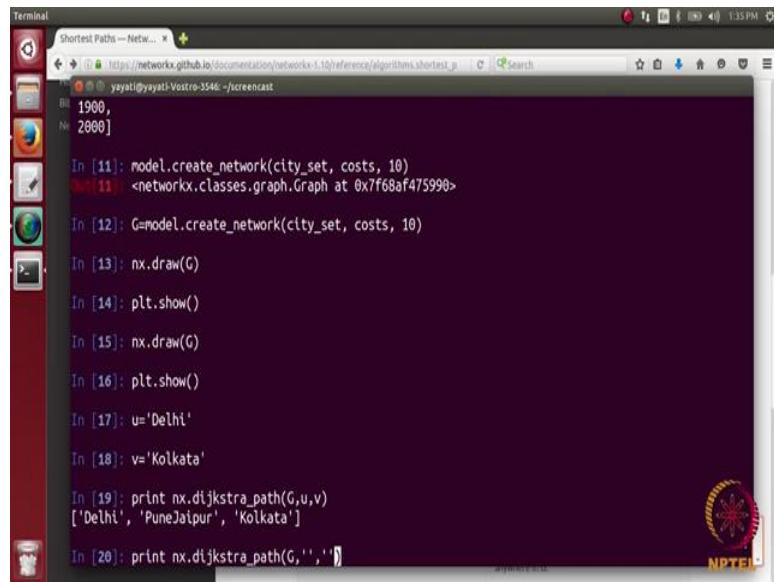
So, let us look a graph once more.

(Refer Slide Time: 29:02)



So, let us choose 2 nodes here let say we choose the aliant Kolkata.

(Refer Slide Time: 29:10)

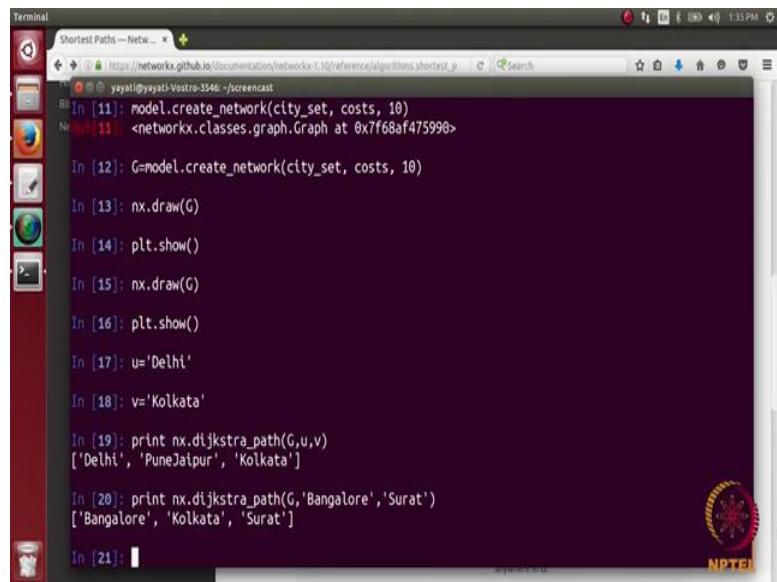


The screenshot shows a terminal window titled "Shortest Path -- Networkx". The window displays a Python script for creating a network and finding shortest paths. The code includes imports for networkx and matplotlib, creation of a network from a city set and costs, and drawing the network. It then defines two nodes, u ('Delhi') and v ('Kolkata'), and prints the shortest path between them using nx.dijkstra\_path(). The output shows the path: ['Delhi', 'PuneJaipur', 'Kolkata']. The terminal is running on a Linux system with a dark theme, and the background shows a blurred view of a presentation slide with the NPTEL logo.

```
In [11]: model.create_network(city_set, costs, 10)
Out[11]: <networkx.classes.graph.Graph at 0x7f68af475990>
In [12]: G=model.create_network(city_set, costs, 10)
In [13]: nx.draw(G)
In [14]: plt.show()
In [15]: nx.draw(G)
In [16]: plt.show()
In [17]: u='Delhi'
In [18]: v='Kolkata'
In [19]: print nx.dijkstra_path(G,u,v)
['Delhi', 'PuneJaipur', 'Kolkata']
In [20]: print nx.dijkstra_path(G,'','')
```

So, let say node number one u is Delhi and second node is now what we are going to do is we are going to look at the shortest path between these 2 nodes. So, what we are going to print is nx.stra and then goes there underscore path and then you paths here network here and the first node and the second node yeah. So, you get here shortest path here. So, from Delhi you go to Pune and then from Pune you can go to Kolkata, similarly we can actually see between every let us choose another some 2 different cities let say let us choose Bangalore and Surat. So, let us write it down Bangalore and let say yeah. So, you see here that Bangalore and Surat are connected with Kolkata and this is the shortest path of it.

(Refer Slide Time: 30:27)



```
Terminal
Shortest Paths -- NetworkX 1.10
yayati@yayati-Vostro-3546:/screencast
In [11]: model.create_network(city_set, costs, 10)
Out[11]: <networkx.classes.graph.Graph at 0x7f68af475990>

In [12]: G=model.create_network(city_set, costs, 10)

In [13]: nx.draw(G)

In [14]: plt.show()

In [15]: nx.draw(G)

In [16]: plt.show()

In [17]: u='Delhi'

In [18]: v='Kolkata'

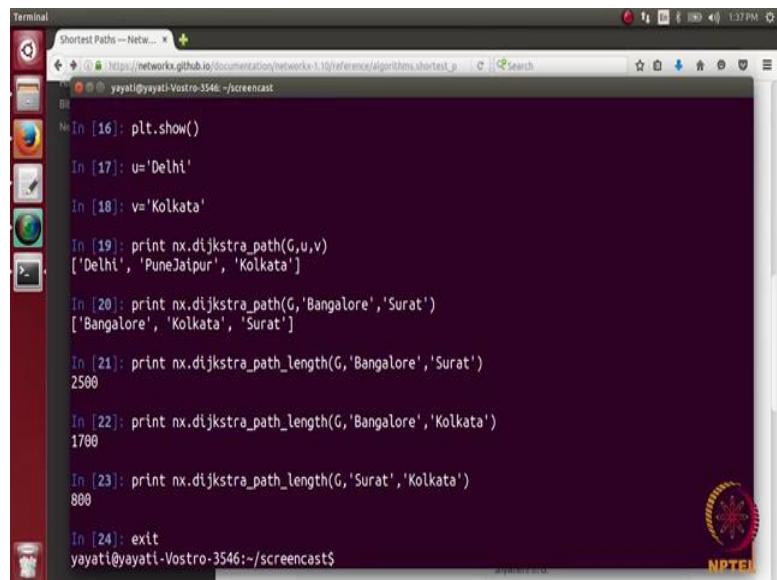
In [19]: print nx.dijkstra_path(G,u,v)
['Delhi', 'PuneJaipur', 'Kolkata']

In [20]: print nx.dijkstra_path(G,'Bangalore','Surat')
['Bangalore', 'Kolkata', 'Surat']

In [21]:
```

Next function is for path length where you do not want to see the path, but you want to see the path length. So, we know here. So, we do a path length here and we. So, this path length is returning here 2500. So, the costs associated with going from Bangalore to Surat is 2500.

(Refer Slide Time: 30:59)



```
Terminal
Shortest Paths -- NetworkX 1.10
yayati@yayati-Vostro-3546:/screencast
In [16]: plt.show()

In [17]: u='Delhi'

In [18]: v='Kolkata'

In [19]: print nx.dijkstra_path(G,u,v)
['Delhi', 'PuneJaipur', 'Kolkata']

In [20]: print nx.dijkstra_path(G,'Bangalore','Surat')
['Bangalore', 'Kolkata', 'Surat']

In [21]: print nx.dijkstra_path_length(G,'Bangalore','Surat')
2500

In [22]: print nx.dijkstra_path_length(G,'Bangalore','Kolkata')
1700

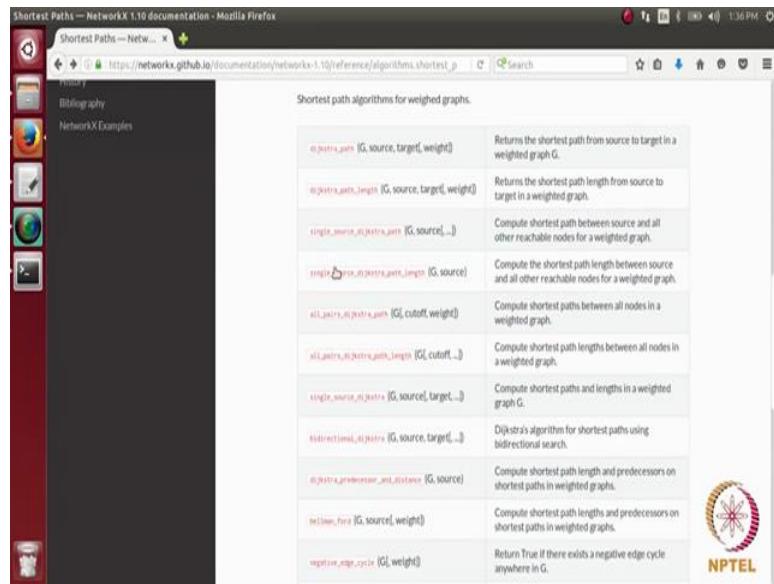
In [23]: print nx.dijkstra_path_length(G,'Surat','Kolkata')
800

In [24]: exit
yayati@yayati-Vostro-3546:/screencast$
```

Why is it turning out to be 2500 let see? So, you let say that if you want to go from Bangalore to Kolkata you can go it in seventeen hundred and let say that you want to go from Kolkata to Surat. So, we can write it this way because it is an undirected graph its 8

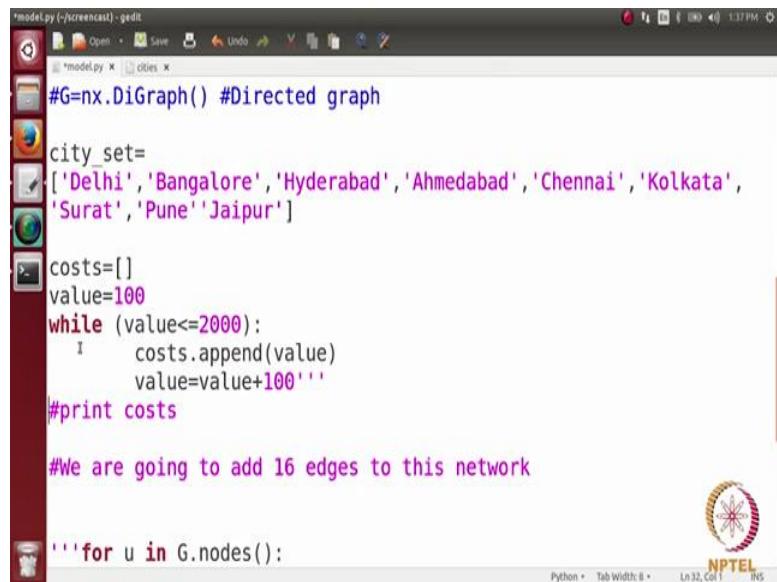
hundred when you some both of these it turns out to be 2500. So, you go from Bangalore to Kolkata which goes you 1700 and then Kolkata to Surat goes you 800 and this some turns out to be 2500 perfect.

(Refer Slide Time: 31:31)



Then we have the single source dijkstra path. So, I will not going to each of this. So, what you can do here is you just give a source node and it will tell you the shortest path from the source node to every then node in the networks. Similarly you can look at its path length and then you have a all pairs extra path which will compute the shortest path between all the nodes in the network and then here also you can look at the path length as well I will just exit.

(Refer Slide Time: 32:05)

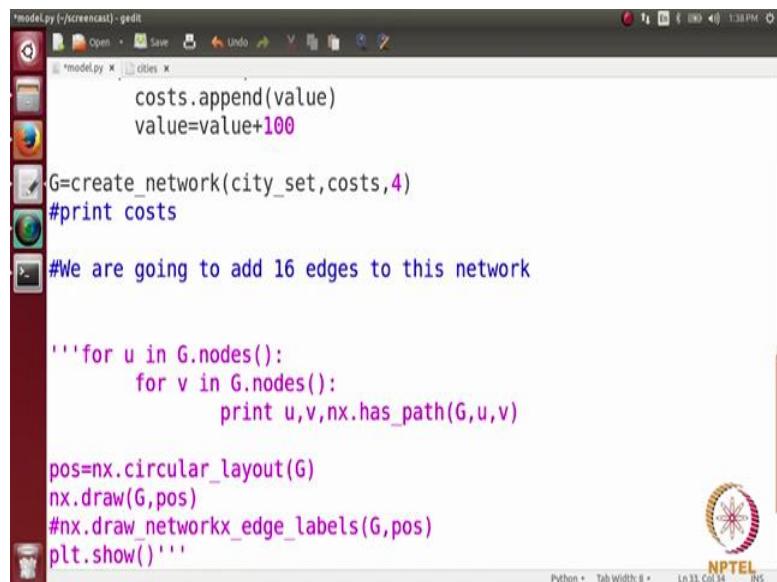


```
#G=nx.DiGraph() #Directed graph
city_set=['Delhi','Bangalore','Hyderabad','Ahmedabad','Chennai','Kolkata',
'Surat','Pune','Jaipur']
costs=[]
value=100
while (value<=2000):
    costs.append(value)
    value=value+100
#print costs
#We are going to add 16 edges to this network
'''for u in G.nodes():
    for v in G.nodes():
        print u,v,nx.has_path(G,u,v)

pos=nx.circular_layout(G)
nx.draw(G,pos)
#nx.draw_networkx_edge_labels(G,pos)
plt.show()'''
```

Coming back to our core now what we are now what we will do is we are going to plot the curve and let see what the curve is going to do. So, let us set first of all this costs rate to be have this city set here to have this costs here and ok.

(Refer Slide Time: 32:57)



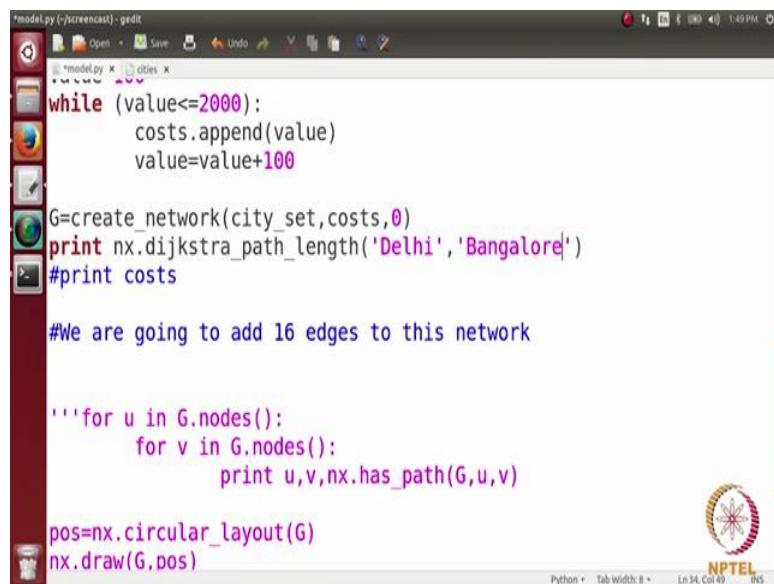
```
costs.append(value)
value=value+100
G=create_network(city_set,costs,4)
#print costs
#We are going to add 16 edges to this network
'''for u in G.nodes():
    for v in G.nodes():
        print u,v,nx.has_path(G,u,v)

pos=nx.circular_layout(G)
nx.draw(G,pos)
#nx.draw_networkx_edge_labels(G,pos)
plt.show()'''
```

We do not need the functions and this is for let us (Refer Time: 32:52) it perfect and we are we had to call this function create underscore network and where we are passing the city set and then our costs and then the number of edges which we want to have. So, let us put very less number of edges 4.

Now, one question what is going to be the path length if there is no connection between 2 nodes. So, we are interested in looking at. So, we have looked at various different extra path finding functions what does this function written if there is no path between a pair of nodes for example, assume that initially we add 0 edges in this network. So, there is no path between any 2 nodes in this networks since there is no edge. So, what happens if I print a; the extra path finding function in this case let us try to do it. So, let us choose to nodes let say Delhi and Bangalore.

(Refer Slide Time: 34:15)



The screenshot shows a terminal window titled "model.py (~/screen cast) - gedit". The code is as follows:

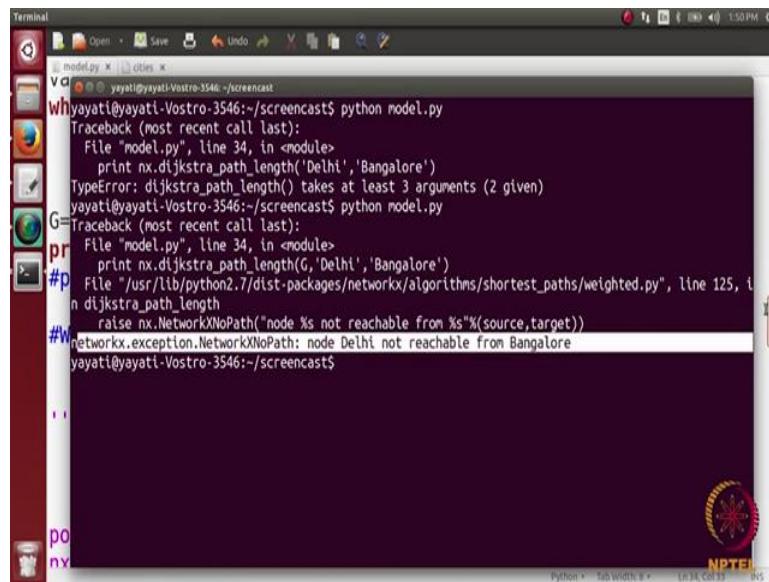
```
model.py (~/screen cast) - gedit
while (value<=2000):
    costs.append(value)
    value=value+100
G=create_network(city_set,costs,0)
print nx.dijkstra_path_length('Delhi','Bangalore')
#print costs
#We are going to add 16 edges to this network
'''for u in G.nodes():
    for v in G.nodes():
        print u,v,nx.has_path(G,u,v)
pos=nx.circular_layout(G)
nx.draw(G,pos)

```

The window includes standard Linux desktop icons in the top bar and a small NPTEL logo in the bottom right corner.

So, what I am going to do is I am going to print n x.dijkstra and then path and then length path length between see Delhi and Bangalore let us always function in a.

(Refer Slide Time: 34:43)

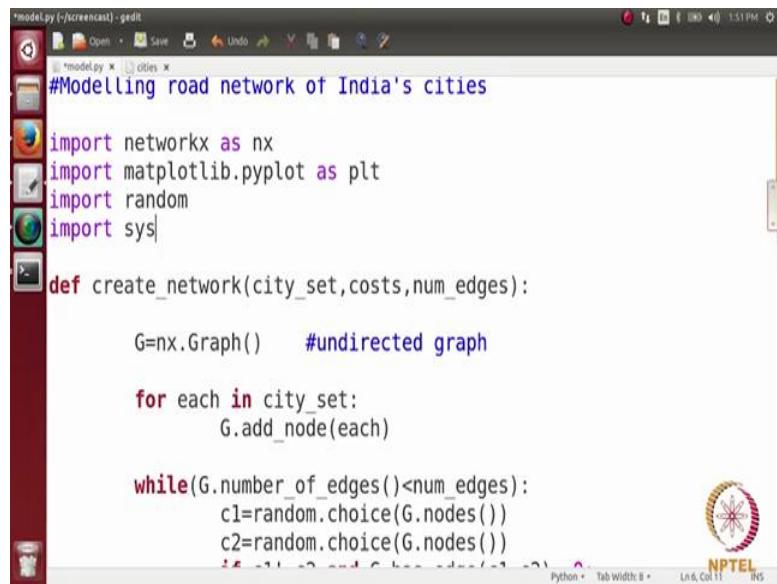


A screenshot of a Linux desktop environment showing a terminal window. The terminal title is 'Terminal' and the current directory is '/model.py'. The user has run the command 'python model.py'. The output shows a traceback for a 'TypeError' from the 'nx.dijkstra\_path\_length()' function. The error message states that the function takes at least 3 arguments (2 given). The user then runs the command again, and the terminal shows an exception from the 'networkx.exception.NetworkXNoPath' module, stating that node 'Delhi' is not reachable from 'Bangalore'. The terminal window has a dark background with light-colored text. The desktop interface includes icons for various applications like a browser, file manager, and system tools.

```
yayati@yayati-Vostro-3546:~/screencast$ python model.py
Traceback (most recent call last):
  File "model.py", line 34, in <module>
    print nx.dijkstra_path_length('Delhi','Bangalore')
TypeError: dijkstra_path_length() takes at least 3 arguments (2 given)
yayati@yayati-Vostro-3546:~/screencast$ python model.py
G=Traceback (most recent call last):
  File "model.py", line 34, in <module>
    print nx.dijkstra_path_length(G,'Delhi','Bangalore')
#P  File "/usr/lib/python2.7/dist-packages/networkx/algorithms/shortest_paths/weighted.py", line 125, i
n dijkstra_path_length
    raise nx.NetworkXNoPath("node %s not reachable from %s"%(source,target))
#W networkx.exception.NetworkXNoPath: node Delhi not reachable from Bangalore
yayati@yayati-Vostro-3546:~/screencast$
```

So, it has given a type error that we should have given 3 arguments. So, what is missing here is graph G right nodes run it. So, you see here that there is an exception. So, the code has written an exception and the exception is node Delhi not reachable from Bangalore. So, we see that how do we catch these exceptions in the network assume that I do not want to return an exception here rather I want my code to show mw as 0 here at the path length from Delhi to Bangalore not 0, it is infinity, right. So, my answer should be infinity or since a graph is very small let us represent infinity here by a big number let say 10,000. So, we are representing infinity with 10,000 though it is not a very well accepted notion, but we are just using it in this code for the sake of convenience.

(Refer Slide Time: 35:43)



```
#Modelling road network of India's cities

import networkx as nx
import matplotlib.pyplot as plt
import random
import sys

def create_network(city_set,costs,num_edges):

    G=nx.Graph()      #undirected graph

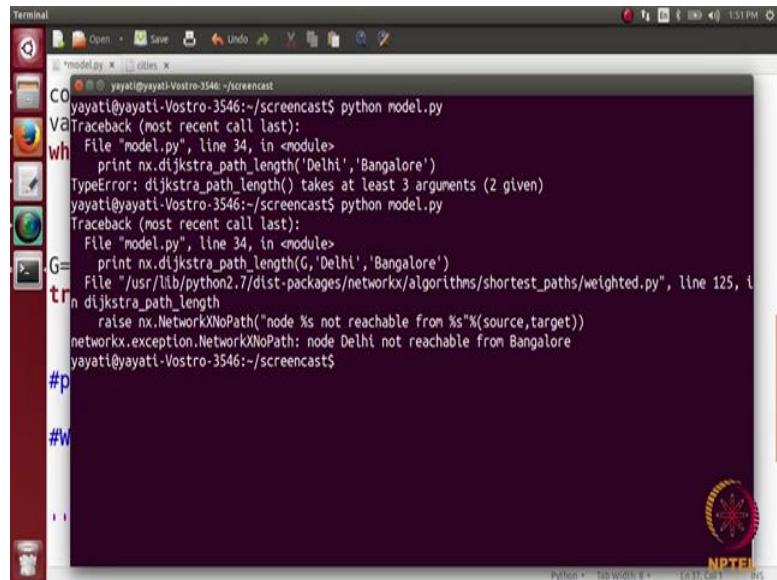
    for each in city_set:
        G.add_node(each)

    while(G.number_of_edges()<num_edges):
        c1=random.choice(G.nodes())
        c2=random.choice(G.nodes())
        if c1!=c2 and G.has_edge(c1,c2)==False:
            G.add_edge(c1,c2,weight=random.randint(1, costs))

    return G
```

So, what we do this in this case is something called the exception handling which is very simple. So, let us look at we just need to import the module sees here after include importing this package what do I do it I just put the statement inside a try (Refer Time: 35:56).

(Refer Slide Time: 35:53)



```
yayati@yayati-Vostro-3546:~/screencast$ python model.py
Traceback (most recent call last):
  File "model.py", line 34, in <module>
    print nx.dijkstra_path_length('Delhi','Bangalore')
TypeError: dijkstra_path_length() takes at least 3 arguments (2 given)
yayati@yayati-Vostro-3546:~/screencast$ python model.py
Traceback (most recent call last):
  File "model.py", line 34, in <module>
    print nx.dijkstra_path_length(G,'Delhi','Bangalore')
  File "/usr/lib/python2.7/dist-packages/networkx/algorithms/shortest_paths/weighted.py", line 125, in dijkstra_path_length
    raise nx.NetworkXNoPath("node %s not reachable from %s%(source,target))"
networkx.exception.NetworkXNoPath: node Delhi not reachable from Bangalore
yayati@yayati-Vostro-3546:~/screencast$
```

So, what my try (Refer Time: 35:58) does? It gets this path length L as nx extra path length G Delhi Bangalore and in case there is an error. So, we will looked at an error here and these exception handling and this error was a key error means here.

(Refer Slide Time: 36:25)

```
model.py yayati@yayati-Vostro-3546:~/screencast$ python model.py
Traceback (most recent call last):
  File "model.py", line 34, in <module>
    print nx.dijkstra_path_length('Delhi','Bangalore')
TypeError: dijkstra_path_length() takes at least 3 arguments (2 given)
yayati@yayati-Vostro-3546:~/screencast$ python model.py
Traceback (most recent call last):
  File "model.py", line 34, in <module>
    print nx.dijkstra_path_length(G,'Delhi','Bangalore')
  File "/usr/lib/python2.7/dist-packages/networkx/algorithms/shortest_paths/weighted.py", line 125, in dijkstra_path_length
    raise nx.NetworkXNoPath("node %s not reachable from %s"%(source,target))
networkx.exception.NetworkXNoPath: node Delhi not reachable from Bangalore
yayati@yayati-Vostro-3546:~/screencast$ exit()
#
```

So, we except in the case of an exception what does we do it let say we print error let see how this piece of code works. So, you see here that they words an exception and this exception was because there was no path and we have printed here error.

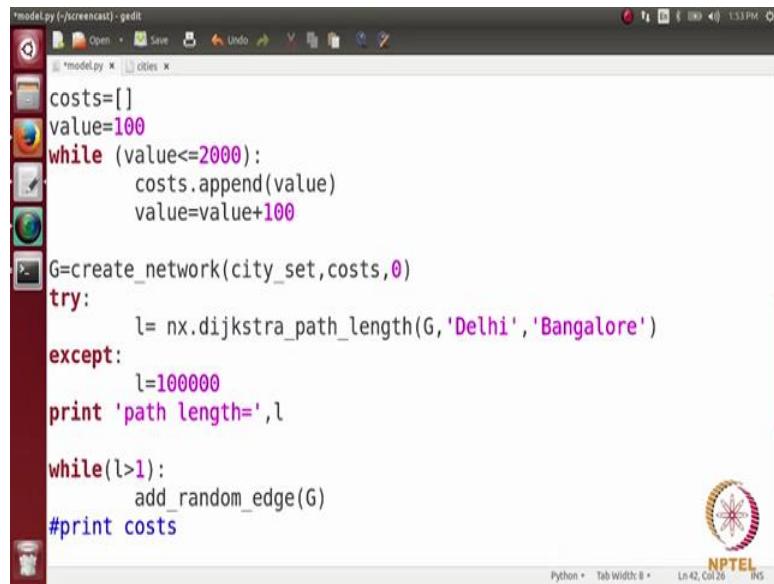
(Refer Slide Time: 36:52)

```
model.py yayati@yayati-Vostro-3546:~/screencast$ python model.py
Traceback (most recent call last):
  File "model.py", line 34, in <module>
    print nx.dijkstra_path_length('Delhi','Bangalore')
TypeError: dijkstra_path_length() takes at least 3 arguments (2 given)
yayati@yayati-Vostro-3546:~/screencast$ python model.py
Traceback (most recent call last):
  File "model.py", line 34, in <module>
    print nx.dijkstra_path_length(G,'Delhi','Bangalore')
  File "/usr/lib/python2.7/dist-packages/networkx/algorithms/shortest_paths/weighted.py", line 125, in dijkstra_path_length
    raise nx.NetworkXNoPath("node %s not reachable from %s"%(source,target))
networkx.exception.NetworkXNoPath: node Delhi not reachable from Bangalore
yayati@yayati-Vostro-3546:~/screencast$ python model.py
path length= 100000
#yayati@yayati-Vostro-3546:~/screencast$ exit()
```

So, what we can do is instead of printing here error we can set this L to be 10,000; 10,000 we will let say even 100,000 and then we print here path length equals to L and let see here when be run this code again we get here a path length of 100,000. So, what this is doing whenever there is no path between 2 nodes it will written 100,000 which is

a very big number according to a code according to a network. So, it is a very small network, now what I want to do is what we want to do is we want to see a plot and what will this plot do there we be 2 cities in our network and will be adding random edges and as the at random edges we are interested to see how this path length decreases.

(Refer Slide Time: 37:59)



The screenshot shows a terminal window titled "model.py (~/screencast) · gedit". The code is as follows:

```
model.py (~/screencast) · gedit
costs=[]
value=100
while (value<=2000):
    costs.append(value)
    value=value+100

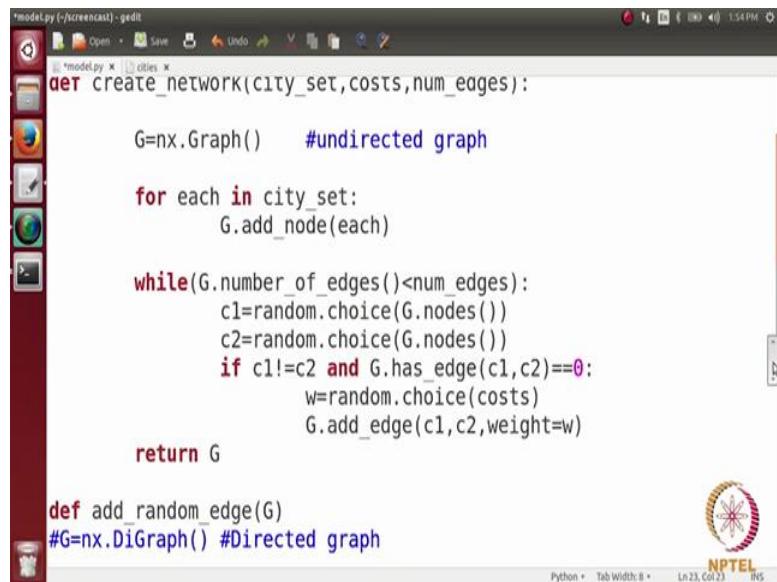
G=create_network(city_set,costs,0)
try:
    l=nx.dijkstra_path_length(G,'Delhi','Bangalore')
except:
    l=100000
print 'path length=',l

while(l>1):
    add_random_edge(G)
#print costs
```

The window includes standard file operations (Open, Save, Undo, Redo) and a status bar at the bottom indicating "Python" and "Ln.42, Col.26". The NPTEL logo is visible in the bottom right corner of the terminal window.

So, let us write a small piece of 4 hold this. So, what we are going to do is initially we have this graph here g. So, what we are going to do is while, we are going to do is we have calculated here I will write and L was here 10,000 what we want L to become is let say one because the minimum path length between 2 nodes is one till they become directly connected. So, till while your L it is greater than one. So, what you do here is you add an edge randomly. So, I will call a function here add random edge right add random edge in G. So, how this function is going to work is.

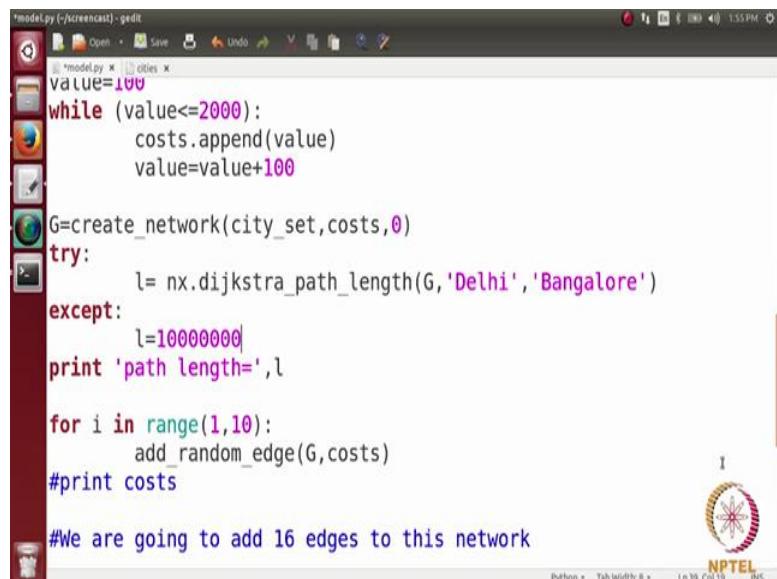
(Refer Slide Time: 38:50)



```
model.py (~/screencast) · gedit
model.py cities
def create_network(city_set,costs,num_edges):
    G=nx.Graph()      #undirected graph
    for each in city_set:
        G.add_node(each)
    while(G.number_of_edges()<num_edges):
        c1=random.choice(G.nodes())
        c2=random.choice(G.nodes())
        if c1!=c2 and G.has_edge(c1,c2)==0:
            w=random.choice(costs)
            G.add_edge(c1,c2,weight=w)
    return G
def add_random_edge(G)
#G=nx.DiGraph() #Directed graph
```

Lets create a function here define add random edge G and so we have the city set here.

(Refer Slide Time: 39:08)



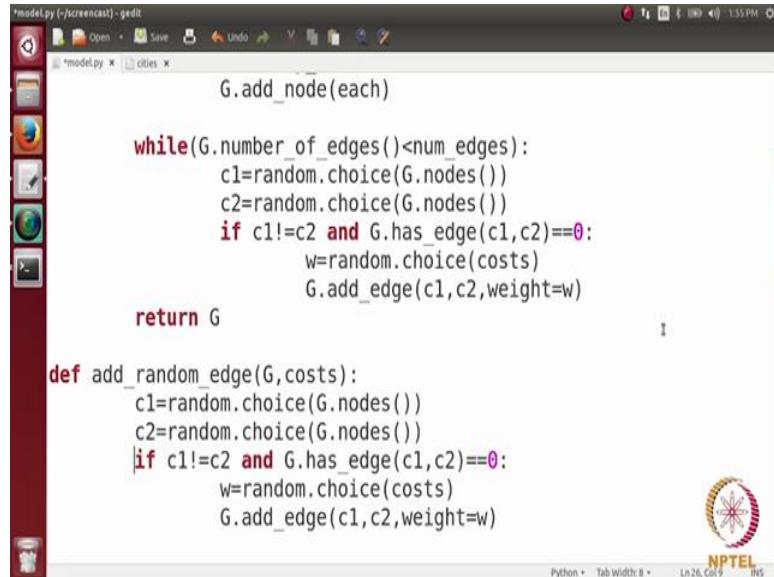
```
model.py (~/screencast) · gedit
model.py cities
value=100
while (value<=2000):
    costs.append(value)
    value=value+100
G=create_network(city_set,costs,0)
try:
    l= nx.dijkstra_path_length(G,'Delhi','Bangalore')
except:
    l=10000000
print 'path length=',l
for i in range(1,10):
    add_random_edge(G,costs)
#print costs
#We are going to add 16 edges to this network
```

But we want to pass here the costs we goes every node should have a costs one thing here this pass length is actually equal to be cost here right. So, this is equal to a travelling cost.

So, we cannot say it is less than one. So, we will just run this code for some number of steps because path length is never going to be one here because our path length is a travelling cost here. So, what will go is for i in range 1 to 10 you will see for the 10 times

what happens here and let us make it a little bit larger number. So, add random edge G costs.

(Refer Slide Time: 40:06)



```
model.py (~/screencast) - gedit
G.add_node(each)

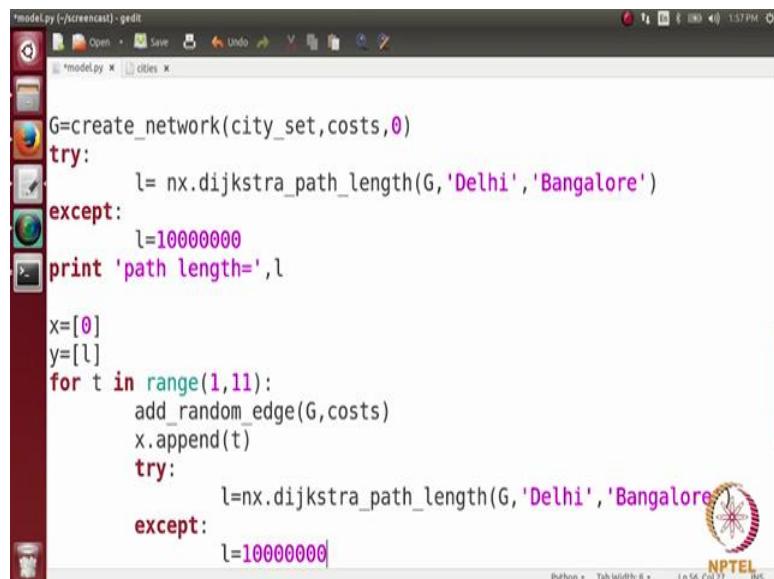
while(G.number_of_edges()<num_edges):
    c1=random.choice(G.nodes())
    c2=random.choice(G.nodes())
    if c1!=c2 and G.has_edge(c1,c2)==0:
        w=random.choice(costs)
        G.add_edge(c1,c2,weight=w)

return G

def add_random_edge(G,costs):
    c1=random.choice(G.nodes())
    c2=random.choice(G.nodes())
    if c1!=c2 and G.has_edge(c1,c2)==0:
        w=random.choice(costs)
        G.add_edge(c1,c2,weight=w)
```

So, what it is going to do is what exactly we have done above. So, what we have done is right we have taken 2 random nodes and if there was no end between them we have chosen edge and added between them. So, exactly the same thing we are going to do here. So, essentially we are just copy pasting this code.

(Refer Slide Time: 40:35)



```
model.py (~/screencast) - gedit
G=create_network(city_set,costs,0)
try:
    l= nx.dijkstra_path_length(G,'Delhi','Bangalore')
except:
    l=10000000
print 'path length=',l

x=[0]
y=[l]
for t in range(1,11):
    add_random_edge(G,costs)
    x.append(t)
    try:
        l=nx.dijkstra_path_length(G,'Delhi','Bangalore')
    except:
        l=10000000|
```

So, what we are doing is we will choose 2 nodes randomly and if there is no edge between these 2 nodes will add an edge with their weight being equal to w. So, since done here add random edge G.edges and after that what we are going to do is. So, since we have to plot this graph over 10 timestamps we need 2 arrays 1 is for x axis, 1 is for y axis in the 11. So, x axis is a time which is essentially I let us represent it as t.

So, we will have an array x and we will have an array y x will be initially having the value 0 right because it is a 0 timestamp when y will initially having the value 1 which is a very big number code we are going to do here now is x.append t that is a time and in y what we have to open again if the dijkstra path length. So, it is again going to be a try function. So, we are going to try what give me are going to try is let us say let us taking L; L equals to n x.dijkstra and then path length then G comma Delhi comma Bangalore right and then we are going to except and adder. So, this exception is in case where there is no path between the 2.

(Refer Slide Time: 42:32)

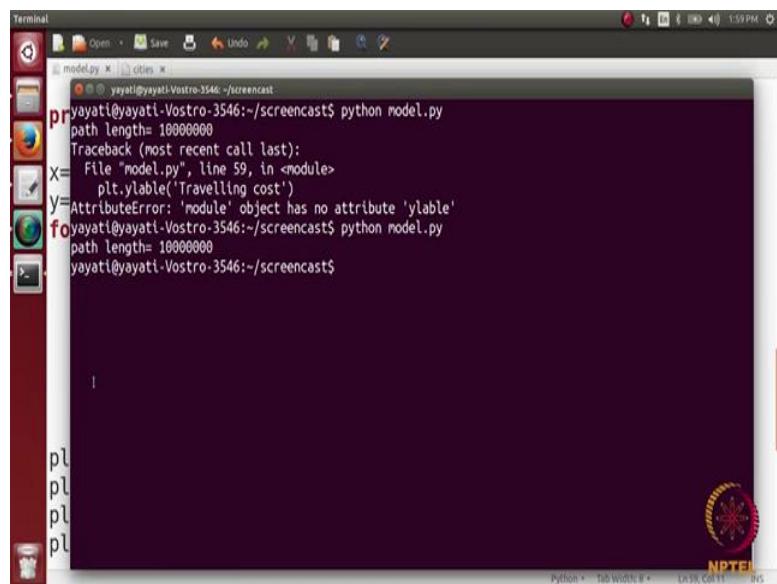
```

Terminal
yayati@yayati-Vostro-3546:~/screencast$ python model.py
Traceback (most recent call last):
  File "model.py", line 34, in <module>
    print nx.dijkstra_path_length('Delhi','Bangalore')
TypeError: dijkstra_path_length() takes at least 3 arguments (2 given)
yayati@yayati-Vostro-3546:~/screencast$ python model.py
Traceback (most recent call last):
  File "model.py", line 34, in <module>
    print nx.dijkstra_path_length(G,'Delhi','Bangalore')
  File "/usr/lib/python2.7/dist-packages/networkx/algorithms/shortest_paths/weighted.py", line 125, in dijkstra_path_length
    raise nx.NetworkXNoPath("node %s not reachable from %s"%(source,target))
networkx.exception.NetworkXNoPath: node Delhi not reachable from Bangalore
yayati@yayati-Vostro-3546:~/screencast$ python model.py
path length= 100000
yayati@yayati-Vostro-3546:~/screencast$ cler

```

So, here we are going to set L equals to it remains the same which is the big number 1, 2, 3, 4, 5, 6. So, there are 7 0s, perfect and then what we are going to do is y.append L and then we want to plot this. So, for plotting this we have plt.plot and we have x and then y and we can actually give your axis name let us say that plt.x label that is a x axis is a time and plt.y label y axis is a let us say travelling cost and then we have a title let us say change in travelling cost as more roads are added in the network.

(Refer Slide Time: 43:34)



A screenshot of a Linux desktop environment showing a terminal window titled "Terminal". The terminal window has a dark purple background. Inside, there is a command-line session:

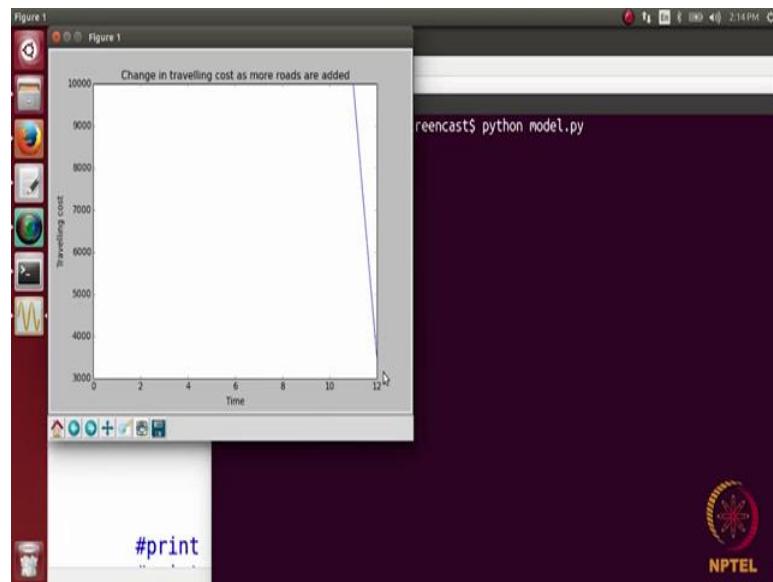
```
yayati@yayati-Vostro-3546:~/screencast$ python model.py
path length= 10000000
Traceback (most recent call last):
  File "model.py", line 59, in <module>
    plt.ylabel('Travelling cost')
yayati@yayati-Vostro-3546:~/screencast$ python model.py
path length= 10000000
yayati@yayati-Vostro-3546:~/screencast$
```

The terminal window is part of a desktop interface with a dock on the left containing icons for various applications like a browser, file manager, and terminal. The desktop background is a dark purple color with a circular logo in the bottom right corner.

So, let us try to see. So, clear here the small spelling mistake here. So, I will take their having some problem it is printed are path length. So, here we have mention the small changes to this code. So, the value of L we have just change it to be 10,000 we can just to so that the plot looks nice and should not we where to high, but it is still higher on the path that we have very less as oppose we are just 2500 or something and then this take when y.append L has been shifted.

So, instead of adding at the end we added inside the loop because we have this break statement inside the try block which is needed. So, if this break statement is there and we could y.append at the end. So, the value is not append it. So, we append the values to are array y in between the try block and the added block and then we have a plt road show here which we were which was initially missing so which is required to show us the plot.

(Refer Slide Time: 44:56)

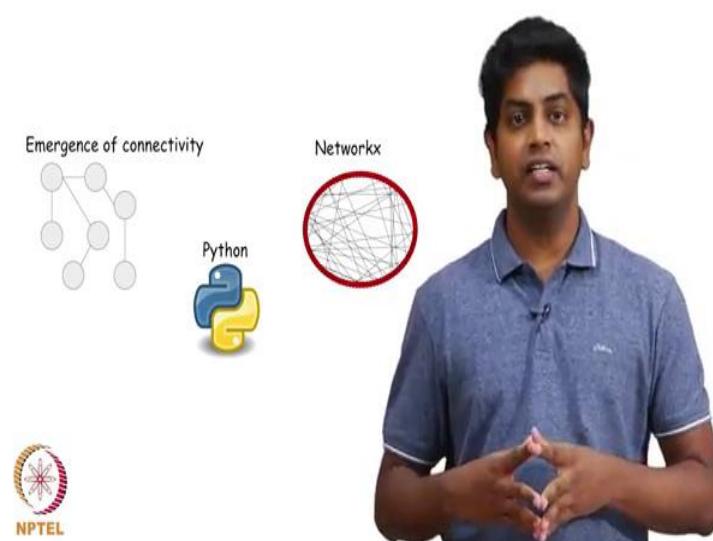


So, let us see here. So, when we execute this code. So, we see here something. So, the initial cost was 10,000. So, this was the cost which was which is actually infinity furors right. So, initially Delhi and Bangalore were not connected and after sometime steps. So, after some 11 time steps, so the random edges are keep creating in the network and after some eleven times steps this cost drops and comes to 3,000. So, it becomes connected here were just a small code. So, it might not be very clear to you. So, if you have any doubt regarding this you can leave it in the discussion forums and we can discuss it further. So, this was just a basing basic plotting function which we can do using python and networkx just an example for that.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

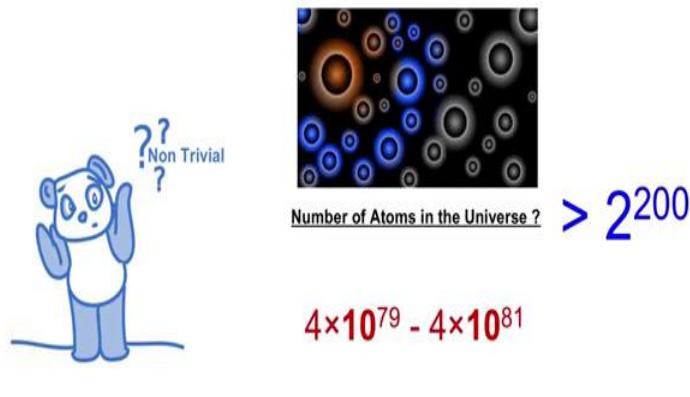
**Lecture – 07**  
**Introduction to Social Networks**  
**Social Networks: The Challenge**

(Refer Slide Time: 00:07)



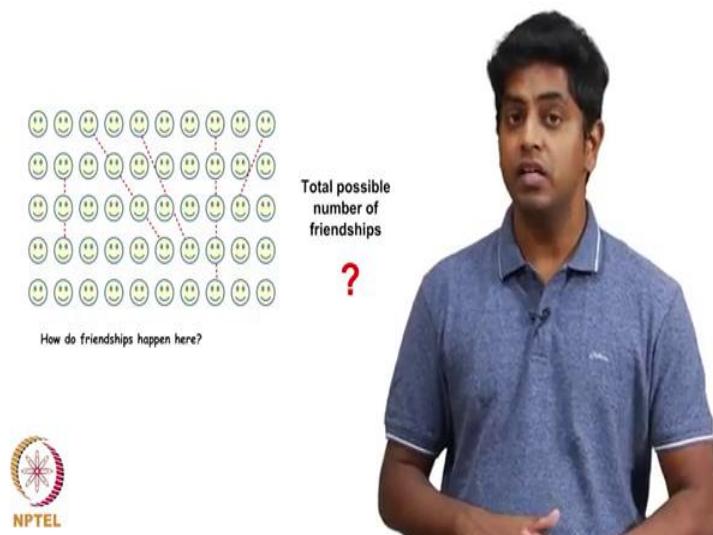
We saw very nice emergent property on social networks and I thought you how to use python on network x, now let shift our gas and try looking at something that is very non trivial about social networks.

(Refer Slide Time: 00:24)



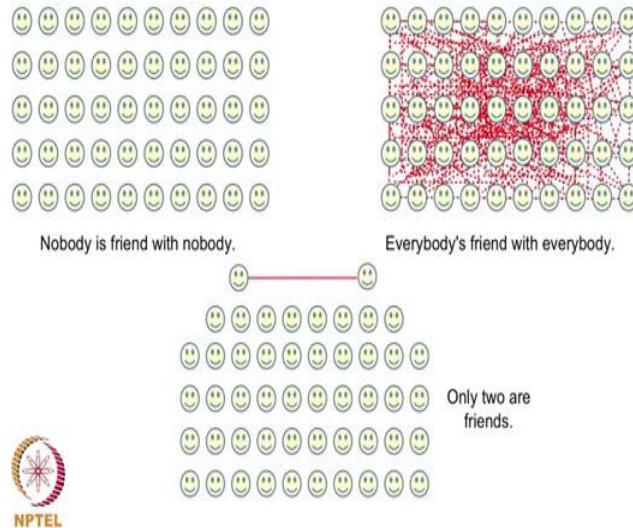
What is a non trivial about social networks? Let me ask a very seemingly unrelated question here; what you think is the atoms in the universe? So, very big number, it is roughly 2 to the 200 plus more than that 2 to the 200 plus is the number of atoms in the universe, so much to be precise. So, what? Observe: look at the example that we just now saw in one of our previous lectures.

(Refer Slide Time: 01:00)



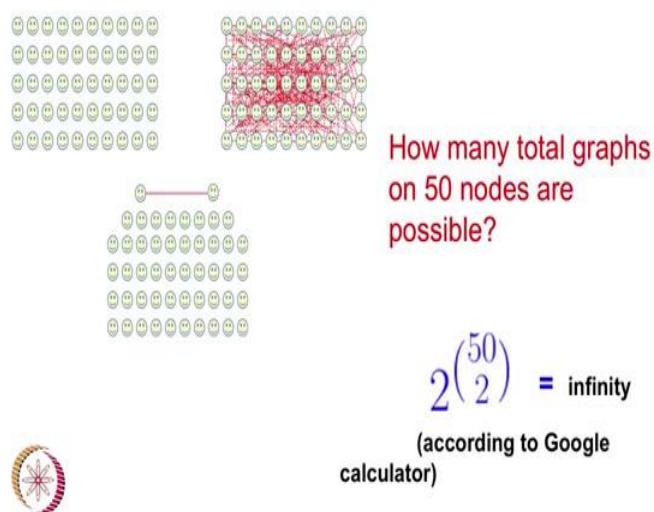
A classroom with 50 people correct and we questioned about friendships there. Now let me ask you this question, what are the total possible friendships that are possible on this graph on just 50 people?

(Refer Slide Time: 01:20)



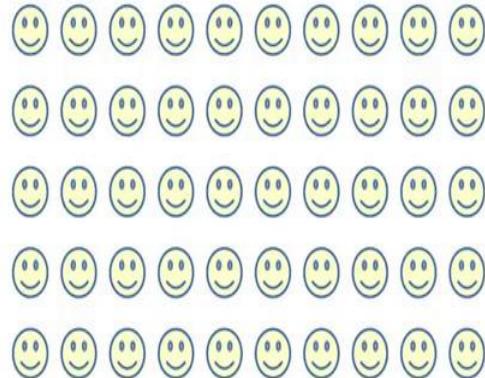
By that I mean look at this, this is one possible friendship where nobody is friends with each other, the other possibilities is all of them are friends with each other, other possibilities there are only these 2 friends rest are not friends so on and so on and so forth.

(Refer Slide Time: 01:39)



If I go on enumerating all possible graphs on 50 nodes, how many will I get? How do I even answer this question? I leave it as an exercise. I will quickly answer it here. The answer is 2 to the power of 50 joules 2 and that is a very big number and the number is so much, this is the total possible graphs on 50 nodes; just 50 nodes.

(Refer Slide Time: 02:06)

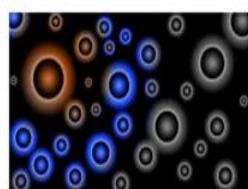


If you take 50 people in this classroom, all of you sitting here 50 people let us say.

(Refer Slide Time: 02:13)

$$2^{\binom{50}{2}}$$

**Much much much more than**



Number of Atoms in the Universe ?

$> 2^{200}$



What are all possible graphs that I can think of on the 50 people, the total number of graphs on 50 people is much much more not just more, but a lot lot lot more than the atoms in the universe.

(Refer Slide Time: 02:31)



And that is what makes analyzing social networks very very difficult, the inherent complexity in it is way too much for even a super computer to handle and this is what makes it interesting that despite the fact that there is so much of complexity, we tend to observe some very nice properties.

(Refer Slide Time: 02:41)

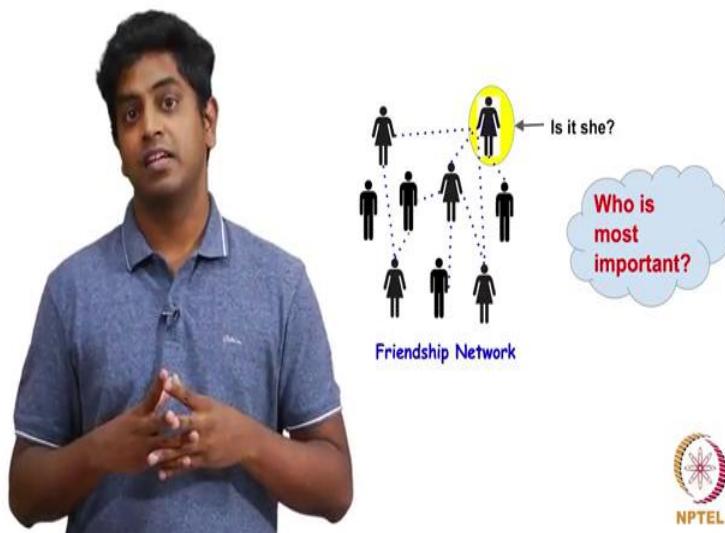


I am going to start enumerating this very nice questions and properties that this entire course is going to cover. Let us start with a first such a property rather what I will do right now is I will start off telling you a couple of nice; may not be couple may be 4 to 5 such nice ideas which I will be covering in the course. In fact, the course will have several nice ideas, but the main ideas I am going to tell you right now on what will be covered in the course.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture – 08**  
**Introduction to Social Networks**  
**Google Page Rank**

(Refer Slide Time: 00:06)



Given a friendship network, who is the most important person? How do we judge who is very important? Do you see who has the maximum number of friendships or is there something more than just the number of links that one has?

(Refer Slide Time: 00:27)



In fact, this is the question which yielded 450 billion dollar industry; I am telling you the story of google.

(Refer Slide Time: 00:31)



So, google basically asked this question; given a bunch of web pages that are linked to each other which is the most important web page of all these things?

(Refer Slide Time: 00:44)



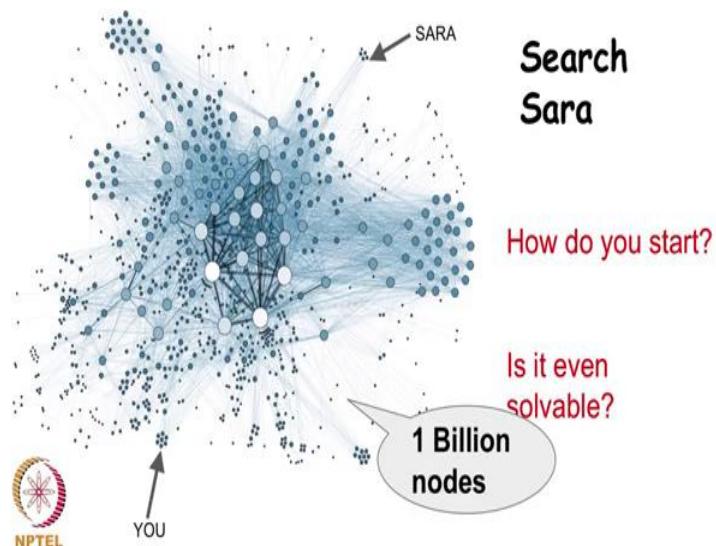
And the answer is so straight forward that when we come to this chapter, you will feel how can such an obvious observation result in 450 billion dollar industry. In fact, it might appear to be very obvious or it took team of 2 people to bring this to the lime light and use this seemingly simple concept to solve one of the most important problems in computer science which is how do you rank nodes in a network.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture – 09**  
**Introduction to Social Networks**  
**Searching in a Network**

So, second in the list is the problem of searching in a network, in a network with roughly let say one billion nodes.

(Refer Slide Time: 00:17)



If I were to tell you go and search for a particular node starting from some random node, how will you even start? Can this even be solved? I told you there are so many possible networks that one can think of on even 15 node graph, and I am giving you a 1 billion node network; a network is 1 billion nodes, how can you search for a node here?

Well such complicated questions very interestingly they have very simple and straightforward answers. In fact, we are going to show that searching on a network especially on a real world network is very easy.

(Refer Slide Time: 01:02)



**log n steps**

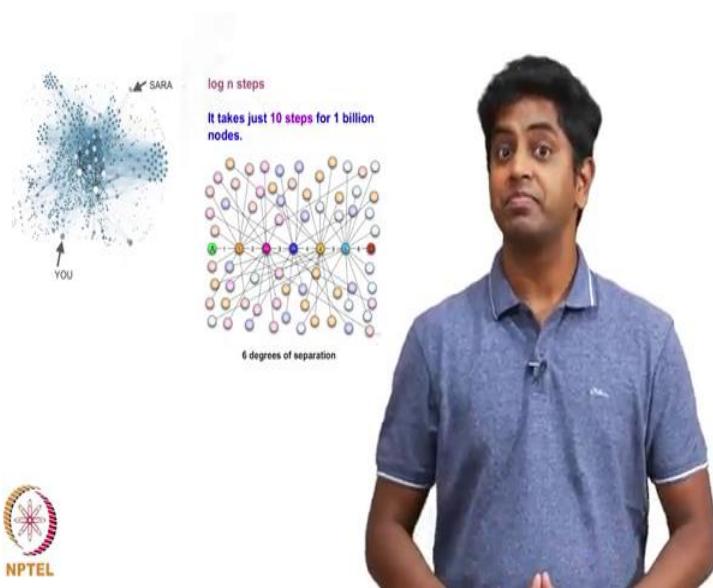
**It takes just  
10 steps for 1  
billion nodes.**



In fact, if you have  $n$  number of nodes you can search with only roughly  $\log n$  number of steps to be precise actually its  $\log^2 n$ , but the point is in  $\log$  number of transactions you can achieve searching, I am sure you know what one means by logarithm, if  $n$  is the number  $\log n$  is the number of digits in  $n$ .

So, if the number of nodes in a network is  $n$  let us say 1 billion, the effort involved in searching in this big network is the number of digits in one billion single digit number or let say maximum 10, 15 or even 20, in 20 transactions you can find any node in the network, why? How?

(Refer Slide Time: 01:50)



We exploit a nice property in such networks called the small world phenomena; gorgon for the time being, but do not worry when we come there you will see it is a very straightforward concept, but very elegant very beautiful and sparkling part is that it gives you an answer like a charm.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture – 10**  
**Introduction to Social Networks**  
**Link Prediction**

Next in the list is something that is actually not believable; what do you know about predictions?

(Refer Slide Time: 00:17)



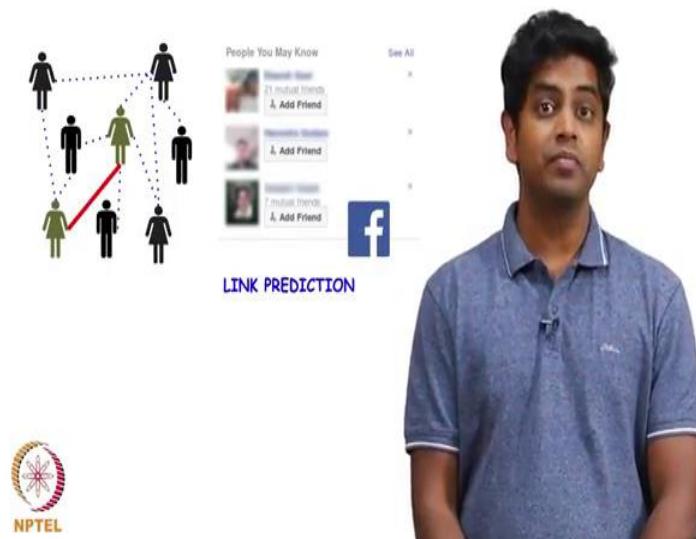
That someone come and tell you that show me your hand and I will tell you, what is your future? Your future is hidden here. I do not know the signs of it. It is may be it is true, it is not true, very debatable.

(Refer Slide Time: 00:31)



But I can assure you of the fact that predictions on networks work really really well, what do I mean by this?

(Refer Slide Time: 00:36)



I will look at your friendship network and tell you who could possibly be your next friend sounds like this is very familiar to you right, what am I talking about? Have you heard of he could be your prospective friend? Friend suggestions in Facebook; it is not a straight forward method that they use it, do not show a random person who has a lot of

common friends with you, there is sophisticated algorithm which is in play there what is that algorithm, we will discuss that in this chapter called link prediction.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture – 11**  
**Introduction to Social Networks**  
**The Contagions**

Later on we will impact look at very interesting humanly questions, what do I mean by humanly questions?

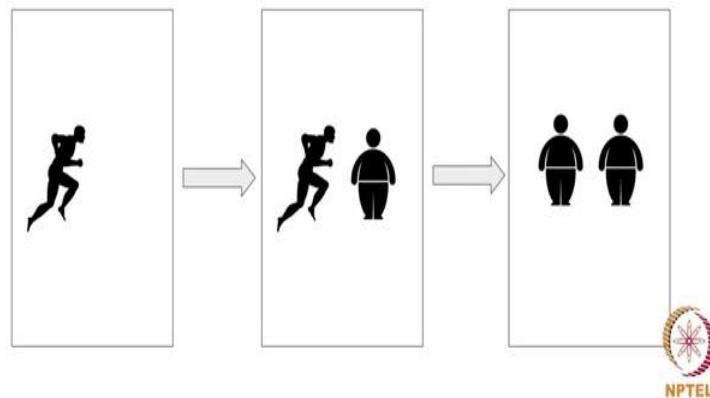
(Refer Slide Time: 00:18)



We are in fact, going to answer this so called philosophical questions with very elegant computing computer science and mathematics; what are those questions?

(Refer Slide Time: 00:25)

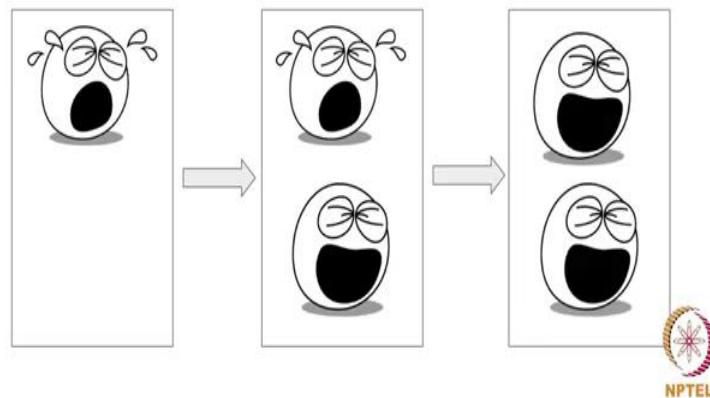
### Is obesity contagious?



Do you think obesity is contagious do you think if you are friends with the person who is obese you run into the danger of becoming obese?

(Refer Slide Time: 00:36)

### Is happiness contagious?

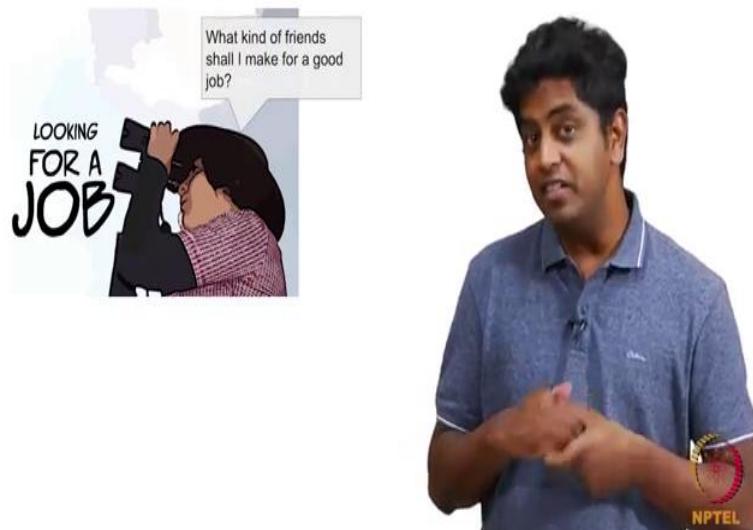


Do you think if you are surrounded by happy people? You will be happy too. In fact, you will be surprised to note that happiness is not just contagious to one level, it is contagious to 2-3 levels by that I mean your friend's friend; if he is happy you will be happy; sounds like cooked up story, no it is not true, this is a very well established scientific fact, we will be seeing such questions and their answers.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture – 12**  
**Introduction to Social Networks**  
**Importance of Acquaintances**

(Refer Slide Time: 00:12)



Very interesting last part which would be a nice moral for all of you on the hunt for a good job is how to get a good job. What kind of friends should you make in order to get a good job?

(Refer Slide Time: 00:27)



This comes under the topic of what is called the strength of weak ties. It is believed that your acquaintances; not close friends; acquaintances friends were not so very close, but yeah they are friends, those kind of people are actually very important for you to get your dream job or only a dream job, your prospective spouse is also a recommendation given by a far away person and acquaintances not a close friend, the strength of weak ties. We will see more of it when we encounter this idea in the forth coming chapters.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

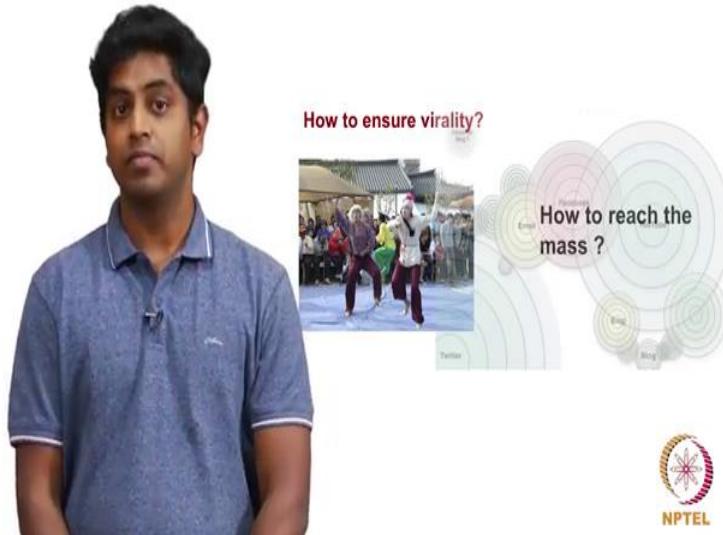
**Lecture – 13**  
**Introduction to Social Networks**  
**Marketing on Social Networks**

(Refer Slide Time: 00:19)



Lastly, how do you market a product on social networks of course, the era of putting advertisements on let us say notice board or let us say a TV commercial is long gone, today people use social networks to market their product.

(Refer Slide Time: 00:30)



What is the science of this? How do you ensure that your product becomes viral or a piece of video becomes viral? What should you note rather what should you do in order to market something so that it reaches the mass? We are going to discuss the whole lot about this and we are also going to see the sort of link between how do you market a product to how does a belief propagate, how does a piece of information propagate, how do things cascade so on and so forth.

So, these are all just some of the cool ideas that we will see in the course. In fact, the course is some 5-6 times more than what I just now introduced. These are some of the cool ideas that we will be seeing, there are lot more ideas coming up.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture – 14**  
**Handling Real-world Network Datasets**  
**Introduction to Datasets**

(Refer Slide Time: 00:20)



So, here we are with a brand-new chapter on datasets.

(Refer Slide Time: 00:26)



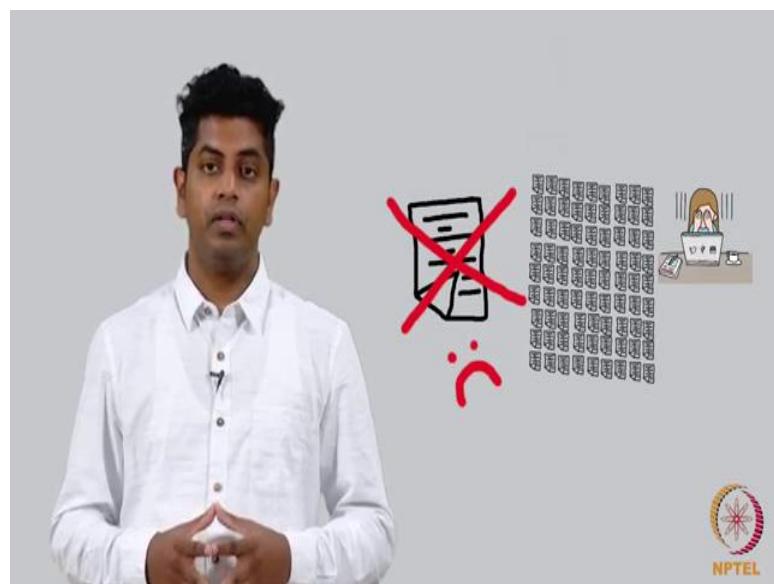
So, we are going to teach you the importance of crunching datasets and how to go about crunching them.

(Refer Slide Time: 00:30)



So firstly, what are datasets? Especially datasets that are relevant to social networks.

(Refer Slide Time: 00:38)



You see there was a time when we did not have any data and today we are in that era where it is all about data we have so much of data that we do not know how to make sense out of this data that a complete contrast over it was let us say some 30-40 years back.

(Refer Slide Time: 00:53)



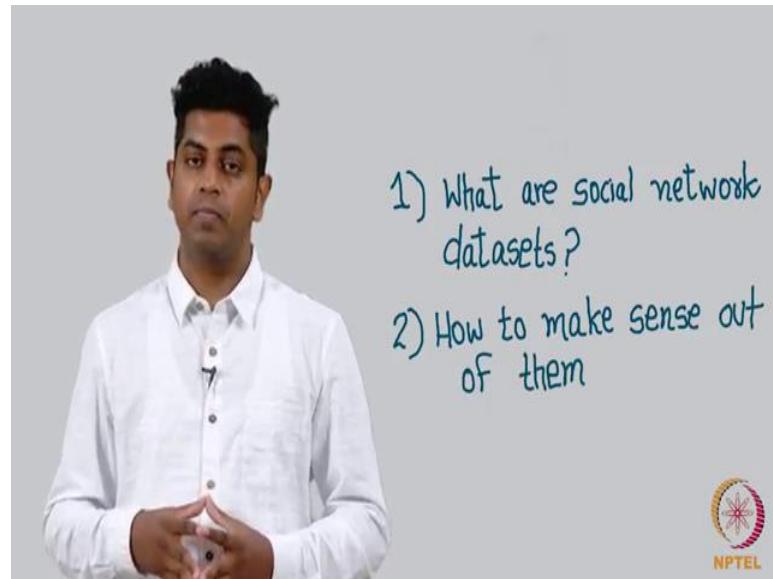
For example there was an; there was time when people look at the friendship network on 30 to 35 nodes, this is called to the Zachary Karate network, it had some 32 nodes and some edges on it, it is very popularly studied, very widely studied and are very popular social network, but today; back then we did we probably did not have a way in which we could get a bigger internet.

(Refer Slide Time: 01:28)



But today with the advent of internet and higher computing and storage facilities, we can crunch a whole lot of data whether we can crunch or not we have a whole lot of data.

(Refer Slide Time: 01:42)



So, this week this a entire module is all about understanding what are social network datasets and how one can make sense out of these things.

(Refer Slide Time: 01:52)



Please ensure that you have understood Python and networkx and another APIs on Python really well, especially the things that we thought in the last week.

(Refer Slide Time: 02:03)



I am sure you all know of this Play-Doh; Play-Doh is where you have this sand like thing that is sort of you can make any shape out of it and you realize that the options are multitude, you can go on making a small box out of it or a cute animal out of it, I mean we are in with limit is just in your imagination.

(Refer Slide Time: 02:28)

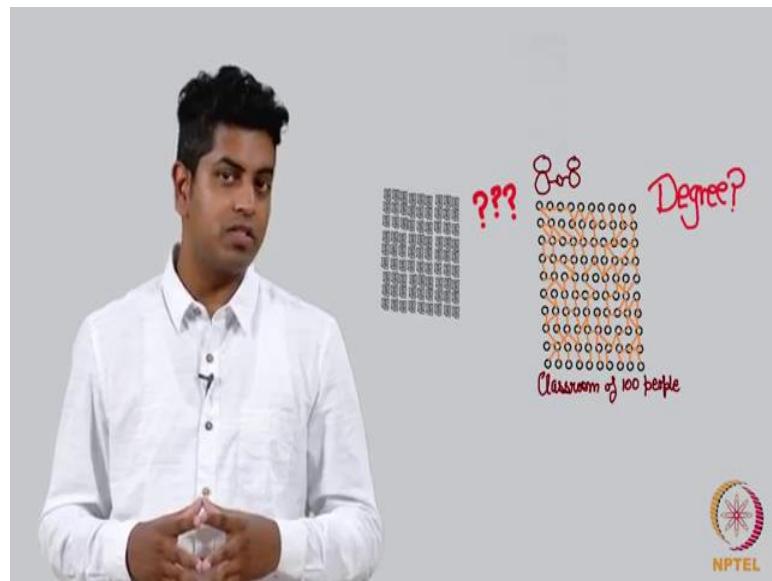


So, similarly when people got hold of this big network data set they realized that one can actually asked the whole lot of questions on it. Let us say first question that one can ask

on our friendship network, there is a classroom full of 100 students and data is available to me of their friendship network, who is friends with whom?

What kind of questions can we ask the obvious question is the graph connected we have discussed in a on this topic before most all graphs are connected, fine?

(Refer Slide Time: 03:06)



Second question; what can you say about the degree of the nodes?

(Refer Slide Time: 03:10)



By degree you mean; pick a person look at how many friends he has, pick another person look at how many friends he or she has so on and so forth, make a list of all these friendships of individual people, what is the average here and what is the standard deviation of this list what are the mean by this? By this I mean is the average sort of do everyone agree with the average or is it very widely spread of what uses is such that kind of research?

(Refer Slide Time: 03:45)



Data sets gave us this ability to question random stuff like this just like we would build a random stuff with Play-Doh and then played around with his datasets and arrived at fantastic conclusions. So, now, what we are going to do is we are going to pick some random datasets for a discussion right now. I am going to go one by one go through some 8 to 10 datasets and explain what these datasets and what kind of question are can one ask and answer.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture – 15**  
**Handling Real-world Network Datasets**  
**Ingredients Network**

Let us now look at some networks as examples; I just told you that the advent of internet era has resulted in; we having a whole lot of plethora of datasets. Let me start discussing them one by one first and the most interesting dataset at least to me is the following.

(Refer Slide Time: 00:30)



Assume I prepare a dish and I look at all the ingredients in this dish; let say there are 5 items in this particular dish I am preparing.

(Refer Slide Time: 00:41)



And I take all possible ingredients in the world and I put a edge between two ingredients if they are part of a dish.

So, let us say there are some 200 ingredients and I start putting a link between two things if they are part of one dish and I keep doing this and I finally, observe how does the graph look like, maybe you think there is definitely some dish where any two ingredients are part of it. So, what I will do is I will look at only those cases which are popular.

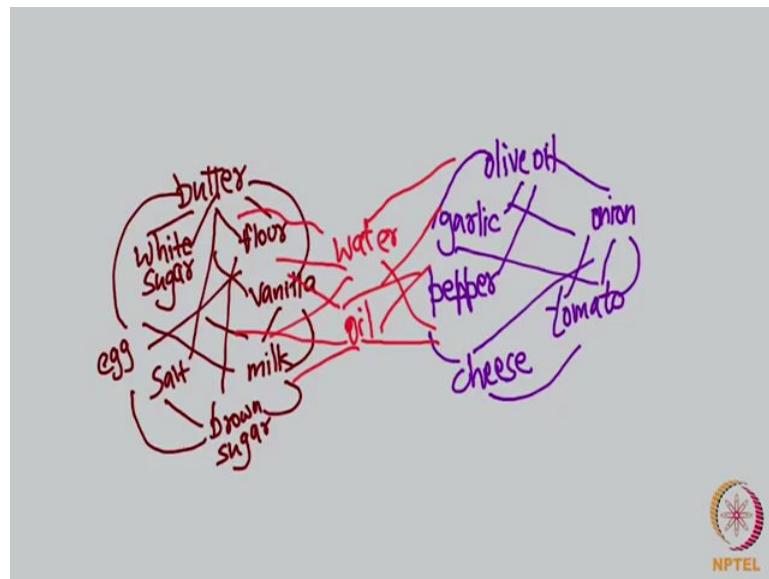
(Refer Slide Time: 01:20)



For example, there are dishes where salt and both are added many dishes one can name. So, I am going to put an edge between salt and sugar, but chili and chili powder and sugar probably there are not many dishes where you have both the both these things one exception could be chili chocolate, but one or two with one or two exceptions it is never used together.

So, there is no edge between chili powder and let say sugar.

(Refer Slide Time: 01:55)



If I look at the entire big network of all these ingredients it is a big network, now what do you expect? Anything interesting; anything at all that is interesting well you will observe that there will be strong communities in this network by communities I mean a bunch of people, bunch of ingredients that are mostly connected to each other and again a bunch of ingredients that are mostly connected to each other, but no connections across.

(Refer Slide Time: 02:27)



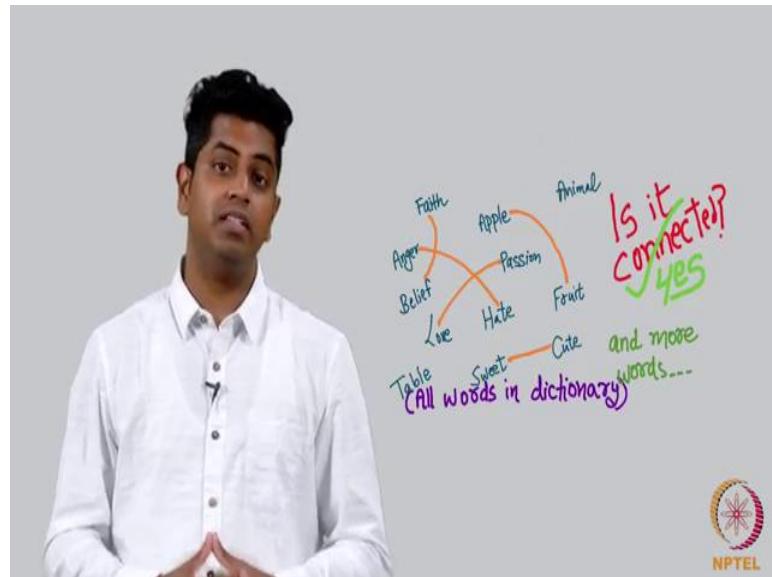
Now why even study this? It is plain interesting number one and secondly, may not be in this network in any other network such community structure has a lot of hints on the overall behavior of the system, so will see more such networks and more such properties of these networks in the forth coming examples.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture – 16**  
**Handling Real-world Network Datasets**  
**Synonymy Network**

I told you a sort of a weird example where one can see networks; I am going to tell you another example which is equally weird.

(Refer Slide Time: 00:18)



So, assume I take all words in English, words that are available from thesaurus and I put an edge between two words if they are synonyms so; obviously, a love and let say passion has an edge because they are synonyms. Let us say one can think of obtain such examples of let us say I say hate synonym for hate is bad, so and so forth.

Now, the same old question; once you have a network, we will ask the following question; is it connected? So, it is surprisingly true that this network is connected, at least it as a huge component; large component. What by mean by this? By this I mean you take all words in English thesaurus and you put an edge between two words, if one is a synonym of the other as you know if A is synonym of B then B is synonym of A; it is symmetric you draw the graph I am telling you its connected, so what? Something surprising happens if it is connected; think about it for a minute.

(Refer Slide Time: 01:31)



There is this word love there is this word hate they are not synonyms given that the graph is connected there has to be some word which love is synonym to and this words is synonym to some other words, so on and so on and finally, you will be getting the word hate. Graph theoretically speaking there should be a path from love to hate and you move synonymously, how do I mean by this? By this I mean do you see that if a is synonym of B and B is a synonym of C, A should be a synonym of C that is what it means, synonym means what these two mean the same if the word A means the same as B, if the word B means the same as C then the word A must mean the C same as C, but you observe that from love we can go to hate through synonyms which means love is synonyms to hate.

(Refer Slide Time: 02:28)



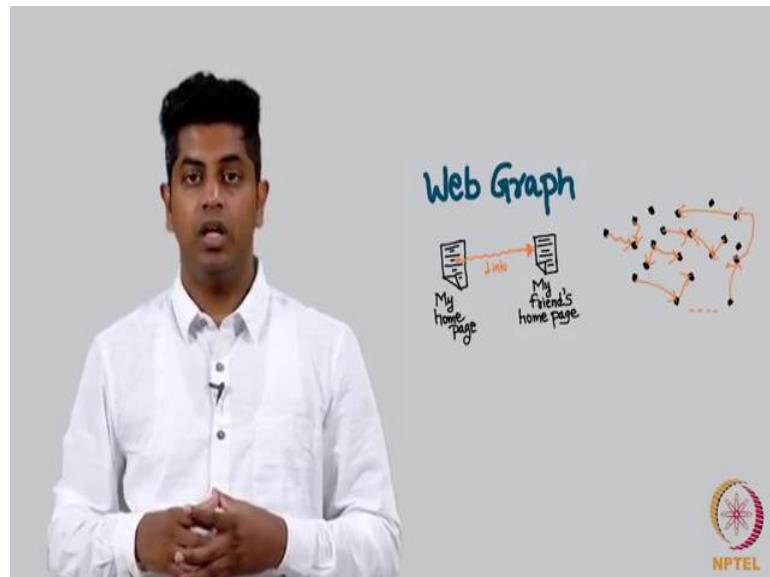
We can go from friend to let say enemy synonymously which means friend is synonyms to enemy I means the how weird is that; a graph theoretic analysis network theoretic analysis tells you of such weird retain English language not necessarily English language may be in many such many well known languages and what does this denote? It denotes that this synonym network synonym relation is not actually symmetric; if A is synonymous to B and B is synonymous to C, you cannot say A and C are actually synonymous, maybe there is a degradation of the synonyms as you proceed because from love you very beautifully get to hate, from enemy you very nicely get to friends synonymously. I just gave you an example of how a network theoretic analysis of question or an entity give some eyebrow raising results which are mostly generally is unexpected.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture – 17**  
**Handling Real-world Network Datasets**  
**Web Graph**

So, let us look at this seemingly useless example, I say useless because it is apparently useless, but it is extremely useful, I have been highlighting this fact from almost the first lecture. The example is this. It is called the network of the World Wide Web also called the web graph.

(Refer Slide Time: 00:24)



So, what I do is the following, I have my home page and I link the homepage of my friend on my homepage. I say for more details about these scores, please refer to my friend's website here and that here is a hyperlink to the other web page. We have discussed this before if you remember. So, now, if I look at the entire World Wide Web it is a bunch of pages; simply a bunch of pages, is it not. So, what I do is I go to a page, call it a node in my graph, in my graph every node corresponds to a page.

So, take 2 nodes means take 2 web pages, put an edge between them if one points to the other. So, my home page is a node, my friend's home page is another node, I put an edge from my page to my friend's home page saying A; my homepage is adjacent to B; my

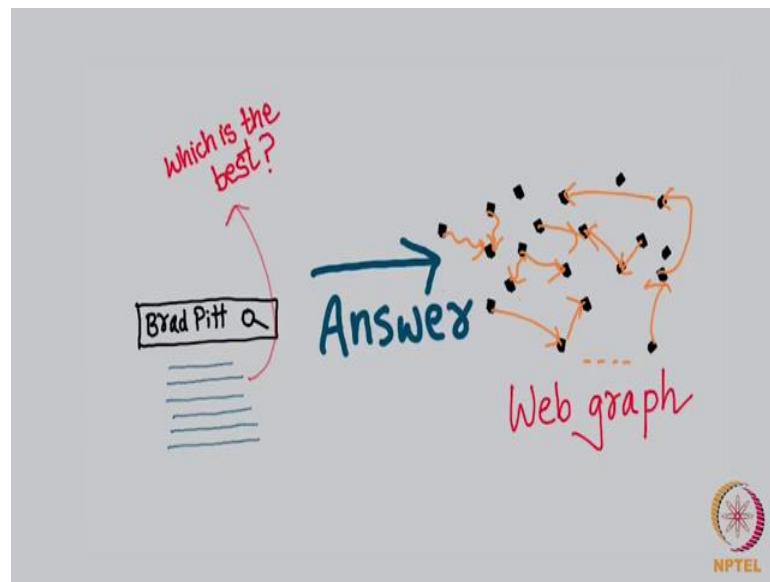
friend's home page and here is the edge. Assume I do this to all possible pages in the world. How does the graph look like? Very weirdly collecting such a graph looks like complete waste of time of what use would such a graph be it does not sound like it is going to be of any use, but as I told you people this resulted in a multibillion dollar industry which today we call google.

(Refer Slide Time: 01:42)



So, google basically harness the fact, but deep within this connections is the clue to the answer; answer to this question as to how can one get the best search result. If someone types a celebrity's name; let us say a Brad Pitt, it has a whole lot of their whole lot of pages on the internet which has the word Brad Pitt which is the best possible hit amongst these; the answer is in this web graph and google unlisted it.

(Refer Slide Time: 02:22)

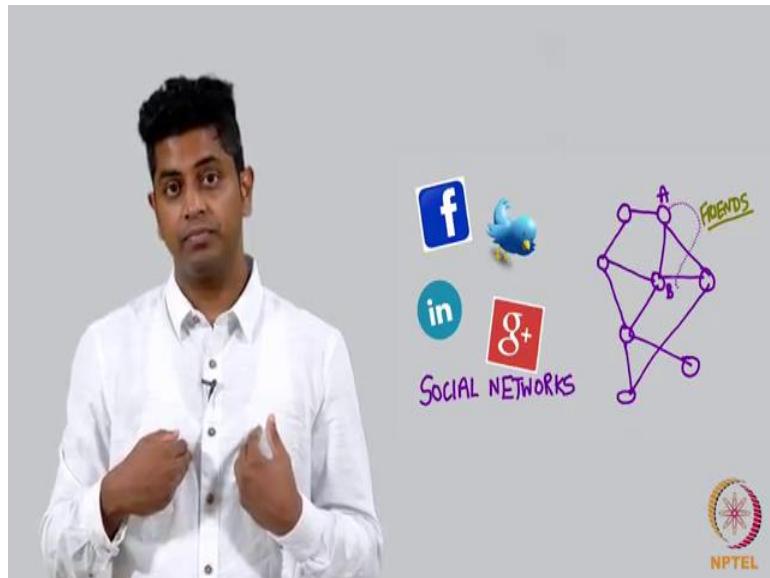


So, now here is an example of a network dataset called the web graph which is appearing of no use, but a very small process of mining, it gives you an answer to a huge question, but people did not realize is a that important in the 90s, but in the late 90s to Larry Page and (Refer Time: 02:50) came out with this beautiful algorithm with says this is where one can search for an answer for the question, how does one rank once search keywords once you type in a keyword whatever you get how do you rank the output of your search.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture – 18**  
**Handling Real-world Network Datasets**  
**Social Network Datasets**

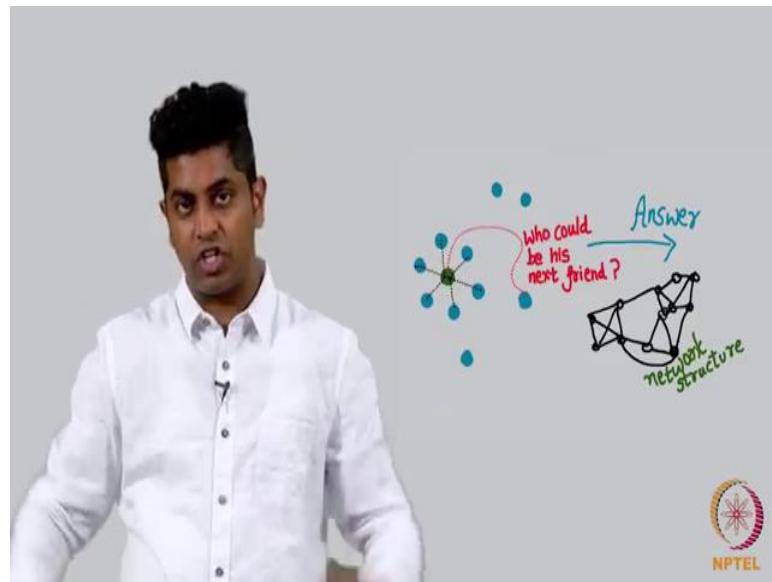
(Refer Slide Time: 00:14)



Let us look at literally a good social network that we all are familiar with. We all know a Facebook or let us say LinkedIn, Twitter, Google Plus, they are all what is called the social networks where a bunch of people get together and then the declared he is my friend he she is my friend and so on. If you look at such a network, what do you mean by such a network? By that I mean you put an edge between 2 nodes; nodes are people here, if one is in the friend list of the other I have you in my Facebook friend list, you have been in my Facebook friend list, you know its mutual and you put an edge between these 2 people and you look at the entire friendship network this way again appearing is useless dataset.

But then Facebook again made good use of this dataset by asking this question, they said fine here is a bunch of friends of this person who could be is next friend no, now here is a very important topics that requires some explanation.

(Refer Slide Time: 01:09)

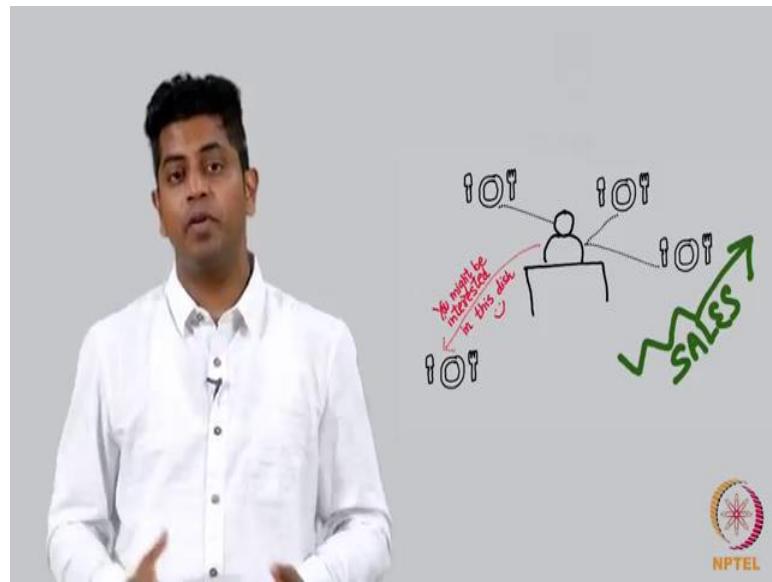


Let us take a pause and try to analyze this question properly, here is a friendship network of who is friends with whom and you would like to say I recommend Sudarshan a new friend who is this, out of let say 7 billion possible friends in this world for me I have some thousand friends on Facebook, anybody outside this thousand friends can be my prospective friend.

Now, how will you choose one most probable friend for me out of this big lot of people of what use is this to a social networking site like let say LinkedIn, Facebook, Google plus, all these sites work based on the numbers the more friends you have more busy you get glued to the monitor being on the social networking sites. So, they want to increase the number of friends they can only increase when they show you a prospective friend. This prospective friend let us say he is your long lost childhood friend or this prospective friend is a friend you find very interesting for whatever reasons how will Facebook know about it well the answer for this is hidden in the network structure there were few things that you must observe not just between you and your friends, but in the entire graph which will give you a hint as to who will be your most probable next friend.

If you show me the most probable next friend I will probably the friend that person makes friends with that person correct it is just like saying what dishes I eat and in a hotel in a restaurant and is an automated recommending system which says you may possibly be interested.

(Refer Slide Time: 03:00)

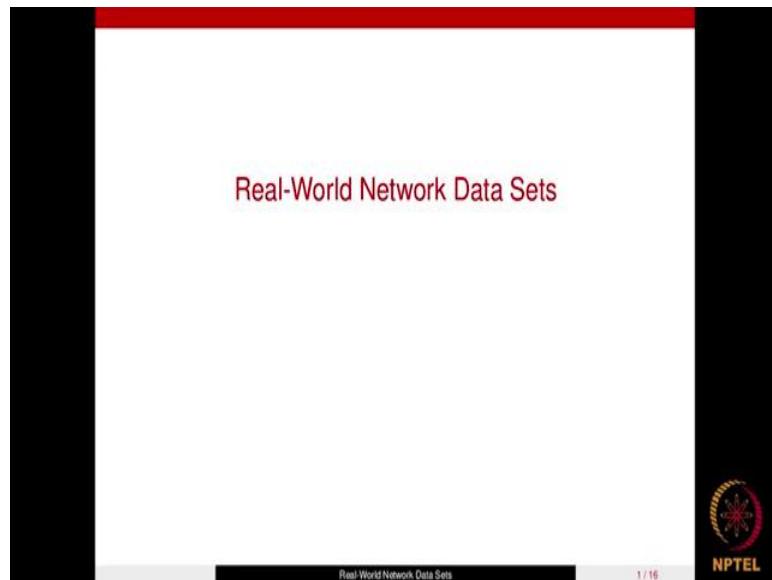


In this particular dish that will increase the sales of the restaurant if they are able to predict properly what do you generally like eating similarly who will be your next friend in case it is predicted can result in increasing your friend list not only will you will it increase your friend list it will also populate your friend list with good friends good whatever you define it as the most important friends whatever I define it as. So, this is this goes into the name link prediction and Facebook friend list is a great playground for one to investigate on who could be the next probable friend.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture – 19**  
**Handling Real-world Network Datasets**  
**Datasets: Different Formats**

(Refer Slide Time: 00:05)



Hi everyone! So, after having got introduced to the course and after having seen the various packages that you can use for the energy the social networks, I am sure you might be all geared up to learn the concept from social networks. I would say if you only learn the concepts from this course and do not apply them to nearby situations, you may not be able to get an enough understanding. So, it is important that whatever knowledge you required you apply them to real world scenarios and for that purpose you obviously, need a dataset from the real world situations.

Now, how do we get that data? Fortunately there are various individuals and organizations who have gathered network datasets for different situations and topics and the good thing is that they have kept this data online publicly available for anyone to access and analyze more over these datasets are available in different formats. So, in this video, we are first going to take a look at the formats into which the datasets are available and after that we are going to look at the sources from which we can download

these datasets and then in the subsequent videos we are going to analyze these datasets which we will download from the internet.

(Refer Slide Time: 01:31)

Examples of Networks

- Friendship Network
- Road Network
- Email Network
- Citation Network
- Collaboration / Co-authorship Network

Real-World Network Data Sets      2 / 16

NPTEL

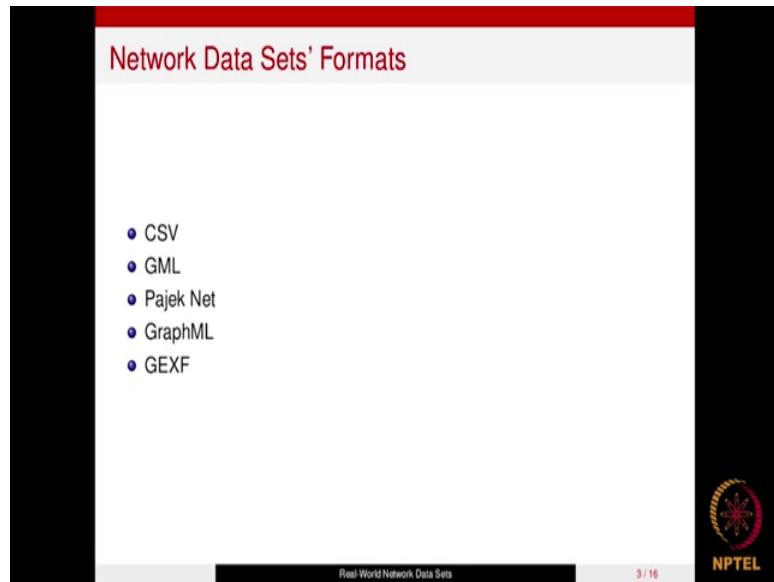
So, let us get started and before we get into the formats of these data sets I want you to take a look at the networks on which data is available of course, they are not limited we have data sets available on diverse range of topics, I am just going to give you a few examples of networks over here. So, we have friendship networks where the nodes are people and there will be a link from one node to the other node if these 2 nodes are if these 2 people are friends with each other similarly we have road networks where the nodes will be cities and there will be link from one node to the other node if these 2 cities are connected to a road.

So, usually these friendship networks and road networks are undirected networks and if you look at directed networks we have email network where the nodes will be the people and there will be a directed edge from node A to node B, if person A sent an email to person B, another example of a directed network is citation network. So, you might be knowing that in scientific scenario if a paper uses some information from another paper then this paper cites the second paper. So, that is what information is captured by a citation network. So, this is of course, directed network.

Another example of a network from scientific domain is collaboration network. So, scientists collaborate with each other for some research project or some scientific

development. So, this sort of information is captured by collaboration or co authorship network which captures which 2 people collaborated with each other or co authored for a people. So, these were just few examples of networks we are going to come across a number of them in on different topics. So, as he said these networks are available in different formats.

(Refer Slide Time: 03:27)



Let us take a look at these formats one by one. So, we have CSV format, we have GML format, we have Pajek format, we have GraphML format, we have GEXF format.

So, I am going to take these networks one by one, I am going to tell you the basic features of these formats and the basic structure that these that the files of these networks carry. So, apart from these formats you might also come across from two three more formats, but these 5 are the most commonly used formats when you download some data sets from internet that might most likely been one of these formats. So, we are going to take a look at these formats in detail will take them one by one.

(Refer Slide Time: 04:16)

CSV Format

CSV: Comma Separated Values  
Extension: \*.txt or \*.csv

- Edgelist
- Adjacency List (Adjlist)

Real-World Network Data Sets 4 / 16

NPTEL

Let us see; what is CSV format? So, we might be knowing that CSV stands for comma separated values this file will be in form having the extension either dot txt or dot csv.

Now, CSV format file can have 2 more types it could either be in Edgelist list format or it could be in adjacency list format. So, if you are from graph background you might know what an adjacency list means and what an Edgelist mean any anyway I am going to give an example of both these formats.

(Refer Slide Time: 04:54)

CSV Format- Edgelist

0	344
0	345
0	346
0	347
1	48
1	53
1	54
1	73
1	88
1	92

Real-World Network Data Sets 5 / 16

NPTEL

So, let us see what is an edge list format you can look at the simplicity of this file this is a snapshot from one of the files which is in CSV format which is in Edgelist type here every row contains two values first value is the source node and the second value is the target node. So, this indicates the row indicates that is going to be an edge from 0 to 344, 0 to 345, 0 to 346 and so on.

(Refer Slide Time: 05:38)

The screenshot shows a presentation slide with a red header bar containing the title 'CSV Format- Weighted Edgelist'. The main content area displays a list of edges with weights, each consisting of three values separated by spaces. The list includes:

- 1 2 0.3
- 3 4 0.1
- 5 7 0.2
- 4 7 0.8

At the bottom of the slide, there is a navigation bar with the text 'Real-World Network Data Sets' and '6 / 16'. On the right side, there is a logo for NPTEL.

So, basically the file will contain a list of the edges and in case you want to add some weight to these edges you can also do that for example, you want to add weights to these edges like this you can add third value to every row. So, for example, the edge which is going to go from 1 to 2 should have an 8.3. So, you can add it like this; however, there is one limitation that the edge list format carries and that you cannot add non numeric weights to the edges you cannot change the labels you cannot change any you cannot add an attribute to the nodes and edges. So, that is a limitation that this format carries. So, there is a trade. So, you can see that there is a trade off between the simplicity and the flexibility that a format provides.

(Refer Slide Time: 06:22)

The screenshot shows a presentation slide with a red header bar containing the text 'CSV Format- Adjlist'. The main content area displays a list of numbers arranged in rows, representing an adjacency list for a network. The numbers are:

```
1 2 5 7  
2 4 6  
3 1 4 7  
4 6 2 3  
5 7 2  
6 1  
7 2 8 4  
8 9  
9 1 7 4 8 |
```

At the bottom of the slide, there is a footer bar with the text 'Real-World Network Data Sets' on the left and '7 / 16' on the right. On the far right, there is the NPTEL logo.

Let us now look at the adjacency list format. So, this is a snapshot from a file which is in adjacency list format you see every row is having more than 2 values, now what is this indicate the first value in every row is basically a source node and the subsequent values in every row are the nodes which are adjacent to this source node. So, this first row indicate that there is going to be an edge from one to 2 this going to be an edge from 1 to 5 and 1 to 7 and if you go to the second row it indicates that there is going to be an edge from 2 to 4, 2 to 6, 3 to 1, 3 to 4 and 3 to 7 and so on.

So, basically every row tells all the nodes that are adjacent to a given node. So, this is the adjacency list format again you can see the simplicity that this format provides; however, you as I already told you this format does not give you much flexibility for example, you just cannot add any sort of attribute or label to the nodes or edges if we want to add some color attribute to the nodes or edges you cannot do that. So, it all depends on your requirement.

(Refer Slide Time: 07:38)

```
graph
[
  node
  [
    id A
  ]
  node
  [
    id B
  ]
  node
  [
    id C
  ]
  edge
  [
    source B
    target A
  ]
  edge
  [
    source C
    target A
  ]
]
```

Let us go to the next format the next format is GML format with stand for graph modeling language now this is one of the most commonly used format for network datasets the reasons include the flexibility that this format provides when it comes to labeling or assigning attributes to the nodes and edges the also this format is not very complex.

Let me explain you the format of this file. So, you start with the graph keyword and when you start the square bracket and inside that you have node keyword and when you again this square bracket and we write the keyword id and then you write the id of that node we close the node and so on you keep adding the nodes. So, if there are some n nodes in the network, they are going to be n nodes keyword once the nodes are done you start with the edge keyword. So, again you start the edge square bracket and then you write the source keyword and this is the source node the idea of the source node and then you write the target keyword and then the idea of the target node and we close it and so on.

So, if there are n edges there will be m edge keywords like this that is that the simplest format of a file which is in GML format in case you want to add some labels to these nodes and edges you can do that as well in this format.

(Refer Slide Time: 09:09)



**GML Format with Labels**

```
graph
  [
    node
    [
      {id A
       label "Node A"
      }
    ]
    node
    [
      {id B
       label "Node B"
      }
    ]
    node
    [
      {id C
       label "Node C"
      }
    ]
    edge
    [
      {source B
       target A
       label "Edge B to A"
      }
    ]
    edge
    [
      {source C
       target A
       label "Edge C to A"
      }
    ]
  ]
```

Real-World Network Data Sets      9 / 16

Let me show you an example for that. So, here you see we are adding a label to this node A. So, you can see this syntax you go inside the node keyword and there you write this keyword label and then you write the label of that node which is node a in this case similarly to write the label of the next node you go inside that in you write the label keyword and then you write the label that is node B and so on you do this for all the nodes individually and similarly you can do the labeling for the edges again you use the label keyword and then you write the label for the edge A which is edge B to A in this case and so on.

So, this is how to add labels and in case you want to add some attributes to the nodes and edges you can do that as well let me show you an example for that.

(Refer Slide Time: 10:00)

```
graph [ hierarchic 1
directed 1
node
[ id 0
graphics
[ x 200.0
y 0.0
]
]
node
[ id 1
graphics
[ x 425.0
y 75.0
]
]
edge
[ source 1
target 0
LabelGraphics
[
text "Happy New Year!"
]
]
```

Real-World Network Data Sets      10 / 16

NPTEL

Here you see this is a graph; this is the starting with graph and inside this, there are 2 attributes here which are the graph attribute. So, apart from nodes node attributes and edge attributes there are graph attributes as well in this case there are 2 attributes hierachic and directed and these are the values of these attributes to said the node attributes you go inside the node and when you start this attribute that you want to assign and then the name of that attribute and the value of that attribute inside the square brackets. So, this is the syntax. So, basically here they are assigning x and y positioning to the node.

Similarly, for the next node they are assigning x and y positions of the second node and here for adding an attribute to the edge again you start another attribute and start square brackets and then assign the value of that attribute now here at this point let me tell you one thing in case you are just making use of the data set download from the internet you might not need to go into the details whatever I am just now explaining. So, only in case you want to create your own datasets only in that case you should know these integrate details that I am telling you otherwise you should not worry about these details I just want to you to be aware of the syntax. So, that in case you come across some file should be able to mix sense sort of it you should be able to know what these a labels and keywords mean.

In other cases, you just want to download the files from internet, you might not need to go into all these details, let us go to the next format now.

(Refer Slide Time: 11:46)

```
*Vertices 82670
1 "entity"
2 "thing"
3 "anything"
4 "something"
5 "nothing"
6 "whole"

*arcs
4244 107
4244 238
4244 4292
4247 107
4248 1
4248 54
```

We have Pajek format which many people use for network datasets this kind of format has extension.net and let me tell you that in some cases you might also find networks with an extension.paj that also indicates a Pajek format network. So, it does not make any difference. So, you have either dot net or dot PNG format networks and the way the way of handling and the syntax is completely the same. So, let me show you the syntax of the file that is a structure of the file. So, you will start with the keyword star vertices and after that will be a number return which basically mean a number of vertices in the graph or the network after that you would have on every row you have these numbers one 2 up to n and after the number you would have some string written which basically indicates the label of that node.

So, you see that for every row you have every node return and once all the nodes are done you start with the information about the edges by writing with a star arcs or star edges this will basically come below it. So, every row contains the source node and the target node and so on.

(Refer Slide Time: 13:16)

The slide title is "Pajek Net Format". The content shows the basic syntax of a Pajek file:

```
*Vertices 9  
*Edges  
1 2  
1 9  
2 9  
2 3  
2 8  
3 8  
3 4  
4 5  
4 7  
5 7  
5 6  
6 4
```

At the bottom, there is a navigation bar with "Real-World Network Data Sets" and "12 / 16". The NPTEL logo is in the bottom right corner.

So, this is the basic syntax of a Pajek file and there is one more thing that I should tell you in case there is no labeling assign to the vertices of the network you just do not write anything below this vertices keywords because that does not make any sense you just write the there is no need to write the nodes consecutively one to n. So, you just leave that blank you just write the number of vertices that the graph contains and afterwards you start with the details on the edges and that is the same and in case you want to add some attribute to the edges you can do that as well.

(Refer Slide Time: 13:49)

The slide title is "Pajek Net Format with attributes". The content shows the syntax of a Pajek file with attributes:

```
*arcs  
4244 107 5
```

At the bottom, there is a navigation bar with "Real-World Network Data Sets" and "13 / 16". The NPTEL logo is in the bottom right corner.

Let me show you an example. So, here this is a source node and the target node and then you have value written this value basically is an attribute for this edge. So, in this case it is sort of a weight assign to this edge. So, that was about the Pajek format.

(Refer Slide Time: 14:08)

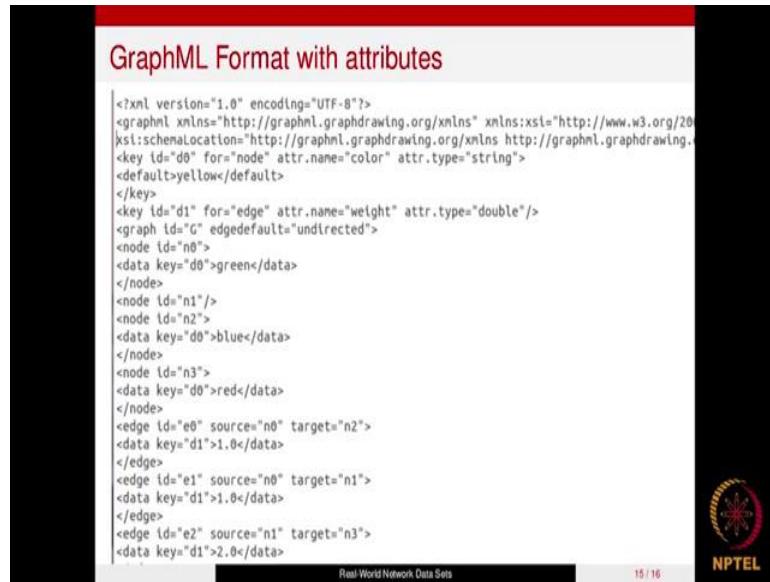
```
<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
  <graph id="G" edgedefault="undirected">
    <node id="n0"/>
    <node id="n1"/>
    <edge id="e1" source="n0" target="n1"/>
  </graph>
</graphml>
```

So, the next format is graph ml format ml here stands for xml basically you might be knowing what xml format is this basically provides the hierarchical structure and it makes use of various tags just like html and that is how you store the details in the form of tags. So, in a graph ml file you would have several tags let me quickly tell you about these tags. So, initially there will be an xml tag and after that you would have this graph ml tag and an inside the graph ml you would have a graph tag right and after the graph tag you would have number of node tags once the node tags are done you would have number of edge tags. So, this is the basic structure.

Now, let me get back here to the graph ml tag here you see some namespaces details basically you see some metadata you might be knowing the advantages of using namespaces and all that they allow different users to use different names and as long as I making use of different namespaces. So, I do not think you need to get into all that this is just a metadata about this network you can start from here you see a graph tag and then you have an attribute of graph which is edge default which is undirected and then you have node you have 2 nodes here and you have one edge here. So, this is a source and target keyword that we used inside the edge tag this is the basic structure of a graph ml

file you might see the structure you might feel that structure is little complex, but the flexibility that it provides in terms of assigning attributes to the nodes and edges is quite better than asking by 2 other formats.

(Refer Slide Time: 16:02)



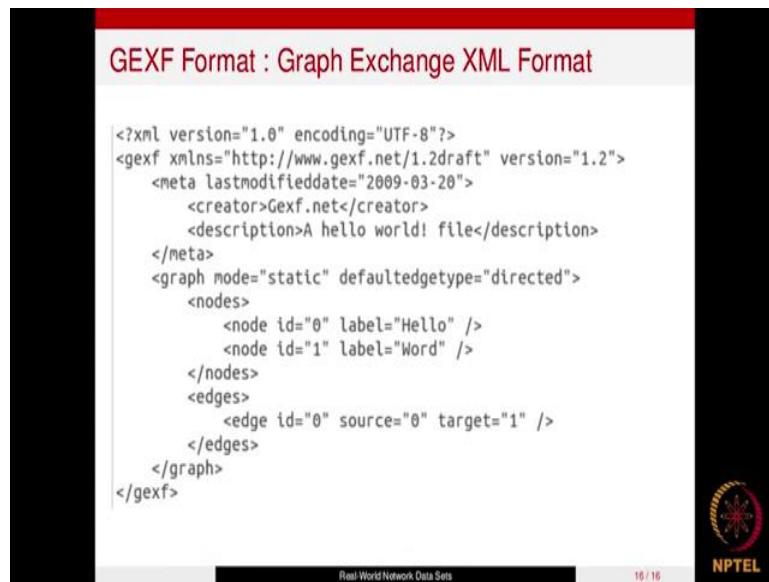
The screenshot shows a presentation slide with a red header bar containing the title "GraphML Format with attributes". Below the title is a block of XML code representing a graph structure. The XML includes declarations for namespaces and schema locations, definitions for keys (d0 for nodes, d1 for edges), and node and edge definitions with their respective attributes like color and weight. At the bottom of the slide, there is a footer bar with the text "Real-World Network Data Sets" and "15 / 16". On the right side of the slide, there is a logo for NPTEL.

```
<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmns" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://graphml.graphdrawing.org/xmns http://graphml.graphdrawing.org/xmns.xsd">
  <key id="d0" for="node" attr.name="color" attr.type="string">
    <default>yellow</default>
  </key>
  <key id="d1" for="edge" attr.name="weight" attr.type="double"/>
  <graph id="G" edgedefault="undirected">
    <node id="n0">
      <data key="d0">green</data>
    </node>
    <node id="n1"/>
    <node id="n2">
      <data key="d0">blue</data>
    </node>
    <node id="n3">
      <data key="d0">red</data>
    </node>
    <edge id="e0" source="n0" target="n2">
      <data key="d1">1.0</data>
    </edge>
    <edge id="e1" source="n0" target="n1">
      <data key="d1">1.0</data>
    </edge>
    <edge id="e2" source="n1" target="n3">
      <data key="d1">2.0</data>
    </edge>
  </graph>
</graphml>
```

So, let us look at one more example from graph ml where there assigning attributes to the nodes and edges. So, after you done with this GraphML tag you start a key tag this key tag for example, is for nodes and this is for assigning colors and after that you have another key tag which is for edges and that is for assigning weights and once you done with the key tags you start the graph tag as usual and when you want to assign an attribute to a node you start a data tag and inside this data tag you have you make use of this key that is d0 you are making use of this key and you are assigning green color to this node in the next one you are assigning blue color to this node and red color and so on.

Similarly, you are assigning a weight 1.0 to the edge by making use of this d1 key, you are assigning a weight 1.0 and so on. So, this is how you assign attributes to the nodes and edges. So, that was about the GraphML format.

(Refer Slide Time: 17:07)



```
<?xml version="1.0" encoding="UTF-8"?>
<gexf xmlns="http://www.gexf.net/1.2draft" version="1.2">
  <meta lastmodifieddate="2009-03-20">
    <creator>Gexf.net</creator>
    <description>A hello world! file</description>
  </meta>
  <graph mode="static" defaultedgetype="directed">
    <nodes>
      <node id="0" label="Hello" />
      <node id="1" label="Word" />
    </nodes>
    <edges>
      <edge id="0" source="0" target="1" />
    </edges>
  </graph>
</gexf>
```

Let us go on to the last format in our list which is GEXF format this stands for graph exchange xml format this format was basically created by Gephi people Gephi is an open source software which is used for visualizing and analyzing social networks. So, this format is again inspired from xml as you can see it is composed of many xml tags. So, you start with an xml tag and we start with GEXF tag and inside that you have this meta tag to store the meta data about the network and then you start the graph tag inside the graph tag you have nodes tag and inside the nodes tag you have all the nodes and then you start the edges tag inside the edges tag you have the edge tag.

So, the format is little similar if it actually quite similar to graph ml format the previous one this is a little cleaner in terms of assigning various attributes to the nodes and edges. So, there is one thing that I would like to point out here. So, I want to say that do not worry if you not understanding anything out of this video because the aim of this video was just to introduce to you the main formats into which the datasets are available and in case you are not creating your own dataset you might not able to me to you know get into also integrate details in case you are just using the datasets downloaded from the internet you might not need to yourself write all these details that I just explain to you.

So, the aim was just for you to know you now get introduced to these statements and in case you get file you would be able to make some sense out of it that was your only aim. So, do not worry if you do not understand these integrate details.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture – 20**  
**Handling Real-world Network Datasets**  
**Datasets: How to Download?**

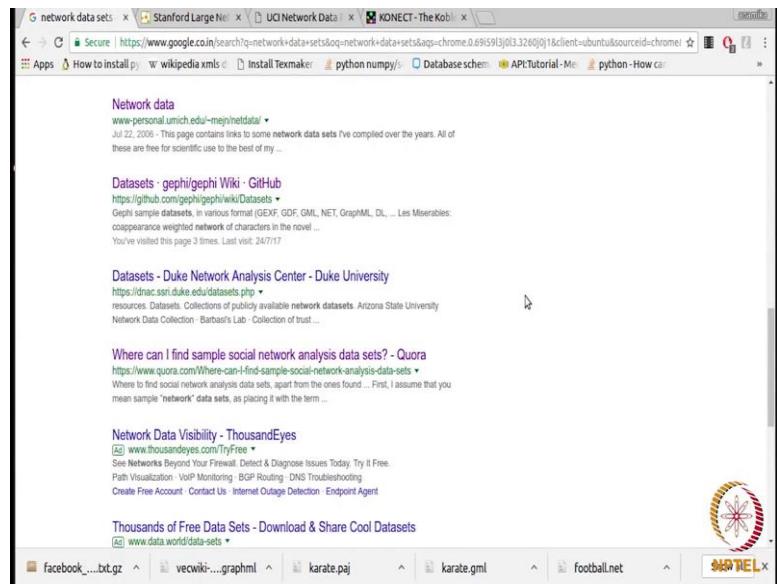
(Refer Slide Time: 00:05)



NPTEL

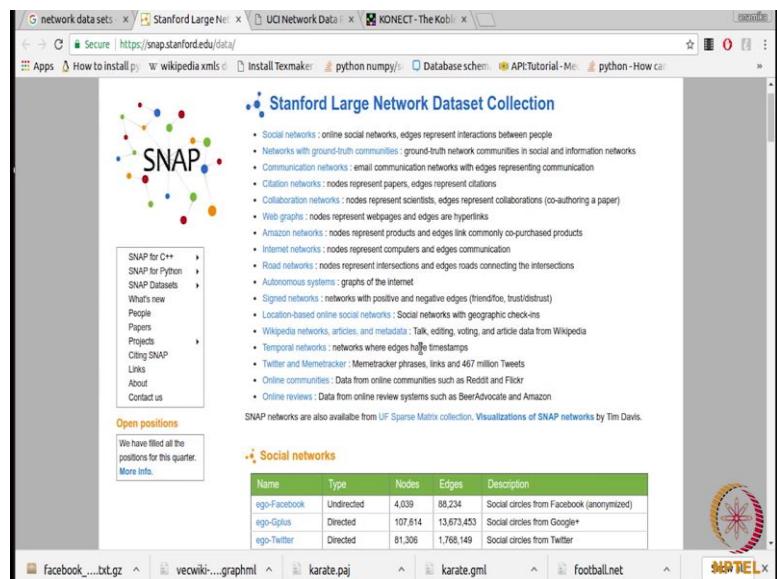
There is one more point to note here, networkx provides various functions through which we can read a network in one format and we can write the network in another format for example, we can read a network in Gmail format and we can write the network in GEXF format. So, those functions available and we can make use of them now let see how we can download these networks in different formats.

(Refer Slide Time: 00:34)



So, when we google for network dataset let us see which options do we get. So, here we get repository wise Stanford university which is called snap dataset let me open this and you also get a repository by UCI that is maintained by university of California and we also get this connect repository which is maintained by Koblenz university as you go down you get a number of more resources.

(Refer Slide Time: 01:05)



Let me show you this one first. So, snap dataset repository is the most commonly used repository for accessing the network data sets. So, here you get a number of networks of

different types for example, you get social network like you like the networks on Facebook, Twitter, Slashdot, Google plus and you also get networks which have ground truth community what do you mean by that you know there are various communities of nodes in the networks. So, if there is a community detection algorithm you will come across these algorithms in the next videos. So, if you want to check whether a community detection algorithm is working fine or not, we might need to verify it.

So, these are the networks where the community details are given as ground truth values. So, they are useful in such scenarios then we have communication networks for example, networks e for example, email networks and we have citation networks and collaboration networks we already discussed about these networks and then we have web graphs we have Amazon networks, we have road networks and we have Wikipedia networks as in the networks between the articles and we have temporal networks now temporal networks are the ones which also contain the temporal information of the events. So, timing details will also be there, and we also have networks on stack over flow here. So, you see there are so many networks available over here and then the basic details you can see the number of nodes number of edges etcetera.

(Refer Slide Time: 02:45)

Name	Type	Nodes	Edges	Description
sx-supuser	Directed, Temporal	194,085	1,443,339	924,866 Comments, questions, and answers on Super User
sx-askubuntu	Directed, Temporal	159,316	964,437	596,933 Comments, questions, and answers on Ask Ubuntu
wiki-talk-temporal	Directed, Temporal	1,140,149	7,833,140	3,309,592 Users editing talk pages on Wikipedia
email-Eu-core-temporal	Directed, Temporal	986	332,334	24,929 E-mails between users at a research institution
CollegeMsg	Directed, Temporal	1,899	20,298	59,833 Messages on a Facebook-like platform at UC-Irvine

Name	Type	Number of items	Description
twitter7	Tweets	47,869,982 users	A collection of 476 million tweets collected between June-Dec 2009
memetracker9	Memes	96 million	418 million links Memetracker phrases and hyperlinks between 96 million blog posts from Aug 2008 to Apr 2009
ksc-time-series	Time Series	2,000	Time series of volume of 1,000 most popular Memetracker phrases and 1,000 most popular Twitter hashtags
higgs-twitter	Tweets	456,631	Spreading processes of the announcement of the discovery of a new particle with the features of the Higgs boson on 4 July 2012.

Name	Type	Number of items	Description
RedditPizzaRequests	Reddit requests	5,671 submissions	Textual requests for pizza with outcome labels
Reddit	Reddit	132,308 submissions	Resubmitted content on reddit.com

(Refer Slide Time: 02:52)

The screenshot shows a web browser window with several tabs open. The active tab displays network statistics for the Facebook dataset. Key statistics include:

Nodes	4039
Edges	88234
Nodes in largest WCC	4039 (1.000)
Edges in largest WCC	88234 (1.000)
Nodes in largest SCC	4039 (1.000)
Edges in largest SCC	88234 (1.000)
Average clustering coefficient	0.6055
Number of triangles	1612010
Fraction of closed triangles	0.2647
Diameter (longest shortest path)	8
90-percentile effective diameter	4.7

Note: These statistics were compiled by combining the ego-networks, including the ego nodes themselves (along with an edge to each of their friends).

**Source (citation)**

J. McAuley and J. Leskovec, Learning to Discover Social Circles in Ego Networks. NIPS, 2012.

**Files**

File	Description
facebook.tar.gz	Facebook data (10 networks, anonymized)
facebook_combined.txt.gz	Edges from all egos combined
readme-Ego.txt	Description of files

Now, let me show you how we can download one of the networks our; of these network. So, let me let me first open this and see the details. So, here you can read the details about the network and then you have basic statistics about the network and as you go down you have different files regarding the network. So, here we see 3 files one is readme file and the first file is having ten networks, the second file is having edges from all the networks you know combined. So, let me download this second file.

(Refer Slide Time: 03:31)

A 'Save File' dialog box is displayed over the browser window. The file name is set to 'facebook\_combined.txt.gz'. The 'Save in Folder' dropdown shows 'anamika' and 'NPTEL'. The 'Places' sidebar lists 'Recently Used' locations like 'anamika', 'Desktop', 'Documents', 'Downloads', etc. The main area shows a list of files in the 'Data Sets New' folder, with 'facebook\_combined.txt.gz' highlighted. At the bottom right of the dialog are 'Cancel' and 'Save' buttons.

So, let me save this file and we will open it later to see its structure. So, this is how you can download the networks.

(Refer Slide Time: 03:43)

The screenshot shows a web browser window with the URL <https://networkdata.cs.cmu.edu/resources.php>. The page title is "Network Data Repository". It features a large circular network graph visualization. Below the title, there's a brief introduction: "The UCI Network Data Repository is an effort to facilitate the scientific study of networks. Feel free to browse and download the currently available datasets. For more information about networks and the terms used to describe the datasets, click Getting Started." A section titled "Additional Public Datasets" lists several categories of datasets:

- Collections of classic network datasets commonly used in social network analysis research
  - UCINET dataset collection
  - Pajek dataset collection (and in Matlab format)
- Dataset directories curated by research groups and organizations
  - Datasets from the CASOS Project
  - Datasets from the Leland Stanford Junior University biological networks from the Link Group
  - Datasets from Jim Lenković's work (SNAP)
  - Web graph datasets from the Laboratory for Web Algorithmics
  - Arlington dataset of a trust network in a online web community
  - Datasets and repositories listed on Gephi community wiki
  - Several datasets involving longitudinal network data
  - Infovis Cyberinfrastructure
  - Columbia's Collective Dynamics Group
  - Barabási's CNR
- Dataset directories curated by individuals
  - Network Datasets from Alex Arenas
  - Ivan Havel
  - Mark Newman
  - Rodan Pozo
  - Christian Sommer

At the bottom of the page, there are links to download files like "facebook\_....txt.gz", "facebook\_....bit.gz", "vecwiki-...graphml", "karate.paj", and "karate.gml".

Let me show another repository this is maintained by university of California and they are also you see their in a given additional public datasets you can you can explore all these lines and get different kinds of network since we also have to download Pajek network and click this link. So, here we have all the Pajek datasets.

(Refer Slide Time: 04:02)

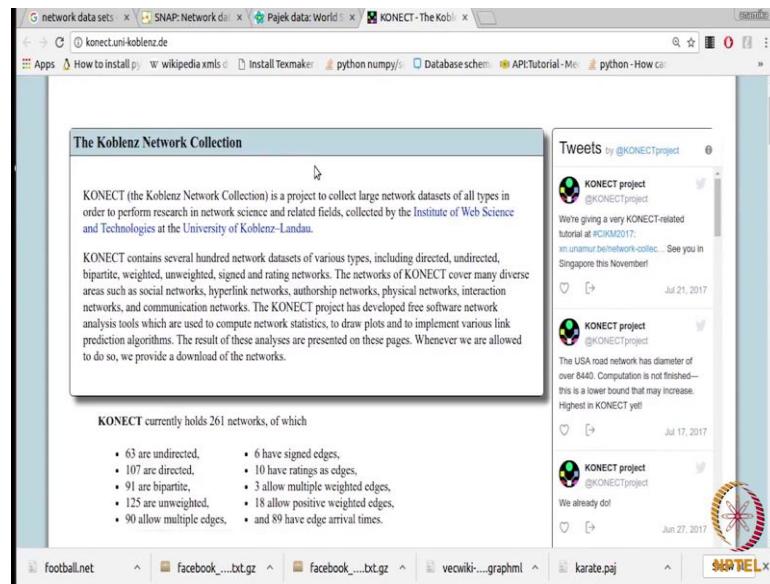
The screenshot shows a web browser window with the URL [vlado.fmf.uni-lj.si/pub/networks/data/](http://vlado.fmf.uni-lj.si/pub/networks/data/). The page title is "Pajek datasets". It displays a table titled "Networks:" with the following data:

network	n	m <sub>E</sub>	m <sub>A</sub>	type
Yeast	2361	0	7182	biology, protein interactions
Tina	11	0	29-48	sociology, (6 relations), measurements
Football	35	0	118	sport, valued
Slovene parties 1994	10	0	90	sociology, valued signed
US presidents	?	0	?	genealogy
Turkish nomads	?	0	?	genealogy
CS phd	1882	?	0	genealogy
US Air lines	332	0	?	transport
Cities and services	101/55	0	?	valued, 2-mode data
Divorce in US	59/50	0	?	binary, 2-mode data
Dutch Elite 2006	3810+937	5221	0	multirelational, 2-mode data
Graph products	674/314	0	?	collaboration, 2-mode
Slovenian municipalities				valued, derived

At the bottom of the page, there are links to download files like "facebook\_....txt.gz", "facebook\_....bit.gz", "vecwiki-...graphml", "karate.paj", and "karate.gml".

Let me download the small network just to show you the kind of structure that Pajek file contains. So, I am saving this I am sorry let me first see this. So, we have to save this file. So, this is a dot net file I told you previously that the Pajek files will be available in one of the 2 formats either it will be dot net, or it will be dot paj. So, this one is dot net a very small network just to show you the structure.

(Refer Slide Time: 04:43)



Let us go to the next one. So, here you see this repository again is having a lot of networks on different topics you see Wikipedia site you like and there are air traffic control there are different topics and you can just choose based on your requirement there is twitter dataset. So, you can download the data set based on the resource that you want to access or you can download dataset based on the format that you want for example, I want the data from say UCI.

(Refer Slide Time: 05:24)

The screenshot shows a web browser window with multiple tabs open. The active tab displays the 'Zachary's Karate Club' dataset from the Network Data Repository. The page includes a network graph visualization, basic description, summary, original website, citation, related literature, and a 'View files' button.

**Basic Description:**  
34 nodes  
3 edges  
Undirected  
Static  
Unweighted

**Summary:**  
Social network of friendships between 34 members of a karate club at a US university in the 1970s.

**Original website:**  
<http://www-personal.umich.edu/~mejn/netdata/>

**Citation:**  
W. W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research* 33, 452-473 (1977).

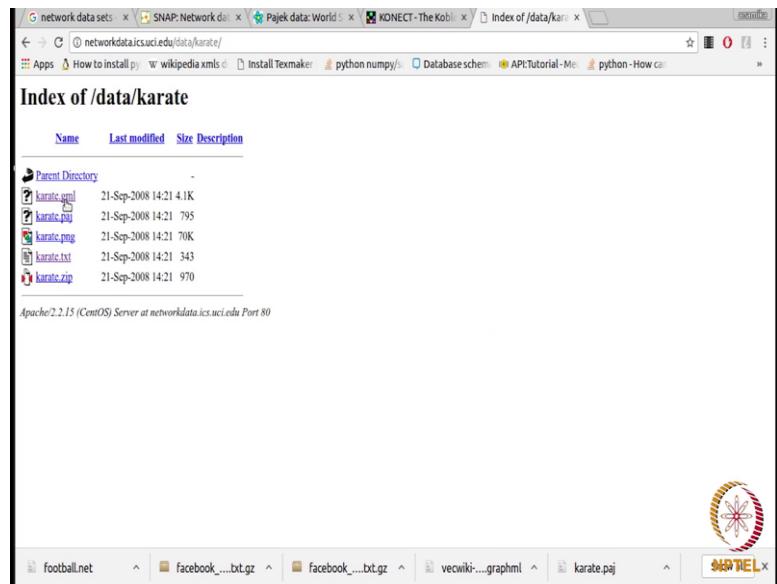
**Related Literature:**  
We hope to maintain a list of papers that use or reference this data set, hosted at CiteULike.com. If you know of others, feel free to add to the list.

Suggest Edits

So, I click this and here you have all the network from this repository there is an interesting network here which is the Zachary Karate, let me download this. So, there is an interesting history attached to this network there was a karate club in us university and it had 34 participants, there was a fight that happened between 2 important people of that club and they were the instructor and the administrator.

So, after a period of 3 year the due to this fight the network got divided into 2 communities Zachary was a person who analyze this network over a over this period and he Indian basically predicted the communities that are going to form. So, this network is well known and it is called the Zachary's karate club network and it is very small as well its use for the community detection algorithms as well and it is also a nice starting step for you to start your analysis on the networks. So, we are going to download this network.

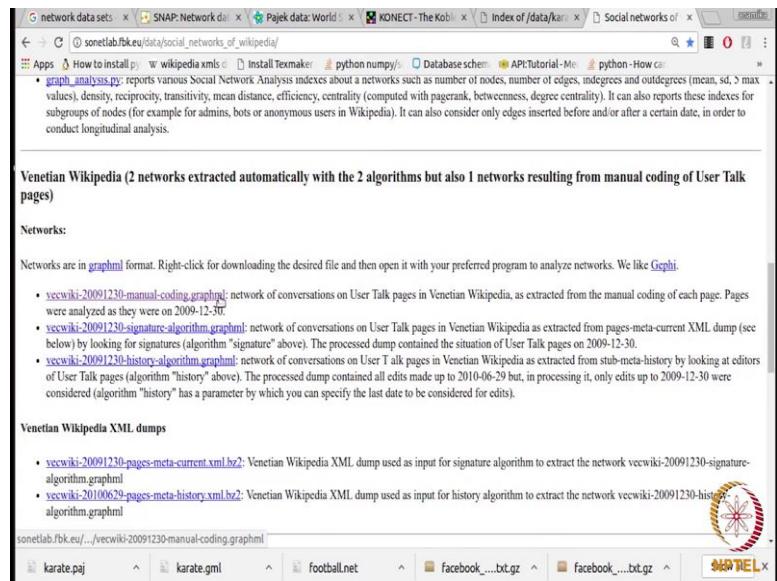
(Refer Slide Time: 06:35)



This network is available in GML format as well as Pajek format. So, let me download both of them and then we have this Pajek. So, we have done with Ajelius format and dot net format and dot gml format and dot Pajek format.

So, we are left with GraphML. So, let me download one network in that format I know one resource a sonnet is a repository that contains data in graph ml format.

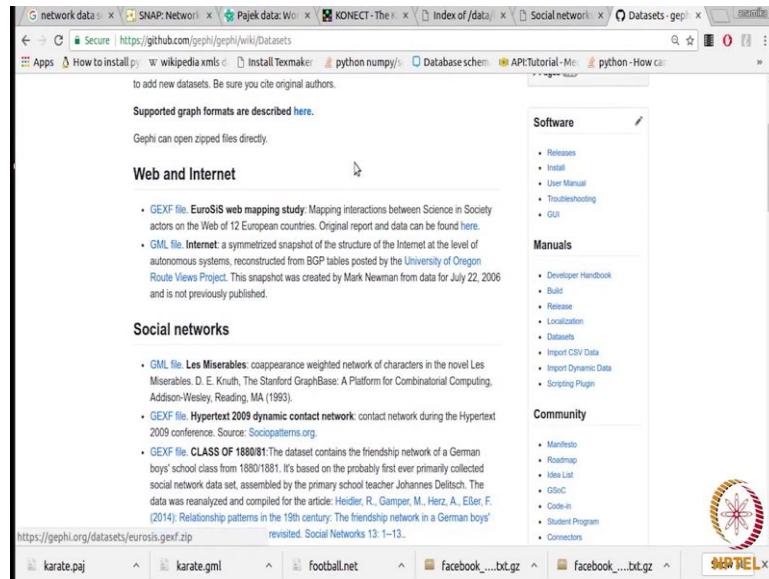
(Refer Slide Time: 07:12)



So, let me just open that only here these are the Wikipedia network which are in graph ml format I had already downloaded this once. So, I will access that only. So, you can

download all any of these networks and you can basically read the details about the network and then accordingly download if it chooses your purpose.

(Refer Slide Time: 07:47)



Now, let see GEXF format let me show you how we can download that let me open this link. So, here you can see various networks in GML as well as GEXF format. So, you can download any of these after reading the details I have already downloaded one in GEXF files. So, I will show you that only.

(Refer Slide Time: 08:05)

```

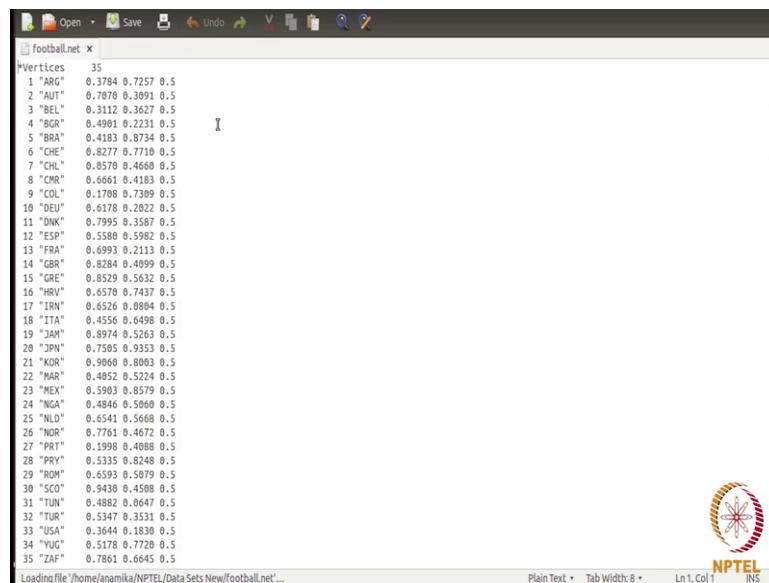
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
0.10
0.11
0.12
0.13
0.14
0.15
0.16
0.17
0.18
0.19
0.20
0.21
0.22
0.23
0.24
0.25
0.26
0.27
0.28
0.29
0.30
0.31
0.32
0.33
0.34
0.35
0.36

```

So, let us look at what we have downloaded we have all these networks let me extract this first. So, here we have six network GEXF format txt format dot net format dot gml format dot Pajek format and dot graph ml format. So, I am quickly going to show you the structure of these files although I had already introduce the formats to you let me first open the txt format which is in edge list format.

So, you can see this simplicity of the format again you just have 2 things in every row and these 2 things are the source and the target of the edges. So, there will be a link from 0 to 1, 0 to 2, basically this is undirected. So, there will be an edge between 0 and 3, 0 and 4 and so on. So, this is the edge list format.

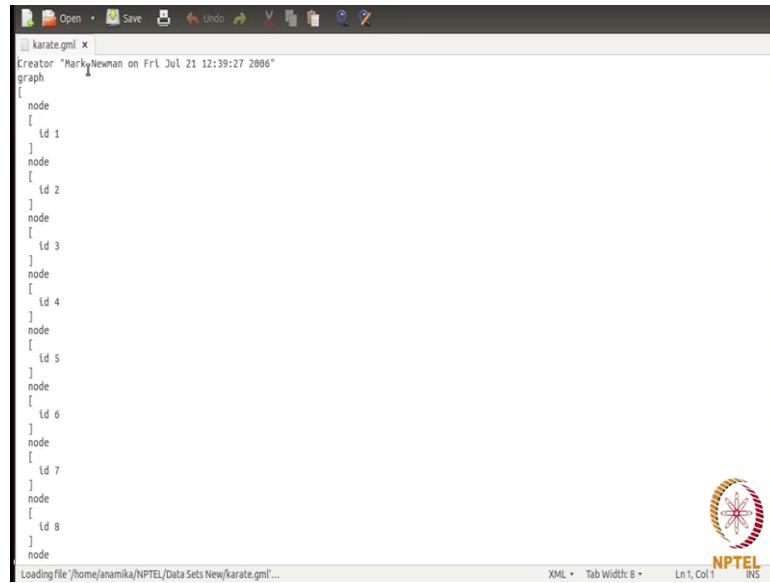
(Refer Slide Time: 08:59)



	Vertices	35
1	"ARG"	0.3784 0.7257 0.5
2	"AUT"	0.7070 0.3091 0.5
3	"BEL"	0.3112 0.3627 0.5
4	"BGR"	0.4981 0.2231 0.5
5	"BRA"	0.4183 0.8734 0.5
6	"CHE"	0.8277 0.7718 0.5
7	"CHL"	0.6570 0.4669 0.5
8	"CND"	0.6661 0.4183 0.5
9	"COL"	0.1708 0.7309 0.5
10	"DEU"	0.6178 0.2022 0.5
11	"DNK"	0.7995 0.3587 0.5
12	"ESP"	0.5580 0.5982 0.5
13	"FRA"	0.6993 0.2113 0.5
14	"GBR"	0.8284 0.4099 0.5
15	"GRE"	0.8520 0.5632 0.5
16	"IRN"	0.6570 0.7437 0.5
17	"IRL"	0.6526 0.0804 0.5
18	"ITA"	0.4556 0.6498 0.5
19	"JAH"	0.8974 0.5263 0.5
20	"JPV"	0.7582 0.9353 0.5
21	"KOR"	0.9060 0.8003 0.5
22	"MAR"	0.4852 0.5224 0.5
23	"MEX"	0.5903 0.8579 0.5
24	"NGA"	0.4384 0.5968 0.5
25	"NLD"	0.6541 0.5668 0.5
26	"NOR"	0.7761 0.4672 0.5
27	"PRT"	0.1990 0.4988 0.5
28	"PRV"	0.5335 0.8248 0.5
29	"ROM"	0.6593 0.5979 0.5
30	"SCO"	0.9430 0.4508 0.5
31	"TUN"	0.4882 0.0647 0.5
32	"TUR"	0.5347 0.3531 0.5
33	"USA"	0.3644 0.1830 0.5
34	"YUG"	0.5178 0.7720 0.5
35	"ZAF"	0.7861 0.6645 0.5

Next let us check the dot net format. So, as I told you this starts with the key words star vertices and then you have the number of vertices in that network. So, we these are the ids and these are the labels of the vertices and then we have 3 details attached to every vertex which are the attributes and you can basically see the documentation to see what these attributes mean then after the nodes are done you have these arcs which are basically the edges and for every edge you have this attributes attached.

(Refer Slide Time: 09:40)

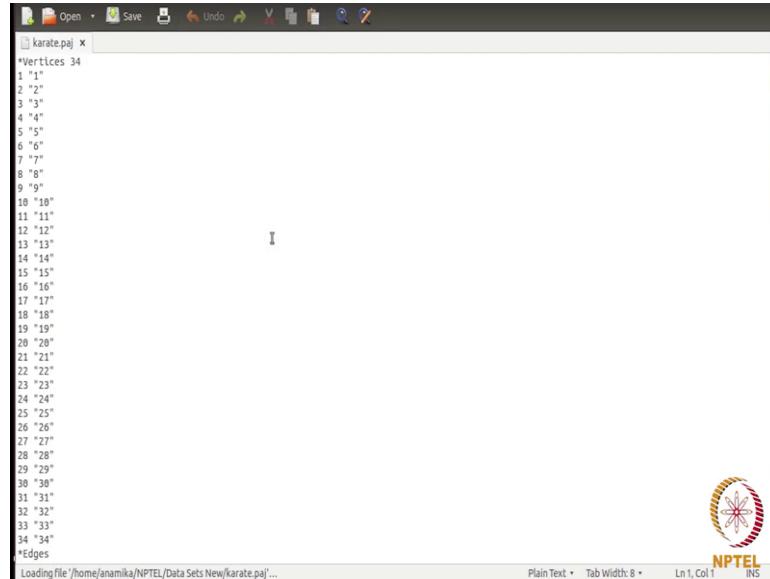


A screenshot of a text editor window titled "karate.gml x". The content of the file is a GML graph definition. It starts with "graph" followed by a series of "node" blocks, each containing an "id" from 1 to 8. After the nodes, there is a closing "graph" block. The file was created by "Mark Newman" on "Fri Jul 21 12:39:27 2006". The status bar at the bottom shows "Loading file '/home/anamika/NPTEL/Data Sets New/karate.gml'..." and "Ln 1, Col 1". The NPTEL logo is visible in the bottom right corner of the window.

```
graph
  node
  [
    id 1
  ]
  node
  [
    id 2
  ]
  node
  [
    id 3
  ]
  node
  [
    id 4
  ]
  node
  [
    id 5
  ]
  node
  [
    id 6
  ]
  node
  [
    id 7
  ]
  node
  [
    id 8
  ]
  node
```

So, this is an example of dot net file and then we have GML. So, you see there is a graph keyword and then you have square brackets and then you have all the nodes. So, first all the nodes will be there and then once the nodes are done you have these edge details. So, this is a GMail format. So, this is Zachary karate network which had thirty-four nodes.

(Refer Slide Time: 10:05)



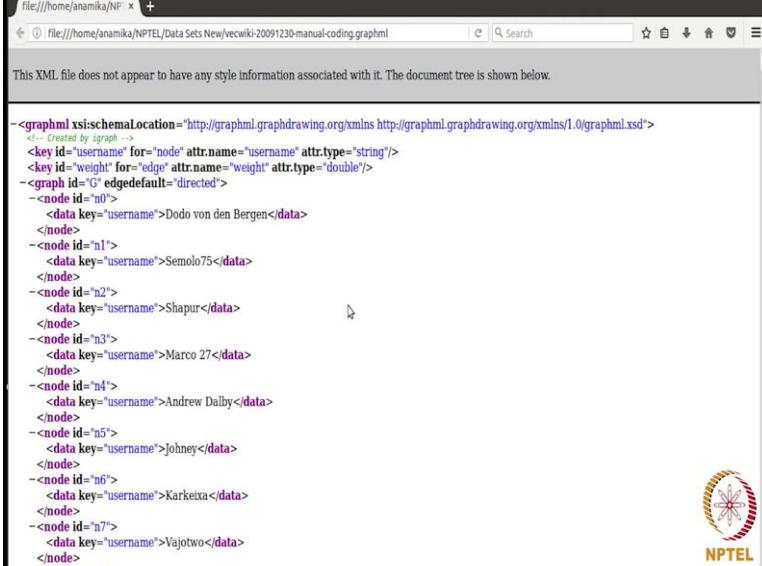
A screenshot of a text editor window titled "karate.paj x". The content of the file is a Pajek format network definition. It starts with "\*Vertices 34" followed by a list of 34 vertex IDs from 1 to 34. Below this, there is a section for edges, indicated by "\*Edges". The status bar at the bottom shows "Plain Text • Tab Width: 8 • Ln 1, Col 1" and "Loading file '/home/anamika/NPTEL/Data Sets New/karate.paj'...". The NPTEL logo is visible in the bottom right corner.

```
*Vertices 34
1 "1"
2 "2"
3 "3"
4 "4"
5 "5"
6 "6"
7 "7"
8 "8"
9 "9"
10 "10"
11 "11"
12 "12"
13 "13"
14 "14"
15 "15"
16 "16"
17 "17"
18 "18"
19 "19"
20 "20"
21 "21"
22 "22"
23 "23"
24 "24"
25 "25"
26 "26"
27 "27"
28 "28"
29 "29"
30 "30"
31 "31"
32 "32"
33 "33"
34 "34"
*Edges
```

Now, this is a karate network in Pajek format, let me show you that here there is no attribute for the vertices and there is no attribute for the edges as well. So, this is again a very simple network in Pajek. Now let me show you graph ml format. So, this is an

example of a GraphML network. So, it has the number of tags the first tag is GraphML xl and after that we have 2 key tags. I told you that key tags for adding the attributes to nodes and edges.

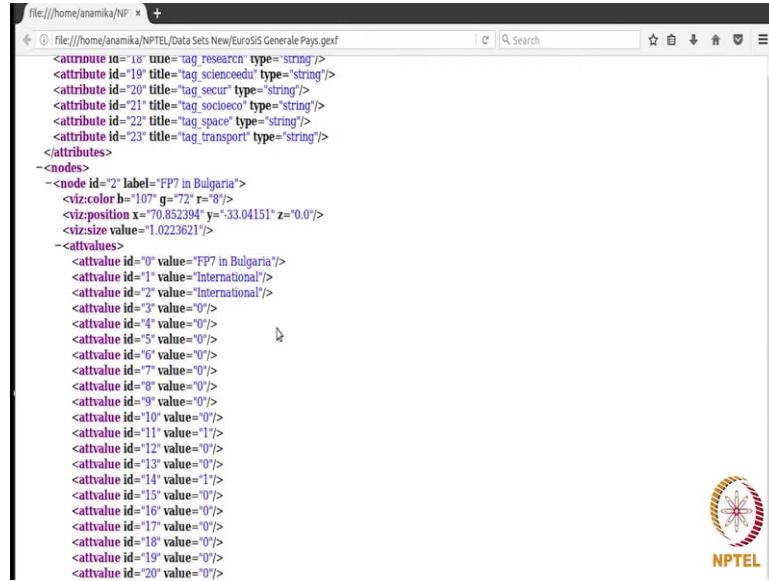
(Refer Slide Time: 10:26)



```
<graphml xmlns="http://graphml.graphdrawing.org/xmlns http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
  <!-- Created by iGraph -->
  <key id="username" for="node" attr.name="username" attr.type="string"/>
  <key id="weight" for="edge" attr.name="weight" attr.type="double"/>
  <graph id="G" edgedefault="directed">
    <node id="n0">
      <data key="username">Dodo von den Bergen</data>
    </node>
    <node id="n1">
      <data key="username">Semolo75</data>
    </node>
    <node id="n2">
      <data key="username">Shapur</data>
    </node>
    <node id="n3">
      <data key="username">Marco 27</data>
    </node>
    <node id="n4">
      <data key="username">Andrew Dalby</data>
    </node>
    <node id="n5">
      <data key="username">Johnney</data>
    </node>
    <node id="n6">
      <data key="username">Karkeixa</data>
    </node>
    <node id="n7">
      <data key="username">Vajotwo</data>
    </node>
    <node id="n8">
```

So, first one is for nodes and the second one is for edges and then we start the graph tag and inside graph we have a number of node tags and the nodes are given attributes the key using the you know data tag and we are making use of this key and as you go down after the nodes are done you can go down gap after the nodes are done you have the edge stacks. So, again the edges are given attributes using the data tag.

(Refer Slide Time: 11:21)

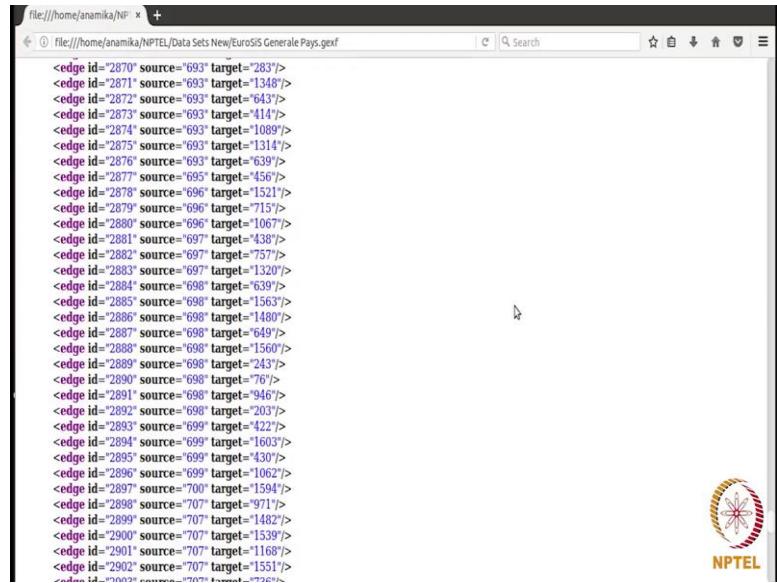


```
<file:///home/anamika/NPTEL/Data Sets New/EuroSIS Generale Pays.gexf>
<graph LR>
<attribute id="18" title="tag_research" type="string"/>
<attribute id="19" title="tag_scienceedu" type="string"/>
<attribute id="20" title="tag_secur" type="string"/>
<attribute id="21" title="tag_socieco" type="string"/>
<attribute id="22" title="tag_space" type="string"/>
<attribute id="23" title="tag_transport" type="string"/>
</attributes>
<nodes>
<node id="2" label="FP7 in Bulgaria">
<viz:color b="107" g="72" r="8"/>
<viz:position x="70.852394" y="-33.04151" z="0.0"/>
<viz:size value="1.0223621"/>
<attvalues>
<attvalue id="0" value="FP7 in Bulgaria"/>
<attvalue id="1" value="International"/>
<attvalue id="2" value="International"/>
<attvalue id="3" value="0"/>
<attvalue id="4" value="0"/>
<attvalue id="5" value="0"/>
<attvalue id="6" value="0"/>
<attvalue id="7" value="0"/>
<attvalue id="8" value="0"/>
<attvalue id="9" value="0"/>
<attvalue id="10" value="0"/>
<attvalue id="11" value="1"/>
<attvalue id="12" value="0"/>
<attvalue id="13" value="0"/>
<attvalue id="14" value="1"/>
<attvalue id="15" value="0"/>
<attvalue id="16" value="0"/>
<attvalue id="17" value="0"/>
<attvalue id="18" value="0"/>
<attvalue id="19" value="0"/>
<attvalue id="20" value="0"/>
</attvalues>

```

Now, let us check the GEXF format. So, here you see there are again because this is also based on xml there is GEXF tag inside that we have graph tag and there are. So, many attributes for this graph and after that we have nodes tag inside these nodes there will be all the nodes. So, the first node is here and then there are. So, many attributes for this node and then second node comes with all these attributes. So, basically here in this network they are assigning a lot of attributes for every node as you can see. So, as you go down once the nodes are done the edges will be there it should be the yeah. So, here you see once the nodes are done, we have edge tags they have not assigned any attributes for the edges as you can see. So, this is an example of GEXF format.

(Refer Slide Time: 12:01)



The screenshot shows a terminal window with the following text:

```
file:///home/anamika/NP... x +  
file:///home/anamika/NPTEL/Data Sets New/EuroSiS Generale Pays.gexf  
<edge id="2870" source="693" target="283"/>  
<edge id="2871" source="693" target="1348"/>  
<edge id="2872" source="693" target="643"/>  
<edge id="2873" source="693" target="414"/>  
<edge id="2874" source="693" target="1089"/>  
<edge id="2875" source="693" target="1314"/>  
<edge id="2876" source="693" target="630"/>  
<edge id="2877" source="693" target="456"/>  
<edge id="2878" source="694" target="1521"/>  
<edge id="2879" source="694" target="715"/>  
<edge id="2880" source="696" target="1067"/>  
<edge id="2881" source="697" target="438"/>  
<edge id="2882" source="697" target="757"/>  
<edge id="2883" source="697" target="1320"/>  
<edge id="2884" source="698" target="630"/>  
<edge id="2885" source="698" target="1563"/>  
<edge id="2886" source="698" target="1480"/>  
<edge id="2887" source="698" target="649"/>  
<edge id="2888" source="698" target="1560"/>  
<edge id="2889" source="698" target="243"/>  
<edge id="2890" source="698" target="76"/>  
<edge id="2891" source="698" target="946"/>  
<edge id="2892" source="698" target="203"/>  
<edge id="2893" source="699" target="422"/>  
<edge id="2894" source="699" target="1603"/>  
<edge id="2895" source="699" target="430"/>  
<edge id="2896" source="699" target="1062"/>  
<edge id="2897" source="700" target="1594"/>  
<edge id="2898" source="707" target="971"/>  
<edge id="2899" source="707" target="1482"/>  
<edge id="2900" source="707" target="1539"/>  
<edge id="2901" source="707" target="1168"/>  
<edge id="2902" source="707" target="1551"/>
```

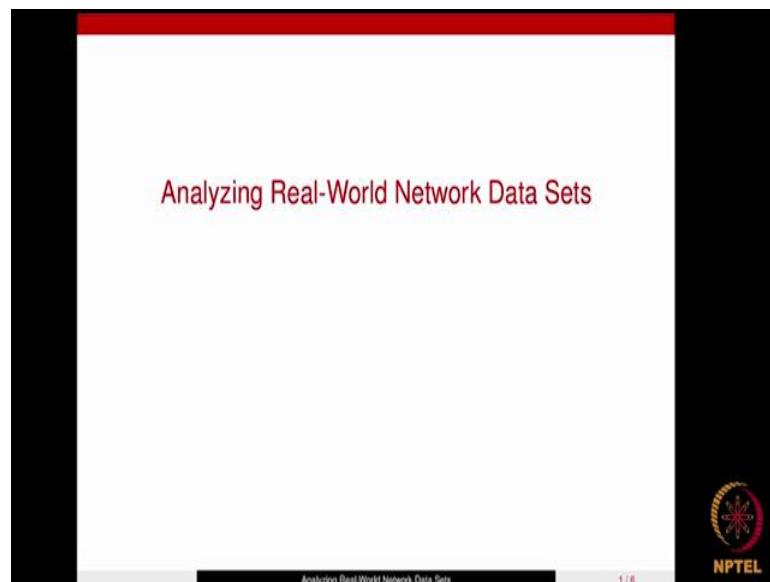
NPTEL

So, you saw there are so many resources available, you can just explore and download the ones which suit your purpose for an analysis. So, this is the basic introduction to how we can download datasets. Next, we will see how we can analyze these network datasets that we have downloaded.

**Social Networks**  
**Prof. S.R.S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

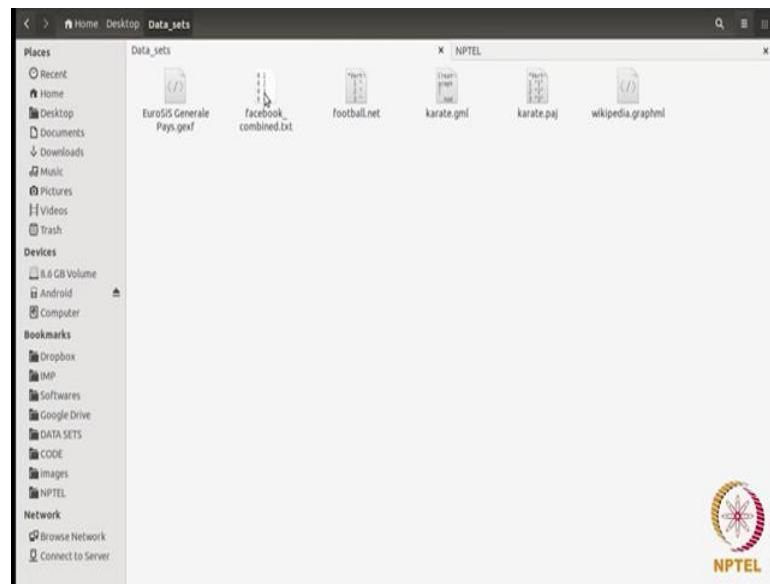
**Lecture - 21**  
**Handling Real-world Network Datasets**  
**Datasets: Analyzing Using Networkx**

(Refer Slide Time: 00:06)



Hey everyone in the previous video we had downloaded a number of network datasets in different formats, in this video we are going to see how we can analyze them using networkx package of python; let us take a look at the datasets that we have downloaded.

(Refer Slide Time: 00:25)



So, these are the datasets that we downloaded in the previous video. So, we had 6 networks, we have a network in gexf format, we also had a Facebook network in Edgelist format, we had a football network in dot net format which it is equivalent to Pajek format and we had karate club network in gml format, we also had karate network in Pajek format and we also had a Wikipedia network in GraphML format.

So, these were the 6 network datasets that we had, now we going to see how we can analyze them. So, I have all the datasets in this folder, I am going to create a new python file here where I will be writing all the 4. So, datasets.py I am going to open in it in an editor I am using sublime text here can use any editor for that matter.

(Refer Slide Time: 01:26)



```
1 import networkx as nx
2 import matplotlib.pyplot as plt
3
4 # G = nx.read_edgelist('Data_sets/facebook_combined.txt')
5 # G = nx.read_pajek('Data_sets/football.net')
6
7 G = nx.read_gml('Data_sets/karate.gml')
8 # G = nx.read_pajek('Data_sets/karate.paj')
9 # G = nx.read_graphml('Data_sets/wikipedia.graphml')
10 # G = nx.read_gexf('Data_sets/EuroSiS_Generale_Pays.gexf')
11
12 # G = nx.read_gml('Data_sets/karate.gml')
13 # print nx.info(G)
14
15 # print nx.number_of_nodes(G)
16 # print nx.number_of_edges(G)
17 #
18 # print nx.is_directed(G)
19
20 nx.draw(G)
21 plt.show()
```

Since we are going to make use of networkx package and I am going import it, we are also going to visualize the networks. So, I am going to import matplotlib as well. Now let us take the first network in this folder we have let us make use of this Facebook combined dot txt network which is in Edgelist format let me copy this name.

So, now since the Facebook network is in Edgelist format, the function that we are going to make use of is read Edgelist. So, I am going to write `g = nx.read_edgelist` and here I am going to get the name of the network. Now our datasets are kept in a folder. I am sorry I think that is the name (Refer Time: 02:30). So, in that we have this is the name of the network. So, the you see the function that we are using is `read_edgelist` function, which is present in networkx package, and as a parameter we are giving the name of the network. Now this function basically takes the network in a in the Edgelist format and returns a graph object; and then we can apply any function on this graph object.

For example, if you want to look at basic information of this network, we can write `nx.info` on as a parameter here give `G`. So, `info` is function which provides a basic detail as to the number of nodes number of edges etcetera about to graph. So, let us save this file.

(Refer Slide Time: 03:26)

```
anamika@anamika-Inspiron-5423:~/Desktop$ python data_sets.py
Name:
Type: Graph
Number of nodes: 4039
Number of edges: 88234
Average degree: 43.6910
anamika@anamika-Inspiron-5423:~/Desktop$ python data_sets.py
Name:
Type: Graph
Number of nodes: 4039
Number of edges: 88234
Average degree: 43.6910
4039
88234
False
anamika@anamika-Inspiron-5423:~/Desktop$
```



And I am going to open my terminal here, I am going to run this file. So, all right. So, here you see firstly, it tells us the type; the type is graph as in it basically tells us whether it is digraph or directed graph or it is multi graph and it tells us the number of nodes, it also tells us the number of edges and the average degree.

So, these are just a basic detail about the graph, now let us go back to our file and add some more things. So, this is what the basic things if you just want to get the number of nodes you can write. So, now, `nx.number_of_nodes` is a function which returns you the number of nodes if you have to just get the number of nodes, you can use this function and similarly if you want the number of edges if you do not want all the other information you can use this. And in case you want to know whether the graph is directed or not, then you can make use of this function `print nx.is_directed`. So, as a parameter you pass the graph.

So, this should tell us whether the graph is directed or not, let us go back and try to run this. So, here you see after the basic statistics it told us the number of nodes and edges and it is false that is it is non directed. So, this Facebook network that was a friendship network it is basically undirected network. So, that was about how we can we read and Edgelist network into networkx object. Now let us see the other kinds of networks that we have, we also have here dot net format as I told you in the previous video that dot net

format is basically the Pajek format. So, in order to read this network into the networkx object we will use Pajek functions.

So, let me show you how to do it. So, what I am going to do is I am going to use this function read Pajek, which is a function that is used to read a dot net or dot paj file. So, let me check the name of the network football dot net. So, change it. So, I am reading this dot net network through this function read Pajek into a graph object g, and then I applying all these operations let us see and we run this.

(Refer Slide Time: 06:10)

```
Average degree: 43.6910
anamika@anamika-Inspiron-5423:~/Desktop$ python data_sets.py
Name:
Type: Graph
Number of nodes: 4039
Number of edges: 88234
Average degree: 43.6910
4039
88234
False
anamika@anamika-Inspiron-5423:~/Desktop$ python data_sets.py
Name:
Type: MultiDiGraph
Number of nodes: 35
Number of edges: 118
Average in degree: 3.3714
Average out degree: 3.3714
35
118
True
anamika@anamika-Inspiron-5423:~/Desktop$
```



So, here you see that the type of graph is multi digraph; that means, there are multiple edges between the nodes, and it is also directed graph and it is telling us a number of nodes number of edges, and since it is a directed graph it is telling us the average in degree and average out degree, and after that again it is giving the result of the functions number of nodes number of edges, and it is true which means it is a directed graph.

So, these are just the basic functions, let us see what other kinds of networks we have. We have gml network we have a Pajek format as well let me show you that for reading the Pajek files as well, you use the same function that is read Pajek. So, the network name is karate dot p a j. So, I will replace this ok.

(Refer Slide Time: 07:15)

```
False  
anamika@anamika-Inspiron-5423:~/Desktop$ python data_sets.py  
Name:  
Type: MultiDiGraph  
Number of nodes: 35  
Number of edges: 118  
Average in degree: 3.3714  
Average out degree: 3.3714  
35  
118  
True  
anamika@anamika-Inspiron-5423:~/Desktop$ python data_sets.py  
Name:  
Type: MultiGraph  
Number of nodes: 34  
Number of edges: 78  
Average degree: 4.5882  
34  
78  
False  
anamika@anamika-Inspiron-5423:~/Desktop$
```



So, when I run this, I am getting that this is a multi graph and the number of nodes is 34 number of edges is 78 and this is a average degree and it is not a directed graph. So, it is. So, I am getting false here ok.

(Refer Slide Time: 07:35)

```
Number of nodes: 921  
Number of edges: 1081  
Average in degree: 1.1737  
Average out degree: 1.1737  
True  
anamika@anamika-Inspiron-5423:~/Desktop$ python data_sets.py  
Traceback (most recent call last):  
  File "data_sets.py", line 9, in <module>  
    G = nx.read_gexf('Data_sets/EuroSiS_Generale_Pays.gexf')  
  File "<string>", line 2, in read_gexf  
    File "/usr/lib/python2.7/dist-packages/networkx/utils/decorators.py", line  
263, in _open_file  
    result = func(*new_args, **kwargs)  
  File "/usr/lib/python2.7/dist-packages/networkx/readwrite/gexf.py", line 1  
61, in read_gexf  
    G=reader(path)  
  File "/usr/lib/python2.7/dist-packages/networkx/readwrite/gexf.py", line 5  
74, in __call__  
    raise nx.NetworkXError("No <graph> element in GEXF file")  
networkx.exception.NetworkXError: No <graph> element in GEXF file  
anamika@anamika-Inspiron-5423:~/Desktop$
```



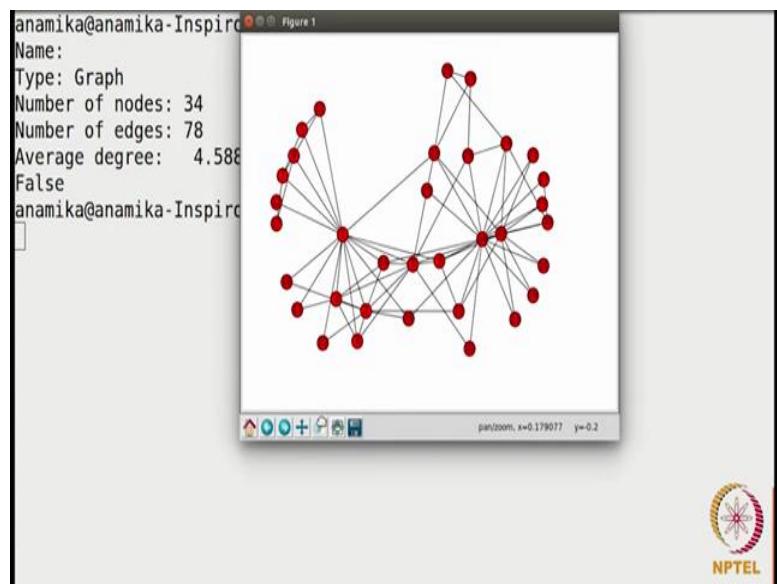
Now, 2 more network formats that we have are GraphML and gexf. So, let me show you quickly show you how we can read them as well. So, graph m 1 how do we handle it we. So, let me the function that we use it read graph m 1 and the file name is Wikipedia dot graph m 1. So, this is how we will use it; since this is written information, I am going to I

am going to comment this, now let me run this all right. So, what we are getting here is that it is a digraph Wikipedia because this is graph between that the nodes are the articles. So, is it basically tells us whether an article is referred to in the other article or not? So, it is basically a directed graph 921 are the number of nodes number of edges are giving and since it is directed we are getting in degree as well as the out degree, the average in degree and out degree and it is true that means, it a directed graph.

So, let us go back here and the only format that is left is gexf, let me quickly show you how to convert into graph object as well and copying it is name and here let us go back. Since this is a gf; gexf format will make use of function read\_gexf and (Refer Time: 09:24) name of the function let me rename this, so that it does not create any problem and let me rename this is well. So, it should work let us see. So, this is how we can read various networks in different formats and convert them into a graph object; once they are converted into graph object, we can apply various functions on them, and we can play around with them. Now let also show you how we can visualize a network in networkx package.

So, let me take a small network, let me take karate network itself and we had it in gml format. I think we have in that video we can quickly add it. So, the function is read\_gml. So, if you have a graph which is in gml format, the function that you will used is nx.read\_gml. So, all right. So, we read a karate network which was in gml format, we made use in function nx.read\_gml.

(Refer Slide Time: 11:05)



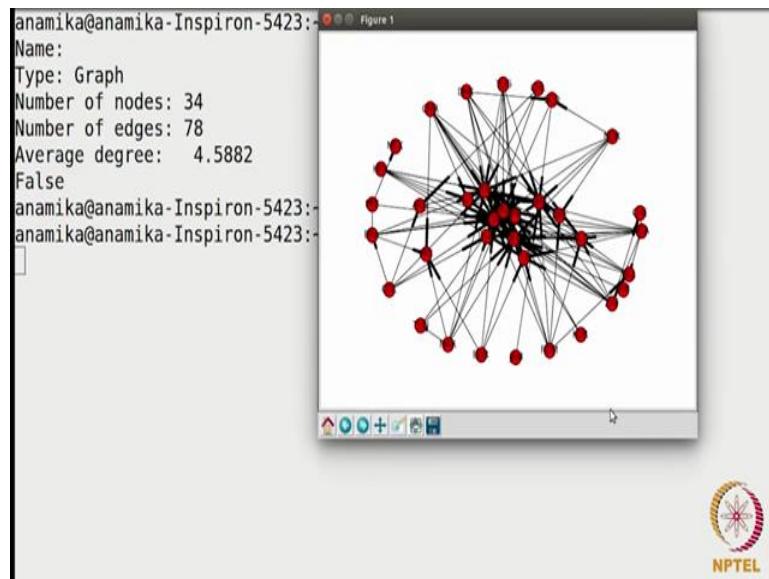
So, let us execute this program. So, this is a simple graph, and this is a number of nodes and edges and it is an undirected graph. Now let me show you how we can visualize this graph? I am going to comment this; I am going to comment this as well.

Now, the function that we used to visualize the graph is `nx.draw`, and the parameter we will do the graph that we have to draw in order to see that graph we have to use this function `plt.show`, basically the `show` function which is available in `matplotlib`. So, that is how we will be able to see this graph, now let me run this all right. So, this is all graph the karate network, and the labels are given here. Let me also show you a few features that this interface provides for example, this when you click this you can just move the graph the way you want, and the next option is zoom to rectangle. So, when you click that if you want to carefully observe some part of the graph you can just zoom it and see and if you further want to do that you can do the way you want and then you can go back then you go then go back. So, these are few functions that few features that you can make use of, and this is configures subplots this will be used when we plot something this is just a graph, I will show you the functionality of this later then this is how you can save the figure ok.

Let me close this window now let me go back to the program; let me also show you how directed graph in `networkx` looks like. Since this karate is in undirected graph let me

comment this and if I am not wrong this football network was a directed network. So, I am doing the same analysis on football dot net.

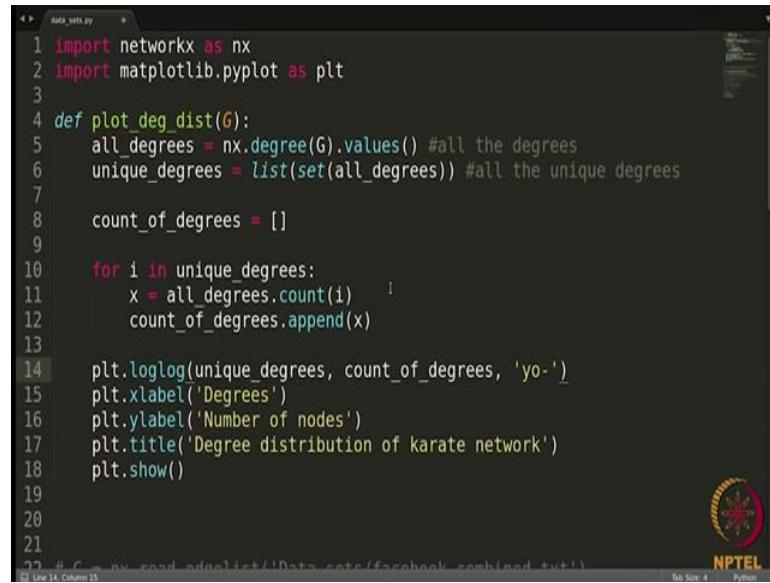
(Refer Slide Time: 13:13)



Let me execute it again, you see this is how directed graph and networkx looks like. So, you see the arrows are represented like this, if you want to zoom it again you can make use of this feature.

You want to zoom this area you can further zoom it is well. So, this is how you can closely analyze whatever you want in the graph right and you can then go back, or you can just press home here. Then you can save the graph is well let me close this now I am going to continue the rest of the analysis on karate networks.

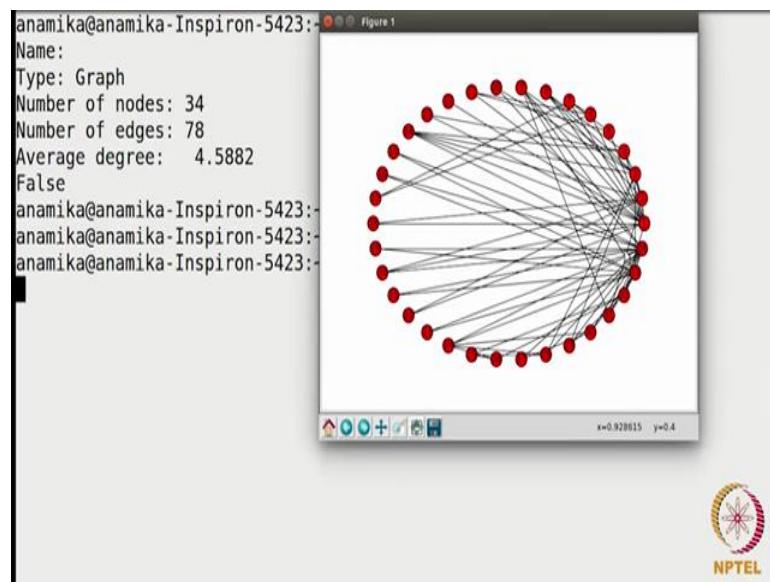
(Refer Slide Time: 13:51)



```
1 import networkx as nx
2 import matplotlib.pyplot as plt
3
4 def plot_deg_dist(G):
5     all_degrees = nx.degree(G).values() #all the degrees
6     unique_degrees = list(set(all_degrees)) #all the unique degrees
7
8     count_of_degrees = []
9
10    for i in unique_degrees:
11        x = all_degrees.count(i)
12        count_of_degrees.append(x)
13
14    plt.loglog(unique_degrees, count_of_degrees, 'yo-')
15    plt.xlabel('Degrees')
16    plt.ylabel('Number of nodes')
17    plt.title('Degree distribution of karate network')
18    plt.show()
19
20
21
22 # C = nx.read_edgelist('Data sets/facebook_combined.txt')
```

So, I am going to on comment object back. Let me also show you that there are different layouts which are available in networkx for example, we have as if now use this function nx.draw, we can use another function if you want to different layout. So, we can use nx.draw\_circular. So, this is one of the layouts there are various layouts available, let me show you the output that we get in this case.

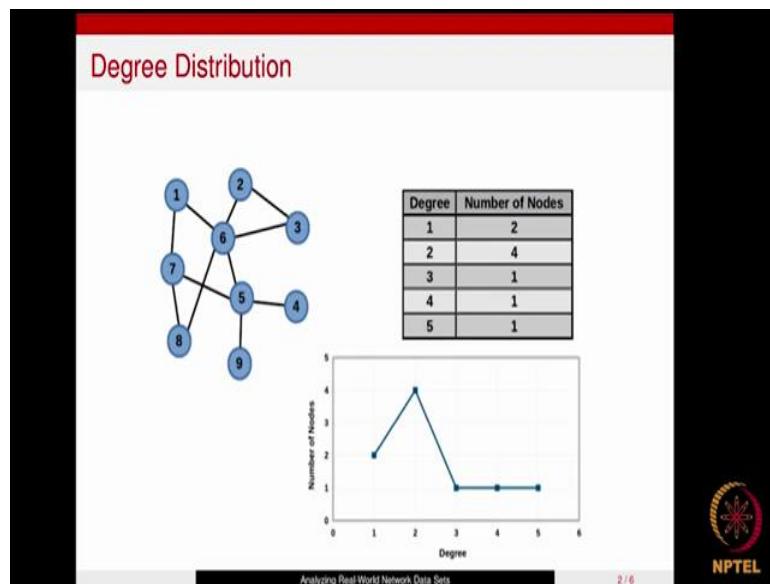
(Refer Slide Time: 14:23)



So, here the all the nodes are arranged along a circle and the edges between them are shown like this. So, this is called circular layout, there are number of other layouts as

well for example, spectral layout and spring layout. So, you can just read the documentation about them. So, we have visualized the network let us close this and let us go back to our program and we can do some basic analysis on this, one of the thing that we can check on this network is we can check the degree distribution.

(Refer Slide Time: 15:00)



Now, what is degree distribution? Degree distribution basically tells us how many nodes there in the network are, that have a degree. So, this is done for all possible degrees that a node can have in the graph; let us take this example graph. So, here how many nodes are having degree 1? So, node number 4 and node number 9 they are having degree 1. So, corresponding to 1 will have 2 and similarly we can check how many nodes are having degree 2, how many nodes are degree 3, 4 and 5. So, these are all the possible degrees that nodes can have in this graph and corresponding to the degree we have the number of nodes that have that particular degree. So, this basically is called the degree distribution of the nodes in a graph. Now it is always nice idea to plot the degree distribution to get a better idea of the graph.

So, when we plot this, we get this kind of distribution. So, on the x axis we maintain the degree and on the y axis we maintain the number of nodes that have that particular degree. So, this is the kind of plot that we get for this example graph, we can check for our datasets what kind of degree distribution to the exhibit. So, let us go back to our program and let us try to check what kind of degree distribution this karate network has.

For that purpose, I am going to create a function to plot the degree distribution of this graph G. So, let me create a function here, before we implement function, I want to show you a few things on the ipython console. So, let me copy this and let me open the ipython console here.

(Refer Slide Time: 17:03)

```
anamika@anamika-Inspiron-5423:~/Desktop$ ipython
Python 2.7.6 (default, Oct 26 2016, 20:30:19)
Type "copyright", "credits" or "license" for more information.

IPython 1.2.1 -- An enhanced Interactive Python.
?           -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help        -> Python's own help system.
object?    -> Details about 'object', use 'object??' for extra details.

In [1]: import networkx as nx

In [2]: import matplotlib.pyplot as plt

In [3]: G = nx.read_gml('Data_sets/karate.gml')

In [4]: nx.degree(G)■
```



So, basically, I am just copied those first 2 statements here and let me also copied this statement here ok.

So, we have the karate network in the object G; now if I want to see the degree of each node in this graph, I can make use of this function nx.degree(G).

(Refer Slide Time: 17:33)

```
16: 2,
17: 2,
18: 2,
19: 2,
20: 3,
21: 2,
22: 2,
23: 2,
24: 5,
25: 3,
26: 3,
27: 2,
28: 4,
29: 3,
30: 4,
31: 4,
32: 6,
33: 12,
34: 17}
```

```
In [5]: nx.degree(G).values
```



(Refer Slide Time: 17:36)

```
In [4]: nx.degree(G)
Out[4]:
{1: 16,
 2: 9,
 3: 10,
 4: 6,
 5: 3,
 6: 4,
 7: 4,
 8: 4,
 9: 5,
 10: 2,
 11: 3,
 12: 1,
 13: 2,
 14: 5,
 15: 2,
 16: 2,
 17: 2,
 18: 2,
```



So, what this function does this degree function it basically returns a dictionary, where the key is the number of the node and the value is the degree of that node. So, here we get the dictionary for all 34 nodes, we are getting the degrees of these nodes. Now let us go one step back and recall what is our (Refer Time: 17:56) is to have the degree distribution of nodes in the graph.

So, basically, we want that for particular degrees how many nodes are there in the network that are have in that degree. So, we basically first of all want to get the possible

degrees that the nodes are having in this network. So, we are getting the dictionary here where we are getting all the possible values of the degrees that is in that the nodes can have. So, what we are interested here in is basically the values. So, what I can write is nx.degree(G). So, this is going to give me a dictionary all I am interested in is the value. So, I am going to write dot values here.

(Refer Slide Time: 18:41)

```
2,  
2,  
2,  
2,  
3,  
2,  
2,  
2,  
2,  
5,  
3,  
3,  
3,  
2,  
4,  
3,  
4,  
4,  
6,  
12,  
17]  
  
In [6]: setnx.degree(G).values()
```



(Refer Slide Time: 18:43)

```
32: 6,  
33: 12,  
34: 17}  
  
In [5]: nx.degree(G).values()  
Out[5]:  
[16,  
 9,  
 10, 1  
 6,  
 3,  
 4,  
 4,  
 4,  
 5,  
 2,  
 3,  
 1,  
 2,  
 5,  
 2,
```



So, it should return me earliest which is having all the values. So, what is get here is basically all the possible degrees that the nodes are having. Now my aim is get the

possible degrees that the nodes can have; here you see there are lot of reputation. So, I want to get the unique degrees. So, in that case what I can do is, I can just write all this inside of function set.

(Refer Slide Time: 19:13)

```
2,
2,
5,
3,
3,
2,
4,
3,
4,
4,
6,
12,
17]

In [6]: set(nx.degree(G).values())
Out[6]: {1, 2, 3, 4, 5, 6, 9, 10, 12, 16, 17}

In [7]: list(set(nx.degree(G).values()))
Out[7]: [1, 2, 3, 4, 5, 6, 9, 10, 12, 16, 17]

In [8]:
```



So, what it will do is, it will convert the output into a set and we know that in set there cannot be in reputations. So, what we get is the unique values, basically the unique degrees that the nodes can have in this particular network. Now a list is more flexible data structure as compared to set because we can perform lot of operation. So, I can further convert this into list if I want; this is basically up to you how or you want to handle the data.

So, I convert this into a list now what I get finally, is a list of all the unique values of degrees that the nodes can have in this network. So, I should you here so that we can see what is going on in the function. Now let us get back and use these functions in the function that we are creating. So firstly, I want all the degrees. So, I will write `nx.degree(G)`, I want all the values only I do not want the dictionary. So, I will write dot values. So, here I get all the degrees let me comment all the degrees.

Now, I am also interesting getting all the unique degrees so that I can see what the possible degree values is. So, what I am going to do the same thing that I did on the console, I am going to pass all degrees here. So, here I will get all the unique degrees. Now to get the degree distribution what I basically want is I want to out of all the values

that are there in this list unique degrees, I want to see how many nodes are having that particular degree. So, basically what I will do is I will fetch one element out of this list that is unique degrees, and I will see how many occurrences of that value there in all degrees are right.

So, probably I can start for loop here. So, I will write for i in unique degrees sorry for i. So, what I want to check is how many occurrences of i are available in all degrees. So, I can start a variable probably x is equal to all degrees dot count. So, we have this function count in a list, which tells us the occurrence number of occurrences of a particular element in that list. So, x will be telling us the number of occurrences of the degree i in the list all degrees. So, probably we can keep a track of all these values. So, we can create another list count of degrees I will I started a empty list and so I am going to append the x values to this list. So, basically the occurrences of the first degree are now stored in the list count of degrees.

So, after this for loop is finished, we will have all the degree distributions in this list that is called count of degrees, and then we can plotted at. So, let us try plotting is it. So, plt.plot as you might be knowing there are 2 parameters that have to be passed here. So, on the x axis we want all the unique degrees, and on the y axis we want have many nodes have that degree which we have stored in count of degrees. So, will pass that here. So, let us try plotting it. So, I will write plt.show we have not call this function x.

(Refer Slide Time: 23:20)



```
22 # G = nx.read_pajek('Data_sets/football.net')
23 # G = nx.read_gml('Data_sets/karate.gml')
24 #
25 G = nx.read_gml('Data_sets/karate.gml')
26 # G = nx.read_pajek('Data_sets/karate.paj')
27 # G = nx.read_graphml('Data_sets/wikipedia.graphml')
28 # G = nx.read_gexf('Data_sets/EuroSiS_Generale_Pays.gexf')
29 #
30 # G = nx.read_gml('Data_sets/karate.gml')
31 # print nx.info(G)
32 #
33 # print nx.number_of_nodes(G)
34 # print nx.number_of_edges(G)
35 #
36 # print nx.is_directed(G)
37 #
38 # nx.draw(G)
39 # nx.draw_circular(G)
40 # plt.show()
41 # plot_deg_dist(G)
42 print 'Density is', nx.density(G)
43
```

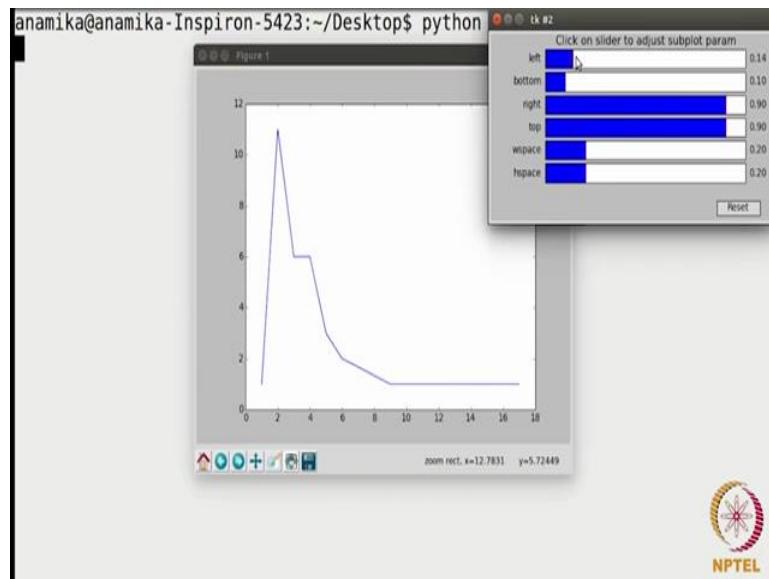
So, let me call this function here and comment this. So, I will call this function plot degree distribution and I will pass this G here ok.

(Refer Slide Time: 23:38)

```
anamika@anamika-Inspiron-5423:~/Desktop$ python data_sets.py
Name:
Type: Graph
Number of nodes: 34
Number of edges: 78
Average degree: 4.5882
False
anamika@anamika-Inspiron-5423:~/Desktop$ python data_sets.py
anamika@anamika-Inspiron-5423:~/Desktop$ python data_sets.py
anamika@anamika-Inspiron-5423:~/Desktop$ python data_sets.py
anamika@anamika-Inspiron-5423:~/Desktop$ clear
```



(Refer Slide Time: 23:40)

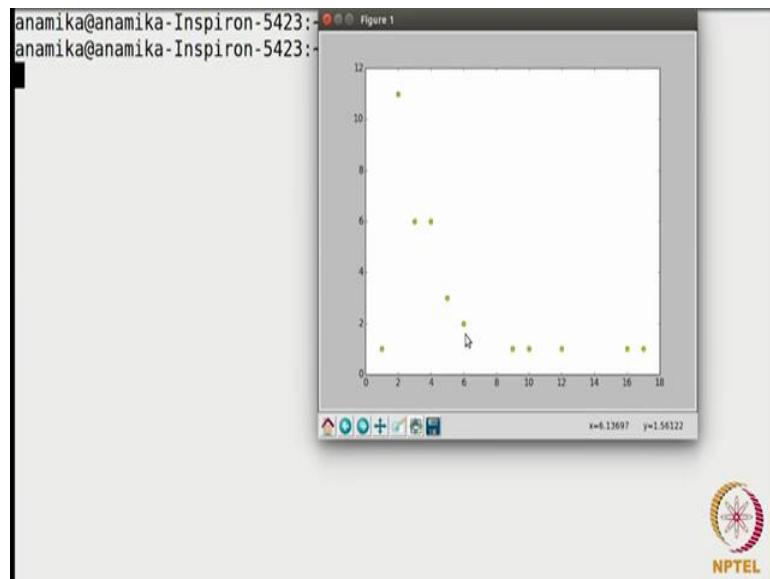


Let us go back here not here will go will try (Refer Time: 23:39) this. So, this is the degree distribution that we are getting for the karate club ah network.

So, this is the x axis where the possible degrees are there and on the y axis the number of nodes having that degree are there, and we you can just again play around with this plot as I told you, you can just use this to move the plot, you can use this to zoom of

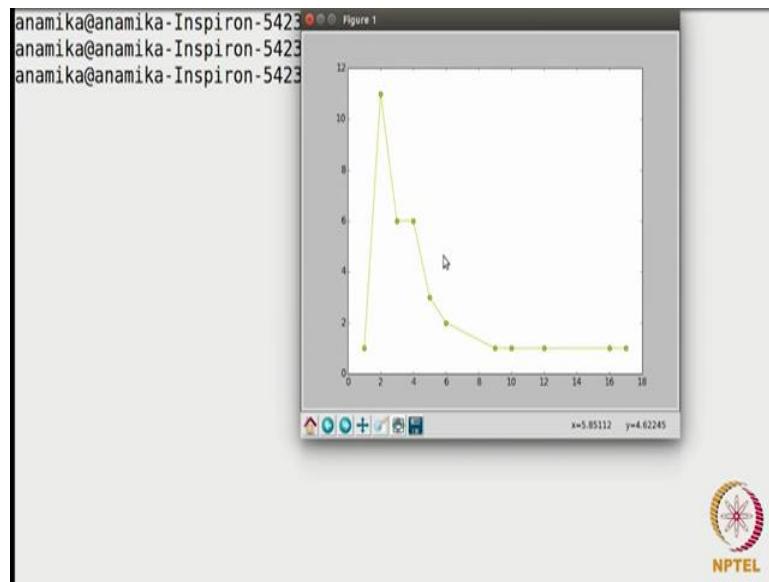
particular part and you can just go back as well and this is a feature that you can use here basically you can just increase or decrease the x and y margins. So, this is up to you whatever way you want it, and you can always reset it is well. So, let me close it. So, this is basically up to you how you want to visualize it. Let be close it there was no x and y axis as you see. So, we can do that. So, let me just quickly do that, before that you can also change the way this plot is appearing so, if I put these dots you will get the plot in the form of these dots ok.

(Refer Slide Time: 24:56)



So, you get yellow dots here and you can also put line here and dots is well. So, we will get both the things.

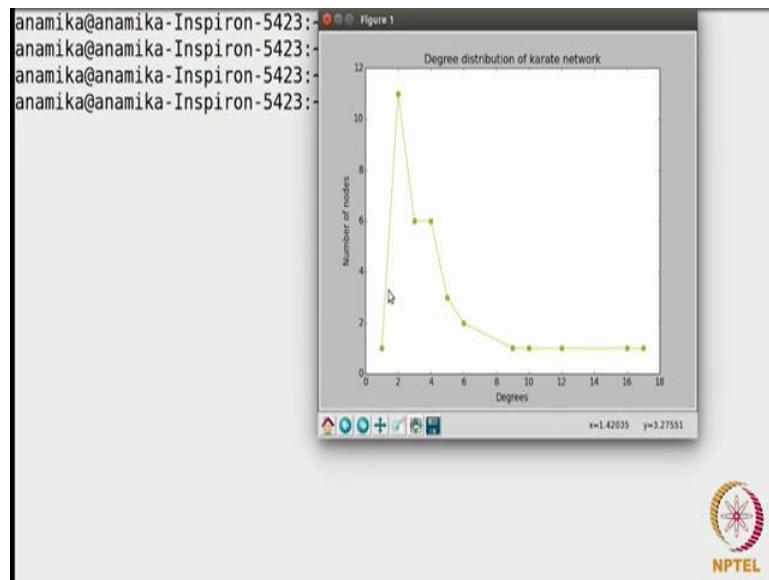
(Refer Slide Time: 25:06)



So, this is plot that you getting; one thing to observe here is that most of the real world networks exhibit power law degree distribution, which means that there are very few nodes which have very high degree and there are lot of nodes which have very less degree. So, the same is being followed in this small network is well, apart from one exception of this node. So, that might happen in some cases, but in general real-world networks have this power law degree distribution ok.

So, let us go back and add a few more things may be let us add the x label. So, let us add degrees here and then we can add the y label as well may be number of nodes and maybe we can add the title as well and degree distribution of karate network.

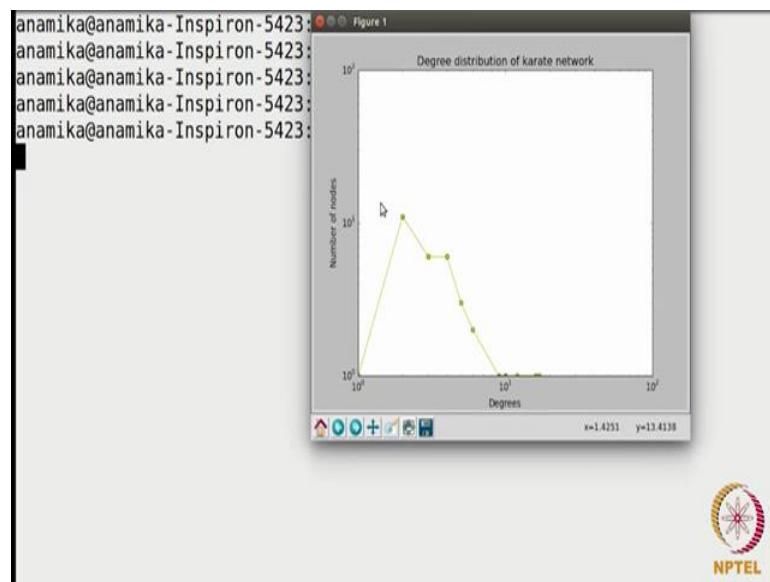
(Refer Slide Time: 26:17)



So, let us sum this you get the title and the x and y labels as well. So, you can just use more features more functions and decorate this plot the way you want. There is one more thing that usually is done in case of power law degree distribution, we can also check the log log plot of that.

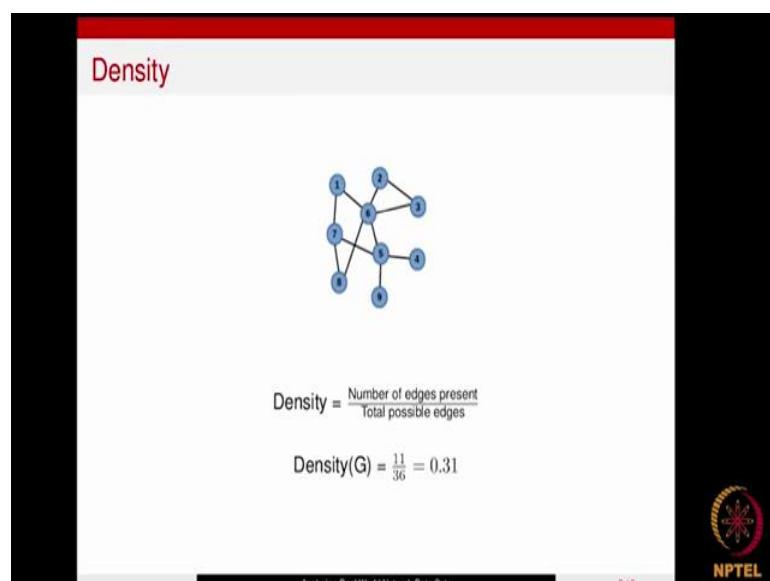
So, how can we do that we can just replace this plot by log log. So, in that case it will give as a log log plot which basically means in take the log of x axis it take the log of y axis, and if network is following complete power law the log log plot should be in a straight line. So, let us see what kind of plot we get in this case.

(Refer Slide Time: 27:01)



So, here firstly, we had section here. Secondly, it was not perfect power law. So, we have this kind of line which is not exactly straight. So, this is the kind of log log plot that we are getting in this case. So, this one about the degree distribution, now I am closing it let us go back here and see some more properties that we can analyze on this network we have done with degree distribution let us go ahead. So, next thing that we can check is density.

(Refer Slide Time: 27:27)



Density value of graph basically tells us whether it is a parts graph or it is a dense graph with respect to the number of edges present. So, if there are n nodes in a network, the total possible edges that that network can have will be  ${}^nC_2$ . Out of these  ${}^nC_2$  edges how many what is the fraction of the edges that are present in the graph is basically what is stored by the density value.

So, if it is a simple graph the density value between will be between 0 and 1, and if it is empty graph the density will be 0 if it is complete graph density will be 1; however, if it is a multi graph where more than one edges are allowed between 2 nodes, in that case density value will be more than one can be more 1 for example, in this diagram you see simple graph with 9 nodes. So, the total possible edges that can be there in this network will be 9 choose 2 which is 9 into 8 divided by 2 that is 36. Now the number of edges that are present in this graph are 11. So, the density is going to be 11 divided by 36 and that is equal to 0.31. So, the graph is not very dense. So, this is the kind of indication that the density value gives us.

(Refer Slide Time: 29:06)

```
In [9]: G = nx.complete_graph(100)
In [10]: nx.density(G)
Out[10]: 1.0

In [11]: H = nx.Graph()
In [12]: H.add_nodes_from([1,2,3,4])
In [13]: nx.density(H)
Out[13]: 0.0

In [14]:
```



So, we can go back to the console and see the density value for few networks for example, let me go here and let us create (Refer Time: 29:04). Let us create a complete graph. So, I am going to write `G = nx.complete_graph` of say 100 nodes. So, since it is a complete graph what should be the density value let us check `nx.density`. So, this is the function density, which is available in networkx, which will give us the density value;

obviously, in case of complete graph it is going to be 1. Let me now create in other graph let me repeat nx.Graph. So, I have not added any edges into it let me add few nodes here. So, I am going to pass list. So, I am passing only 4 nodes.

Let us check the density value of this network. So, since we have not added any edge; obviously, the density value was going to be 0; and let us go back here and see what is going to be the density value for our network karate network. So, what I will write is print (Refer Time: 30:26) density is. So, n x dot density sorry and I am to pass G here let us go back (Refer Time: 30:40) density is 0.139.

(Refer Slide Time: 30:40)

```
anamika@anamika-Inspiron-5423:~/Desktop$ python data_sets.py
Density is 0.139037433155
anamika@anamika-Inspiron-5423:~/Desktop$ █
```



So, basically it is sort of parts graph, so that we can check. So, that is about the density, let us go back and see few more things that we can check on these networks.

(Refer Slide Time: 30:55)

## Clustering Coefficient

$$\text{Clustering Coefficient} = \frac{\text{Actual Number of friendships}}{\text{Total possible friendships}}$$

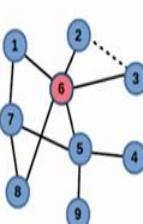
Analyzing Real-World Network Data Sets 4 / 6



Next you can see clustering coefficient. So, for a given node clustering coefficient basically tells us the number of lengths that are present amongst the neighbors of this node with respect to the total number of lengths that can be possible.

(Refer Slide Time: 31:15)

## Clustering Coefficient contd.


$$\text{Clustering Coefficient}(6) = \frac{1}{10}$$

Analyzing Real-World Network Data Sets 5 / 6

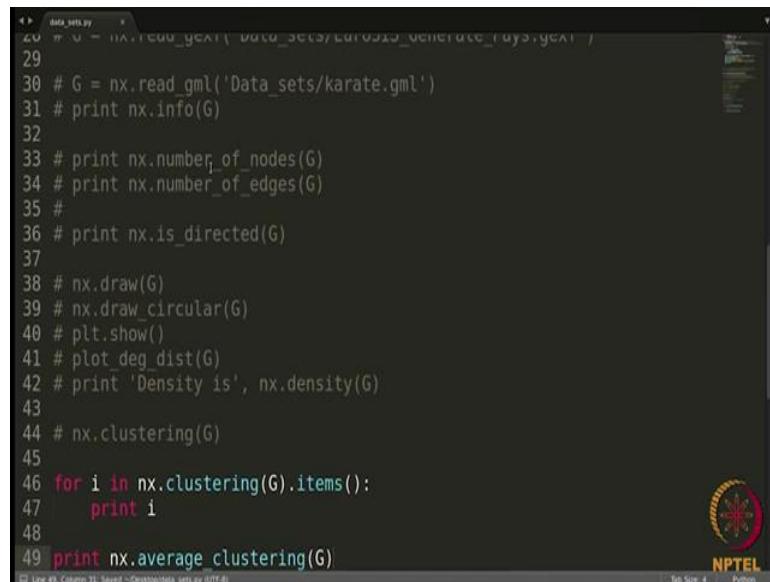


I will show you using example let us take this network let us try to find out the clustering coefficient for these nodes 6. So, as you can see this node as 5 neighbors write these are 1 2 3 5 and 8. So, there are 5 neighbors. So, what we have to check here is the number of

links that are present amongst these neighbors, that is the number of links amongst 1, 2, 3, 5 and 8.

As you can see, we see only one link that is there between 2 and 3; there is no other link present amongst these neighbors. So, on the numerator we will put 1 and, on the denominator, we will put the total possible links that that can be there amongst these 5 nodes. So, amongst 5 nodes total possible links can be  ${}^5C_2$  which is  $5 * 4$  divided by 2 that is 10. So, in this case the clustering coefficient of this node 6 will be 1/10. So, in case of friendship network, this clustering coefficient basically tells us how closely net the friends of node are. So, we can calculate the clustering coefficient value for every node and then we can find the average as well. So, average clustering coefficient for they tell us the amount of clustering present amongst the nodes in the graph.

(Refer Slide Time: 32:44)



```
28
29
30 # G = nx.read_gml('Data_sets/karate.gml')
31 # print nx.info(G)
32
33 # print nx.number_of_nodes(G)
34 # print nx.number_of_edges(G)
35 #
36 # print nx.is_directed(G)
37
38 # nx.draw(G)
39 # nx.draw_circular(G)
40 # plt.show()
41 # plot_deg_dist(G)
42 # print 'Density is', nx.density(G)
43
44 # nx.clustering(G)
45
46 for i in nx.clustering(G).items():
47     print i
48
49 print nx.average_clustering(G)
```

So, let us go back and try to check the same for our network. So, I am going to comment this, the function that we can use for finding the clustering is `nx.clustering`; however, this function basically returns a dictionary which gives the clustering coefficient value for every node. So, you can always treat over this dictionary. So, what I will do is, for `i` in `nx.clustering G`. So, I am interested in all the items. So, I will write `dot items` I want to print `i`. So, I am going to comment this.

So, what we are doing here is this nx.clustering is returning a dictionary, which contains the clustering coefficient values for all the nodes, we are just going to print them. So, let us run this.

(Refer Slide Time: 33:38)

```
anamika@anamika-Inspiron-5423:~/Desktop$ python data_sets.py
(1, 0.15)
(2, 0.333333333333333)
(3, 0.2444444444444444)
(4, 0.666666666666666)
(5, 0.666666666666666)
(6, 0.5)
(7, 0.5)
(8, 1.0)
(9, 0.5)
(10, 0.0)
(11, 0.666666666666666)
(12, 0.0)
(13, 1.0)
(14, 0.6)
(15, 1.0)
(16, 1.0)
(17, 1.0)
(18, 1.0)
(19, 1.0)
(20, 0.333333333333333)
```



(Refer Slide Time: 33:45)

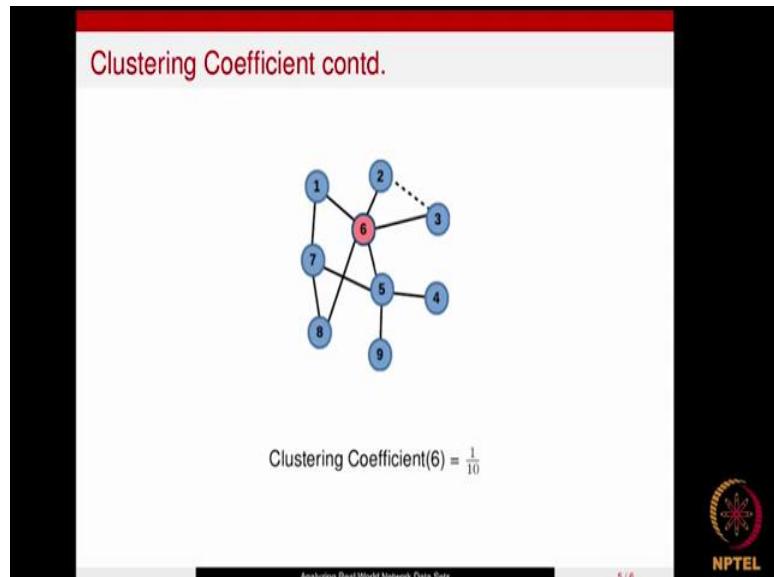
```
(18, 1.0)
(19, 1.0)
(20, 0.333333333333333)
(21, 1.0)
(22, 1.0)
(23, 1.0)
(24, 0.4)
(25, 0.333333333333333)
(26, 0.333333333333333)
(27, 1.0)
(28, 0.166666666666666)
(29, 0.333333333333333)
(30, 0.666666666666666)
(31, 0.5)
(32, 0.2)
(33, 0.196969696969696)
(34, 0.11029411764705882)
0.570638478208
anamika@anamika-Inspiron-5423:~/Desktop$ python data_sets.py
Diameter is 5
anamika@anamika-Inspiron-5423:~/Desktop$
```



So, we are getting this dictionary, where for every node we are getting the clustering coefficient value. So, if you want the average clustering coefficient, we can either average these values or we can directly make use of another function which is n x dot average I am sorry average clustering. So, this should tell us the average clustering

present in the network. So, you see 0.57 is an average clustering. So, more this value more the clustering and more tight in it the people in the friendship network r.

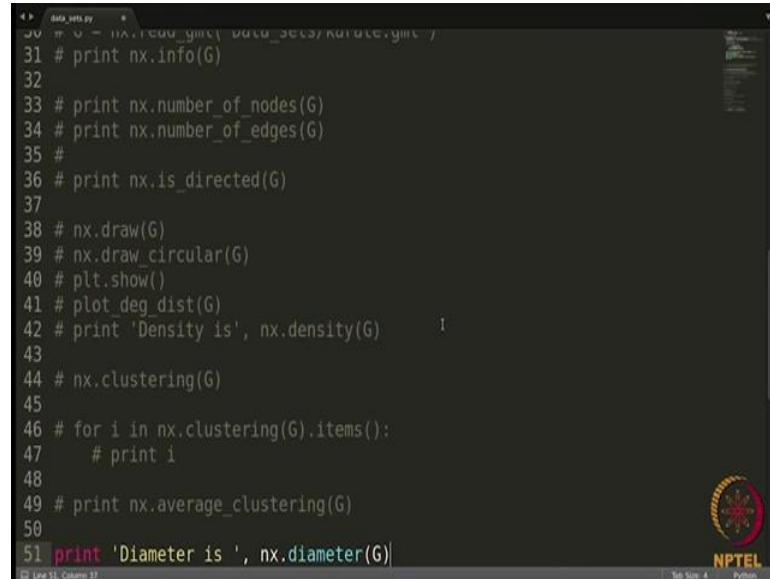
(Refer Slide Time: 34:26)



That was about the clustering coefficient let us go back and check few more properties. So, what is the diameter of a network? Diameter is basically the maximum shortest path that we have to travel to go from one node to the other for example, if you know about all pair shortest path algorithm, it basically returns the metrics where the values are the length of the shortest path being the 2 nodes. So, is it as that for every pair of the nodes? So, whatever is a maximum value in that metrics will be the diameter of the network.

In other words, it is the shortest path between 2 most distant nodes in the network. So, for example, if you see node 1 and node 9. So, if you have to go from 1 to 9, you will have to traverse this path 1 to 6 to 5 to 9, there is no other shortest path between them. So, the length of this path is 3, and we do not see any other shortest path which is longer than this 3, if you want to go from 1 to 4 again the length of the path is 3, if you want to go from 3 to 9 the length is 3. So, we do not see any other shortest path which is more than 3. So, that is why the diameter of this network will be 3, we can check the diameter of r network here is well.

(Refer Slide Time: 35:54)



```
30 # G = nx.read_gml('data_sets/karate.gml')
31 # print nx.info(G)
32
33 # print nx.number_of_nodes(G)
34 # print nx.number_of_edges(G)
35 #
36 # print nx.is_directed(G)
37
38 # nx.draw(G)
39 # nx.draw_circular(G)
40 # plt.show()
41 # plot_deg_dist(G)
42 # print 'Density is', nx.density(G)
43
44 # nx.clustering(G)
45
46 # for i in nx.clustering(G).items():
47     # print i
48
49 # print nx.average_clustering(G)
50
51 print 'Diameter is ', nx.diameter(G)
```

So, I am going to comment this and let us check the diameter. So, I will write diameter is. So, I will the function that we can use is `nx.diameter(G)`.

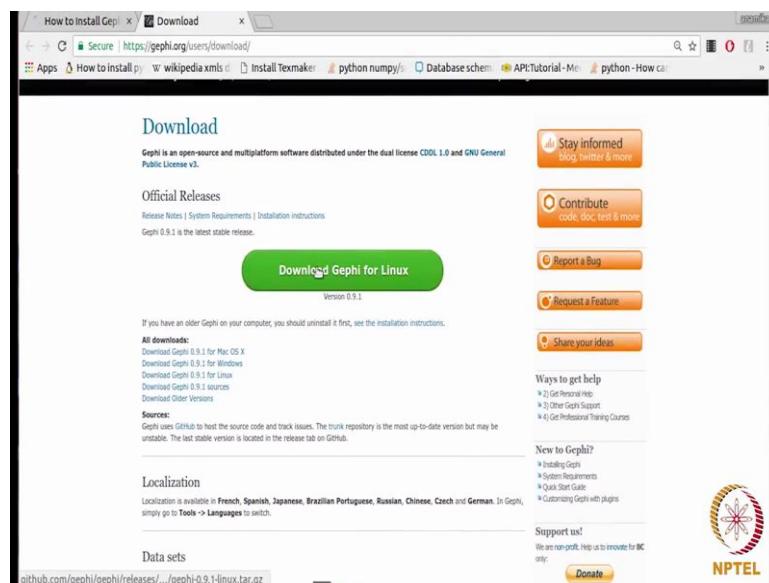
So, it should give us the diameter. So, diameter is 5. So, so there are 34 nodes here, and the diameter is 5; it is basically observe that in real world networks the diameters is basically very less because the nodes are connect to each other and that makes the distance between them very small and let us how the diameter reduces. So, these were just the few point of an analysis that we performed on the networks that we downloaded. That the main thing to notice here is that once you get the network in the networkx graph object you can just play around with it you can apply all the functions that are available, you can just read the documentation and apply the functions which are relevant for that network and you can go ahead with your analysis.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture - 22**  
**Handling Real World Network Datasets**  
**Datasets: Analyzing using Gephi**

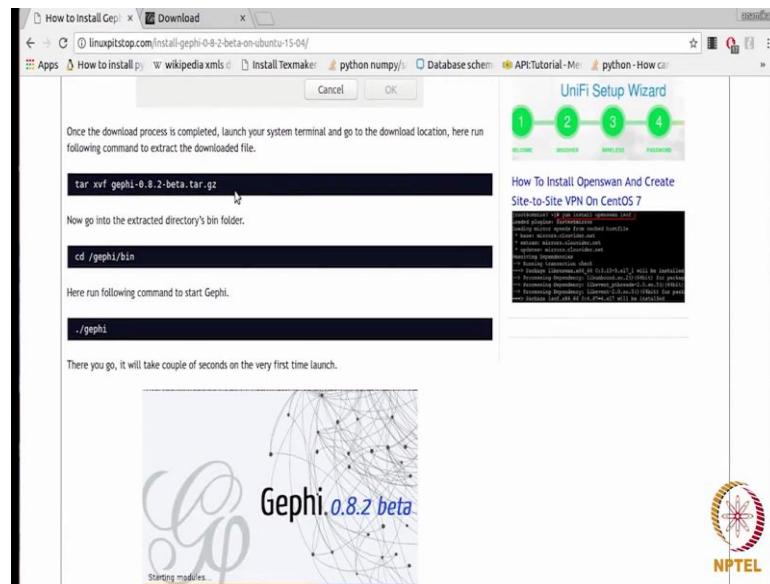
Hi everyone. In the previous video we saw how we can use networkx package on Python to analysis network data sets. In this video we are going to learn some basic features of software called Gephi which is also used for the analysis and visualization of networks. Gephi is in open source tool, which is written in Java, although networkx provides a number of functions to analysis networks. However, when it comes to visualization Gephi provides much more flexibility. So, let us get started and look at the interface of Gephi software, I all ready have a Gephi installed in my system which is a one two, but in case you want to know how we can do that I quickly show you that.

(Refer Slide Time: 00:52)



So, it is pretty simple I will just download Gephi for one two. So, we can follow the first link. So, we have to download a file I will click this link and this while that we have to download the version is 0.9.1.

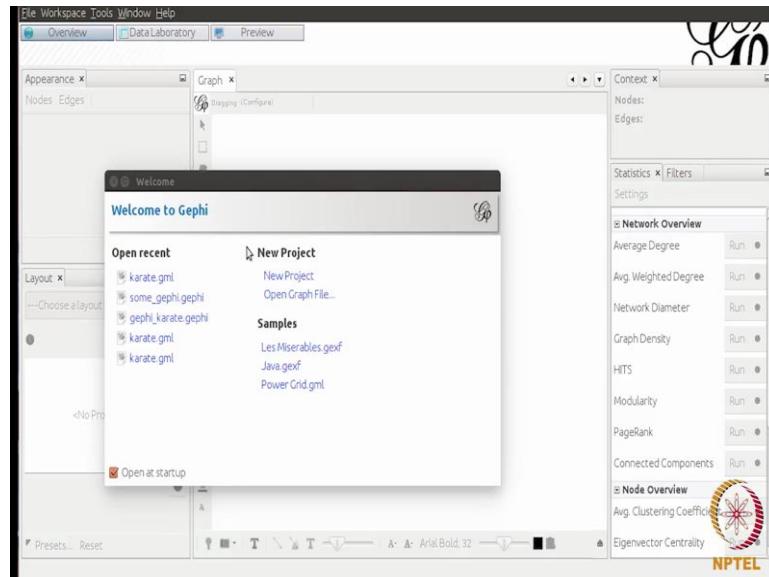
(Refer Slide Time: 01:14)



So, we are going to download this file after you download it you where ever you have saved it you will open the terminal there in that folder, and you will run this command to extract the Gephi folder out of it and after that you will get Gephi folder extracted out of the file, and you will go inside bin and after that you will run dots slash Gephi that is how you will run this software.

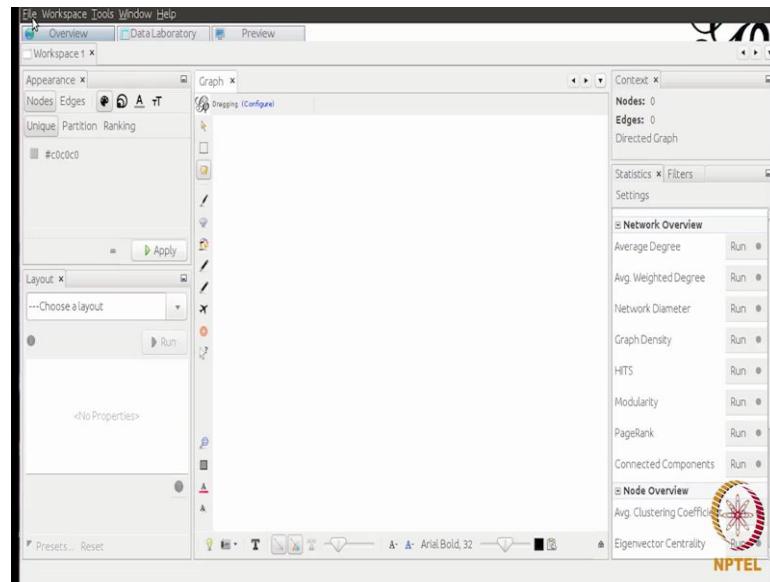
So, I already have a downloaded in my systems. So, I am going to show you how it can started; I am going to open the terminal here.

(Refer Slide Time: 01:49)



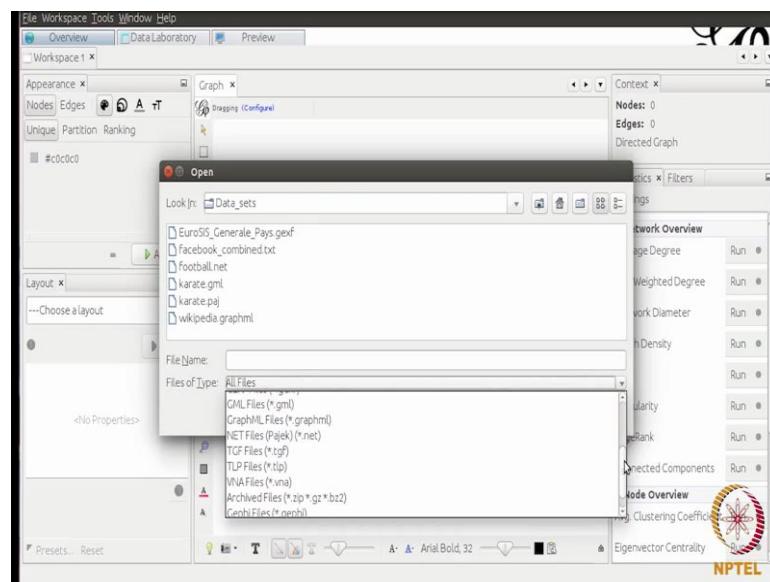
Since, I mean Gephi two to bin and then I will write .\ Gephi. So, that is how we start the software this is the interface of Gephi we are going to start a new project. So, I will click here.

(Refer Slide Time: 02:05)



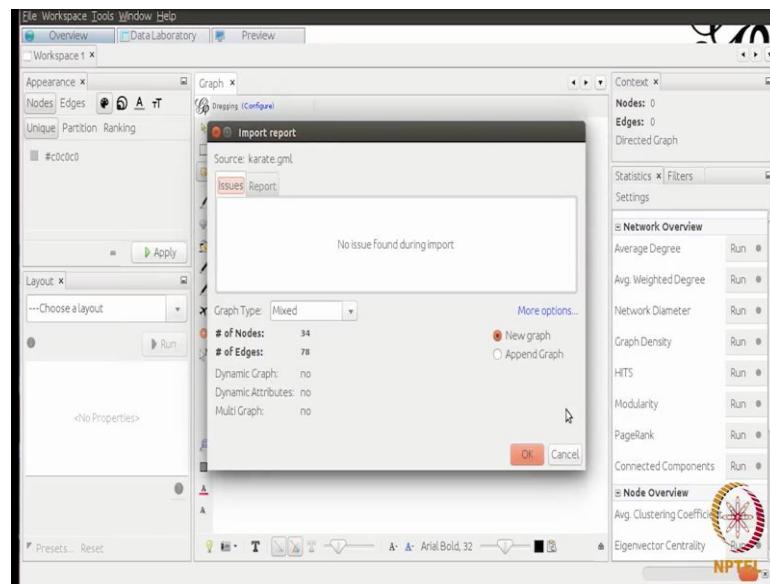
Now, this are the various options; now in order to analyze a network in Gephi we have to first open that the file of that network here. The most commonly used format by Gephi community is dot gexf. However, there are lots of other formats which are also supported in Gephi.

(Refer Slide Time: 02:29)



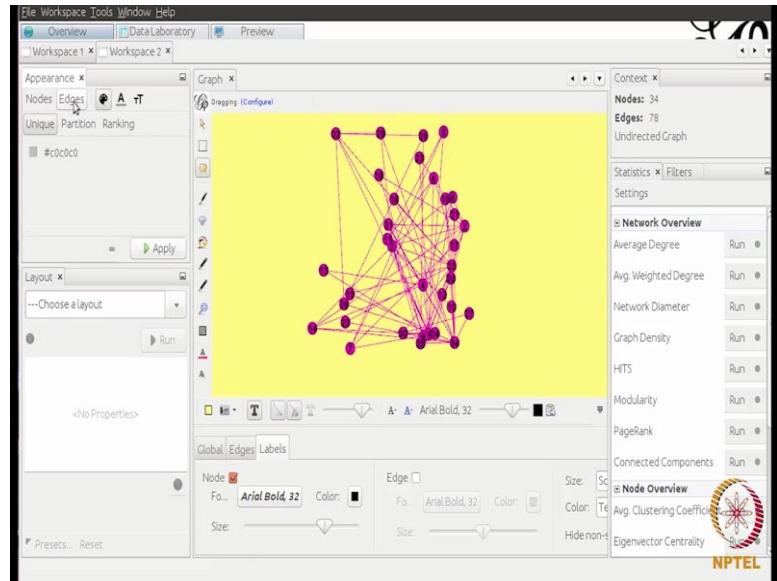
For example, let me show you when you click open here this are the formats that are supported by Gephi. As you can see there is a CSV, there is GXF there is GML GraphML L.net, Pajek files. So, number of formats are fortunately supported by Gephi; in this video we are going to take a small network in we are already downloaded this karate dot GML in the previous video. So, we are going to continue with that. So, I am selecting this file and I am opening it. So, this is what I get.

(Refer Slide Time: 03:01)



So, this is graph type which is mixed here since it is not mentioned anywhere what kind of graph it is whether it is there it is directed or undirected. Since we know that karate is undirected graph I am going to select undirected here then it showing me the number of nodes and edges and it is not a dynamic graph, and it is not a multi graph I am I want a new graph. So, I am going to click.

(Refer Slide Time: 03:30)



So, as I do that this is the graph that I get let me click here and see more options here. So, on this interface there are few options in the reference side there are few option about bottom there are few options on the right hand side, my aim is to introduce you to this software that we can use it for your analysis.

So, let me quickly start this, on the bottom you see some of the options for example you can change the background color. So, you can always change that in case you want then there is an option of zooming, let me zoom in this network. So, that I can visualize it better and then you can go to this edges tab. Here you have this pointer which you can use to in case of thickness of the edges. So, I am going to keep it like this, and then you can go to labels I am telling you the most used features out here. So, when you go to labels and if you click this node, you get the label of the nodes here and this is how you can change the size of the labels as well.

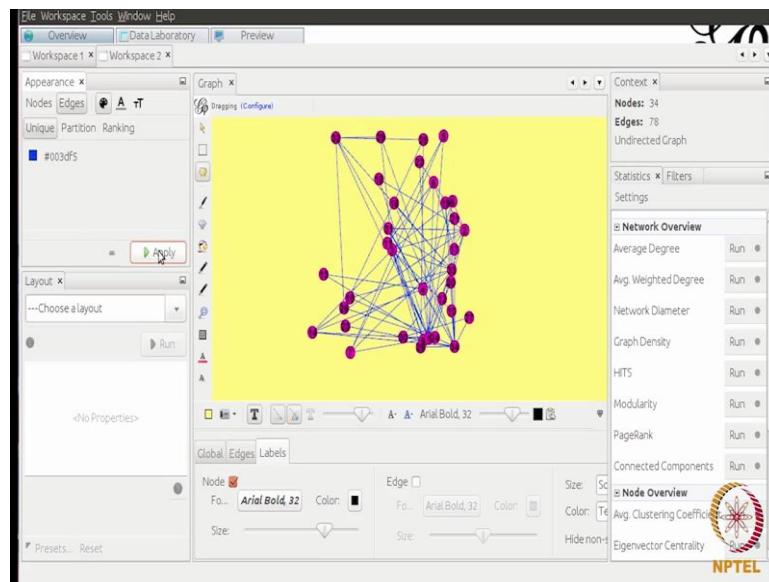
Now this is not looking so good because the size of the node is very small, we can always change that I will show you how we can do that as well and here let me show you can change you can also display the labels of the edges, in this graph there is no label sign to the edges. So, it is not going to make any difference. So, these are the main features of available of here. So, this are some short cuts that you can use for changing the thickness and this is what changing this font size of the label of the nodes. So, that is about it.

Now, let us go to the left hand side panel. So, here we have this appearance panel and under that we have nodes tab and edges tab. So, once this nodes tab is clicked we have some four options over here, the first option is for the color and second option is for the size. Since you want to change the size of the nodes so I am going to show you this first. So, right now the size is 10 let me make it 30, let us say and then I apply and this, what I see. So, labels have gone because the color of the nodes is black. So, we can change that I am going to click this color pallet.

And this is the color which is there by default we can change it let me give some color here and when I apply you see the color of the nodes as well as the edges as change by this is by default it changes the color of the edges as well, we can change that further I will show you the and this for the color of the labels let us keep it black itself as of now and this for the size of the labels.

Now, let us click the edges tab and see what options are here. So, we have the color of the edges since we have to change it, let us change it to some bluish color and apply and the color of the edges as change.

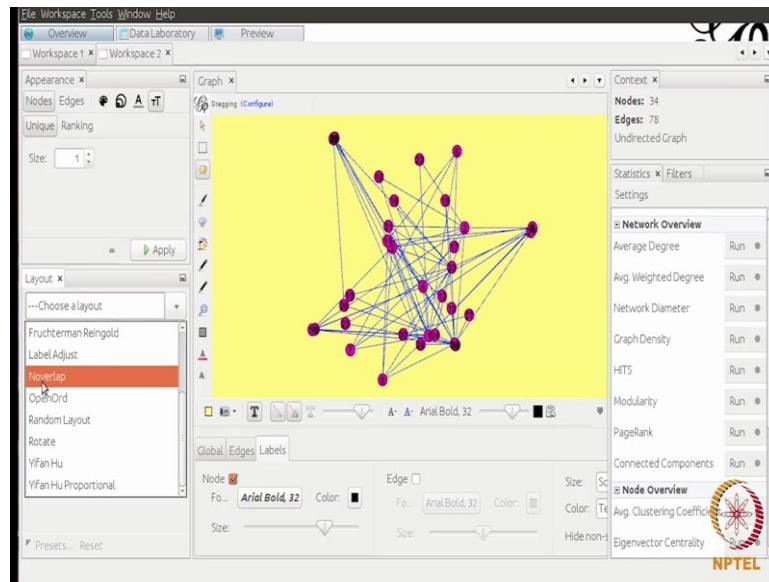
(Refer Slide Time: 06:55)



Now, this is for changing of the labels of edges since there are no labels; so would not make any difference, and this is for changing the size of the labels alright. Now in the bottom left corner out here you see an option for changing the label layouts. So, layouts basically tell us the way the node should be position and how the graph should look like.

So, there are number of layouts here and we can change it, I am you can just pair with them I am going to show you one of the layouts here which may be used full in certain scenarios. Sometimes what happens that these nodes are overlapping? So, one node is hidden behind the other so that way you cannot see them properly.

(Refer Slide Time: 07:56)



So, I am going to show you one format that can help such a scenario, before that I have to show you one of the nice features of Gephi and that is you can manually position the nodes. Now this is something that you cannot do in network x, if you have to change the positioning you have to write code for that you cannot simply do this.

So, this is one of nice features of Gephi. Now as I was telling you sometimes nodes overlap each other I am expressly over lapping some of the nodes so that I can show you functioning of one of the layouts. So, I am just expressly putting some nodes on top of the other nodes. So, let us go here and see there is one layout that is called no overlaps. So, I am taking it and I am joining it and you see the nodes have scattered, the ones which are overlapping they are no more over lapping. So, that is what is the effect of this layout. There are a number of other layouts as well so you can just read the documentation of these layouts and see what they do. Let us go to the right panel now on the top you see it just face a number of nodes number of edges and it is an undirected graph. As you go down you have a number of functions that we can apply on the given network.

Now, before we get into these functions, I want to show you two things over here. So, we are currently working in this overview tab, there are two more tab on the top left corner. So, one of them is data laboratory.

(Refer Slide Time: 09:39)

The screenshot shows the Data Laboratory interface with a 'DataTable' window open. The 'Nodes' tab is selected. The table has columns for 'Id', 'Label', and 'Interval'. Rows 1 through 22 are listed with their respective labels and intervals. Below the table is a toolbar with various data manipulation icons: Add column, Merge columns, Clear column, Copy data to other column, Fill column with a value, Duplicate column, Create a boolean column from regex match, and Create column with list of regex matching groups. The NPTEL logo is visible in the bottom right corner.

Id	Label	Interval
1	1	
2	2	
3	3	
4	4	
5	5	
6	6	
7	7	
8	8	
9	9	
10	10	
11	11	
12	12	
13	13	
14	14	
15	15	
16	16	
17	17	
18	18	
19	19	
20	20	
21	21	
22	22	

(Refer Slide Time: 09:47)

The screenshot shows the Data Laboratory interface with a 'DataTable' window open. The 'Edges' tab is selected. The table has columns for 'Source', 'Target', 'Type', 'Id', 'Label', 'Interval', and 'Weight'. Rows 2 through 12 are listed with their respective source, target, type (Undirected), id, label, interval, and weight. Below the table is a toolbar with various data manipulation icons: Add column, Merge columns, Clear column, Copy data to other column, Fill column with a value, Duplicate column, Create a boolean column from regex match, and Create column with list of regex matching groups. The NPTEL logo is visible in the bottom right corner.

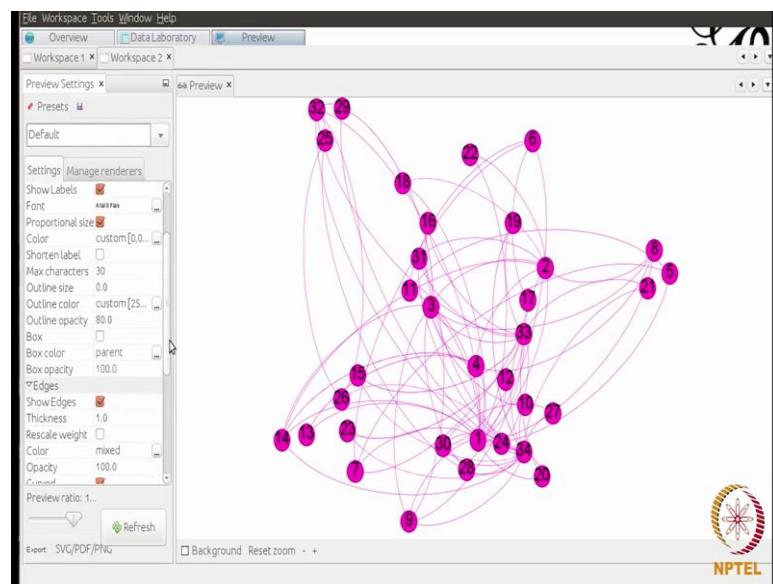
Source	Target	Type	Id	Label	Interval	Weight
2	1	Undirected	0			1.0
3	1	Undirected	1			1.0
3	2	Undirected	2			1.0
4	1	Undirected	3			1.0
4	2	Undirected	4			1.0
4	3	Undirected	5			1.0
5	1	Undirected	6			1.0
6	1	Undirected	7			1.0
7	1	Undirected	8			1.0
7	5	Undirected	9			1.0
7	6	Undirected	10			1.0
8	1	Undirected	11			1.0
8	2	Undirected	12			1.0
8	3	Undirected	13			1.0
8	4	Undirected	14			1.0
9	1	Undirected	15			1.0
9	3	Undirected	16			1.0
10	3	Undirected	17			1.0
11	1	Undirected	18			1.0
11	5	Undirected	19			1.0
11	6	Undirected	20			1.0
12	1	Undirected	21			1.0

Let us see what is there. So, here you see the id of the nodes and the label of the nodes and when you click on the edges you see all the edges in terms of source and target. So, this is the basic information about the network in the form of a spread sheet. So, there are

some other features also available over here for example, you can add new nodes here you can add edges here and you can also export this table.

So, one more feature that should be noted here that when you apply some sort of operation on the network the values that will be calculated out of that operation will also be added to this spread sheet. So, once you apply lot of operations and then you want to save the details, you can just export the table and you can make use of that table in some other tool as well. So, that is nice feature.

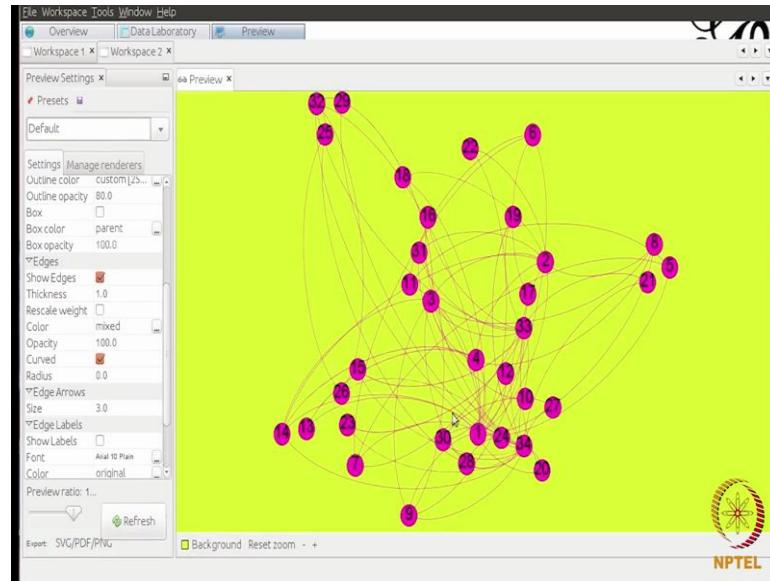
(Refer Slide Time: 10:42)



So, let us get back to the over view and let me show you the third tab is well the preview tab, when you click this and you press refresh your network appears over here with a very nice rendering, which you can download and use it the way you want. So, you can export it this formats that is SVG, PDF, PNG you can also change various properties of this network for example, if you want to show the labels you can click here and you can change the forms after clicking it you have to press refresh, now you see the labels here you can also change the font.

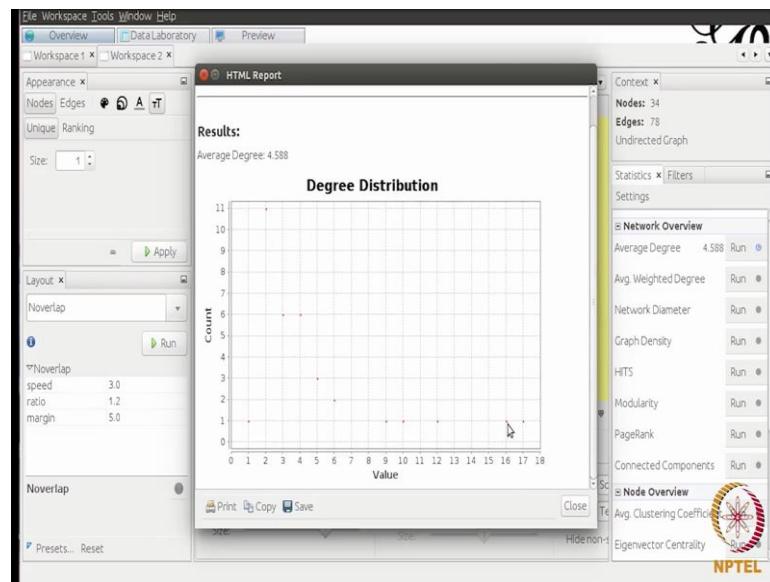
For example; let me reduce the forms to 8 because it is not looking so nice here let me press refresh and label as reduced. So, you can change it the way you want there are lot of other features as well for example, you want to show the labels or show the edges or not of course, you want to show the edges you can just play around this features and you can also change the back ground color let me just show you this features.

(Refer Slide Time: 11:48)



So, this is how you can play around here and then you can download it and you can use it. Let us go back to the overview tab and we were checking the functions that Gephi provides here. So, we have this function average degree let us click on run.

(Refer Slide Time: 12:09)

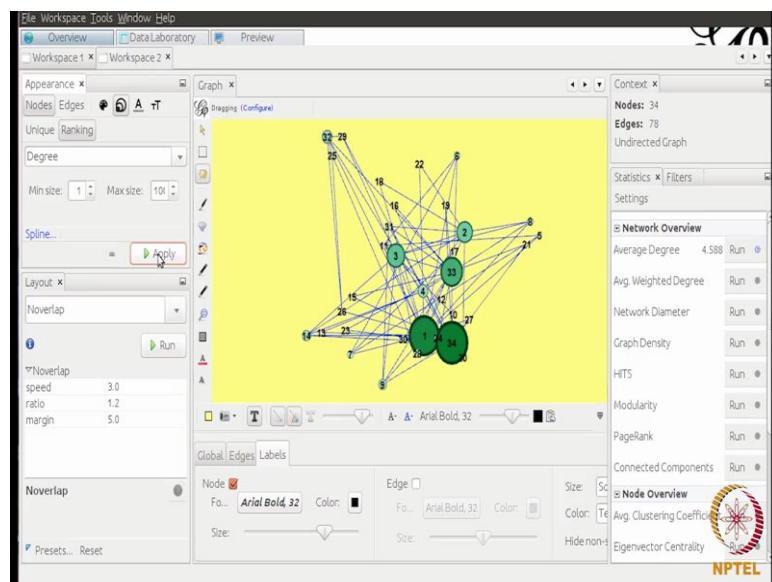


So, just like a button and the average degree and number of other things will be calculated. In this network average degree is open filed and it also given the degree distribution automatically as you can see this is sort of power loss plot as we (Refer Time: 12:26) previous video as well. Now you can save this plot you can copy it you can

print it. So, it is up to you want to use this information. Once you calculate the degree there are a couple of more features that you can see on the left-hand side, now let me show you that. So, here nodes tab is clicked, and you see one more option here that is ranking.

So, sometimes what we want; we want to change the appearance of certain nodes based on certain properties for example, we want that nodes which have high degree should appear little different be it in terms of color or be it in terms of the size of that node. So, such features are easily available in this software. So, we have this node clicked here and let me click this color here, and when I clicked the ranking here it is asking me how we want to rank the nodes.

(Refer Slide Time: 13:31)

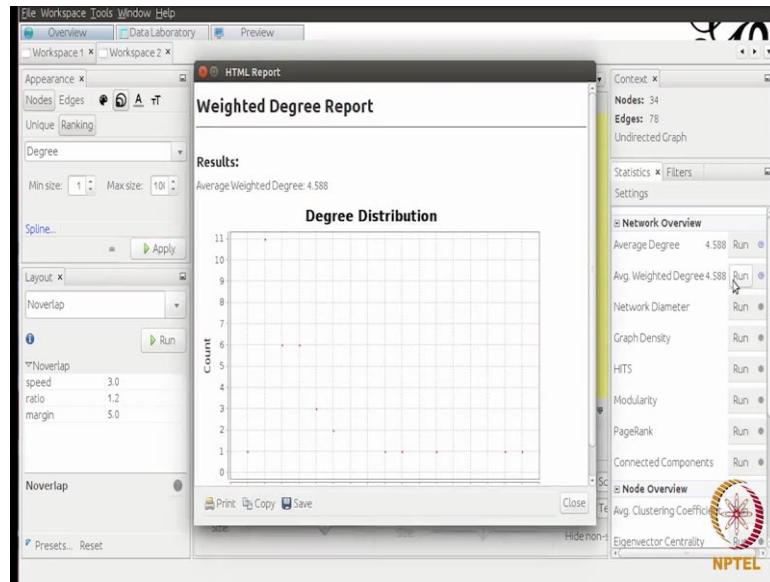


Now, I selected degree here, after this when you press apply you see the color of nodes as change the nodes which have high degree or darker in color and nodes which have less degree are lighter in color. So, you can quickly see by looking at the network you can see which are the nodes that are having high degree. So, this is one visualization feature, similarly if you want that the nodes which have high degree should appear bigger in size, you can also do that. So, I am clicking this button here and I will. So, as of now the size is unique that is 30 right.

So, we want to change it based on degree. So, I will select degree here and the minimum size is one and maximum size is 100, I will go with a head with that press apply. So, this

is what you see the size is of the nodes as change based on the degree. As you can see there are two nodes two three nodes which have very high degree compared to the other nodes, so according the size of the nodes as change. So, this is one another nice visualization features so that is about degree, let us check out more features over here we have average weighted degree.

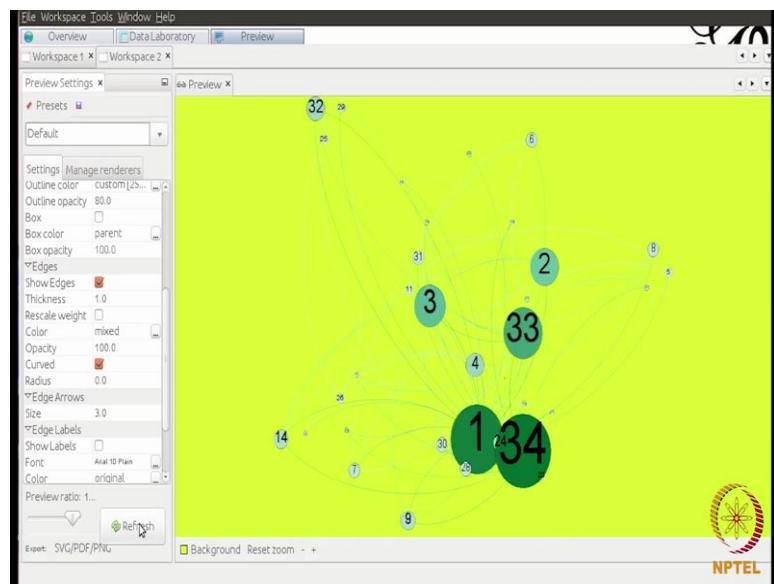
(Refer Slide Time: 14:47)



Since this graph is unweighted. So, this is now going to make difference. So, we getting same average degree here, this is because the weight is one for all the edges. We also have a number of other functions here for the example diameter we have density, where modularity, page rank, connected components clustering coefficients. So, when learn the other concepts in the throughout the course, you might want to come back here and then analyze your network based on that parameters that we have learnt. So, that will be nice because now you see what all features this software provides.

So, as when you learn about that feature that you can come back here and import here graph file and then see the features so that will be a good start for you. Let us go back here and as I want to show you one more thing in detail laboratory, since we calculated degree of the nodes that degrees as come over here. So, another column as be added over here where degrees of the nodes have been displayed. So, you can export this table and you can use it the way you want if you go the preview tab again if you press the refresh button you get that network over here.

(Refer Slide Time: 16:04)



So, this is the final rendering of the network and from the left-hand side panel you can change the various properties of this network. So, I think that was pretty much basic introduction to Gephi software and may be a good starting point for you.

Thank you.

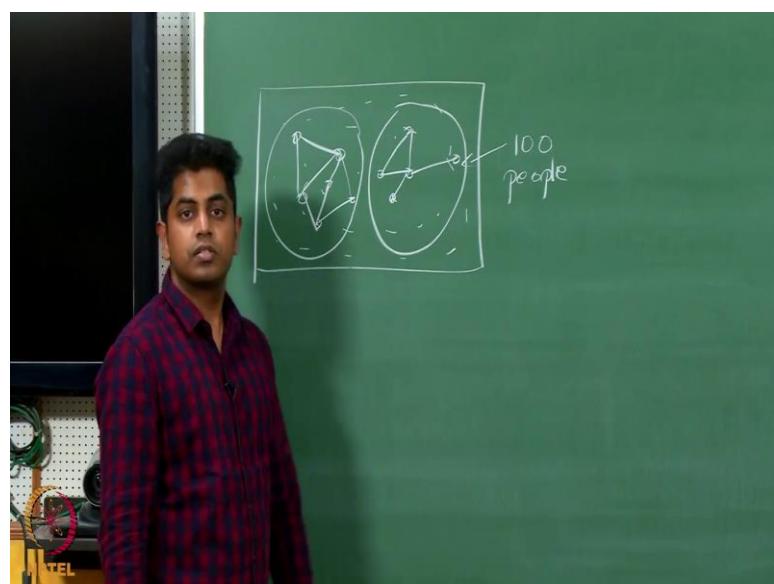
**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture – 23**  
**Handling Real-world Network Datasets**  
**Introduction: Emergence of Connectedness**

So, we saw an introduction to datasets, importance of crunching datasets and thankfully as I said we are in era where lot of datasets are available when we can try to make sense out of it. But there are times when we may have to not worry so much about data sets to unravel some mysteries of networks in general. I am going to teach you right now in the forth coming few lectures on how one can actually keep a side data sets and look at graphs in general and make sense out of it.

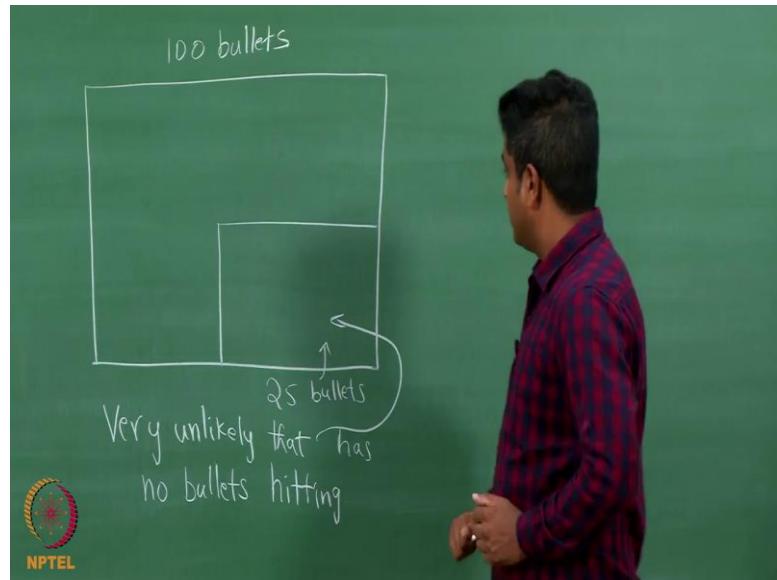
So, what will do is will revisit or age old puzzle which we have been talking from a long time of how does a graph become connected, why is it that almost always a graph is connected in the real world. Given a class room of 100 people as friendships start happening when will you observe the graph becoming connected, will it at all become connected, what if there is a bunch of people this side a bunch of people this side such that nobody knows each other across where friendships only happen within, what do I mean by this? By this I mean following.

(Refer Slide Time: 01:31)



Assume this is my class room and a class room full of some let us say 100 people, let me call this 100 people and you look at the friendships between them some random friendships like this friends with him, friends with him, friends with him etcetera. Will you see a bunch of people this side a bunch of people this side where connections are only within, there are no connections across can this even happen? Let me give you an intuition.

(Refer Slide Time: 02:23)



By looking at a similar example look at this assume I am giving you a shooting target by a shooting target I mean I will give you a gun and ask you to shoot within this square only within this is rectangle only you can go on shooting as many bullets as you want right, but then I will give you one quadrant of this rectangle let us say this is one-fourth of the total rectangle correct, in this one-fourth of this rectangle small rectangle your bullet will definitely fall unless you aim really well you are a great archer, you may not throw your arrows only in a particular place.

The assumption here is that you are shooting uniformly random your blind folded and simply shooting and all your bullets assuming here coming within this rectangle I will ask you how many bullets came here, let us say you shoot 100 bullets, how many of those bullets came and fell here? If you are shooting uniformly it random is not it very obvious that this captures one-fourth the total area. So, one-fourth the bullet should come and fall here right. If you are shooting 100 bullets 25 bullets should fall inside can it. So,

happen its very straight forward and intuition for all of us can it. So, happen that none of the 100 bullets fell inside this and all of them fell outside this, let me remind you, you are not shooting it with any intend you are uniformly at random shooting at it, your blind folded you cannot see where to shoot you just hold a rifle and keep shooting. It is very unlikely that no bullets come inside.

Let me write that down very unlikely that this place has no bullets at all, no bullets hitting is not this very unlikely, it is very unlikely correct. Now, if you have this intuition I am going to ask you a question on this it is exactly the same as this observation, what do I mean by that?

(Refer Slide Time: 05:12)



If I take 100 friends, 100 people they are not friends 100 people - let me say 70 people are this side, 30 people are this side totally there are 100 people, total possible friendships, total possible friendships is clearly  ${}^{100}C_2$  as you all know  ${}^{100}C_2$  which is roughly  $100^2/2$ ,  $100^2/2$ , it is actually  $(100 * 99)/2$  for you know from day one I have been writing it as square by 2 because 99 can be seen as 100 its almost all close to 100. So, there total number of friends which is  $100^2/2$  which we all know is 5000 these are the total possible friendships out of which there are some friendships within, within there are some friendships across. Let me see how many friendships are across, across friendships are simply  $70 * 30 - 2100$  right. What do I mean by this? Let us take a pause and

understand what I have done. So, far total possible friendships are 5000, but the friendships across are 2100 in number.

What I am saying? Here is the situation where there was a possibility for people to make 5000 something friendships and every single friendship that they choose did not fall into this, did not fall into the across friendship, there were so many possible ways in which they could have had an across friendship they did not, all of them fell within only this is very unlikely.

The intuition is this do you see this is roughly half of this, this is roughly half of this. There are let us say twenty balls in a basket 10 black, 10 white you pick some balls from this basket all of them are black, you are picking them uniformly it random and all of them are black is this even possible, it is not possible, it is not impossible it is improper. So, I repeat if you take a bunch of 100 people and look at friendships it is very unlikely that there are no friendships across because a whole lot of possibilities for that to happen and you are saying none of this possibilities happened.

So, what did I just say? Let me connect the analogy properly when you start putting a lot of edges it is as good as you choosing the edges out of 5000 edges it choosing some edges, half the edges are between and when you are choosing none of these edges are chosen, half the edges are across between you choose not to touch even one of them and you are picking the edges look at the analogy I gave you of the basket and balls. There is a basket full of balls half of them are black half of them are white, you blind fold yourself and then pick and you see there all the balls that you are picking are all black. It is unlikely when you are blind folded you should pick uniformly it random white as to come somewhere. So, the black balls all the balls here or all the edges here, the white balls are the edges across, and the black balls are the edges within. Edges across are roughly half the total number of edges, black balls are half the total number of balls, white balls is half the total number of balls, but you when you are picking you are only picking black balls you are not picking white balls and your blind folded look this is very unlikely.

And I am just using argument to say this is very unlikely that you have two components most of them are out of 5000. Half of them are across and you are not choosing even one of them right. Same thing here one-fourth of the place is this part and all your bullets are

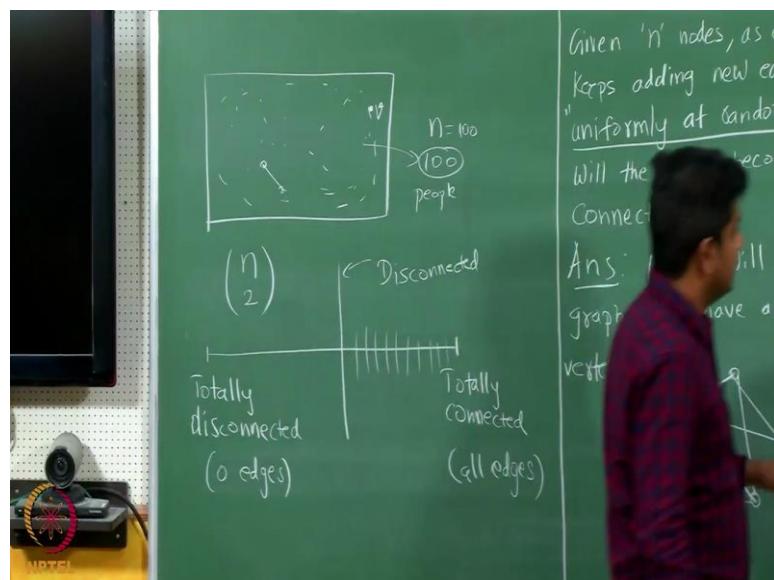
not falling here its falling out side this only that is unlikely some bullets should fall here, some balls should be white, some edge should be across that is the intuition correct. When you create a graph, you cannot see two compartments like this when you have sufficiently many edges by sufficient many edges I mean when you pick one ball it can so happen that it is black only. Second ball can be black only when you pick some 50 balls all of them being black is very unlikely. So, when you put a lot of edges all these edges being within this only is very unlikely.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture – 24**  
**Handling Real-world Network Datasets**  
**Advanced Material: Emergence of Connectedness**

Let us see something nice and deep with some mathematical analysis, let us continue with our question of when exactly we see a connected graph as we keep putting edges.

(Refer Slide Time: 00:21)



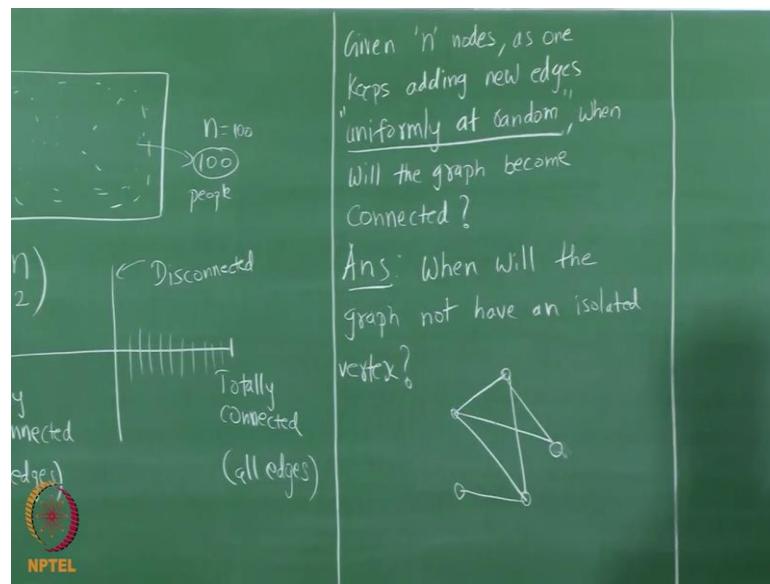
So, the question is this, the same question that we been discussing so far, assume there are 100 people or let us say  $n$  people  $n = 100$  here, a particular example there are  $n$  people here, if you put all the edges what is all the edges?  ${}^N C_2$  or let us say  ${}^{100} C_2$ , if you put all the edges; obviously the graph becomes connected.

Now, when you put all the edges and remove just 1 edge it remains connected. If you keep plugging out edges 1 at a time from this complete graph of  ${}^N C_2$  edges, there is a time when it becomes disconnected. I repeat take 100 nodes put all possible edges between any 2 nodes, which happens to be 100 choose 2 and from this you start removing 1 edge at a time remove and then throw it, keep doing this you will finally, get a disconnected graph somewhere at some point. The question is the investigation of when exactly you will get disconnected graph as you keep removing edges.

So, complete graph no completely disconnected graph; complete graph means all edges completely disconnected graph means absolutely no edges, as you keep removing edges you come closer and closer to this, there is a point when the graph becomes disconnected right. So, I am going to use the word totally disconnected which stands for no edges at all and then I am going to say totally connected which means it is a complete graph. 0 edges no edge at all, all edges  ${}^N C_2$  edges.

As you keep removing edges you will observe that there is a point when the graph becomes disconnected, what is that point should you remove half the edges or only a few edges when removed will make the graph disconnected or should you remove a lot of edges? This question is same as take no edges keep putting 1 edge uniformly at random, when exactly will the graph become connected. So, we will write this down for clarity sake, given n nodes as 1 keeps adding new edges.

(Refer Slide Time: 03:26).



I use the word uniformly at random by this I mean, I do not have any preference for the edges I keep putting edges uniformly at random randomly basically. As I keep putting adding edges uniformly at random, when will the graph become connected? Now a question like this does not have an answer, we always say mostly if you put so many edges becomes connected; whatever mean what do you mean this? If you say I need to lose weight, I need to lose 5 kilo grams of weight, you will say then you start running half an hour a day for the next 1 month, this is just guess work and you will say an

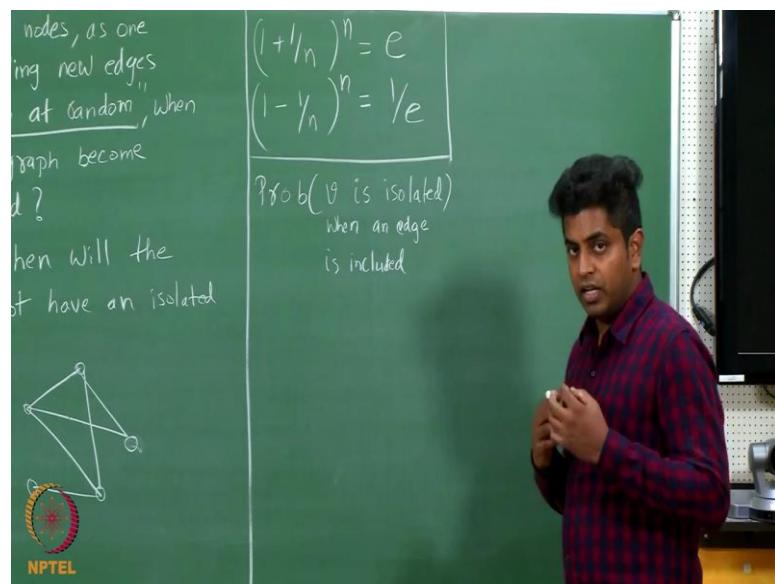
average if you run half an hour day in let us say 1 month time, you probably lose 2 to 3 kilograms of weight I am not a dietitian I am not a exercise expert, I am just telling you an example right.

Similarly, I am asking this question as you keep putting edges what do you guess? Will be the number of edges it takes for you to make the graph become connected. Let us answer this question with a small observation. I will answer this question by asking this question when will the graph not have an isolated vertex. To answer this question I am asking another question, I will ask you this question as you keep putting edges when is the time you do not have even a single point without edges, what do you mean by that?

So, let me take 1 2 3 4 5 vertices here, as you know how many possible edges can put; little bit of observation tells you 10 edges totally. Let me put 1 edge when I put 1 edge this is isolated, this is isolated, this is isolated, this was isolated does not continue to be isolated because you have put an edge. Now I will put another edge let us say this there are 2 isolated vertices now, still 2 isolated vertices only 1 isolated vertices vertex only 1, none there are no isolated vertices right now correct? After putting 1 2 3 4 5 6 edges isolated vertices came to an end, but then I should be putting this edges uniformly at random, I should not try to put edges just in order to make the graph appear connected without any isolated edges.

Please note you cannot have isolated vertex sorry; you cannot continue to have an isolated vertex by simply putting all possible edges within the remaining vertices, not touching our vertex that is all possible. But the question is uniformly at random when you start putting edges, when does the graph become connected? Let us answer this question.

(Refer Slide Time: 07:46)



So, we have all seen this result which says  $1 + 1/n$  whole to the  $n$ , is as  $n$  become bigger and bigger, this quantity is between 2 and 3 this is called  $e$  we have all seen this, it is some basic calculus limits basically. As  $n$  increases  $(1 + 1/n)^n$  becomes  $e$  for example, if  $n$  is 10,  $1 + 1/10$  which is 1.1 to the power of 10, 1.1 to the power of 10 is around 2.7 which is  $e$  value.  $1 + 1/20$  whole to the power 20 is roughly equal to a number between 2 and 3.

So, it is observed that this is the limit of this is a constant. Little bit of observation says we have all observed this are 11 12th standard mathematics, this is  $1/e$  do not worry much if you we recollect this things or I am trying to say is  $(1 + 1/n)^n$  as  $n$  becomes bigger and bigger becomes a quantity between 2 and 3 which is  $e$ ,  $(1 - 1/n)^n$  let us say  $n$  is 10 then this becomes  $1 - 1/10$  which is 0.9, 0.9 to the power of 10 is 1 over 2.7. As  $n$  is very bigger it becomes very accurate now we are going to use only this 2 things ok.

What is the probability let us get back to the question, what is the probability of you having a vertex  $v$  becoming isolated? When an edge is included there are  $n$  nodes you put 1 edge and  $v$  a vertex  $v$  is isolated, what is the probability what you I mean by this? Look at this, here are 100 nodes this is my vertex  $v$ , and we put 1 edge what is a probability that  $v$  continues to be isolated when you put 1 edge. In this big a graph when you put an edge where there no edges are put an edge, it is very unlikely that this edge

will fall on v, it will be some alpha and beta some 2 vertices it may not be b v, what is the probability that is v?

There are 100 people v is one of them, you choose 2 people to put an edge, what is a probability that it is v? It is 2/100 simple observation, it may be pause and thing for a minute I am going to I am assuming that you know that is 2/100. So, when I take when I put an edge and that edge not being on v on a graph with n vertices is 2/n.

(Refer Slide Time: 11:01).

Chalkboard content:

$$\left(1 - \frac{1}{n}\right)^n = \frac{1}{e}$$

Prob(v is isolated)  
when an edge  
is included

$$= \frac{2}{n}$$

Prob(v not isolated) =  $\left(1 - \frac{2}{n}\right)$

Prob(v not isolated after K edges being put) =  $\left(1 - \frac{2}{n}\right)^K$

As I keep doing this you know the probability of me the probability of v having rain today if it let us say 1 by 4, the probability of not raining today is 3 by 4 correct. So, let me say probability of v not being isolated is  $1 - 2/n$ , when you put 1 edge. You see probability of it raining today if its 1 by 4, probability of it not raining today is 3 by 4. The probably the probability of not raining today and tomorrow is 3 by 4 into 3 by 4, the probability of not seeing any kind of rain happening today tomorrow day after is simply product of 3 by 4, 3 by 4, 3 by 4 again straight forward concept of probability right.

So, probability of v not being isolated after K edges being put is,  $1 - 2/n$ , times  $1 - 2/n$ , times  $1 - 2/n$  so on K times right so on K times. So, we saw that the probability is  $1 - 2/n$  whole to the K, the power probability of v not being isolated after K edges being put.

(Refer Slide Time: 13:06)

$$\begin{aligned}
 K=1 \quad & \text{Prob}(v \text{ not isolated}) = 1 - \frac{2}{n} \\
 K=2 \quad & \dots \dots \dots = (1 - \frac{2}{n})^2 \\
 K=n \quad & \dots \dots \dots (1 - \frac{2}{n})^n \\
 \text{When we include 'n' edges,} \\
 \text{Prob}(v \text{ not isolated}) &= (1 - \frac{2}{n})^n \\
 &= \left( \left(1 - \frac{1}{\frac{n}{2}} \right)^{\frac{n}{2}} \right)^2 = \left( \frac{1}{e} \right)^2 = \frac{1}{e^2}.
 \end{aligned}$$

Now, let me ask this question when  $K = 1$ , the probability of  $v$  not being isolated is  $(1 - 2/n)^K$ ,  $K = 1$  correct. If  $K = 2$  this will be  $(1 - 2/n)^2$ . If  $K = 3$  it will be  $(1 - 2/n)^3$ , if let us say  $K = n$  then this is going to be  $(1 - 2/n)^n$ , observe this let me observe this what am I saying. When  $v$  include  $n$  edges which means  $K = n$ , the probability of  $v$  not being isolated is  $(1 - 2/n)^n$ , which is  $((1 - 1/(n/2))^{n/2})^2$ , 2 goes down and becomes  $n/2$ , whole to the power of  $n$  I will do a small find tuning here.

I divide this by 2 and multiply it by 2, usual trick we do in mathematics. If something does not look nice you make it appear nice by multiplying something on both sides multiplying and dividing by the same number adding something on LHS and RHS we know those tricks. So, I am writing it. So, that this appears like sometime that we know already, in place of  $n$  we have  $n/2$  1 minus some 1 over something to the power of the same thing is 1 by  $e$ . So, this is  $1/e$  which means is equal to  $1/e$  this  $e$  is  $1/e$ ,  $1/e^2$ , what I mean by this?  $E$  is a number between of 2 and 3.

So, let me say  $e$  is 2, if  $e$  is 2 then the probability of  $e$  is not being isolated is 1 by 4; is a quarter probability that pause and think what I am saying there is a quarter probability 1 over let us say  $2^2$  or  $2.7^2$  actually whatever a constant, there is roughly quarter probability that when you put  $n$  edges on  $n$  vertices you put  $n$  edges, there is some decent probability 1 by 4, but your  $v$  is not isolated it is a very small probability, but then I will do a small trick, I will take  $K = n * \log n$  ok.

(Refer Slide Time: 16:21)

NPTEL

$$P(v \text{ not isolated}) = 1 - \frac{2}{n}$$

$$= (1 - \frac{2}{n})^n$$

We include edges,

$$K = n \cdot \log n$$

$$\text{Prob}(v \text{ not isolated}) = \left(1 - \frac{1}{\frac{n}{2} \log n}\right)^{\frac{n}{2} \log n}$$

$$= \left(\frac{1}{e^2}\right)^{\log n} = \left(\frac{1}{e^{\log n}}\right)^2$$

$$= \left(\frac{1}{n}\right)^2$$

Prob of v becoming isolated after including  $n \log n$  edges is

$$(1 + \frac{1}{n})^n = e$$

$$(1 - \frac{1}{n})^n = \frac{1}{e}$$

Prob(v is isolated when an edge is included) =  $\frac{2}{n}$

Prob(v not isolated edges being) =  $\sqrt{\frac{1}{10,000}}$

100 node  
v is isolated after 100 Log<sub>100</sub> edges

Do not ask me why I am taking  $n$  equal to  $\log n$ , just for fun you can take whatever you want alright there is a reason I am taking this way, but for the time being assume I am taking slightly more than  $n$  which is  $n \log n$ , if  $K = n \log n$  the probability of  $v$  not being isolated happens to be.

Let me rewrite this  $1 - 1/(n/2)$  this thing, instead of  $n$ , I put  $n \log n$  here,  $n \log n$  here,  $n \log n$  here and here this becomes same thing I repeat 1 divided by  $n/2$  whole to the  $n/2$  times  $\log n$ , we know this part is we saw its  $e$  correct. So, there is a 2 here, 2 here this becomes  $1/e^2 * \log n$ , which is  $(1/e^{\log n})^2$  I just take a log here and put 2 that side, which is equal to 1 over we know  $e^{\log n}$  is  $(1/n)^2$ . Let me take a minute and stare at this and observe what just happens. The probability of a vertex  $v$  becoming isolated after including  $n \log n$  edges is this. I repeat the probability for vertex  $v$  becoming isolated after including  $n \log n$  edges is  $1/n^2$ ; which means if you take a graph with 100 vertices and if you only put 100  $\log 100$  edges in that graph, then you will find a vertex  $v$  isolated with 1 over 100 square probability. So, by that I mean 100 node network  $v$  is isolated after putting 100  $\log 100$  edges, the probability for this to happen is 1 over 10,000 is what my observation says.

My observation says when you take 100 nodes a vertex  $v$  is isolated with very small probability if you put 100  $\log 100$  edges, which means when you take a graph  $g$  with  $n$  nodes you take a graph  $g$  with  $n$  nodes by just putting  $n \log n$  edges, with a very high

probability you will not have any isolated vertices; not having isolated vertices does not guarantee you of a connected graph, but then not having an even isolated vertex, but the graph is disconnected is very unlikely.

I told you before correct you cannot have 2 components; you cannot even have 1 disconnected vertex because it is very small probability. So, this tells you that as you keep putting more and more edges very soon the graph becomes connected; when you put  $n \log n$  edges the graph with a very high probability does not have any isolated vertices, does not have any isolated components there is no isolated vertex having an isolated component is very unlike correct. With a very high probability the graph is connected when you put just  $n \log n$  edges.

So, we saw if 1 puts only  $n \log n$  edges in a graph, the graph becomes connected. You see what  $n \log n$  is? given a graph with let us say 1 million nodes, 1 million times  $\log 1$  million so many edges suffice to make the graph become connected. Point to note  $\log n$  means number of digits in  $n$ . So, you are not multiplying  $n$  by a big factor here when I say  $n \log n$ ; by  $n \log n$  I mean something almost as good as  $n$ . So, if you have  $n$  number of vertices by putting just a few edges as many as the vertices itself,  $n$  times  $\log n$  you achieve connectedness.

Now, we saw the theoretical framework for it, do not worry much about all the proof that I gave you it is not especially important I just gave it for completeness sake. All that you need to know is the following let me go slowly, all that you need to know is this. Given  $n$  nodes you keep putting edges you keep putting  $n \log n$  edges, then the graph becomes connected; how true is this statement. Whatever I wrote on the board may not be true the mathematical statements I have derived is in no way a validation for what is going to happen if I am going to program and see it maybe or may not be, I do not know I am at least not convinced. So, let us do one thing, let us take our laptops and try to write piece of code and then check if it is true or not.

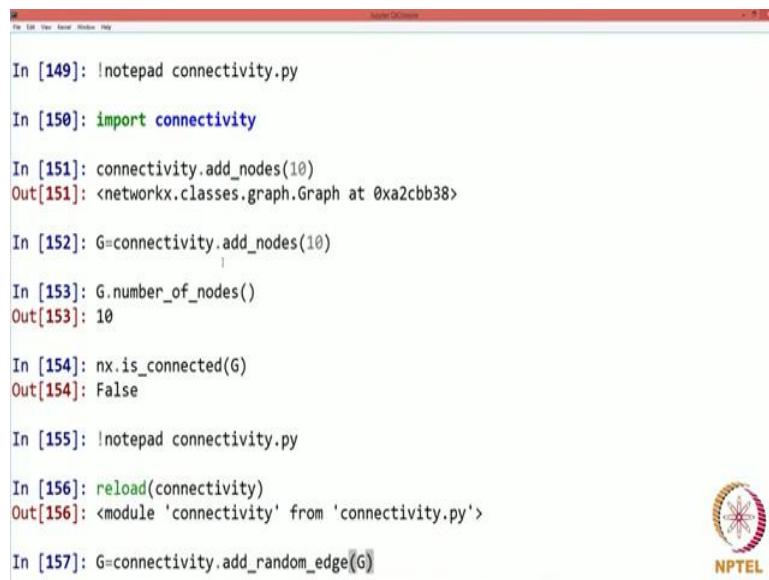
**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture – 25**  
**Handling Real-world Network Datasets**  
**Programming Illustration: Emergence of Connectedness**

We just now looked at a very surprising and interesting result which says that on a graph if we have  $n$  nodes then one just needs  $n \log n$  number of edges to make this graph connected that is if we take  $n$  nodes and we randomly put  $n \log n$  edges in this graph, the graph becomes connected. So, you can start at any node in this network and from this node you can find a path to every other node in the network.

Now, we will be doing a screen cast for it we will be writing a piece of code in which we will take random graphs of different sizes and then we will look at the number of edges required to make this graph connected. So, we will do this, we will look at the plot, let us get started.

(Refer Slide Time: 00:50)



The screenshot shows a Jupyter Notebook interface with the following code history:

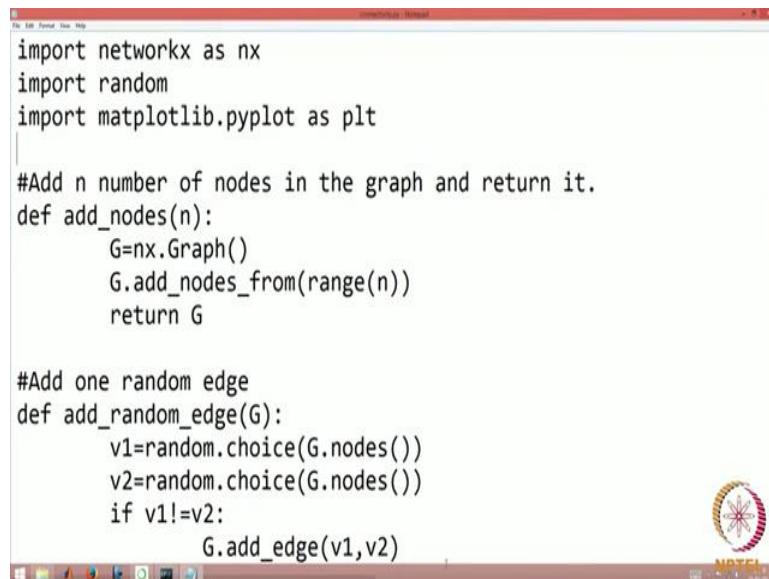
```
In [149]: !notepad connectivity.py
In [150]: import connectivity
In [151]: connectivity.add_nodes(10)
Out[151]: <networkx.classes.graph.Graph at 0xa2cbb38>
In [152]: G=connectivity.add_nodes(10)
           |
In [153]: G.number_of_nodes()
Out[153]: 10
In [154]: nx.is_connected(G)
Out[154]: False
In [155]: !notepad connectivity.py
In [156]: reload(connectivity)
Out[156]: <module 'connectivity' from 'connectivity.py'>
In [157]: G=connectivity.add_random_edge(G)
```

The notebook is running on a system named "Slicer-Opti". In the bottom right corner, there is an NPTEL logo.

Let us start from very basic coding. So, what we want to do is we want to have some nodes in the graph and then we add random edges in this network one at a time and our aim is to see after how many edges does this network becomes

connected. So, we first create a file for this. So, let us open a new file in notepad let us name it as connectivity.py.

(Refer Slide Time: 01:26)



```
import networkx as nx
import random
import matplotlib.pyplot as plt

#Add n number of nodes in the graph and return it.
def add_nodes(n):
    G=nx.Graph()
    G.add_nodes_from(range(n))
    return G

#Add one random edge
def add_random_edge(G):
    v1=random.choice(G.nodes())
    v2=random.choice(G.nodes())
    if v1!=v2:
        G.add_edge(v1,v2)
```

So, here we have a file oh we first of all import our basic packages which we will need import networkx as nx see I think it is sufficient for the time being. So, what we do? We first define a function to add some specified number of nodes in the network. So, the function already exists in networkx you all know, but just for the sake of simplicity and clarification I will again make a function here and let us name this function as add nodes and you pass a number n as a parameter to this function. So, what this function is going to do is it will create a graph  $G = nx.Graph$  and add n number of nodes in this graph. So, we can do  $G$  dot I hope you remember the function add nodes from. So, this function adds the nodes in the graph from a list and the list we give here.

So, the list is from number 0 to number  $n - 1$ . So, n nodes get added to this graph and we finally, return the graph simple function we will just keep commenting everything side by side. So, we comment it add n number of nodes in the graph and return it perfect. So, we will save it and let us come back to our window.

So, what we need to do now is import this file we import connectivity. So, this gets imported let us call a function to call a function we have the syntax connectivity dot function name our function name is add underscore nodes and me and we had let us say we want to add 10 nodes in the graph you execute it. So, we get a graph here. So, our

function return the graph. So, let us collect the output in a graph name G. So, `G = connectivity.add_nodes(10)` now I want to see whether 10 nodes have been actually added in the graph or not. So, we can see that using the function G dot number of nodes you see that there are 10 nodes in this graph next, we will write a function for adding one random edge to this network.

So, if you see currently in this network there are just 10 nodes there are no edges. So, this network is disconnected completely disconnected. So, to check that we have command `nx.is_connected` let say you want to see whether this graph is connected or not let us call. So, before using `nx.is_connected` parse graph G here which shows us false if the graph is not connected now one by one, we add edges to this network. So, first we write a code for adding one random edge to this network we go back to our file which was `connectivity.py` and now we add second function here and the function is for add one random edge in this network and one random edge.

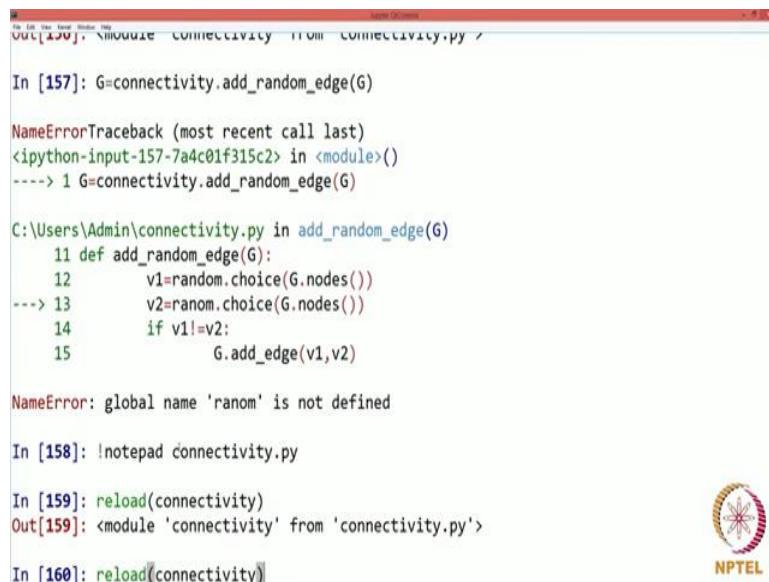
Let us name the function as add random edge and it takes us input the graph G graph G which has been already created with no edges. So, we pass that graph G actually you can pass a graph G with the edges also that is up to us. So, the function takes as input one in graph and adds one extra random edge to this graph define add underscore random edge G, now we need 2 random vertices from the network. So, how do we pick a random vertex? So, to put to pick a random entry we need one package name random. So, we have discussed about it before in the first week what does it do, we are going to use the function `random.choice` first vertex v1. So, this is the first random vertex which we want to pick it is `random.choice` function takes as the input one list and outputs as one random value from that list.

So, the list which we are going to pass here is a `G.nodes` that is the list of nodes in the network. So, v1 gets first random node from this network let us choose another node v2 v2 equals to another random node `random.choice` and again we have here `G.nodes` we got 2 random vertices now and we want to create an edge between them, but there might be a case where v1 and v2 might be same nodes. So, we would have picked both the random nodes to be same although that is a very rare case that might happened, but we do not want that to happen because we do not want any self loops in our graph. So, we just put a condition here if `v1 != v2` only then we add an edge that is these 2 vertices

should be separate. So, if  $v_1 \neq v_2$  we go ahead and add an edge  $G.add\_edge$  from  $v_1$  to  $v_2$  after adding this edge we return a graph  $G$  save it we come back to our console.

Let us now we not need to reload our file. So, we reload connectivity since we have made changes to it reload connectivity now the spelling is correct reload connectivity. So, it is reloaded we already have a graph here which has 10 nodes. So, what do we do now is  $G$  equals to connectivity dot and we call our function add random edge on the graph  $G$  itself seems to be some problem?

(Refer Slide Time: 08:07)



```
In [157]: G=connectivity.add_random_edge(G)
NameError: Traceback (most recent call last)
<ipython-input-157-7a4c01f315c2> in <module>()
      1 G=connectivity.add_random_edge(G)

C:\Users\Admin\connectivity.py in add_random_edge(G)
     11     def add_random_edge(G):
     12         v1=random.choice(G.nodes())
--> 13         v2=ranom.choice(G.nodes())
     14         if v1!=v2:
     15             G.add_edge(v1,v2)

NameError: global name 'ranom' is not defined

In [158]: !notepad connectivity.py

In [159]: reload(connectivity)
Out[159]: <module 'connectivity' from 'connectivity.py'>

In [160]: reload(connectivity)
```

So, it is a name error which says that global name there is a spelling mistake. So, let us go back and correct it if we see here yeah just small spelling error, we correct it save it come back reload it and call the function again it works fine. So, we have we now one random edge should have been added in a graph.

(Refer Slide Time: 08:36)



```
In [15]: G.add_edge(v1,v2)
NameError: global name 'randon' is not defined

In [158]: !notepad connectivity.py

In [159]: reload(connectivity)
Out[159]: <module 'connectivity' from 'connectivity.py'>

In [160]: G=connectivity.add_random_edge(G)

In [161]: G.edges()
Out[161]: [(0, 7)]

In [162]: !notepad connectivity.py

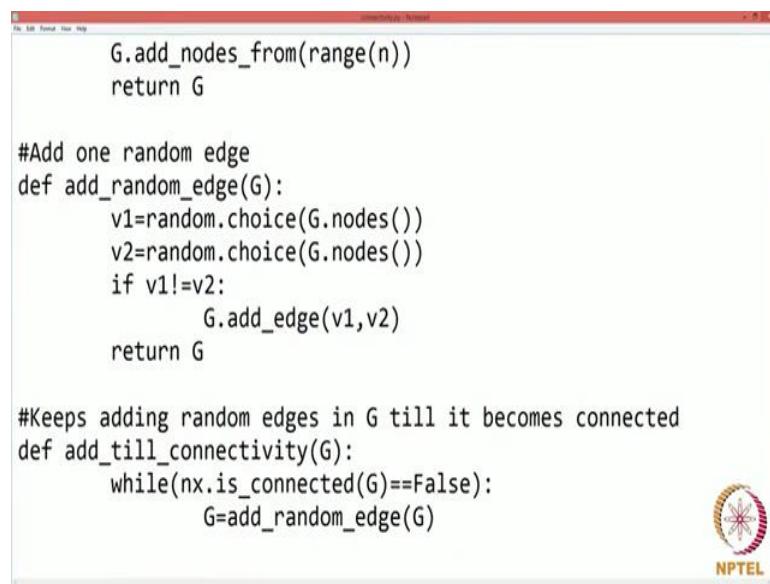
In [163]: reload(connectivity)
Out[163]: <module 'connectivity' from 'connectivity.py'>

In [164]: G=connectivity.add_till_connectivity(G)
           |
In [165]: G.edges|
```



So, let us see it let us look at the list of edges in a graph. So, we do `G dot edges` and we can see here that a random edge and edge has been added from 2 vertices 0 to 7. So, we have written the code till now for adding a specific number of vertices in the graph and adding one random edge what is our aim? Our aim is to keep adding these random edges in the network till our network becomes connected and then to look at how many edges were these which made this network connected for doing that we go back to our code.

(Refer Slide Time: 09:22)



```
G.add_nodes_from(range(n))
return G

#Add one random edge
def add_random_edge(G):
    v1=random.choice(G.nodes())
    v2=random.choice(G.nodes())
    if v1!=v2:
        G.add_edge(v1,v2)
    return G

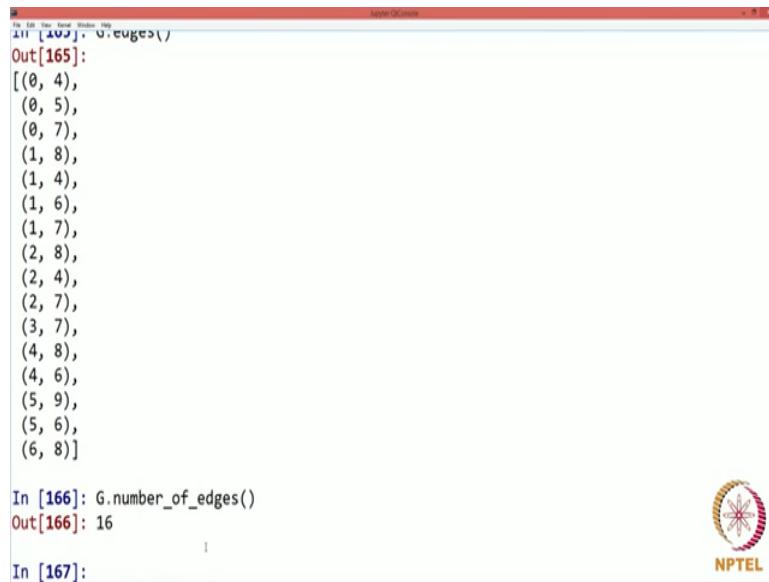
#Keeps adding random edges in G till it becomes connected
def add_till_connectivity(G):
    while(nx.is_connected(G)==False):
        G=add_random_edge(G)
```



And let us create another function here what is function does is it add random edges let rather say keep keeps adding random edges in G till it becomes connected and let us name this function as add till connectivity add till connectivity and it takes the graph G as input. So, what it is going to do till the graph does not become sorry till the graphs become connected.

So, while nx.is\_connected g. So, this is a function in networkx which returns true if the graph is connected false if it is not connected. So, while nx.is\_connected(G) = false it keeps running and what does it do? It adds one random edge to the graph for graph G becomes add random edge G. So, one random edge is added to this graph and it keeps adding more random edges till the graph becomes connected and at the end, let us return the graph. So, let us look at what this code is doing again we reload connectivity and now what we are going to do is G = connectivity.add\_till\_connectivity. So, let us see how many how many edges it adds. So, we already had one edge which we have added previously, and we pass the graph G here and let us now look at G.edges.

(Refer Slide Time: 11:19)



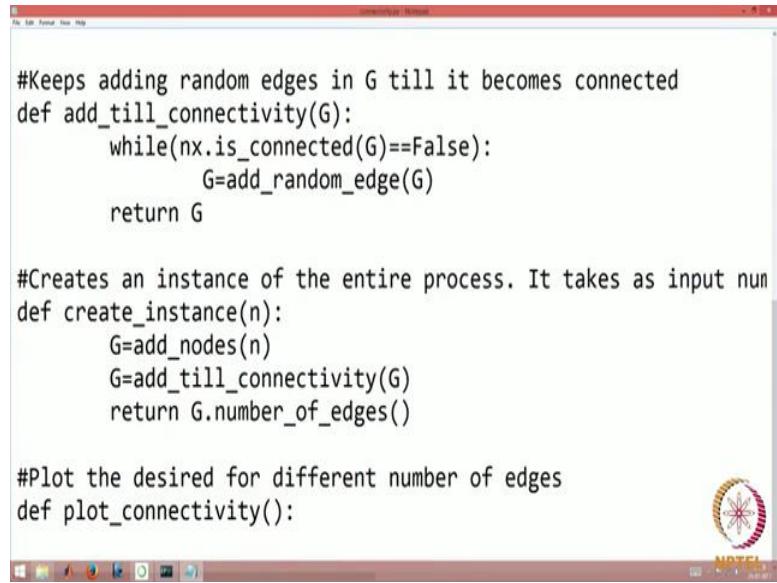
```
In [165]: G.edges()
Out[165]:
[(0, 4),
 (0, 5),
 (0, 7),
 (1, 8),
 (1, 4),
 (1, 6),
 (1, 7),
 (2, 8),
 (2, 4),
 (2, 7),
 (3, 7),
 (4, 8),
 (4, 6),
 (5, 9),
 (5, 6),
 (6, 8)]
```

```
In [166]: G.number_of_edges()
Out[166]: 16
```

In [167]:

So, we see that many edges have been added to this network to be precise the number of edges added is 16.

(Refer Slide Time: 11:36)



```
#Keeps adding random edges in G till it becomes connected
def add_till_connectivity(G):
    while(nx.is_connected(G)==False):
        G=add_random_edge(G)
    return G

#Creates an instance of the entire process. It takes as input num
def create_instance(n):
    G=add_nodes(n)
    G=add_till_connectivity(G)
    return G.number_of_edges()

#Plot the desired for different number of edges
def plot_connectivity():
```

So, it seems like in a graph having sixteen nodes just sixteen edges are required to make this graph connected let us also quickly check whether its connected or not. So, `nx.is_connected G` we can say that it is we can see that it is connected it is true.

Till now what we have done is we have written a small piece of code which takes a graph as input keeps adding edges to this graph till the graph becomes connected next what we want to do is to repeat this procedure for different number of nodes. So, here for example, in this we did it for 10 nodes and we looked at that the number of edges required was 16, now want to do it for 20 look at the number want to do for 30 look at the number and so on.

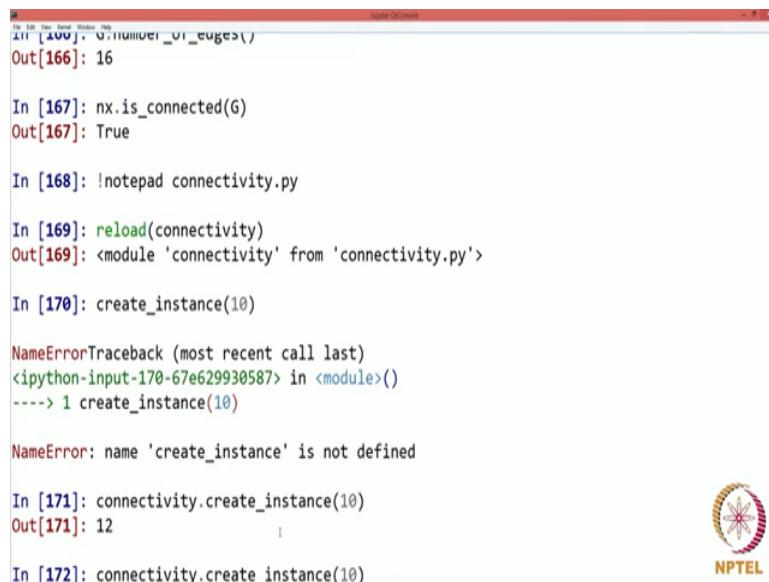
So, for doing that first of all let us create a new function here. So, this function is going to simplify what all we have written here let us name the function `create` creates an instance of the entire process. So, what do I mean here by instance of an entire process? It takes as input. So, the function will simply take as input number of nodes and returns the number of edges required for connectivity. So, it will make a task easier. So, we will just input a number and a number of nodes and it will output us it will tell us how many edges are required to connect a graph having these many nodes.

So, let us define this function let us name it as `create instance` and whatever is required inside the code for this function, we have already done that. So, define `create underscore instance` and let us pass a number and here what we do next is first of all we have to add

nodes. So,  $G = \text{add\_underscore\_nodes}$  and we add  $n$  nodes. So, we called the first function which we defined previously. So, whatever we have done till now in Ipython that only we are replicating here in the file.

So, we first call this function `add underscore nodes` and which returns us the graph and next we can simply call `G equals true` keep adding till the graph gets connected. So, `add till connectivity` we again pass here `G` and it will return us what it will return us the number of edges return `G.number_of_edges`.

(Refer Slide Time: 14:50)



The screenshot shows a Jupyter Notebook interface with the following code execution history:

```
In [166]: G.number_of_edges()
Out[166]: 16

In [167]: nx.is_connected(G)
Out[167]: True

In [168]: !notepad connectivity.py

In [169]: reload(connectivity)
Out[169]: <module 'connectivity' from 'connectivity.py'>

In [170]: create_instance(10)

NameErrorTraceback (most recent call last)
<ipython-input-170-67e629930587> in <module>()
      1 create_instance(10)

NameError: name 'create_instance' is not defined

In [171]: connectivity.create_instance(10)
Out[171]: 12

In [172]: connectivity.create_instance(10)
```

The notebook also features the NPTEL logo in the bottom right corner.

Let us see how it does? how is it working reload the file and let us call `create underscore instance` and let us pass the number 10 here. So, it is in our file `connectivity.py`. So, `connectivity.create_instance(10)`. So, we see that if there are 10 nodes, we need 12 edges.

(Refer Slide Time: 15:14)



The screenshot shows a Jupyter Notebook interface with the following code and output:

```
NameErrorTraceback (most recent call last)
<ipython-input-170-67e629930587> in <module>()
      1 create_instance(10)

NameError: name 'create_instance' is not defined

In [171]: connectivity.create_instance(10)
Out[171]: 12

In [172]: connectivity.create_instance(20)
Out[172]: 27

In [173]: connectivity.create_instance(30)
Out[173]: 50

In [174]: !notepad connectivity.py

In [175]: reload(connectivity)
Out[175]: <module 'connectivity' from 'connectivity.py'>

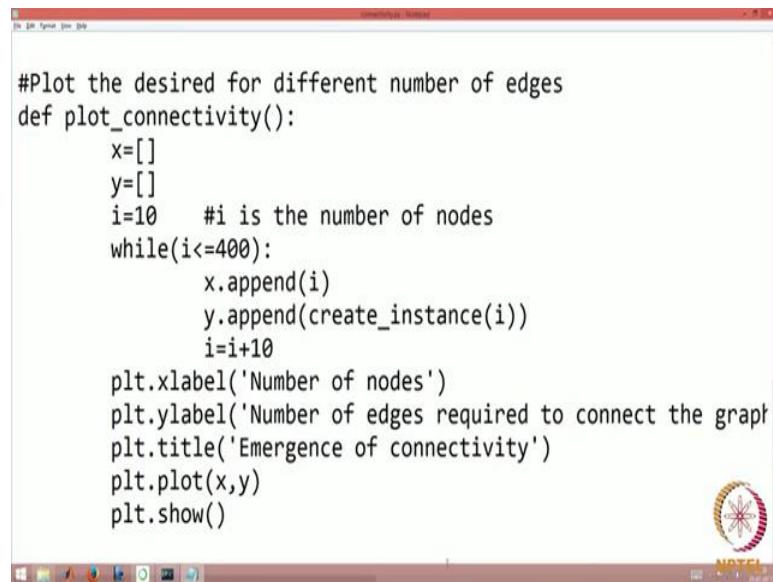
In [176]: connectivity.plot_connectivity()
```

The notebook window has a red header bar with the title "Jupyter Notebook". In the bottom right corner, there is an NPTEL logo.

Let us make it twenty nodes we need 27 edges we have thirty nodes it is 50 edges and so on and what we are interested in I want to plot this and visualize how does this plot looks like. So, we just need to we just need to plot what we have done. So, for plotting it we go back to the file, we can define a function here. So, what is this function do plot the desire let say we know what is desired. So, desired is the number of edges required for connectivity plot the desired for different number of edges and let us define a function and let us call it plot and let us call it plot connectivity.

Let us define this function plot connectivity and we it does not require as input anything we are not taking any input for simply for different number of nodes will be plot it. So, for plotting always we need the arrays corresponding to a 2 axis; x axis and y axis.

(Refer Slide Time: 16:36)



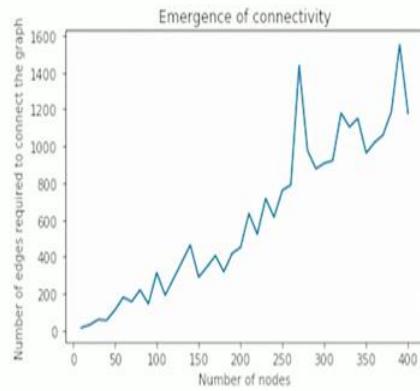
```
#Plot the desired for different number of edges
def plot_connectivity():
    x=[]
    y=[]
    i=10    #i is the number of nodes
    while(i<=400):
        x.append(i)
        y.append(create_instance(i))
        i=i+10
    plt.xlabel('Number of nodes')
    plt.ylabel('Number of edges required to connect the graph')
    plt.title('Emergence of connectivity')
    plt.plot(x,y)
    plt.show()
```

So, let us the array associated with x axis be x and the array associated with y axis be y. So, we have here x and y then we take a number  $i = 10$ . So,  $i$  what does  $i$  tells us  $i$  is the number of nodes for which we are experimenting. So,  $i$  is the number of nodes while and let us take the number of nodes till 100 or let us rather make it 400. So, while  $i \leq 400$  what is loop will do in x axis, we append the number of nodes which is nothing, but  $i$  and in y axis we can append or do we append the number of edges required to connect these many nodes which we can get from our function create instance. So,  $y.append$  create instance and we pass  $i$  here and we need let us increment  $i$  with 10 every time  $i = i + 10$ .

Now, we need to plot this for plotting this we need the package matplotlib, we import matplotlib.pyplot as plt come back. So, what do we do plt.plot what we want to plot is x against y and plt.show. So, let us just do a little bit of decoration here before plotting it let us add some labels and the title for the curve let us say plt.xlabel which is the label for the x axis is nothing, but the number of nodes plt.ylabel label for the y axis is the number of edges required to connect the graph and plt.title let us give a title to this curve and let us title it as emergence of connectivity it done.

Let us look at how does this plot look like save the file we will reload it and then we call connectivity dot plot connectivity I will take some time to execute yes.

(Refer Slide Time: 19:33)



So, here we can see the curve here we have the number of nodes here we have the number of edges and we can see that as the number of nodes increases the number of edges required to connect it also increases although we see that although we see this plot where number of nodes with number of edges we see a lot of spikes in this curve.

(Refer Slide Time: 19:49)

```
In [175]: reload(connectivity)
Out[175]: <module 'connectivity' from 'connectivity.py'>

In [176]: connectivity.plot_connectivity()

In [177]: !notepad connectivity.py

In [178]: reload(connectivity)
Out[178]: <module 'connectivity' from 'connectivity.py'>

In [179]: connectivity.plot_connectivity()
```



So, why are these spike occurring it is because our experiment is a random experiment which gives a different output every time you execute it whenever you execute a random random experiment it will give you a slightly different outputs. So, one time it can throw

a value three other time 5 other time 4, but if you take the expected value it gives you a better estimate. So, to remove these spikes in the plot instead of taking one value corresponding to one instance we might want to average it out. So, let us how can we do that we go back. So, what are we doing here is corresponding to one value of I that is corresponding to one particular instance given a particular number of nodes stand here we execute this code only one time?

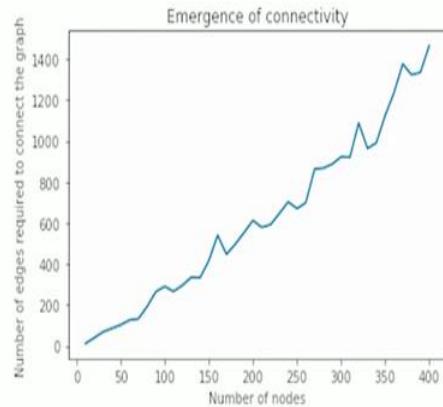
So, we take a graph having 10 nodes we will look at how many edges are required to connect it and then we return the number of edges what will give us a better result is if we do it multiple times. So, we take a graph having 10 nodes then look at the number of edges we again do this again do this do it 10 times and take an average over all those experiments that will give us a better estimate. So, let us try doing that. So, doing that is very easy. So, in this create instance. So, create underscore instances for one instance you give it one number of nodes and it gives you number of edges.

Let us average it over some particular number of instances let us average it over 4 instances. So, n see what I am going to do here is we define a new function n and let us name it create average instance n and what it does is it call the function create underscore instance 4 times and takes an average over them. So, we create a list here list one and for i in range 0 to 4. So, its repeated 4 times what does it do is list one dot append what does it append to list one is create underscore instance n. So, it appends it to list and what it returns is the average. So, it return numpy.average of list one.

So, we have yet node imported numpy. So, let us import numpy here import numpy. So, its returns numpy dot average list one. So, what we are going to do is while plotting in plotting connectivity instead of using creating underscore instance now we create we use create underscore average underscore instance. So, the value is averaged over 4 particular instances now let us look at the change in the curve it creates we reload connectivity and we again call this function.

So, since it is doing it 4 times it takes some time and you can see you can say that the difference in 2 curves is visible this curve is more smoother because we have averaged out the values over 4 instances where it was only one instance.

(Refer Slide Time: 23:21)



In this case let us play around with it a little bit further and let us change this 4 to let say 10 we do an average over 10 instances we make this 4 as 10.

(Refer Slide Time: 23:45)

```
def create_instance(n):
    G=add_nodes(n)
    G=add_till_connectivity(G)
    return G.number_of_edges()

#Average it over 10 instances
def create_avg_instance(n):
    list1=[]
    for i in range(0,10):
        list1.append(create_instance(n))
    return numpy.average(list1)

#Plot the desired for different number of edges
def plot_connectivity():
    x=[]
    y=[ ]
```



(Refer Slide Time: 23:52)

In [177]: !notepad connectivity.py

In [178]: reload(connectivity)  
Out[178]: <module 'connectivity' from 'connectivity.py'>

In [179]: connectivity.plot\_connectivity()

In [180]: !notepad connectivity.py

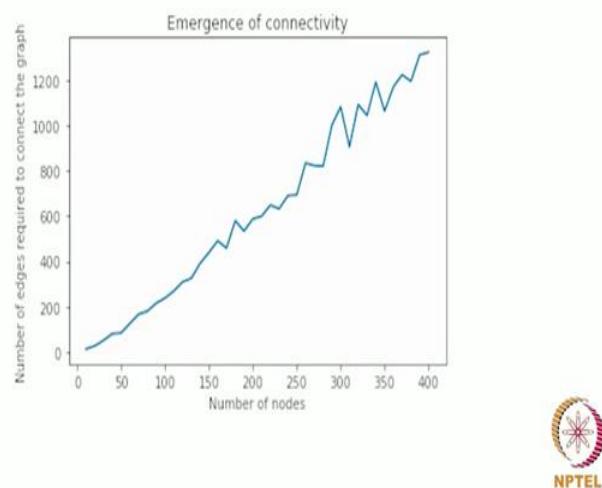
In [181]: reload(connectivity)  
Out[181]: <module 'connectivity' from 'connectivity.py'>

In [182]: connectivity.plot\_connectivity()

The screenshot shows a Jupyter Notebook interface with several code cells and their outputs. The first two cells are standard Python imports and module reloads. The third cell calls the `plot\_connectivity` function, which generates a line graph titled "Emergence of connectivity". The x-axis is labeled "Number of nodes" and ranges from 0 to 400. The y-axis is labeled "Number of edges required to connect the graph" and ranges from 0 to 1200. The plot shows a jagged, increasing curve that becomes smoother as it approaches 1200 edges at approximately 400 nodes. The NPTEL logo is visible in the bottom right corner of the slide.

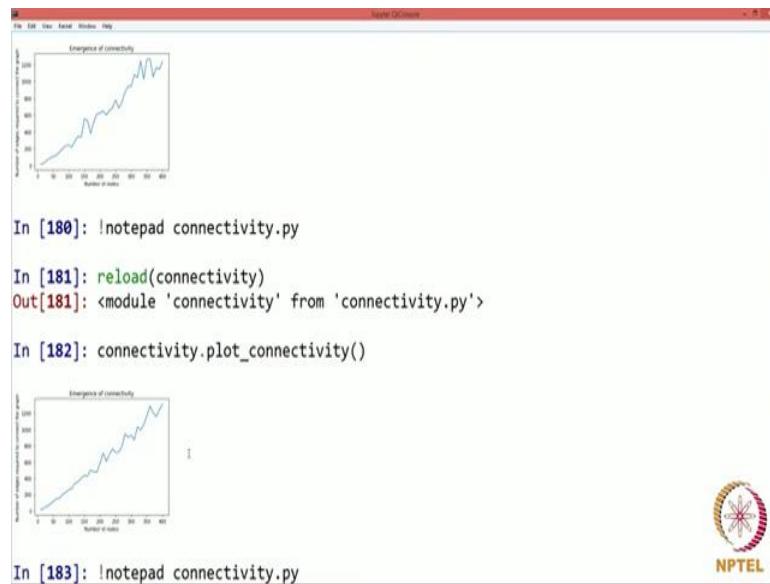
We reload it and we call the function again. So, let us wait for the output.

(Refer Slide Time: 24:01)



So, again you can see a visible change in the curve it gets all the more smooth. So, you can change this value from 10 to 20 to 3200 and the curve will become more and more smooth.

(Refer Slide Time: 24:12)



So, we can try it for higher values now the final and the biggest question what was our claim was that if there are  $n$  nodes in the network it takes just  $n \log n$  number of edges to connect this network the question is are these number of edges  $n \log n$ . So, if I look at this curve is it the curves of  $n \log n$  oh let us write a piece of code and check that as well.

So, what I am going to do is I am going to draw this plot. So, I am going to have 2 plots in the same figure one plot is the plot which we have achieved from our code which is the actual data and let us plot it against  $n \log n$  and look at what is the result.

(Refer Slide Time: 24:58)

```
def create_instance(n):
    G=add_nodes(n)
    G=add_till_connectivity(G)
    return G.number_of_edges()

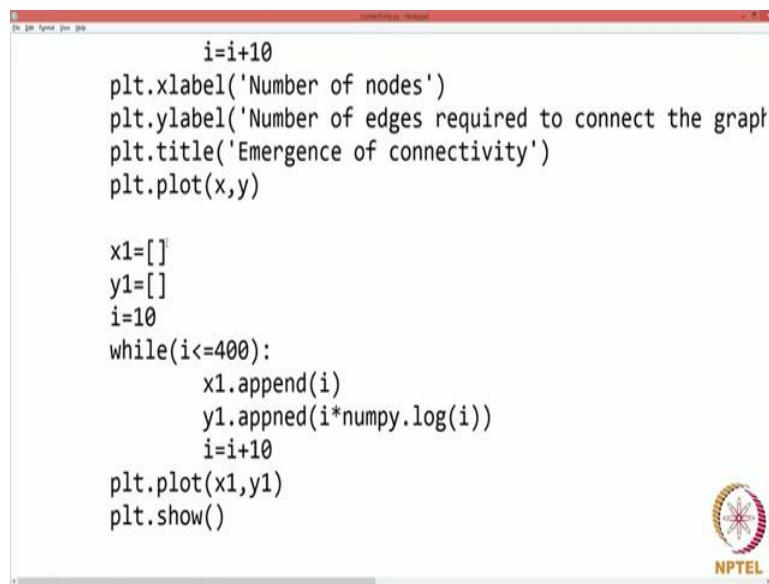
#Average it over 100 instances
def create_avg_instance(n):
    list1=[]
    for i in range(0,100):
        list1.append(create_instance(n))
    return numpy.average(list1)

#Plot the desired for different number of edges
def plot_connectivity():
    x=[]
    y=[ ]
```

So, I want a very smooth curve. So, for a very smooth curve what I do is I average it to hundred instances. So, I average it 400 instances and now I want to plot it against the  $n \log n$  curve. So, what I do is after doing this before plotting I do exactly the same procedure, which is written above, but here I am going to plot  $n \log n$ .

So, whatever I am going to do here is simply a replication of the above code with a little bit of change and you can write it as a separate function as well and I am writing it here.

(Refer Slide Time: 25:38)



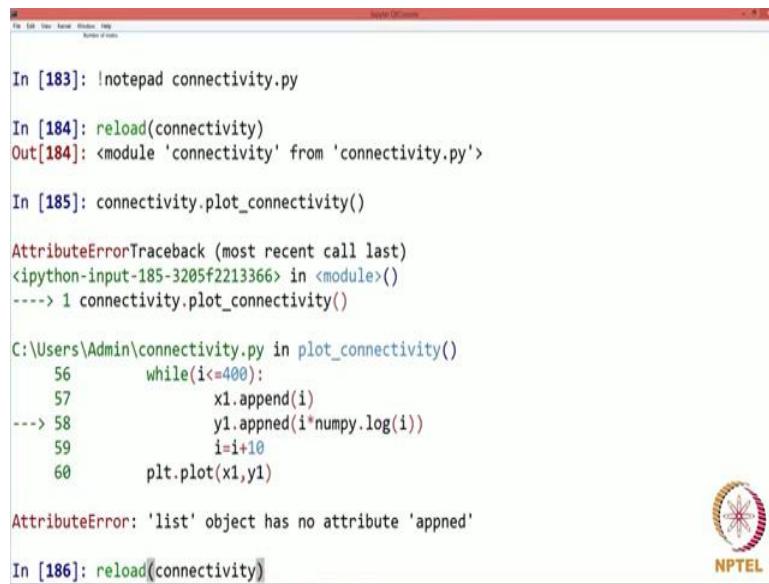
The screenshot shows a code editor window with Python code. The code defines two plots: one for actual data (x, y) and one for the theoretical curve (x1, y1). The x-axis for both is labeled 'Number of nodes' and the y-axis is labeled 'Number of edges required to connect the graph'. The title of the plots is 'Emergence of connectivity'. The code uses numpy for logarithmic calculations.

```
i=i+10
plt.xlabel('Number of nodes')
plt.ylabel('Number of edges required to connect the graph')
plt.title('Emergence of connectivity')
plt.plot(x,y)

x1=[]
y1=[]
i=10
while(i<=400):
    x1.append(i)
    y1.append(i*numpy.log(i))
    i=i+10
plt.plot(x1,y1)
plt.show()
```

So, I take a new array  $x_1$  and a new array  $y_1$  one and then again I set  $i = 10$  which is the number of nodes and while  $i \leq 400$  what I do is  $x_1.append(i)$ .  $y_1.append$  is the curve I want and the curve which I want is  $i * \log(i)$ , sorry, it is  $x_1.append$  and it is  $y_1.append(i * \log(i))$  and then I increment  $i$  with 10 and I do  $plt.plot(x_1, y_1)$ . So, it is going to show me 2 plots in the same figure 1 is for  $xy$  which is the actual data number of nodes and the number of edges required to connect them and second is the  $x_1 y_1$  which tells me a number on the x axis and if the number is  $I$ , it tells me  $i \log i$  on the y axis and again reload the file and let us run it.

(Refer Slide Time: 27:08)

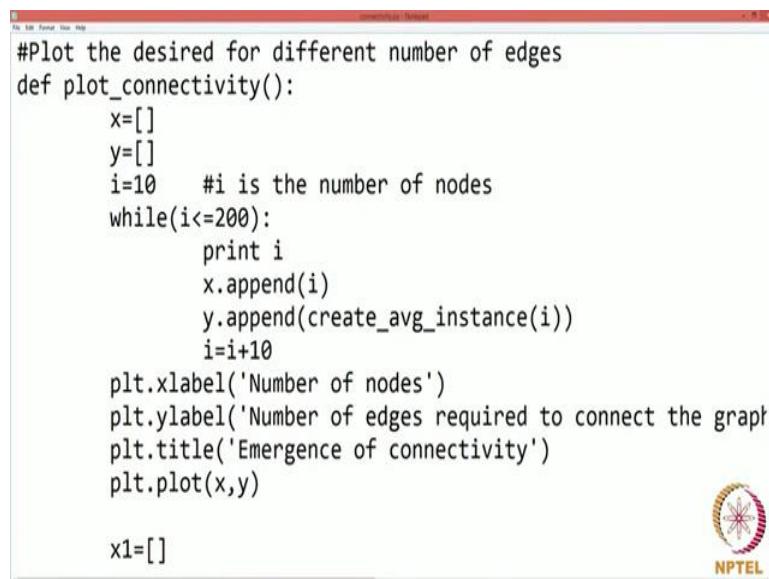


```
In [183]: !notepad connectivity.py
In [184]: reload(connectivity)
Out[184]: <module 'connectivity' from 'connectivity.py'>
In [185]: connectivity.plot_connectivity()
AttributeErrorTraceback (most recent call last)
<ipython-input-185-3205f2213366> in <module>()
      1 connectivity.plot_connectivity()
C:\Users\Admin\connectivity.py in plot_connectivity()
    56     while(i<=400):
    57         x1.append(i)
--> 58         y1.append(i*numpy.log(i))
    59         i=i+10
   60     plt.plot(x1,y1)
AttributeError: 'list' object has no attribute 'appned'
In [186]: reload(connectivity)
```



And let us give it some time to execute since it is averaging over hundred instances it will take some decent amount of time the code here ends within a error and next again a spelling mistake let us go back and correct it. So, just correct it here and what we do here is I have added an extra statement here print i which keeps showing us how many iterations has the code complete it.

(Refer Slide Time: 27:34)



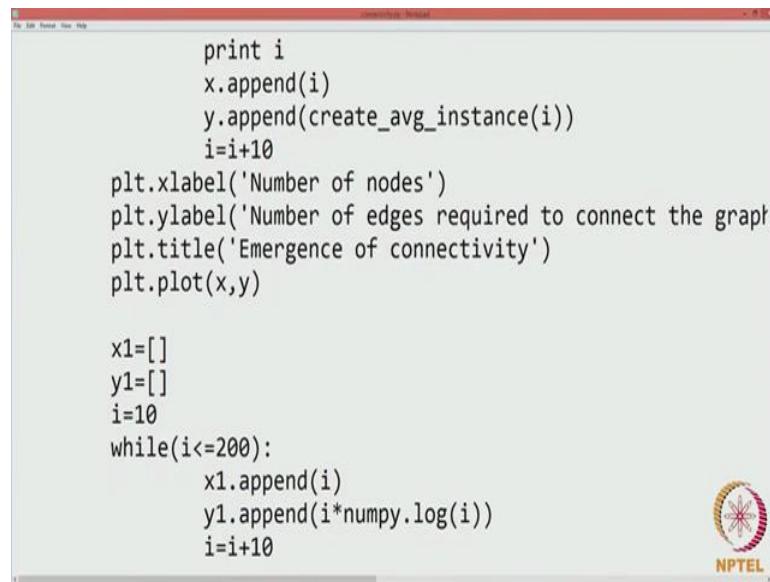
```
#Plot the desired for different number of edges
def plot_connectivity():
    x=[]
    y=[]
    i=10    #i is the number of nodes
    while(i<=200):
        print i
        x.append(i)
        y.append(create_avg_instance(i))
        i=i+10
    plt.xlabel('Number of nodes')
    plt.ylabel('Number of edges required to connect the graph')
    plt.title('Emergence of connectivity')
    plt.plot(x,y)

x1=[]
```



So, for the time being let us make this 1 2 hundred since for 400 it takes a lot of time. So, let us see how it works for 200, but if we see a code below.

(Refer Slide Time: 27:59)

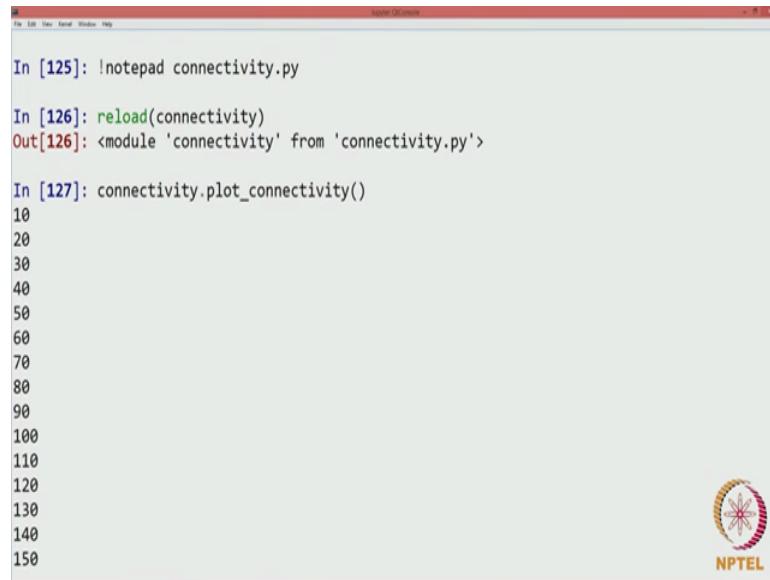


```
print i
x.append(i)
y.append(create_avg_instance(i))
i=i+10
plt.xlabel('Number of nodes')
plt.ylabel('Number of edges required to connect the graph')
plt.title('Emergence of connectivity')
plt.plot(x,y)

x1=[]
y1=[]
i=10
while(i<=200):
    x1.append(i)
    y1.append(i*numpy.log(i))
    i=i+10
```

So, for the second plot it is still plotting till the till 400 nodes. So, we also change this and make it here 200 and now we just save this code and let us run it.

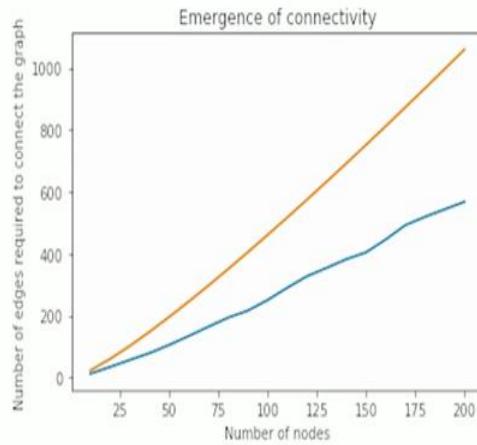
(Refer Slide Time: 28:09)



```
In [125]: !notepad connectivity.py
In [126]: reload(connectivity)
Out[126]: <module 'connectivity' from 'connectivity.py'>
In [127]: connectivity.plot_connectivity()
10
20
30
40
50
60
70
80
90
100
110
120
130
140
150
```

So, let us reload our file we call function `connectivity.plot_connectivity`. So, here you can see 2 lines in this plot one is the blue line which corresponds to our data where we took different number of nodes and looked at the number of edges required to connect the corresponding network and the red line it corresponds to the  $n \log n$  plot. So, the lines are not totally overlapping in this case.

(Refer Slide Time: 28:25)



So, here I would like to remind you that according to the analysis that we have done the number of edges required to connect the graph was not exactly  $n \log n$  it is of the order of  $n \log n$ .

So, let us do a small; a little bit of tweak in our code let us go back to a notepad file.

(Refer Slide Time: 29:06)

```
print i
x.append(i)
y.append(create_avg_instance(i))
i=i+10
plt.xlabel('Number of nodes')
plt.ylabel('Number of edges required to connect the graph')
plt.title('Emergence of connectivity')
plt.plot(x,y)

x1=[]
y1=[]
i=10
while(i<=200):
    x1.append(i)
    y1.append(i*float(numpy.log(i))/2)
    i=i+10
```



And what I am going to do here is instead of plotting  $n \log n$  what I plot here is  $n \log n$  divided by two. So, please note that even if I divide this term  $n \log n$  by 2 in the final

analysis the number of edges required to connect the network still remains of the order of  $n \log n$ . So, let us try doing that. So, what I do here is I divide the term  $\log n$  by 2.

(Refer Slide Time: 29:35)

In [128]: !notepad connectivity.py

In [129]: reload(connectivity)  
Out[129]: <module 'connectivity' from 'connectivity.py'>

In [130]: connectivity.plot\_connectivity()

The screenshot shows a Jupyter Notebook interface. Cell 128 runs a command to open a file named connectivity.py in a text editor. Cell 129 reloads the module. Cell 130 executes a function to generate a plot titled "Emergence of connectivity". The plot shows two curves: a straight orange line representing a baseline and a blue curve that rises more slowly, indicating the reduced complexity of the connectivity requirement. The x-axis is labeled "Number of nodes" and ranges from 0 to 200. The y-axis is labeled "Number of edges required to connect the network" and ranges from 0 to 200.

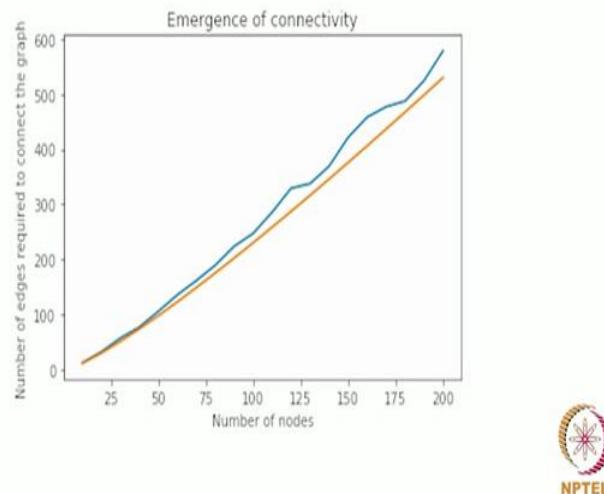
(Refer Slide Time: 29:38)

In [129]: <module 'connectivity' from 'connectivity.py'>

In [130]: connectivity.plot\_connectivity()

The screenshot shows a Jupyter Notebook interface. Cell 129 reloads the module. Cell 130 executes a function to generate a plot titled "Emergence of connectivity". The plot shows two curves: a straight orange line representing a baseline and a blue curve that rises more slowly, indicating the reduced complexity of the connectivity requirement. The x-axis is labeled "Number of nodes" and ranges from 0 to 200. The y-axis is labeled "Number of edges required to connect the network" and ranges from 0 to 200.

(Refer Slide Time: 29:46)



And let us see how our plot changes let us plot it. So, here you can see that both the curves become very close to each other establishing the fact that it is essentially  $n \log n$  by 2 number of edges almost of the order of  $n \log n$  which are required to make our graph connecting hence satisfying a previous claim.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture – 26**  
**Handling Real-world Network Datasets**  
**Summary to Datasets**

This week we saw the side of with introduction to datasets we saw why datasets, the fact that this error has helped us collect a whole lot of datasets and even we have good computational facilities to make sense out of these data sets. We should you using python networkx like APIs, how one can write a piece of code and infer make inferences from these data sets. We also told you a powerful tool like Gephi where you input a graph and it beautifully shows you the different nodes and edges of the graphic, we can visualize it in different ways, it appears very nice.

Gephi is not just used to visualize graphs it is also used to analyze graphs especially for people who are not from programming background. If you were to analyze a given a graph quickly and you want to make sense out of it in let us say no time, python is not your destination it has to be Gephi like tool. But python again gives you the flexibility because to write your own code the way you want a Gephi does not it is sort of standardized. Nevertheless we saw a tool like Gephi which we realized is both powerful and fun to use and then we moved on and saw a piece of puzzle which we have been discussing from the first weeks first lecture which is the emergence of connectedness. We saw how a graph becomes connected when you take a bunch of nodes without any edges and you start putting edges one edge at a time and you will see that there is a movement when it becomes connected, when does it become connected? If you remember it is when you put  $n \log n$  number of edges.

Then we also saw that it is not just mathematically true that its  $n \log n$  our program also says it is something to do with a little more than  $n$  number of edges for a graph with  $n$  vertices to become connected and that talks about the power of what is called synthetic data sets. Synthetic means you synthesize a dataset of your choice for example, you have a huge dataset of 2000 nodes or 2 million nodes let say, but you want to observe what happens on a graph of 100 nodes say you would like to create your own network of 100 nodes. We this the experiment with which we concluded this week was a typical example

of how one can do this. We took, you saw the plot of as you take vertices and you look at the emergence of a connectedness and the plot looks plot has something to tell you right. So, that is a good example of how you can make your own data sets as and when required.

There are many ways to do this we will cover that in the forthcoming chapter. To put the entire chapter in a nutshell we learnt about datasets we saw the emergence of connectedness and we saw the importance of synthetic data sets that us with this week. So, next week onwards we are going to shift our gates completely, the past two weeks is only been playing around with python and understanding some basics of a network x and crunching datasets and things like that. Next week onwards we are going to see some hard-core results in network science. So, all that I told you in the first weeks lectures on do you think you should watch your company, how does Google work, how do you predict who is your next friend so on and so forth all those things are coming next.

So, next week especially we are going to start off with a clip of Harry Potter and we will see what has one to learn from this clip.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture – 27**  
**Strong and Weak Relationships**  
**Introduction**

Who is there?

Dear Mister Potter, we are pleased to inform you that you have been accepted at Hogwarts School of Witchcraft and Wizardry.

You will not be going thorough; we saw when we took him through, and we could have stopped all these rubbish.

You knew; he knew all along and he never told me.

Cruize, did you?

I am sure most of you are watched Harry Potter. And this clip is a epic clip, we all remember this scene. What has this clip to do in social networks? Guess what just happened here, Hagrid comes to take Harry Potter and when Hagrid comes, Harry Potter has a; he is perplexed he is not sure who this person is; 8 foot long person who knocks the door off and then enters, he asks who are you? Why are you here? Hagrid says I am here to take you; take you to Hogwarts, he does not even know what is Hogwarts and he says here is this school of Witchcraft and Wizardry and you are invited to come and join the school.

What just happened? Hagrid comes to meet Harry Potter and take him and Harry Potter does not know of Hagrid; pause for a minute and let us think is this what happens in a real life.

(Refer Slide Time: 02:22)



Do you think opportunity knocks Orlando through someone whom we have no clue of or is it someone who is a friend and close relative who comes and gives us info of an opportunity? So, here are two possible cases case number one someone whom you know comes and tells you about a job opportunity case two is the person who comes and tells you about the job opportunity is someone whom you do not know well, says it is a second one where second one I mean most of the jobs that people get is basically through someone they are not very close with is with someone who is called a weak tie, tie means friendship weak friendship.

So, if you have some hundred friends most probably your next job opportunity will come from someone who is a friend who is basically acquaintance I am not a close friend is what I meant very good friends will not give you the info for the next job it is from someone whom you are not very familiar not very close to is the one who is going to come and give you a job opportunity why is that is what makes this chapter. So, let see the concepts let us develop the ideas and finally, answer the question what the importance of weak ties is.

(Refer Slide Time: 04:04)



So, the title of the chapter is the strength of weak ties, the strength of weak friendships.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture – 28**  
**Strong and Weak Relationships**  
**Granovetter's Strength of weak ties**

(Refer Slide Time: 00:07)



Let us do this thought experiment. Assume I went to some 100 people randomly and ask them. So, you are working in this place, how did you get to know of this job? Who told you? You will be surprised to note that more than 90 percent of them will say; this person who was my friend's friend or this person I was not close to told me about this job so on and so forth. You will never here a person saying oh my dad told me of this job opportunity very rarely you might encounter such an answer, but the most frequently heard answer will be this answer that a very far away relative or a very not. So, close friend told me of this job opportunity more than 90 percent.

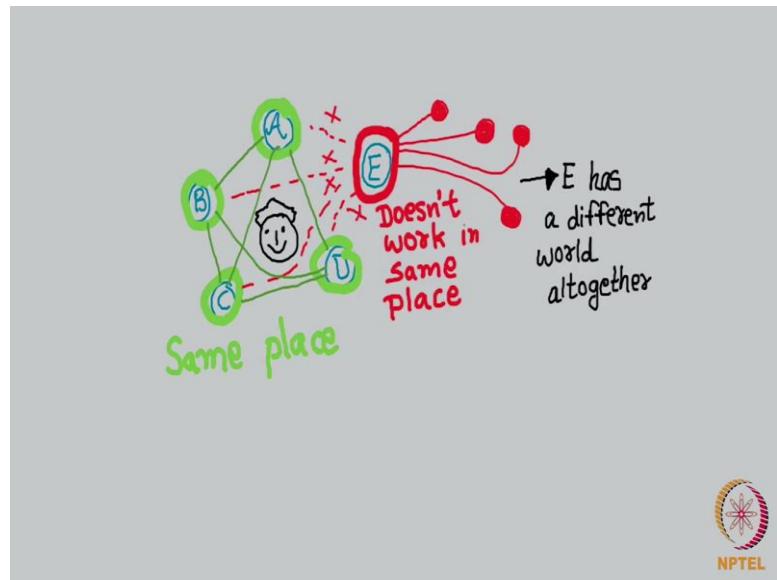
Now, I am sure most of you are finding this against your common sense your finding it very counter intuitive because an intuition says close people of the ones we will help us with job opportunities.

(Refer Slide Time: 01:08)



Just what is happening here is it because or close friends do not want to tell us the of the job opportunities, not real the reason is something else, let see what exactly is the reason the answer is actually convincing if you know the right reasoning I am going to give you an example.

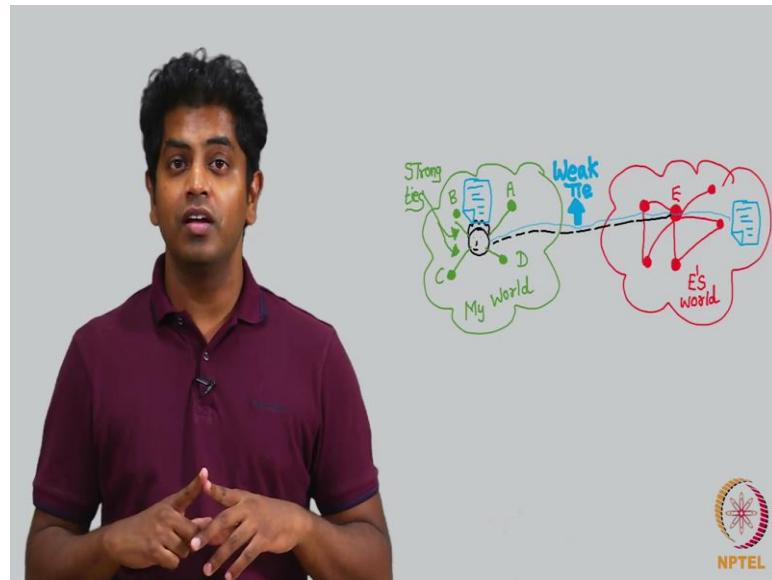
(Refer Slide Time: 01:34)



Assume I have 5 friends A, B, C, D and E, A, B, C, D work in the same place where I work, while E is someone who does not work in the same place as me and E does not know A, B, C, D, while A, B, C, D, they all know each other assume such a scenario.

Now, whatever A, B, C, D knows mostly even I know because I am from the same place there are may be many other people who would have told me the same information which A, B, C, D, might tell me well occasionally A, B, C, D my tell me something new, but mostly whatever they say even I would, but look at E is roll here E is actually as you can see here E is from a different world all together and hence this new information that E has is that of this new world that I am slightly away from.

(Refer Slide Time: 02:32)



So, E is the one who helps me get to know of things from his world which increases my sample space of information which A, B, C, D, cannot give me given that E is a distant friend of mine because he stays in a different world he probably is a weaker of my friends.

I call that a week tie while A, B, C, D, they are strong ties strong ties are week when it comes to getting job related information new job related information while weak tie such as E is strong with the sense that any new information that I get which is worth knowing let say in terms of new job is from E and hence this weak tie is actually strong while strong ties are actually weak Granovetter in the late 1960s he conducted this experiment by going and asking people how did you find this new job and most of them said it was through acquaintance. And that is when he got this idea that there is something hap counter intuitive stuff happening here and let me experiment it properly he conducted is

experiments concluded that it is a strength of weak ties that is in play here as and always it sounds a little controversial and the scientific community did not accept his claims.

In fact, he first published tried publishing it in 1969, but this paper got rejected it was only 1973 is at people shall I accepting is used and it got published and after it got published it became really famous, simply because it is such an important piece of information at very counterintuitive you see the fact that you are acquaintances are very very important is something that is to be learnt in the sense that the moral of the story is that keep your acquaintances happy, you should not just read your good friends you should reach your acquaintance is very happy. Now there is some kind of a fallacy in this statement if you keep your acquaintance is very happy they will become your good friends and when they become your good friends probably will not get job new information from them new job related information from them.

So, the point is you should have a big friend circle your opportunities increase when you know a lot of people, but then new information actually comes from someone who is not. So, close to you for the reasons that I explain just now. So now, let us go ahead and see some slightly mathematically abstract principles that surround this concept.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture – 29**  
**Strong and Weak Relationships**  
**Triads, clustering coefficient and**  
**Neighborhoods overlap**

Now we will see some important concepts, I will define them, tell you some results about it and we will slowly go ahead and understand this notion of the strength of weak ties very clearly. Firstly, I would like to define this as structure as you can see.

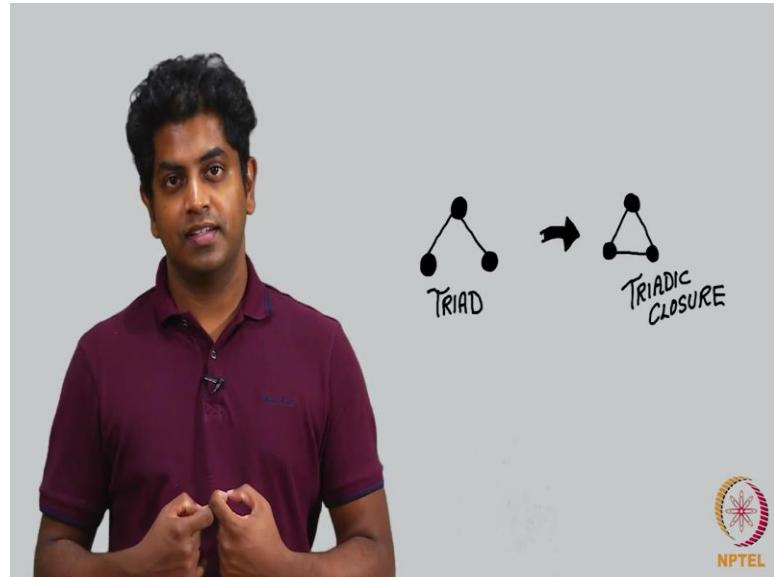
(Refer Slide Time: 00:23)



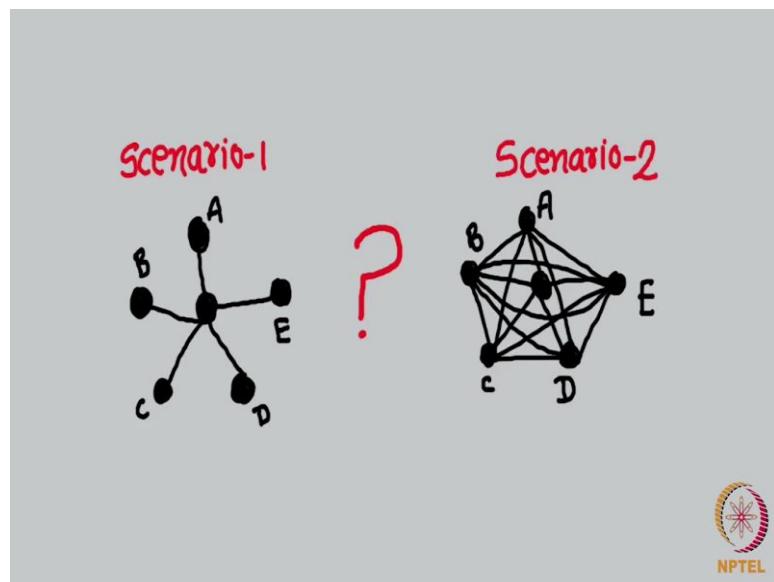
This is called a triad; a triad is simply speaking Sudarshan knows 2 people A and B and this is called a triad and A and B; they do not know each other, what will happen? You have 2 good friends and they do not know each other, eventually you can expect that they become friends with each other, A and B will know each other. This structure is called a triad and the friendship that happens in this triad making it a triangle is called; this phenomenon is called the triadic closure.

So, you see there is a triad and it closes and that is called triadic closure let this sink into your mind, this is one very important topic in social networks which will keep revisiting us in the forthcoming chapters.

(Refer Slide Time: 01:21)



(Refer Slide Time: 01:28)



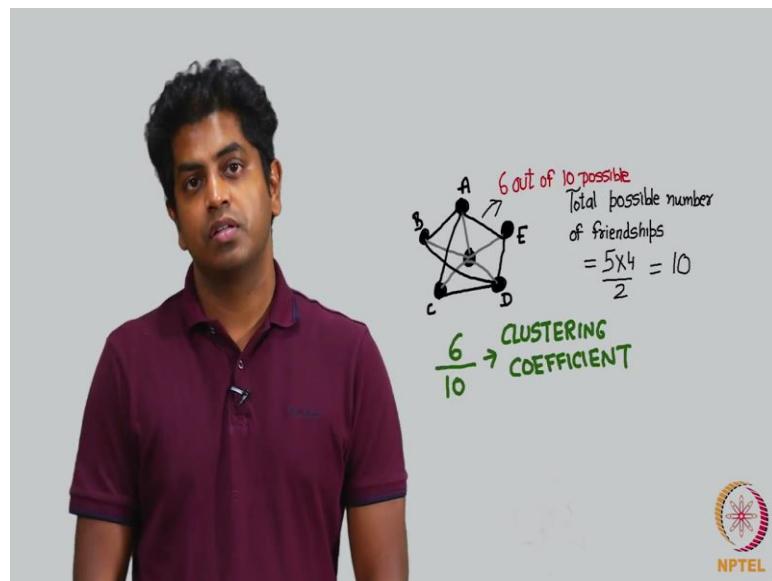
Triad becomes a triangle and that concept is called the triadic closure assume 2 scenario; scenario 1 is I have 5 friends; A, B, C, D and E and none of them know each other; that is scenario 1, scenario 2 I have this 5 friends; A, B, C, D and E and all of them know each other, now scenario 1; I have 5 friends, scenario 2; I have 5 friends, but what kind of which scenario would you favor scenario one or scenario 2.

Obviously, scenario 2, you want your friends to know each other. So, that if there is any problem to you all of them can team up and then come and help you, correct, there are

situations where scenario 1 is also very helpful I am reminded of a Bollywood movie; I am unable to recollect the name. So, it is Shah Rukh's movie and he has 3 girlfriends and he ensures that they do not know each other because if they know each other they will come and destroy him given that he is cheating on them by being in a relationship with 3 people simultaneously. So, here is a scenario where the friends not knowing girlfriends not knowing about each other might help any way, jokes apart.

So, it is important that you maintain relationship with your friends in such a way that your friendship circle your friends they know each other let me try mathematically quantifying what I just now said here is me and I have 5 friends A, B, C, D, E and they all are friends with each other this was scenario 2, scenario 1 is none of them are friends with each other they can be something in between these two things what do I mean by this?

(Refer Slide Time: 03:23)



Imagine a scenario where I am friends with 5 people and there are some friendships between them how many possible friendships can happen between 5 people; 5 into 4 by 2 you see that that is very simple, right for 5 into 4 by 2 gives you 10, there are 10 possible friendships.

How many friendships are there in this example there are 6 friendships. So, I would say the strength of my friends these 5 friends depend on the friendships between them. So, this fraction 6 by 10 denotes the strength of my friendship with this 5 people this goes by

the name clustering coefficient we will be calling this the clustering coefficient, from now onwards what does clustering coefficient mean it means in the numerator you put the total number of friendships between your friends and the denominator you put the total possible friendships this fraction if it is one it means all your friends know each other if it is 0 then none of them know each other.

(Refer Slide Time: 04:29)



Anything in between tells; how strong are the friendships between your friends? So, clustering coefficient is a particularly important concept in social networks and I just now defined it what is the use of this.

(Refer Slide Time: 04:53)



A very eye opening research says in most of the cases where few sides was committed they observed that these people had very less clustering coefficient I mean is it surprising I do not think it is very surprising people who are into depression are the ones not with many friends not that they do not have many friends it is that they have friends, but this friends are from different circles. So, they do not get to meet a bunch of people at the same time.

So, I go and meet this one person and come back another person and then come back another person and then come back I do not go and meet 5 of my friends because 5 of my friends do not know each other. So, clustering coefficient it is observed is less in people who have attempted suicide. So, this way I think clustering coefficient can be used to say many things about a person, but let us just use this definition in the forthcoming units I just introduce this as a concept that is close to triadic closure I explain what is triadic closure and I explained quick implication of what it is triadic closure which is clustering coefficient.

(Refer Slide Time: 06:14)



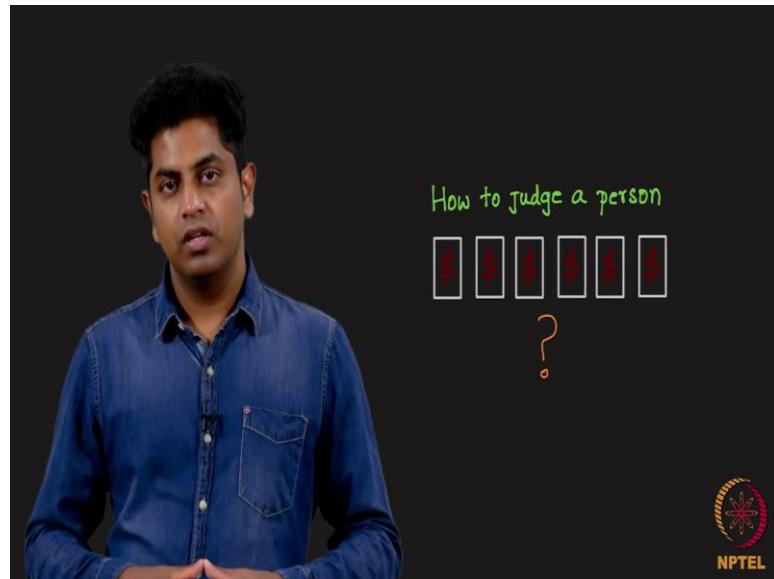
So, let me define what is a neighborhood overlap it sounds slightly complicated let me use the right analogy to ensure that the definition is well understood by all of us.

(Refer Slide Time: 06:27)



If I were to judge a person philosophically speaking is it based on how much he earns, let me give a definition of judging a person look at a look at B if A earns more than B.

(Refer Slide Time: 06:36)



(Refer Slide Time: 06:41)



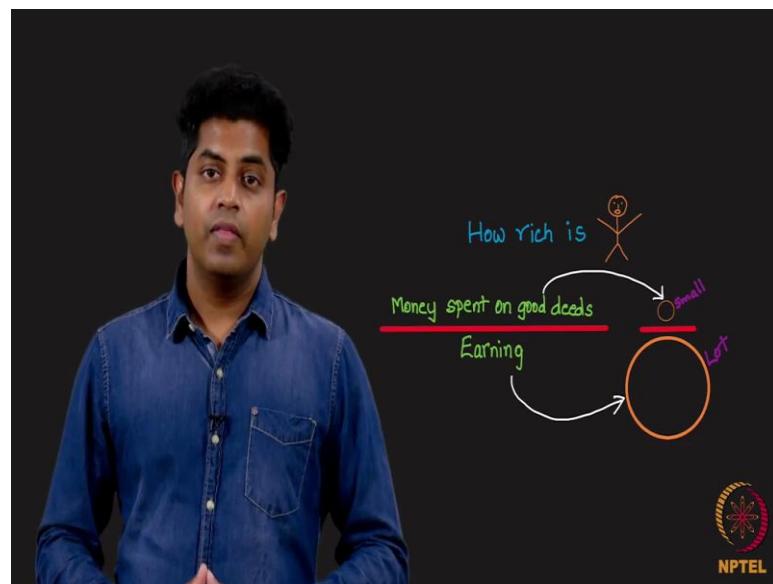
Then you say A is richer than B, let me try redefining this look at this definition that one can get I will not judge a person's richness by looking at his bank balance.

(Refer Slide Time: 06:55)



I will instead look at the following I will see how much he earns put that in the denominator and in the numerator, I will put how exactly he spends the money where exactly he spends the money.

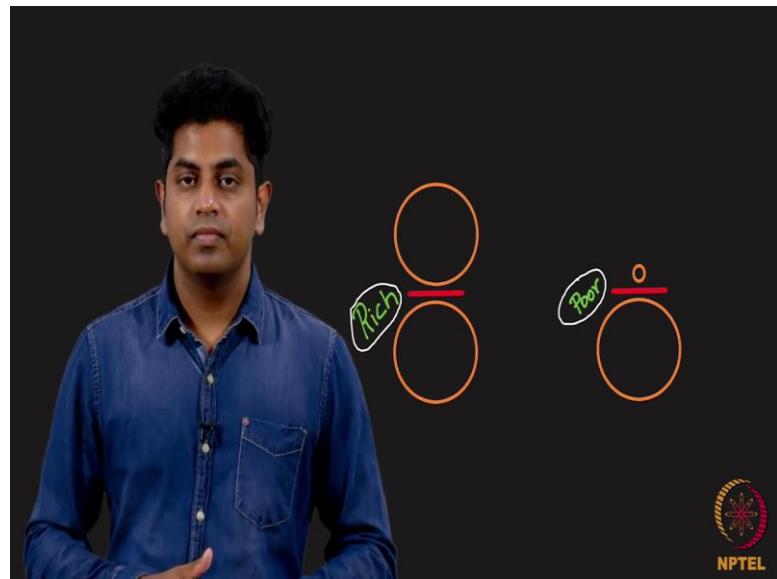
(Refer Slide Time: 07:03)



If he is spending a good chunk of his money for let say a noble cause then I will call that person the richest. Now, I am re defining what is rich by rich I mean the proportion of how well the money is spent divided by how much you earn why do I use this fraction

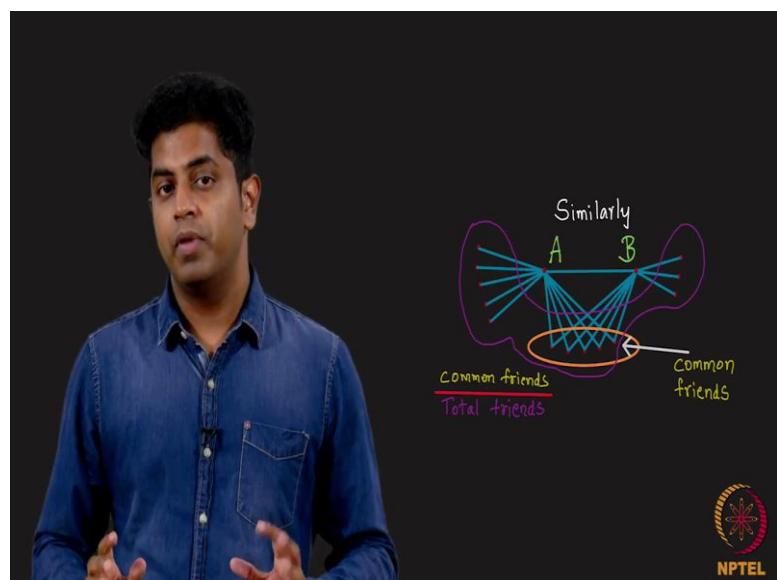
that is because you earn a lot you spend a small chunk worthy way I do not call you rich whatever you earn you spend a good chunk of it forever the cause then I call you rich.

(Refer Slide Time: 07:55)



If this fraction is close to one you are very rich if this fraction is close to 0 you are not very rich. So, I am redefining richness with the help of this fraction.

(Refer Slide Time: 08:06)

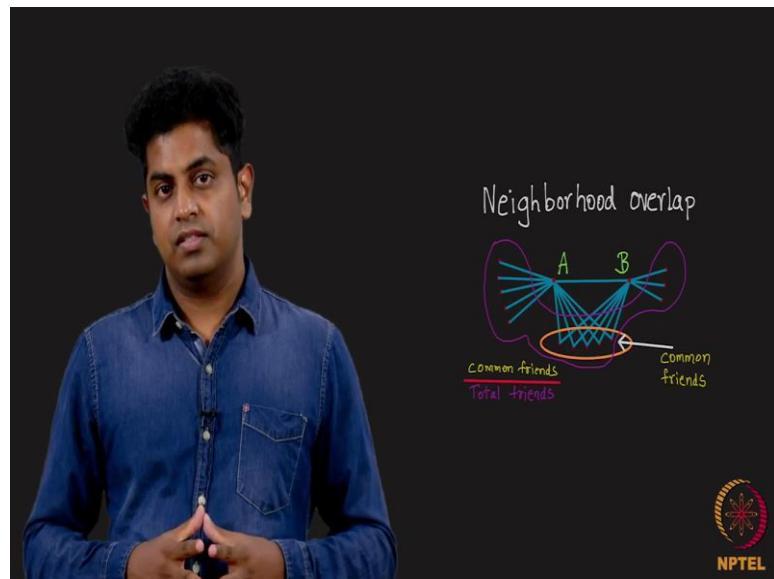


Similarly, the strength of a friendship let say between A and B can be defined the following way you can simply see how many common friends do A and B have and

designate that as the strength of their friendship or you can put this in the numerator and in the denominator you can put the sum total of friendships of A and B; I repeat.

You look at what fraction of the friends of A and B are common friends what fraction of the friends of A and B are common friends. So, this is called the neighborhood overlap.

(Refer Slide Time: 08:54)

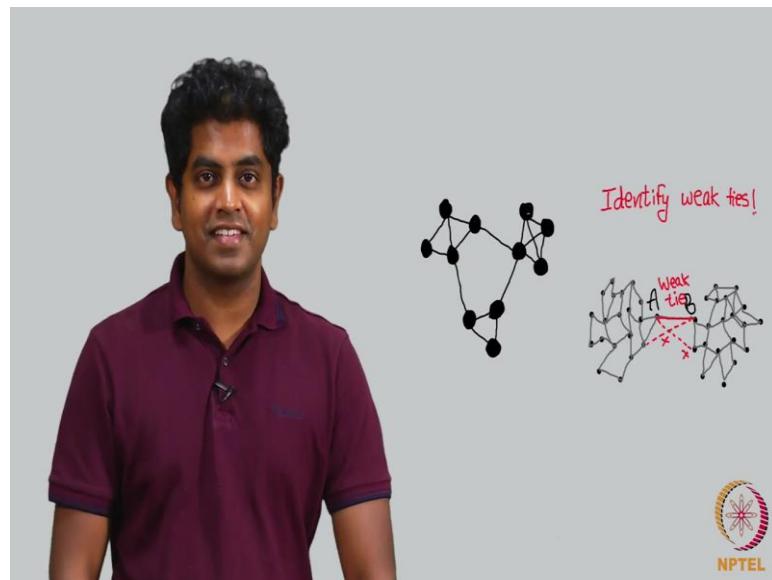


What does the neighborhood overlap of the friendship between A and B it is in the numerator total number of common friends and the denominator all possible friends? So, mathematically speaking it is the number of elements in a intersection B divided by A union B by A intersection of B, I mean friends of a intersection friends of B divided by friends of A union friends of B this is called neighborhood overlap it is close to 1; means neighborhood overlap is maximum it is close to 0 means neighborhood overlap is very less.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture – 30**  
**Strong and Weak Relationships**  
**Structure of weak ties, bridges, and local bridges**

(Refer Slide Time: 00:08)



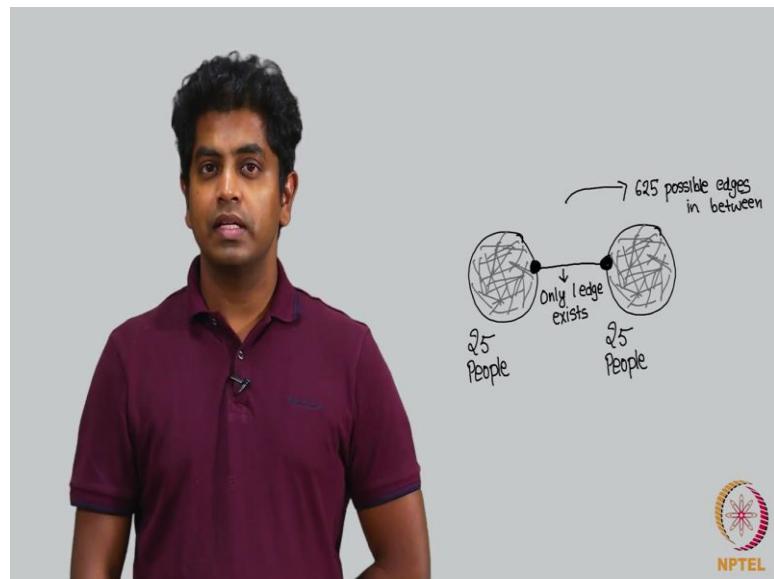
Now, assume I give you a graph  $g$ , a graph is basically a bunch of vertices and some edges, I show you the such a graph here and I ask you; can you tell me what are the weak ties here? How can you tell me? How do you know what is the intensity of friendship here? You obviously, cannot tell me, or can you? Look at the underline graph; the graph that is shown here, remember this graph is we discussed about this graph, you do not have to remember, I will help you recollect it. There are 25 nodes this side and 25 nodes that side and is only one friendship from someone this side to someone that side.

First, there are 625 possible friendships between these 2 clusters because they are 25 this side, 25 that side. Total possible is 625 out of which I am saying this only one friendship happening here, alright, fine, let us assume the graph is like this do you sense that this particular edge here this red colored edge is actually a weak tie is it not that obvious because this person is in some other island all together and in this island this person

knows a lot of people that this person does not know mainly I am saying B knows a whole lot of people that A does not know and A and B are friends with each other.

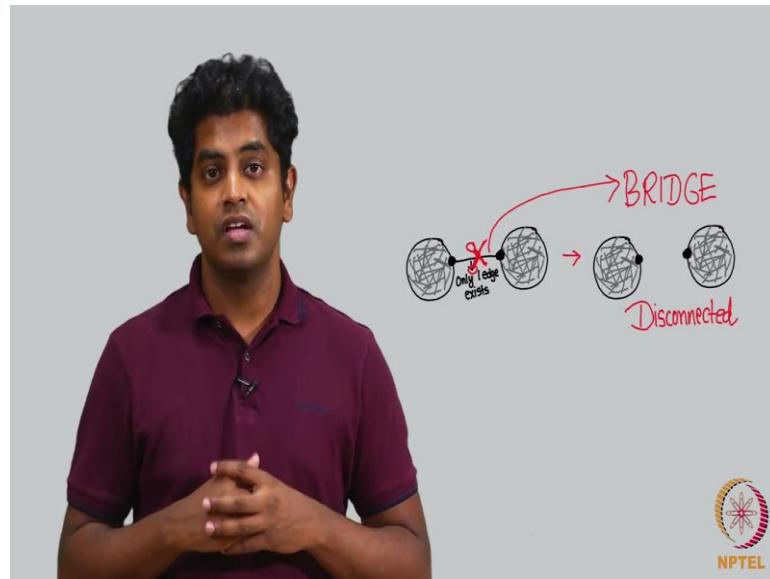
Now, A and B by definition would you think it should be a weak tie now we can tell me how do you know it is a weak tie maybe they are good friends now that is slightly not possible probably not possible for a simple reason that if they were good friends let say A and B were good friends then B has a lot of friends there and A should also be friends with them because of triadic closure correct given the triadic closure is not happening itself is an indication of the fact that A and B are weak they form of weak tie their friendship is not in this now remember what I said a while back.

(Refer Slide Time: 02:38)



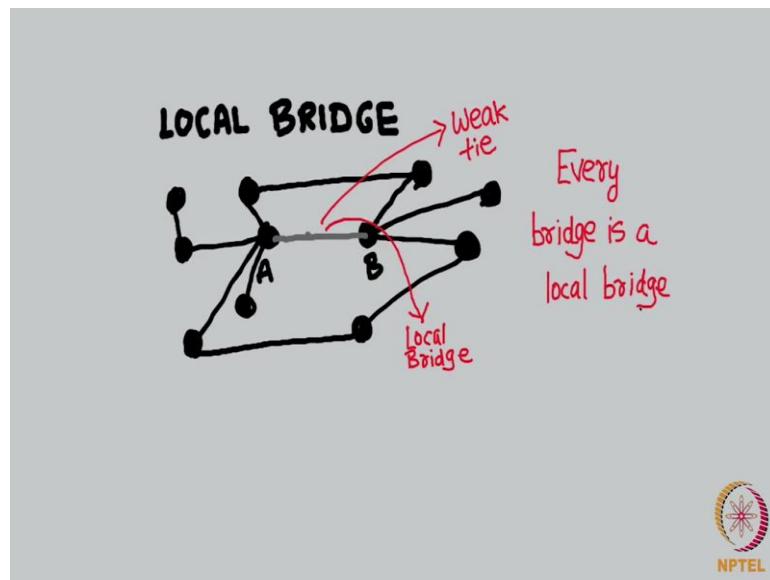
I said there are 625 possible friendships as well one friendship happening this is rare such a structure may not even exist in real life.

(Refer Slide Time: 02:50)



So, this such a edge is called a bridge the definition of a bridge is basically this you remove that edge the graph becomes disconnected that is a definition now this does not definition does not seem to be of any help, because we may not see such structures if we see such structures then we know it is a weak tie we may not see such structures.

(Refer Slide Time: 03:29)



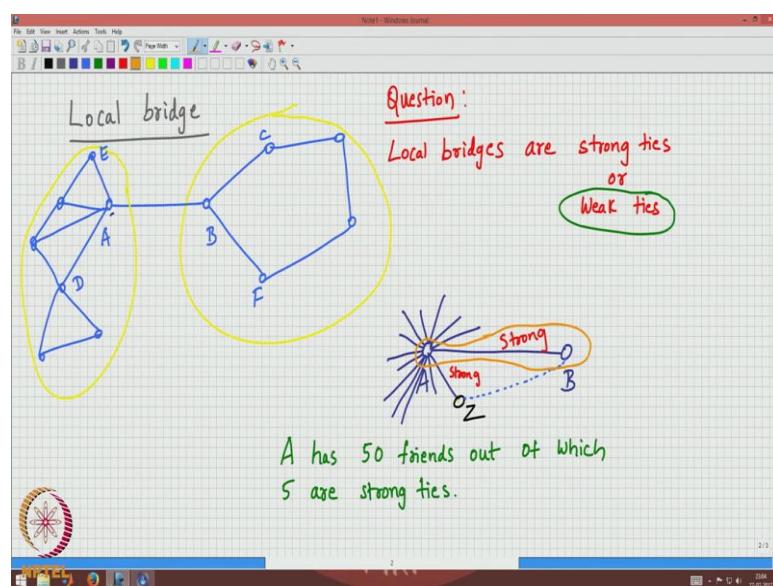
So, I am going to relax this definition slightly and I am going to define what is called a weak bridge it is called the local bridge, but it is intuitive a mind to call it a weak bridge.

So, what is a local bridge? Local bridge is bread an edge without any triad on the either sides of the vertices.

A and B is called a local bridge if there is no triad on that not even a single triad now do you see every bridge is a local bridge by definition because a bridge means what A and B do not have any common friends nor is there a path from A to B apart from the A B edge, correct. So, A B is a local bridge if there is no triad on it as you can see from the figure here in this example A B is not a bridge, but is a local bridge because there is no triad on it correct now such local bridges are also can also be considered as a weak tie why think about it here again a is friends with B A B is a local bridge none of B's friends a knows why if a well to know A B's friend that would sort of result in a triad on A and B you see.

So, none of B's friends A knows none of A's friend B knows sounds like a weak tie you see. So, Granovetter actually observed that these local bridges are what forms is what one means by a weak tie it forms a weak tie and a weak tie is mostly is a local bridge they both do not have a common friend. So, this is the definition of local bridge and this is where Granovetter argued that such weak ties is what resulted in a being friends with someone who connects him to a different island all together from where very good opportunity can knock his door through B, I am going to explain a very nice way in which we can analyze the local bridges that we just now discussed.

(Refer Slide Time: 06:04)



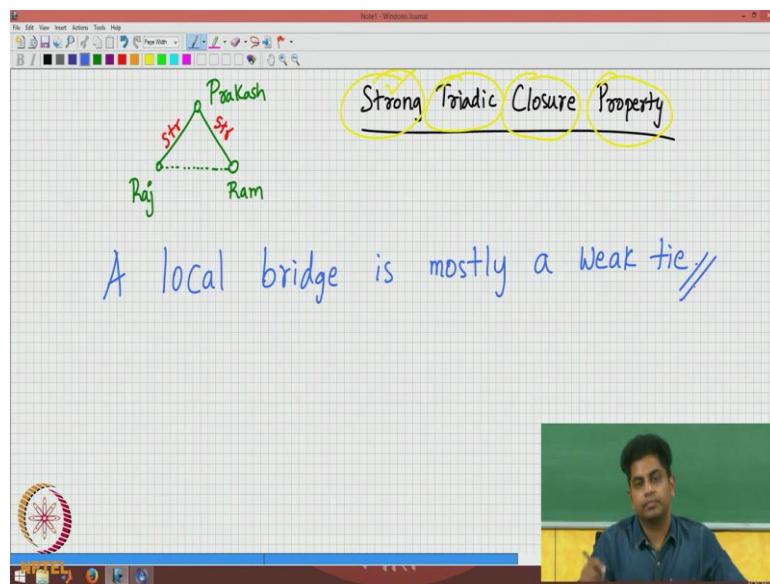
So, what is the local bridge a local bridge between two people is that edge that relationship where let say A and B, A and B are friends you call this friendship as local bridge if there are no common friends right for example, as you can see A and B are friends and A is the left side, B is in the right side and there are lot of friendships for B lot of friendships for A, but A and B do not have a common friend, A and E; they have common friends A and D have common friends, but A and B do not have common friends this is called a local bridge here goes my question let me write that down the question if you are going to ask an answer right now is the following.

What can one say about this local bridges are they strong ties or weak ties we saw strong ties we saw weak ties we discussed about the strength of weak ties right we saw local bridges now I am asking this question are local bridges are there strong ties or weak ties by this I mean A and B are friends and this A and B do they enjoy very close friendship or is it friendship very weak something in me tells me that it is indeed weak tie why let us try looking at it mathematically. Let us discuss logically why one can actually show whether it say a weak tie what I am trying to say let say as a real world analogy or I am saying is if I were it to have of friend in let say Indonesia will have be very good friends with this person. Firstly, it is the proximity.

We may not meet each other very regularly, but that aside if you where it have a friends with some on in Indonesia will that friendship be very strong looks like no is the answer there could be exceptions, but mostly no when you have friendship across 2 different clusters the way have showed here as you can see this is a cluster this is another cluster right between these 2 clusters is there is a friendship that is mostly a weak friendship how why what makes me say this let us look at it mathematically. Now, assume this a let me just write a here once again A and B and there is friendship here a has a lot of friends let say forget this case I am looking at another situation you has a lot of friends and you see some of A's friends let say A has; assume A has a has 50 friends; out of which 5 are strong friendships strong ties.

What do I mean by this a we all have friendships some of them are close some of them are not so close, A has 50 friends and let say 5 of them are very close.

(Refer Slide Time: 11:09)



Now, let us look at this in detail if you see what I just say I said if a has 2 very close friendships really close friendships assume A; let me. In fact, give them some name for clarity sake assume there is a person called Prakash, here and then raj and then Ram, Prakash is friends with Raj, Prakash is friends with Ram and they very good friends, their tie is really strong what can you say about the friendship between Raj and Ram? What can you say about it? Do you think that is also strong? That we cannot say, but for sure they should be friendship between Raj and Ram.

Why imagine you are really close to 2 people very close in the past ten years you are close to these 2 people is there a possibility that these 2 people do not know each other every possibility that you would have made these 2 people meet each other I am not saying any 2 friends of mine now each other I am only saying if there are 2 really best friends of mine from a long time strong friendships they both will at least know each other they may not be a strong tie not necessarily they could be not necessarily, but definitely there is a tie between these 2 people, right, this property which is very straight forward intuitively goes by the name strong it is not of self explanatory strong triadic closure property.

What do you mean by this? We just mean when this a triad and they are strong, they are closed that is what I mean here by a strong that is what I mean by strong triadic strong it is a triad and the friendships as strong and that will lead to the triad getting closed always

it is a property that is very easy for us to sort of visualize. Now, when this happens what do you mean by this happens is this always happens you see, but all I am saying is if we assume that this property is through in the network whenever you have 2 strong friendships there is a friendship between these 2 people. So, Raj and Ram become friends if Prakash is strongly in friendship with Raj and Ram.

Let us get back to your previous slide as you can see, A has 50 friends out of which 5 were strong ties then let us say a and this person let say this person Z A and Z are very close friends my claim was local bridges are weak ties assume A and B was indeed strong tie and A is friends with 5 people with strong ties. So, A and Z is let us say strong see what happens A is friends with B, A is friends with Z both are strong friendships. So, what does the triadic closure property is say let us go and then see the triadic closure property says that wherever you have 2 close friends they are basically friends with each other. So, what happens here you observe that Z and B will be friends because a triadic closure property does resulting in A and B does resulting in A and B not being a local bridge why what is a definition of a local bridge there are no common friends, but here. So, see or seeing a common friend between A and B which is Z, correct.

So, what have we just infer lets paraphrase A and B if you assume were local bridge was a local bridge then a naturally has a lot of friends and some of them are strong ties assume a has at least one strong tie namely Z and A and B is a local bridge my claim is this local bridge cannot be a strong tie if it becomes a string tie then there should be link between Z and B by the strong triad a closure property which is self explanatory and intuitive correct which while lets the fact that A B is a local bridge logically speaking local bridge is mostly weak tie. So, let me write that down. So, our conclusion is that local bridge is always rather mostly weak tie.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture – 31**  
**Strong and Weak Relationships**  
**Validation of Granovetter's experiment using cell phone data**

Back then when Granovetter observed, it was in the 60s, he did not have any means to get the friendship network of people he observed the following he said people who get a new job or get it get the information through weak ties and that is it and he made up the story of possible network like thing with weak tie thing and things like that, but he had no clue whether this was this can actually be validated in real life.

(Refer Slide Time: 00:52)



But then in some 10 years back, in 2007, we had data of cell phone usage between people and let see how can this be used to validate the Granovetter's strength of weak ties all you need to show is this so called local bridges actually are weak ties remember what we have been discussing all this 5 strength of weak ties means there are weak ties which are useful which are helpful what I am going to tell you right now is the fact that the local bridges turn out to be weak ties that Granovetter could not show back them how does he; for example, if he made an attempt to show how would he do that probably any scientist would look at the entire big network of friends and then go ask what is the

strength of your friendship with this person what is the strengths of your friendship with this person this person this person.

How would I do that I am in that is not a very nice way to do it may be your survey results might have noise may be people may not give the right answer for your questions when you are surveying in 2007, we had this data set of whose speaks to who on cell phone. So, the huge network and an edge this put between 2 people if they have conversed on cell phone and we would record the total number of minutes that they have spent on cell phone to the past let say 2 to 3 weeks. So, to be precise 18 weeks not 2 to 3 weeks, 18 weeks is what they have observed 18 weeks is roughly what? Four and half months.

They observed people for 4 and half months and then they saw whose speaks to whom how many minutes if I call you over cell phone then there is an edge between me and you we both are friends and they saw this big network the network was connected. In fact, around 85 percent of the nodes were belong to one big component the rest were another component you remember I discussion from first chapter that there is it is generally a connected network. Network is always connected or at least there is a big component of connected this a big, connected component in the network the rest is a small component if its disconnected. So, that a part. So, my point is when you look at this cell phone network it is a connected graph. So, what we will look at the local bridges here source a local bridge we just now saw that a proxy for local bridge better definition would be neighborhood overlap.

So, we look at the neighborhood overlap of an edge if it is higher than it is local bridge if it is lower than it is not a local bridge.

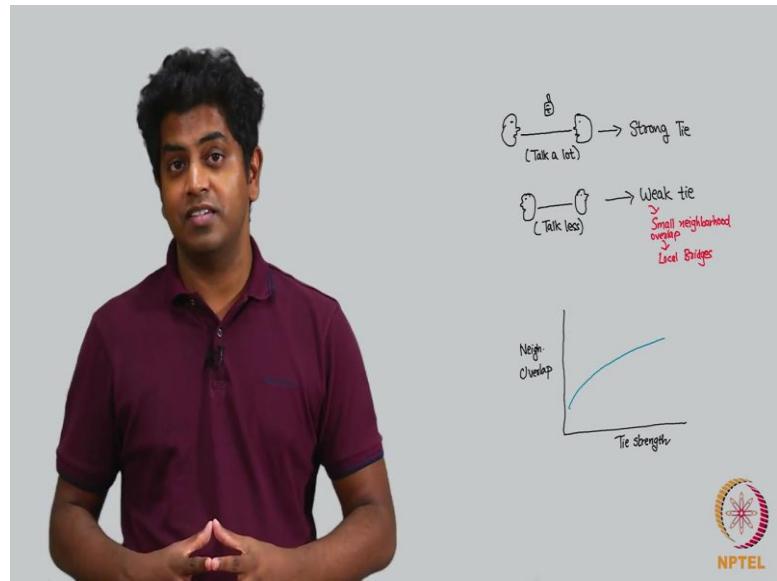
(Refer Slide Time: 04:07)



So, we saw the definition of neighborhood overlap if the neighborhoods overlap is very small then it comes close to being a local bridge. So, it was observed on this cell phone data that for those edges with very less neighborhood overlap which means it is close to being a local bridge it was observed that the cell phone talking that happened across this friendship was also very less which means higher the cell phone conversation between 2 friends higher it is local bridge properties smaller lesser the duration of the phone call more closer it is towards the local bridge.

So, what does this tell us he just tells us that see local bridge is a very binary definition is it a local branch or not correct the neighborhood overlap definition tells you as I told you in the previous video tells you nice grey scale values it gives you between 0 and one 0 means local bridge one means not a local bridge there could be values between and that is what neighborhood overlap captures and what happens in this data of cell phone usage you observe that if 2 people are talking a whole lot.

(Refer Slide Time: 05:31)



Then they are not weak tie they are strong tie if they are talking very less then there are they are weak tie and you observe that in the graph the graph structure when you observe the edges that correspond to smaller neighborhood overlap which means edges that are close to being local bridges.

We observe that the cell phone usage between these two people is of lesser duration which means it is actually a weak tie think about it a very I opening research you need which is possibly doable only in this era because not only do we have data we even have sophisticated programming possibilities that we can check such a big data.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture – 32**  
**Strong and Weak Relationships**  
**Embeddedness**

(Refer Slide Time: 00:08)



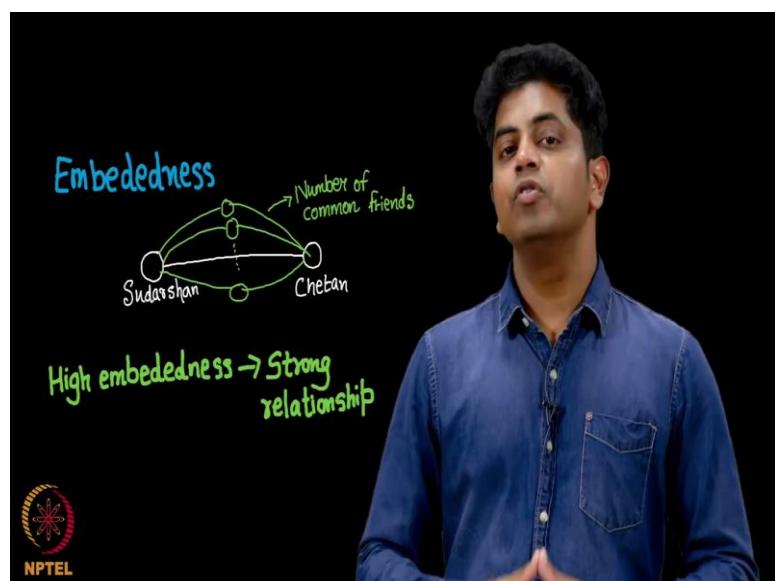
Imagine a situation where you meet a stranger on the road and you strike a conversation you both become quick friends and that friend asks you for some money what is your immediate reaction your immediate reaction is what is this we just friends in the past few minutes I do not even know this person really well and this person is asking me from money how would do I trust him?

(Refer Slide Time: 00:41)



Maybe it is a genuine request, but you will not trust this person this as supposed to a situation where your good friend whom you know from the past several years comes and asks you for some help you trust that person more one reason probably is the duration of friendship there is another reason to it which we will explore slowly in this lesson and that concept is called the embeddedness.

(Refer Slide Time: 01:03)



So, what do you mean by embeddedness of a relationship? Embeddedness of a relationship; assume I am friends with Chetan; embeddedness of this relationship

between me and Chetan is defined as the number of common friends that we have we have seen similar definitions before, but this is pretty well studied in social sciences literature where they have observed that if the embeddedness of the relationship also known as number of common friends if the common friends are high then the relationship is strong. In the example I gave you where I meet this person on the street the embeddedness there was 0, we did not know any common friends.

(Refer Slide Time: 01:49)



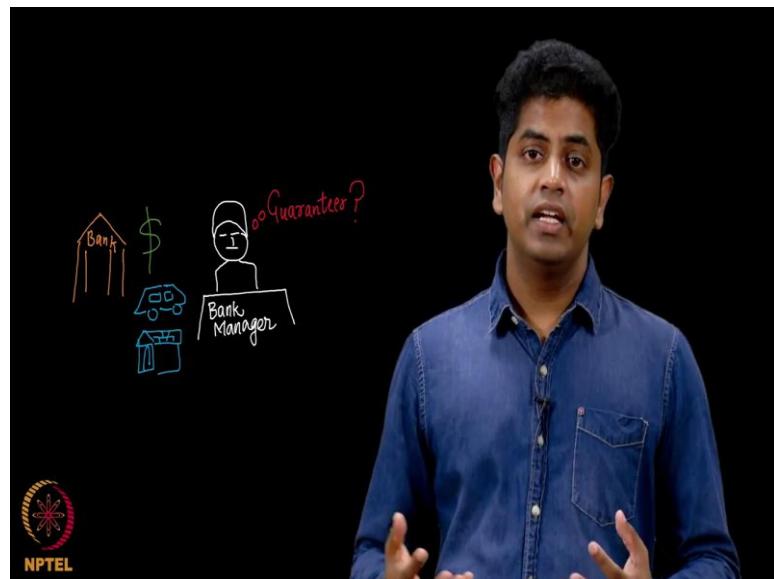
(Refer Slide Time: 01:57)



Imagine I was speaking to him and then he talks something about a good common friend of as and I realize that he is so close to him and he is close to me and then probably I will trust this person and you will give is give you money in case he wants money you see the trust increased many folds when we realize that we had a common friend imagine we realize that we have 5 common friends it is very easy to check you know it is very what are the what are the what is the likelihood that you meet a stranger on the road and that person knows 5 people whom you know he keep probably is not trying to cheating you sort of feel like what you say? You sort of feel the trust in him and you are ready to give him many money some money at least.

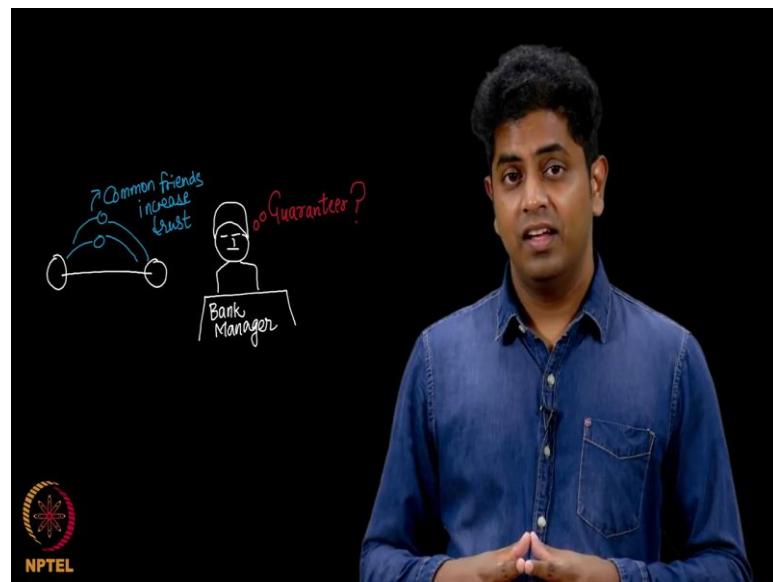
As a postal let say case where you know this person just now you got a know this person just now we got to know of this person just now and there is no common friends. So, all in all I am trying to say that if we have common friends also called is embeddedness if the embeddedness is high then there is trust probably trust is directly proportional or sort of correlated with the number of friends that we have.

(Refer Slide Time: 03:16)



Why let us illustrate a couple of examples you see you go to a bank manager asking for loan to buy let say a car or a house bank managers generally expect you to get someone who can act as surety for you guarantee here for you why does they do that they want a third person involved. So, that the chances of you being a defaulter becomes less in case you became a defaulter they can ask the third person.

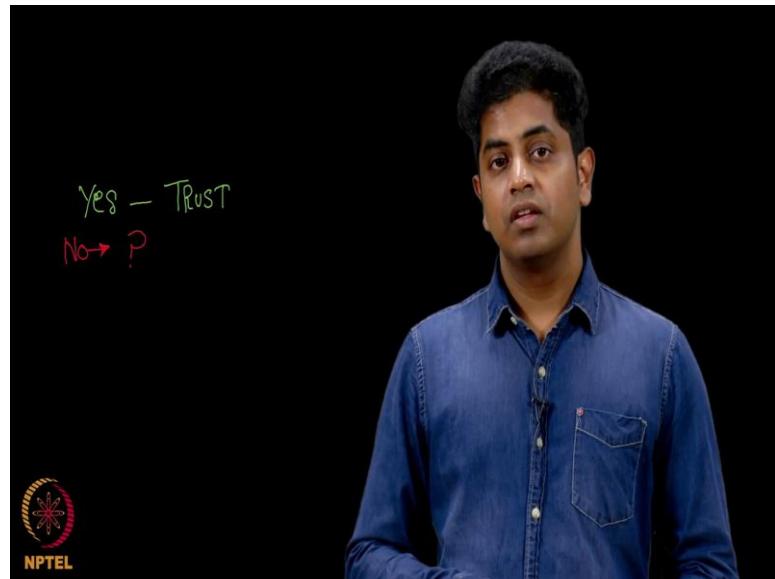
(Refer Slide Time: 03:48)



A similar situation arises in friendships 2 when 2 people are friends and they do not have any common friends the trust between them is a lot less why. So, think for a minute if I do any mischief for mistake or any anything unto words to the other person he has nobody to pull in and ask for justice generally if 2 friends fight they involve a common friend and it is look at this, this is happening between us resolve it at least in the fear of. So, many common friends two friends may not cheat, it is all these psychological factors involved where which results in having more common friends means more trust having less common friends means less trust.

Now, do you think every single relationship should have high embeddedness I repeat what is embeddedness? Embeddedness is not with respect to a person it is with respect to a person's friendship with another person this edge when we say embeddedness it is about an edge it is not about a node correct here is my question do you think it is always important to have high embeddedness in a relationships.

(Refer Slide Time: 05:09)



The answer is yes and no yes because I just told you why it is important for us to have high embeddedness the relationship gets stronger there is trust between 2 people if the relationship is high embeddedness of the relationship is high no because not always is high embeddedness good how why what we will discuss in our next lesson.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

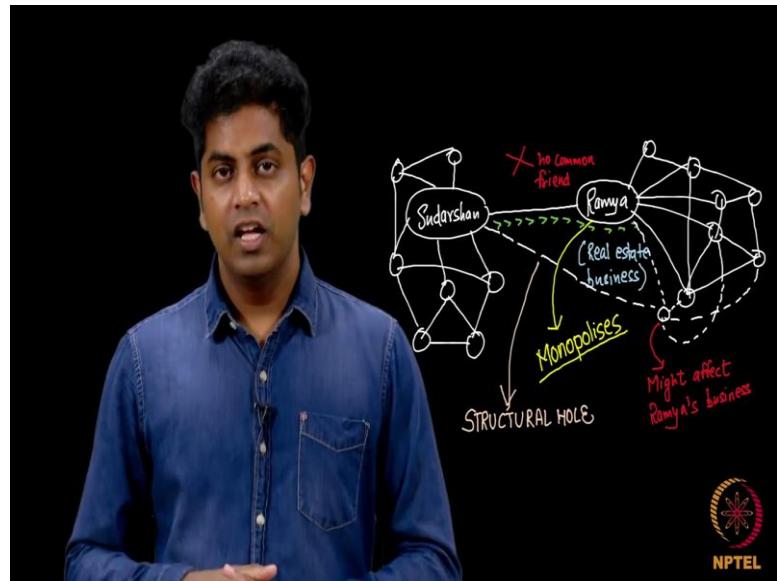
**Lecture – 33**  
**Strong and Weak Relationships**  
**Structural Holes**

(Refer Slide Time: 00:12)



Assume I have 2 friends, one by name Chethan, the other one by name Ramya, with Chethan I have high embeddedness, he is my colleague, we are very good friends, we have a lot of common friends. As I told you common friends means more trust; it is actually the other way round by other way round I mean the definition is slightly reversed the miss trust is not there the definition of trust here is there cannot be mis trust, right. So, me and Chethan I am a lot of common friends. So, we cannot cheat each other right there is a high penalty we pay if we cheat each other.

(Refer Slide Time: 00:51)



So, high; the higher the embeddedness more is the trust that is what we saw in our previous lesson, now me and Ramya do not have any common friends, now is the dangerous a sort of yes it is to some extent as you know where the same old theory if you do not have any mutual friends it is fraught with the danger of me cheating that that person if we are into a business transaction or money transaction right it sort of intuitive not always true, but sort of intuitively it is mostly true will this be of any advantage that I know Ramya, but our relationship is not high on embeddedness we do not have any common friends me and Chethan have a lot of common friends me and Ramya do not have any common friends it can also be advantages why because imagine Ramya is into real world business.

Look at this graph Sudarshan that is me and Ramya are friends and look at the graph here Sudarshan's community is very different Ramya's community is very different. In fact, for suggestion to reach out to anyone that that side in that community he must go through Ramya and imagine Ramya is a real estate agent if I want to buy a house I should consult Ramya because she is the only one who is into real estate business I do not know anyone who is into real estate anyone else are probably on the other community and I should go through Ramya only if I want to contact them.

So, Ramya's sort of monopolizes here. So, for a fact that we have low embeddedness imagine we had a common friend may be that common friend is common because he is

slightly on the other community side you see here is a common friend a new node between me and Ramya and that fellow is also friends with people from the other community right may be through him I can reach some other real estate agent correct. So, the fact me and Ramya have zero embeddedness is a huge advantage to Ramya in a situation where she is going to harness this fact that I do not know anyone from the other side community.

This is very similar to Granovetter's weak ties theory. So, if she is a connection to me for the other world do you see what is called a whole here whole in this graph structure where a lot of people on this side they should if they want to contact someone from the other side they have to go through a long chain of path. In fact, Sudarshan has to go through Ramya to contact anyone any friend of Sudarshan as you can see should go through Sudarshan through Ramya to contact the other world Ramya holds a very strong position here this concept is called the structural holes there are structural holes in these friendship networks in real life transaction networks in business transactions.

We some people do not want come people from this side community to meet anyone from that side community Ramya here acts more like a broken she ensures that nobody from this side gets know anyone from that side because she will lose business in case anyone here gets know anyone that side.

So, high embeddedness in the context of personal friendship adds in trust even in the context of business ads in trust higher the embeddedness more the trust. So, the relationship benefits, but in such a situation a huge structural whole exist in the network structure and whenever there is an edge between a people like Sudarshan and Ramya, Ramya benefits from this; any transaction between Sudarshan and her simply because she monopolizes.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture – 34**  
**Strong and Weak Relationships**  
**Social Capital**

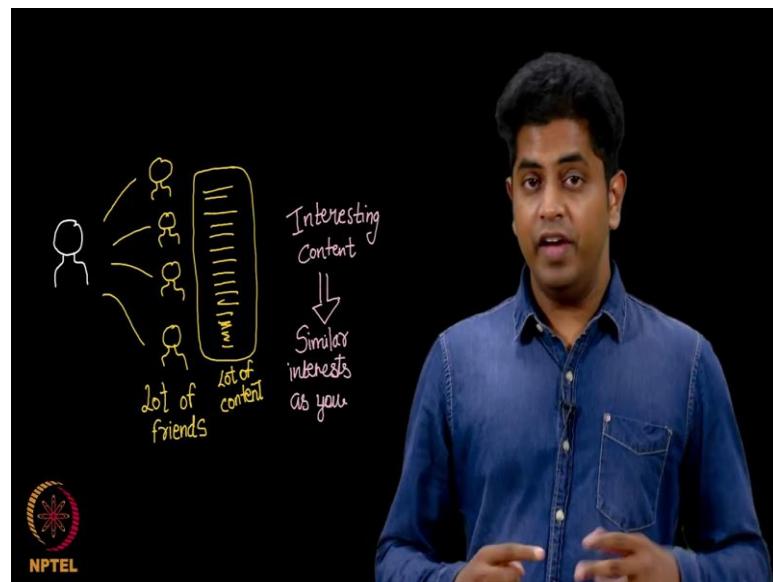
Let say Facebook wants to become popular, they are already popular assume they want to become even more popular what should they do.

(Refer Slide Time: 00:18)



What is one nice factor about Facebook because of which it is so popular, so famous? It is that if you sit in front of Facebook you find it very entertaining, its joyful to see what is happening with your friends life the pictures, the videos, the status messages, and the comments and the likes, and the re-shares right now here are two things that you should observe.

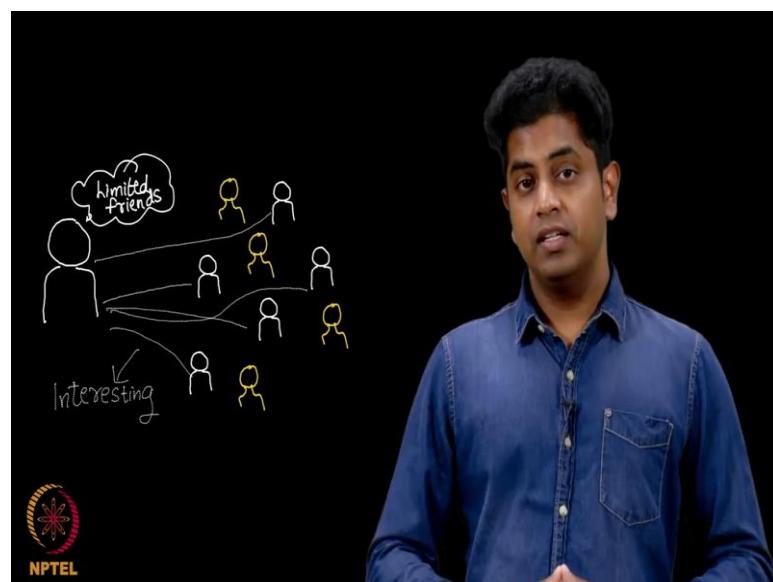
(Refer Slide Time: 00:45)



Firstly, I should have a lot of Facebook friends, so that is a lot of content on my news feed. Firstly, it is the content. Secondly, it should be interesting.

So, interesting content lot of content interesting content; when will you get these two things? If you have the kind of friends on your friend list who are whose interest are similar to that of yours similar or least they are something exciting to share, with this in mind Facebook may want to ensure that new friends that you make not only should Facebook make you have.

(Refer Slide Time: 01:30)



Not only should Facebook ensure that you have new friends, it should ensure that you do not make randomly new friends who are random. You may want the Facebook to ensure that you make friends more and more friends, such that you find their shares and comments and photos remarkably interesting right. So, the point is every person tends to have some limited amount of friendship that he can maintain whom he can follow right; you better ensure that these friends whom he makes they all are interesting enough right.

(Refer Slide Time: 02:15)



So, what I am trying to say if I am allowed to make a bunch of people become friends what exactly will I sprinkle the edges; of course, I cannot make everybody become friends with everyone else, it comes with its own a bunch of problems for example, a friendship budget is very minimum, we cannot have everybody as our friends. So, if god were to be sprinkling edges on a bunch of nodes, on this, this person should ensure that he has only 200 edges to be sprinkled let say 500 edges to be sprinkled on 100 people, where exactly will he put edges. So, that it becomes what is called a social capital. You want to benefit from the fact that you are creating a social network, the output is at its best for Facebook the output is a lot of attention for the user on their interface.

So, as I told you one should have a lot of friends, one should also have exciting friends. So, that they should have exciting stuff that is what Facebook like portal we will try to

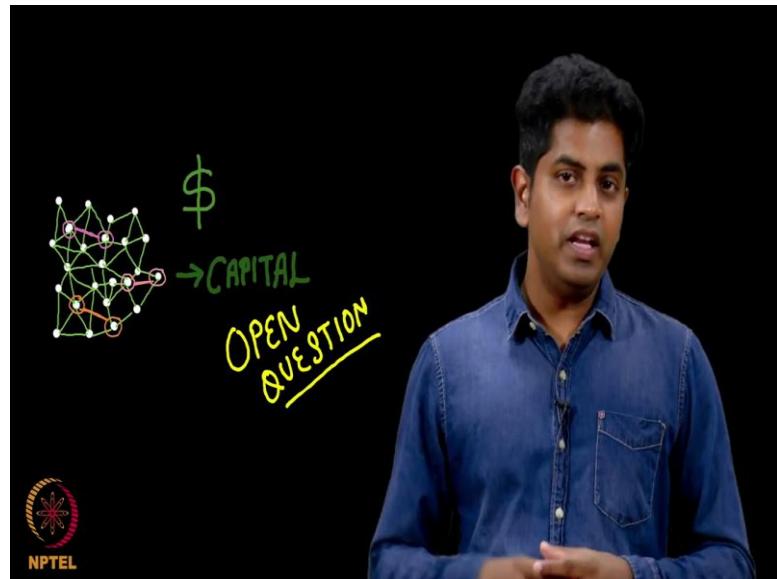
maximize. Twitter like portal would want you to follow a whole lot of people and it would want ensure that you follow those people who is newsfeed is interesting to you right where other what is called the time line is interestingly. Now let shift guess and look at a different problem.

(Refer Slide Time: 03:45)



In your family if you were to enhance the relationship between two people, if you given a budget of enhancing the friendship between two people, which is that pair which you will like to enhance maybe if you can put an edge between these two people who are always fighting with each other, because of which is a lot of commotion is a family probably would want to enhance their relationship.

(Refer Slide Time: 04:13)



But an example would be this assume this is a team of 20 people there is some kind of complicated friendship between them, there are this bunch of pairs let say three pairs of people who are who do not like each other, and you have some budget, you have some money to spend to ensure that their relationship gets enhanced what will you do? Maybe me as CEO of this company with 20 people, I will ask these three pairs of people to go out and have let say lunch together or I will give them free coupon to probably go to a water park and then play and then sort of bond well right.

So, when I have limited money how will I ensure that I put some nice edges between the existing nodes, so that I create capital out of the network. All in all the fact here is the social network itself is a capital sometimes the reason why an organization works its best is because of the complicated network structure between people which probably is yielding the best possible productivity right. So, how do I ensure that I maximize this is an open question, it is not very easy to come out with let say proto call where this kind of friendship network is yields maximum benefit.

(Refer Slide Time: 05:45)



As an example if there is a society of some thousand people all are friends with each other, is there any fun in playing a foot ball game between two teams.

As an example, if I have my son and daughter playing chess with each other, who will I cheer for I must take a side and cheer for this person and hope that the other person loses. I cannot do it when I love two people equally well there must be two communities where a lot of love is within and not across, so that you can form two football teams and they can play with each other. While I say that a lot of friendships unity is good; at the same time there should be 100 percent unity. 100 percent unity as you know is boring, I gave you foot ball gave example. So, this is called closure versus brokerage.

(Refer Slide Time: 06:39)



Closure means friends friend should become friends, when a friends friend become is a friend the network completely become sort of complete everybody will be friends with each other is not it. Here is a friends friend you become friends with him, here is a friend friends you become friends with him so on and so forth you see this sort of friends friend becoming a friend cascades and finally, everyone will become friends and there is no fun here, but is closure. Brokerage means there should be an edge, as I told you in my previous lecture me and Ramya. Ramya acts as a bridge here she benefits with the fact that I am we do not have any common friends, it is not easy for me to reach the other side of the community right Ramya is the bridging gap.

So, there has to be what is called the structural holes two at the same time some closures should also happen. As an within the previous example me and Chethan were good friends we had a lot of common friends, we benefit with that kind of relationship me and Ramya are of course friends, but it is a sort of a local bridge it is a bridge she is a bridge for the for me to connect to the other community that is also important brokerage is important closure is also important. In general the bigger question is given a choice for you to come out with a nice network, I will give you 100 people and I will give you a 1000 friendships and ask you place this friendships properly so that this particular 100 friends maximum yield maximum benefit if they work in a team.

Let say how will you sprinkle this friendships right is seems to be a tough question to answer, there has been a whole lot of research in this direction, but it is not very well understood what should be the exact social structure which will yield the maximum benefit.

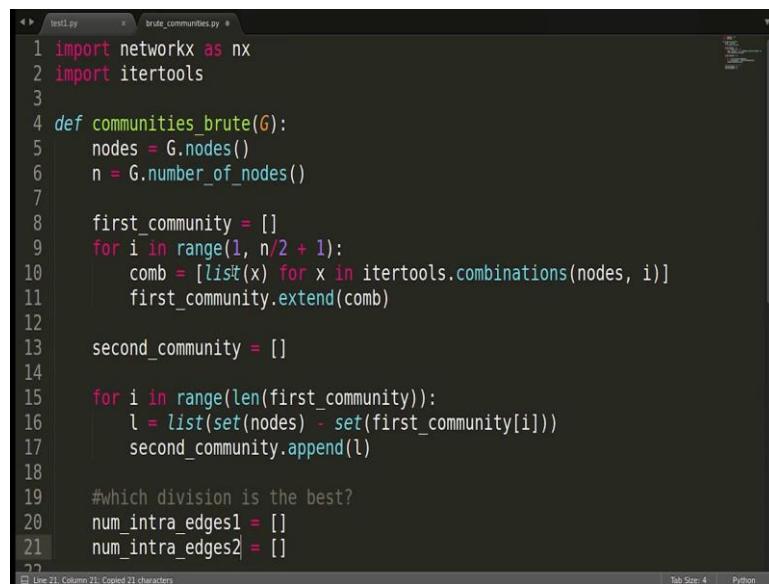
**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture - 35**  
**Strong and Weak Relationships**  
**Finding Communities in a graph**  
**(Brute Force Method)**

Hi everyone! In this video we are going to divide the nodes of a given graph into 2 communities using a Brute Force approach. Now Brute Force approach means we will be trying all possible divisions of the nodes of the graph into 2 parts and after that we will check which of these divisions is the best? Now how do we check which of these divisions is the best? As per the definition of communities, the nodes which are a part of a community; they form a lot of connections to the nodes of the same community and they form very less connections with the nodes of the other communities.

So, here we are going to make use of this property and we are going to find the best division out of all possible divisions and that is what you will display. So, let us get started.

(Refer Slide Time: 00:59)



```
test1.py      brute_communities.py
1 import networkx as nx
2 import itertools
3
4 def communities_brute(G):
5     nodes = G.nodes()
6     n = G.number_of_nodes()
7
8     first_community = []
9     for i in range(1, n/2 + 1):
10         comb = [list(x) for x in itertools.combinations(nodes, i)]
11         first_community.extend(comb)
12
13     second_community = []
14
15     for i in range(len(first_community)):
16         l = list(set(nodes) - set(first_community[i]))
17         second_community.append(l)
18
19     #which division is the best?
20     num_intra_edges1 = []
21     num_intra_edges2 = []
22
```

The screenshot shows a code editor with two tabs open: 'test1.py' and 'brute\_communities.py'. The code in 'brute\_communities.py' is a Python script that defines a function 'communities\_brute' to find communities in a graph using a brute-force approach. It imports 'networkx' and 'itertools'. The function takes a graph 'G' as input and initializes two lists, 'first\_community' and 'second\_community', both starting as empty lists. It then iterates over all possible sizes of the first community from 1 to half the number of nodes. For each size, it generates all combinations of nodes and adds them to 'first\_community'. The remaining nodes are added to 'second\_community'. Finally, it initializes two empty lists, 'num\_intra\_edges1' and 'num\_intra\_edges2', which likely represent the number of intra-community edges for each division.

So, let me import networkx. So, let me create a function here, let us pass the graph here. So, we are given the graph object, our aim is to divide the nodes of this graph into 2

partitions. So, let us first get the nodes of this graph. So, we can make use of this function G.nodes you might be knowing that this returns a list of the nodes of the graph let us also check the total number of nodes.

So, we can write G.number\_of\_nodes, now our aim is to divide the nodes of this graph into 2 communities; let us call them first community and second community now on. So, we must divide the nodes this list of nodes into first community and second community. Since you have to try all possible cases, we would have to try all those cases where the first community has one node and the second community has  $n - 1$  node. All those cases where the first community has 2 nodes and the second committee has  $n - 2$  nodes; and all those cases were the first community has 3 nodes and second community has  $n - 3$  nodes and so on.

So, we must check for all these possible cases. So, if we find the combination of nodes that are going to fall in the first community, it is very easy to get the nodes which are going to fall in the second community because they are going to be all the remaining nodes. So, our aim is to get all the nodes that may fall into the first community; basically, all the combinations of the nodes that may fall in the first community. So, how do we do that? To get the combination, there is a function in the package called itertools. So, I am going to import that package and function name is combinations.

(Refer Slide Time: 03:39)

```
In [2]: import itertools
In [3]: itertools.combinations([1,2,3,4],2)
Out[3]: <itertools.combinations at 0x7fe4b975c208>
In [4]: for i in itertools.combinations([1,2,3,4],2):
...:     print i
...:
(1, 2)
(1, 3)
(1, 4)
(2, 3)
(2, 4)
(3, 4)
In [5]: for i in itertools.combinations([1,2,3,4],2):
...:     print list(i)
```

Let me show you the functionality of this function first. So, let me show you on ipython console how this work function works, let me import itertools here and let me show you the functioning of the combinations function itertools dot combination. So, we pass 2 parameters in this function; first is the list from which we want the combinations and second is some number that we pass which goes from one to the length of the list. So, length of the list is 4 here, we can pass any number from 1 to 4 here let me pass 2 over here.

So, what it does is it returns all possible combinations out of this list which are of length 2. So, let me run this, this another point to note here is that this function does not return a list it returns an object, which has tuples of the combination as in the combinations are in the form of tuples. So, when if you want to see that we will write maybe we can use for loop for i in let me use this for i in this object print i. So, it has returned the combinations in the form of tuples, since we are working in list, we would want to convert them into list. So, we can do that by just writing list over here. So, now, it is returning in the form of list.

(Refer Slide Time: 05:15)

```
(3, 4)

In [5]: for i in itertools.combinations([1,2,3,4],2):
    print list(i)
    ...
[1, 2]
[1, 3]
[1, 4]
[2, 3]
[2, 4]
[3, 4]

In [6]: for i in itertools.combinations([1,2,3,4],3):
    print list(i)
    ...
[1, 2, 3]
[1, 2, 4]
[1, 3, 4]
[2, 3, 4]

In [7]:
```

Similarly, if you want all the combinations which are of length 3 then you will write like this. So, our aim here is to get all possible combinations of the nodes list which are of length 1 2 3 and so on. So, let us get back here we are going to put all the combinations in a list, let me create a list first say first community. Now I am going to start a loop for i

in range one to what is the length of our list that is  $n$ . So, I will write  $n/2 + 1$ . I will explain why I get this. So, I will store these combinations that this function will written in the form of a list and since it is it returns the tuples, I will pass convert them into list. So, I will write list of  $x$  for  $x$  in whatever it returns.

Let me also pass the parameters. So, the parameters are going to be the nodes list and  $i$ . So, as you can see the  $i$  is ranging from 1 up to  $n/2 + 1$ . So, why did we do  $n/2$  here? This is because the case is where the first community has  $k$  nodes, and the second community has the remaining  $n - k$  nodes and the cases where the first community has  $n - k$  nodes and the second community have  $k$  nodes. So, these cases are going to be the same these divisions are basically the same. So, there is no point of checking all the  $n$  to 1 for the first community combinations. So, that is why we are starting from 1 and we are stopping at  $n/2$  that is if the list nodes list has 10 elements. So, we will start from 1, 2, 3, 4, up to 5 and we are not going to check for 6, 7, 8, up to 10 cases, this is because they are going to give us the same divisions and there is no point checking for the extra cases.

And the reason why we put plus one here is that you might be knowing the functionality of this range function if you pass  $1, n$  in range it goes from 1 to  $n - 1$ . So, since we wanted to go exactly up to  $n/2$  we are going to add plus 1 over here. So, what is this comb variable going to have; this is basically going to have a list this comb is a list of lists where all the list is having  $i$  elements. So, as I showed you on the screen comb is going to have all these 3 all these 4 lists for the value 3 and for the value 2 it is going to have all these list, and after this our aim is to add all these combinations to this list first community. So, what I am going to do here is first community dot extend and I am going to add all the list that I am getting from this comb based.

(Refer Slide Time: 08:57)

```
[1, 2, 4]
[1, 3, 4]
[2, 3, 4]

In [7]: l1 = [1,2]

In [8]: l2 = [3,4]

In [9]: l3 = [5,6]

In [10]: l1.append(l2)

In [11]: l1
Out[11]: [1, 2, [3, 4]]

In [12]: l2.extend(l3)

In [13]: l2
Out[13]: [3, 4, 5, 6]      I

In [14]:
```

Now, let me tell you why I am using this extend function here and not the append function, let me show you an over here why i used extend let me take some example lists; let me take another list and yet another list. So, if I do l1.append(l2) what will happen let me print l1. See how it is adding the elements my aim was basically to add this 3 4 2 this list which is having 12 already. So, when I write l1.append(l2) were l2 is itself a list this is how it does.

In case I want to add only the elements we make use of the function extend. So, let me show you l2.extend(l3). So, let me print l2, now this is how it works and this is how I wanted to be; that is why I have used this function extend this is because comb is a list of list and I want to add the elements only to this first community. So, we are using extend function here now we have stored all possible combinations of nodes in in this list first community, now we should also do the same for the second community as I am going to create a list here, now this is going to be very easy because all the remaining node are going to fall into this community.

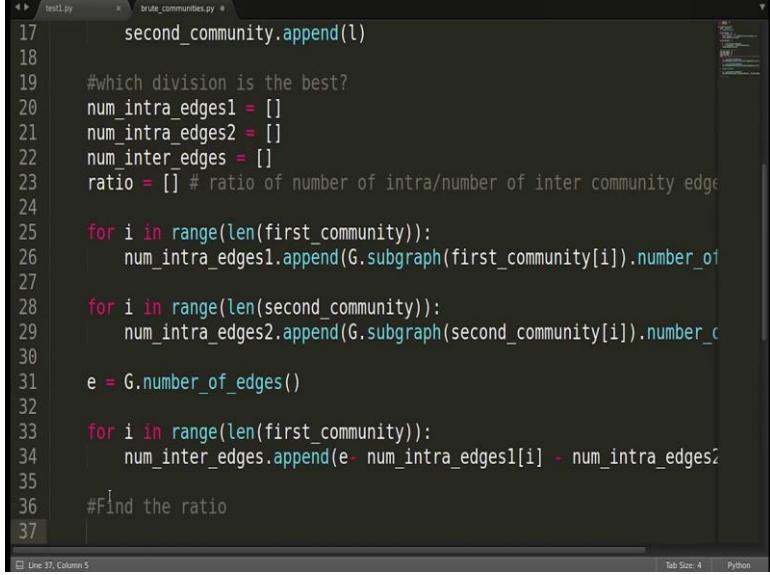
Now, you see we have the list of all the nodes we have the list of nodes which are going to fall in the first community, we must get the list of remaining nodes in the second community. So, how do we do that? Basically, we must subtract the elements of first community from the elements in nodes. So, how do we do subtraction in list? If you want to do subtraction set is the appropriate data structure. So, what we can do is we can

convert the list into set and after that we can apply the subtraction operation for example, I can write set nodes minus set first community. So, whatever it will give me it will give me the difference of the elements from nodes, nodes minus first community it will give me the difference of elements and that will also be in the form of set. So, I can later convert them into a list. So, that is how I will get the second community lists. So, let me store it here.

Now, since we have to do it for all the elements of first community, we have to start a loop here. So, I will write for i in range. So, I am going to do it for all the elements of first community. So, I will write length of first community all right. So, we are going to do it for all the elements of the first community and once we get the list in l we have to append it to the second community list. So, I will write second community dot append l. So, now, we have all the combinations in this list first community and all the combinations of the second community in this list second community, and please note that these 2 lists are themselves list of lists. So, we have the divisions in these lists, and we have to find which division is the best.

Now, as I told you as per the definition of community communities, the nodes which are a part of one community have a lot of connections with the nodes in the same community that is there are going to be a lot of intra community edges and there are going to be very less intercommunity edges. So, if a division is good then the number of intra community edges should be high and the number of intercommunity edges should be less. So, this is what we are going to use here. So, we are going to keep track of the number of intra and intercommunity edges for all the possible divisions that we have made so far.

(Refer Slide Time: 13:49)



```
test1.py          brute_communities.py
17     second_community.append(l)
18
19     #which division is the best?
20     num_intra_edges1 = []
21     num_intra_edges2 = []
22     num_inter_edges = []
23     ratio = [] # ratio of number of intra/number of inter community edges
24
25     for i in range(len(first_community)):
26         num_intra_edges1.append(G.subgraph(first_community[i]).number_of_
27
28     for i in range(len(second_community)):
29         num_intra_edges2.append(G.subgraph(second_community[i]).number_c
30
31     e = G.number_of_edges()
32
33     for i in range(len(first_community)):
34         num_inter_edges.append(e - num_intra_edges1[i] - num_intra_edges2[i])
35
36     #Find the ratio
37
```

The screenshot shows a code editor with two tabs: 'test1.py' and 'brute\_communities.py'. The code in 'test1.py' is as follows:

```
test1.py          brute_communities.py
17     second_community.append(l)
18
19     #which division is the best?
20     num_intra_edges1 = []
21     num_intra_edges2 = []
22     num_inter_edges = []
23     ratio = [] # ratio of number of intra/number of inter community edges
24
25     for i in range(len(first_community)):
26         num_intra_edges1.append(G.subgraph(first_community[i]).number_of_
27
28     for i in range(len(second_community)):
29         num_intra_edges2.append(G.subgraph(second_community[i]).number_c
30
31     e = G.number_of_edges()
32
33     for i in range(len(first_community)):
34         num_inter_edges.append(e - num_intra_edges1[i] - num_intra_edges2[i])
35
36     #Find the ratio
37
```

So, what you going to do now is which division is the best. So, this is what we are going to check now, since we have to keep track of the number of intra community edges in the first community we will create a list for it and we have to keep a track of intercommunity edges in the second community we will create a list for it and we also have to keep track of the number of intercommunity edges. So, we have these 3 lists, now apart from this to check which division is the best we are going to find the ratio where (Refer Time: 14:05) in the numerator will have the sum of intra community edges, and in the denominator will have the intercommunity edges.

So, whichever division has highest value of this ratio will mean that in that community inside community; there are dense connections and outside community there are sparse connections. So, we are going to find this ratio for every division that we have made so far. So, I am going to create this list as well, let me write here what this means ratio of intra divided by this is basically number of intra and number of inter community edges; alright.

Now, we have the nodes which are falling into the first community and the nodes which are fall into the second community, arrange to find the number of edges that are existing amongst the nodes in the first community and the same for the second community as well. Now how do we find the number of edges amongst the nodes in a given community because we have only as of now, we have only the list of nodes. So, for this purpose we

are going to make use of function that is subgraph, which is given which is inside the graphic object. So, we will make use of a the subgraph function let me tell you the syntax for this function. So, this function is available in graph object. So, I will write G.subgraph and here I will pass the list of the nodes. So, in our case the list of nodes is kept in first community and second community also.

So firstly, we are doing for the first\_community and for the first community also let us do it for i we are going to start loop here, for i in range basically whatever we are doing we are going to do it for all the lists given in the first community. So, we will start it for i in range will write length of first community yeah. So, I was saying you about this sub graph function this function basically takes as input a list of edges, and it returns a graph object itself which is an induced subgraph over this list of elements, over this list of nodes. So, whatever it returns is basically a graph and whatever operations you can apply on graph you can apply on the written value of this function as well. So, if you want to find the number of edges, we can just use the formula that is number of edges. Since this is itself or graph object as I told you subgraph returns an induced subgraph which is a sort of graph itself. So, you can apply this function number of edges on the return value.

Now, whatever it returns we must keep store that value in this list it is number of intra edges, because this will give us the number of intra edges in that subgraph. So, I will write number of intra edges in first\_community.append all right. So, let me explain you once again what this statement is doing. So, we have the nodes that are falling into the first\_community and these are here first\_community right. We are starting this for loop from i goes from zero to the length of the first\_community , basically we are checking it for all the possible combination. So, we have this list of nodes which we are passing into this subgraph function which. So, this function will return an object of graph itself. So, that will basically be an induced subgraph over these nodes. So, since that is a graph, we are calling this function number of edges to find the number of edges which are existing amongst those nodes.

(Refer Slide Time: 18:31)

```

7
8     munity = []
9     range(1, n/2 + 1):
10    = [list(x) for x in itertools.combinations(nodes, i)]
11    t_community.extend(comb)
12
13 community = []
14
15    range(len(first_community)):
16    list(set(nodes) - set(first_community[i])))
17    nd_community.append(l)
18
19 vision is the best?
20 a_edges1 = []
21 a_edges2 = []
22 r_edges = []
23 [] # ratio of number of intra/number of inter community edges
24
25 range(len(first_community)):
26 intra_edges1.append(G.subgraph(first_community[i]).number_of_edges())
27

```

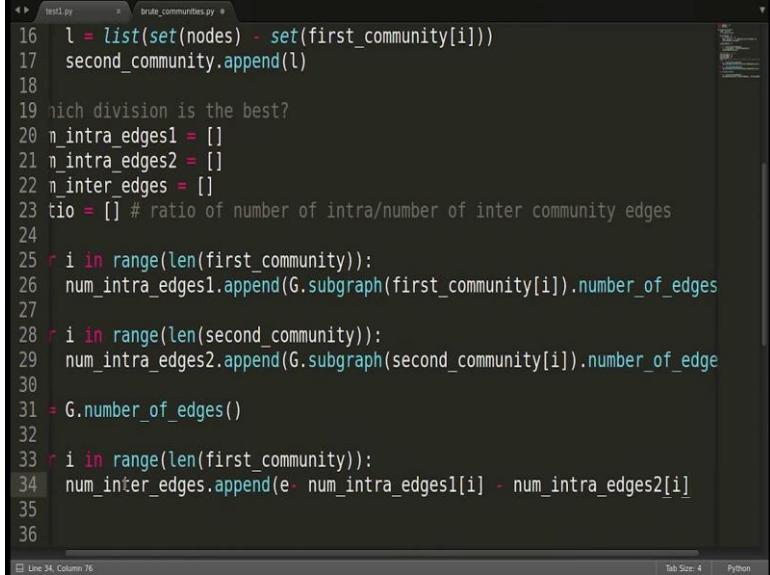
For example, if you pass one 2 3 here this will tell us the number of edges amongst these 1 2 and 3 nodes ,if there is an edge from 3 to 4 let us say, that will not be counted here that is because 4 is not a part of this subgraph. So, that is how we will get the number of edges and that information we are storing in this list as we know created this list number of intra edges one that is for first\_community . So, we are appending that information here.

Now, the same thing you have to do for the second\_community as well. So, we are just going to copy paste it and I am going to write second\_community here and information has to go to this second list and here I will write second yeah, I think we are done. So, now, we have the number of intra edges in the first\_community in this list and we have the number of intra edges in the second\_community in this list, now we have to find the number of inter edges.

So, we can easily find the total number of edges in the graph right let us do that e is equal to G dot number of edges right. So, we got the total number of edges here and we have the intra edges in first\_community and we have to intra edges in the second\_community . So, we subtract these all these intra edges from the total number of edges, we will get the number of inter edges intercommunity edges. So, let us do that. So, what we will do is e minus number of intra edges in first\_community ; we are going to start a loop now minus number of intra edges in the second\_community .

And we have to do it for all values of i. So, I will start for i in range length of here we can use length of first\_community or second\_community because they are the same. So, number of intra edges or to maintain uniformity I will write first\_community itself here. So, we have to keep this information in this list that is number of inter edges. So, i like I will write it here I am sorry. So, in this list number of inter edges dot append.

(Refer Slide Time: 21:34)



```

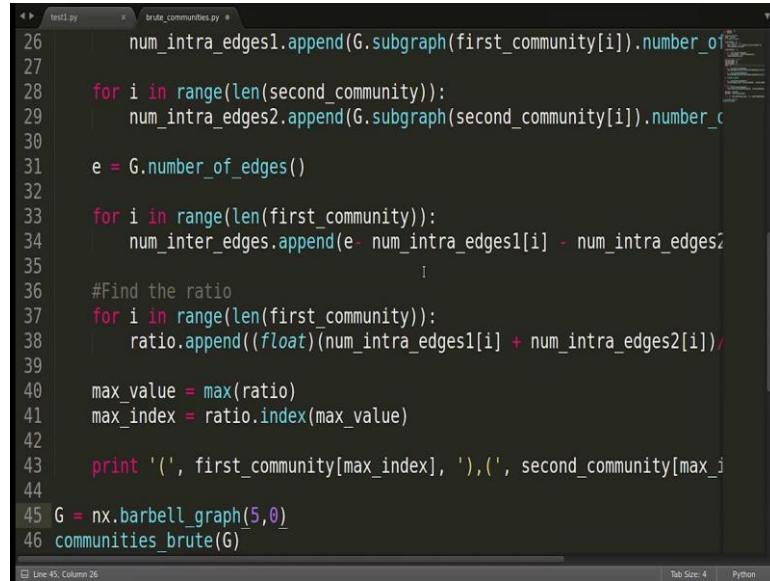
16 l = list(set(nodes) - set(first_community[i]))
17 second_community.append(l)
18
19 which division is the best?
20 n_intra_edges1 = []
21 n_intra_edges2 = []
22 n_inter_edges = []
23 rto = [] # ratio of number of intra/number of inter community edges
24
25 for i in range(len(first_community)):
26     num_intra_edges1.append(G.subgraph(first_community[i]).number_of_edges)
27
28 for i in range(len(second_community)):
29     num_intra_edges2.append(G.subgraph(second_community[i]).number_of_edges)
30
31 G.number_of_edges()
32
33 for i in range(len(first_community)):
34     num_inter_edges.append(e - num_intra_edges1[i] - num_intra_edges2[i])
35
36

```

So, we have to append this information here. So, now, we have the number of intra edges in this list, we have the number of inter intra edges in the second\_community in this list we have the number of inter edges in this list.

Next thing is to find is the ratio. So, let us find the ratio. So, again we have to start this loop am going to copy here.

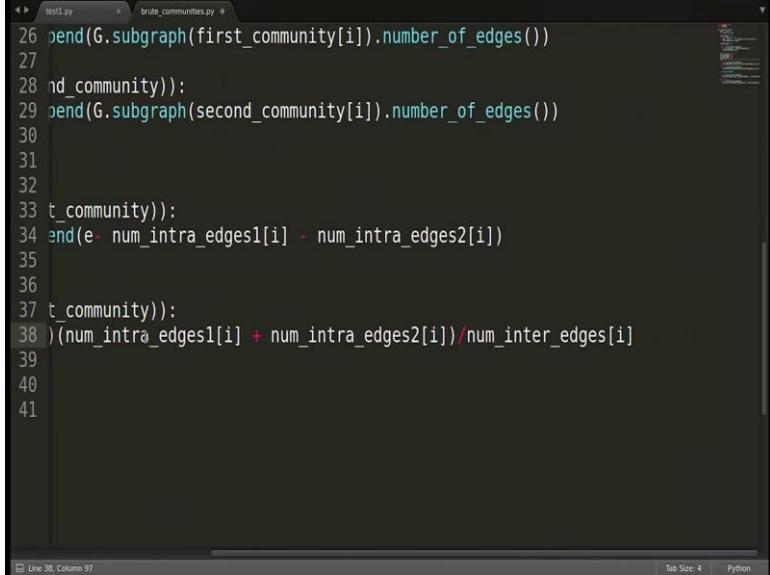
(Refer Slide Time: 22:03)



```
test1.py          brute_communities.py
26     num_intra_edges1.append(G.subgraph(first_community[i]).number_of_
27
28     for i in range(len(second_community)):
29         num_intra_edges2.append(G.subgraph(second_community[i]).number_o_
30
31     e = G.number_of_edges()
32
33     for i in range(len(first_community)):
34         num_inter_edges.append(e - num_intra_edges1[i] - num_intra_edges2[_
35
36     #Find the ratio
37     for i in range(len(first_community)):
38         ratio.append((float)(num_intra_edges1[i] + num_intra_edges2[i]) / e)
39
40     max_value = max(ratio)
41     max_index = ratio.index(max_value)
42
43     print ('(', first_community[max_index], ','), ('', second_community[max_i_
44
45 G = nx.barbell_graph(5,0)
46 communities_brute(G)
```

I am sorry. So, we have to find the ratio for all these possible combinations; ratio is going to be the divisions of number of intra edges with the number of inter edges. So, numerator is going to be the total number of intra edges, we have intra edges in 2 lists. So, we are going to sum them up the value from this list plus the value from this list. So, I will write number of intra edges 1 for the ith combination, plus number of intra edges in 2 for the ith combination. So, we have to sum them up and then we have to divide them with the number of inter edges for the ith combination. So, since I want all the precise values. So, I will write float here and this will give me the ratio and this ratio I am going to keep in the list that I have already created by the name ratio. So, I will write ratio dot append.

(Refer Slide Time: 23:15)



```
test1.py          brute_communities.py
26     end(G.subgraph(first_community[i]).number_of_edges())
27
28     end_community)):
29     end(G.subgraph(second_community[i]).number_of_edges())
30
31
32     t_community)):
33     end(e - num_intra_edges1[i] - num_intra_edges2[i])
34
35
36     t_community)):
37     (num_intra_edges1[i] + num_intra_edges2[i]) / num_inter_edges[i]
38
39
40
41
```

So, let me explain you once again what this statement is doing, we are finding the ratio of the intranet inter edges. So, on the numerator we have the total number of intra edges and in the denominator we have the total number of inter edges, and this ratio we are just keeping in this list which is named ratio and we have already created that list. Now we have all the ratio in this ratio list and our aim is to find the list the best value of this ratio that is the maximum value of this ratio. So, what we can do is. So, let us create the variable max value is equal to. So, this is a function that you can use max function and the parameter can be the list this returns us the maximum value in the list ratio.

Now, our aim is not to get the maximum value basically, our aim is to get the index at which the maximum is coming so that we get to know; what was the best division? We can make use of the index function from the list that is ratio. So, what we can write is ratio dot index max value. So, what it will return is it will return the index of the max value here. So, I will write max index is equal ratio dot index. So, this index is what we are interested in; this is because this index tells us the division which is the best division out of all possible division.

So, now we got this max division I am sorry max index that is the best index, and in the first\_community and in the second\_community list we have the divisions of nodes into first and second committee. So, we are going to access the division out of this list and out of this list which is falling at this index. So, what we are going to do is we are going to

basically print the final result. So, I will write first\_community and which element the one which is falling at max index and similarly second\_community we are going to access the element which is falling at max index.

Now, I am going to print these. So, let me just do some representation thing here. So, after this is done and I will add a bracket here and then I will start the other 1 and this and then I will close it this is about the representation. So, this is how our lists 2 divisions will be printed divisions of nodes into 2 communities. Now the function is return we have to now call it and use it we want an example graph here. So, let me take an example graph here the function name is communities brute. So, let me call this and let me pass this graph as an example graph let us take this graph which is called barbell graph, I am going to show you what this graph looks like let us pass 2 parameters here. So, let me go back to the ipython screen and I will show you how this graph looks like.

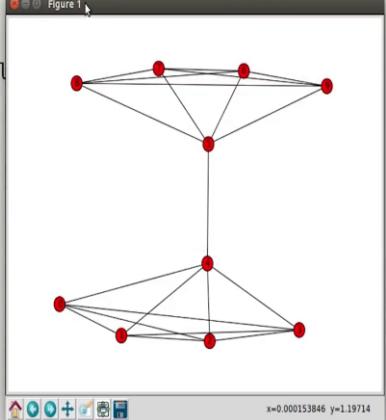
(Refer Slide Time: 26:50)

```
In [28]: import networkx as nx
In [29]: G = nx.barbell_graph(5,0)
In [30]: nx.draw(G)
In [31]: import matplotlib.pyplot as plt
In [32]: plt.show()
```

So, let us go here, here let me import the package networkx and I will show you how this graph looks like. So, I will take a graph  $G = \text{nx.barbell\_graph}$ , and I am going to pass the same parameters. Now what it indicates I will show you in the visualization. So, I will write `nx.draw` to show this I need `matplotlib`. So, I will import `matplotlib` dot my plot as `plt` and I will write `plt.show`.

(Refer Slide Time: 27:39)

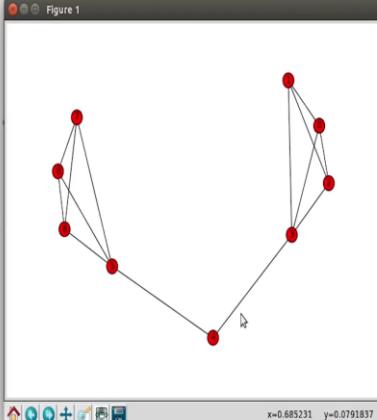
```
In [28]: import networkx as nx
In [29]: G = nx.barbell_graph(5,0)
In [30]: nx.draw(G)
In [31]: import matplotlib.pyplot
In [32]: plt.show()
```



Now this is a graph that comes out of these 2 parameters 5 and 0. So, let me explain you what this parameter indicate; the first parameter which is 5 here it indicates that there should be 2 clicks of size 5, as you can see here first click and the second click and they should be connected by a path which has these nodes which is 0 node here.

(Refer Slide Time: 28:08)

```
In [28]: import networkx as nx
In [29]: G = nx.barbell_graph(5,0)
In [30]: nx.draw(G)
In [31]: import matplotlib.pyplot
In [32]: plt.show()
In [33]: G = nx.barbell_graph(4,1)
In [34]: nx.draw(G)
In [35]: plt.show()
```



So, this is a path which is connecting these 2 clicks and it has 0 node. So, this is how it looks like let me change the parameters a bit and show you how it comes out to be.

So, if I pass 4 and 1 let us see how it looks like. So, we passed 4 , 1 which means it should be 2 clicks of size 4 as you can see, and they should be connected with the path which has one node. So, the same is showing here.

(Refer Slide Time: 28:43)

```
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_3_(Strength of weak tie  
s)/Videos$ python brute_communities.py  
( [0, 1, 2, 3, 4] ),( [8, 9, 5, 6, 7] )  
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_3_(Strength of weak tie  
s)/Videos$ █
```

So, this is one node in between. So, this is a kind of barbell graph, I plan to take this graph because the kind of community is required to visible here and it is also small. So, it would not take much time for the testing. So, I am going to take the graph.

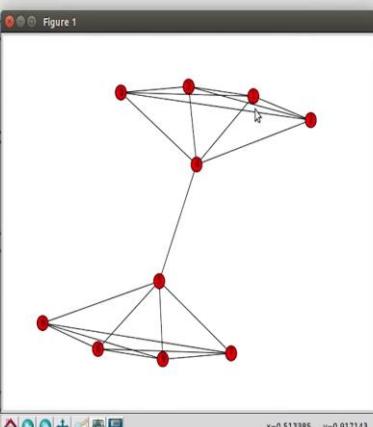
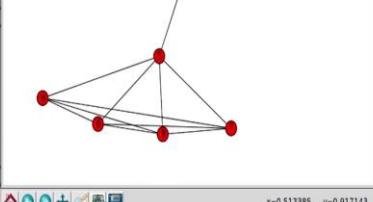
So, let us go and a test this function this is yeah. So, these are the 2 communities that we get if you remember we passed 5 , 0 which means  $5 + 5$  there are going to be 10 nodes, and these 10 nodes are divided into 2 communities where the first\_community have 0, 1, 2, 3, 4 and the second has 8, 9, 5, 6, 7. Let me quickly show you once again what kind of plot we were getting when we pass these parameters see this is what we got.

(Refer Slide Time: 29:20)

```
In [28]: import networkx as nx  
In [29]: G = nx.barbell_graph(5,0)  
In [30]: nx.draw(G)  
In [31]: import matplotlib.pyplot as plt  
In [32]: plt.show()  
In [33]: G = nx.barbell_graph(4,1)  
In [34]: nx.draw(G)  
In [35]: plt.show()  
In [36]: G = nx.barbell_graph(5,0)  
In [37]: nx.draw(G)  
In [38]: plt.show()
```

So, here you can see that communities are clearly visible that these should be the communities and our brute force algorithm is nicely able to differentiate the communities.

(Refer Slide Time: 29:28)

```
In [29]: G = nx.barbell_graph(  
In [30]: nx.draw(G)  
In [31]: import matplotlib.pyplot as plt  
In [32]: plt.show()  
In [33]: G = nx.barbell_graph(  
In [34]: nx.draw(G)  
In [35]: plt.show()  
In [36]: G = nx.barbell_graph(  
In [37]: nx.draw(G)  
In [38]: plt.show()
```

So, as you see 0, 1, 2, 3, 4, 5 are falling to one community, and the rest are in the second one. So, that is what we are getting here. So, the brute force algorithm is working fine, next we are going to see the Girvan Newman algorithm for community detection.

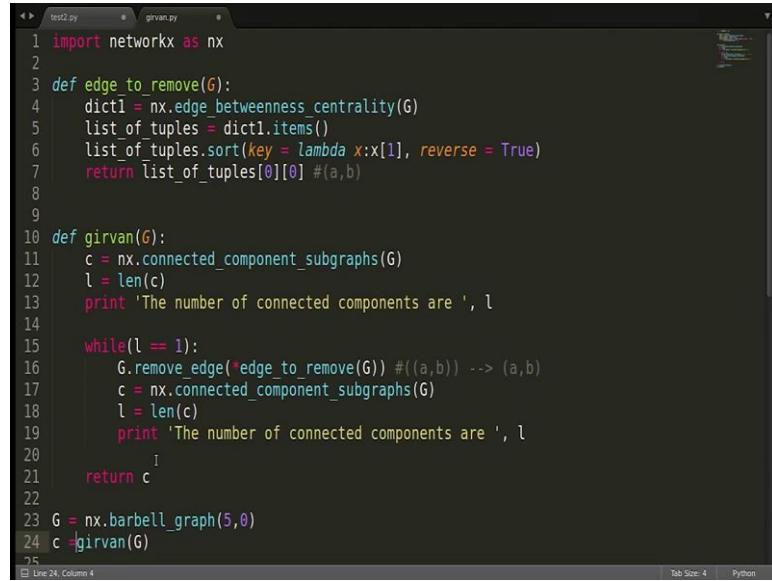
**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture - 36**  
**Strong and Weak Relationships**  
**Community Detection Using**  
**Girvan Newman Algorithm**

Hey everyone, in the previous video we divided the graph into two communities using brute force approach. Brute force approach means we tried all possible divisions of nodes into two communities which was very time-consuming process. So, if the graph is very huge it is going to take whole lot of time and incase of very large graphs it might not even run it may get stuck. So, that was not a feasible way to find communities in a given graph.

On the other hand we have this algorithm which is called Girvan Newman Algorithm and this algorithm is basically based on the concept of edge betweenness. Now the key idea is that the edges which are inter-community that is the edges which are linking the nodes from one community to the other community, they tend to have high value of betweenness now; that means, that number of shortest paths through these edges. So, if we find the edge betweenness of these edges and we keep removing them we tend to get the community structure of the graph. So, this is the whole idea behind this algorithm and this is what we are going to implement and after that will try this algorithm on the sample graph that we took in previous video that is bubble graph, and after that will also execute this algorithm on Zachary karate club as well and will see the communities that get formed.

(Refer Slide Time: 01:55)



```
1 import networkx as nx
2
3 def edge_to_remove(G):
4     dict1 = nx.edge_betweenness_centrality(G)
5     list_of_tuples = dict1.items()
6     list_of_tuples.sort(key = lambda x:x[1], reverse = True)
7     return list_of_tuples[0][0] #(a,b)
8
9
10 def girvan(G):
11     c = nx.connected_component_subgraphs(G)
12     l = len(c)
13     print 'The number of connected components are ', l
14
15     while(l == 1):
16         G.remove_edge(*edge_to_remove(G)) #((a,b)) --> (a,b)
17         c = nx.connected_component_subgraphs(G)
18         l = len(c)
19         print 'The number of connected components are ', l
20
21     return c
22
23 G = nx.barbell_graph(5,0)
24 c = girvan(G)
25
```

So, let us get started. So, I am going to import this package import networkx as nx and let us create a function for Girvan algorithm. So, I will write Girvan G. So, let us first of all check whether the graph is connected or not. So, I will write nx.connected. I am sorry connected component sorry component sub graphs G. So, what is function does connected component sub graphs this function returns the connected component of the graph as sub graphs. So, if there are two connected components in the graph G it will return two sub graphs. If there is only one connected component that is graph is connected it is going return only one sub graph and that is graphic itself.

So, let us store that in this list c. So, if the length of this list is one; that means, graph the is connected right because it return only one connected component, and if the list if the length of the list is more than one that means, the graph is disconnected and there are more than one connected components right. So, let us store the length of this list c and let us print that because we are going to keep checking whether the graph is now connected or not. So, we are going to make use of this function again and just to keep track of algorithm we are going to print whether the graph is connected or not at that particular moment.

So, I am going to print the number of connected components is. So, write 1. So, 1 is number of connected components. So, initially it will be one if the graph is connected of course. Now as I told you we will keep removing the edges from this graph G. So, the

function that we will use to remove the edges from the graph is `G.remove_edge`, and as the parameter we will pass the edge that has been removed basically we will pass two parameters first is the source of the edge and second is the target of the edge that has to be removed.

Now, the big question is which edge to remove right as I told you we are going to remove the edges based on the betweenness value. So, we have to calculate the betweenness value of the edges first, now since you have to keep removing the edges I am going to keep that code in a function. So, let me better create a function here, I will create a function say `edge_to_remove` I will pass the graph `G` here. So, let us first create this function, `edge_to_remove` and let us pass the graph here `edge_to_remove`.

Now, in order to be able to know which edge to remove we have to first calculate the edge betweenness values, and for that we have a function `nx.edge_betweenness_centrality` or of `G`. So, so edge betweenness centrality is a function in `networkx` which returns a dictionary of values. So, I am going to name it like `dict1`. So, that you know the data structure of the return value. So, what does this dictionary contain? This dictionary contains as a key the edge and as a value it contains the edge betweenness centrality of that edge correct. So, that is what this dictionary contains.

Now, what is our aim? Our aim is to sort the edges based on their betweenness centrality value. So, since dictionary does not keep track of the sorted elements basically there is no association between the element and the index. So, the elements can appear in any order. So, order is not a thing of dictionary. So, let us take the values from this dictionary and keep them in the form of a list. So, what I will do is I will write `dict1.items` and I will store these in the form of list of tuples and give a similar name to it. So, that you it is easy if your list of tuples is equal to `dict1.items`.

So, you might be knowing that these `items` function basically returns the tuples of the dictionary, where the first element of each tuple is a key and second element of each tuple is the value. So, this list is going to have tuples where the first element will be the edge of each tuple and the second element of each tuple is going to be the edge betweenness value. Now make sense for us to sort this list as we know sort this list based on the edge betweenness value right now how we do that let me show you list of tuples dot `sort` this is the function that we will use to sort this list. Now the list is having tuples

and tuples have two values, second value is the edge betweenness value and based on that only we must sort the list. So, whenever we must sort list of tuples there is syntax that we follow let me show you that, key = lambda x : x1.

Now, let me tell you why I write x : x1 and one more thing reverse is equal to true. So, this is how we sort list of tuples you can see the documentation of this lambda key word and. So, this is basically a shortcut to sort list based on either you know first value of the tuple or the second value of the tuple. So, here we want to sort based on second value of the tuples. So, I am writing x : x1, in case you want to sort the list based on the first value of the tuple you will write x : x0. Second thing is we want in the descending order that is we want the edge which has highest betweenness to be on the top. So, I will write reverse is equal to true in case you want opposite you do not need write anything here.

So, we have now the list which is sorted. So, the first element of the list is the tuple, where the first element of the tuple is the edge and the second element of the tuple is the betweenness value. What do we have to return here by this function? This function should return the edge right. So, what we are going to return is list of tuples first I am sorry first element first tuples first element which is 0 because the index start from 0 I hope I am clear here 0th with element means 0th tuple and in that tuple 0th element which is the first element that is the edge.

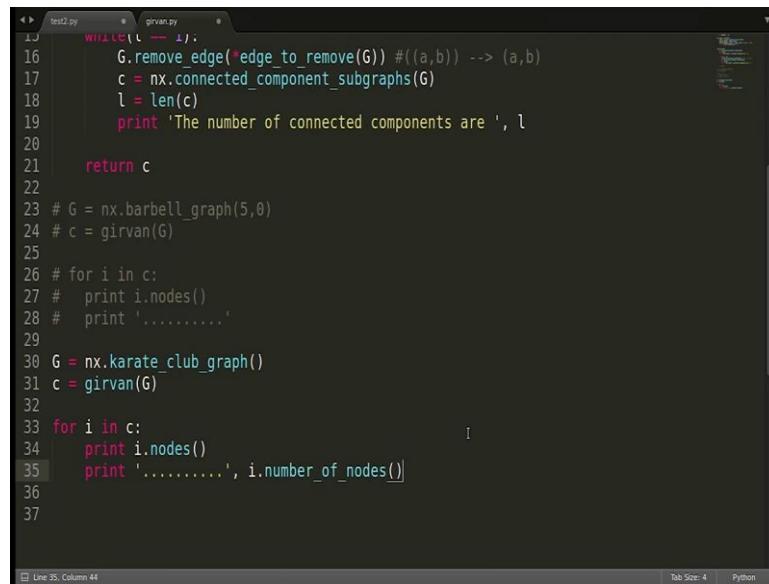
So, we are going to return this alright I am sorry return this, alright. So, you know which edge to remove now, there is one catch here which should be noted this edge to remove is going to return tuple right let me write here as a comment is going to return the edge which is of the form a comma b where a is the source and b is target . So, when we pass it here it becomes like this a comma b right this becomes this comes over here instead of this which will give you syntax error. So, what is needed is we instead of this we want this thing a comma b I am sorry a comma b, basically out of the tuple we have to extract the edge. So, what we can put here is star, when we write star here, we get the content of the tuple and not the tuple ok.

So, we now we got the edge which has to be removed after we remove the edge from the graph, we again have to check whether the graph is corrected or not right. So, again we will make use of this function and again we will find the length and again we make print whether the as in what are the number of connected components. So, I am going to use it

here, we have to keep removing the edges until the graph becomes disconnected. So, which means we have to start while loop here. So, I will write while l is equal to 1 which means as long as the graph is connected keep doing this and as soon as this l becomes 2 right l becomes 2 means the graph becomes disconnected and number of connected component in the graph is 2 the control will come out this loop and here we have two connected components correct.

So, and these two connected components are there in this list c which is what we are going to return here. So, will write return c. So, we are going to return this c and we will see in while we are calling it we will see how we can extract the connected component out of this c. So, I think we are done with the functions let us try to make use of them, let us take the example graph that we took in previous video nx.barbell\_graph with the same values I will take as a from 5 , 0 and I will pass them in this function, and let me take the return values now c is going to have the two connected components right. So, and these two connected components in the form of graph which means we can apply all the functions of graph on these connected components because they are actually graph objects.

(Refer Slide Time: 12:45)



```

15     while(l == 1):
16         G.remove_edge(*edge_to_remove(G)) #((a,b)) --> (a,b)
17         c = nx.connected_component_subgraphs(G)
18         l = len(c)
19         print 'The number of connected components are ', l
20
21     return c
22
23 # G = nx.barbell_graph(5,0)
24 # c = girvan(G)
25
26 # for i in c:
27 #   print i.nodes()
28 #   print '.....'
29
30 G = nx.karate_club_graph()
31 c = girvan(G)
32
33 for i in c:
34     print i.nodes()
35     print '.....', i.number_of_nodes()
36
37

```

So, I can start to look here, or I can manually do that as well. So, I will write for i in c. So, c was less which has two elements. So, I will write for i and c let me. So, what is our aim are our is to print the nodes which are falling in the first community and in the

second community, community means connected component here right. So, let us print i because i is a graph I can use i.nodes and that is all. So, i goes from 0 to 1 and both the connected components nodes will be printed here let me put some separately here something anything.

(Refer Slide Time: 13:39)

```
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_3_(Strength of weak tie
s)/Videos$ python girvan.py
The number of connected components are 1
The number of connected components are 2
[0, 1, 2, 3, 4]
.....
[8, 9, 5, 6, 7]    I
.....
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_3_(Strength of weak tie
s)/Videos$
```

So, we should get the two connected component the nodes of the two connected components by this let us see, let us check the program what was the name yeah right. So, you see the number of connected components initially was one, and then we removed some edge and the number of connected components became 2.

So, it was very small graph. So, quickly it conversed as we as see how quickly it conversed as compared to the brute force method. In that method we it took good amount of iteration to finally, give us the communities. Here it quickly given as the communities and as we saw in the previous video the communities were 0 1 2 3 4 in the first one and 8 9 5 6 7 in the second community. So, it is giving us correctly the communities in the given graph let us go back to our code and we can also check the karate clubs current communities here ok.

So, let us check this code for the Zachary karate club as well, let me copy paste this and let me common this. So, let us takes the Zachary karate club. So, this is the function that we can use karate club graph. Let me tell you can also follow another method to get the is Zachary karate club in the graph object which I think we used in the previous video

you if you have the karate dot gml or karate club graph in any format you can just read that graph file and then that is how you can you know convert that into graph object.

(Refer Slide Time: 15:43)

```
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_3_(Strength of weak tie
s)/Videos$ python girvan.py
The number of connected components are 1
The number of connected components are 2
[32, 33, 2, 8, 9, 14, 15, 18, 20, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31]
.....
[0, 1, 3, 4, 5, 6, 7, 10, 11, 12, 13, 16, 17, 19, 21]
.....
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_3_(Strength of weak tie
s)/Videos$
```

So, you can use that, or you can use this inbuilt function which is given by networkx. So, we are going to run this again on this graph let us see, let me its let me show you. So, you see the number of connective components in initially was one and then it removed some good number of edges based on the betweenness until it reached to the two connective components part, and this is these are the nodes in the first community and these are the nodes in the second community, and yeah I think these are the these are the nodes which actually constitute the two communities in the karate club network, you can also verify you can google and you can see the communities that would actually formed in the karate club network.

So, this given (Refer Time: 16:26) is doing good job of giving the communities here, you can also see the number of nodes if you want to print the number of nodes in each community let us do that. So, as I told you this is i is a graph. So, you can you know use any graph function here number of nodes it should work.

(Refer Slide Time: 16:51)

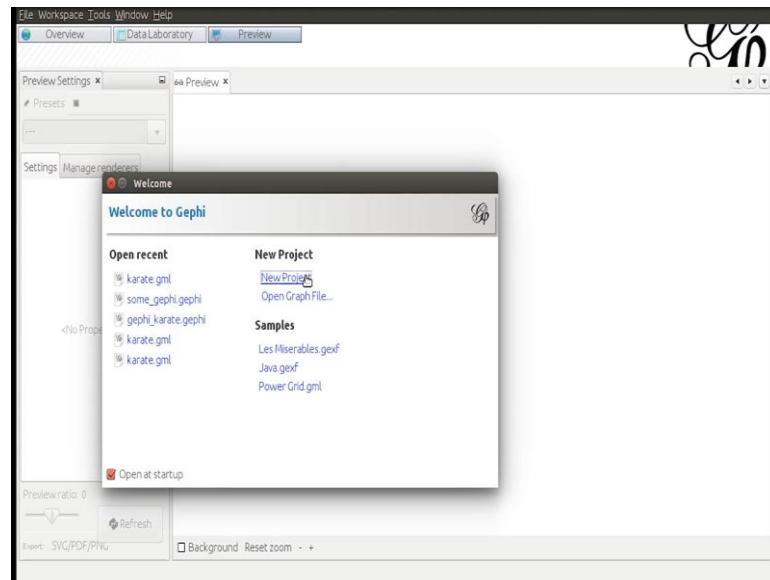
```
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_3_(Strength of weak tie
s)/Videos$ python girvan.py
The number of connected components are 1
The number of connected components are 2
[32, 33, 2, 8, 9, 14, 15, 18, 20, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31]
..... 19
[0, 1, 3, 4, 5, 6, 7, 10, 11, 12, 13, 16, 17, 19, 21]
..... 15
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_3_(Strength of weak tie
s)/Videos$
```

So, I want to see the number of nodes in each community. So, there are 19 and 15, I think if I am not wrong there were 16 or 18 or something, but nevertheless this is pretty good approximation of finding the communities in the given graph.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

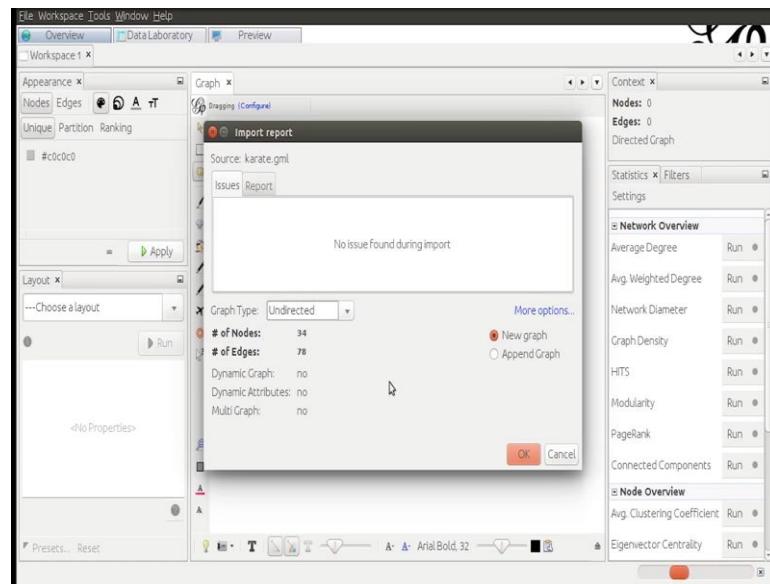
**Lecture – 37**  
**Strong and Weak Relationships**  
**Visualizing Communities using Gephi**

(Refer Slide Time: 00:06)



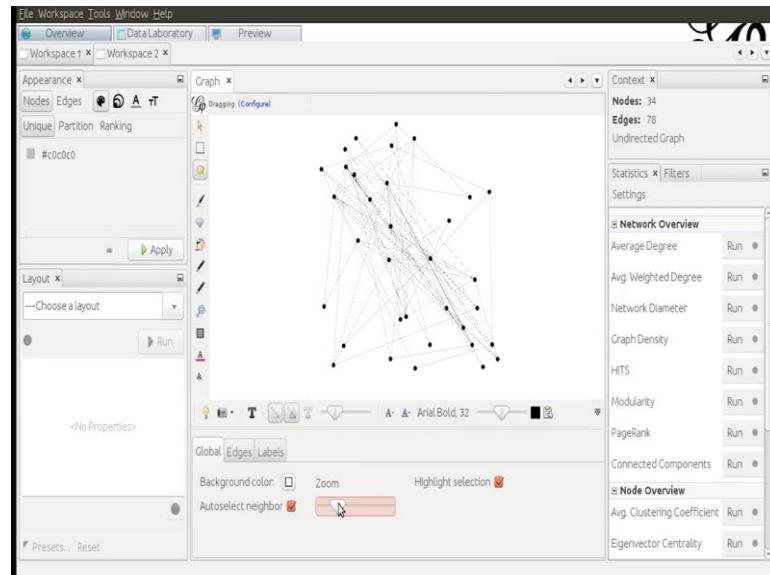
Hey everyone, in this video we are going to see how we can visualize the communities in the Zachary karate network. So, I have this Gephi opened already I am going to open the new project.

(Refer Slide Time: 00:23)



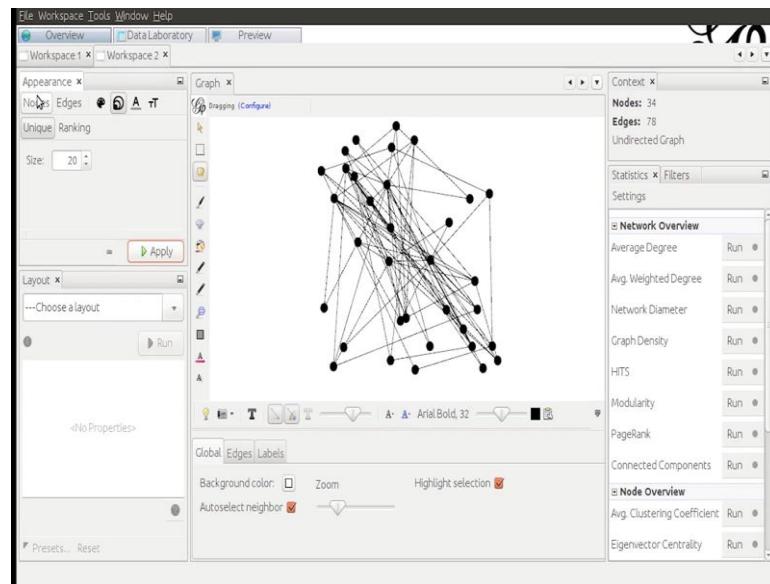
And in the overview, I am going to open the; I am sorry; I am going to open the file which is containing the karate network. So, I already have this karate.gml file, I am going to open that I know that this is undirected. So, I am going to select that this has 34 nodes and 78 edges.

(Refer Slide Time: 00:50)



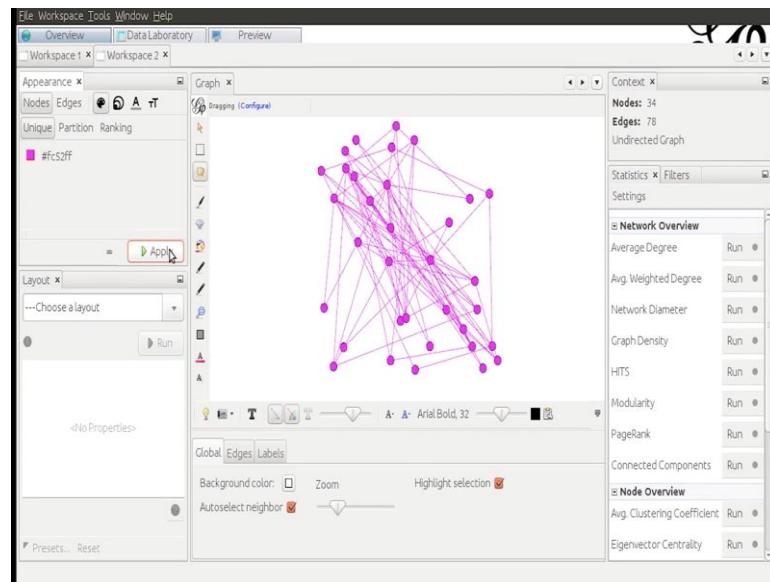
This is the graph that we are getting I think we did in the previous it as well let me zoom in, so this is the karate network let me thick in the edges.

(Refer Slide Time: 01:01)



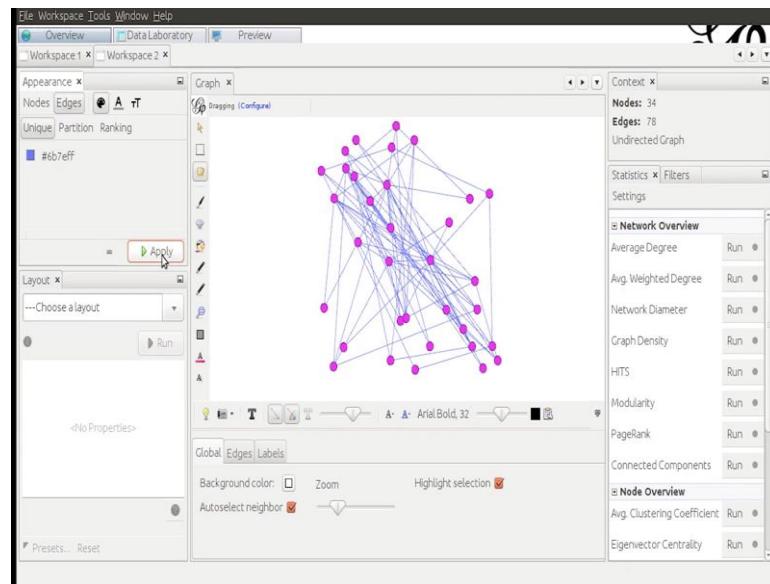
Let me increase the node size maybe. So, I will go to nodes and I will change the size of the nodes let me keep it 20; does not look so nice because they are black in color.

(Refer Slide Time: 01:25)



So, let me change the color of the nodes let me have something like this may be. So, it has changed the color the edges as well.

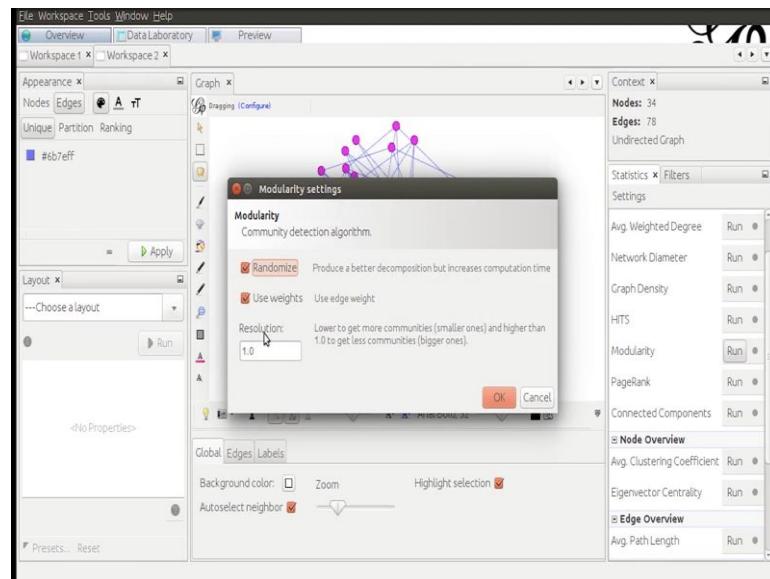
(Refer Slide Time: 01:36)



Let me change the color of the edges maybe with this one let see not so good, I will little darker decent.

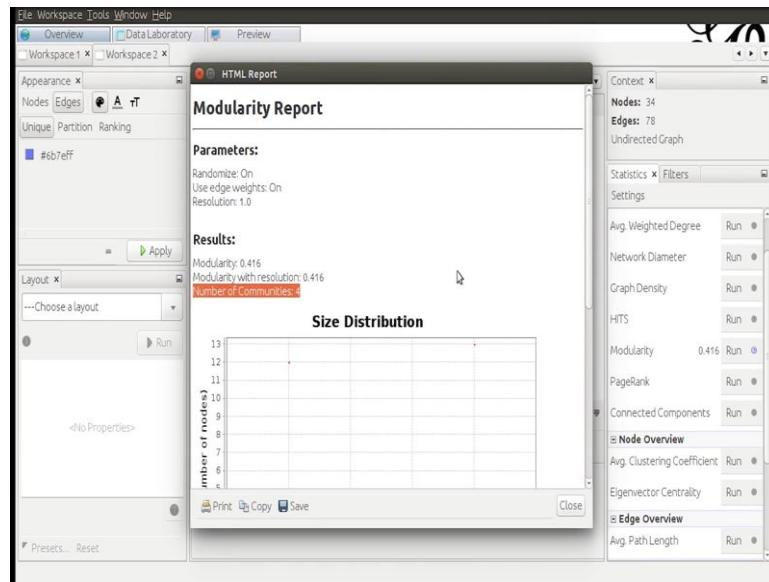
So, I have to show you how to visualize the communities in this network. So, I am going to go to this right-hand side panel here we have this modularity feature I am going to run this.

(Refer Slide Time: 02:01)



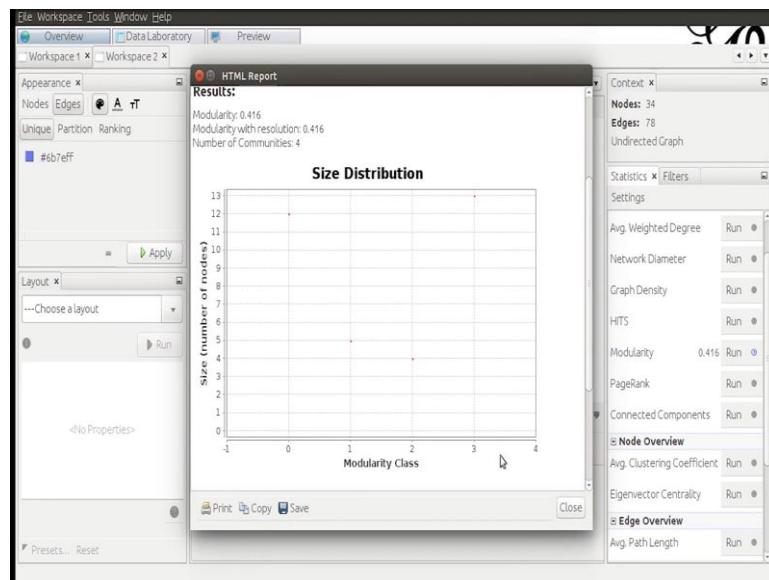
And here we have this resolution value if the value is less than we get more communities is resolution values higher we get less communities- let see how many communities we get with this value of resolution.

(Refer Slide Time: 02:18)



So, this is what we get we get 4 communities here.

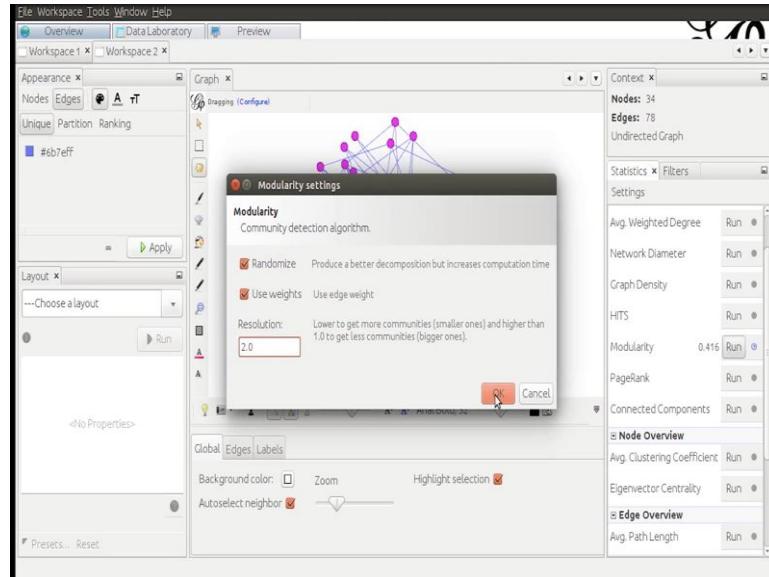
(Refer Slide Time: 02:24)



And these are community 0, 1, 2 and 3, these are you see the small red dots, these are the number of nodes, in these communities, I am going to close it, it is well known that

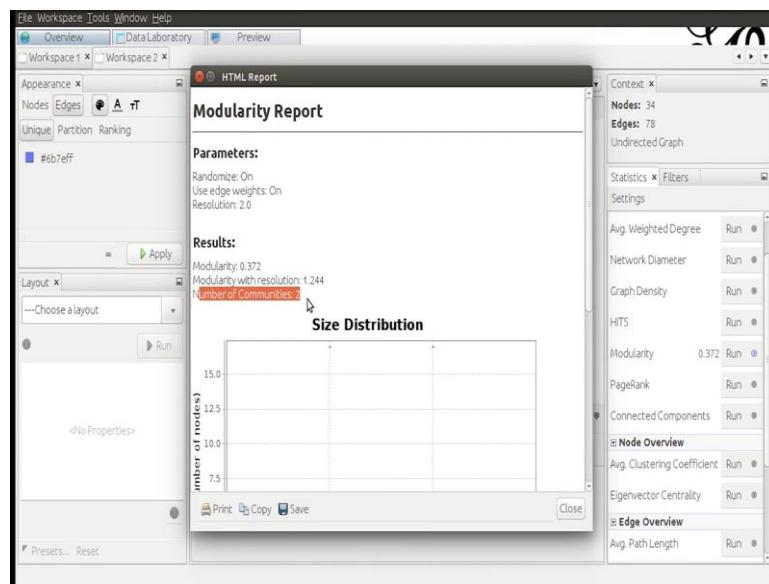
Zachary karate network had 2 communities. So, I am going to change the value of this resolution so that we get to 2 communities.

(Refer Slide Time: 02:50)



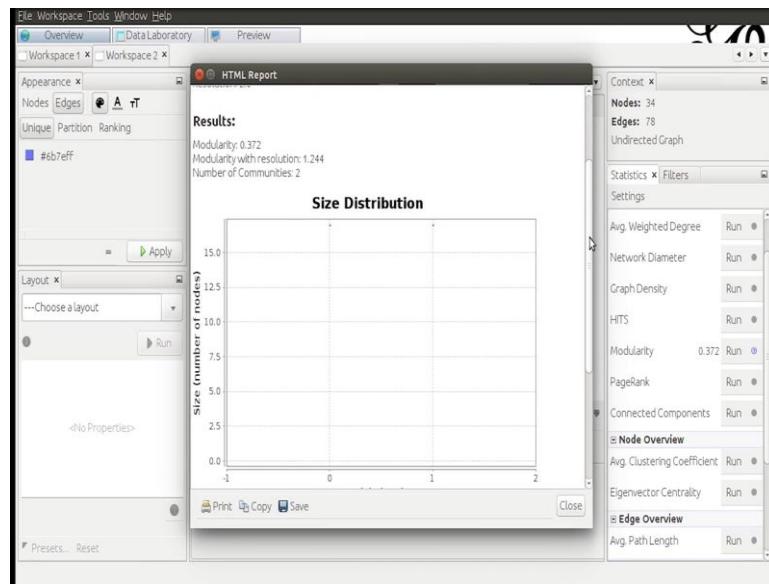
Let us change this where it 2 and let see how many communities to be get.

(Refer Slide Time: 02:52)



We are getting 2 communities here the 2 most prominent communities that existed the time of fight that happened there by knowing the history behind this network.

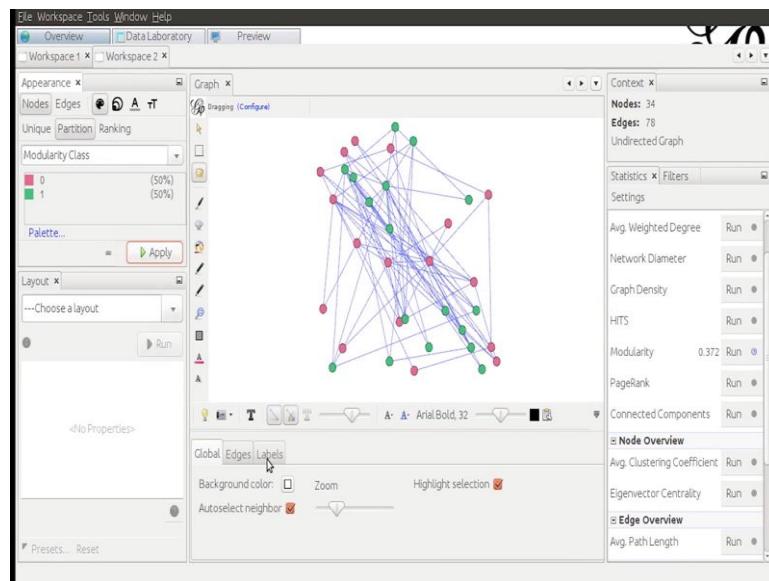
(Refer Slide Time: 02:59)



Let us close this and let us try to visualize the communities I am going to nodes and we basically want to partition the nodes into 2 communities. So, communities I have already been found we just want to display them accordingly. So, we want to change the colors color of the nodes based on the partitioning that has happened because of communities.

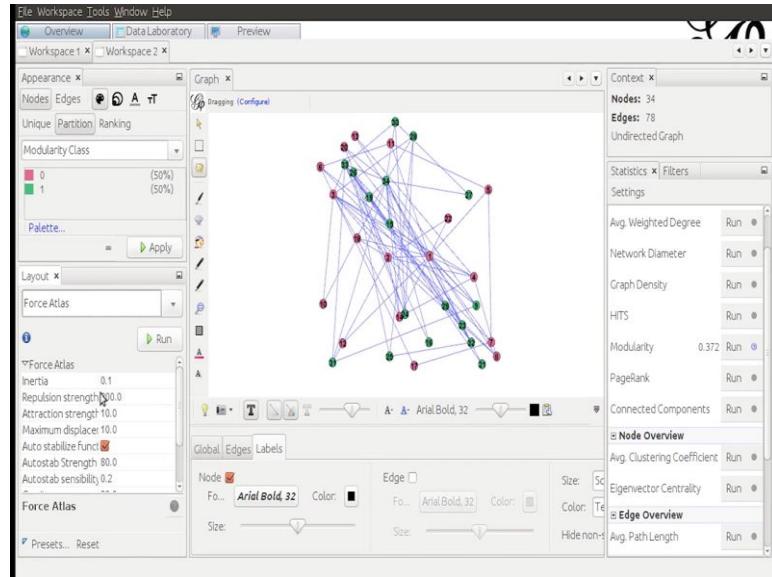
So, nodes just selected colors are selected partition we will click and here we have this modularity class.

(Refer Slide Time: 03:43)



You see there are 2 communities. So, we are getting 2 different colors here when I apply. So, you see the nodes have changed color based on the communities that they belong to.

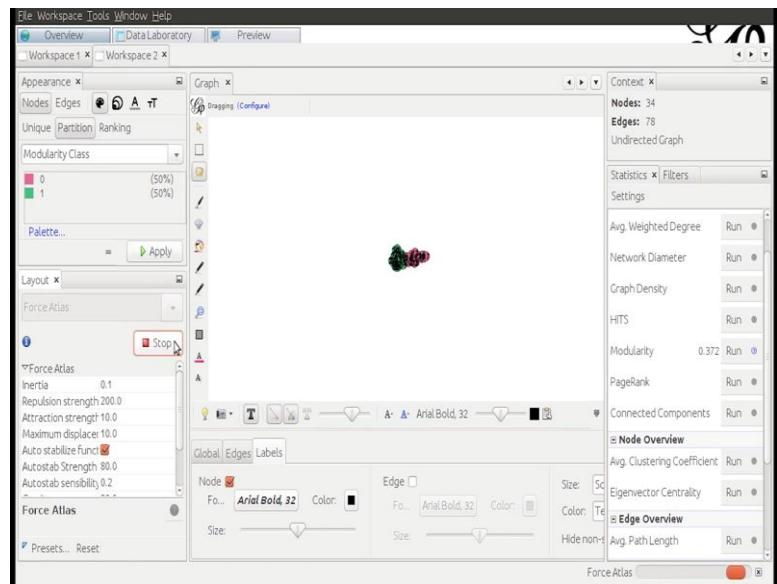
(Refer Slide Time: 04:07)



So, this is nice; we can see the nodes which belong to different communities and in case you want to display the labels so that you get to know which node belongs to which community you can do that. So, I am going to click this. So, you see the labels of the nodes are also visible.

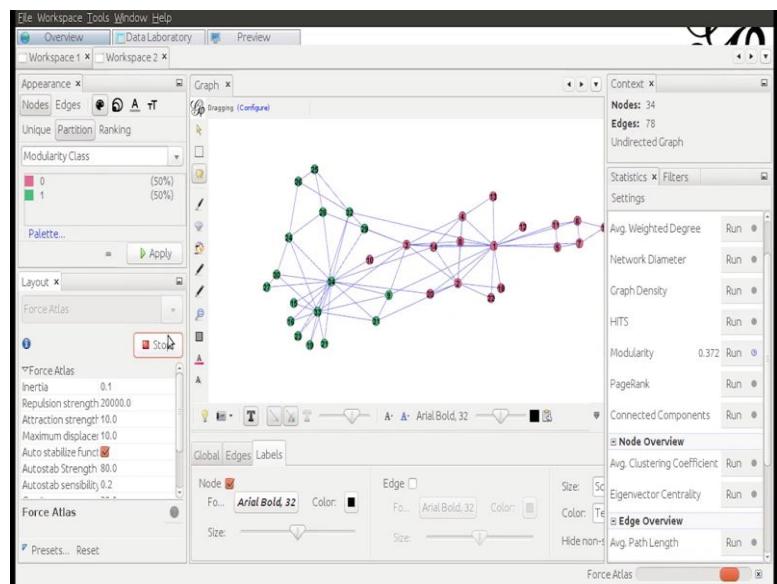
Let me show you some layouts that might be useful maybe in this situation let me show you this force atlas layout. So, what this layout does is it forces the nodes in the given community to be displayed together.

(Refer Slide Time: 04:38)



So, let us run this, it is not looking so nice and going to stop it. So, I am going to increase this value of repulsion strength. So, as of now the nodes I have gotten clustered together based on the communities you see all the green nodes are on one side and all the pink nodes are on the other side, but it is difficult to see them because the repulsion is less. So, let me increase the value of repulsion may be to this value you can just do it and try.

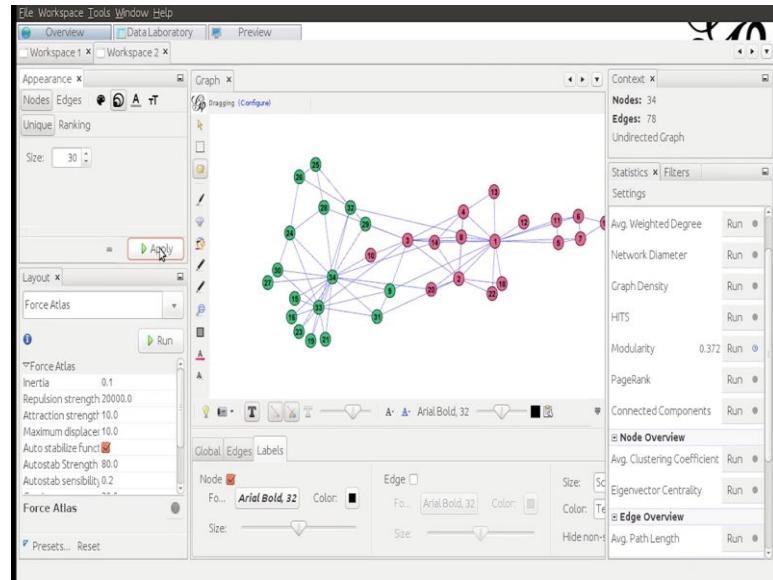
(Refer Slide Time: 05:03)



So, this is a little better. So, I am going to stop it. So, green nodes are in the left hand side and pink nodes are in the right hand side. So, this is a little better visualization let

me increase the size of nodes. So, that the labels are visible better. So, I am going to nodes unique and color partitioning we have done size basically we have to change size. So, let me make it 30.

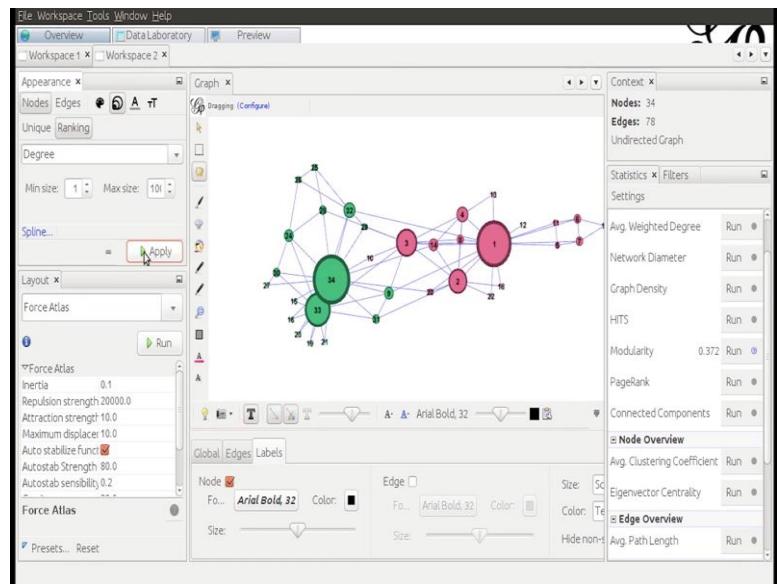
(Refer Slide Time: 05:41)



Little better; so this is one thing another thing that we can do here is that we can change the size of the nodes based on degree. So, I mean nodes I will change the size basically I am going to run the nodes based on decrease. So, I am going to click on ranking and the attribute that I will choose will be degree.

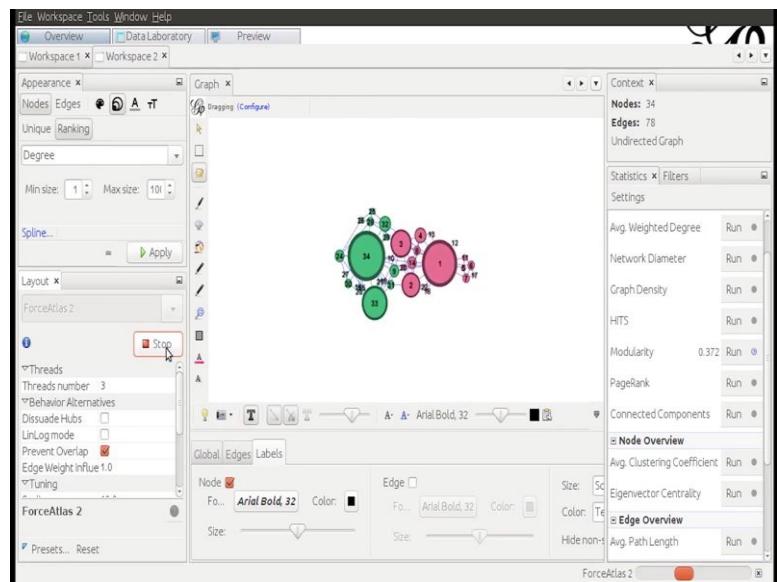
So, I think this is good enough hundred go should be the maximum size.

(Refer Slide Time: 06:06)



So this is what I see. So, you can quickly see which are the nodes that are having the high degree and you can; obviously, see the community is that they belong to let me also you should one more layout here which is force atlas 2 there is one particular nice feature that I will like about it and let me show you that let me click prevent overlap because I see some of the nodes overlapping. So, I do not want that to happen I think this we can go ahead.

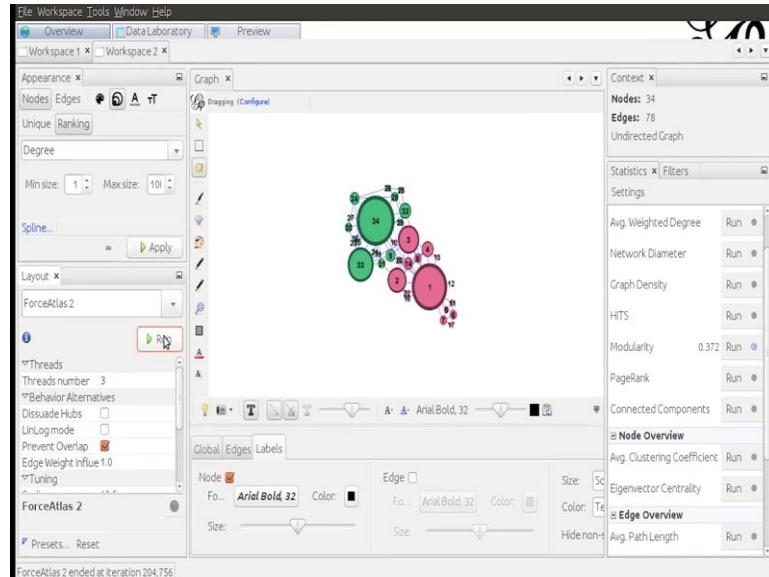
(Refer Slide Time: 06:37)



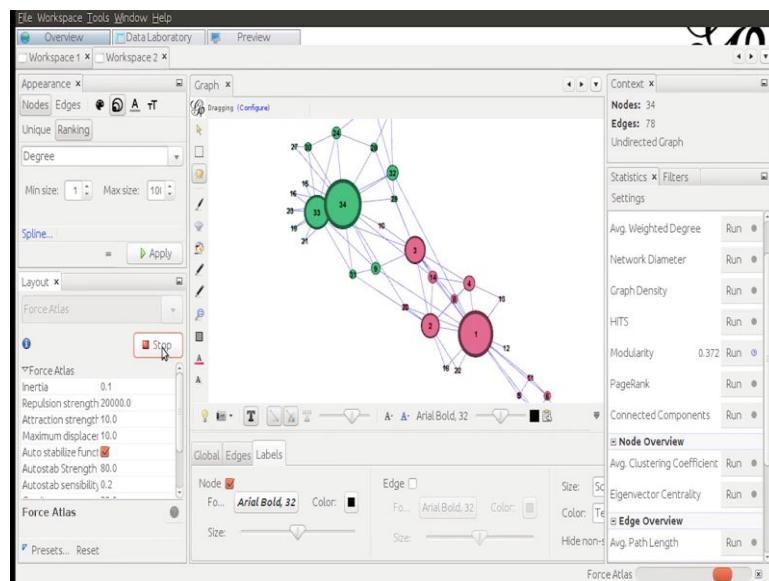
So, you see what it is doing it is sort of moving the graph.

So, whenever you like it you get this stop maybe I like the green nodes to be in this selection I will stop it.

(Refer Slide Time: 06:47)

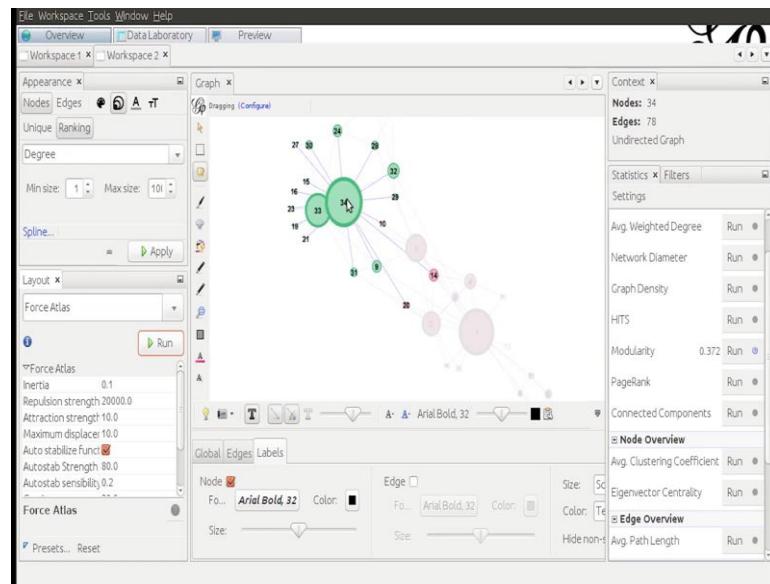


(Refer Slide Time: 06:57)



So, that is one thing and since they; the repulsion has reduced. So, I can go back to this force atlas and then run it and then again, I get the nodes you know for a part from each other so that I can nicely see them.

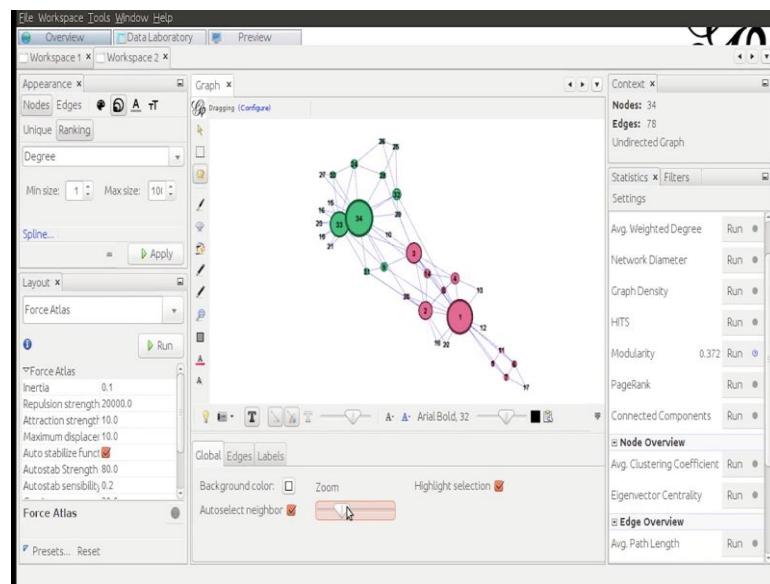
(Refer Slide Time: 07:06)



So when you see, they kind of high lighting that is happening when you when you how are your mouse on one node all the nodes in that community are visible and so on.

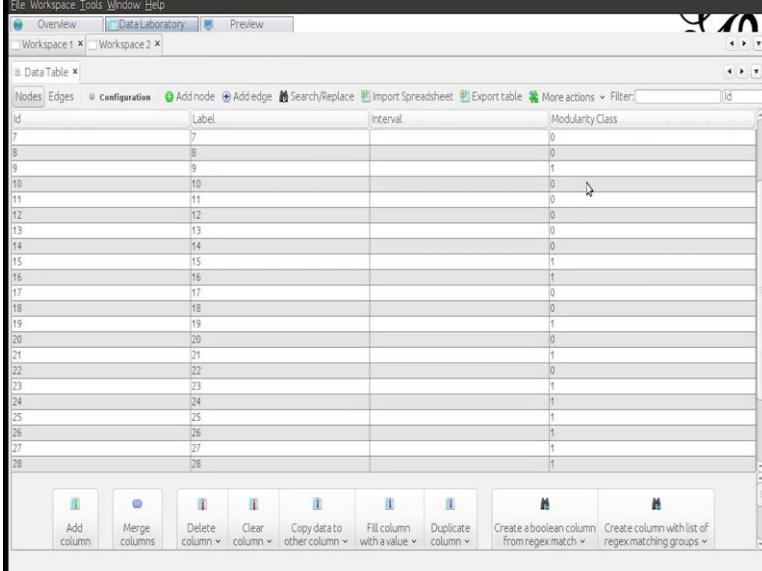
Let me reduce this size. So, that I can this is the labeling let me go back here let me.

(Refer Slide Time: 07:27)



So, this is the network that you can see.

(Refer Slide Time: 07:36)



Id	Label	Interval	Modularity Class
7	7		0
8	8		0
9	9		1
10	10		0
11	11		0
12	12		0
13	13		0
14	14		0
15	15		1
16	16		1
17	17		0
18	18		0
19	19		1
20	20		0
21	21		1
22	22		0
23	23		1
24	24		1
25	25		1
26	26		1
27	27		1
28	28		1

And you can also go to data laboratory. So, here you see modularity class for every node is given you can just export this table. So, that you can use it to the way you want in any other tool as well. So, there are 2 modularity classes; class is 0 and 1 because there are 2 communities here and you can also go to preview and let me refresh; refresh you are that was about how we can visualize communities in Gephi.

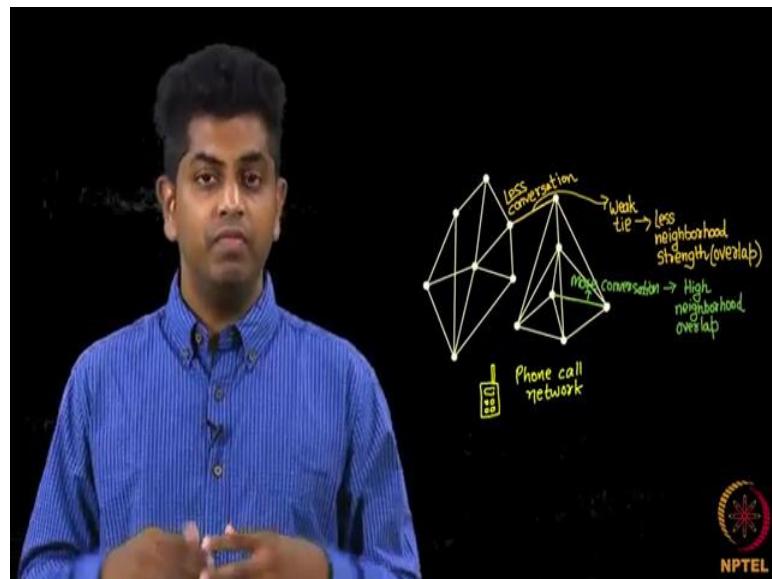
So, that was a brief introduction to a few features of Gephi with respect to visualizing the communities in the graph.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture - 38**  
**Strong and Weak Relationships (Continued) and Homophily**  
**Tie Strength, Social Media and Passive Engagement**

We observed that Granovetter indeed had guessed it right. The data set that is available today tells us the following.

(Refer Slide Time: 00:26)



Given an edge, given a link you observe that in the phone call network they take the phone call network and they see that are given link between two people you have an edge if they have had a phone call. If they have had less conversation across a period this experiment said that it is mostly a weak tie and it is topologically also a weak tie which means its neighborhood strength is less. What do I mean by this? It just means in a network if you look at all the links, links denote two people are have had conversation on phone and you see how much conversation they have had and you observe that more the conversation more is the neighborhood overlap, lesser the conversation less is the neighborhood overlap. We saw this in detail, we made are inferences, we saw the plot as well.

Now the world has been there done that they were phone conservations, how do conversations happened today, it is mainly through Facebook, Whatsapp, different social media, right.

(Refer Slide Time: 01:40)

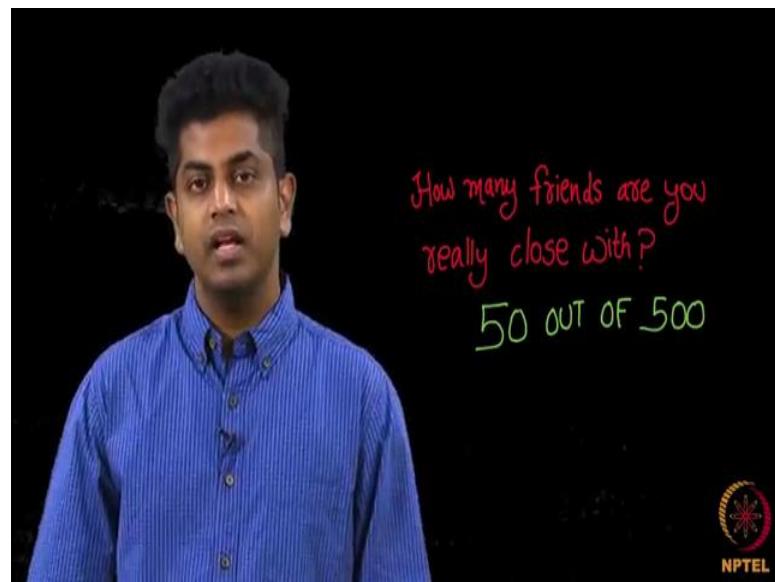


So, let us now shift gears and ask a nice question. So, we have seen Granovetter strength of weak ties and we saw its validation on the cell phone, phone usage network. Now we will slightly take move, will drift slightly away from Granovetter strength of weak ties theory and ask an entirely different seemingly different, but related question. The question is the following.

We all use Facebook, imagine when was the last time you met your old friend and this person comes and says hey dude how are you, I saw your updates on Facebook, very interesting you seem to have, you seem to doing this, you seem to doing, that I see that you got married recently, did you invite me, you had a kid, oh videos are very cute. Now you are wondering this friend of mine which I have lost touch for a fairly long time, since to be in touch with me he is following what is happening with my life, this is what we call passive engagement we may not be in touch with our friends, but we know what is happening with their lives.

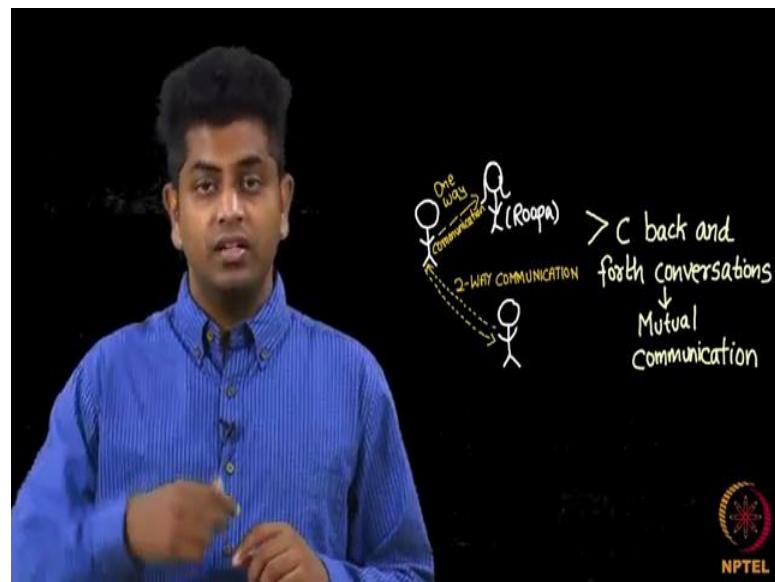
Now, we have thousands of friends on Facebook, if not thousands at least hundreds. Each one on an average, let us say each one of you have around 500 friends.

(Refer Slide Time: 03:18)



But if you ask this question with how many people are you really close? The answer probably will be very less if you have let say 500 friends you hardly will be close to let us say 50 friends or even less correct, let us do this experiment. Let me take you your Facebook profile and let me try to see with how many people you had a conversation.

(Refer Slide Time: 03:46)



So, let us say you say hi Roopa how are you it is been long since we spoke and Roopa does not reply back. I would call this one-way communication; you have spoken to Roopa and Roopa has not replied back. Two-way communication, whether mutual

communication is when you say something that person also say something you say something, she says something so on and so forth right.

There has been some amount of conversation. I keep a threshold and say if it has caused this many back and forth conversations then I call them as people who are in mutual communication. So, one-way communication is just a few messages, mutual communication means a lot of messages is a third type.

(Refer Slide Time: 04:35)

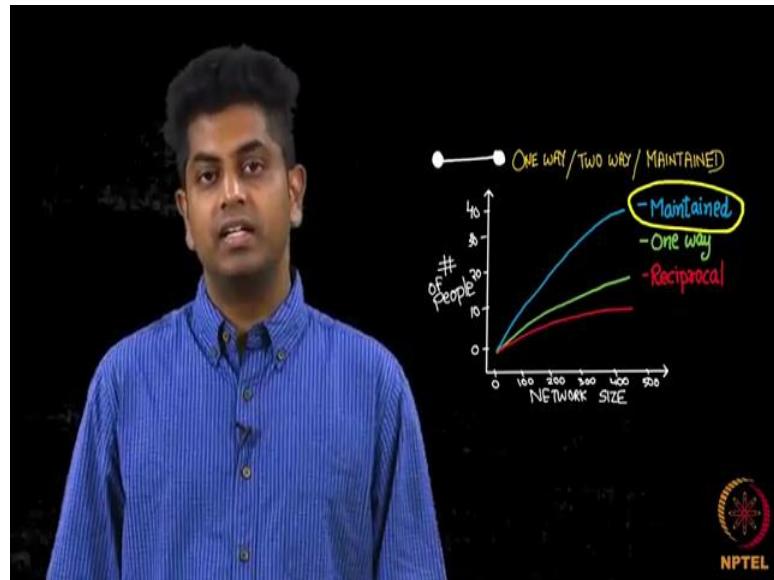


People observed there first-time one-way communication is a lot in number, we just add people we just say hi and then we forget on Facebook, we do not care. They also add us, but they were not respond back to our replies, I mean our messages, one-way communication. Mutual communication is you have a very close we keep talking, mutual communication is very less one-way communication is lot more. This is a third type as I was telling you.

The third type is if a person is following you, on your timeline on whatever is happening with you whatever news you keep posting he comes and clicks on like or comments there and things like that right I call it the third type. So, what is so interesting about this third type thing about it, is it, is the third type better than one-way communication or is it less than mutual communication, let us think about it for a minute. So, this third type is called the maintained relationships. Let me ask what happens on Facebook, given a friendship what how do you classify this, is it one way communication or two way communication

or is it a maintained relationship. You probably will say some of the there could be overlap between reciprocal communication, which is two way communication and maintained relationships, two people were talking to each other, might even follow each other.

(Refer Slide Time: 06:21)



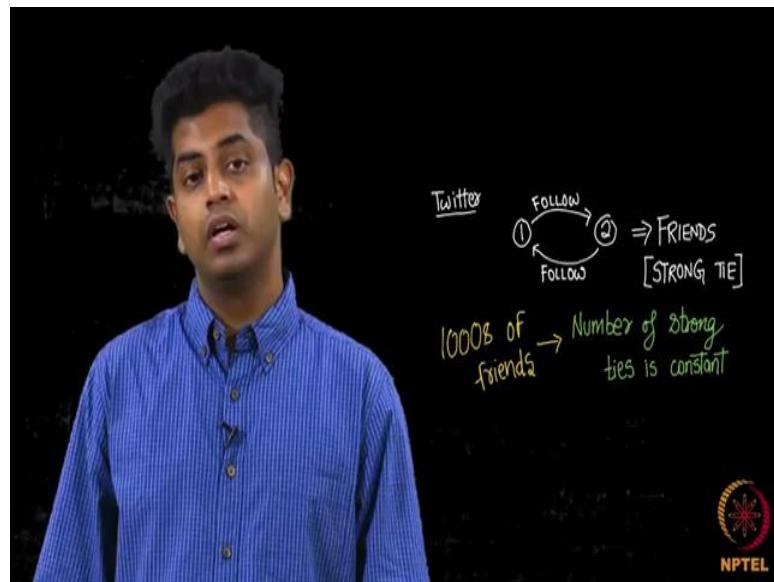
Now, what I will do is I will see which one is more and classify and link as either one way or two way or the third one maintains relationship. And what I will do is I will plot with x axis being the network size, actually the neighborhood size, but took keep it simple on your minds I will call it the network size and then on y axis I will plot three things - one is one way communications, second one is mutual communications reciprocal communications or two way communications whatever recall it that is a second plot, the third one is the maintained relationships. You see what is happening here lets at any point on x axis, let me say network of some 100 people when I see the total number of friends that a person has is basically this these many people are one way relationships, these many are two way relationships and these many are maintained relationships right.

What do you see from the plot? The maintained one also called the passive relationships is the highest, why is that, why does that happen, is it that obvious Facebook keeps you glued by showing you some interesting stories about my life, but we seldom talk there. More often on Facebook what happens is you checking my news feed, me checking your

recent photograph that you uploaded, me click clicking a like on it, me seeing your status message commenting on it etcetera right. There is not much of talking that happens would me and you and that is what happens on Facebook, there is a whole lot of mutual engagement, but very less of first and the second type. So, the plot clearly says most the friendships on Facebook is basically about two people just checking there Facebook profiles clicking on like, commenting and stuff like that.

So, what we mean by this? What is the inference? The inference is that on social media the most important thing that sells that keeps people engaged is the third type, if you let them just talk to each other probably that particular media will not be very popular. You should make them indirectly be updated about each other's life by not talking to each other, but by just seeing each others Facebook profile, that is what the plot says.

(Refer Slide Time: 09:18)

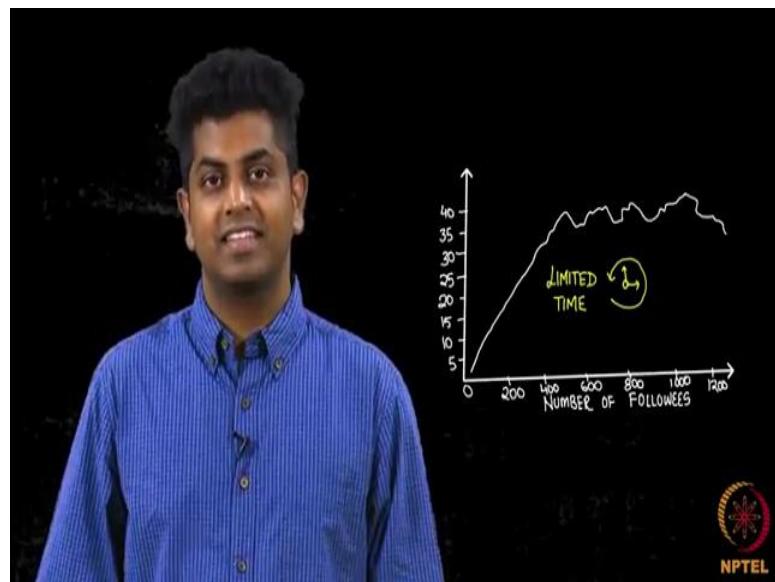


So, a similar question was also asked on twitter what the question was asked on twitter. Let me classify friends as strong friends and weak friends on twitter, you know in twitter you basically follow a person and that person may also follow you, only if a person follows can you send that person a direct message which will personally go to that persons twitter inbox let us say, they observed the following.

They saw if two people are friends and if they have exchanged personal messages then it is a strong tie otherwise it is a weak tie: so what. They saw something very expected. People may have 1000s of friends, if you see Obama or Donald Trump or Narendra

Modi you will see that they have millions and millions and millions of followers, but you will observe something very startling about all the celebrities. The amount of direct messages that they would have exchanged and hence the strong ties that I define. I repeat a link is a strong tie if they have exchanged direct messages with each other, more than let us say 3-4 direct messages, they observed that this seems to be a constant. No matter how many people you are following or being followed this does not seem to go beyond some 50 or 60.

(Refer Slide Time: 10:55)



Look at the plot the x axis is the number of friends and the y axis signifies the number of strong ties, you see that it sort of converges, converges to a value which is a lot less than 100 which means converge may the number of people to whom you can be closed cannot go beyond a particular number. Now is not this obvious, why is this obvious? This is obvious because we all have limited time, we cannot endlessly keep communicating with people, but we can follow some 1000-2000 people I know what is happening with their life. But we will not be able to talk to them regularly that is probably because the limited cognitive budget that we have, limited time that we have correct. So, we can only do this much and that is what the plot says.

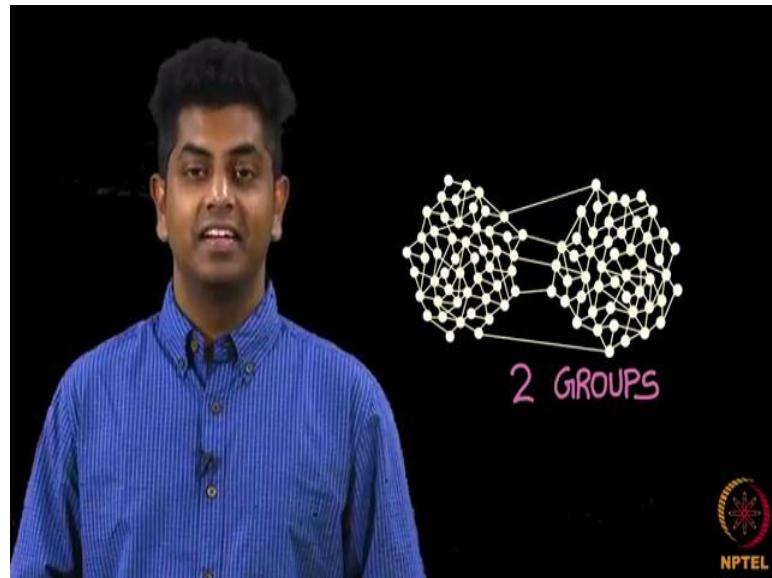
Very eye-opening plot in the sense that we think many people who have a lot of; few people have a lot of friends and probably they know a lot of people that is not very true

even. If they have a lot of followers or if they have following so many people those people that they are close to or let us say strong tie is way too less.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

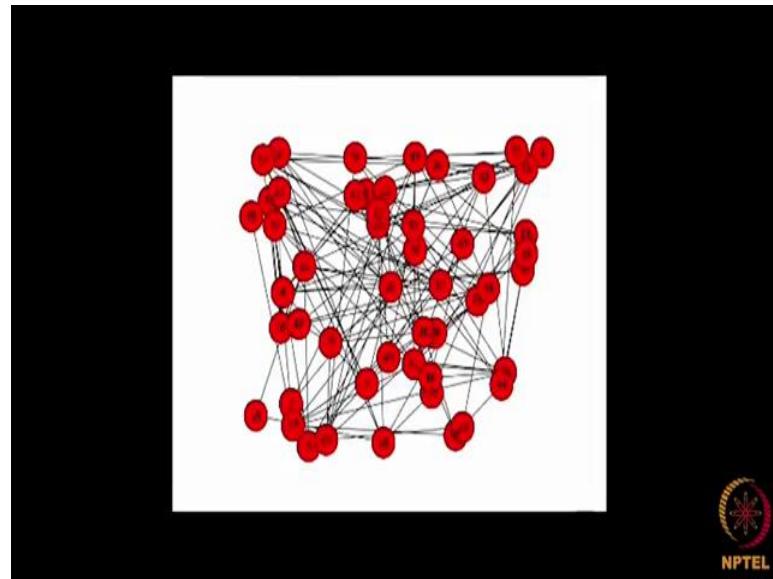
**Lecture - 39**  
**Strong and Weak Relationships (Continued) and Homophily**  
**Betweenness Measures and Graph Partitioning**

(Refer Slide Time: 00:08)



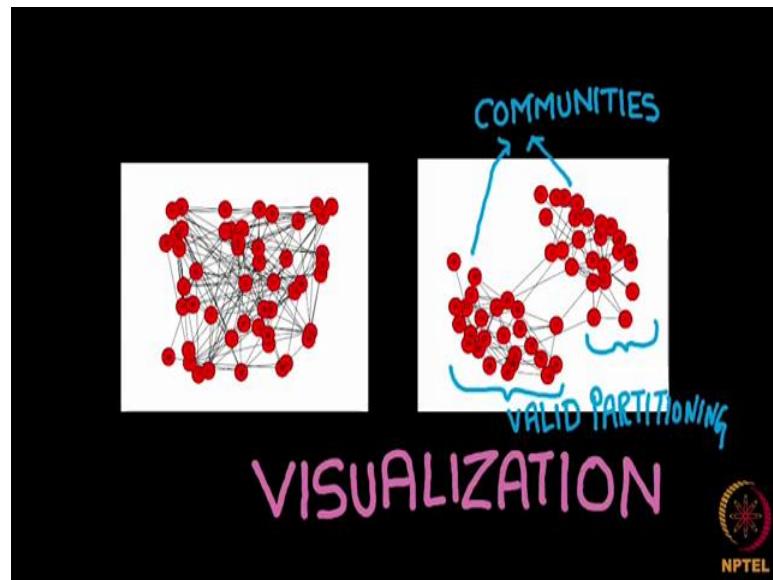
Let us start with the nice puzzle. Here is the classroom of 100 people and there are precisely two teams in this, two groups. Now what do I mean by two groups? Guess what I am saying, two groups means what? We say there is groupism in this place, in this organization means there are two teams, two groups and friendships are within and not across. By not across I mean, I do not mean there is no nothing across, I mean it is less across and more within then we say oh the classroom right now has two segregations, is segregated into two, right, and there is groupism here.

(Refer Slide Time: 00:57)



Now, look at this graph what can you say about the graph? Does this graph have this is the friendship network of some 50 students, do you see any kind of what to say grouping here, are there two groups here, I think no there is a look like there are two groups.

(Refer Slide Time: 01:23)



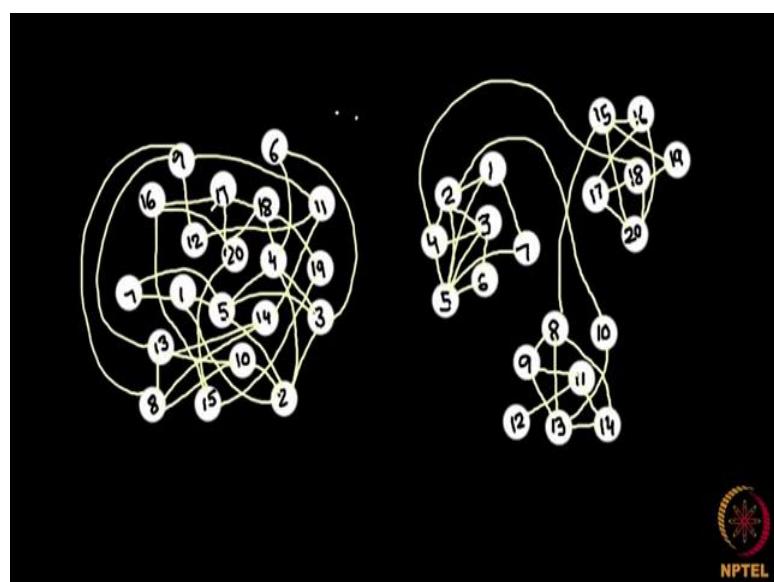
But probably you are wrong there are two groups here observe, there are two groups there. The way it was shown before to you probably did not recognize that there were two groups, but actually there were two groups it all depends on how you visualize the graph, I tried visualizing a different way and I see that there are actually two groups. The

visualization before was very important that said there is only one group, it is all in the visualization. In fact, visualization is a separate branch of research in computer science anyway I will we will not get there, but we will answer this question given a network like this how many groups are there.

These groups are called communities, in the languages or subject it is called communities. So, how do you define a community? A community is a bunch of nodes where the connections are a lot within them and a lot less outside. This is technically called, a bunch of nodes is called communities these nodes partitioning of a vertex at basically, you have 50 nodes that is the vertex said and you classify them as some 20 and 30 or something like that and you call this partitioning as a valid community partitioning if such a partitioning is such that there is a lot of intercommunity edge spar city and intra community edge density intra means what within inter means between. So, between it is less, within it is more then you call such a partitioning, or the vertex set as community partitioning.

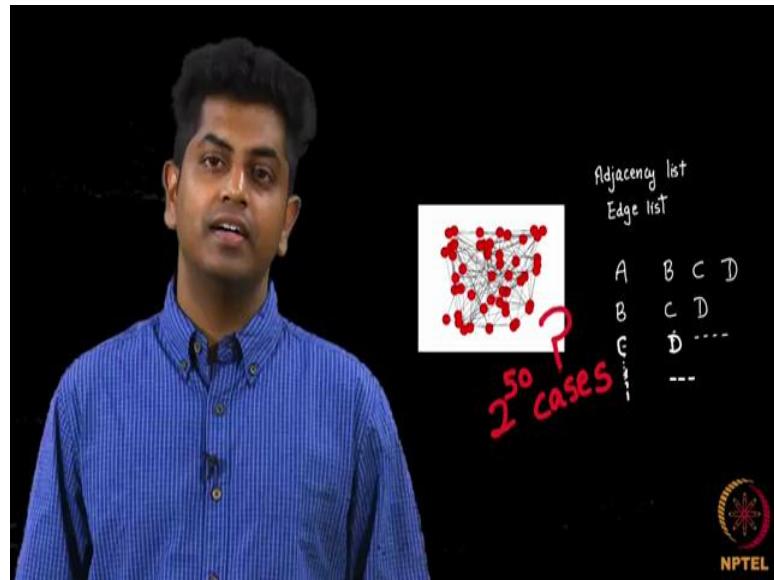
Now let us get back the network that I showed you 50 people you we all initially thought this only one group here, but we realized is two groups here. Given a graph  $g$  which is input to a computer how do you find how many groups are there, maybe there are two or may be more why more? I will give you an example.

(Refer Slide Time: 03:36)



Look at this network how many groups do you have? 2, 1, observe if you visualize it properly you see that there are 3 groups, how do we deduct its groups, such groups. I will give you a quick answer to it intuitive answer and a technical answer to it.

(Refer Slide Time: 04:02)



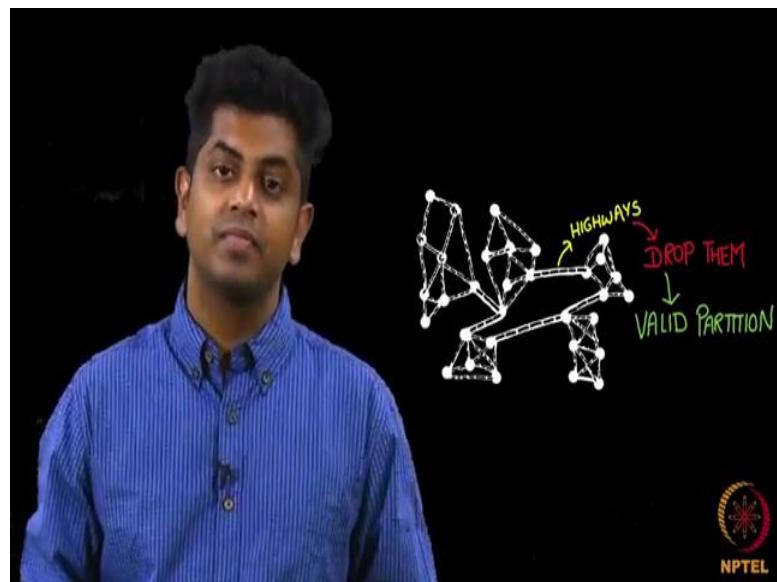
50 people there is some partitioning here and I need to know what is the partition. Is it even easy to answer this the graph is given to me in the form of let us say adjacency list or edge list, what do you mean by this I just know all the links in the graph it is given to me in a file let us say.

We have seen these things in our programming assignments right. This is how I know the graph let us say this way, now A is adjacent to B, C is adjacent to D so on and so forth. There is some complicated graph like this, and I see that there are 50 people, and these are the links and the links are known to me. Only with this information can I get to see get to know if there are teams here, this is hard, very hard to solve, it is a lot of computation whole lot of computation. In fact, precisely speaking you may have to, you observe that there are actually around  $2^{50}$  possible partitioning of this 50 people, how is slightly complicated do not worry about it trust me it is  $2^{50}$  number of partitions.

For each partition you must check is this partition leading to community, is there inter community sparsity and intra community density, is technical languages all I am saying is there more adjust within and let us say just across then you say this is the community, I have found the community there are two communities. There are  $2^{50}$  cases that you must

check that is the huge number, you probably cannot do it. Is there a easy way out? Yeah, there is an easy way out and I will give you the intuition the intuition is very clear. Assume I gave you all the road network of India, I will give you all the roads connecting two points you must and I will remove all the names, you do not know which node represents what just given the nodes. Nodes are all regions let us say town, cities, villages whatever and edges are the roads connecting these two places, direct roads connecting these two places.

(Refer Slide Time: 06:02)



So, where is a road that connects two points through region then you do not put a link between these two nodes, you put a link through this node. Let us say city A and city B do not have a direct road, but you should pass through C then you put a link between A to C and then C to B clear. How do you find out what are the main, what are states here let us say of the country? Yeah, states are basically probably communities here do you see, there are very few roads between the states and a lot of roads within this state. There are very few roads across cities and a lot of roads inside a city, grab on this sentence. There are a lot of roads inside the city, but a lot less roads across the city why? You see highways connecting two cities there will not be many roads between any two locations of the city, the only way you can get to this location of the city to some other location of some other city is through this highway correct.

This highways act as bridges in connecting two cities, if you can identify these highways and remove them then you get a partition that represents communities. I repeat in the road network if you identify the highways drop these highways remove them basically from the graph, what is remaining? Remember you are removing edges not nodes in the road network you have nodes and edges nodes represent cities towns and edges represent roads between them what you do is you look at the highways and drop these highways and the resulting network should be, the resulting nodes that you see should be partitioning of the graph. What you mean by resulting nodes? It results in a disconnected bunch of; you will see a lot of edges within when you remove the highways. They are probably the states or cities. This is basically the intuition behind what I am going to say next. Our puzzle that we are we were trying to solve is 50 student classroom how do you detect whether there are groups here or not.

(Refer Slide Time: 09:02)



You just go on the hunt for highways here. What do you mean by highways here? Of course, there are highways here even in friendship networks, how do you see these are the edges that connect between these two clusters, they act as highways right. How do you detect them when the graph is not really viewer friendly, it does not tell you the communities when you see it unless you pluck it out like this, place it like this and then you see there – yes, there are two teams. These edges if you can detect and remove them then you see a disconnected two components - component 1, component 2 and then you can say ha yes there are two teams.

So, it all boils down to you detecting these links which act as highways. How do you detect them? So, we need to quantify this, how stronger a highway is this road correct, there is this notion of betweenness that does this. Now let us go slowly and let me define what one means by betweenness. Look at this edge in this graph take a node from this side and a node from that side, consider a path connecting these two nodes you see it always passes through this red edge, red link correct, it always passes through this red link. Take some other node these sides, some other node that side you have to invariably pass through this edge if you want to take a connecting path between these two nodes. But when you pick two nodes well within the left side then you can have a path which connects these two links directly like this correct.

Now betweenness of an edge is defined as the total number of paths that go on it, this is the very dumb definition. A very regress definition is slightly mathematically involved. All it says is the total number of paths that pass through this edge the total number of shortest paths that pass through these edges versus the total number of shortest paths between two vertices. What do I mean by this?

(Refer Slide Time: 11:35)

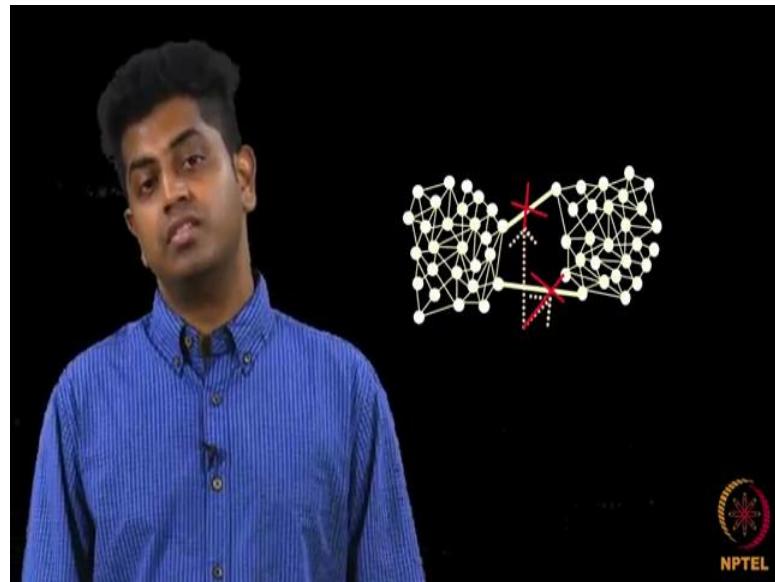


Betweenness of an edge is defined as  $\sigma_{st}(e) / \sigma_{st}$ , this fraction means what? You consider basically a path from s to t, you take a path from s to t and you see if edge e is coming on it; what are those shortest path from s to t where edge e comes on it; and what are all the shortest paths from s to t.

This fraction you see I take the sum of all such fractions across all possible s and t as a little mathematically involved all that it says is I am going to take the fraction of edges, a fraction of paths that uses this edge this is complicated version of this very straightforward statement and this is called the betweenness value of v. So, betweenness of value v is computed by using this rational of edge is has high betweenness if it comes in between as a link in the paths connecting two nodes. So, formally speaking betweenness a value of an edge is  $\Sigma_{st} \sigma_{st}(v) / \sigma_{st}$  running through all the vertices by this simply speaking all that we say is straightforward intuition that what are the fraction of shortest paths that cross e.

This quantifies betweenness, now how can we use this. Do not you think intuitively betweenness means this, higher the betweenness more does it connect two components that otherwise are not connected correct.

(Refer Slide Time: 13:19)



Look at this example this edge there are two edges that connect these two components and the betweenness of these two edges will be high as compared to the betweenness of some edges within the clusters that is easy to see. So, what I do is remember the question that we started with, given a classroom of 50 people we check are there two groups here, how do we do that? You take this graph and compute the betweenness of all the edges here other is seemingly not so complicated, it is very well known that this runs in a time

that is doable on a computer it does not take two to the power of 50 computations, it can be done much faster.

I do this and I find out the betweenness values of all the edges and then look at those edges that I have betweenness I observed that these two edges have high betweenness and what I do is I remove these two vectors and I observed that the graph becomes disconnected. When the graph becomes disconnected it has two components and these two components are the two groups in the classroom. So, we can find out what are the communities by looking at high betweenness edges.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture - 40**  
**Strong and Weak Relationships (Continued) and Homophily**  
**Summary - Strong and Weak Relationships**

(Refer Slide Time: 00:10)



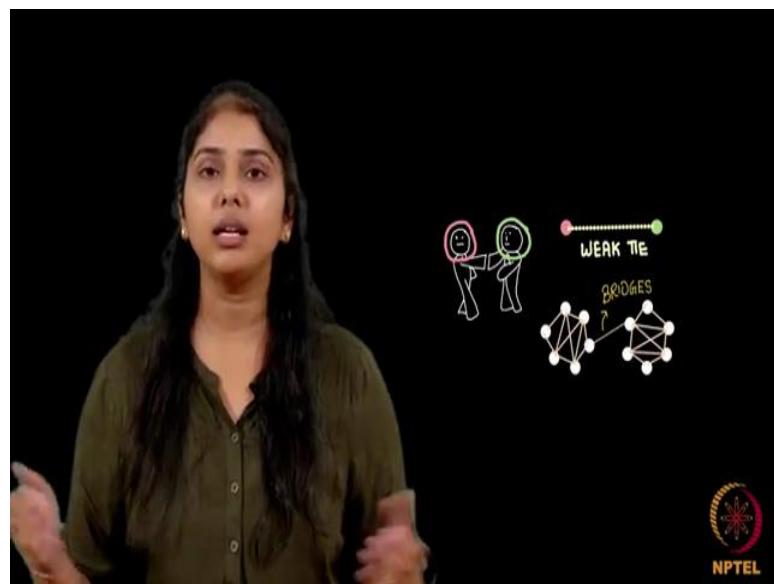
Congratulations, we have reached the end of week 3. Let us quickly recap what all we have studied. So, the first important lesson that we have learnt was we should be nice, we should be caring towards our close friends, towards our strong friends, but probably we should forget them when it comes about finding a nice job opportunity.

(Refer Slide Time: 00:20)



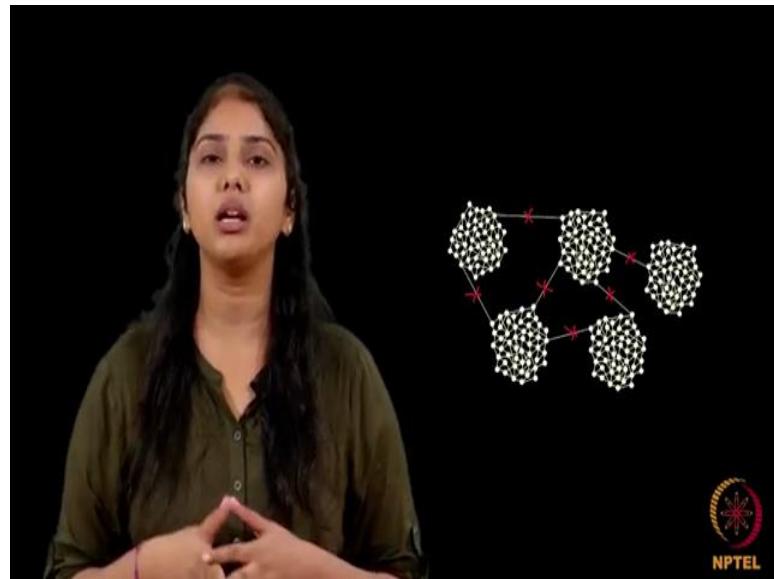
When it comes to finding a job opportunity it is better to contact a friend who is just an acquaintance, with whom we talk really, with whom we talk lesser less frequently and probably that person is my future employer or maybe a link to my future employer.

(Refer Slide Time: 00:53)



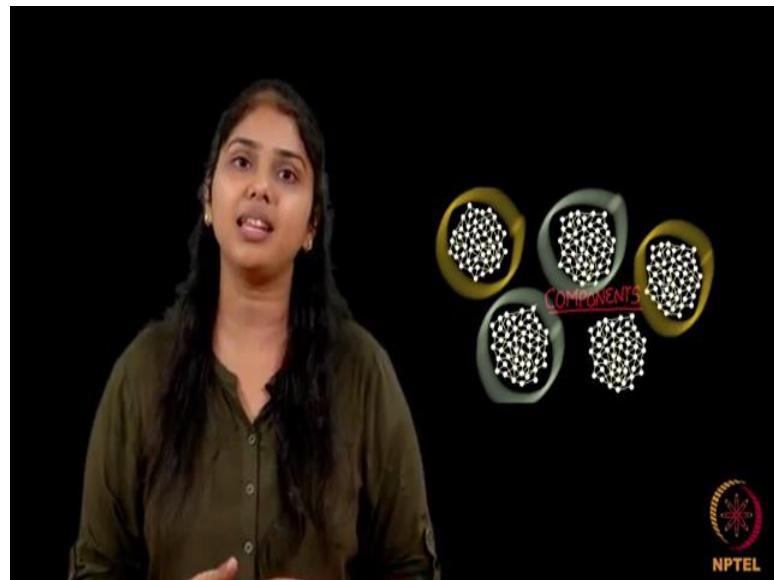
So, these people these kind of ties between me and a person with whom I talk less often we term it as a weak tie and we have looked at the importance of these weak ties in the networks. We have seen that in a network these kinds of weak ties they act as a connector between two different worlds.

(Refer Slide Time: 01:19)



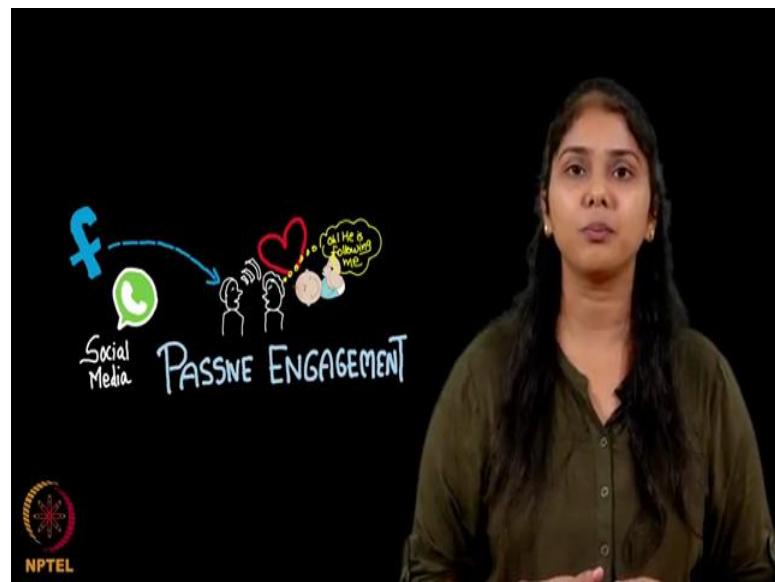
So, in our entire world these kinds of weak ties they connect two communities what will happen if we pull these weak ties and throw them away, we see what will happen to the world then. The world will get divided into individual components where there are no ties between these two.

(Refer Slide Time: 01:26)



What happens in this world, in this component nobody in this component comes to know about it. What happens here nobody here comes to know about it and all these different components evolve independently and probably that is not nice.

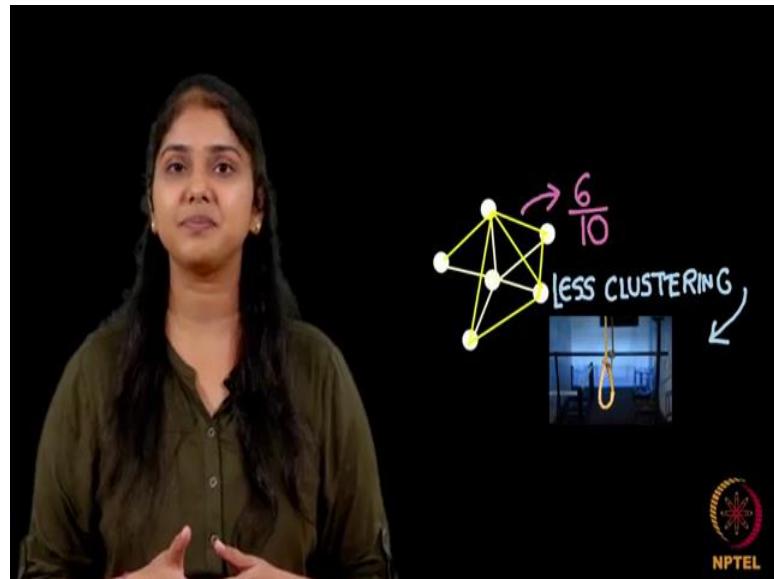
(Refer Slide Time: 01:56)



Next cute concept which we looked at was what kind of ties are there how do the ties they behave in our social networks, online social networks - like Facebook, Twitter, LinkedIn and all such places and we have seen that what happens on Facebook is we have so many friends, so many friends and we do not talk to them on a regular basis, we are not we are not very much close to them, but we always open a news feed and we are always interested in looking what is happening in their lives and this kind of a tie is termed as passive engagement. So, we do not talk to them regularly, but occasionally we look at what is happening in their life in a news feed and probably then we follow it up.

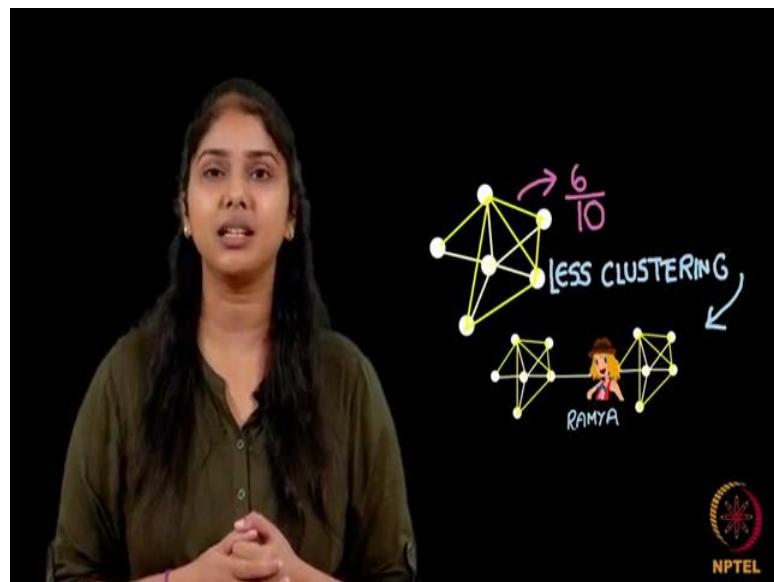
The last incredibly beautiful concept and personally my best is that of clustering coefficient. I hope you remember what clustering coefficient is.

(Refer Slide Time: 02:49)



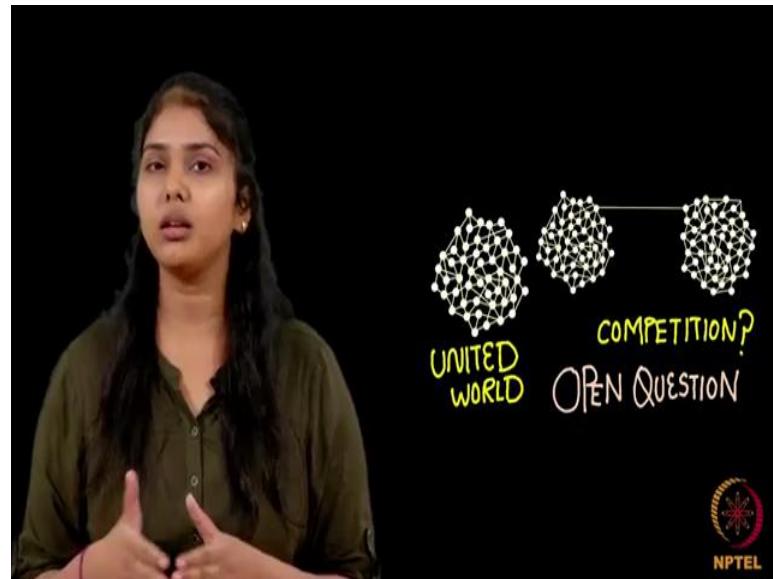
So, it is mainly what fraction of my friends are friends with each other. So, it kind of indicates the amount of the level of bonding between my friends and we have seen that if the clustering coefficient is very less very less of my friends know each other I am more prone to commit suicide.

(Refer Slide Time: 03:10)



But then there are people like Ramya we talked about who have a very less clustering coefficient, but they make use of this clustering coefficient to flourish their business.

(Refer Slide Time: 03:26)



So, how should our world look like was the next question, how should our social networks look like - should everybody here has a very high clustering coefficient, so that the world is united or we need some people like Ramya who kind of act as weak ties connecting these different worlds, so that there kind of immergence a competition between the two. So, which is the best should everything be united or there should be different worlds connected with weak ties and we do not know, it is an open question what should happen this one or this one or a nice combination something in between both of these.

(Refer Slide Time: 04:03)



Next in the league is coming an interesting chapter on homophily. So, you might find Homophily to be a quite a technical term, but in simple words Homophily simply means like a treks like birds of a feather flocked together, that is mainly if I am fat and you are fat we tend to go together, we tend to becomes friends with each other. It might appear to you as a silly idea or stupid idea and realistic, but in the next chapter we will look at how it is true and how it is a very important component in our life.

The second interesting concept we will be talking is about social influence. Influence we know what a social influence and it is again an interesting concept which says that beware of your fat friends.

(Refer Slide Time: 04:52)



Probably if you are hanging out with that fat guy you also tend to gain weight over a period. So, this chapter kind of reinforces our old parental advice watch your company. So, I will not trivial many of the secrets here you go head and watch week 4.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture - 41**  
**Strong and Weak Relationships (Continued) and Homophily**  
**Introduction to Homophily - Should you watch your company?**

(Refer Slide Time: 00:05)



Remember the age-old parental advice. Our parents keep telling us keep good company do not get into bad company.

(Refer Slide Time: 00:11)



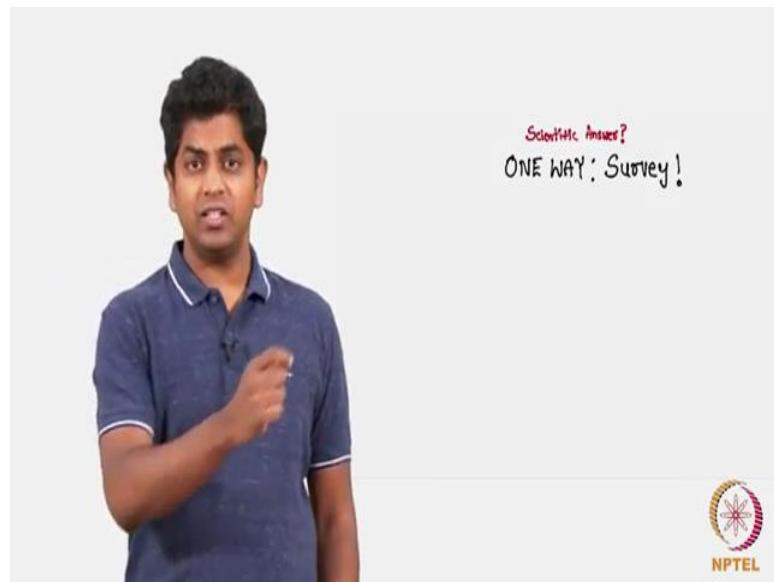
Does this statement even mean anything? How does it matter who your friends are? Do not you think it is you and only you who make makes your character, how on earth will it matter if I have a spoiled brat on my as my friend. It does not matter if my morals or in place, I think I will do well in my life. Is this true? Do you think your friends really make or break your life? Do you think the age-old parental advice that I told you is really true? There is a very well-known proverb it says tell me who your friends are I will tell you who you are - how much of truth does this proverb carry.

(Refer Slide Time: 00:52)



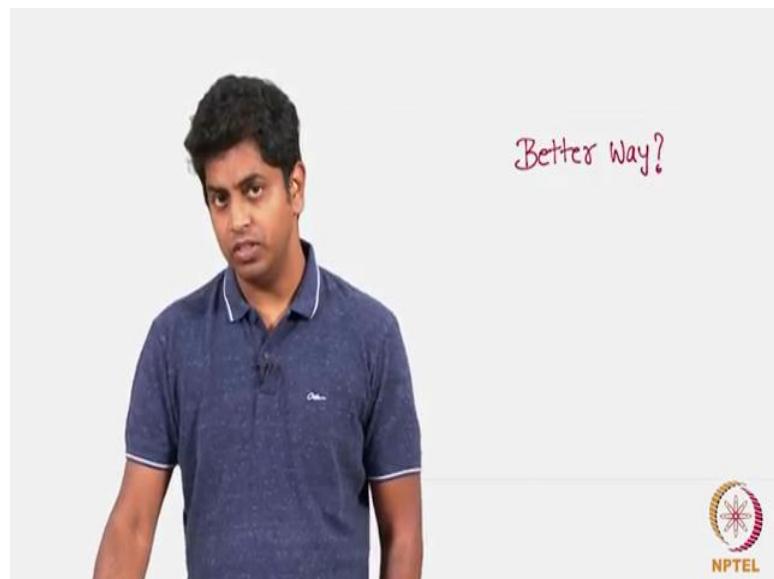
Now, how can one even answer such a question as to how this is true?

(Refer Slide Time: 01:04)



One way of doing it is to go and look at many people and ask who their friends are, look at their traits. For example, I will go and ask this person who are your friends, what are they doing, what is their social status or what is your social status and try to see whether friendships and the character of the person is sort of related, I can go on doing this right.

(Refer Slide Time: 01:30)

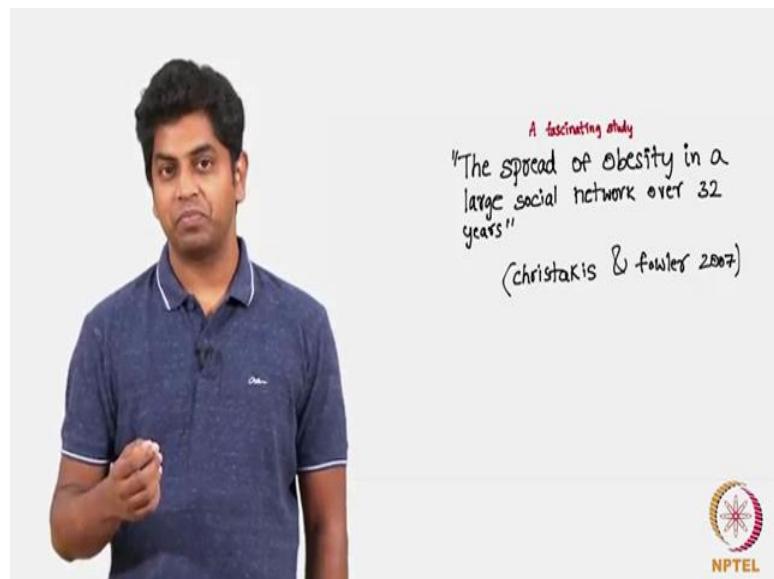


(Refer Slide Time: 01:35)



Is there a better way to do this? Yeah there is there is there is a much better way to do this as a very fascinating study that I am going to tell you people.

(Refer Slide Time: 01:40)



There is a paper that was published in 2007 by Christakis and Fowler which said obesity is contagious what do I mean by this.

(Refer Slide Time: 01:48)



If I am friends with people who are obese, I enter the danger of becoming obese. How do you think this is true? Probably it some fictitious statement that that one can make just like that, can we back it up with some evidence.

(Refer Slide Time: 02:13)



Scientists also observed that it is not just obesity that is contagious, there are several other things that are contagious too on our friendship network and it is the inferences beyond something being contagious. This eye-opening piece of research which says happiness is contagious, takes this to a brand-new level what is it mean? It goes without

saying that if you amidst high energy happy people, you also tend to be happy. You see this is especially true of dogs if you have a pet you will realize that when you are happy and energetic it is very contagious the dog also becomes incredibly happy and energetic and vice versa sometimes.

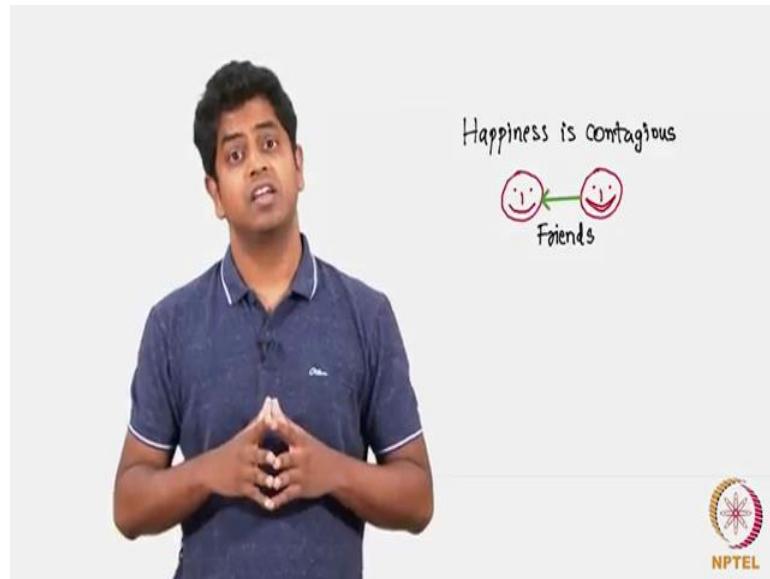
If you meet an old friends who is sort of very who has a smiling face and who is always positive optimistic you miss being with him or her right I mean it is in all I am trying to say that this happiness and energy levels are very contagious. Now what if we repeat this experiment of the obesity being a contagion, with obesity replace by happiness, do you think happiness also is contagious? In the previous question we saw how obesity was contagious, but now if we ask the same question if I am happy is it because of someone who is happy around me, well again a longitudinal study said people who tend to be happy by again by longitudinal study I mean study across several years. This was some 10 to 12 years study of some 5000 people and they observed that happiness is also very contagious, and it spreads in social networks.

(Refer Slide Time: 03:56)



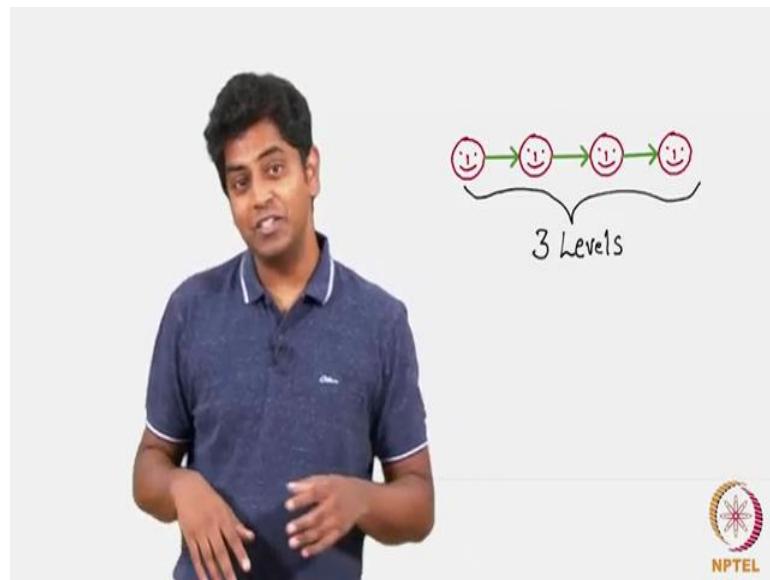
Not just that lot more fascinating than just this.

(Refer Slide Time: 04:03)



They observed, but this happiness is contagious up to three levels which means if you are happy your friends are happy and their friends are happy and their friends are happy I mean that is why the startling, that it is not just at one level, it is up to three levels.

(Refer Slide Time: 04:07)



Assume I have 100 friends, 100 friends have 100 other friends and they have 100 more friends. So, up to the last level people are happy if I alone is happy. So, which means happiness is it percolates from the top to bottom one person is happy all his friends are happy on so on and so forth.

(Refer Slide Time: 04:23)



Very tough to believe, but yeah research says this is true. At least to a particularly good extant that many of your friends are happy when you are happy.

(Refer Slide Time: 04:48)



Getting back to the parental code that I said, keep good company this also means something more than that, not just keep good company work on understanding what is the company that your company keeps because their happiness implies his happiness which is your friend implies your happiness. We should watch up to several levels. So, the proverbial code of tell me who is your friend I will tell you who you are seem to be

partially true in the sense that there is a lot more to it than just your friends. It is your friends' friends who can influence you and their friends who can influence you. Maybe a modern code after this kind of research should be to tell me who your friends are who you are friends' friends or who you are friends' friends are and I will tell you who you are right.

So, this eye-opening research establishes a brand-new line of understanding in social networks that we are not just connected, we permeate our thoughts, we spread our thoughts like a contagion. Just the way yarning is contagious, happiness is contagious. Just the way flues spreads obesities spread unbelievable but true. We will see in the forthcoming lectures how we can program this and then make our observations.

(Refer Slide Time: 06:09)



I will give you a fictitious data set on which you will be writing a piece of code and you will be inferring all these facts. The fact that obesities contagious, happiness is contagious, I will give a big network and I will give you the longitudinal network data across the timeline and you will infer that yes obesity has become contagious from here to here. Happiness does spread to people across several levels.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture - 42**  
**Strong and Weak Relationships (Continued) and Homophily**  
**Selection and Social Influence**

(Refer Slide Time: 00:12)



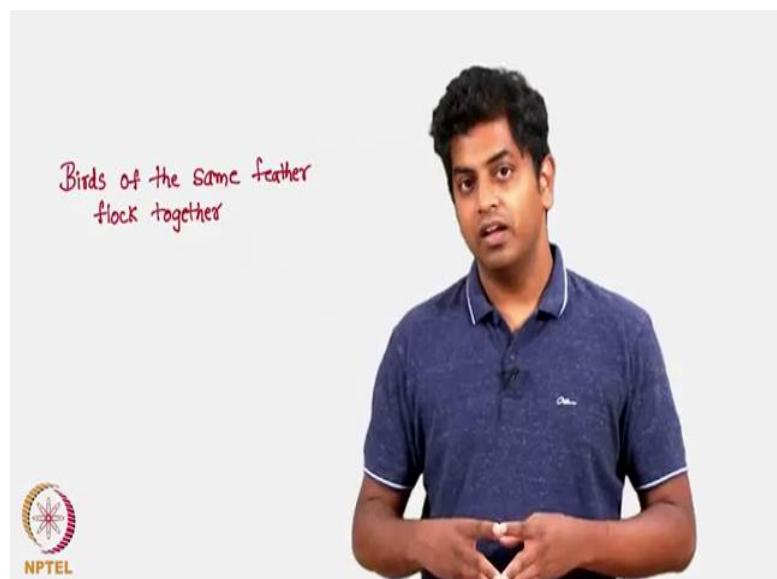
We just now saw a nice, question the question was how we pick friends? Do we pick friends who are our types, or do we pick friends and then we become their types? Let us ask a few questions through examples.

(Refer Slide Time: 00:25)



First question I meet this girl who speaks French I also speak French, we start speaking in French and we become friends. See it is very unlikely that in a place like Chennai I mean it is someone who speaks in French while I also know French this is some common interest, we become friends. This is called selection; we select friends based on the fact that they are our types they are just like us.

(Refer Slide Time: 00:58)

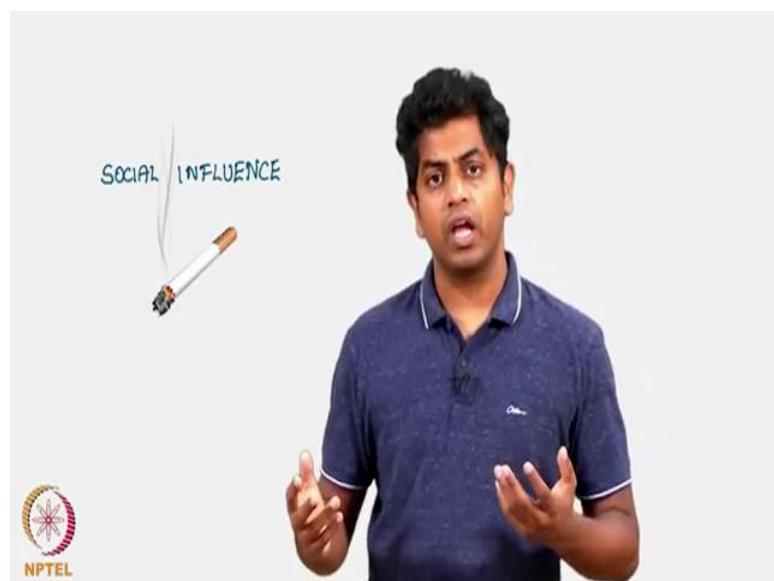


This goes like, you probably know of this code which goes like this - Birds of the same feather flock together, correct. This may not always be true. Let us look at examples and classify these examples into selection or social influence.

(Refer Slide Time: 01:08)



(Refer Slide Time: 01:11)



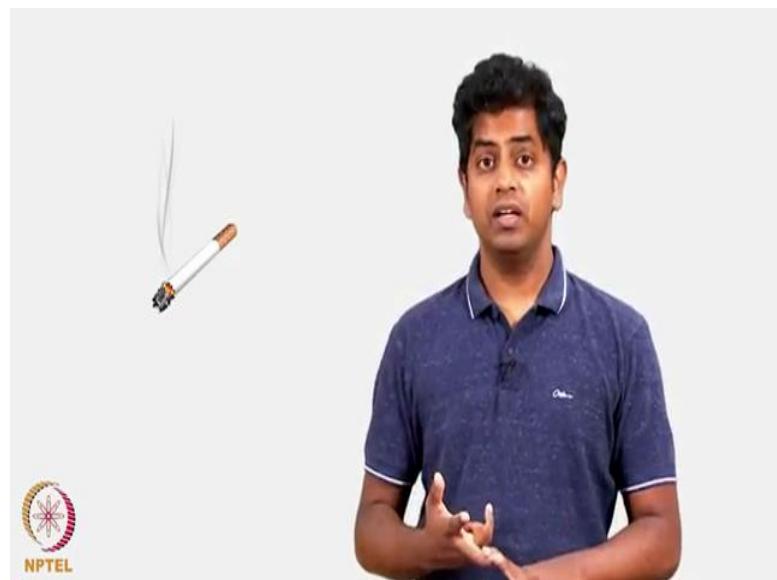
A good example of social influence is smoking. A good example of selection is a girl example I gave you.

(Refer Slide Time: 01:15)



When I meet a person it is known that generally people start smoking because of friends. I meet a friend who is a smoker and he influences me to smoke. Same with drinking too, people do not start drinking because they are alone rather, I mean they do not pick up this habit all by themselves, they generally pick up this habit because of their friends.

(Refer Slide Time: 01:56)



So, example number one: I meet a French girl, I know French we start talking I select her as my friend because we both are similar and then there is this friend who asks me to try

smoking I try smoking I pick up this smoking habit after becoming friends with him I become like him this is called social influence. Let me repeat there is one thing called selection another thing called social influence. The question that you just now saw was what exactly is in play is it selection or social influence. Let us see more examples.

(Refer Slide Time: 02:27)



What about social status? Do you think I become friends with someone whose social status is higher than mine and my social status rises? Not really.

(Refer Slide Time: 02:48)



In fact, two people of the same social status they become friends this again falls under the selection category. Class topper if someone is friends with him, he or she is also a class topper, if she is a class topper if he or she is friends with him he is also class topper, this here again selection is implied.

(Refer Slide Time: 03:03)



Eating habits, you see a person who goes, and eats junk every second day, I probably cannot be friends with him unless I also eat junk right. Here it is social influence. I do not pick a person who eats junk as my friend. I start developing this habit if I am with friends with people, with the person who eats junk.

(Refer Slide Time: 03:30)

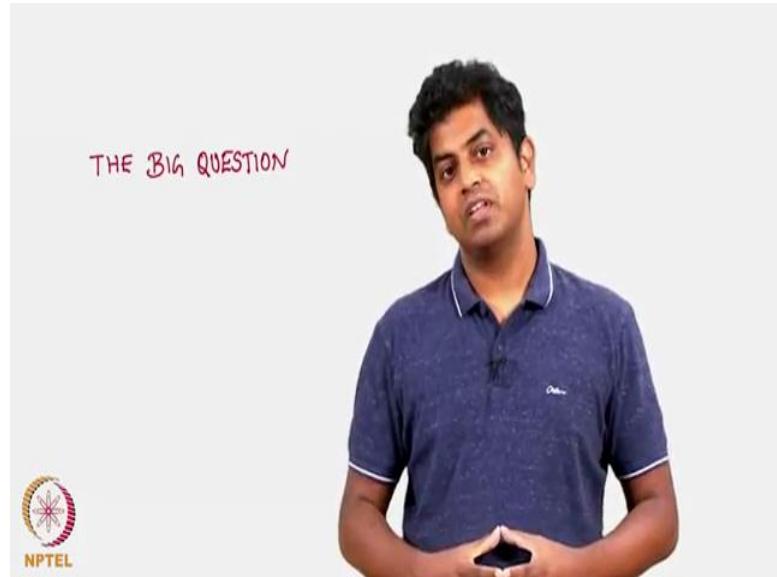


Partying, what is it fall under? I am friends with someone who parties a whole lot I also start partying a whole lots social influence. Smoking as I told you social influence.

(Refer Slide Time: 03:45)

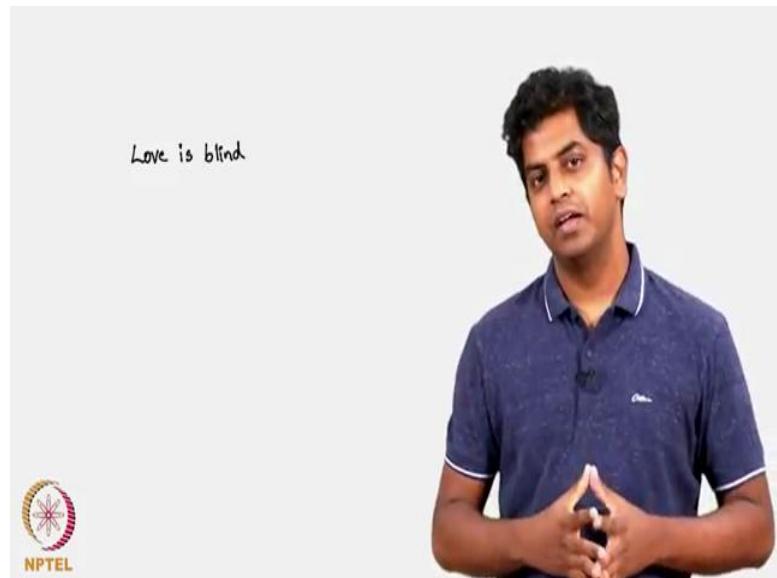


(Refer Slide Time: 03:48)



So, we can classify these things into two broad categories namely social influence and selection. One of the biggest questions is what amount of friendships have social influence in them and what amount of friendships have selection factor into them. Let us investigate more of this slowly.

(Refer Slide Time: 04:05)



So, we all know that they say love is blind I do not know how far that is true, I am not even sure if there is any study which says it is true, but for sure parents do not agree with a fact that they would say it better not be blind, love better not be blind when you fall in

love keep your eyes wide opened and then youngsters definitely agree to this, they say oh yeah love is blind right.

But then if you look at psychologies behavioral economies, evolutionary biologist they have claimed that there is a lot of selfishness in love of course, that aside we probably will not debate more on how people fall in love and what happens in love and things like that.

(Refer Slide Time: 04:46)



But in friendships when people make friendships the dynamics of these friendships seems to be very well understood at least under the domain of social networks.

(Refer Slide Time: 04:49)



So, we will be observing, we will be investigating, we will be seeing how friendships happen and what exactly is in play when friendships happen. As we discussed just now is it social influence or selection or both, if it is both what amount of fit has social influence part what amount of it has the selection part.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture – 43**  
**Strong and Weak Relationships (Continued) and Homophily**  
**Interplay between Selection and Social Influence**

(Refer Slide Time: 00:08)



So, we saw the big question, what is the difference between social influence and selection; which one of the two is happening, are both happening. Now, it is very difficult for us to go ahead and look at the real-world dataset of friendships and ask this question. If we ask this question of, can we have some very concrete dataset where this experiment can actually be conducted, and the facts be unraveled.

(Refer Slide Time: 00:48)



There was one such data set which gave some very promising observations. So, our motive for this video chunk is to understand this piece of research on Wikipedia.

(Refer Slide Time: 00:57)



Let me now motivate the concept by explaining; how Wikipedia works, number one and number two by explaining what one means by similarity measure; these two are important for us to go any further.

(Refer Slide Time: 01:08)



(Refer Slide Time: 01:12)



So how does Wikipedia work? Wikipedia basically is a blank slate; you can basically anyone can go write anything on Wikipedia. For example, you can open the Wikipedia entry of India; edited. If you think some information is incorrect there, delete it add the right content; that you think is right and if someone else thinks; what you have entered is wrong, they will come and edit it and change it to what they think is right this goes on and on.

(Refer Slide Time: 01:42)



One frequently asked question is, can we really trust such a database, where anyone can come and enter anything. Very intriguing, but it is known to be true that when a lot of people get together and start discussing and debating, when they have conflicts of their ideas; true knowledge is known to emerge.

Initially of course, there will be some false information here and there but a lot of people watching it, will converge to the right answer. So, Wikipedia is actually believed to be very trustworthy these days; that is about Wikipedia.

(Refer Slide Time: 02:14)



But what happens in the background of Wikipedia? There are many people who will be coming and editing it.

(Refer Slide Time: 02:20)



If there are conflicts on their belief of what should be put and what should not be put, they talk in the background with each other and the background talking is also visible for all of us; they are called user talk page. One can click on this tab and then take a look at what all discussions they have about the content.

Now, this big dataset is available to our disposal and one can conduct any kind of research on this data set.

(Refer Slide Time: 02:57)



I am now going to explain one such research which actually helped us understand the inter play between social influence and selection.

(Refer Slide Time: 03:09)



As I told you, just now I helped you understand how Wikipedia works; that was the first prerequisite. The second prerequisite is understanding what one means by a similarity measure. So, let me motivate what is the similarity measure now, so let me define what is the similarity measure; it is actually quite intuitive. Let me motivate it with a good example.

(Refer Slide Time: 03:22)



Assume, I like the following dishes I like upma, let us say idli, dosa, pizza and let us say fruit punch; this is the five things that I like upma, idli all those things are South Indian dishes.

(Refer Slide Time: 03:46)



(Refer Slide Time: 03:56)



And let us say my friend Priya she likes pizza, burger let us say pasta and idli; these four things. As you can see, we do not have a lot of commonalities, but we have; we like pizza and idli that is common between both of us. So, what I do is in the denominator I put the total number of items that we like, whatever I like and whatever Priya likes; both put together as you can see is 7 items, out of which we both like 2 items.

(Refer Slide Time: 04:35)



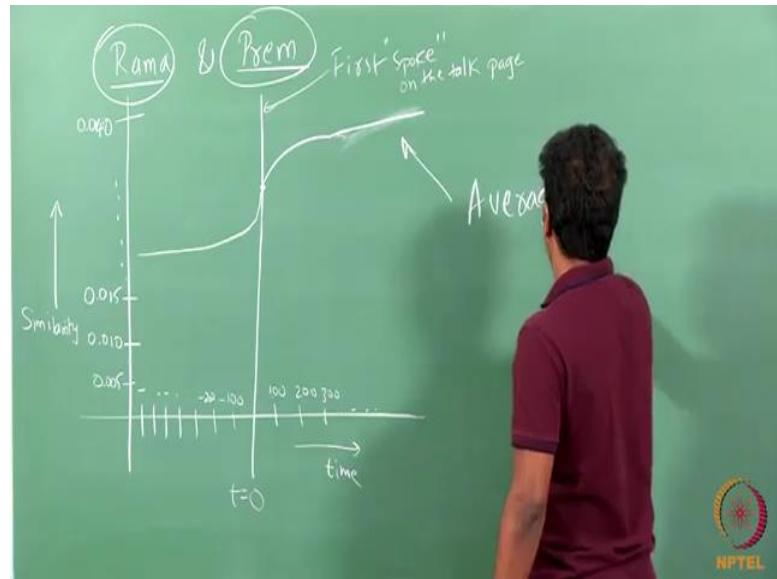
So, I say the similarity measure between me and Priya is 2 by 7; this is what we mean by similarity measures. Now, I am going to use this on my Wikipedia dataset that I was

telling you. So, what do I do? There are two people two authors who are editing Wikipedia pages randomly here and there let us say; I am editing the Wikipedia entry of India and let us say Chennai floods and Wikipedia entry on iPhone 8 and things like that; different Wikipedia pages I am editing. And a friend of mine let us say Peter is editing Wikipedia pages, again some other Wikipedia pages, some ten other Wikipedia pages.

The similarity measure between me and Peter is the total number of Wikipedia pages that we both are editing, total number of unique Wikipedia pages that we both are editing in the denominator; and in the numerator we write those Wikipedia pages we both have edited, just like the similarity measure that I defined between me and Priya. The similarity measure between me and Peter on Wikipedia is total number of Wikipedia articles that we both have edited in the numerator; divided by the total possible unique Wikipedia pages that we have edited, both put together; that is in the denominator. This gives a good measure of how similar we both are on our Wikipedia editing transactions.

So, now that you understand what is; how Wikipedia works, and you also understand the definition of similarity measure. I am going to use these two things in answering the big question of social influence versus selection. So, here it goes; I will do the following, I will look at two people who have spoken to each other on Wikipedia. I told you people do talk in the talk pages, in the background; if I observe that two people have spoken to each other, please note I have a dataset here; in that data set I observe if two people have spoken to each other.

(Refer Slide Time: 06:52)



If let us say two people; I will say Rama and Prem have spoken to each other on the wiki talk page, then what I do is I plot this. What is the plot? The plot is the following, this is my Y axis, this is my X axis, I will see when exactly they spoke. Let me call that time T equal 0 and I will observe, what were they doing before and what were they doing, what did they do after this time equal 0, What do I mean by that?

Let me define what is the X and Y axis properly; on the Y axis, I will put their similarity measure. Their similarity measure is given by let us say 0.005, 0.010 so on and so forth; 0.015 so on up to let us say 0.040; I am doing this on the Y axis. This is my similarity and this is my time; I repeat Rama and Prem have started editing some Wikipedia articles and their similarity measure, I am seeing when they first spoke; this line denotes the time when they first spoke; by spoke let us be clear.

I mean first time they spoke on the talk page and I observed what was their similarity measure before and what is their similarity measure after, it was observed that the plot looks something like this; it looks like this. Let me correct this, it just goes steep it just goes like this that is it. Point to note is the steep increase and further increase and then it becomes sort of constant. What is happening here? Let us just pause and observe; let us revise time, similarly, measure Rama and Prem.

Their similarity measure was; so, this is time when they started let us say some few days before and the first time they met, this probably some minus something minus 100,

minus 200 or whatever is the scale and the time when they first met and then time t equals 100 and then 200 and then 300 so on. Whatever you want, 200 minutes or 200 seconds or you can have any scale here, but this is the time when they first spoke to each other.

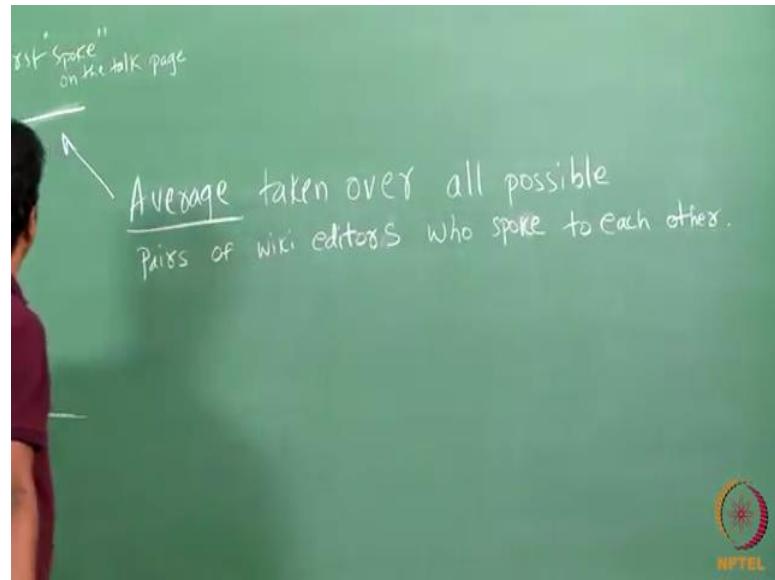
Do you see something happened here? Something fascinating, what does the curve tell you. The curve tells you that before they first spoke to each other, there was a sudden surge of similarity between them. These two people were similar because of which they spoke maybe, because of which they had a talk on the spoke on the talk page maybe. Do you see; after they spoke there was increase in their similarity measure. What does this translate to? This translates the following because that their interest were common, that is what similarity measure means because that their similarity was high, their interest were common; they actually spoke to each other, after speaking there was further increase in their similarity.

This translates the following: two people become friends, if they agree on lot of; let us say dishes in my previous example of me and Priya; maybe if we have 2 by 7 similarity, we may not become friends. If we agree on a lot of food interest, maybe we will become friends; I do not know whether if this is true or not, but assume that people become friends because they have a lot of common food interest. What will happen after we become friends? Once we become friends, we tend to have all the more common food interest, we will probably will hang out together and eat more often the same kinds of dishes, give each other our testimonials of this kind of food in this kind of place and we may want to try these things in each other's presence.

So, the fact that we are similar makes us talk; the fact that we have spoken right now makes us become all the more similar. Fine, this plot was given to you for reasons of better explaining things to you, but the fact here is; if you look at the textbook, this plot is the plot of the average. By average, I mean they have looked at all possible people who have actually spoken and they observed; what was their similarity measure before and what was their similarity measure after. This is the average plot that you see in the text book; which says that on an average people tend to become more similar and then talk and then become all the more similar not just this.

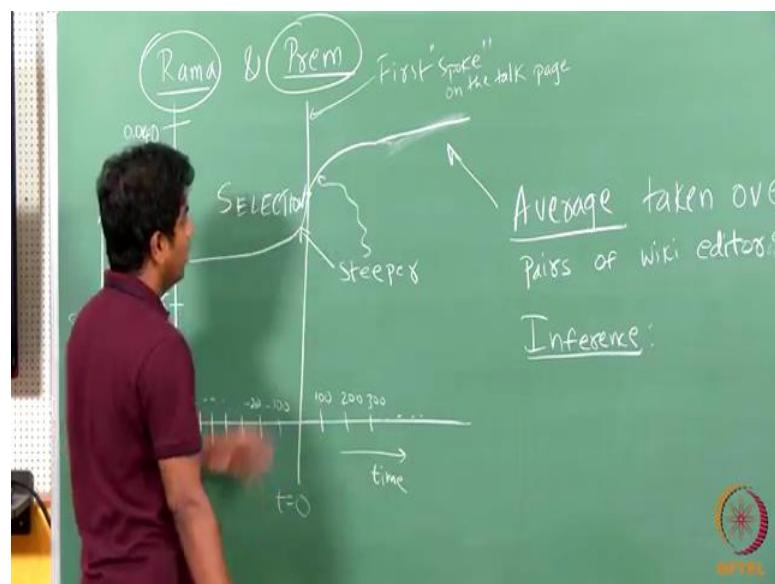
This; what is coming next is what makes the research all the more interesting. They observed; I repeat on an average people become similar; people because this is average, I told you.

(Refer Slide Time: 12:59)



This is average taken over all; let me write that down, average taken over all possible pairs of wiki editors; who spoke to each other, sorry who spoke to each other and we observed that; when they speak at this time, there is sudden increase in their similarity measure before speaking and after speaking; this is the average curve.

(Refer Slide Time: 13:49)



A mega inference that one can make here is that this is steeper than this, now that is the climax point in the piece of research. This is steeper than this; let me write that, this is steeper as compared to this. This is steeper; than this, what does that mean? That means, that you become similar and then you talk and then you continue to become better and better similar. So, what is this and what is this; take a minutes pause and observe, what is this and what is this in our language that we are motivating from the past few minutes.

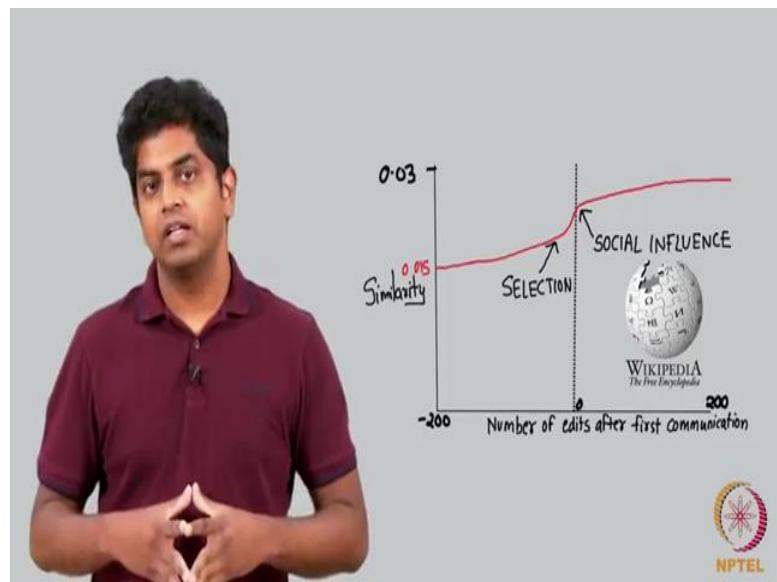
Which one is social influence, and which one is selection? When do we select? We select when people are similar. So, I am sure you would have figured out the answer right now, two people select each other to talk when they are similar. This is selection here happening on this part of the vertical T equals 0 line selection and on this part; once they spoken to each other, maybe they will share more common interest and they will started eating pages together, that is what is happened, that is what this means; there is steep increase in the curve here as well and this is going to be our social influence.

(Refer Slide Time: 15:21)



So, we just now saw the plot and we observed the interplay between selection and social influence on a particular type of dataset.

(Refer Slide Time: 15:34)



Now, let us get a little critical and ask this question; fine the plot is very clear, it says that initially there is a lot of selection happening here and then social influence is also happening, selection seems to be more in play than social influence while both actually happening; that is with this dataset. Maybe this kind of a observation is very context dependent, dataset dependent.

(Refer Slide Time: 16:16)



Now, instead of taking Wikipedia dataset, if I were to take some other data set, let us say the way smokers become friends with each other. Maybe there is a lot of social influence

factor there, the un-selection factor correct and in fact, obesity research says otherwise whatever we saw here it is the reverse of that. Here, there is more selection than social influence, while in the obesity research what we inferred was there is more social influence than selection.

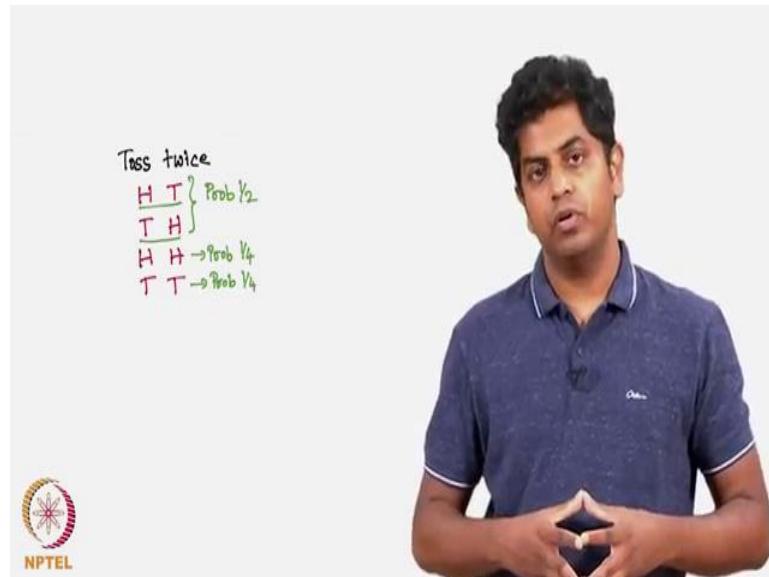
This is a very context dependent observation the people have made in fact; this opens up brand new questions namely. Which is more in play in the society? So, be it with for bad habits, is it more of social influence than selection. For good habits, is it selection and less of social influence; one can ask many such questions and thankfully we do have some datasets on which we can experiment these questions and then find our answers.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture – 44**  
**Strong and Weak Relationships (Continued) & Homophily**  
**Homophily – Definition and Measurement**

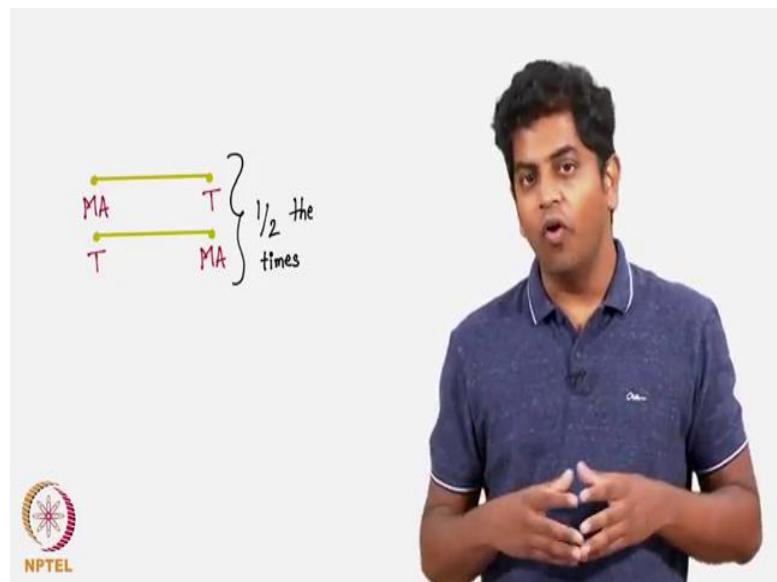
Let us now consider an example assume there is a party going on with 50 teenagers and 50 middle aged people. Now there is some kind of a friendship that gets formed between these two people these two groups of people, and now if I pick a friendship let say uniformly it random I should see on an average one person should be the middle aged person, another person should be the teenage person right. So, this is analogous to you toss a coin twice you will expect a head and a tail or a tail and a head or will be head and a head and tail and a tail, but you observe that head and a head comes with a probability with  $1/4$  head and a tail comes with a probability.

(Refer Slide Time: 00:51)



1/2 by that I mean a head and a tail and tail, and a head comes with probability 1/2 a tail and a tail come with probability 1/4.

(Refer Slide Time: 01:06)



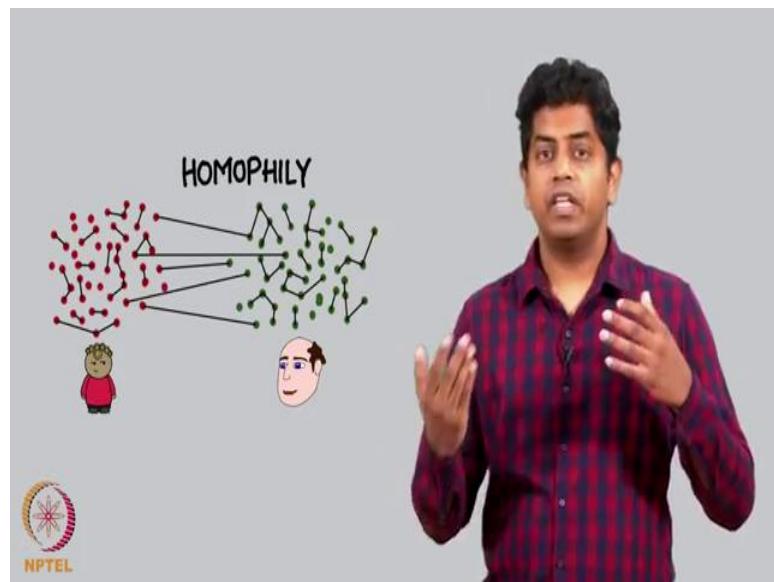
So, it is very likely that if you pick some friendship out of this 100 people, that friendship will have one middle aged and one teenage person half the times that is how I think mathematically it appears to us.

(Refer Slide Time: 01:21)



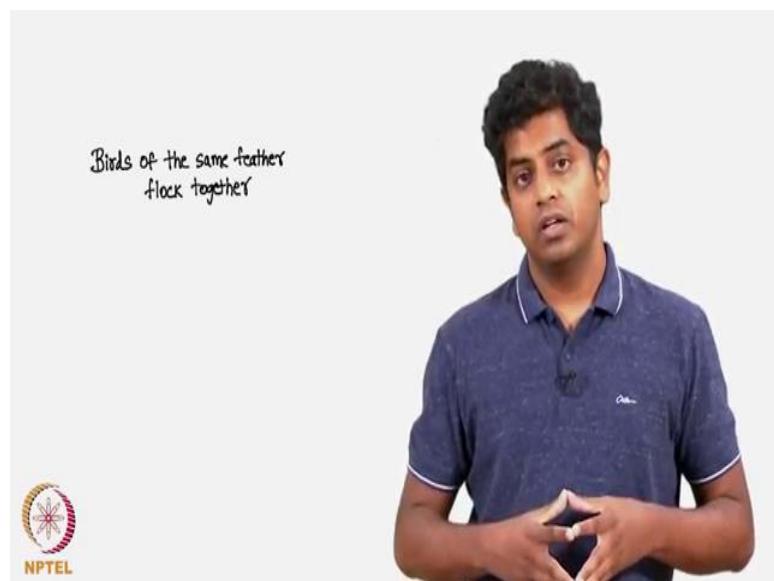
But you all know the teenagers may want to talk to only teenagers, and middle-aged people.

(Refer Slide Time: 01:31)



May want to talk to only middle-aged people this is called homophily again the code that I was telling in a previous lecture videos.

(Refer Slide Time: 01:38)



Birds of the same feather flock together.

(Refer Slide Time: 01:42)



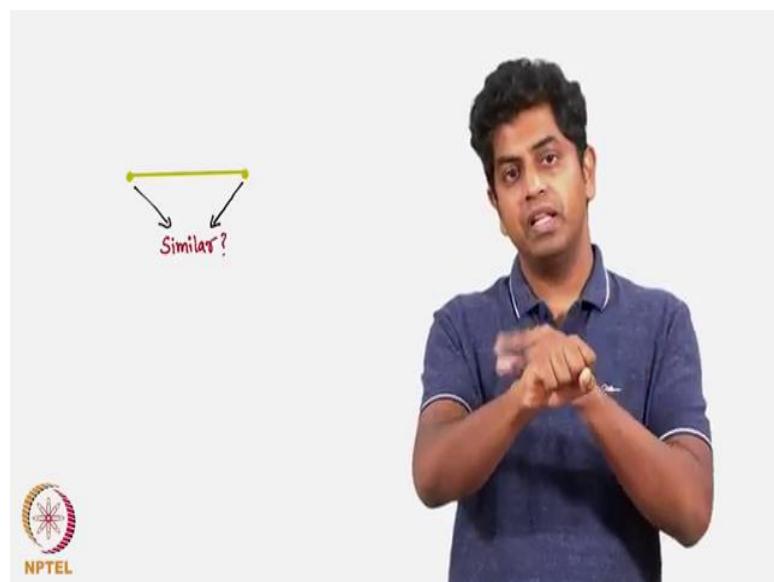
Now, let me ask this question do you think this is true if there is a bunch of people let say 100 people of the same age group and they are sort of there are 50 men and 50 women half men, and half women and they are sort of in a dating spry they want to talk with each other then definitely across genders you will observe that any friendship will have opposite genders definitely for sure.

But then the previous example of middle-aged men and teenagers you saw that tend to cluster middle aged men would be friends with middle aged men, teenagers would be friends with teenagers correct.

(Refer Slide Time: 02:27)

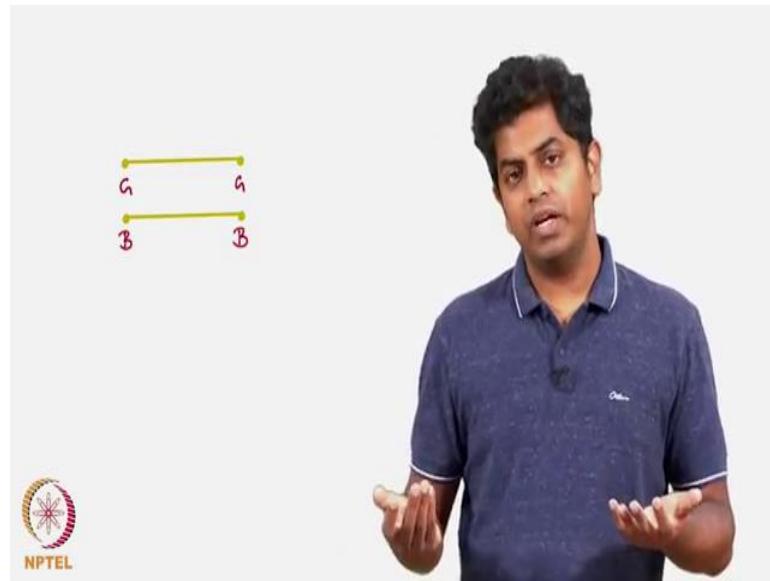


(Refer Slide Time: 02:38)



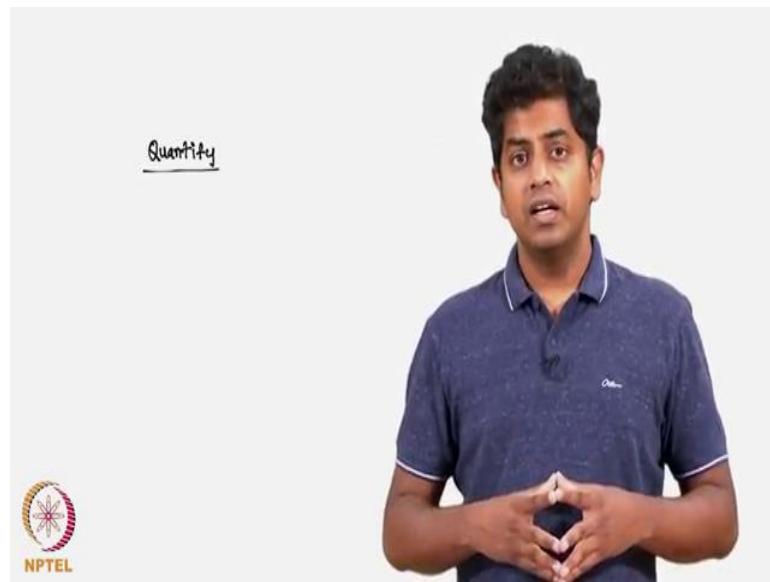
So, what will do, what will now do is given a network will observe, does it exhibit homophily what do you mean by this? By this I mean given a network if you pick a friendship an edge across the edge are people similar.

(Refer Slide Time: 02:47)



For example, in a classroom if you go and observe that girls and girls are basically friends with each other boys and boys friends with each other girls and boys are also friends, but quite comparatively rare right unless it is a romantic relationship.

(Refer Slide Time: 03:01)

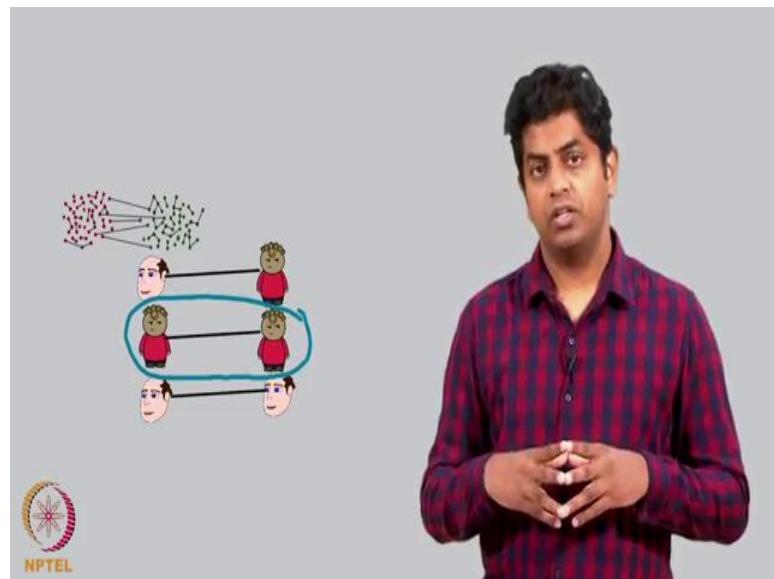


So, we will now try to quantify; what do I mean by quantifying this? First question that we can ask is does the network exhibit homophily this is like asking do you have milk in the vessel the answer is yes or no. If the answer is yes, then you ask another question how much milk you have in the vessel. Similarly, you ask this question does the network

exhibit homophily if the answer is yes you ask the next question to what extent does it exhibit.

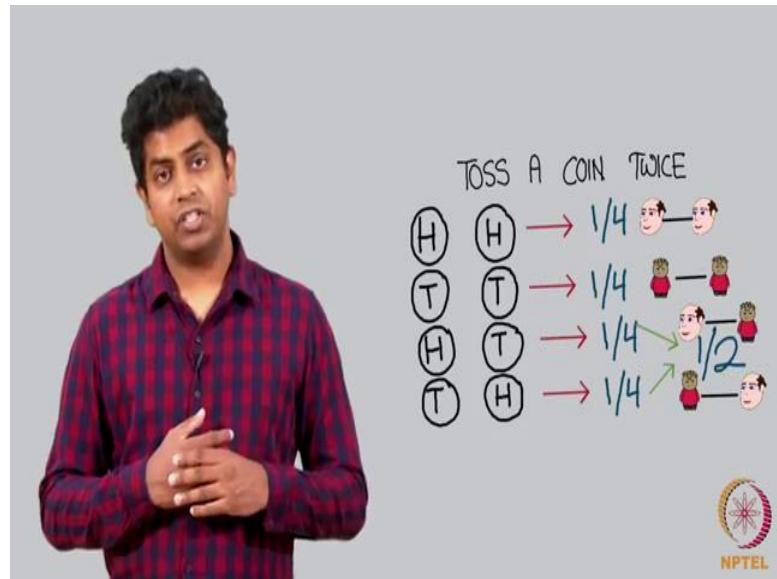
Let us now define what is homophily rigorously? Let me use the same example assume there are 100 people in a party hall, 50 of them are middle aged people and 50 of them are teenagers. Now there is friendship between these 100 people which is actually classified as teenagers and middle-aged people. Now what do I expect? I expect that all of them talk to each other become friends with each other irrespective of whether they are teenagers or middle ages.

(Refer Slide Time: 04:17)



Let me take one such friendship and then observe whether this friendship I pick a random friendship and I see whether this friendship is across a middle-aged person and a teenager or between two teenagers or between two middle aged people. Now what do you suspect would be the answer for this? If I ask you what is the possibility that if you pick a friendship, uniformly at random, that friendship is actually between two teenagers, it is in fact easy to see that this is very similar to a coin tossing experiment.

(Refer Slide Time: 04:53)



I toss a coin twice let say and I get what are the possibilities that I will get a head and a head a tail and a tail a head and a tail or a tail and a head right all these 4 possibilities are equiprobable

Now what is the possibility that I get head and head  $1/4$  pretty obvious, tail and a tail  $1/4$  a head and a tail and a tail and a head both put together is  $1/4 + 1/4$  which is  $1/2$ . Now is in this what is happening in our example where I said a party with some 100 people half of them are teenagers, half of them are middle aged people and friendships between them is what we are observing; half the friendships should be between middle aged and teenager people, and quarter of them should be between teenagers and quarter should be between middle aged and middle aged.

Now, here are two contrasting examples.

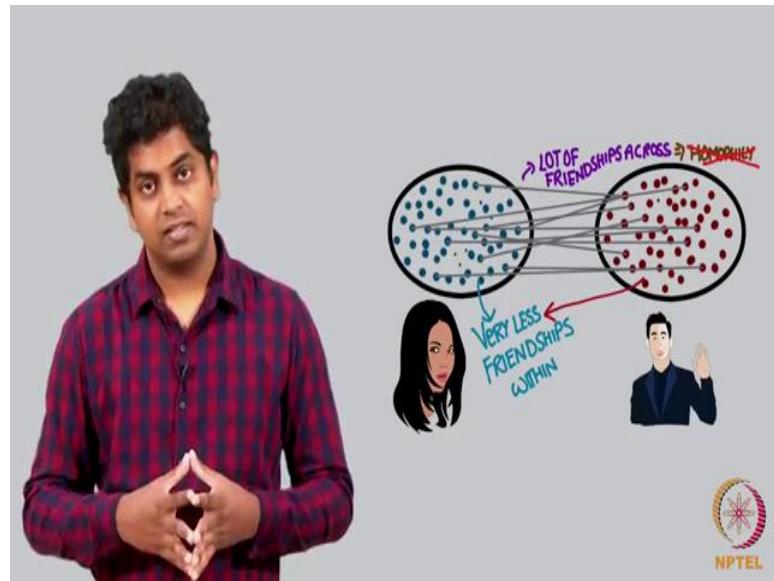
(Refer Slide Time: 06:03)



One example is middle aged and teenagers in a party hall, what do you expect? You will definitely expect there is a whole lot of friendships between teenagers and between middle aged people by that I mean the expected probability of half which is the friendships between middle aged and teenagers across that is half right that will not divisible infact that will be a lot less it will be some 18. If the friendships across these two parities are less than half, then we say we observe homophily here this makes perfect sense you see let me rephrase.

Party with 100 people half of them 50 of them are teenagers, 50 of them are middle aged people you pick friendships and observe what kind of friendships people have, you will observe that the friendships between teenagers and middle aged people are a lot less what do you mean by a lot less? You expect it to be half the friendships out of all the friendships half of them should be between middle aged and teenagers, but you observe that less than half is between middle aged and the teenagers, which means there is homophily happening here, there is the that the same old court I have been repeating birds of the same feather flock together is what is happening here.

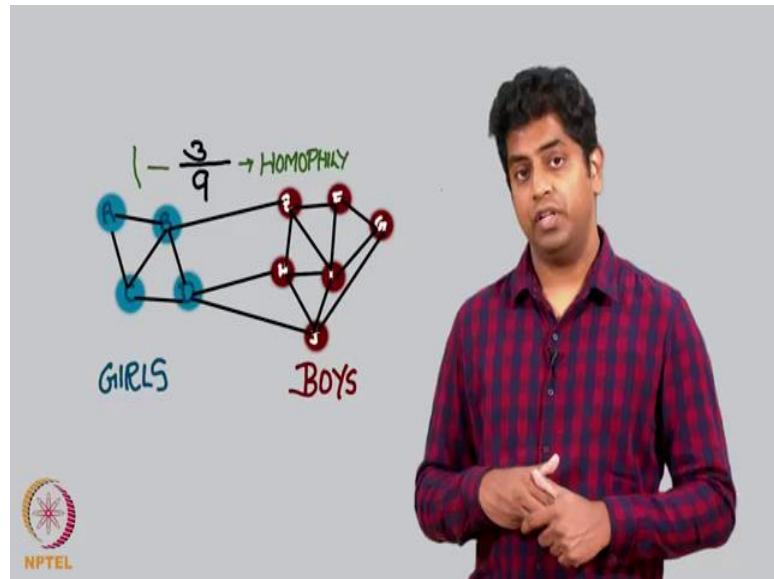
(Refer Slide Time: 07:43)



Now, let us look at the contrasting example; the contrasting example is this assume there is a party hall full of Bollywood hero and heroines right. So, a 50 of them are Bollywood heroes 50 of them are Bollywood actress. Now what kind of friendships will happen here? I suspect that the actors have the tendency of going and talking to the actress more than actors themselves. An actress has the tendency of talking to an actor than a fellow actress. Now if I look at the kind of friendships that happen in a party hall between a Bollywood actors and actresses what would you observe think for a moment. You will observe the exact opposite of the previous example right.

Now, there will be very less friendships within the actresses and there will be a very less friendships within the actors there will be a lot of friendships across. Now what is happening here? There is no homophily here absolutely if anything there is very less homophily here correct. Why? Pick an edge out of all friendships that happen in this party hall of Bollywood actors and actresses, pick all the friendships and see how many of them are actually between the actresses and between the actors, you will observe that a lot less is between the same type of people, a lot more will be across right which means every friendship has different types of people across the edge, which means homophily is a lot less.

(Refer Slide Time: 09:25)



So, let us now look at a nice example; look at this network of 10 people let say in a class of 10 there are four girls namely a b c and d and then 6 boys e f g h i and j, and this is the network between them there is some friendships there is a friendships fine done. Now let me see what the nature is of friendships. Let me look at one such friendship let say between a and b as I can see friendship between a and b is present and a and b are both girls. If I randomly pick a friendship and look at who are on the other side of the friendship do, I mostly see two boys or two girls or do I also see a boy and a girl now let me quantify this properly. Let me first count the total number of edges on this network as you can see there are 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 edges here, which means 18 friendships are there out of this, 18 friendships I expect half of them as we are being discussing I expect half of them to be between boys and girls.

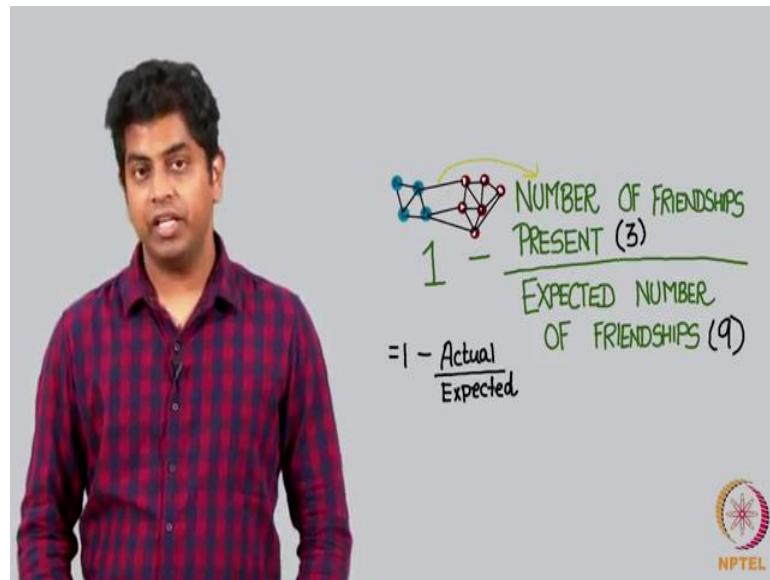
Now, let see how many of the friendships are actually between boys and girls. I observe that 1 2 3 and 4 these are the only four friendships between boys and girls, no not four actually it is three one two and three just 3 friendships between boys and girls, the rest is either between boys or between girls.

So, I now try to define what is homophily. By Homophily I mean take look at this fraction in the numerator you put what are the actual number of friendships across the divisions boys and girls that is in the numerator, in the denominator I put the expected number of friendships between boys and girls. Since there are 18 friendships, I expect

the number of friendships between boys and girls to be half of 18 which is 9. Under the numerator is the actual friendship, across boys and girls which is 3. So, 3 by 9denotes homophily correct. Lesser this fraction more is the homophily think about it, more this fraction lesser the homophily why is that? If you think the numerator is very small, then the fraction becomes very less. What do you mean by numerator is very small? By numerator small I mean that the actual number of friendships across boys and girls is very small, which means there are lot of friendships happening within them which means there is a lot of homophily.

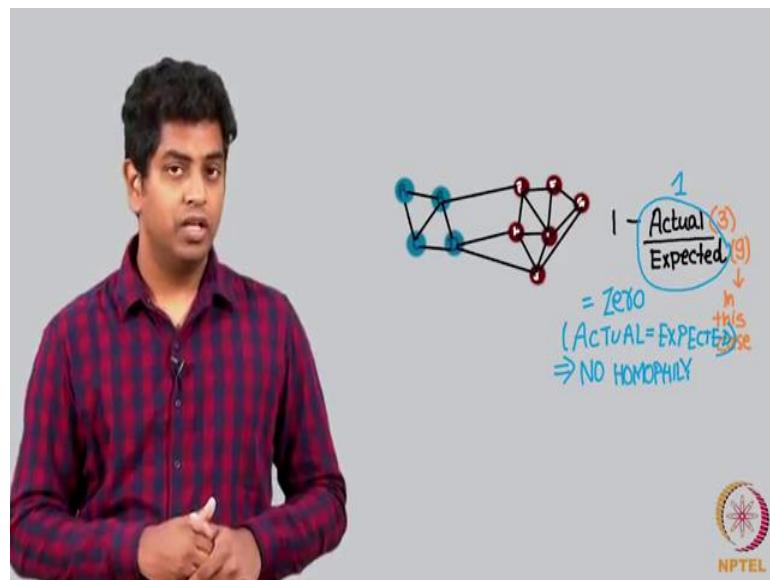
Now, what we do is a typical trick that we do in math physics, computer science in general is try to subtract this from one. So, what I do is I take 1 minus this fraction, actual number of edges divided by expected number of edges what does this denote? This simply denotes that as this number goes higher there is more homophily, as this number goes lower there is lesser homophily.

(Refer Slide Time: 13:03)



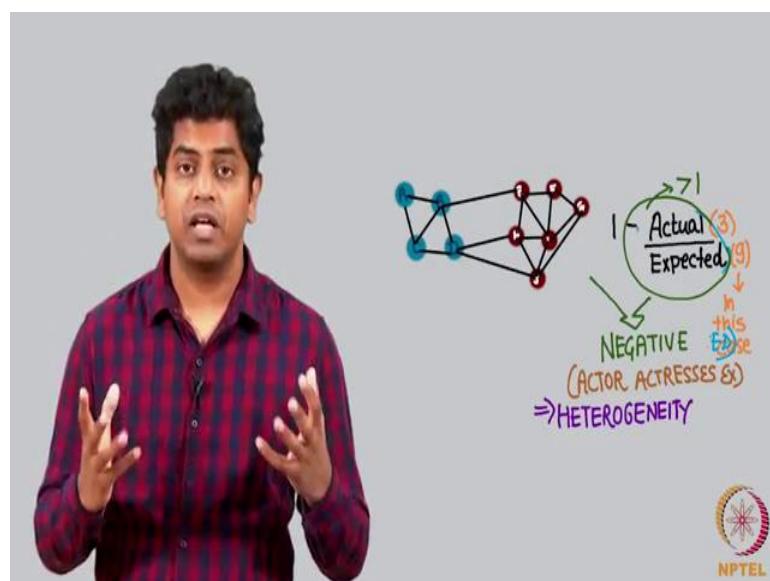
Let us look at this number do you look at the fraction? The fraction can actually become equal to 1 and hence 1 minus 1 can be 0 what does 0 denote here?

(Refer Slide Time: 13:10)



0 denotes that the number of edges across these two parities boys and girls, is the expected number of edges which means homophily is not happening here. Please note there is also possibility that this big number this entire number 1 minus this fraction can become negative; when is it negative? It is negative when the actual number of edges are more than the expected number of edges.

(Refer Slide Time: 13:36)



Do you see the actor actresses party example that I gave you, where friendships are mostly between two people of the opposite gender? Now what will happen? The

expected number in the denominator and the actual number in the numerator the actual number will be more than the expected number in that case 1 minus of this will become negative; and the negative number denotes the heterogeneity in the network, where friendships are actually between two people who are of opposite types, and a whole lot of friendships are like this.

(Refer Slide Time: 14:31)



Now let summarize if the friendships are such that across the friendship two people are similar more often, then you call such a network as homophily the network exhibits homophily. If they are half the times different then you say the network looks very random, then this number is 0, 1 minus actual by expected is 0. When almost always the types of people across this edge are different like in the actor actresses example party example, then you say such a network exhibits heterogeneity and is no way homophilic.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture – 45**  
**Strong and Weak Relationships (Continues) and Homophily**  
**Foci Closure and Membership Closure**

Neeru (Refer Time: 00:10), I wanted to ask you something (Refer Time: 00:12) I think we are batch mates.

Yeah, you are right; I was there but I remember seeing you in Delhi University.

Yeah, you are correct and see we are here in the same yoga class.

Yes.

What a co-incidence?

Yes.

I think we should catch up sometime (Refer Time: 00:25) for a coffee?

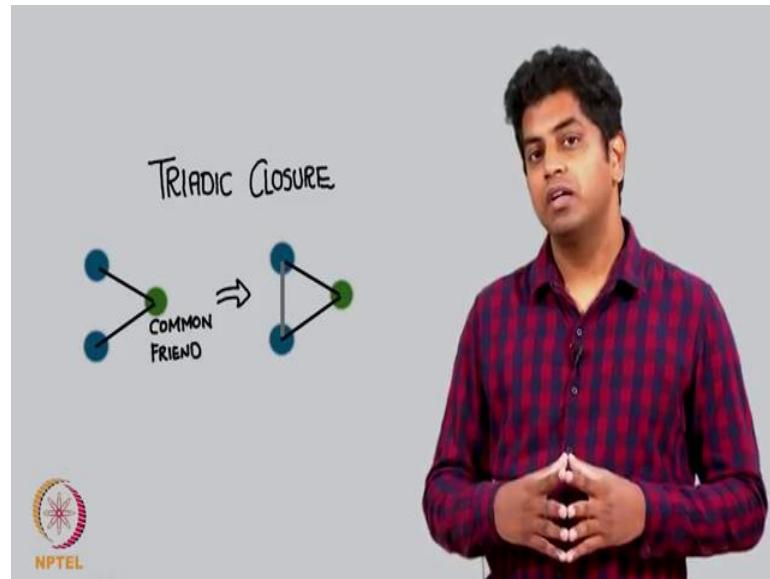
Sure; no problem.

Tell me regarding; which batch you are?

I was in (Refer Time: 00:31)

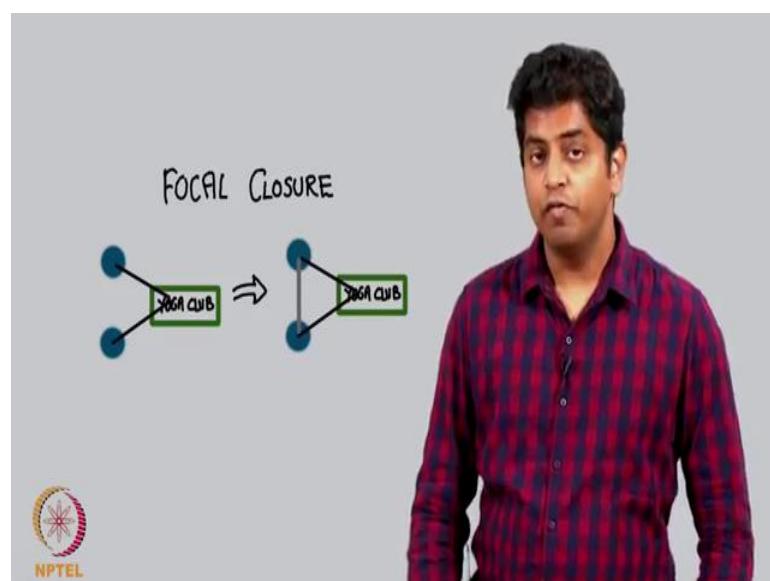
We just now saw a video clip of Amit and Neeru being part of yoga club and then they realized that they are actually part of several affiliations while, we have been looking at triadic closures; where 2 people who have a common friend trying to become friends with each other.

(Refer Slide Time: 00:47)



It is not just a common friend which makes 2 people get closure, sometimes it also affiliations. If they are part of a common focus; let us say me and you are part of kung fu club, we trying to be friends with each other. If we both are part of yoga club, we will be friends with each other. In the example between Amit and Neeru, they were part of yoga club and then they realized that they are part of many other common foci, they are now becoming friends with each other; this is called focal closure.

(Refer Slide Time: 01:29)



Where, we saw what a triadic closure is. Similar is focal closure, triadic closure meant common friend 2 people become friends. Focal closure means 2 people have a common place where they meet, where they talk, where they see each other and that can also result in the friendship that can happen between these 2 people that is called focal closure.

Now, there are basically three types of closures that we will now observe; first is of course triadic closure, second is focal closure that I told you, third is membership closure.

(Refer Slide Time: 02:07)



Assume me and Ramesh are 2 people who know each other and I go to the music club and now because he is friends with me, I will take him to the music club; this is called membership closure, I make my friends as members of this focal point; which is let us say music club.

(Refer Slide Time: 02:31)



So to per phrase, we saw triadic closure; common friend means 2 people can become friends, focal closure; if there is a common focus, let us say Kung Fu Club, yoga club; 2 people can meet there and become friends. The third one is membership closure; I am part of a music club; I can make Ramesh become part of the music club.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture - 46**  
**Strong and Weak Relationships (Continued) and Homophily**  
**Introduction to Fatman Evolutionary model**

Now, we are going to do a very interesting screencast, programming screen cast. So, what will be doing in this programming screen cast is, to make an evolutionary model. What is an evolutionary model? We have studied 3 main concepts in this chapter.

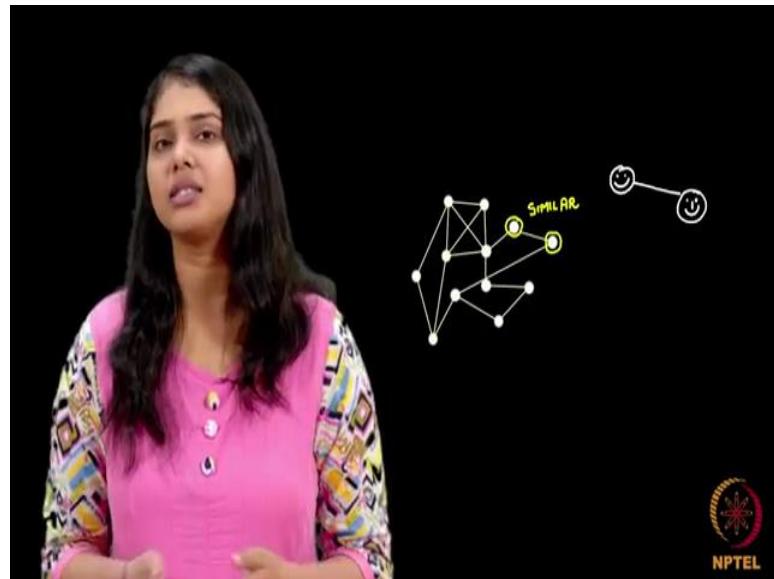
(Refer Slide Time: 00:26)



So, the first one is homophily; people who are alike similar to each other, they tend to become friends with each other. Then we looked at 3 different kinds of closures triadic closure, focal closure, membership closure and we also looked at social influence.

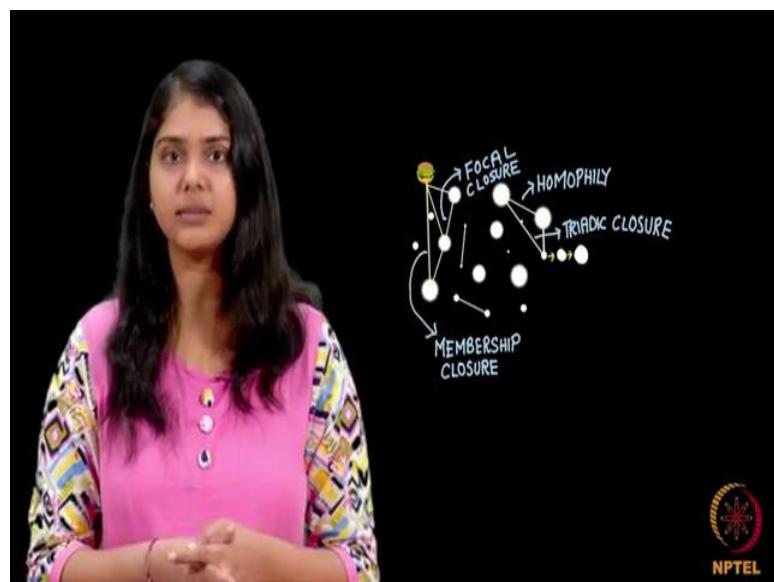
So, in a graph, in a network of these people these people keep changing, so first of all new edges keep coming in this network.

(Refer Slide Time: 00:54)



So, when 2 people look at each other and they find that they are similar to each other; they become friends with each other. So, you see that this network is evolving with time, more friendships are coming in this network and probably some friendships are getting broken also. And then people they change their properties because of social influence; for example, somebody who is not getting good marks becomes friend with a topper and starts getting good marks. So, we have also seen these kinds of things known as social influence.

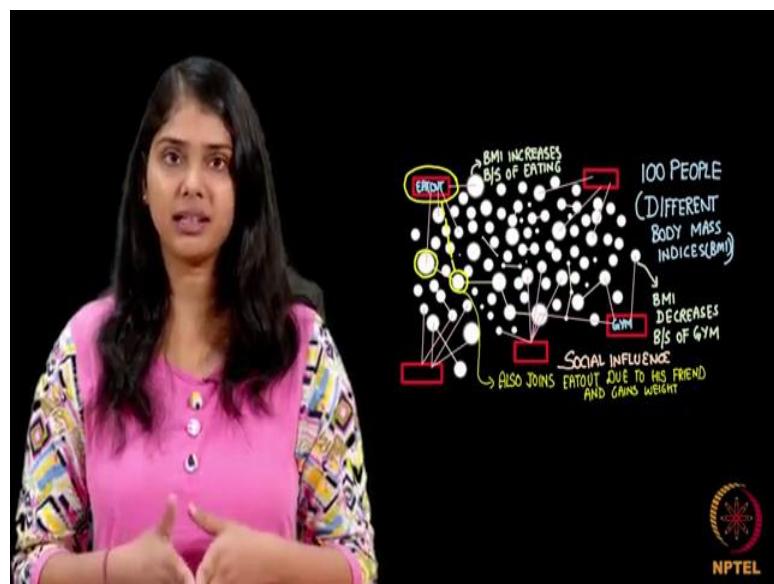
(Refer Slide Time: 01:34)



So, we will try to model all these 3 things as an evolutionary model; what will be modeling is, there is a city and this city has different-different kinds of people; kinds of people in terms of their body weight. So, I hope you remember this Fatman hypothesis which we discussed during the chapter that; beware of your fat friends, if your friend is fat the probability of you gaining weight increases. So, we will be modelling a city in which there will be people with different-different body weights, different-different BMI's and then we will look at all these 3 phenomena happening over this network. So we will see that, we will code it; mainly we will code it that people who are having similar body weight, they tend to become friends with each other. We will be looking at triadic closures also; 2 people having same friend tend to become friends with each other.

We will be looking at foci closure also; that is if 2 people they are going to the same eat out place or let us say the same gym, they tend to become friends with each other and we will also be looking at membership closure. So, assume I am a friend with somebody who regularly goes to an eat out place, so I also start going to that eat out place. So, all these different kinds of closures and we will also be modelling social influence that will be in a more subtle way, which I will tell you, so how does this programming screen cast proceed.

(Refer Slide Time: 03:04)



So, first of all we have this network and let us say there are 100 people in this network and these 100 people have some BMIs. So, we assign some random BMIs to each of these persons and then we look at; when the model proceeds, when the evolution continues people who are having similar BMI, they make more and more edges with each other. And then we also have; five, we will introduce five social foci in this system as I tell you and 2 social foci out of these will be one is an eat out and another is a gym; which has a direct impact on the BMI's of these people.

So, these people they will be linked to different-different social foci and this social foci will change the property associated with these people; that is, if you are part of a gym you will be losing weight, if you are part of an eat out place you will be gaining weight and then how do we implement social influence. One of the reasons for social influence, if you remember or the sheared context, so if one person goes to an eat out place; I will also start going to that eat out place and I start gaining weight.

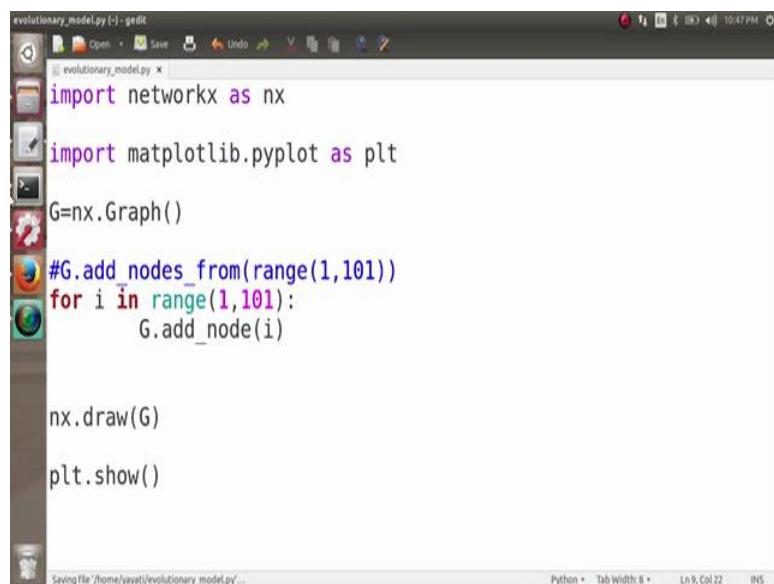
So, this is the way in which we will capture the influence of; we will capture the phenomenon of social influence. So, I might have less weight, but then over time I might become friends with somebody because of maybe let us say a triadic closure. So, we have a common friend and this common friend makes both of us friend, but now this new friend which I have made is going to an eat out place; I am in his company. So, I will also start going to this eat out place and I start gaining weight.

So, we will see how we code such a complex scenario and one assumption in this programming screen cast that we will be taking, although there are many assumptions. One main assumption is that, new edges only come with time; previous edges as in the existing edges they are not deleted, only the new edges come in this network. But you will see once when we code the entire thing, it is actually very easy and you can modify this model in many ways you can have old edges leaving the network, you can increase the number of social foci, you can increase or decrease the number of people and you can incorporate many new interesting things in this model.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture - 47**  
**Strong and Weak Relationships (Continued) & Homophily**  
**Fatman Evolutionary Model - The Base Code (Adding people)**

(Refer Slide Time: 00:05)



```
evolutionary_model.py (~) - gedit
import networkx as nx
import matplotlib.pyplot as plt
G=nx.Graph()
#G.add_nodes_from(range(1,101))
for i in range(1,101):
    G.add_node(i)
nx.draw(G)
plt.show()

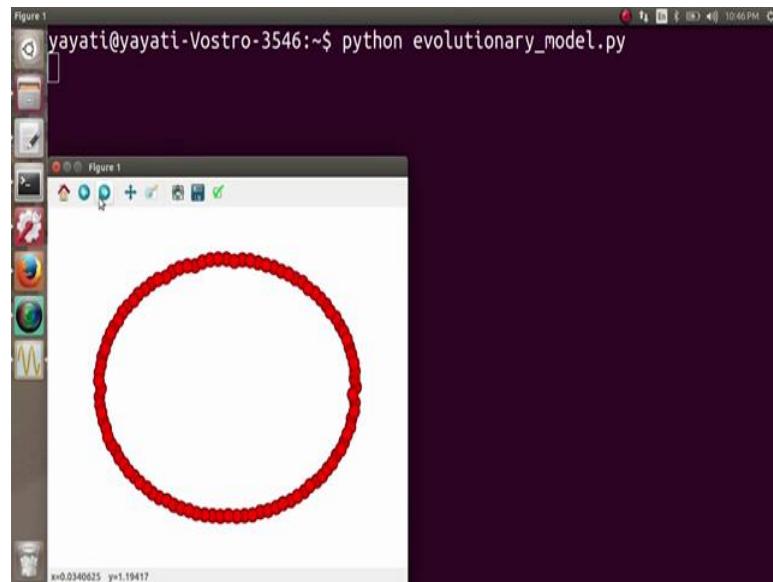
Saving file '/home/yeyati/evolutionary_model.py'...
Python Tab Width: 8 Ln 9, Col 22 IN5
```

So, what do we do first is we have to create a graph on 100 nodes where everybody has a different BMI? So, mainly we have this city and the city has 100 people which we depict as 100 nodes. So, to make this graph first we import a function sorry be import a package import networkx as nx and let us save this file, let us save this file as let say evolutionary model.py. So, we save this file a evolutionary model.py. The import networkx is nx, now we have to create a graph on 100 nodes which we know is very simple we have already created many graphs.

So, what we simply need to do is first of all create a graph  $G = nx.Graph$  and then we have to add 100 nodes here. So, we write  $G$  dot `add_nodes` from and this function demands a list here. So, what is the list here? Range 1 to 101. So, what will this function do it will add 100 nodes to this graph with the indices being 1 2 3 4 up to 101. So, this is one we do will get will soon look at another wave of doing the same thing.

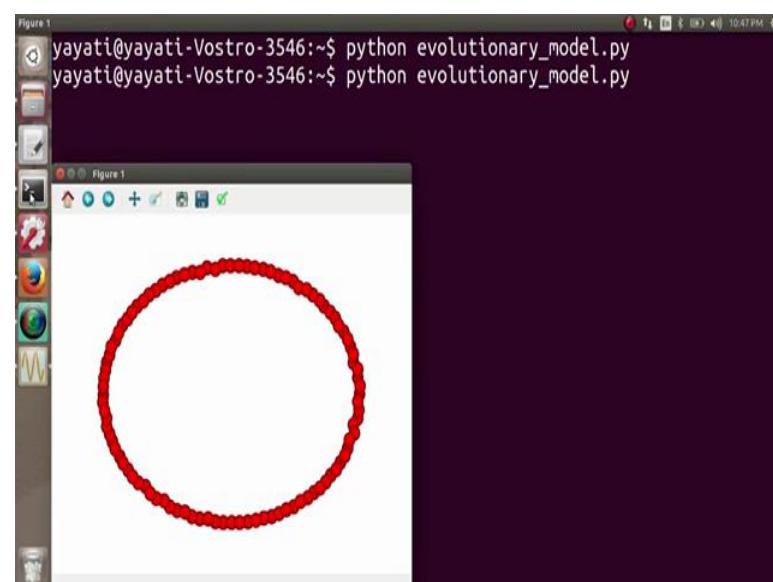
And next I want to plot this graph and see how does it look like, so far as we know for plotting we need import mat plot lib dot pyplot as plt and then we can use a function and next dot draw G and then plt.show.

(Refer Slide Time: 02:07)



I save this file let us run it and see, python evolutionary model.py and here we can see a graph with 100 nodes. Let us quickly look at another way of doing the same thing for you might be already knowing it I am just repeating it.

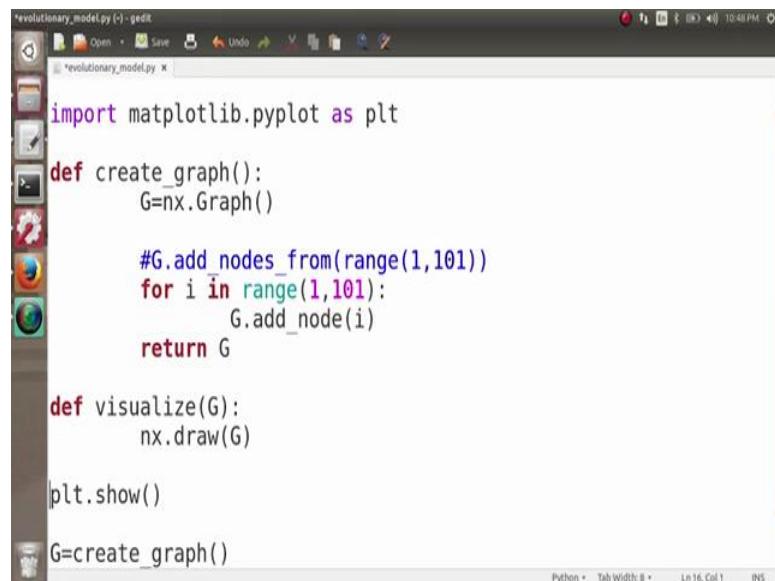
(Refer Slide Time: 02:32)



So, here what we have done is we have made a list here and then passed it what we can do is have a four loop here. So, we can do for i in range 1 to 101 what do we do G.add\_node(i), and it creates the same output as before. So, when we run this file and we do python evolutionary underscore model.py we get the same graph perfect.

Next what we want to do is we want to see some random BMIs to these people. So, for assigning random BMIs to these people what we will do is. So, we will be first before doing that we want to use a good use of we want to make a good use of functions here. So, whatever will be doing will keep writing it everything in functions only. So, this particular piece of code G = nx.Graph to add\_node(i) was for creating our graph.

(Refer Slide Time: 03:44)



```
evolutionary_model.py (~)-edit
evolutionary_model.py x
import matplotlib.pyplot as plt

def create_graph():
    G=nx.Graph()

    #G.add_nodes_from(range(1,101))
    for i in range(1,101):
        G.add_node(i)
    return G

def visualize(G):
    nx.draw(G)

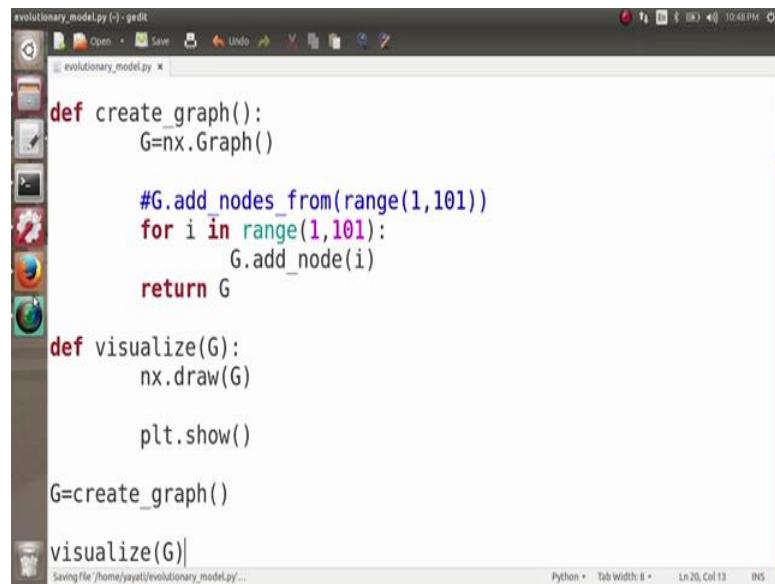
plt.show()

G=create_graph()
```

So, what we do here is we simply pass a function here G = create\_graph and this entire thing we shift under the function create underscore graph. So, we have this function here - define create underscore graph and what this function does is it creates a graph having 100 nodes and it returns the graph G.

And these two statements nx.draw(G) and plt.show we are using it to visualize our graph. So, what do we do is we define another function here define and then we name this function as visualize, define visualize G what it does is it will draw or graph and showing?

(Refer Slide Time: 04:39)



```
evolutionary_model.py (~) - gedit
evolutionary_model.py x

def create_graph():
    G=nx.Graph()

    #G.add_nodes_from(range(1,101))
    for i in range(1,101):
        G.add_node(i)
    return G

def visualize(G):
    nx.draw(G)

    plt.show()

G=create_graph()

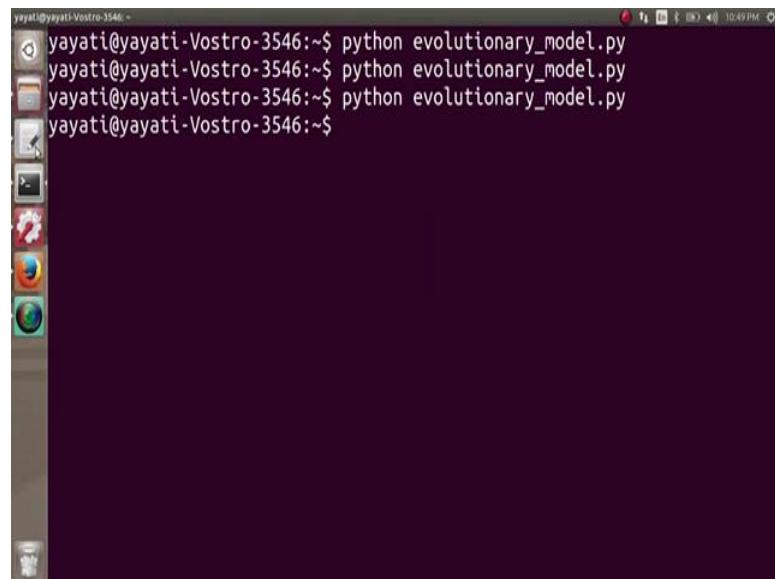
visualize(G)|
```

Saving file /home/yayati/evolutionary\_model.py...

Python • Tab Width: 8 • Ln 20, Col 13 INS

So, we pass both of these statements inside this function and whenever we have to see this graphic simply called visualize G. So, let us run it and see quickly.

(Refer Slide Time: 04:47)



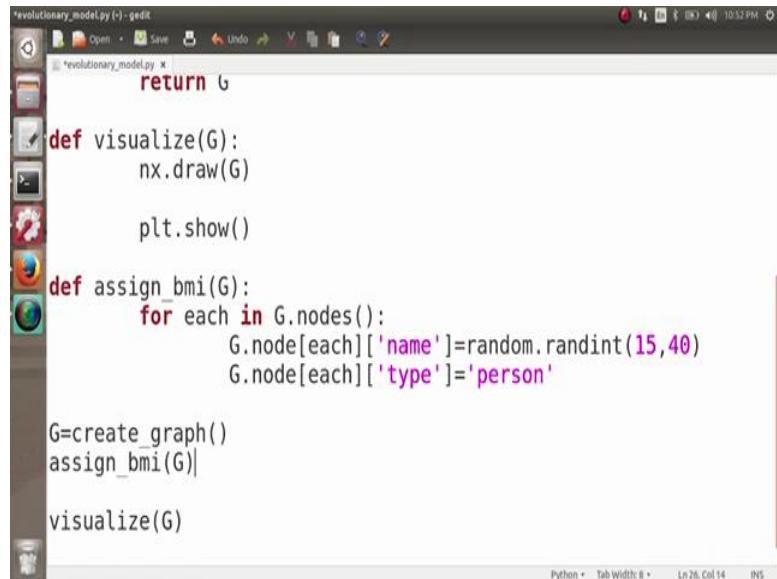
```
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$
```

So, we get here the output which we want it. So, till now everything is perfect. So, we have a graph having 100 nodes.

Next what we want to do is we want to assign a body mass index to every person here in this graph. So, here I have 100 people and you want to assign a BMI to every person. So,

for that again we take the help of a function and we call the function assign BMI and we pass the graph G in this function.

(Refer Slide Time: 05:16)



```
evolutionary_model.py (~) - gedit
return G

def visualize(G):
    nx.draw(G)
    plt.show()

def assign_bmi(G):
    for each in G.nodes():
        G.node[each]['name']=random.randint(15,40)
        G.node[each]['type']='person'

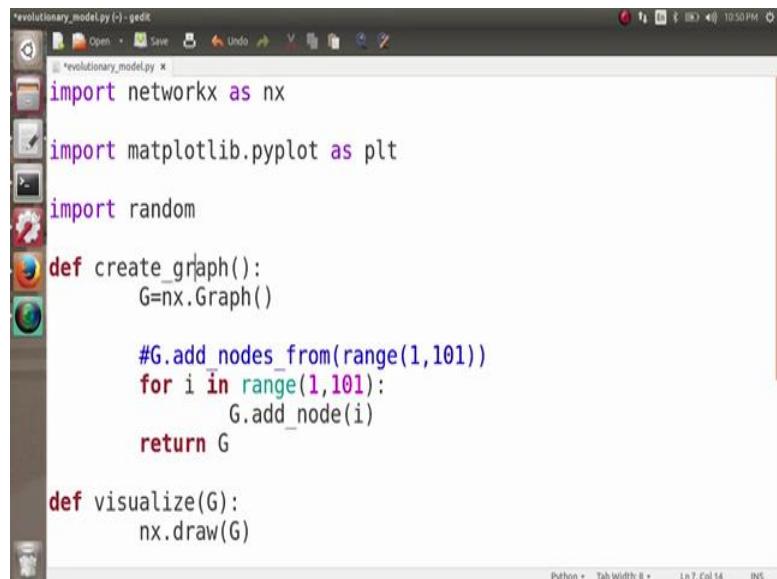
G=create_graph()
assign_bmi(G)

visualize(G)

Python+ Tab Width: 8+ Ln 26, Col 14 INS
```

So, let us write down the body of this function, we have define, assign, underscore BMI and we have a graph G here. How do we assign a BMI, so we know that BMI is going to be a number from 15 to 40? So, what we are going to do is for every node in the network we generate a random number from 15 to 40 and assign that number as the BMI of this node.

(Refer Slide Time: 05:56)



```
evolutionary_model.py (~) - gedit
import networkx as nx

import matplotlib.pyplot as plt

import random

def create_graph():
    G=nx.Graph()

    #G.add_nodes_from(range(1,101))
    for i in range(1,101):
        G.add_node(i)
    return G

def visualize(G):
    nx.draw(G)

Python+ Tab Width: 8+ Ln 7, Col 14 INS
```

So, since we need a random number, we need derive package random here. So, we import our package random here, and we come back to our function assign underscore BMI. How do we assign a BMI to our node? So, probably you will remember we have used the attributes of the node, we have learnt about the attributes of the node. So, we are going to make the use of the attributes of the nodes. So, how do we do it for each in G dot nodes? So, we have an iterator which iterates you every node in G. So, for each in G dot nodes what do we do is we add an attribute G.node each and let us have the name of this attribute as name. So, better option was having the BMI here, but we are using the name of the attribute to be name itself have a reason why I am using the attribute name to be name here which you will understand as you see more of this code.

So, every person has here, and attribute and the attributes name are name. So, the attribute is name and the value of the attribute is random number, random integers from 15 to 40 what I do is random.randint and it should be from 15 to 40. In addition to an attribute name I want one more attribute associated with each node here and you guess what that attribute is going to be. I told you that we are going to have two types of nodes in this network. So, one type of nodes is corresponding to the people in the network and another type of node is corresponding to the social for kind. So, will need to distinguish between these two kinds of nodes, so hence I take an attribute here G dot node each and I name this attribute as type.

So, G dot node each type equals to and these are 100 people. So, I name this as I keep the value as person. So, for each of these 100 nodes there type is equal to person. So, we have assigned a BMI to every node and let us put this assign BMI function before visualizing the graph.

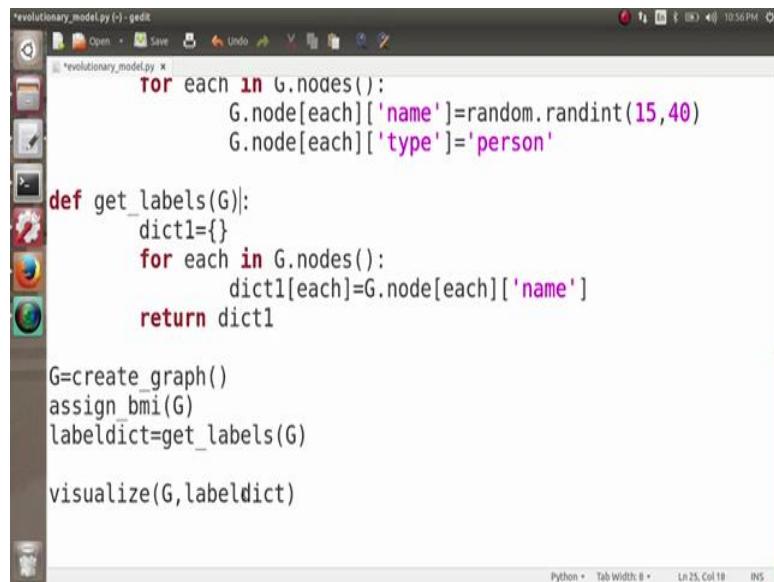
(Refer Slide Time: 08:31)



And let us run the code and see it now run the code and we see it so, but the graph looks like the previous graph only so every node here has a BMI, but we do not see any BMI here why, because currently our nodes are having no labels. So, we need to label our nodes and this particular labeling which we want is the BMI. So, every node should be labeled with its BMI. Every node should have a number written on it and that number should be the BMI of this node.

So, how do we do that? After assigning BMI to these nodes, we want these as labels. So, mainly we need a dictionary for all the labels which can be displayed.

(Refer Slide Time: 09:20)



```
evolutionary_model.py (~) - gedit
evolutionary_model.py x
for each in G.nodes():
    G.node[each]['name']=random.randint(15,40)
    G.node[each]['type']='person'

def get_labels(G):
    dict1={}
    for each in G.nodes():
        dict1[each]=G.node[each]['name']
    return dict1

G=create_graph()
assign_bmi(G)
labeldict=get_labels(G)

visualize(G,labeldict)

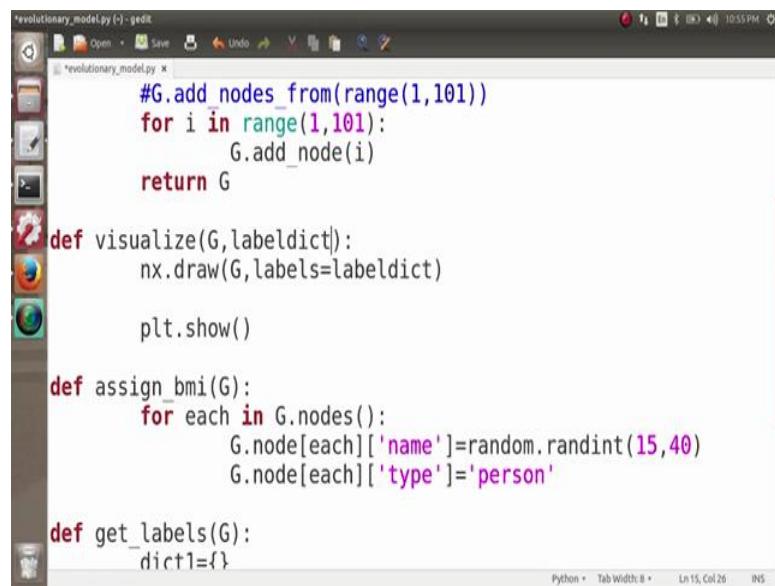
Python + Tab Width: 8 + Ln 25, Col 18 INS
```

We want the dictionary here let us name this dictionary as labeldict and to get this dictionary labeldict we use a function get labels from G and this is not an implicit function get labels we need to define these function. So, what do we do? We define this function get underscore labels G and what this function is going to do first of all we create a dictionary we have a dictionary here let us name it as dict to 1 and what will add to this dictionary is the label for every node.

Hence for each in G dot nodes what we are going to do is dict1 and the value in the value for each, so each is the key here, dict to 1 of each. So, each is am node here dict1 of each should be equal to, what it should be equal to the BMI of this person, the BMI of each. So, how to access the BMI of each BMI is are attribute we do G.node each under BMI was stored in the attribute named name and at the end we return our dictionary dict to 1.

So, here you see we get a dictionary labeled dict equals to get underscore labels G and this dictionary has all the labels. Now to see these labels in our graph we look at this function visualize G, we go above will see this function visualized G. So, it was having nx dot draw G we need to add a small code here, we want the labels with the each node.

(Refer Slide Time: 11:16)



```
#G.add_nodes_from(range(1,101))
for i in range(1,101):
    G.add_node(i)
return G

def visualize(G,labeldict):
    nx.draw(G,labels=labeldict)
    plt.show()

def assign_bmi(G):
    for each in G.nodes():
        G.node[each]['name']=random.randint(15,40)
        G.node[each]['type']='person'

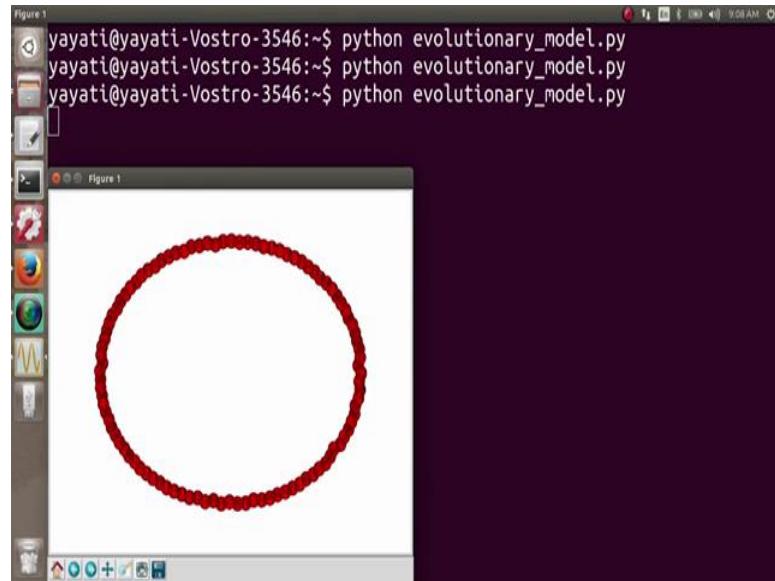
def get_labels(G):
    dict1={}
```

So, we add labels = and what should be the labels = a dictionary and what was the dictionary label dict. Do you see a problem here? I will see a problem here, we are using labels = labeldict how does my code node what is, does my code node what is label dict. So, you need to pass labeldict as an argument here. So, we pass this labeldict as an argument here and similarly here while visualizing the graph G we need to pass labeldict here. So, I hope that the code is clear do you.

So, I will just quickly recap what did we do is after assigning the BMI we created a dictionary the name of the dictionary is labeldict and what is the value what is this dictionary carrying is this dictionary is having one key for each of the nodes in the network and the value corresponding to that key, corresponding to that node is the BMI for that node and then we written this dictionary dict1 and its collected in label dict. Now we have a dictionary for all the labels in the graph to do visualize the graph we pass this labeled dict and here by visualizing we have an extra parameter here labels which is equal to label dict.

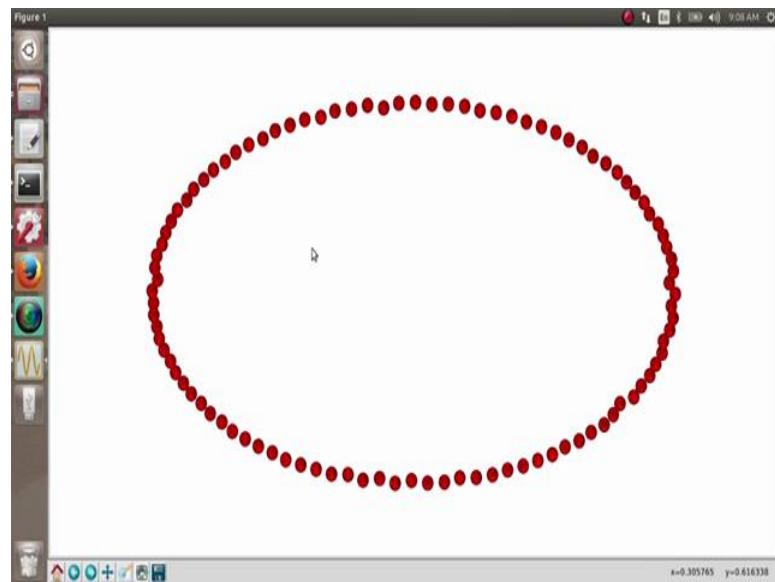
So, now we execute this code and see our graph. So, I execute our code and you see the graph here.

(Refer Slide Time: 12:42)



So, when we look here at this graph you can see that every node is marked with the label and the label is its BMI.

(Refer Slide Time: 12:51)



So, these are 100 nodes if you look at the labels for all these nodes the label the range from BMI that is from 15 to 40 and then you can also see that there are two nodes it is having a label 31 and this is also label 31. So, these two nodes they are having the same BMI. So, here we get the graph where we get different people and every person is labeled with his or her own BMI.

(Refer Slide Time: 13:29)

```
revolutionary_model.py (~) - edit
evolutionary_model.py
dict1={}
for each in G.nodes():
    dict1[each]=G.node[each]['name']
return dict1

def get_size(G):
    array1=[]
    for each in G.nodes():
        array1.append(G.node[each]['name'])
    return array1

G=create_graph()
assign_bmi(G)
labeldict=get_labels(G)
nodesize=get_size(G)

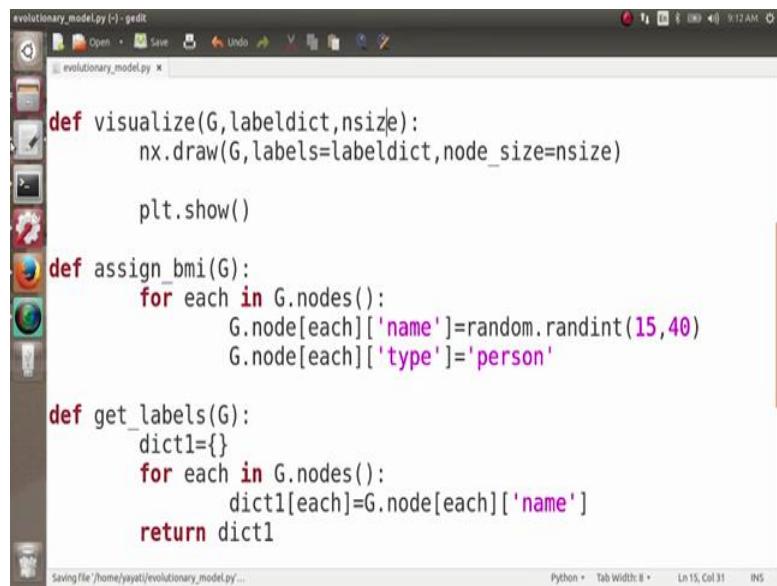
visualize(G,labeldict,nodesize)
```

Next what am I interested in, what I want in the graph. Suppose that we want to see this difference in the form of the node size that is I want a person having a higher BMI to appear larger as compared to a person who is having a lower BMI. So, what do we do for that is I want to assign a size to every node and the size of the node should be equal to or proportional to the size of a node should be proportional to its BMI? So, what do we do for that is, I want an array here and this array consists of, this array consists of these sizes of different nodes? So, I take an array here node size and the value of this node size is to get the value I call a function get underscore size G.

And again, this function gets underscore size we have to define. So, we define this function here - define get underscore size G and what this function should output is an array of the sizes of different nodes whatever size we want to keep in the graph. I create an array here let us say array one and then for each in G.nodes. So, I put an iterator for all the nodes in the graph for each in G dot nodes what do we do? We do array one dot append and what do we append in this array in this array we append the BMI value extract that node and as we know how do we extract the BMI value of a node is G dot node each and then we have here name. This gives us the BMI value of each node and at the end we return array1. So, we get here an array node size which has the size of different nodes.

We want to see it in the graph. So, to see it in the graph we first of all pass it in the function visualize, so in the function visualize we also pass here the array node size. And then when we are visualizing the graph what I do here is I include a new parameter which says node underscore size. So, and the value for every node is to be extracted from the array node size and also, we need to pass this array here. So, let me pass node size here.

(Refer Slide Time: 15:42)



The screenshot shows a Gedit text editor window with the file 'evolutionary\_model.py' open. The code defines three functions: 'visualize', 'assign\_bmi', and 'get\_labels'. The 'visualize' function takes a graph G, a label dictionary labeldict, and a node size nsize. It uses nx.draw to draw the graph with labels from labeldict and node sizes from nsize. The 'assign\_bmi' function takes a graph G and assigns random BMI values (between 15 and 40) and 'person' type to each node. The 'get\_labels' function takes a graph G and returns a dictionary where each node's name is its key and its name value is its value. The code is as follows:

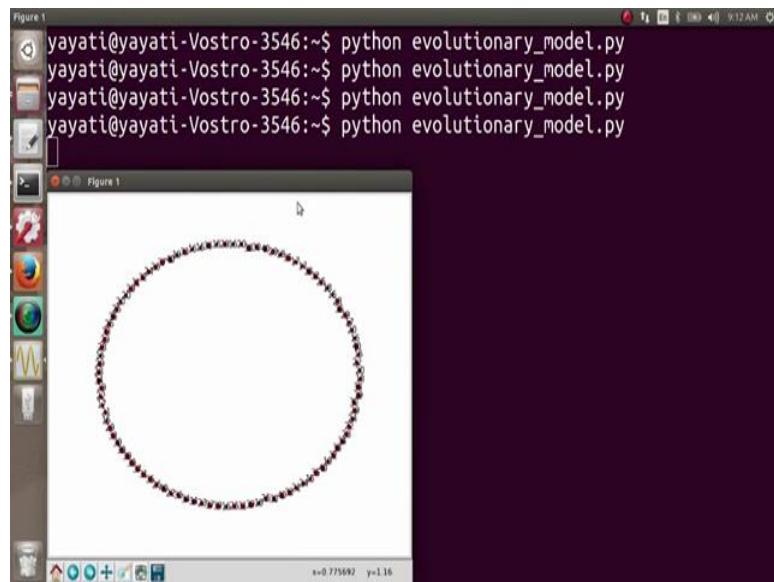
```
def visualize(G,labeldict,nszie):
    nx.draw(G,labels=labeldict,node_size=nszie)
    plt.show()

def assign_bmi(G):
    for each in G.nodes():
        G.node[each]['name']=random.randint(15,40)
        G.node[each]['type']='person'

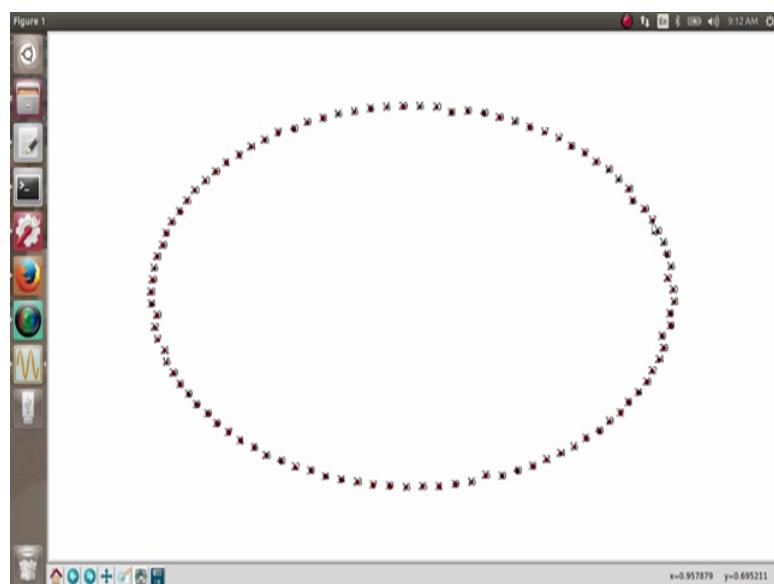
def get_labels(G):
    dict1={}
    for each in G.nodes():
        dict1[each]=G.node[each]['name']
    return dict1
```

You can also give different names for like we pass there the argument node size. So, here I can actually keep it nszie and at the end I pass here n size. Just doing it this way for the sake of clarity, we can also have, we can keep the same name for the argument and we parameters and we can also give different names. Let us go back execute this code and see.

(Refer Slide Time: 16:28)

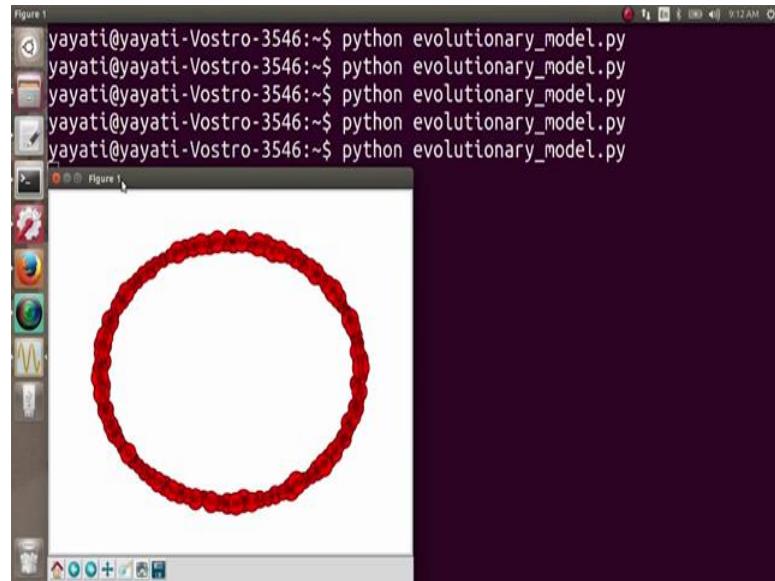


(Refer Slide Time: 16:33)



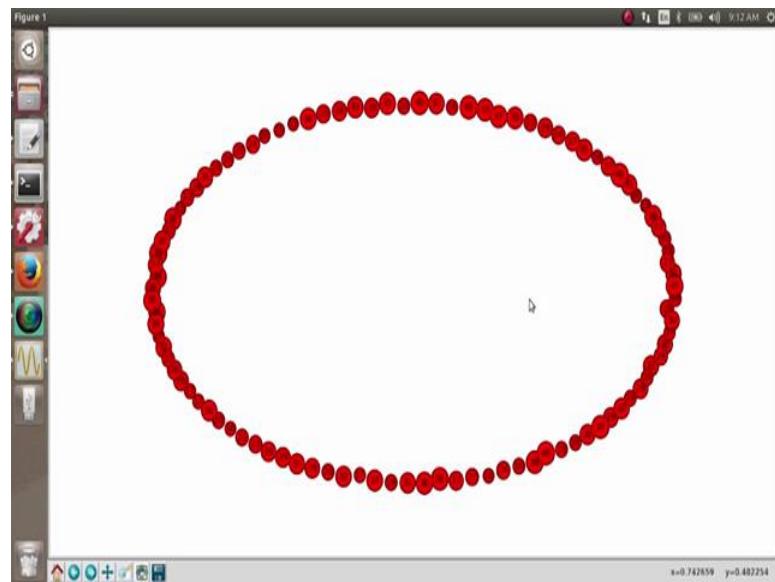
Let us execute this code and what we see here is the size of the nodes have changed, but we are not being able to. So, you can see here that some nodes are of smaller size some nodes are of larger size, but still the difference is not very clear the graph does not look very nice. So, what I do is I go back to the code and for each value way wherever we are collecting the size here. So, for each value what I do? I multiply it by 20. So, I scale the value for the size of every node by 20. So, that the graph looks cleaner.

(Refer Slide Time: 17:10)



And then we go back and then we execute this code. So, you can see here this looks more beautiful from the previous graph.

(Refer Slide Time: 17:14)

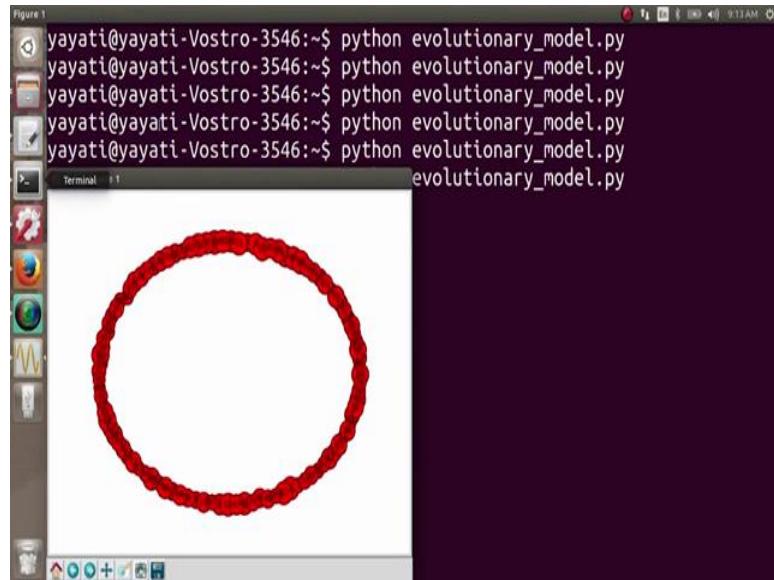


So, you can say here the nodes which are having a high BMI such as 40 they appear in the biggest; they appeared biggest in size and the nodes which are having a lower BMI such as 15 they appear smaller in size.

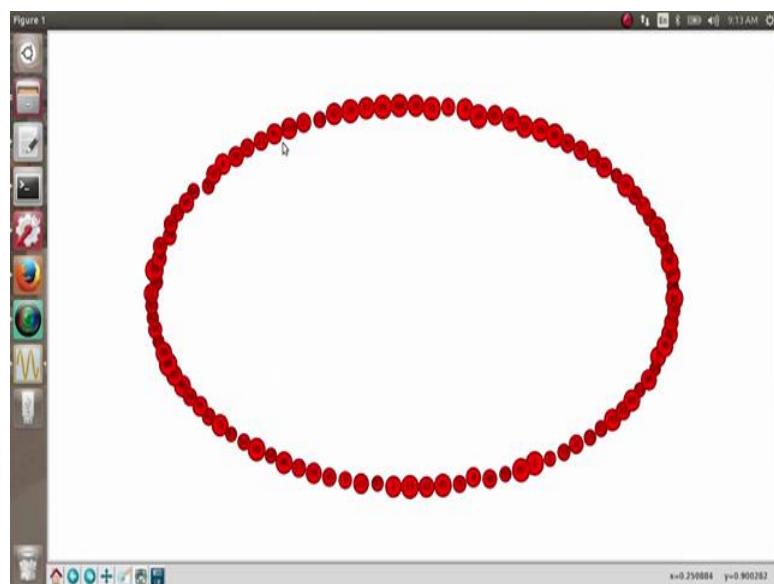
So, till now we have just created a network on 100 people where every person has a different BMI and the size of every person is proportional to his or her BMI when we

visualize this network. So, for the time being what I want to do is while visualizing my graph I will now remove the labels. So, these are the labels which shows the BMI in our graph, I will just I will remove the labels.

(Refer Slide Time: 18:04)



(Refer Slide Time: 18:07)

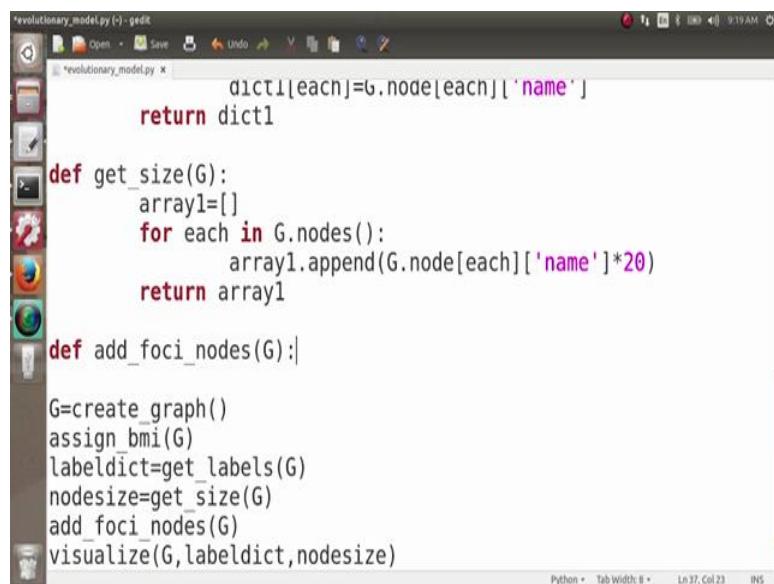


So, here we say that the node 10. So, can see that the node 10 is of the is having a very high BMI, node 87, node 13 these are all having very high BMI and the nodes like 83, 64, 66 these are having smaller BMI.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture - 48**  
**Strong and Weak Relationships (Continued) and Homophily**  
**Fatman Evolutionary Model - The Base Code (Adding Social Foci)**

(Refer Slide Time: 00:05)



```
evolutionary_model.py (~)-gedit
dict1[each]=G.node[each]['name']
return dict1

def get_size(G):
    array1=[]
    for each in G.nodes():
        array1.append(G.node[each]['name']*20)
    return array1

def add_foci_nodes(G):

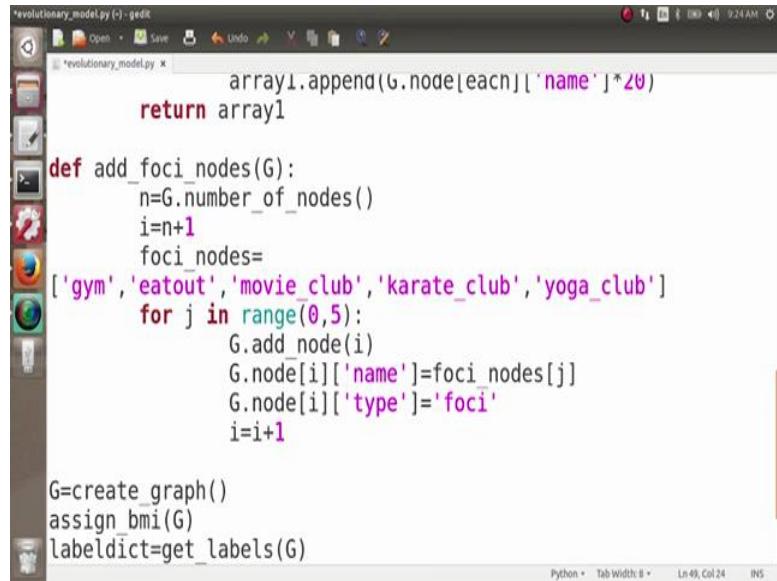
    G=create_graph()
    assign_bmi(G)
    labeldict=get_labels(G)
    nodesize=get_size(G)
    add_foci_nodes(G)
    visualize(G,labeldict,nodesize)

Python + Tab Width: 8 + Ln 37, Col 23 8NS
```

Next, what we want to do is; we want to add foci\_nodes here. So, we have now all the people node here, we want different-different foci like the gym, eat out, movies, yoga etcetera. So, let us see; how do we add foci\_nodes here, so as before we are writing a function for everything. So, I create a new function here and I name the function as at foci\_nodes and I pass my graph G here. So, in my graph G, I have to add foci\_nodes, I define here; define add foci\_nodes and I pass the function G here. Now, what I have to do is as discussed before we have 5 foci, so we have to create 5 new nodes in this graph. There, the simplest way to do is we know that the nodes in our graph till now, are numbered from 1 to 100.

So, we create 5 nodes; 101, 102, 103, 104 and 105, but I do not want to do it this way, what if my graph was having 200 nodes or 300 nodes. I might not know; what is the number of nodes in my graph and how should I proceed.

(Refer Slide Time: 01:18)



```
evolutionary_model.py (~) - edit
array1.append(G.node[each]['name']*20)
return array1

def add_foci_nodes(G):
    n=G.number_of_nodes()
    i=n+1
    foci_nodes=
    ['gym','eatout','movie_club','karate_club','yoga_club']
    for j in range(0,5):
        G.add_node(i)
        G.node[i]['name']=foci_nodes[j]
        G.node[i]['type']='foci'
        i=i+1

G=create_graph()
assign_bmi(G)
labeldict=get_labels(G)

Python + Tab Width: 8 Ln 49, Col 24 INS
```

So, what will we do it; we will get an iterator, let us say  $i$  or let us say a number  $n$ . So what is  $n$ ?  $N$  is  $G.number\_of\_nodes$ . So, it returns as the number of nodes in the graph, so the number of nodes was 100. So, enhance this value 100 unit and now we want the new node to get started from 101, it means that we have to start adding nodes from the number  $n$  plus 1.

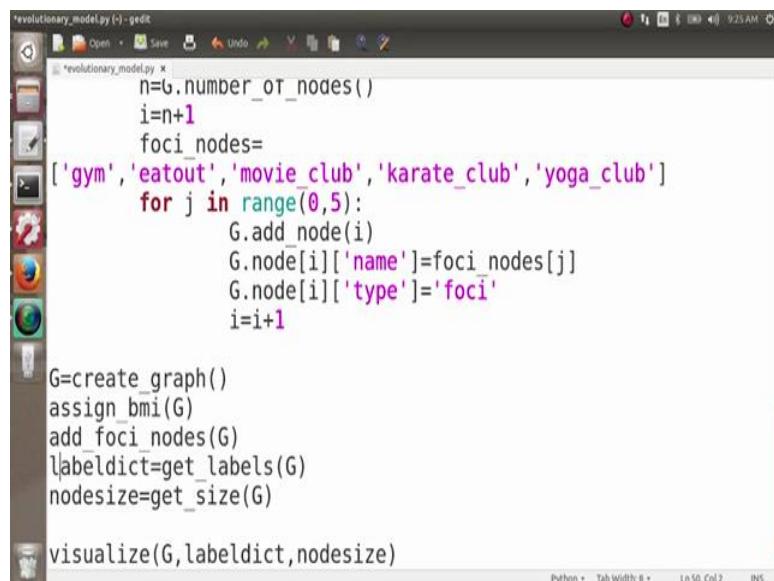
Now, you might want to note here; here the numbering of our nodes, it started from 1 to 100. If the numbering of the nodes was from 0 to 100; sorry was from 0 to 99, so still these were the 100 nodes, but the numbering and deduct 99. So, there we would have to start adding the nodes from the number  $n$  itself because  $n$  is 100 but provided now, we have the nodes from number 1 to 100. We have to start creating the new nodes from the number  $n + 1$ ; so, I take the value of  $i = n + 1$ .

Next, I create an array for all of my  $foci\_nodes$ ; so I say  $foci\_nodes$  equals to and this is an array and what all values this array has, it has a gym and then it has eat out and then it has a movie club and then it has a karate club and it also has a yoga club, so you have all these value in our  $foci\_nodes$ . Now, I want to add 5 nodes in my graph; what we can do is four let us take a new variable;  $j$  in range 0 to 5, so it goes from 0 to 4. So, for  $j$  in range 0 to 5; what we have to do is,  $G$  dot  $add\_node$  and what should be the index of this node, index of this node should be  $i$  and then we increment  $i$  with  $i + 1$ . So, these 2 lines 0 to add node  $i$  and  $i = i + 1$ ; add a new node in the graph.

Now, this new node as before should have 2 properties; the first property is the name which is the attribute of this node, which will tell you which of these 5 social foci it is. So, you see here now; so, to maintain a consistency in the person node also we have taken this attribute to be named. So when you look at a person, the main feature of a person is; his BMI which we depict as name and if you look at a social foci node its main feature is what so foci node it is; that is name. So, that is why we have used the attribute name in both the cases, so we need here 2 attributes; one is the name, and another is the type; so, that we can distinguish it from the person node.

So, what we do is; G.node[i] and what should be its name; its name should be chosen from our array. So, its name should be foci\_nodes and we pass the value j here because j ranges from 0 to 4. So, it gets a name here and then we have to add a type; we do G.node[i] and then we have a type and this type should be equal to foci node, that it tell us that it is a foci node.

(Refer Slide Time: 05:57)



```

revolutionary_model.py -> gedit
*revolutionary_model.py
n=G.number_of_nodes()
i=n+1
foci_nodes=
['gym', 'eatout', 'movie_club', 'karate_club', 'yoga_club']
for j in range(0,5):
    G.add_node(i)
    G.nodes[i]['name']=foci_nodes[j]
    G.nodes[i]['type']='foci'
    i=i+1

G=create_graph()
assign_bmi(G)
add_foci_nodes(G)
labeldict=get_labels(G)
nodesize=get_size(G)

visualize(G,labeldict,nodesize)

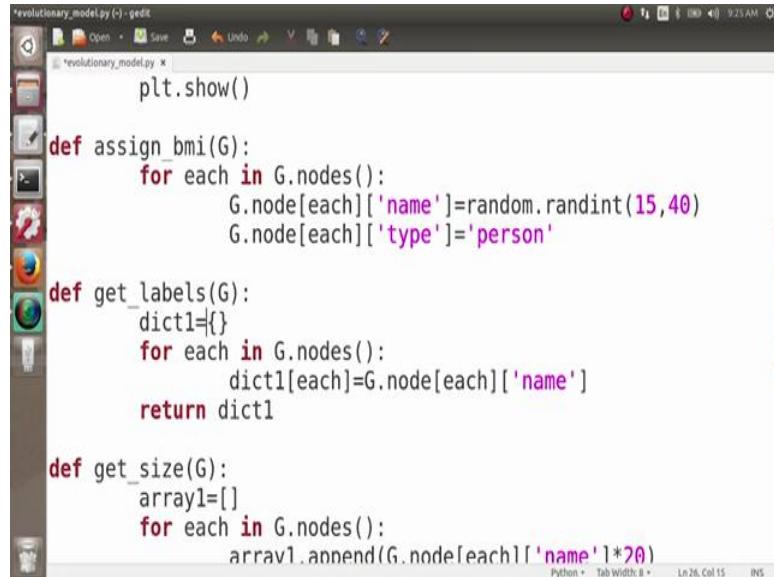
```

So, we have added 5 foci\_nodes to our graph G and then for visualizing it as you see that; so, one thing we have to take care where we have to keep this function call is should be after assigning BMI.

So, once we assign BMIs to the people then we add the foci\_nodes, then next we get the labels and as we know labels for every node was its attribute name; so, we need not do any change here. Another was this function get under score size, so here we will have to

do a little bit of modification. So, if you see at this function get labels; we do not need any modification here.

(Refer Slide Time: 06:31)



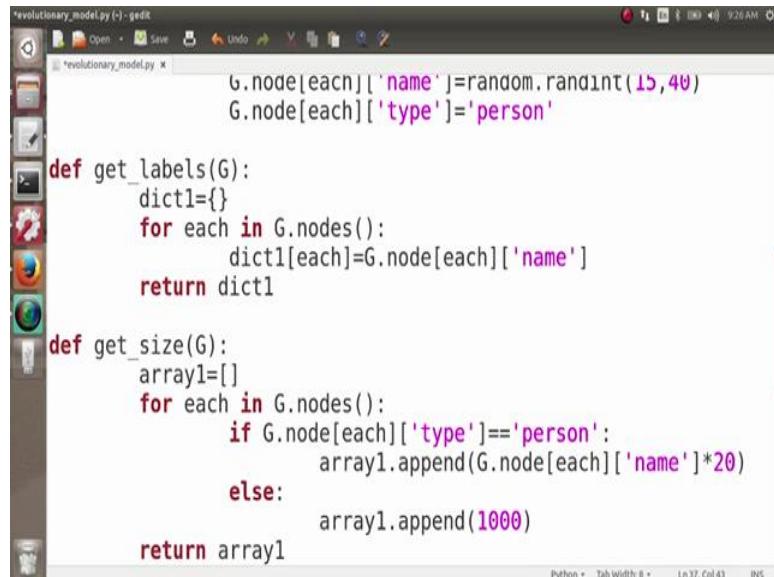
```
revolutionary_model.py (~) - gedit
plt.show()

def assign_bmi(G):
    for each in G.nodes():
        G.node[each]['name']=random.randint(15,40)
        G.node[each]['type']='person'

def get_labels(G):
    dict1={}
    for each in G.nodes():
        dict1[each]=G.node[each]['name']
    return dict1

def get_size(G):
    array1=[]
    for each in G.nodes():
        array1.append(G.node[each]['name']*20)
```

(Refer Slide Time: 06:38)



```
revolutionary_model.py (~) - gedit
G.node[each]['name']=random.randint(15,40)
G.node[each]['type']='person'

def get_labels(G):
    dict1={}
    for each in G.nodes():
        dict1[each]=G.node[each]['name']
    return dict1

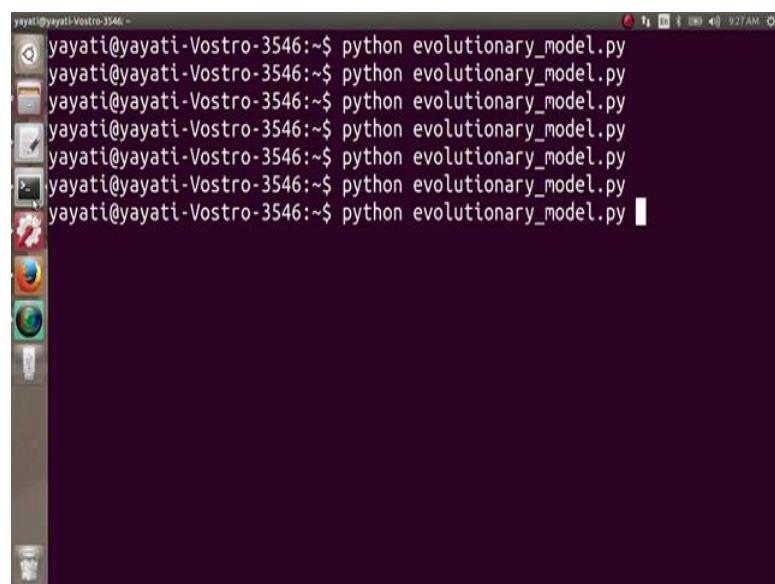
def get_size(G):
    array1=[]
    for each in G.nodes():
        if G.node[each]['type']=='person':
            array1.append(G.node[each]['name']*20)
        else:
            array1.append(1000)
    return array1
```

Because it is simply returning the name of every node, which is well defined for a person node as well as for a foci node, but if we look at these get size here. So, as we know that in our graph now, if we look at this name attribute for nodes. So, for the people node this name attribute returns the BMI, which is a number.

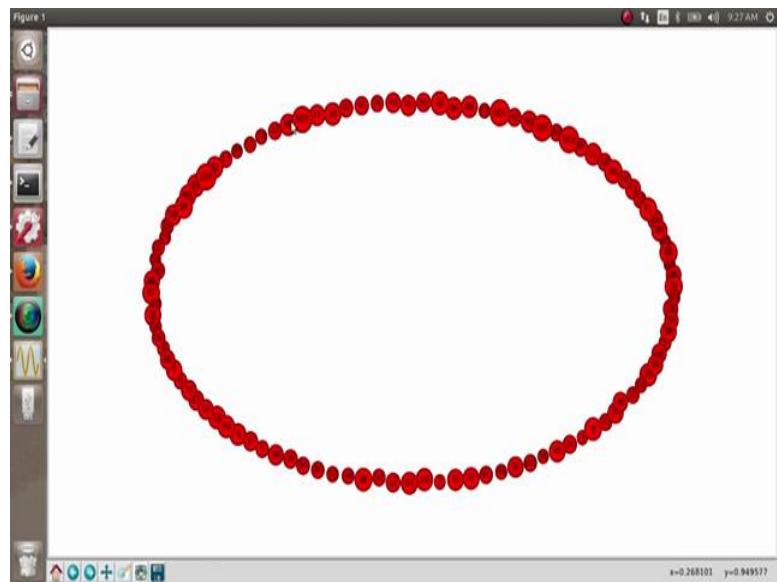
But, if we look at the social foci node; its name attribute is a string and multiplying the string by a 20 or using it in the array of size makes no sense. So, what we will do it; for each in G.nodes, if G dot each; if G.node[each]['type']. So, if its type == person; then what we do is, we go ahead and have this value, but else if its type is not of person; it is a foci node. Let us have some well-defined value fix here, so what we append here; array 1.append and let us append a value 1000 here.

Now, if it is a person node; the size of the node is proportional to its BMI and if it is a foci node, we have a fixed size of that node which is 1000 and as before we then visualize this graph.

(Refer Slide Time: 08:02)



(Refer Slide Time: 08:05)



Let us execute this code, so we see this graph here; so, what is the problem here? We see all the nodes, but we do not see the names. So, what we had done previously is; we have to use this labeldict, so we were getting the label for every node; we have also passed this label and by visualizing the graph; we have actually removed that statement from here.

(Refer Slide Time: 08:28)

```
evolutionary_model.py (~) - gedit
*evolutionary_model.py x
for i in range(1,101):
    G.add_node(i)
return G

def visualize(G,labeldict,nsize):
    nx.draw(G,labels=labeldict,node_size=nsize)
    plt.show()

def assign_bmi(G):
    for each in G.nodes():
        G.node[each]['name']=random.randint(15,40)
        G.node[each]['type']='person'

def get_labels(G):
    dict1={}
    for each in G.nodes():
```

A screenshot of a Python code editor window titled "evolutionary\_model.py (~) - gedit". The code in the editor is as follows:

```
for i in range(1,101):
    G.add_node(i)
return G

def visualize(G,labeldict,nsize):
    nx.draw(G,labels=labeldict,node_size=nsize)
    plt.show()

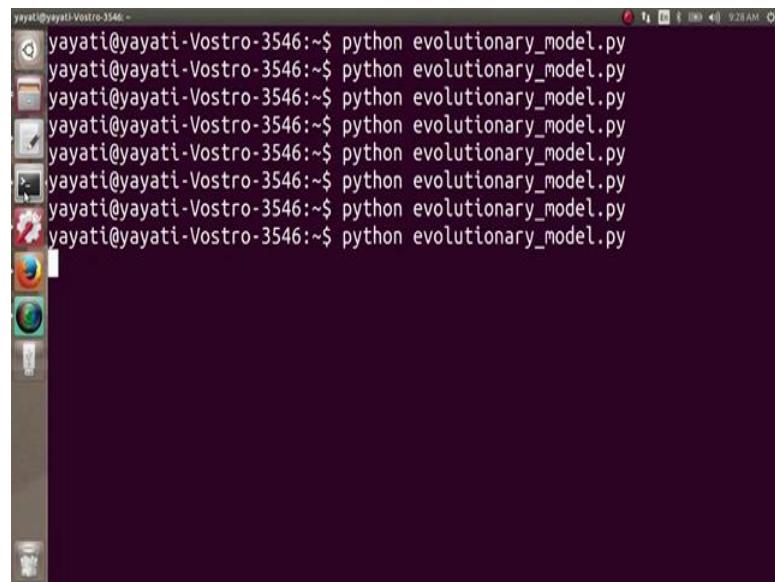
def assign_bmi(G):
    for each in G.nodes():
        G.node[each]['name']=random.randint(15,40)
        G.node[each]['type']='person'

def get_labels(G):
    dict1={}
    for each in G.nodes():
```

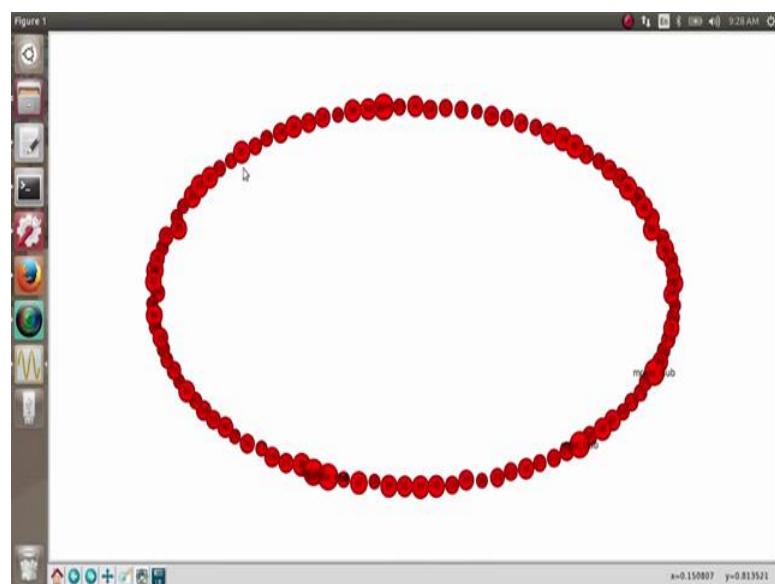
The code defines four functions: `get_labels`, `assign_bmi`, `visualize`, and `create_G`. The `visualize` function uses the `nx` and `plt` modules to draw the graph with labels from the `labeldict` and a specified node size. The `assign_bmi` function assigns random names and types ('person') to the nodes. The `get_labels` function creates a dictionary `dict1` to store node labels. The `create_G` function adds nodes from 1 to 101 to a graph `G`.

So, what we want is; we want the labels and the labels should be equal to labeldict so that we can see the labels for all the nodes, we want the labels; so, we keep = labeldict.

(Refer Slide Time: 08:47)

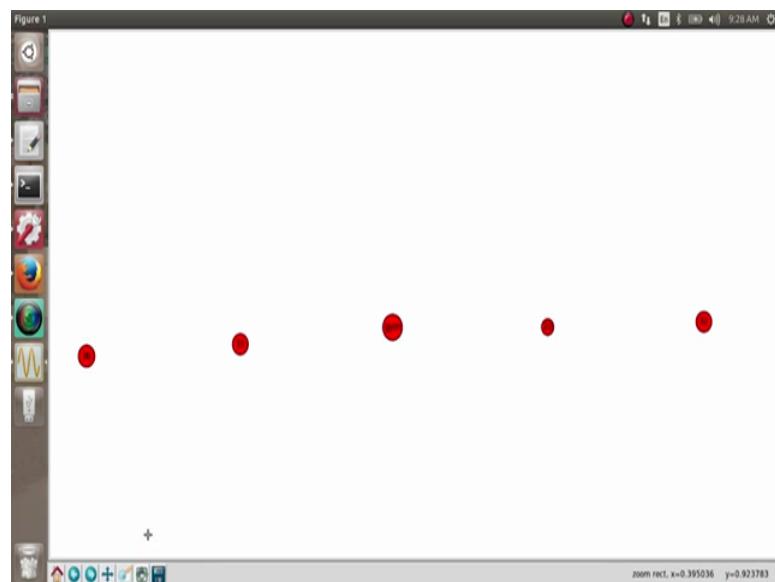


(Refer Slide Time: 08:52)

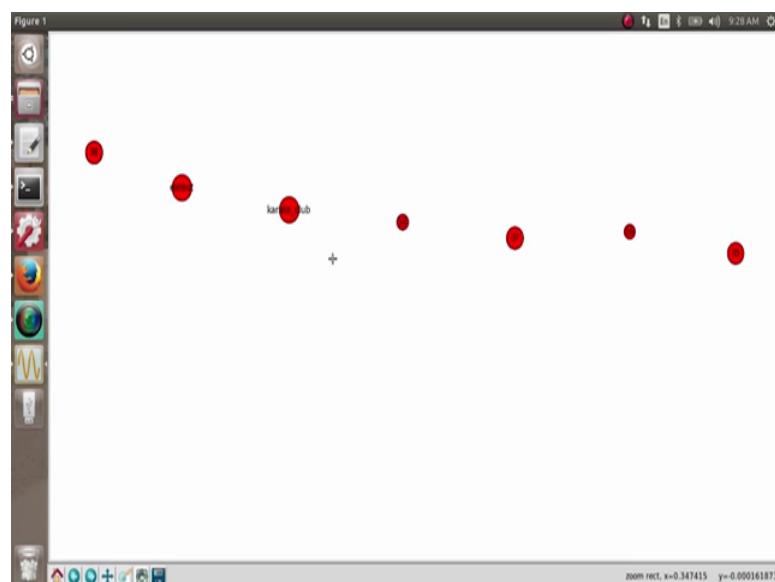


We go back then execute this code, now if we look at this graph; you can see that we have all different nodes labeled with their BMI, with their size, but here we have 5 social foci\_nodes as well.

(Refer Slide Time: 09:02)

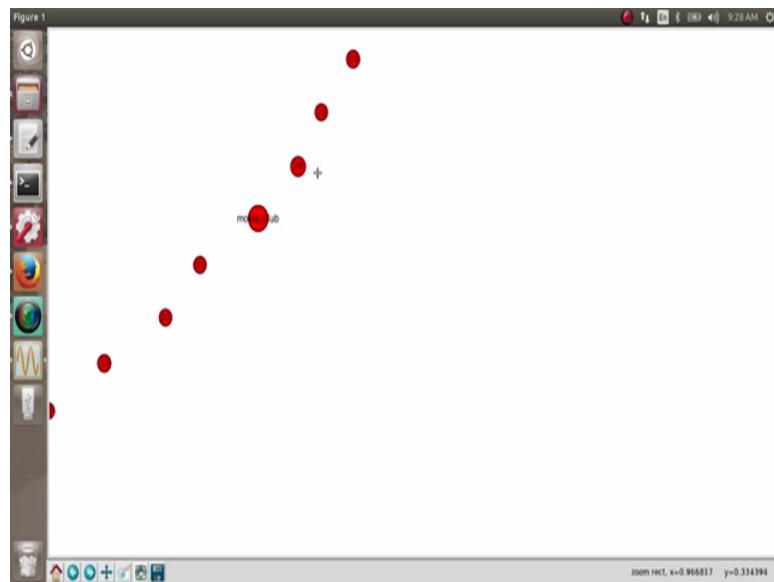


(Refer Slide Time: 09:07)



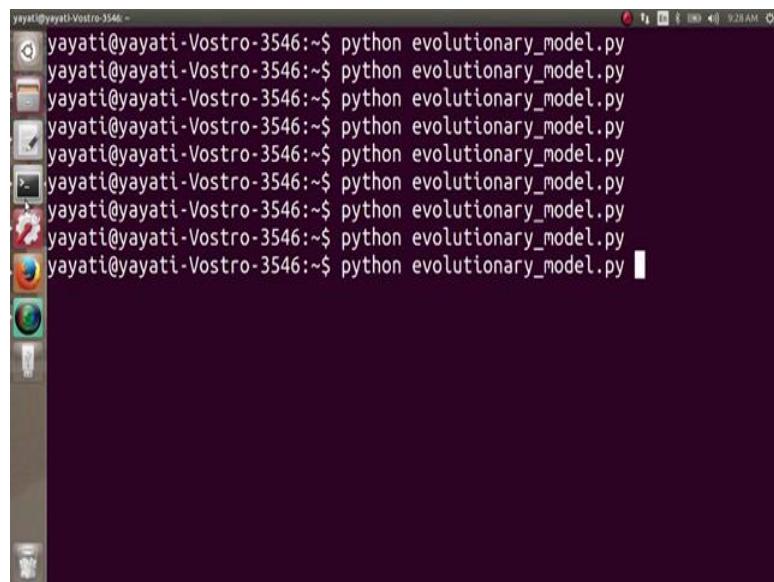
So, here we have gym node and then here we have eat out and we have karate club and here we have movie club.

(Refer Slide Time: 09:11)

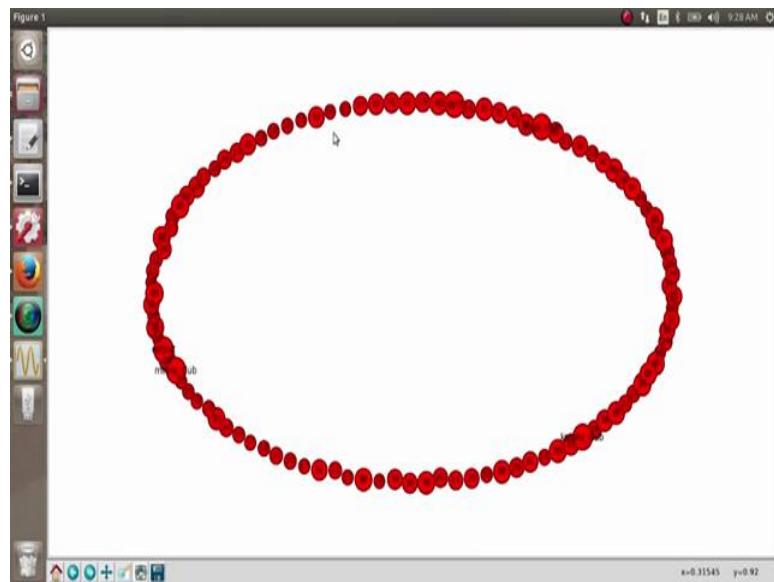


So, we have all the social foci\_nodes also here in this graph; so, we have 100 nodes for people and 5 nodes for social foci. What is more interesting to do next, so something which is coming next is all the way more interesting; what do I want?

(Refer Slide Time: 09:37)



(Refer Slide Time: 09:39)



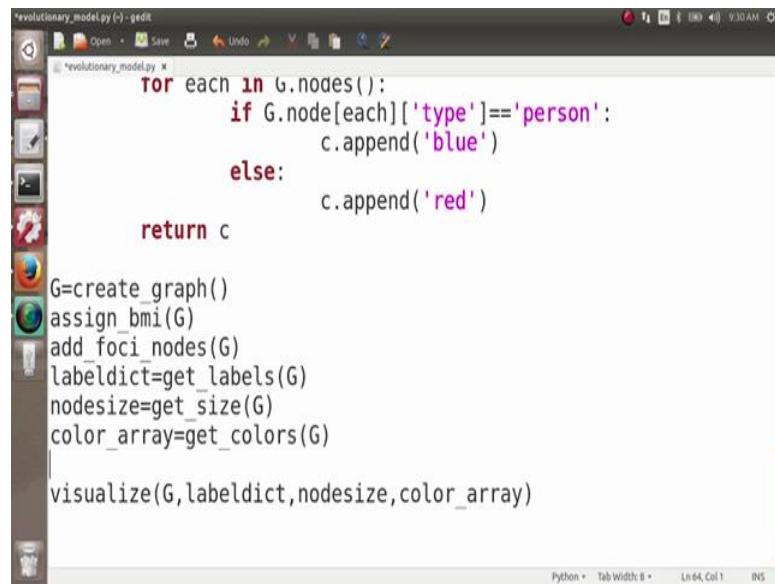
So you have seen that when we looked at this graph here; everything comes everything is in a red color. So, it makes difficult for me to identify the social foci\_nodes here; I have to search for them. Next, what we will do is; we write a piece of code, so that the social foci\_nodes appear in red color, but all the other nodes that is the people nodes; they appear in blue color and this is very simple. So, like we did for the labels and for the node size, we also do a similar thing.

(Refer Slide Time: 10:06)

```
revolutionary_model.py - gedit
  TOC1_nodes=
  ['gym', 'eatout', 'movie_club', 'karate_club', 'yoga_club']
  for j in range(0,5):
      G.add_node(i)
      G.nodes[i]['name']=foci_nodes[j]
      G.nodes[i]['type']='foci'
      i=i+1
  def get_colors(G):
      c=[]
      for each in G.nodes():
          if G.nodes[each]['type']=='person':
              c.append('blue')
          else:
              c.append('red')
      return c
  G=create_graph()
```

Here, we make an array; color array and get this array from the function get colors G and then we define this function get colors G here; define get colors G and what do we do here is, we have an array let us say c, then for each in G.nodes; if G.node each, type, so if its type == person, What do we do? We append a blue in this array else we append a red in this array and then we return this array.

(Refer Slide Time: 11:05)



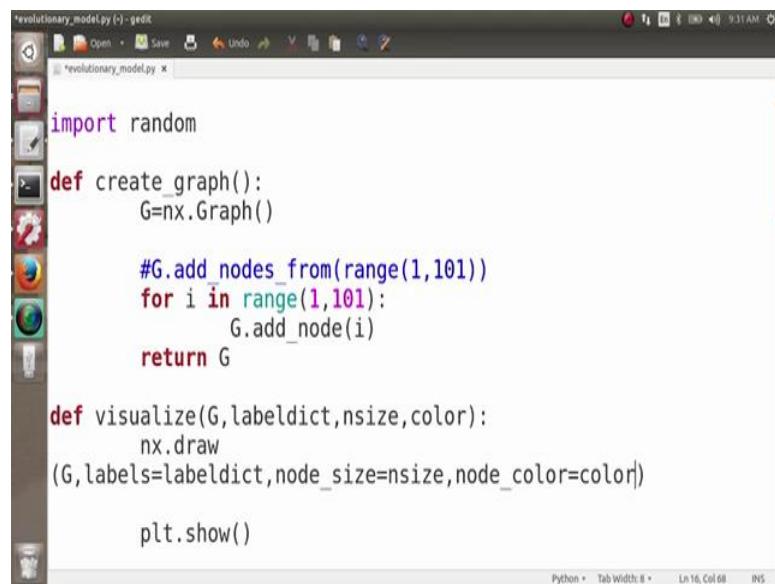
```
evolutionary_model.py (~) - gedit
*evolutionary_model.py *
for each in G.nodes():
    if G.node[each]['type']=='person':
        c.append('blue')
    else:
        c.append('red')
return c

G=create_graph()
assign_bmi(G)
add_foci_nodes(G)
labeldict=get_labels(G)
nodesize=get_size(G)
color_array=get_colors(G)
|
visualize(G,labeldict,nodesize,color_array)

Python + Tab Width: 8 + Ln 64, Col 1 0%
```

So, we get here color array, so this color array has a blue color if a node is a person and it has a red color, if the node is a social foci and what do we do here is we pass this color array here and the visualize function.

(Refer Slide Time: 11:27)



```
evolutionary_model.py (~) - gedit
import random

def create_graph():
    G=nx.Graph()

    #G.add_nodes_from(range(1,101))
    for i in range(1,101):
        G.add_node(i)
    return G

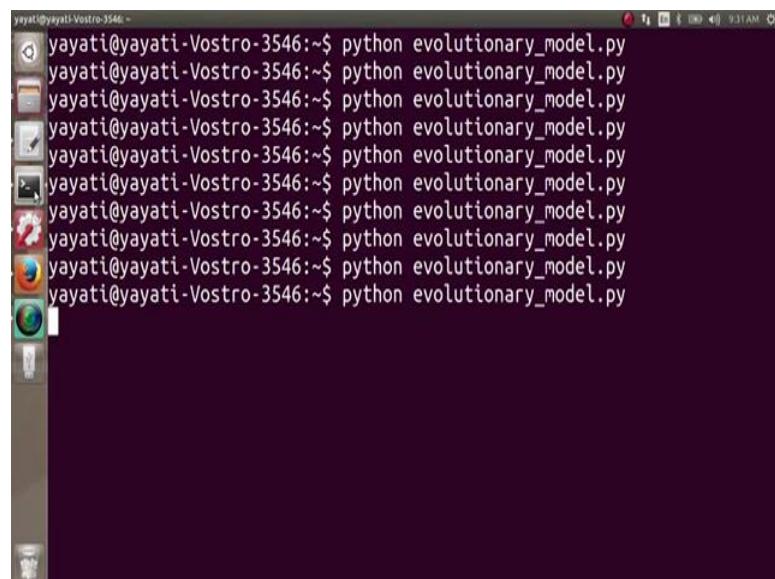
def visualize(G,labeldict,nsize,color):
    nx.draw
    (G,labels=labeldict,node_size=nsize,node_color=color)

    plt.show()

Python + Tab Width: 8 Ln 16 Col 68 INS
```

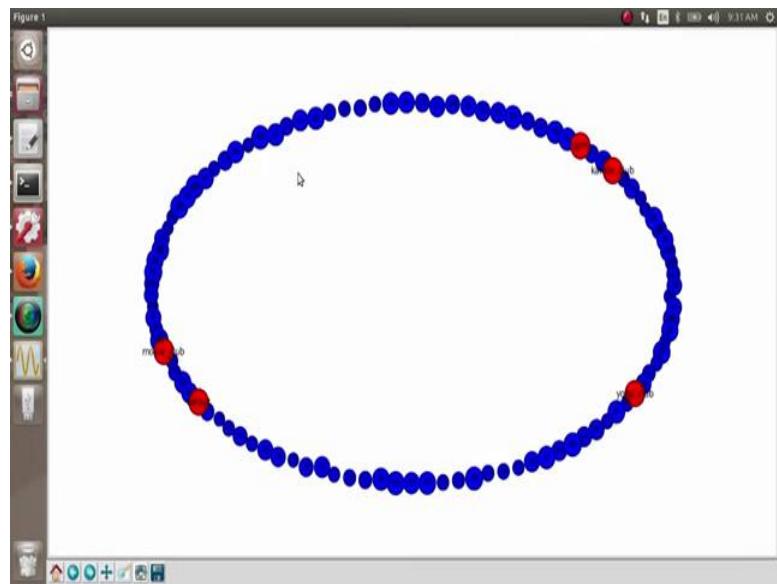
We go back to the definition of the visualize function; we have an extra array color here let us say color. So, what we do here is we add one more parameter which is node underscore color and from where every node takes its color is our array color, so we are done; go back.

(Refer Slide Time: 11:45)



```
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
```

(Refer Slide Time: 11:51)



Execute our code and you can see now; every social foci node comes in a red color and all the people node, it comes in a blue color; so far, so good. Now, we have a city and the city has 100 people and the city has 5 social foci. Next, what we want to do is in the beginning every person should be a part of exactly one social foci. So, now we must add edges between the people and the social foci.

(Refer Slide Time: 12:26)

```
revolutionary_model.py - gedit
革命性模型.py x
def categorize_nodes(G):
    c = []
    for each in G.nodes():
        if G.node[each]['type']=='person':
            c.append('blue')
        else:
            c.append('red')
    return c

def get_foci_nodes(G):
    f = []
    for each in G.nodes():
        if G.node[each]['type']=='foci':
            f.append(each)
    return f

def add_foci_edges(G):
    foci_nodes = get_foci_nodes(G)
    person_nodes = get_persons_nodes(G)
    for each in person_nodes:
        for foci in foci_nodes:
            G.add_edge(each, foci)

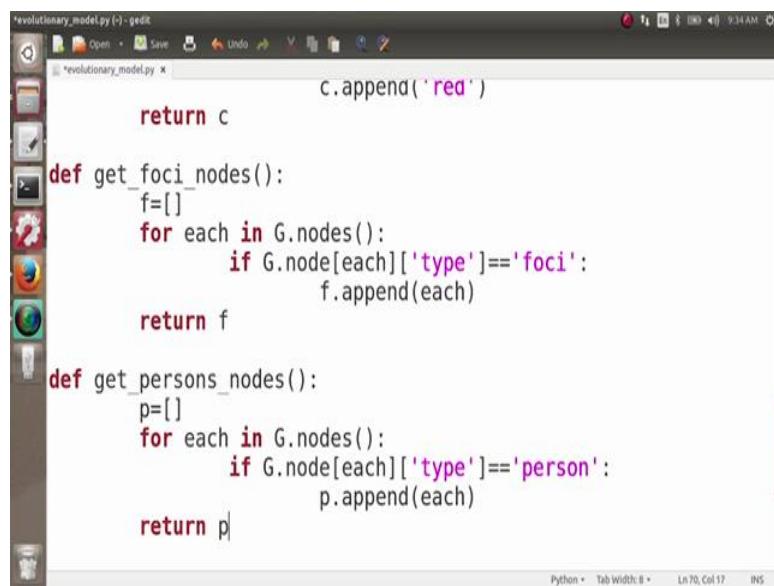
G = nx.Graph()
G.add_nodes(100, {'type': 'person'})
G.add_nodes(5, {'type': 'foci'})
```

So, again we write a function for that and let the name of the function be add\_foci\_edges. So, we want to add some edges from every person to a social foci; so,

we have here define add; foci edges and now what we want to do is, we chose one person one by one and from this person, we add an edge to every foci node. So, for that first of all we need all the nodes which are the foci\_nodes at one place, so that we know to which nodes we have to make connections to. Although, we know that the foci\_nodes are from 101 to 105, but that might not be the case with every graph given to you. We have made this graph this way, so let us try to get an array which consists of all the nodes, which are the foci\_nodes.

So, we make an array foci\_nodes equals to and then and let us make an array; people node. Rather, a better approach is to use functions for both of these; so, for foci\_nodes equals to get. So, the array foci\_nodes = get\_foci\_nodes and the array person\_nodes = get\_persons\_nodes and then we quickly define both of this functions here; define get\_foci\_nodes. So, what do we do is; we take an array f and then few of we want to make the foci\_nodes for each in G.nodes; if G.node each and if its type is equal to equal to what? If its type == 'foci', then what do we do?

(Refer Slide Time: 14:42)



```

evolutionary_model.py (?) - gedit
  File  Open  Save  Undo  Redo  Help  9:34 AM 0
  *evolutionary_model.py x
      c.append('red')
      return c

  def get_foci_nodes():
      f=[]
      for each in G.nodes():
          if G.node[each]['type']=='foci':
              f.append(each)
      return f

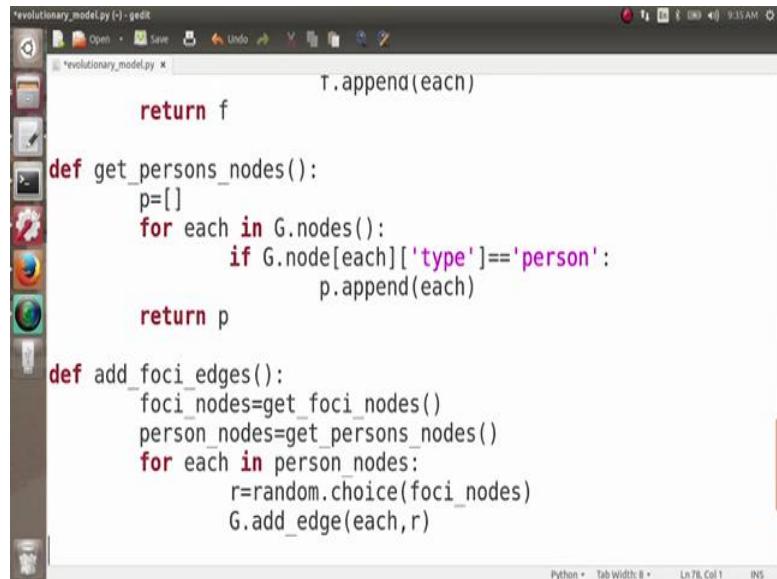
  def get_persons_nodes():
      p=[]
      for each in G.nodes():
          if G.node[each]['type']=='person':
              p.append(each)
      return p

```

Then what do we do is; f.append; f.append each and then you return f. So, this particular function will return you an array or foci\_nodes and we do exactly the same thing for the person's nodes. So, instead of writing the code again; let us just copy paste this code and make the changes. So, get\_foci\_nodes and here we have get persons nodes and what we do not want to do here is let this array be p. So, for each in G.nodes, if

`G.node[each]['type']` equals to `equals to` and we have here person, then we append this in array `p` and then we return `p`.

(Refer Slide Time: 15:37)



The screenshot shows a Gedit text editor window titled "revolutionary\_model.py - gedit". The code in the editor is as follows:

```
t.append(each)
return f

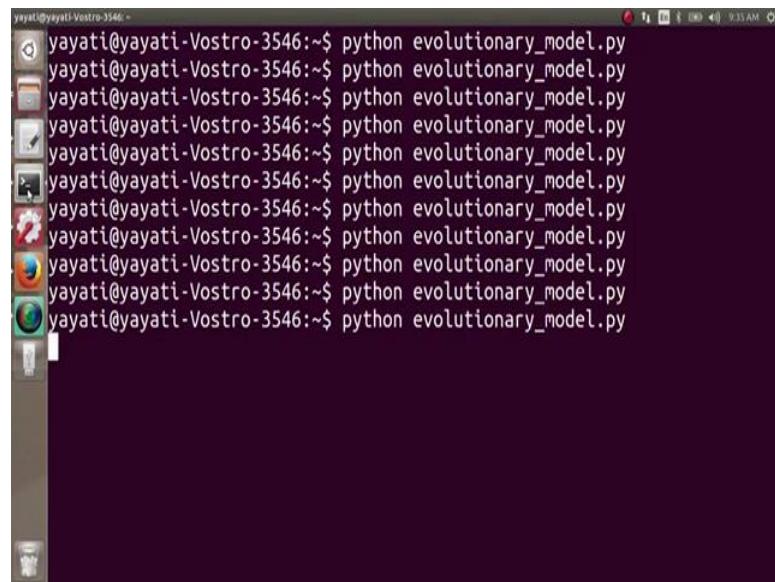
def get_persons_nodes():
    p=[]
    for each in G.nodes():
        if G.node[each]['type']=='person':
            p.append(each)
    return p

def add_foci_edges():
    foci_nodes=get_foci_nodes()
    person_nodes=get_persons_nodes()
    for each in person_nodes:
        r=random.choice(foci_nodes)
        G.add_edge(each,r)
```

The code defines two functions: `get_persons_nodes` and `add_foci_edges`. The first function creates an array `p` and appends nodes to it if their type is 'person'. The second function creates arrays for foci nodes and person nodes, then iterates over person nodes to add edges from each to a random node in the foci nodes.

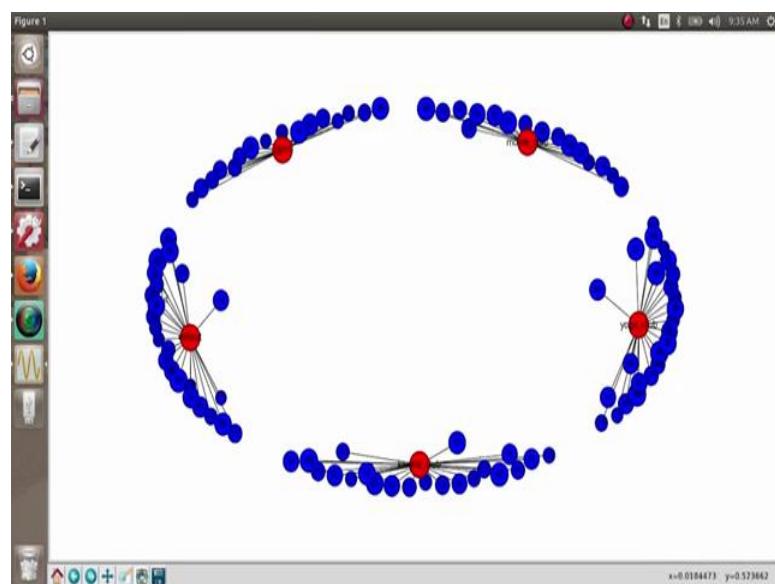
So, here we have got an array one is for the `foci_nodes` and another is for the person's node. Now, what we want to do is add an edge, so what do we do for adding an edge is for each in and we want to add an edge from every person nodes. So, for each in person nodes, what we do is, we choose a random node from the `foci_nodes`. So, `r = random.choice`, so `random.choice` is used to get a random value from an array; `r` equals to `random.choice` from `foci_nodes` and then what we do is `G.add_edge` and we add edge from each to `r`. So, for every node in the person's node, we choose a node randomly from the `foci_nodes` and add an edge between the 2.

(Refer Slide Time: 16:26)



```
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
```

(Refer Slide Time: 16:31)

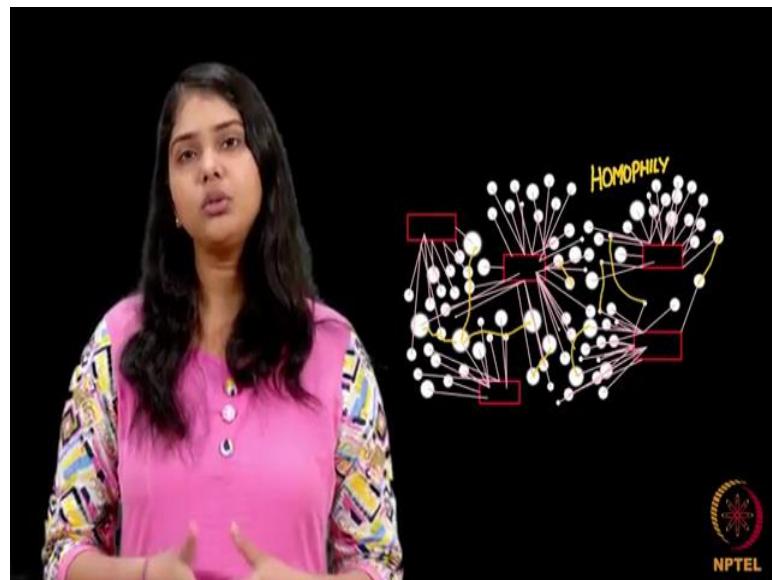


So, let us go back and look at the output; so, you can see here what we have is; every person has chosen here one foci\_nodes. So, these people here are part of gym, these people are part of movie club, these people are part of yoga club, karate club and eat out.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture – 49**  
**Strong and Weak Relationships (Continued) and Homophily**  
**Fatman Evolutionary Model - Implementing Homophily**

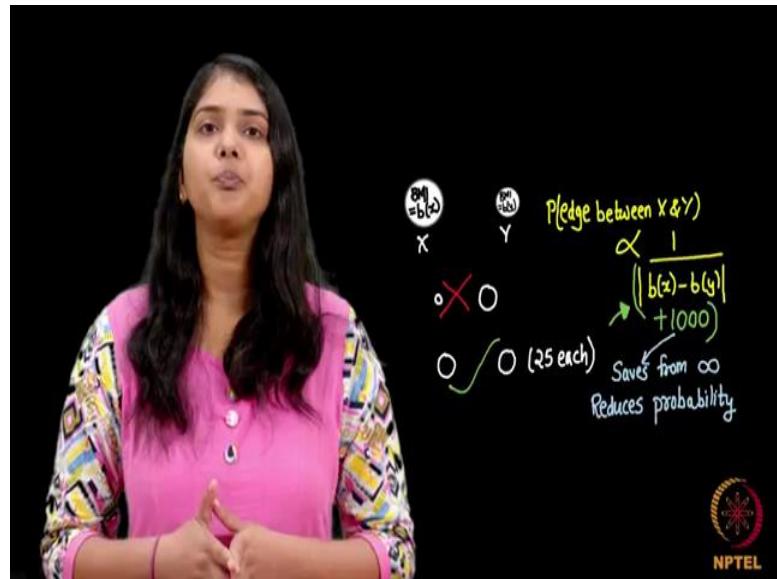
(Refer Slide Time: 00:10)



Till now we have made a network which has 100 people of different body weights and then we have 5 social foci including a gym and an eat out place and we have seen that initially everybody is a part of exactly one social foci and with time people will start becoming parts of more social foci because of membership closer.

Next what we want to implement in this coding what we want to happen in this network is homophily that is the people who are having singular BMI should become friends with each other. So, we are going to use a very simple method to do it. So, we see that the probability of two people becoming friends with each other is inversely proportional to the difference between their BMIs.

(Refer Slide Time: 00:53)



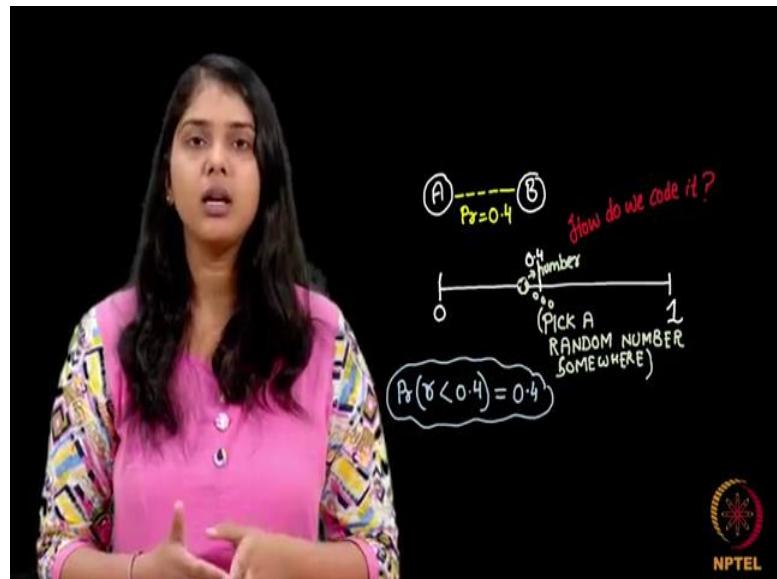
So, more is the difference between your BMIs less probable two people are to become friends with each other. So, if let say a person is underweight having a BMI of 15 and a person is overweight having a BMI of 40. So, it is very unlikely that they will become friends. And if two people they have almost the same BMI. So, let us say 25 each then they are likely to become friends. So, we can directly use a formula that probability of two people becoming friends with each other is inversely proportional to the difference between their BMIs.

So, probability of an edge between  $u$  and  $v$  is proportional to one divided by the absolute difference between them. So, we take absolute to avoid a negative there. So, we have absolute there and we add an extra factor of plus 1000 here. Why do we add an extra factor of plus thousand here is if you see by this formula if we do not have a 1000 here and let us say the BMIs of two people are equal, let us say 25 each then the probability of them becoming friends becomes 1 by 0 which is infinity, so we have a 1000 here which not only help us deal with this infinity issue what it does is it reduces all the probabilities.

So, although the probability of one person becoming connected to another is remains inversely proportional to the difference between their BMIs, but we want this probabilities to be less because since we want to see the network to be evolving we do not want a lot of edges to enter at the same time. So, adjust should come slowly and

slowly. So, this plus 1000 helps us reduced all the probability value, it scales all the probability values down. So, we use this formula for implementing homophily.

(Refer Slide Time: 03:16)



And one interesting concept here which I would like to tell you, you see here what we are going to do is the here are two people. Let us say A and B and we are saying that assume the probability with which A and B become friends is 0.4. So, there is a 40 percent chance that A and B become friends and 60 percent chance that they do not become friends. How do we implement this in our coding? So, this random package helps us do this. So, what I do is assume, I pick a random number random real number from a number line from 0 to 1 as shown here.

I want some event to happen with a probability of 0.4. So, now, I have a random number here between 0 and 1. So, I pick a random number let us say it is here, what is the probability that this random number is less than 0.4. So, since this random number can be anywhere on this number line if you see the probability that this random number is less than 0.4 is actually 0.4 itself. So, 0 to 0.4 is a smaller slot, 0.4 to 1 is a bigger slot. So, there is a lesser probability of this number falling below 0.4 and there is a higher probability it lies in the range 0.4 to 1. So, we will be using this method to implement what we are saying. So, assume there are two nodes u and v and we want them to become friends with probability 40 percent 0.4. What will do is generate this random

number are from 0 to 1 and if its value is less than 0.4, we make an edge between these two people otherwise we do not make an edge.

(Refer Slide Time: 04:55)



So, now before implementing homophily I want to do one extra thing. So, we have added colour to a node. So, we have added blue colour for the person node and red colour for the foci nodes.

(Refer Slide Time: 05:05)

```
i=i+1
def get_colors(G):
    c=[]
    for each in G.nodes():
        if G.node[each]['type']=='person':
            if G.node[each]['name']==15:
                c.append('green')
            elif G.node[each]['name']==40:
                c.append('yellow')
            else:
                c.append('blue')
        else:
            c.append('red')
    return c

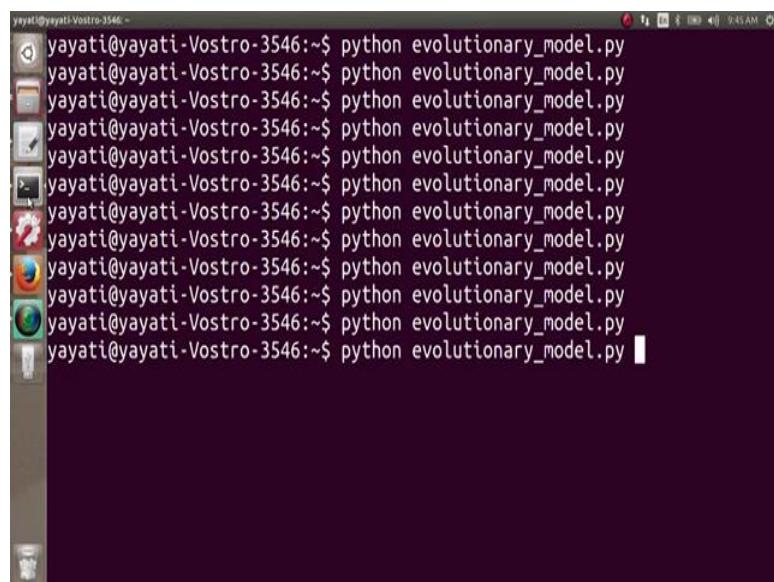
def get_foci_nodes():
    f=[1]
```

What I want to see I want to see underweight and overweight people separately, I want the underweight people to appear in let say green colour and overweight people to appear

in yellow colour, so that when we look at this graph across different timestamps we can see that how the number of people are changing and how the clustering between them is happening.

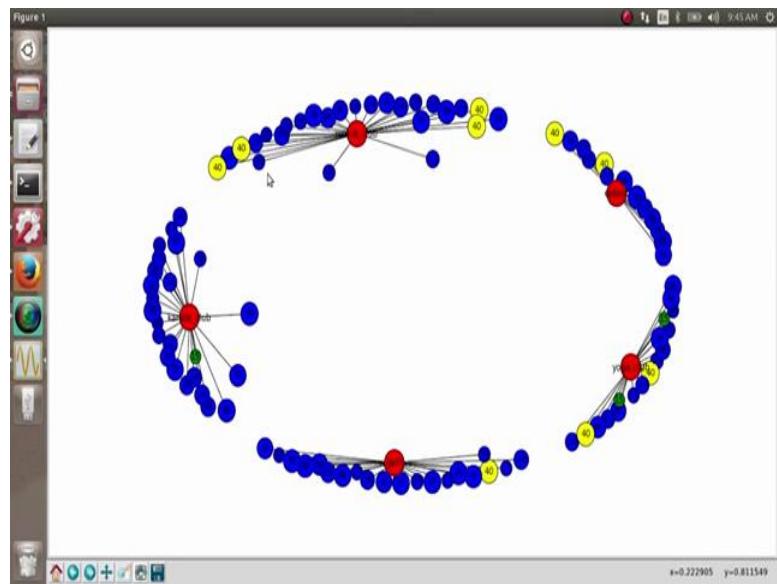
So, what we do here is if G node each type equals to equals to person and then further we have if G.node each name equals to equals to 15. So, what we do is in this case c.append what do we append here is a green and then we have else if which is written as elif in python. So, elif G.node each name equals to equals to 40, if it is 40, c.append and what do we append here is a yellow colour. And still if some node is left; obviously, many nodes are left having the BMI in between these two then we do c.append(blue). Let us save it and next let us look at the output.

(Refer Slide Time: 06:55)



So, you see here the graph looks I think all the more beautiful now. So, we have these nodes. So, blue nodes are mostly the normal nodes red nodes are the foci nodes. So, here you see all the nodes having BMI of 40 comes in the yellow colour and all the nodes having this BMI of 15 comes in the green colour.

(Refer Slide Time: 06:59)



Next if we go head and implement homophily in this network, let us see how we implement homophily. So, after adding foci edges what we want to do if we want to implement homophily.

(Refer Slide Time: 07:30)

```
evolutionary_model.py - gedit
evolutionary_model.py *
def add_foci_edges():
    foci_nodes=get_foci_nodes()
    person_nodes=get_persons_nodes()
    for each in person_nodes:
        r=random.choice(foci_nodes)
        G.add_edge(each,r)

def homophily(G):
    pnodes=get_persons_noded()

G=create_graph()
assign_bmi(G)
add_foci_nodes(G)
labeldict=get_labels(G)
nodesize=get_size(G)
color_array=get_colors(G)
```

Now what does homophily do? So, what will be implementing here will be one iteration of homophily what does homophily says nodes which are alike similar to each other they tend to become friends with each other, and we have already looked at the formula for this.

So, we define homophily here, define homophily G and what does it do first of all it gets all the person nodes. So, person node equals to get\_persons\_nodes because you see homophilies only going to happen between the people not the social foci. So, this function is explicitly defined for the person nodes. So, we get all the person nodes and for simplicity, let us simply name this array as pnodes. pnodes refer to person nodes pnodes equals to get\_persons\_nodes. You can also notice here that I am nowhere pass the parameter G here because G here is being considered as a global parameter - pnodes equals to get\_persons\_nodes.

(Refer Slide Time: 08:57)

```

revolutionary_model.py (~)- gedit
*revolutionary_model.py x
person_nodes=get_persons_nodes()
for each in person_nodes:
    r=random.choice(foci_nodes)
    G.add_edge(each,r)

def homophily(G):
    pnodes=get_persons_nodes()
    for u in pnodes:
        for v in pnodes:
            if u!=v:
                diff=abs(G.node[u]['name']-
G.node[v]['name'])
                p=float(1)/(diff+1000)
                r=random.uniform(0,1)
                if r<p:
                    G.add_edge(u,v)

```

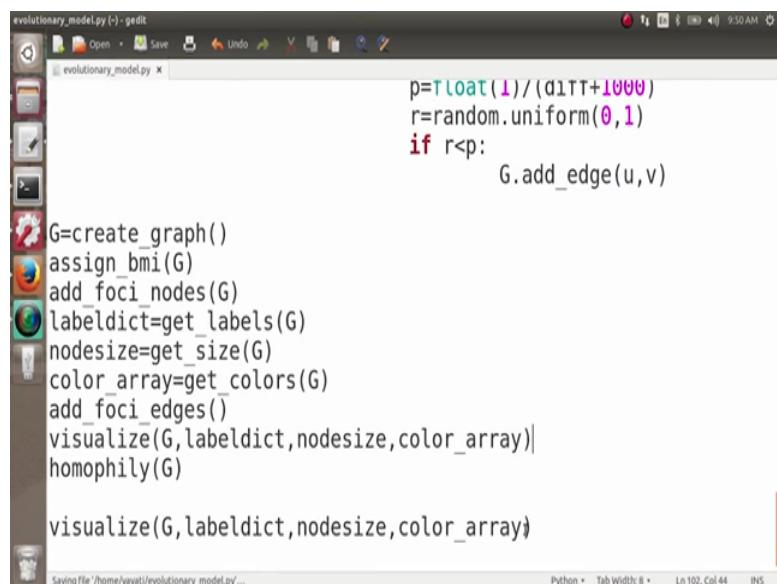
So, I have all the person nodes, now I want to see whether there BMI similar to each other and if it is similar to each other I will add edges between them what I do for u in pnodes. So, u is my first node for u in pnodes and then I have for v in pnodes, so it picks each of the person nodes every possible combination of two person nodes one by one and obviously, my u should not be equal to v because the person is not going to make tie with himself. So, if  $u \neq v$  what do we do is we calculate this difference in their BMI.

So, what is the difference in their BMI? It is the absolute value of, absolute value of  $G.\text{node}[u]['name']$  which carries the BMI of this person minus  $G.\text{node}[v]['name']$ . So, I have this difference here, after having this difference what is the probability that these two nodes will connect to each other as we have seen is inversely proportional to this difference - greater the difference lesser the probability,

lesser the difference higher the probability. So, probability of these people connecting is one divided by this difference plus 1000. So, we have this 1000 here to keep these probabilities small.

And next as discussed before we have a random number  $r$ , random real number  $r$  which takes its value from 0 to 1 and if this  $r < p$  what do we do; if  $G$  do not add edge between  $u$  and  $v$ . I think it is clear. So, what we do is we calculate the difference between two nodes, probability of these nodes connecting is inversely proportional to this difference.

(Refer Slide Time: 11:28)

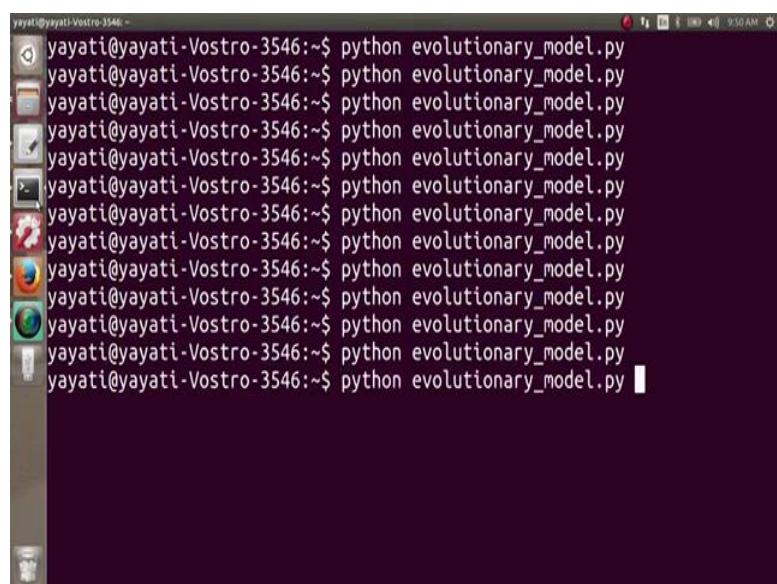


```
p=Tfloat(1)/(diff+1000)
r=random.uniform(0,1)
if r<p:
    G.add_edge(u,v)

G=create_graph()
assign_bmi(G)
add_foci_nodes(G)
labeldict=get_labels(G)
nodesize=get_size(G)
color_array=get_colors(G)
add_foci_edges()
visualize(G,labeldict,nodesize,color_array)
homophily(G)

visualize(G,labeldict,nodesize,color_array)
```

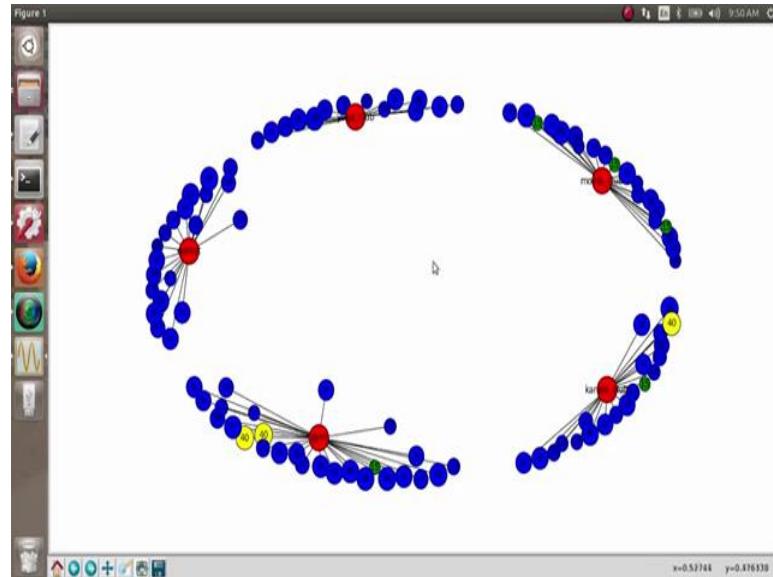
(Refer Slide Time: 11:33)



```
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
```

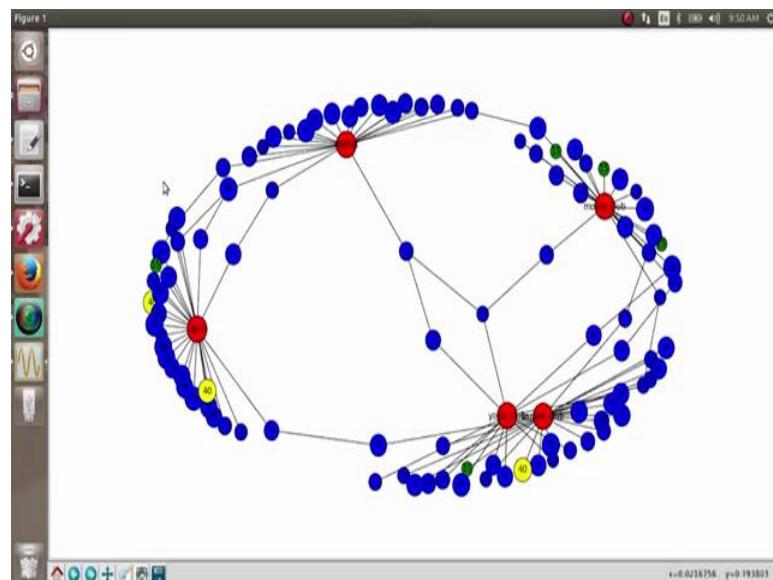
And with this probability these two nodes connect in every iteration. Let see what we want to do is let us visualize the graph once before homophily and once after homophily.

(Refer Slide Time: 11:39)



So, this is our graph before homophily, this is our graph before homophily, and this is our graph after homophily.

(Refer Slide Time: 11:45)

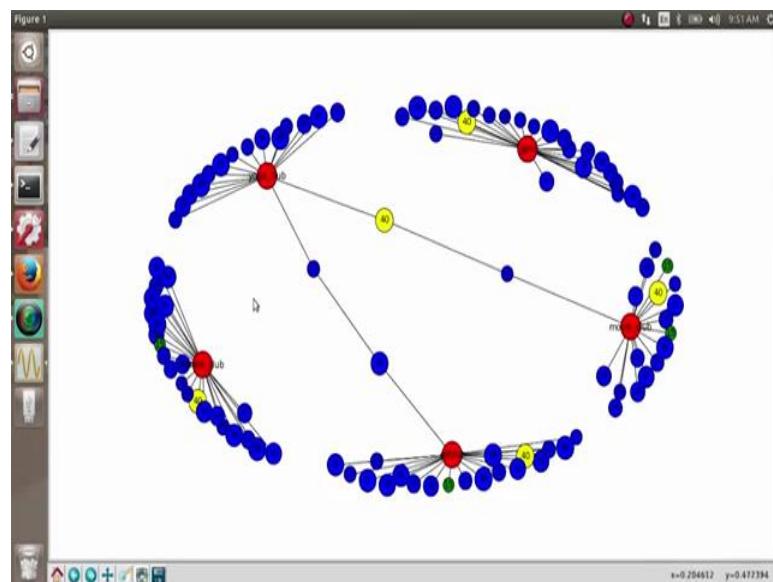


So, you can see here now we have many relationships across different clusters. So, these nodes previously they existed in different clusters, but you can see now our graph has become connected and more and more people has connected with each other. So, here

we have an edge between 35 25, 25 39, 39 27 and so on and these edges are in accordance with their body weights BMIs.

So, we can make this probability actually a little lesser if you want a smaller number of edges to be added at every step that is totally up to us. So, we can increase this probability of connection, we can decrease this probability of connection, but the probability should be inversely proportional. So, if you want to decrease the number of edges being added at every step what we can do is we can make this value all the higher and this probability gets less here. So, you see here that this was our initial network.

(Refer Slide Time: 12:51)

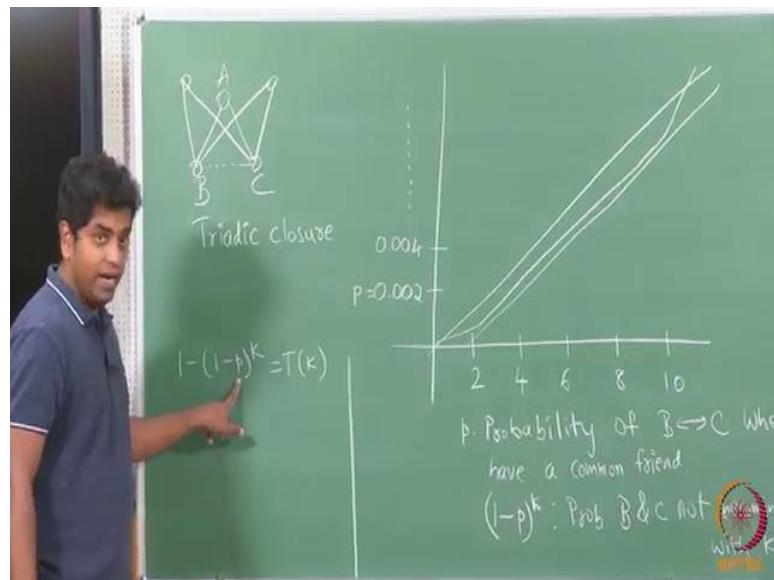


And this is our new network. So, you see here a smaller number of edges have been added, a smaller number of edges have been added it here.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

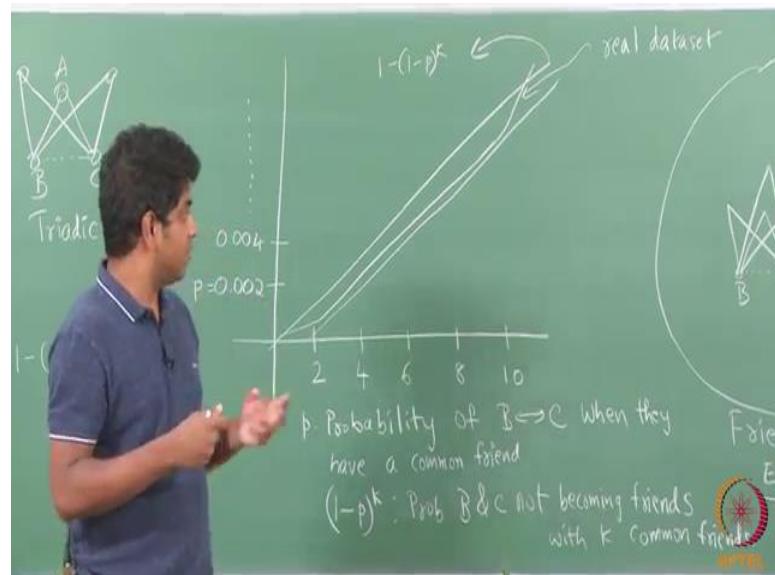
**Lecture – 50**  
**Strong and Weak Relationships (Continued) and Homophily**  
**Quantifying the Effect of Triadic Closure**

(Refer Slide Time: 00:14)



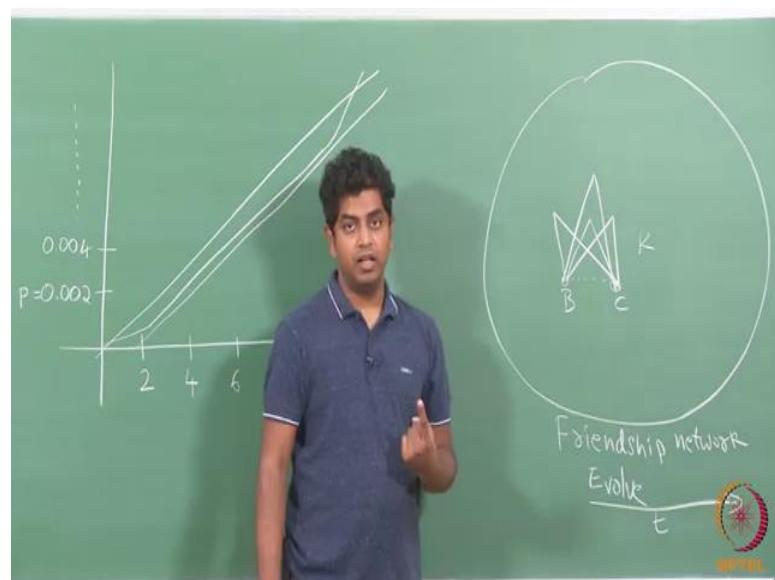
So, we saw about triadic closure where a person A has two friends namely B and C, and there is enormous pressure for B and C to become friends, this is called the Triadic closure perfect. So, now, as you can see if B and C have a common friend there is pressure for them to become friends. As they say a friend's friend is a prospective friend right now what if there are many such common friends do you think the pressure increases for B and C to become friends, if there are a lot of common friends you just now watch the video clip which said two people are sitting and then they discussing they should probably become friends, simply because they have a whole lot of common friends.

(Refer Slide Time: 01:28)



What is this whole lot of common friends mean do you think two people becoming friends has something to do with their common friendship? There is a space of free such that answers this question, people considered the email communication network and then tried plotting the following. They said if there are two common friends 4 common friends, 6 common friends, 8 and 10 common friends let us say and if you look at the probability of them becoming friends let us say 0.002, 0.004 and so, on right you observe it is a very small probability.

(Refer Slide Time: 02:05)



So, what we do is we look at our friendship network over a timeline; friendship network you look at it evolve with time  $T$ , and then we observe that two people who have  $K$  number of friends  $K$  number of common friends

Let us say what is the probability of them becoming friends in the next time step? For example, today I look at this friendship network this  $B$  and  $C$  are not friends, but they have  $K$  common friends with them, tomorrow I will observe are they still friends may be not I keep going like this and I observe that eventually they become friends. So, now, I will consider the following, given two people  $B$  and  $C$  if they have  $K$  number of common friends and if they become friends right tomorrow then I will make a note of that and across this timeline which is sort of this friendship network which is evolving across this timeline I will try to draw this plot of what is the probability with this data I can write a plot here what is the probability of we having our closure between  $B$  and  $C$ , when the number of common friends are 2, the number of common friends are 4, the number of common friends are 6 so on and so forth.

They observe that the plot looks something like this very linear which means increase in the number of common friends results in the increase in the probability of them becoming friends. Although as you can observe the probability is actually on the very small side its very small 0.0002 and we probably we can go up to 0.008 or 0.010 or something right and this looks linear what does it signify they tried looking at the fact if this can be fitted over some linear curve and they were successful. How did they do this? This, simply it the following they tried looking at the fact that if there are  $K$  number of friends with a probability of me becoming friends with someone if I have if the probability of  $B$  and  $C$  becoming friends if they have one common friend let us call that  $p$ .

If  $p$  is the probability.

If  $p$  is the probability of  $B$  and  $C$  becoming friendship when they have a common friend  $1 - p$  is the probability of them becoming not becoming friends. If  $p$  is the probability of them becoming friends when they have a common friend  $1 - p$  is the probability of them becoming them not becoming friends right and assume you have  $K$  such common friends as we know  $1 - p$  whole to the case the probability of you having  $K$  common friends, at  $B$

and C are not friends this is the probability let me write this down, this is the probability of B and C not becoming friends despite the fact that they have K common friends.

Now, what I will conclude here, what will  $1 - p$  whole to the K denote this will be denote the probability of this not happening, because I am considering  $1 - p$  of that. So, what this signify this signifies probability of B and C becoming friends with K common friends, let me denote this as T of K. I repeat T of K signifies the probability of two people becoming friends when they have K common friends provided  $p$  is the probability of B and C becoming friends when they have a common friend, they observed that for an appropriate  $p$  you will basically get this straight line and this straight line and this curve that you observed that I wrote here sort of agree.

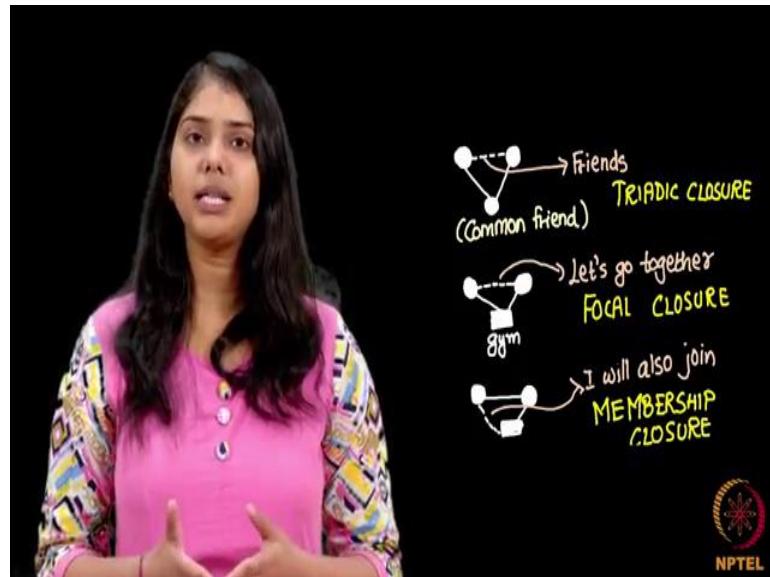
This is the curve that they observed and real-world data set real data set and this straight line is the curve  $(1 - (1 - p))^K$ . As you can see it is a pretty straight forward analysis they observe its linear and there then try to see what could be this  $p$  they observe that the  $p$  is very small whatever B it was observed the moral of the story is that it was observed that the probability of two people becoming friends goes higher and higher, as the number of common friends go higher and higher.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture – 51**  
**Strong and Weak Relationships (Continued) and Homophily**  
**Fatman Evolutionary Model - Implementing Closures**

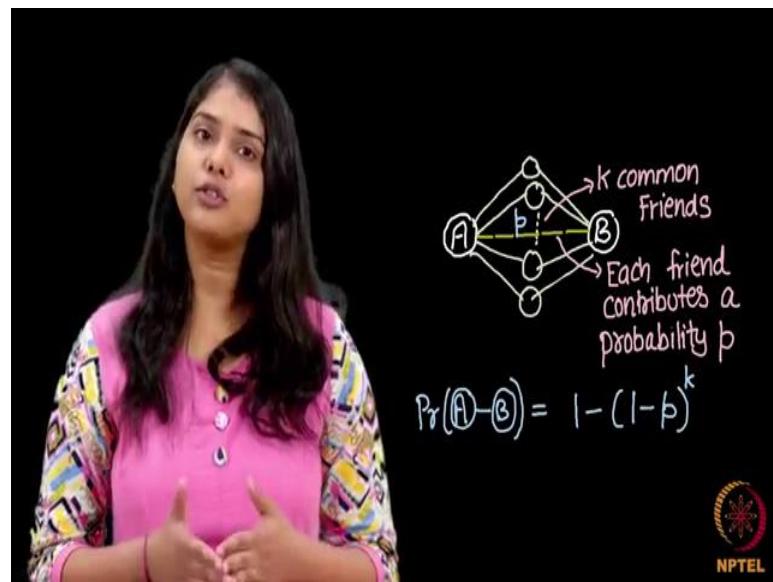
We next want to implement closures in our model, whereas we have seen there are 3 kinds of closures; triadic closure we both have a common friend we become friends.

(Refer Slide Time: 00:13)



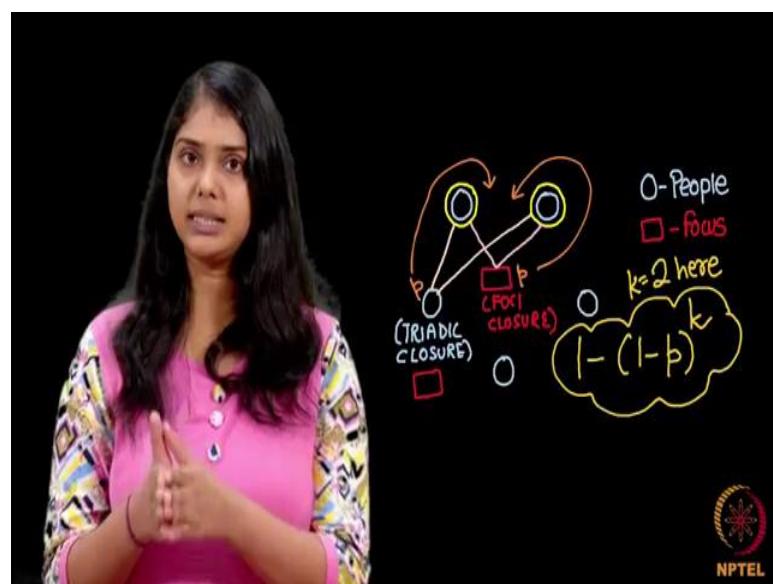
Foci closure: we both are going to the same gym, so we become friends; membership closure I am your friend, you go to gym, so I start going to gym. Now, how do we implement these closures in our model is now the question. So, how have we implemented these closures before? So, we have used a formula if you remember.

(Refer Slide Time: 00:44)



So, what we have done is; if there are 2 people A and B and these people have let us say some K number of common friends, then assume that each of these friends; each of these common friends contributes to a probability p. So, one common friend contributes to a probability of p that these 2 people will become friends. Assume the; it is 0.4, so one person says that with 40 percent probability, these 2 people A and B will become friends and then we looked at this formula that the probability that these 2 people will become friends in the next iteration is 1 minus; 1 minus p to the power k.

(Refer Slide Time: 01:38)

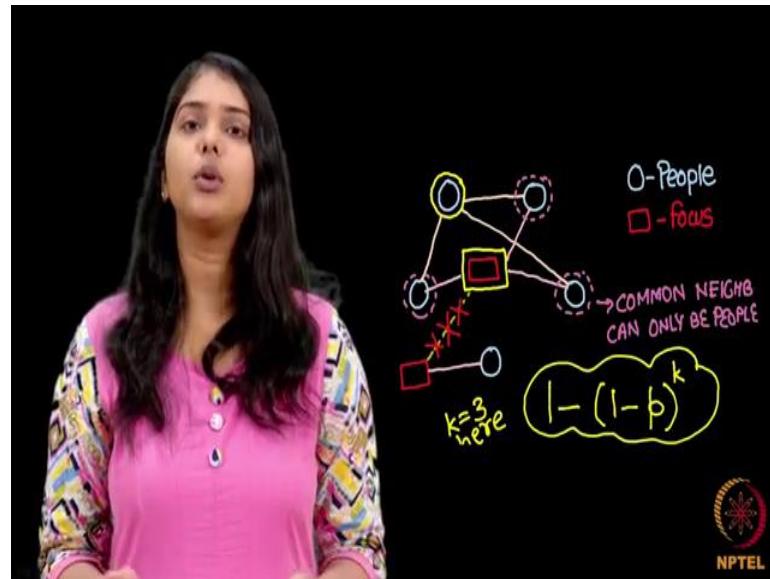


So, we will be using the same formula for implementing all 3 kinds of closures. Now, how do we implement it? First, we will look at every 2 possible combination of nodes. So, when we pick 2 nodes from this network; we will pick every possible pair of 2 nodes. So, when we pick these 2 nodes from the network what can be their types; so, both of these can be the person nodes; person A, person B or one can be a person node, and another can be a foci node.

So person and foci node or vice versa foci node and person node, so, mainly one person node and one foci node and in the last case both of these can be the foci node. How do closures happen in all of these 3 cases? So, in the first case when there are 2 people node; so if we look at the common neighbors, so these 2 nodes. So, we know that in our model the common neighbors can be a person as well as the foci. So, we use the same formula for both of these things, so we capture both the triadic closure and the foci closure using the same formula.

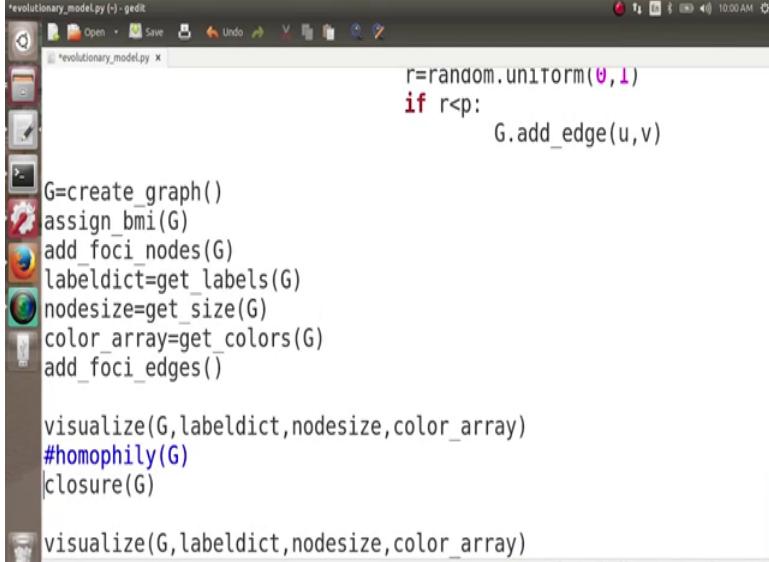
So, we assume that whether it is a person which is your common friend, or it is a foci where both of these are going contributes to a probability  $p$  of them becoming friends. So, as I told you that this is a very; we are trying to keep things very simple here. So, once we write the complete coding; we can actually change both of these probabilities, we can assume that or we can code it that if there is a person node; it has a higher probability of making them friends and if it is a foci node, it has a less probability of making them friends, but for now we keep both of these the same. So, whether it is a common friend or common foci; it contributes to a probability  $p$  of both of these becoming friends and then again, we use the same formula. So, that formula helps us in achieving both things here triadic closure and foci closure.

(Refer Slide Time: 03:38)



Let us now look at the second case, so here is a person and then here is a social foci; now we look at their common neighbors. Now, if you look at their common neighbors; you observe that their common neighbors can only be persons, their common neighbors cannot be foci, why, because there are no edges between the foci nodes, so 2 foci nodes are never connected to each other, it is the edges are only between person to person and person to foci. So, we have a person and we have a foci here and all their common neighbors are different-different persons. So, this means that this person here can have  $k$  of his friends going to the social foci and again we assume that each of these common friends contributes to a probability of  $p$ .

(Refer Slide Time: 04:44)



```
r=random.uniform(0,1)
if r<p:
    G.add_edge(u,v)

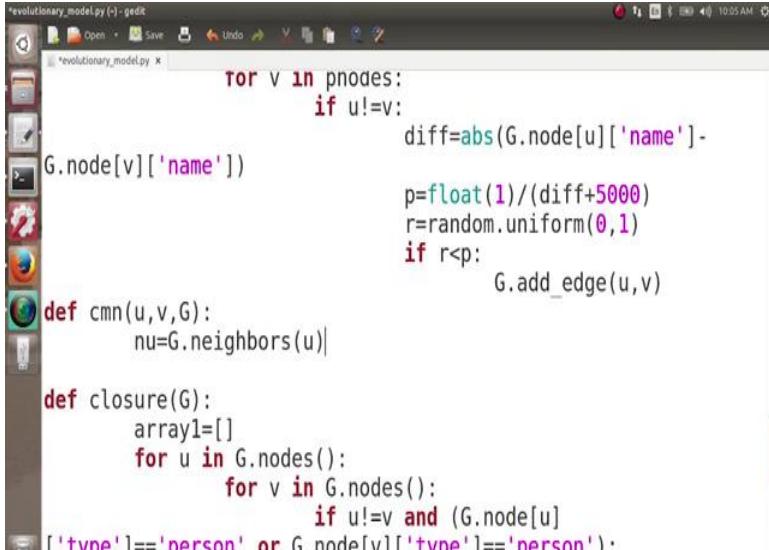
G=create_graph()
assign_bmi(G)
add_foci_nodes(G)
labeldict=get_labels(G)
nodesize=get_size(G)
color_array=get_colors(G)
add_foci_edges()

visualize(G,labeldict,nodesize,color_array)
#homophily(G)
closure(G)

visualize(G,labeldict,nodesize,color_array)
```

For this person also to go to this social foci and again we use the same formula. So, you see how we just use one formula for implementing all these 3 kinds of closures; once we call it we can actually change the things here and make it more advanced. Next, we want to implement closure property in this graph, in the meantime; we comment the homophily. So, that we can clearly see what is happening in closure and we have here closure(G).

(Refer Slide Time: 05:03)



```
for v in nodes:
    if u!=v:
        diff=abs(G.node[u]['name']-
G.node[v]['name'])
        p=float(1)/(diff+5000)
        r=random.uniform(0,1)
        if r<p:
            G.add_edge(u,v)

def cmn(u,v,G):
    nu=G.neighbors(u)

def closure():
    array1=[]
    for u in G.nodes():
        for v in G.nodes():
            if u!=v and (G.node[u]
['type']=='person' or G.node[v]['type']=='person'):
```

So as we have discussed there are 3 kinds of closures which we have to implement become here define closure( $G$ ) and what you have to do for closure is we know that in one iteration everybody is going to look at their common friends to any 2 people or any 2 people there going to look at all their common friends and they are going to decide whether in the next iteration, they are going to connect to each other or not.

So, what we do here is take an array let us say array1 and this array will have 3 values. So, how will this array look like; this array will be composed of sub arrays and each sub array will be having 3 values. The first value is the first node, second value is the second node and third value are the probability of connection. So, when we are going in an iteration of closure( $G$ ); we do not keep adding edges side by side like we did for homophily. Because, if you add an edge between you looked at 1 and 2 and you added an edge between them, it can have complications for the addition of the coming up edges.

So, the addition of every edge should be independent of what has been added before. So, to maintain that independence we have here this array; array1 and so these are the kinds of values this array1 will holds, so here it can be 2, 5 and then 0.7 and so on. So, how do we go about doing this, so initially we have this array1 which is empty and then now by after looking at all the nodes and their common neighbors, we have to decide whether we have to add an edge between them or not.

As we have seen that for any kind of closure to happen, the only condition required is whatever nodes we are picking both of though node should not be the foci nodes. Otherwise, if both of them are the people nodes; it results in a simple closure and otherwise the other 2 cases result in a membership closure and a foci closure; the only condition is both of the node should not be foci nodes.

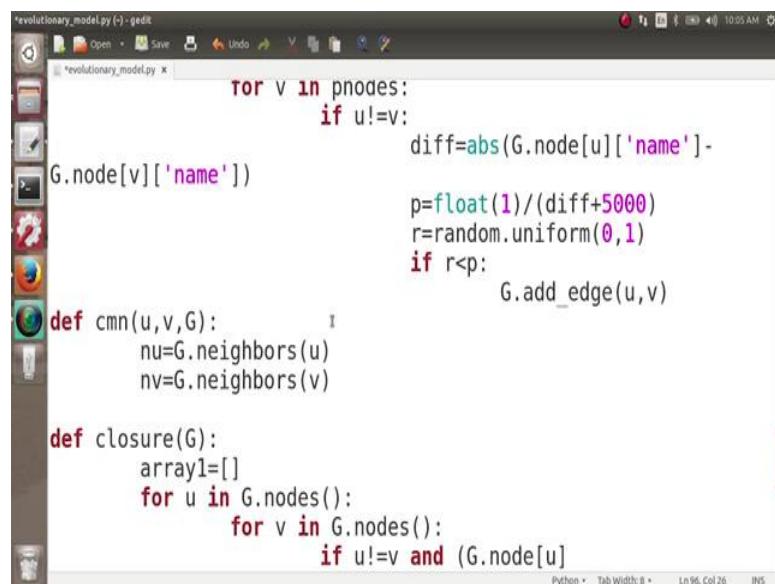
Now, what do we do is for  $u$  in  $G.nodes$  and then for  $v$  in  $G.nodes$ ; now both of these nodes should be the foci nodes at least one of these should be the person nodes. So, if first of all  $u \neq v$  and either  $G.node[u]['type'] == 'person'$  or  $G.node[v]['type'] == 'person'$ . So, one of they should be person; in that case we have to implement that closure. We already know; what was our formula for closure, and we are using that same formula for all 3 kinds of closure.

So, here we find out the value of a probability  $p$  which is nothing, but our formula so our formula was  $(1 - (1 - p))^k$ . So, for having  $(1 - p)^k$ ; we have this function math. power.

So, we have to add this package math at the (Refer Time: 08:47) which will do, so  $1 - \text{math.pow}(1 - p, k)$ , but now what is k here; k is the number of common neighbors . So, we need the number of common neighbors between u and v. So, we have here k equals to; find k equals to common neighbors.

So, let us have simple function in cmn common, common neighbors between u and v in the graph G. So, now we need to define this function here; define cmn common neighbors of u, v, G and how do we find its value is first we need the neighbors of u.

(Refer Slide Time: 09:57)



The screenshot shows a Gedit text editor window titled "evolutionary\_model.py". The code is written in Python and defines several functions related to a social network model:

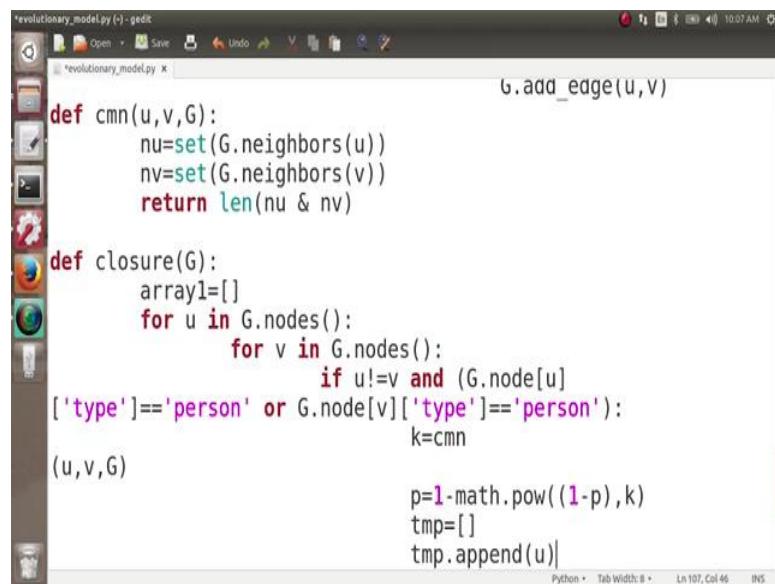
```
for v in nodes:
    if u!=v:
        diff=abs(G.node[u]['name']-G.node[v]['name'])
        p=float(1)/(diff+5000)
        r=random.uniform(0,1)
        if r<p:
            G.add_edge(u,v)

def cmn(u,v,G):
    nu=G.neighbors(u)
    nv=G.neighbors(v)

def closure(G):
    array1=[]
    for u in G.nodes():
        for v in G.nodes():
            if u!=v and (G.node[u]
```

So, neighbors of u = G.neighbors(u) and then neighbors of v = G.neighbors(v) and we want the common one.

(Refer Slide Time: 10:28)

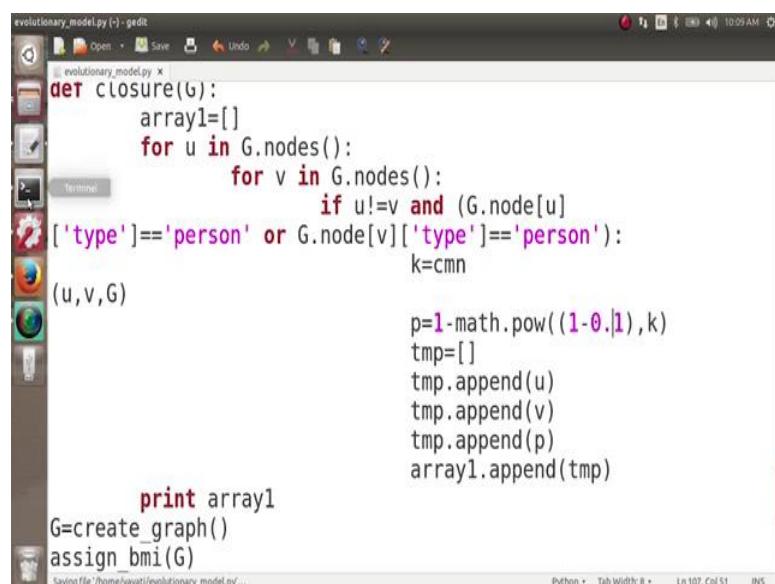


```
evolutionary_model.py (~) - gedit
  File  Open  Save  Undo  Redo  Help  10:07 AM
  evolutionary_model.py
def cmn(u,v,G):
    nu=set(G.neighbors(u))
    nv=set(G.neighbors(v))
    return len(nu & nv)

def closure(G):
    array1=[]
    for u in G.nodes():
        for v in G.nodes():
            if u!=v and (G.node[u]['type']=='person' or G.node[v]['type']=='person'):
                k=cmn
                (u,v,G)
                p=1-math.pow((1-p),k)
                tmp=[]
                tmp.append(u)
                tmp.append(v)
                tmp.append(p)
                array1.append(tmp)
    print array1
G=create_graph()
assign_bmi(G)
Saving file /home/yayati/evolutionary_model.py...
Python  Tab Width: 8  Ln 107, Col 46  INS
```

So, for having common one we need a set intersection, so let us make both of these value a sets. So, nu is the set having the neighbors of u and n, v is a set having the neighbors of v and what we have to return is the length of the set which set n, u and n, v which only looks at the common elements of u and v; to be written length of nu and nv. So, we have this probability here;  $\text{math.pow}(1 - p, k)$ . Now, we have to feed these values inside our array which has to had 3 parts u, v and the probability of connection.

(Refer Slide Time: 11:33)

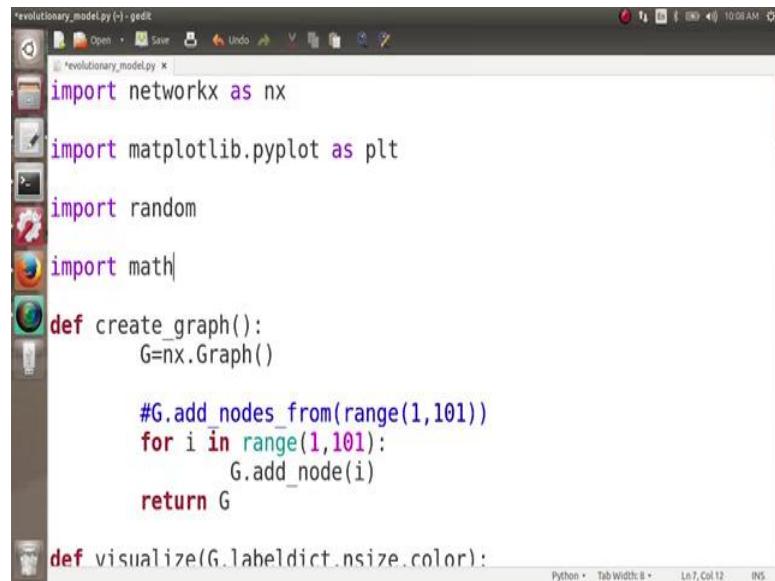


```
evolutionary_model.py (~) - gedit
  File  Open  Save  Undo  Redo  Help  10:09 AM
  evolutionary_model.py
def closure(G):
    array1=[]
    for u in G.nodes():
        for v in G.nodes():
            if u!=v and (G.node[u]['type']=='person' or G.node[v]['type']=='person'):
                k=cmn
                (u,v,G)
                p=1-math.pow((1-0.1),k)
                tmp=[]
                tmp.append(u)
                tmp.append(v)
                tmp.append(p)
                array1.append(tmp)
    print array1
G=create_graph()
assign_bmi(G)
Saving file /home/yayati/evolutionary_model.py...
Python  Tab Width: 8  Ln 107, Col 51  INS
```

So, we need a sub array here let us name it strength, so the first value which we are going to append(tmp) phase u and the second value which we are going to append(tmp) is v and then the third value which we append(tmp) is p.

So, tmp has these 3 values and at the end we append(tmp) in array1; array1.append(tmp). So, we have made here this array1; for all possible pairs of nodes, let us quickly look at how this array1 looks like.

(Refer Slide Time: 12:14)



```
evolutionary_model.py (~) - gedit
import networkx as nx
import matplotlib.pyplot as plt
import random
import math
def create_graph():
    G=nx.Graph()
    #G.add_nodes_from(range(1,101))
    for i in range(1,101):
        G.add_node(i)
    return G
def visualize(G,labeleddict,nsize,color):
```

So, let us print array1 and see how it looks like; now we forgot to import the package math, we go up and here we import math.

(Refer Slide Time: 12:35)

```
evolutionary_model.py (-) - gedit
Saving file /home/yayati/evolutionary_model.py ...
Python • Tab Width: 8 • Ln 126, Col 2 INS
array1.append(tmp)
print array1
G=create_graph()
assign_bmi(G)
add_foci_nodes(G)
labeldict=get_labels(G)
nodesize=get_size(G)
color_array=get_colors(G)
add_foci_edges()

#visualize(G,labeldict,nodesize,color_array)
#homophily(G)
closure(G)

#visualize(G,labeldict,nodesize,color_array)
```

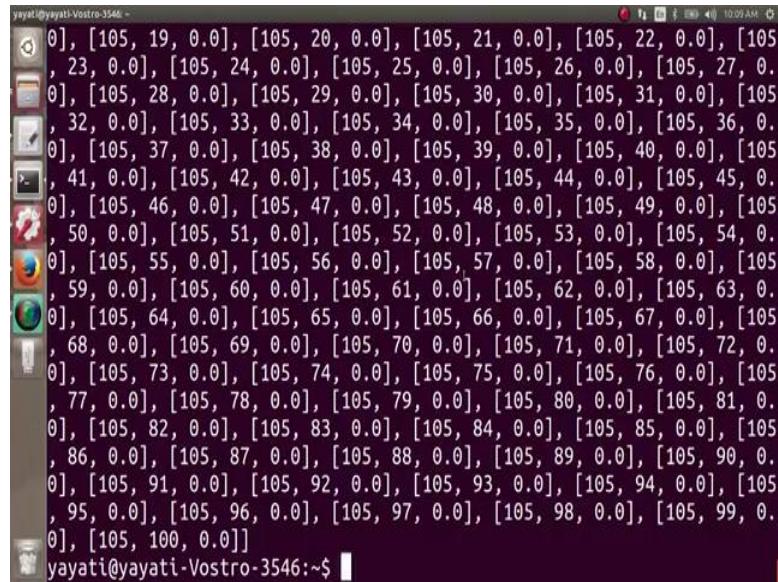
And at the end is this is become a (Refer Time: 12:37) code, so at the end for the time being we comment and visualize functions as well and now let us look at the array.

(Refer Slide Time: 12:42)

There is some problem, so this  $p$  here;  $p$  in the formula, this  $p$  here was supposed to be a constant. What was that constant value of probability? That is the probability each of your common neighbor contributes to  $u$  forming a connection. Let us come back, when we come here to closure  $p$  equals to this probability is supposed to be some value here.

So, let us say each of my neighbour contributes 0.01 probability of connection; let us rather say 0.1 probability of connection; then the more common neighbors we have, more probability of connection becomes; let us go back execute it.

(Refer Slide Time: 13:52)



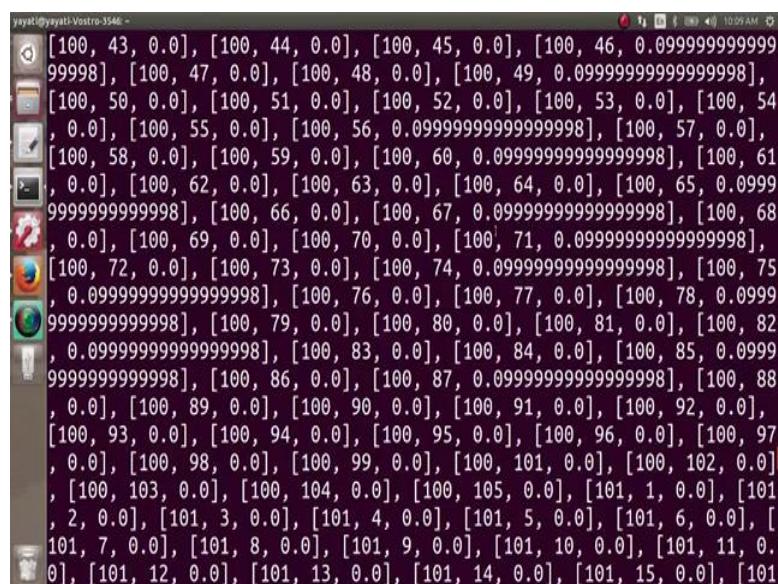
```
yayati@yayati-Vostro-3546:~
```

```
[0], [105, 19, 0.0], [105, 20, 0.0], [105, 21, 0.0], [105, 22, 0.0], [105, 23, 0.0], [105, 24, 0.0], [105, 25, 0.0], [105, 26, 0.0], [105, 27, 0.0], [105, 28, 0.0], [105, 29, 0.0], [105, 30, 0.0], [105, 31, 0.0], [105, 32, 0.0], [105, 33, 0.0], [105, 34, 0.0], [105, 35, 0.0], [105, 36, 0.0], [105, 37, 0.0], [105, 38, 0.0], [105, 39, 0.0], [105, 40, 0.0], [105, 41, 0.0], [105, 42, 0.0], [105, 43, 0.0], [105, 44, 0.0], [105, 45, 0.0], [105, 46, 0.0], [105, 47, 0.0], [105, 48, 0.0], [105, 49, 0.0], [105, 50, 0.0], [105, 51, 0.0], [105, 52, 0.0], [105, 53, 0.0], [105, 54, 0.0], [105, 55, 0.0], [105, 56, 0.0], [105, 57, 0.0], [105, 58, 0.0], [105, 59, 0.0], [105, 60, 0.0], [105, 61, 0.0], [105, 62, 0.0], [105, 63, 0.0], [105, 64, 0.0], [105, 65, 0.0], [105, 66, 0.0], [105, 67, 0.0], [105, 68, 0.0], [105, 69, 0.0], [105, 70, 0.0], [105, 71, 0.0], [105, 72, 0.0], [105, 73, 0.0], [105, 74, 0.0], [105, 75, 0.0], [105, 76, 0.0], [105, 77, 0.0], [105, 78, 0.0], [105, 79, 0.0], [105, 80, 0.0], [105, 81, 0.0], [105, 82, 0.0], [105, 83, 0.0], [105, 84, 0.0], [105, 85, 0.0], [105, 86, 0.0], [105, 87, 0.0], [105, 88, 0.0], [105, 89, 0.0], [105, 90, 0.0], [105, 91, 0.0], [105, 92, 0.0], [105, 93, 0.0], [105, 94, 0.0], [105, 95, 0.0], [105, 96, 0.0], [105, 97, 0.0], [105, 98, 0.0], [105, 99, 0.0], [105, 100, 0.0]]
```

```
yayati@yayati-Vostro-3546:~$
```

So, now you can see here your array and as we can see that as of now most of the values are here 0 and there is a reason for it to be 0 because there are very fewer common neighbors here.

(Refer Slide Time: 14:10)



```
yayati@yayati-Vostro-3546:~
```

```
[100, 43, 0.0], [100, 44, 0.0], [100, 45, 0.0], [100, 46, 0.09999999999999998], [100, 47, 0.0], [100, 48, 0.0], [100, 49, 0.09999999999999998], [100, 50, 0.0], [100, 51, 0.0], [100, 52, 0.0], [100, 53, 0.0], [100, 54, 0.0], [100, 55, 0.0], [100, 56, 0.09999999999999998], [100, 57, 0.0], [100, 58, 0.0], [100, 59, 0.0], [100, 60, 0.09999999999999998], [100, 61, 0.0], [100, 62, 0.0], [100, 63, 0.0], [100, 64, 0.0], [100, 65, 0.09999999999999998], [100, 66, 0.0], [100, 67, 0.09999999999999998], [100, 68, 0.0], [100, 69, 0.0], [100, 70, 0.0], [100, 71, 0.09999999999999998], [100, 72, 0.0], [100, 73, 0.0], [100, 74, 0.09999999999999998], [100, 75, 0.09999999999999998], [100, 76, 0.0], [100, 77, 0.0], [100, 78, 0.09999999999999998], [100, 79, 0.0], [100, 80, 0.0], [100, 81, 0.0], [100, 82, 0.09999999999999998], [100, 83, 0.0], [100, 84, 0.0], [100, 85, 0.09999999999999998], [100, 86, 0.0], [100, 87, 0.09999999999999998], [100, 88, 0.0], [100, 89, 0.0], [100, 90, 0.0], [100, 91, 0.0], [100, 92, 0.0], [100, 93, 0.0], [100, 94, 0.0], [100, 95, 0.0], [100, 96, 0.0], [100, 97, 0.0], [100, 98, 0.0], [100, 99, 0.0], [100, 101, 0.0], [100, 102, 0.0], [100, 103, 0.0], [100, 104, 0.0], [100, 105, 0.0], [101, 1, 0.0], [101, 2, 0.0], [101, 3, 0.0], [101, 4, 0.0], [101, 5, 0.0], [101, 6, 0.0], [101, 7, 0.0], [101, 8, 0.0], [101, 9, 0.0], [101, 10, 0.0], [101, 11, 0.0], [101, 12, 0.0], [101, 13, 0.0], [101, 14, 0.0], [101, 15, 0.0], [101,
```

So, you can see here in some cases we have here this value here 0.09998, so since it is the very beginning of our codes. So, initially we had seen that the common neighbors only exist between, so there is no homophily as of now; we have commented the function homophily. The common neighbors will exist only when 2 people are the part of same social foci.

So, you can see here if you look at the nodes here 100, 87 and then 100, 74; 100, 56. So, these are mostly the people node, which are connected to the same social foci and this is the probability of these 2 nodes getting connected with each other. So, we apply a closure here, it is mostly since there is no edge between 2 people; it is mostly going to be happening a foci closure only. So, here was this array1; now we must make edges in accordance with this array1.

(Refer Slide Time: 15:21)

```

revolutionary_model.py (~) - gedit
*revolutionary_model.py x
K=cmn
(u,v,G)
for each in array1:
    u= each[0]
    v=each[1]
    p=each[2]
    r= random.uniform(0,1)
    if r<p:
        G.add_edge(u,v)
G=create_graph()
assign_bmi(G)

```

So, that is very simple; so, we go for each in array1; so, for every entry in array1; what do we do; our first value, first node is each 0 and then our second node is each1 and then our probability of connection is each 2. As we have done before, we choose our random real number from 0 to 1;  $r = \text{random.uniform}(0, 1)$  and if  $r < p$ , we add an edge  $G.\text{add\_edge}$ ; edge between  $u$  and  $v$ .

So, this is one iteration of our function closure let us execute and see it.

(Refer Slide Time: 16:10)

```
evolutionary_model.py (~) - gedit
evolutionary_model.py x
G.add_edge(u,v)

G=create_graph()
assign_bmi(G)
add_foci_nodes(G)
labeldict=get_labels(G)
nodesize=get_size(G)
color_array=get_colors(G)
add_foci_edges()

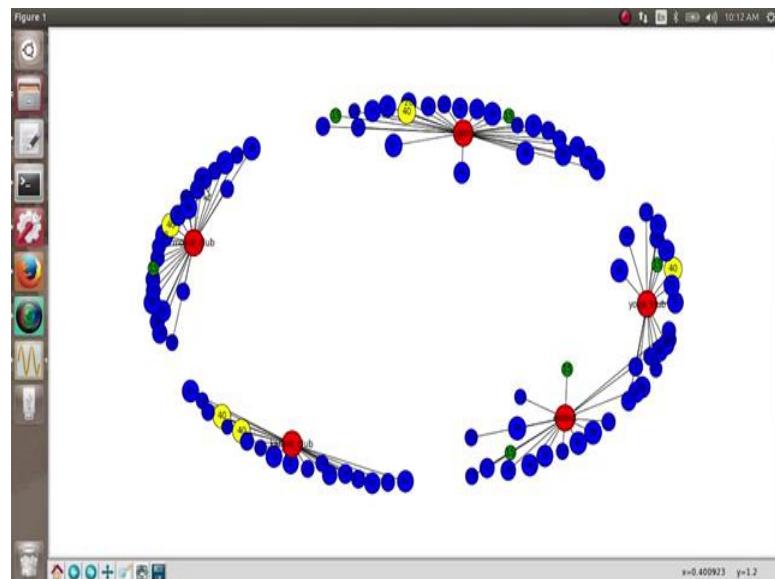
visualize(G,labeldict,nodesize,color_array)
#homophily(G)
closure(G)

visualize(G,labeldict,nodesize,color_array)

Saving file '/home/yayati/evolutionary_model.py...
Python • Tab Width: 8 • Ln 133, Col 1 1NS
```

So, here we will visualizing our graph as well, so this is our initial graph looks like; let us see what is going on here. So, actually there are no intersections here it has just, ok.

(Refer Slide Time: 16:13)

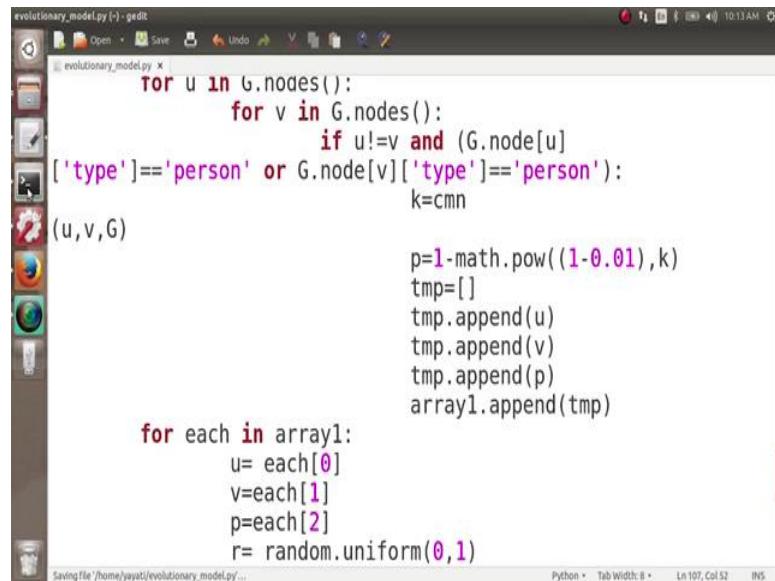


So, you can see here every node is actually the part of exactly one social foci here, we will zoom this graph and see and now this is what happens after closure. So, let us see what is happened after closure; let us look at this. So, you can see that there are more and more edges which have been added between people. So, 18 has now become connected

to 28; 28 has become connected to 24 because they were all the part of the same social foci gym.

So, you see how here the edges between people have been added, so as we can see here that probability of connection seems to be very high. We do not want this high probability of connection, so for that what we can actually do is; we go back and we deduce this probability here.

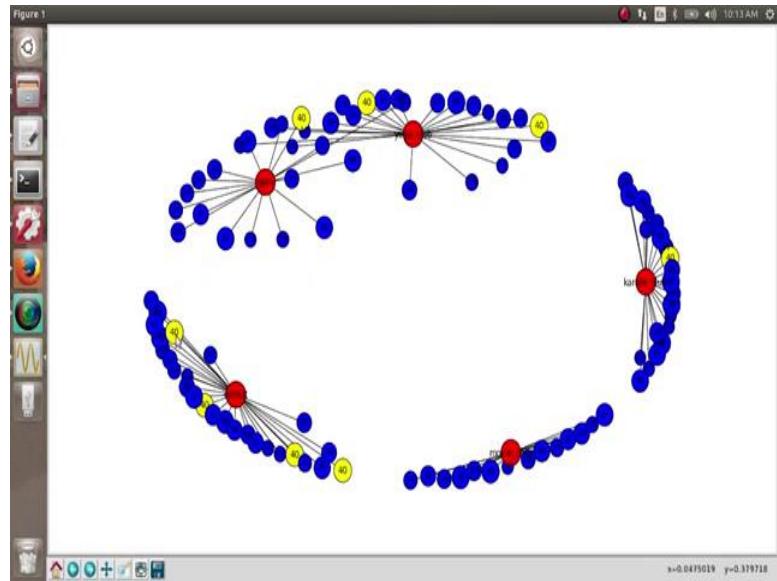
(Refer Slide Time: 17:22)



```
evolutionary_model.py (~) - gedit
evolutionary_model.py x
 10:11 AM
for u in G.nodes():
    for v in G.nodes():
        if u!=v and (G.node[u]
['type']=='person' or G.node[v]['type']=='person'):
            k=cmn
            (u,v,G)
            p=1-math.pow((1-0.01),k)
            tmp=[]
            tmp.append(u)
            tmp.append(v)
            tmp.append(p)
            array1.append(tmp)
for each in array1:
    u= each[0]
    v=each[1]
    p=each[2]
    r= random.uniform(0,1)
Saving file /home/yayati/evolutionary_model.py...
Python Tab Width: 8 Ln 107, Col 52 INS
```

So, let us say one common neighbor contributes 0.01 probability of connection. Let us now look at, so this is our initial graph and let us see what happens here now. Let us again pick 1 here; let us say we pick this karate club and then you can, now here see that the probability of connection has reduced. So, this karate club leads to the formation of some number of edges between 2 people.

(Refer Slide Time: 17:31)



We might want to see other kinds of closures as well, so till now here only 2 people were getting connected to each other because of having a common social foci.

(Refer Slide Time: 18:05)

```
evolutionary_model.py (-) - gedit
G.add_edge(u,v)

G=create_graph()
assign_bmi(G)
add_foci_nodes(G)
labeldict=get_labels(G)
nodesize=get_size(G)
color_array=get_colors(G)
add_foci_edges()

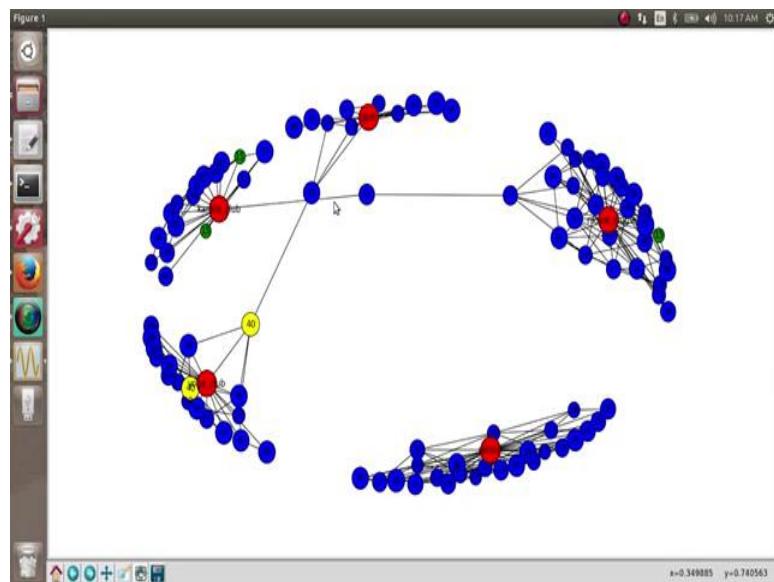
visualize(G,labeldict,nodesize,color_array)
homophily(G)
visualize(G,labeldict,nodesize,color_array)
while 1:
    closure(G)
    visualize(G,labeldict,nodesize,color_array)

Saving file '/home/yayati/evolutionary_model.py...
Python Tab Width: 8 Ln 134, Col 9 IHS
```

So, for that we would like to do homophily first so that we have some edges between the people as well and then we visualize and then we apply closer. So, for looking at, so let us for the meantime reduce this value and differentiating between the triadic closure and foci closure is a little bit difficult because there we edge edit between 2 people only. We would like to look at the membership closure, where one person becomes a part of social

foci because his friend is a part of social foci. So, let us try to look at that; we put this coding in a loop. So, while one closure happens and it keeps visualizing it again and again; now this is homophily, so, this is the graph. Now you can see here, this node 35; it is going to the karate club as well as it is going to the gym.

(Refer Slide Time: 19:02)



So, this node 35; it has 2 social foci is here in which it is participating, but according to our code initially node was assigned to exactly one social foci. So, most probably 35th node was initially a part of karate club only as we see then it has identified that one of its friends 1920 or somebody was going to gym and hence 35 also decides to go and be a member of gym. So, we can actually see a membership closure also happening here; you can write this code, play around with it and look and verify it further and it will keep happening till I do not stop it.

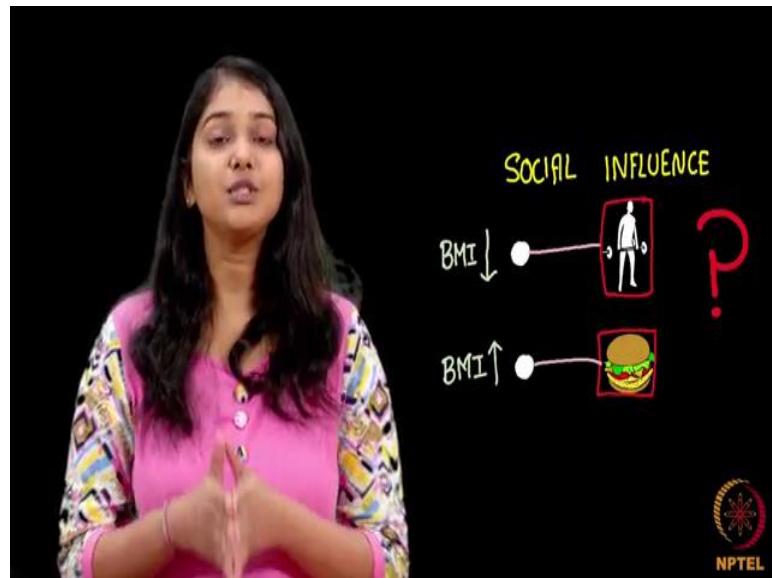
So, let us just stop it here control z; I will stop this code here. So, till now what we have done is; we have made a graph having 100 nodes with different BMIs where every node is part of exactly one social foci. Then we implemented homophily here and then we implemented closures here and we implemented all 3 kinds of closures; triadic closure, foci closure and membership closure and we have also seen them happening on this graph.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture – 52**  
**Fatman Evolutionary Model – Implementing Social Influence**

Finally, we have implemented homophily, we have seen how do implement three kinds of closures scenario evolutionary model and the last and the most interesting thing which remains is the social influence and it is actually the easiest. We need not do anything here in this model to achieve social influence, as I have told before.

(Refer Slide Time: 00:20)



So, there is a very subtle way in which we capture these social influence happening; how do we do that is, assume that if a person is connected to a gym so; obviously, he is being connected to gym should result in a decrease in his BMI and if a person is connected to an eat out place, they should result in increase in the BMI of this person.

So, what do we do; for every node which is connected to a gym, we reduce its BMI by 1; every iteration and every node which is connected to an eat out place, we increase its BMI by one every iteration and we just do this and the social influences captured; How? So, as I have told you; we have seen here now, when a person gets connected to a gym; he is losing weight. Now, I have a friend and this friend is going to the gym; now

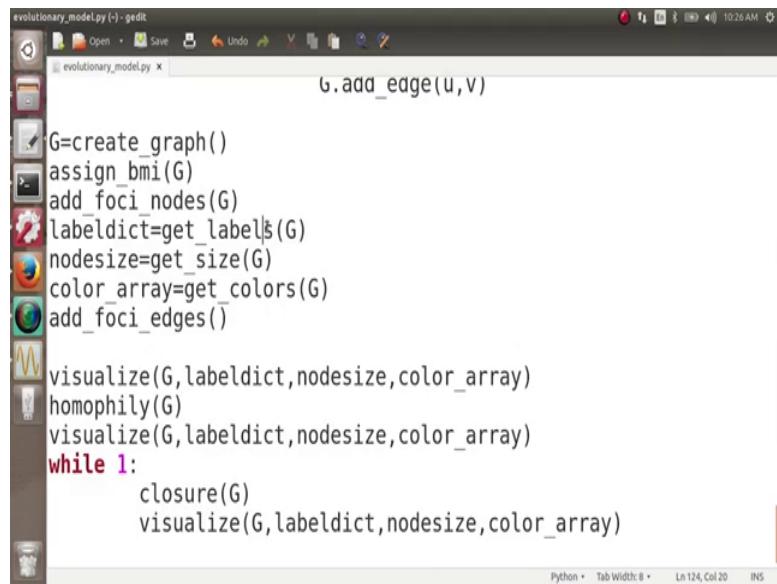
because of membership closure, I tend to become a part of the gym and when I become a part of the gym; as you saw I will start losing weight.

(Refer Slide Time: 01:22)



So, this is one aspect of social influence, so we have mainly looked at three reasons why in which social influence can happen, but we capture one of those here and that reason is the sheared context. When 2 people are the part of the same social foci, it leads to an influence. So, we have seen that it is like; if 2 people both have a brilliant teacher, both of them will start having good marks. Similarly, if 2 people here both of them are part of gym; so one person who was going to gym, another also started going to gym will start losing weight and similarly with an eat out place; the effect is reversed, so this is how we capture social influence here.

(Refer Slide Time: 02:25)



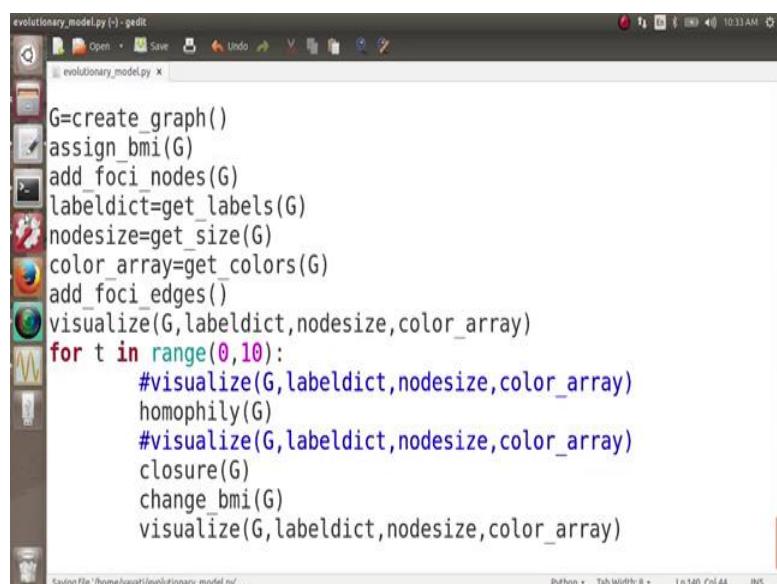
```
G.add_edge(u,v)

G=create_graph()
assign_bmi(G)
add_foci_nodes(G)
labeldict=get_labels(G)
nodesize=get_size(G)
color_array=get_colors(G)
add_foci_edges()

visualize(G,labeldict,nodesize,color_array)
homophily(G)
visualize(G,labeldict,nodesize,color_array)
while 1:
    closure(G)
    visualize(G,labeldict,nodesize,color_array)
```

Finally, what we want to do here is; we want to implement social influence. How we are implementing social influence? We have already discussed before; well it is an indirect implementation of social influence. So, it is mainly at every iteration a person who is associated with gym; lose his weight and a person will associated with an eat out, gains weight and then as we know that membership closure is happening. So, let us say I am a part of gym and then I have a friend, who is fat, then he looks at me and I have joined gym, he also joined gym and hence he started losing weight. So, this the kind of social influence we are going to implement here.

(Refer Slide Time: 03:18)



```
G=create_graph()
assign_bmi(G)
add_foci_nodes(G)
labeldict=get_labels(G)
nodesize=get_size(G)
color_array=get_colors(G)
add_foci_edges()
visualize(G,labeldict,nodesize,color_array)
for t in range(0,10):
    #visualize(G,labeldict,nodesize,color_array)
    homophily(G)
    #visualize(G,labeldict,nodesize,color_array)
    closure(G)
    change_bmi(G)
    visualize(G,labeldict,nodesize,color_array)
```

So, for meanwhile let us just comment everything and now what we want to do here is; we want to change weights, we want to change BMIs of people.

(Refer Slide Time: 03:41)

```
if r<p:
    G.add_edge(u,v)

def |change_bmi(G):
    fnodes=get_foci_nodes()
    for each in fnodes:
        if G.node[each]['name']=='eatout':
            for each1 in G.neighbors(each):
                if G.node[each1]['name']!=40:
                    G.node[each1]
['name']=G.node[each1]['name']+1
        if G.node[each]['name']=='gym':
            for each1 in G.neighbors(each):
                if G.node[each1]['name']==15:
                    G.node[each1]
['name']=G.node[each1]['name']-1
```

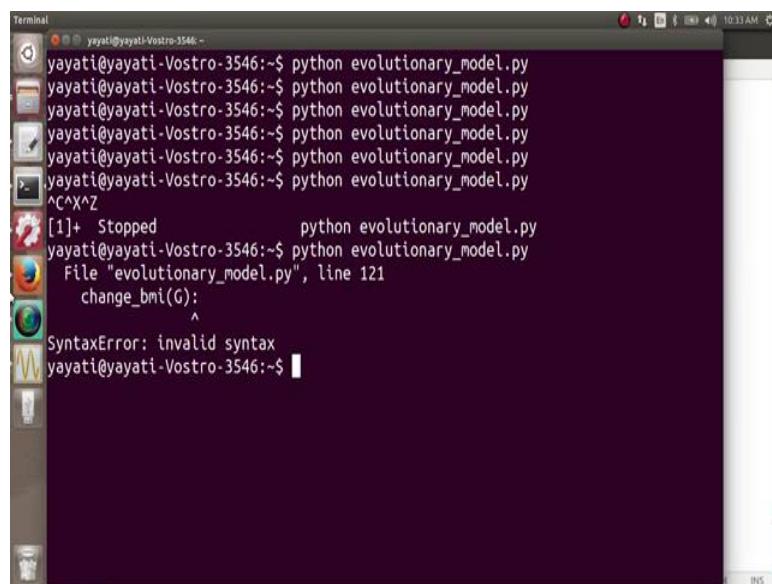
So, we called `change_bmi(G)` and what this function `change_bmi(G)` does is `change_bmi(G)`. So, for each in `f nodes`; what do we do for `each1 in G.neighbors(each)`; what do we do, if `G.node[each1]`, but again we are doing it for the foci node which are either an eat out or a gym. So, we want to see that so for each in `f nodes`; we again need to put a node here, if `G.node[each]`. So, if this each node its name; its name == ‘eat out’.

If its name = ‘eatout’ only then we are going to insecure this code, so if its eat out we look at each of its neighbors and if each `G.node[each1][‘name’]`. So, now since we are going to increase the weight, first this name should not be equals to 40 because if it is equal to 40, we cannot further increase the BMI of this person. So, if each if `G.node[each1]; name != 40`; what do we do is we increase it by 1.

`G.node[each1], name = G.node[each1][‘name’] + 1` and then we again do this same thing for gym. It is essential we have to copy paste the same code here, if `G.node each, name = ‘gym’`. So, if it is a gym we look at all of its neighbors and if for a neighbour this value should not be equals to 15; if this value is not equals to 15; we decrease its BMI by 1, so this is how we are going to change the BMI of people.

Now, what are we going to do is; we have all three components in our network. We have homophily, we have closure and we have social influence. We are going to put all of these three together and we are going to visualize our network. So, let us take time for t in range; let us do it over a time period of; let us do it for 10-time instants. So, for t in range 0 to 10; what are we going to do is, we do homophily we apply (Refer Time: 07:00) closure, we apply social influence and then we visualize this graph and add the beginning when nothing is done that time also we want to visualize our graph. So, we have the statements here; now let us execute our code and see what is happening.

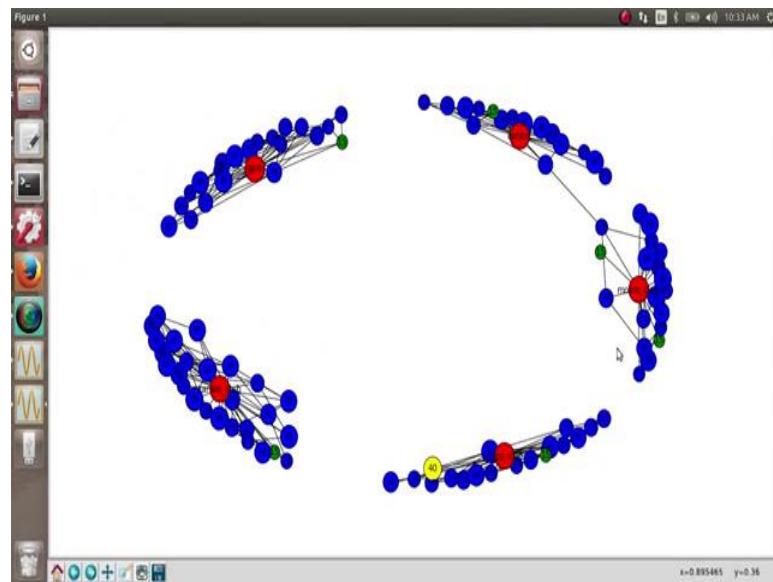
(Refer Slide Time: 07:16)



```
Terminal
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
^C^X^Z
[1]+  Stopped                  python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
File "evolutionary_model.py", line 121
    change_bmi(G):
          ^
SyntaxError: invalid syntax
yayati@yayati-Vostro-3546:~$
```

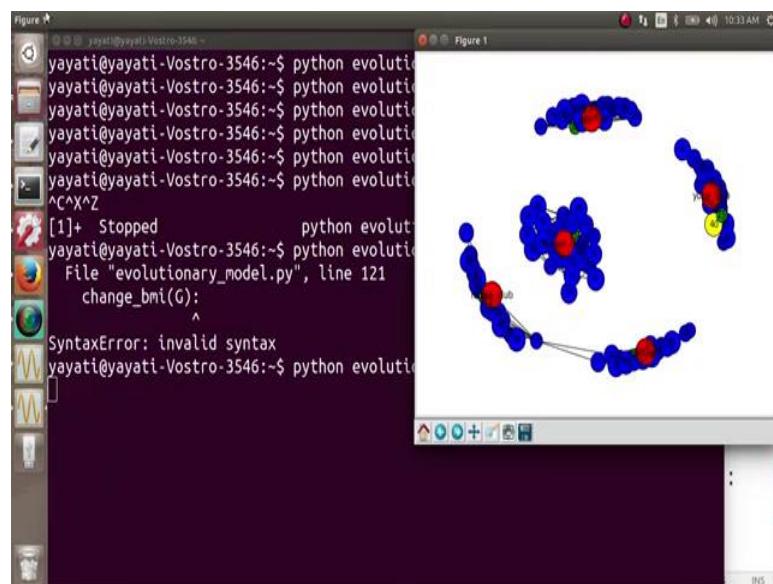
change\_bmi(G); so, this is a function definition; so, we need a define here; go back and secure it. So, this is our initial network where we have five social focus and we have different nodes; I mean different BMIs.

(Refer Slide Time: 07:31)



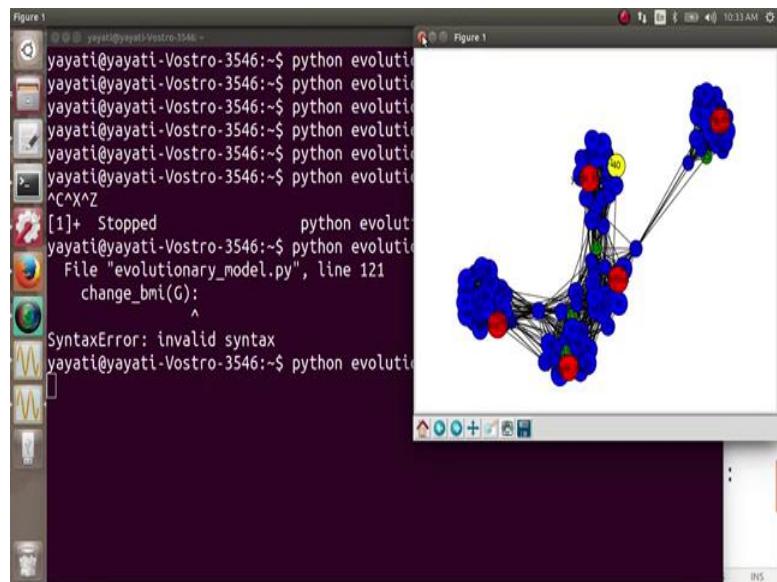
Now, we see here that some nodes have become friends with each other and there are more number of edges between these nodes and let us look at the number of obese people in this network, so currently we have one obese node which is 40.

(Refer Slide Time: 08:00)



We again have one obese node; which is 40 and is a greater number of nodes are getting added; Do you see some problem in this code? I see some problem in this code.

(Refer Slide Time: 08:10)

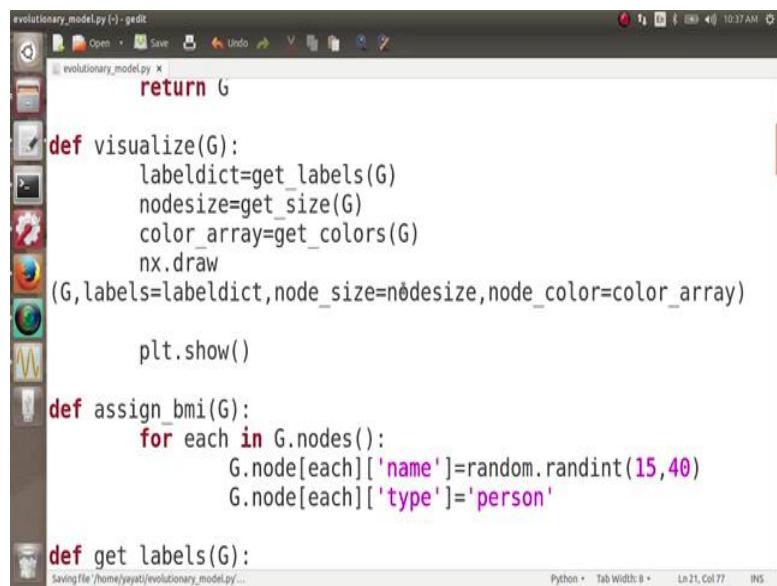


So, as we can see the number of obese people is now changing in this network that should actually happen because we have implemented social influence. So, what can be the problem with this code, so let us try to troubleshoot it together when we write a piece of code, we should be able to troubleshoot this code nicely? So, you will see here when we were visualizing a graph  $G$  here; we have passed these various arrays here; color array, node size, label dict. So, here when we have here when the BMI of people have changed and then the social influence has occurred. Do not you think we need to again calculate all these values?

So, it is like in the beginning we saw which nodes were yellow, we faded it here and we visualize we fed it here and we visualized it and now when we are running this graph; we are again and again using the same color array, the same size array; so, we need to change this. So, best way to change is to have these values; took all these functions inside our visualize.

So, this functions like let us say labeldict function and then the node size function and the color array function. So, all these three things need to be calculated in the visualize function because whenever you visualize it, these values are keeping, these value that changing. So, what we do is we just delete these values from here and we put all these values inside all these function calls; inside a visualize function. So, here is our function visualize; so, what we do here is we have all these values here.

(Refer Slide Time: 10:04)



```
evolutionary_model.py (~) - gedit
evolutionary_model.py x
return G

def visualize(G):
    labeldict=get_labels(G)
    nodesize=get_size(G)
    color_array=get_colors(G)
    nx.draw
    (G,labels=labeldict,node_size=nodesize,node_color=color_array)

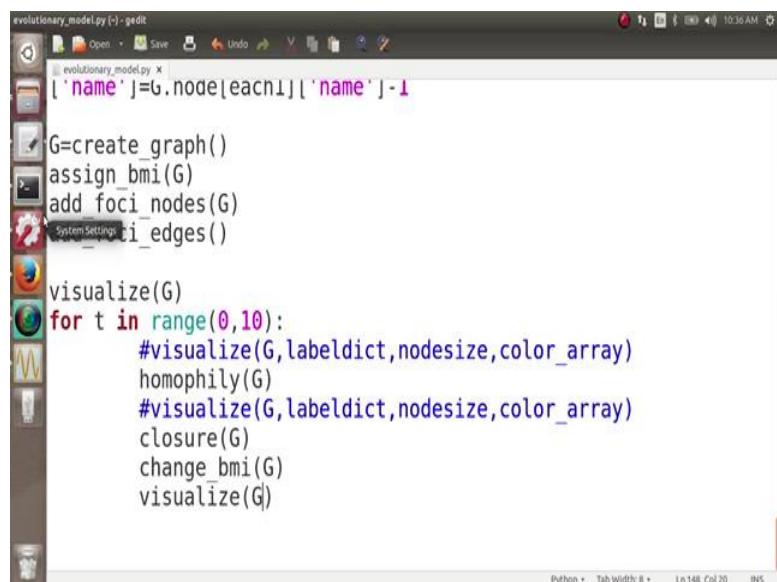
    plt.show()

def assign_bmi(G):
    for each in G.nodes():
        G.node[each]['name']=random.randint(15,40)
        G.node[each]['type']='person'

def get_labels(G):
Saving file /home/yayati/evolutionary_model.py...
Python Tab Width: 8 Ln 21, Col 77 INS
```

So, whenever we want to visualize a graph; we again look at the labels, we again look at the size of nodes and we again look at the colors and then we do not need to have all these things here, we just have a define visualize(G) and when we call these function towards the end, there also we just need a visualize(G). So, whenever you will try to code sometimes when weird things will happen. So, the best ways to keep printing your different parameters at different places and see what is going wrong where.

(Refer Slide Time: 10:32)



```
evolutionary_model.py (~) - gedit
['name']=G.node[each]['name']-1

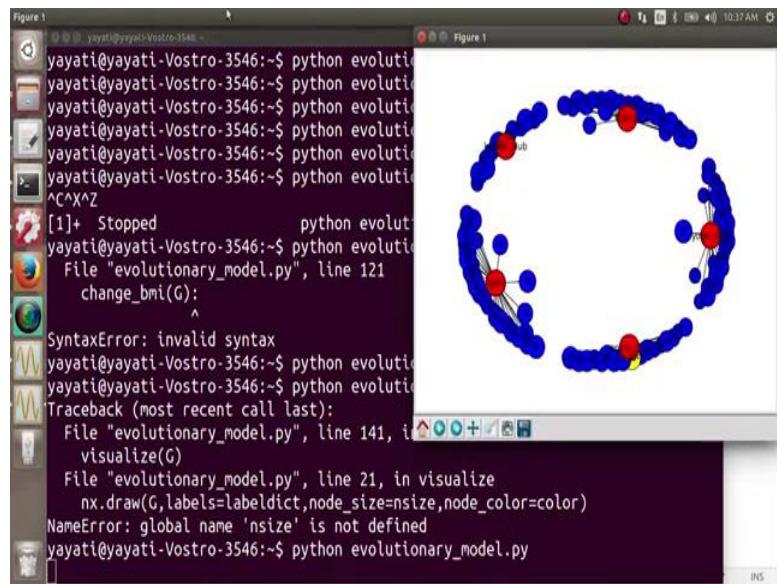
G=create_graph()
assign_bmi(G)
add_foci_nodes(G)
System_Settings_i_edges()

visualize(G)
for t in range(0,10):
    #visualize(G,labeldict,nodesize,color_array)
    homophily(G)
    #visualize(G,labeldict,nodesize,color_array)
    closure(G)
    change_bmi(G)
    visualize(G)

Python Tab Width: 8 Ln 148, Col 20 INS
```

Now, let us again execute it.

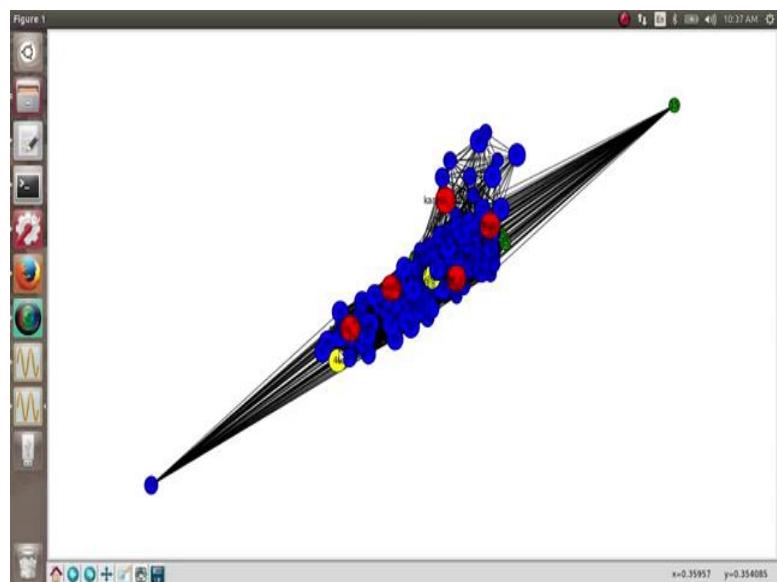
(Refer Slide Time: 10:21)



```
Figure 1
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
^C^X^Z
[1]+  Stopped                  python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
File "evolutionary_model.py", line 121
    change_bmi(G):
          ^
SyntaxError: invalid syntax
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
Traceback (most recent call last):
  File "evolutionary_model.py", line 141, in <module>
    visualize(G)
  File "evolutionary_model.py", line 21, in visualize
    nx.draw(G,labels=labeldict,node_size=nsize,node_color=color)
NameError: global name 'nsize' is not defined
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
```

Line 21; there seems to be a problem, so let us go to the line 21, so we have calculated labeldict here, we have calculated node size here. So, instead of n size here; it should be node size and then node under scroll color is color under scroll array node color. Now, let us look at this graph, so this is our initial graph and then you see some 2 friendships have been added to this graph.

(Refer Slide Time: 11:56)



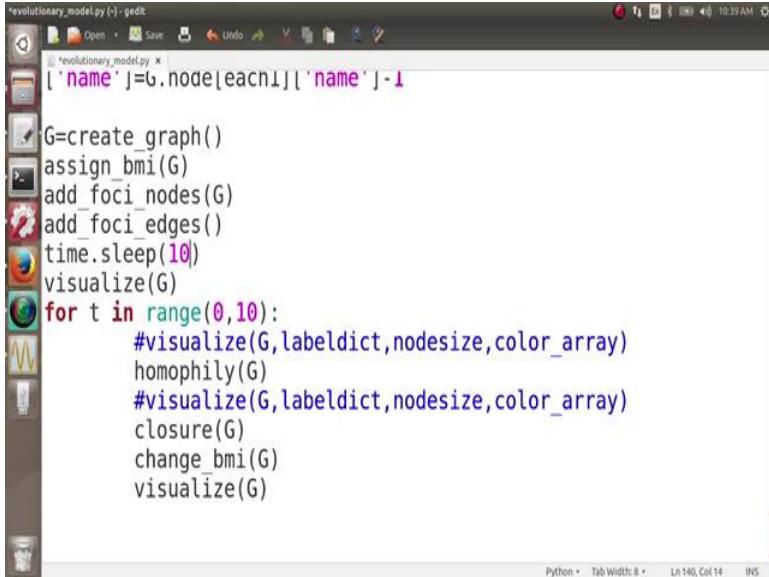
So, now you see there is one yellow node and there is another yellow node, so you see that the BMI of one node has been increased and it is hit the obesity parameter. So, this

how a graph will keep changing; to have a battle visualization, we can actually change the parameters, we have used in the code; which you will see that with time more and more number of people will keep becoming obese because of social influence and yes people will use their weights also because of this.

So, let us quickly have a look on the parameter, so, this was for homophily and then this is for closure. So, let us keep it small for the time being let us keep it 0.01 and then we can again visualize this graph.

Now, let us visualize this graph in a very nice way, so what we are going to do for this visualization is we store this graph in a file jpg file and when we run this code at jpg file, we will keep changing with time and we observe that change in that jpg file. So, for that what do we do is; when you call this function visualize(G), this function will store your graph; will save your graph as a jpg file; instead of plotting it here.

(Refer Slide Time: 13:19)



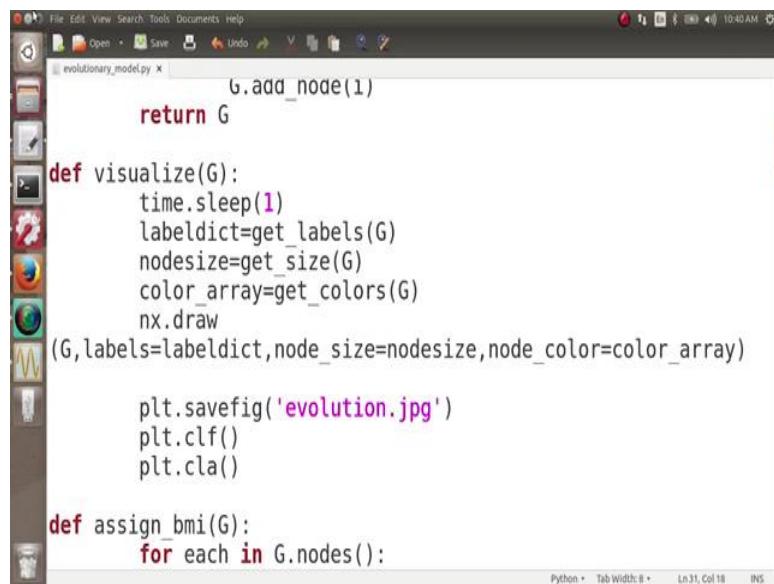
The screenshot shows a Gedit text editor window titled "revolutionary\_model.py". The code in the editor is as follows:

```
L['name']=G.node[each1]['name']-1
G=create_graph()
assign_bmi(G)
add_foci_nodes(G)
add_foci_edges()
time.sleep(10)
visualize(G)
for t in range(0,10):
    #visualize(G,labeldict,nodesize,color_array)
    homophily(G)
    #visualize(G,labeldict,nodesize,color_array)
    closure(G)
    change_bmi(G)
    visualize(G)
```

The status bar at the bottom of the editor indicates "Python" as the language, "Tab Width: 8", "Ln 140, Col 14", and "INS" for insert mode.

Now, first let us put a sleep statement here so that before making your file, the code sleeps for some time.

(Refer Slide Time: 13:39)



```
File Edit View Search Tools Documents Help
evolutionary_model.py X
    G.add_node(1)
    return G

def visualize(G):
    time.sleep(1)
    labeldict=get_labels(G)
    nodesize=get_size(G)
    color_array=get_colors(G)
    nx.draw
    (G,labels=labeldict,node_size=nodesize,node_color=color_array)

    plt.savefig('evolution.jpg')
    plt.clf()
    plt.cla()

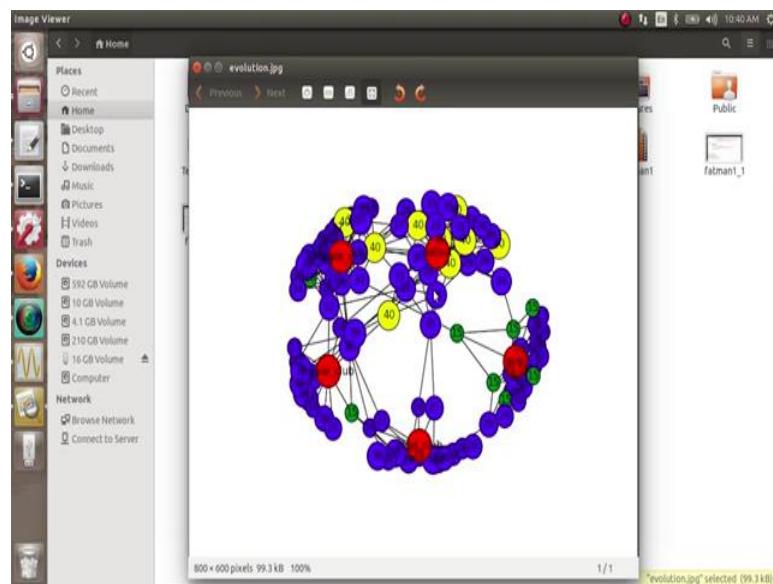
def assign_bmi(G):
    for each in G.nodes():

Python Tab Width: 8 Ln 31, Col 18 INS
```

So, let us say some 10 seconds and whenever you call your function; `visualize(G)`, what this function is going to do is; it will wait for let us say 1 second and for this, use this sleep statement we need to import time. It may put some delays in your code, so `time.sleep(1)` and here instead of showing our plot, what do we do is; we save our figure `plt.savefig` and let us say `evolution.jpg`.

So, this plot is saved and once this plot is saved; we need to clear this plotting screen for the next plotting to occur. So, what we have done is; we have removed the `plt dot show` statement here and whatever the output graph is, we are having that output graph in a separate file and that separate file keeps changing with time.

(Refer Slide Time: 14:50)



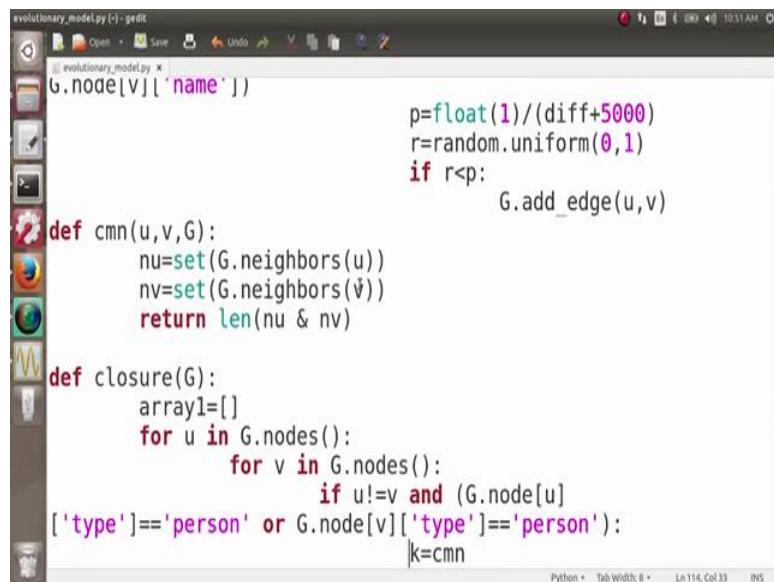
Let us execute this code now and see what is happening. So, let us come back, so here we are and let us wait. So, here is our file evolution dot jpg let us open it and you can see that now this code has started changing with time. Now this figure has started changing with time and you can see that how more and more number of yellow nodes are popping up in the network; how more and more number of green nodes are popping up in the network, which shows that with time the number of obese people in this network is increasing and with time the number of underweight people in this network is also increasing, so it mainly depends on how you execute this code.

So, this was one way of looking at this animation; there actually milli nines python functions to look at this animations. So, for the sake of simplicity we have done it like this time, but next time we will try to use a better approach, we will try to use a inbuilt functions of python.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture - 53**  
**Strong and Weak Relationships (Continued) and Homophily**  
**Fatman Evolutionary Model - Storing and analyzing longitudinal data**

(Refer Slide Time: 00:05)



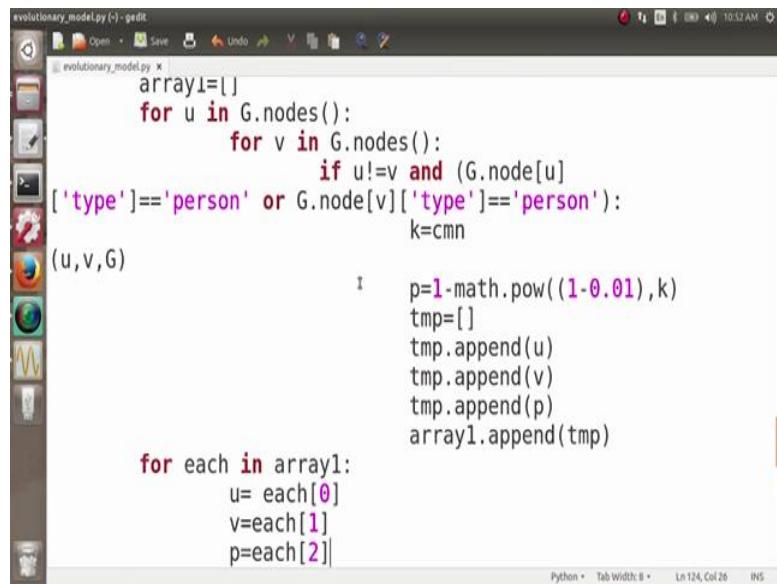
```
evolutionary_model.py (~) · gedit
evolutionary_model.py *
G.nodes[v]['name'])
p=float(1)/(diff+5000)
r=random.uniform(0,1)
if r<p:
    G.add_edge(u,v)

def cmn(u,v,G):
    nu=set(G.neighbors(u))
    nv=set(G.neighbors(v))
    return len(nu & nv)

def closure(G):
    array1=[]
    for u in G.nodes():
        for v in G.nodes():
            if u!=v and (G.node[u]
['type']=='person' or G.node[v]['type']=='person'):
                k=cmn
                array1.append(k)
                print k
```

Next what we want to do assume we want to analyze what all is happening in this graph. So, it is not actually possible to analyze it very nicely by visualizing the graph and looking at it as a figure every time. So, some nicely written functions can help us in the analysis of this graph.

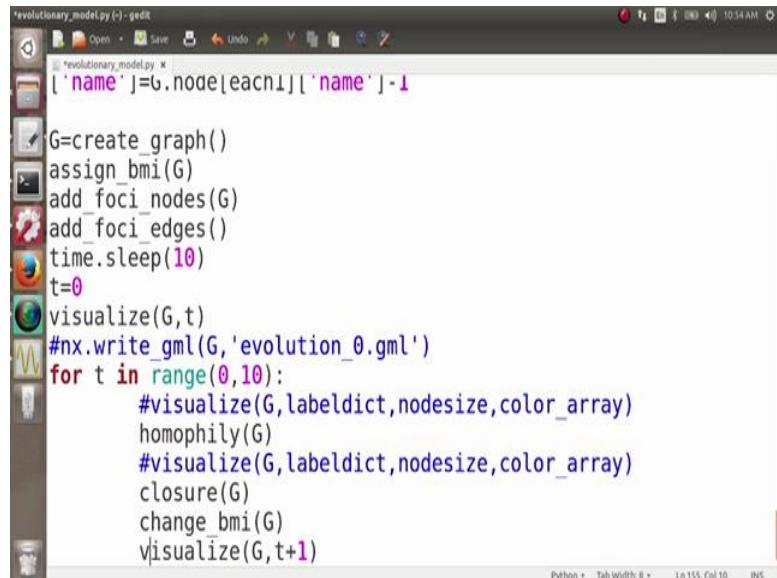
(Refer Slide Time: 00:25)



```
evolutionary_model.py (~) - gedit
array1=[]
for u in G.nodes():
    for v in G.nodes():
        if u!=v and (G.node[u]['type']=='person' or G.node[v]['type']=='person'):
            k=cmn
            (u,v,G)
            p=1-math.pow((1-0.01),k)
            tmp=[]
            tmp.append(u)
            tmp.append(v)
            tmp.append(p)
            array1.append(tmp)
for each in array1:
    u= each[0]
    v=each[1]
    p=each[2]
```

So, for that what we will do is, first of all we will store whatever is happening in this graph as different snapshots. So, we will have different network store for different snapshots, next we will look at these networks one by one and look at how this network is changing over time.

(Refer Slide Time: 00:41)



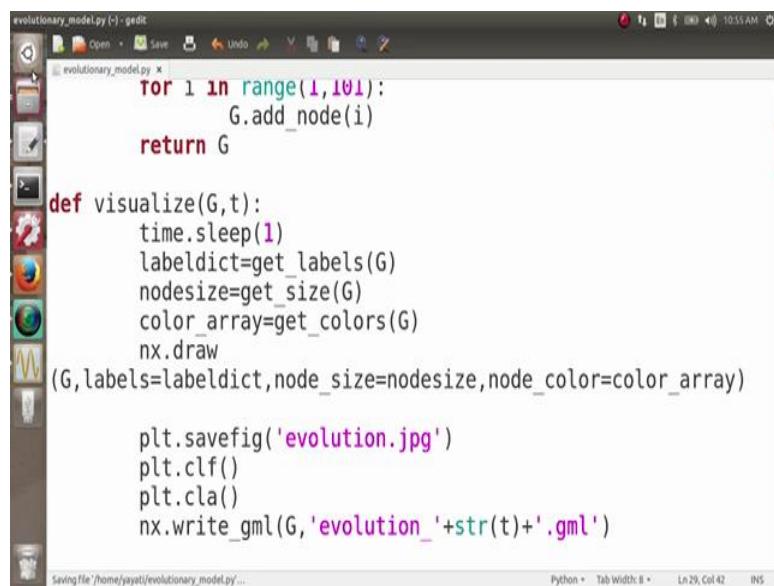
```
evolutionary_model.py (~) - gedit
l['name']=b.node[each1]['name']-1
G=create_graph()
assign_bmi(G)
add_foci_nodes(G)
add_foci_edges()
time.sleep(10)
t=0
visualize(G,t)
#nx.write_gml(G,'evolution_0.gml')
for t in range(0,10):
    #visualize(G,labeldict,nodesize,color_array)
    homophily(G)
    #visualize(G,labeldict,nodesize,color_array)
    closure(G)
    change_bmi(G)
    visualize(G,t+1)
```

So, what we do here is along with visualizing G we try to store this network. So, we call a function here let us say store G, and this store G takes a time value with itself let us say its t or let us say the time value is i. So, initially this i is 0 when nothing is done. So, we

call store G , i, and a better option is when we visualize this graph that time only, we can store this network what do we do is. So, for the very first of all outside the loop we store it manually. So, for storing it we use we stored it in a gml format. So, for storing it in a gml format we use the function nx.write\_gml as you know we have done it before nx.write\_gml(G) and let us say the file name is gain evolution\_0.gml. So, this is a first file name.

Next when the time runs from 0 to 10 what we are doing is we are calling these three functions and we are visualizing this network. So, let us pass the value t + 1 here. So, why t + 1? t was here t starts from 0, but we want the indices of this network from 1. So, that is evolution\_1.gml, evolution\_2.gml. So, we have here visualize (G , t + 1) and actually we should be doing it here also because the format of the function is same. So, we need to pass the same parameters here. So, let us first have t = 0 and then we call visualize(G , t) you can actually come and take and then our t then we called visualized (G , t + 1).

(Refer Slide Time: 02:56)



```

evolutionary_model.py (~) - gedit
Saving file /home/jayati/evolutionary_model.py...
10:55 AM
evolutionary_model.py x
for t in range(1,101):
    G.add_node(i)
    return G

def visualize(G,t):
    time.sleep(1)
    labeldict=get_labels(G)
    nodesize=get_size(G)
    color_array=get_colors(G)
    nx.draw
    (G,labels=labeldict,node_size=nodesize,node_color=color_array)

    plt.savefig('evolution.jpg')
    plt.clf()
    plt.cla()
    nx.write_gml(G,'evolution_'+str(t)+'.gml')

Python Tab Width: 8 Ln 29, Col 42 IN5

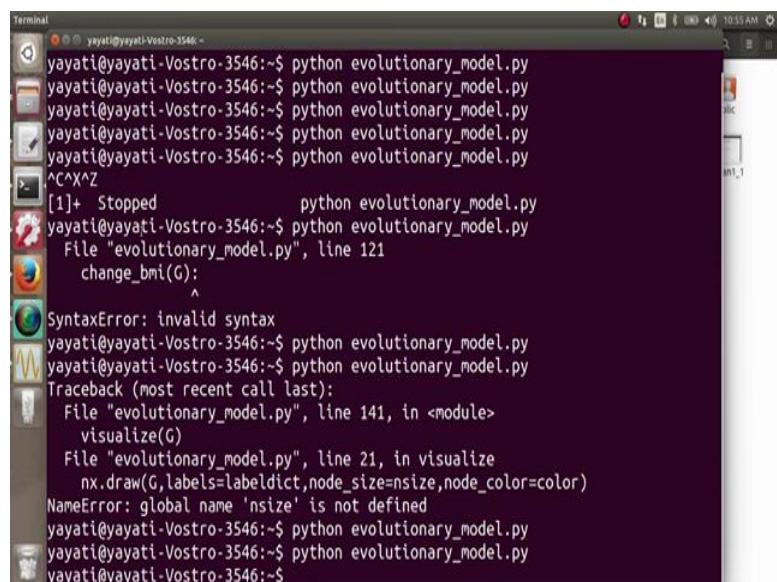
```

And what happens here coming back to our visualization function. So, it now it is taking an extra parameter which is the t at which the time t the graph is to be stored. So, what we do here is to store the graph after doing all this, we have a function here nx.write\_gml and then we write this graph G as what should be its name its name should be evolution underscore, and then we have this value of t here. So, we do a string append operation

and what do we append to this string is the value of t as a string. So, it is evolution\_0 then evolution\_1 and we append.gml to it.

So, how this graph gets stored is G evolution\_0.gml, evolution\_1.gml and so on and then will be loading this graphs and we look at some properties of this evolution of this completes evolution. So, let us see let us try to run it.

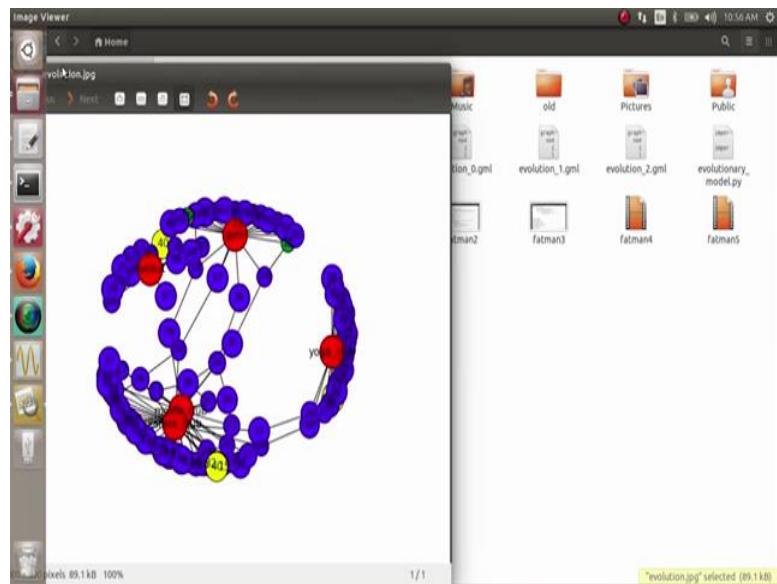
(Refer Slide Time: 04:03)



```
Terminal
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
[1]+  Stopped                  python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
File "evolutionary_model.py", line 121
    change_bmi(G):
          ^
SyntaxError: invalid syntax
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
Traceback (most recent call last):
  File "evolutionary_model.py", line 141, in <module>
    visualize(G)
  File "evolutionary_model.py", line 21, in visualize
    nx.draw(G,labels=labeldict,node_size=nsize,node_color=color)
NameError: global name 'nsize' is not defined
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$
```

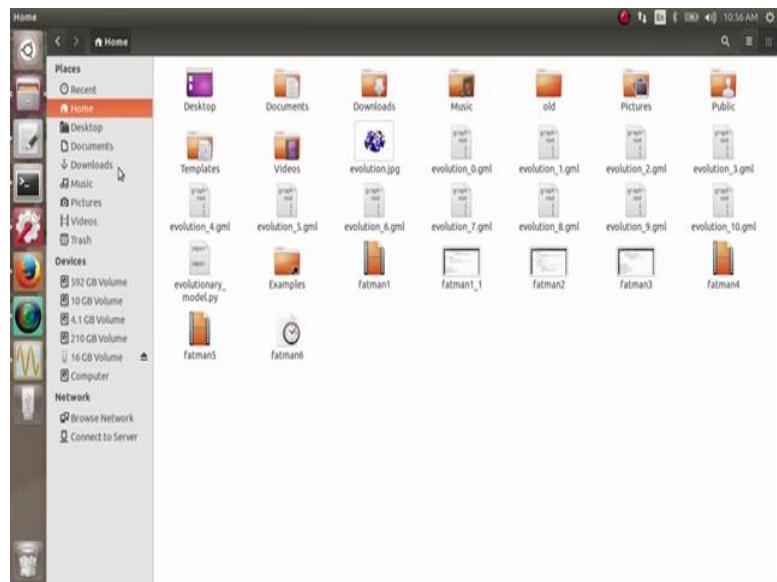
So, let us see how this graph gets generated here we let us (Refer Time: 04:06) python evolutionary\_model.py and let see what happens here. So, we have seen as before this file evolution.jpg is going to change with time it will keep changing.

(Refer Slide Time: 04:16)



So, and if you see here let us just let us just close it. So, you see here that we have here these different `gml` files for me `evolution_0.gml`, `evolution_1.gml`, `2.gml`.

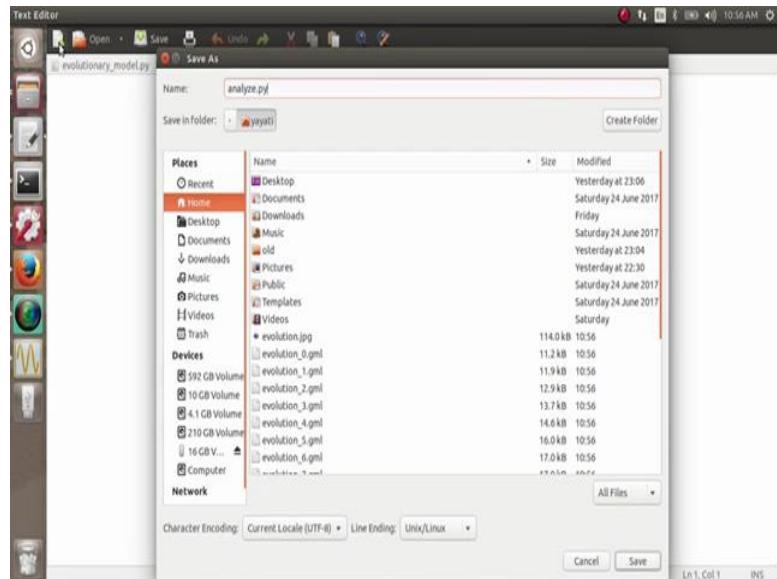
(Refer Slide Time: 04:24)



So, we have done it over some ten snapshots. So, you can do it over as many number of snapshots as you want. So, here we have 10 networks here I do not say ten networks it is the same network over 10 times stumps. So, it is my initial network there is some homophily some closure etcetera are happening on this network and then we have these different `gml` files coming up here.

Now, I want to analyze these files why I want to analyze how this how the change in this graph is happening. So, to analyze this file we create a new file.

(Refer Slide Time: 05:10)



And let us name it analyze.py, and what I want to do in analyze.py see I want to analyze three things I want to see how the number of obese people in the network change then I want to see how the number of edges between these obese people change, and then I want to look at the change in density on this network. Let us keep it simple for the timing let just look at 2 things.

(Refer Slide Time: 05:44)

```
import networkx as nx
import matplotlib.pyplot as plt

def plot_density():
    x=[]
    y=[]
    for i in range (0,11):
        G=nx.read_gml('evolution_'+str(i)+'.gml')
        x.append(i)
        y.append(nx.density(G))
    plt.xlabel('Time')
    plt.ylabel('Density')

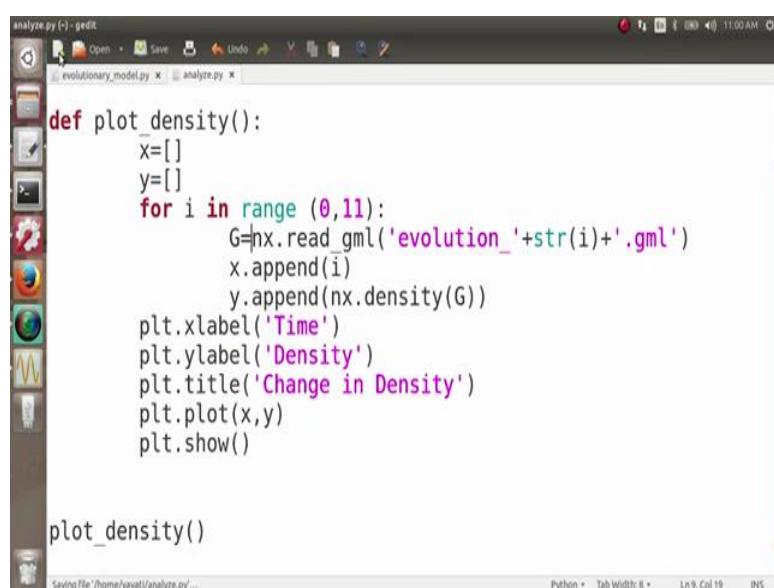
plot_density()
```

Let us look at how the number of obese people change and how the density of these networks changes. So, we need to import network x as nx and we also need to import matplotlib.py plot as plt.

And next what we want to do is let us say our function is let us have our function plot density. So, what plot density does? It plots how the density in a network is changing. So, we defined here plot underscore density and what this function is going to do is. So, we need have. So, you know that for plotting we need an x axis, we need a y axis; x axis here is the iterations and y axis here is the density of these different networks. So, our x axis will be the iterations and our y axis will be the density.

So, what we do is we know that for i in range we have 10 snapshots. So, we take from range 0 to 11 we need a graph here. So, what is our graph G = nx.read\_gml and what is the file name? File name we know is evolution under score and whatever is the value of i plus and we have here gml. When we read this file and then we want to look at the density of this network. So, what was in the x axis? x.append its very simple. So, we append i here and what do we appending y is the density of this network. So, we have a direct function for density let us call that nx.density(G). In y we append the density of this graph. So, we have an x axis, we have a y axis and at the end we plot both of these or we can actually put the titles for the plot plt.xlabel and let us see it is the time and then we can have plt.ylabel.

(Refer Slide Time: 08:13)



```
analyze.py (-) - edit
analyze.py evolutionary_model.py analyze.py

def plot_density():
    x=[]
    y=[]
    for i in range (0,11):
        G=nx.read_gml('evolution_'+str(i)+'.gml')
        x.append(i)
        y.append(nx.density(G))
    plt.xlabel('Time')
    plt.ylabel('Density')
    plt.title('Change in Density')
    plt.plot(x,y)
    plt.show()

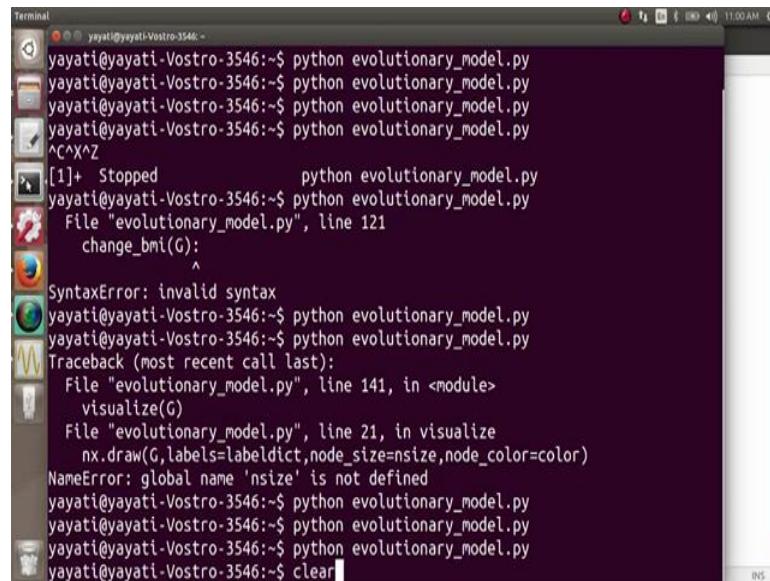
plot_density()

Saving file '/home/yayat/analyze.py...
Python Tab Width: 8 Ln 9, Col 19 INS
```

And let us have it as density, and then we have a title for this plot will it change in density; and then what we do is plt.plot(x, y) as we do, and then we do a plt.show.

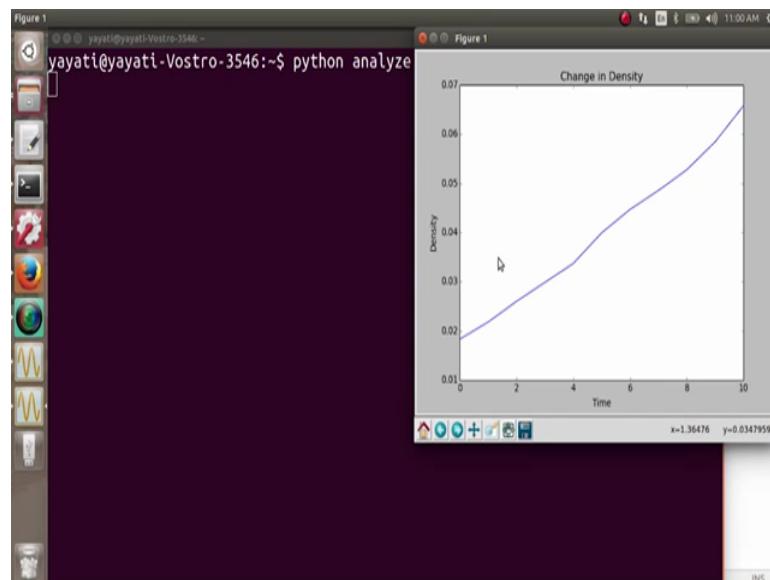
So, this is simple right we have these 10 gml graphs we read each of these graph one by one look at their density and then we clued their density let us execute it for.

(Refer Slide Time: 08:47).



```
Terminal
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
^CAXZ
[1]+  Stopped                  python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
File "evolutionary_model.py", line 121
    change_bmi(G):
          ^
SyntaxError: invalid syntax
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
Traceback (most recent call last):
  File "evolutionary_model.py", line 141, in <module>
    visualize(G)
  File "evolutionary_model.py", line 21, in visualize
    nx.draw(G,labels=labeldict,node_size=nsize,node_color=color)
NameError: global name 'nsize' is not defined
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ clear
```

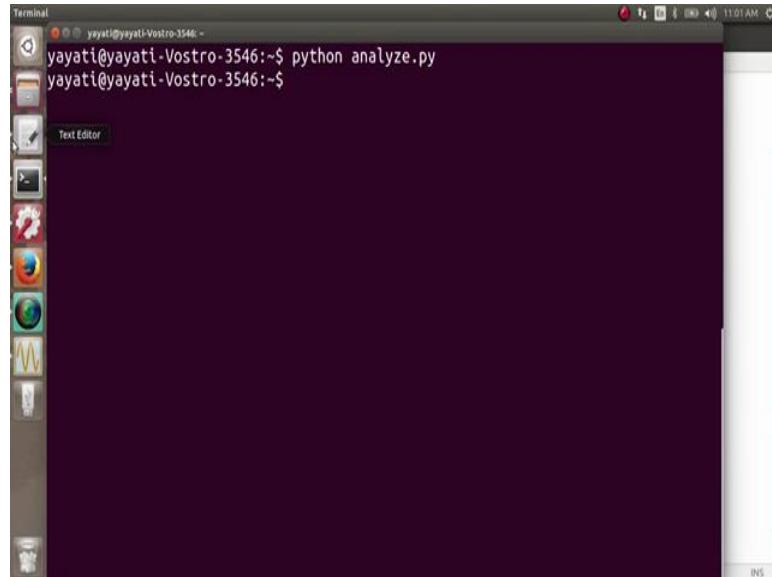
(Refer Slide Time: 08:50)



So, we have here python and we have analyze dot p y. So, you can see here how the density of this graph is changing. So, we trailed over ten iterations the density is increasing very fast, not we cannot say very fast densities increasing almost linearly this

is about the density of this network more and more links are being formed because of homophily and because of all these three different kinds of closures.

(Refer Slide Time: 09:17)



So, you can actually see whatever you want to see across this evolution with the while loading this graphs and looking at them.

(Refer Slide Time: 09:35)

A screenshot of a code editor window titled 'analyze.py (~) - gedit'. The code in the editor is as follows:

```
analyze.py (~) - gedit
plt.title('Change in Density')
plt.plot(x,y)
plt.show()

def plot_obesity():
    x=[]
    y=[]
    for i in range (0,11):
        G=nx.read_gml('evolution_'+str(i)+'.gml')
        x.append(i)
        y.append(obesity(G))
    plt.xlabel('Time')
    plt.ylabel('Obesity')
    plt.title('Change in Obesity')
    plt.plot(x,y)
    plt.show()
```

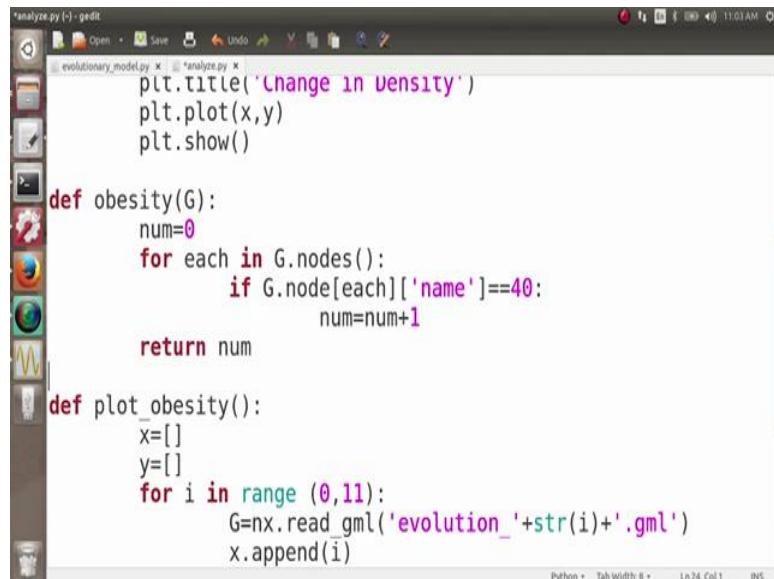
The code uses the NetworkX library to read graphs from files named 'evolution\_0.gml' through 'evolution\_10.gml'. It plots the 'obesity' value for each graph over time (index 0 to 10).

Let us say next I want to plot the obesity in this network and how do I plot the obesity by looking at the number of obese people in this network. So, we can write a similar code for this. So, we have defined plot underscore obesity and actually it is going to be almost

the same. So, let us just copy paste we use copy paste again and again in computer science. Define plot no (Refer Time: 10:10) computer science in coding define plot underscore obesity and we have this till here it is perfect.

Now, the problem is only here y dot append we have appended here the density of graph G, but what we want to append here is let us call our function obese and let us name this function as obesity. So, obesity is a function which will take as input your graph G and return you the number of obese people in this graph, and we just remove the density here by obesity, but yes we need to define this function obesity G.

(Refer Slide Time: 10:51)



```
analyze.py (~) - gedit
analyze.py x evolutionary_model.py x
plt.title('Change in Density')
plt.plot(x,y)
plt.show()

def obesity(G):
    num=0
    for each in G.nodes():
        if G.node[each]['name']==40:
            num=num+1
    return num

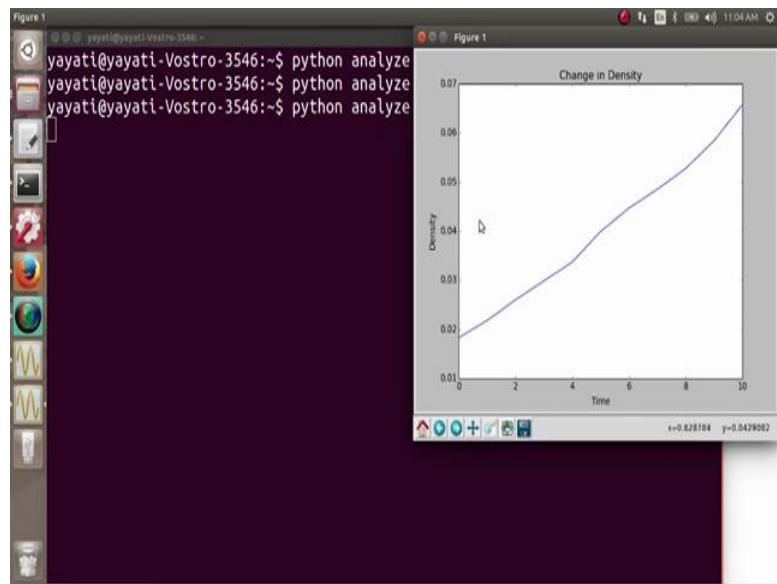
def plot_obesity():
    x=[]
    y=[]
    for i in range (0,11):
        G=nx.read_gml('evolution_'+str(i)+'.gml')
        x.append(i)
        y.append(obesity(G))

Python + Tab Width: 8 × Ln 24, Col 1 INS
```

Lets quickly define this function obesity G and it is quite easy to find the number of obese people in this graph. So, let us say the number is 0 then for each in G.nodes we see whether the node is obese or not and if the node is obese, we increase num by 1.

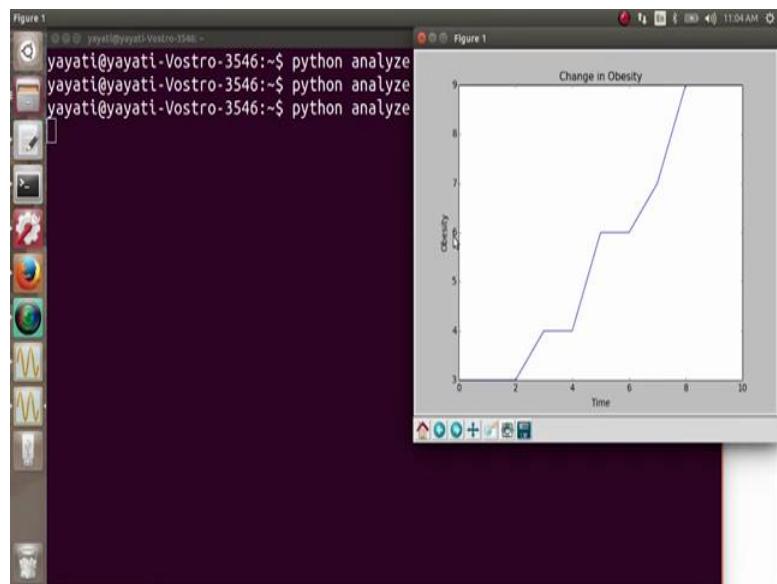
So, for each in G.nodes if G.node[each] and we look at its name right if its name equals to equals to 40 that is if its BMI is 40, we increase num by one and then we return num. So, this is how it goes.

(Refer Slide Time: 11:36)



Let us plot it and let us see it now. So, let us plot it and see. So, here we have python analyze.py. So, this first plot here is for the change in density, which we can see is increasing with time.

(Refer Slide Time: 11:51)



And now let us look at obesity, how the number of obese people in this network increase is. So, you can see here. So, it is an interesting curve you can see that it increases that then it remains constant for some time it then increases, then again remains constant for some time and so on.

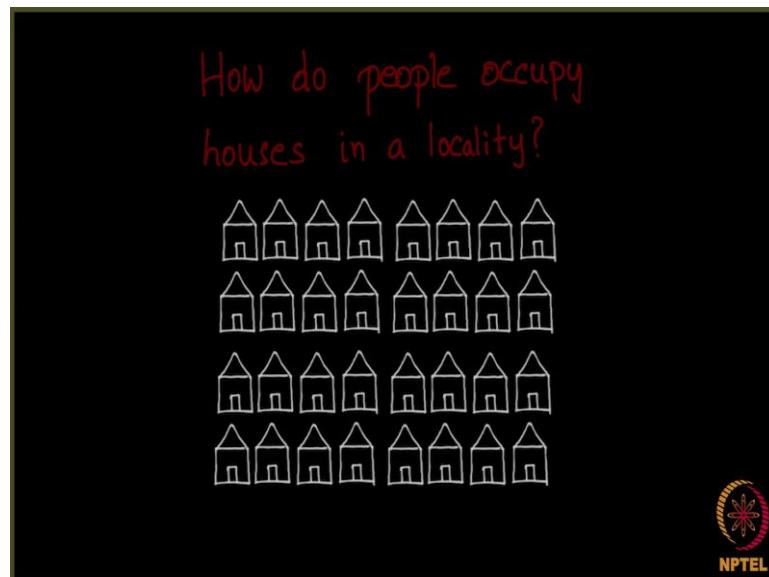
We can actually see a lot of the change in a lot of interesting parameters in this graph you can actually see how the number of membership closures is changing in this graph, you can look at the number of triads in a lot of other things. So, that is a little bit of more complicated coding, but; obviously, fun to do if you are interested in any of that coding you can try it yourself and leave your questions on discussion forum will be always happy to help you.

So, I will leave you here with these very 2 simple observations. So, over this complete evolution and you can try your own coding and let us see if there is any problem. I hope you enjoyed this screen cast.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

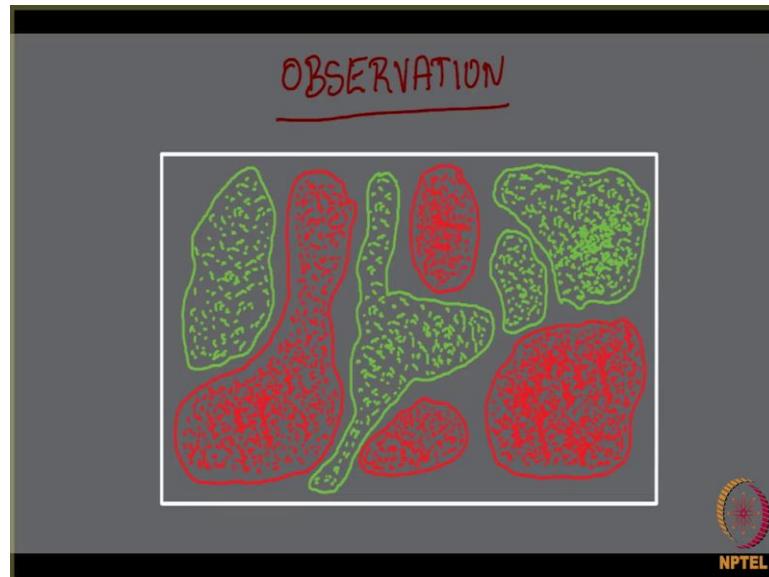
**Lecture - 54**  
**Homophily (Continued) & Positive and Negative Relationships**  
**Spatial Segregation: An Introduction**

(Refer Slide Time: 00:16)



So, let us discuss an interesting question on how do people occupy houses in a locality. So, assume you had a few houses and you want to choose a house for yourself what kind of house will you like living in that you think who your neighbors are matters to some extent would you like to be in a locality.

(Refer Slide Time: 00:44)

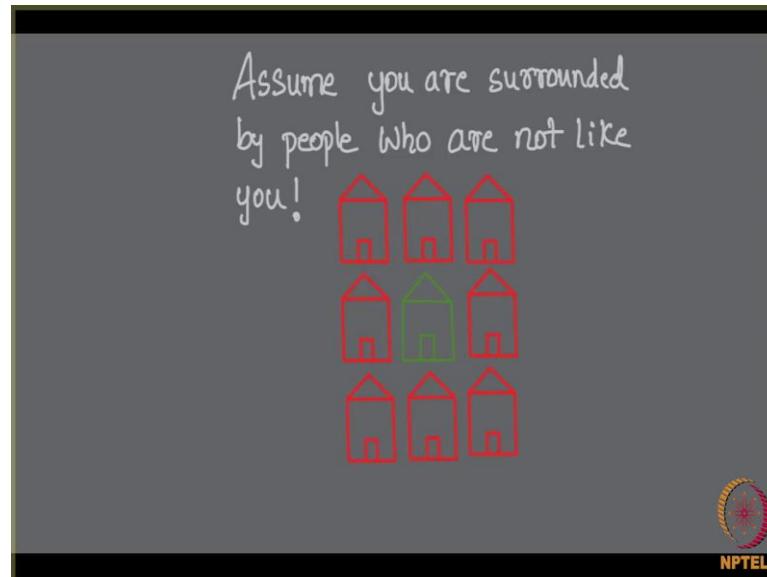


Where your neighbors are someone with whom you cannot have a conversation or your neighborhood is full of people who are unlike you would you like staying in such locality.

Now, here is an observation made several times in the literature. The observation is people are known to go and occupy a place a locality or a city uniformly at random, but which time they sort of start segregated by segregating I mean they all get together. You will see clusters of different types of people here is a type of people let us call it type green they all are together, they all are together and there is this type red as well they all are together too. Then you can see the clustering happening here they wish to stay with each other people have observed why exactly this happens as I told you this is not observed initially they come and occupy the place uniformly at random, but which time you see this kind of segregation that I am showing here.

They get segregated into different classes. So, assume in the united states you will see a lot of Indians, Chinese and people from other countries, but you will see that Indian tend to sort of stay with each other in colonies and there will be colonies of Chinese as well although there will be some sort of mingling, but there are places where you will see only people of one type this is observed in many places. So, assume you were to live in a foreign country or even in your own country would you not like to live in a place where you are surrounded by people who are of your type.

(Refer Slide Time: 02:41)



Now these is observed and let us assume you are surrounded by people who are not like you it is observed that people then they came to migrate.

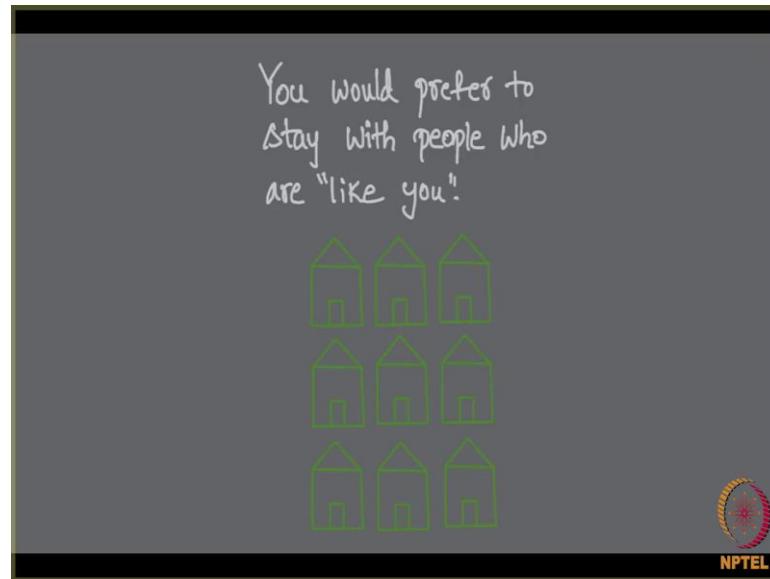
This is you and you are surrounded by people who are unlike you what will happen?

(Refer Slide Time: 03:03)



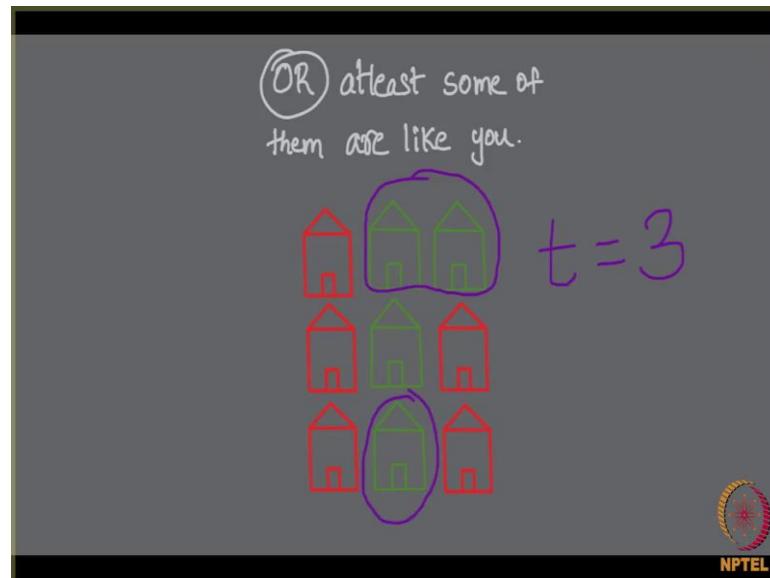
Nobody would like to stay in such a place would you like to stay in such a place absolutely not mostly not.

(Refer Slide Time: 03:14)



But then would you not prefer to stay they your type of people are present right. So, people who are around you are all like you anybody would like to stay in such a place not just this. In fact, at least some of them should be like you right at least some of them should be like you for example, you are here.

(Refer Slide Time: 03:34)



And you are surrounded by at least some 3 people who are like and the other one 2 3 4 five people are not like you would not care much.

So, let us call this the threshold value  $t$  equals 3 which means whenever take a pos and understand what I am saying this is the important concept in this lecture wherever you are surrounded by at least  $t$  number of people out of 8 people who can surround you in a neighborhood 3 behind 3 in the front and 2 people on a sideways. Let us say at least when you have  $t$  number of people who are like you then you will stay there if they are not if you are not surrounded by people at least  $t$  people who are like you then you will migrate here are 3 people who are like me and my threshold is actually 3.

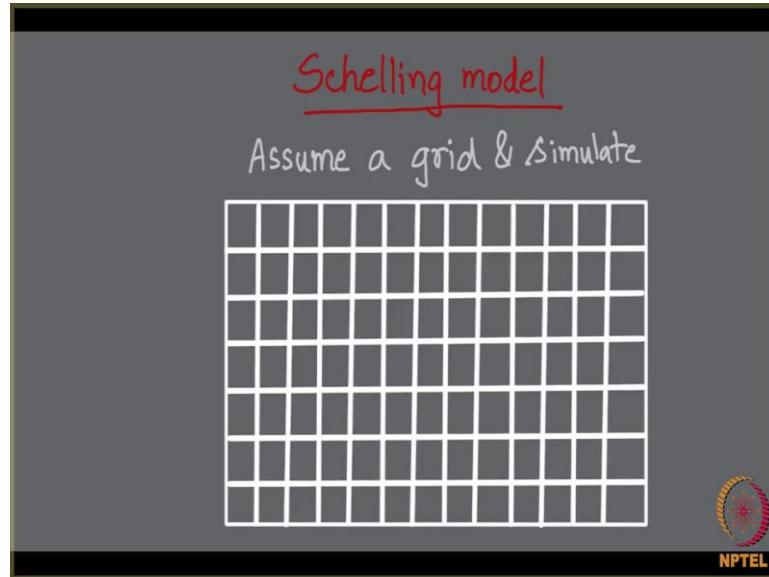
(Refer Slide Time: 04:46)



So, I like living in such a neighborhood.

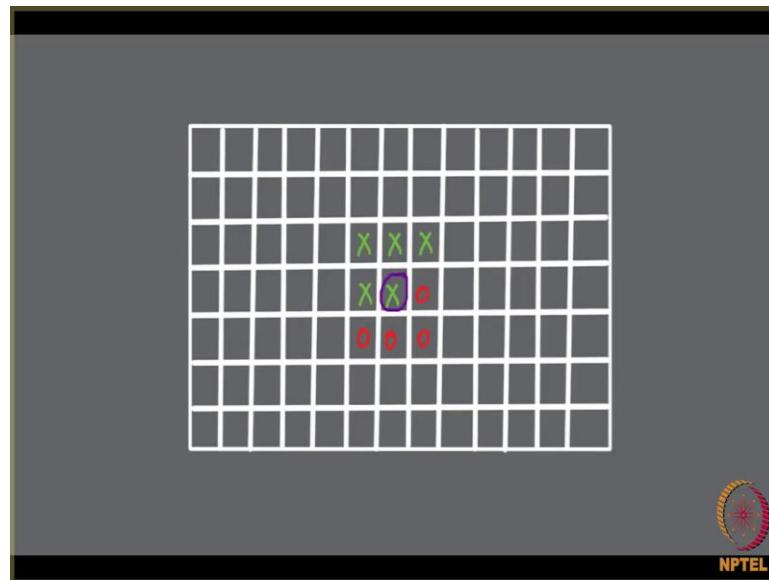
So, it is believed that unhappy people change their house to a happier destination by unhappy people I mean people who are surrounded by less than  $t$  number of people in the previous case we saw  $t$  was 3 which means this fellow was happy if in case this particular thing instead of being green was red. Then he would be unhappy because his neighbors would be less than 3 can we model this as a question can this be modeled can we model and see what exactly happens when people are unhappy and they migrate they come and occupy place uniformly at random a house uniformly at random in a city and then later they realize that because of the neighborhood they want to move to some other place.

(Refer Slide Time: 05:43)



How do we model this there is a very popular model called the Schelling model it is a very interesting model very simple and straight forward simulation it goes like this we assume that a grid simulates city grid is a analog to a city and we simulate the behavior of people.

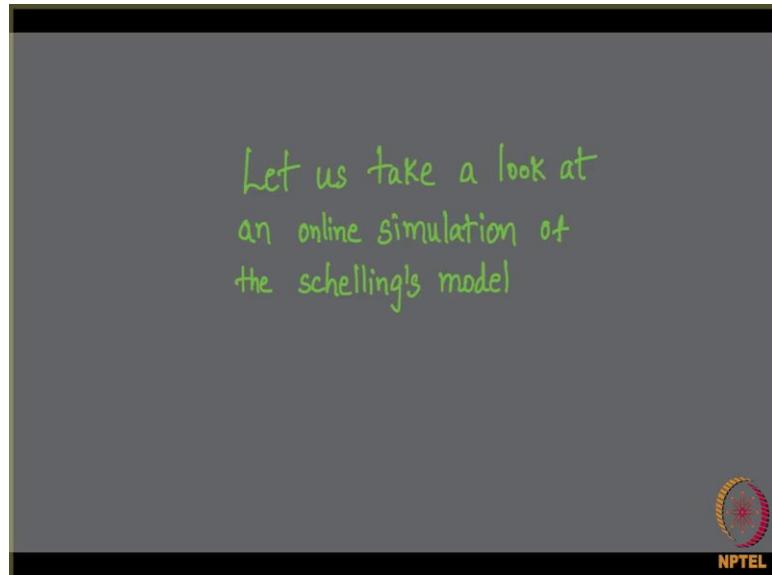
(Refer Slide Time: 06:13)



How do we do that we take a grid like this and then we put into for 1 parity that is let us say type green and then oh; for a red parity type red. So, this particular person x is

surrounded by the 4 red people and 4 green people as I told you if the threshold is 3 then this green fellow is actually happy.

(Refer Slide Time: 06:45)



Now, let us take a look at an online simulation of the Schelling's model. What is Schelling's model? Schelling's model simply says that as and when people are unhappy because they are surrounded by neighbors below their threshold value, what do I mean by that? Sounds a little (Refer Time: 07:02). All I am trying to say is my threshold is 3 or may be 2 if I have 2 people in my neighborhood who are of my type, I will stay there. Otherwise, I will move. Let me take a big grid and then try simulating this and seeing. Let us see what one can observe. There are a lot of online simulations of this model. I am just going to open a browser right now and then I am going to show you one such simulation.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture – 55**  
**Homophily (Continued) & Positive and Negative Relationships**  
**Spatial Segregation: Simulation of the Schelling Model**

(Refer Slide Time: 00:05)



The screenshot shows a web browser window with the title "Schelling's Model of Segregation". The page content includes a bio for Frank McCown, a section titled "Overview" with a text about racial segregation, a section titled "Meta Information" with a table, and a circular logo for NPTEL.

**Frank McCown**  
Computer Science Department  
Harding University  
[fmcrown@harding.edu](mailto:fmcrown@harding.edu)

**Overview**

Racial segregation has always been a pernicious social problem in the United States. Although much effort has been extended to desegregate our schools, churches, and neighborhoods, the US continues to remain segregated by race and economic lines. Why is segregation such a difficult problem to eradicate?

In 1971, the American economist [Thomas Schelling](#) created an agent-based model that might help explain why segregation is so difficult to combat. His *model of segregation* showed that even when individuals (or "agents") didn't mind being surrounded or living by agents of a different race, they would still choose to segregate themselves from other agents over time! Although the model is quite simple, it gives a fascinating look at how individuals might self-segregate, even when they have no explicit desire to do so.

In this assignment, students will create a simulation of Schelling's model. The user should be able to set a number of parameters of the model and watch it go. Bonus is given for implementing extra model features.

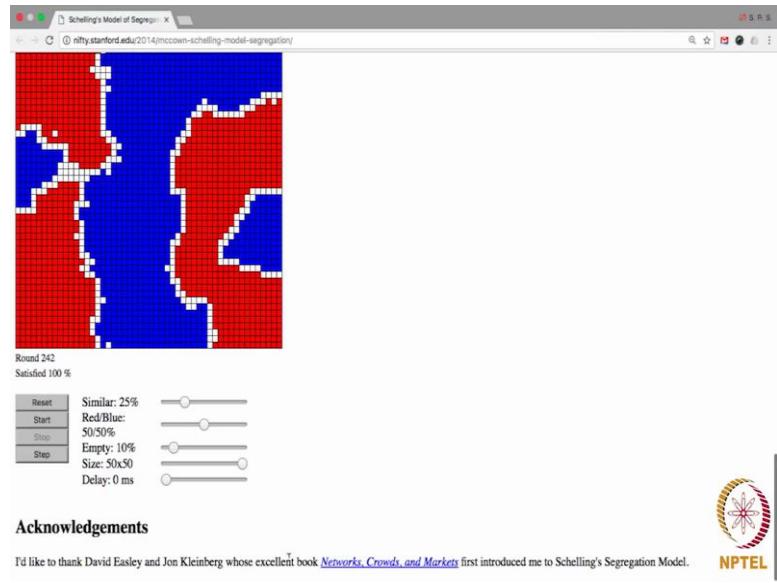
**Meta Information**

Summary	Watch as two population groups self-segregate over time into clumps. This assignment gives students the chance to implement a game-like model from the social sciences that gives an unintuitive result. It gives the instructor a chance to address a societal problem that students are already naturally interested in.
Topics	2D arrays, generating random numbers. In advanced courses other topics can be addressed like: timers, using various GUI widgets, MVC, threads, etc.
Audience	CS1 or above
Difficulty	This assignment is intermediate to advanced. Students should be given a week or two to complete it.
Strengths	1) It addresses a real-world problem that students are naturally interested in. 2) It's visually appealing and fun to play with. 3) Can be implemented in a console using X's and O's or in a GUI or web environment. 4) Uses 2D arrays and simple algorithms which do not require advanced data structures.

NPTEL

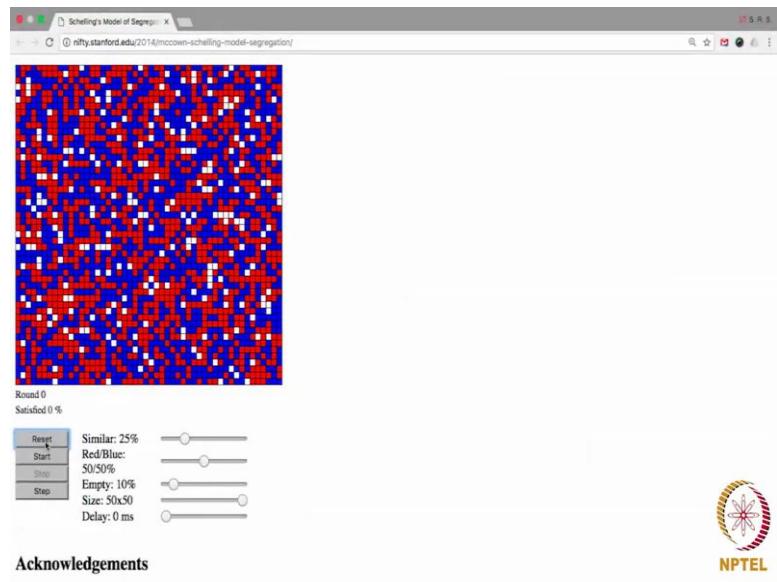
So, here is a website that I have opened, it has a very nice animation of the concept, it is by Frank McCown as you can see it there is a beautiful description of the model and a very nice simulation of the module as well.

(Refer Slide Time: 00:21)



If you can see in the acknowledgement in fact, the creator of this simulation acknowledges the text book that we have we are using for the course network stores in markets, alright.

(Refer Slide Time: 00:41)



So, here is the simulation facility let me hit the reset button and as you would have guessed that red and blue here denote 2 different types of people who have populated this city and white represents blank where anybody can shift. So, here have some parameters

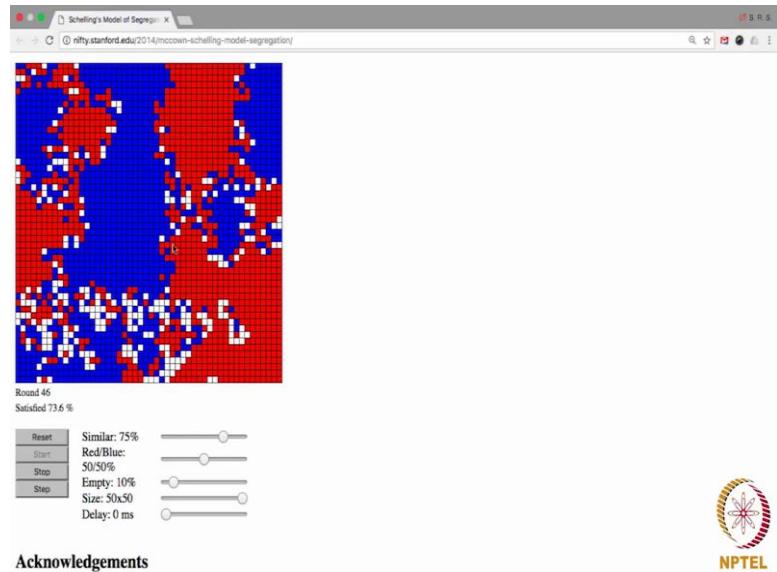
which are again, which are again quite self explanatory all though I will try explaining them for you.

So, this is this time for how similar do you expect the how similar do each node expect their neighbours if I put 25 percent here it is the t value that I explained just now, the t value I say  $t = 3$ . If its 25 percent I am surrounded by as you know 8 people 25 percent of 8 people is how much -  $8 * 1/4$  which is 2. So, if any person has 2 neighbours of the same colour then he is deemed to be happy and satisfied he will not shift and anyone who is not surrounded by at least 2 people of the same colour that person will try to shift to a blank space. And the rest is the distribution of red and blue it is the same when I say 50-50 you can in fact change it. Empty space I have kept some ten percent you can in fact, change these 2 sizes is 50 ka 50 grid delay is the delay between 2 steps of the animation I have kept it is 0.

So, let me reset it once again and then let me start the simulation as I start you will see that the number of people who are satisfied will be displayed here as of now nobody is satisfied it says 0 percent people are satisfied which means nobody is surrounded by at least 25 percent of people who are happy. Now basically once you start the animation only this will start counting as you can see I hit the start button and you see 100 percent people are satisfied it took just 8 rounds as you can see every single person is surrounded by at least 2 neighbours of its type. That is the similarity percentage that I gave t equals basically 2 now it will make t equals 4 which is 50 percent out of 8 people who my know I would expect half of them to be of my type if the reset and then start the simulation and see the beautiful convergence of segregation that happens here right now.

I hit the start button now please observe look at this they are getting segregated and you see there is some beautiful segregation happening here right. Then you expect half the people to be of your type you observe segregation happening right let me try increasing it all the more reset and increase this to let us say 75 percent what you expect now whether lets pause for a minute and then see what I am doing here I am saying 75 percent of my neighbourhood should be of my type which means if you take a blue red cell here I should have at least more than what is 75 percent of 8 neighbours it is 6. This red person should have 6 people surrounding him only then he is redeemed happy otherwise you would want to shift which means as you can see none of them are happy here, right. Let me start the simulation and see what happens.

(Refer Slide Time: 04:10)

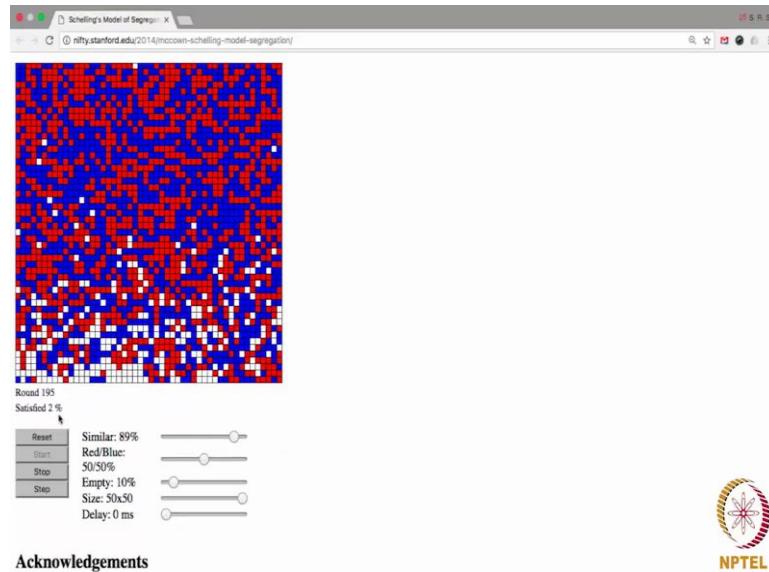


See, they are now moving to the empty spaces now roughly if 50 percent of the people are happy and then it goes on and on look at the beautiful segregation that is happening.

See more and more people get segregated and there are clusters that are emerging you see of huge blue cluster here and a red cluster here and you see 80 percent people are happy you see the stick more and more correct you see that is the segregation is beautifully going towards 100 percent. Now very soon you will see all of them are happy as you can see the number of rounds here in this case when similarity percentage is high when t is high seems to take a lot of time. You see the some problem in the boundary now with converges beautifully you see there is basically simply three clusters cluster 1; I am sorry 4 clusters cluster 1 cluster 2 cluster 3 and cluster 4 of course, you have clusters here as well, but I think we can ignore it, right.

So, basically there are 4 big clusters. So, the simulation says that more the similarity that you expect better is the segregation. If you actually I am sure you would know what we can guess what will happen if I make it 89 percent it will probably take a long time, but let us see what happens from here.

(Refer Slide Time: 05:29)



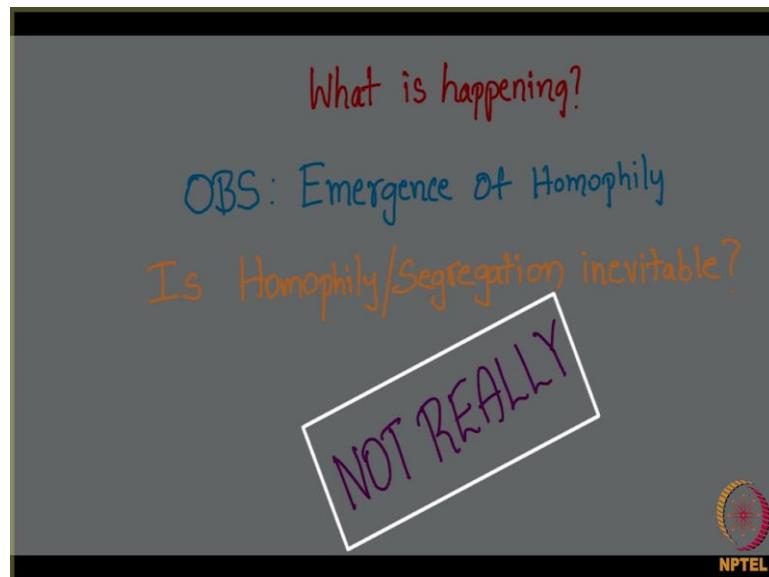
You see you see the people become less and less satisfied because 89 percent is what it is not achievable as you know 89 percent is more than 7 people should be of your type I mean that is not possible right not even more than 7 almost 8 people all your neighbour should be of your type that is not possible. So, it is a sort of simulation gets confused this might even go forever.

So, if you sort of reduce it you will observe that the convergence happens very quickly. Reset start for 30 percent which very quick yeah thanks to the create of this simulation we can explain this very easily with the simulation than verbal explanation, also it is interesting to note that the people have not understood this mathematically really well how exactly this happens how is it a function of this similarity percentage and how does the clustering combined although it is common sensical, but the exact mathematical details of it is not very well known.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture – 56**  
**Homophily (Continued) and Positive and Negative Relationships**  
**Spatial Segregation: Conclusion**

(Refer Slide Time: 00:05)

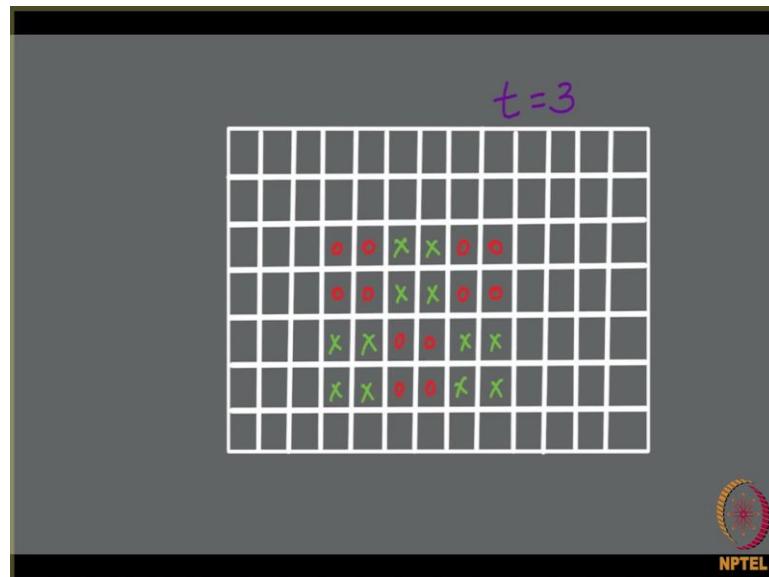


We saw the simulation; what we just observe we observed that homophily sort of emerges, now please note the chapter the discussion has been of homophily. So, do you observe that there is some sort of a homophily here right. There is a lot of people who sort of get together and they cluster together now do you observe something. In fact, it is startling to see that local behaviour results in a global structure. What do I mean by that? A local behaviour such as one person trying to decide about his locality and trying to shift to a better locality person who is trying to see if he has a good neighbours if he does not have the required number of neighbours he shifts to a different locality he is just trying to do something locally, but each and every person doing this it results in a global phenomena correct.

So, we observe that there is this emergence of homophily here, but that is this homophily or segregation inevitable, what do I mean by this, probably my next figure that I am

going to show you we will make it clear is this homophily inevitable will it always happen not really and why do I say this.

(Refer Slide Time: 01:35)



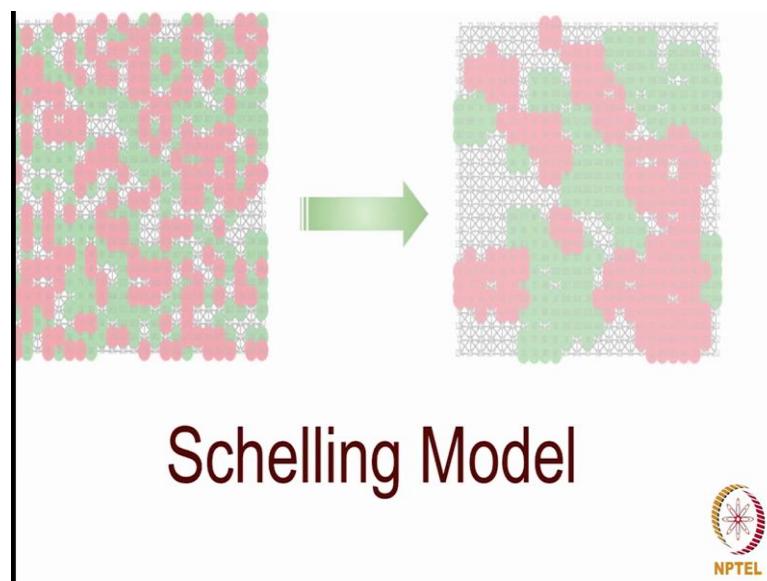
Look at this beautiful example. Here are a few people right green and red. In fact, for  $t = 3$  you observe that everybody is actually happy everybody has 3 neighbours who are like them correct. So, observe it carefully every green has 3 green neighbours every red has 3 red neighbours you can filled the entire grid like this; this is not a lot of clustering here all though there is some kind of clustering there is lot of clustering here people. In fact, are living harmoniously with each other there are both types of people and there is no high because segregation here and still they do satisfy this property of the threshold being 3 which means they have the required number of neighbours, but segregation is not happening.

Now, this is a surprising factor that all though there is a possibility that segregation need it happen segregation does happen most of the time. It is observed that in on the map of some countries people basically are distributed evenly rather I mean uniformly at random, but later on this slowly drift and you see this patterns of segregation emerging. All in all Schelling model beautifully captures the this notion of a people getting segregated and rises many questions like can we come out to the nice mathematical model to explain this which remains to be an open question till date.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

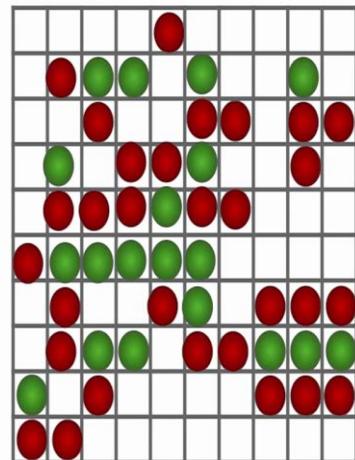
**Lecture – 57**  
**Homophily (Continued) & Positive and Negative Relationships**  
**Schelling Model Implementation – Introduction**

(Refer Slide Time: 00:05)



Hey everyone, Schelling model shows how the effect of homophily that operates at a local level leads to some interesting global patterns in the network. In this video we are going to implement this model and we will observe the patterns that emerge. Before we start the implementation I will give you a brief of the model, let us assume that there is a population of people and these people are one of the 2 types that is every person is either of type 0 or type 1.

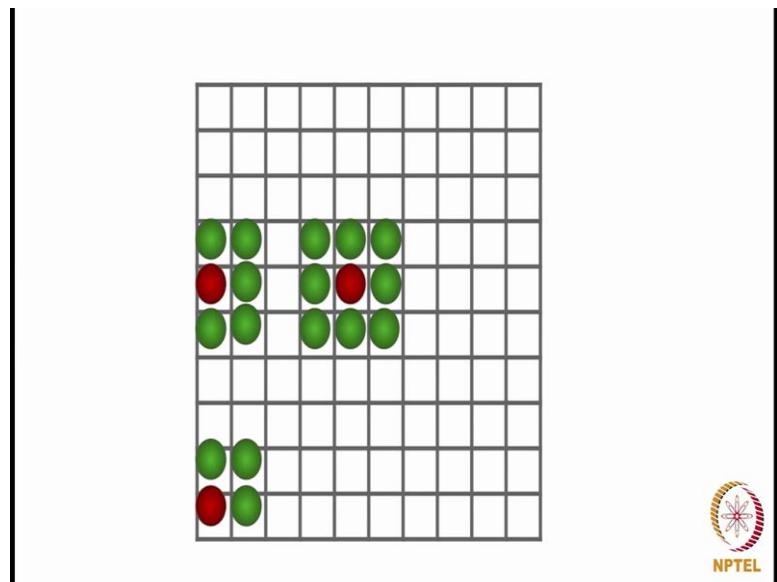
(Refer Slide Time: 00:44)



Further assume that these people live in a city which is like a grid that is there are cells in this grid where these people live just like this. Also, some cells of this grid have people whereas, some cells will be empty, in other words every cell will either have a person of type 0 or a person of type 1 or it will be empty. For example, like this as you can see every cell either has a red node or you can say red person or a green person or it is empty, correct.

Now, cell's neighbours are the cells that touch it including the diagonal cells. So, you can imagine that the cell that is not on the boundary will have 8 neighbours.

(Refer Slide Time: 01:24)



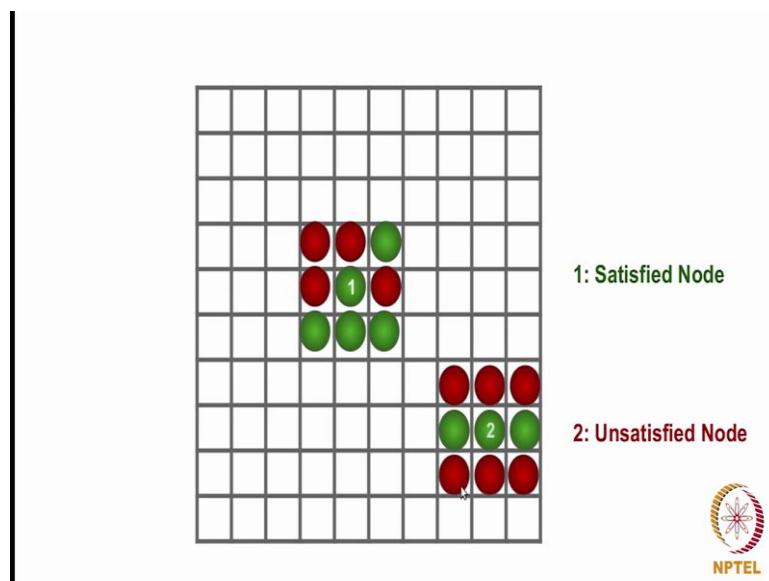
For example, if you look at this scenario this red node is an internal node that is not on the boundary. So, every node which is not on the boundary will have 8 neighbours like this the red person has basically the cell that has red person has 8 cells which are its neighbouring cells. Similarly, if you look at the boundary cells they can be 2 cases first case could be where the cell is a corner cell like here the red person is staying in a cell which is a corner cell. So, this cell has 3 neighbours as you can see here the other kinds of boundary cells can be like this which are not the corners.

So, as you can see this red person which is residing in the cell has five neighbours I think it is not difficult to understand that we can convert this whole scenario into a graph where the cells are the nodes and there is an edge between 2 cells if they are neighbours on this grid. So, that is what we are going to do we are going to analyze this grid using a network now if you look at a metropolitan city where there are people from different states of the country everyone tends to or rather prefers to stay in an area where there are more people from their own state. So, that practically happens we are going to apply the same scenario in our given grid. So, the people in our model wish to have at least some other people of their own type as neighbours.

Now, as an analogy if you look at a metropolitan city for example, where there are people from different states of the country everyone tends to or rather prefers to stay in an area where there are more people from their own state in a similar fashion the people

in our model wish to have at least some other people of their own type as neighbours. So, what we will do is we will assume a common threshold  $t$  for each person if a person discovers that it is surrounded by fewer than  $t$  people of its own type he tends to move to a new cell such a person is called unsatisfied with his current location.

(Refer Slide Time: 03:48)



As an example if we set the threshold  $t$  to be 3 for instance and if you look at this node a labelled one it has 8 neighbours as you can see and out of these 8 neighbours 4 neighbours are of its own type and the rest 4 neighbours are of different type. So, since the threshold is 3 it has more than 3 neighbours of its own type. So, we will call this node one to be a satisfied node if you look at the other scenario.

For example if you look at this node labelled one it is of green type and it has 8 neighbours out of these 8 neighbours 4 neighbours are of the different type and 4 neighbours are of the same type as the neighbour as the node one.

Now, since as an example if we set the threshold  $t$  to be equal to 3 let us say and we look at the node which is labelled one here as you can see there are 8 neighbours and out of the 8 neighbours 4 neighbours are of different type and 4 neighbours are of the same time as the given node one since the threshold  $t = 3$ . And this node one has more than 3 neighbours of its own time we will call this node one to be a satisfied node, right and if you take one more example and if you look at this node which is labelled 2 here it again has 8 neighbours out of these 8 neighbours 6 neighbours are of different types and 2

neighbours are of the same type as the node 2 and threshold t we have said to be 3 since this node 2 has lesser than t neighbours of its own type this node 2 will be called an unsatisfied node.

Now, as I already explained the unsatisfied nodes tend to move to new locations which are already empty and where they are likely to be satisfied their different versions of Schelling model in this context. For example, in some of the versions does the unsatisfied nodes we will move to those locations where there are more likely to become satisfied or in other versions the unsatisfied nodes will move to any random location. So, so there are different versions which you can implement.

Ah another thing is that whenever a node moves from one cell to the other cell it affects other neighbouring nodes as well for example, in this process some other nodes might become satisfied some other nodes might become unsatisfied. So, it basically changes the scenario of the cell of the grid, right. So, in every iteration, we choose some unsatisfied nodes and we move them to new locations and the cell and the grid structure changes and this keeps on repeating and we can analyze the structure that we get to in the end that is after certain number of iterations.

So, this is what we are going to implement we are going to implement the version of Schelling model where the unsatisfied nodes move to the random locations and we are going to run several iterations of this process we are basically going to recalculate the unsatisfied nodes in every iteration because that is; obviously, required because one shifting affects the rest of the nodes as well. So, let us get started with implementation now.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture – 58**  
**Homophily (Continued) & Positive and Negative Relationships**  
**Schelling Model Implementation – Base Code**

Before we start the implementation in the editor I want to show you certain commands in the ipython console, so that it becomes easier for you to understand these commands when I write there in the editor. I basically want to show you some intermediate results. So, I will be easier for you to understand. So, let me ask you what is our aim our aim is to how a grid then we have to put people on the cells of the script then we have to identity the nodes which are the people which are unsatisfied and then we have to move these are unsatisfied people to different locations that is what is the cracks. Now what is the first step? The first step is you want to grid, right.

So, how do we get the grid is your any network expansion that can help us to get the grid. So, apparently, we have this function grid to be graph which we can make use of although not to suit our complete purpose, but we can make changes into that and then basically we can manipulate that.

(Refer Slide Time: 01:16)

```
anamika@anamika-Inspiron-5423:~$ ipython
Python 2.7.6 (default, Oct 26 2016, 20:30:19)
Type "copyright", "credits" or "license" for more information.

IPython 1.2.1 -- An enhanced Interactive Python.
?           -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help        -> Python's own help system.
object?    -> Details about 'object', use 'object??' for extra details.

In [1]: import networkx as nx

In [2]: N = 10

In [3]: G = nx.grid_2d_graph(N,N)

In [4]: import matplotlib.pyplot as plt

In [5]: nx.draw(G)

In [6]: plt.show()
```



So, let us get started I will start my ipython console and lets import. So, let us create a 10 cross 10 grid. So, I will take N to be 10 and then I will use the function. So, I will write n x.grid 2 d graph and we want to N to be N cross and that is a 10 cross 10 in this case.

So, this is the function that we will use let me show how it looks like. So, I need to show I need to show that. So, I need to import matplot lib.pyplot as plt. So, I will write n x.draw(G) and plt.show.

(Refer Slide Time: 02:17)

The screenshot shows an IPython notebook interface with a code editor on the left and a plot area on the right. The code editor contains the following Python code:

```
Python 2.7.6 (default, Oct 26 2016, 20  
Type "copyright", "credits" or "license"  
  
IPython 1.2.1 -- An enhanced Interactive  
?      -> Introduction and overview  
%quickref -> Quick reference.  
help    -> Python's own help system.  
object? -> Details about 'object', u  
  
In [1]: import networkx as nx  
  
In [2]: N = 10  
  
In [3]: G = nx.grid_2d_graph(N,N)  
  
In [4]: import matplotlib.pyplot as pl  
  
In [5]: nx.draw(G)  
  
In [6]: plt.show()
```

The plot area displays a circular grid graph with red nodes and black edges, representing a 10x10 grid. The plot has a coordinate system at the bottom right corner. The NPTEL logo is visible in the bottom right corner of the slide.

So, this is how it looks like; obviously, it is not looking. So, good it is not even looking like a grid of, but it is a grid although it is not looking like the way we want to show it. So, we might need to make some changes into it let us see what kind of changes we can make here I am going to close it the graph is not looking. So, nice because the nodes are not displayed in the form of a grid apparently, we do not have an in grid layout a networkx.

So, we might need to manually assign the positions to the grid to the nodes now how can we do that before I tell you that let me show you the nodes of this graph.

(Refer Slide Time: 03:05)

```
In [2]: N = 10
In [3]: G = nx.grid_2d_graph(N,N)
In [4]: import matplotlib.pyplot as plt
In [5]: nx.draw(G)
In [6]: plt.show()
In [7]: G.nodes()
Out[7]:
[(7, 3),
 (6, 9),
 (0, 7),
 (1, 6),
 (3, 7),
 (2, 5),
 (8, 5),
 (5, 8),
 (4, 0),
```



So, I will write G.nodes now this something different here in the other kinds of graphs we get either some number or some use a define string as the label of the nodes, but here see I wrote G.nodes and what I am getting is and getting a tuple for every node and the tuple consist in 2 values which is basically the row number and the column number of that node. So, basically it is like a grid, but it just that is not looking like a grid.

So, we might need to display it accordingly we might need to assign appropriate positioning to every node. So, I will show you how we can do that please note that we are getting both x and y positions which are relative positions of these nodes with respect to each other.

(Refer Slide Time: 03:58)

```
(5, 6),  
(7, 7),  
(6, 3),  
(1, 2),  
(4, 9),  
(3, 3),  
(2, 9),  
(8, 1),  
(4, 4),  
(6, 3),  
(0, 0),  
(7, 9),  
(3, 8),  
(2, 0),  
(1, 8),  
(8, 8),  
(4, 3),  
(9, 5),  
(5, 2)]
```

In [8]:



(Refer Slide Time: 04:00)

```
(4, 9),  
(3, 3),  
(2, 9),  
(8, 1),  
(4, 4),  
(6, 3),  
(0, 0),  
(7, 9),  
(3, 8),  
(2, 0),  
(1, 8),  
(8, 8),  
(4, 3),  
(9, 5),  
(5, 2)]
```

In [8]: pos = dict((n, i) for n in G.nodes())

In [9]: nx.draw(G, pos)

In [10]: plt.show()



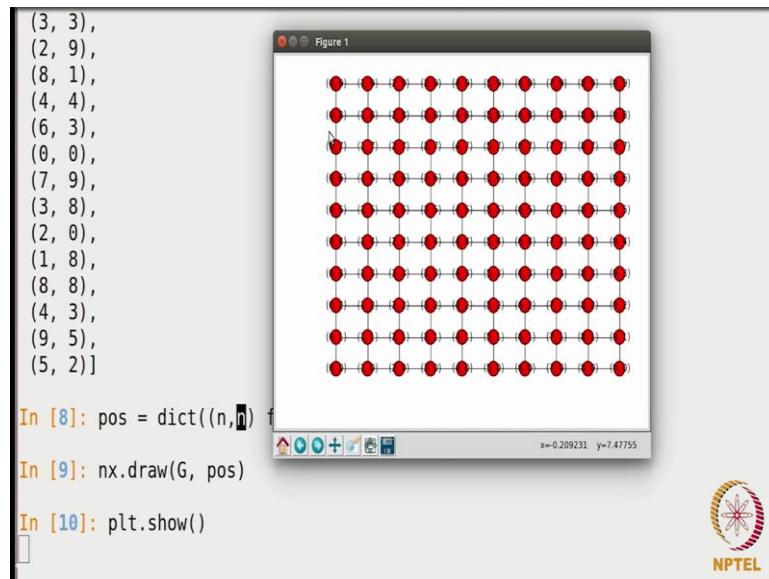
So, what we can do is I will write  $((N, N))$  for  $N$  in  $G.nodes$  or since we have used  $N$  capital  $N$  already maybe I should use small  $n$ . So, what I will write is I will convert this into a dictionary, and I will assign that dictionary to a variable say  $pos$ .

Let me explain you what I have done. So, these are  $G.nodes$  as you can see please note the pattern of the nodes are we getting here. So, these  $(8, 8)$  for example, will be  $N$  will be 1  $N$ . So, what I am doing here for  $N$  in  $G.nodes$  that is for all these tuples I am writing  $(N, N)$ . So, I am creating tuple where there are 2  $N$ s  $(N, N)$ . So, it will be like 8 8 comma

8 8 what I am doing is for node I am fixing its position which is the same as its label, right so, and after that I am converting that into dictionary because that is how we pos the parameter that we pos should be in the form of dictionary. So, this dict function this is the function in python what it does is it converts a tuple into a dictionary where the first value of the tuple will be the key and the second value of the tuple will be the value of the dictionary.

So, this pos this is the positioning that we have fixed and we can assign the these positions should a nodes and how we can do that is we call this function and it is not draw G we can pos this positioning here and then we can write plt.show.

(Refer Slide Time: 05:47)



Alright, this looks beautiful now listen it right. So, as you can see, we are getting the grid and now other thing that we have to note here is that the labels are not in. So, nice one thing that you should note here is that 0 0 is here and the bottom left corner and it goes like this 0 0 0 1 0 2 0 3 and then comes back from 1 0 1 1 and so on. So, this is the default sequencing of the nodes that this function provides that we used what grid 2 d graph if you want you can change this label in such a way that 0 0 pairs here, but I do not think we need that. So, we will let it be like this, but one thing that you would want to do is that we would want to change the label of basically we have 10 cross 10 that is hundred nodes here and it will be nice if we assign the no labels like 0 2 0 1 2 3 up to 99.

So, that is one thing that we would like to do here now let me show you how we can do that again we will assign different label to every node now the name of the node itself carries the row and column number that is what we will be using in order to assign the label. So, the 0 0 should be having the label 0 0 1 should be having label 1 and so on 0 1 2 3 4 up to 10; now 0 up to 9 and then 10 should be here than 11 and so on. So, this is how we have to assign the labels how can we do that if you know a little math you should be able to understand what I am doing.

(Refer Slide Time: 07:35)

```
(0, 0),
(7, 9),
(3, 8),
(2, 0),
(1, 8),
(8, 8),
(4, 3),
(9, 5),
(5, 2])

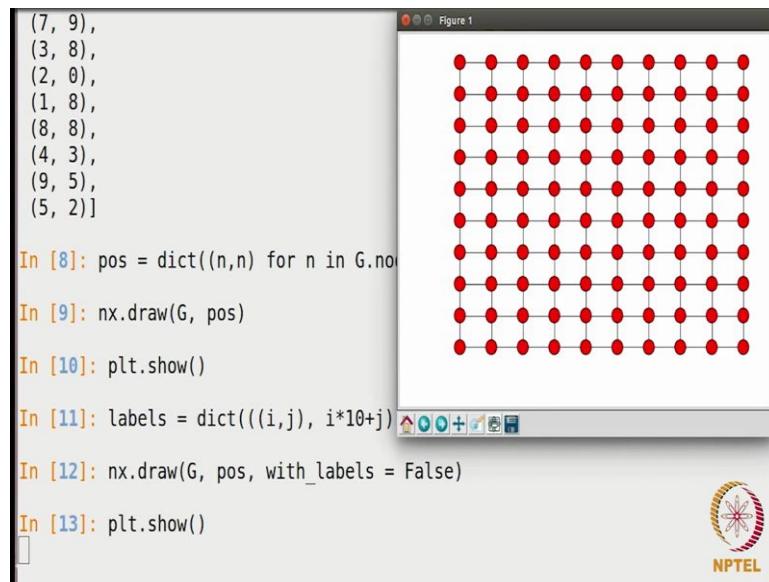
In [8]: pos = dict((n,n) for n in G.nodes())
In [9]: nx.draw(G, pos)
In [10]: plt.show()
In [11]: labels = dict(((i,j), i*10+j) for i,j in G.nodes())
In [12]: nx.draw(G, pos, with_labels = False)
In [13]: plt.show()
```



So, what I will do is  $(i, j)$  to  $(i, j)$  what I am assigning is  $i * 10 + j$  this is what I am assigning to first node and to the  $(i, j)$ th node this I will do for every node. So, I will write for  $(i, j)$  in  $G.nodes$  and again I will convert this into dictionary. So, I will use the same function and let me call this dictionary labels or when they should go.

So, what I am doing here is that to every node which is  $(i, j)$  like here like 4 3. I am assigning label which is  $i * 10 + j$  in the case  $(4, 3)$  it will be 43. So, that label I am assigning to every node and that information I am keeping in the form of a dictionary in this dictionary called labels which I am going to use. So, let me show you how we can do that. So, we used `nx.draw` a previously which displace some labels by default which is the labels that I showed you just now we do not want those labels to be displayed. So, what I will write is `with labels = false` right. So, the labels that it will showing it will not be visible you can check that is well if you I can pretty show you.

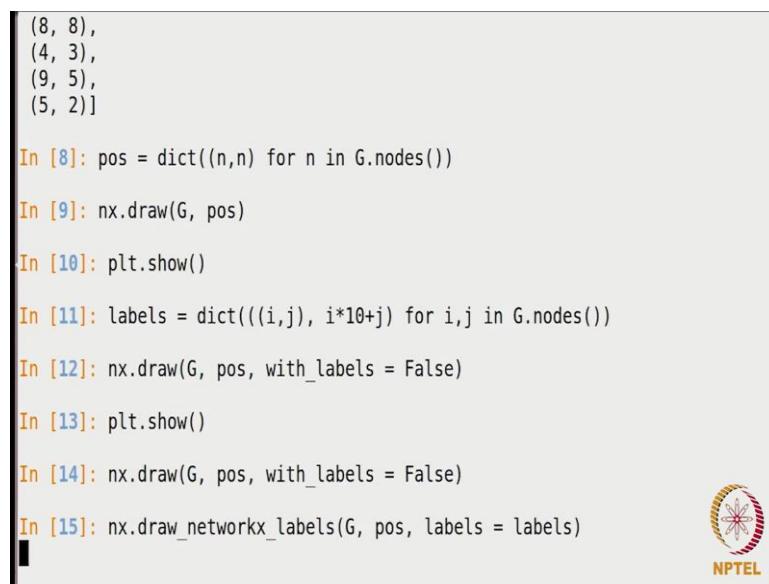
(Refer Slide Time: 09:02)



```
(7, 9),
(3, 8),
(2, 0),
(1, 8),
(8, 8),
(4, 3),
(9, 5),
(5, 2])

In [8]: pos = dict((n,n) for n in G.nodes())
In [9]: nx.draw(G, pos)
In [10]: plt.show()
In [11]: labels = dict(((i,j), i*10+j))
In [12]: nx.draw(G, pos, with_labels = False)
In [13]: plt.show()
```

(Refer Slide Time: 09:04)



```
(8, 8),
(4, 3),
(9, 5),
(5, 2))

In [8]: pos = dict((n,n) for n in G.nodes())
In [9]: nx.draw(G, pos)
In [10]: plt.show()
In [11]: labels = dict(((i,j), i*10+j) for i,j in G.nodes())
In [12]: nx.draw(G, pos, with_labels = False)
In [13]: plt.show()
In [14]: nx.draw_networkx_labels(G, pos, labels = labels)
In [15]: nx.draw_networkx_labels(G, pos, labels = labels)
```

So, there are no label displayed here; here now we want to display our own labels for that we have a function `nx.draw_networkx_labels`. Now again we have to `pos` the graph we have to `pos` the positioning and then we have to `pos` the labels dictionary `labels` dictionary.

So, this is how we write labels is it would labels this should work now the labels have the new labels have been assigned that is display them oh, yeah, this looks nice.

(Refer Slide Time: 09:28)

```
(8, 1): <matplotlib.text.Text at 0x7fd4668fd8d0>,
(8, 2): <matplotlib.text.Text at 0x7fd466118490>,
(8, 3): <matplotlib.text.Text at 0x7fd465f48a10>,
(8, 4): <matplotlib.text.Text at 0x7fd466958510>,
(8, 5): <matplotlib.text.Text at 0x7fd465f44510>,
(8, 6): <matplotlib.text.Text at 0x7fd465f7cdd0>,
(8, 7): <matplotlib.text.Text at 0x7fd4660de150>,
(8, 8): <matplotlib.text.Text at 0x7fd466914b50>,
(8, 9): <matplotlib.text.Text at 0x7fd4660fcf90>,
(9, 0): <matplotlib.text.Text at 0x7fd46611c090>,
(9, 1): <matplotlib.text.Text at 0x7fd466963210>,
(9, 2): <matplotlib.text.Text at 0x7fd465f967d0>,
(9, 3): <matplotlib.text.Text at 0x7fd466118d10>,
(9, 4): <matplotlib.text.Text at 0x7fd465f78410>,
(9, 5): <matplotlib.text.Text at 0x7fd466921410>,
(9, 6): <matplotlib.text.Text at 0x7fd4660de9d0>,
(9, 7): <matplotlib.text.Text at 0x7fd465fa3690>,
(9, 8): <matplotlib.text.Text at 0x7fd4660b6e90>,
(9, 9): <matplotlib.text.Text at 0x7fd4660fc2d0>

In [16]: plt.show()
```



Now, So, we have the labels from 0 1 2 3 up to 99. So, it will be nice to refer to them when we do the later manipulations.

(Refer Slide Time: 09:35)

```
(8, 2): <matplotlib.text.Text at 0x7fd466118490>,
(8, 3): <matplotlib.text.Text at 0x7fd466118490>,
(8, 4): <matplotlib.text.Text at 0x7fd465f48a10>,
(8, 5): <matplotlib.text.Text at 0x7fd465f48a10>,
(8, 6): <matplotlib.text.Text at 0x7fd465f48a10>,
(8, 7): <matplotlib.text.Text at 0x7fd465f48a10>,
(8, 8): <matplotlib.text.Text at 0x7fd465f48a10>,
(8, 9): <matplotlib.text.Text at 0x7fd465f48a10>,
(9, 0): <matplotlib.text.Text at 0x7fd46611c090>,
(9, 1): <matplotlib.text.Text at 0x7fd466963210>,
(9, 2): <matplotlib.text.Text at 0x7fd465f967d0>,
(9, 3): <matplotlib.text.Text at 0x7fd466118d10>,
(9, 4): <matplotlib.text.Text at 0x7fd465f78410>,
(9, 5): <matplotlib.text.Text at 0x7fd466921410>,
(9, 6): <matplotlib.text.Text at 0x7fd4660de9d0>,
(9, 7): <matplotlib.text.Text at 0x7fd465fa3690>,
(9, 8): <matplotlib.text.Text at 0x7fd4660b6e90>,
(9, 9): <matplotlib.text.Text at 0x7fd4660fc2d0>
```

x=5.27385 y=5.00408

```
In [16]: plt.show()
```



Now, one thing that is important to note here is that I told in the explanation of scaling model that every node which is an internal node that is node on the boundary should have 8 neighbours right. Now here if you look at this node number 45 it has 3; 35, 46, 55, 44; it has only 4 neighbours it is not neighbours with the diagonal nodes as you can see. So, this is the kind of grid that the function provides as by default.

But this is not serving our purpose our purpose is that this 45 should be connected to 56 as well as all these 4 nodes 36, 34, 54 and 56 now how do we do that. So, we would have to add them the edges manually which are not already present here now how can we do that I want to show you.

(Refer Slide Time: 10:40)

```
(8, 8): <matplotlib.text.Text at 0x7fd466914b50>,
(8, 9): <matplotlib.text.Text at 0x7fd4660fcf90>,
(9, 0): <matplotlib.text.Text at 0x7fd46611c090>,
(9, 1): <matplotlib.text.Text at 0x7fd466963210>,
(9, 2): <matplotlib.text.Text at 0x7fd465f967d0>,
(9, 3): <matplotlib.text.Text at 0x7fd466118d10>,
(9, 4): <matplotlib.text.Text at 0x7fd465f78410>,
(9, 5): <matplotlib.text.Text at 0x7fd466921410>,
(9, 6): <matplotlib.text.Text at 0x7fd4660de9d0>,
(9, 7): <matplotlib.text.Text at 0x7fd465fa3690>,
(9, 8): <matplotlib.text.Text at 0x7fd4660b6e90>,
(9, 9): <matplotlib.text.Text at 0x7fd4660fc2d0>}

In [16]: plt.show()

In [17]: nx.draw_networkx_labels(G, pos, labels = labels)
%% latex labels labels= lambda

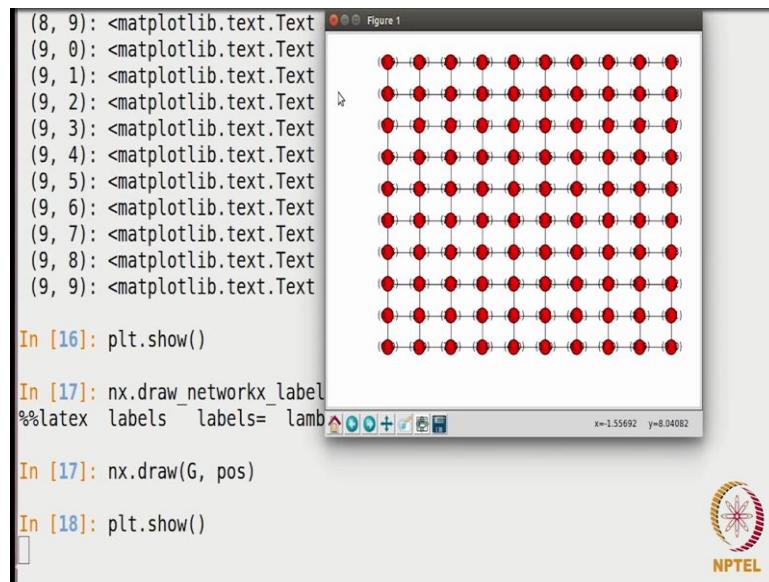
In [17]: nx.draw(G, pos)

In [18]: plt.show()
```



The previous representation of the graph because that will help us understand how we can add the edges, so, what I am going to do is I am not going to use this command `nx.draw` the usual function that we will use and this position we were using I would like to display this only.

(Refer Slide Time: 11:06)



I am showing you the previous version of the graph for a reason because I want to show you how can we have diagonal edges now if you look at this 1 1 node we have to connect to 2 2 this 2 2 has to be connected to 3 3 and if you look at some other node this 2 6 has to be connected to 3 7 4 8 3 7 has to be connected to 4 8.

So, if you observe carefully the node  $(i, j)$  has to be connected to  $(i + 1, j + 1)$  if you look at 6 7 it has to be connected to 7 8 which is again  $(i + 1, j + 1)$ . So, we will take all the nodes  $(i, j)$  and we will connect them to the nodes  $(i + 1, j + 1)$  that should be half of the work done. So, let us see one thing that we have to note here again is that when we are doing  $(i + 1, j + 1)$  it should never cross the 9 right we cannot connect 9 9 to 10 10. So, that limit we have to take care of, so that we will take care of that thing.

(Refer Slide Time: 12:15)

```
(9, 7): <matplotlib.text.Text at 0x7fd465fa3690>,
(9, 8): <matplotlib.text.Text at 0x7fd4660b6e90>,
(9, 9): <matplotlib.text.Text at 0x7fd4660fc2d0>

In [16]: plt.show()

In [17]: nx.draw_networkx_labels(G, pos, labels = labels)
%& latex labels labels= lambda

In [17]: nx.draw(G, pos)

In [18]: plt.show()

In [19]: for (u,v) in G.nodes():
....:     if (u+1 <= N-1) and (v+1 <= N-1):
....:         G.add_edge((u,v),(u+1,v+1))
....:

In [20]: nx.draw(G, pos)

In [21]: nx.draw(G, pos)
```



So, I will write for  $(u, v)$  let say for  $(u, v)$  in  $G.\text{nodes}$  what do we want to do  $(u, v)$  has to be connected to  $(u + 1, v + 1)$ , but again we have to take care that  $u + 1$  and  $v + 1$  does not exist does not exceed  $N - 1$ . So, I will write if  $u + 1 \leq N - 1$  and we say to be done for  $v$ . So, I will write  $v + 1 \leq N - 1$  then only we will do this otherwise you will not add that edge. So, I will write  $G.\text{add\_edge}$  which edge  $(u, v)$  edge to  $(u + 1, v + 1)$  edge this should work let us see let me show you in the graph at this stage I will write  $\text{nx.draw}$  and then (Refer Time: 13:16)  $\text{plt.show}$ , yeah.

(Refer Slide Time: 13:16)

```
(9, 9): <matplotlib.text.Text at 0x7f

In [16]: plt.show()

In [17]: nx.draw_networkx_labels(G, pos
%& latex labels labels= lambda

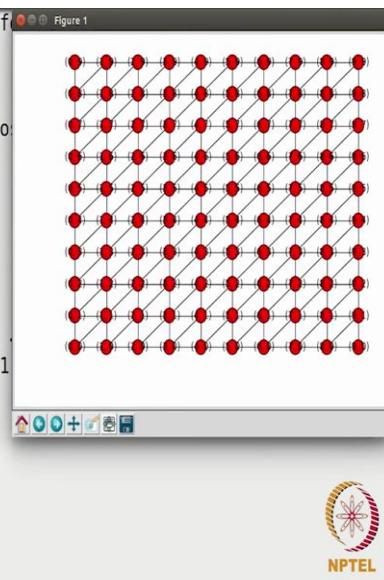
In [17]: nx.draw(G, pos)

In [18]: plt.show()

In [19]: for (u,v) in G.nodes():
....:     if (u+1 <= N-1) and (v+1 <= N-1):
....:         G.add_edge((u,v),(u+1,v+1))
....:

In [20]: nx.draw(G, pos)

In [21]: plt.show()
```



So, we have 1 p of half of the diagonal edges in corporate already the other direction of the diagonal edges is still yet to be added for example, we must connect this 3 9 2 4 8 4 8 to 5 7. So, that must be done.

Now, if you observe carefully 4 6 has to be connected to 5 5 which means  $(u, v)$  has to be connected to  $(u + 1, v - 1)$  yeah that that whole is true for every node. Now again one thing that we have to take care of here is that while doing  $v - 1$  it should not come less than 0, so that part we will take care of let see how we can do that.

(Refer Slide Time: 14:01)

```
In [18]: plt.show()

In [19]: for (u,v) in G.nodes():
....:     if (u+1 <= N-1) and (v+1 <= N-1):
....:         G.add_edge((u,v),(u+1,v+1))
....:

In [20]: nx.draw(G, pos)

In [21]: plt.show()

In [22]: for (u,v) in G.nodes():
....:     if (u+1 <= N-1) and (v-1 >= 0):
....:         G.add_edge((u,v),(u+1,v-1))
....:

In [23]: nx.draw(G, pos)

In [24]: for (u,v) in G.nodes():
....:     if (u+1 <= N-1) and (v-1 >= 0):
....:         G.add_edge((u,v),(u+1,v-1))
```



I am going to make changes here itself or yeah. So, what I will do is I will go back here and if  $u + 1$ . So,  $(u, v)$  must be connected to  $(u + 1, v - 1)$ . So,  $u + 1$  should be this and  $v - 1$  should be less than equal to now should be greater than equal to 0 it should not be less than equal to. So, this we must check and if this is true then we will have  $N$  h from  $(u, v)$  to  $(u + 1, v - 1)$ ; they should work.

(Refer Slide Time: 14:46)

```
In [18]: plt.show()

In [19]: for (u,v) in G.nodes:
....:     if (u+1 <= N-1)
....:         G.add_edge(
....:

In [20]: nx.draw(G, pos)

In [21]: plt.show()

In [22]: for (u,v) in G.nodes:
if (u+1 <= N-1) and (v-1 >= 0):
    G.add_edge((u,v),(u+1,v-1))
....:

In [23]: nx.draw(G, pos)

In [24]: plt.show()
```

Doubtful about the in indentation but let me see if there is work if it works fine, yeah. So, here you see all the edges have been incorporated all the diagonal edges are there now the great seems complete.

(Refer Slide Time: 14:58)

```
....:     G.add_edge((u,v),(u+1,v+1))
....:

In [20]: nx.draw(G, pos)

In [21]: plt.show()

In [22]: for (u,v) in G.nodes():
if (u+1 <= N-1) and (v-1 >= 0):
    G.add_edge((u,v),(u+1,v-1))
....:

In [23]: nx.draw(G, pos)

In [24]: plt.show()

In [25]: nx.draw(G, pos, with_labels = False)

In [26]: nx.draw_networkx_labels(G, pos, labels = labels)
```

Let me just show you the labels they updated labels that we recently showed which command it will be use yeah, we use this and then we used this one yeah let us see.

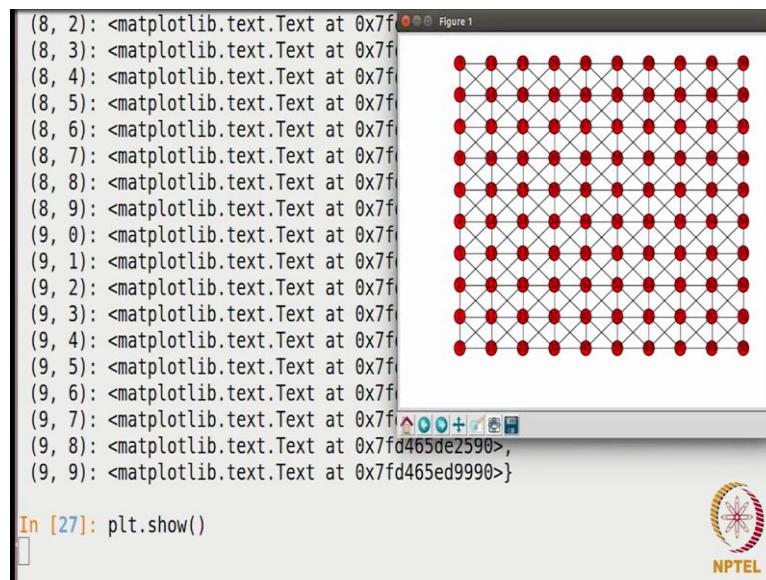
(Refer Slide Time: 15:11)

```
(8, 1): <matplotlib.text.Text at 0x7fd465d03f90>,
(8, 2): <matplotlib.text.Text at 0x7fd465c6db50>,
(8, 3): <matplotlib.text.Text at 0x7fd465dbd610>,
(8, 4): <matplotlib.text.Text at 0x7fd465d5ebd0>,
(8, 5): <matplotlib.text.Text at 0x7fd465c4abd0>,
(8, 6): <matplotlib.text.Text at 0x7fd465c72250>,
(8, 7): <matplotlib.text.Text at 0x7fd465d3b810>,
(8, 8): <matplotlib.text.Text at 0x7fd465d27250>,
(8, 9): <matplotlib.text.Text at 0x7fd465ed0690>,
(9, 0): <matplotlib.text.Text at 0x7fd465c568d0>,
(9, 1): <matplotlib.text.Text at 0x7fd465d6c8d0>,
(9, 2): <matplotlib.text.Text at 0x7fd465dbde90>,
(9, 3): <matplotlib.text.Text at 0x7fd465eac410>,
(9, 4): <matplotlib.text.Text at 0x7fd465ed0ad0>,
(9, 5): <matplotlib.text.Text at 0x7fd465d27ad0>,
(9, 6): <matplotlib.text.Text at 0x7fd465d460d0>,
(9, 7): <matplotlib.text.Text at 0x7fd465c7a7d0>,
(9, 8): <matplotlib.text.Text at 0x7fd465de2590>,
(9, 9): <matplotlib.text.Text at 0x7fd465ed9990>

In [27]: plt.show()
```



(Refer Slide Time: 15:14)

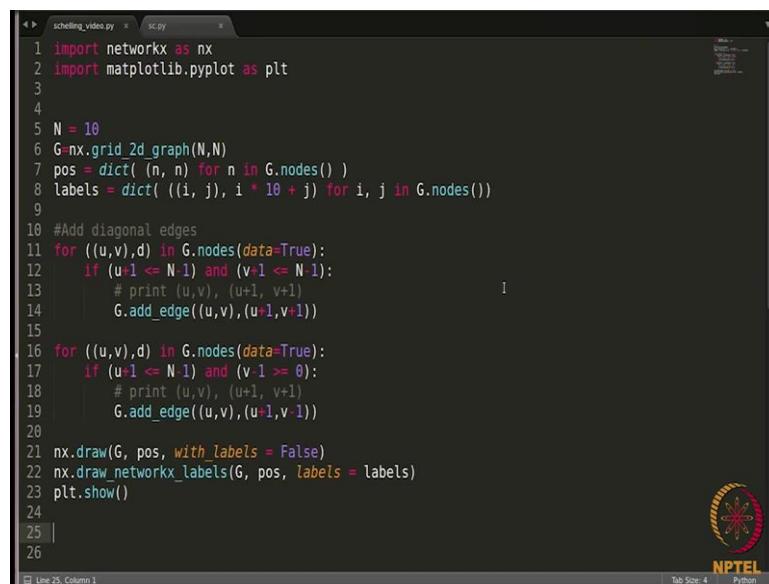


This looks nice. So, we have the great we have the diagonal edges we have the nodes now we have said the playground for playing with these nodes where we will put the people of different types there will be some people of type 0, there will be some people of type 1; there will be some nodes its will be empty we will not put any people there. So, that is how we are going start in the next video I will be using all these commands that I would show shown you already in the ipython console. So, I will use them in the editor and then we will get started with manipulating the nodes.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture – 59**  
**Homophily (Continued) & Positive and Negative Relationships**  
**Schelling Model Implementation – Visualization and Getting a list of boundary & Internal nodes**

(Refer Slide Time: 00:05)



```
schelling_video.py x sc.py x
1 import networkx as nx
2 import matplotlib.pyplot as plt
3
4
5 N = 10
6 G=nx.grid_2d_graph(N,N)
7 pos = dict( (n, n) for n in G.nodes() )
8 labels = dict( ((i, j), i * 10 + j) for i, j in G.nodes())
9
10 #Add diagonal edges
11 for ((u,v),d) in G.nodes(data=True):
12     if (u+1 <= N-1) and (v+1 <= N-1):
13         # print (u,v), (u+1, v+1)
14         G.add_edge((u,v),(u+1,v+1))
15
16 for ((u,v),d) in G.nodes(data=True):
17     if (u+1 <= N-1) and (v-1 >= 0):
18         # print (u,v), (u+1, v-1)
19         G.add_edge((u,v),(u+1,v-1))
20
21 nx.draw(G, pos, with_labels = False)
22 nx.draw_networkx_labels(G, pos, labels = labels)
23 plt.show()
24
25 |
26
```

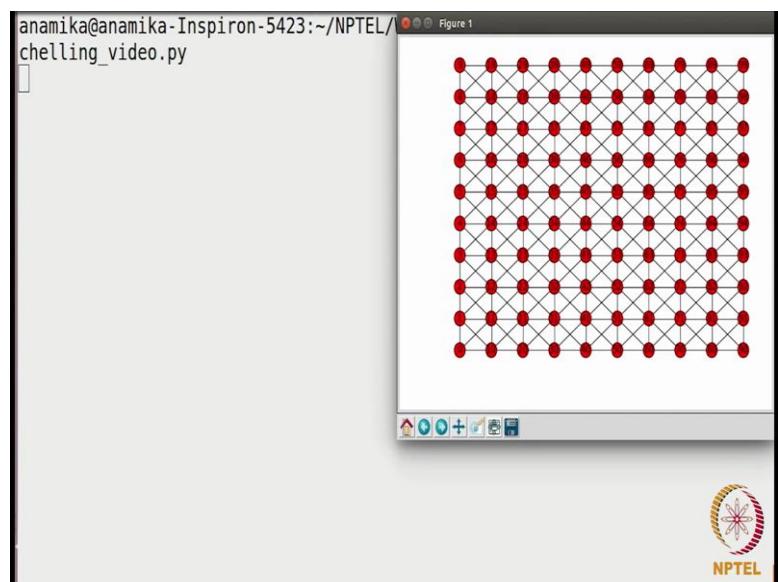
So, I have created this file and whatever we did in ipython console, I copied all that all those commands. So, here; so, there is nothing new here.

(Refer Slide Time: 00:20)

```
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_4_(Homophily)$ python schelling_video.py
```

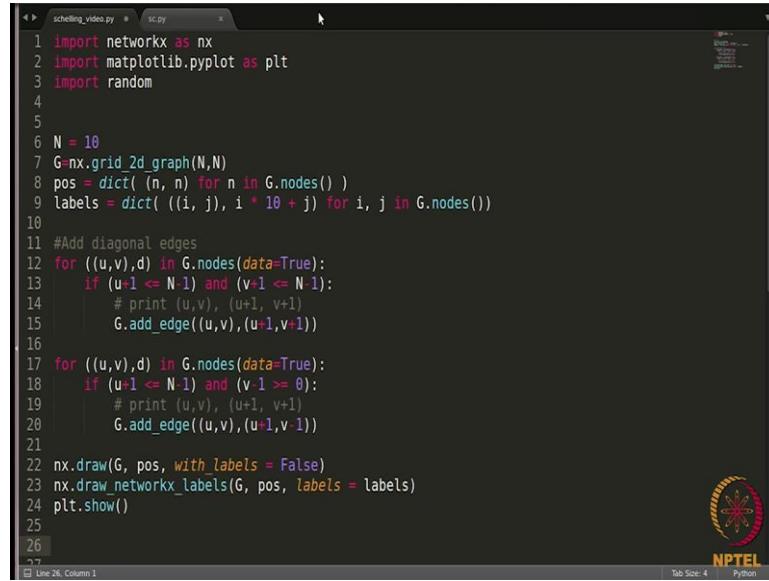


(Refer Slide Time: 00:27)



Let me show you the output of this. So, this is where we left off in the previous video and this is the graph that we were getting now we will continue with from here after having created the network we have to assign people to the nodes. So, as I explained we have to assign people of 2 kinds people of type one people of type 2 and there will be some nodes which will be empty. So, what we can do is we can make use of package random.

(Refer Slide Time: 01:00)

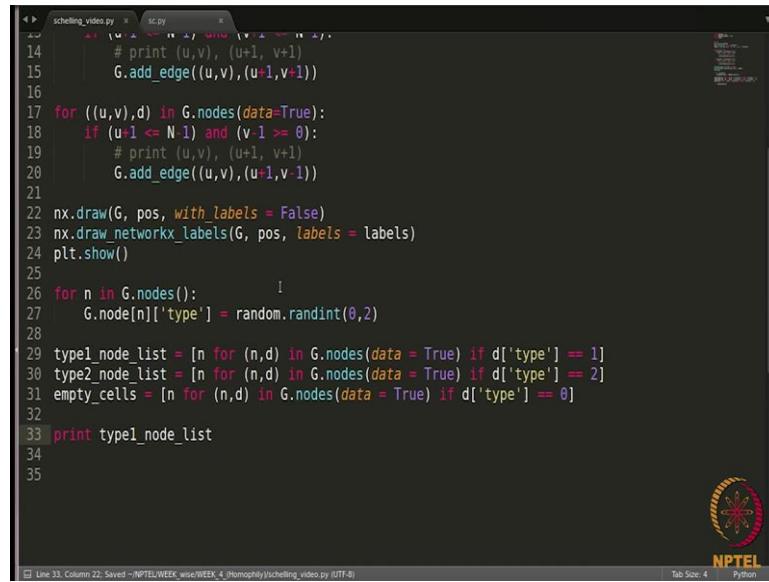


```
1 import networkx as nx
2 import matplotlib.pyplot as plt
3 import random
4
5
6 N = 10
7 G=nx.grid_2d_graph(N,N)
8 pos = dict( (n, n) for n in G.nodes() )
9 labels = dict( ((i, j), i * 10 + j) for i, j in G.nodes())
10
11 #Add diagonal edges
12 for ((u,v),d) in G.nodes(data=True):
13     if (u+1 <= N-1) and (v+1 <= N-1):
14         # print (u,v), (u+1, v+1)
15         G.add_edge((u,v),(u+1,v+1))
16
17 for ((u,v),d) in G.nodes(data=True):
18     if (u+1 <= N-1) and (v-1 >= 0):
19         # print (u,v), (u+1, v-1)
20         G.add_edge((u,v),(u+1,v-1))
21
22 nx.draw(G, pos, with_labels = False)
23 nx.draw_networkx_labels(G, pos, labels = labels)
24 plt.show()
25
26
27
```

So, I will write both random to assign people to nodes what we can do here is that we can assign and attribute that is type to every node in the network.

So, this attribute type will have values from 0 1 or 2 if the type is equal to 0 the node will be considered empty if the type is equal to 1, the node will be considered a person of type 1 if the type = 2 the node will be considered a person of type 2. So, we will be following this convention.

(Refer Slide Time: 01:41)



```
14     # print (u,v), (u+1, v+1)
15     G.add_edge((u,v),(u+1,v+1))
16
17 for ((u,v),d) in G.nodes(data=True):
18     if (u+1 <= N-1) and (v-1 >= 0):
19         # print (u,v), (u+1, v-1)
20         G.add_edge((u,v),(u+1,v-1))
21
22 nx.draw(G, pos, with_labels = False)
23 nx.draw_networkx_labels(G, pos, labels = labels)
24 plt.show()
25
26 for n in G.nodes():
27     G.node[n]['type'] = random.randint(0,2)
28
29 type1_node_list = [n for (n,d) in G.nodes(data = True) if d['type'] == 1]
30 type2_node_list = [n for (n,d) in G.nodes(data = True) if d['type'] == 2]
31 empty_cells = [n for (n,d) in G.nodes(data = True) if d['type'] == 0]
32
33 print type1_node_list
34
35
```

Let me assign the type to the nodes. So, what we can do is for  $n$  in  $G.nodes$ ,  $G.node$ . So, they must assign a type to  $n$ . So, I will write type I want to assign a type randomly. So, I will write `random.randint` and this should re this should go from 0 to 2. So, 0 1 2 any random value could be assigned to the type of to the type of node  $n$ . Now after we have assign the type to every node, we should maintain a list of type one people 2 people and the list of empty nodes or you can say empty cells.

So, we will create three lists for that purpose. So, I will write type one node list sorry is equal to. So, this has to be list and what we can do here is  $n$  for  $(n, d)$  now I will explain what this means  $G.nodes$  (`data = true`) if the type of  $t = 1$ . So, let me tell you what I have done here. So, when you when you can  $G.nodes$  you get only a list of nodes, but in case we have assigned an attribute to the nodes and still we write  $G.nodes$  still we get only the list of nodes we do not get the get the tie we do not get the value of that attitude in case you want the value that attribute you have to explicitly write `data = true`.

So, when you write  $G.nodes$  (`data = true`) you get  $n$  and  $d$  where  $n$  is a node and  $d$  is a dictionary which will contain the attribute and its value. So, since we want to get the type of every node we will write  $G.nodes$  (`data = true`) and that will write us  $(n, d)$  where  $n$  is a node and  $d$  will give us the type. So, we have written if  $d['type'] = 1$ . So, if  $d['type'] = 1$  the node will be of type 1 sorry please pattern by back though out. So, this is for type one node list similarly we will get the type 2 list it just copies paste I am sorry. So, I will write type 2 node list is equal to  $n$  for  $(n, d)$  in  $G.nodes$  that as will true if  $d['type'] = 2$ , right.

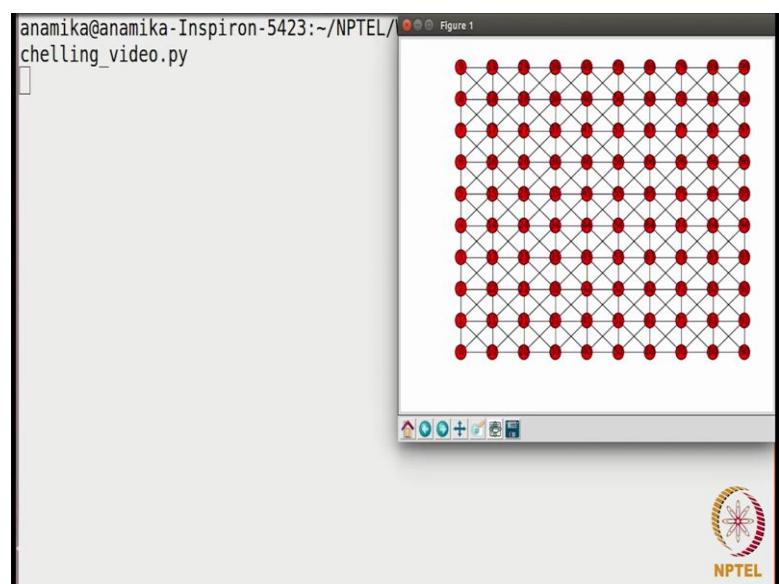
So, similarly we can get a list of empty nodes. So, we can write empty cells is equal to, I will just copy paste where the type = 0. So, they should work we can also verify let me try printing the type one node list type one node list let me check whether it works find.

(Refer Slide Time: 05:07)

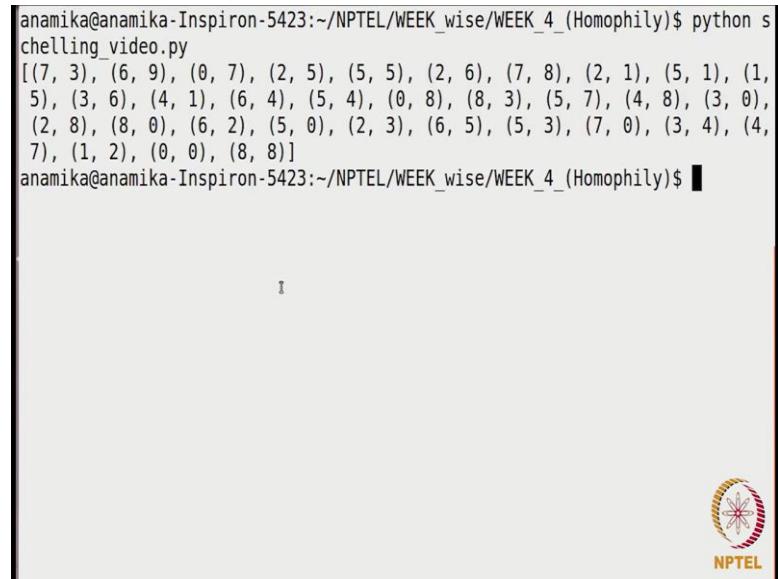
```
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_4_(Homophily)$ python s
chelling_video.py
```



(Refer Slide Time: 05:10)



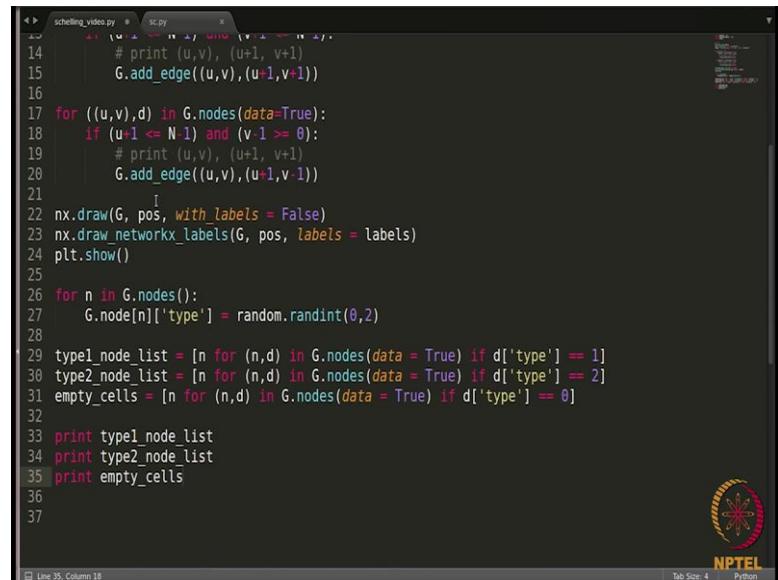
(Refer Slide Time: 05:11)



```
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_4_(Homophily)$ python schelling_video.py
[(7, 3), (6, 9), (0, 7), (2, 5), (5, 5), (2, 6), (7, 8), (2, 1), (5, 1), (1, 5), (3, 6), (4, 1), (6, 4), (5, 4), (0, 8), (8, 3), (5, 7), (4, 8), (3, 0), (2, 8), (8, 0), (6, 2), (5, 0), (2, 3), (6, 5), (5, 3), (7, 0), (3, 4), (4, 7), (1, 2), (0, 0), (8, 8)]
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_4_(Homophily)$
```

So, this is the list of the list of nodes which are n type 1 similarly we can print the other lists as well let me just verify and then we will continue empty cells yeah.

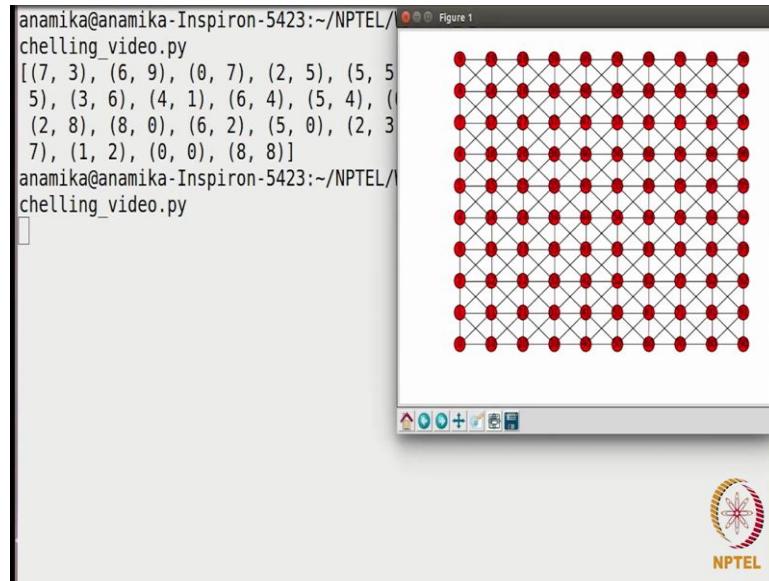
(Refer Slide Time: 05:20)



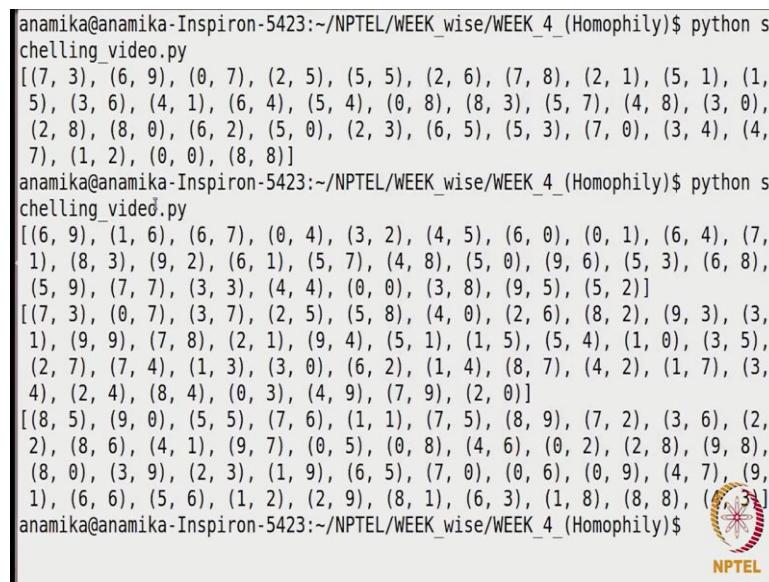
```
<ipython> schelling_video.py # sc.py
14     # print (u,v), (u+1, v+1)
15     G.add_edge((u,v),(u+1,v+1))
16
17 for ((u,v),d) in G.nodes(data=True):
18     if (u+1 <= N-1) and (v+1 >= 0):
19         # print (u,v), (u+1, v+1)
20         G.add_edge((u,v),(u+1,v+1))
21
22 nx.draw(G, pos, with_labels = False)
23 nx.draw_networkx_labels(G, pos, labels = labels)
24 plt.show()
25
26 for n in G.nodes():
27     G.node[n]['type'] = random.randint(0,2)
28
29 type1_node_list = [n for (n,d) in G.nodes(data = True) if d['type'] == 1]
30 type2_node_list = [n for (n,d) in G.nodes(data = True) if d['type'] == 2]
31 empty_cells = [n for (n,d) in G.nodes(data = True) if d['type'] == 0]
32
33 print type1_node_list
34 print type2_node_list
35 print empty_cells
36
37
```

So, it is giving all the three lists we are going to make use of these lists. So, I am going to comment this because we do not need to print.

(Refer Slide Time: 05:32)



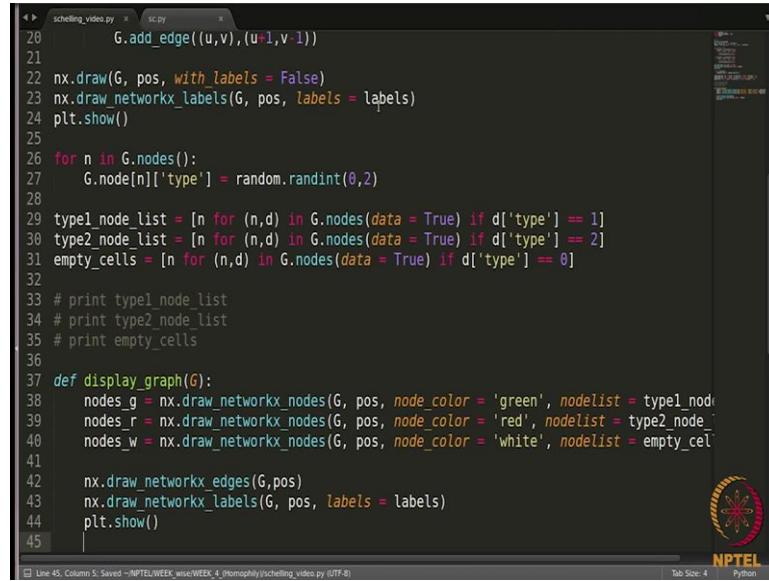
(Refer Slide Time: 05:34)



Now, what we can do is now that we have assigned type to every node. So, way we have 2 different kinds of people in our network it will be nice if he represents them with a different colour.

So, that it is easy to visualise the kind of people that are present in a given cell.

(Refer Slide Time: 06:09)



```
schelling_video.py x sc.py x
20     G.add_edge((u,v),(u+1,v-1))
21
22 nx.draw(G, pos, with_labels = False)
23 nx.draw_networkx_labels(G, pos, labels = labels)
24 plt.show()
25
26 for n in G.nodes():
27     G.nodes[n]['type'] = random.randint(0,2)
28
29 type1_node_list = [n for (n,d) in G.nodes(data = True) if d['type'] == 1]
30 type2_node_list = [n for (n,d) in G.nodes(data = True) if d['type'] == 2]
31 empty_cells = [n for (n,d) in G.nodes(data = True) if d['type'] == 0]
32
33 # print type1_node_list
34 # print type2_node_list
35 # print empty_cells
36
37 def display_graph(G):
38     nodes_g = nx.draw_networkx_nodes(G, pos, node_color = 'green', nodelist = type1_node_list)
39     nodes_r = nx.draw_networkx_nodes(G, pos, node_color = 'red', nodelist = type2_node_list)
40     nodes_w = nx.draw_networkx_nodes(G, pos, node_color = 'white', nodelist = empty_cells)
41
42     nx.draw_networkx_edges(G, pos)
43     nx.draw_networkx_labels(G, pos, labels = labels)
44     plt.show()
45
```

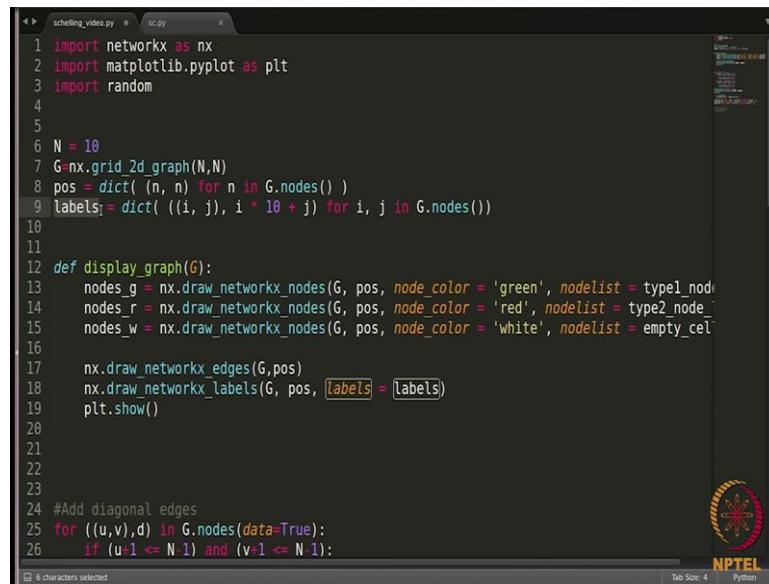
So, let me create a function for that I will create a function say display graph. So, I am just going to decorate the graph here it is up to you how much you want to do here how much you want t, how you want to display how much you want to decorate it is up to you can skip some of the a steps here. If you want who is a what I want is type one node should be displayed in say green colour type 2 nodes should be displayed in say red colour and empty cell should be say display displayed in white colour. So, the function that we will use and it is not draw network x nodes where we can specify a list of nodes that should be displayed by a given colour. So, what we can do is nx.draw networkx nodes the parameters will be G and this pos parameter that we posed already.

And what is to be want I everyone the node colours. So, I will write node colour is equal to green I wanted to be green and then what should be the nodes for which we want to as for whom we want to assign the pink colour. So, I will write node list is equal to type one node list right and going to copy the next kind of nodes we want in red colour. So, I will write nodes are I want in red colour and then type to we are doing this all this here because it will be nice to visualise the kind of transitions that take place. So, all though this is optional step it will be in it will help us to visualise the things I have just to go out the yeah. So, we want empty cells here I will write empty cells and I want why it here.

So, now we have drawn the nodes and, but the edges have not been drawn. So, we have to draw the edges. So, we will write nx.draw networkx edges and there again the 2

parameters we will pos G and pos the positioning since we are since we are manually draw in the nodes and edges labels we will not come. So, again we have to manually drop the labels as well. So, you write nx.draw\_networkx\_labels and again we I think we used at already. So, yeah this one we can just copy this, they should work and anything that we skipped let us see plt.show. So, now, we have to call this function let me put this function on the top so that we can call it.

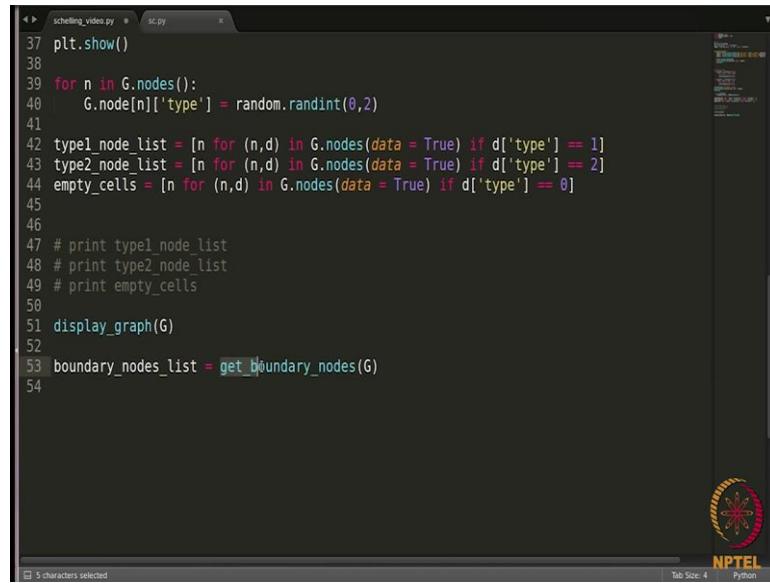
(Refer Slide Time: 09:37)



```
schelling_video.py # sc.py
1 import networkx as nx
2 import matplotlib.pyplot as plt
3 import random
4
5 N = 10
6 G=nx.grid_2d_graph(N,N)
7 pos = dict( (n, n) for n in G.nodes() )
8 labels = dict( ((i, j), i * 10 + j) for i, j in G.nodes() )
9
10
11 def display_graph(G):
12     nodes_g = nx.draw_networkx_nodes(G, pos, node_color = 'green', nodelist = type1_node)
13     nodes_r = nx.draw_networkx_nodes(G, pos, node_color = 'red', nodelist = type2_node)
14     nodes_w = nx.draw_networkx_nodes(G, pos, node_color = 'white', nodelist = empty_cell)
15
16     nx.draw_networkx_edges(G, pos)
17     nx.draw_networkx_labels(G, pos, labels = labels)
18     plt.show()
19
20
21
22
23
24 #Add diagonal edges
25 for ((u,v),d) in G.edges(data=True):
26     if (u+1 <= N-1) and (v+1 <= N-1):
```

So, I will cut it from here and I will right it here this is because we want to make use of labels. So, I want to define the labels dictionary before I declare this function.

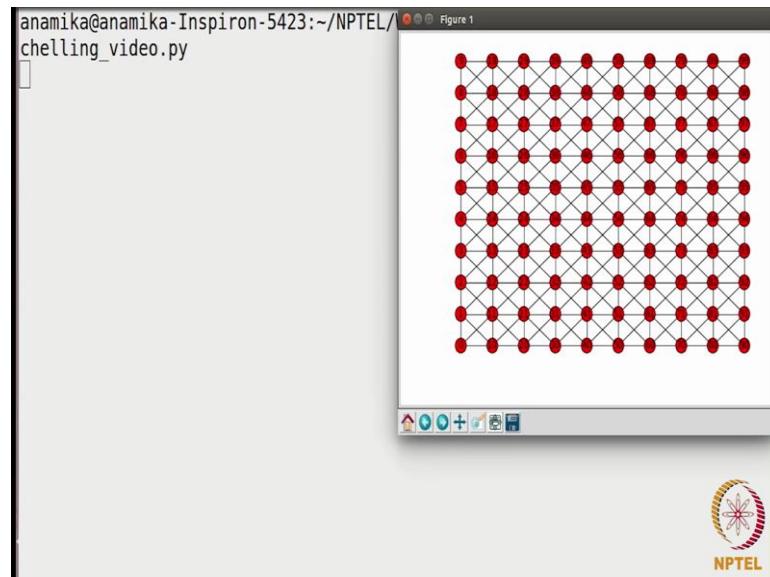
(Refer Slide Time: 09:54)



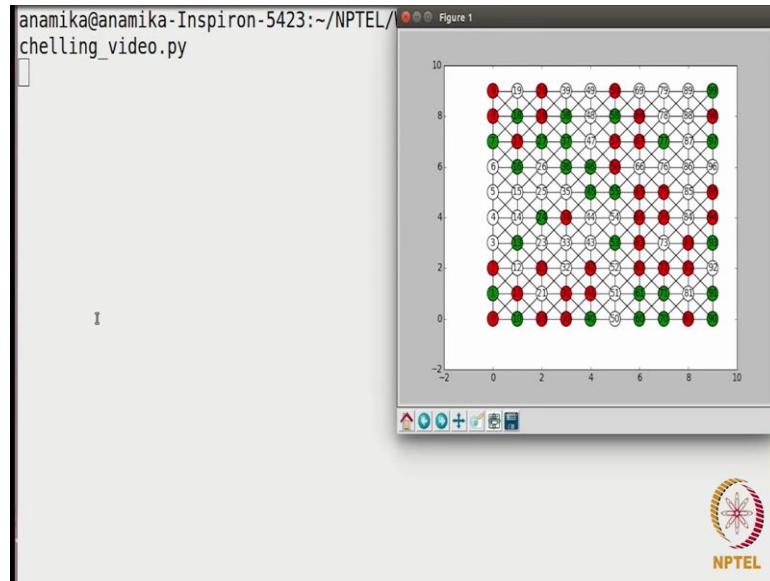
```
37 plt.show()
38
39 for n in G.nodes():
40     G.nodes[n]['type'] = random.randint(0,2)
41
42 type1_node_list = [n for (n,d) in G.nodes(data = True) if d['type'] == 1]
43 type2_node_list = [n for (n,d) in G.nodes(data = True) if d['type'] == 2]
44 empty_cells = [n for (n,d) in G.nodes(data = True) if d['type'] == 0]
45
46
47 # print type1_node_list
48 # print type2_node_list
49 # print empty_cells
50
51 display_graph(G)
52
53 boundary_nodes_list = get_boundary_nodes(G)
54
```

So, I will call this function here a write display graph G let me execute it.

(Refer Slide Time: 10:01)



(Refer Slide Time: 10:05)

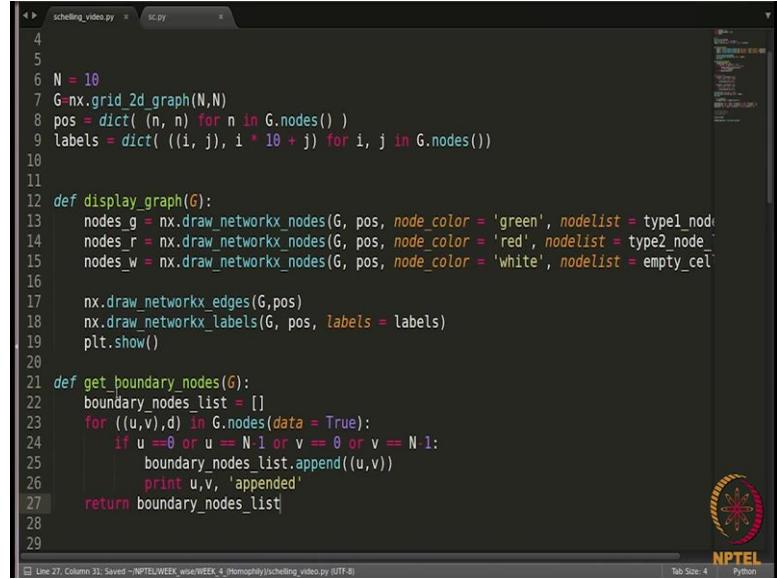


This is the graph that we initially displayed in this is the graph that now we are displaying which looks quite better and we are able to see which nodes belong to which type.

So, it looks pretty good if you want you can change the font size you can change the node size so that all you can play around. So, what is the next step we have the network we have the people. The next step is to identify the people were unsatisfied now how do if how do you find which people are unsatisfied we should be able to know for a given node which are its neighbour right it only then we will be able to you know tell whether this node is satisfied or unsatisfied. Now to be able to know which the neighbours of a are given node we should know whether the node is an internal node or is it a boundary node. So, that we must find out. So, for that matter it will be helpful if we create a list of boundary nodes as well as a list of internal nodes.

So, maybe we can create a function for that to get the list of boundary nodes. So, what we can write is get boundary nodes lets pos the graph here boundary nodes list is equal to this. So, basically we are going to put all the nodes which are aligning in the boundary into this list through this function which is get boundary nodes let us create this functional we will see how we can do that.

(Refer Slide Time: 11:58)



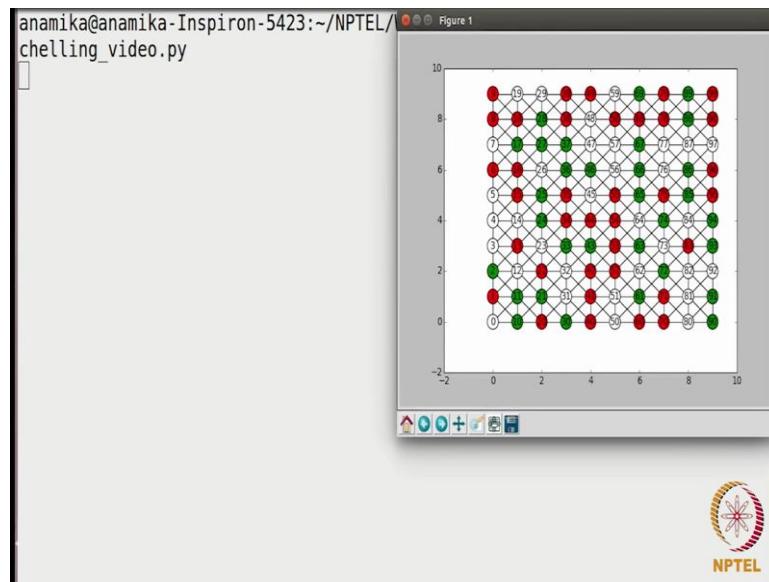
```
4
5
6 N = 10
7 G=nx.grid_2d_graph(N,N)
8 pos = dict( (n, n) for n in G.nodes() )
9 labels = dict( ((i, j), i * 10 + j) for i, j in G.nodes() )
10
11
12 def display_graph(G):
13     nodes_g = nx.draw_networkx_nodes(G, pos, node_color = 'green', nodelist = type1_node)
14     nodes_r = nx.draw_networkx_nodes(G, pos, node_color = 'red', nodelist = type2_node)
15     nodes_w = nx.draw_networkx_nodes(G, pos, node_color = 'white', nodelist = empty_cell)
16
17     nx.draw_networkx_edges(G, pos)
18     nx.draw_networkx_labels(G, pos, labels = labels)
19     plt.show()
20
21 def get_boundary_nodes(G):
22     boundary_nodes_list = []
23     for ((u,v),d) in G.nodes(data = True):
24         if u == 0 or u == N-1 or v == 0 or v == N-1:
25             boundary_nodes_list.append((u,v))
26             print u,v, ' appended'
27     return boundary_nodes_list
28
29
```

So, I will create this function here now how to do this we should know which nodes are boundary nodes if you remember the kind of grid that we were getting you will know that the nodes which have either  $u$  or  $v$  if  $(u, v)$  is a node the nodes which have either  $u$  as 0 or  $v$  as 0 is a boundary node also the nodes which have either  $u$  as  $n - 1$  or  $v$  as  $n$  minus one they are also boundary nodes.

So, you can check the grid and see how it comes out to be. So, let us write this function lets create of lists which we will be writing. So, I will write boundary nodes list is equal to empty. So, I will write for  $u$ ,  $v$  what else is there  $t$  which is a type we can actually skip the type here because we do not need this lets see in  $G.nodes(data = true)$ . So, what I was saying was data is not needed for these particular steps. So, we can skip that is one and we can keep that is one. So, and choosing to keep it just like the now as I told you if  $u = 0$  or  $u = n - 1$  or  $v = 0$  or  $v = n - 1$  then the node is a boundary node.

So, what we will do is the boundary nodes lists.append we are going to append  $u$   $v$  that yeah we should go let me print also it is to cross check whether things are fine or not. So, let us print this whenever a node is appended to the list there will be is this state. So, once that is we are going to written this list. So, we have already called this function yes let us call let us execute. Before that let me remove this displaying of the graph because now we are because now we have created a function and we are getting a better visualizations.

(Refer Slide Time: 14:35)



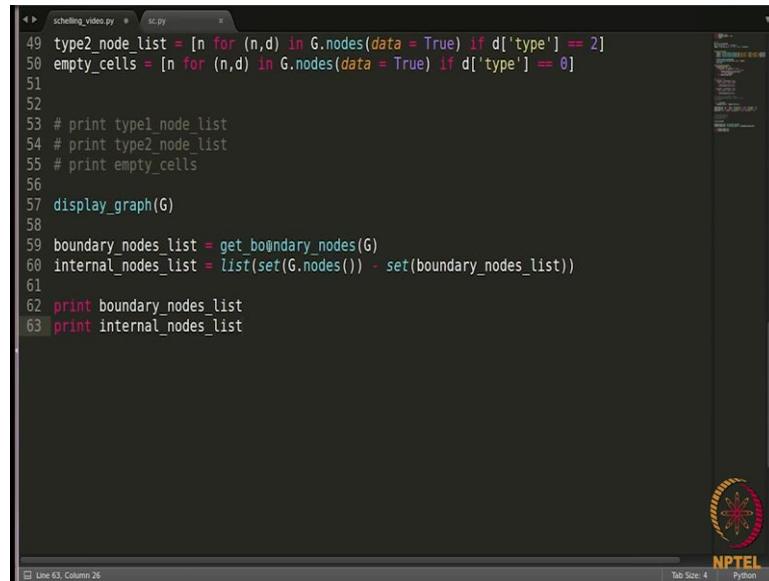
(Refer Slide Time: 14:38)

```
0 2 appended
3 0 appended
9 8 appended
8 0 appended
5 0 appended
3 9 appended
1 9 appended
9 6 appended
7 0 appended
0 6 appended
0 9 appended
5 9 appended
9 1 appended
0 3 appended
4 9 appended
2 9 appended
0 0 appended
7 9 appended
2 0 appended
9 5 appended
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_4_(Homophily)$
```

So, I have commented on that. So, this is the graph I am closing it yeah. So, we are we are getting things working fine these are the nodes which are appended to the list.

Now, the second step is to get the list of the internal nodes right. So, what we can do is if we subtract the boundary nodes from the total nodes; obviously, whatever remains is the list of internal nodes.

(Refer Slide Time: 15:05)



```
49 type2_node_list = [n for (n,d) in G.nodes(data = True) if d['type'] == 2]
50 empty_cells = [n for (n,d) in G.nodes(data = True) if d['type'] == 0]
51
52
53 # print type1_node_list
54 # print type2_node_list
55 # print empty_cells
56
57 display_graph(G)
58
59 boundary_nodes_list = get_boundary_nodes(G)
60 internal_nodes_list = list(set(G.nodes()) - set(boundary_nodes_list))
61
62 print boundary_nodes_list
63 print internal_nodes_list
```

So, what we can do is I will create a list for internal nodes internal nodes list is equal to; now what is this I told you this technique if you want to subtract one list from the other you can use is set function. So, I am going to convert list of all the nodes into a set. So, I will write G.nodes; yeah minus set or should I write boundary nodes list yeah this work. So, let me try printing these 2 lists just to cross it.

So, I will write print boundary nodes list and print internal nodes list yeah let me command that appending. So, that we get only what we want to print (Refer Time: 16:08) that yeah in this function I am going to commentary this let's check yeah. So, we are getting 2 lists here first lists are boundary node list and second list is internal nodes. So, we are doing well.

(Refer Slide Time: 16:20)

```
0 3 appended
4 9 appended
2 9 appended
0 0 appended
7 9 appended
2 0 appended
9 5 appended
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_4_(Homophily)$ python s
chelling_video.py
[(6, 9), (0, 7), (4, 0), (9, 0), (0, 4), (9, 3), (6, 0), (0, 1), (9, 9), (8,
9), (9, 4), (9, 7), (0, 5), (1, 0), (0, 8), (9, 2), (0, 2), (3, 0), (9, 8),
(8, 0), (5, 0), (3, 9), (1, 9), (9, 6), (7, 0), (0, 6), (0, 9), (5, 9), (9,
1), (0, 3), (4, 9), (2, 9), (0, 0), (7, 9), (2, 0), (9, 5)]
[(7, 3), (4, 7), (5, 7), (4, 8), (5, 6), (2, 8), (5, 4), (2, 1), (1, 3), (1,
6), (3, 7), (5, 1), (2, 5), (8, 5), (7, 2), (1, 2), (3, 8), (6, 7), (5, 5),
(8, 1), (7, 6), (6, 6), (4, 4), (6, 3), (1, 5), (8, 8), (3, 3), (5, 8), (3,
6), (2, 2), (8, 6), (5, 3), (4, 1), (1, 1), (6, 4), (3, 2), (2, 6), (8, 2),
(7, 1), (4, 5), (1, 4), (7, 7), (7, 5), (2, 3), (8, 7), (6, 8), (4, 2), (6,
5), (3, 5), (2, 7), (8, 3), (4, 6), (3, 4), (6, 1), (3, 1), (5, 2), (7, 4),
(1, 8), (6, 2), (4, 3), (1, 7), (7, 8), (2, 4), (8, 4)]
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_4_(Homophily)$
```

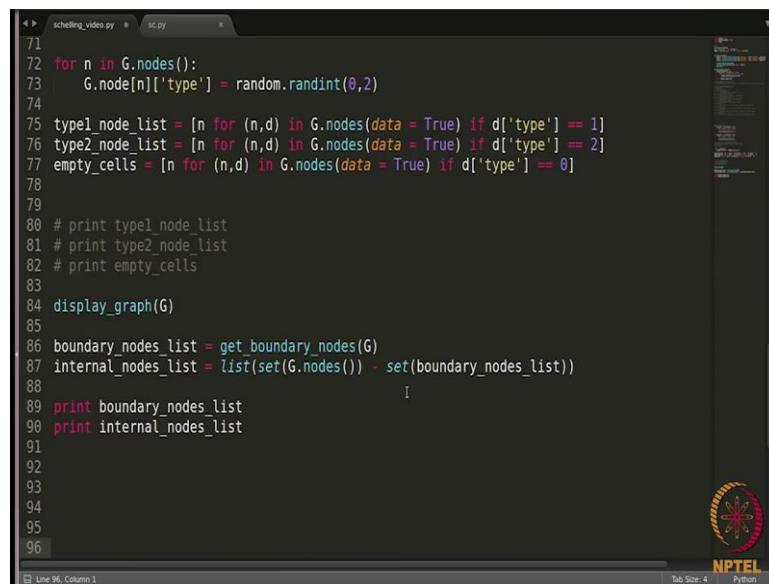


So, now we have a list of boundary nodes internal nodes what is the next step the next step is to identify the unsatisfied nodes and then moving them to a new location we will do that in the next part of the video.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture – 60**  
**Homophily (Continued) & Positive and Negative Relationships**  
**Schelling Model Implementation – Getting a list of unsatisfied nodes**

(Refer Slide Time: 00:06)



The screenshot shows a Jupyter Notebook interface with two tabs: 'schelling\_video.py' and 'sc.py'. The code in 'schelling\_video.py' is as follows:

```
71
72 for n in G.nodes():
73     G.node[n]['type'] = random.randint(0,2)
74
75 type1_node_list = [n for (n,d) in G.nodes(data = True) if d['type'] == 1]
76 type2_node_list = [n for (n,d) in G.nodes(data = True) if d['type'] == 2]
77 empty_cells = [n for (n,d) in G.nodes(data = True) if d['type'] == 0]
78
79
80 # print type1_node_list
81 # print type2_node_list
82 # print empty_cells
83
84 display_graph(G)
85
86 boundary_nodes_list = get_boundary_nodes(G)
87 internal_nodes_list = list(set(G.nodes()) - set(boundary_nodes_list))
88
89 print boundary_nodes_list
90 print internal_nodes_list
91
92
93
94
95
96
```

The code initializes node types, creates lists for type 1, type 2, and empty cells, prints the graph, and then defines boundary and internal nodes.

In the previous video we computed a list of boundary nodes and we computed a list of internal nodes as well. Now what is the next step? We have to find a list of unsatisfied nodes so that we can move to a new location one by one. Now how do we get a list of unsatisfied nodes? So, basically, we will take all the nodes from `G.nodes` one by one and we will test whether they are unsatisfied or not. Now how do we do that? We need to get a list of neighbours of that node if it is a boundary node it will have a different set of neighbours if it is an internal node it will have a different set of neighbours.

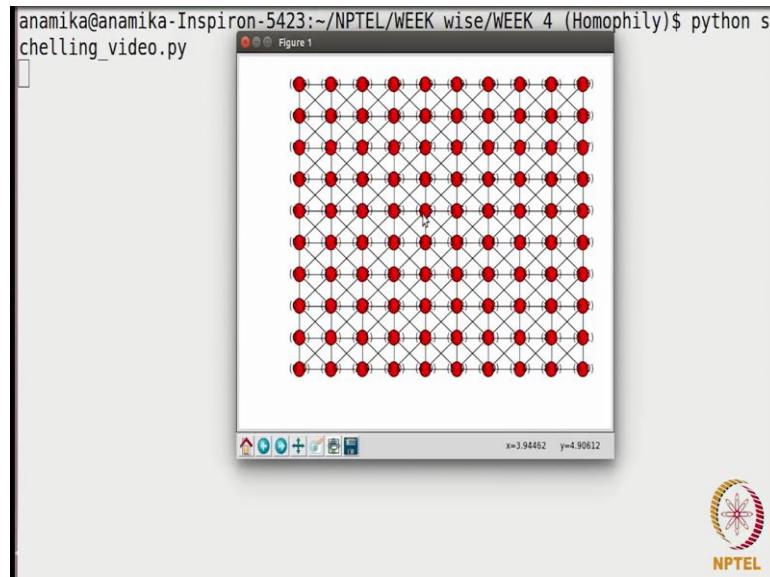
So, the first step is to get a list of neighbours for the given node. So, we might need to create 2 functions for that one function for getting a list of neighbours if the node is a boundary node and another function for getting a list of neighbours if the node is an internal node. Now let me show you how we can do that let me comment this I want the previous version of the graph here.

(Refer Slide Time: 01:20)

```
51
52
53
54
55
56
57 #Add diagonal edges
58 for ((u,v),d) in G.nodes(data=True):
59     if (u+1 <= N-1) and (v+1 <= N-1):
60         # print (u,v), (u+1, v+1)
61         G.add_edge((u,v),(u+1,v+1))
62
63 for ((u,v),d) in G.nodes(data=True):
64     if (u+1 <= N-1) and (v-1 >= 0):
65         # print (u,v), (u+1, v-1)
66         G.add_edge((u,v),(u+1,v-1))
67
68 nx.draw(G, pos, with_labels = False)
69 # nx.draw_networkx_labels(G, pos, labels = labels)
70 # plt.show()
71
72 nx.draw(G, pos)
73 plt.show()
74
75
76
```

So, let me show you that once, I will just draw the graph the normal way here. So, I will write `nx.draw(G)` and `pos` position also I want to `pos` and then `plt.show`.

(Refer Slide Time: 01:37)



Now, let us see what we get yeah. So, see if it is an internal node for example, let us take the case of 4, 5.

Now, what are its neighbours 4, 5 has 8 neighbours now what are these; the left one that is 3, 5 the right one that is 5, 5 the bottom one the top one now how do we get them. So, given a node  $(u, v)$  how do we get the neighbours  $(u, v)$  we will have a neighbour which

is  $(u - 1, v)$  it will have a neighbour which is  $(u + 1, v)$  it will have a neighbour which is  $(u, v + 1)$  it will have another neighbour which should be  $(u, v - 1)$ . Now next come the diagonal nodes. So, they will be  $(u - 1, v + 1)$ ,  $(u + 1, v - 1)$ ,  $(u + 1, v + 1)$ ,  $(u - 1, v - 1)$ .

So, whatever I told you are going to be the neighbours of an internal node which is  $(u, v)$ . So, that was about the internal nodes. So, we can write a function where we pos  $(u, v)$  and we get the list of all the neighbours now if the node is a boundary node it will have different set of neighbours. So, we need to create a separate function for that now in case again there are 2 kinds of boundary nodes the nodes which are in the corner the 4 nodes and the rest of the nodes now again there will be special cases there if the node is  $0, 0$  I do not think they can generalize that. So, I think they going to have to do that case by case.

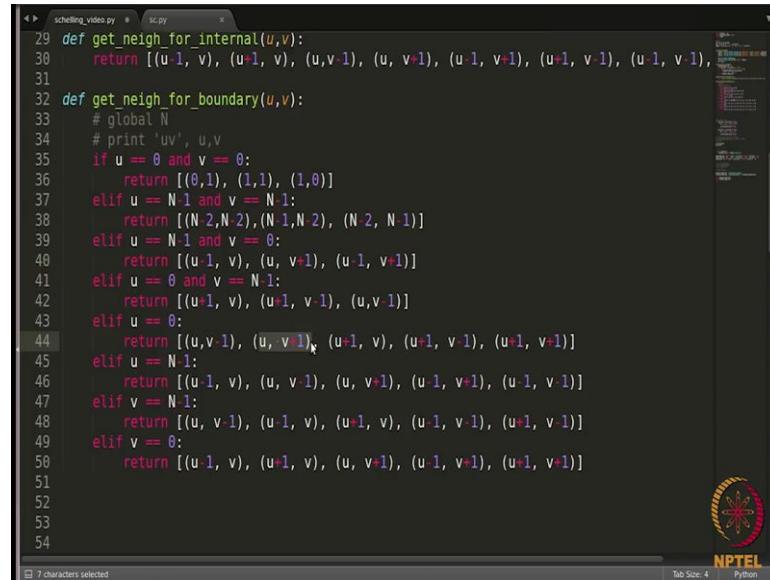
So, if the node is  $0, 0$  its neighbours will be  $0 1 1 1 1 0$  if the node is a nine ,  $0$ . So, if you want to generalize you can just write if the node is  $N - 1, 0$ . So, you remember we passed  $N = 10$  here. So,  $N - 1, 0$  is going to be corner node similarly  $N - 1, N - 1$  is going to be in other corner node and  $0, N - 1$  is going to be another corner node now these for special cases for which the neighbours are going to be only 3. So, we can write these special cases there the rest of the boundary nodes are these once. So, one list of one chunk of boundary nodes are going to be the once where  $u$  is equal to 0 right and in those cases the boundary the neighbours are going to be  $1 2 3 4 5$ .

So, if it is  $0, v$  it will be  $(0, v + 1), (0, v - 1), (1, v + 1), (1, v), (1, v - 1)$  you see how I am doing that I think you can do that for the rest of the cases where let me give you one more example. So, if it is amongst the nodes where  $v$  is equal to and minus 1 as you can see it will give us this row of boundary nodes right. So, again and those cases you can write the and the neighbours that are going to be there and this is going to be the row where  $u = N - 1$  and this is going to be the row where  $v = 0$ . So, you can write all those special cases and you can get a list of the neighbours for the given boundary node.

Let us go back now to save time I have already written all the cases. So, that we do not spend a lot of time in the video making. So, let me show you all though I have already explained to you. So, this is the function get neighbours for an internal node. So, we are posing  $(u, v)$  and for  $N$  internal node the cases are very straight forward. So, then you completely generalized it. So, you are you are returning a list there which is having all

the 8 neighbours. So, that is straight forward another function that I have created already is forgetting a list of neighbours for a boundary node as I told you.

(Refer Slide Time: 06:02)



```
29 def get_neigh_for_internal(u,v):
30     return [(u-1, v), (u+1, v), (u, v-1), (u, v+1), (u-1, v+1), (u+1, v-1), (u-1, v-1),
31
32 def get_neigh_for_boundary(u,v):
33     # global N
34     # print 'uv', u,v
35     if u == 0 and v == 0:
36         return [(0,1), (1,1), (1,0)]
37     elif u == N-1 and v == N-1:
38         return [(N-2,N-2),(N-1,N-2), (N-2, N-1)]
39     elif u == N-1 and v == 0:
40         return [(u-1, v), (u, v+1), (u-1, v+1)]
41     elif u == 0 and v == N-1:
42         return [(u+1, v), (u+1, v-1), (u,v-1)]
43     elif u == 0:
44         return [(u,v-1), (u, v+1), (u+1, v), (u+1, v+1)]
45     elif u == N-1:
46         return [(u-1, v), (u, v-1), (u, v+1), (u-1, v-1)]
47     elif v == N-1:
48         return [(u, v-1), (u-1, v), (u+1, v), (u-1, v-1), (u+1, v-1)]
49     elif v == 0:
50         return [(u-1, v), (u+1, v), (u, v+1), (u+1, v+1), (u-1, v+1)]
51
52
53
54
```

There are 1 2 3 4 8 cases. So, this you can get. So, there are this also I comment. So, you see there are 8 cases here 1 2 3 4 5 6 8 cases all the cases that their as I told you first 4 cases are the ones for the corner nodes and the rest 4 cases are the ones for the rest of the boundary nodes.

So, I do not think I need to explain much there. In fact, I would like you to try yourself N case you need help we can just check the values here. So, these are all possible values you can just open the graph the way I opened and then you can note down the neighbours are going to be there that that should be simple to do now we have to get a list of unsatisfied nodes.

(Refer Slide Time: 06:56)

```
48     return [(u, v-1), (u-1, v), (u+1, v), (u-1, v-1), (u+1, v-1)]
49 elif v == 0:
50     return [(u-1, v), (u+1, v), (u, v+1), (u-1, v+1), (u+1, v+1)]
51
52 def get_unsatisfied_nodes_list(G, internal_nodes_list, boundary_nodes_list):
53     unsatisfied_nodes_list = []
54     t = 3
55     for u,v in G.nodes():
56         type_of_this_node = G.node[(u,v)]['type']
57         if type_of_this_node == 0:
58             continue
59         else:
60             similar_nodes = 0
61             if (u,v) in internal_nodes_list:
62                 neigh = get_neigh_for_internal(u,v)
63             elif (u,v) in boundary_nodes_list:
64                 neigh = get_neigh_for_boundary(u,v)
65
66             for each in neigh:
67                 if G.node[each]['type'] == type_of_this_node:
68                     similar_nodes += 1
69
70             if similar_nodes <= t:
71                 unsatisfied_nodes_list.append((u,v))
72
73     return unsatisfied_nodes_list
```

So, we will create a function for that we will write get unsatisfied nodes list lets pos graph here. Now we will check all the nodes one by one and we will check how many neighbours of the given node have the same type as the as the type of the given node right. So, let us create a list; unsatisfied nodes list and (Refer Time: 07:34) initially now we have to check for each node. So, we will write for  $(u, v)$  in  $G.nodes$ .

We have to keep a node of the type of this node that is  $(u, v)$ . So, we can write type of this node because we will be comparing write type of this node is equal to  $G.node$ , we have to see the type of  $u$   $v$ . So, you write  $u$   $v$  and then type here there should give us the type of the node. So, we have stop here in case the type of this node is equal to (Refer Time: 08:18) what are we going to do we basically do not have to do anything as in if the type of this node is equal to 0 it can never be unsatisfied right. So, we will just continue, and we will check the next node. So, we will write if type of this node is equal to 0 we just continue and check the next node as for now we are going to do.

We are going to see in the list of neighbours of this node how many nodes are having a similar type as this nodes type. So, we will have to keep drag of that. So, we will write similar nodes is equal to 0 initiate initially now we have to check whether this node  $u$   $v$  is a internal node or is it a boundary node because accordingly we will get the list of neighbours. So, we have to check here now do we check whether or node is internal node

or boundary node we have already created a list of internal, internal nodes in boundary nodes here, right.

(Refer Slide Time: 09:22)

```
97
98 for n in G.nodes():
99     G.node[n]['type'] = random.randint(0,2)
100
101 type1_node_list = [n for (n,d) in G.nodes(data = True) if d['type'] == 1]
102 type2_node_list = [n for (n,d) in G.nodes(data = True) if d['type'] == 2]
103 empty_cells = [n for (n,d) in G.nodes(data = True) if d['type'] == 0]
104
105
106 # print type1_node_list
107 # print type2_node_list
108 # print empty_cells
109
110 # display_graph(G)
111
112 boundary_nodes_list = get_boundary_nodes(G)
113 internal_nodes_list = list(set(G.nodes()) - set(boundary_nodes_list))
114
115 # print boundary_nodes_list
116 # print internal_nodes_list
117
118 unsatisfied_nodes_list = get_unsatisfied_nodes_list(G, internal_nodes_list, boundary_nodes_list)
119 print unsatisfied_nodes_list
120
121 make_a_node_satisfied(unsatisfied_nodes_list, empty_cells)
122
```

So, it will be nice if you pass these lists into the function `boundary_nodes_list`. So, we will pass these 2 lists in the function and we will pass these lists and we will check whether the node belongs to this list or not that we will tell us whether the node is boundary node or internal node.

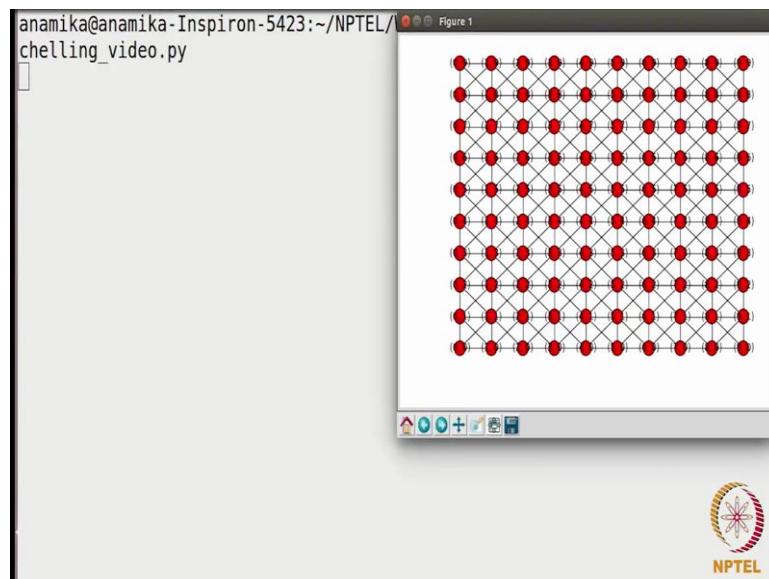
`internal_nodes_list`: so, what we have to do here we will check if  $(u, v)$  sorry in `internal_nodes_list` which will mean that it will be an internal node then its neighbours. So, you must get all its neighbours how we get its neighbours we have already created a function for getting the neighbours of an internal node we will write `get_neighbours` for internal, right. So, we are going to pass the node here in case this  $u$  is not an `internal_nodes_list` it will (Refer Time: 10:33) in the other list that is `boundary_nodes_lists`. So, we will write for  $(u, v)$  in `boundary_nodes_list` what do we do now we get a list of neighbours again we have created a function for that `get_neighbour` of boundary right for boundary yeah `get_neighbour` for boundary and we will pass the node here you will.

So, now we have got a list of neighbours what do we have to do now we have to check amongst these neighbours how many of them are having the same type as the type of this node that is what we have stored already. So, we will start a loop for all the nodes in the

neighbour. So, you write for each in neighbour if G.node. So, we have to check the type of each. So, I will write if I am sorry if this is equal to type of this node, we have we have to increment the counter that is we already initial right similar nodes plus is equal to one. So, number of similar loads nodes well be incremented after this loop is over, we will have a count of the similar nodes. So, if that similar node is less than equal to the threshold. So, maybe you can initialize the threshold here maybe  $t = 3$ . I will write here.

So, if the same number similar node is less than equal to  $t$  this should mean that the node is unsatisfied. So, we will at this append this node to the list of unsatisfied nodes. So, write unsatisfied node list or append u v. u v and this function is going to write the list of unsatisfied nodes I will write return unsatisfied nodes list. Now let us call this function here. So, we will write unsatisfied nodes list is equal to get unsatisfied no nodes list and the parameters are going to be the graph and internal\_nodes\_list and boundary\_nodes\_list yeah in order to verify let us print whether it works it works fine or not. So, I will print unsatisfied nodes list let me comment this I do not want to see them. So, like see. So, this is the graph we are getting the list of unsatisfied nodes here. So, we are doing good. Now what we have to do?

(Refer Slide Time: 13:19)



(Refer Slide Time: 13:22)

```
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_4_(Homophily)$ python s
chelling_video.py
[(0, 7), (1, 6), (5, 8), (4, 0), (5, 5), (7, 6), (0, 4), (1, 1), (2, 6), (9,
3), (6, 0), (7, 5), (0, 1), (3, 1), (9, 9), (8, 9), (9, 4), (7, 2), (1, 5),
(3, 6), (9, 7), (0, 8), (4, 6), (9, 2), (6, 1), (5, 7), (7, 4), (0, 2), (1,
3), (3, 0), (2, 8), (9, 8), (1, 4), (3, 9), (2, 3), (9, 6), (6, 5), (3, 4),
(2, 4), (4, 7), (9, 1), (6, 6), (7, 7), (0, 3), (4, 9), (3, 3), (8, 1), (7,
9), (2, 0), (8, 8), (9, 5)]
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_4_(Homophily)$
```

I



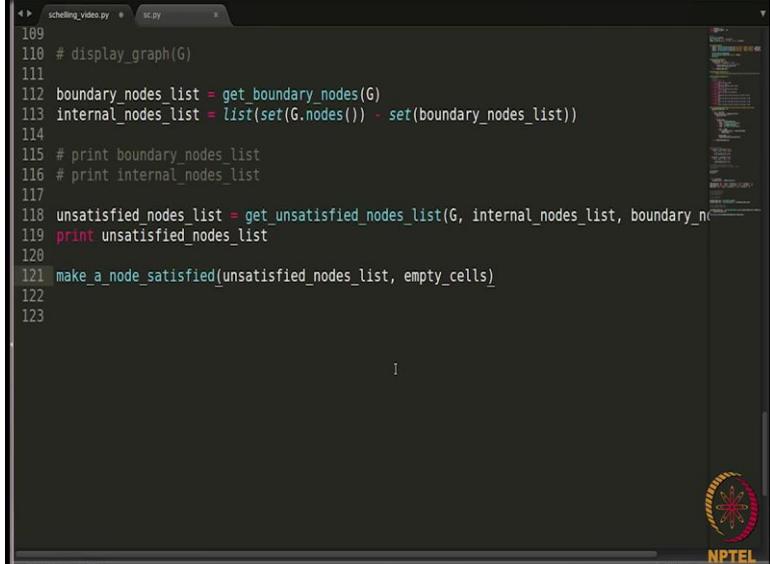
After getting a list of unsatisfied nodes we have to take the nodes take one of the unsatisfied nodes and we have to move it to a new place now that new place should be empty, right. So, we also have to keep a track of the empty places we initially did that we kept a list of empty cells. So, that we have already. So, what we will do is I will go back what we will do is from this list of unsatisfied nodes we will choose one node randomly from the list of empty cells we will choose one place randomly we and then we will put the randomly chosen node from the unsatisfied list into this randomly chosen position from the empty cells and after that; obviously, the number of unsatisfied and satisfied nodes we will change. So, the graph we will actually change after that we will re compute all the values that is unsatisfied empty satisfied everything will be re computed and then we will see how the graph looks like this we will keep on repeating for several iterations and in the end we will compare how we are how the graph is looking like and how the graph is looking like initially.

So, as I said the next step is to make or node satisfied we can create a function for that make a node satisfied what do we need to pass their we; obviously, need to pass the list of unsatisfied nodes because we will randomly choose one out of that and second thing that has to pass just the anti cells list because we will run issues one position out of that I think this is good. So, in the next video, we will implement this function make a nodes satisfied.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture – 61**  
**Homophily (Continued) & Positive and Negative Relationships**  
**Schelling Model Implementation - Shifting the Unsatisfied Nodes & Visualizing the Final Graph**

(Refer Slide Time: 00:05)



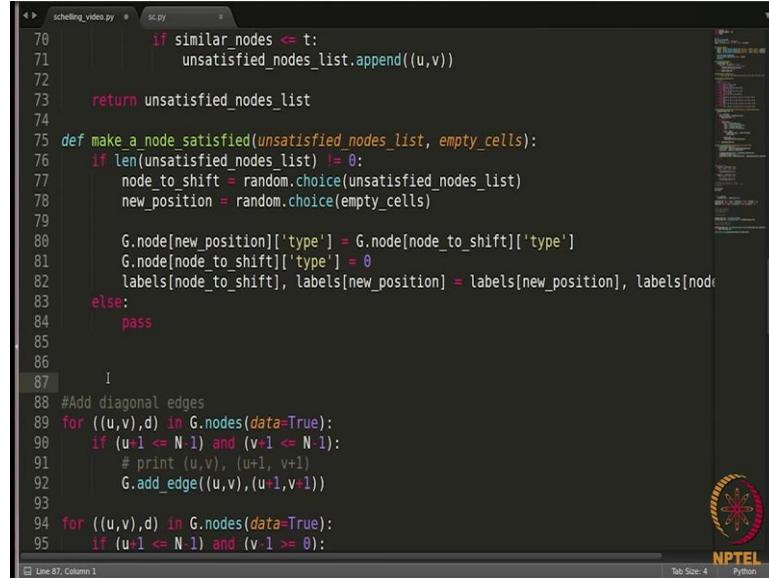
The screenshot shows a Jupyter Notebook interface with two code cells. The first cell contains the following Python code:

```
109
110 # display_graph(G)
111
112 boundary_nodes_list = get_boundary_nodes(G)
113 internal_nodes_list = list(set(G.nodes()) - set(boundary_nodes_list))
114
115 # print boundary_nodes_list
116 # print internal_nodes_list
117
118 unsatisfied_nodes_list = get_unsatisfied_nodes_list(G, internal_nodes_list, boundary_nodes_list)
119 print unsatisfied_nodes_list
120
121 make_a_node_satisfied(unsatisfied_nodes_list, empty_cells)
122
123
```

The second cell is currently empty, indicated by a large white area.

We now have list of unsatisfied nodes which we completed in the previous video and we have to now choose one node out of that list and we have to make it satisfied; how do we do that we have to move that node to an empty cell which also we will be randomly choosing. So, let us create this function.

(Refer Slide Time: 00:35)



```
70     if similar_nodes <= t:
71         unsatisfied_nodes_list.append((u,v))
72
73     return unsatisfied_nodes_list
74
75 def make_a_node_satisfied(unsatisfied_nodes_list, empty_cells):
76     if len(unsatisfied_nodes_list) != 0:
77         node_to_shift = random.choice(unsatisfied_nodes_list)
78         new_position = random.choice(empty_cells)
79
80         G.node[new_position]['type'] = G.node[node_to_shift]['type']
81         G.node[node_to_shift]['type'] = 0
82         labels[node_to_shift], labels[new_position] = labels[new_position], labels[node_to_shift]
83     else:
84         pass
85
86
87     I
88 #Add diagonal edges
89 for ((u,v),d) in G.nodes(data=True):
90     if (u+1 <= N-1) and (v+1 <= N-1):
91         # print (u,v), (u+1, v+1)
92         G.add_edge((u,v),(u+1,v+1))
93
94 for ((u,v),d) in G.nodes(data=True):
95     if (u+1 <= N-1) and (v-1 >= 0):
```

So, let us create this function, here I will write this. So, in case the list is empty we will I add just one condition here if length of unsatisfied nodes list is not equal to 0 then we will do or this and what is that we have basically have to choose one node randomly out of this list we will make use of the function rand dot choice.

So, let us write a node to shift the one we which we will be shifting in this particular iteration. So, in one iteration we will be shifting 1 node. So, we will choose the node randomly random dot choice unsatisfied nodes list now where do we shift this node again, we have to choose one empty location randomly. So, maybe we can write new position is equal to random dot choice from empty cells now we got the node and we have to move it what kind of changes we will happen when a node is shifted. So, basically the, so, here we have on the left-hand side assume we have a node which is to be shifted on the right-hand side we have another node which is empty.

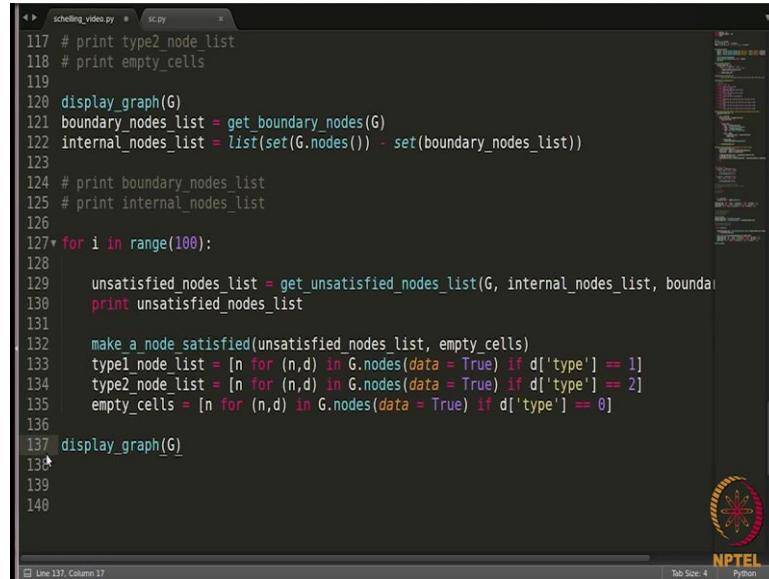
So, the type of the empty node is right now 0. So, the type of this empty node will change to the type of the node which is now coming into this cell that is one thing also when we look at the left hand side when this node is shifted this location will become empty. So, the type of this node will become 0. So, this changing of type will happen as the first thing second thing that has to happen is the changing of the labels. So, the label of the node on the left-hand side and the node on the right-hand side will be exchange will be exchange yeah basically. So, I hope you are able to imagine how I am going to

explain you. So, let us first change the type. So, I am going to change the type of the empty cell to the type of a node which is going to be shifted. So, I will write G dot node new position is basically the location of the news of the empty cell type is going to be equal to the type of the node which is node to shift.

So, now we have change the type of the empty cell to the type of the node which is going to be coming into this cell and the node which got shifted will now be 0 as type. So, I will write G.node to shift it is type will now be 0 done second thing that has to be done is to exchange the labels. So, in python exchanging is very straight forward 1 1 simple command we can just do; now where are we storing the labels if you remember initially, we created this dictionary that is labels, right. So, this dictionary is having all the labels where this is where we have to make changes. So, in this dictionary labels there will be a label for node to shift there will be and in this dictionary labels there will be a label for the node new position already we have to exchange that.

So, let me I am sorry how do we do it x comma y is equal to y comma x as implies that this is how you exchange contents of two variables this no need for any temporary variable. So, that is this while do is well labels node to shift comma labels new position is equal to labels new position comma labels node to shift I think where done in the function let me write the else part of this if. So, in case the there is nothing in the unsatisfied nodes list basically all the nodes are satisfied we are not going to do anything. So, just we will write a pass nothing to be done. So, we are done with this function which is make a node satisfied now let us get back, here we call this function make a node satisfied after which the list of type a nodes and the list of type two nodes and the list of empty cells we will change and we will need the those lists again. So, they need to update them. So, what is going to write is type one I think we did that in this copy paste let me yeah.

(Refer Slide Time: 06:02)

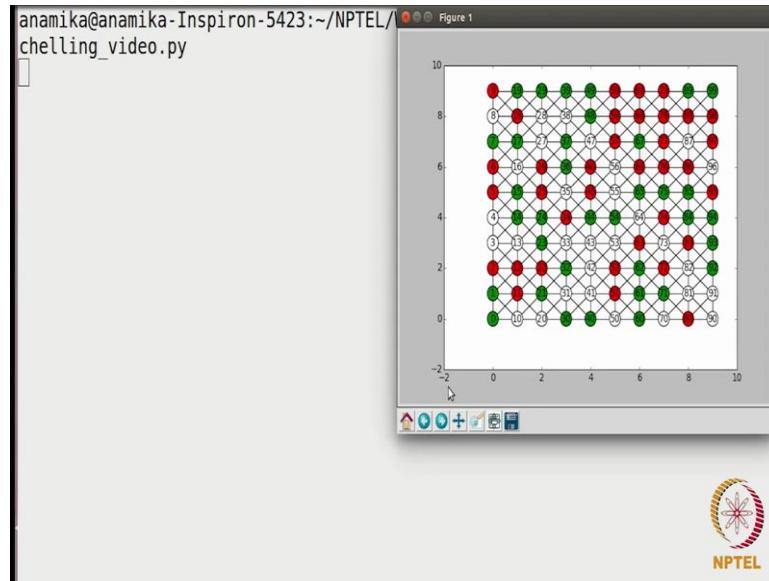


```
117 # print type2_node_list
118 # print empty_cells
119
120 display_graph(G)
121 boundary_nodes_list = get_boundary_nodes(G)
122 internal_nodes_list = list(set(G.nodes()) - set(boundary_nodes_list))
123
124 # print boundary_nodes_list
125 # print internal_nodes_list
126
127 for i in range(100):
128
129     unsatisfied_nodes_list = get_unsatisfied_nodes_list(G, internal_nodes_list, boundary_nodes_list)
130     print unsatisfied_nodes_list
131
132     make_a_node_satisfied(unsatisfied_nodes_list, empty_cells)
133     type1_node_list = [n for (n,d) in G.nodes(data = True) if d['type'] == 1]
134     type2_node_list = [n for (n,d) in G.nodes(data = True) if d['type'] == 2]
135     empty_cells = [n for (n,d) in G.nodes(data = True) if d['type'] == 0]
136
137 display_graph(G)
138
139
140
```

So, this is where we computed these 3-list type one node list type two node list empty cells now after node is shifted to another location these lists are going to undergo changes is well. So, we need to re compute them. So, I am just copying pasting here over here now how many times we do and then do we have to do it as you see make a node satisfied only makes one nodes satisfied. So, we have to keep doing this for all the nodes which are unsatisfied. So, maybe we can start a loop here. So, I will right for I in range you can change recording or you can keep a number over here which is dependent on the total number of nodes in the grid I am for now I am keeping it just hundred. So, basically, we have to repeat all these steps until all the nodes gets satisfied. So, this all will go under this loop once this is done, we are going to display the graph again. So, we will call this function display graph G.

Now, before I will execute let me it removes these least commons common based commons which where displaying the graph in the very first form. So, let me trend the let me display the initial graph is well display graphs. So, we are going to we are just displaying the initial version of the graph and after that we have this loop in which all the shifted will happen and in the end this is how the graph this is the command that will display the final graph. So, this should be displaying the initial graph they should be displaying the final graph and in the middle, they are these hundred iterations let me increase the number of iterations here. So, what should basically happen the nodes should move towards nodes of their own time?

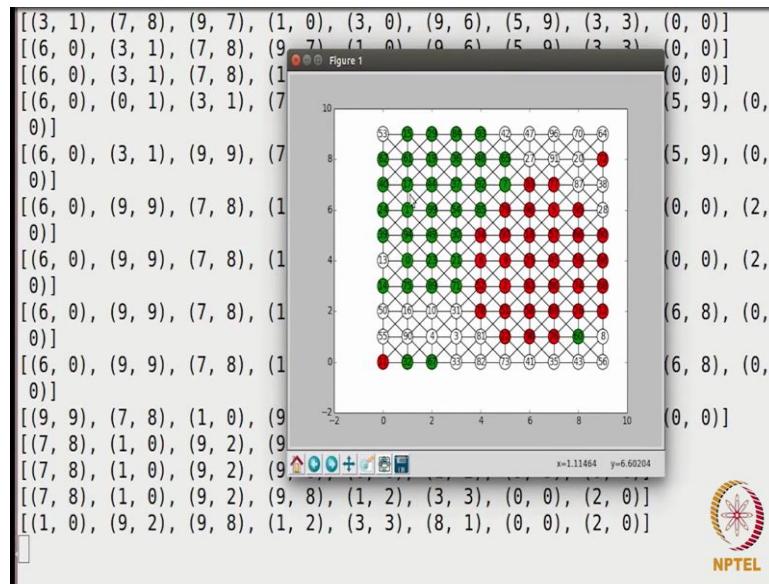
(Refer Slide Time: 08:31)



So, that is what is expected let us see how it works lets executed and see how it works.

So, this is the initial graph as you can see the nodes of different colours are scattered. I am closing it and now.

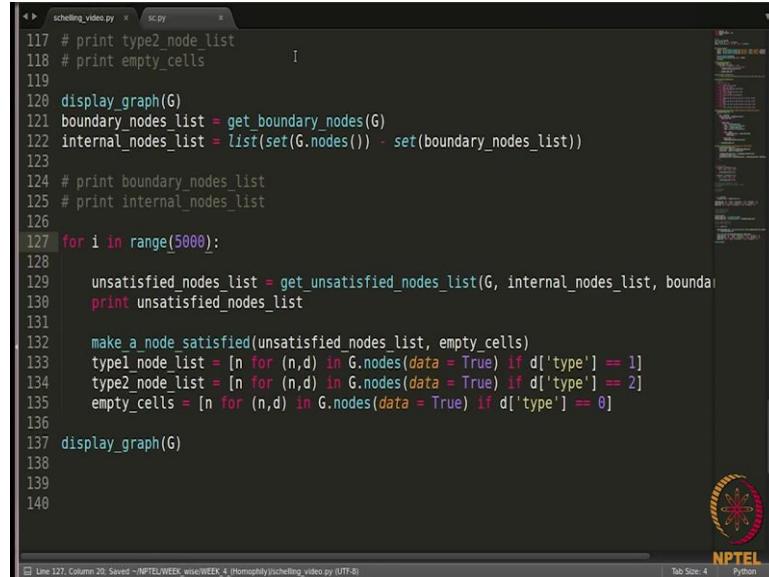
(Refer Slide Time: 08:42)



So, this is the final graph as you can see there are all the green nodes are on the one side all the red nodes are on the other side and there are few other nodes as well with scene unsatisfied is well maybe we can increase in the on the counter there. So, that counter will basically depend on many things it could depend on the size of the network that if

we have taken. Secondly, we have assigned the nodes randomly. So, for different random settings you might require different number of iterations you can also keep a track of the number of hydration that that were required to convert this graph into a graph where the people are same type on together.

(Refer Slide Time: 09:25)

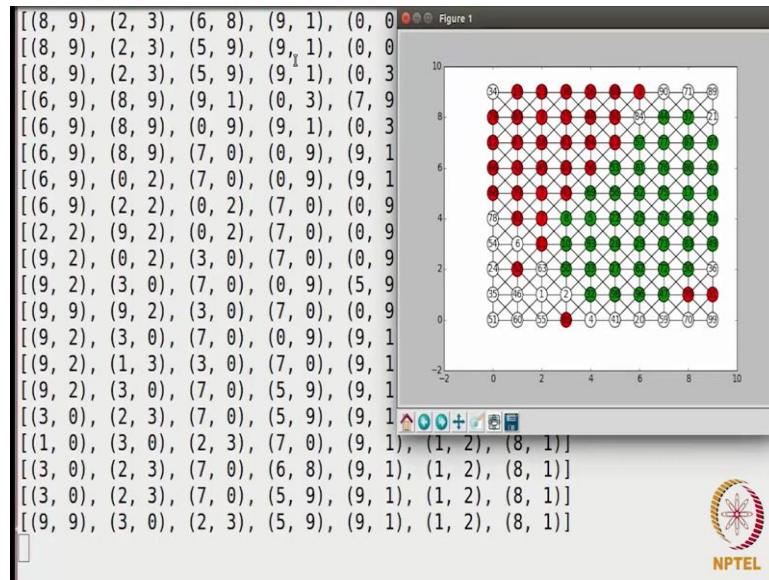


```

117 # print type2_node_list
118 # print empty_cells
119
120 display_graph(G)
121 boundary_nodes_list = get_boundary_nodes(G)
122 internal_nodes_list = list(set(G.nodes()) - set(boundary_nodes_list))
123
124 # print boundary_nodes_list
125 # print internal_nodes_list
126
127 for i in range(5000):
128
129     unsatisfied_nodes_list = get_unsatisfied_nodes_list(G, internal_nodes_list, boundary_nodes_list)
130     print unsatisfied_nodes_list
131
132     make_a_node_satisfied(unsatisfied_nodes_list, empty_cells)
133     type1_node_list = [n for (n,d) in G.nodes(data = True) if d['type'] == 1]
134     type2_node_list = [n for (n,d) in G.nodes(data = True) if d['type'] == 2]
135     empty_cells = [n for (n,d) in G.nodes(data = True) if d['type'] == 0]
136
137 display_graph(G)
138
139
140

```

(Refer Slide Time: 09:29)



10,000 seems too much; may be 5000; 5000 let us try for these many iterations. So, it is going its going good still there are a few unsatisfied nodes, but in more iterations they will converge they will converge basically it. So, happens that all the nodes become

satisfied another thing to note here is that you can also change the value of  $t$ . So, here we have taken  $t$  is equal to 3. So, even for an internal node since it has 8 neighbours even out of 8 neighbours only 3 are of its own type the node will not move to any other place. So, so you can change  $t$  to 4 5 or any other value and you can see the number of iterations that were required to get to the convergence. So, you can I think you have gotten enough tools now you can play around with all these values you can keep changing and you can observe and if needed you can plot things as well as in the number of iterations required for converging the these many number of nodes you can change the value of capital  $N$  which we initially to a capital  $N$  was 10 which means aware 100 cells.

So, that was about this Schelling model I would suggest you code yourself and you know watch the video only when it is needed see in the initial video I think I gave the structure as to what has to be done. So, if for every video you do that in the sense you just listen to the structure and the aim that has to be implemented and then code is code it yourself without just following the video that will be actual learning for you. So, I would suggest you to code yourself and check the video whenever needed only.

Thank you.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

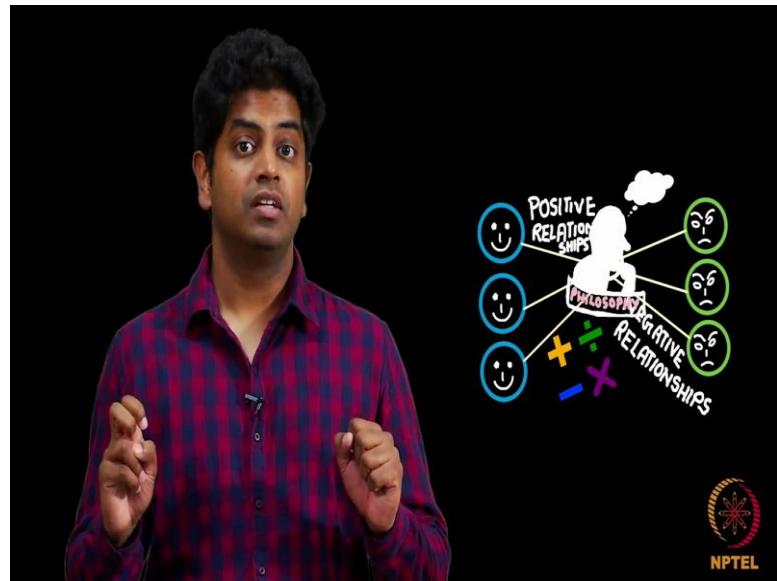
**Lecture – 62**  
**Homophily (Continued) & Positive and Negative Relationships**  
**Positive and Negative Relationships**  
**Introduction**

(Refer Slide Time: 00:15)



So, let us start with a new chapter, here is an interesting chapter for more than many reasons one being it is both philosophical as well as mathematically very elegant. We will sort of talk less of the data sets here and we will speak more of logic probably the only chapter where this kind of an approach we are going to take.

(Refer Slide Time: 00:37)



Let me motivate you with a nice question; in our life we make a lot of friends, some of them are good, some of them are not so good. Now in fact, we have I can classify them as positive and negative relationships with some of them we get along very well we are very close to them it feels very good being with them on the contrary we sometimes do not like being with some people this is how life is that is the philosophy part of it.

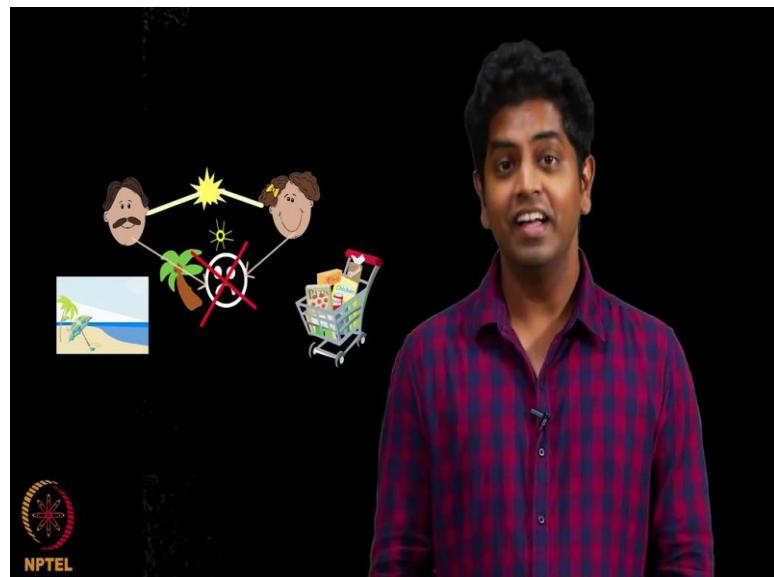
But what should entry give is the mathematics of this can be very well studied and the mathematics of it is both eyebrows rising as well as eye opening. So, let us see more of this. In fact, this is one chapter that is my personal favourite given the counter intuitive results that we all will see let us get started.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture – 63**  
**Homophily (Continued) & Positive and Negative Relationships**  
**Structural Balance**

So, let me motivate you with a very personal example and then let me talk about an abstract example a personal example.

(Refer Slide Time: 00:15)



I used to spend my vacations with my uncle and aunt a very loving couple very nice to hang around with them, but occasionally they used to sort of have some disagreement and they would fight for a minute or 2 and that is when I would be in a fix for a simple reason that uncle and aunt would start controlling me. Uncle would say come Sudarshan, let us go to the beach; aunt would say no, no, no, no, come let us go shopping, I get you something to eat uncle would say no, no, no, if you going with your aunt and I am not going talk to you; aunt would say the same; if you are going with your uncle I am not going talk to you.

So, I am in a fix here, look I cannot go with only one person here, there will be hatred from the other person; what do I do I cannot go with both of them, I cannot go with one of them. The best thing here is to say call it quits I am not going with anyone, I am

neither going with uncle nor with aunt this is to be on the safer side right. We all of us would have encountered such a situation in our life you love 2 people equally and they ask you to take a call love me or love the other person you cannot love both. So, it is so happened that I would take the excuse of I have homework and I would say start scribbling something and things would come to a standstill and then they would become normal.

This is my personal example let me try using abstract example to explain the same thing and I just use personal example. So, it people can relate to it and abstract example.

(Refer Slide Time: 01:57)



Let me consider this friend Ram and Krishna, Rama and Krishna; these 2 people are very close to me let us say very good friends best friends, but unfortunately, they do not like each other. Here is the similar problem I told you of the uncle and aunt example I cannot invite Rama and Krishna home for a party for a simple reason that in case they come for the party they would not talk to each other.

Party is done to distress yourself if you have 2 such people in our party it will add to your stress. So, what will happen? Think about it for a second Rama and Krishna come home they sit together they do not talk to each other; they do not want me to talk to one-person other person gets offended what will I do. There are 2 consequences of such a thing, in my life if I want to continue being friends with these 2 people I must take a decision one by hook or crook make them become friends that is a nice decision if it can

work. But generally does not work 2 people who hate each other now we would want to continue hating each other what do I do? I will try my best to make Rama and Krishna become friends with each other I will convince them take them out with them I will play football with them party with them no, it may not work if it works well and good if it does not I am in a fix.

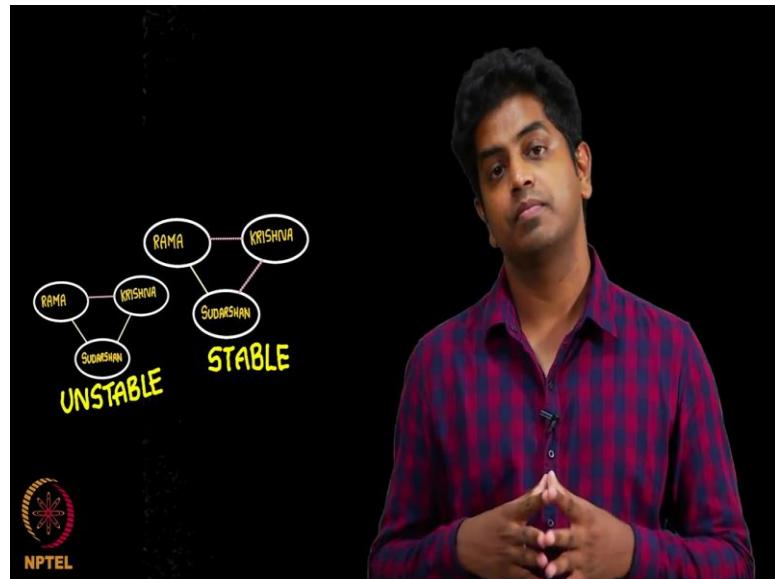
Why I am in a fix it is because Rama will say Sudarshan enough is enough stop being friends with Krishna otherwise I will stop being friends with you Krishna will tell me the same thing. So, what is happening here think for a minute let me draw a picture and then observe what is happening here?

(Refer Slide Time: 03:53)



Here is me and I have I am friends with Rama I am friends with Krishna Rama and Krishna hate each other now this kind of a triangle with 2 lines and one dotted line dotted line representing hatred and straight line representing friendship is a dangerous combination. This will not continue to be stay this way for a long time eventually either this dotted line should become a straight line you say what I am saying either Rama and Krishna should become friends because of Sudarshan's involvement or whatever or Sudarshan should start hating one of them.

(Refer Slide Time: 04:38)



In case Sudarshan starts hating Krishna the triangle will look like this Sudarshan likes Rama, Rama likes Sudarshan of course, and Rama of course, hates Krishna as and always Sudarshan has started hating Krishna this is a stable situation. The previous one was actually an unstable situation unstable simply because there is pressure for the structure friendship structure to change now this was with 3 people imagine there are thirty people in a organization is a lot of hatred lot of friendship what will happen.

(Refer Slide Time: 05:10)



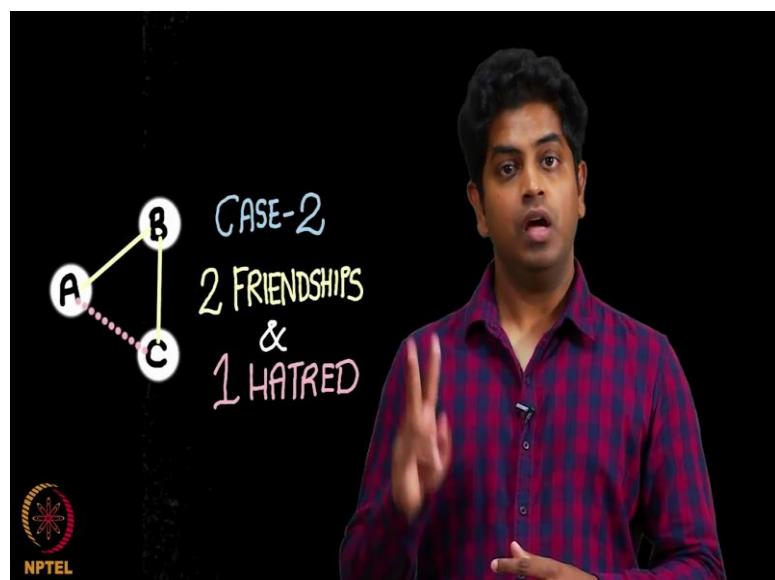
This entire chapter is about studying this and studying this and inferring some really cool results lets gets started with more results right now I will now make an attempt to enumerate all possible cases what do I mean by all possible cases.

(Refer Slide Time: 05:35)



You see there are 3 people A B and C between A B and C; what all are possible 3 friendships are possible as you can see case 1.

(Refer Slide Time: 05:46)



(Refer Slide Time: 05:55)



Case 2; 2 friendships and 1 hatred, this is case 2 what is the next case as you can guess 2 hatred one friendship.

(Refer Slide Time: 05:59)



The third the fourth case is all 3 hatreds, right, let us take this one at a time and then observe what are the consequences of these cases what are the cases that are stable what are the cases that are unstable. I hope you are following what I mean by stable and unstable by stable I mean this kind of a friendship does not have any dangers.

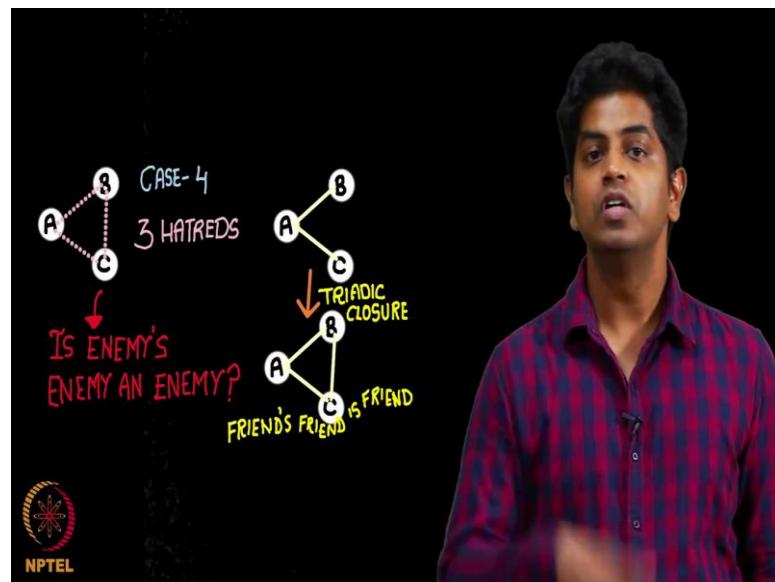
(Refer Slide Time: 06:29)



For example 3 friendships I am there is no problem there at all versus the case that I told you awhile before Rama Krishna case what you can see here 2 friendships one hatred. Now that is dangerous it is unstable eventually something we will happen to this kind of a triangle right either it should become a complete 3 friendships or 2 hatred one friendship as discussed earlier what I will do right now is I will go slowly with all these cases and see what case has what consequence. So, first case 3 friendships a happy with need I same over 3 people who are friends with each other I mean there you have a wonderful life.

There is absolutely no hatred between anyone period there is just nothing to study here at least in this chapter 3 friendships is stable there is no pressure it similar to my Rama Krishna that is a tough life here is a happy life.

(Refer Slide Time: 07:38)



Now, what if there is 3 hatreds A B and C, there 3 people hate each other by the way let me ask you a question remember triadic closure friends friend is a friend do you think and enemies enemy is an enemy think about it a friends friend is a friend the same rule applies here it is. In fact, true that an enemies enemy is actually an enemy think about it I am right is I just because of professor says something then he will be right pause for a second analyze the situation and tell me is it really true that enemies enemy is an enemy if you are not clear just watch this video clip and come back and I am sure you will understand what is wrong with this statement enemies enemy is an enemy.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture – 64**  
**Homophily (Continued) & Positive and Negative Relationships**  
**Enemy's Enemy is a Friend**

Hey, are you Anamika, I have been teamed up with to do the artificial intelligence assignment given by Professor Sam.?

Yeah, I am Anamika, but I have to go home this weekend, can we start on Monday?

Oh, no, this requires a lot of hard work and you going home will only delay the submission.

I am sorry, but I have to go home, I can start from Monday only.

Fine, but you do not simply understand that sometimes we have to be adjusting.

You do not tell me what to do I know who I am and what I have to do.

Well, anyway I will start with the assignment I have no other option other than working with you, the only other girl left is Deepika and I do not like her at all, it is better to give a late submission than working with her.

Are you talking about that short, thin Deepika having long hair (Refer Time: 01:02) also?

Yes exactly, do you know her?

Yeah, I have been an dance class with her, she is a great show off and moreover she has ruined my rehearsal many a times.

Oh, I am sorry for you, yeah, she is a great show off she thinks she is some great person.

Yes, actually I have to show her what I am, wait, I will come, I will cancel my plan of going to home and work with you in on this weekend, we will show great show.

Come let us go for a coffee.

Yeah, we are friends up here.

(Refer Slide Time: 01:37)



So, we watch the video what is your inference enemy's enemy is a friend comes as a counter intuition in the beginning, but the video made it very clear this 2 people initially they were they were not liking each other, but they got it topic to discuss and the topic is the third person who is not there and the third person is an enemy to both of them over we have seen many such situations in our life where 2 people only thing that they need is an excuse to talk about a third person who is not there and there is extreme bonding between these 2 people. We have seen this in many occasions we have done this too, right yeah. So, this is prevalent to an extent that is a lot more than what you can imagine this always happens.

(Refer Slide Time: 02:37)



I am going to motivate you by giving you very nice example as to why this happens lot of studies which says the following. When 2 people struggle together some kind of a bonding that happens between them I do not know whether it happens everywhere, but it is to happen in the place where I studied we used to have what is called the freedom run every august 15th, august 15th of every year we wish to have what is called the freedom run. We would run until this sweat and take a big ground and then come back. I was always wondering why do we run and why is it called a freedom run the point is very simple when you sweat together sweat together struggled together when you run together there is some kind of a bonding that happens with each other.

In fact, try this out if you want to increase your friendship with someone try working out with that person, try taking a long jog with that person you will see a difference in bonding rather the bonding will increase. So, maybe I should try that with my Rama and Krishna friends, one way to make them become friends is to run with them if we sweat together we become friends it seems say is research. Anyway I am going to give you some personal a tip here the course is not all about abstraction and exams and assignments I am going to give a life tip right now, I hope NPTEL does not edit this piece of video.

(Refer Slide Time: 04:17)



Here is a very nice tip especially for single people it is believed that if you work out sweat it out with a person of the opposite gender people can even fall in love what do I mean by this if you are seeing someone if your relationship status a sort of complicated. So, try taking a long jog with them or try playing badminton or workout go to a gym and try some aerobic; so, whatever.

In case you sweat together there is a high possibility that you people will fall in love with each other I am not saying this research says this try Googling for this information falling in love because of sweating together or something like that there is psychology to the article which has description of all the research in this direction anyway. So, let me know in case there is any success with this step alright.

(Refer Slide Time: 05:16)



Getting back to my original discussion there are 2 people struggling there is extreme bonding between them and 2 people beat friends or enemies just in case there talking to each other about their struggle imagine there is a bully in the office someone who troubles these 2 people and these 2 people actually do not like each other very much. But there is a common thing for them to talk which is this office person who bullies this people let us say the boss is always the bully.

So, boss bully is this 2 people and this 2 people they start talking about the boss and they get very close now do you do you see what is happening here, do you see the reasoning behind why an enemies enemy is a friend this is the reason. So, what is the inference?

(Refer Slide Time: 06:12)



The inference is the following and enemies enemy is a friend take one message hatred, hatred, hatred 3 hatreds in a triangle is not stable when 2 people come together for some odd reasons and may they get to know that there is a third person who is a common enemy that is enough for these 2 people to become friends say psychology research. So, now, we are going to put this particular case of 3 hatreds under the unstable umbrella we are going to say 3 friendships means stable happy world as I told you 3 enemies means it is not stable there will be a friendship that will originate very soon. So, let us paraphrase what is happen?

(Refer Slide Time: 07:08)

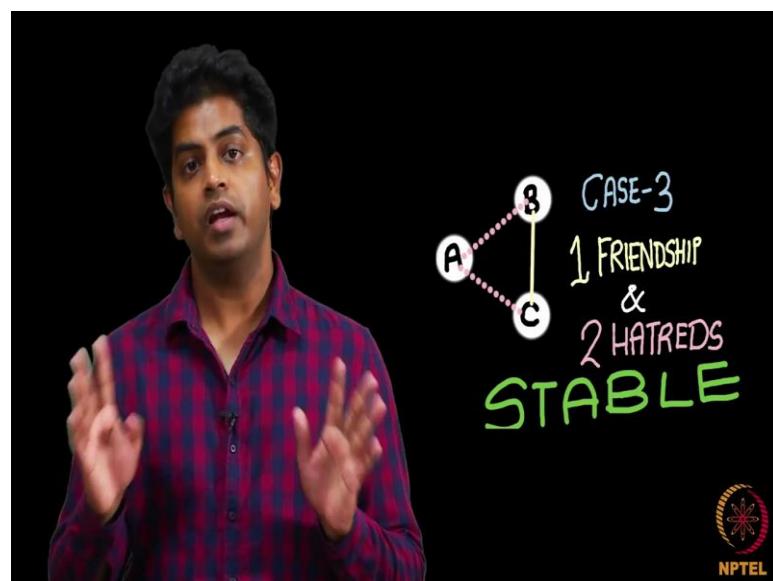


(Refer Slide Time: 07:14)



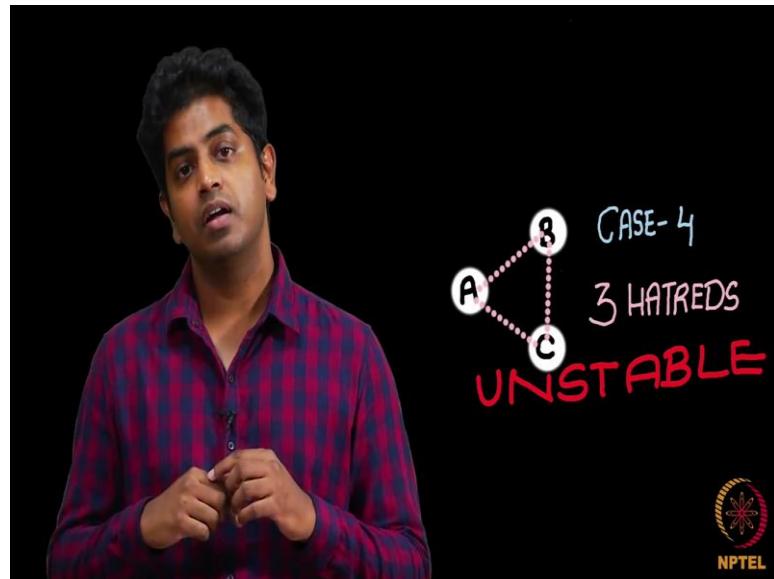
We had four cases just quickly go through this four cases case one 3 friendships stable no problem case 2-2 friendships one hatred Rama Krishna example unstable this will result in at least one friendship breaking this is an unstable state case.

(Refer Slide Time: 07:28)



3 2 hatred 1 friends stable nothing goes wrong here it will continue to be like this. In fact, the same hold theory that I told you just now holds good here you have a common enemy and that will increase your bonding when you talk about this common enemy with each other what is called Crebbing; Crebbing complaining whatever you call it.

(Refer Slide Time: 07:54)



Last case is 3 hatreds we discussed that us again that brings in some sort of an imbalance by imbalance I mean a it is not stable where in 2 people get together out of this 3 enemies mutual enemies when 2 people get together tells start talking about the third person and the bonding increases. So, 3 enemies; 3 hatreds unstable, so, you see these four cases 2 of them are stable 2 of them are unstable.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

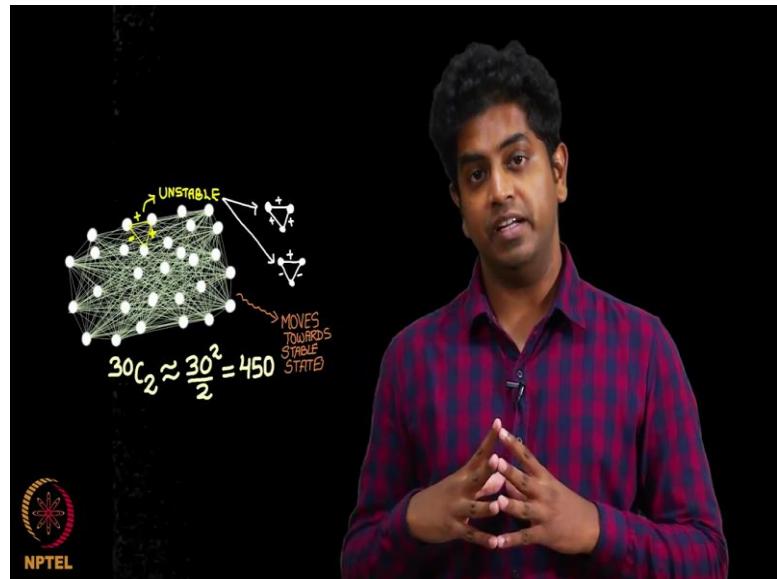
**Lecture – 65**  
**Homophily (Continued) & Positive and Negative Relationships**  
**Characterizing the structure of balanced networks**

(Refer Slide Time: 00:16)



Let us now shift gears and look at something slightly deeper you are now going to observe how in an organisation such positive and negative relationships can actually affect the organisation. Let me now ask the big question the main question in this chapter. So, he has the puzzles that impose the big question as a puzzle.

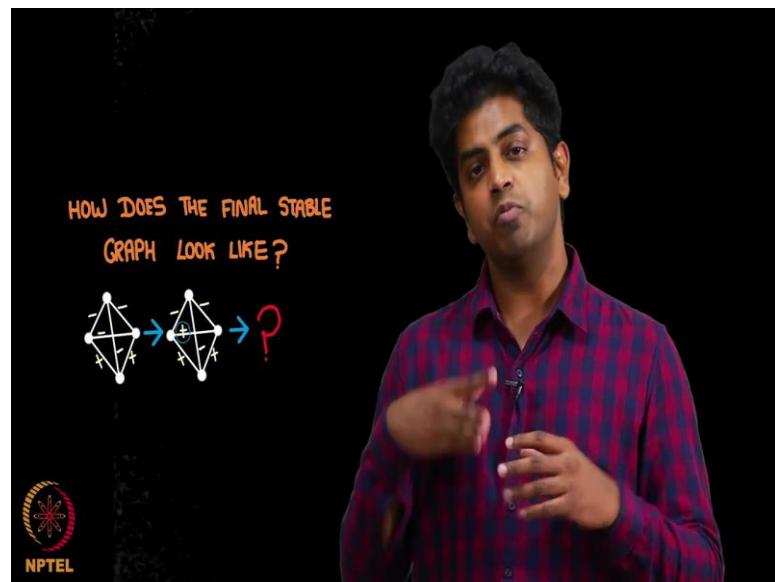
(Refer Slide Time: 00:34)



There are 30 people in an organisation 30 out of these 30 people since its only 30 people they know each other. So, is what is called a complete graph 30 people who know each other perfect? So, how many possible friendships are their here we have discussed this before it is roughly  $30^2/2$  to be precise it is  $(30 * 29)/2$ , but throughout the discussion in this course a graph on n number of vertices we assume has roughly  $n^2/2$  number of edges.

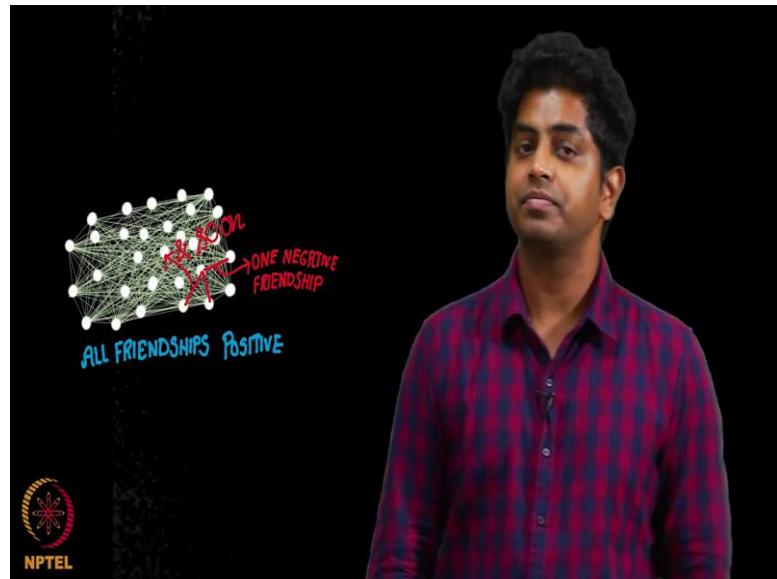
So, with 30 people you have  $30^2/2$  number of friendships which is 450, right. So, far so good out of these four fifty friendships that you are seeing some of them can be positive some of them can be negative. So, observe carefully here is the graph with 30 nodes 30 people and roughly 450 friendships which can be positive and negative. We also know the four cases if there is a triangle here with a positive, positive, negative the it is unstable it will move towards positive, positive, positive or positive, negative, negative we have discuss this enough or you should remember by looking at this figure is that given a complete graph with 30 nodes and some random plus minus symbols on these edges in an organisation it moves towards stable state because every triangle here if you see may not always be stable.

(Refer Slide Time: 02:48)



So, what is the big question the big question is as it goes towards the stable state in an organisation how will the graph look like when there are no unstable triangles here I repeat out of 30 people given 450 friendships with some random plus minus symbols on the edges which is friendship and hatred what is what does one mean by this particular graph becoming stable, alright. So, in (Refer Time: 03:18) heard to analyze that that see one first of all one should get the question right once we get the question right we should make an attempt to answer to do you see the motivation behind this question the motivation is if a bunch of people get together there could be there could be friendships or hatred between them if there are unstable triangles it moves towards stable state what exactly happens eventually.

(Refer Slide Time: 03:48)



Let us consider a happy about assume 30 people how 450 friendships and all this friendships are positive never happens, but let us assume this happens all 30 people are friends with each other and their all positive now remember the code that I started this chapter with what is the code a bunch of apples you have and you just put one rotten apple there one rotten apple spoils the barrel right. So, this is just like this you have milk you put some if you drops of curd into it and the entire milk eventually turns into crud right given this kind of a happy world with all relationships being positive just try introducing one negative friendship what will that result in observe.

One negative friendship will result in a triode around it at least one triode right one triangle one negative friendship and that results in Rama Krishna kind of a phenomena and this one negative friendship ensures that one of their friendships becomes negative and this cascades whatever mean by this cascades one negative friendship results in another negative friendships so on and so forth. But wait a minute we will this result in all of them becoming negative my curd milk example is not actually true here why introducing one negative friendship we will result in negative friendships, but then if there is a triangle with three negative friendships that will result in positive friendship. So, it is not really true that introducing one negative friendship results in all negative friendships it may not really be true.

Now what else us what else will happened here I repeat the question given 30 people I am using 30 for example, reasons you can it an 100 200 whatever let us stick to 30. Given 30 people 450 friendships positive or negative there are some unstable triangles here it will reach stability slowly how does a organization with all stable triangles look like is the big question that we are going to answer now.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture – 66**  
**Homophily (Continued) & Positive and Negative Relationships**  
**Balance Theorem**

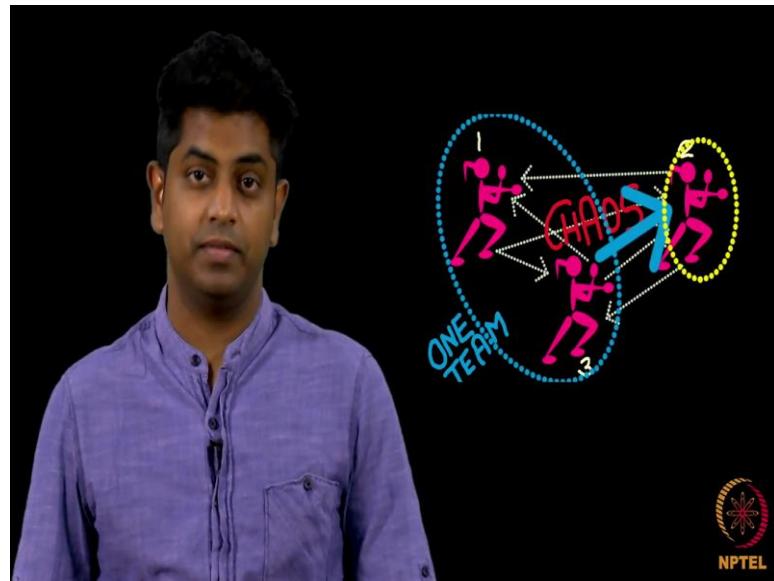
So, we are now going to show a very real world observed phenomena. In fact, I will tell you about it and it will be surprisingly true you will even say why do we have to talk more on this isn't it obvious, but then this is very elegant mathematical proof for it what is the question imagine this is a boxing ring.

(Refer Slide Time: 00:34)



And there are 2 people in the boxing ring and what you do is you introduce a third person when you introduce a third person what happens 2 people are trying to fight. This is a third person who is trying to fight and the third person does not join the first 2 people he is trying to fight the 2 people every person here is trying to fight the other 2 people we will this work what will happen eventually.

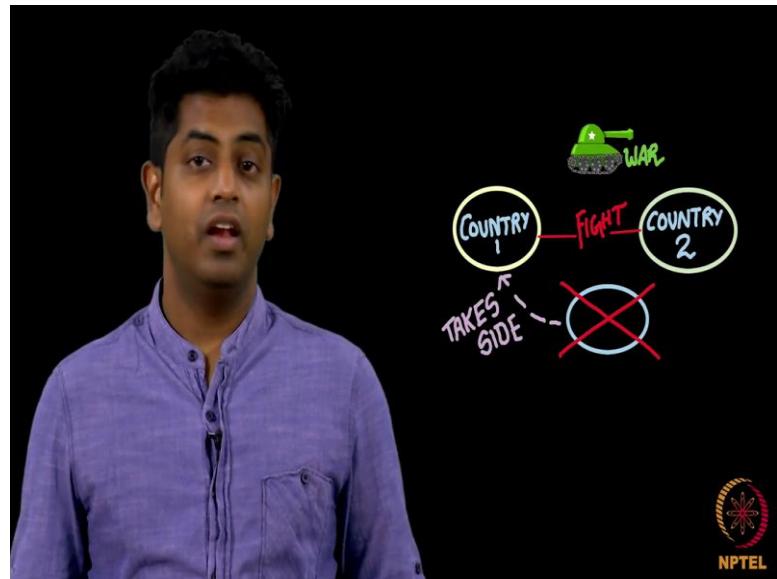
(Refer Slide Time: 01:09)



The first person is trying to hit the third person and second person second person is trying to hit the first person and the third person the third person is trying to hit the second person and the first person.

It will be a chaos it cannot be a proper fight unless 2 people get together into one team in that case these 2 people we will fight with this one person and they will be a when or a loss, but then when you have these 1 1 and 1 the boxing ring gets very chaotic does it sound very familiarity have not you seen in a group the fight is generally between 2 teams.

(Refer Slide Time: 01:52)



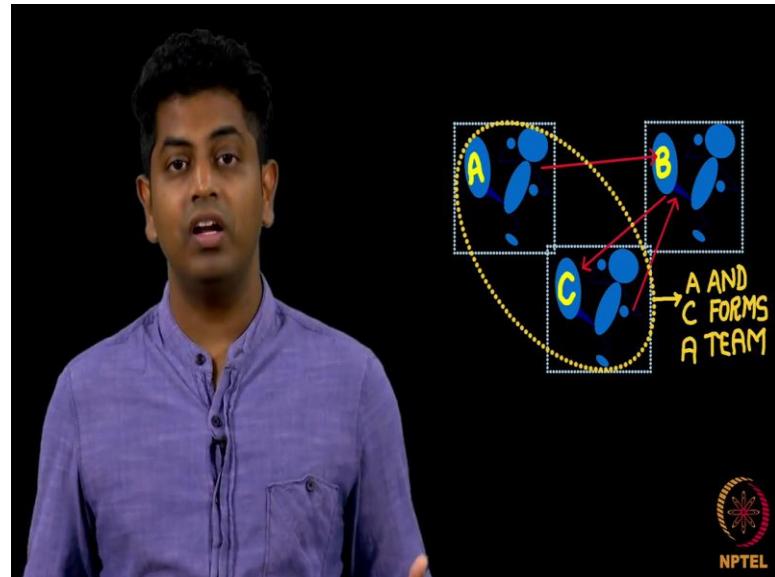
It is never between 3 teams why is that this sounds all the more familiar to you war when its waged it is never between 3 countries it is always between 2 countries may be is the third country, but the third country we will take some one side. In fact, it is observed in world war one and 2 that countries did take the side of some other country and the war was generally based between 2 groups in a bunch of people the war is always between 2 groups not between 3 groups.

(Refer Slide Time: 02:42)



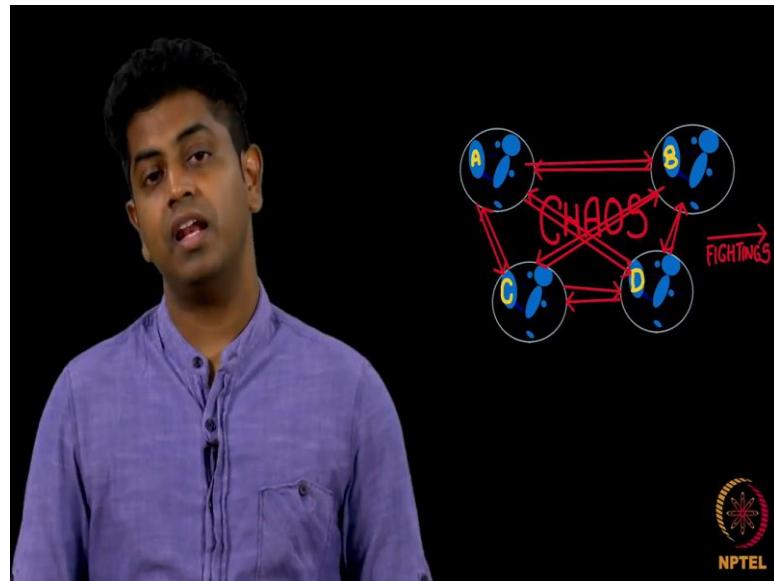
Now, can you connect the dots and tell me why exactly this happens that is because enemies enemy is a friend correct as simple is that how.

(Refer Slide Time: 02:50)



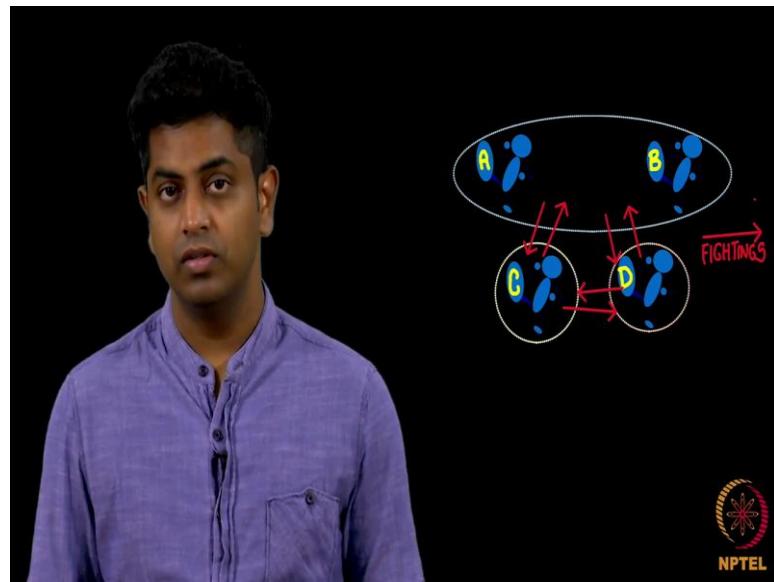
Assume you have a boxing ring and there is A B C fighting with each other when A realises that he needs to hit B and B realises that he should hit C and C should hit B. A and C also should hit C there A C can get together and hit B, right so on and so forth. This is the same kind of a chaos here and then it can only be resolved by 2 people taking one side. So, when these people take a start getting taking in sides that enemies enemy becoming friends sort of a thing.

(Refer Slide Time: 03:29)



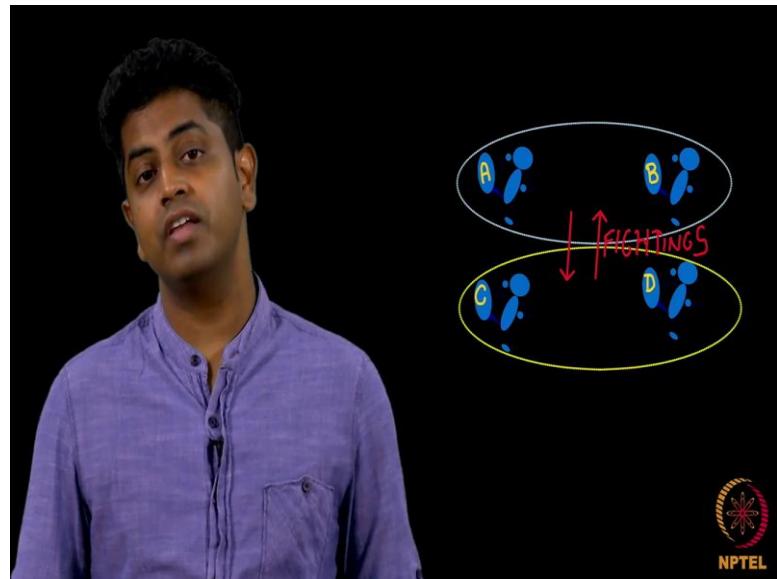
Then there becomes 2 teams imagine there are 4 people who are trying to box in a boxing ring now what happens again the same thing there are there is 1 2 A B C and D trying to box with each other and then A hits B, A hits C, A hits D and every single person is trying to hit the other 3 people chaos.

(Refer Slide Time: 03:53)



So, some prefer order emerges there and then this split into few teams maybe 3 teams from 4 teams A B C D is 4 teams and they realise this cannot happen this cannot continue and this aggregate themselves into 3 teams still there is chaos.

(Refer Slide Time: 04:07)

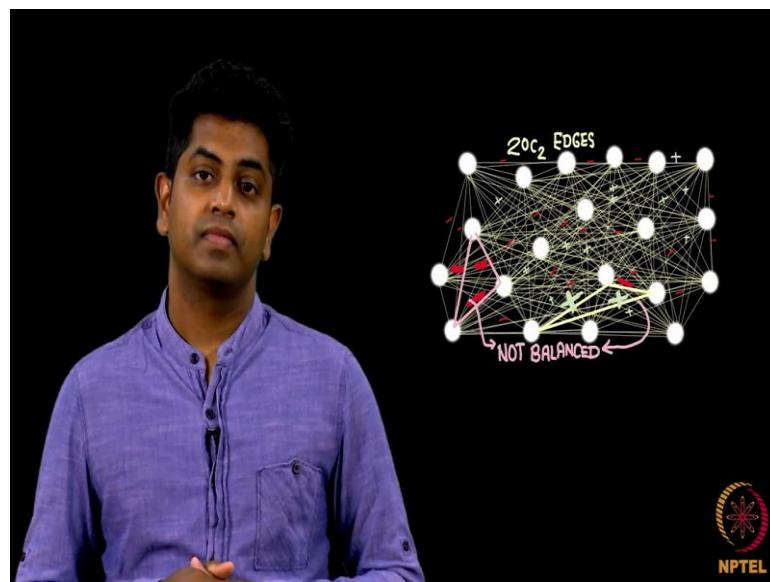


As I told you before these 3 teams again gets converted to 2 teams no matter how many people you put it is always 2 teams, why is this true let us try to look at it mathematically right now.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Lecture – 67**  
**Homophily (Continued) & Positive and Negative Relationships**  
**Proof of Balance Theorem**

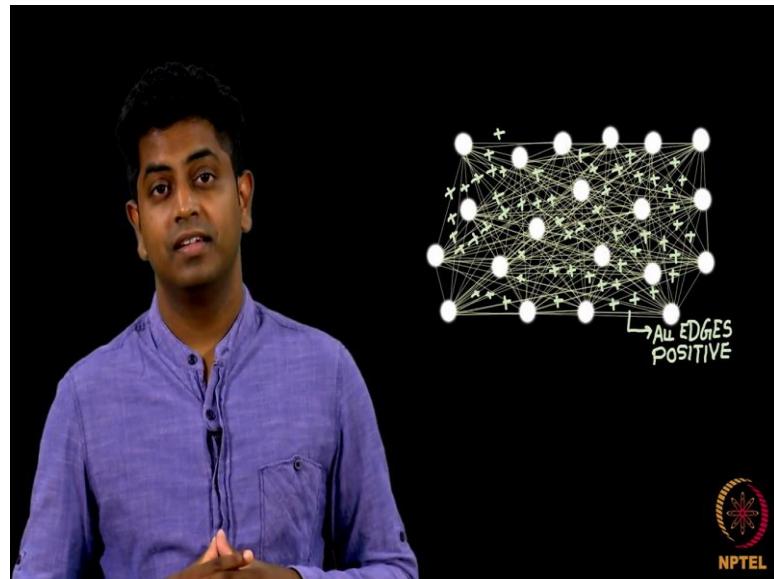
(Refer Slide Time: 00:21)



So, let us try to mathematically rigorously try to understand what is happening here. What I will do is firstly, I will develop the necessary machineries we will assume there is a complete graph, the graph on  $n$  nodes let us say some 20 nodes and you put edges between any 2 people,  ${}^{20}C_2$  edges. Now there are some plus and there are some minus relationships, I put that 2 is this structurally balanced maybe not, you see there is a minus, minus, minus triangle there is also a plus, plus, plus minus triangle this is not structurally balanced. But what if it was structurally balanced let us see.

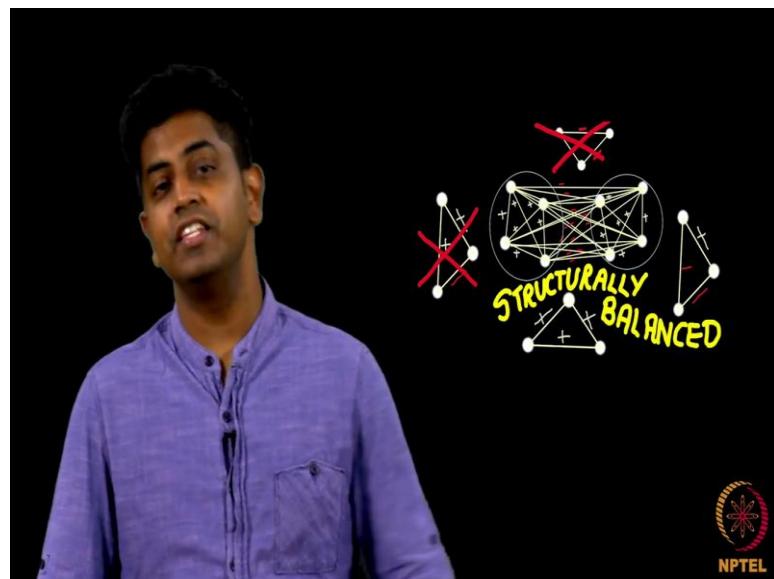
Let me now think of an example where is where a given graph is structurally balanced how we will look like this is an example of our graph that is structurally balanced see how it looks like.

(Refer Slide Time: 01:20)



Obviously, all of them are plus I know this is structurally balanced what is so great about this. Is there any graph with negative edges as well that is structurally balanced?

(Refer Slide Time: 01:44)

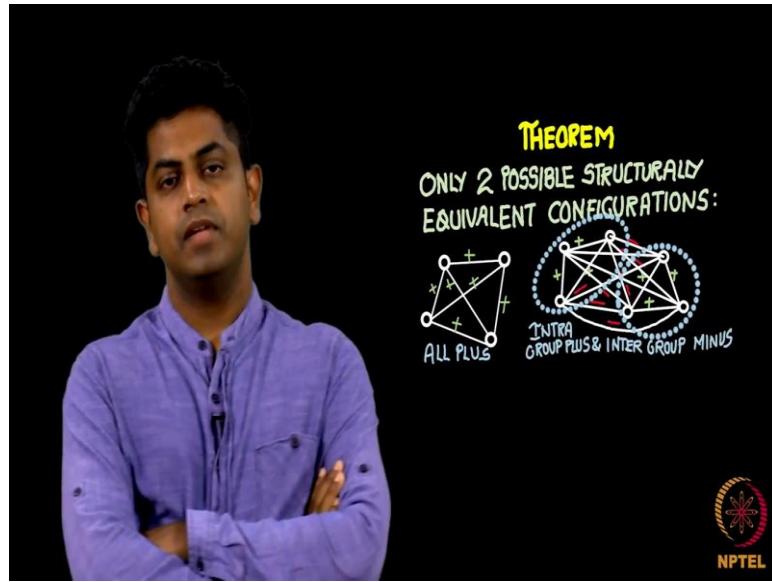


Let me think of finding one as I keep drawing. I converged to only structure like this do you see there are 2 clusters and edges inside these clusters are positive and edges across are negative. Basically there are 2 teams here 2 teams means what within the team the relationships is positive right you will see that in the figure relationships between 2 people inside the team is positive relationships between 2 people across the team is

negative. This is structurally balanced why, let me take a triangle here a triangle here is either positive, positive, positive as you observe or positive, negative, negative correct can you find a triangle here which has a positive, positive, negative plain impossible just observe can you give me a triangle here which is negative, negative, negative no impossible which means looks like this is structurally balanced.

So, when you have one team full of positivity structurally balanced when you have 2 teams where positivity is within negativity is across structurally balanced some basic observation that you did this structurally balanced. Now if I give you a graph  $g$  and college structurally balanced how will it look like we will it look like all positives or only 2 teams with positives inside negatives across is there are third type. So, we prove that there is no third type there is there only these 2 types the type one where there is only one team the type 2 where there are 2 teams.

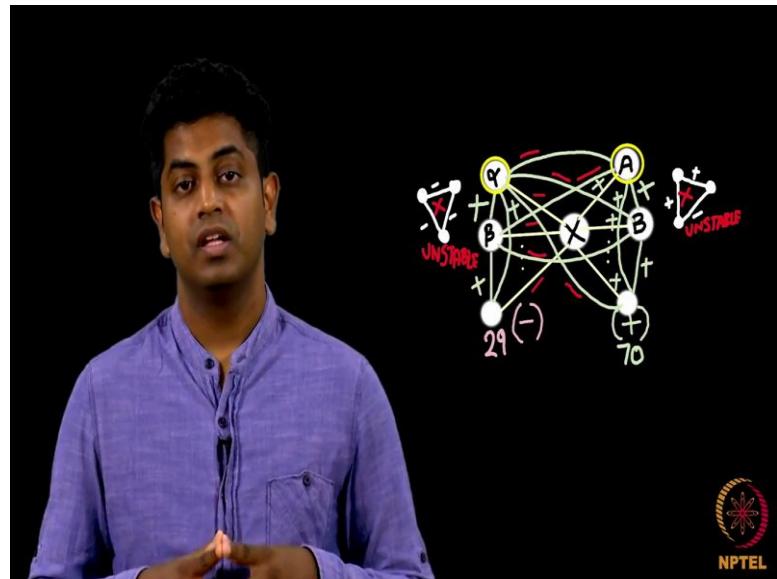
(Refer Slide Time: 03:53)



And period nothing else can happen. So, here is a theorem which states these are the only 2 possible structures when you say a given network is structurally balanced.

Let we let us try proving this now with good amount of rigger how do you go about it.

(Refer Slide Time: 04:17)



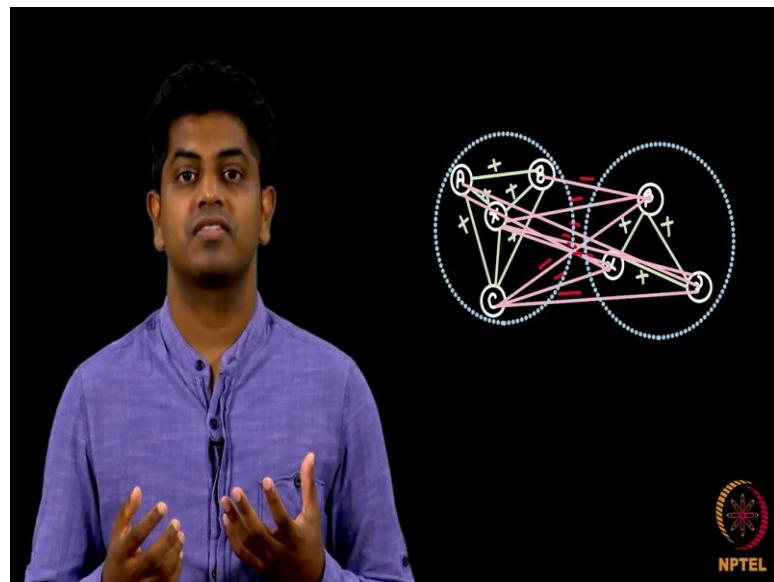
Let me take a given graph  $g$  with many nodes let us say 100 nodes and I pick one node from this network as you know in these graph with 100 nodes there are edges across any 2 nodes and there is a sign for every single edge plus or minus. As you can observe let me pick some node here call it let say  $x$  node  $x$ .  $x$  is just sent to ninety-nine other nodes some of them positive some of them negative.

Let me say sorry roughly seventy of them is positive and twenty nine of them are negative seventy are positive twenty nine of them are negative now is you need obvious that this vertex  $x$  is friends with positive edges and the encompassing triangle on these edges as you can see  $x$   $A$   $B$  this triangle  $A$   $B$  cannot be negative because I told you it is structurally balanced correct. Now look at this side  $x$   $\alpha$ ,  $\beta$   $x$  to  $\alpha$  negative  $x$  to  $\beta$  negative now what can be the edge between  $\alpha$  and  $\beta$  it has to be positive because its structurally balanced correct think about it. This vertex  $x$  if it is adjacent to 2 people positively the relationship between these 2 people as I told you in the example  $A$  and  $B$  should be positive with the vertex  $x$  is adjacent to  $\alpha$  and  $\beta$  and  $x$   $\alpha$  is negative  $x$   $\beta$  is negative and what should be between  $\alpha$  and  $\beta$  enemy is enemy right it should be positive.

So, I observe that when you pick an element  $x$  and look at all its friends which are positive and all its friends which are negative you observe that amongst the positive friends of  $x$  relationships are all positive amongst the negative friends of  $x$

relationships are all positive, but if you pick a friend A of x and a friend  $\alpha$  of x.  $x A$  is positive,  $x \alpha$  is negative then what is the relationship between A and  $\alpha$ ? 1 positive one negative which means A and  $\alpha$  should be negative that is straight forward. So, what do I gather all the friends that x is adjacent with they are friends with each other with positive friendship all the people that x is adjacent to with negative friendship negative relationship?

(Refer Slide Time: 07:22)



They are all friends with each other which means there are 2 clusters here what are those 2 clusters the clusters x and all his friends there all positive within and all the negative friends of x there all positive within, but negative across as simple as that that closes the theorem. So, the theorem just states that in case you have a structurally balanced network it better we type one all positive friendships within there is only one team type 2 there are precisely 2 teams where friendships within them is positive friendships occurs is negative.

**Lecture – 68**  
**Homophily (Continued) & Positive and Negative Relationships**  
**Introduction to Positive and Negative Edges**

(Refer Slide Time: 00:05)

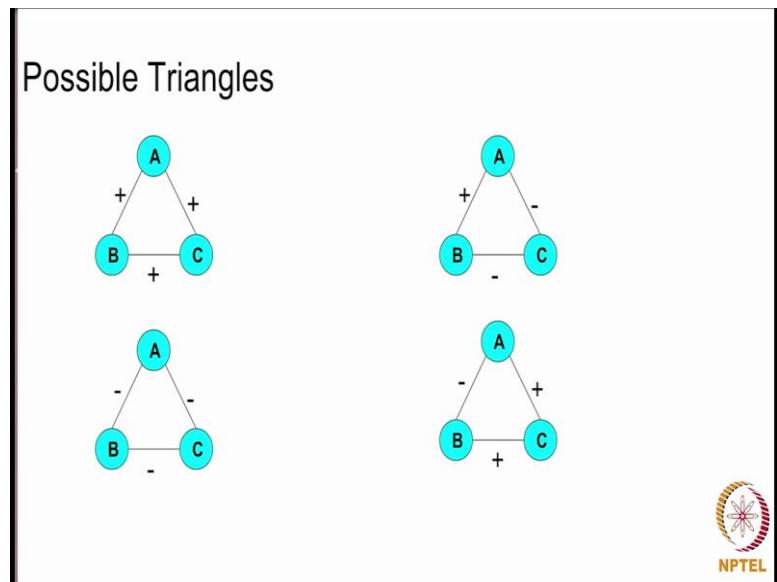
# Signed Networks



Hi everyone, in this video sequence we are going to implement signed networks where the edges have either positive or negative weights attached to them. And if the nodes represent individuals then the positive way the positive edges indicate friendships amongst these individuals and the negative edges indicate animosity amongst these individuals. In such networks there exist some sort of triangular structures which when analysed lead to interesting observations, in this video sequence we are going to analyse those structures which are basically different combinations of positive and negative weights across the edges of the triangles present in the network.

So, I assume you have been already introduced to these networks in the previous videos, I will just be briefly discussing the possible structures that are prevalent in the networks and then we will go ahead to implementation.

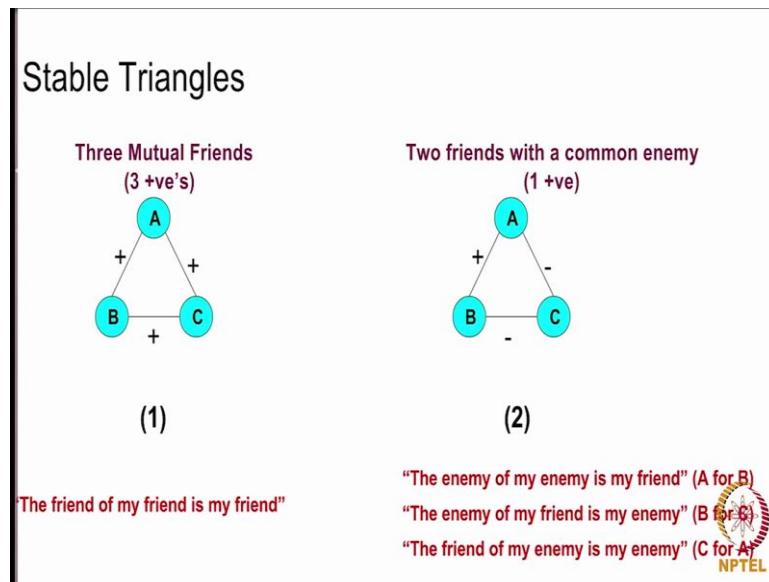
(Refer Slide Time: 01:08)



These are the possible combinations of positive and a negative way across the triangles present in the networks. Now some of these triangles are called stable and some of these are called unstable. Now what do we mean by unstable triangles or unstable combinations? These are the combinations that do not stay in the network for a long time simply because they are inconsistent with respect to the relationships among amongst the nodes and they tend to move towards a stable state.

Now, we are going to see which of these are stable combinations or stable triangles and which are unstable triangles. Please note that they could be different permutations of these signs across the triangles you anything to note here is that there are three positive signs here and there is one positive sign here there are 2 positive signs here and there is no positive sign here.

(Refer Slide Time: 02:18)

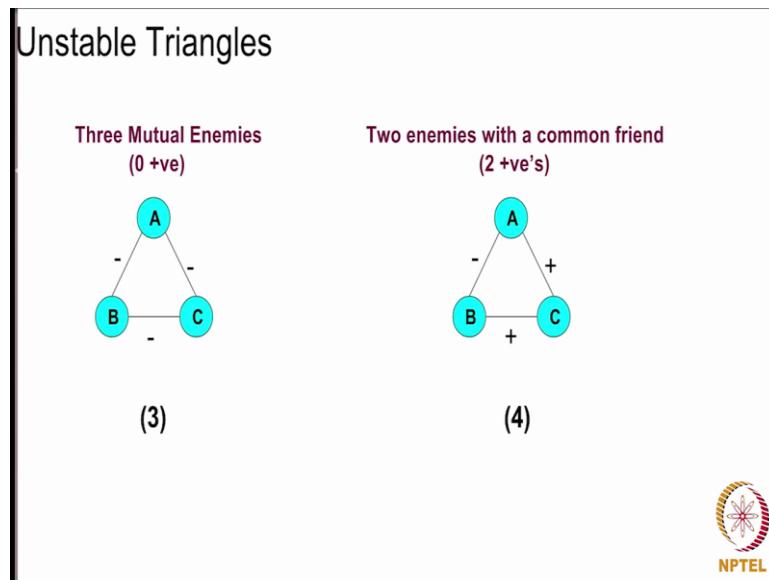


So, this is these are the different combinations of course, the different permutations can exist. So, let us look at these stable triangles first. So, a triangle having all positive edges is one of the stable triangles. So, there are three mutual friends they cannot be a better situation done this, so, every bodies every body's friend.

This kind of structures seems to follow this popular social belief that the friend of my friend is my friend the next social sorry next stable triangle is this one where you have one positive edge and 2 negative edges. So, this basically shows that there are 2 friends who have who have one common enemy, right. So, the triangle will have one positive edge, but it indicates that 2 friends have one common enemy this is also one of this stable triangles this triangle seems to follow a number of interesting social believes. For example, if you look at node A with respect to be the node a seems to follow the enemy of my enemy is my friend when it comes to node B with respect to C its seems to follow this belief that the enemy of my friend is my enemy as you can see here and if when it comes to node C with respect to A it seems to follow this belief that the friend of my enemy is enemy.

So, it is pretty interesting to see that all the nodes seem to be satisfied with the kinds of relationships they have they have with their neighbours in the triangle. So, that is how it another stable triangle. So, you can remember that triangle which has either one positive edge or three positive edges are stable triangles let us now check the unstable triangles.

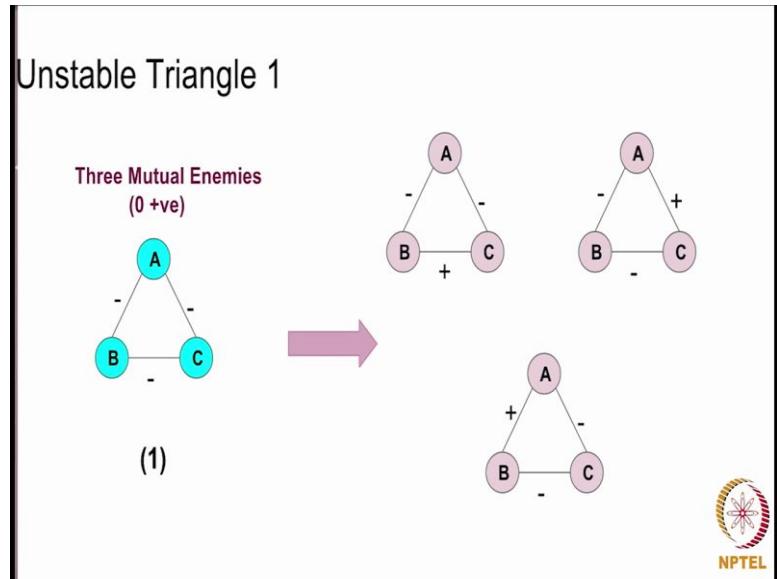
(Refer Slide Time: 04:12)



So, the first one as you can see is the triangle which has all negative edges. So, this is one of the unstable triangles because all the nodes are enemies of each other and this; this kind of situation cannot stay for a long time given the belief that enemy of my enemy is my friend. So, this seems to actually play a role here which leads to some of the nodes befriending other nodes and hence changing the configuration and hence moving towards stable state.

So, this is the one where there is 0 positive edge let us look at the second unstable triangle now here as you can see there are 2 positive edges and there is one negative edge. So, this basically means that there are 2 enemies which is A and B here there are 2 enemies and they have one common friend. So, this is again another unstable situation which does not tend to stay for a long time because 2 enemies would not like to have one common friend. So, this again leads to another state which is stable hence changing changes the configuration of the network. So, you can remember that the; and that a triangle which has either 0 positive edge or 2 positive edges is an unstable triangle right. So, you can remember it that way now what we are going to do is we are going to look at each of these unstable triangles and we are going to see which stable states can these triangles move towards.

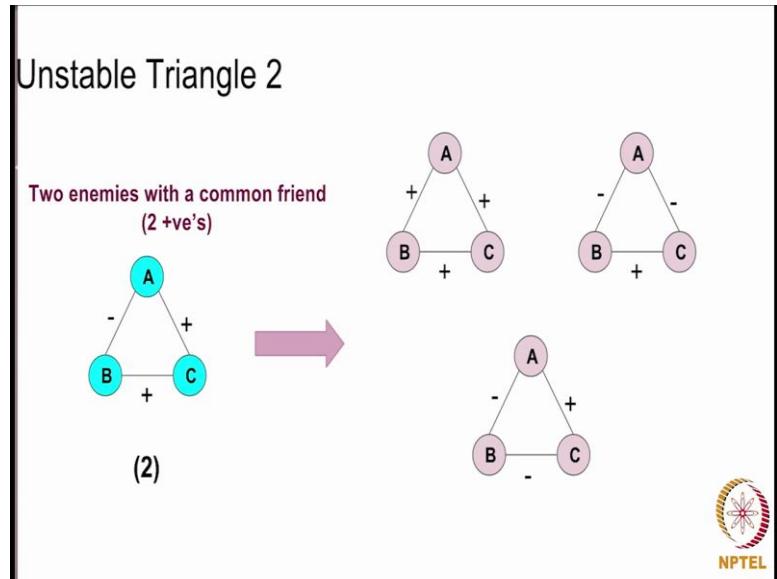
(Refer Slide Time: 06:01)



So, we are taking the first unstable triangle here where there are three mutual enemies as I said the beliefs that enemy of my enemy's enemy is my friend seems to play a role here in the network. So, when there are three negative edges which means that some of these nodes tend to make friends make friendship friendships with each other. So, it may happen that B and C become friends of each other because they are enemies there because they will then have A as a common enemy which is A or it may happen that these three negative signs lead to A and C becoming positive because they will have B as a common enemy or it may happen that A to B becomes positive because they will have a common enemy C.

So, whenever we have a triangle which has all negative edges that are 0 positive edges it can move to any one of these their stable states where there is one positive edge each. So, let us talk about the next unstable triangle which has one negative edge and 2 positive edges.

(Refer Slide Time: 07:18)



So, here there are 2 enemies who have one common friend which is see now there are they can we they can we three things which can happen in this scenario first thing is since C is a common friend to A and B C might help A and B also to become a friend right. So, A and B, A to B, h can be come positive like this. So, this is one thing that can happen second thing that can happened here is A has a friend C who is friends with an enemy of A. So, that may lead to animosity between A and C. So, this is what can happen in this case third thing that can happen is since B is friends with C who is friends with an enemy of B which is A.

So, that can lead to animosity between B and C as well. So, it can lead to this kind of situation. So, as you can see on the right hand side the three configurations one of them has three positive edges which is a; which is a stable one that that you have already know and 2 of the configurations have one positive edge which is again a stable relation. So, all these three are stable relationships. So, the relationship on the on the left hand side leads to these three configurations in the implementation we are going to start with a few configurations and we will see how they move towards stable states in the next video we are going to see the steps that you are going to follow for the implementation.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Homophily (Continued) & Positive and Negative Relationships**  
**Lecture – 69**  
**Outline of implementation**

(Refer Slide Time: 00:05)

**Steps for Implementation**

1. Create a graph with 'n' nodes, where the nodes are the countries.
2. Make it a complete graph by adding all possible edges. Also, assign '+' or '-' signs as weights to all the edges randomly.
3. Display the network.



Before we start the implementation, we are going to look at the steps that we are going to execute during with the implementation. So, that its easier for you to follow and understand the subsequent videos.

So, let us look at the bigger picture first. Our aim is to have a network of countries where the friendship or animosity details amongst these countries is stored in the network. Given this scenario a number of unstable relationships amongst the countries emerged which tend to move towards a stable state.

So, over time what happens the network starts from an unstable state and it gradually moves towards a stable state. Now, what do we mean by a stable network? Its a network where there is no unstable relationship. So although, all the triangular relationships will be stable and that network does not change after that. So, our first aim is to observe that how the network evolves from unstable state to a stable state. Now second thing that can be observed in this scenario is that. Once the network becomes stable the nodes in the

network can be divided into two groups such that: the nodes in the first group are all friends to each other, the nodes in the second group are all friends to each other. However, the nodes in the first group are enemies to the nodes in the second group

So, that is an interesting pattern that emerges. And the second aim of our implementation will be to observe this division of nodes into two groups. And we will see how the relationships existed amongst the countries initially, and how the relationships exist amongst the countries later after this evolution.

So, to start with we will create a graph with  $n$  nodes where the nodes are countries. We are going to make it a complete graph by adding  $n$  choose two edges and we are also going to randomly assign a weight either positive or negative to all these edges. And after that we will display the network. So, this is the starting step. Once the network is created our aim is to analyze the unstable triangles in the network.

(Refer Slide Time: 02:25)

### Steps for Implementation

1. Create a graph with 10 nodes, where the nodes are the countries.
2. Make it a complete graph by adding all possible edges. Also, assign '+' or '-' signs as weights to all the edges randomly.
3. Display the network.
- 4.1 Get a list of all the triangles in the network.
- 4.2 Store the sign details of all the triangles.
- 4.3 Count the number of unstable triangles in the network.



So, what we are going to do is: we will create a list of all the triangles in the network. So, if there are  $n$  nodes, they are going to be  $n$  choose three possible triangles in the network. Since we have to check whether these triangles are stable or unstable, we will store the sign details of all these triangles. Maybe, we will create a list where we will store the sign details of each and every triangle.

Now, you might remember from the previous video that if the triangle has 3 or 1 positive edges then it is stable. And if the triangle has 0 or 2 positive edges then that triangle is unstable right. So, we are going to use this detail to check whether a given triangle is unstable or not. And we will finally get the total number of unstable triangles in the network

Our aim now is to move the network from an unstable state to a stable state. Basically, we have to see what happens when there are unstable triangles in the network. Since we know that they tend to move towards a stable state so that is what we are going to implement in our video.

(Refer Slide Time: 03:34)

5. While the number of unstable triangles is not zero, do the following:
  - 5.1. Choose a triangle in the graph that is unstable.
  - 5.2. Make that triangle stable.
  - 5.3. Count the number of unstable triangles



So, what we are going to do is we are going to choose a triangle which is unstable and will move this triangle to a stable state. Now I will show it during the implementation.

So, we will make that triangle stable and after the triangle becomes stable a number of relationships change and due to that the adjacent relationships also get affected. So, the adjacent triangles may become stable or they can become unstable as well. So, this changes the total number of unstable triangles it can increase it can decrease. So, we will actually see during the implementation how it fluctuates

After this moving of one triangle from unstable state to stable state the total number of unstable triangles will change, so we are going to keep a count of that. So, we are going

to do this until the total number of unstable triangles becomes 0. So, we are going to repeat the steps 1 2 and 3 here until the whole network becomes stable.

So, that is the whole idea of making the network move from unstable to stable state.

(Refer Slide Time: 04:49)

6. Now that there is no unstable triangle in the network, it can be divided into two coalitions, such that in each coalition, the intra-edges are positive, and the inter-edges are negative.



Now, as I told you once the network becomes stable it can be divided into two groups right, where the nodes in first group are all friends and the nodes in second group are all friends, and they all and the nodes in first group are enemies to the nodes in second group. So, how can we create that network? Now we have a stable network and we have to divide it into two groups, how do we do that? There can be various ways that we can use to divide the network; what we are going to follow here is that.

(Refer Slide Time: 05:24)

6. Now that there is no unstable triangle in the network, it can be divided into two coalitions, such that in each coalition, the intra-edges are positive, and the inter-edges are negative.

6.1. Choose a random node. Add it to the first coalition.



We are going to choose a random node and we will add it to first group, after that we will look at the neighbors of this node some of these neighbors will be positive some of these neighbors will be negative. As in the some of these neighbors will be friends, some of these neighbors will be enemies.

So, we are going to take the friends of this node and we will add the all of them to the first group. And the enemies of this node we are going to add to the second group. After that we will take the friends of this node which we added to the first group, we will take them one by one and we will repeat the same process for them. That is we will we will look at their neighbors, we will add the friends to the first group, we will add its enemies to the second group.

(Refer Slide Time: 06:17)

6. Now that there is no unstable triangle in the network, it can be divided into two coalitions, such that in each coalition, the intra-edges are positive, and the inter-edges are negative.
  - 6.1. Choose a random node. Add it to the first coalition.
  - 6.2. Also put all the 'friends' of this node in the first coalition.
  - 6.3. Put all the 'enemies' of this node in the second coalition.



So, we will keep taking all the nodes from the first group which we have added, if there if they have not been processed they are going to repeat this process. So, we will put all the friends in the first group, we will put all the enemies in the second group.

(Refer Slide Time: 06:25)

6. Now that there is no unstable triangle in the network, it can be divided into two coalitions, such that in each coalition, the intra-edges are positive, and the inter-edges are negative.
  - 6.1. Choose a random node. Add it to the first coalition.
  - 6.2. Also put all the 'friends' of this node in the first coalition.
  - 6.3. Put all the 'enemies' of this node in the second coalition.
  - 6.4. Repeat steps 6.2 and 6.3 for all the 'unprocessed' nodes of first coalition.



And we will keep repeating this process until we have processed all the nodes in the first group. So, at the end of this process we will be having two groups of nodes as stated.

(Refer Slide Time: 06:37)

6. Now that there is no unstable triangle in the network, it can be divided into two coalitions, such that in each coalition, the intra-edges are positive, and the inter-edges are negative.

6.1. Choose a random node. Add it to the first coalition.

6.2. Also put all the 'friends' of this node in the first coalition.

6.3. Put all the 'enemies' of this node in the second coalition.

6.4. Repeat steps 6.2 and 6.3 for all the 'unprocessed' nodes of first coalition.

7. Display the network with coalitions



And then we will display the network with the two groups, and we will observe the kinds of relationships that existed amongst the countries initially, and the kinds of relationships that existed amongst the countries after all this evolution.

So, these were the steps that we are going to follow let us now start the implementation.

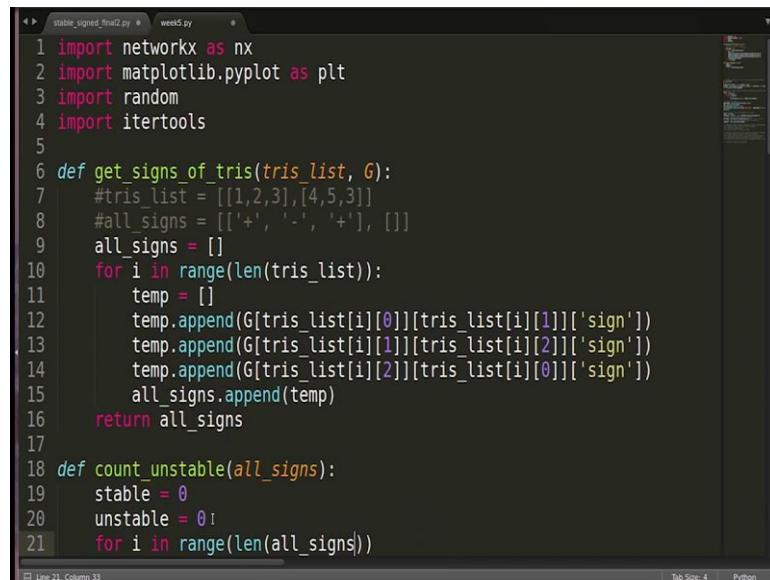
**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Homophily (Continued) & Positive and Negative Relationships**

Lecture - 70

**Creating graph, displaying it and counting unstable triangles**

(Refer Slide Time: 00:05)



```
stable_signed_final2.py  week5.py
1 import networkx as nx
2 import matplotlib.pyplot as plt
3 import random
4 import itertools
5
6 def get_signs_of_tris(tris_list, G):
7     #tris_list = [[1,2,3], [4,5,3]]
8     #all_signs = [['+', '-', '+'], []]
9     all_signs = []
10    for i in range(len(tris_list)):
11        temp = []
12        temp.append(G[tris_list[i][0]][tris_list[i][1]]['sign'])
13        temp.append(G[tris_list[i][1]][tris_list[i][2]]['sign'])
14        temp.append(G[tris_list[i][2]][tris_list[i][0]]['sign'])
15        all_signs.append(temp)
16    return all_signs
17
18 def count_unstable(all_signs):
19     stable = 0
20     unstable = 0
21     for i in range(len(all_signs)):
```

Let us get started with the implementation. So, for your convenience I have added all the steps so here so that we will pick these steps one by one and we will implement them ok. So, initially let me import the packages that we are going to need, since we are going to work with networkx.

Let me import networkx. We are also going to display the network. So, we might need matplotlib so I will import that. We will be randomly choosing the triangles out of all possible triangles. So, we will need the random package. And to choose all possible triangles we might need to use combinations function from itertools package we used it in previous video as well. So, I will import itertools. So, these were the packages.

Now let us look at the first step we have to create a graph with n nodes, where the nodes are the countries. So, let me create graph this is an empty graph as of now. So, let me add the nodes to this graph let us initialize this n to 5 and we can change it later. So, we are

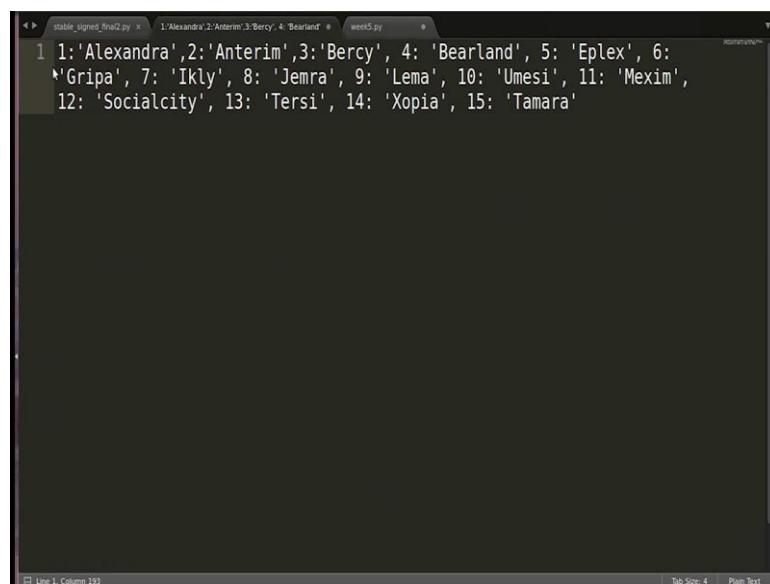
just going to start with a simple network, where there are 5 countries here increase that later.

So, n is equal to 5, we are going to add 5 nodes to it. So, I will write we have to add 5 nodes. So, let me pass a list of 5 nodes over here as a parameter. So, I will write i for i in range; the nodes should be numbered from 1 to n. So, I can write i to n + 1 because range if it starts from 1 to n + 1 it goes from 1 to n. So, this is what we are going to pass here.

Now we have added the nodes to this graph we have to assign the names to these nodes, the country names to these nodes. So, let me use dictionary; we are we are going to use dictionary for this because we have this function a networkx. That is nx.relabel\_nodes; what this function does is? It labels the nodes, because as of now the nodes are labeled from 1 to n right that is 1 to 5 in this case. We have to assign the country name as a label to all these nodes. So, they are going to make use of this function relabel nodes from networkx.

The parameters are two: the first one is a dictionary let me name this dictionary is mapping that is what will pass. This dictionary will contain the labels the new labels of the nodes. So, the key will be the old label and the value will be the new label right. The second parameter is the graph itself. So, we are going to make use of this function, for this we first need a dictionary which has the new labels.

(Refer Slide Time: 03:33)



```
stable_signed_final2.py  1/Alexandra 2/Anterim 3/Bercy 4/Bearland  week5.py
1 1:'Alexandra', 2:'Anterim', 3:'Bercy', 4: 'Bearland', 5: 'Eplex', 6:
  'Gripa', 7: 'Ikly', 8: 'Jemra', 9: 'Lema', 10: 'Umesi', 11: 'Mexim',
12: 'Socialcity', 13: 'Tersi', 14: 'Xopia', 15: 'Tamara'
```

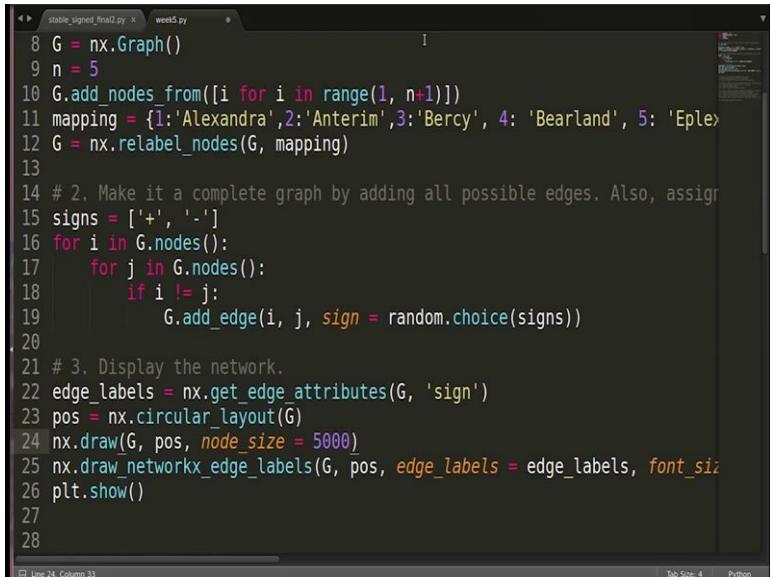
I already have this here; I have some random names of the countries. So, I am going to use these fictitious names for the countries. So, this is our mapping dictionary which we will pass to this graph and the nodes will be relabeled by this function.

So, the first step is over, we have created a graph with n nodes where the nodes are the countries. Let us go to the second step. The second step is we have to add all possible edges to this graph and we also have to assign either positive or negative to these edges. So, let us create a list with two weights: positive and negative. So, what we are going to do is: we will randomly choose one of these weights and we will assign to the edges.

Since we have to create a complete graph, we have to add all possible edges. So, how can we do that? I can start the loop here. So, I can write i in G.nodes for j in G.nodes, if i is not equal to j then we will add this edge. So, we will write G.add\_edge we have to add the edge between i and j, we also have to assign the weight.

Let us call the attribute sign; sign is equal to. So, we have to randomly choose the sign out of these. So, the function that we can use is random.choice. What this function does is: it chooses, so the parameter is a list that is the signs here out of this the values of this list it randomly chooses one value. So, out of plus and minus it will randomly choose either plus or minus and we will assign to this edge.

(Refer Slide Time: 05:22)



```
stable_signed_final2.py week5.py
8 G = nx.Graph()
9 n = 5
10 G.add_nodes_from([i for i in range(1, n+1)])
11 mapping = {1:'Alexandra', 2:'Anterim', 3:'Bercy', 4: 'Bearland', 5: 'Eplex'}
12 G = nx.relabel_nodes(G, mapping)
13
14 # 2. Make it a complete graph by adding all possible edges. Also, assign
15 signs = ['+', '-']
16 for i in G.nodes():
17     for j in G.nodes():
18         if i != j:
19             G.add_edge(i, j, sign = random.choice(signs))
20
21 # 3. Display the network.
22 edge_labels = nx.get_edge_attributes(G, 'sign')
23 pos = nx.circular_layout(G)
24 nx.draw(G, pos, node_size = 5000)
25 nx.draw_networkx_edge_labels(G, pos, edge_labels = edge_labels, font_size = 10)
26 plt.show()
27
28
```

So, now we are done with the edges. Next step is to display the network; the command to display the network is `nx.draw(G)`. However, we want to display number of other things on the network. So, we are going to use some extra features of drawing functions from networkx. For example, we can use a layout let me use circular layout so that the nodes are nicely visible. So, I will write `nx.circular_layout`.

Now, by default the edge labels are not displayed. So, we will have to use an extra function `nx.draw` networkx edge labels. So, we are going to use this function and I will tell you the parameters to this. Another thing is that when we use this function the labels will be displayed. However, the labels will be like sign is equal to plus sign is equal to minus. We do not want that to be displayed like this, you just want plus we just want minus, we just want the signs not sign is equal to plus. So, for that we will have to use one more extra function. We will write edge labels is equal to `nx.get_edge_attributes` from `G` we must take the attributes and the attribute whose values we have to take is sign. Now these edge labels we are going to pass here the first parameter is `G` itself, the second parameter is the layout that we are using and then we will pass the edge labels.

Edge labels is equal to edge labels which means the values that we are getting out of this function those values only should be displayed here. If you want, you can change the font size you can change the font color.

(Refer Slide Time: 07:25)

The screenshot shows a code editor window with a dark theme. The file name is 'stable\_signed\_final2.py'. The code is as follows:

```
10 range(1, n+1)])
11 Anterim', 3: 'Bercy', 4: 'Bearland', 5: 'Eplex', 6: 'Gripa', 7: 'Ikly', 8:
12 , G)
13
14 n by adding all possible edges. Also, assign '+' or '-' signs as weights
15
16
17
18
19 sign = random.choice(signs)
20
21
22 tributes(G, 'sign')
23
24
25 (G, pos, edge_labels = edge_labels, font_size = 20, font_color = 'red')
26
27 triangles in the network.
28 of all the triangles.
29 stable triangles in the network.
30
```

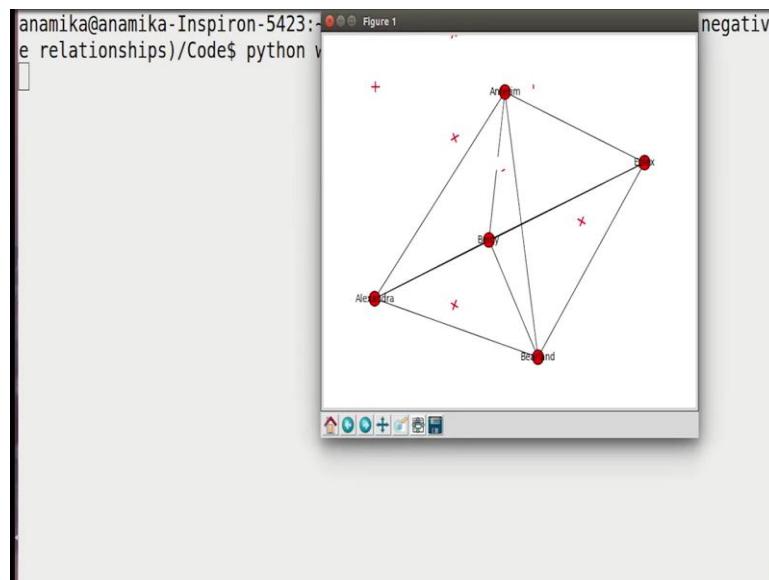
The code uses the `random` module to choose signs ('+' or '-'). It then uses the `tributes` function to assign these signs to edges. Finally, it uses the `nx.draw` function to draw the network with circular layout, edge labels, and red font.

So, let us use this font size is equal to say 20, you can also change the font color let us do that is well let us make it red. So, these are just some notification parameters you can use them if you want. So, just to show you how they work and going to show them in this video.

I think this should be sufficient. So, to display the network we have to use plt.show. So, to sum up we first added the nodes here, and we really build; oh, I think first we pass the graph and then we pass the dictionary ok. So, first we added the nodes, we relabeled the nodes, then we added the edges, and we assigned weight to these edges, and now we are just going to display this.

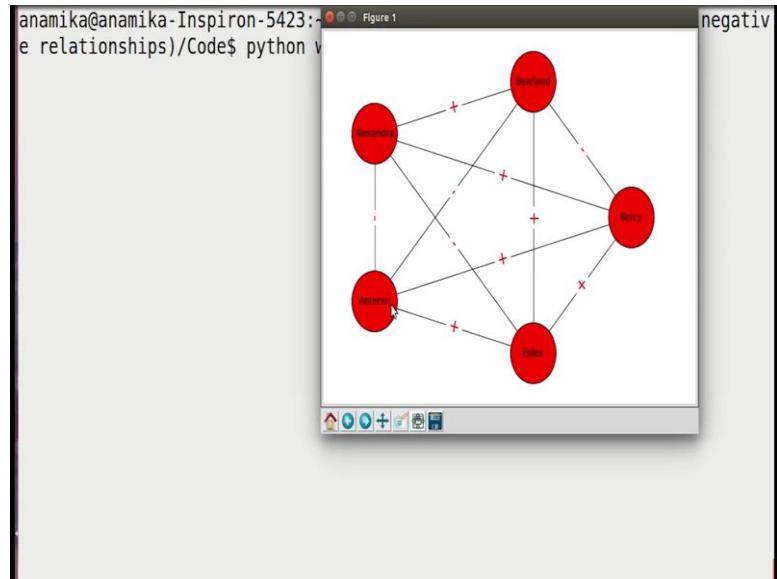
So, let us try executing this, ok.

(Refer Slide Time: 08:18)



So, we go back here and let us execute this ok. So, this is not coming in a circular layout, let us go back here ok. We have not passed this pos here while we are drawing, so we can pass this. Another thing that we can do is we can change the node; the node size is well since you saw that the country names are not visible nicely in the nodes. So, we can change the size. Let me do that node size you can specify any size I am just giving this, ok.

(Refer Slide Time: 08:55)



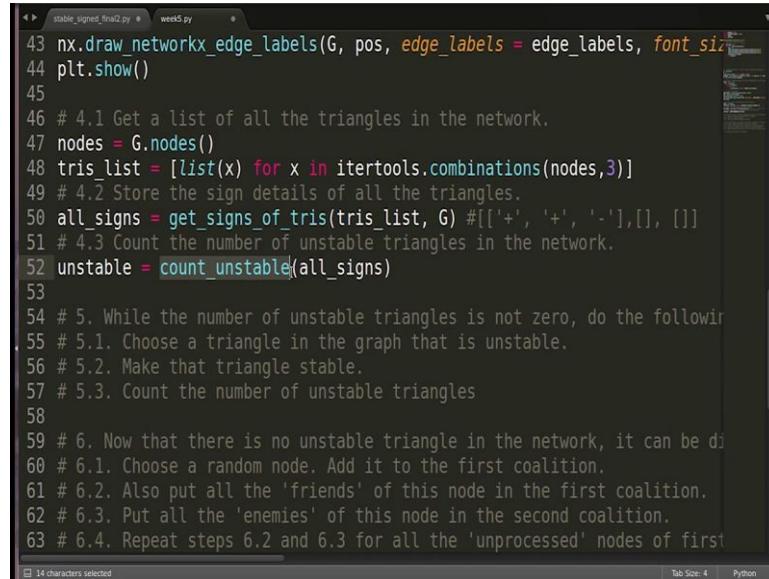
So, let us go back here and try to run it: oh, now this looks better. So, as you can see here. So, these are the 5 countries just for simplicity we have kept a smaller number of countries and will increase them later. So, the edge labels indicate whether they are friends or enemies to each other.

Now, you can see there are some unstable triangles here. For example, Alexandra, Bearland and Bercy you can see that they are all enemies to each other. So, we see that this kind of these kind of relationships do not tend to stay for a long time in the network, they tend to go towards the stable state. That is what we will show in our implementation. So, that is one unstable triangle. Another unstable triangle let us see Alexandra, Anterim, and Bearland you see they are all again enemies to each other that is another unstable triangle.

If you look at Anterim, Eplex and Bercy they are all friends to each other. So, this kind of relationship should stay as it is. So, this is the first three steps completed: we have the network with us, now we have to make it evolve from one state to the other.

Let us close it and go back to our program.

(Refer Slide Time: 10:15)



```
43 nx.draw_networkx_edge_labels(G, pos, edge_labels = edge_labels, font_size=10)
44 plt.show()
45
46 # 4.1 Get a list of all the triangles in the network.
47 nodes = G.nodes()
48 tris_list = [list(x) for x in itertools.combinations(nodes,3)]
49 # 4.2 Store the sign details of all the triangles.
50 all_signs = get_signs_of_tris(tris_list, G) #[['+', '+', '-'],[],[],[]]
51 # 4.3 Count the number of unstable triangles in the network.
52 unstable = count_unstable(all_signs)
53
54 # 5. While the number of unstable triangles is not zero, do the following
55 # 5.1. Choose a triangle in the graph that is unstable.
56 # 5.2. Make that triangle stable.
57 # 5.3. Count the number of unstable triangles
58
59 # 6. Now that there is no unstable triangle in the network, it can be deleted.
60 # 6.1. Choose a random node. Add it to the first coalition.
61 # 6.2. Also put all the 'friends' of this node in the first coalition.
62 # 6.3. Put all the 'enemies' of this node in the second coalition.
63 # 6.4. Repeat steps 6.2 and 6.3 for all the 'unprocessed' nodes of first
```

So, let us go to the fourth step now.

Now, we have the network with us and we have to observe the triangles. So, let us execute this 4.1 that is get a list of all the triangles in the network. How do we do that? To get all the triangles all the possible triangles we will have to choose three nodes from the list of nodes right, all possible combinations of three nodes. So, for that we first need list of all the nodes.

So, I am going to get that nodes is equal to G.nodes. So, we got a list of all the nodes. Now we have to out of this list we have to choose three nodes at a time. So, let us create a list: tris list which will keep which will keep all the triangles in the network. So, how do we do that? We will create a list and we will make use of combinations function here.

So, let us call it x for x in itertools.combinations out of wish list we have to choose we have to choose out of nodes list. How many choose nodes we have to choose we have to choose three nodes. So, basically what we are doing we are taking out all the combinations of three nodes from this list nodes. And since that is in the form of a tuple, we are going to store that in the form of a list and that list we will store in this list that is tris list.

So, basically this tris list will be a list of list where every item of this list will be a list having the three nodes from the network. These three nodes will indicate a triangle. Now

that we have the triangles, we have to see whether these triangles are stable or not, and which ones of these are unstable so that we can you can we can implement moving them towards a stable state.

So, for that we have to see what the weights on the edges on these triangles are. We are going to create a function for that. Before that we will create a list that we will keep track of all the signs of these edges. So, I will create a list and we will call a function we will just create that. We will pass the triangles list in that and we will pass the graph. So, this function we will just create: what this function we will do is it will take the triangles list and the graph as the parameter and it will return a list where the list where this list is a list of list. And every element of this list will have the signing details of the triangles let me write it as a comment here.

For example, this would be a list and this list will have number of other lists like this, and these lists will have the signing details of every edge. For example, this will have plus minus and so on. So, this will be list of lists this all signs which will be return by this functional will just create this function.

Now, go to the next step. Once we have gotten all the sign details of the triangles. We have to check how many unstable triangles are there. So, let me create variable here and let me create a function also: count unstable. We will pass this all signs list here so that this function can tell us how many unstable triangles in all these in all this list are, ok.

So, once these two functions are created, we are will be done with the four steps. Now, let us go back and implement this function first, get signs of triangles. So, we will go up and we will create a function. The parameters as we passed here. We basically need a triangle list and we need the graph as well, so we will pass them here. So, what do we have to do in this function as an input we have tris list? Let me show you the format of tris list. So, it's going to be list with element lists in it and this list will have the nodes; for example, 4, 5, 3. So, this is an example.

And the output has to be the sign list: like this call this one. So, basically, we have to the triangle is the one which is comprising of the nodes 1, node 2, node 3. So, we have to see the sign between 1 and 2 and we have to store it here, we have to see the sign between 2 and 3 we have to store it here, and we have to see the sign between 2 and 3 and we have to store it here right. So, the output will be something like this for example.

So, this plus indicates that the sign between first and second node of the triangle is plus. This minus indicates that the sign between second and third node of the triangle is this, and this indicates that the sign between first and third node of the triangle is this. So, this is how we are going to create the all signs list and that is what we will return from this function. Similarly, the format of the signal list and all the subsequent lists will be like this only.

So, we will create this entry list that we are going to return. Now for every list inside this what we have to do? We have to do something, so we are going to start a loop for every element list inside this list; so for i in range length of tris list. Now how do we get the sign details of these edges? The sign details are stored in the graph.

So, we are going to take the signs from the graph only. So, all the signs is an empty list it should be returned as this list. So, we have to add a list as an element here. So, I will create our temporary list and I will keep adding values to this list and ultimately we will add this list to all signs. So, how can we get the sign? The sign we will get from G and tris list i-th elements 0th entry and the first entry. This will be the first node.

And the second node will be tris list i-th element first element; I am sorry one will be there ok. So, what we are doing is; we are passing first parameter here which is tris list i-th elements 0th value. So, if this is the i-th element the 0th value will be 1. And second parameter is tris list i-th elements first entry; so i-th elements first entry which is 2 here. So, we want to get the sign of the edge between 1 and 2. So, the third parameter is the sign that we will pass here.

Now whatever sign we get out of here we have to store that in temp. So, I will write temp.append this whole thing. Similarly, we have to get the sign between 2 and 3 and we have to get the sign between 1 and 3. So, I am just going to copy paste this. So, initially this is the sign between 0 and 1, the second will be the sign between 1 and first and second entry and the third will be sign between second and 0th entry right or 0th were second entry this is undirected. So, you can use it either way.

So, by this time we have gotten this temp list which we have to store in all signs. So, I am going to write all signs.append this temp right. So, we have done it for all the all the lists in the tris list, in the end of this list we can in the end of this far loop we can return all signs. This was the implementation of the first function that is get signs of triangle.

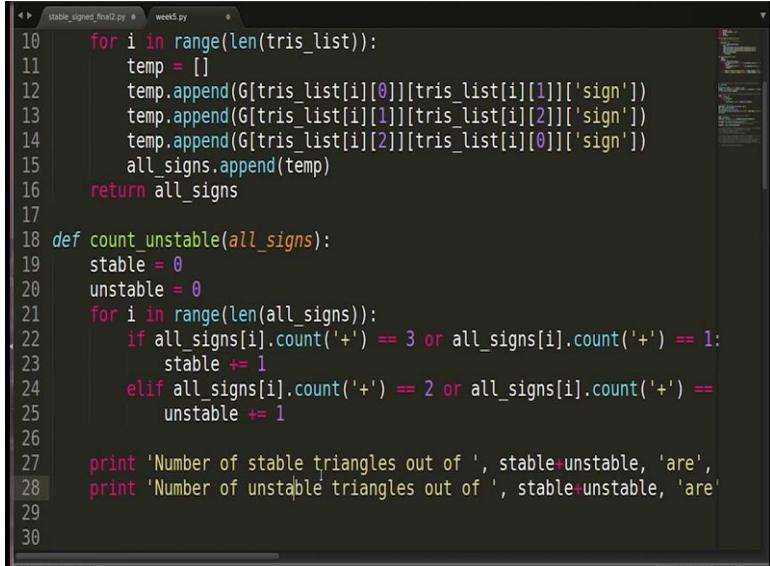
So, we have now stored the signing details of all the triangles. And now let us get back here the second function that we have been using and I am we are yet to implement is count unstable.

So, now, we have all the sign details in this list, all the signs which has been returned by this function. This list we will pass to the next function which will count the number of plus in every list and we will tell us whether it is stable or not and it also we will keep a track of the total number of unstable triangles.

So, let us now implement this function count unstable. Let us go up here, the parameter was all signs right and it is returning variable which is the unstable. So, we will write all signs here and it is returning a variable unstable. Let us also keep a track of stable triangles. So, initialize to 0, Now we have all signs as a parameter, and you know that the format of all signs is like this where every element list has the number of causes plus and minus given in it.

So, we can simply count the number of plus and minus, and we can get to know whether the triangle is stable or not. So, we can start loop here length off all signs I am sorry, length off all signs.

(Refer Slide Time: 21:14)



```
stable_signed_final2.py  week5.py
10     for i in range(len(tris_list)):
11         temp = []
12         temp.append(G[tris_list[i][0]][tris_list[i][1]]['sign'])
13         temp.append(G[tris_list[i][1]][tris_list[i][2]]['sign'])
14         temp.append(G[tris_list[i][2]][tris_list[i][0]]['sign'])
15         all_signs.append(temp)
16     return all_signs
17
18 def count_unstable(all_signs):
19     stable = 0
20     unstable = 0
21     for i in range(len(all_signs)):
22         if all_signs[i].count('+') == 3 or all_signs[i].count('+') == 1:
23             stable += 1
24         elif all_signs[i].count('+') == 2 or all_signs[i].count('+') ==
25             unstable += 1
26
27     print 'Number of stable triangles out of ', stable+unstable, 'are',
28     print 'Number of unstable triangles out of ', stable+unstable, 'are'
29
30
```

So, for every element we have to pick one element list from all the signs, and we have to count the number of plus minus in that. Or we can simply count the number of plus as we discussed in the previous video.

So, if all the signs i-th list.count; what do we have to count we have to count plus if it is equal to 3 or if this is equal to 1; then what we have discussed in that case the list will be stable. So, we can increase the stable count in this case. On the other hand, let us use elif. On the other hand, if the total number of plus is equal to 0 or 2 then to be unstable; so, I am just copying pasting here. So, if the total number of plus is 2 or it is 0, then it will be unstable. So, we will increase the unstable count here. So, I think we are done.

So, by the end of this for loop we would have the total number of stable and unstable triangles in the network, which we want to keep track of to know whether the complete network has become stable or not. So, we can print this information so that we can keep track of it. So, let us print. Number of stable triangles; I let us also within the total number of triangles ok. Similarly, I am just doing it so that while executing the code we can keep track of what is going on; number of unstable triangles out of stable plus unstable are unstable.

(Refer Slide Time: 23:27)

```
10 n range(len(tris_list)):
11 o = []
12 o.append(G[tris_list[i][0]][tris_list[i][1]]['sign'])
13 o.append(G[tris_list[i][1]][tris_list[i][2]]['sign'])
14 o.append(G[tris_list[i][2]][tris_list[i][0]]['sign'])
15 signs.append(temp)
16 all_signs
17
18 nstable(all_signs):
19 = 0
20 e = 0
21 n range(len(all_signs)):
22 all_signs[i].count('+') == 3 or all_signs[i].count('+') == 1:
23 stable += 1
24 f all_signs[i].count('+') == 2 or all_signs[i].count('+') == 0:
25 unstable += 1
26
27 Number of stable triangles out of ', stable=stable, 'are', stable
28 Number of unstable triangles out of ', stable-unstable, 'are', unstable
29
30
```

So, I think we are done with these two functions implementation let us go back here and ok.

So, we have implemented this, and we have implemented this. So, it should work fine let us try executing it and see whether it is telling us the stable and unstable triangle count or not. Let us execute this and let us check. This is the graph that we just visualized; I am going to close it ok.

(Refer Slide Time: 23:57)

```
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_5_(Positive and negative relationships)/Code$ python week5.py
Number of stable triangles out of 10 are 4
Number of unstable triangles out of 10 are 6
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_5_(Positive and negative relationships)/Code$
```

We are getting this number of stable triangles out of 10 or 4. So, total number of triangles are 10, and the stable are 4 and unstable are 6 ok. So, this is working fine, let us go back.

In the next part of the video we will see how we can choose an unstable triangle and how we can move it to a stable state. So, we will do it for all the triangles and the network will eventually become stable.

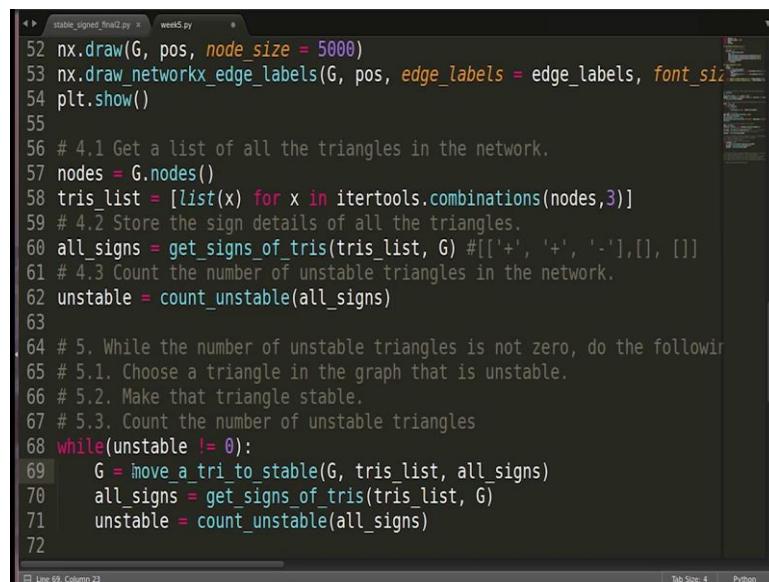
**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Homophily (Continued) & Positive and Negative Relationships**

**Lecture – 71**

**Moving a network from an unstable to stable state**

(Refer Slide Time: 00:05)



```
stable_signed_final2.py
52 nx.draw(G, pos, node_size = 5000)
53 nx.draw_networkx_edge_labels(G, pos, edge_labels = edge_labels, font_size = 10)
54 plt.show()
55
56 # 4.1 Get a list of all the triangles in the network.
57 nodes = G.nodes()
58 tris_list = [list(x) for x in itertools.combinations(nodes,3)]
59 # 4.2 Store the sign details of all the triangles.
60 all_signs = get_signs_of_tris(tris_list, G) #[[ '+', '+', '-' ], [], []]
61 # 4.3 Count the number of unstable triangles in the network.
62 unstable = count_unstable(all_signs)
63
64 # 5. While the number of unstable triangles is not zero, do the following.
65 # 5.1. Choose a triangle in the graph that is unstable.
66 # 5.2. Make that triangle stable.
67 # 5.3. Count the number of unstable triangles
68 while(unstable != 0):
69     G = move_a_tri_to_stable(G, tris_list, all_signs)
70     all_signs = get_signs_of_tris(tris_list, G)
71     unstable = count_unstable(all_signs)
72
```

In the previous video we created a network on countries; we also stored the positive and negative signs across the edges. We also computed the total number of unstable triangles in the network. Our next aim is to move this network to a stable state for that what we will do is we will choose one triangle which is unstable, and we will move it to a stable state. We will keep doing this until the complete network become unstable.

So, we have to see how the number of unstable triangles fluctuates and how it changes with respect to each iteration. So, let us get started on that. We have this variable `unstable` which keeps track of the number of unstable triangles. So, we can start the loop here while this `unstable` become 0. So, we can write `while unstable is not equal to 0` we have to keep repeating this. Now, what do we have to keep repeating?

Since we have to repeat this it will be better if you we create a function for this particular task. So, let me create a function here maybe move a triangle to stable state, we need to pass the graph here because that is what we will get changed during this task. And we

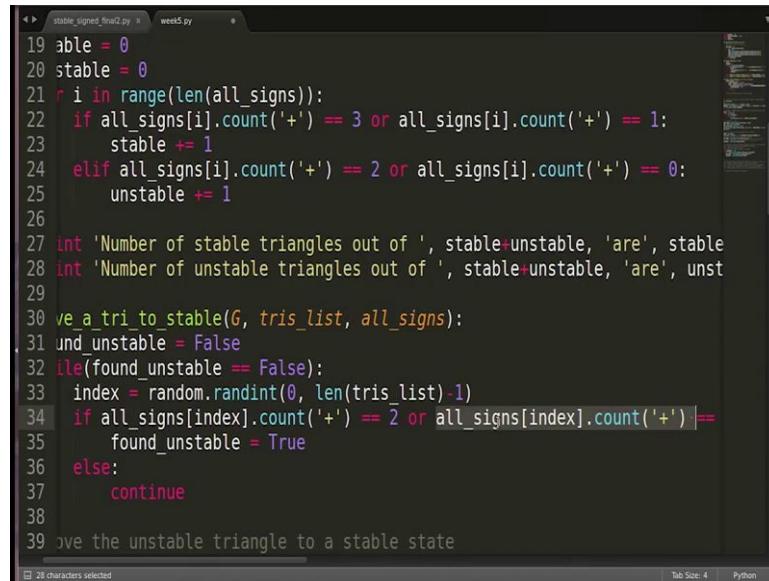
also need to pass all the triangles list we might also need all the signs there. So, let us pass this and in this function, we will make some changes in the graph with respect to the signs of one of the triangles.

So, we too want the updated graph back. So, this is the function we will just implement. Once we get the updated graph that is once one of the triangles has moved to a stable state; the signs of the adjacent triangles might also get affected. So, we need to recompute the signs of the adjacent triangles. So, we can do one thing we can do one of the two things. We can either recompute the signs of the adjacent triangles only or we can recompute the signs of all the triangles.

Here for simplicity I am recomputing the signs of all the triangles. So, basically, I am going to call this function once again; get signs of triangles which we have already created. So, I am going to call this once again on the updated graph G here; also once we get all the signs we need to check how many unstable triangles are there now right. So, we need to get the updated count here. So, we will call this function count unstable once again.

So, now up this variable unstable has the updated count of the number of unstable triangles. So, after that the control will go back here if this is not equal to 0 it will go back again inside; where it will go inside this function; where it will choose one of the one of the triangles which is unstable and we will move it to the stable state and so on. So, this loop will keep running until the entire network become unstable. So, we are done with this part we are just we just have to implement this function or move a triangle to stable.

(Refer Slide Time: 03:31)



```
stable_signed_final2.py  week5.py
19 able = 0
20 stable = 0
21 for i in range(len(all_signs)):
22     if all_signs[i].count('+') == 3 or all_signs[i].count('+') == 1:
23         stable += 1
24     elif all_signs[i].count('+') == 2 or all_signs[i].count('+') == 0:
25         unstable += 1
26
27 int 'Number of stable triangles out of ', stable+unstable, 'are', stable
28 int 'Number of unstable triangles out of ', stable+unstable, 'are', unst
29
30 def a_tri_to_stable(G, tris_list, all_signs):
31     und_unstable = False
32     while found_unstable == False:
33         index = random.randint(0, len(tris_list)-1)
34         if all_signs[index].count('+') == 2 or all_signs[index].count('+') ==
35             found_unstable = True
36         else:
37             continue
38
39 move the unstable triangle to a stable state
```

So, let us see how we can do that; let me copy the syntax let go up and let us create this function. Now we have the graph G, we have the list of all the triangles, we have the list of all the signs on these triangles. Our aim is to choose one triangle which is unstable, now how do we do that? So, what we can do here is; we can choose one triangle out of this list randomly and then we can check whether this triangle is unstable or not. If this triangle is unstable, we have founded a triangle which we have to work on, if this triangle is not stable. I am sorry if this triangle is stable we will check another triangle.

So, we have to keep checking the triangles until we get an unstable triangle. So, how do we do that? Let us create a flag maybe found unstable = false. So, initially we have not found any unstable triangle we will keep running this loop until we find one. So, write while found unstable; while found unstable = false we have to keep running this. We have to keep running what basically we have to choose one triangle randomly out of it.

So, what we can do is; we can make use of a function run in which is there in random package. So, maybe index is the variable that will store the index of the triangle. So, so we can write random.randint. So, we can start from 0 length of triangles list minus 1. So, basically what we are doing is this triangle this tris list has some length say 0 to x. We have to choose one index out of 0 to x randomly, right.

So, what we are doing here random.randint this randint function will provide a random integer out of 0 to len(tris\_list) - 1. We are doing minus 1 because the next is starting

from 0. So, that index will be stored here; now we have to check whether the triangle which is stored at this index is unstable or not. Now how do we know can we check that? We have this list all signs, which is storing all the signs for every triangle. So, we can check whether the triangle which is existing at this index has 0 pluses 1 pluses 2 pluses 3 pluses. You know what I mean. We will count the number of plus their positive sign is there.

So, accordingly we will judge whether the triangle is unstable or not let us see how we do that. So, what we will do if all signs the triangle which is available at this index all signs index.count; we have to count the number of plus there.

(Refer Slide Time: 06:51)

```

19 = 0
20 le = 0
21 in range(len(all_signs)):
22 all_signs[i].count('+') == 3 or all_signs[i].count('+') == 1:
23 stable += 1
24 if all_signs[i].count('+') == 2 or all_signs[i].count('+') == 0:
25 unstable += 1
26
27 'Number of stable triangles out of ', stable+unstable, 'are', stable
28 'Number of unstable triangles out of ', stable+unstable, 'are', unstable
29
30 tri_to_stable(G, tris_list, all_signs):
31 unstable = False
32 found_unstable = False:
33 dex = random.randint(0, len(tris_list)-1)
34 all_signs[index].count('+') == 2 or all_signs[index].count('+') == 0:
35 found_unstable = True
36 se:
37 continue
38
39 the unstable triangle to a stable state

```

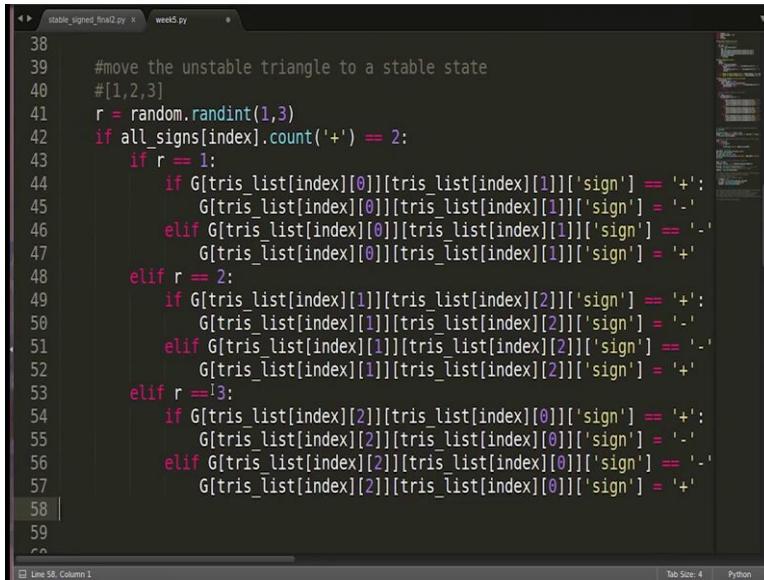
If this = 2 or if this = 0; that means the triangle is unstable. So, we can write flag we have this found\_unstable = true; which means we have found an unstable triangle.

In case the number of plus is not equal to 2 or 0, which means the triangle that we found was a stable one so we will continue with our loop. Now that we have found an unstable triangle, now that we have found an unstable triangle, we have to move it to a stable state. Now how do we do that? Firstly, let me write it here move the unstable triangle to a stable state ok.

Now, how do we do this? Firstly, the point to be noted is that there are two kinds of unstable triangles; one kind of unstable triangles are the ones where the number of

positive edges is two. And the other kinds of unstable triangles are the ones where the number of positive edges is 0. So, we are going to take different action in both the cases.

(Refer Slide Time: 08:08)



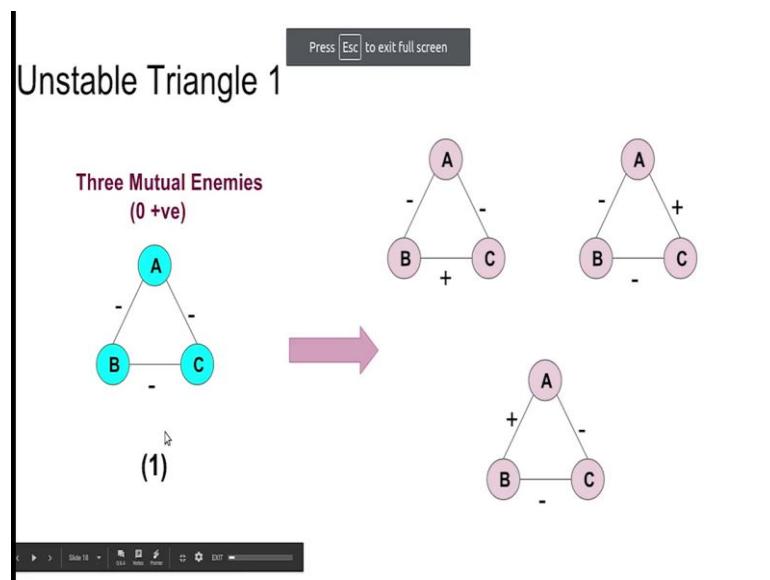
```

38
39     #move the unstable triangle to a stable state
40     #[1,2,3]
41     r = random.randint(1,3)
42     if all_signs[index].count('+') == 2:
43         if r == 1:
44             if G[tris_list[index][0]][tris_list[index][1]]['sign'] == '+':
45                 G[tris_list[index][0]][tris_list[index][1]]['sign'] = '-'
46             elif G[tris_list[index][0]][tris_list[index][1]]['sign'] == '-':
47                 G[tris_list[index][0]][tris_list[index][1]]['sign'] = '+'
48         elif r == 2:
49             if G[tris_list[index][1]][tris_list[index][2]]['sign'] == '+':
50                 G[tris_list[index][1]][tris_list[index][2]]['sign'] = '-'
51             elif G[tris_list[index][1]][tris_list[index][2]]['sign'] == '-':
52                 G[tris_list[index][1]][tris_list[index][2]]['sign'] = '+'
53         elif r == 3:
54             if G[tris_list[index][2]][tris_list[index][0]]['sign'] == '+':
55                 G[tris_list[index][2]][tris_list[index][0]]['sign'] = '-'
56             elif G[tris_list[index][2]][tris_list[index][0]]['sign'] == '-':
57                 G[tris_list[index][2]][tris_list[index][0]]['sign'] = '+'
58
59

```

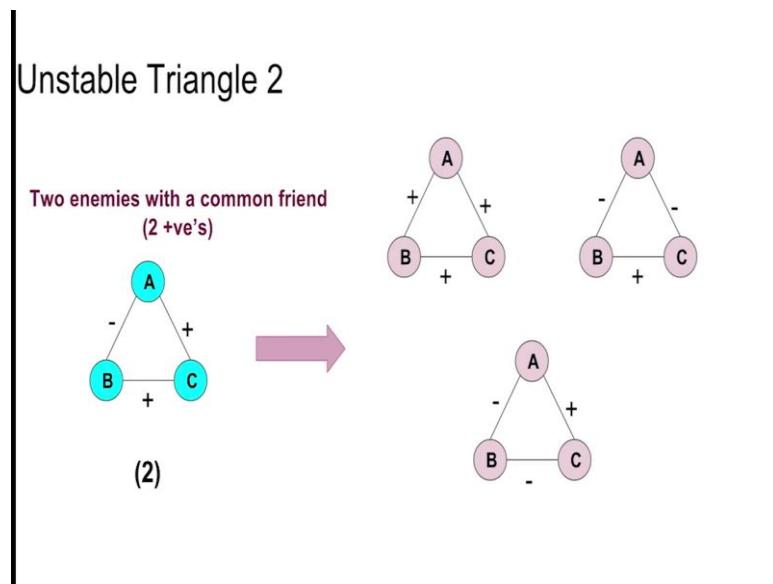
So, we are going to check in the given triangle the number of positive edges; based on that only we will move it to the appropriate state.

(Refer Slide Time: 08:22)



Now, if you remember from one of the previous videos, I had explained the two cases of unstable triangles let me show you once again. So, here you see there are two kinds of unstable triangles unstable triangle 1.

(Refer Slide Time: 08:33)



Unstable triangle 2 let us consider this case first. Now this is an unstable triangle where the number of plus is 2; in this case what happens now this state can be moved to any one of these three states. Now, what are these states? In the first case what happens this negative sign between A to B this negative sign has become positive and the rest signs remain the same.

In the second case this minus and plus remain the same, but this plus sign becomes negative this is the second case. In the third case this minus and this plus remain the same here and this plus sign becomes negative.

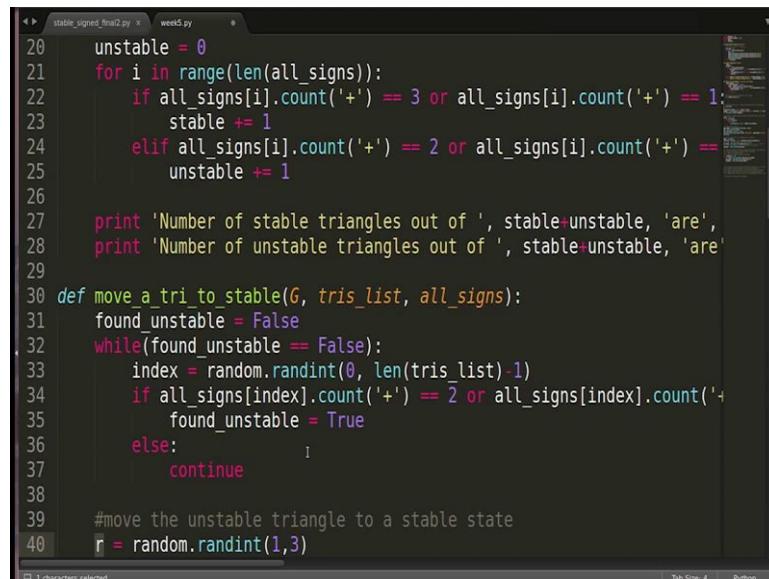
So, basically you see what is happening here in the first case this one this sign is getting inverted. In the second case the sign of the second edge is getting inverted. In the third case the sign of the third edge is getting inverted. So, basically if you take this triangle and choose any one of the edge and invert its sign; whatever you will get that will be a stable state right. So, this is the case of a triangle which has 2 positive edges. If you go back and see the case of the triangle which has 0 positive edges the scenario is a little different here.

So, accordingly we are going to implement as well so let us try to understand this. In this case this triangle which is unstable has 0 positive edges that is all edges are negative. Now these three states are the ones which are stable states which can emerge out of this state. Now what is happening in all these states? In the first state this minus these two

states are these two edges are same and the third edge becomes positive from negative. In the second state this edge becomes positive in the third state A to B edge becomes positive.

So, basically what is happening here is you choose any one of the edges and make it positive; whatever state you get will be a stable state. So, this is how we are going to implement as well. So, you see the moving of unstable state to a stable state is different in both the cases when we have 0 positives or when you have 2 positives. So, we have to keep this thing in mind while we are implementing it. So, let us go back here.

(Refer Slide Time: 11:11)



```

20     unstable = 0
21     for i in range(len(all_signs)):
22         if all_signs[i].count('+') == 3 or all_signs[i].count('+') == 1:
23             stable += 1
24         elif all_signs[i].count('+') == 2 or all_signs[i].count('+') ==
25             unstable += 1
26
27         print 'Number of stable triangles out of ', stable+unstable, 'are',
28         print 'Number of unstable triangles out of ', stable+unstable, 'are'
29
30 def move_a_tri_to_stable(G, tris_list, all_signs):
31     found_unstable = False
32     while(found_unstable == False):
33         index = random.randint(0, len(tris_list)-1)
34         if all_signs[index].count('+') == 2 or all_signs[index].count('+')
35             found_unstable = True
36         else:
37             continue
38
39     #move the unstable triangle to a stable state
40     r = random.randint(1,3)

```

Another thing to note here is that there are three stable states corresponding to one unstable state. So, we will randomly assign one of the three stable states here. So, we will choose a random number here. So, you again use random package random.randint. So, we want a random integer from 1 to 3. So, any one of these three integers can come in r. So, if r = 1 we will move it to the first stable state if r = 2 we will move it to the second stable state and so on.

So, let us handle the first case where the number of a positive signs = 2; let us handle this case first. I am copying pasting here. So, let us see what we do if all\_signs[index].count('plus') = 2. So, as I explained you the 3 stable states that can correspond that can come out of this state. So, the strategy that we are going to follow here is that if r = 1; we will invert the sign of the first edge.

If  $r = 2$  we will invert the sign of the second edge if  $r = 3$  we will invert the sign of the third edge. So, that is the approach that we are going to follow here. Now how do we invert the sign of a given edge for that we should know what the sign of that edge is. How do we get to know what is the sign of a given edge that detail is stored in  $G$  basically  $G$  has the detail of the sign of a given edge? And how do we access the edges? These edges we will access from tris list.

So, let us see how we can do this. So, write if  $G$  so we will pass two parameters here and this the third parameter will be sign as usual simple syntax. So, what is going to the first parameter? First parameter is going to be the node the node of the first edge. How do we access the node of the first edge? All the nodes of the triangle are available in tris list. So, we are going to write tris list which index the index the index that we computed here over here. So, tris list index first node and in the second parenthesis what will come I am pasting here.

Tris list index first a value that is the first index basically means the second value ok. Let me explain it here if you find it is difficult. So, tris list is a list of lists and every list is like this ok. So, 1 will be accessed by the 0th index, 2 will be accessed by the first index and 3 will be accessed by the second by the index two which we are going to use in the next sentence. So, this will basically give us 1 and this will basically give us 2.

So, we are going to pass 1 and 2 to this  $G$  to get the sign of that  $G$ . So, so we will check whether that is positive or not; if that is positive will make it negative then that is negative and make it positive. So, we will just copy paste here. So, this was positive so we will make it negative ok. Now the same has to be done with the rest of the edges as well. So, I will copy paste I remove this; now this is for the second edge.

The second edge will be accessed with index 1 and 2 here again index 1 and 2 ok; so again, if that is positive this one catch here. So, see what we are doing here is; if the sign of the first edge was positive, we are making it negative, but if the sign of the first edge is negative, we have to make it positive as well. So, this thing we are going to do in the next statement.

So, here we are just copy pasting this. So, let us check this if the sign of the first edge was positively made it negative and if it was negative sorry negative. We will make it positive. So, we are done with the first edge and we will handle the first edge if the value

of  $r$  was equal to 1. What is the value of  $r = 2$ ? Then we have to choose the second edge. So, I will write else if  $r = 2$  we will now handle the second edge.

Now, again we will check the sign of the second edge we just copy paste things here. If the sign of the first edge was positive, we will make it negative and otherwise I will just copy paste again. So, I will write elif if it was negative, we will make it positive. Since we are repeating things here you can also create a function over here. So, that you pass the values and it makes the changes I am just copy pasting here. And the third thing is if  $I$  if the value of  $r = 3$  we will choose the third edge and we will invert its sign.

So, again I will copy paste the entire thing. So, I will write if  $r = 3$  now we have to access the third edge. So, we will write 2 comma 0 or you can write 0 comma 2; write 2 0 like here like here.

(Refer Slide Time: 17:49)

```
38
39 #move the unstable triangle to a stable state
40 #[1,2,3]
41 r = random.randint(1,3)
42 if all_signs[index].count('+') == 2:
43     if r == 1:
44         if G[tris_list[index][0]][tris_list[index][1]]['sign'] == '+':
45             G[tris_list[index][0]][tris_list[index][1]]['sign'] = '-'
46         elif G[tris_list[index][0]][tris_list[index][1]]['sign'] == '-':
47             G[tris_list[index][0]][tris_list[index][1]]['sign'] = '+'
48     elif r == 2:
49         if G[tris_list[index][1]][tris_list[index][2]]['sign'] == '+':
50             G[tris_list[index][1]][tris_list[index][2]]['sign'] = '-'
51         elif G[tris_list[index][1]][tris_list[index][2]]['sign'] == '-':
52             G[tris_list[index][1]][tris_list[index][2]]['sign'] = '+'
53     elif r == 3:
54         if G[tris_list[index][2]][tris_list[index][0]]['sign'] == '+':
55             G[tris_list[index][2]][tris_list[index][0]]['sign'] = '-'
56         elif G[tris_list[index][2]][tris_list[index][0]]['sign'] == '-':
57             G[tris_list[index][2]][tris_list[index][0]]['sign'] = '+'
58
59
```

So, if it was positive, we made it negative if it is negative, we made it positive. So, all set; now this was the case of the triangles where the number of positive edges were 2.

Next, we have to handle the case where the number of positive edges = 0. In that case as I told you what we will do we will choose any of the edge since all the edges are negative already, we will choose any one of the edges and we will make it positive. So, the resultant state is a positive is a stable state. So, what I will do is; we will write in the else part, if I know this if ok.

(Refer Slide Time: 18:23)

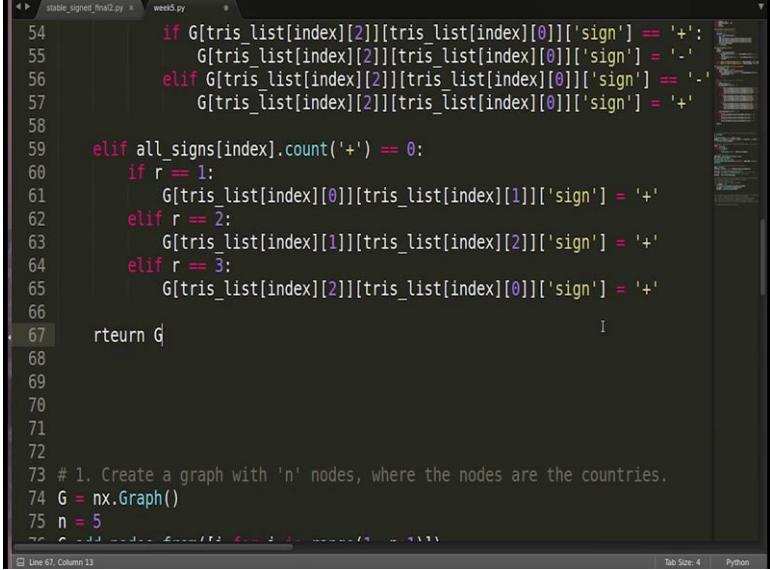
```
stable_signed_final2.py  week5.py  0
46     elif G[tris_list[index][0]][tris_list[index][1]]['sign'] == '-':
47         G[tris_list[index][0]][tris_list[index][1]]['sign'] = '+'
48     elif r == 2:
49         if G[tris_list[index][1]][tris_list[index][2]]['sign'] == '+':
50             G[tris_list[index][1]][tris_list[index][2]]['sign'] = '-'
51         elif G[tris_list[index][1]][tris_list[index][2]]['sign'] == '-':
52             G[tris_list[index][1]][tris_list[index][2]]['sign'] = '+'
53     elif r == 3:
54         if G[tris_list[index][2]][tris_list[index][0]]['sign'] == '+':
55             G[tris_list[index][2]][tris_list[index][0]]['sign'] = '-'
56         elif G[tris_list[index][2]][tris_list[index][0]]['sign'] == '-':
57             G[tris_list[index][2]][tris_list[index][0]]['sign'] = '+'
58
59     elif all_signs[index].count('+') == 0:
60         if r == 1:
61             G[tris_list[index][0]][tris_list[index][1]]['sign'] = '+'
62         elif r == 2:
63             G[tris_list[index][1]][tris_list[index][2]]['sign'] = '+'
64         elif r == 3:
65             G[tris_list[index][2]][tris_list[index][0]]['sign'] = '+'
66
67
```

If the number of a positive edge = 0; what we are going to do ok. Again, there can be three cases here based on the value of r we will choose the edge. So, we will write if  $r = 1$  we will choose the first edge and I we will do that else if  $r = 2$  we will choose the second edge else if  $r = 3$  we will choose a third edge. Now how do we choose the edges the same way that we have done. So, like you can choose like this.

Now, there is no checking here you just have to make the sign positive; because we know that already the sign is negative. So, I will write it here the sign of the first edge we are making to be positive ok. If  $r = 2$  we will choose the second edge and second edge will be accessed by a 1 and 2; here we will make it positive. And if  $r = 3$  we will choose the third edge and that will be accessed by 2 and 0 here ok.

So, by doing this we have handled both the cases of unstable triangles and we have. So, basically what we have done in this function we have chosen 1 unstable triangle over here ok. We have found the index of a triangle which is unstable then we are checking whether; what is the kind of that unstable triangle; based on that we are taking the action. Let us go back sorry based on that we are taking action if the signs are 2 we are executing this if the signs are 0 positive signs is 0 we are executing this.

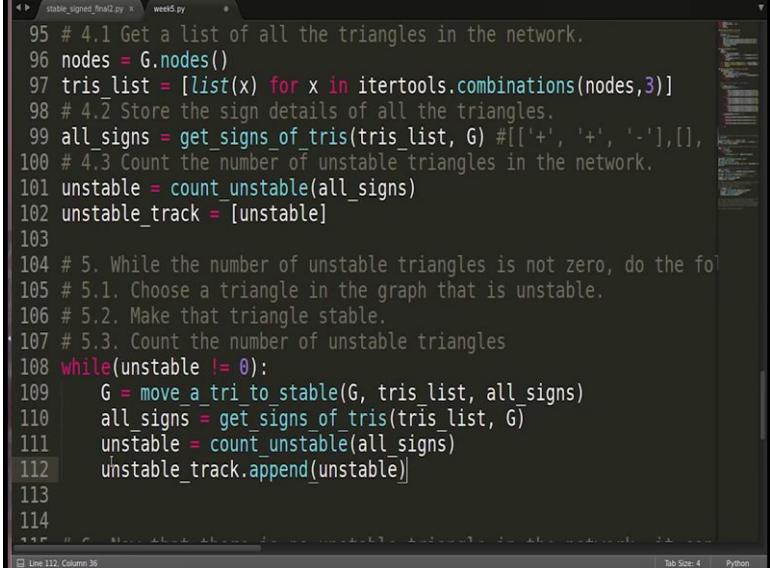
(Refer Slide Time: 20:24)



```
54     if G[tris_list[index][2]][tris_list[index][0]]['sign'] == '+':
55         G[tris_list[index][2]][tris_list[index][0]]['sign'] = '-'
56     elif G[tris_list[index][2]][tris_list[index][0]]['sign'] == '-':
57         G[tris_list[index][2]][tris_list[index][0]]['sign'] = '+'
58
59 elif all_signs[index].count('+') == 0:
60     if r == 1:
61         G[tris_list[index][0]][tris_list[index][1]]['sign'] = '+'
62     elif r == 2:
63         G[tris_list[index][1]][tris_list[index][2]]['sign'] = '+'
64     elif r == 3:
65         G[tris_list[index][2]][tris_list[index][0]]['sign'] = '+'
66
67 return G
68
69
70
71
72
73 # 1. Create a graph with 'n' nodes, where the nodes are the countries.
74 G = nx.Graph()
75 n = 5
```

So, we are done here and then we will return the updated graph here.

(Refer Slide Time: 20:32)



```
95 # 4.1 Get a list of all the triangles in the network.
96 nodes = G.nodes()
97 tris_list = [list(x) for x in itertools.combinations(nodes, 3)]
98 # 4.2 Store the sign details of all the triangles.
99 all_signs = get_signs_of_tris(tris_list, G) #[['+', '+', '-'], [], ...]
100 # 4.3 Count the number of unstable triangles in the network.
101 unstable = count_unstable(all_signs)
102 unstable_track = [unstable]
103
104 # 5. While the number of unstable triangles is not zero, do the following:
105 # 5.1. Choose a triangle in the graph that is unstable.
106 # 5.2. Make that triangle stable.
107 # 5.3. Count the number of unstable triangles
108 while(unstable != 0):
109     G = move_a_tri_to_stable(G, tris_list, all_signs)
110     all_signs = get_signs_of_tris(tris_list, G)
111     unstable = count_unstable(all_signs)
112     unstable_track.append(unstable)
113
114
```

Let us go back here and see if everything works fine. So, here we are calling this function which we have just now implemented and after calling this function, we are again computing all the signs, and we are counting the number of unstable triangles. This function is returning this variable unstable we can also keep track of this variable.

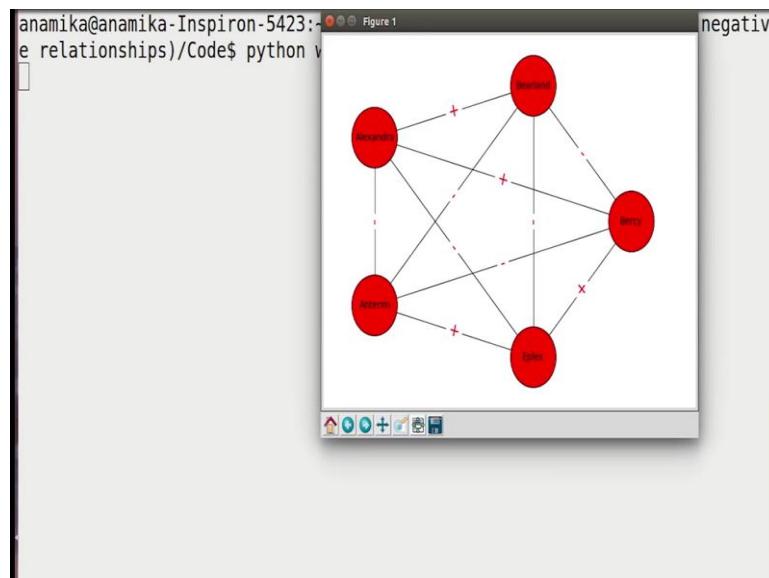
(Refer Slide Time: 20:57)

```
stable_signed_final2.py  week5.py
10 for i in range(len(tris_list)):
11     temp = []
12     temp.append(G[tris_list[i][0]][tris_list[i][1]]['sign'])
13     temp.append(G[tris_list[i][1]][tris_list[i][2]]['sign'])
14     temp.append(G[tris_list[i][2]][tris_list[i][0]]['sign'])
15     all_signs.append(temp)
16 return all_signs
17
18 count_unstable(all_signs):
19     stable = 0
20     unstable = 0
21     for i in range(len(all_signs)):
22         if all_signs[i].count('+') == 3 or all_signs[i].count('+') == 2:
23             stable += 1
24         elif all_signs[i].count('+') == 2 or all_signs[i].count('+') == 1:
25             unstable += 1
26
27 print 'Number of stable triangles out of ', stable+unstable, 'are'
28 print 'Number of unstable triangles out of ', stable+unstable, 'are'
29 return unstable
30
```

This one statement missing here I am sorry; we did not return this variable.

We have to return unstable here.

(Refer Slide Time: 21:19)



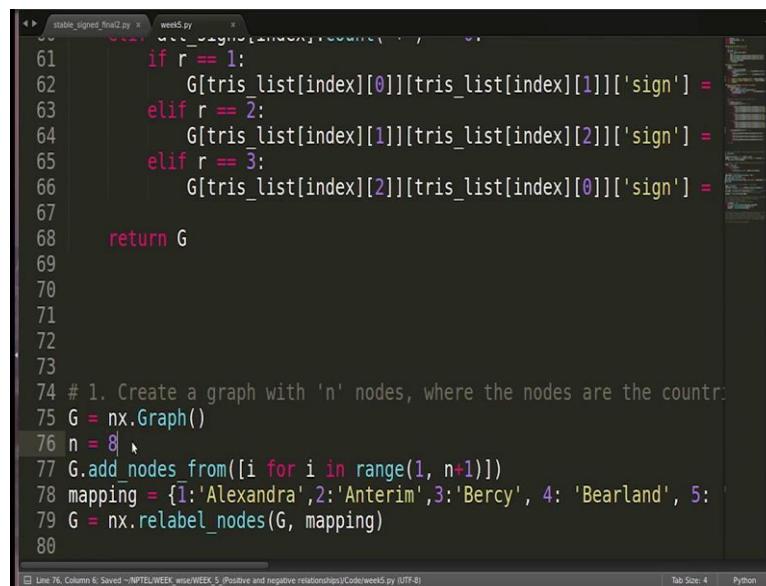
So let us check, let us verify our code we can check this. So, so this is the network that we had plotted that we had drawn earlier.

(Refer Slide Time: 21:28)

```
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_5_(Positive and negative relationships)/Code$ python week5.py
Number of stable triangles out of 10 are 6
Number of unstable triangles out of 10 are 4
Number of stable triangles out of 10 are 7
Number of unstable triangles out of 10 are 3
Number of stable triangles out of 10 are 10
Number of unstable triangles out of 10 are 0
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_5_(Positive and negative relationships)/Code$
```

This is the output you see initially the number of unstable triangles was 4 and then it became 3 and then it became 0. So, this was very small network. So it happened it converged quite quickly.

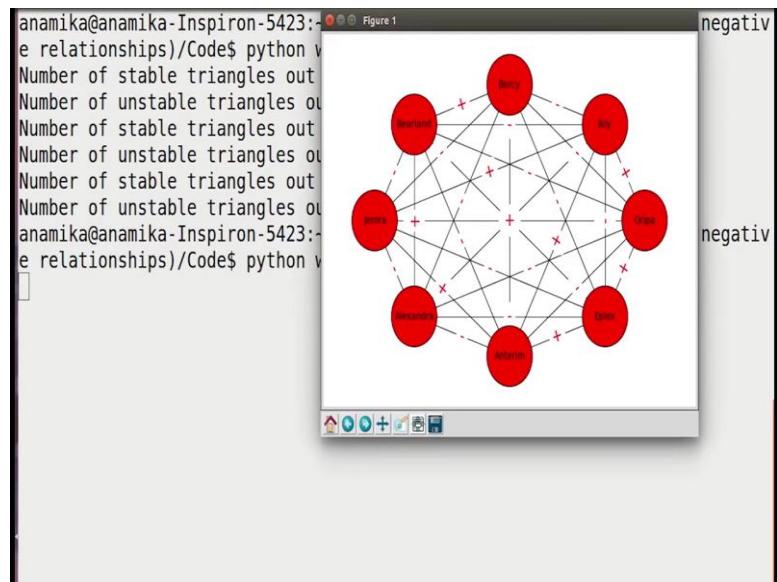
(Refer Slide Time: 21:47)



```
61     if r == 1:
62         G[tris_list[index][0]][tris_list[index][1]]['sign'] =
63     elif r == 2:
64         G[tris_list[index][1]][tris_list[index][2]]['sign'] =
65     elif r == 3:
66         G[tris_list[index][2]][tris_list[index][0]]['sign'] =
67
68     return G
69
70
71
72
73
74 # 1. Create a graph with 'n' nodes, where the nodes are the countries
75 G = nx.Graph()
76 n = 8
77 G.add_nodes_from([i for i in range(1, n+1)])
78 mapping = {1:'Alexandra', 2:'Anterim', 3:'Bercy', 4: 'Bearland', 5:
79 G = nx.relabel_nodes(G, mapping)
80
```

Let us for example, increase this number of countries to say 8 and see how it works.

(Refer Slide Time: 21:53)



So, this is the network that gets created out of 8 countries and these are the edges I am closing it.

(Refer Slide Time: 22:05)

```
Number of stable triangles out of 10 are 7
Number of unstable triangles out of 10 are 3
Number of stable triangles out of 10 are 10
Number of unstable triangles out of 10 are 0
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_5_(Positive and negative relationships)/Code$ python week5.py
Number of stable triangles out of 56 are 30
Number of unstable triangles out of 56 are 26
Number of stable triangles out of 56 are 28
Number of unstable triangles out of 56 are 28
Number of stable triangles out of 56 are 32
Number of unstable triangles out of 56 are 24
Number of stable triangles out of 56 are 30
Number of unstable triangles out of 56 are 26
Number of stable triangles out of 56 are 28
Number of unstable triangles out of 56 are 28
Number of stable triangles out of 56 are 26
Number of unstable triangles out of 56 are 30
Number of stable triangles out of 56 are 28
Number of unstable triangles out of 56 are 28
Number of stable triangles out of 56 are 32
```

Here you see; you go up the number of unstable triangles was the 26 then it became 28 you see. After converting 1 unstable triangle to a stable one; the number of unstable triangles increased to 28 then it reduced then it increased, increased, increased and then it decreased. So, as you can see it kept fluctuating.

(Refer Slide Time: 22:29)

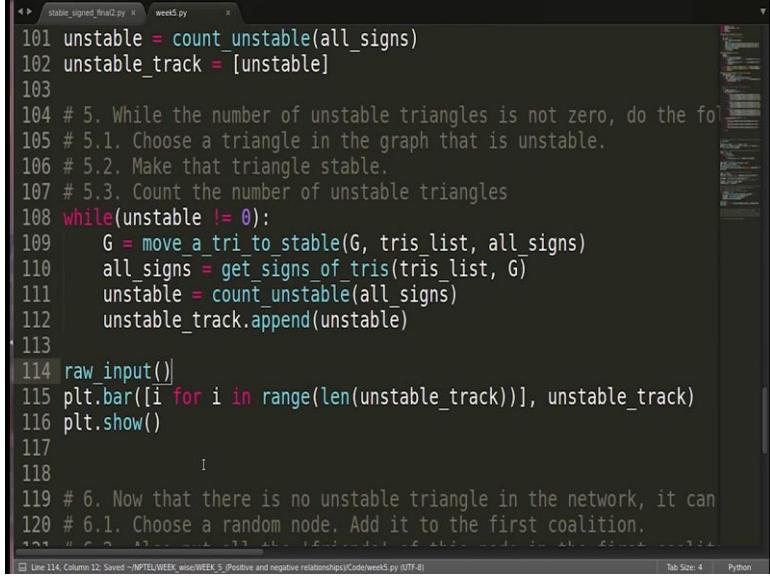
```
Number of unstable triangles out of 56 are 22
Number of stable triangles out of 56 are 34
Number of unstable triangles out of 56 are 22
Number of stable triangles out of 56 are 32
Number of unstable triangles out of 56 are 24
Number of stable triangles out of 56 are 36
Number of unstable triangles out of 56 are 20
Number of stable triangles out of 56 are 40
Number of unstable triangles out of 56 are 16
Number of stable triangles out of 56 are 40
Number of unstable triangles out of 56 are 16
Number of stable triangles out of 56 are 44
Number of unstable triangles out of 56 are 12
Number of stable triangles out of 56 are 46
Number of unstable triangles out of 56 are 10
Number of stable triangles out of 56 are 501
Number of unstable triangles out of 56 are 6
Number of stable triangles out of 56 are 56
Number of unstable triangles out of 56 are 0
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_5_(Positive and negative relationships)/Code$ █
```

So, it keeps fluctuating, and, in the end, it became 10 and then quickly 0 and then quickly it became 0.

So, if you want you can also plot this let me show you quickly how we can do that. So, basically, we want to see how the number of unstable triangles is changing when we keep moving 1 unstable triangle to a stable state in each iteration. So, this is the count of the unstable triangles; let us create a list and keep storing the number of unstable triangles there. So, let me create a list here unstable track and initially let it contain the number of the initial number of unstable triangles.

And as and when in every iteration we are getting the number of unstable triangles we will keep appending this number unstable to this list alright. So, we will write unstable track sorry unstable track.append unstable. So, this unstable track is a list which has all the number of unstable triangles with respect to each iteration and if you want to plot you can plot that as well.

(Refer Slide Time: 23:46)



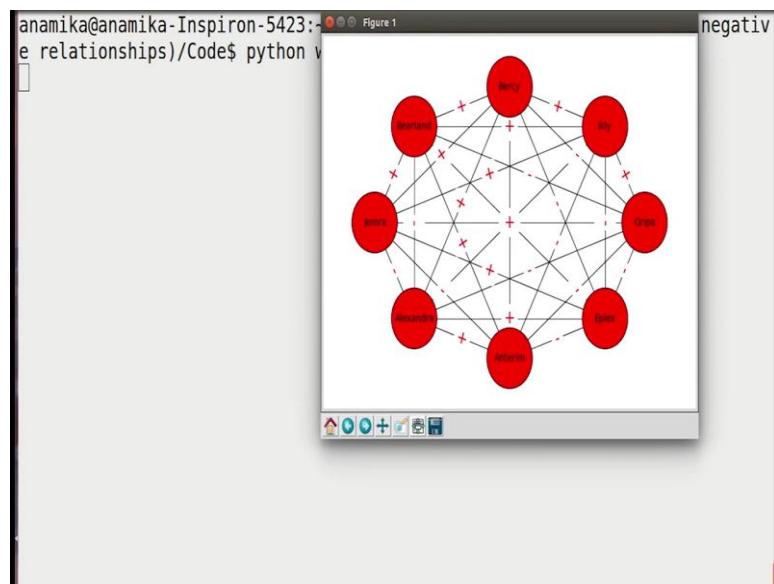
```
101 unstable = count_unstable(all_signs)
102 unstable_track = [unstable]
103
104 # 5. While the number of unstable triangles is not zero, do the fo
105 # 5.1. Choose a triangle in the graph that is unstable.
106 # 5.2. Make that triangle stable.
107 # 5.3. Count the number of unstable triangles
108 while(unstable != 0):
109     G = move_a_tri_to_stable(G, tris_list, all_signs)
110     all_signs = get_signs_of_tris(tris_list, G)
111     unstable = count_unstable(all_signs)
112     unstable_track.append(unstable)
113
114 raw_input()
115 plt.bar([i for i in range(len(unstable_track))], unstable_track)
116 plt.show()
117
118
119 # 6. Now that there is no unstable triangle in the network, it can
120 # 6.1. Choose a random node. Add it to the first coalition.
121
```

So, we can do plt.plot if you want to bar chart you can do that also.

So, the function is plt.bar and it requires 2 inputs the first will be the values on it on the x axis. So, we can have the values to start from 0 on the x axis. So, maybe we can pass the list here I can write I for I in range we can just pass the length of this list here so that the number of parameters match. So, this was our x axis and the y axis will of course be unstable track and then we can show it plt.show.

So, we can check whether it works before that we can use this function. So, that it us for an enter before it precedes. So, let us check the function we can.

(Refer Slide Time: 24:42)



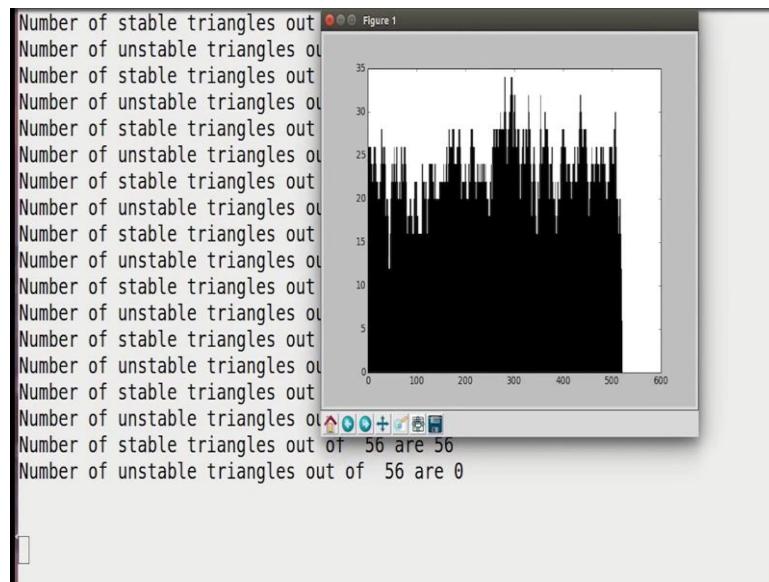
So, this is the network.

(Refer Slide Time: 24:47)

```
Number of stable triangles out of 56 are 36
Number of unstable triangles out of 56 are 20
Number of stable triangles out of 56 are 40
Number of unstable triangles out of 56 are 16
Number of stable triangles out of 56 are 40
Number of unstable triangles out of 56 are 16
Number of stable triangles out of 56 are 38
Number of unstable triangles out of 56 are 18
Number of stable triangles out of 56 are 36
Number of unstable triangles out of 56 are 20
Number of stable triangles out of 56 are 36
Number of unstable triangles out of 56 are 20
Number of stable triangles out of 56 are 40
Number of unstable triangles out of 56 are 16
Number of stable triangles out of 56 are 44
Number of unstable triangles out of 56 are 12
Number of stable triangles out of 56 are 50
Number of unstable triangles out of 56 are 6
Number of stable triangles out of 56 are 56
Number of unstable triangles out of 56 are 0
```

Let us close it and this is the how the number of unstable triangles changed.

(Refer Slide Time: 24:53)



And I am just pressing enter here and this is the plot that we got. So, as you can see the fluctuations that happened in the number of unstable triangles; I have not written anything on the x and y axis, but you can understand the y axis means the number of unstable triangles.

So, it reached here somewhere here and then quickly came to 0. So, that is how it happens let me close it now. Let us go back.

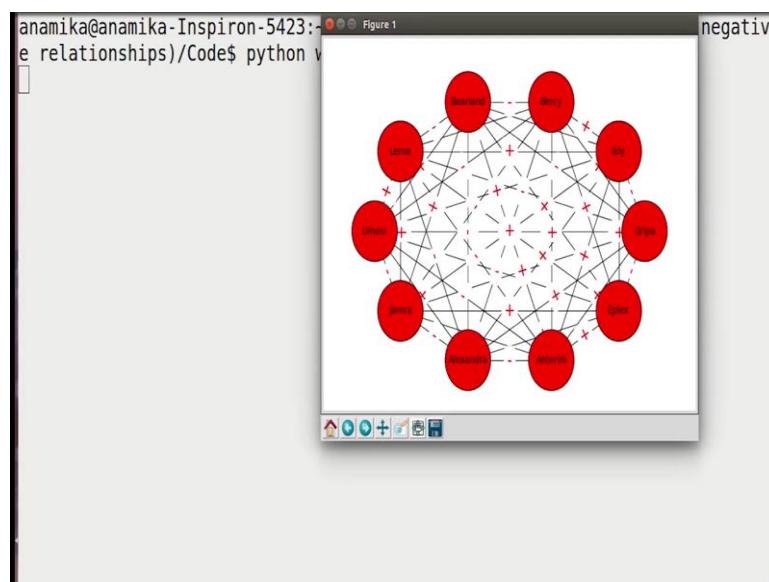
(Refer Slide Time: 25:21)

```
103
104 # 5. While the number of unstable triangles is not zero, do the following:
105 # 5.1. Choose a triangle in the graph that is unstable.
106 # 5.2. Make that triangle stable.
107 # 5.3. Count the number of unstable triangles
108 while(unstable != 0):
109     G = move_a_tri_to_stable(G, tris_list, all_signs)
110     all_signs = get_signs_of_tris(tris_list, G)
111     print all_signs
112     unstable = count_unstable(all_signs)
113     unstable_track.append(unstable)
114
115 # raw_input()
116 # plt.bar([i for i in range(len(unstable_track))], unstable_track)
117 # plt.show()
118
119
120 # 6. Now that there is no unstable triangle in the network, it can
121 # 6.1. Choose a random node. Add it to the first coalition.
122 # 6.2. Also put all the 'friends' of this node in the first coalition.
```

Now, if you want you can also keep printing this list which is all signed so that you can see how the signs of the triangles are changing. So, let me quickly show you that I am printing all signs. Since we have already seen the plot. I am just commenting this part and let me increase the number of cities to say 10, alright.

So, basically, we want to see the signs how they are changing from the initial configuration through the last configuration where all the triangles are stable.

(Refer Slide Time: 25:56)



So, let us check this is the network with ten countries. So, it is taking some time as you keep increasing the countries it takes a whole lot of time from 8 to 10; it is taking a quite a lot of time; it is taking some time.

So, as you can see this is the list, which is all signs and once it stops, I will show you some examples here; it is actually taking a lot of time ok. I think I can pause the video and show you the final result ok; I will show you the final result now.

(Refer Slide Time: 26:46)

You see this is the final result: in the end out of one twenty triangles we in the end we have all the stable triangles, and if you go back I am just going a little up.

(Refer Slide Time: 26:59)

Here, you can see a number of unstable triangles for example, here there are 2 positives. So, this is one of the unstable triangles and there are 0 positives. Here this is another unstable triangle. However, over time it keeps evolving and in the end whatever you get. So, this all list is completely stable there, there is no triangle which has 0 or 2 positive signs. So, this is how it looks like we can go back here.

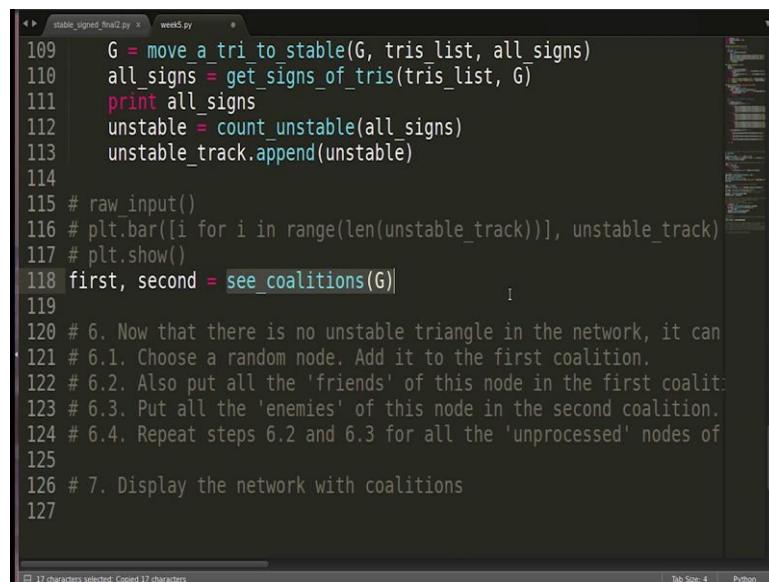
So, we done with evolving the network from us from an unstable state to a stable state As I told you once a network becomes stable it can be divided into two groups two coalitions I am not calling them communities for a reason because in we did community division in one of the previous videos. So, you should not get confused with that algorithm and this method, because community is basically; community detection is completely different algorithm depending on the number of links. Here the network is complete, here we are concerned with the signs of the edges, so that is why we are not calling them division in two committees recalling them division to coalitions or groups.

So, in the next video we are going to divide this network into two groups given that it has become stable now.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Homophily (Continued) & Positive and Negative Relationships**  
**Lecture – 72**  
**Forming two coalitions**

(Refer Slide Time: 00:05)

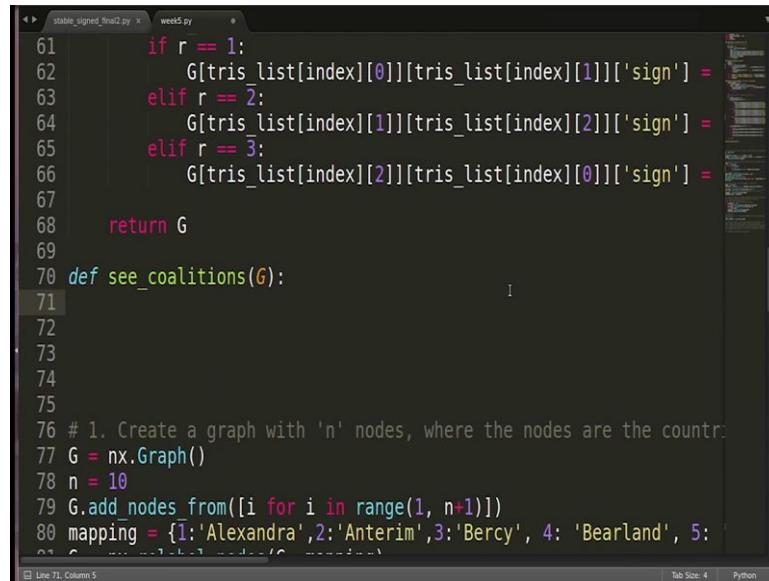


```
stable_signed_final2.py  week5.py
109 G = move_a_tri_to_stable(G, tris_list, all_signs)
110 all_signs = get_signs_of_tris(tris_list, G)
111 print all_signs
112 unstable = count_unstable(all_signs)
113 unstable_track.append(unstable)
114
115 # raw_input()
116 # plt.bar([i for i in range(len(unstable_track))], unstable_track)
117 # plt.show()
118 first, second = see_coalitions(G)
119
120 # 6. Now that there is no unstable triangle in the network, it can
121 # 6.1. Choose a random node. Add it to the first coalition.
122 # 6.2. Also put all the 'friends' of this node in the first coalition.
123 # 6.3. Put all the 'enemies' of this node in the second coalition.
124 # 6.4. Repeat steps 6.2 and 6.3 for all the 'unprocessed' nodes of
125
126 # 7. Display the network with coalitions
127
```

In the previous video we moved the network from an unstable state to a stable state, where there are no unstable triangles. So, once that happens it is possible to divide the nodes of the network into two groups. So, that is what we are going to do here now.

Let us create a function for that and let the function return to lists, where the first list will contain the nodes of the first row and second list will contain the nodes of the second row. So, let us create this function and let us call it see coalitions and let us pass the graph here.

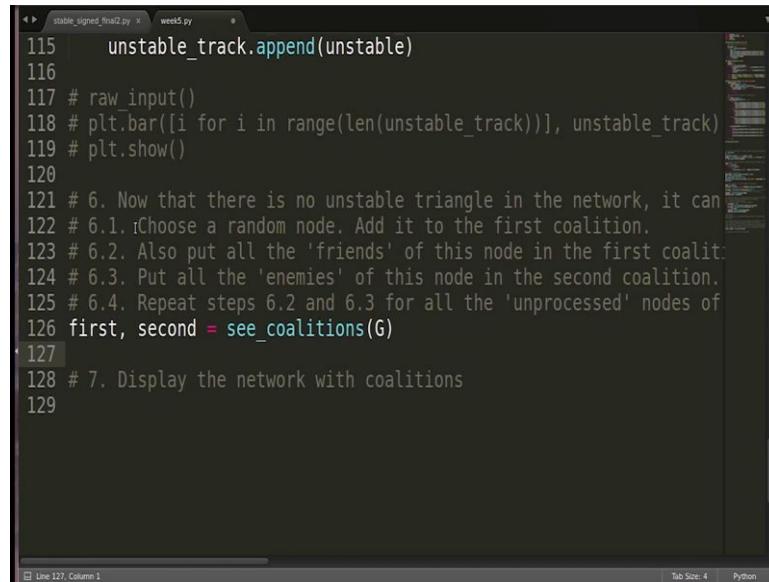
(Refer Slide Time: 00:53)



```
stable_signed_final2.py week5.py
61     if r == 1:
62         G[tris_list[index][0]][tris_list[index][1]]['sign'] =
63     elif r == 2:
64         G[tris_list[index][1]][tris_list[index][2]]['sign'] =
65     elif r == 3:
66         G[tris_list[index][2]][tris_list[index][0]]['sign'] =
67
68     return G
69
70 def see_coalitions(G):
71
72
73
74
75
76 # 1. Create a graph with 'n' nodes, where the nodes are the countries
77 G = nx.Graph()
78 n = 10
79 G.add_nodes_from([i for i in range(1, n+1)])
80 mapping = {1:'Alexandra', 2:'Anterim', 3:'Bercy', 4: 'Bearland', 5:
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115     unstable_track.append(unstable)
116
117 # raw_input()
118 # plt.bar([i for i in range(len(unstable_track))], unstable_track)
119 # plt.show()
120
121 # 6. Now that there is no unstable triangle in the network, it can
122 # 6.1. Choose a random node. Add it to the first coalition.
123 # 6.2. Also put all the 'friends' of this node in the first coalition.
124 # 6.3. Put all the 'enemies' of this node in the second coalition.
125 # 6.4. Repeat steps 6.2 and 6.3 for all the 'unprocessed' nodes of
126 first, second = see_coalitions(G)
127
128 # 7. Display the network with coalitions
129
```

So, let us create this function here let us go up and we will create this function. So, what do we have to do here? As I told you previously the strategy that we are going to follow as we explained here with me.

(Refer Slide Time: 01:04)

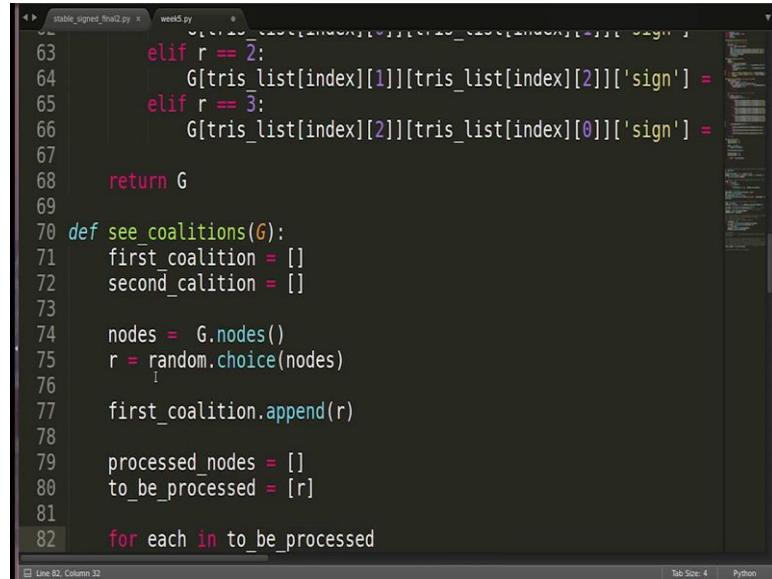


```
stable_signed_final2.py week5.py
115     unstable_track.append(unstable)
116
117 # raw_input()
118 # plt.bar([i for i in range(len(unstable_track))], unstable_track)
119 # plt.show()
120
121 # 6. Now that there is no unstable triangle in the network, it can
122 # 6.1. Choose a random node. Add it to the first coalition.
123 # 6.2. Also put all the 'friends' of this node in the first coalition.
124 # 6.3. Put all the 'enemies' of this node in the second coalition.
125 # 6.4. Repeat steps 6.2 and 6.3 for all the 'unprocessed' nodes of
126 first, second = see_coalitions(G)
127
128 # 7. Display the network with coalitions
129
```

Let me write it here because this is the sixth step the second last step.

So, we are going to choose a random node and we will add it to the first coalition.

(Refer Slide Time: 01:18)



```
stable_signed_final2.py  week5.py
63     elif r == 2:
64         G[tris_list[index][1]][tris_list[index][2]]['sign'] =
65     elif r == 3:
66         G[tris_list[index][2]][tris_list[index][0]]['sign'] =
67
68     return G
69
70 def see_coalitions(G):
71     first_coalition = []
72     second_coalition = []
73
74     nodes = G.nodes()
75     r = random.choice(nodes)
76
77     first_coalition.append(r)
78
79     processed_nodes = []
80     to_be_processed = [r]
81
82     for each in to_be_processed
```

Now, how do we choose a random node? Very simple, we will first access the list of nodes; nodes is equal to G.nodes. So, we got a list of nodes now what we have to do is we have to choose one node randomly out of it. The function that we can use for that is ran int that we have previously used as well or we can simply use the function random.choice which gives us one entry out of a given list. So, we have this list as nodes. So, we can use any of the functions.

Let us use the function choice r is the random node. So, we will write random.choice; the choice should be out of this list nodes. So, we got a random node here now since we have return also and we have to keep track also let us create two lists for each coalition, first coalition empty list and similarly second coalition another entry list. So, we are going to add nodes to this coalition.

Now, as I told you we will choose a random node and we will add it to the first coalition. So, we will write first I am sorry first coalition.append r right now basically what we have to do is; we have to run a sort of a (Refer Time: 02:49) first search on this node we can added condition that all the enemies of this node are not going to be a part of the bfs tree right.

So, as I told you we will look at the neighbours of this node and some of these neighbours will be positive some of these neighbours will be its friends and some of its neighbours will be its enemies. So, the neighbours which are its friends are going to be a

part of the bfs tree, they will be explored further. However, the nodes which are going to be which are its enemies are not going to be part of the bfs tree they will not be explored they will simply be put in the second coalition list. So, that is the strategy you can use any other strategy as well. So, this is one simple method that works well. So, as we do in bfs also we keep a track of the nodes which have been processed already the nodes which are yet to be processed. So, we are going in the same fashion.

So, let us create for that purpose two lists processed nodes initialized to empty to be processed same. So, this list will keep track of all the nodes which are to be processed. Now since r is the node which has to be processed, we will put r into it ok. Now we will we have to keep running the loop of bfs until this to be processed list becomes empty right you will write for each in to be processed.

(Refer Slide Time: 04:32)

```

74     nodes = G.nodes()
75     r = random.choice(nodes)
76
77     first_coalition.append(r)
78
79     processed_nodes = []
80     to_be_processed = [r]
81
82     for each in to_be_processed:
83         if each not in processed_nodes:
84             neigh = G.neighbors(each)
85
86             for i in range(len(neigh)):
87                 if G[each][neigh[i]]['sign'] == '+':
88                     if neigh[i] not in first_coalition:
89                         first_coalition.append(neigh[i])
90                     if neigh[i] not in to_be_processed:
91                         to_be_processed.append(neigh[i])
92
93

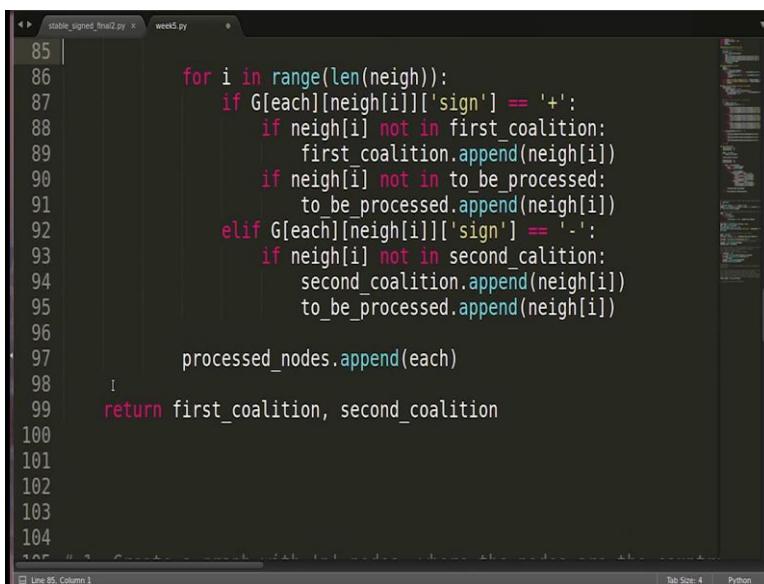
```

Now, in case the node is an enemy what has to be done? It has to be added to the second coalition and it does not have to be processed. So, we will write elif let me copy paste if it is a negative sign, we will add it to second coalition if it is not already there. So, we will check if `neigh[i] not in second_coalition` we will add it there. So, we will write `second_coalition.append(neigh[i])` that is one thing. Secondly, it does not have to be process. So, we are going to add it to be processed list, we will write to be processed.append(`neigh[i]`) right, ok.

Now, by this time r node that is each which we started with has been completely processed. So, we will write here, we will write since it does that it has been processed we will write processed\_nodes.append each. So, whatever we just processed, we will added to this list. So, that we have done.

So, this was the method to divide into first\_coalition and second\_coalition now this function must return these two lists.

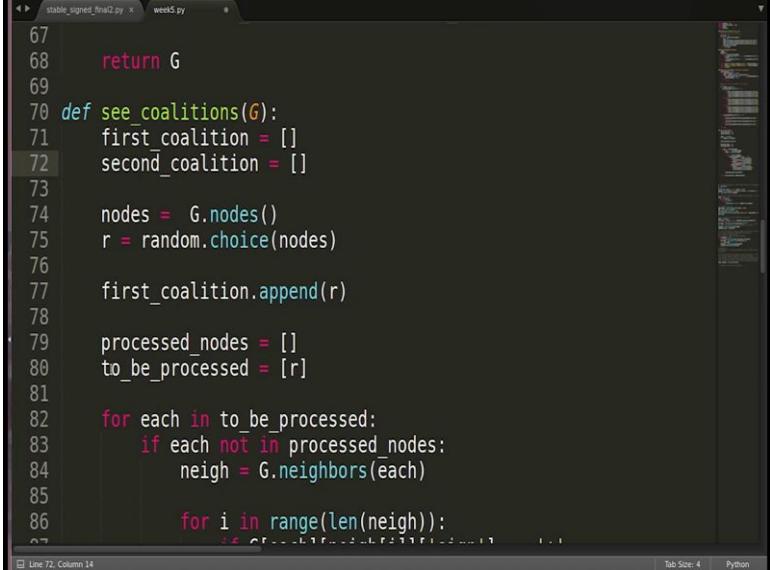
(Refer Slide Time: 05:57)



```
85 |     for i in range(len(neigh)):
86 |         if G[each][neigh[i]]['sign'] == '+':
87 |             if neigh[i] not in first_coalition:
88 |                 first_coalition.append(neigh[i])
89 |             if neigh[i] not in to_be_processed:
90 |                 to_be_processed.append(neigh[i])
91 |             elif G[each][neigh[i]]['sign'] == '-':
92 |                 if neigh[i] not in second_coalition:
93 |                     second_coalition.append(neigh[i])
94 |                     to_be_processed.append(neigh[i])
95 |
96 |     processed_nodes.append(each)
97 |
98 |     return first_coalition, second_coalition
99 |
100|
101|
102|
103|
104|
```

So, we will make it return here ok. We will write return first coalition second coalition if the spelling is wrong is it; oh, I am sorry second coalition its wrong everywhere I am sorry where else here ok.

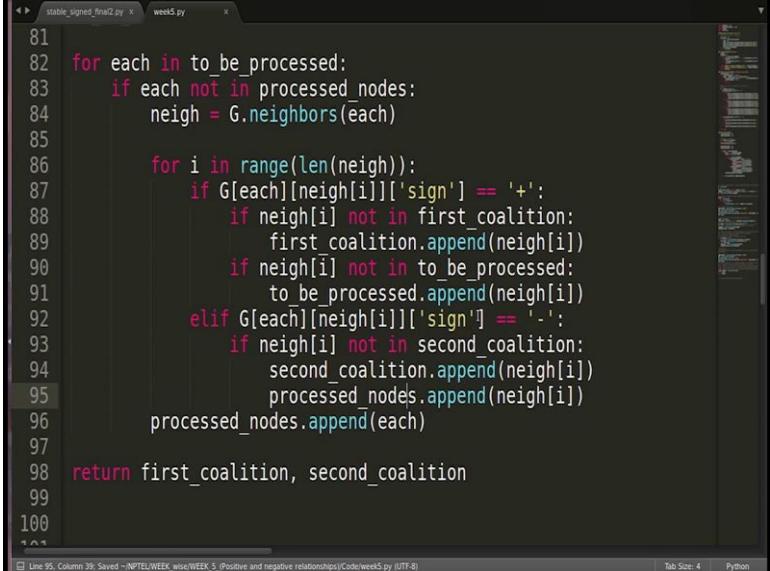
(Refer Slide Time: 06:23)



```
67     return G
68
69
70 def see_coalitions(G):
71     first_coalition = []
72     second_coalition = []
73
74     nodes = G.nodes()
75     r = random.choice(nodes)
76
77     first_coalition.append(r)
78
79     processed_nodes = []
80     to_be_processed = [r]
81
82     for each in to_be_processed:
83         if each not in processed_nodes:
84             neigh = G.neighbors(each)
85
86             for i in range(len(neigh)):
```

I think we are done (Refer Time: 06:28).

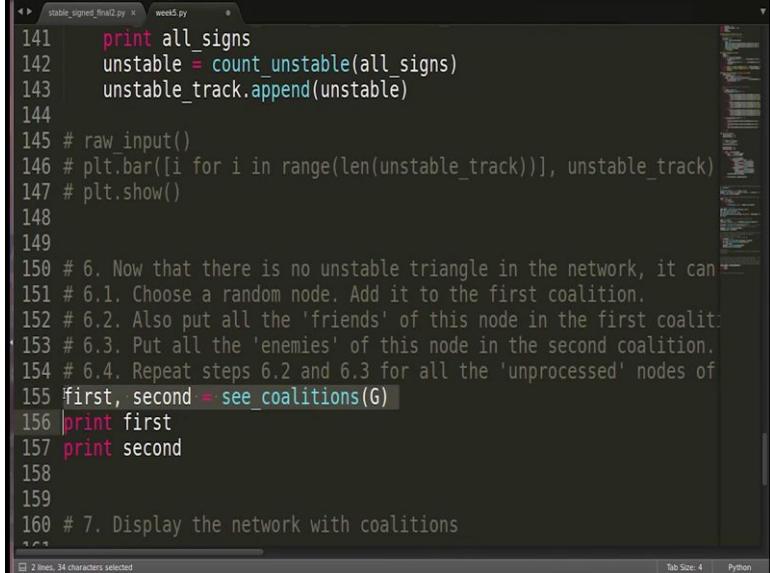
(Refer Slide Time: 06:28)



```
81     for each in to_be_processed:
82         if each not in processed_nodes:
83             neigh = G.neighbors(each)
84
85             for i in range(len(neigh)):
86                 if G[each][neigh[i]]['sign'] == '+':
87                     if neigh[i] not in first_coalition:
88                         first_coalition.append(neigh[i])
89                     if neigh[i] not in to_be_processed:
90                         to_be_processed.append(neigh[i])
91                 elif G[each][neigh[i]]['sign'] == '-':
92                     if neigh[i] not in second_coalition:
93                         second_coalition.append(neigh[i])
94                         processed_nodes.append(neigh[i])
95
96     processed_nodes.append(each)
97
98 return first_coalition, second_coalition
99
100
```

We have added the friends of a given node to the first coalition and the enemies of the given node to the second coalition, and we are not processing the nodes which are going to go in the second coalition. Here, I am sorry we do not need to process the nodes which are added to the second coalition. So, they will go to processed nodes here, because they are not going to be processed again ok.

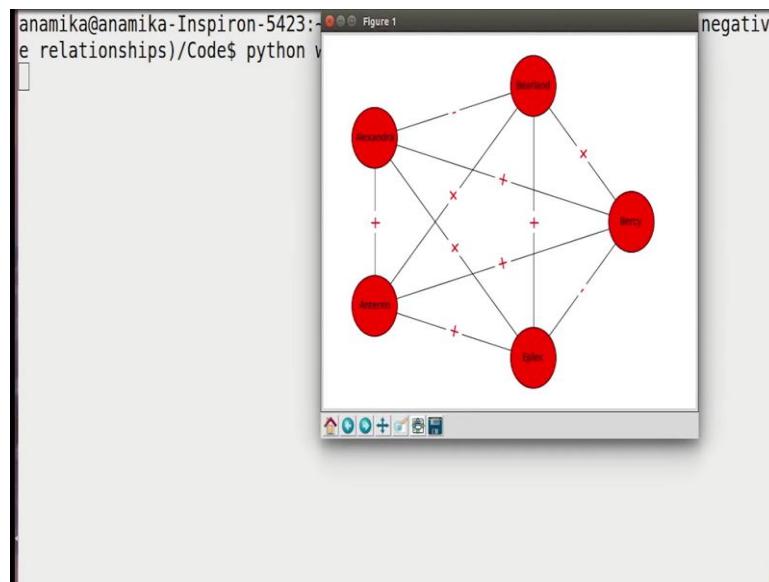
(Refer Slide Time: 07:00)



```
141 print all_signs
142 unstable = count_unstable(all_signs)
143 unstable_track.append(unstable)
144
145 # raw_input()
146 # plt.bar([i for i in range(len(unstable_track))], unstable_track)
147 # plt.show()
148
149
150 # 6. Now that there is no unstable triangle in the network, it can
151 # 6.1. Choose a random node. Add it to the first coalition.
152 # 6.2. Also put all the 'friends' of this node in the first coalition.
153 # 6.3. Put all the 'enemies' of this node in the second coalition.
154 # 6.4. Repeat steps 6.2 and 6.3 for all the 'unprocessed' nodes of
155 first, second = see_coalitions(G)
156 print first
157 print second
158
159
160 # 7. Display the network with coalitions
```

So, this is the function and let us go back here and see if it works well. So, we call this function see coalitions and its returning the two lists of nodes first and second and then we are printing these two lists.

(Refer Slide Time: 07:15)



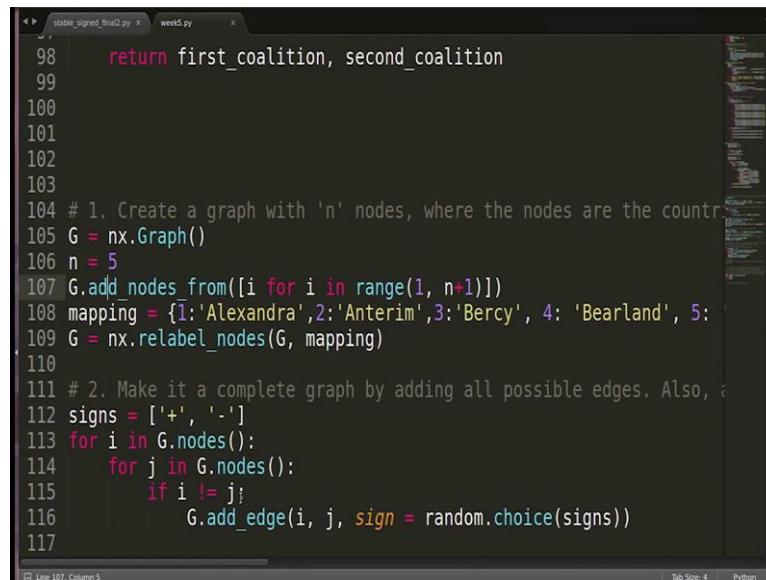
Let us see how it functions. So, let us run this, this is the nodes this is the networks with five countries I am closing it. So, that it can go ahead.

(Refer Slide Time: 07:30)

```
Number of unstable triangles out of 10 are 5
[['+', '-', '+'], ['+', '+', '+'], ['+', '+', '+'], ['+', '+', '+'], ['+', '+', '-'],
['+', '+', '+'], ['+', '+', '+'], ['-', '+', '+'], ['-', '+', '+'], ['+', '+', '+'],
['+', '+', '-']]
Number of stable triangles out of 10 are 6
Number of unstable triangles out of 10 are 4
[['+', '+', '+'], ['+', '+', '+'], ['+', '+', '+'], ['+', '+', '+'], ['+', '+', '-'],
['+', '+', '+'], ['+', '+', '+'], ['+', '+', '+'], ['+', '+', '+'], ['+', '+', '+'],
['+', '+', '-']]
Number of stable triangles out of 10 are 7
Number of unstable triangles out of 10 are 3
[['+', '+', '+'], ['+', '+', '+'], ['+', '+', '+'], ['+', '+', '+'], ['+', '+', '+'],
['+', '+', '+'], ['+', '+', '+'], ['+', '+', '+'], ['+', '+', '+'], ['+', '+', '+'],
['+', '+', '+']]
Number of stable triangles out of 10 are 10
Number of unstable triangles out of 10 are 0
Eplex is random
['Eplex', 'Alexandra', 'Bercy', 'Anterim', 'Bearland']
[]
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_5_(Positive and negative relationships)/Code$
```

Let me show you one some more country.

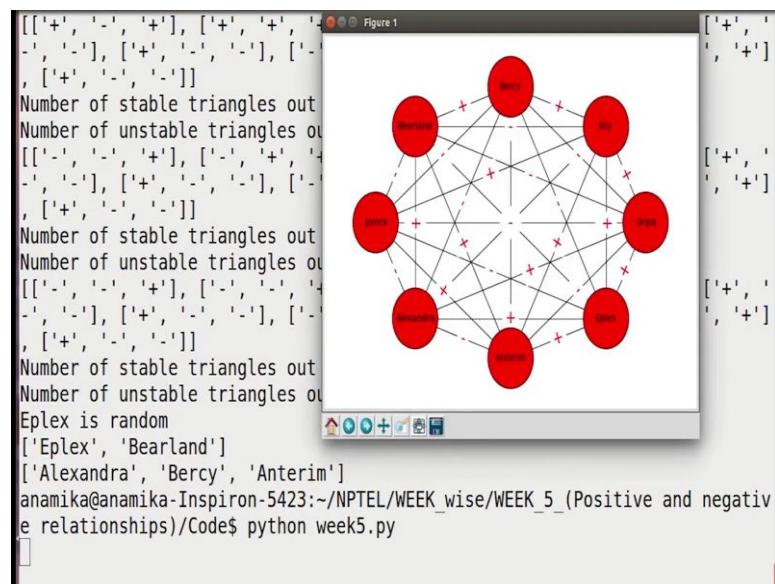
(Refer Slide Time: 07:34)



```
98     return first_coalition, second_coalition
99
100
101
102
103
104 # 1. Create a graph with 'n' nodes, where the nodes are the countries
105 G = nx.Graph()
106 n = 5
107 G.add_nodes_from([i for i in range(1, n+1)])
108 mapping = {1:'Alexandra', 2:'Anterim', 3:'Bercy', 4:'Bearland', 5:'Eplex'}
109 G = nx.relabel_nodes(G, mapping)
110
111 # 2. Make it a complete graph by adding all possible edges. Also, assign
112 signs = ['+', '-']
113 for i in G.nodes():
114     for j in G.nodes():
115         if i != j:
116             G.add_edge(i, j, sign = random.choice(signs))
117
```

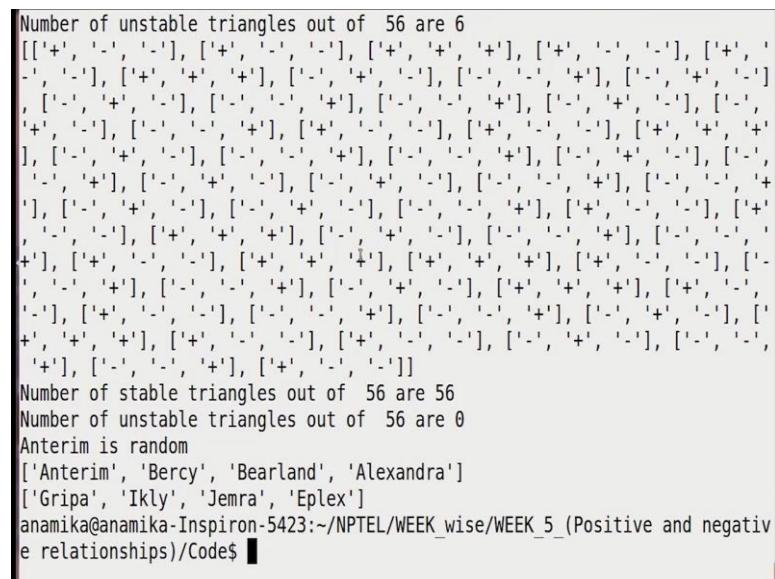
So, that we are able to nicely see the division into two groups: let me increase this to 8 and in c.

(Refer Slide Time: 07:44)



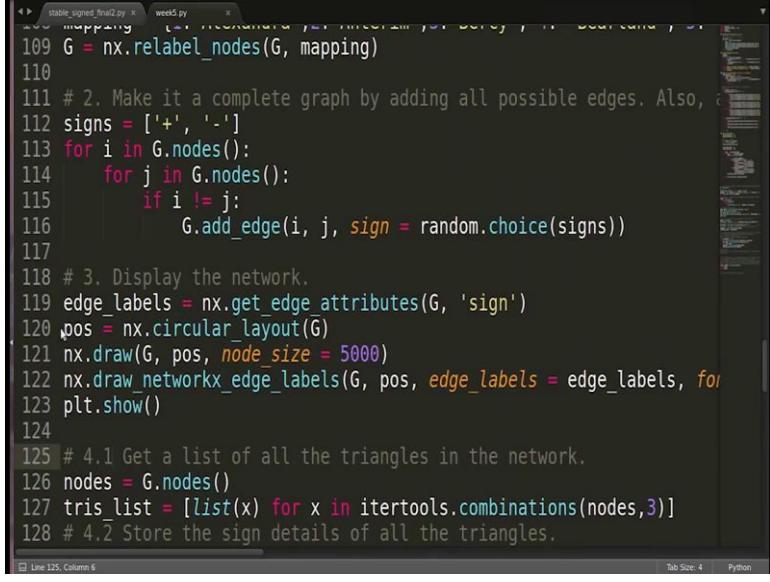
So, these are the 8 countries let me close this.

(Refer Slide Time: 07:46)



So, it moves the triangle to stable state and then it divided into two groups. As you can see out of the 8 countries four are in first group and the rest four are in the second group if we want to visualize as we can see.

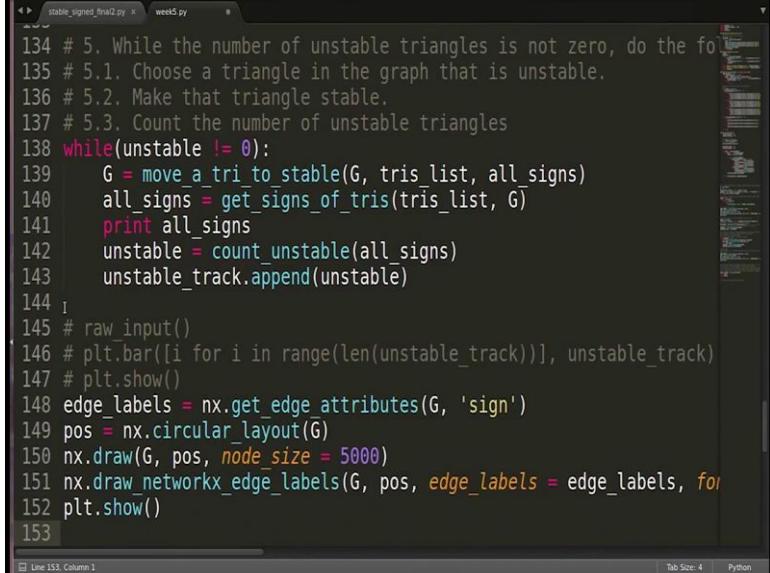
(Refer Slide Time: 08:08)



```
109 G = nx.relabel_nodes(G, mapping)
110
111 # 2. Make it a complete graph by adding all possible edges. Also,
112 signs = ['+', '-']
113 for i in G.nodes():
114     for j in G.nodes():
115         if i != j:
116             G.add_edge(i, j, sign = random.choice(signs))
117
118 # 3. Display the network.
119 edge_labels = nx.get_edge_attributes(G, 'sign')
120 pos = nx.circular_layout(G)
121 nx.draw(G, pos, node_size = 5000)
122 nx.draw_networkx_edge_labels(G, pos, edge_labels = edge_labels, font_size = 10)
123 plt.show()
124
125 # 4.1 Get a list of all the triangles in the network.
126 nodes = G.nodes()
127 tris_list = [list(x) for x in itertools.combinations(nodes, 3)]
128 # 4.2 Store the sign details of all the triangles.
```

Let me add code of displaying the communities as well the coalitions ok. This is how we displayed the graph previously; I am just copying the code; so that we can display it once again.

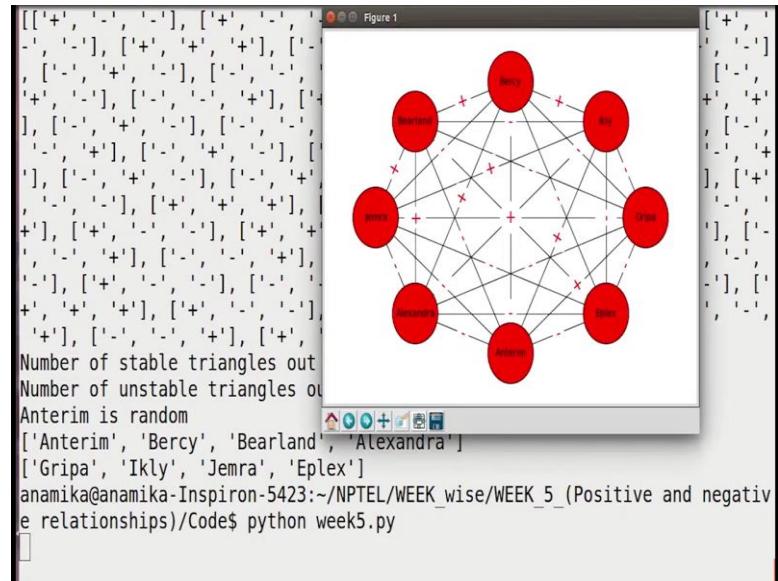
(Refer Slide Time: 08:21)



```
134 # 5. While the number of unstable triangles is not zero, do the following:
135 # 5.1. Choose a triangle in the graph that is unstable.
136 # 5.2. Make that triangle stable.
137 # 5.3. Count the number of unstable triangles
138 while(unstable != 0):
139     G = move_a_tri_to_stable(G, tris_list, all_signs)
140     all_signs = get_signs_of_tris(tris_list, G)
141     print all_signs
142     unstable = count_unstable(all_signs)
143     unstable_track.append(unstable)
144
145 # raw_input()
146 # plt.bar([i for i in range(len(unstable_track))], unstable_track)
147 # plt.show()
148 edge_labels = nx.get_edge_attributes(G, 'sign')
149 pos = nx.circular_layout(G)
150 nx.draw(G, pos, node_size = 5000)
151 nx.draw_networkx_edge_labels(G, pos, edge_labels = edge_labels, font_size = 10)
152 plt.show()
153
```

So, I am executing it again.

(Refer Slide Time: 08:24)



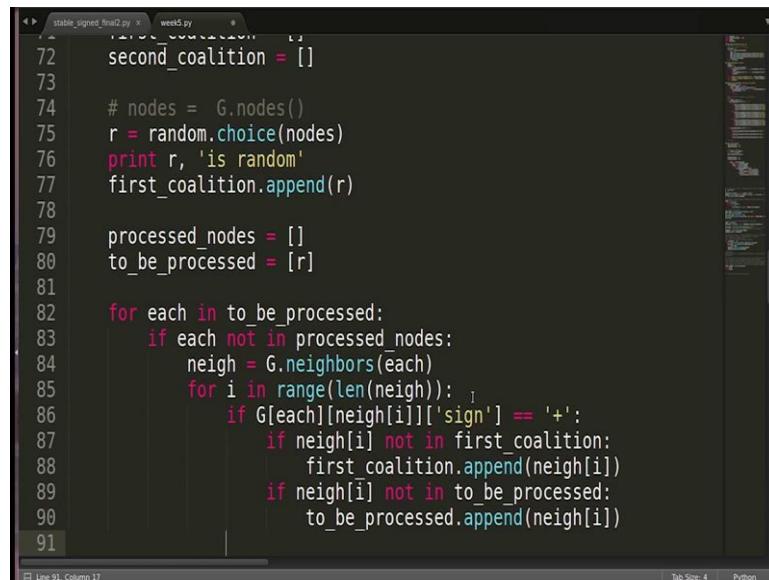
So, this is the initial network; let us take one example triangle. So, Bearland, Jenera and Bercy they have an unstable relation, see there are two positive relationships. So, basically Jenera and Bercy are enemies to each other, but they have a common friend which is Bearland. So, let us see what happens to this. I am closing it and its doing its thing. And yes, this is what we had to see. Bearland, Jenera and Bercy they all are friends now. So, over time what happens? Jenera and Bercy which were initially enemies they become friends to each other. So, you can check more examples as well. So, overall, this network is completely stable.

In the next video we are going to visualize these two groups that get formed.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Homophily (Continued) & Positive and Negative Relationships**  
**Lecture – 73**  
**Forming two coalitions (Continued)**

(Refer Slide Time: 00:05)



```
stable_signed_final2.py  week5.py
72     second_coalition = []
73
74     # nodes = G.nodes()
75     r = random.choice(nodes)
76     print r, 'is random'
77     first_coalition.append(r)
78
79     processed_nodes = []
80     to_be_processed = [r]
81
82     for each in to_be_processed:
83         if each not in processed_nodes:
84             neigh = G.neighbors(each)
85             for i in range(len(neigh)):
86                 if G[each][neigh[i]]['sign'] == '+':
87                     if neigh[i] not in first_coalition:
88                         first_coalition.append(neigh[i])
89                     if neigh[i] not in to_be_processed:
90                         to_be_processed.append(neigh[i])
91
```

We will take nodes from this list to be processed and we will keep checking their neighbors. So, we will start a loop for each in to be processed, if each not in process node. So, if this node has not already been processed, if each not in processed node then we will look at his neighbors; so let us have this list `neigh[i]`s equal to `G` dot neighbors we have to look at the neighbors of each. So, we will write like this.

Now we have take these neighbors one by one and we have to see whether these neighbors are friends or enemies. If they are friends, they will be added to the first list, if they are enemies they will be added to the second list. Apart from that if they are friends and they are being added to the first list, they have to be processed afterwards. So, they will be added to the list to be processed.

And if they are enemies and they are being added to the second list, then they will not be processed further. So, they will be added to the list process nodes. So, we will write for `i` in `range`, length of this list because to get to know how many neighbors are there of

course, in this particular example there are  $n - 1$  precisely  $n - 1$  neighbors, but we can generalize this and we can write up to length of neighbors.

So, once we take the neighbor, we have to check the relationship of this neighbor with the node that is each. So, we will write  $G$  each to  $\text{neigh}[i]$  and how do we check the relation by using this attribute sign. So, we will start this if this is equal to plus which means it is a friend, as I told you what has to be done it has to be added to the list of first coalition.

So, if it is not already there. So, we will check if  $\text{neigh}[i]$  not in first coalition, then it will be added there. So, you write  $\text{first\_coalition.append(neigh[i])}$  done. And second thing that has to be done is it has to be added to the list of to be process nodes if it is not all be there. You will write if  $\text{neigh}[i]$  not in to be processed and it will appended there to be processed dot append  $\text{neigh}[i]$  alright

And in the other case when the neighbor is an enemy, what we are going to do is.

(Refer Slide Time: 02:55)

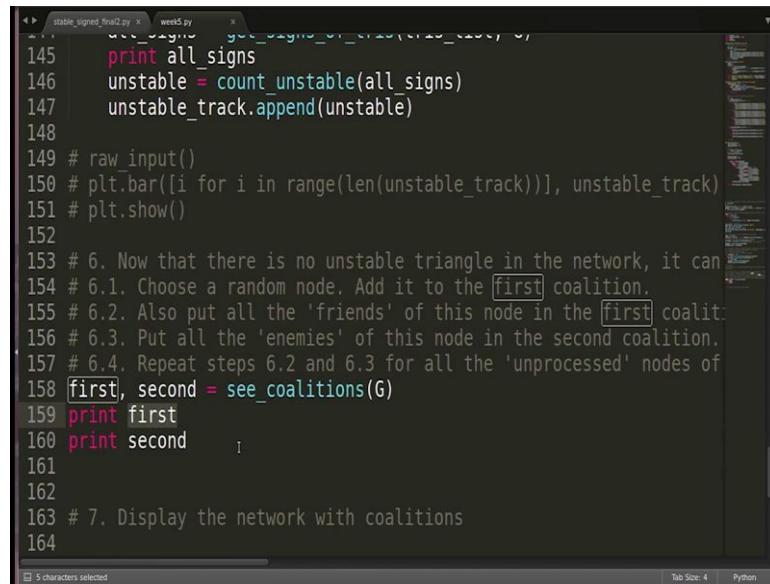
```
81
82     for each in to_be_processed:
83         if each not in processed_nodes:
84             neigh = G.neighbors(each)
85             for i in range(len(neigh)):
86                 if G[each][neigh[i]]['sign'] == '+':
87                     if neigh[i] not in first_coalition:
88                         first_coalition.append(neigh[i])
89                     if neigh[i] not in to_be_processed:
90                         to_be_processed.append(neigh[i])
91                 elif G[each][neigh[i]]['sign'] == '-':
92                     if neigh[i] not in second_coalition:
93                         second_coalition.append(neigh[i])
94                         processed_nodes.append(neigh[i])
95             processed_nodes.append(each)
96
97     return first_coalition, second_coalition
98
99
100
```

So, will write `elif` and this thing which means this is minus, so its an enemy. What has to be done? It has to be added to the second coalition list if it is not already there. So, we will write if  $\text{neigh}[i]$  not in  $\text{second\_coalition}$ , it will be appended  $\text{neigh}[i]$  ok. Second in that has to be done is as I told you it does not have to be processed again. So, it has to be added to the list of processed nodes.

So, we can directly just add it there, processed\_nodes.append(neigh[i]). Next thing that has to be done is this each the node which is called each has been process now. So, it has to be added to the list of process nodes. So, we will write here processed nodes dot append each ok. So, that is done

So, there this loop should basically divide the nodes in the two lists that is first coalition and second coalition and that is what this function will return. So, we are going to return this. So, we will write return first coalition, second coalition this should work.

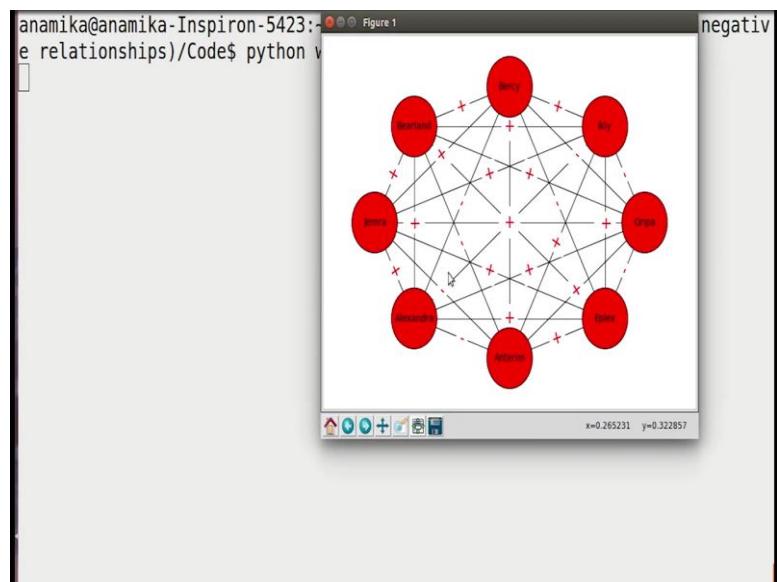
(Refer Slide Time: 04:25)



```
stable_signed_final2.py  week5.py
145     print all_signs
146     unstable = count_unstable(all_signs)
147     unstable_track.append(unstable)
148
149 # raw_input()
150 # plt.bar([i for i in range(len(unstable_track))], unstable_track)
151 # plt.show()
152
153 # 6. Now that there is no unstable triangle in the network, it can
154 # 6.1. Choose a random node. Add it to the [first] coalition.
155 # 6.2. Also put all the 'friends' of this node in the [first] coalit:
156 # 6.3. Put all the 'enemies' of this node in the second coalition.
157 # 6.4. Repeat steps 6.2 and 6.3 for all the 'unprocessed' nodes of
158 [first, second = see_coalitions(G)
159 print first
160 print second
161
162
163 # 7. Display the network with coalitions
164
```

Let us go down and see this is where we call this function and after it returned the two lists, we are just spending those two lists. So, let us see how it works ok.

(Refer Slide Time: 04:32)



These are the air cities and you see the relationships amongst them as well positive and negative relationships I am closing it.

(Refer Slide Time: 04:41)

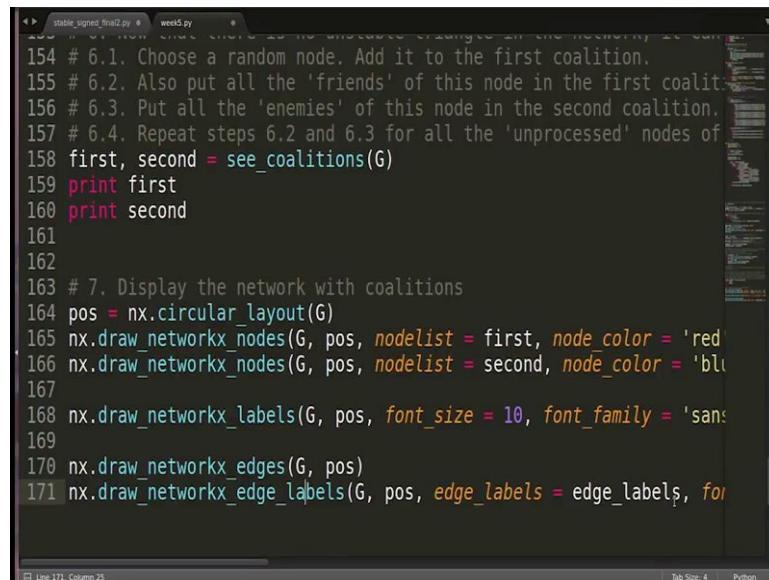
So, it has processed, and you see these are the two coming the two coalitions formed in the first one there are 5 countries and in the second one there are 3 countries. So, this is the division

In the next video we are going to visualize this network and we are going to compare it with the initial configuration that was entered initially.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Homophily (Continued) & Positive and Negative Relationships**  
**Lecture – 74**  
**Visualizing coalitions and the evolution**

(Refer Slide Time: 00:00)



```
154 # 6.1. Choose a random node. Add it to the first coalition.
155 # 6.2. Also put all the 'friends' of this node in the first coalition.
156 # 6.3. Put all the 'enemies' of this node in the second coalition.
157 # 6.4. Repeat steps 6.2 and 6.3 for all the 'unprocessed' nodes of
158 first, second = see_coalitions(G)
159 print first
160 print second
161
162
163 # 7. Display the network with coalitions
164 pos = nx.circular_layout(G)
165 nx.draw_networkx_nodes(G, pos, nodelist = first, node_color = 'red')
166 nx.draw_networkx_nodes(G, pos, nodelist = second, node_color = 'blue')
167
168 nx.draw_networkx_labels(G, pos, font_size = 10, font_family = 'sans-serif')
169
170 nx.draw_networkx_edges(G, pos)
171 nx.draw_networkx_edge_labels(G, pos, edge_labels = edge_labels, font_size = 10)
```

Of the implementation where we are going to visualize the communities that is the coalitions that we informed in the previous video. So, one thing that we want to do here is that we want to display the nodes in one coalition by a different color and the nodes in the second coalition by different color; so that we can clearly see the kinds of relationships amongst the nodes inside; the inside the coalition and across the coalition we want to visualize that.

So, what we can do is we can use the same layout circular layout; so we will write `nx.circular_layout G`. Now if we write `nx.draw`, it will just draw the network simply without any added features there. Since we want to display the nodes in a different in two different colors, we will have to use separate commands.

So, what we can use here is `nx.draw` in `networkx nodes`; in that function we will pass the lists list of nodes which have to be colored with the single color. So what we can write is

`nx.draw` networkx nodes the parameters will be first the graph and then the layout and the third parameter which is the most important is the node list.

So, this list will contain the nodes which have to be colored in one color. So, here we want the nodes which are there in the first list this list to be colored in the same colors. So, you will write node list is equal to first we can specify the color that you want node color is equal to say red. And we can also give the size node size is equal to say 5000; so, that was the displaying the nodes in the first list.

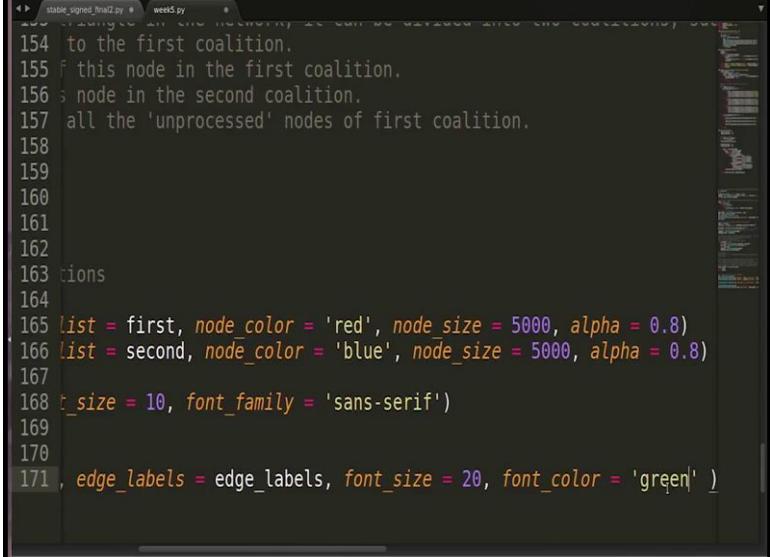
Similarly, we have to display the nodes in the second list we will write `nx.draw` networkx nodes `G` and then `pos` then node list is equal to second. Now node color is equal to say we will change the color here say blue and node size is equal to 1 second (Refer Time: 02:33) node size is equal to 5000, there is one more parameters which we can use that is alpha which basically gives the transparency value for the for the node color.

So, we can keep it and we can change it later if we require; so, as of now I am giving it 0.8. So, now we have drawn the nodes since we have not made use of `nx.draw` the rest of the things will not be drawn by default we have only; so we are manually doing it in a way. So, we have drawn the nodes we have not drawn the edges, we have not drawn the edge labels, we have not drawn the node label; so we are going to do that one by one ok.

So, let us first write the labels of the nodes, so we will write `nx.draw` networkx labels. So, again the parameters will be `G` and the layout and then the font size, font size is equal to 10. And then font we can give the font as well by this parameter font family is equal to say sans serif; I am sorry we will put hyphen here.

Now, we have drawn the nodes and the labels on the nodes; now next thing is the edges. So, we will write `nx.draw` networkx edges; again, the parameters will be `G` and `pos` you can give more parameters as well; as of now I think this is enough. Now we have not yet drawn the labels on the edges; so for that we will use `nx.draw` networkx edge labels; so this is the command, I think we used it at here as well. So, the parameters again will be `G` and then the layout and then edge labels is equal to edge labels; we are yet to retrieve the labels firstly, we are going to do that after writing this command.

(Refer Slide Time: 04:47)



A screenshot of a Python code editor showing a script named 'stable\_signed\_final2.py'. The code is part of a larger program and includes the following snippet:

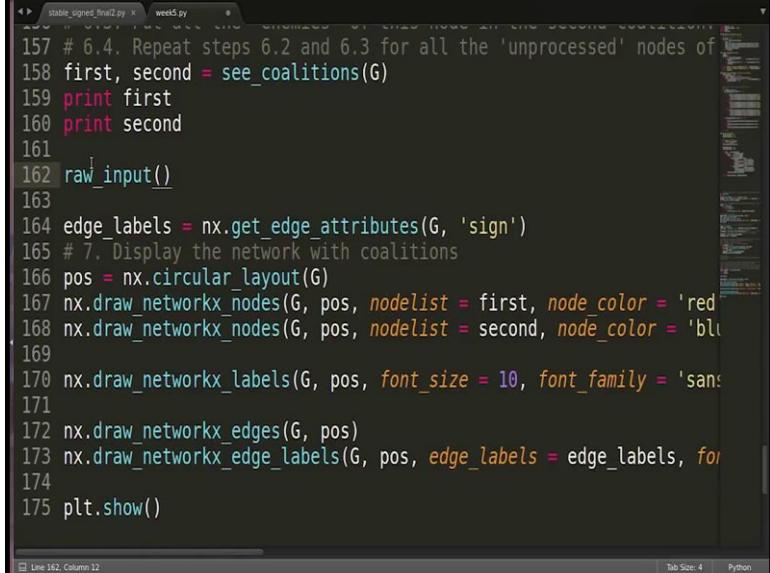
```
154 to the first coalition.
155 f this node in the first coalition.
156 s node in the second coalition.
157 all the 'unprocessed' nodes of first coalition.
158
159
160
161
162
163 tions
164
165 list = first, node_color = 'red', node_size = 5000, alpha = 0.8)
166 list = second, node_color = 'blue', node_size = 5000, alpha = 0.8)
167
168 t_size = 10, font_family = 'sans-serif')
169
170
171 , edge_labels = edge_labels, font_size = 20, font_color = 'green' )
```

The code uses the NetworkX library to draw a graph with red and blue nodes, and green edge labels.

So, as we did in the previous representation of the graph; we have to retrieve the labels. If you do not do that by default it displays the attribute name as well for example, sign is equal to plus, sign is equal to minus we do not want that, we only want to display plus and minus there. So, we are going to retrieve the edge labels in the next command and apart from that we can display the font size as well; font size is equal to say 20 and in font color is equal to say green maybe.

So, the only thing left is retrieving the edge labels ok; for that we will go up here, we will write this command edge labels is equal to nx.get\_edge\_attributes. And the column address will be the graph and the attribute for which we want to retrieve the values. So, this should fetch the latest attributes of sign for all the edges. So, I think we have drawn pretty much everything, and it should work.

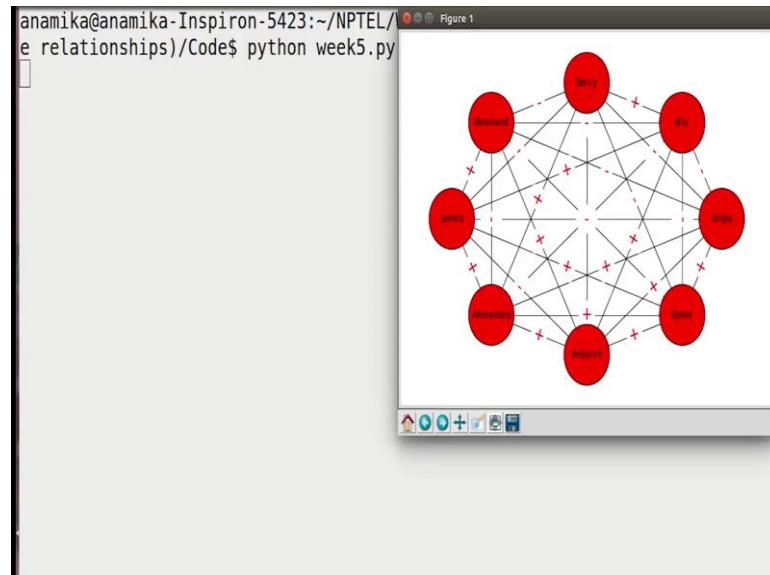
(Refer Slide Time: 05:56)



```
157 # 6.4. Repeat steps 6.2 and 6.3 for all the 'unprocessed' nodes of
158 first, second = see_coalitions(G)
159 print first
160 print second
161
162 raw_input()
163
164 edge_labels = nx.get_edge_attributes(G, 'sign')
165 # 7. Display the network with coalitions
166 pos = nx.circular_layout(G)
167 nx.draw_networkx_nodes(G, pos, nodelist = first, node_color = 'red')
168 nx.draw_networkx_nodes(G, pos, nodelist = second, node_color = 'blue')
169
170 nx.draw_networkx_labels(G, pos, font_size = 10, font_family = 'sans-serif')
171
172 nx.draw_networkx_edges(G, pos)
173 nx.draw_networkx_edge_labels(G, pos, edge_labels = edge_labels, font_size = 10)
174
175 plt.show()
```

So, we can write plt.show let me also add one more thing; let me add a function here draw input. Now what this function does is; it will ask you for an enter before it proceeds. So, we want it to pos for a few moments and then we will press enter and then will display the graph the network ok. Let us save it and we will check here.

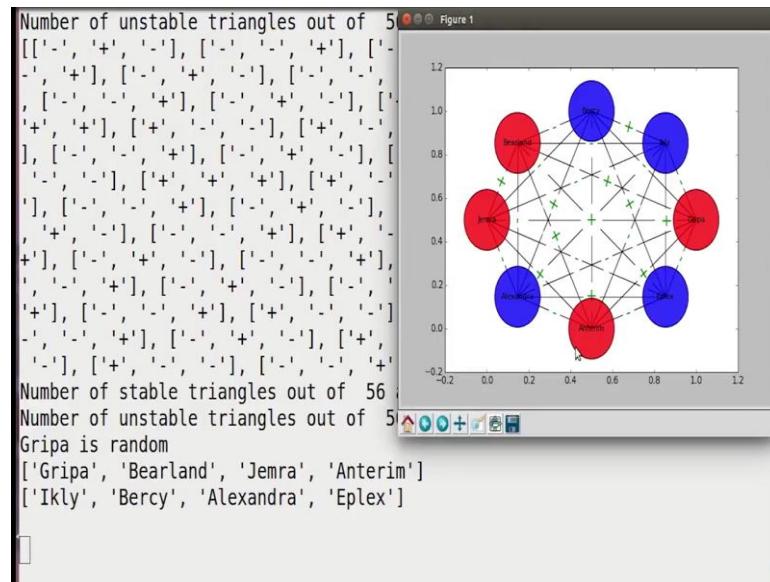
(Refer Slide Time: 06:23)



Let us run this; so, this is the initial graph; the initial network. Let us take an example of an unstable relationship here, let me see Bearland, Alexandra and Anterim this is a triangle which has two positive edges and one negative edge.

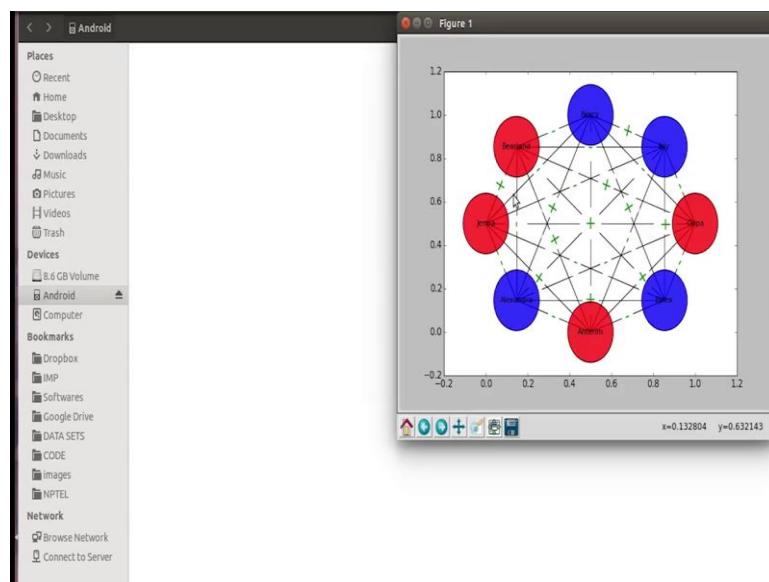
So, this is an unstable relationship this should not stay for a long time in the network. So, let us note down Bearland, Alexandra and Anterim; so, between Bearland and Alexandra there is negative and this two are positive.

(Refer Slide Time: 07:10)



So, let us see what happens to this triangle let me close it and it is now I am pressing enter; this is the final network with the different coalitions colored in a different color.

(Refer Slide Time: 07:19)



So here you see blue color nodes are in the first come in one of the communities I do not I do not remember which community; so, the second coalition nodes are in the red color.

Now, if we look at Bearland, Alexandra and Anterim; this has now become stable by; so, now, there is only one positive edge here.

So, what exactly is happened Alexandra and Anterim which were already friends; they have now turned enemies. So, this is as per our initial configurations that this is the kind of relationship that exists for a long time, but not the previous one. So, that previous one has led to the animosity between the Alexandra and Anterim that is one thing to see. Second thing is, if you look at all the red nodes there are all positive edges amongst them right Bearland and Gripa is positive Gripa and Anterim is positive you can check all the cases.

Similarly, if you took look at all the blue nodes; there are all positive edges amongst them Alexandra, Bercy, Eplex and Icly you see all positive edges are there; they are all friends. However, if you look at the inter edges which is the edges from one coalition to the other coalition; they are all negative which is quite an interesting pattern to see. So, Bercy to Bearland there is negative, Bercy to Gripa is negative Bercy to Jemra is negative; and similarly, Alexandra to Anterim is negative Alexandra to Bearland is negative.

So, precisely we have divided the network into two parts: where people in the first part are all friends, people in the second part are all friends, but these the people in these two parts hate each other; they are all enemies of each other. So, this is a nice and interesting phenomena to observe.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Link Analysis**  
**Lecture – 75**  
**The Web Graph**

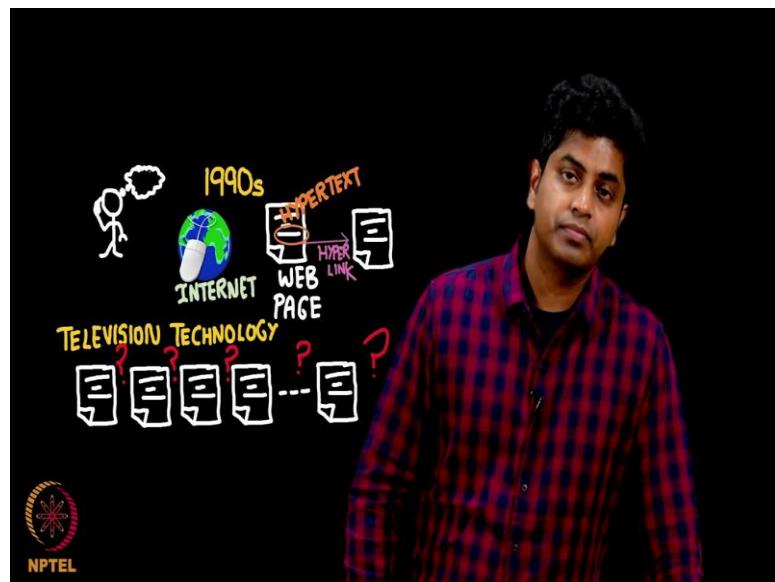
Let us now switch directions and talk about something that I spoke in at length in the introductory video.

(Refer Slide Time: 00:19)



It is about the origin of Google as an industry and the algorithm that they used to search the web. So, I will now call upon you all to imagine the following story.

(Refer Slide Time: 00:29)



Please note you really have to imagine otherwise, otherwise; you cannot get the gist what I am going to say. Please imagine that me and you are in the 1990s and I have this word with you that, see there is something called the internet which is fast developing. There is something called a web page where people put some information and there is something called hyper link, a hypertext where in the page you can link to another page. You click here, it just goes there right that is what you mean by hyperlink. By a link you mean something that links to somewhere else, but hyperlink means something that not just links hyper means what more than a link hyperlink is more than a link. Hyperlink means it just does not link you to a new place; it even takes you to that place.

So, a hyperlink was the invention of hypertext was very big leap in computer science. So, there are pages with hyperlinks to other pages and now me and you are thinking of a start up idea. In the 1990 where we see that internet is booming with a lot of web pages which are linked one-page links to the other page through a hyper link and we all start thinking. Now there are so many pages here thousands and thousands of pages. Today actually its billions of billions pages, billions of pages, but I am talking about 1990; me and you are having coffee table conversation in nineteen ninety.

Look there are so many web pages. How do you know what to search for? How do you know which page has the right information? Assume there are so many pages, web pages

talking about television technology, the television technology. Which page is a authoritative source for me to refer? If there are 100 pages which talk about 100 different web pages written by 100 different technicians, technology experts. Which is the best possible webpage? Now I see I repeat we are in 1990s, I see that this probably becomes a problem is going to become a problem very soon. Now that there are few pages people can sort of go through all that page and decide which one is good and then read it, but very soon the web pages will be populated with hundreds of thousands of pages. How on earth will people know what to look for where right? What is the immediate idea, you think we both will get over this coffee table discussion in 1990?

I for a reason would give this dumb idea that fine we will hire some thousand people. Assume we had we had so much money to pay thousand people; I will say we will hire thousand people and ask them to go through each and every page and maintain a sheet which has the following entries following columns.

(Refer Slide Time: 03:39)



First column will be the link second column will be the keywords which tells me what ~~is this link~~ ~~this link is~~ all about. This is the link I will say it talks about television technology. There is another link, this talks about health tips for good health; another link that talks about latest cars, another link that talks about World War-I so on right. First column will have the link, second column will have ~~this keyword~~ ~~these keywords~~ what I what is the link all about. The third column will be, I will read by me I mean ~~this~~

~~thousand employees~~ these thousand employees I am going to hire, each one of them will read the link and give the article rating. Or this article on World War-I is 8 on 10 rating, 8 on 10 like how we give rating for movies. Another person sees another link on World War-I and says this is not so good; it is 3 on 10 so on and so on and so on. These thousand employees in my start up industry will go through every single possible page ever created, ~~somewhat somehow~~, we get access to all the pages let us say and we rate all the pages and we also put keywords.

So, that whenever someone searches for a keyword television technology, then it will come to this database of all the entries that my thousand people whom I have hired have populated. It will look at the highest rated page which has the keyword television technology and display that to him. Maybe it will sort it in the order of rating, best rated link first next best second, third, fourth descending order. It will show it you wonderful idea, everybody will come to my page to search for information on what is the ~~so called~~ so-called internet with so many pages. Very soon I will become a millionaire because everybody will be coming to my page, my page will become popular. And once you have a popular portal, you can make a lot of money through ads or whatever right.

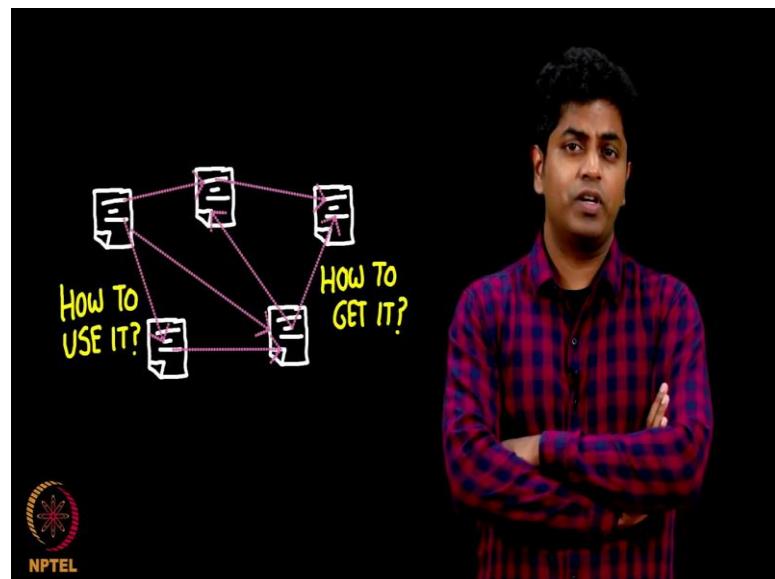
So, here is my idea in 1990 let us say, but then nobody ever dreamt that there could be a much ~~much~~ better way of doing this. I am going to explain next: what is the much ~~much~~ better way which paved its way for the birth of as I said a ~~450 billion dollar~~ 450-billion-dollar industry called Google; obviously, my start up idea will not scale up. Thousand people can do how much of work? The number of pages that gets populated on the internet is in terms of millions. So, very soon I will run out of man hours I am I will be need of more man hours and less employees; obviously, thousand is not enough. You should think of some other way of doing it right. What is this way of doing? It is ~~very~~ simple I will tell you the simple idea which is behind searching and again I will try explaining it bit by bit with analysis examples and questions; very simple.

(Refer Slide Time: 07:23)



| Larry Page and Sergey Brin, they came out with his idea of maintaining what is called a web graph.

(Refer Slide Time: 07:35)



By a web graph you mean take a page a webpage, call it a node. Take all the web pages, each web page is represented by a node. You wire a link between this page and this page namely this node and that corresponding node by a link. If this page has a hyperlink to that page; for example, the same example I have been giving you people. This is my homepage and I link the homepage of my friend on my homepage. Then my homepage is

a node, my friends homepage is a node; there is an edge from my home page to my friends homepage.

Please note this is a directed graph, why? Me linking to Prime Minister's homepage does not imply that the Prime Minister's homepage also links to mine. It is completely one sided correct. I like her, she does not mean she likes me. It is a very asymmetric relation. So, this is what is called a web graph where pages are denoted by nodes and a link represents a page pointing to another page through a hyperlink. The point is if you collect this web graph that is enough it does the trick of having thousand people or more. First question, how do I collect the web graph, what is the use of this web graph, how is this web graph going to solve the problem that I was trying to resolve with some thousand employees by hiring thousand employees right.

I have thousand employees in my organisation and now I am saying you can replace these thousand employees or even more by simply making node of the web graph. How do we get a web graph? What is the web graph? Think about it. It is a graph containing so many nodes and so many links. How do you even now the in some nook corner of the world, there will be a web page point into some other web page I am saying that you should have the all this information. How do you get this information? Do you have any idea how one can get this information? Any rough idea? Think about it.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Link Analysis**  
**Lecture - 76**  
**Collecting the Web Graph**

So, let us be organised. We are only asking these two questions. Let us answer them one at a time.

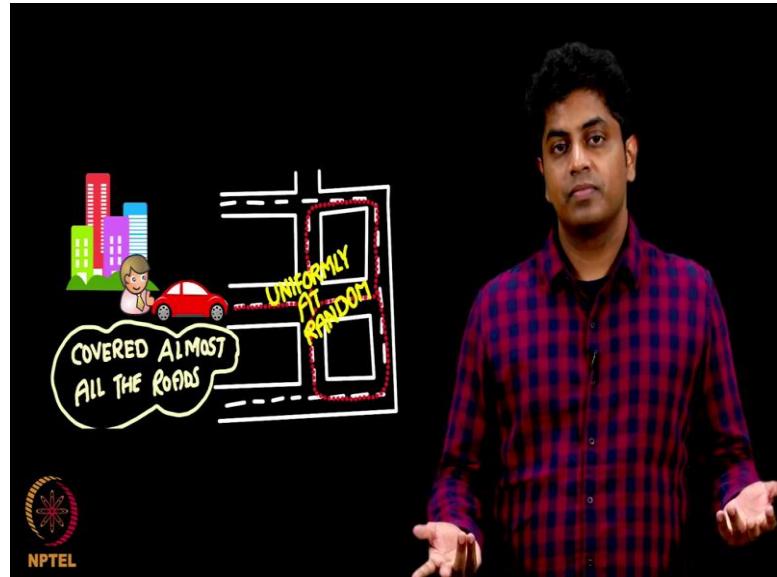
(Refer Slide Time: 00:15)



There is this question of web graph which will solve our problem; our problem of searching the web looks like unrelated you see. A graph of who points to whom which page points to what, has nothing to do with your search problem. So, first question is how we collect this web graph? I am saying this web graph will solve the problem. How will it solve the problem? Two things: firstly, how do you collect this web graph; secondly, how do you think this web graph will solve your problem.

Let us go step by step. So, first question. How do we collect this web graph? Here is a technique to collect the web graph. You should imagine a small situation.

(Refer Slide Time: 01:03)

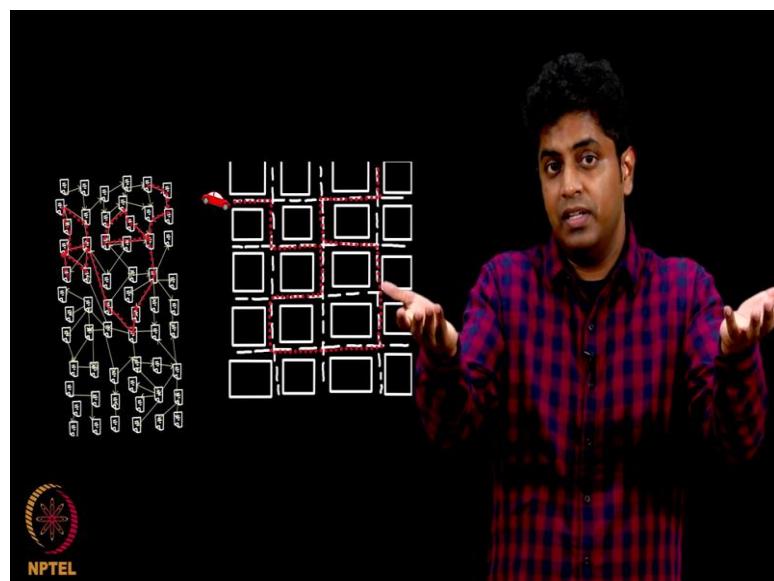


Imagine a city that you have never visited. And what I will do is I will put you to this new city; I will put you to this new city. Assume there are no vehicles in the city at all, you are the only one with a nice vehicle. I have given you a driver, a car road are empty; I repeat a new city you do not know of. Let us say you are not given a driver you drive the car, because if you are given a driver will know the localities. So, you are in a new city you do not know the localities, you are given the car; there is no traffic at all. What you do is you fix a camera on your car and just keep taking random turns here and there and keep going and going and going for the next one week. Assume you are given enough rest in between; you are given enough food in between; all you need to do is to simply take random turns and keep going.

Obviously you will not go a left and a right and a right and; obviously, you will not take a right and a right and a right and a right; you will come back to the same place, I said randomly. In every junction you toss a coin, you will take a decision whether you want to move towards your left or towards your right or towards straight. Or if there is a junction with 5 roads going from there, you roll a dice and you decide which side to go uniformly a track. You keep doing this for let us say, 1 week or more be even more may be 1 month. At the end of 1 month do you think there will be a road that you would not have explored?

May be yes, you would not have intuitively you feel you would not have. But do you think, you would have explored most of the roads in the city? Intuitively yes, you feel you would have if you take a car and keep going randomly wherever is an intersection, you break the tie by choosing a road uniformly at random and then move in that road and keep doing this for about a month's time in the city. You would have covered almost all roads right almost all roads you would have covered. Now what does this tell you? Does it tell you anything about the question that I just now asked?

(Refer Slide Time: 03:31)



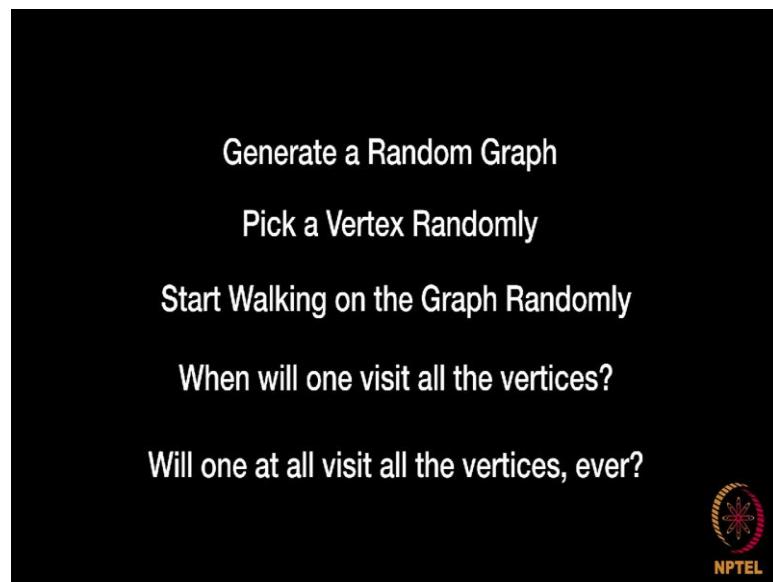
Is there a way in which you can explore this entire web graph and capture it? I gave you an example and you have this question in your hand, how do you get the web graph. The car, new city, taking random turns example; do you think you can link these two things? And do you think a question here has a solution which I stated in the form of an example?

(Refer Slide Time: 04:01)



So let us now do a quick experiment. I am curious to see if one takes a random walk on a given graph, how much time will it take for one to reach the entire graph by that I mean all vertices in the graph?

(Refer Slide Time: 04:19)



So, let us go slowly. I now need to generate a random graph on some  $n$  vertices with some probability  $p$  and then I pick a vertex randomly call it the start vertex. I will start walking on the graph randomly from this start vertex and then my question is when one will visit all the vertices in the graph. Will one at all visit all the vertices, ever? Pause for

a minute and think about this question and the what you think is your answer for this question.

Do you think it will take a long time or sometimes it is never possible to reach all the nodes by just taking a random walk? What do you mean by this? In a city, if you take a cab and then keep visiting places uniformly at random by tossing a coin and deciding whether to take a left turn or a right turn, how much time will it take for you to explore the entire city?

(Refer Slide Time: 05:21)

Let us write a quick python code to  
check how many random walks  
does it take to reach the entire  
graph...



Let us now write a piece of code a quick python code to check how many random walks does it take to reach the entire graph. By reach the entire graph I mean reach all the vertices of the given graph.

(Refer Slide Time: 05:41)

```
 1 import random          "
 2 import matplotlib.pyplot as plt
 3 import networkx as nx
 4 import numpy
 5
 6 def walk(n,p):
 7     start=random.randint(0,n-1)
 8     G=nx.erdos_renyi_graph(n,p)
 9     S=set({})
10     v=start
11     count=0
12     while(len(S)<n):
13         Nbr=nx.neighbors(G,v)
14         v=random.choice(Nbr)
15         S.add(v)
16         count=count+1
17     return count
18
19 l=[]
20 for i in range(20,300):
21     z=[]
22     for j in range(10):
23         z.append(walk(i,0.3))
24     l.append(numpy.average(z))
25     print i,"-->",numpy.average(z)
26 plt.plot(l)
27 plt.show()
28
~
~
```

"checkrandom.py" 28L, 522C written



Here is a piece of python code that have written. I am sure you people are, I am sure you people all of you are very familiar with python right now. You know how to write a piece of code. So, I have not gone a line by line. I have finished this code and you can probably take a pause and then take a look at this code and then judge what the code is doing. So, as you can see, I have imported a few functions import random matplotlib networkx and NumPy.

NumPy is used to compute average; it is a nice package for scientific computations. I am not using it much except for the 24th line as you can see, I am computing the average here in 24th line ok. Let me go slowly. This function walk (n, p); all it does is it generates a random graph with n nodes and p vertices starts from a random vertex in the given graph G (n, p). By that I mean an addition graph with n nodes and p probability p. And then I start with a with a vertex v which is uniformly picked from the vertex z.

And then I keep walking on this graph and then count how much time it takes until I visit all elements of the vertex set ok. Take a look at this code. It is quite self explanatory and then I what I do is I call this walk function quite often with i ranging from 20 to 300 with probability 0.3. Take a look at the walk function, you will understand what I am doing here. And every time I get an answer, what is the answer? The answer here is walk of i comma 0.3, simply returns the number of attempts number of random walks you must

take until you finish going through all the given vertices of the graph with i vertices and p equals 0.3.

And then I append this to a list called z and repeat this experiment some 10 times and then take the average of the number of random walks it takes for me to reach the entire graph. And then I do it for graphs of the size 20, 21, 22, 23 up to 300. And, then I plot on the x axis this range namely the number of vertices in the graph and in the y axis this plots the list l which stands for the number of random walks you have taken to cover the entire graph. So, let me execute this program and see what happens.

(Refer Slide Time: 08:37)

```
 1 import random
 2 import matplotlib.pyplot as plt
 3 import networkx as nx
 4 import numpy
 5
 6 def walk(n,p):
 7     start=random.randint(0,n-1)
 8     G=nx.erdos_renyi_graph(n,p)
 9     S=set([ ])
10     v=start
11     count=0
12     while(len(S)<n):
13         Nbr=nx.neighbors(G,v)
14         v=random.choice(Nbr)
15         S.add(v)
16         count=count+1
17     return count
18
19 l=[]
20 for i in range(20,300):
21     z=[]
22     for j in range(10):
23         z.append(walk(i,0.3))
24     l.append(numpy.average(z))
25     print i,"-->",numpy.average(z)
26 plt.plot(l)
27 plt.show()
28
~
```

!python %



(Refer Slide Time: 08:39)

```

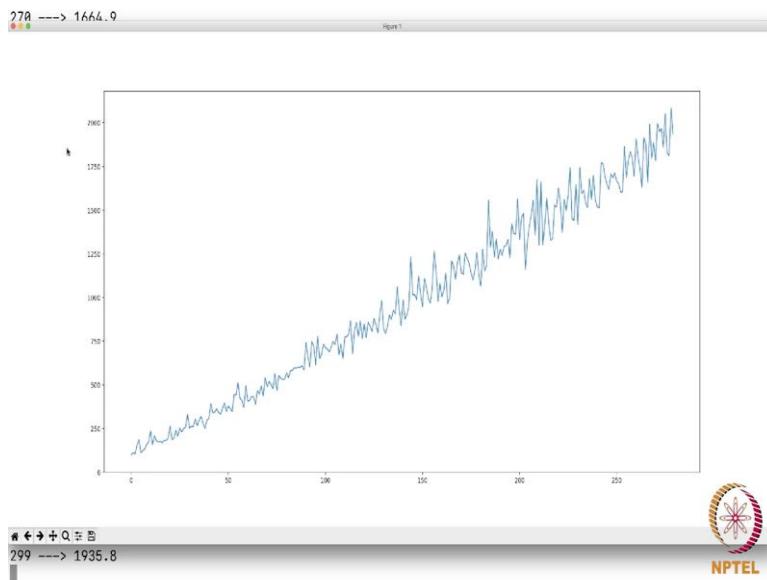
247 ----> 1449.6
248 ----> 1448.6
249 ----> 1647.8
250 ----> 1418.0
251 ----> 1743.5
252 ----> 1594.5
253 ----> 1612.7
254 ----> 1543.9
255 ----> 1513.7
256 ----> 1679.3
257 ----> 1568.9
258 ----> 1699.8
259 ----> 1556.9
260 ----> 1517.5
261 ----> 1512.9
262 ----> 1772.9
263 ----> 1763.5
264 ----> 1691.0
265 ----> 1646.6
266 ----> 1618.5
267 ----> 1706.6
268 ----> 1684.8
269 ----> 1712.6
270 ----> 1664.9
271 ----> 1652.6
272 ----> 1603.6
273 ----> 1604.9
274 ----> 1864.9
275 ----> 1687.6
276 ----> 1786.5

```



As you can see, these are the number of nodes and the number of random walks it takes for you to cover the entire graph. So, let us wait and see what happens. So, 263, 4, 5, 6, 7 so, on; it goes on till 300. So, as you can see the right side is the answer which is the number of random walks it takes for you to go to the entire graph.

(Refer Slide Time: 09:13)



So, now I get the plot the plot looks beautiful as you can see um. The x axis is the number of nodes in the graph and y axis is the number of attempts it takes for you to number of random walks, you need to take to cover the entire graph.

What does the signify? It signifies that it is just a little bit of an effort a few times the number of nodes in a network is what it takes for you to cover the entire graph and that is a very nice observation that we make. As you can see the plot clearly says it is looking very linear, although it is not very linear a little bit of observation will tell you that it is very close to a  $n \log n$  function. Anyway details aside we will not worry much. All that I need to understand is this is very much possible.

(Refer Slide Time: 10:07)

```
271 --> 1652.6
272 --> 1603.6
273 --> 1604.9
274 --> 1864.9
275 --> 1687.6
276 --> 1786.5
277 --> 1834.0
278 --> 1798.0
279 --> 1695.5
280 --> 1906.5
281 --> 1807.1
282 --> 1737.9
283 --> 1629.6
284 --> 1915.0
285 --> 1879.5
286 --> 1659.3
287 --> 1991.1
288 --> 1796.8
289 --> 1886.4
290 --> 1784.4
291 --> 1996.0
292 --> 1950.0
293 --> 1967.2
294 --> 1860.2
295 --> 2850.5
296 --> 1825.8
297 --> 1810.9
298 --> 2882.6
299 --> 1935.8

Press ENTER or type command to continue
```



For example, when I take a node on 299 when I take a graph on 299 nodes, when I take a graph on 299 nodes in roughly 2000 random walks, I cover the entire graph. As you can see 298 seems to have taken a little more time as you know this is averaged. So, if you have average a lot more, this will be a properly increasing function.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Link Analysis**  
**Lecture - 77**  
**Equal Coin Distribution**

Going with the flow, we had two questions.

(Refer Slide Time: 00:11)



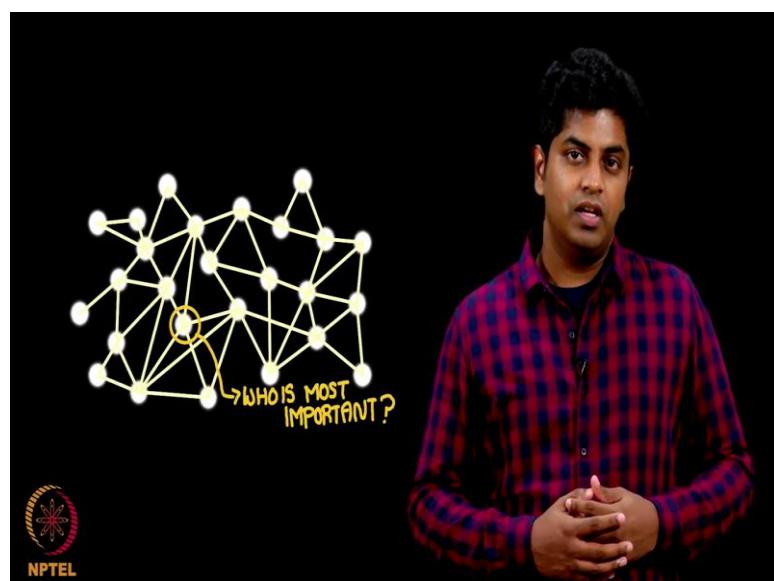
Question 1 was collect the web graph. Question 2 is use this web graph to solve your search problem. Question 1 you all are convinced is very doable, why? You saw the plot just now. When there are  $n$  number of nodes, it is not very difficult to explore the entire graph with  $n$  nodes. Random walk makes you search for all possible nodes and visit all the places that way you would know what is linking to what; not only will you get all the nodes; you will get a whole lot of edges as well right.

(Refer Slide Time: 00:45)



So, which means a partial structure of the graph is reveal to you. The second question that we asked how can we use such a web graph to solve our search problem? That we should go slightly slowly. I am going to divide my explanations in several pieces. I will go piece by piece; at the end put all these pieces together and you will see that we can solve the search problem just by using this web graph. So, I am going to go in bits and pieces and solve the problem of how we can use this web graph to solve the search question. The first piece is as and always a puzzle.

(Refer Slide Time: 01:43)



You are given 30 people with complicated friendship between them, you have to decide who is the most important person here and I am going to use a very weird strategy to find out who is the most important person. Repeat 30 node graph, they are all friends. I am going to find out a nice way in which, I am going to use a nice way in which I can find out who is the most important person here. What do I do? As I told you it is a very weird strategy; do not break your head on why am-I doing it.

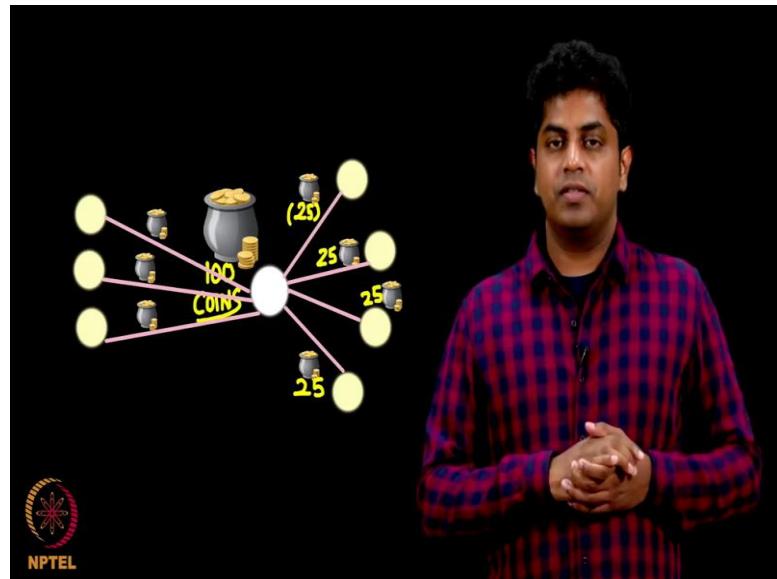
What has this to do with a question? As I told you I am going to go in bits and pieces. The first piece is the stand alone question the standalone question; I mean do not try to relatedrelate with anything else just look at this is a problem in its own right. 30 people and I am going to use the following strategy to find who is good here, who is the best here.

(Refer Slide Time: 02:37)



Basically, I am going to rank everyone who is the best, who is the second best, third best. What I do is I take 100 gold coins and give it to every single person on this network of 30 people. So, the first person gets 100 gold coins second person gets 100 gold coins 100 100 100 100 30 people; each of them get 100 gold coins. And what isdoes the first person do?

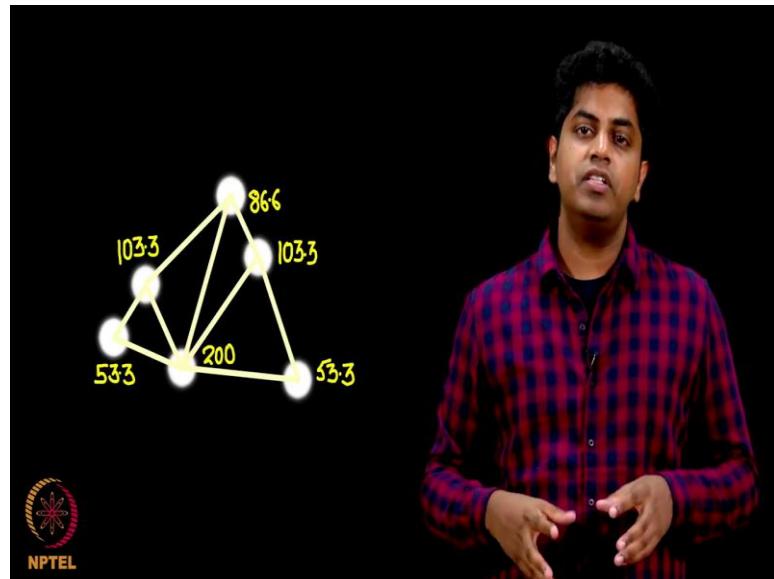
(Refer Slide Time: 03:15)



Assume a person has 3 edges. This node has 3; let us just take one node it has 3 outgoing edges. So, 100 can be divided into 33 33 33; I am going to round off ok. So, let us say 90 let us say 4 edges are going not 3, 4 edges are going 100 gold coins to a node and this node gives 25 gold coins to each of its neighbours. If it was 3, I will just divide it by 3 and 33.33 each. I will assume gold coins can you can consider decimal part of it and then give it and by powdering it let us say.

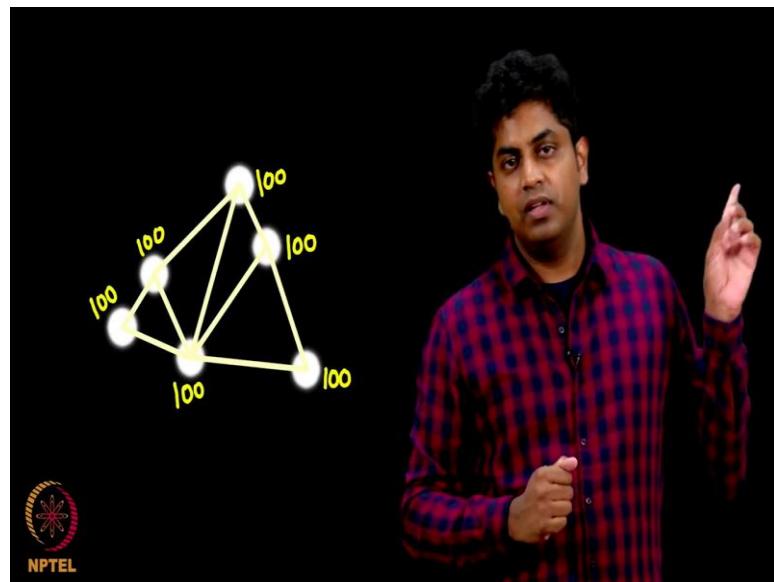
So, every person distributes the gold coins that they have to their neighbours. Now this one who has 4 neighbours to him will have some people who have him as a neighbour. So, he will also get some gold coins right.

(Refer Slide Time: 04:19)



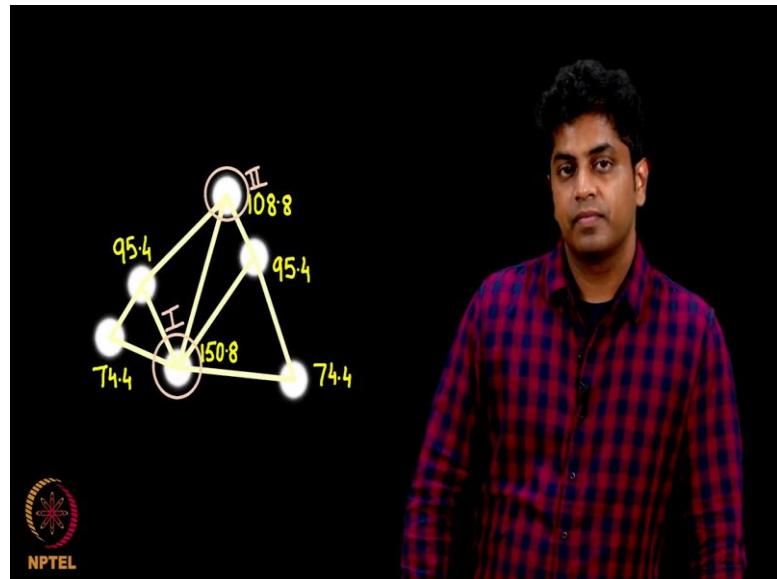
So, all I do is I take a network. As you can see 100 gold coins, each one iteration one snap like this will make everyone distribute their existing gold coins to all their neighbours.

(Refer Slide Time: 04:33)



You see with one snap, everything changes in the network on these 30 nodes. Another snap again it gets distributed. Observe very closely.

(Refer Slide Time: 04:37)



What is the distribution? Whatever gold coins you have, whoever are your neighbours you should just go and give it to them and whoever is pointing to you they will give their gold coins to you. In every snap this happens, every snap this happens. Keep doing this, snaps are basically iterations you see the graph changing; first snap, change; second snap, change; third snap, change. You see it keeps changing like this, keeps changing like this and so on and so forth. I keep doing this up to eternity guess what happens.

Very interestingly, this converges which means there is a stage after which the gold coins get accumulated and it does not change; just stays stagnant. And in that state, you go and see who has the maximum gold coins, you will see there is this node which has the maximum gold coins. You call him rank 1. Now what is it denote how did you accumulate so many gold coins? You did some random process, you simply started giving gold coins that you have what isdoes this even denote in the first place?

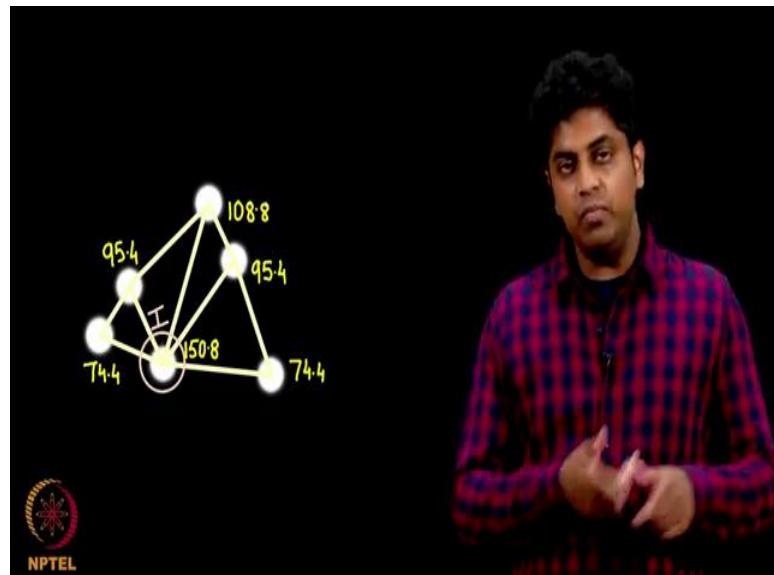
Let us see that in a while, but I just did this for fun, a very weird strategy to find out who is the most important person. Whoever accumulates the maximum gold coins eventually with time is the most important person. The second most gold coin collector is the second most important person; third, third most important person; fourth, fourth; fifth, fifth so on. What just happened? Pause and think about it.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**  
**Link Analysis**

**Lecture - 78**  
**Random Coin Dropping**

So, just now we saw a strategy of every node being given 100 gold coins and these nodes distribute all the gold coins to their immediate neighbours; equally. And as this process continuous and continuous the number of gold coins they get distributed not evenly although, we started evenly; they get distributed haphazardly.

(Refer Slide Time: 00:26)



And I go and see that node which has accumulated the highest number of gold coins, call him the winner first king; first top one person; rank 1 person. Second most gold coin accumulator, I call him the second best, second king I rank him accordingly. That was one way.

I will now give you yet another way of ranking the nodes. Now, I will not use gold coins; I will use something else of course, I will use gold coins, but I will give the gold coins in a quite different way.

(Refer Slide Time: 01:10)



What I will do is, I will stand on a node, some node and from 1 node assume I have 4 neighbours, I jump to a neighbour uniformly at random and give that neighbour a gold coin. From that node I will jump to yet another node, whatever it is pointing to maybe, it is pointing to 5 nodes. I will pick 1 node uniformly at random jump there, give him a gold coin. Remember the car and roads experiment I keep visiting the places I said, and I explore entire city, I am doing something similar here. From 1 node is just take a random hop and go to another node. Whenever I visit a node, I drop a gold coin on that node, I keep doing this, keep doing this, keep doing this, and I see, how many gold coins are accumulated by these nodes?

After some 1 million such random walks; I am walking randomly right. So, I call it a random walk. Random walk, gold coin drop; random walk, gold coin drop; random walk and I drop a gold coin. So, on and so, on and so, on and million times and I observe after a million iterations, every node has accumulated gold coins.

Although, they are not equal different nodes have accumulated different number of gold coins right. We can see that the figure says, if I take 1 million random walks which node gets how many gold coins? At the end I again, look at which node got the highest gold coin? I observe that the node which got the highest number of gold coins here was the same node which got the highest gold coins in my previous strategy previous experiment.

In fact, the second best here will also be a second best there. In fact, third best, fourth best, fifth best, all of them here we will agree with everything else there. It is very surprising that this is true for you, but mathematically if one sees it is not at all a surprise. In fact, as part of the advanced material for this chapter I am going to cover the mathematical details of why and how this is happening as well.

You are free to skip that its sort of it is not required that you go through it, but if you are curious to know what is happening, here where these 2 things the same, you may want to take a look at the mathematical material that I am going to record in the next few days, I mean in the next few hours of the video well.

(Refer Slide Time: 03:59)

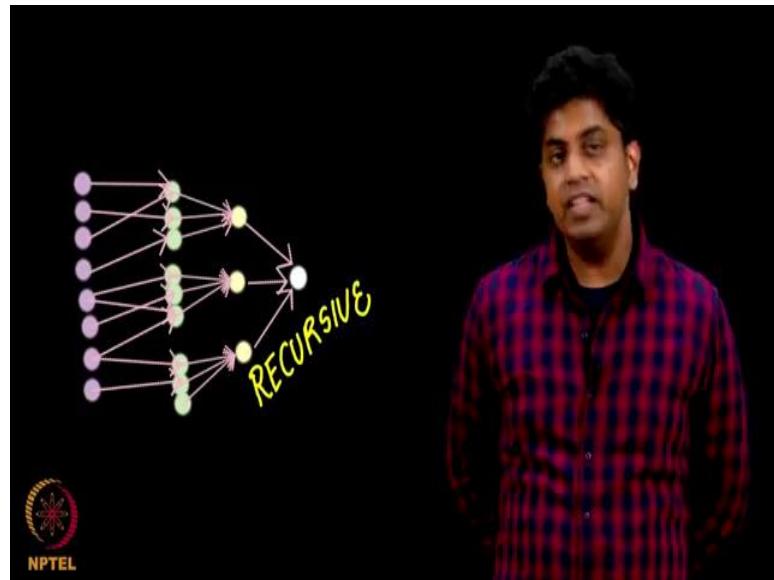


All that you need to buy from this explanation is that the strategy of 100 gold coins being distributed equally versus taking a random walk and dropping a gold coin, both these strategies rank the nodes in a same way; in exactly the same way. So, the moral of the story is sharing gold coins game and dropping gold coins game yield the same ranking of the nodes. So what? I am going to analyse this first game and the second game properly, for reasons of better explanation I am going to call the first game. The equal sharing of gold coins game, the second game the random walk drop the coin game. So, simply speaking equal sharing and random walk; game 1, game 2.

Let us slowly analyse and understand what these 2 games mean. What has happened so far? 2 games; this game is same as this game in terms of ranking the nodes. I will first

look at the first game which is 100 coins and equal sharing. Let us just think what happened here? What kind of nodes will attract the highest number of coins? A node has highest number of coins, if it is getting a lot of coins from its neighbours.

(Refer Slide Time: 05:19)



When will the neighbours give this a lot of coins? If the neighbours are getting a lot of coins. When will the neighbours get a lot of coins? If they are adjacent to whatever if there pointed by neighbours who get a lot of coins. You see this is a very recursive argument here. This captures the very notion of the following definition: you are famous, if famous people say you are famous, and those people are famous if famous people say that they are famous so on and so forth. This goes on like this correct.

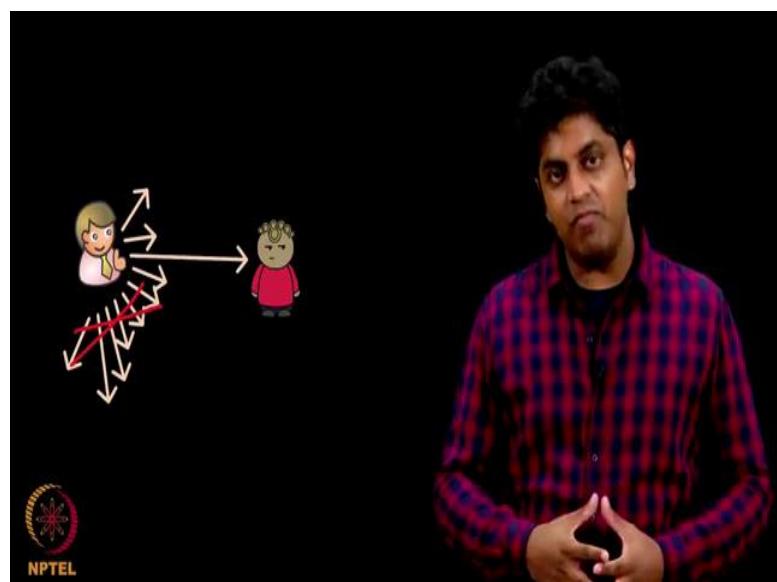
We discuss this in the introductory lecture in the first chapter right. You are famous if someone famous says you are famous. It is a very recursive definition. So, you see there is a definition for there is a abbreviation called GNU. If you are to computer science, you will know what is GNU. GNU stands for GNU is not UNIX and now what does that GNU stand for? Again, GNU is not UNIX; this is called a recursive definition.

For fun they have called it this way, but you see the all the all the time trying to say is the definition is sort of recursive here. You are famous if someone famous says you are famous and how do you know they are famous? What do you, what do you mean by famous then? Famous they are famous if someone famous says they are famous. And this famous is defined very recursively. Now you see what is happening in the first game. A

node accumulates a lot of gold coins if a lot of people, the people who are adjacent to it accumulate a lot of gold coins right. Only then you will pass on the gold coins to me.

Please note, for me to accumulate a lot of gold coins people who point to me should accumulate a lot of gold coins and they should not have a lot of people adjacent to them. Now, what do I mean by this?

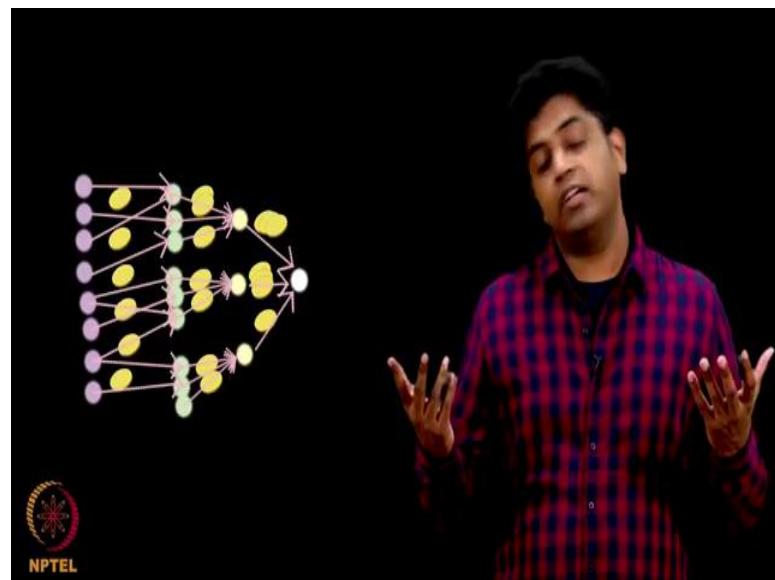
(Refer Slide Time: 07:28)



I have a rich cousin brother who helps me out whenever I want money let us say and I have 1 such cousin brother, but assume he has 10 such people to whom he is ready to help. Then I may not get all his help because his help sort of gets divided into 10. So, there are 2 aspects here; you should be having some good backups like a rich cousin brother to help you out in times of distress and he must not be ready to help a lot of people.

So, he must have very few friends outgoing, and he must be rich. When will he be rich? When a lot of people point to him, so, a rich person, a rich node is being pointed at by a lot of people, lot of rich people let us say. But that person should point to a few people and one of those few people should be you I am just set up dissecting this problem and then observing it. But what I am trying to say is the moral of the story so far.

(Refer Slide Time: 08:38)



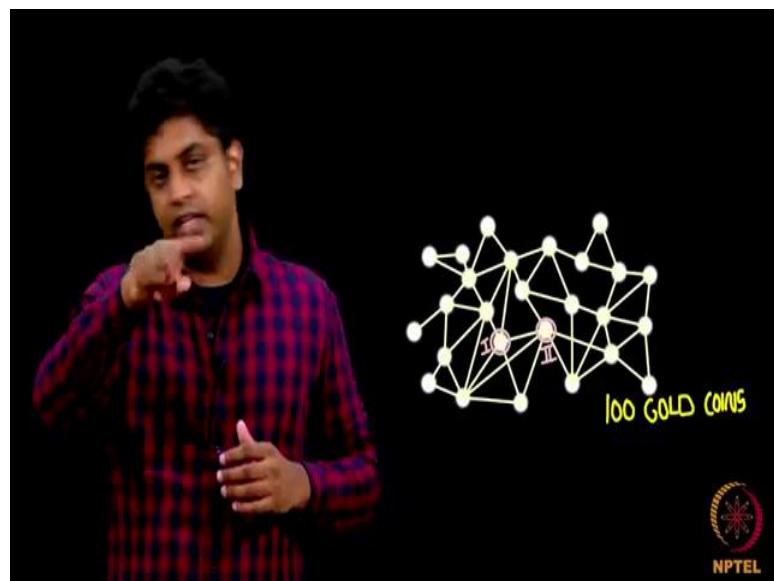
If you have understood that a node accumulates a lot of gold coins, if it is being pointed by nodes that accumulate a lot of gold coins. This much is enough and then you call a node to be highly ranked if it is being pointed by highly ranked nodes.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**  
**Link Analysis**

**Lecture – 79**  
**Google Page Ranking Using Web Graph**

Let us now, get back to the web graph question, 2 questions that we observed. We asked rather, how do you collect the web graph? How are you going to use it to solve your search problem? I am going to collect the web graph by taking random walks I collected. As you are collecting, as you are visiting nodes through random walks, you give points to each node.

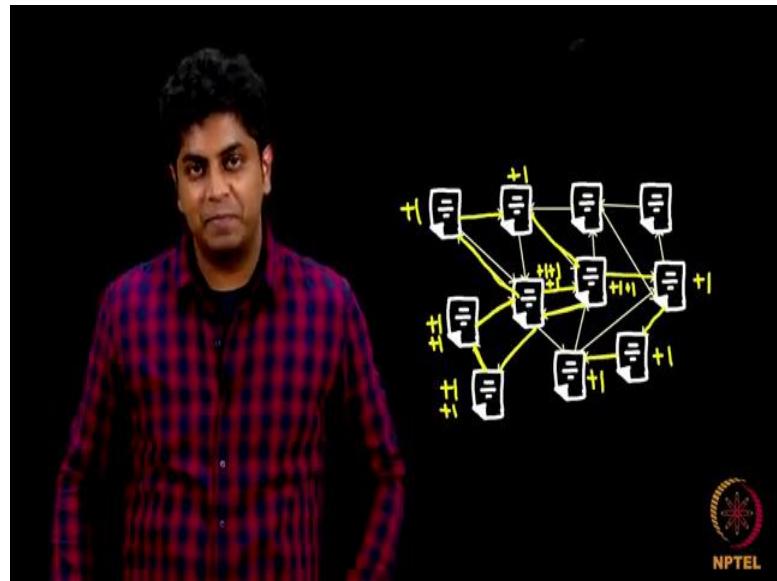
(Refer Slide Time: 00:36)



For example, on this 30-node network which we did in let us say, second game the random coin dropping game. We dropped the coins and we saw which node got the highest gold coins.

On the web graph we are going to do the same thing. We are going to walk on this web graph.

(Refer Slide Time: 00:53)



We keep walking, keep walking, keep walking, we walk like let us say billion times and keep dropping the coins and we look at which node has the highest coins. And when you search for a keyword in the search engine what Google does, is this. It firstly, keeps walking on the web that is called crawling; it keeps crawling on the web keeps dropping the gold coins and sees which node has the highest gold coin.

Every single page has basically, the gold coin accumulation points ok. When you type in let us say a keyword namely ah, I will say Britney Spears; if you type in the keyword Britney Spears we all know she is very popular, very popular pop singer. There is a there are lot of pages which talk about Britney Spears, which page should come first? What Google will do is it will go and retrieve all those pages, which has anything to do with Britney Spears. You will have some 10000 pages maybe, even more and as an example let me say 10000 pages.

Now, Google does not display in fact, it displays all the 10000 pages, but it should display the most relevant page in the beginning. What is the most relevant page? That page, that has accumulated the maximum gold coins; this is how Google works. Google just displays all these 10000 results in descending order of the points that page is accumulate as Google crawls the web and drops a gold coin, on each page that it visits. Very simple, very straight forward, very easy to understand algorithm, very easy to implement technique it works like a charm.

Google harnessed the potential of what is called a web graph, where web pages are related by hyperlinks and Google as an industry thought one should do before it became Google, in fact in fact, story is very interesting, these 2 people Larry Page and Sergey Brin, they were sitting in a old garage and they were trying to implement this; this particular algorithm of crawling through the web. In fact, their crawling was also very suspicious that people thought that they are trying to hack something and things like that, in fact, they gone into trouble because of this. You can read up the literature for this.

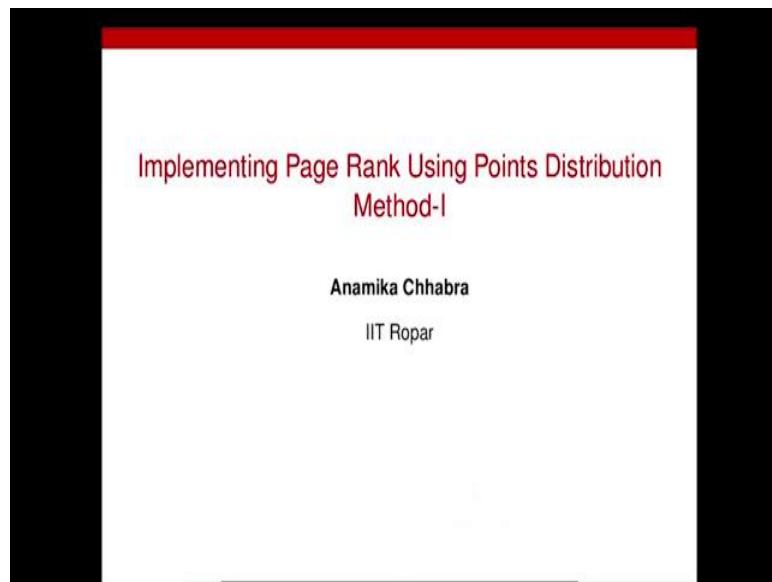
I mean articles on this, in fact, there is a nice documentary made on this as well. I strongly suggest that you watch it. So, these people were doing something out of the box, which was like crawling the entire web and then giving points to each node and using that to give you the right search results. Work like a charm something that is unexpected which is the web graph seemingly useless of what use you think a person like me and you, remember when we had this coffee table discussion in 1990, do not you think? By no means one could think of seemingly useless ah data like the web graph coming in use in a problem such as search.

And that seemingly useless data made this industry a big billion-dollar industry for a simple reason that it avoided hiring thousands of people which is like the first thought that one will get when it comes to solving the search problem. All that they did was take the web graph. How easy is to take? It is easy to take just crawl, take random walk; you will get all the web graph. Not only will you get as you crawl, as you take random walk, drop a gold coin and see which page has how many gold coins. And then when someone asks for a keyword search and show all the pages which has this keyword but sort them according to the gold coins accumulated by this web pages and that is precisely what Google does.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Prof. Anamika Chhabra**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**  
**Link Analysis**

**Lecture – 80**  
**Implementing Page Rank Using Points Distribution Method – 1**

(Refer Slide Time: 00:06)



Hey everyone, in this video we are going to implement page rank using points distribution method. I assume, you have been already introduced to this method in the previous videos. So, I will just be briefing out the technique and then we will start the implementation. So, in this method what happens is, we start with the signing of fixed equal number of points to every node in the graph. After that every node distributes its points to its neighbours, by neighbours I mean the nodes that are connected to this node by an out link.

So, basically what we do is we take a node and we look at its out links and the nodes we are connected to this node through an out link get an equal share of points from this node. For example, if there is node which has 100 points and it has 2 out links; so, basically it has 2 neighbours. These 2 neighbours should be getting 50 points each from this node. That is, that is called points distribution in case the node is having 4

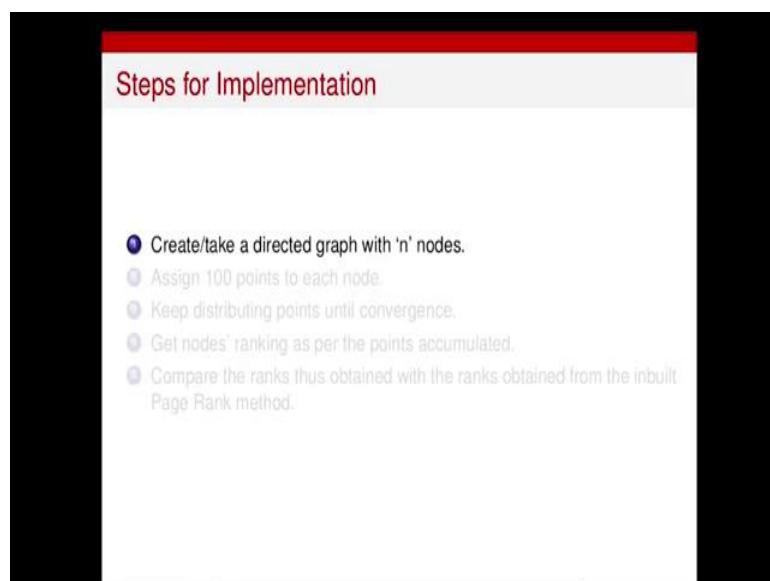
neighbours and its having 100 points in that case every neighbour will be getting 25 points each from this node.

Now, that point distribution happens for every node in the graph that is called one iteration. So, every iteration, so, after every iteration the points that every node contains keeps changing. For example, initially if we assign some 100 points to every node and we perform this points distribution after one iteration, the points that every node contains will change. And we keep repeating this process and after every iteration the points will keep changing; however, after some point after some number of iterations what happens? The points do not change, that is even after distributing points the points that the nodes were containing earlier is same as the points that the nodes will be containing after the iteration.

That happens because the number of points that the node distributes is equal to the number points and it gets from the nodes through which it relates to an in link. So, we basically get a convergence, and as soon as we get the convergence, we stop. And this point every node will be having some number of points after attaining the convergence, every node will be having some points and we can sort the nodes based on that that point distribution. Basically, we can rank the nodes and that ranking is called page rank.

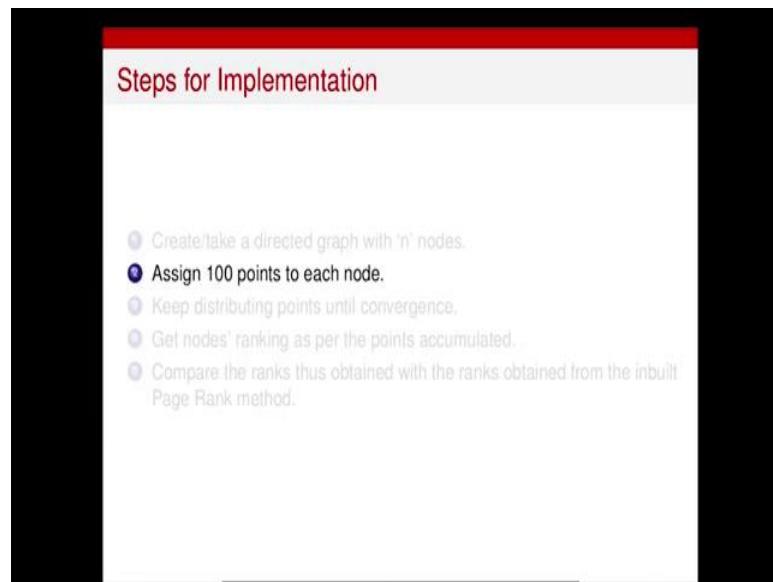
So, that was about the points distribution method. Now let us see the steps that we are going to follow for the implementation.

(Refer Slide Time: 01:49)



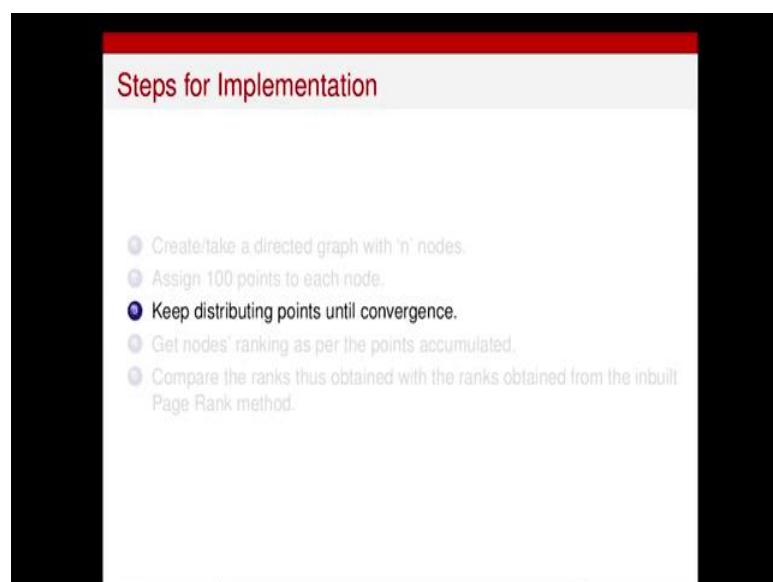
So, as a first step we are going to take a directed graph as you might be knowing for page rank it is better to take a directed graph because the points are distributed based on the out links. So, we are going to create a directed graph, we can also take make use of a function from networkx to generate the directed graph, but we will be creating a graph here by ourselves.

(Refer Slide Time: 03:19)



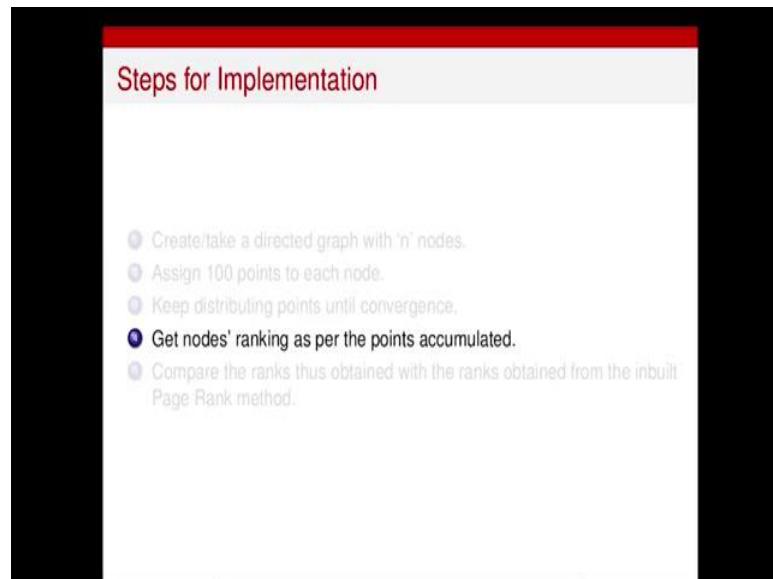
After that we will be assigning 100 points to each node. So, that is the initialisation of the points to every node. As I told you after that every node will distribute the points.

(Refer Slide Time: 03:30)



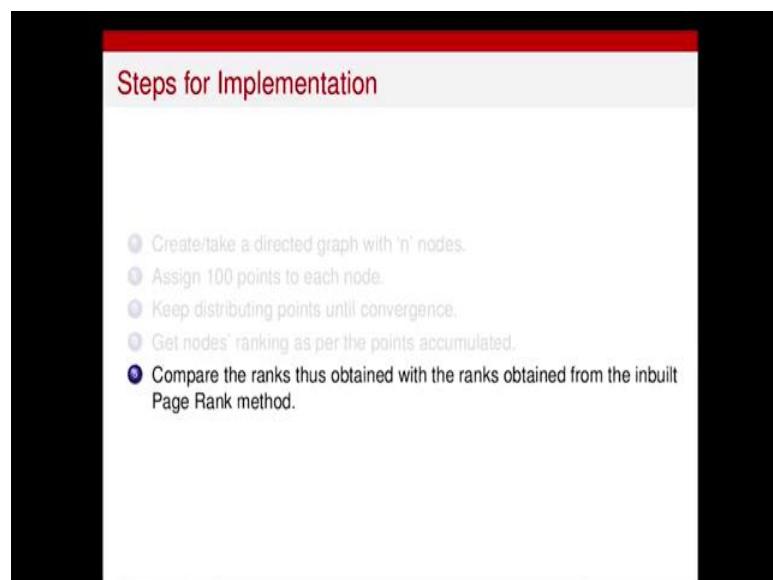
So, as a third step every node will keep distributing its points to its neighbours and this process will keep repeating until we get a convergence that is, until a point where the nodes' points stop changing, that is where we will stop.

(Refer Slide Time: 03:49)



Now, after that we will be getting some distribution of points amongst these nodes and we can rank the nodes based on the points that they have accumulated. That ranking will be called page ranking; page rank basically, to validate whether the method is working fine or not.

(Refer Slide Time: 04:08)



What we will do is as the last step is, we will make use of the inbuilt page rank method from network x and we will see the ranking that we get from that method and we will compare the ranking that we got from our implementation. So, these are the steps that we are going to implement. Let us get started with implementation.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Prof. Anamika Chhabra**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**  
**Link Analysis**

Lecture - 81  
Implementing Page Rank Using Points Distribution Method – 2

Let us start with the Implementation of Page Rank Using Points Distribution Method, as we discussed in the previous video.

(Refer Slide Time: 00:12)

```
points_convergence1.py * points_convergence_recording.py *
1 import networkx as nx
2 import random
3
4 def add_edges(G, p):
5     for i in G.nodes():
6         for j in G.nodes():
7             if i != j:
8                 r = random.random
9
10
11
12 def main():
13     # Create/take a directed graph with `n` nodes.
14     G = nx.DiGraph()
15     G.add_nodes_from([i for i in range(10)])
16     G = add_edges(G, 0.3)
17
18     # Assign 100 points to each node.
19     # Keep distributing points until convergence.
20     # Get nodes' ranking as per the points accumulated.
21     # Compare the ranks thus obtained with the ranks obtained from the i
```

I have pasted to 5 steps from the previous video over here and we will take the steps one by one. So, let me create a main function here ok. So, let me import networkx first. So, let us take these steps one by one. So, the first step is create or take a directed graph with n nodes as I told you previously. So, we can use a generator for creating a directed graph that is some function from network, we can make use of. For example, we have a function gnr graph which gives us a directed graph. We can use any function, or we can create our own function to create a directed graph as well. In this video we are going to created at directed graph ourselves.

So, let us see how we can do that. So, as a first step I am just creating an empty directed graph by using this function `G = nx.DiGraph` ok. This graph is empty, but it is a directed

graph. Now, let us add the nodes and then we will add the edges. Let us add some nodes to it. So, I will write `G.add_nodes` from. So, we will add a list, I will write `i` for `i` in range, let me add some 10 nodes here. We can add more as well, but I want to show you the conversions, if you took take and a number of nodes it will take more nodes, it will take more time for convergence; you can always change that. So, I am taking 10 nodes at this point.

Now, we have added the nodes, we must now add the edges. So, I am going to use a technique of course, you can use many different methods to create a graph. The method that I am going to use here is that I will be randomly adding edges between the nodes, that is also well known technique to create a random graph which I think we will be discussing in the next weeks video. But I will be just briefing out the technique here because we will be creating this directed graph using that technique.

So, for that purpose I will be creating a function, let me create a function `add_edges` ok. I will pass the graph here which has nodes and no edges as of now. So, I will be probabilistically adding the edges, I will briefly explain you the method as well. I am passing a probability value of 0.3 here and this function will be written in graph where the edges have been added.

So, let me create this function here I will create it here so, I will write. So, as a parameter this function requires the value of `p` which is a probability value. So, let me tell you what we are you doing here. So, we are going to take all possible edges that can ever be added to the graph and we will toss I mean, and we will add these edges to the graph after tossing a coin. By that what I mean is we will take an edge and we will toss a coin, if a coin turns out to be head we will add that edge and if the coin turns out to be tail we will not add that edge.

So, here in this case as you can see that we are passing a probability value `p` here and the value of `p` we have passed as 0.3; you can pass any value here, you can pass any value between 0 to 1 over here. So, by passing this probability value what we mean is that, you can assume that the coin is biased by a probability 0.3 which means with the probability 0.3 the coin will turn out to be head. And, with the probability 0.7 the coin will turn out to be a tail.

So, we will take an edge, we will toss the coin which is biased by this probability value and if a coin turns out to be head, we will add that edge otherwise we will not add that edge. So, this is the technique that we are going to follow. This is a well-known technique which is used to basically create random graphs. We are using this technique; you can use any other technique as well.

So, let us see how we can implement this. So, we have to take all possible edges, how can we do that. I can start loop I can write for i in G.nodes. So, I got all the nodes here because, we want the edges I will start another loop here. So, I will write for j in G.nodes ok. So, i comma j is we will give us all possible edges, we are now going to add self edges. So, I am going to check over here whether i is equal to j or not. So, I will write if i != j only then go ahead.

So, we have to implement that coin tossing here now, can you imagine how we can do that. We are going to take a random value for that we need to random package. So, let me import that, import random. So, we are going to make use of function random.random which gives us a random value between 0 and 1. Let me store that here r = random.random.

(Refer Slide Time: 05:56)

```

 2 import random
 3
 4 def add_edges(G, p):
 5     for i in G.nodes():
 6         for j in G.nodes():
 7             if i != j:
 8                 r = random.random()
 9                 if r <= p:
10                     G.add_edge(i,j)
11                 else:
12                     continue
13     return G
14
15 def initialize_points(G):
16     points = [100 for i in range(G.number_of_nodes())]
17     return points
18
19 def distribute_points(G, points):
20     prev_points = points
21     new_points = [0]
22

```

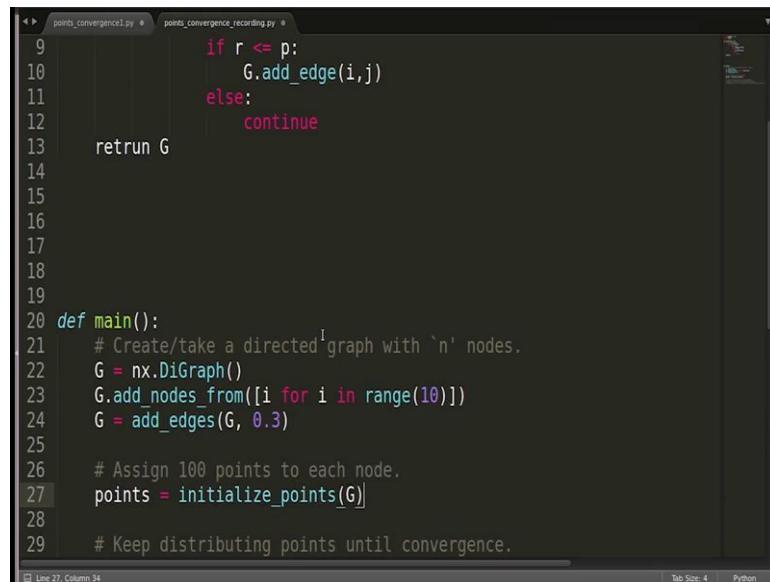
Now, we got some value here in r, we will check whether that value  $r \leq p$  value or not. If that if  $r \leq p$ ; that means, we got a head and if  $r > p$ ; that means, we got a tail. So, that is that how we are going to implement that coin tossing part. So, we got this random

value, let us check whether  $r \leq p$ ; if it is we are going to add that edge. So, I will write  $G.add\_edge$ , the edge is going to be  $(i, j)$  right and the edge part when  $r > p$ , we assume that that is that is a tail.

So, we will not add that edge, we will just continue here. So, this is how we are going to add the edges. Now, you know that this  $G$  is a directed graph. So, the edges that are going to be added here by using this add edge function are going to be directed edges. So, after this loop we would have added several edges, then we can return this graph. So, this is our add edge function and now we have created this graph  $G$  and we can go ahead.

So, we are done with the first point, let us implement the second one. It says assign 100 points to each node. So, we have to maintain a list points which will be storing the points for every node and we have to initialise this points list to 100. Let me create a function although you can directly write it, but I am going to create a function for this.

(Refer Slide Time: 07:35)



```
 9     if r <= p:
10         G.add_edge(i,j)
11     else:
12         continue
13
14
15
16
17
18
19
20 def main():
21     # Create/take a directed graph with `n` nodes.
22     G = nx.DiGraph()
23     G.add_nodes_from([i for i in range(10)])
24     G = add_edges(G, 0.3)
25
26     # Assign 100 points to each node.
27     points = initialize_points(G)
28
29     # Keep distributing points until convergence.
```

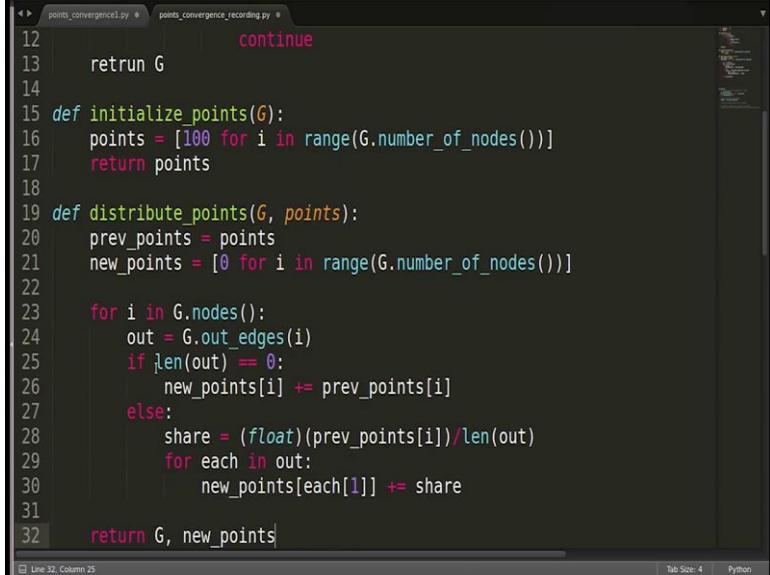
So, I will write points is equal to initialize points and I will pass the graph here ok. So, let me create this function ok. So, this function is just going to create a list points and that points list will be having 100 points for each node. So, I will write 100 for  $i$  in range, what is going to be the size of this list. And, the size of this list is going to be equal to the number of nodes that are there in the graph although we know that there are 10 nodes as of now, but we can always change that. So, let me generalized it by writing

`G.number_of_nodes` and then we should return this list points we are done ok. Let us go here; we have done the second point. So, every node has 100 points right now.

Let us go to third step, third step says keep distributing points until convergence. Now, how do we distribute\_points? Let us see how we do that. For that purpose, let me create a function `distribute_points`. So, I will write define `distribute_points`. So, this is going to be one iteration and we are going to distribute\_points in this function. As you know that after the nodes distribute their points their points change. So, we are going to need previous points and `new_points`.

Let us pass the graph here because, we are going to need that and let us pass the points that are there as of now. As I told you previous points and `new_points` let me create to let me create two lists here. So, I will write previous points to be the same as points and this is just for a ease of understanding. Another list I will create `new_points`, this is what we will be returning. So, this list I am going to initialize to 0 and during the distribution part, this list will get more points and that is what we will written.

(Refer Slide Time: 09:58)



```
12     continue
13     return G
14
15 def initialize_points(G):
16     points = [100 for i in range(G.number_of_nodes())]
17     return points
18
19 def distribute_points(G, points):
20     prev_points = points
21     new_points = [0 for i in range(G.number_of_nodes())]
22
23     for i in G.nodes():
24         out = G.out_edges(i)
25         if len(out) == 0:
26             new_points[i] += prev_points[i]
27         else:
28             share = (float)(prev_points[i])/len(out)
29             for each in out:
30                 new_points[each[1]] += share
31
32     return G, new_points
```

So, for now I will write 0 for `i in range G.number_of_nodes`. I have not copied `G.number_of_nodes` ok. So, this is the initialization of the `new_points` list which will be returned by this function. Now, how do we distribute? As I told you for every node, we will look at its out links. So, let me start a loop at these points, I write for `i in G.nodes`. Now, `i` is a node and we must distribute its points. How do we distribute? We look at its

out links right. So, let us keep a track of the out links of this node  $i$ , the function that we can use from networkx for this purpose is `out_edges`.

So, I will call that function I will write `G.out_edges`. Now, this function is there for directed graphs only, the parameter will be the node. So, we write `G.out_edges`. What this function does is, for the node  $i$  it will return a list of the edges which are `out_edges` from this node  $i$ . Now after this, this node  $i$  has to distribute its points and equal share of its points to its neighbours right. So, we have the points that this node  $i$  has in this list previous points right. We are going to distribute these points equally amongst its neighbours.

And the number of neighbours we can get by getting the list and by getting the length of this list. Now, what happens what will happen if the list is empty, that is the node has no out link. In that case, the length of this list will be 0. So, it is a good idea to check for that condition over here. So, let me check if the length of this out is 0 not, because in that case division by 0 exception might come. So, it is better to check it here.

So, I will write if length of out is equal to 0 which means this node has no out link ok. If a node has no out link, what should happen? It will not be distributed; it will not be distributing any of its points right. So, what we write is, `new_points` for  $i$  are going to be plus equal to I will tell you why I have written plus equal to previous points  $i$  ok. So, I have written plus equal to because, this point is going to retain its points it is not going to give its points to anybody because there is no out link. However, it might have some in links and through those in links it might be getting more points.

So, that is why we have to keep adding it. So, we have to write plus equal to. Now, this is what we will do in case a node has no out link. In the general case that is the nodes have out links, what are we going to do? We are going to distribute an equal share of these nodes points to its neighbours. Now, what is that share going to be? Let us create a variable `share`. Now, the share is going to be the number of points that are going to be distributed to every neighbour. What is this share going to be? So, as I told you if there are if there are 3 neighbours, the points the points that are going to be distributed is total divide by 3.

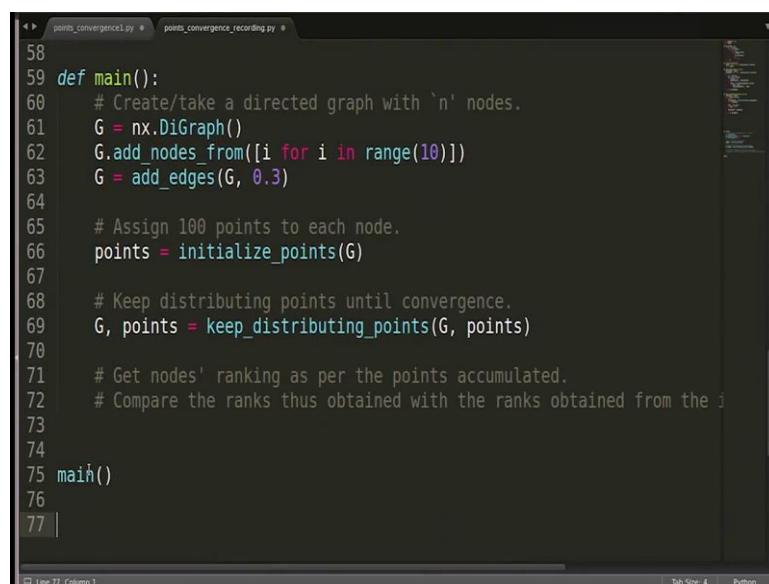
And what is the number of neighbours here? The number of neighbours is equal to the length of out here right. So, what are we going to do is we will write previous points that

is what is to be distributed right, you remember that you have to only distribute the previous points. So, previous points are to be distributed in equal share so, we divide by length of out this. So, this is what is to be distributed, let me sorry let me do the float casting here so, that we get the exact values ok.

So, this is a share that has to be distributed to the neighbours. Now, where are the neighbours? The neighbours are in this list; this list is a list of edges. So, for every edge 0th value is the source that is our node  $i$  and first value that is the value at index 1 is going to be the neighbour ok. So, let us start a loop here in order to distribute the points. So, I will write for each in out which means for every neighbour, we are going to update the new\_points of every neighbours. So, I will write new\_points for each now each is an edge.

So, I will write each 1, each 1 is going to be the neighbour. So, new\_points plus equal to share so, that is how we will be updating the points of every neighbour. I think we are done, let us return. What are we going to return? We are going to return the graph  $G$  and we are going to turn the list new\_points ok. So, we have created this function, this distributes points which is going to be one iteration where every node will be distributing its points. Let us go back to main and let us see what we have to do.

(Refer Slide Time: 15:22)



```
58
59 def main():
60     # Create/take a directed graph with `n` nodes.
61     G = nx.DiGraph()
62     G.add_nodes_from([i for i in range(10)])
63     G = add_edges(G, 0.3)
64
65     # Assign 100 points to each node.
66     points = initialize_points(G)
67
68     # Keep distributing points until convergence.
69     G, points = keep_distributing_points(G, points)
70
71     # Get nodes' ranking as per the points accumulated.
72     # Compare the ranks thus obtained with the ranks obtained from the i
73
74
75 main()
76
77 |
```

We have to keep distributing the points until the convergence. So, we have to basically keep calling this function distribute\_points. So, what we can do is let me create another

function which will keep calling this function that we have just created. Let us call this function keep distributing points and the parameter will be G and the points that we have and this function should return G and points again. So, we are passing something and that is getting updated and that is what we will be returned. Now, what should this function do? Keep distributing points, this function has to keep calling the function distribute\_points until we get the convergence ok. So, let us create this function ok.

(Refer Slide Time: 16:22)

```
33
34
35 def keep_distributing_points(G, points):
36     prev_points = points
37     print 'Enter # to stop'
38     while(1):
39         G, new_points = distribute_points(G, prev_points)
40         print new_points
41
42         char = raw_input()
43         if char == '#':
44             break
45         prev_points = new_points
46
47     return G, new_points
48
49
50
51
52
53
```

So, as I told you this function has to keep calling this distribute\_points function, let me start a while loop here. So, what I can write is while 1, they should keep going on ok. What should keep going on? Before that let me rename these previous points is equal to points, this is for the ease of variable name that is all ok. So, we are going to start this loop, this will keep going on and on. We have to call the function this distribute\_points.

So, I will copy it and I will call it here, this function returns two things again. So, I will write G, new\_points is equal to this right. What are we passing? We are passing previous points. So, let me rename these previous points done. So, we have called this function and one iteration is happened and we have got the new\_points in this list new\_points. If we want to keep track let us let us print this so, that we can see the updated points ok.

Now, this will keep going on and on, it has to stop somewhere basically when the convergence happens. So, you can do one thing, or you can do one or the two things, you can either through your code check whether the convergence has happened or not that is

one thing. And, other simple thing that we can do is we can keep observing as we are printing it right. So, we can keep observing these points and whenever we see that convergence has happened, we can stop it.

So, for that let us let us ask the user to stop it, enter may be # to stop and let us ask the user to enter a hash when he wants it to stop. So, we will take that through the function may be raw input and if that char is equal to # we are going to stop. So, I will write break ok. Now, after one iteration, we have to assign the new\_points to be the previous points and we have to repeat the process. So, it is important to write previous points is equal to new\_points and after that we will go back here, and the same thing will start alright.

I think we have done, let us return let us return the same two things G and new\_points. I think this function should work ok, we can, may be check the code as well if it is working fine, wait there is an error here ok. So, let us check the functionality of this code whether it is working fine or not. So, let me call main here ok.

(Refer Slide Time: 19:34)

```
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_6_(Link Analysis)/Code/  
Video_code$ python points_convergence_recording.py  
Enter # to stop  
[33.33333333333336, 58.33333333333336, 58.33333333333334, 33.333333333333  
36, 166.66666666666669, 116.66666666666667, 116.66666666666667, 158.3333333  
33334, 75.0, 183.333333333334]  
  
[38.8888888888889, 70.1388888888889, 46.52777777777778, 31.94444444444446,  
195.8333333333334, 126.388888888889, 120.8333333333334, 190.97222222222  
23, 79.86111111111111, 98.61111111111111]  
  
[42.12962962962963, 76.9097222222223, 55.84490740740741, 38.31018518518519,  
167.24537037037038, 84.95370370370371, 141.087962962963, 206.0763888888889,  
85.59027777777779, 101.851851851851]  
  
[28.317901234567902, 69.70968364197532, 60.595100308641975, 41.3676697530864  
25, 161.23649691358025, 91.13618827160494, 137.7411265432099, 218.2339891975  
3092, 76.62519290123457, 115.03665123456791]
```

Let us get back to our screen and let us try calling this file ok. So, these are the points, we did not print the initial points right. They were obviously, 100 100 each we can print that as well. So, this is these are the values that we are getting after one iteration.

And we have to press enter after it, after I press enter the next the next iteration starts. And, as you can see the points are changing the first node was earlier having 33 points

and now it is having 38 points. I am keeping on pressing enter and we are seeing the points how they are changing.

(Refer Slide Time: 20:12)

```
[31.58367769325456, 70.08542734973662, 57.65617335916762, 40.15481123715133,  
166.98780936924356, 94.70919693251918, 133.53098480321384, 209.376402517846  
54, 80.91890968419077, 114.99660705367603]  
  
[31.569732310839726, 70.0766464628731, 57.68137020595101, 40.16001336851685,  
166.97913346467644, 94.69764998627318, 133.5737007500799, 209.4233153473427  
4, 80.8798427369394, 114.95859536650778]  
  
[31.565883328757725, 70.0800537063799, 57.68466955874868, 40.165507943030406  
, 166.94981125499515, 94.68042390226654, 133.56434139787567, 209.45337258920  
688, 80.89011177146926, 114.96582454726986]  
  
[31.560141300755514, 70.07110447468555, 57.68988941292241, 40.16987598632743  
, 166.95547093018016, 94.68507364471003, 133.5623544407881, 209.436806425741  
37, 80.89020412197718, 114.97907926191232]  
  
[31.561691214903345, 70.07429599662399, 57.683934073087535, 40.1661579544161  
5, 166.95771594944134, 94.68981165298408, 133.55830874697523, 209.4367291146  
7579, 80.89029797242418, 114.98105732446848]
```

So, I keep on pressing the enter ok, as you can see we are sword of reaching the convergence.

(Refer Slide Time: 20:21)

```
[31.565883328757725, 70.0800537063799, 57.68466955874868, 40.165507943030406  
, 166.94981125499515, 94.68042390226654, 133.56434139787567, 209.45337258920  
688, 80.89011177146926, 114.96582454726986]  
  
[31.560141300755514, 70.07110447468555, 57.68988941292241, 40.16987598632743  
, 166.95547093018016, 94.68507364471003, 133.5623544407881, 209.436806425741  
37, 80.89020412197718, 114.97907926191232]  
  
[31.561691214903345, 70.07429599662399, 57.683934073087535, 40.1661579544161  
5, 166.95771594944134, 94.68981165298408, 133.55830874697523, 209.4367291146  
7579, 80.89029797242418, 114.98105732446848]  
  
[31.56327055099469, 70.073555501419, 57.68497738741918, 40.16640338826319, 1  
66.96135109375416, 94.69036804881267, 133.56012386367235, 209.43345522322838  
, 80.89037558837921, 114.97611935405726]  
  
[31.56345601627089, 70.0750280447285, 57.68450983772526, 40.16612096237051,  
166.95861942811962, 94.68823799107062, 133.56088743273813, 209.436501458685  
2, 80.89027623366374, 114.97636259458332]
```

So, it may stop, it may happen that as you keep pressing enter it will not change at all. So, nothing will change, or it may happen that values are changing by very nominal decimal points. So, you can stop wherever you want and or you can go on and on. So, it

is up to you. So, I am keeping on pressing enter and I can see the values changing slightly only by decimal points. So, I can keep going on and on or I can stop.

(Refer Slide Time: 20:53)

```
[166.95891578309104, 94.6887137215057, 133.56057750610447, 209.4360957703087,  
80.89018534602846, 114.97681126169684]  
  
[31.562904573835237, 70.07423673817209, 57.685059241900376, 40.1665000573573  
3, 166.95891578309113, 94.68871372150575, 133.56057750610452, 209.4360957703  
086, 80.89018534602843, 114.97681126169681]  
  
[31.56290457383525, 70.07423673817215, 57.68505924190033, 40.16650005735731,  
166.95891578309107, 94.68871372150574, 133.5605775061045, 209.4360957703086  
7, 80.89018534602846, 114.97681126169685]  
  
[31.562904573835244, 70.0742367381721, 57.685059241900355, 40.16650005735732  
, 166.95891578309116, 94.68871372150576, 133.5605775061045, 209.436095770308  
64, 80.89018534602843, 114.97681126169684]  
  
[31.562904573835254, 70.07423673817215, 57.68505924190033, 40.16650005735731  
, 166.95891578309113, 94.68871372150574, 133.5605775061045, 209.436095770308  
67, 80.89018534602845, 114.97681126169682]  
#  
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_6_(Link Analysis)/Code/  
Video_code$ █
```

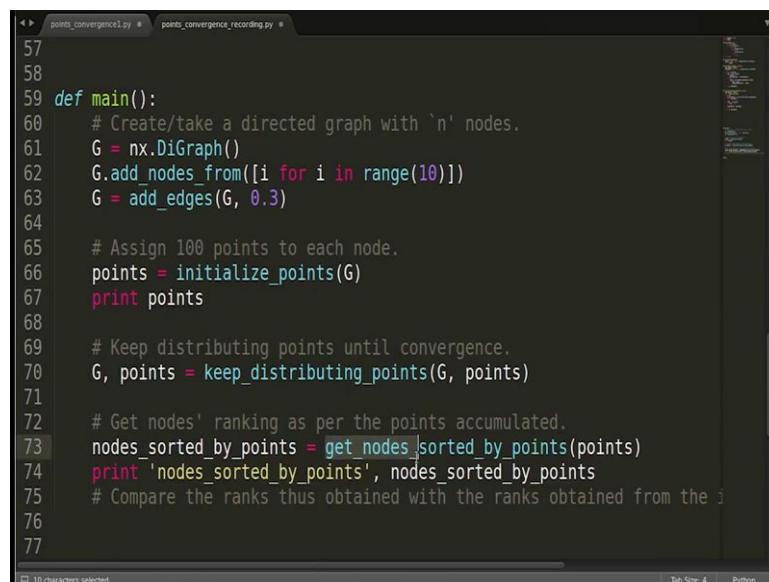
So, I want to stop let me press hash and we are stopping. So, these are the points distribution that we are getting, and we are sort of seeing the convergence taking place as well. Now, let us get back to our code and let us see what is pending ok, next step is to get the nodes ranking as per the points accumulated. So, we have to basically rank the nodes based on the points that they have acquired overtime after number of iterations, after the convergence point.

So, I think we will do that in the next video.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Prof. Anamika Chhabra**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**  
**Link Analysis**

**Lecture - 82**  
**Implementing Page Rank Using Points Distribution Method – 3**

(Refer Slide Time: 00:07)

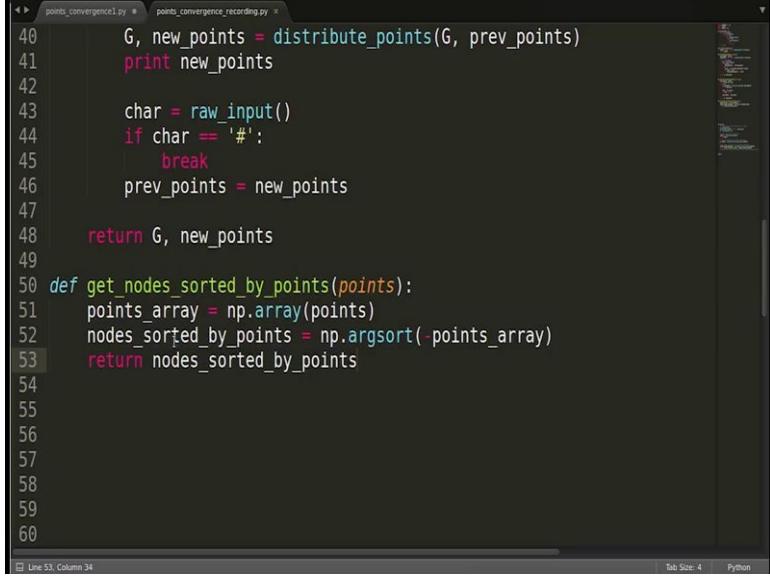


```
57
58
59 def main():
60     # Create/take a directed graph with 'n' nodes.
61     G = nx.DiGraph()
62     G.add_nodes_from([i for i in range(10)])
63     G = add_edges(G, 0.3)
64
65     # Assign 100 points to each node.
66     points = initialize_points(G)
67     print points
68
69     # Keep distributing points until convergence.
70     G, points = keep_distributing_points(G, points)
71
72     # Get nodes' ranking as per the points accumulated.
73     nodes_sorted_by_points = get_nodes_sorted_by_points(points)
74     print 'nodes_sorted_by_points', nodes_sorted_by_points
75     # Compare the ranks thus obtained with the ranks obtained from the j
76
77
```

In the previous video, we saw the conversions taking place when it comes to the points that the nodes are able gather in every iteration. So, we saw that the points are converging. Now, the next step is how we can rank the nodes based on the points that they have gathered. Now, let us create a function to sort the nodes based on points. So, let me create get nodes sorted by points, I am going to pass the points list and it should return me the list of nodes. I am sorry I am sorry ok.

So, I will write `nodes_sorted_by_points = get_nodes_sorted_by_points` which is a function which I am going to create. And, then let me print the nodes that we will be getting. So, we will just be printing the nodes that we are getting, I should print the initial points as well. So, that we know how the points are changing. So, I am printing the initial points here ok, getting back to the function that you have to create `get_nodes_sorted_by_points`. Let us create this function here. Sorry ok.

(Refer Slide Time: 01:27)



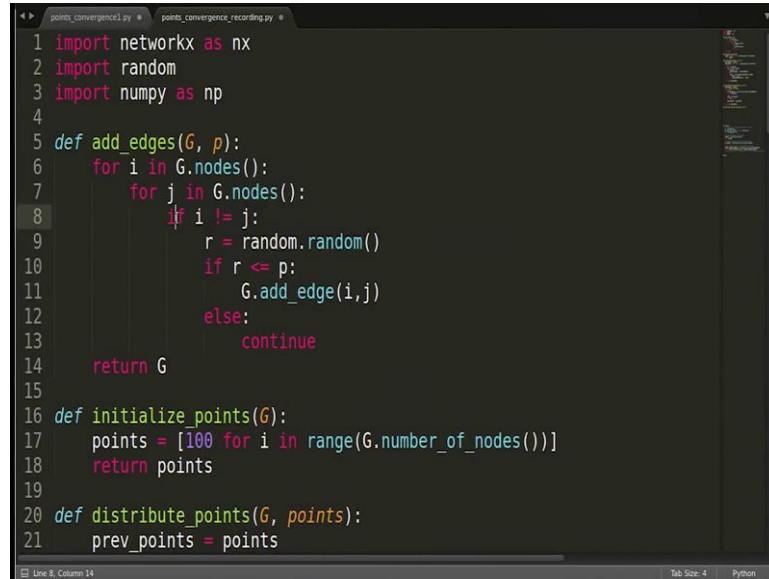
A screenshot of a code editor showing a Python script named `points_convergence_recording.py`. The code contains two functions: `distribute_points` and `get_nodes_sorted_by_points`. The `distribute_points` function takes a graph `G` and a list of previous points `prev_points`, prints the new points, and then reads a character from input. If the character is '#', it breaks the loop. Otherwise, it updates `prev_points` to `new_points` and returns `G` and `new_points`. The `get_nodes_sorted_by_points` function takes a list of points `points`, converts it to a NumPy array, and then uses `argsort` to get the indices sorted by the values. The code editor interface shows tabs for `points_convergence1.py` and `points_convergence_recording.py`, and a status bar indicating "Line 53, Column 34" and "Tab Size: 4 Python".

```
40     G, new_points = distribute_points(G, prev_points)
41     print new_points
42
43     char = raw_input()
44     if char == '#':
45         break
46     prev_points = new_points
47
48 return G, new_points
49
50 def get_nodes_sorted_by_points(points):
51     points_array = np.array(points)
52     nodes_sorted_by_points = np.argsort(-points_array)
53     return nodes_sorted_by_points
54
55
56
57
58
59
60
```

Now, what should this function do? This function has as an input a list which is `points` which is a list having the points for every node. And, what should this function do? This function should return a list of the nodes sorted by their points. Now, what are the nodes? How are the nodes labeled? The nodes are basically from as you can see here from 0 to 9 right; that is how the nodes are numbered.

So, we can just simply get the index of the nodes from this `points` list. So, basically what we have to do is, we have to sort the index of the values which are there in the `points` right. We can use again several methods for that, one method which might be very quick is we can make use of the package `numpy`. So, `numpy` has function `argsort` which basically returns the index, the indices sorted by the values of a list. So, we are going to use that function `argsort` that function is there in `numpy`.

(Refer Slide Time: 02:49)



```
1 import networkx as nx
2 import random
3 import numpy as np
4
5 def add_edges(G, p):
6     for i in G.nodes():
7         for j in G.nodes():
8             if i != j:
9                 r = random.random()
10                if r <= p:
11                    G.add_edge(i,j)
12                else:
13                    continue
14
15    return G
16
17 def initialize_points(G):
18     points = [100 for i in range(G.number_of_nodes())]
19
20 def distribute_points(G, points):
21     prev_points = points
```

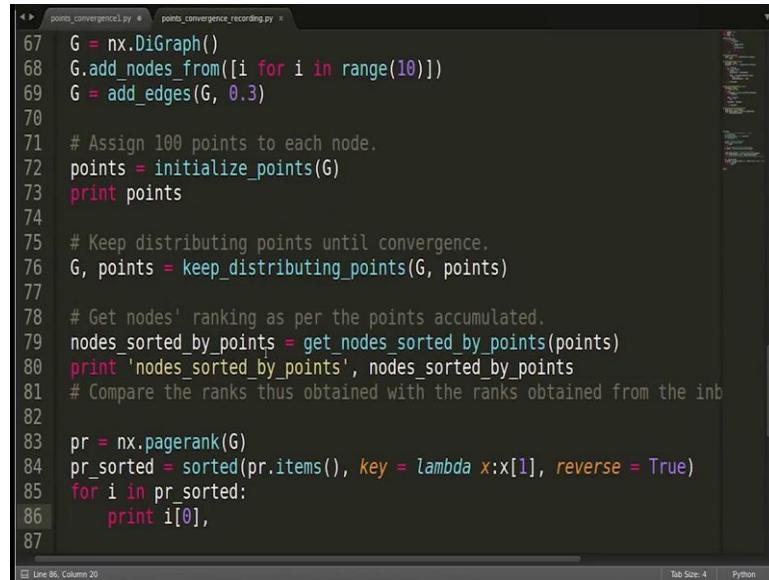
So, let me import that package. So, I will write import numpy as np ok. So, the input to wherever we yeah here so, the input to that function is basically an array which is a numpy array and here as an input we have a points which is a list. So, we have to convert that list into a numpy array. Now, how we do that? So, I will write points array is equal to np numpy.array. So, so this array is the function which converts a list into an numpy array ok. Now, points array is a numpy array and we can apply the function argsort on it. And, what is that function do?

It returns the indices sorted by the values of the list. So, write np numpy.argsort points array ok. What is it return? It returns the indices. So, indices basically are the nodes in our case. So, maybe I can write nodes\_sorted\_by\_points equal to this thing ok. Now, one thing that has to be noted is by default argsort sorts in the ascending order right. And, which order do you want? We want descending order that is the node which has high points should be the first to be ranked ok; so, simple thing that we can do here just put a minus here.

By putting a minus all the values are converted to negative and the ordering that we will be getting will be negative. So, as simple as that we can work with that simple method. So, this nodes\_sorted\_by\_points is going to have the nodes sorted by of course, the points in descending order right. So, the first value will be the node which is having

highest ranking and hence the highest page rank ok. Let us return this nodes\_sorted\_by\_points, this should work.

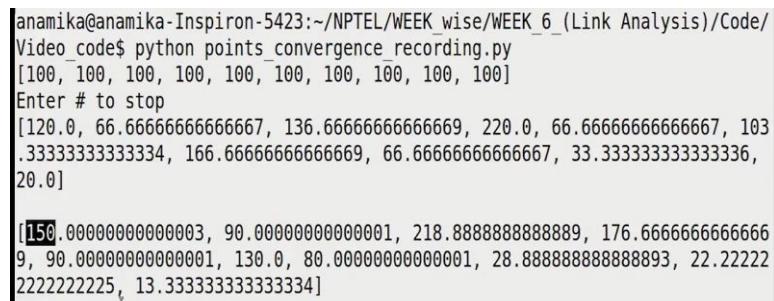
(Refer Slide Time: 05:07)



```
67 G = nx.DiGraph()
68 G.add_nodes_from([i for i in range(10)])
69 G = add_edges(G, 0.3)
70
71 # Assign 100 points to each node.
72 points = initialize_points(G)
73 print points
74
75 # Keep distributing points until convergence.
76 G, points = keep_distributing_points(G, points)
77
78 # Get nodes' ranking as per the points accumulated.
79 nodes_sorted_by_points = get_nodes_sorted_by_points(points)
80 print 'nodes_sorted_by_points', nodes_sorted_by_points
81 # Compare the ranks thus obtained with the ranks obtained from the inb
82
83 pr = nx.pagerank(G)
84 pr_sorted = sorted(pr.items(), key = lambda x:x[1], reverse = True)
85 for i in pr_sorted:
86     print i[0],
87
```

So, let us go back to main and let us see is anything pending. So, we had already called this function and we had already printed the values. So, let us see whether this function works fine or not ok.

(Refer Slide Time: 05:18)



```
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_6_(Link Analysis)/Code/
Video_code$ python points_convergence_recording.py
[100, 100, 100, 100, 100, 100, 100, 100, 100]
Enter # to stop
[120.0, 66.66666666666667, 136.66666666666669, 220.0, 66.66666666666667, 103
.333333333334, 166.6666666666669, 66.66666666666667, 33.3333333333336,
20.0]

[150.0000000000003, 90.0000000000001, 218.888888888889, 176.66666666666666
9, 90.0000000000001, 130.0, 80.0000000000001, 28.8888888888893, 22.22222
222222225, 13.3333333333334]
```

So, let us call this function, let us let us see how the code works. So, let me execute this ok. So, these are the initial points I am sorry these are the initial points 100 each, as we

assigned initially. And then we started the iterations, after the first iteration as you can see the first node which was having 100 points is now having 120 points and second node the points for second node have reduced and so on. So, that that keeps changing; this is the scenario after first iteration. I press enter and this is this scenario after second iteration. So, the points have 150 for the first node, I keep on pressing enter.

(Refer Slide Time: 06:01)

```
[150.00000000000003, 90.00000000000001, 218.888888888889, 176.66666666666666  
9, 90.00000000000001, 130.0, 80.00000000000001, 28.8888888888893, 22.2222  
222222225, 13.3333333333334]  
[236.888888888889, 70.0, 183.0, 170.222222222223, 70.0, 110.77777777777777  
9, 76.66666666666667, 34.44444444444445, 30.00000000000004, 18.000000000000  
004]  
[197.0, 62.48148148148149, 203.62962962962962, 187.66666666666669, 62.481481  
48148149, 105.1111111111111, 114.96296296296298, 29.33333333333332, 23.333  
33333333332, 14.0]  
[216.1259259259259, 73.35802469135803, 210.31728395061728, 163.3604938271605  
2, 73.35802469135803, 110.99629629629631, 93.66666666666667, 25.493827160493  
83, 20.82716049382716, 12.496296296296297]  
[224.9888888888888, 68.220987654321, 199.61604938271606, 173.6588477366255,  
68.220987654321, 100.5172839506173, 97.03456790123457, 28.618106995884773,  
24.45267489711934, 14.671604938271605]
```

And I keep on seeing the values changing as you can see. So, let us concentrate on the first node as of now. So, 224 I keep pressing enter because it is keeping on changing it is 2219.

(Refer Slide Time: 06:14)

```
[219.39556973427028, 68.033045972233, 205.78146969744535, 171.19545489990605  
, 68.033045972233, 103.74098876662828, 100.33045574218573, 27.20960588577329  
6, 22.675227080828137, 13.605136248496882]  
  
[219.38807889189195, 68.02381483627133, 205.77967846988497, 171.206083237930  
41, 68.02381483627133, 103.73938206056526, 100.34212907508386, 27.2127274069  
0996, 22.677681990744333, 13.6066091944466]  
  
[219.38444143713923, 68.02717037854971, 205.7845405752114, 171.2012489704070  
5, 68.02717037854971, 103.74334098436833, 100.34257801952384, 27.21014134357  
2644, 22.674604945423777, 13.604762967254265]  
  
[219.38997465092135, 68.02863966796406, 205.7817317131345, 171.2005596859683  
3, 68.02863966796406, 103.74097954999822, 100.3376730802216, 27.210644448601  
33, 22.675723459516572, 13.605434075709942]  
  
[219.3874596467273, 68.0262175434066, 205.78189035362462, 171.202903671822,  
68.0262175434066, 103.7411524684803, 100.340859701727, 27.211357914558, 22.6  
76213222654685, 13.605727933592812]
```

And only the decimal values are changing ok.

(Refer Slide Time: 06:20)

```
[219.38772275683706, 68.02723014132114, 205.78234360369206, 171.201791592383  
6, 68.02723014132114, 103.7415191726658, 100.3401506994707, 27.2108670143670  
75, 22.675715548713345, 13.605429329228008]  
  
[219.38778963195628, 68.02722329071216, 205.78229964322526, 171.201803732905  
94, 68.02722329071216, 103.74148493419868, 100.34009957740169, 27.2108864901  
8305, 22.67574338044038, 13.605446028264229]  
  
[219.3877443013677, 68.02719483720011, 205.78230959438142, 171.2018304339478  
6, 68.02719483720011, 103.74149520068347, 100.34015526718055, 27.21088977299  
213, 22.67574109690405, 13.605444658142432]  
  
[219.38774856182144, 68.02721682262134, 205.78232070726335, 171.201806853515  
9, 68.02721682262134, 103.74150240379477, 100.34013741674076, 27.21087983178  
085, 22.67573161240004, 13.605438967440023]  
  
[219.38776407178761, 68.02721327351185, 205.7823087841363, 171.2018124590509  
8, 68.02721327351185, 103.74149311376222, 100.34012745548719, 27.21088526335  
3788, 22.67573894087378, 13.605443364524268]
```

Let us let us look at any other node if the values are changing drastically. So, if the values are not at all changing, we will stop of course, if the values are changing slightly, we can either keep going on and on or we can stop there. So, that that should be enough and that is what I am mean to say; let us see if any other value is changing. I am giving on pressing enter and the values are only slightly changing. So, I think I can stop at any moment, let me press hash here.

(Refer Slide Time: 06:46)

```
[484, 22.67573696145134, 13.605442176870804]  
[219.38775510204064, 68.02721088435334, 205.78231292516995, 171.201814058957  
32, 68.02721088435334, 103.74149659863939, 100.34013605442216, 27.2108843537  
41655, 22.67573696145139, 13.605442176870833]  
[219.38775510204061, 68.02721088435385, 205.78231292517026, 171.201814058956  
76, 68.02721088435385, 103.74149659863961, 100.34013605442189, 27.2108843537  
4139, 22.67573696145111, 13.605442176870667]  
[219.38775510204104, 68.02721088435383, 205.78231292516998, 171.201814058956  
84, 68.02721088435383, 103.74149659863937, 100.34013605442154, 27.2108843537  
41505, 22.675736961451282, 13.60544217687077]  
[219.38775510204073, 68.02721088435364, 205.78231292517003, 171.201814058957  
, 68.02721088435364, 103.74149659863944, 100.34013605442188, 27.210884353741  
534, 22.67573696145128, 13.605442176870767]  
#  
nodes_sorted_by_points [0 2 3 5 6 1 4 7 8 9]  
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_6_(Link Analysis)/Code/  
Video_code$
```

So, our loops stops ok, this function is looking fine. So, I getting the nodes\_sorted\_by\_points, these are the sorted nodes that we are getting. Basically, these are rankings; this is the ranking of the nodes. So, the node 0 is having the highest page rank node 2 is on the second number and so on. So, this is the ranking that we obtained. Now, let us see whether this ranking is correct or not as per the page rank; by third that is there in networkx. So, let us get back get to the last step which is compare the ranks thus, obtained the ranks obtained with the inbuilt page rank method.

Now, let me tell you there is an inbuilt function which we can use to compute the page rank values for the nodes for a given graph. We have ourselves implemented it over here, there are several other methods as well let me tell you. We have implemented one of the methods which is using the distribution of the points. We might look at other methods as well maybe in the subsequent videos; let us check the ranking that we obtained from page rank. So, the function is nx.pagerank and we will pass the graph here ok.

Let me tell you what it returns, it basically returns a dictionary where the keys are the nodes and the values are the page ranks of the nodes. So, basically it does not tell the ranking of the nodes, it tells us the page ranks page rank values of each node. So, we would have to rank them ourselves. So, how can we do that? Let me sort this dictionary ok, we cannot sort the dictionary because the dictionary does not contain any sorting

order. So, we can create a list of the tuples from the dictionary as we have been doing in the previous videos as well.

So, how can we do that? Let us see so, this is going to be a list of the sorted tuples based on the values. So, the function that we can use is sorted, I write pr.items and same old method that we use key is equal to lambda. How can how do we want to sort it? We want to sort it based on the value which is the second thing in the tuple which is x1. So, we will write x : x1 and by default it is a ascending right and we want it to be descending.

So, we will write reverse is equal to True. So, pr sorted is a list of the tuples ok. What we want? We want the nodes; we want the nodes sorted by the page rank value only. So, maybe what we can do is, we can only print the first value from each tuple. So, write for i in pr\_sorted ok, I am going to only print the first value for every i ok. So, I want all the values in same line so, I have put a comma here ok. Let us check the working of the code. So, basically what we have to check is the ranking that we were getting r by r method and the ranking that we are getting by this method are the same or not.

(Refer Slide Time: 10:20)

```
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_6_(Link Analysis)/Code/  
Video_code$ python points_convergence_recording.py  
[100, 100, 100, 100, 100, 100, 100, 100, 100, 100]  
Enter # to stop  
[50.0, 133.333333333334, 66.66666666666667, 116.66666666666669, 183.333333  
3333334, 83.333333333334, 116.66666666666669, 0, 33.3333333333336, 216.  
66666666666669]  
  
[108.333333333334, 116.66666666666669, 27.7777777777782, 111.1111111111  
113, 211.111111111114, 113.888888888889, 105.55555555555557, 0, 0.0, 205.  
55555555555557]
```

Let us check alright. So, these are the initial values 100 each and after first iteration these are the values that we get. We got 50 points for the first node now, I am pressing enter and the values are changing ok. So, the values are changing.

(Refer Slide Time: 10:41)

```
[106.78213447711533, 97.02664487720955, 38.820685788002365, 123.006478422182  
84, 207.07357587026456, 116.53929027684927, 97.13663886767405, 0, 0.0, 213.6  
145514207021]  
  
[106.80727571035105, 97.13663886767405, 38.84643009228309, 122.9537973565929  
9, 207.15694501372556, 116.47701653113307, 97.04759586031068, 0, 0.0, 213.57  
430056792958]  
  
[106.78715028396479, 97.04759586031068, 38.82567217704436, 122.9964168476650  
7, 207.08972113930565, 116.52728253762382, 97.11955470138173, 0, 0.0, 213.60  
6606452704]  
  
[106.803303226352, 97.11955470138173, 38.84242751254127, 122.96194220401821,  
207.14393916272581, 116.48675129533427, 97.0614054171584, 0, 0.0, 213.58067  
648048842]  
  
[106.79033824024421, 97.0614054171584, 38.828917098444755, 122.9897955245862  
7, 207.10022644069807, 116.51944541887667, 97.10835426365529, 0, 0.0, 213.60  
151759633644]
```

Let us see, for first node has 106, only the decimal values are changing. And, let us check the second one, second one is also slightly changing third, fourth I think all the nodes are only slightly changing. So, we can stop at this moment or maybe you can go ahead a little more ok. I am just keeping on pressing enter; it is just changing very slightly. It might take number of iterations to stop. So, maybe we can press hash and stop at this point.

(Refer Slide Time: 11:15)

```
[106.79611650485441, 97.08737864077673, 38.8349514563107, 122.97734627831719  
, 207.11974110032372, 116.50485436893207, 97.0873786407767, 0, 0.0, 213.5922  
3300970882]  
  
[106.79611650485441, 97.0873786407767, 38.83495145631069, 122.9773462783172,  
207.1197411003237, 116.50485436893209, 97.08737864077673, 0, 0.0, 213.59223  
300970882]  
  
[106.79611650485441, 97.08737864077673, 38.8349514563107, 122.97734627831719  
, 207.11974110032372, 116.50485436893207, 97.0873786407767, 0, 0.0, 213.5922  
3300970882]  
  
[106.79611650485441, 97.0873786407767, 38.83495145631069, 122.9773462783172,  
207.1197411003237, 116.50485436893209, 97.08737864077673, 0, 0.0, 213.59223  
300970882]  
#  
nodes_sorted_by_points [9 4 3 5 0 6 1 2 7 8]  
9 4 3 5 1 6 0 2 8 7  
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_6_(Link Analysis)/Code/  
Video_code$ python points_convergence_recording.py
```

This is something that I want to tell you. So, this is the first thing that you can see is the ranking that we got by our method ok. And, the second row is the values that we got by the page rank method from networkx. Now, let us compare 9 4 3 5 is correct after that 0 6 1 2 7 8. So, we can see some changes here.

This is what I wanted to show you, why these changes are coming and how we can handle that ok. So, can you imagine what can happen if there is a node which has no out link. So, as we also implemented if a node has no out link, this node does not distribute its points. So, in every iteration, what happens it keeps on changing; it keeps on getting points ok. It keeps on getting point from other nodes and it never distributes its own points. So, what happens is that the point sink happens.

Sink basically all the points are getting accumulated at 1 node or a set of nodes. So, this example I gave for 1 node, it may also happen that in the graph that we generated there are a bunch of nodes which are not distributing any points outside. So, basically there is a link from 1 node to the other and there is a link from second node to the first node, but there is no link outside. So, these 2 nodes will keep accumulating all the points and that is how the ranking that we will get will not be accurate.

So, for that we need to take some measures, we must handle that case. The networkx function is basically handling those cases whereas; our function is not handling those cases; that is why we are getting these differences in the ranking. Now, we are going to handle these cases. Let me tell you it will not happen in all the cases, because in all the cases your graph might not be having such sinks right. Let us check one more example, if we are getting the same. So, let us check for the convergence first.

(Refer Slide Time: 13:31)

```
[77.89855072463769, 64.31159420289856, 38.94927536231884, 74.72826086956522,
85.14492753623189, 181.15942028985506, 83.78623188405797, 144.9275362318840
6, 106.43115942028987, 142.66304347826087]
#
nodes sorted by points [5 7 9 8 4 6 0 3 1 2]
5 7 9 8 4 6 0 3 1 2
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_6_(Link Analysis)/Code/
Video_code$ anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_6_(Link Analysis)/Code/
Video_code$ python points_convergence_recording.py
[100, 100, 100, 100, 100, 100, 100, 100, 100, 100]
Enter # to stop
[58.33333333333336, 100.0000000000001, 133.3333333333334, 58.33333333333
336, 66.66666666666667, 41.66666666666667, 100.0000000000001, 166.666666666
66669, 241.6666666666666, 33.33333333333336]

[47.91666666666667, 59.02777777777786, 90.97222222222223, 47.22222222222223
, 72.2222222222223, 30.55555555555556, 152.77777777777777, 162.5, 256.25, 8
0.55555555555556]
```

Again the values are keeping changing ok, now I am pressing enter and it is not going dead basically the values are not at all changing. So, it is completely stopped, there is no change now I am keeping on pressing enter. So, we can just press hash, let us see the ranking that we getting we are getting 5 7 9 8 4 6 0 3 1 2 ok, that is again another thing I want to show. So, this might be a graph where there is no such problem as I just explained you there is no sink.

There is no 1 node or a set of nodes which are keeping on accumulating all the points and they are not distributing that case is not here and that is why we are getting the same thing here. So, our method is accurate just in case there is no sink there in the graph ok. Let me just take another case and I am sure there will be some problem here. I actually want some problem to happen so, that the next video we can cover that because, we are going to handle such cases.

(Refer Slide Time: 14:28)

```
[39.937061936775855, 78.95866113574596, 88.05607209269061, 54.47003289944214
5, 57.84580174510085, 49.00586468316407, 140.7523959376341, 159.347732799313
4, 248.71978257759977, 82.90659419253326]

[39.937061936775855, 78.95866113574596, 88.05607209269061, 54.47003289944214
5, 57.84580174510085, 49.00586468316407, 140.7523959376341, 159.347732799313
4, 248.71978257759977, 82.90659419253326]

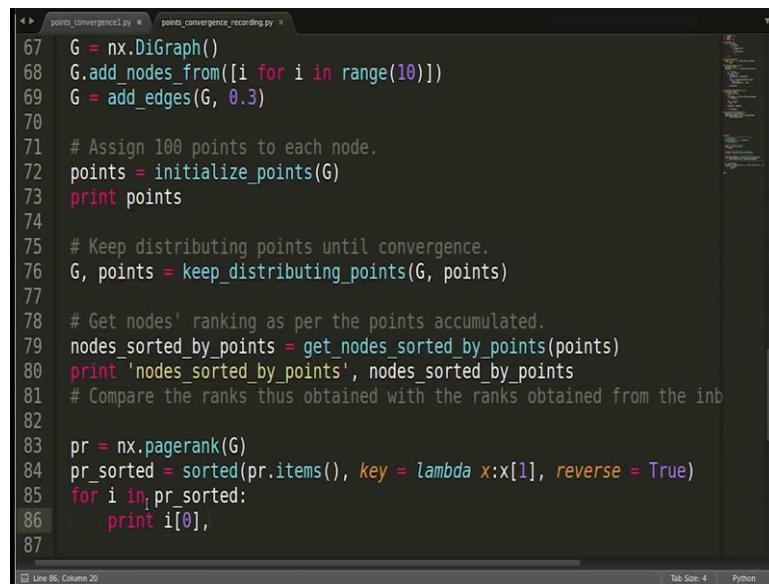
[39.937061936775855, 78.95866113574596, 88.05607209269061, 54.47003289944214
5, 57.84580174510085, 49.00586468316407, 140.7523959376341, 159.347732799313
4, 248.71978257759977, 82.90659419253326]

[39.937061936775855, 78.95866113574596, 88.05607209269061, 54.47003289944214
5, 57.84580174510085, 49.00586468316407, 140.7523959376341, 159.347732799313
4, 248.71978257759977, 82.90659419253326]
#
nodes_sorted_by_points [8 7 6 2 9 1 4 3 5 0]
8 7 6 2 1 9 4 3 5 0
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_6_(Link Analysis)/Code/
Video_code$
```

I am again pressing the enter and it is not going ahead, it is stopped. So, 8 7 6 2 1 you pick 2 1. Yeah, we got the difference nice. So, 2 9 1 and 2 1 9 so, we got a difference right; let me tell you one more thing sometimes there might be a slight you know variation for another reason as well. And, that another reason could be that 2 nodes are having exactly same page rank values right.

So, in that case your method can rank them in a different order and page rank method from networkx for example, can rank them in a different order. So, that sort of difference can happen even if your method is completely working fine and taking care of the sink cases as well. So, that you can just ignore, but as of now we know that our code is not taking care of the sink cases. So, we need to do that ok. So, let us get back to our code.

(Refer Slide Time: 15:30)



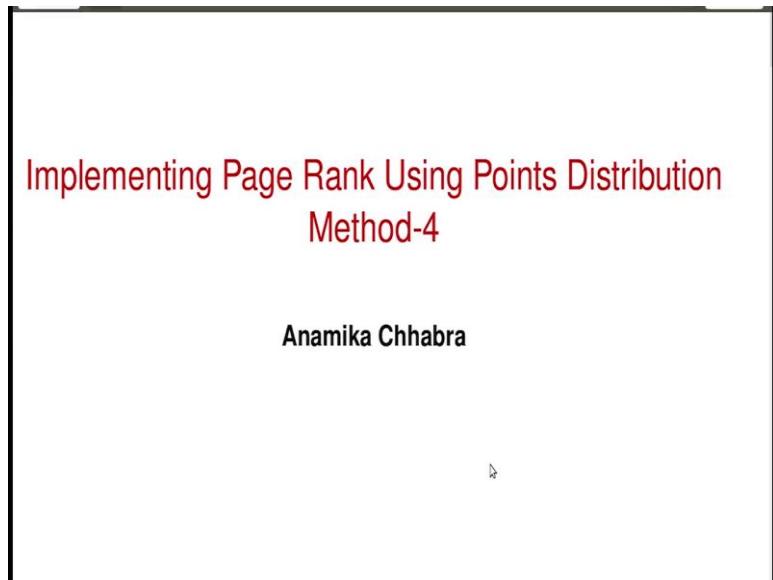
```
67 G = nx.DiGraph()
68 G.add_nodes_from([i for i in range(10)])
69 G = add_edges(G, 0.3)
70
71 # Assign 100 points to each node.
72 points = initialize_points(G)
73 print points
74
75 # Keep distributing points until convergence.
76 G, points = keep_distributing_points(G, points)
77
78 # Get nodes' ranking as per the points accumulated.
79 nodes_sorted_by_points = get_nodes_sorted_by_points(points)
80 print 'nodes_sorted_by_points', nodes_sorted_by_points
81 # Compare the ranks thus obtained with the ranks obtained from the inb
82
83 pr = nx.pagerank(G)
84 pr_sorted = sorted(pr.items(), key = lambda x:x[1], reverse = True)
85 for i in pr_sorted:
86     print i[0],
87
```

And in the next video we will be handling that case.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Prof. Anamika Chhabra**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Link Analysis**  
**Lecture – 83**  
**Implementing Page Rank Using Points Distribution Method – 4**

(Refer Slide Time: 00:06)



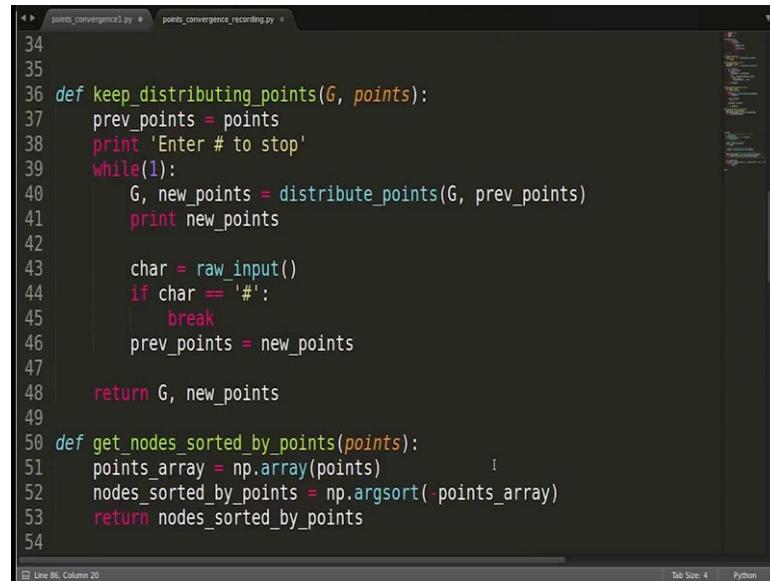
**Implementing Page Rank Using Points Distribution  
Method-4**

Anamika Chhabra

In the previous video, we finished Implementing Page Rank using Points Distribution Method and it was in most of the cases coming same as the ranking that the networkx a function page rank provides us. However, in some cases the ranking was not similar; the reason for that was that our code was not handling the cases, where one node or a set of nodes have no out links. So, in that case all the points get accumulated in that node or that set of nodes. So, our code has to handle those cases. That is what we will be doing in this video.

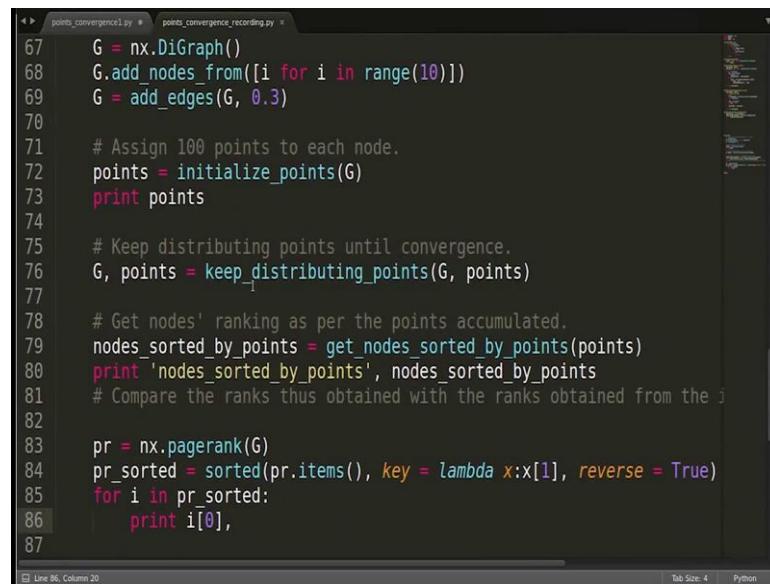
Let us get back to our code and see what kinds of measures are required to handle those cases.

(Refer Slide Time: 00:51)



```
34
35
36 def keep_distributing_points(G, points):
37     prev_points = points
38     print 'Enter # to stop'
39     while(1):
40         G, new_points = distribute_points(G, prev_points)
41         print new_points
42
43         char = raw_input()
44         if char == '#':
45             break
46         prev_points = new_points
47
48     return G, new_points
49
50 def get_nodes_sorted_by_points(points):
51     points_array = np.array(points)
52     nodes_sorted_by_points = np.argsort(-points_array)
53     return nodes_sorted_by_points
54
```

(Refer Slide Time: 00:56)



```
67     G = nx.DiGraph()
68     G.add_nodes_from([i for i in range(10)])
69     G = add_edges(G, 0.3)
70
71     # Assign 100 points to each node.
72     points = initialize_points(G)
73     print points
74
75     # Keep distributing points until convergence.
76     G, points = keep_distributing_points(G, points)
77
78     # Get nodes' ranking as per the points accumulated.
79     nodes_sorted_by_points = get_nodes_sorted_by_points(points)
80     print 'nodes_sorted_by_points', nodes_sorted_by_points
81     # Compare the ranks thus obtained with the ranks obtained from the i
82
83     pr = nx.pagerank(G)
84     pr_sorted = sorted(pr.items(), key = lambda x:x[1], reverse = True)
85     for i in pr_sorted:
86         print i[0],
87
```

So, this was our main. And here, we were calling this function `keep_distributing_points` and let us get back to this function and see what it was doing. So, `keep_distributing` functions was keeping on calling this function `distribute_points`. This `distribute_points` was basically, 1 iteration that was returning us updated points after that iteration right.

Now, after every iteration we have to make some changes to the points in order to handle the cases of sinks, point sinks. Now, how that is usually done let me explain that to you. So basically, what we do is after every iteration, we gather a fraction of points from

every node and the points that we gather, this way we then distribute these points equally amongst all the nodes right. So basically, for example, after 1 iteration we get some points with all the nodes. And after that to handle the sink cases, we take one more step and, in that step,, we let the node have 80 percent of it is points and we take some 20 percent of the points from that node. And we do that with every node. And that is how we gather some number of points, after that we divide these points that we have gathered this way amongst all the nodes equally.

Now this is the way to handle the problem of sink. You can consider this to be similar to the tax system in any country, where we take some say 30 percent of the salary from every person, if their salary is exceeding some value. So, the amount of money that we gather this way that is distributed for the development of the rest of the people or all the people in general. So, it is the same scenario here as well, the number of points that the node will be all the nodes will be keeping.

And the fraction of points that they will be letting go is basically decided by a parameter, which is usually denoted by s. So, usually we take  $s = 0.8$ , which means 80 percent of the nodes points will be with the nodes and 20 percent of the nodes points will be let go by all the nodes right. So, let us increment that thing here. As you can see in this function `keep_distributing_points`, we call this function and that is how we got the new points.

(Refer Slide Time: 03:44)

```

31             new_points[each[1]] += share
32
33     return G, new_points
34
35
36 def keep_distributing_points(G, points):
37     prev_points = points
38     print 'Enter # to stop'
39     while(1):
40         G, new_points = distribute_points(G, prev_points)
41         print new_points
42
43         new_points = handle_points_sink(G, new_points)
44
45         char = raw_input()
46         if char == '#':
47             break
48         prev_points = new_points
49
50     return G, new_points
51

```

Now, after we get the new points, let us take one more step to handle um the sinks let me for that purpose create a function. So, I will write new points, is equal to handle points sink let us call this function. So, let us pass the graph and the new points here ok. So, we have to create this function, now which will be doing, whatever I just explained that is it will be taking 20 percent of the points from every node and whatever we gather this way we will be distributed to all the nodes equally.

(Refer Slide Time: 04:16)

```

23
24     for i in G.nodes():
25         out = G.out_edges(i)
26         if len(out) == 0:
27             new_points[i] += prev_points[i]
28         else:
29             share = (float)(prev_points[i])/len(out)
30             for each in out:
31                 new_points[each[1]] += share
32
33     return G, new_points
34
35 def handle_points_sink(G, points):
36     for i in range(len(points)):
37         points[i] = (float)(points[i])*0.8
38
39     n = G.number_of_nodes()
40     extra = ((float)(n)*100*0.2)/n
41     for i in range(len(points)):
42         points[i] += extra
43
44     return points

```

So, let us create this function ok. So, as a first step every nodes point will be reduced right, every nodes point will be reduced to 80 percent of what they were. So, let me start look here for `i in range(len(points))` we can generalize it to may be points and that is what it will be returning. So, for `i in range(len(points))`, we must reduce the points for every node. So, we will write `points[i] = points[i] * 0.8`. So, they become 80 percent or what they were earlier let me do the casting ok.

So now, every node has reduced points reduced to 80 percent, now what should be done with the remaining 20 percent of the point that we have gathered from every node? Now if you know that what are the total number of points that the graph had initially: you can understand that then the total number of points never change. They are only sorry; they are only distributed amongst each other or the nodes will keep giving the points to each other, but the total number of points never changed ok.

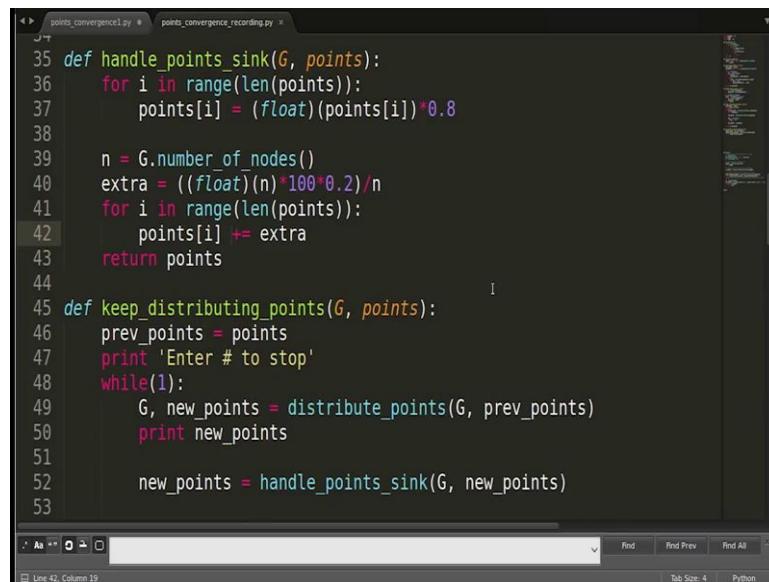
So, we are going to use that fact to know, what are the points that we have gathered? So, if every nodes point has become 80 percent, we have gathered 20 percent of the total number of points that the graph had ok. Now what is the total number of points that had? We had assigned 100 points to every node ok. Let us see the total number of nodes, I will just show that here. So, I will write `g.number_of_nodes` ok. So, I got `n` here what will be the total number of points in the graph that will be `n * 100`, because we assigned 100 points.

Now all of this these total points 20 percent is what we have now so that we can distribute to every node. So, I will write into 0.2. So, this is what we have ok. Now how many points to we have to distribute to every node? We must distribute these points equally ok. So, we will divide by `n` ok, so that we get an equal share for every node that will be the casting here ok. Let us call it extra now because, this is the extra points that every node will be having may be, we can took this as well ok.

So, these are the extra points that have to be assigned to every node apart from the point that they already have. So again, we will start a loop. I will write for `i` in `range(len(points))` again ok. So, points of every node will become plus equal to extra ok. So, every node is going to get extra points I think we have done. So, just return the points these are the new points ok. So, we passed old points to this function `handle_points_sink` and it will return the updated points, which will go back to this function keep distributing points here ok.

So, new points have changed, this is what we passed, and this is what we got ok. So, new points have changed since, we had not change the name. So, I do not think we have to change any other thing in the function.

(Refer Slide Time: 08:09)



```
35 def handle_points_sink(G, points):
36     for i in range(len(points)):
37         points[i] = (float)(points[i])*0.8
38
39     n = G.number_of_nodes()
40     extra = ((float)(n)*100*0.2)/n
41     for i in range(len(points)):
42         points[i] += extra
43     return points
44
45 def keep_distributing_points(G, points):
46     prev_points = points
47     print 'Enter # to stop'
48     while(1):
49         G, new_points = distribute_points(G, prev_points)
50         print new_points
51
52         new_points = handle_points_sink(G, new_points)
53
```

(Refer Slide Time: 08:14)

```
[227.1947057381879, 37.592249033734355, 105.85868838452122, 83.2034754699315
, 67.25192153018345, 140.35142718231808, 54.75776969874026, 76.4957101716420
5, 93.83281785922962, 113.46123493151191]

[227.1947057381879, 37.592249033734355, 105.85868838452122, 83.2034754699315
, 67.25192153018345, 140.35142718231808, 54.75776969874026, 76.4957101716420
5, 93.83281785922962, 113.46123493151191]

[227.1947057381879, 37.592249033734355, 105.85868838452122, 83.2034754699315
, 67.25192153018345, 140.35142718231808, 54.75776969874026, 76.4957101716420
5, 93.83281785922962, 113.46123493151191]

[227.1947057381879, 37.592249033734355, 105.85868838452122, 83.2034754699315
, 67.25192153018345, 140.35142718231808, 54.75776969874026, 76.4957101716420
5, 93.83281785922962, 113.46123493151191]
#
nodes_sorted_by_points [0 5 9 2 8 3 7 4 6 1]
0 5 9 2 8 3 7 4 6 1
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_6_(Link Analysis)/Code/
Video_code$ python points_convergence_recording.py
```

Let see how our program works. So, let us execute, so these are the initial values, and these are this is how the values are changing, I am just keeping on pressing enter and now I will press # ok. So, these are the values that we are getting 0592837. So, they are exactly in matching. So, we are handling the sink cases. So, the values are exactly matching.

Let me tell you one more thing sometimes there might be few differences with the with the ranking that you are getting from the networkx function as I told you, I guess one of

the reasons could be 2 of a nodes are having the same page rank another reason could be you have not let it converge, it is going on and on and it is taking a lot of iterations to reach an exact convergence point.

And you are just stopping somewhere in the middle. So, that might also lead to a slight deviation from the ranking that you are getting from networkx. Let me check one more example here ok.

(Refer Slide Time: 09:21)

```
[64.01028097141224, 58.520807692687356, 98.25552970807644, 110.8290816197897  
2, 218.3354120891941, 121.90356318812216, 182.42474313462776, 93.26529858218  
694, 0, 52.45528301390334]  
  
[64.01028097141224, 58.520807692687356, 98.25552970807644, 110.8290816197897  
2, 218.3354120891941, 121.90356318812216, 182.42474313462776, 93.26529858218  
694, 0, 52.45528301390334]  
  
[64.01028097141224, 58.520807692687356, 98.25552970807644, 110.8290816197897  
2, 218.3354120891941, 121.90356318812216, 182.42474313462776, 93.26529858218  
694, 0, 52.45528301390334]  
#  
nodes_sorted_by_points [4 6 5 3 2 7 0 1 9 8]  
4 6 5 3 2 7 0 1 9 8  
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_6_(Link Analysis)/Code/  
Video_code$
```

So, this is what we are getting let me stop here. 4653270198 here; so, we are getting the same values because, we are handing the case.

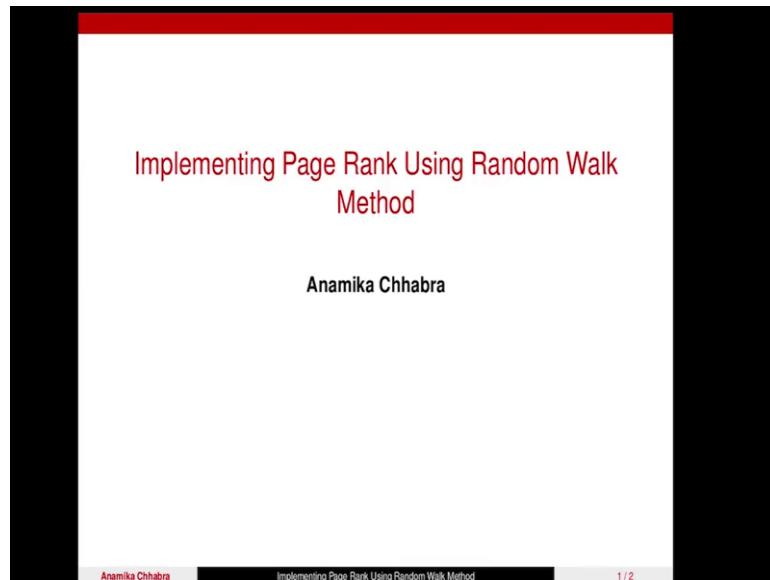
So, in case you get the slight deviations also you need do not worry because, you have understood the whole concept; how basically points distribution is similar to getting a page rank because, apart from the mathematical complexities this is what basically is the main idea behind getting the nodes rank based on the page rank values.

So, this was about the implementation of page rank using point distribution method. So, we are going to implement random walk method in the next videos.

**Social Networks**  
**Prof. S.R.S. Iyengar**  
**Prof. Anamika Chhabra**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Link Analysis**  
**Lecture – 84**  
**Implementing PageRank Using Random Walk Method – 1**

(Refer Slide Time: 00:05)



Hey everyone, in the previous video we implemented page rank using points distribution method. In this video we are going to Implement Page Rank Using Random Walk Method. Now what exactly is random walk method? So, basically, we are going to choose 1 node from the network uniformly at random. And, we will look at its neighbors; we will choose one neighbor out of them uniformly at random. And then we will look at the neighbors of that node, and we will keep doing this. And that is how we will randomly walk on the network that precisely what random walk is.

Now, there are two things that we are going to do along with this walk. And that is first thing that we will do is as and when we choose a node, we will increment its points by 1. Since we used points in the previous method as well which are different sort of points, let us call these points as random walk points just to differentiate things.

So, as I was saying as and when we reach a node, we increment its random walk points by 1 ok. That is one thing that we do, second thing is while we are randomly walking on the network it might so happen that we reach a node which has no out links, which has no neighbors where we can go right. So, in that case what we do is we out of all the nodes in the network we choose one uniformly at random and we resume ours our random walk from that node.

Now this is precisely called teleportation, which is a frequently used term when you talk about page rank and random walks. So, teleportation is basically choosing a node uniformly at random from the network and starting our basically resuming our random walk from that point.

Now, what is the basic idea behind this concept? How is it I mean it is, it looks counterintuitive that we are trying to implement page rank using random walk; it sort of does not make sense intuitively. So, let me tell you the basic idea that is used here. As we randomly walk on the network, a node which has a number of huge number of in links. There is a high probability that we will reach at this node more often right. And anytime we reach a node we are incrementing its random walk points.

So, the number of times we reach a node we are keeping a track of that. And a node which has high number of in links we will be reaching that node quite often. On contrary if we have a few nodes which have very a smaller number of in links, we will be reaching them very less often and that is what is recorded by the number of random walk points. So, that is a basic idea because in page rank also the idea is same.

If lot of nodes point to you then you are in important ok. And another thing that it captures is if a lot of important nodes points to you then you are important. So, in the random walk also that thing will be captured that if a lot of nodes, which are reached quiet often and they are pointing to you, you will be reached quiet often. And, the same thing is captured by the count that we are keeping random walk points that we are keeping. So and based on those points we are going to rank the nodes; which will be precisely same as a ranking that we get from the page rank technique. So, let us see the steps that we are going to implement ok.

(Refer Slide Time: 03:54)

Steps for Implementation

- 1 Create/take a directed graph.
- 2 Perform a *random walk*.
- 3 Get sorted nodes as per points accumulated during random walk.
- 4 Compare with the inbuilt Page Rank method.

Anamika Chhabra      Implementing Page Rank Using Random Walk Method      2 / 2

As in the previous case we are going to take a directed graph. So, we can either take it using some generator from network x function or we can create it ourselves. So, we created in the previous video we are just going to make use of that code again.

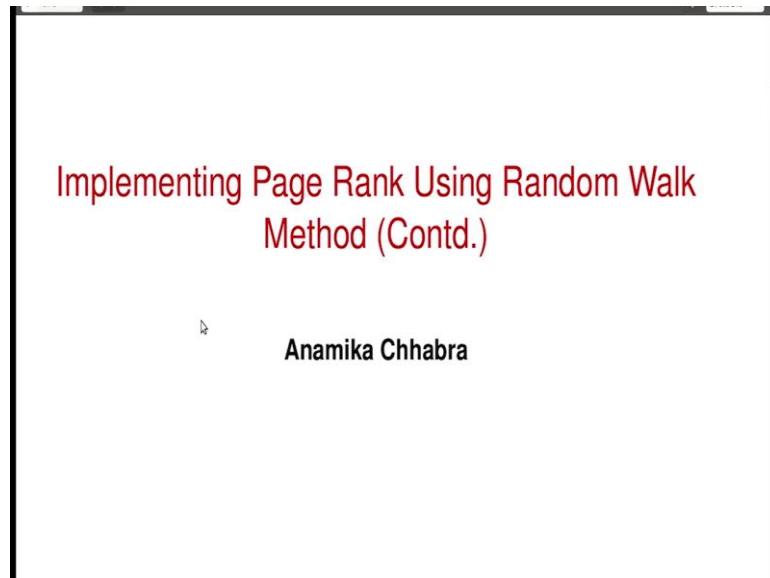
Second step that must be implemented is the main step which is performing a random walk. As I just explained to you, we are going to implement those steps. After performing a random walk, we are going to have count of the number of times we reached a node that is the random walk points. So, we are going to rank the node based on the random walk points that we obtained in the previous step.

In the last step as we did in the previous video as well. We are going to compare a results, with the results that we obtained from the inbuilt page rank method from networkx. So, these are the sequence of steps that we are going to follow; first, third and fourth steps are the same as in the previous video. So, the main thing that has to be implemented is the second one that is the random walk which we are going to in the next video.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Prof. Anamika Chhabra**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

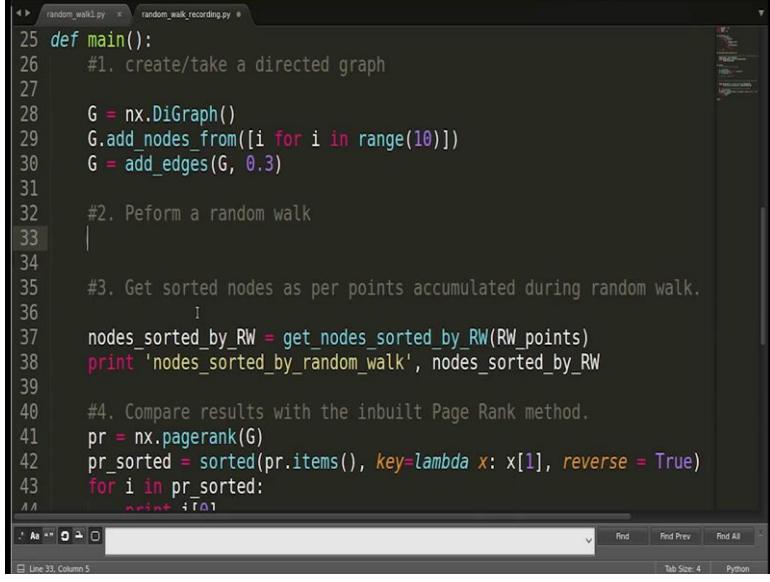
**Link Analysis**  
**Lecture – 85**  
**Implementing Page Rank Using Random Walk Method – 2**

(Refer Slide Time: 00:07)



In this video we are going to Implement Page Rank Using Random Walk. So, let us start the code.

(Refer Slide Time: 00:12)



A screenshot of a code editor window titled "random\_walk1.py" and "random\_walk\_recording.py". The code is as follows:

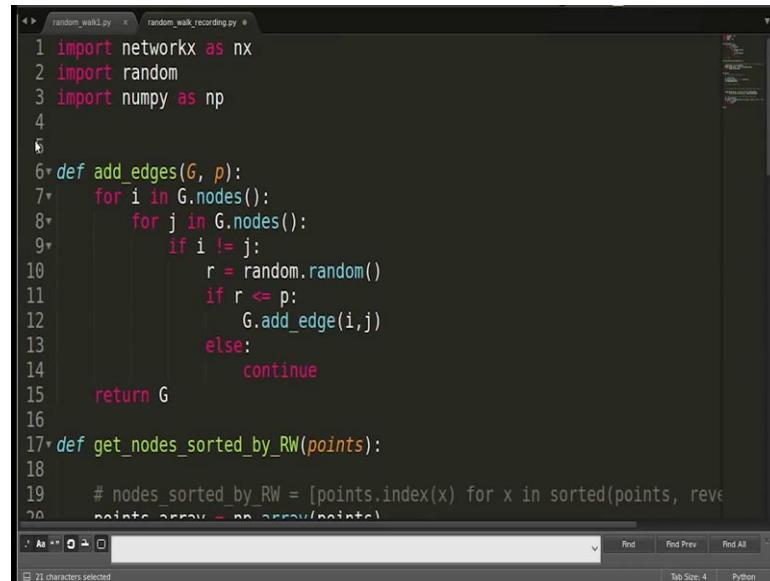
```
25 def main():
26     #1. Create/take a directed graph
27
28     G = nx.DiGraph()
29     G.add_nodes_from([i for i in range(10)])
30     G = add_edges(G, 0.3)
31
32     #2. Perform a random walk
33     |
34
35     #3. Get sorted nodes as per points accumulated during random walk.
36     |
37     nodes_sorted_by_RW = get_nodes_sorted_by_RW(RW_points)
38     print 'nodes_sorted_by_random_walk', nodes_sorted_by_RW
39
40     #4. Compare results with the inbuilt Page Rank method.
41     pr = nx.pagerank(G)
42     pr_sorted = sorted(pr.items(), key=lambda x: x[1], reverse = True)
43     for i in pr_sorted:
44         print i
```

The code implements a random walk on a directed graph. It starts by creating a directed graph with 10 nodes. Then it performs a random walk. After the random walk, it gets the sorted nodes based on the accumulated points. Finally, it compares the results with the inbuilt Page Rank method.

As I told you, in the previous video there are 4 steps which are to be implemented. And out of those 4 steps first third and forth steps are the same that we implemented in our previous video on implementation of page rank using point distribution method. So, we are going to reuse that code and there is no point of writing the code again.

So, I have copied pasted the code which is required over here in this file. Let me show you, we will start from the first step. First step is to take a directed graph. So, as we had created the graph. How had we created? We had taken an object of nx.DiGraph. So, we created an empty directed graph here and after that we add nodes to it, we added 10 nodes to this graph. And after that we added edges to the node to the graph using this function add\_edges.

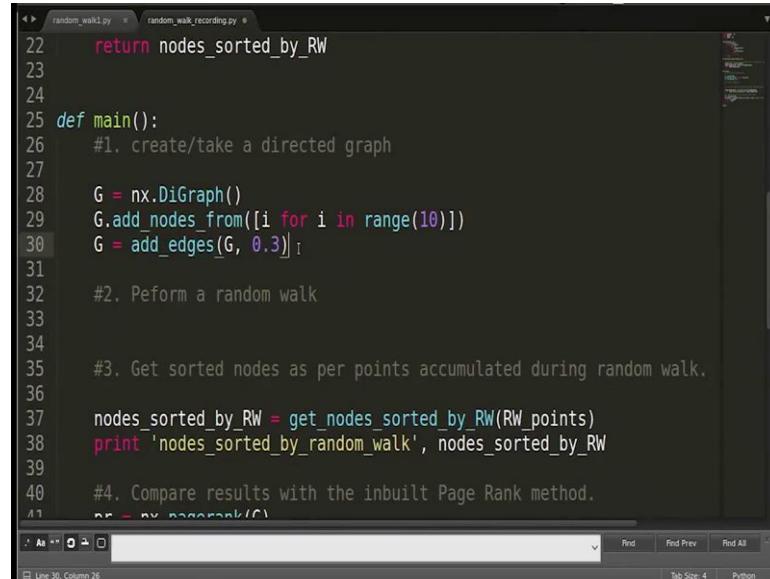
(Refer Slide Time: 01:07)



```
random_walk1.py random_walk_recording.py
1 import networkx as nx
2 import random
3 import numpy as np
4
5
6* def add_edges(G, p):
7    for i in G.nodes():
8        for j in G.nodes():
9            if i != j:
10                r = random.random()
11                if r <= p:
12                    G.add_edge(i,j)
13                else:
14                    continue
15    return G
16
17* def get_nodes_sorted_by_RW(points):
18
19    # nodes_sorted_by_RW = [points.index(x) for x in sorted(points, reverse=True)]
20    points_array = np.array(points)
21    points_array = np.argsort(points_array)
22
23
24
25 def main():
26     #1. create/take a directed graph
27
28     G = nx.DiGraph()
29     G.add_nodes_from([i for i in range(10)])
30     G = add_edges(G, 0.3)
31
32     #2. Perform a random walk
33
34
35     #3. Get sorted nodes as per points accumulated during random walk.
36
37     nodes_sorted_by_RW = get_nodes_sorted_by_RW(RW_points)
38     print 'nodes_sorted_by_random_walk', nodes_sorted_by_RW
39
40     #4. Compare results with the inbuilt Page Rank method.
41     pr = nx.pagerank(G)
```

So, let us go back here. These add edges function I have copied as it is from the previous code. So, here just to brief about we are tossing coin for every edge. And if the coin turns out to be head, we are adding that edge otherwise we are not keeping that edge. So, that is how we are randomly assigning edges to the graph.

(Refer Slide Time: 01:35)



```
random_walk2.py random_walk_recording.py
22     return nodes_sorted_by_RW
23
24
25 def main():
26     #1. create/take a directed graph
27
28     G = nx.DiGraph()
29     G.add_nodes_from([i for i in range(10)])
30     G = add_edges(G, 0.3)
31
32     #2. Perform a random walk
33
34
35     #3. Get sorted nodes as per points accumulated during random walk.
36
37     nodes_sorted_by_RW = get_nodes_sorted_by_RW(RW_points)
38     print 'nodes_sorted_by_random_walk', nodes_sorted_by_RW
39
40     #4. Compare results with the inbuilt Page Rank method.
41     pr = nx.pagerank(G)
```

So, after this our graph is ready and we can perform a random walk on this. So, that is a second step performing a random walk which is to be done, we will just do that. After forming the random walk we will be having random walk points. And then we have to

rank the node based on random walk points which is what we have done in the previous video. I have just pasted the code; I have just renamed this function get node sorted by random walk. I just renamed it here that is all, the functionalities entirely the same.

(Refer Slide Time: 02:08)

```

4
5
6 def add_edges(G, p):
7     for i in G.nodes():
8         for j in G.nodes():
9             if i != j:
10                 r = random.random()
11                 if r <= p:
12                     G.add_edge(i,j)
13                 else:
14                     continue
15     return G
16
17 def get_nodes_sorted_by_RW(points):
18
19     # nodes_sorted_by_RW = [points.index(x) for x in sorted(points, reverse=True)]
20     points_array = np.array(points)
21     nodes_sorted_by_RW = np.argsort(-points_array)
22     return nodes_sorted_by_RW
23

```

So, this function I have added as it is. So, that part is done.

(Refer Slide Time: 02:13)

```

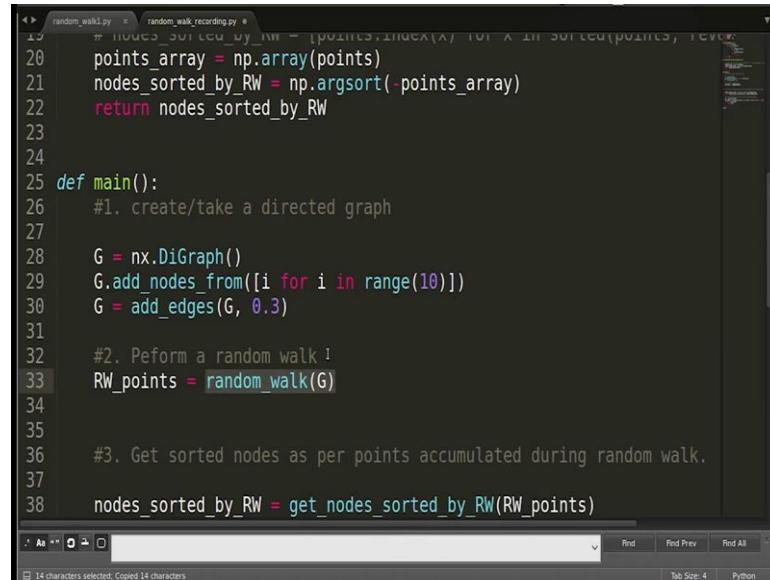
28
29     G.add_nodes_from([i for i in range(10)])
30     G = add_edges(G, 0.3)
31
32     #2. Perform a random walk
33
34
35     #3. Get sorted nodes as per points accumulated during random walk.
36
37     nodes_sorted_by_RW = get_nodes_sorted_by_RW(RW_points)
38     print 'nodes_sorted_by_random_walk', nodes_sorted_by_RW
39
40     #4. Compare results with the inbuilt Page Rank method.
41     pr = nx.pagerank(G)
42     pr_sorted = sorted(pr.items(), key=lambda x: x[1], reverse = True)
43     for i in pr_sorted:
44         print i[0],
45
46
47 main()

```

And finally, once we are done, we use the page rank method from network x. And we compare the results, our results with the functions results; again, this is entirely the same code ok. So, what has to be done is the second step which is performing a random walk,

now let us do that. So, this function is going to we are going to create a function random walk and that function will return random walk points ok.

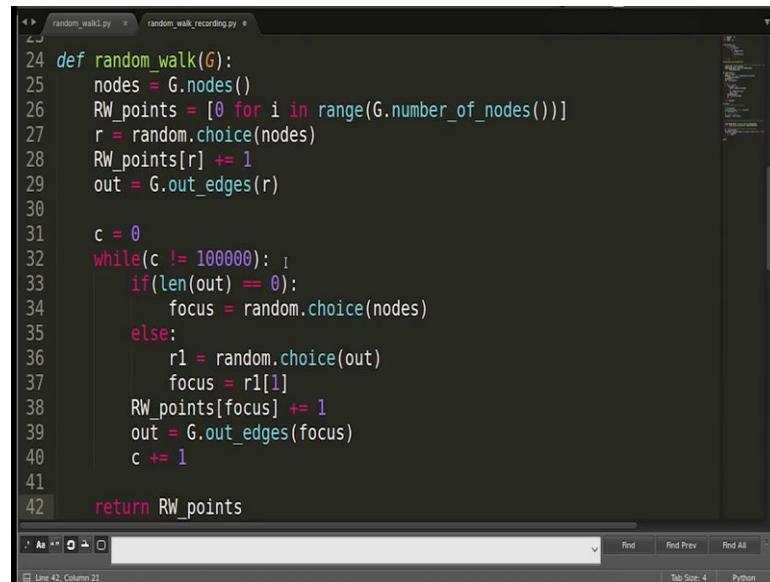
(Refer Slide Time: 02:43)



```
random_walk_recording.py
1  #!/usr/bin/python
2  # This file records the random walk points.
3
4  import networkx as nx
5  import numpy as np
6
7  def get_nodes_sorted_by_RW(RW_points):
8      #1. Create a sorted array of nodes based on the number of points accumulated during the random walk.
9      points_array = np.array(RW_points)
10     nodes_sorted_by_RW = np.argsort(-points_array)
11
12     return nodes_sorted_by_RW
13
14
15
16
17
18
19
20
21
22
23
24
25 def main():
26     #1. Create/take a directed graph
27
28     G = nx.DiGraph()
29     G.add_nodes_from([i for i in range(10)])
30     G = add_edges(G, 0.3)
31
32     #2. Perform a random walk !
33     RW_points = random_walk(G)
34
35
36     #3. Get sorted nodes as per points accumulated during random walk.
37     nodes_sorted_by_RW = get_nodes_sorted_by_RW(RW_points)
38
```

So, let us write that: random walk points is equal to random walk. So, we are going to create this function random walk, we will pass the graph there ok. So, let us create this function.

(Refer Slide Time: 02:59)



```
random_walk.py
1
2
3
4  def random_walk(G):
5      nodes = G.nodes()
6      RW_points = [0 for i in range(G.number_of_nodes())]
7      r = random.choice(nodes)
8      RW_points[r] += 1
9      out = G.out_edges(r)
10
11     c = 0
12     while(c != 100000): 
13         if(len(out) == 0):
14             focus = random.choice(nodes)
15         else:
16             r1 = random.choice(out)
17             focus = r1[1]
18         RW_points[focus] += 1
19         out = G.out_edges(focus)
20         c += 1
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42     return RW_points
```

Now, as I told you what do we mean by random walk? We will randomly choose, 1 node from all the nodes that, is the first step ok how do we do that? Let us create let us get a

list of nodes firstly. So, let us write node = G.nodes because, we have chosen one randomly we will choose out of these out of these list. And as and when we visit a node, we increment its counter, increment its random walk points. So, it will be nice to create a list which will be returned by this function, random walk points is equal to we have to initialize it to 0. So, we can write 0 for i in range G.number\_of\_nodes. So, random walk points initialize to 0.

Now, we have this list of nodes and we have to choose one out of them out of these nodes randomly. So, we can do r is equal to random dot. We have to choose 1 node randomly out of these lists; for that we can use a function random dot choice, random dot choice out of nodes ok. So, we got 1 node uniformly at random. Since, we are have visited it we must increment its counter its point. So, we will write RW points for this node r, must be incremented by 1.

After that we have to look at the nodes which are neighbors to this node ok. We have to choose one neighbor uniformly at random. So, let us first get a list of the neighbors of this node are. How do we get that? For a directed graph we have function G dot out edges ok. G dot out edges for r, out will contain a list of all the edges, out edges from r ok. And after that we have to choose one of the, one of the neighbors uniformly at random and we have to repeat the same process. So, basically what are we are doing, we have to keep repeating it. How many times we should do, as much as possible ok.

So, let us create a counter for that. And we can start a loop because, we have to keep repeating things, we can take a very big number probably whatever as much as you can handle. I have taken this big number. So, we will visit these many nodes. So, we will basically have these many iterations of a random walk from one node to the other. So, we can keep any number here. In the basic idea is that we should be able to visit all the nodes; we there should not be nodes which we have not visited because if you do not visited node the counter for them will be 0. And, hence we will not be able to rank the nodes properly.

So, the idea is that we should be able to visit all the nodes. And that too multiple times so, that we are able to capture the frequency of visiting in our counts; in our random walk points ok. So, what do we have to keep repeating? We are started the loop. What do we have to keep repeating? We had reached r and these are the neighbors of r, we have to

choose one uniformly at random out of them ok. So, that is what we will do. Before that what if this out is 0, the length of out is 0 that is the node which we have reached has no out links that may also happen right. And, I told you in that cases what we do? In that case we do teleportation, that we that is we choose one node uniformly at random from all the nodes.

So, let us check that if length of out is equal to 0, what am I doing? If the length of out is equal to 0. What we do? We have to choose one uniformly random let us create a new variable for that to keep the node which is currently under focus ok. This focus will contain the node which is currently under focus, which is currently being visited basically. So, we have to choose one node uniformly at random from all the nodes we will write random dot choice nodes ok. This is the list which we already created. In case the node which we have reached at has number of neighbors, what do we have to do? We have to choose one neighbor out of them.

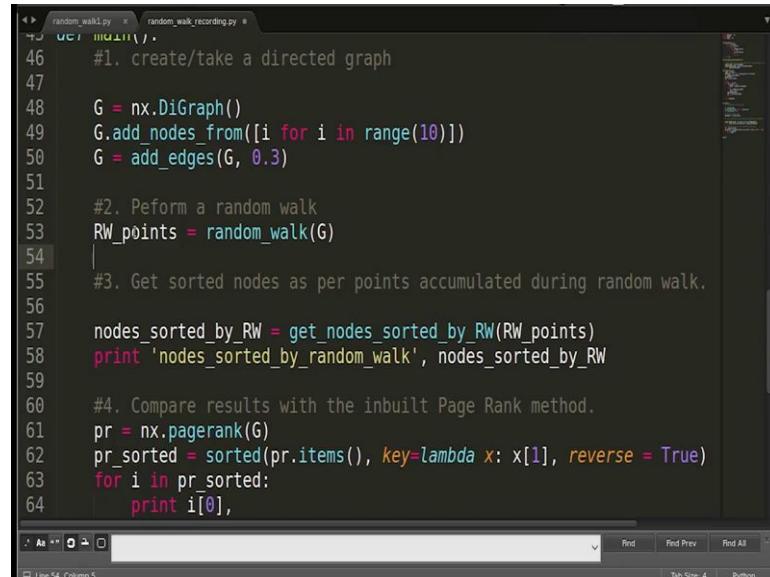
So, maybe we can have another random variable here, we can write  $r1$  dot. I am sorry `random.choice`. Now this choice has to be made out of what? This choice has to be made out of `out`. Because `out` is what is containing all the edges and we will choose one edge out of them. Now, what is going to be the focus? The focus is not going to be the edge. The focus is going to be the node which is on the other side. How do we get the node, which is on the other side, using  $r1[1]$  because  $r1[0]$ ,  $r1[1]$  is basically the edge ok yeah exactly?

So, we got the focus where focus is the current node where we are currently. Since we have reached at this node. We have to implement the random walk points for this node. So, we will write random walk points for focus plus is equal to 1. Now again we have to get the list of its neighbors. So, we will write `out` is equal to we can copy paste this, I am just set `r` will be replaced by `focus`; because `focus`, `focus` is what is a current node.

Now, this is one iteration; what is iteration having? This iteration consists of looking at the neighbors choosing one out of them randomly and then it be and then repeating the same process. That is what this while looping is doing. Let us implement the count `c` that is the number of times this while loop will continue ok. I think we are done. So, what we have to return? This function should return random walk points right. So, let us return

this; I think we are done. Let us go down to the main, in the main also we have implemented everything.

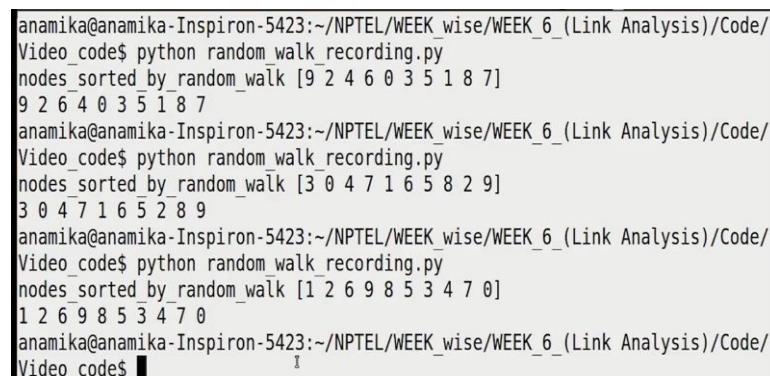
(Refer Slide Time: 10:05)



```
random_walk.py      random_walk_recording.py
45 def main():
46     #1. Create/take a directed graph
47
48     G = nx.DiGraph()
49     G.add_nodes_from([i for i in range(10)])
50     G = add_edges(G, 0.3)
51
52     #2. Perform a random walk
53     RW_points = random_walk(G)
54
55     #3. Get sorted nodes as per points accumulated during random walk.
56
57     nodes_sorted_by_RW = get_nodes_sorted_by_RW(RW_points)
58     print 'nodes_sorted_by_random_walk', nodes_sorted_by_RW
59
60     #4. Compare results with the inbuilt Page Rank method.
61     pr = nx.pagerank(G)
62     pr_sorted = sorted(pr.items(), key=lambda x: x[1], reverse = True)
63     for i in pr_sorted:
64         print i[0],
```

We only have to write this instruction rest all are the same from the previous code. I think we are good to go, let us check whether this code works.

(Refer Slide Time: 10:19)



```
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_6_(Link Analysis)/Code/
Video_code$ python random_walk_recording.py
nodes_sorted_by_random_walk [9 2 4 6 0 3 5 1 8 7]
9 2 6 4 0 3 5 1 8 7
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_6_(Link Analysis)/Code/
Video_code$ python random_walk_recording.py
nodes_sorted_by_random_walk [3 0 4 7 1 6 5 8 2 9]
3 0 4 7 1 6 5 2 8 9
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_6_(Link Analysis)/Code/
Video_code$ python random_walk_recording.py
nodes_sorted_by_random_walk [1 2 6 9 8 5 3 4 7 0]
1 2 6 9 8 5 3 4 7 0
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_6_(Link Analysis)/Code/
Video_code$
```

So, let us see, as you can see the outputs 9, 2, 4, 6 and here 9, 2, 6, 4 and rest of the values are matching. So, there can be two reasons for the mismatch. One could be the rank of 4 and 6 might be same, 4 and 6 here and 6, 4 and 4 here. So, they can give different

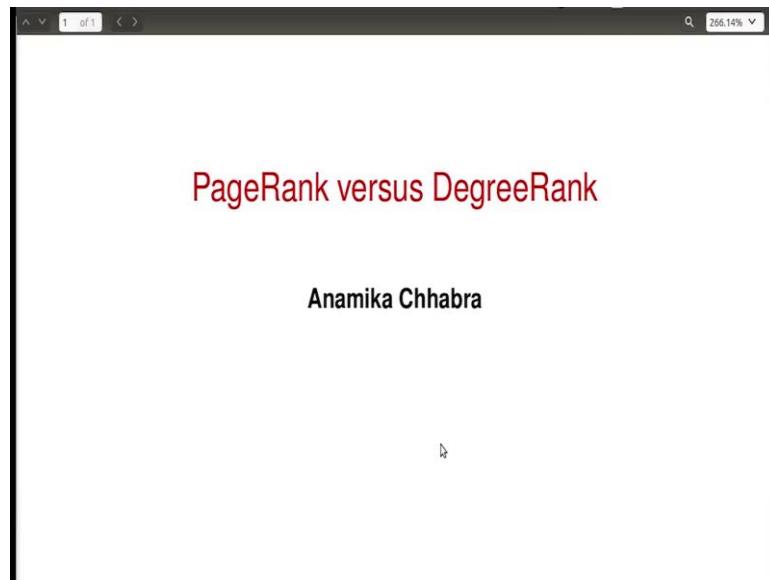
ordering. Second reason could be that we needed more iteration so, that we were able to visit all the nodes properly and we could achieve the effect. Because random walk is basically based on probability right, you randomly choose the next node. It might so, happen that you are ending up reaching one node, more than the other node whereas, they both have the same number of in links right.

So, that may happen, let us check another example. Again, here everything is same except 8 2 and 2 8 over here; we can get one more. So, in this case used as you can see it is precisely the same, everything is matching. So, it might happen because it is based on probability and it is never possible that there are 2 nodes which have the same configuration in terms of the number of in links. And, we are reaching these two nodes precisely the same number of times. So, that sort of difference is going to happen. So, this was about the implementation of page rank using random walk technique.

**Social Networks Link Analysis**  
**Prof. S. R. S. Iyengar**  
**Prof. Anamika Chhabra**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

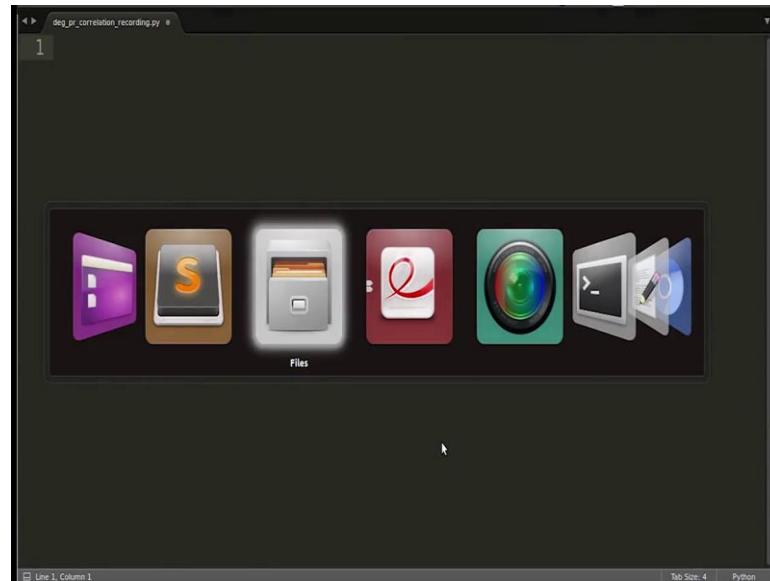
**Lecture – 86**  
**Degree Rank versus Page Rank**

(Refer Slide Time: 00:05)



Hey everyone, in this video we going to see the relationship between Degree Rank and Page Rank that is we are going to see whether they correlate with each other or not? We are going to see whether the nodes which have less degree do they have less page rank and vice versa. So, in order to check this, we are going to use citation network which is a directed graph I already have downloaded it from snap repository, so I am going to use that. So, let us start the implementation.

(Refer Slide Time: 00:34)



Let me show you the citation data set how it looks like?

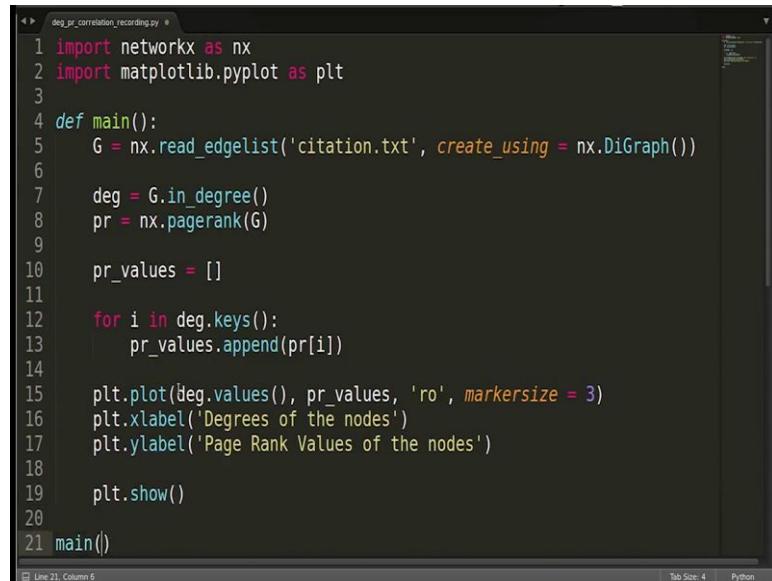
(Refer Slide Time: 00:38)

```
citation.txt x
1 2
3 4
5 6
7 8
9 10
11 12
13 14
15 16
17 18
19 20
21 22
23 24
25 26
27 28
29 30
31 32
33 34
35 36
37 38
39 40
41 42
43 44
45 46
47 48
49 50
51 52
53 54
55 56
57 58
59 60
61 62
63 64
65 66
67 68
69 70
71 72
```

A screenshot of a text editor window titled "citation.txt". The window displays a list of edges in the form of pairs of numbers, representing citation links between research papers. The file contains approximately 70 such pairs, starting from (1, 2) and ending at (71, 72). The text editor interface includes standard toolbar icons and status bar information.

So, this is the citation data set as you can see it is in the form of edge list. Every row contains an edge these are the IDs of the research papers. So, if you take an edge it indicates that the first paper is citing the second paper. So, the paper which there is ID 17 is citing in the paper with the ID 18. So, this is a sort of information that we have.

(Refer Slide Time: 01:04)



```
1 import networkx as nx
2 import matplotlib.pyplot as plt
3
4 def main():
5     G = nx.read_edgelist('citation.txt', create_using = nx.DiGraph())
6
7     deg = G.in_degree()
8     pr = nx.pagerank(G)
9
10    pr_values = []
11
12    for i in deg.keys():
13        pr_values.append(pr[i])
14
15    plt.plot(deg.values(), pr_values, 'ro', markersize = 3)
16    plt.xlabel('Degrees of the nodes')
17    plt.ylabel('Page Rank Values of the nodes')
18
19    plt.show()
20
21 main()
```

Let us implement this now. Since the network is in the form of a edge list we are going to use the function read edge list in order to read it into a graph for object. First, let me import networkx we are going to plot there is a so let me import matplotlib ok.

So, let us start the main function we have to read the network. So,  $G = \text{nx.read\_edgelist}$ , the network name is citation.txt. Since we have to create a directed graph, we will write create using is equal to  $\text{nx.DiGraph}$ .

Now since this is a directed graph, we are going to keep a track of the in degrees of the nodes that is basically what we will compare with the page rank. So, we have to check whether a node which has more in degree that is a greater number of in links. Does it have more page rank is well or not and vice versa.

So, let us first get all the in degrees we can use a function  $G.in\_degree$  for that purpose. Now what this function return is a dictionary. So, this is going to be a dictionary where the keys are going to the nodes and the values are going to be the in\_degree values of these nodes.

Similarly, in order to compute the pr we are going to use the inbuilt functions from networkx. We will write the  $\text{page rank} = \text{nx.pagerank}(G)$ , again page rank function will return a dictionary where the keys are the nodes and the values are the page rank values of these nodes.

Since these two are two different dictionaries and we must plot them. We basically we are going to keep the in\_degree values on the x axis and the corresponding page rank values on the y axis. Since there are two different dictionaries let us keep one of them in the order of the other.

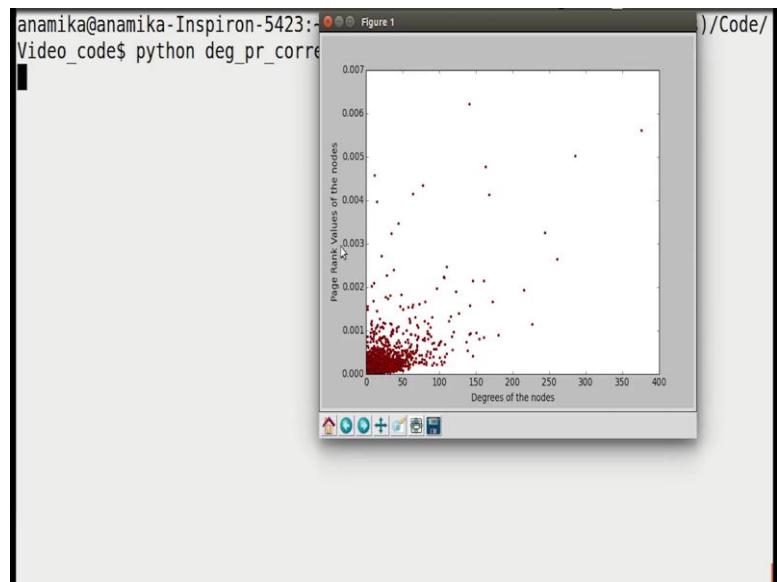
So, I am going to create a list to keep the page rank values in the same order in which degree values are coming. So, I am creating this list page rank values is equal to a list. So, I am going to create this list which will have the page rank values of the nodes in the same order in which they are pairing in this degree dictionary ok.

So, for that I am starting a loop in deg.keys. So, deg.keys will give the keys in the sequence, then we will append the page rank values pr\_values.append. What do we have to append? We have to append the page rank value of the corresponding node i ok. So, we will write pr[i] ok, so we got the both the things in the same sequence and then we can plot them.

So, I will write plt.plot on the x axis we have in degrees which are available in this dictionary. So, we will write deg.values corresponding to these degree values what are the page rank values is what is available in pr\_values list that we just created. Let us display it in red circles; we can also change the marker size ok.

Let us also give the labels, degrees of the nodes and we can also give the y label on y axis we have page ranks, page rank values ok. After that we have to show we will write plt.show sorry ok. Then we will call this main function they should work. Let us see how it works?

(Refer Slide Time: 05:55)



As you can see in the plot degree and page rank values are not correlated with each other. Because if they had been correlated, they would have been linear sort of plot over here which is not there. And, you can see that there are few nodes which have high in degree; however, very less page rank.

For example, if you take this node it has high in degree, but if you look at its page rank which is less. In the context of citation network, it indicates that this is a research paper which is being cited by several other papers. However, those papers are not so important right that is why this paper got less page rank.

Similarly, if you look at the other side you have a few nodes which have very less in degree; however, extremely high page rank. Now in the context of citation network, it indicates that these are the research papers which are being cited by very fewer other papers. However, those papers are important themselves that is those papers are being cited by a lot of other papers, or those papers are being cited by a lot of important papers. So, the conclusion is that in degree and page rank values are not correlated with each other.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Prof. Anamika Chhabra**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Cascading Behaviour in Networks**  
**Lecture – 87**  
**We Follow**

Do you think we follow each other? What do I mean by follow? So, all of us have seen ants in our home or in our backyard happily following each other.

(Refer Slide Time: 00:14)



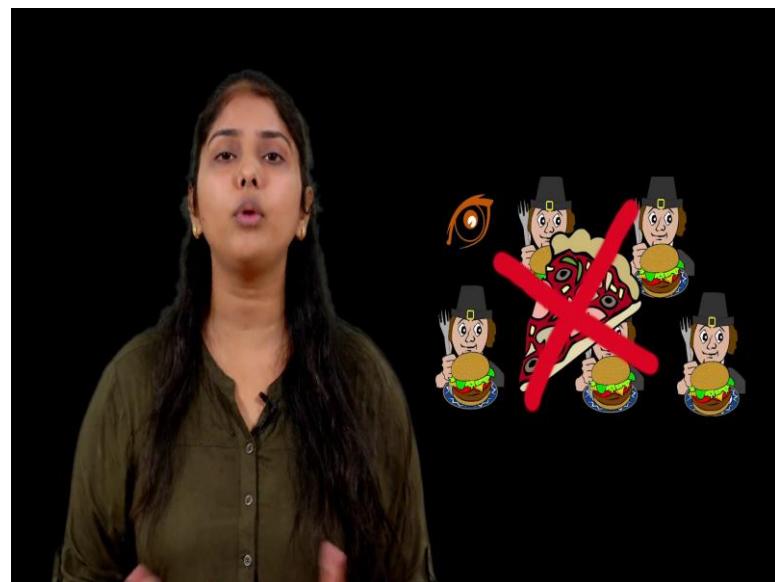
One ant moving behind the next ant, ant and all of them are moving in the same way and as a kid we have tried to disturb their movement as well, but they again go and make their own line. But then do human beings follow each other? Yes, we know we do not move behind each other the way ants do. But we might be following each other in some different manner. What different manner? Let me give you some examples..examples.

(Refer Slide Time: 00:48)



Assume you go to this restaurant and there are two choices. There are two things you can order: one is a burger; another is a pizza. And, let us say you like pizza more, but then you look around to all the people.

(Refer Slide Time: 01:05)



And you see each of them is eating a burger. There is not even a single person in the restaurant who has ordered a pizza and he is eating it. What will you do now? Will you go and order burger or will you go ahead and order pizza?

(Refer Slide Time: 01:17)



Let me give you another example.

(Refer Slide Time: 01:25)



You are going on this road and suddenly the road ends. The road gets split in two parts; one part of the road goes towards the right and other part goes towards the left. And then again you see at the people around you, and most of the people they are moving towards right. Almost everybody is taking the road towards the right. What will you do in this case? Will you go towards the right or will you be bold enough to move and go towards the left side, where nobody is going. We do not know. The answer to this question, do

we follow or not, what do we do in these situations is in the clip which is coming next.

(Refer Slide Time: 02:10)



The gentle man in the elevator now is a candid subject. These folks who are entering the man with the white shirt, the lady with the trench coat and subsequently one other member of our staff will face the rear. And, you will see how this man in the trench coat tries to maintain his individuality. But, little by little looks at his watch but, he is really making an excuse for turning just a little bit more to the wall. Now, we try it once again, here is the candid subject. Here comes the candid counter staff, 3 of them at least and this man has apparently been in groups before.

Now, here is a fellow with his hat on in the elevator. [FirstFirst](#), he makes a full turn to the rear and Charlie closes the door. A moment later we will open the door, everybody has changed positions. Now, we will see if we can use now, we will see if we can use group pressure for some good. Now, in a moment on Charlie's signal everybody turns forward. Notice they take off their hats and now do you think we could reverse the procedure watch.

So, what did we see just now in the clip? So, we have seen that the person in the elevator when he looks at everybody was turned towards the wall, also turns towards the wall. And how silly however silly it might sound, why will one turn towards the wall? But

when we will see other people around us doing some silly stuff, because of our sense of conformity; we also tend to do the same.

(Refer Slide Time: 04:55)



So, in the examples which we have discussed before, when you are in a restaurant it might occur to you; that probably everything in the restaurant except burger is spoiled and hence everybody is eating burger. Or there is an offer on burger may be some Saturday offer and everybody is getting a discount. And you might also go and order a burger.

In another example when there was a road and one of it went towards left and the other one towards right. And everybody is going towards right, you might feel there is something wrong with the road towards the left maybe, some construction work is going on or it leads to let us say; some place where nobody wants to go something like that. And you mostly take the road towards the right. These are two very simple examples. But while doing almost everything in our life, we follow others our decisions are not independent.

(Refer Slide Time: 05:55)



You would have encountered it again and again in your life. When you have to maybe choose a subject after 10th or you have to decide your career; we do not take this decision all by our self.

We take all the feedback from our good friends, from our parents, from our teachers. And then we weigh all these options and then take a decision. So, we almost follow others in almost every aspect of our life.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Prof. Anamika Chhabra**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**  
**Cascading Behaviour in Networks**

**Lecture – 88**  
**Why do we Follow?**

So, we have just seen that yes, we do follow. Now I would like to ask this question to you. Should we follow?

(Refer Slide Time: 00:14)



Should we follow others? If everybody around me is going and jumping into a well. Should I also go and jump in the well? So, it is a very popular statement which is again and again used by your parents, when they do not want us to do something which all of our friends are doing. Let us take a more realistic example.

Let us say you are writing this math's exam.

(Refer Slide Time: 00:37)



In a classroom, and the invigilator the teacher goes out for some five minutes. And everybody in this classroom gets a chance to cheat. So, you do not have enough time to copy and match the exact answers. But you have time to validate your answers. What is your result? And everybody is going around and validating their answers with each other. But then there is this question number 4 whose concept you know well. You have calculated it. And your answer is 40. And you are quite sure about it. Then when you go about to validate the answer to this question; your first friend says no my answer is 30. You go to your second friend; he says no my answer is 30 third friend 30, fourth friend 30 so on and so forth.

Then mostly, a time will come where you will cut off that 40 on your sheet and write a 30 there. So, it might not be good to follow others all the time. If it is not good to follow others then, why do we follow others?

(Refer Slide Time: 01:53)



Before giving you answer to this question, I want to refine the definition of follow, not refine I want to explain in a little bit of more detail. What all kind of circumstances do you count in follow? Let us say my friend is doing something. I look at him he is doing this thing and I also start doing the same thing. It is a follow.

Let us say he is smoking cigarette. I looked at him, I feel smoking cigarette is cool. And I also start smoking cigarette, I followed him. Next my friend is using some product which is really nice. I looked at him, I like the product, I also started using it. Let us say an apple iPod; he is using an apple iPod, I liked it very much. I started using it, I followed him. A third kind of follow, rather a more involved a more intrinsic one is about the information we receive from other people. How information? My friend has information, which is of interest, some information which is of interest. It can be about a recent assignment we have to submit; it can be about news related to a Hollywood or Bollywood celebrity or something else some nice information.

So, when we I talk to my friends. He tells me about this information, I like the information, I believe in the information. This is also a kind of follow; I followed him.

(Refer Slide Time: 03:25)



For example, let us say last night; my friend watched this Kolavari Di song. And he really liked it. Next morning, he came to me; and told me why you do not watch this video, it is very nice. And I go ahead, and I watch this video; and I also really liked it. So, I followed him. So, this is the definition of follow which we will be using. So, all these examples which I gave you come in the category of following.

(Refer Slide Time: 03:55)



So, the question which we were addressing was, why do we follow each other. Like we looked at some examples where we saw that we should not every time we should not be

following each other's, then why human beings do intrinsically they follow each other? The answer is in these examples I am going to tell you.

First example, one of my friend, he is in a health care industry. And he has come up with some new kind of sport shoes, which have a lot of nice feature, is what he says. But then I might not find those features to be very nice. There are a lot of sport shoes which keep on coming in the market. And all of them have nice features why should I care about this one my friend came up with? But then one nice day, one fine day, my friend comes to me and he gives me an offer.

He says that Yayati, if you take my sport shoes; if you look at my sport shoes you advertise it to 5 of your friends; I will give you one pair for free. Now my mind changes just advertising it to five people; and I am getting a pair of sport shoes no matter how bad it is. I will likely do that, I will do that; I kind of followed him, I followed him for some explicit benefit for some monetary benefit. So, this kind of following we can say it is some explicit gain I am getting out of this following.

Let us look at some more scenarios. Why do we follow others? Agriculture let us say it is one of the most prominent occupations in India. There is this small village; and this in the small village is a small villager.

(Refer Slide Time: 05:43)

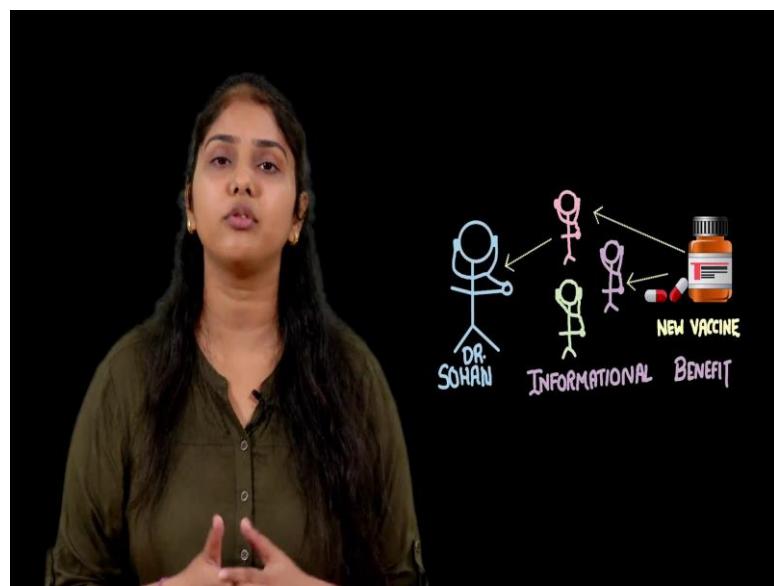


Let us say Sohan, Sohan does agriculture. Sohan has friends who do agriculture; Sohan has a very good friend. And one day suddenly sales of the crops of his friend doubles. And then Sohan is all very curious to know what happened in one day.

How come his sales doubled? And then Sohan goes and ask him. And his friend tells him, that he has imported he has got some imported hybrid seeds from America which makes his crop grow at a double pace. So, he has double the crops and hence he get better sales. How did Sohan get this information? So, Sohan will need to have some friends, whom he can follow, whom he can keep asking things? And then get to know of these nice things; like about these new seeds which we talked about. So, Sohan needs to follow to get useful information.

Let us talk about a different world another analogy. And this is the world of medical science. So, let us take our example; Sohan only and make him doctor Sohan here.

(Refer Slide Time: 06:49)



So, now, doctor Sohan is here. Again, he here needs to follow others, why? There is this very nice new vaccine which came in the market? And now the Sohan's patients can gain out of it.

So, it is Sohan's responsibility to be updated to keep talking to other doctors, to know what all is coming in the market. So, he follows other to get this useful information about the advances in the medical field. So, he needs to follow others. So, this is the second

reason why we follow other people to get useful information. And think not always useful, but mostly useful information we follow others to get this information. So, this kind of gain which we get out of following others is the informational gain.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Prof. Anamika Chhabra**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**  
**Cascading Behavior in Networks**

**Lecture – 89**  
**Diffusion in Networks**

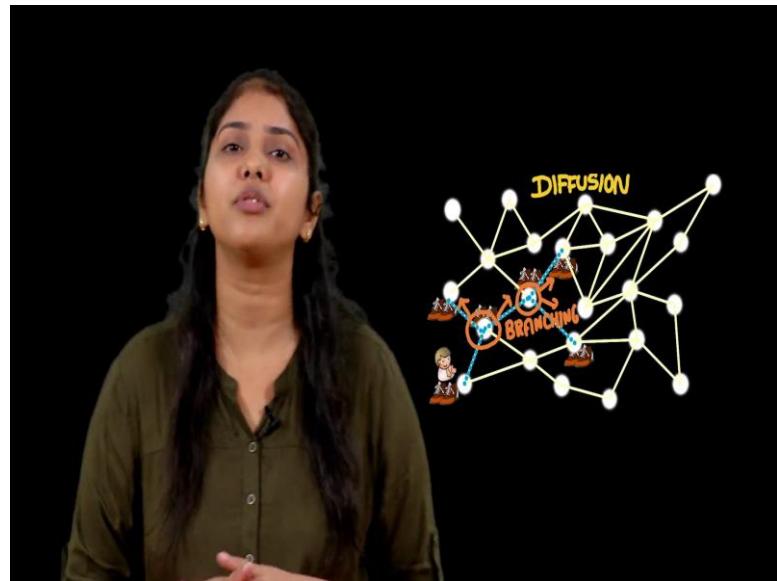
So, we have seen that we follow each other, and we have looked at why do we follow each other. Let us now take this concept of following one step further. What do I mean by one step further? It is not me who is following others; all of us are following each other. Let us take this sport shoes example.

(Refer Slide Time: 00:27)



My friend came up with nice sport shoes and then he gave me some offer and I kind of I adopted his sport shoes, his product of sport shoes I adopted. Then I have many more friends, they might look at me and one or two of them will then adopt these sport shoes and then their friends will look at them and some of them might adopt these sport shoes. So, do you kind of see a network phenomena going on here.

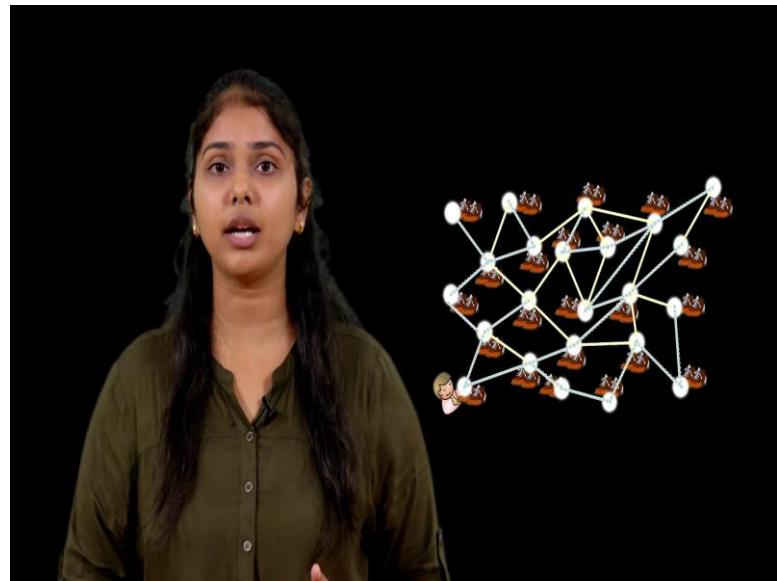
(Refer Slide Time: 01:01)



If I model this complete world as a network like this where every person in this billion of population, I represent every person as a node. And, then again like before I put an edge between two people, if they talk to each other, if they kind of follow each other they listen to each other I put an edge between them. So, this product starts from somewhere, it starts from my friend then I adopt it, then my friends adopt it, their friends adopted us and so on.

So, do you see that this product is kind of travelling in this big social network? Traveling is actually a misnomer; we can actually say it is diffusing through the network. So, travelling is like when something makes a path and goes, but it is branching. So, my friend would have shared it with some people, let us say 2 people and then these 2 people if they share it with 2 people more it becomes 4. So, it is not just one path, this product is branching through this entire network and we call it diffusion. Now, one question of interest here which my friend would be very interested in how many people at the end adopt his product.

(Refer Slide Time: 02:07)



So, whenever a piece of information or a product or something else diffuses in a network from person to person it starts from someplace, then hit some people; hit some more people and so on. What happens at the end? What will happen at the end, can we tell? Will this product sweep the entire population, and everybody will adopt it or it will die away quickly? Nobody will adopt it, maybe very few people. How do we tell it? How do we say what happens and now; obviously, what would be coming in your mind is it actually depends upon what is traveling as well what is diffusing as well.

So, let us say it was a code snippet about some algorithm I am sharing not many people would be interested, but if it is a juicy piece of gossip about some Hollywood celebrity, Bollywood celebrity or rather one of my close friends it will very quickly diffuse through the network. So, yes it depends upon the product, but how do we actually quantify this complete process of a product or an information starting from some part in the network, infecting some more, getting adopted by some more and so on. So, we will look at it in detail in this chapter with the help of nice mathematical notations and conceptual entities which are coming next.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Prof. Anamika Chhabra**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Cascading Behaviour in Networks**  
**Lecture – 90**  
**Modeling Diffusion**

So, how do we model this process of diffusion? So, let us design a simple framework for this, probably a kind of game. What kind of game? Let me again give you a scenario.

(Refer Slide Time: 00:23)



This is a sunny Sunday and you wake up early in the morning and you want to decide how you are going to spend the rest of your day. And, you have two options either you go to the library and finish that assignment whose deadline is coming this Wednesday, or you go outside have fun, watch movies, do shopping and enjoy your day.

Obviously we would always want to do the second; obviously, we would always want to do the second sitting in library and reading a book is boring while going out and having fun is so interesting; that might be your option you go out and have fun. But, now let me broaden this picture and bring your friends in this picture. So, now you have many friends; let us say you have some 20 friends. Let me change your mind, you wake up this Sunday and you decide that you are going outside, and you are going to have fun.

But then comes into picture your 20 friends and out of these 20 friends 18 of your friends tell you that they are going to library and finishing that assignment which is coming up this Wednesday and they invite you. And, then here are your 2 friends which are asking you to come with them and go enjoy.

Now, what will you do you? You might think for a moment, you might now want to go to library and finish that assignment. So, how do we weigh our options? How do you weigh your options which choice to make what to do? So, the first factor which we have seen is how much that work interests you. So, in this case let us call it a payoff.

(Refer Slide Time: 02:13)



There is a payoff which working in the library gives you and there is a payoff which going outside and having fun gives you. And, for you, the payoff which you get in the library is a little bit lesser, then the payoff which you get when you go outside and have fun first thing. Second thing as we saw the number of friends is important, what your friends are doing is important. So, how do we capture this notion of friends?

(Refer Slide Time: 02:45)



So, let us consider one friend for simplicity, let us say just one friend in the picture. So, you have this friend Zara and both of you have to make a decision, what you both are going to do today. So, one option both Zara and you go to library and study and both of you get some payoff which is associated with studying in the library. Another option both you and Zara go outside and have fun you get another pay off which is actually high because, you enjoy more there.

Remaining two options Zara goes to library and you go outside; probably you do not get any payoff because you do not go along with each other. And, second option the reverse way you sit in library Zara goes and have fun outside, again results in no payoff. So, this tie these two people they get the payoff when they do the same thing and the payoff is equal to the payoff associated with that particular thing. This was about one friend.

So, if there were only if you had only this one friend Zara. So, the ideal situation ideal decision you both should have taken was to go out and have fun, but the situation is not that simple. You have lot of other friends in picture you do not just have Zara.

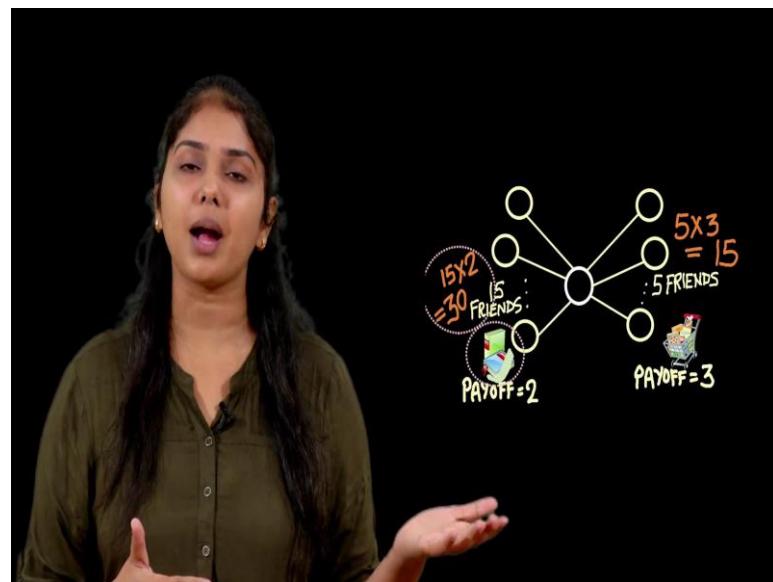
(Refer Slide Time: 04:16)



So, let us say you have some 20 friends here and out of these 20 friends 18 want to go and study in library and 2 wants to go and have fun outside. Now, how do you weigh your option? So, it is very simple. So, if you look at every tie here you are here and your friend is here and if both of you go to library, both of you get some payoff; let us say A associated with library.

Similarly, here are you and here is the another friend both of you go to the library and you both get a payoff of A and so on for all the friends. And, similarly if you look at the friends this side who are going to go and have fun outside. This is again if you go and have fun you get some payoff here, both of you get a payoff of let us say B which is associated with going outside and enjoying and similarly in this case.

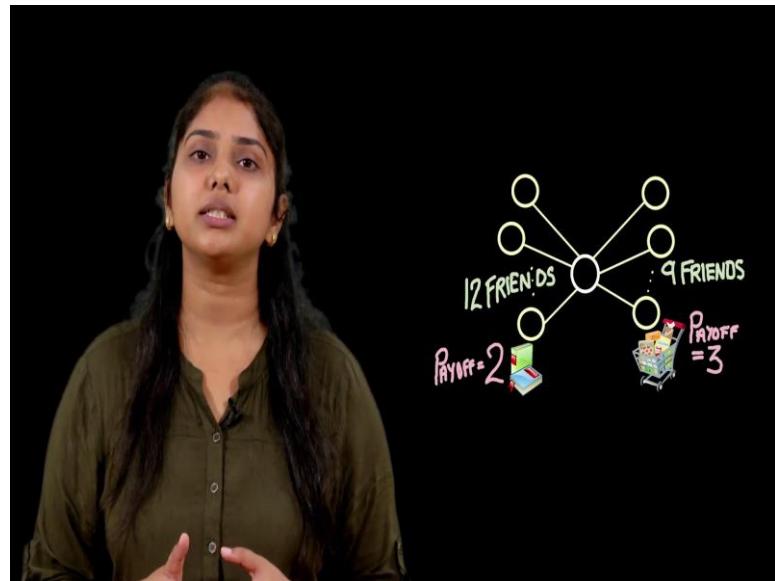
(Refer Slide Time: 05:12)



Now, what you will do is you will calculate the total payoff if you go to library and study what is the total payoff you get. And, if you go outside and have fun what is the total payoff you get. So, for the time being let us assume that the payoff associated with library is 2 and the payoff associated with going outside is 3. And, let us say 15 of your friends are ready to sit with you in the library and study while, 5 of your friends want to go outside and have fun.

So, how much payoff do you get this side is 15 into 2, 30 and how much payoff do you get the other side when you go outside and have fun is 3 into 5, 15. So, what do you do? So, you see that you are getting a better payoff when you are sitting in the library and studying with your friends. So, you take that particular path.

(Refer Slide Time: 06:15)



Let me change these parameters and ask you this, ask you a puzzle, ask you an exercise question and the question is: again assume that the payoff associated with sitting in the library and studying is 2 while, the payoff for going outside and enjoying is 3. 12 of your friends want to sit in library and study with you while, 9 of your friends want to go outside and enjoy with you. So, which action do you take in this place?

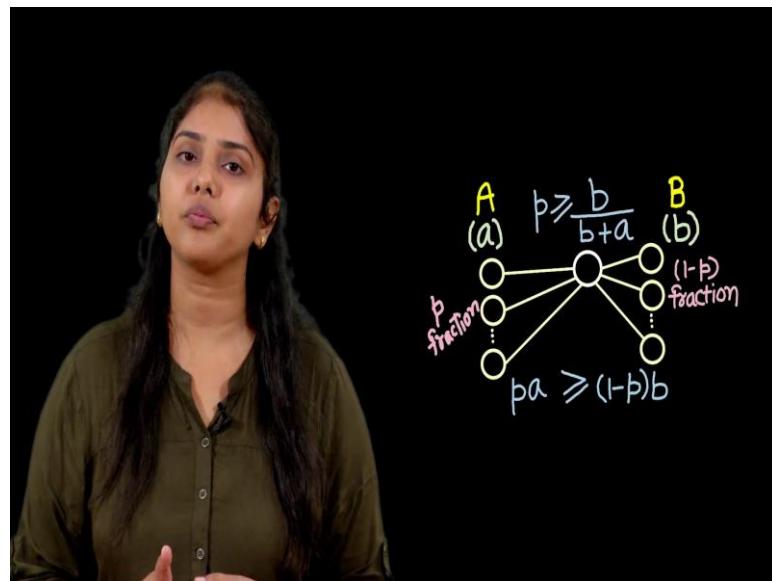
**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Prof. Anamika Chhabra**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Cascading Behavior in Networks**  
**Lecture – 91**  
**Modeling Diffusion (Continued)**

So, I hope you have correctly solved it. So, the answer is very simple. Sitting in library the payoff is 2 and you have 12 friends. So, it gives you a total payoff of 24. Going outside and having fun the payoff you get is 3, but you have 9 friends. So, you get a total payoff of 27.

So obviously, the second is better, and, in this case, you go outside and have fun. So, here we talked about one person how do you weigh your options. Let us define it a little bit more rigorously. So, I will use some mathematical notations.

(Refer Slide Time: 00:46)



So, assume that you have 2 choices, 2 actions A and B: capital A and capital B. The payoff associated with doing capital A is small a and payoff associated with doing capital B is small b. And then you have friends, so let us say B fraction of your friends have adopted capital A remaining 1 - p fraction of your friends has adopted the action

capital B. So, what is the payoff you get if you adopt capital A is  $p * a$  and what is the payoff you get if you adopt capital B is  $(1 - p) * b$ .

So, simple mathematics tells me that if you want to adopt A what should happen. If you have to adopt A, if the product A has to be adopted if the action A has to be taken  $p * a \geq (1 - p) * b$ , which says that  $p \geq b/(b + a)$ .

(Refer Slide Time: 01:58)



So, here you go to a value of  $p$ . So, the value of  $p$  is this much fraction of your friends have adopted A, you also adopt A. So, you kind of have a threshold here. Let us say the value of  $p$  here is 40%, 40/100; it means that if you have 100 friends if 40 of your friends adopt A, you will also adopt A you will see a very nice concept here.

So, let us say this 40% is associated with going outside. If 40% of your friends come and they tell you that they are going outside, you also go outside. And for going to library it is 60%, if 60% of your friends go to library, you also go to library. So, what is happening here?

(Refer Slide Time: 02:49)

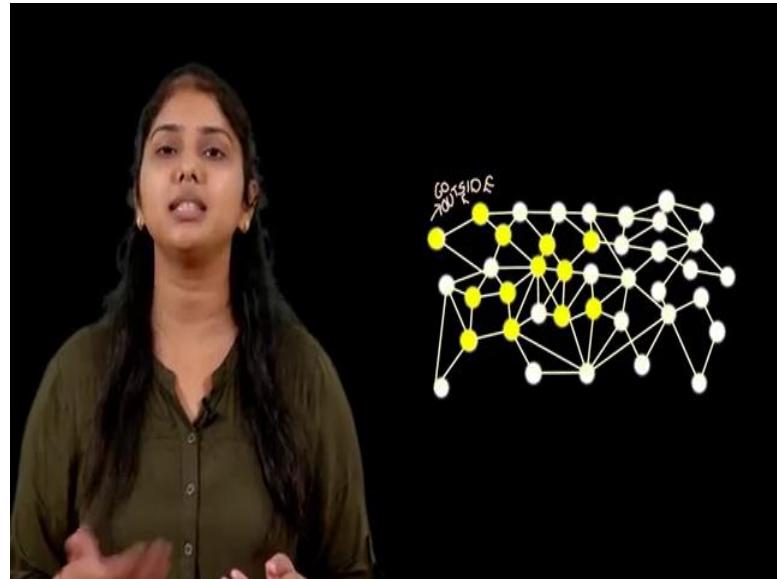


Assume that you have 10 friends, out of these 10 friends 4 should go outside for you to get outside. So, let us say let us look at it sequentially. One of your friend comes and tells you that he is going outside and previously you have decided that you will study in library and then one of your friend comes and tells you that he is going to go and have fun outside. You will not be bothered much.

Then second of your friend comes and then tells who is going to go outside, you will be like fine. Third of your friend comes and tells you he is going outside, you little bit feel that, so many people are going outside I should also go, but still somehow you refrain yourself because, you want to complete that assignment. But, then when now this fourth friend comes and tells you that he is going outside your mood gets completely changed you get ready and hang around with them. So, this phenomenon is known as social reinforcement. When more and more people come and tell us about something, we will tend to believe that information.

Let us extend this scenario from our self to the complete network. So, in the complete network it is not just one person who is deciding what to do everybody. So, it is not just we who are looking at each other and following them. It is like everybody is looking at each other and weighing their options and getting impacted by what others are doing and they are doing something. So, how does this complete thing operate?

(Refer Slide Time: 04:31)

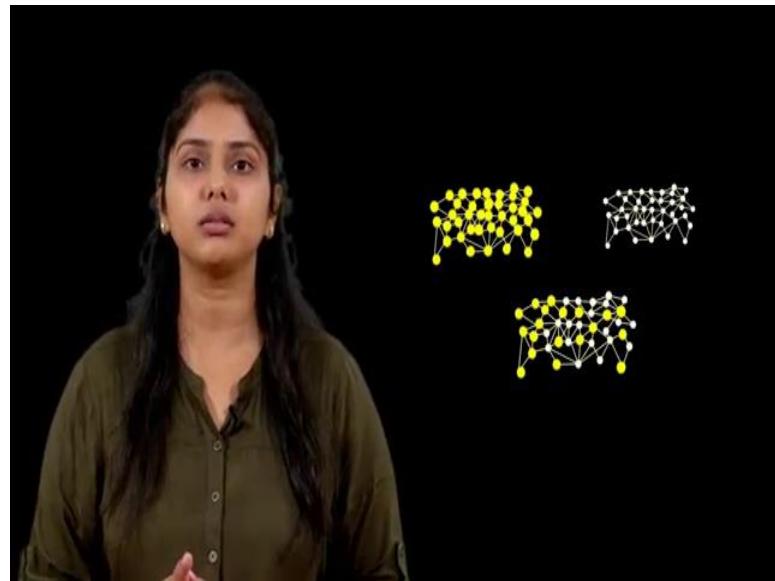


So, for understanding it nicely let us again take the same example, there is this class of 40 people let us say and they all have the they all are friends, their friendships between them and they all have this pending assignment on Wednesday. And the entire class has decided each and every student in the class has decided that they are going to sit in the library and complete this assignment. But what happens next is there are these 2 outrageous people in this class who says that we will do the assignment later, we are going to go outside since it is a Sunday and have fun.

So, the 2 nodes in this network they flip, and they change their mind from doing the assignment to going outside and having fun. And now what happens? Now everybody is kind of start changing their opinion. So, where everybody wanted to sit and work on this assignment, they look at these 2 people and see that they are going outside, the things in the class start changing.

So, every person one by one looks at these 2 people; one person might look and then decide fine, these two people are going outside I will also go outside and have fun with them. And then other person and then other person, so the entire class which has actually decided to sit for the assignment and do it, people start changing their mind and many of them start going outside. So, this is how it happens.

(Refer Slide Time: 05:56)



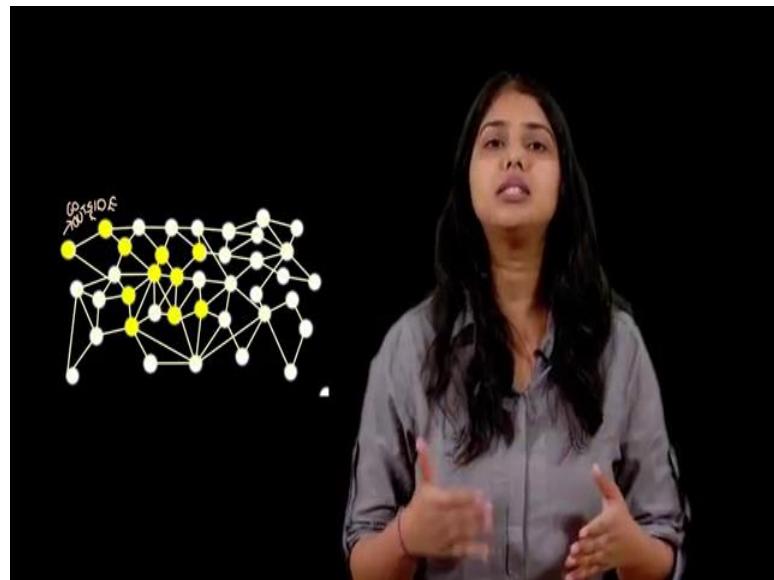
So, what do you think will happen at the end? What will, how will this network stabilize? Will everybody at the end go outside and have fun and nobody will be doing the assignment or people will keep flipping, flipping and ultimately the people who have thought of going outside and have fun they also change their mind? All of them sit together and do assignment or a third option what can be the third option? So, the third option is there emerges 2 parties in this class.

So, one party goes to the library and do their assignment and second party go outside and have fun, so it is also possible.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Prof. Anamika Chhabra**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Cascading Behavior in Networks**  
**Lecture – 92**  
**Impact of Communities on Diffusion**

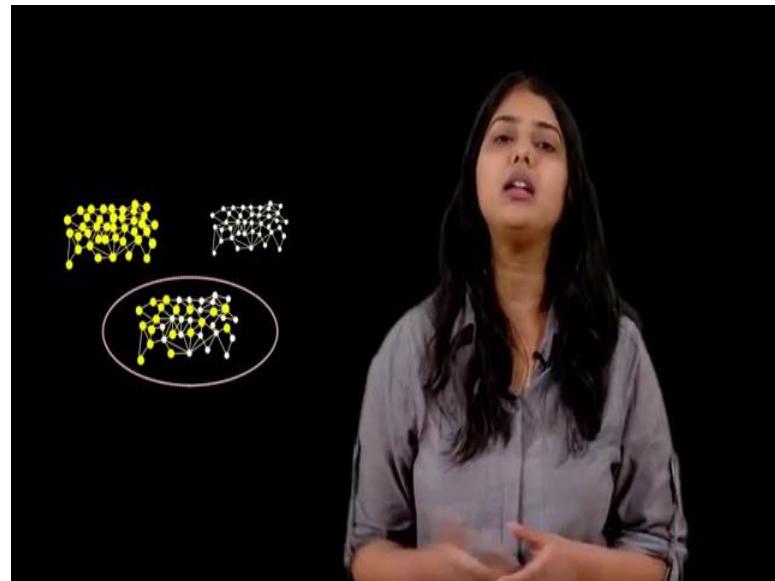
(Refer Slide Time: 00:08)



If we looked at this example where there was this class of 40 people. And initially everybody in the class has decided that they are going to sit in the library and complete that assignment which is pending on Wednesday, but then there were these 2 people who decided to do the assignment a little bit later and go outside and enjoy.

And then we looked at since the payoff with going outside and enjoying is higher, slowly, slowly people in this class they start flipping and more and more people decide to go outside and enjoy. And then we ask these 3 questions, where will be this stability? So, when this network is changing, changing in this sense the decisions of the people are changing. So, initially everybody wanted to sit in library and study, then these 2 people want to go outside, then maybe some 4 people want to go outside and so on.

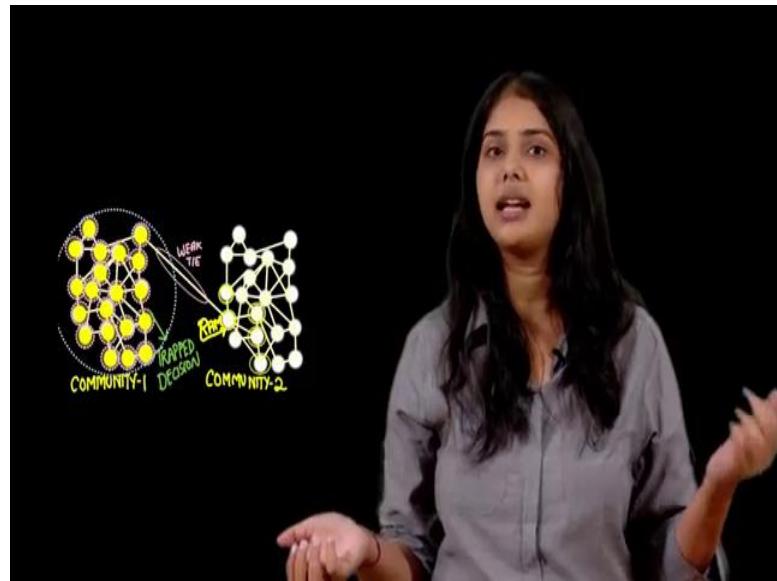
(Refer Slide Time: 01:08)



So, what happened at the end? What is the convergence whether at the end everybody has decided to go outside and enjoy or these two people have kind of looked at other people and they also decide to sit in library and study; that is at the end everybody sits in the library and study. But there was a third interesting possible outcome we have looked at and it was the class gets divided in 2 parties.

So, 1 party who wants to sit in the library and study and other party goes outside and enjoy. In which scenario can this third case happen that, the class gets split in 2 parties and 1 party sit in library and other party go outside. So, here I would like to take you back to a chapter on weak ties which was in week 3 and compare what is happening there. So, in the chapter on weak ties, we studied about communities.

(Refer Slide Time: 02:06)



So, assume that this class has 2 really strong communities; community 1 and community 2. And this was in community 1, that these 2 people decided that they are going to go outside and enjoy. Slowly, slowly people in this community start changing their decision and slowly and slowly this entire community wants to go outside, but we still have this another community, let us say C2, where everybody is still wants to be in library and study. And then there is this weak tie between these 2 communities.

So, in this community C2 currently everybody is going to library and study. So, how will people here start changing? So, these weak tie if we look at this person here, let us say Ram. So, Ram is the one of the endpoint of weak tie and somehow if Ram gets convinced to go outside then maybe people in this community also start flipping and change their decision, but you see changing Ram's decision is difficult because, Ram is here and most of the Ram's friends they have already decided to sit in library and study and there is only this one tie where this person is going outside and enjoying.

So, you see this community has kind of trapped these phenomena of going outside inside itself and the decision does not easily gets passed to this another community. So, there are these 2 communities, 1 is going outside and enjoying and another sits in the library and study. So, till now we have looked at this example where there were 2 actions; each person can take going to library or going outside. And the same phenomena repeat itself and it is about something else. So, let us say it is about the adoption of 2 software. There

is this college, let us say where everybody is using an operating system, say Windows and then and all together new operating system very nice operating system with new features come in the market. And then a few people adopt it and then people start flipping and changing to this new software.

So, this is the diffusion phenomena which happens in almost all the cases, but do you see a problem here? The problem here when any new technology diffuses on a network the actual problem is with getting started, so in the beginning when the entire population is using windows, it is risky for somebody for 1 person in this population to adopt this new operating system and start working with it. That is since it is risky, so people they resist adopting a new technology or a new innovation. So, let me give you one example. So, the example is very fictitious.

(Refer Slide Time: 05:05)



Let us say a scientist from NASA comes to your school and he has a big-time machine with himself and he invites anyone of you. He says that I invite one of you to come sit in my time machine and maybe go back in your life change something about it or do something or this might give you a very fruitful result, but I ask you will you be daring enough, will you take that risk of going inside and then sitting in his time machine? No, probably no, nobody will come forward.

(Refer Slide Time: 05:42)



But now, imagine a world a highly technical world like we see in Hollywood movies Matrix every now and then there is time machine. People sit in this time machine go back change their future and happily enjoy their life. So, there it will not be very difficult for you to adopt this new innovation. So, the there is only the starting trouble for any product or any innovation to come in the market. We only need a few people to adopt it in the beginning and then rest of the diffusion is kind of easy, people look at each other and start following as we discussed and this innovation spreads in the network.

So, how do we overcome this starting trouble? So, one way of overcoming this starting trouble is to increase the payoff associated with your product. So, as payoff increases, people see that if they adopt it they are going to get a lot of advantage. So, probably initially people will be likely to adopt it more, but that might always not be possible. So, the second very nice option is to use the key people in the network.

So, instead of trying to advertise your product or your innovation to every single person in the population, you carefully choose a few key people and infect them with your idea or your innovation. What am I saying? I am saying that let us say you want a new bike that a new bike in the market probably which all of your friends are having and you want to look cool in your college and you want that new bike.

Should you really go and convince your sibling, or you should you go and convince your mother? Actually no, whom should you be convincing, you should be convincing the

head of your family, your father and once if you convince your father, it does not actually matter whether your mother or your sibling is in your favor or not they look at your father and they will adopt what he is saying.

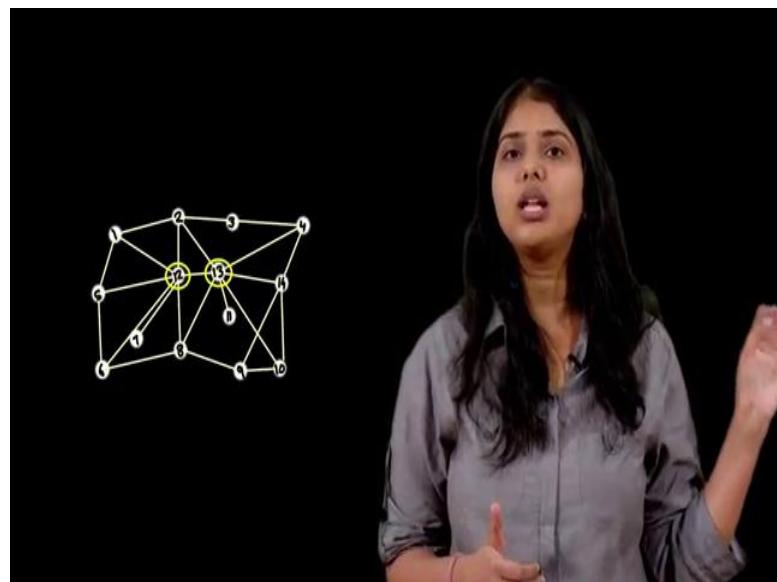
So, similarly in a network if you want to advertise your product or your technology choose those key people. So, that is why this is the reason why we see a photograph of Shahrukh Khan on the Chavan Prash, Chavan Prash box or the photo of Amitabh Bachchan on the Navratna [FL] and stuff like that.

(Refer Slide Time: 08:08)



So, this particular thing is popularly known as viral marketing.

(Refer Slide Time: 08:13)



So, given a network and given the people in this network, identify this bunch of key people, if you which if you convince your product gets spread really very, very fast. So, if you see in this example, we have many nodes many nodes, but if somehow here you convince node 12 and node 13 to adopt your idea. Probably everybody will soon start adopting it because, these 2 nodes are very well connected to rest of the network. We just looked at the importance of key people in a network.

So, we looked at if we convince these key people in the network, it is beneficial to us in the way that my product or my innovation gets a way to quickly spread on this network. And, we can use this very same strategy in our previous example which we were discussing.

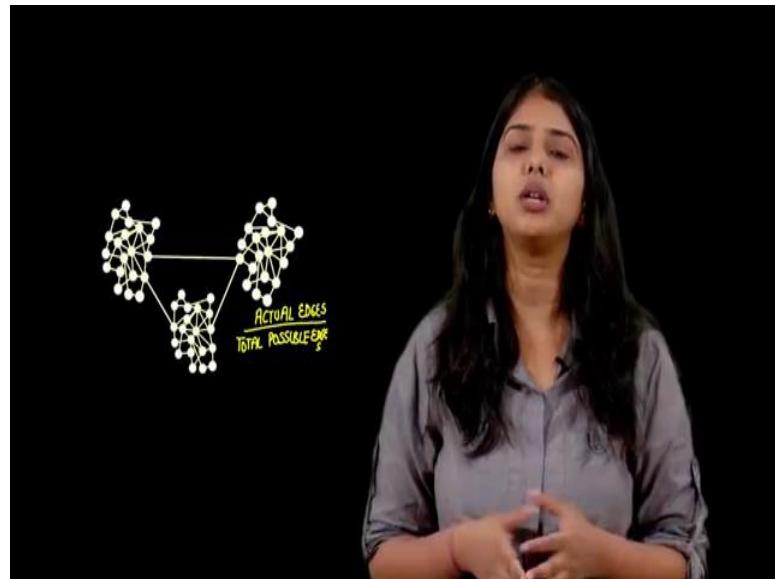
(Refer Slide Time: 09:09)



So, we have looked at here were 2 communities and, in this community,, community C1 2 students they had decided that they are going to go outside and have fun. And slowly, slowly this entire community decides to go outside, but this community remains unaffected by the decision here. Because, there are not enough number of links here to transfer the decision and probably this weak tie is not working here because, it is a single link. And the person at the other side of this weak tie has more number of friends, who are studying in the library and not going outside.

So, what we can do in this case if we want the people of this second community also to go outside and have fun. So, we looked at some key people in this community and convince them to go outside and probably we will be able to attain our objective.

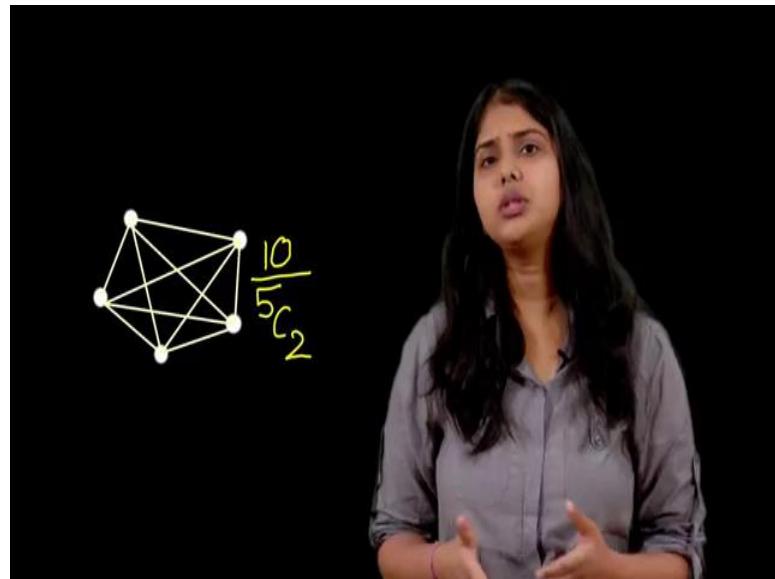
(Refer Slide Time: 10:09)



Here I would like to ask you a very interesting question. Do you think that when this big social network is divided into many, many communities and something is spreading on this network some product, some idea, some information is spreading on this network does the density of a community, so here we have a community do the density of this community has anything to do with? Whether this idea will spread on this community or not? What is the density of a community you will be wondering?

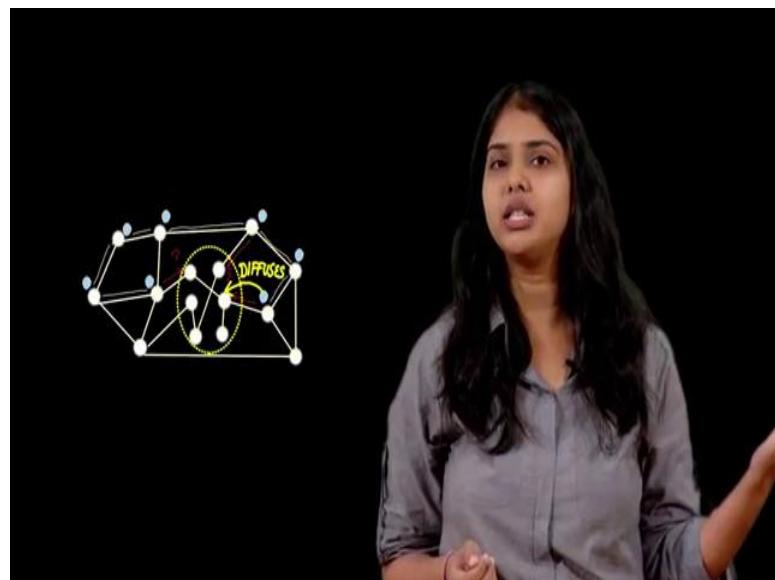
So, density of community is mainly tells you how well connected this community is. Mathematically speaking it is the ratio of the number of edges which exist in this community divided by total possible number of edges in this community.

(Refer Slide Time: 10:56)



For example, let us say there is a bunch of 5 people and they are forming a community. If they have 10 links among them, so their density the density of this community will be 10 divided by 5 choose 2. So, that is a density. So, do you think the density of a community has something to do with? Whether this information will spread in this community or not? So, I will again make this question a little bit clearer.

(Refer Slide Time: 11:26)



Here is this community and it is having some density and here is this entire network some information is spreading on this entire network and he is trying to enter into this

community. So, does the density of this community has something to do here? Whether this information will be able to come inside and infect these people or it has nothing to do with this?

So, to help you answer this question, I tell you I will not tell you I will revise a small story most of you would be knowing. So, there was this king who was about to die and he had a big kingdom which needs to be looked after when he dies. And he had these 3 sons who always used to fight each other, and he wanted to teach them a lesson. So, when he was trying, he was on the deathbed, he asked his sons to come with a bundle of sticks, wood sticks.

And then they come with a bundle of wood sticks and he gave one wood stick to each of them and asks them to break it. The first person tries breaking the wood stick it easily gets broken, second wood stick easily gets broken third one also, but then he gives them all the wood sticks together as a bundle and asks them to break it. But then it does not get broken. So, there is an important lesson that there is power in unity. What it has to do with our question?

So, do you see here that this community here, if the people here kind of do not like each other or not many people are friends with each other, then if somebody will say something from outside, probably they will believe. And, they probably they will adopt this idea, but if they are good friends with each other it will be very difficult to convince them, it is very intuitive as well.

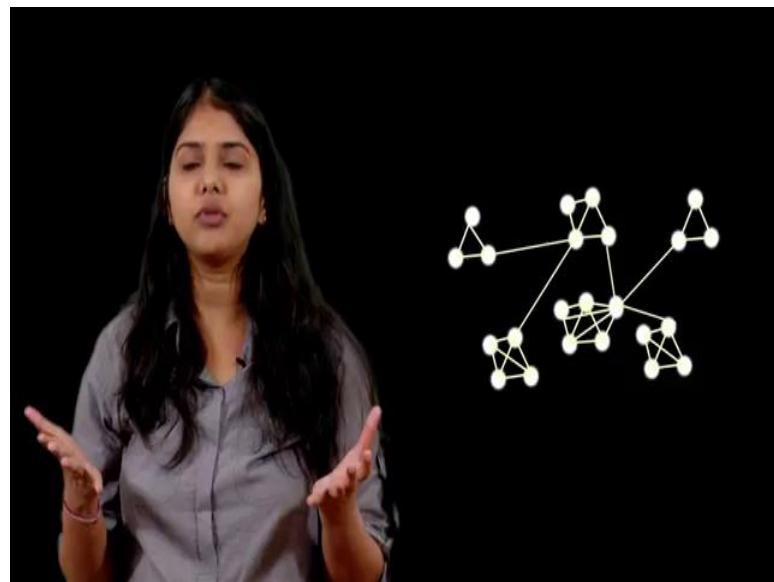
So, here are these bunch of 5 people who are really, really close friends and then they have decided to sit in library and do their assignment; no matter how hard you convince them to come outside with you and enjoy they probably will not come, but if their friendship was not that strong 1 or 2 of them would come outside with you. So, do you see here the role of the density of a community in diffusion? Higher the density the difficult it is for an idea to be injected in this community.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Prof. Anamika Chhabra**  
**Department of Computer Science**  
**Indian Institute Technology, Ropar**

**Cascading Behavior in Networks**  
**Lecture – 93**  
**Cascade and Clusters**

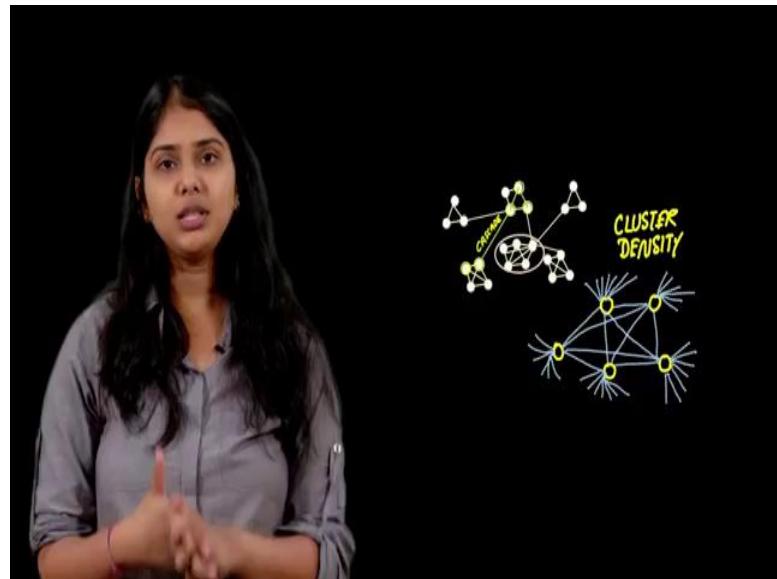
So, we have just looked at a story and have very easily said that, if there is a community having a high density probably and piece of idea traveling through this network cannot enter this community. We intuitively know this, but now let us try to look at it mathematically and it is actually very easy. So, to prove it mathematically, let us first refine a scenario and revise our definitions. Let us refine our scenario. So, our scenario was we had this class right; previously we looked at just 2 communities.

(Refer Slide Time: 00:42)



We said that this class where people are deciding whether to be in library or whether to be outside and then we looked at just 2 communities. What we do now is, let us take let us make this a little bit more detailed. Instead of just having two communities let us have a lot of small, small clusters in this entire network.

(Refer Slide Time: 01:01)



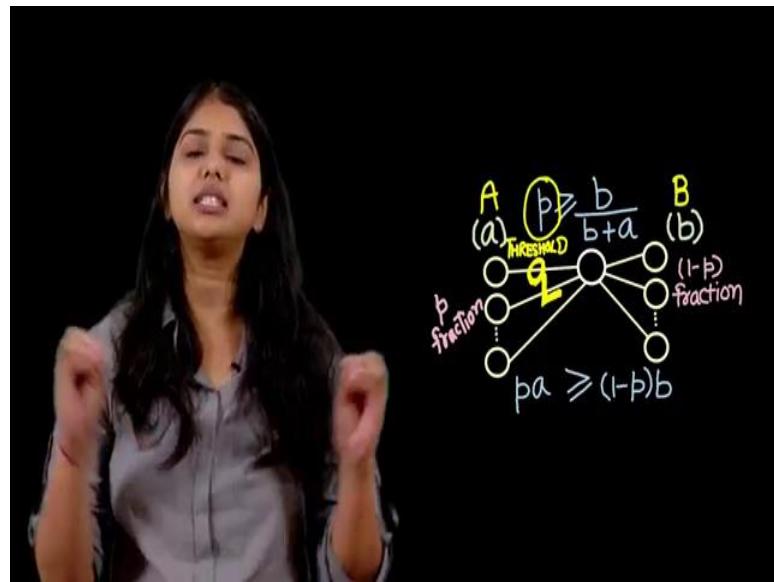
So, here is this class and people have made small, small clusters, it is actually what generally happens. A class is generally not split in 2 halves, people have their own bunch of friends and if it is a class of 100 people everybody has some bunch of 6 or 7 friends and people exist in small, small clusters.

And now the question which we ask is, in this class there are these 2 people somewhere in some part of this network who decide to go outside and have fun. And the question which we are asking is, will this idea of going outside this decision of going outside and having fun quickly sweep the entire class or will here be a bunch of people 1 cluster who despite whatever will not be able to convince to go outside? So, despite all of the rest class despite the complete of the rest class is going outside, these people stay together and say that we are not going outside and hence this diffusion process. So, we call it a cascade.

Now, whenever an idea or information diffuses, this the trajectory of this diffusion we get we call it a cascade as shown in the figure. So, we look at whether it becomes a complete cascade or not. Simply whether everybody in the class will decide to go outside and have fun or this bunch of people will not be convinced. So, this actually used to happen in our B. Tech class where it was not about sitting in library or going outside, it was about whether to have a bunk or not.

So, whether to have a bunk or not and then there were 3 girls in the class whose friendship was really, really strong and they would always say that no we are not going to go for bunk and the entire class had to cancel their decision. So, keeping this example apart, just for fun our question was whether this will be a complete cascade or not. So, for looking at it let us revise our definitions.

(Refer Slide Time: 03:18)



So, we have previously seen that if we look at 1 node, 1 person in this class, this person decides whether to go to library or whether to go outside based on the payoff functions and the number of friends. So, we have there come up with a value called threshold. And we have looked at if the fraction of the, fraction of these persons friends who are going outside exceeds this threshold, this person will also go outside.

So, it was for, so it is 40% let us say, so it means that if 40% or more of my friends are going outside and having fun I also go outside and have fun. And if 60% of my friends have to, are sitting in library and working then I sit in library and work. So, the threshold is 40% for going outside and 60% for staying in the library and working. So, this value threshold we denote it by a number let us say  $q$ . So,  $q$  is a threshold associated with every node. So, at least  $q$  fraction of this nodes friends should be convinced, should be adopting a particular decision for this node to adopt the same decision.

Now, what was our question? It was related to the cluster. So, we have these clusters in this class and in every cluster, there are some nodes and each node is having some

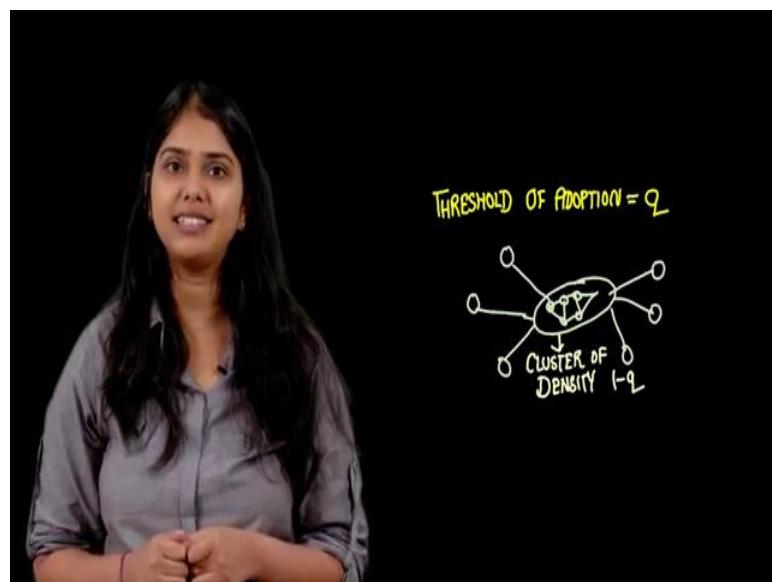
threshold value  $q$ . So, here we are assuming that every node is having the same threshold value  $q$ , I will come in on it later. So, every node in every cluster is having this threshold value  $q$ . Now the next important thing as we know which needs to be captured is like we call the density of the community or the amount of unity.

So, here we defined something which is called the density of a cluster. What is the density of a cluster? So, I will give you a one liner let us see. So, the density of a cluster is we say that the density of a cluster is  $D$ , if you look at every node in this cluster and at least  $D$  fraction of these nodes friends is in this cluster itself. It might be a bit difficult let us look at it with an example.

So, let us say that here is this cluster of 5 people. So, there are these 5 people in this cluster and let us say I say that the density of this cluster is 0.3 if I look at every person in this cluster and 30% of the friends of this person is in this same cluster. So, let us say that in this cluster everybody is having 10 friends.

So, I look at this node A out of these 10 friends 3 should be inside this cluster. For B out of these 10 friends 3 should be inside this same cluster and so on for the rest 3 of the nodes. So, that is what we mean by density of a cluster. So, I hope the definition is clear now.

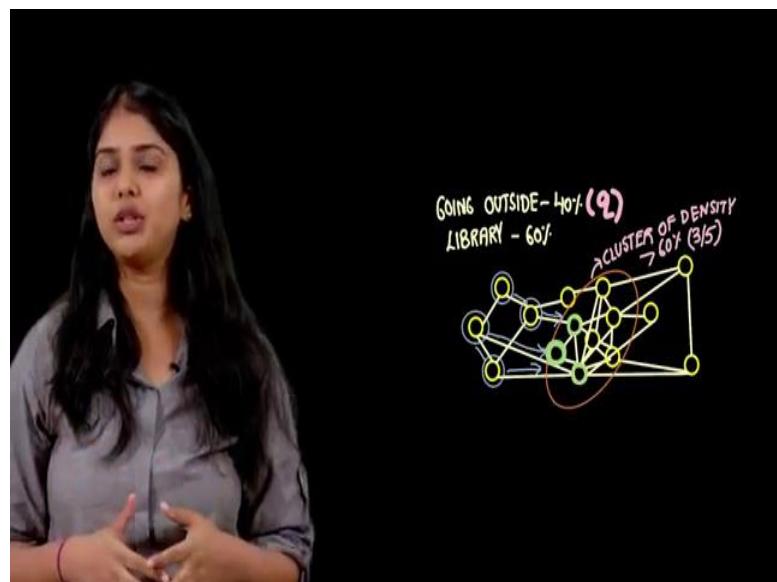
(Refer Slide Time: 06:34)



Now, I am going to give you a very weird claim and the claim says that in this network where the threshold of adoption for every person is  $q$  as we said before. Then the cascade cannot complete itself, complete itself means starting from some nodes it cannot sweep the entire network, the cascade cannot complete itself. If there exist a cluster in this network of density greater than  $1 - q$ .

So, how trivial or nontrivial is that? How do we prove it? So, the proof is very straightforward. We will let us look at it with our example, that will make it easier.

(Refer Slide Time: 07:15)



For example, what did it say that threshold for going outside is 40%, threshold for staying in library and studying is 60%? And, then in this entire class having different, different clusters, these are those 2 people who decide that they are going to go outside and this idea of going outside starts diffusing on this network.

Now, assume that there is this cluster of density greater than  $1 - q$ . So,  $q$  is here 40%. Cluster of density greater than  $1 - q$  means that, there is a cluster of density greater than 60%. What does that mean, cluster of density greater than 60%? It means that if I look at a cluster here and look at every person here 60% of their friends are in the same cluster.

And we know that initially the complete class had decided to stay in library and work and here these two people have changed their mind and they said that we are going to go outside. Now this idea of going outside is diffusing through the network. And let us say

while diffusing this idea comes close to this cluster. We are talking about and tries to enter this cluster.

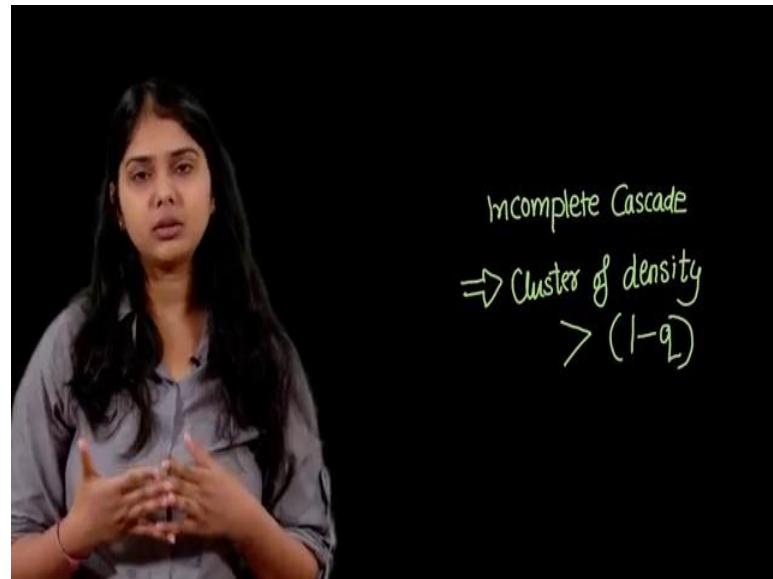
Now, see what happens. When this idea tries to enter this cluster, there is this node So, this node since it belongs to a cluster of density  $1 - q$ , density greater than  $1 - q$  that is density greater than 60%. 60% of his friends are in the same cluster, which means that 60% more than 60% of its friends have already decided to sit in library and study.

So, the threshold condition is not reached. For this node to go outside the number of its friends going outside, the fraction of its friends going outside and enjoying should be greater than 40%, but that cannot be achieved right. Because, 60% of its friends are already in the same cluster and have decided to stay in library and work.

Similarly, this happens for the second node of this cluster, third node of this cluster. So, this cascade is unable to enter inside this cluster and hence you see our claim is achieved. So, if there is this cluster having density greater than  $1 - q$ ; no matter what this cascade which was occurring on this network cannot enter inside this cluster and hence cannot complete itself.

Now I am going to give you another weird statement and let us see how we prove it. So, the first statement, the first claim which we made here was if a cascade is diffusing, if a an idea is diffusing on a network and if there is a cluster of density greater than  $1 - q$ , then the cascade cannot complete itself.

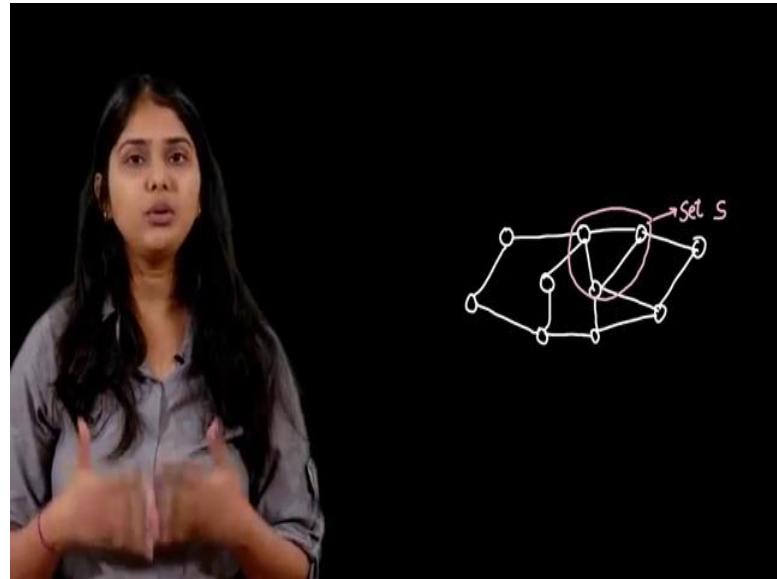
(Refer Slide Time: 10:27)



Now, what am I going to say? I am going to say that I give you a network and I give you the information about the cascade and I tell you that the cascade on this network is not complete. There is something which happened inside this network and the cascade could not complete itself, there are some nodes which did not adopt the idea; it means that if this happens that the cascade is not complete.

This implies that this shows that there is a cluster of density greater than  $1 - q$  in the network. So, it is a by implication, if you know propositional logic else, we can just leave it aside. So, it is both ways. So, I leave it as an exercise problem to prove it. It is actually very intuitive like the similar way we did this proof. Let me give you a small hint.

(Refer Slide Time: 11:17)



So, when this cascade is diffusing on this network and we say that the cascade is not complete, it means that there are some nodes in this network who have not adopted this idea of going outside. Why did not they adopt this idea of going outside? Because, their threshold was not reached; so, let us represent these set of nodes as set  $S$  who did not adopt your idea.

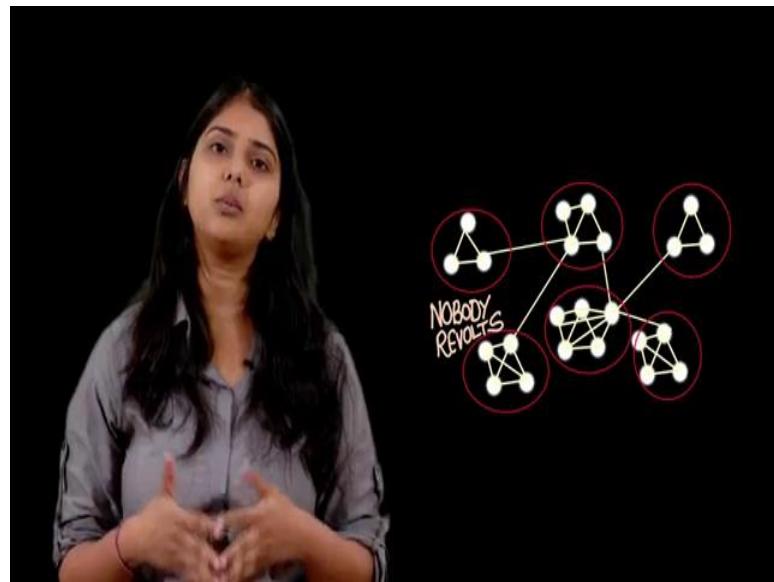
So, these people who are staying in library and studying, we represent them as a set  $S$  and then one by one you examine this person. Why have this node not adopted the idea is because, its threshold is not reached, so using this you can prove this another side of the claim. So, I leave it for you to prove it.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Prof. Anamika Chhabra**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Cascading Behavior in Networks**  
**Lecture – 94**  
**Knowledge, Thresholds, and the Collective Action**

So, we have looked at this beautiful example till now, where there was a class and there were these 2 competing decisions, competing actions which the people in this class could have taken. So, they could have decided to go outside and enjoy, or they could have decided to sit in the library and work. Now I will give you a similar, but a little bit different situation.

(Refer Slide Time: 00:31)



So, we also looked at here the case where the people in this class they are sort of divided into small, small clusters, where the interaction between two clusters is very less. And the interaction inside a cluster is strong. So, which leads to the, this problem sometimes that one cluster does not adopt the idea which others are saying and goes by itself. And now let us take the same scenario and let us take the same bunch of people, the same clusters and put them in another scenario the scenario for company.

So, assume that there is this multinational company and then there are these people who exist in clusters. So, here are different, different clusters of people who do not communicate much with each other. Now assume that the manager of this company comes up with a stupid decision or a heavy decision which these people are not ready to take. So, every person in this company has a problem with his decision, but they are unable to revolt. Why? So, you see here that the people of this company here exist in clusters.

So, this cluster is ready to revolt. Independently this cluster is ready to revolt. Independently this another cluster is ready to revolt, third cluster ready to revolt, all clusters are ready to revolt, but these clusters are small, and they are suspicious about whether the other clusters are ready to revolt or not.

So, you see what is happening here, it is the cluster is ready to revolt, but it will be a problem if this cluster alone revolts it can be very dangerous. So, this cluster all though everybody is ready to revolt, but since this particular cluster does not know that everybody else is also ready to revolt. Because, it has a less communication with the other clusters it cannot go and revolt against a manager because of this fear of getting fired.

Now, you see what is happening here. So, that is probably the reason why companies they do not allow much of the communication between different, different groups of people working there. So, it is in a sense breaking the unity, it is in a sense the divide and conquer strategy because, if you allow everybody to get united, they can take this action and revolt. But if the people live separately in different, different clusters and do not communicate much with each other it becomes difficult to revolt.

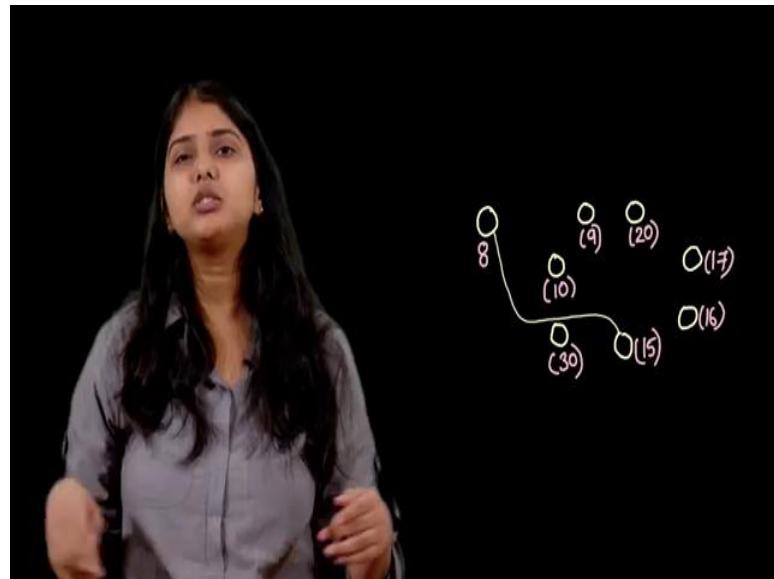
Same is the scenario as so you will see here the importance of this collective decision. What do I mean by a collective decision? So, assume that now the government has come up with a policy which I am not happy with. So, what if tomorrow morning I go on the street alone and start shouting, what will happen? Either I will be declared mad or I will be put in jail, but if with me there are these 100 people revolting then it makes some sense to revolt.

So, what I will be doing is I will be looking at these other people whether they are ready to revolt or not. And if I am suspicious that yes these people are not going to unite with

me I will also not go and revolt. Same is the situation of let us say people want to put some people want to put a coffee machine in an institute, but then there should be enough number of people who want that coffee machine only then this institute is going to fund for that machine. So, this kind of spatial scenario is known as collective action.

So, how do we model this scenario? It is a little bit different from what we have discussed previously, but modeling it as previously is also very easy. So, how do we model it? So, for modeling it, first of all we assume that every person has an intrinsic threshold.

(Refer Slide Time: 04:30)

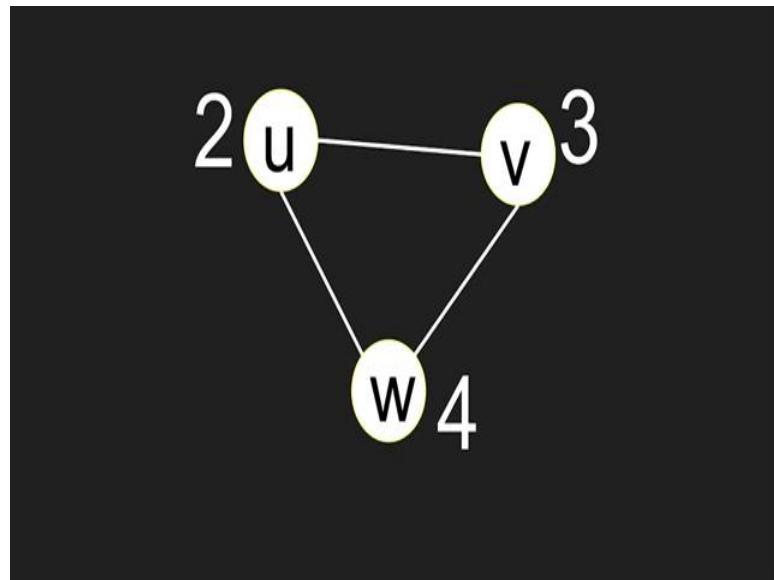


What do I mean by intrinsic threshold? So, say I have an intrinsic threshold of 8. So, it means that if including me, I say that if 8 persons are ready to revolt. So, 7 others total 8, 1 is me 7 others, so if 7 more people are ready to revolt, I will go ahead and revolt. And then there will be a threshold associated with every person.

So, each of the person will have a different, different threshold. Somebody will say I need 20 people at least to go and revolt, somebody can say that I need 40 people to go and revolt. This is the first, this is the first parameter of the model that we all have an intrinsic threshold. Now how do we decide what to do? So, to decide what to do, here we kind of know: what is the threshold of our close friends.

So, if I have a friend, let us say Mira, I will know that Mira is a kind of girl, she will require at least some 15 people to go and revolt. So, here I have an idea of my threshold, I have an idea of my friend's threshold and based on just these two ideas I have to decide whether to revolt or not. So, in the next screen cast I am going to show you 3 situations and in those 3 situations we will see whether the people they collectively revolt or not.

(Refer Slide Time: 05:58)



So, let us look at the screen casts for the collective behavior. So, in this screen cast we will be looking at 3 cases and we will see whether the in these 3 cases there will be a protest or not. So, what was our model? According to our model every node has a threshold which is it in like the threshold for u is 2, for v is 3 and for w is 4. And what does threshold mean? Thresholds mean that this node needs at least these many people to participate in the protest only then this node is going to participate in the protest. So, the node u needs 2 people to participate in the protest.

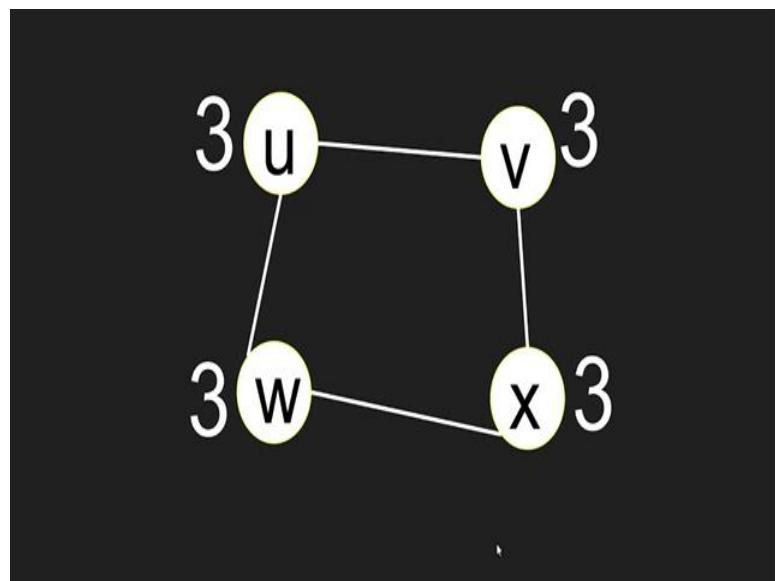
So, 1 person 2 imputing itself, so 1 obviously will be node u and then it needs 1 more person to participate in the protest for it to participate in the protest. Let us see what happens in this case. Let us look at node w here. So, the threshold for node w is 4 and w has only 2 friends right. So, even if both friends participate in the protest, w cannot participate because, it needs at least 4 people, so w drops the idea of participating.

Let us look at node v. It needs 3 people. So, node v looks at w and then it knows that w cannot participate in the protest because, its threshold is 4 and these many people are not

available. So, node v means node v now knows that w is not going to participate. So, again its threshold will not be reached, and node v drops the idea to participate. Node u only requires 2 people, right? Only requires 2 people to participate, but then node u it knows the thresholds of both nodes right.

And then hence it can determine that none of this people neither v nor w is going to participate. Hence node u also decides to drop the idea of protesting. Hence the protest does not happen in this case. Let us look at another example. So, if we look at this example, we can directly look at this example and we only can tell that yes node w needs 4 people which are not available and hence the protest cannot happen in this case.

(Refer Slide Time: 08:21)



But if we look at this case, it seems quite easy for a protest to happen. So, any 3 nodes along this cycle can get together and do a protest; u, v and w if they get together they can do a protest, w, x and v can do a protest and so on right, u, v and x can do a protest.

But then every node has some incomplete knowledge, they cannot see the entire network. For example, node u here has no idea about node x right. So, node u cannot see this entire network and then come out with the conclusion that yes, it can go get together with v and w and protest. So, let us see how it happens at the level of every individual node.

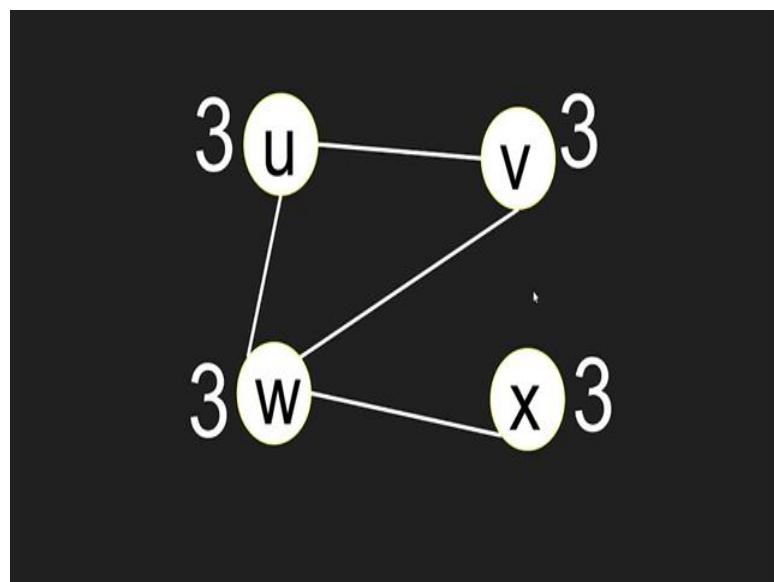
So, if we look at node u. Node u knows that node v and node w. They have a threshold of 3 right. And hence they can get together and do a protest, but then the problem is node u

also knows that see, v and w both are the friends of node u and then node u knows that v and w are node friends. They do not know about each other's thresholds. Node u knows that w has a threshold of 3 v has a threshold of 3, but then node v and w have no information about each other right.

So, we do not know what they will be doing and then node u does not have any idea about the threshold of node x. So, node u can think that may be the threshold of node x is very high let us say 5. So, if the threshold of node x is very high, here it is 5 then what will happen? v might decide node to protest because, x will not be protesting because, it has a very high threshold, then v will not be protesting and then how can you know this node u protest here right.

So, it is unsafe for you to participate in the protest and it drops its idea of participating. And the same thing happens with all of these three nodes v, x and w because, v does not know about w's threshold, x does not know about u's threshold, w does not know about v's threshold. So, even if in this case actually they can get together and do a protest, but then when these people look at the networks from their perspective, they drop the idea of protesting.

(Refer Slide Time: 10:42)



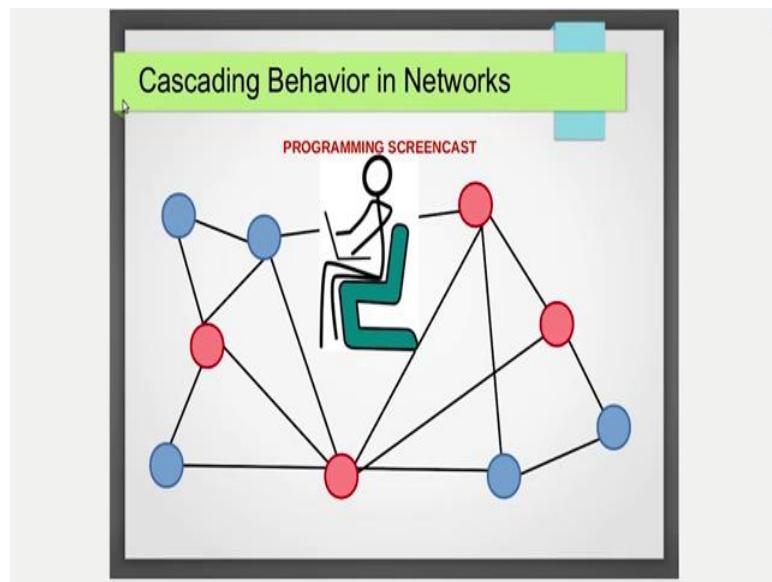
Let us now look at this case right. So, we have rewired this edge from v to x to v to w and what happens in this case. Can you guess? Yes. A protest can very easily happen here. Can happen here and these people you can see that a protest can happen here; u can

look at v and w and then look at their thresholds. And then here you can be sure that yes v and w will participate because, v and w also know about each other's threshold. So, here mainly these 3 people u, v and w can get together and go for a protest.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

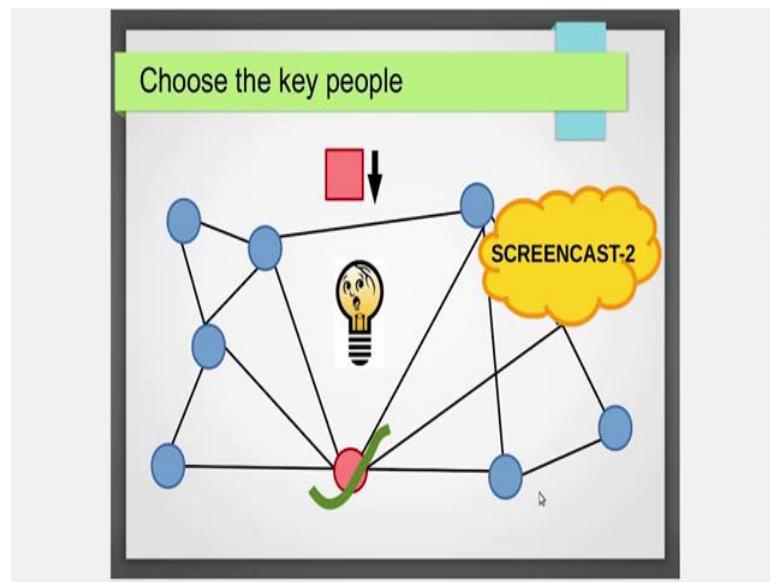
**Cascading Behaviour in Networks**  
**Lecture - 95**  
**An Introduction to the Programming Screencast**  
**(Coding 4 major ideas)**

(Refer Slide Time: 00:05)



So now, we had shifting to the coding part, the programming screen cast part for the chapter Cascading Behaviour in Networks we have discussed just now. What we will be doing here in this programming screen cast is, we will be taking the major concepts that we have discussed in this chapter. So, I will be taking 4 major concepts that we have discussed in this chapter and we will be coding and validating all those 4 major concepts one by one. So, I will quickly recap these concepts for you and then we will start with the coding part.

(Refer Slide Time: 00:38)



In this entire chapter we talked about a scenario and the scenario was, there is a network and on this network every person has adopted some action or some behaviour. So, every person has adopted the same action or same behaviour, and everybody was living comfortably with it on this network, but then with time comes a new idea, new action or new behaviour in this network and then, people start changing.

This new idea or new behaviour starts cascading on this network. So, the example which we have discussed in the chapter is there is a network on a class people in a class, where all of them have initially decided to work in library or an pending assignment, on a pending assignment. And then, came this new idea of going outside and having fun and then slowly and slowly this idea started cascading and more and more people started going outside and having fun, instead of sitting in library and working.

So, that is the exact thing which we will be talking about. So, as you can see that here is a network. Initially, all the nodes in this network were blue. So, they have adopted an action and idea, a behaviour; let us call it a blue action, blue idea or blue behaviour, but then comes in picture a red idea, a red action, a red behaviour and it wants to defuse on this network. And then we have looked at the obvious problems with this. What is the problem with this blue idea, blue sorry, what is the problem here with this red idea to defuse on this network and make all the nodes red here? The problem was, as we discussed that people find it risky to adopt a new idea or a new behaviour. They are not

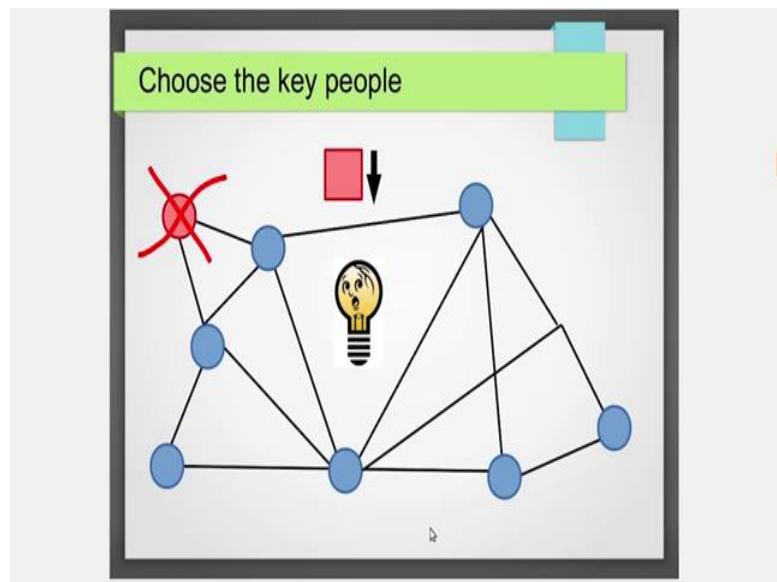
comfortable shifting to a new thing specially, when all of their friends are doing the old thing. So, no node in this network actually wants to shift to this red idea; they find it risky to adopt.

Then what do we do? What should this red idea do to defuse on this network? And we have looked at 2 solutions to this problem. The first solution was to increase the payoff associated with this red idea, to increase the benefits associated with this red idea. As the benefit associated with this red idea increases, more and more people start adopting it. So, you have looked at how we have modelled the entire thing with the help of pay off and number of friends. Each idea here let us be the blue one or a red one gives you a payoff. And then, based on that payoff and your number of friends which I have adopted that idea you decide, what to do right. So, you increase the payoff associated with this red idea and probably it will cascade on this network and rather cause a complete cascade.

What is a complete cascade? Complete cascade is when every node in this network turns red, this red idea is adopted by all the nodes in this network, we call it a complete cascade. There will be modelling; we will be modelling it in screen cast 1 sorry. So, what we will be doing in screen cast 1 is we will take a network where everyone has adopted an old idea and then, a new idea comes which is having some payoff which is; obviously, greater than the payoff of the old idea, but still on a lower side. And, we see that with this payoff this new idea is unable to cascade on this network. It is unable to cause a complete cascade on this network.

But as we keep increasing the pay off associated with this new idea finally, it creates a complete cascade. We will be looking at it in screen cast 1. So, screen cast 1 is associated with the first idea, you increase the pay off associated with this new idea.

(Refer Slide Time: 04:40)

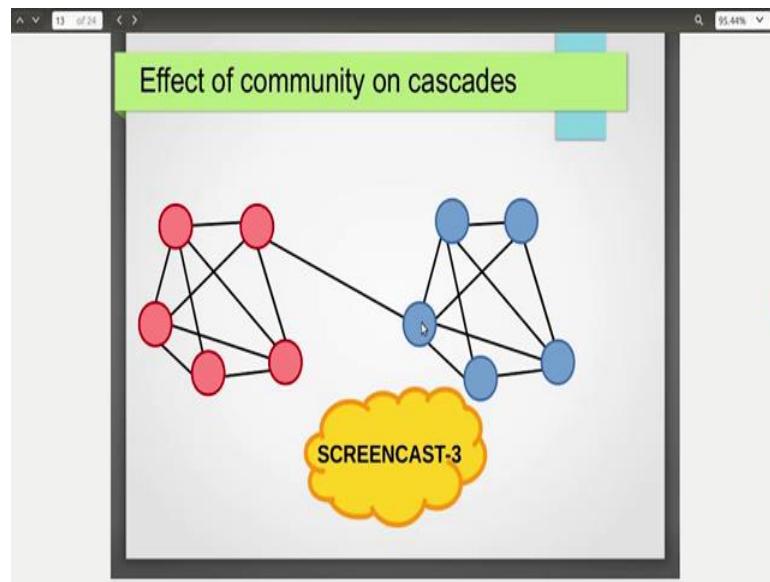


What is the second idea? What was the second idea that we discussed? The second idea was about choosing the right people. So, maybe if you start with this node here you are unable to create a complete cascade or if you start with this node here you are unable to create a complete cascade. But if you start from this node here which is very well connected to rest of the network you might end up creating a complete cascade.

So, if you start with some nodes in this network, the cascade does not occur. If you start with some other nodes in this network the cascade might occur. And, I am quite sure you remember this we talked about this bike example. If you want a new buy it should not be your mother here or your siblings here which you should be convincing, it should be your father. So, this thing we will be discussing in screen cast 2 where we will again all do almost the same thing.

We will have this network which on which every person has adopted an old idea and then, this new idea comes and now, we will keep the payoff associated with this new idea to be same. We are not going to change the pay off, but we are going to change the people from where this cascade starts and which show that if you start from some bunch of people, it is unable to create a complete cascade, but if you start from some other bunch of people it is able to create a cascade. So, we will be talking about it in screen cast 2.

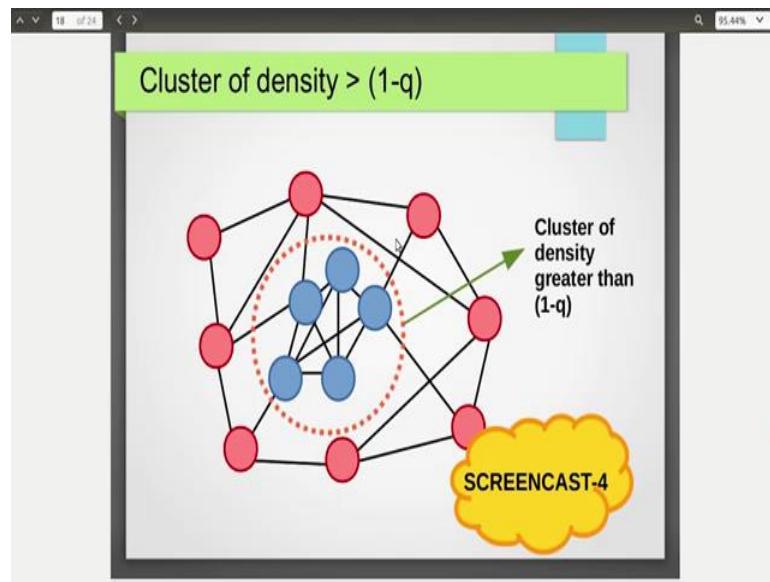
(Refer Slide Time: 06:05)



Screen cast 3: this cute idea of the effect of community on cascades. We have looked at community drapes cascades. So, here if there are 2 communities and this idea originates here; it cascades, cascades and this entire community has adopted this idea, but it is difficult for this idea to come and defuse to this community.

So, one community remains with the old idea and other has adopted this new idea and we have looked at the reason for why it happens. So, we will be coding it in screen cast 3 where we will take a network having 2 communities and we will start our cascade from one community and we will see that it infects everybody in this community, but this unable to reach out to this another community.

(Refer Slide Time: 06:57)



In the last part, we will be implementing the theorem we talked about. So, the theorem was, the statement was, if there is a cluster of density greater than  $1 - q$  a complete cascade is not possible. So, many technical words here; I suggest that you go back and quickly watch what do I mean by cluster density  $1 - q$  here. I will quickly recap it here.

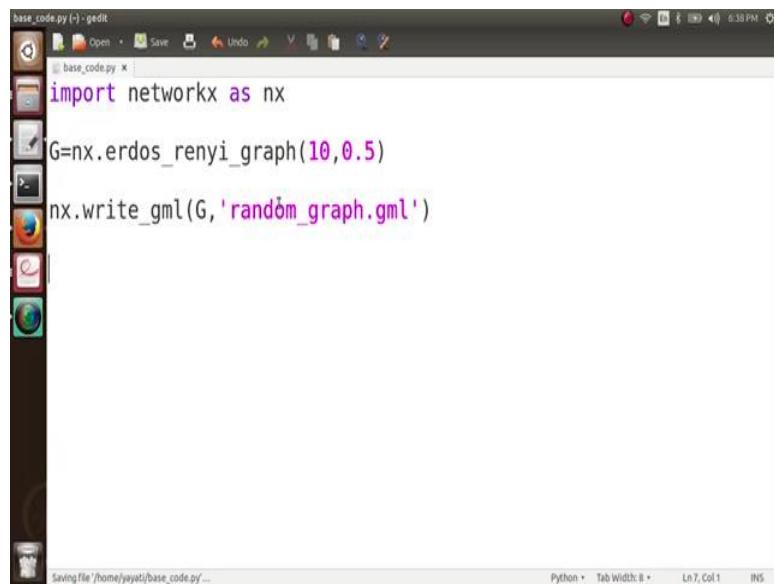
So, we say that there are clusters in every network and we say that a cluster is of density  $p$ , if for every node in the cluster, each and every node in this cluster has at least  $p$  fraction of their friends in the same cluster then, we call it a cluster of density  $p$ . And  $q$  here is a threshold associated with the new idea which means that, if  $q$  of my friends,  $q$  fraction of my friends or greater than  $q$  fraction of my friends adopt a particular idea, adopt this new idea, I will also adopt this new idea.

So, the theorem said that if there is a cluster of density greater than  $1 - q$  in this network then, your idea cannot defuse inside this cluster here. So, here is a cluster of density greater than  $1 - q$  and your cascade starts from here. Even if your cascade infects all these nodes in this network, all the nodes in this network, we will see that it is unable to defuse inside this cluster. And, we have talked about the proof of this theorem in the chapter, but here we will be coding it and validating it with the help of a code. So, this will be our screen cast 4.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Cascading Behaviour in Networks**  
**Lecture - 96**  
**The Base Code**

(Refer Slide Time: 00:05)



A screenshot of a terminal window titled "base\_code.py (~) - gedit". The window contains the following Python code:

```
import networkx as nx
G=nx.erdos_renyi_graph(10,0.5)
nx.write_gml(G,'random_graph.gml')
```

The terminal window has a dark theme. The status bar at the bottom shows "Saving file /home/yeyati/base\_code.py..." and "Python • Tab Width: 8 • Ln 7, Col 1 INS".

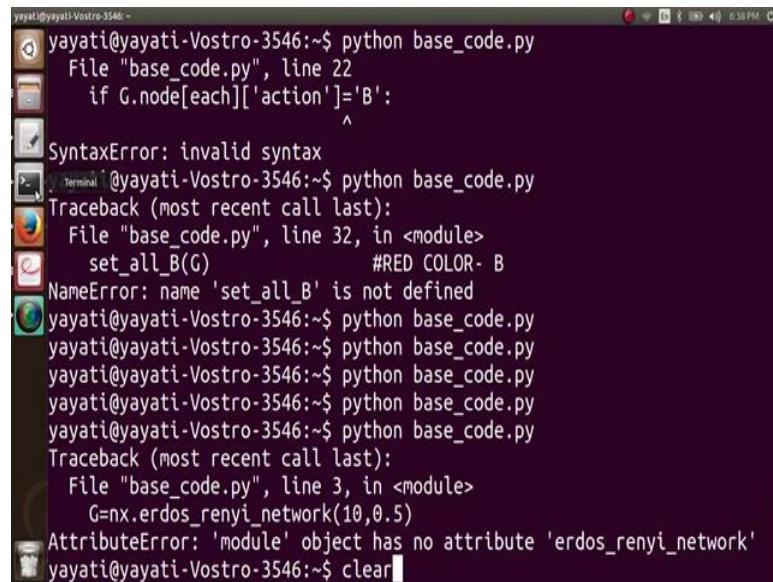
So, we have a file here. So, I am using this file base code.py for the very first code. What we are first going to do is import networkx as nx since we want to work on graphs. Next what we need is an underline network. So, for implementing anything how is action a diffusing on this network or action b diffusing on this network, how is the cascade going.

So, for everything I need an underline graph. And I am going to take this underline graph as an Erdos-Renyi graph as a random graph. So, I make here Erdos-Renyi graph  $G = nx.erdos_renyi_graph$  and then I want this graph to have 10 nodes.

So, first parameter is the number of nodes which is 10 and the second parameter is the probability with which an edge is represent, which is 0.5 And what I will also do is I want to use the same graph for some reasons I want to use the same underline graph for the coming up codes.

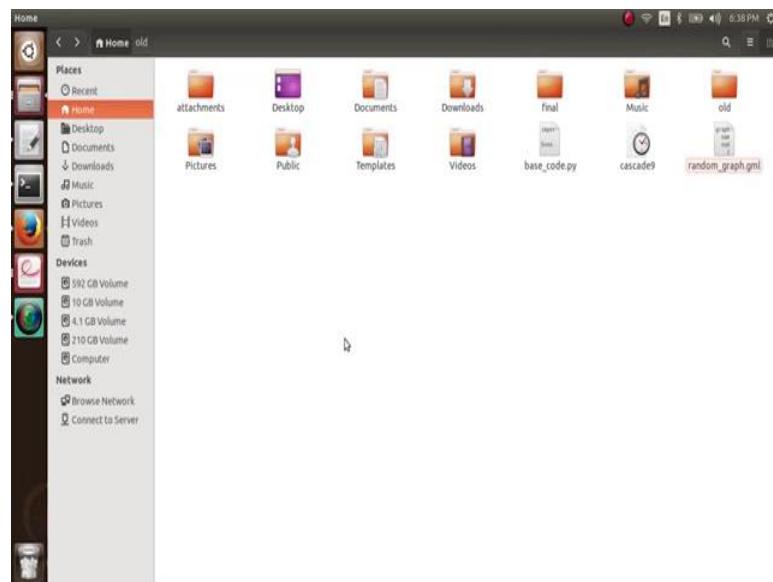
So, what I am going to do is, I will do G, I am going to store this graph as a gml file, so I use nx.write\_gml and I write this graph G as random graph.gml random\_graph.gml And we will be using this graph further let us just execute it.

(Refer Slide Time: 01:43)



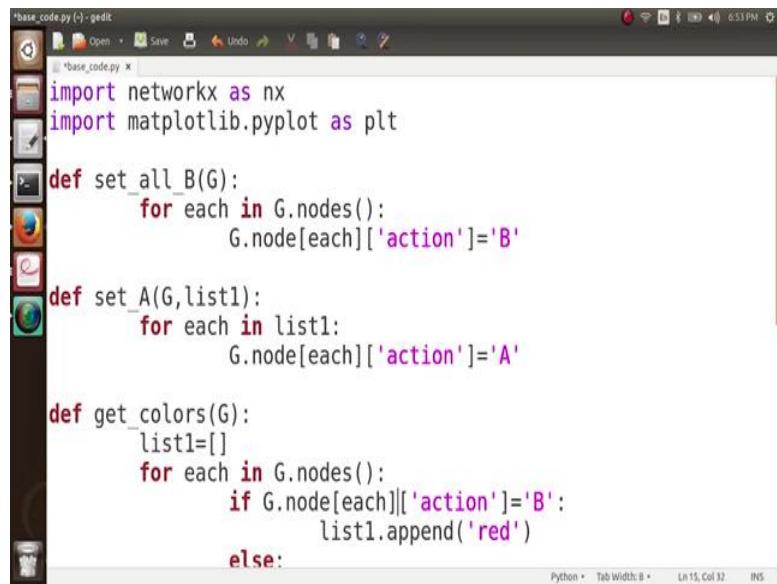
```
yayati@yayati-Vostro-3546:~$ python base_code.py
  File "base_code.py", line 22
    if G.node[each]['action']=='B':
                           ^
SyntaxError: invalid syntax
yayati@yayati-Vostro-3546:~$ python base_code.py
Traceback (most recent call last):
  File "base_code.py", line 32, in <module>
    set_all_B(G)                      #RED COLOR- B
NameError: name 'set_all_B' is not defined
yayati@yayati-Vostro-3546:~$ python base_code.py
Traceback (most recent call last):
  File "base_code.py", line 3, in <module>
    G=nx.erdos_renyi_network(10,0.5)
AttributeError: 'module' object has no attribute 'erdos_renyi_network'
yayati@yayati-Vostro-3546:~$ clear
```

(Refer Slide Time: 01:50)



So, we have executed it and you can see that here we have these graph random\_graph.gml. So, now, on we are using only this graph random\_graph.gml ok.

(Refer Slide Time: 02:01)



```
base_code.py (~)- gedit
base_code.py *
import networkx as nx
import matplotlib.pyplot as plt

def set_all_B(G):
    for each in G.nodes():
        G.node[each]['action']='B'

def set_A(G,list1):
    for each in list1:
        G.node[each]['action']='A'

def get_colors(G):
    list1=[]
    for each in G.nodes():
        if G.node[each]['action']=='B':
            list1.append('red')
        else:
```

So, I have remove the other statements from this code we have already run it once. So, we have that graph, random\_graph stored in our system. So, first I will load that graph. So, what I do is `G = nx.read_gml` and then I have the graph here, `random_graph.gml`, I have this graph here. Next what we want to do is, we want to implement the concept that initially every node in this network is having an action B associated with it and then the action A is introduced. How do you introduce action A is, by some nodes in this network flipping their status from B to A.

So, implementing that is quite easy. We can implement it with the help of attributes associated with the nodes. So, initially what we will do is every node will have an attribute action and the value of this attribute action for every node will be equal to B So, we call a function for that is `set_all_B (G)`; which mean that in your graph G for all the nodes set its action to be equals to B and then we can quickly define this function here.

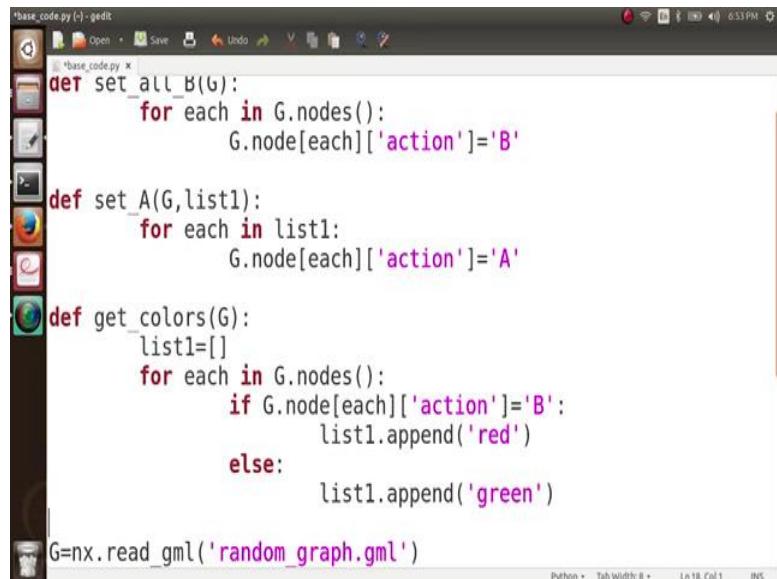
Define `set_all_B (G)` and how do we define it now we get an iterator for `each in G.nodes` for `each in G.nodes` what we have to do is set the attribute action associated with this node to be equal to B. So, what do we do is `G.node each` and then what is the action associated with this node it is equal to B.

So, all the nodes in this network currently has this action B. Next what I want to do is I will pick some 2 nodes from this network and for those 2 nodes I want to set the action associated with them to be A. So, how do I do that? One way to do that is I will take a

list here and this list here consists of some 2 nodes. So, let us say 3 and 7 So, I want these 2 nodes 3 and 7 in this network to be the nodes which initially adopt action A. So, I want to change the attributes corresponding to these nodes, so what do I call is set\_A (G, list1).

So, whatever is there in list1, what all nodes are there in list1? I want to set their attribute to be equals to A. And that is again quiet simple define set\_A and then I have a G here and I have a list1 here and what do I do is for each in G.node each action and what is this action is going to be is A.

(Refer Slide Time: 05:23)



The screenshot shows a code editor window titled "base\_code.py". The code defines three functions: set\_all\_B, set\_A, and get\_colors. The set\_all\_B function sets the 'action' attribute of all nodes to 'B'. The set\_A function sets the 'action' attribute of nodes in list1 to 'A'. The get\_colors function creates a list of colors ('red' or 'green') based on the 'action' attribute of each node. At the bottom, there is a line of code: G=nx.read\_gml('random\_graph.gml').

```
def set_all_B(G):
    for each in G.nodes():
        G.node[each]['action']='B'

def set_A(G,list1):
    for each in list1:
        G.node[each]['action']='A'

def get_colors(G):
    list1=[]
    for each in G.nodes():
        if G.node[each]['action']=='B':
            list1.append('red')
        else:
            list1.append('green')

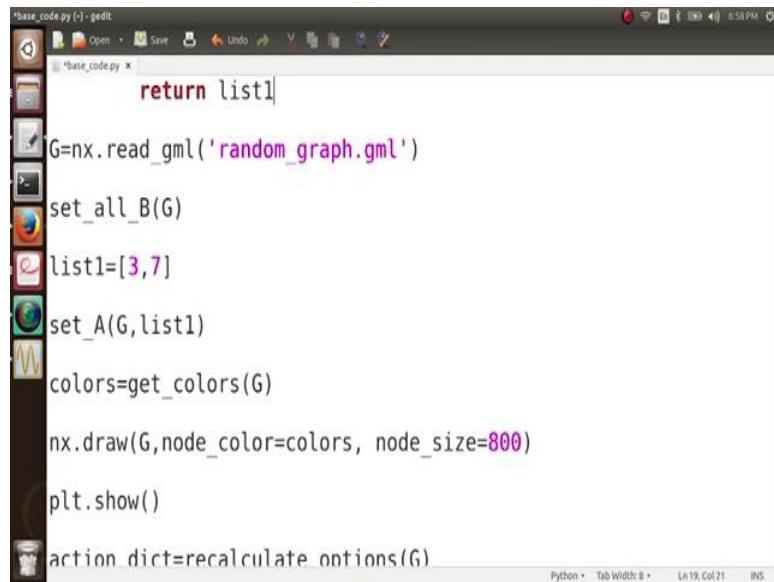
G=nx.read_gml('random_graph.gml')
```

So, I have a graph here in which all the nodes have adopted the action B except these 2 nodes which are the initial adopters 3 and 7 which are the initial adopters and I have adopted this action A and next I want to visualise this graph. So, visualising this graph is easy and let us visualise it with colours.

So, we have done it previously as well. So, let us associate it a colour red with the action B and the colour green with the action A. So, initially all the nodes in this network will be red and this green behaviour will be coming and trying to spread on this network. So, for that I will need an array colour. So, let me get this array from the function get colours G and then this is simple.

So, I have a function here define colours G and what this function is going to do is I have a list here and for each in G.nodes what we are going to do is if G.node each and if the action associated with this particular node is B, which means that it should have a it should have a red colour. So, list1.append ('red') else list1.append ('green').

(Refer Slide Time: 07:15)



```
base_code.py (~) - gedit
return list1

G=nx.read_gml('random_graph.gml')

set_all_B(G)

list1=[3,7]

set_A(G,list1)

colors=get_colors(G)

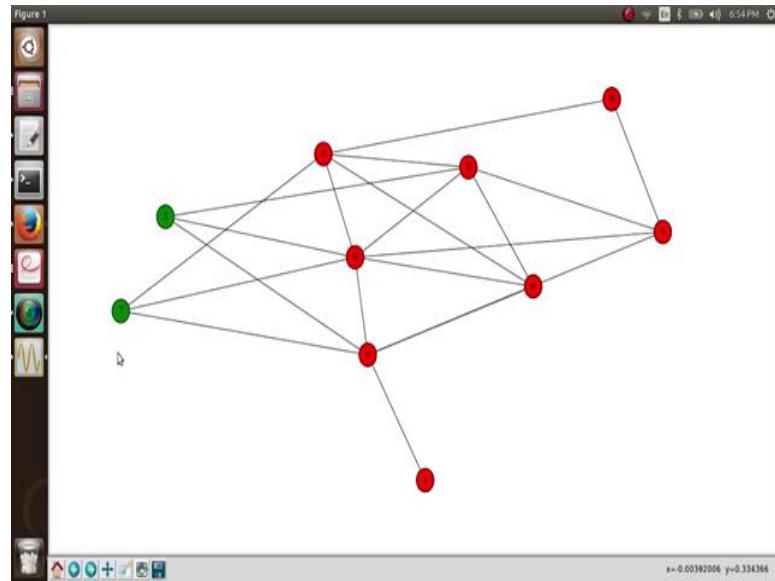
nx.draw(G,node_color=colors, node_size=800)

plt.show()

action_dict=recalculate_options(G)
```

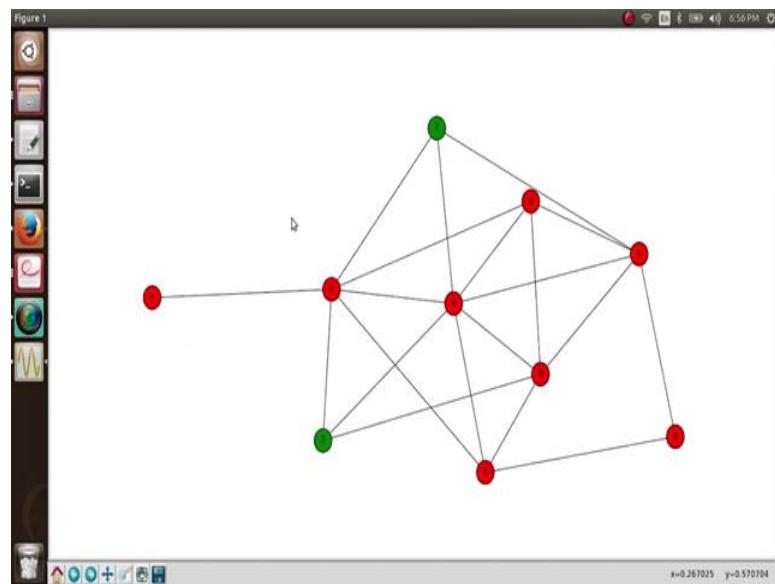
And then return list1, so, we have an array of colours here. Let us quickly visualise this graph. So, for visualise this visualising this graph we have. So, first of all we need to import the matplotlib; import matplotlib.pyplot as plt then come down then what we do is nx.draw (G) and what should be the node colour; node colour should be taken from the array colours and what should be the node size? Let us see the size of every node to be 800, so that it is be and then we do plt.show. Let us now turn this piece of code and see. A small mistake in line 15 which should be equals to equals to go back run it ok.

(Refer Slide Time: 08:18)



So, you can see here there is a network and in this network initially all the nodes they were green they had adopted this behaviour B and then there are these two nodes three and seven which have been initially decided to adopt the behaviour A.

(Refer Slide Time: 08:39)



So, we have the graph here. Next what we want to do is if we look at this graph here you see what is going to, you see what is going to happen next this nodes 7 and 3 each node here from 0 to 9 we will look at their neighbours, we will look at their neighbours and

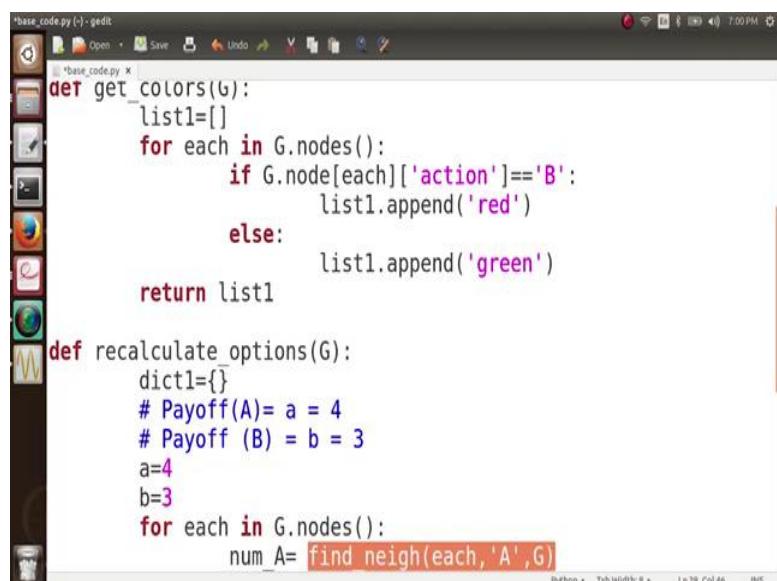
look at the action their neighbours are taking the payoffs and according to that re way their option; for example, this node 2 here looks at all of its neighbours.

So, first it looks at it, look at its neighbours who have adopted action B. So, it looks at 1, 8, 0 and 5, so 4 nodes and then let us say the payoff associated with the action B is let us say 2, so 4 into 2 8. 8 is the payoff it gets from here and then it looks at these 2 nodes 7 and 3 and then let us say the payoff associated with this action A is 3. So, gets a payoff of 6 from here. So, it decides to remain in this state B only.

And similarly, every node is going to re way this option and according to that the cascade is going to occur is this network. So, let us try to code that. So, what is now going to happen is every node is going to re way its options based on its neighbours, based on their actions and based on the payoffs of those actions as we have seen before. So, we are going to define a function now recalculate options. So, this function recalculate options will one by one look at every node in the network, look at re way its options and it will put the next it will put the decision of this node in a dictionary.

So, we get a dictionary here action dictionary. So, what is this action dictionary? It is a dictionary where keys are the nodes and the values are the actions associated with these nodes. And these dictionaries we format from the function recalculate options in G.

(Refer Slide Time: 10:55)



```
base_code.py -e gedit
def get_colors(G):
    list1=[]
    for each in G.nodes():
        if G.node[each]['action']=='B':
            list1.append('red')
        else:
            list1.append('green')
    return list1

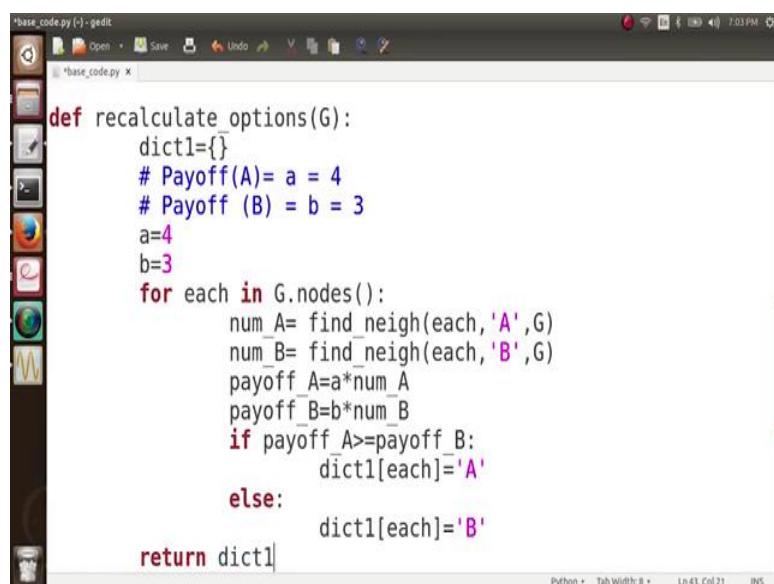
def recalculate_options(G):
    dict1={}
    # Payoff(A)= a = 4
    # Payoff (B) = b = 3
    a=4
    b=3
    for each in G.nodes():
        num_A= find_neigh(each,'A',G)
```

Let us see how it works. So, we define the function here recalculate options in G and we have a dictionary here let us name it dict1, this is the dictionary which will hold the decision for every node. Now, before moving in further we need to decide the playoffs associated with the action.

So, let us say Payoff associated with the action A. So, let us call it small a and let us take its value to be 4 and then we have a Payoff associated with B let us say b and let us say value to be equal to 3. So, based on these payoffs so,  $a = 4$  and  $b = 3$ , these are the payoffs associated with the 2 actions. Now for each in G.nodes, but this node is going to do is. First of all, this node needs to know how many of its neighbours have adopted the action A, how many of neighbours its neighbours have adopted the action B.

So, let us say number of neighbours who have adopted the action A equals to and we call a function here find neighbours and we pass 2 parameters to this function c and G. So, c is basically c we pass it A and G. So, A is the action which we want to see how many of its neighbours have adopted this action A and G and we want to pass this node also, so each A and G. So, let us now define this function.

(Refer Slide Time: 12:58)



```

base_code.py (~) - gedit
File Open Save Undo V Help
base_code.py *
def recalculate_options(G):
    dict1={}
    # Payoff(A)= a = 4
    # Payoff (B) = b = 3
    a=4
    b=3
    for each in G.nodes():
        num_A= find_neigh(each,'A',G)
        num_B= find_neigh(each,'B',G)
        payoff_A=a*num_A
        payoff_B=b*num_B
        if payoff_A>=payoff_B:
            dict1[each]='A'
        else:
            dict1[each]='B'
    return dict1
Python Tab Width: 8 Ln 43, Col 21 INS

```

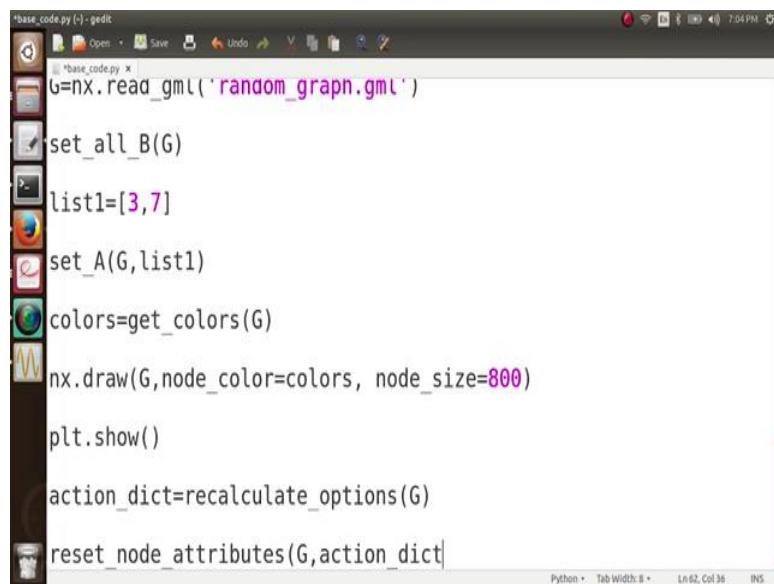
Define and instead of A, we have a character here. So, what this function is going to return in the graph G, the node each we are going to look at the neighbours of the node each and we are going to see how many of these neighbours have adopted this action c.

So, what do we do it is simple we take num to be equal to 0 and then for each I am sorry for each one in G.neighbours G.neighbours each if G.node each one and the action associated with this node is c. Then what we will do is num = num + 1 and then we return this number simple right. So, we have this num\_A and similarly we can find how many of its neighbours have adopted B. So, find\_neigh each and then we have this B and then we have this G right.

Next what is the total payoff this node gets from adopting the behaviour A. It is nothing, but the payoff associated with this behaviour which is A multiplied by the number of its neighbours which have adopted this behaviour A and similarly what is the payoff it gets from adopting the behaviour B is nothing, but b multiplied by the number of neighbours which have adopted B.

Now this node decides can easily decide whether it wants to adopt the behaviour A, or it wants to adopt the behaviour B. So, we know that if payoff associated with A is greater than or equal to pay off with B, what is going to happen is dict1 for this node which is each is going to be A, it is going to adopt A else dict1 with each is going to be B and then we return dict1.

(Refer Slide Time: 15:29)



```

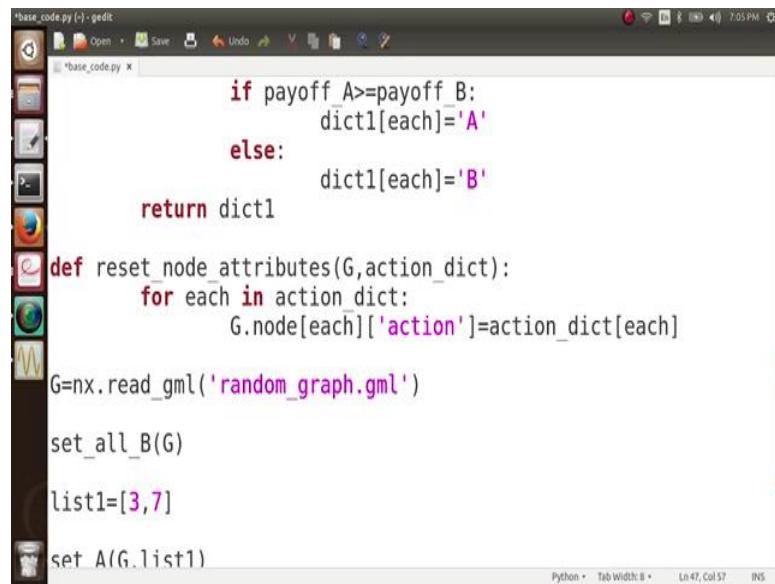
base_code.py (-) - gedit
base_code.py x
G=nx.read_gml('random_graph.gml')
set_all_B(G)
list1=[3,7]
set_A(G,list1)
colors=get_colors(G)
nx.draw(G,node_color=colors, node_size=800)
plt.show()
action_dict=recalculate_options(G)
reset_node_attributes(G,action_dict)

```

So, we get here this action dictionary which tells me what the decision for the next snapshot is that the node makes. So, you see here this action dictionary it is nothing, but it tells me a next snapshot of the graph. It tells me in the next a snapshot in the next

iteration in this process what action every node would have adopted. And now we simply need to draw this graph. So, we have this snapshot stored in action dictionary. So, we want the for the next snapshot we change the attribute associated with the associated with each node based on this action dictionary. So, we do reset node attributes in G and which is based on your action dictionary right action dictionary.

(Refer Slide Time: 16:25)



```

base_code.py (~) - gedit
(base_code.py) 7:05 PM
if payoff_A >= payoff_B:
    dict1[each] = 'A'
else:
    dict1[each] = 'B'
return dict1

def reset_node_attributes(G, action_dict):
    for each in action_dict:
        G.node[each]['action'] = action_dict[each]

G = nx.read_gml('random_graph.gml')
set_all_B(G)
list1 = [3, 7]
set_A(G, list1)

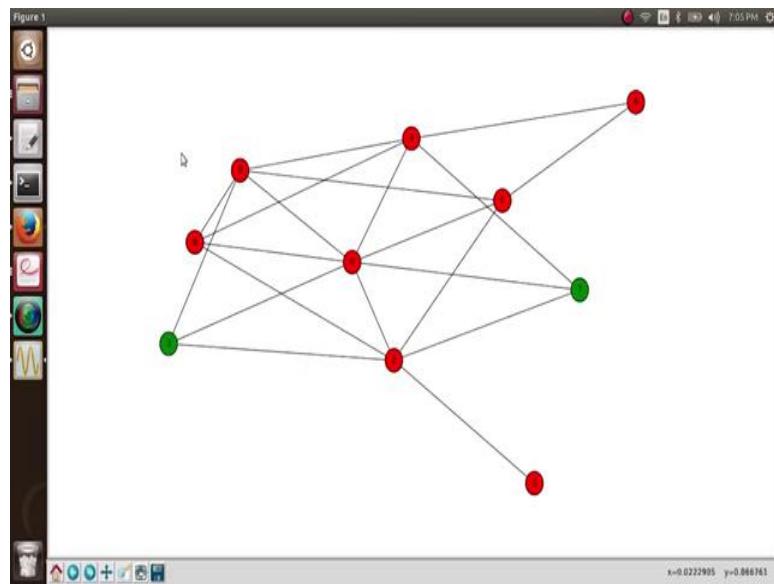
```

The screenshot shows a terminal window titled "base\_code.py (~) - gedit". The code defines a function "reset\_node\_attributes" that takes a graph "G" and an "action\_dict" as parameters. It iterates through the keys in the action dictionary, setting the "action" attribute of each node in the graph to the corresponding value from the action dictionary. The code also includes a "set\_all\_B" function and a "set\_A" function. Finally, it reads a graph from a file named "random\_graph.gml" and applies the "set\_all\_B" function to all nodes. Then, it calls the "set\_A" function with a list containing the values 3 and 7.

We define it here define reset node attributes and what are the parameters G and action dictionary let us say G and action dictionary would we are going to do here if it is simple. So, for each in action dictionary for each for every key in action dictionary G.node each action, what is it going to be the value associated with that which is action dictionary each is the action.

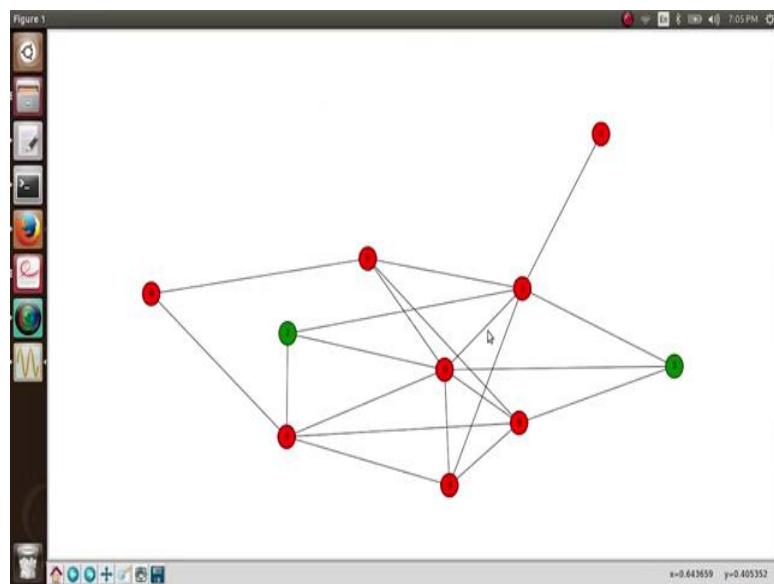
So, we have reset the node attributes here. So, our graph is ready, and we just have to visualise this graph and what do we need for visualise, we need to set the right colours of nodes and we need to draw this graph. Just simply copy and paste this node here; let us now execute it and see.

(Refer Slide Time: 17:31)



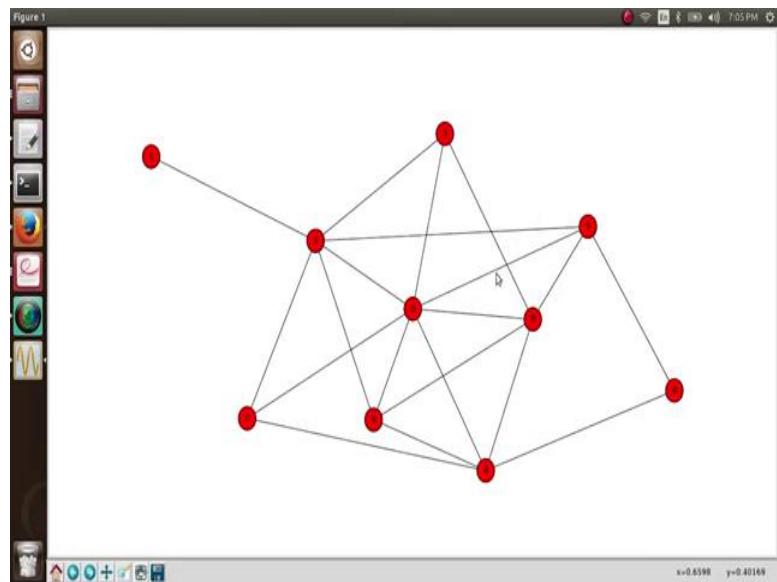
So, you can see here this is the initial graph where our nodes 7 and 3 they have adopted the behaviour A and just small mistake in the spelling of neighbours ok.

(Refer Slide Time: 18:02)



So, this is the initial graph where these nodes 7 and 3 they were they have adopted the behaviour A and you see what happened in next iteration.

(Refer Slide Time: 18:08)



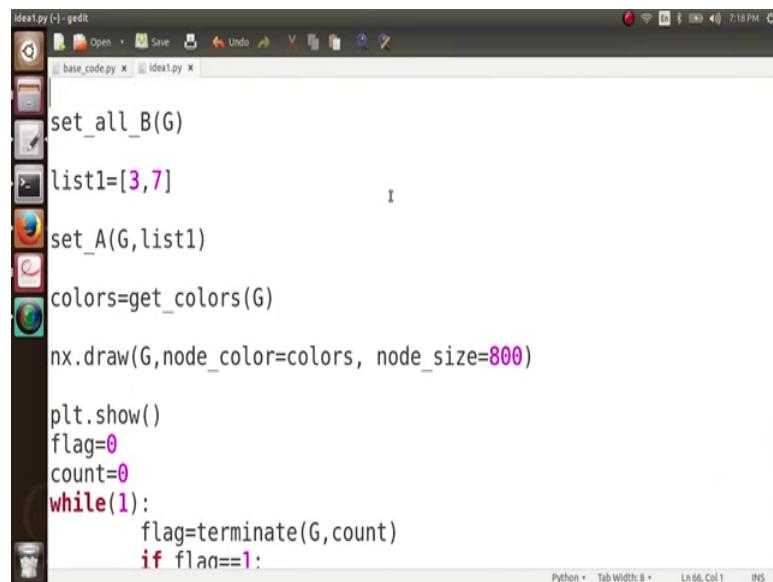
Everybody reviewed their option and these node 2 nodes initially 3 and 7 which had adopted which had decided to adopt this behaviour A, they also backed off and they went back to this behaviour B. We have seen that although the payoff associated with A was high it was 4 and with B was 3, but everybody has switched back to the same condition of having adopted the behaviour B right.

So, it validates that it is actually difficult for any new behaviour or action to cascade on a network. Now, in the next screen cast will see that how as we keep increasing the payoff associated with this action A, a graph ultimately the cascading in a graph is the cascading of action A in the graph is more and rather it can create a complete cascade also.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Cascading Behaviour in Networks**  
**Lecture - 97**  
**Coding the First Big Idea - Increasing the Payoff**

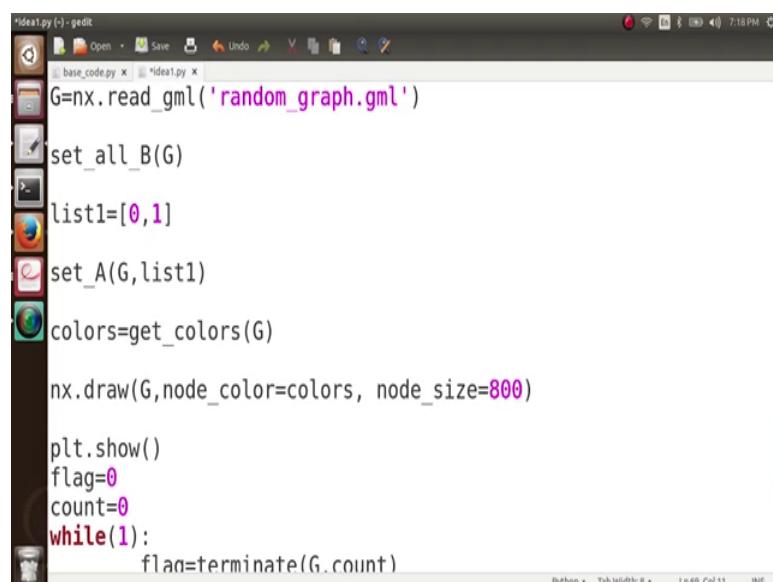
(Refer Slide Time: 00:05)



```
ideat.py (~) - gedit
base_code.py  ideat.py
set_all_B(G)
list1=[3,7]
set_A(G,list1)
colors=get_colors(G)
nx.draw(G,node_color=colors, node_size=800)
plt.show()
flag=0
count=0
while(1):
    flag=terminate(G,count)
    if flag==1:
```

So, now let us experiment with this code a little bit. Everything is ready now.

(Refer Slide Time: 00:11)



```
ideat.py (~) - gedit
base_code.py  *ideat.py
G=nx.read_gml('random_graph.gml')
set_all_B(G)
list1=[0,1]
set_A(G,list1)
colors=get_colors(G)
nx.draw(G,node_color=colors, node_size=800)
plt.show()
flag=0
count=0
while(1):
    flag=terminate(G,count)
```

So, we have a system where we can define, who are the initial adopters of the action a and then, we can run an run the code and see how this action a is cascading in the network whether it dies away, or whether it persist in the network or what happens. So, for the time being what I am going to do is, I am going to set the initial adopters to be 0 and 1 in the network and I am actually, also going to change our payoffs associated with a and b.

(Refer Slide Time: 00:34)

```
num=num+1
return num

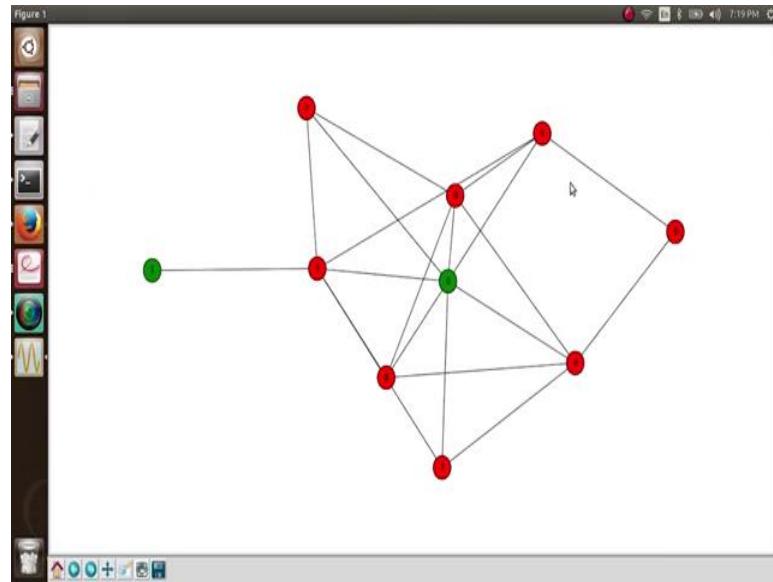
def recalculate_options(G):
    dict1={}
    # Payoff(A)= a = 4
    # Payoff (B) = b = 3
    a=7
    b=5
    for each in G.nodes():
        num_A= find_neigh(each,'A',G)
        num_B= find_neigh(each,'B',G)
        payoff_A=a*num_A
        payoff_B=b*num_B
        if payoff_A>=payoff_B:
            dict1[each]='A'
        else:
```

So, let me keep the payoffs associated with  $a = 6$  and with  $b = 5$ .

(Refer Slide Time: 00:46)

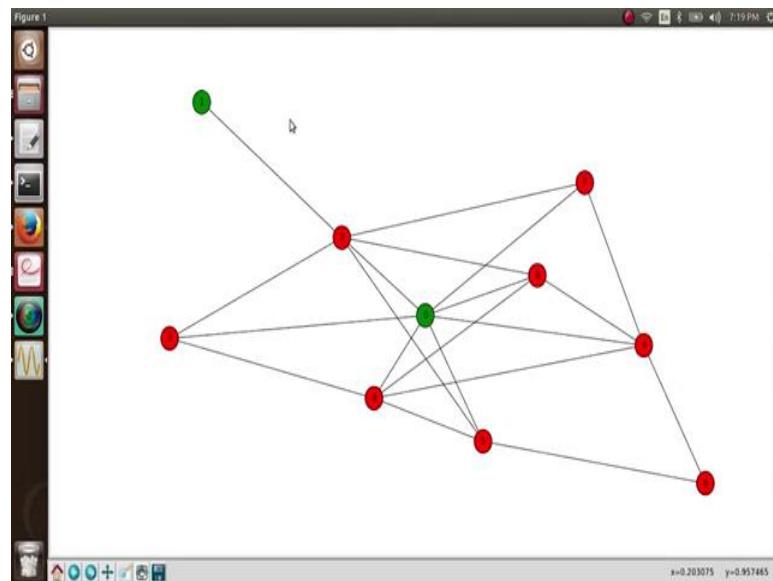
And let us execute this code and see what is going to happen.

(Refer Slide Time: 00:47)



So, initially 0 and 1 in this network I have adopted the idea and then, you see it dies away. So, seems like the payoff 6 is not enough for this action a to cascade on this network right. So, what I am going to do is, I am going to increase the payoff and I am going to keep this payoff of to be 7. Let us see what happens now.

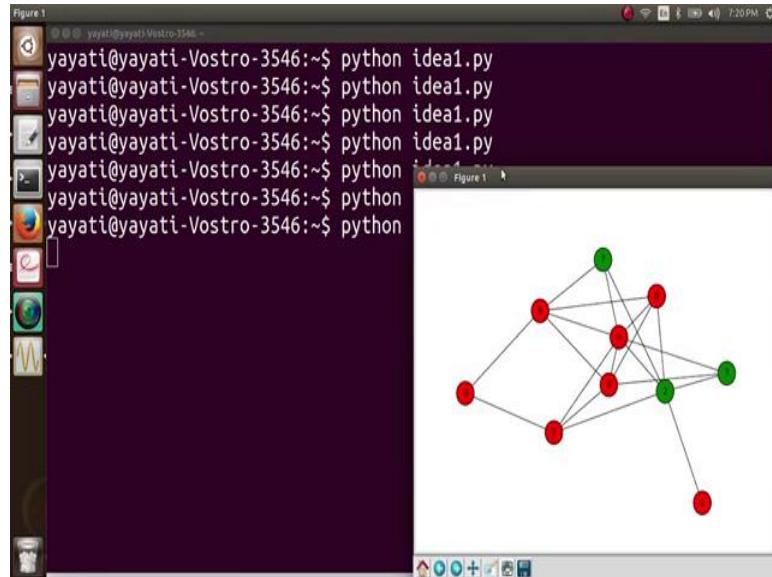
(Refer Slide Time: 01:10)



So, when I keep this payoff of to be 7 initially, 1 and 0 and then again, it dies away. So, the payoff 7 is also not working. Let us make it 8. I am make it 8 and then we have 0 and

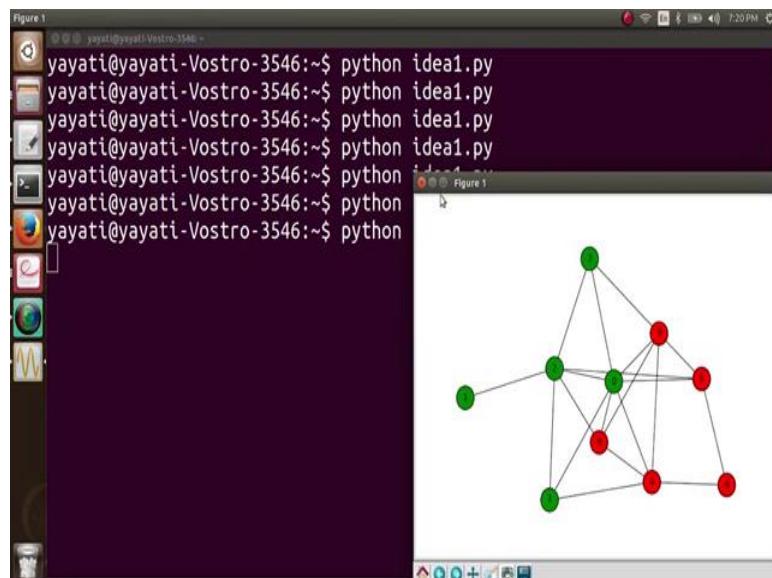
1, in 8 also it dies away ok. Let us make it 9 dies away. And let us make it 10 and then you see when I make it 10.

(Refer Slide Time: 01:43)



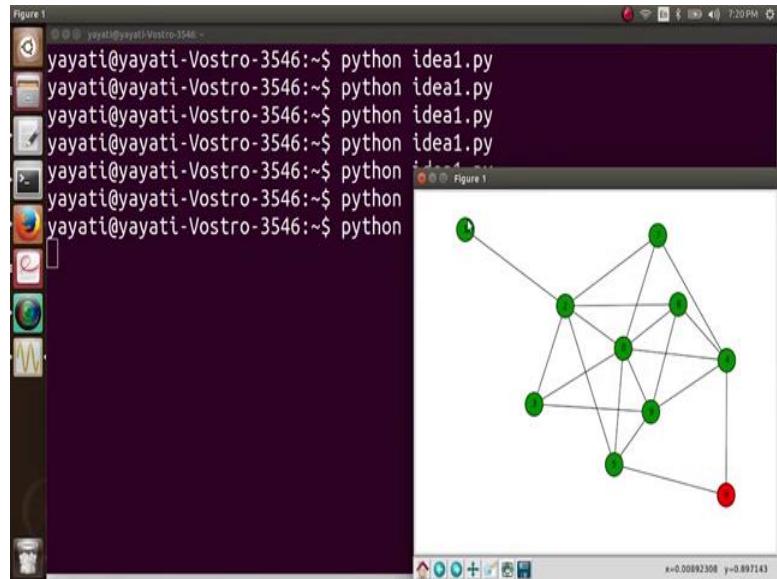
Now, although 0 and 1 they have adopted the behaviour b now, but these three nodes 2, 3 and 7, they have adopted the behaviour a. So, the behaviour is persisting.

(Refer Slide Time: 02:01)



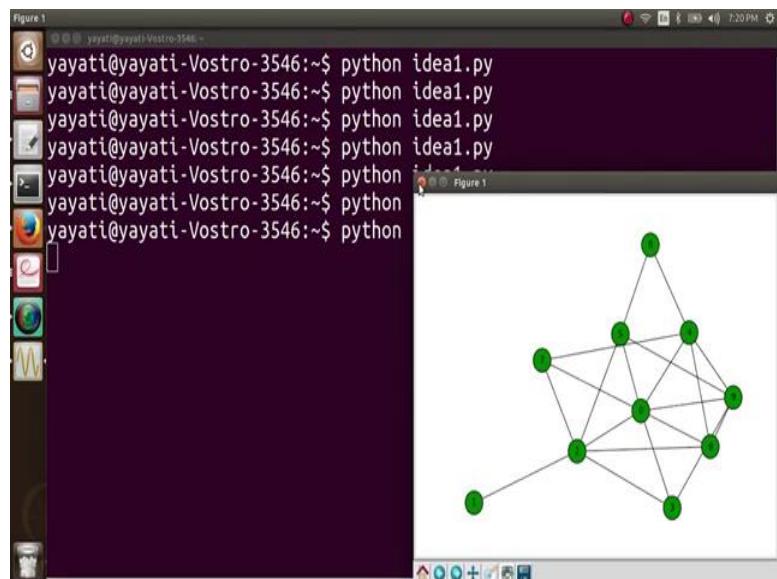
Let us see in the next iteration. In the next iteration now, 0 and 1 also get back the behaviour a. So, now, you can see this behaviour a cascading on this network.

(Refer Slide Time: 02:10)



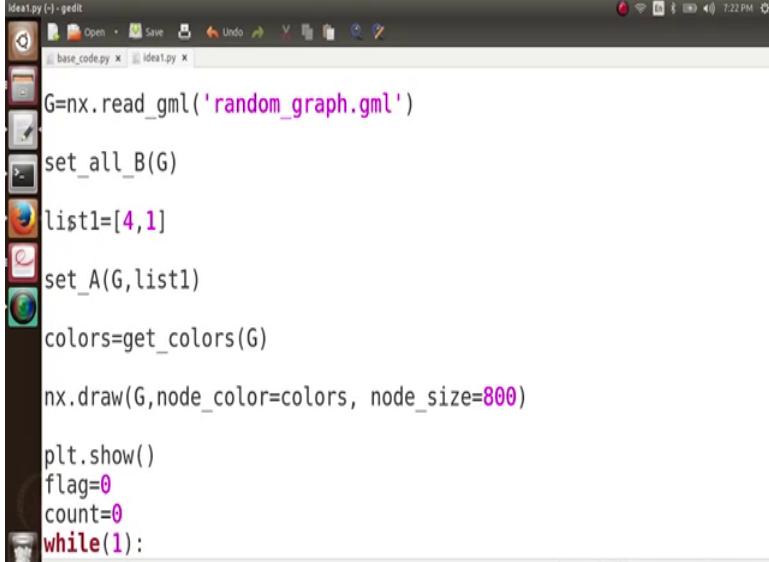
And then you see some more nodes adopt the behaviour a.

(Refer Slide Time: 02:14)



And then finally, all the nodes adopt this behaviour a and the process ends. So, have you seen just now what happened? The, we have started with a payoff 6. So, when the payoff was 6 nothing happened 7, 8, 9, the behaviour ultimately died away, but when you made its payoff to be equal to 10, everybody in this network adopted the behaviour a. Let us try it something else also. So, let us make it 6 and 5.

(Refer Slide Time: 02:42)



```
idea1.py - gedit
base_code.py x idea1.py x
G=nx.read_gml('random_graph.gml')

set_all_B(G)

list1=[4,1]

set_A(G,list1)

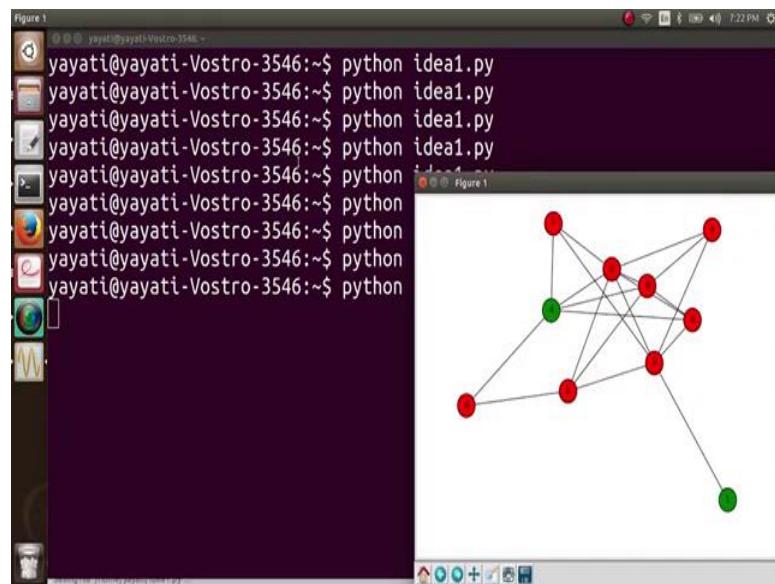
colors=get_colors(G)

nx.draw(G,node_color=colors, node_size=800)

plt.show()
flag=0
count=0
while(1):
```

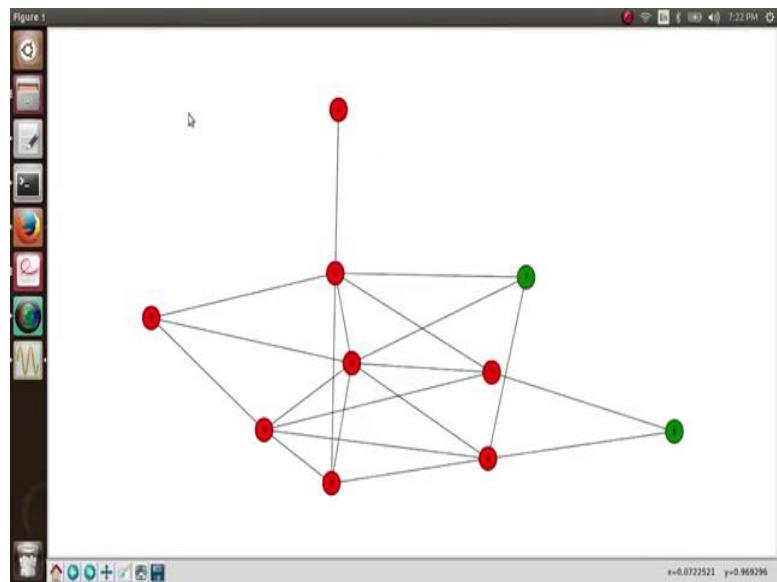
And let us change my initial adopters to be let us say 4 and 1 and let us see what happens now. 4 and 1 and then, 6 and then, dies away ok. Let us increase the payoff to let us say 8, 4 and 1, 6, dies away and say 9, 4 and 1 and 6 and then, dies away and then 10 ok.

(Refer Slide Time: 03:06)



Do you see here what is happening now in this case?

(Refer Slide Time: 03:36)



So, this was what I wanted to show you. In some situations what can happen, you see 7, 6 and then, 7 and 6 in fact 4. And then, 4 in fact 7 and 6 and then, 4 and then, this and this, and this, code goes into an infinite loop. And it is going to run 100 times. It is going to run 100 times ok.

(Refer Slide Time: 04:09)

A screenshot of a Python IDE window titled "ideal.py - edit". The code in the editor is as follows:

```
plt.show()
flag=0
count=0
while(1):
    flag=terminate(G,count)
    if flag==1:
        break
    count=count+1
    action_dict=recalculate_options(G)

    reset_node_attributes(G,action_dict)

    colors=get_colors(G)

    c=is_complete_cascade(G)

    nx.draw(G,node_color=colors, node_size=800)
```

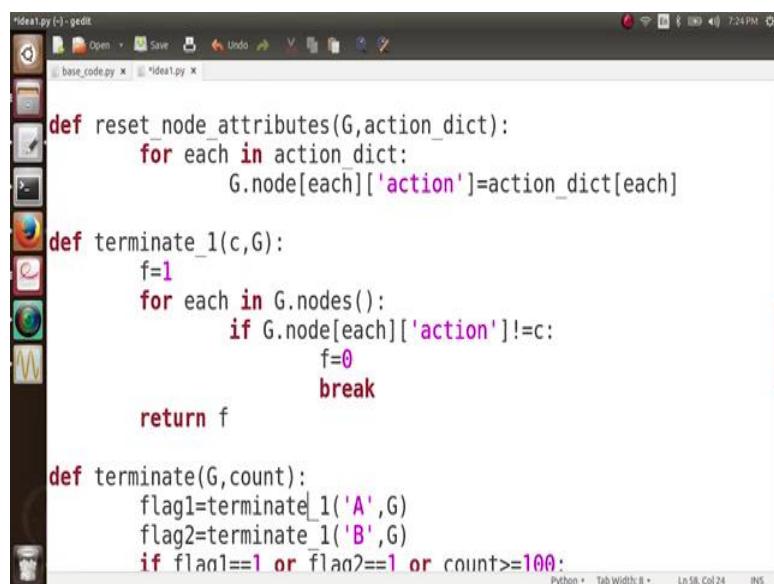
The code uses the NetworkX library (nx) to draw the graph with node sizes of 800. The script includes functions for terminate, calculate options, reset node attributes, and get colors. It also imports plt from matplotlib.

So, what I want to do is instead, of seeing this code to run for 100 times, I actually want to know just a simple thing. I want to know whether my cascade is complete or not.

What do I mean by my cascade is complete is, everything at the end has adopted this behaviour which is behaviour a shown in green colour or not?

So, that is my only aim. So, I do not need, I do not want to see these intermediary graphs. So, I just want to see my graph once at the starting, once at the end and in addition I want to know whether my cascade is complete or not. So, we can quickly see whether my cascade is complete or not. So, we call a function c here sorry, variable c here which holds the value is my cascade complete or not; is complete cascade. So, whether this cascade which has occurred on my graph it is complete or not? And how do we know it is complete?

(Refer Slide Time: 05:21)



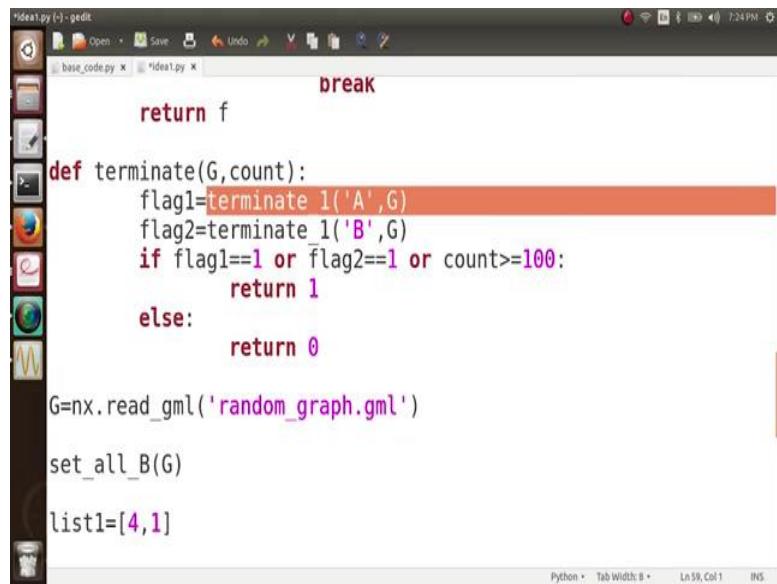
```
ideat1.py (~) - gedit
base_code.py x  ideat1.py x
def reset_node_attributes(G,action_dict):
    for each in action_dict:
        G.node[each]['action']=action_dict[each]

def terminate_1(c,G):
    f=1
    for each in G.nodes():
        if G.node[each]['action']!=c:
            f=0
            break
    return f

def terminate(G,count):
    flag1=terminate_1('A',G)
    flag2=terminate_1('B',G)
    if flag1==1 or flag2==1 or count>=100:
        Python + Tab Width: 8 + Ln 58, Col 24 INS
```

So, you see, we have used this function here, terminate 1('A', G). So, this function it used to return 1. When everything in this graph has adopted the behaviour a, so, instead of using a new function, we can use the same function.

(Refer Slide Time: 05:35)



```
ideat.py (~) - gedit
base_code.py * | ideat.py *
break
return f

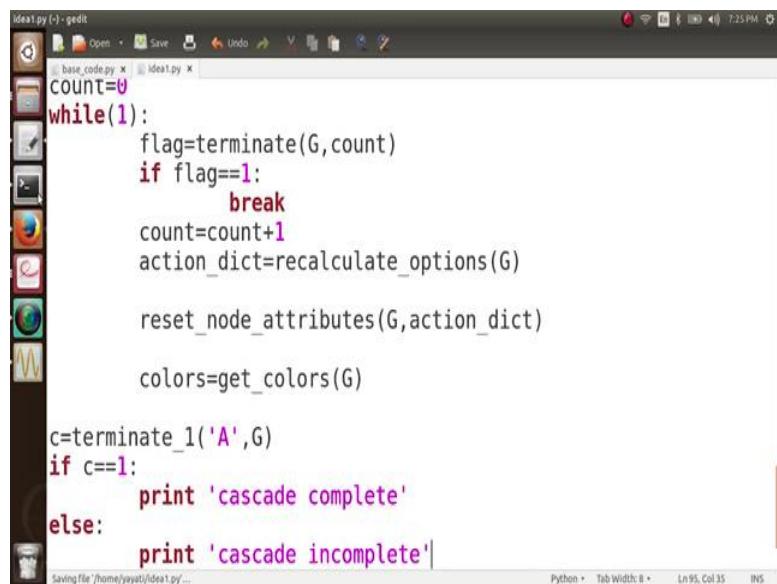
def terminate(G,count):
    flag1=terminate_1('A',G)
    flag2=terminate_1('B',G)
    if flag1==1 or flag2==1 or count>=100:
        return 1
    else:
        return 0

G=nx.read_gml('random_graph.gml')
set_all_B(G)
list1=[4,1]

Python Tab Width: 8 Ln 59, Col 1 INS
```

So, we can use this function terminate 1('A', G), which tells me whether all the nodes in my graph has adopted the behaviour a or not right.

(Refer Slide Time: 05:40)



```
ideat.py (~) - gedit
base_code.py * | ideat.py *
COUNT=0
while(1):
    flag=terminate(G,COUNT)
    if flag==1:
        break
    COUNT=COUNT+1
    action_dict=recalculate_options(G)

    reset_node_attributes(G,action_dict)

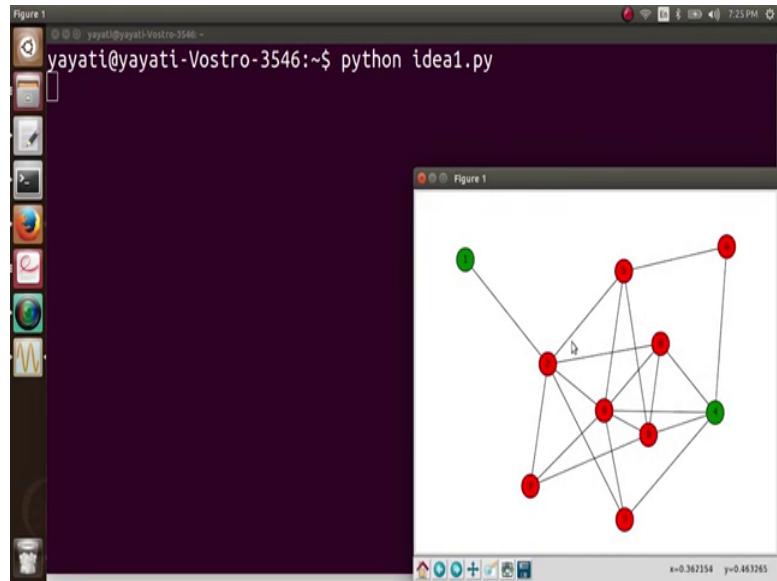
    colors=get_colors(G)

    c=terminate_1('A',G)
    if c==1:
        print 'cascade complete'
    else:
        print 'cascade incomplete'

Saving file /home/yayati/ideat.py...
Python Tab Width: 8 Ln 95, Col 35 INS
```

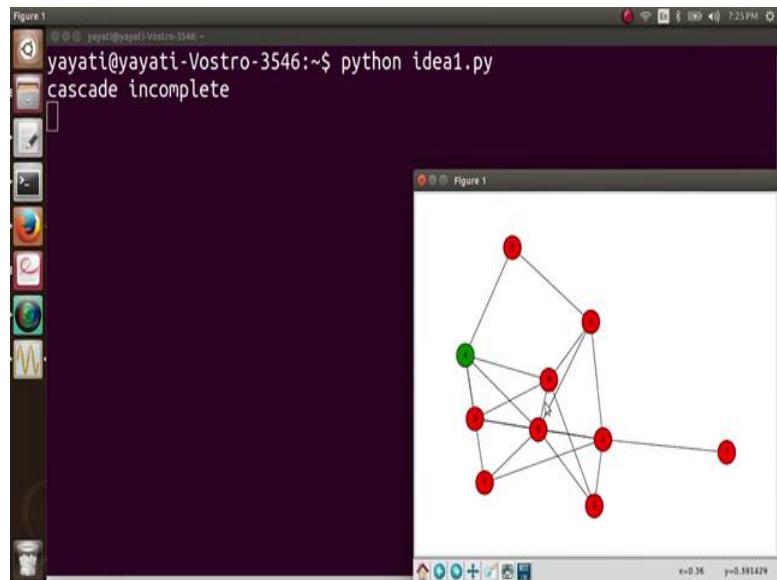
What I will do is,  $c = \text{terminate 1}('A', G)$  and what I want to see is, I want to print. So, if your  $c == 1$  then, you print cascade is complete else you print cascade is incomplete. Let us run it and see now; ideat.py.

(Refer Slide Time: 06:24)



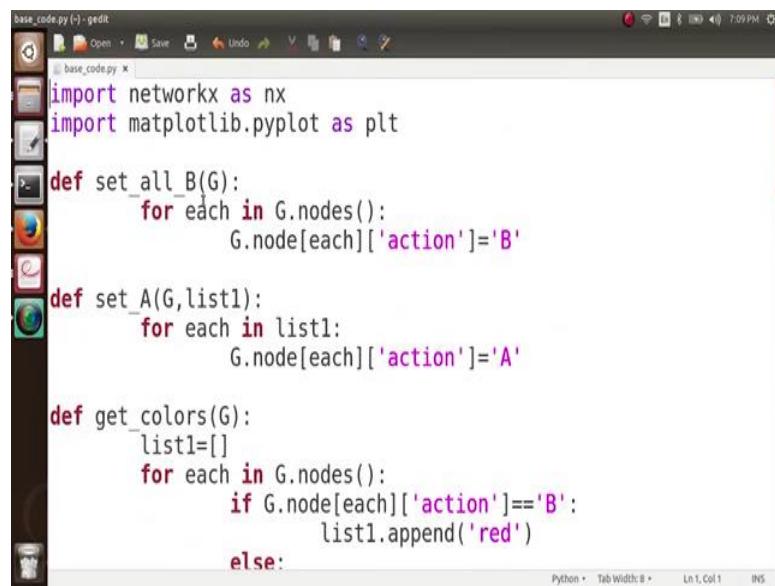
And then, this is the initial graph where 4 and 1 are infecting.

(Refer Slide Time: 06:29)



And this keeps running. So, it shows a result at the end after 100 iterations and it shows you that the cascade is incomplete. So, this is the only result we are interested in and we need not see all the intermediate cascades. We have just introduce this here because, we will be using it in the coming up programming screen casts.

(Refer Slide Time: 06:50)



```
base_code.py (~)-gedit
import networkx as nx
import matplotlib.pyplot as plt

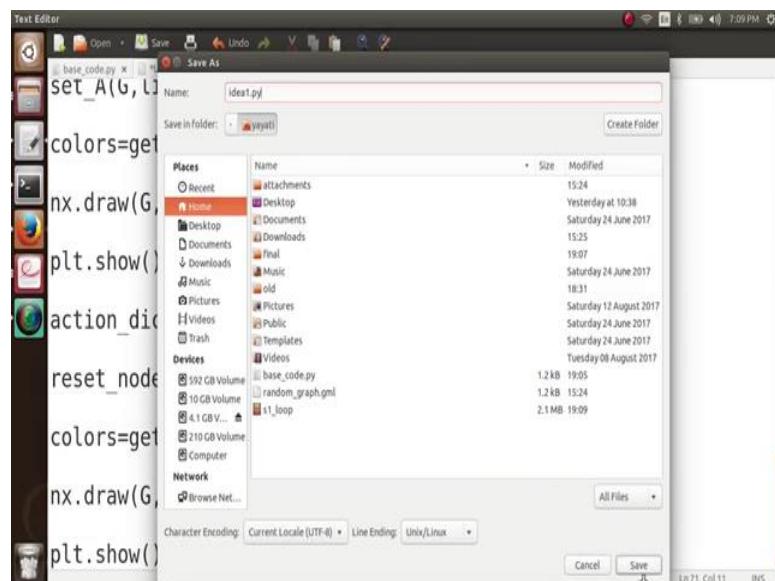
def set_all_B(G):
    for each in G.nodes():
        G.node[each]['action']='B'

def set_A(G,list1):
    for each in list1:
        G.node[each]['action']='A'

def get_colors(G):
    list1=[]
    for each in G.nodes():
        if G.node[each]['action']=='B':
            list1.append('red')
        else:
```

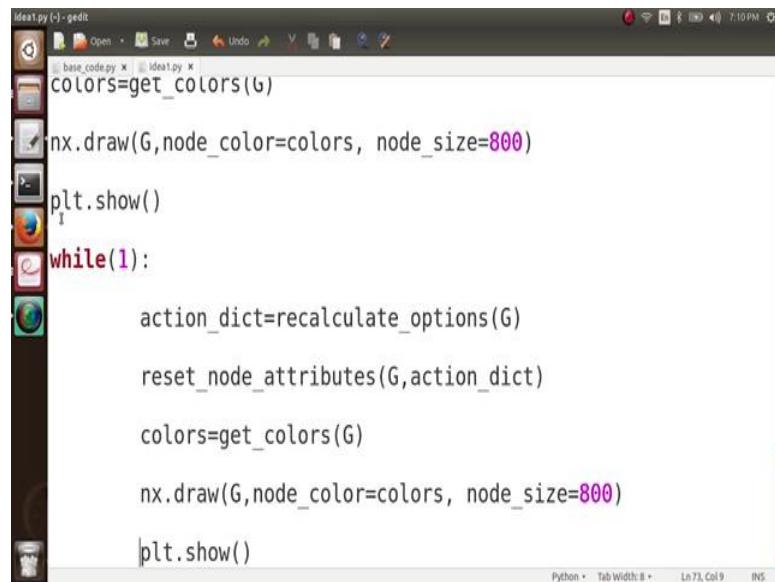
Let us get started with the first idea where we must increase the payoff and see what happens. Well basically, take this code and I will just paste it in another file.

(Refer Slide Time: 06:57)



And let me name this atom, let me name it as idea1.or idea1.py ok.

(Refer Slide Time: 07:15)



```
ideat.py (-) - gedit
base_code.py x  ideat.py x
colors=get_colors(G)

nx.draw(G,node_color=colors, node_size=800)
plt.show()

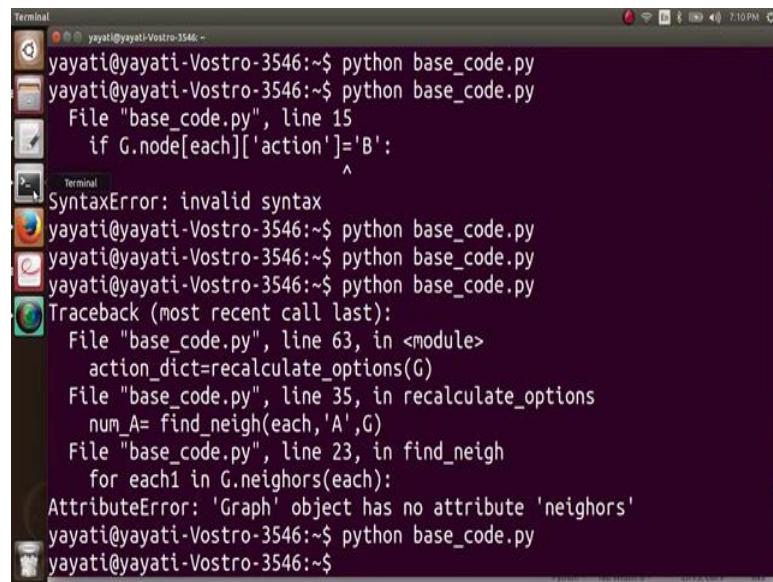
while(1):
    action_dict=recalculate_options(G)
    reset_node_attributes(G,action_dict)
    colors=get_colors(G)
    nx.draw(G,node_color=colors, node_size=800)
    plt.show()

Python Tab Width: 8 Ln 73, Col 9 INS
```

Now, one thing here, you see here currently we are running this process manually. So, this our initial graph right, `nx.draw` where we set 2 nodes to have, where we set this 2 nodes 3 and 7 to have this initial, to have this behaviour a and then, we draw this graph and then we recalculate the options and then we again draw the graph. So, let us automate this process. So, what we will do if we can put this thing inside a loop right.

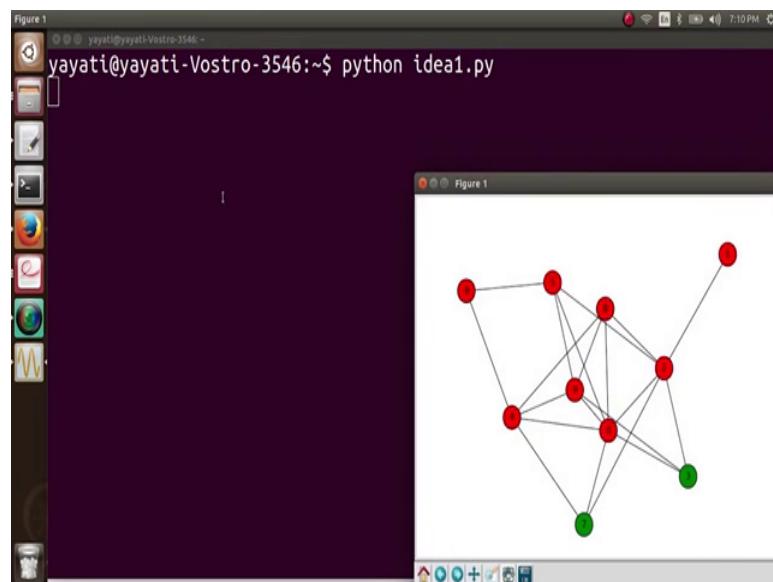
This complete thing can be put inside a loop. So, what will happen if I put it inside a loop? So, while 1, while 1, I just keep repeating this thing. So, every time my, so, this process is now become iterative. So, we start with some initial configuration and then every node recalculates its options and then it keeps drawing the graph again and again. So, I will just quickly show you.

(Refer Slide Time: 08:24)



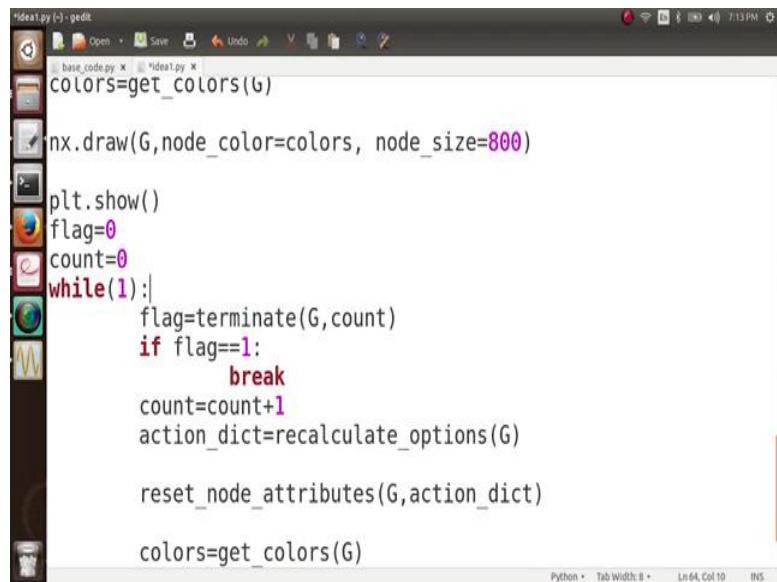
```
yayati@yayati-Vostro-3546:~$ python base_code.py
yayati@yayati-Vostro-3546:~$ python base_code.py
  File "base_code.py", line 15
    if G.node[each]['action']=='B':
                           ^
SyntaxError: invalid syntax
yayati@yayati-Vostro-3546:~$ python base_code.py
yayati@yayati-Vostro-3546:~$ python base_code.py
yayati@yayati-Vostro-3546:~$ python base_code.py
Traceback (most recent call last):
  File "base_code.py", line 63, in <module>
    action_dict=recalculate_options(G)
  File "base_code.py", line 35, in recalculate_options
    num_A= find_neigh(each,'A',G)
  File "base_code.py", line 23, in find_neigh
    for each1 in G.neighbors(each):
AttributeError: 'Graph' object has no attribute 'neighbors'
yayati@yayati-Vostro-3546:~$ python base_code.py
yayati@yayati-Vostro-3546:~$
```

(Refer Slide Time: 08:26)



So, if I implement this here, python idea1.py, so you see what? So, this is our initial graph and then you see it will keep drawing it again, and again, and again right. Now, we can see how the cascade is growing in the network. So, currently the cascade dies at the first step itself. So, we are unable to see anything ok.

(Refer Slide Time: 08:54)



```
ideal1.py (~) - gedit
base_code.py x  ideal.py x
colors=get_colors(G)
nx.draw(G,node_color=colors, node_size=800)
plt.show()
flag=0
count=0
while(1):
    flag=terminate(G,count)
    if flag==1:
        break
    count=count+1
    action_dict=recalculate_options(G)
    reset_node_attributes(G,action_dict)
    colors=get_colors(G)

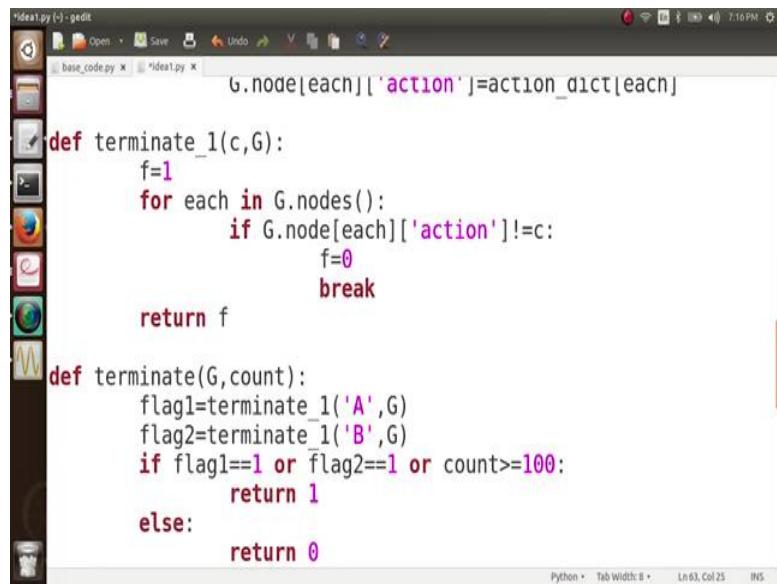
Python  Tab Width: 8  Ln 64, Col 10  INS
```

So what will do is, we want to put a terminating condition here. When should my code terminate? So, for that when should my code terminate? When should this process end? So, the process should end when a complete cascade occurs whether for the behaviour a or b. So, if everything has adopted the behaviour b, behaviour a die away and if everything, every node adopts behaviour a, behaviour b dies away.

So, in both of these conditions the process should come to an end; however, there is one more condition where it does not converge to either a, or b. The node keeps flipping. So, it might be, so, it goes into an infinite loop. So, what do we do for that condition is we keep a maximum iteration limit for 100? So, we do not want to see this process beyond 100 iterations. 100 is actually a big number. So, we put a terminating condition here. First of all, we take a variable flag here, flag = 0.

In while 1, we recalculate this flag. It will call a function terminate G which will decide whether we terminate or not. And if our flag == 1 then we break. So, this function terminate it runs some code and if the process has to be terminated, it should set flag = 1. Let us also take a variable count here, count equals to 0 and we pass count also here. And count should increase every time ok.

(Refer Slide Time: 10:55)



```
G.node[each]['action']=action_dict[each]

def terminate_1(c,G):
    f=1
    for each in G.nodes():
        if G.node[each]['action']!=c:
            f=0
            break
    return f

def terminate(G,count):
    flag1=terminate_1('A',G)
    flag2=terminate_1('B',G)
    if flag1==1 or flag2==1 or count>=100:
        return 1
    else:
        return 0
```

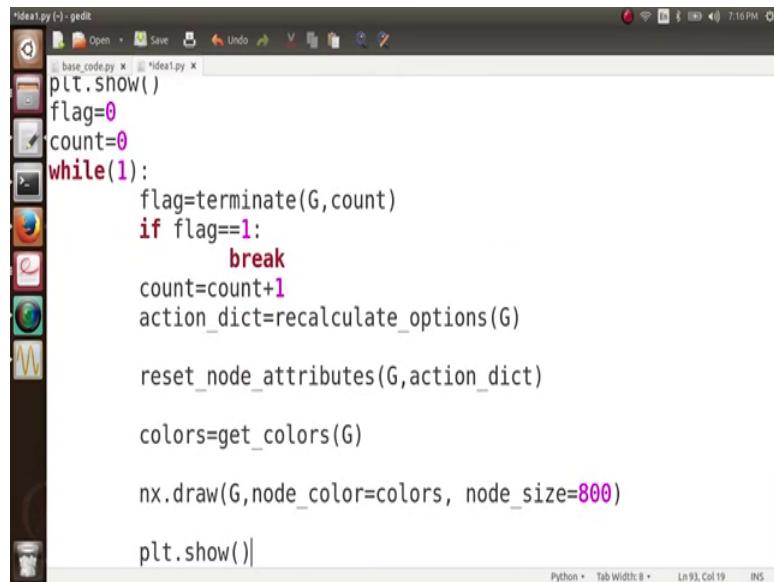
So, let us now define this function terminate ( $G$ ,  $count$ ), define  $terminate(G, count)$ . What is this function going to do is, if all the nodes have adopted the behaviour a or adopted the behaviour b, we terminate it? So, we take further two values flag 1, flag 1 is  $terminate\_1$ , a new function and it checks whether, so, flag 1 is going to check whether all the nodes in this network have adopted the behaviour a. All the nodes in this behaviour in this network have adopted the behaviour a and it is actually easy to check.

So, we define  $terminate\_1$  here and here we have A sorry, here we have a character  $c$  and the graph  $G$ . So, what we are going to do is, we take again indicator variable flag. What we do is for each in  $G.nodes$ , for everyone should have one behaviour  $c$ . So, as soon as we find a node, if  $G.node[each]$ ; as soon as we find a node whose action is not equal to  $c$ , but something else. It means that all the nodes in this network does not have this behaviour. We set  $f$  equals to 0 and we break. And we return  $f$  here.

So, you see  $f$  is going to return a 1 only if all the nodes in this network have this behaviour character  $c$ . So, flag 1 tells me whether all the nodes in this network have this behaviour a and similarly, flag 2 tells me whether all the nodes in this network have this behaviour b. These are 2 conditions and then, we have a count also. So, count should be less than 100. So, what do we do is if your, so, we have 3 conditions?

If our flag 1 is 1 or our flag 2 is 1 or our count value is greater than 100. Now either greater than or equals to 100, what do we do is we return 1 that, you should terminate else we return 0 ok.

(Refer Slide Time: 13:50)



```
idea1.py (~) - gedit
base_code.py x *idea1.py x
plt.show()
flag=0
count=0
while(1):
    flag=terminate(G,count)
    if flag==1:
        break
    count=count+1
    action_dict=recalculate_options(G)

    reset_node_attributes(G,action_dict)

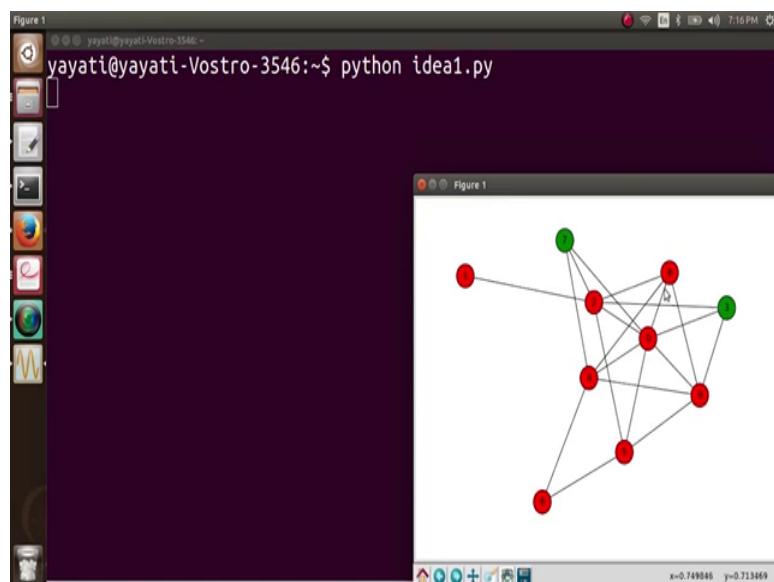
    colors=get_colors(G)

    nx.draw(G,node_color=colors, node_size=800)

    plt.show()|
```

And now also ok so, first let us execute and see what is happening.

(Refer Slide Time: 14:12)

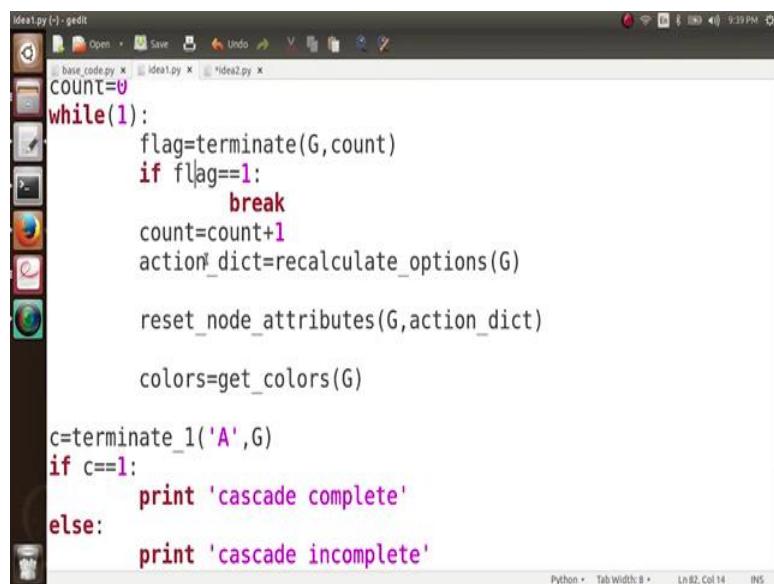


So, we execute it. So, we have this network where initially 7 and 3 have adopted the idea and then, this idea goes away. And you see what happened: 7 and 3 are infected and then this idea goes away. Everybody here has adopted b. So, the process stops.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**  
**Cascading Behavior in Networks**

**Lecture – 98**  
**Coding the Second Big Idea – Key People**

(Refer Slide Time: 00:05)



A screenshot of a terminal window titled "ideat.py - gedit". The window contains Python code for implementing the second big idea of cascading behavior. The code uses a while loop to repeatedly call functions like terminate and recalculate\_options until a condition is met. It also includes logic to print 'cascade complete' or 'cascade incomplete' based on a variable 'c'. The terminal interface shows tabs for "base code.py", "ideat.py", and "ideat2.py". The status bar at the bottom right indicates "Python", "Tab Width: 8", "Ln 82, Col 14", and "INS".

```
ideat.py (-) - gedit
base code.py  ideat.py  ideat2.py
COUNT=0
while(1):
    flag=terminate(G,count)
    if flag==1:
        break
    count=count+1
    action_dict=recalculate_options(G)

    reset_node_attributes(G,action_dict)

    colors=get_colors(G)

    c=terminate_1('A',G)
    if c==1:
        print 'cascade complete'
    else:
        print 'cascade incomplete'
```

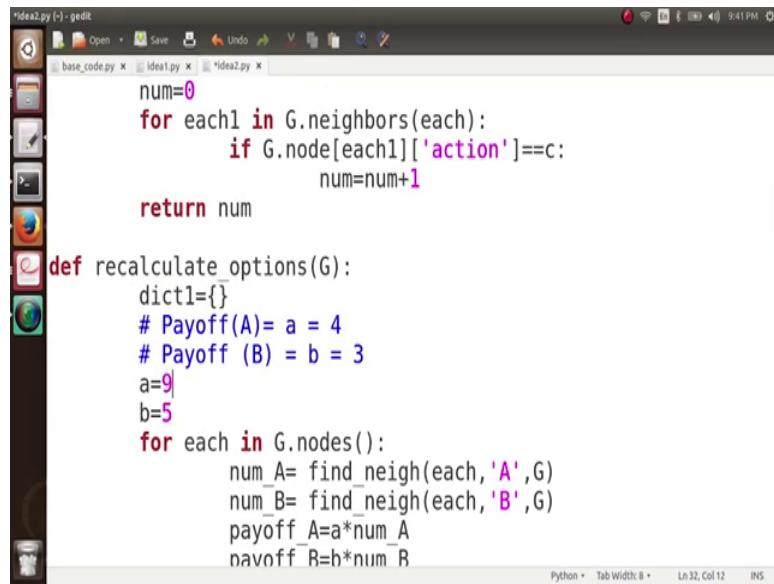
So, now we are going to implement the second idea. What was the second idea? In the first idea we looked that increasing the payoff associated with this new action or this new behaviour can cause a complete cascade. In a second idea, we want to talk about key people right. We, are going to keep the payoff associated with a particular action same, but what we are interested in?

What we are interested in looking at is, we will take this network again and we see that when we start with some particular bunch of nodes it creates a complete cascade, but we start from some other bunch of nodes, they are not able to create a complete cascade. It means that different nodes in this network have different power to create your cascade and these bunch of people which if chosen, creates your complete cascade they are known as the key people.

Again, I would like to remind you about digital marketing in choosing the right set of people to popularize your idea. So, what we are going to do in the second screen cast is,

we are going to take a network and then we are going to choose different sets of initial adopters and will look at why certain sets are able to create a complete cascade, other sets are unable to create a complete cascade. So, I will take this same code and I will put it in a different file idea2.py ok.

(Refer Slide Time: 01:36)

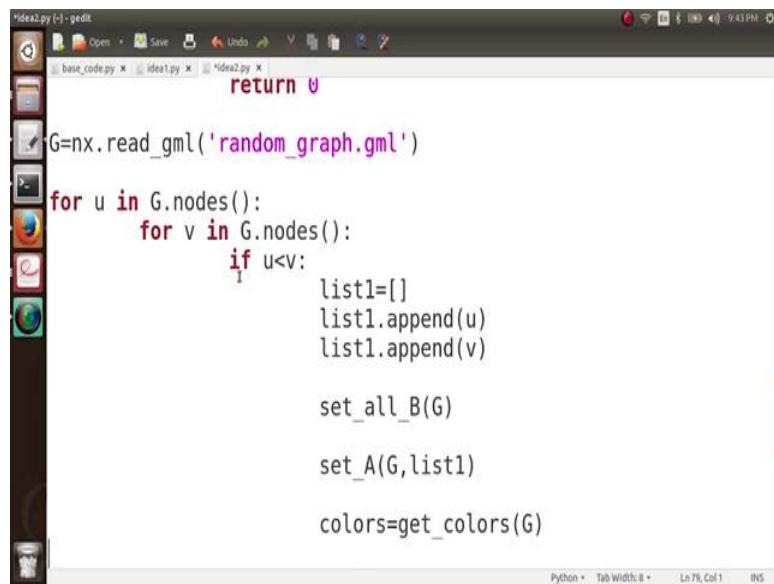


```
idea2.py (~) - gedit
base_code.py  idea1.py  idea2.py
num=0
for each1 in G.neighbors(each):
    if G.node[each1]['action']==c:
        num=num+1
return num

def recalculate_options(G):
    dict1={}
    # Payoff(A)= a = 4
    # Payoff (B) = b = 3
    a=9
    b=5
    for each in G.nodes():
        num_A= find_neigh(each,'A',G)
        num_B= find_neigh(each,'B',G)
        payoff_A=a*num_A
        payoff_B=b*num_B
```

So, now what we do is, first I change my payoff let us say the payoff associated with a is 9 and with b is 5.

(Refer Slide Time: 01:49)



```
idea2.py (~) - gedit
base_code.py  idea1.py  idea2.py
return v

G=nx.read_gml('random_graph.gml')

for u in G.nodes():
    for v in G.nodes():
        if u<v:
            list1=[]
            list1.append(u)
            list1.append(v)

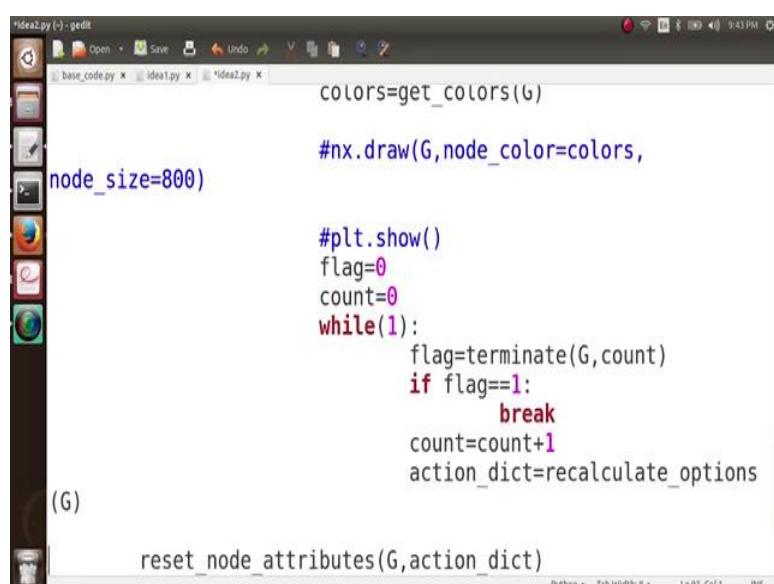
            set_all_B(G)
            set_A(G,list1)
            colors=get_colors(G)
```

What we want to do is, I am going to take 2 initial adopters. So, these 2 initial adopters can be any 2 nodes in the networks. So, for every possible set of 2 nodes in the network, we will start the cascade from there and we check whether this node have been able to create a complete cascade or not.

And we are using the same previous graph as before random\_graph.gml. So, what I am going to do here is for  $u$  in  $G.nodes$  and then, for  $v$  in  $G.nodes$ , I do not want my sets to be repeated. So, if I do it for like this for  $v$  in  $G.nodes$ ,  $v$  in  $G.nodes$  1, 2 means node 1 and node 2 will come once and node 2 and node 1 will come once, rather node 1 and here again it will choose 1 again. I do not want such kind of repetitions. So, I put a loop here node loop condition here. If  $u < v$  only then, it is what I am doing will be considered.

So, if  $u$  is less than  $v$  now, I create my list 1. So, for every possible values of  $u$  and  $v$  what I do is, I append both values in my list,  $list.append(u)$  and then,  $list.append(v)$ . So, what is happening here? Now list 1 consists of my initial set, initial set of adopters. So, this loop condition will both view and take each set of 2 nodes in my network and after taking each and every set of 2 nodes in the network, we can see whether my cascade was complete or not. While I just shift everything inside these loops ok. I do not want to draw my graphs here because there will be so many sets and we cannot see the graph for each and every set.

(Refer Slide Time: 04:08)



The screenshot shows a Gedit code editor window with the file 'ideaz2.py' open. The code implements a social network cascade simulation using NetworkX and Matplotlib. It starts by importing necessary modules and defining a function to get colors for nodes. The main logic involves drawing the graph, initializing variables (flag=0, count=0), and entering a while loop. Inside the loop, it checks if the cascade has terminated (flag=1) and increments the count if not. It also recalculates options and updates node attributes. Finally, it calls a function to reset node attributes.

```
ideaz2.py (-) - gedit
base_code.py * ideaz1.py * ideaz2.py
colors=get_colors(G)

#nx.draw(G,node_color=colors,
node_size=800)

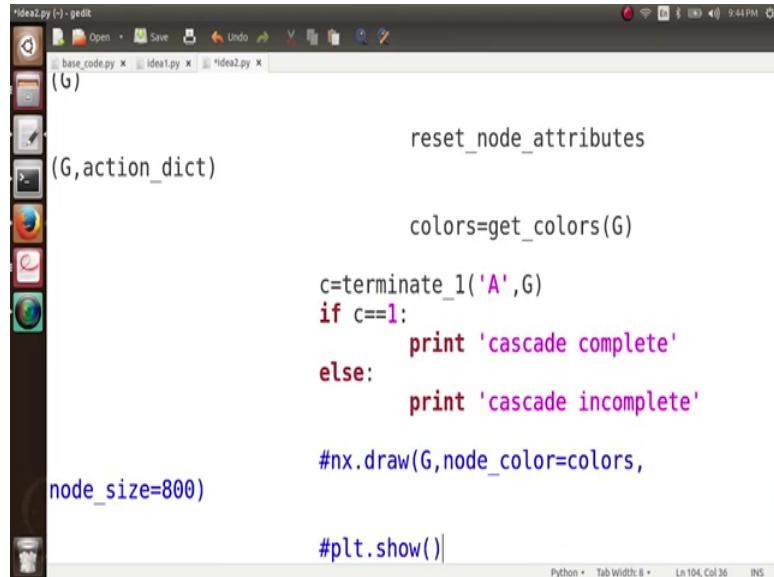
#plt.show()
flag=0
count=0
while(1):
    flag=terminate(G,count)
    if flag==1:
        break
    count=count+1
    action_dict=recalculate_options

(G)

reset_node_attributes(G,action_dict)
```

It will be a time-consuming process. We just want to see whether the cascade is complete or not.

(Refer Slide Time: 04:27)



```
idea2.py (~) - gedit
(base_code.py x : ideat.py x : *idea2.py x | (G)
reset_node_attributes
(G,action_dict)

colors=get_colors(G)

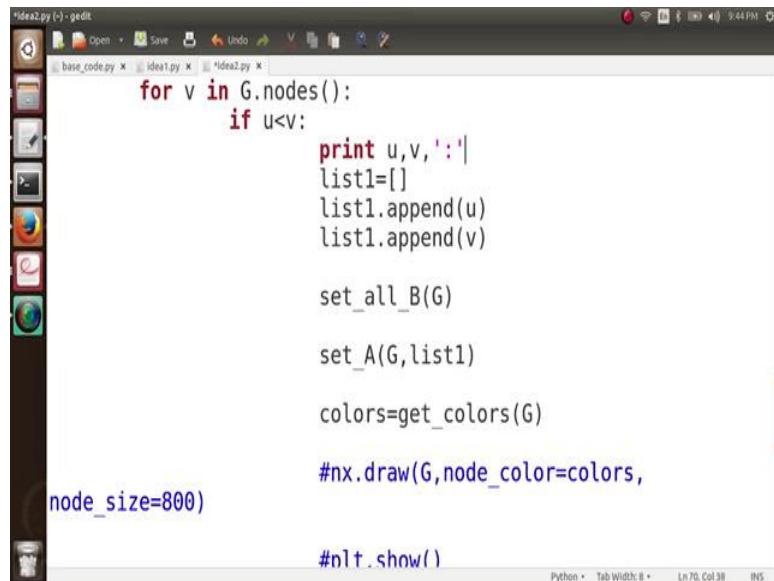
c=terminate_1('A',G)
if c==1:
    print 'cascade complete'
else:
    print 'cascade incomplete'

#nx.draw(G,node_color=colors,
node_size=800)

# plt.show()
Python • Tab Width: 8 • Ln 104, Col 36 INS
```

And here also we do not want to draw this graph ok.

(Refer Slide Time: 04:49)



```
idea2.py (~) - gedit
(base_code.py x : ideat.py x : *idea2.py x | (G)
for v in G.nodes():
    if u<v:
        print u,v,':'
        list1=[]
        list1.append(u)
        list1.append(v)

        set_all_B(G)
        set_A(G,list1)

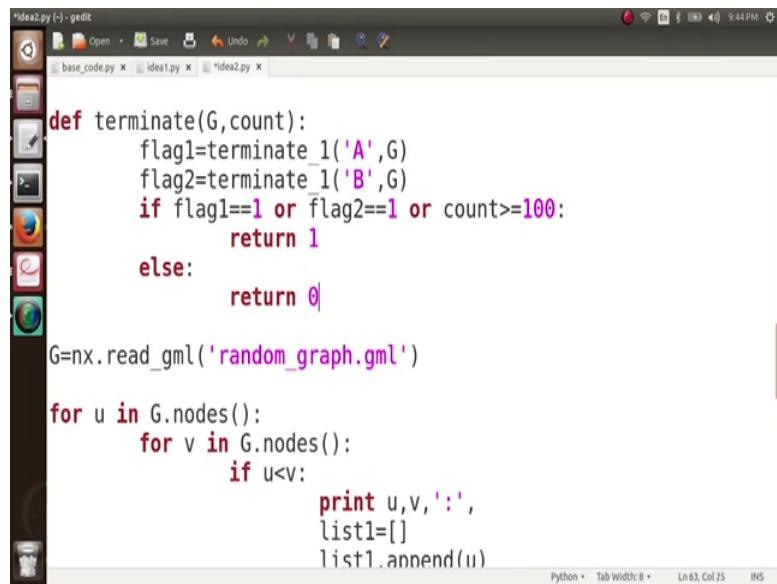
        colors=get_colors(G)

        #nx.draw(G,node_color=colors,
node_size=800)

        # plt.show()
Python • Tab Width: 8 • Ln 70, Col 38 INS
```

And also, what I do is, I print the values of  $u$  and  $v$  here. So, that I know what is happening, for which nodes we are getting the particular result ok. So, what this piece of code is doing? Let me quickly show you.

(Refer Slide Time: 05:04)



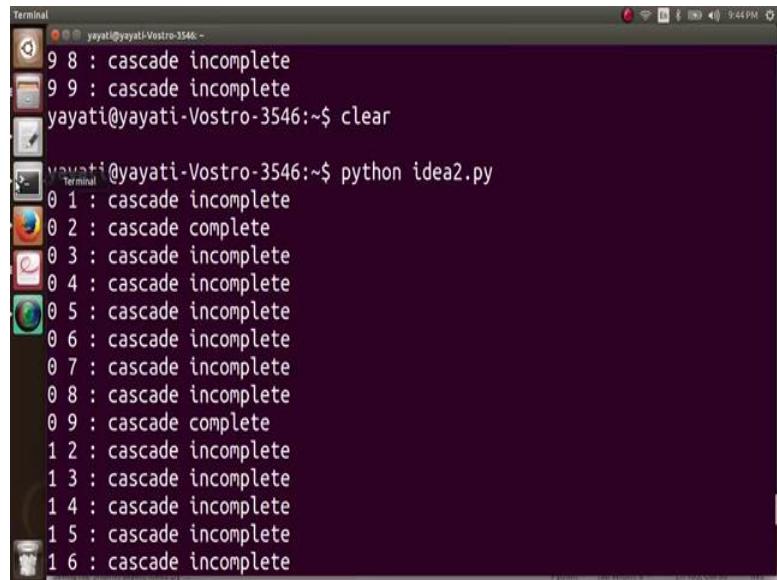
```
idea2.py (~) - gedit
base_code.py  idea1.py  idea2.py
def terminate(G,count):
    flag1=terminate_1('A',G)
    flag2=terminate_1('B',G)
    if flag1==1 or flag2==1 or count>=100:
        return 1
    else:
        return 0

G=nx.read_gml('random_graph.gml')

for u in G.nodes():
    for v in G.nodes():
        if u<v:
            print u,v,':',
            list1=[]
            list1.append(u)
```

So, we have a network here random\_graph.gml. For every possible set of 2 nodes we take them as the initial adopters in list1. We start the cascade from these 2 nodes, and we see at and whether the, it these 2 nodes the result in a complete cascade or not. So, it is a simple code.

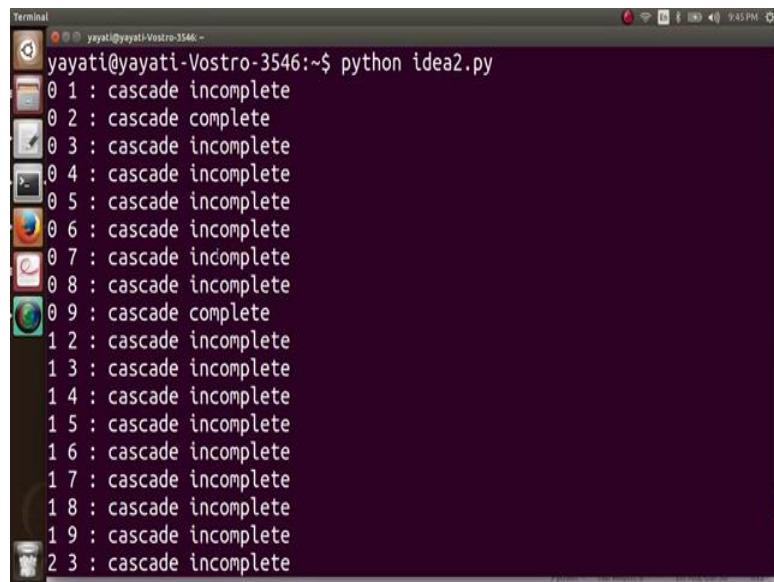
(Refer Slide Time: 05:22)



```
yayati@yayati-Vostro-3546:~$ python idea2.py
0 1 : cascade incomplete
0 2 : cascade complete
0 3 : cascade incomplete
0 4 : cascade incomplete
0 5 : cascade incomplete
0 6 : cascade incomplete
0 7 : cascade incomplete
0 8 : cascade incomplete
0 9 : cascade complete
1 2 : cascade incomplete
1 3 : cascade incomplete
1 4 : cascade incomplete
1 5 : cascade incomplete
1 6 : cascade incomplete
```

Let us run it here now ok. So, let us see this is the results so for all possible set of nodes. You see here for most of the sets the cascade remains incomplete. But there are few sets here. Like 0 2 and then, 0 9 and then 1 here 2 9.

(Refer Slide Time: 05:45)



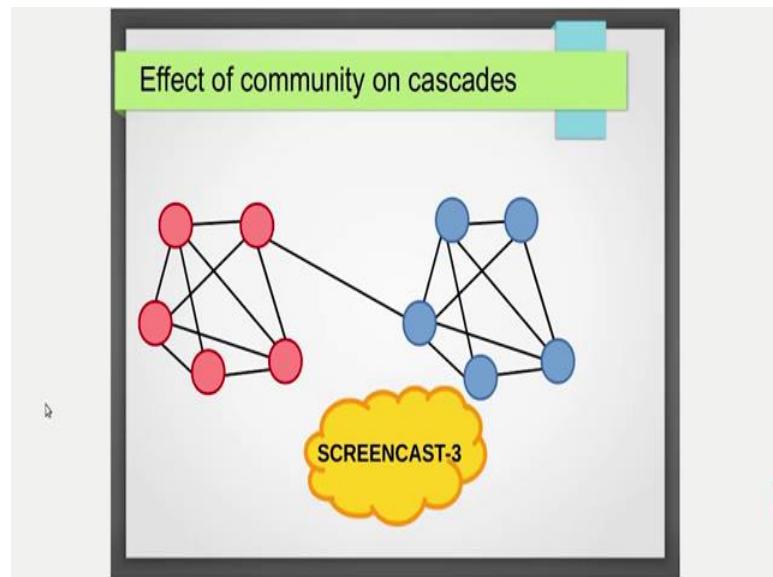
```
yayati@yayati-Vostro-3546:~$ python idea2.py
0 1 : cascade incomplete
0 2 : cascade complete
0 3 : cascade incomplete
0 4 : cascade incomplete
0 5 : cascade incomplete
0 6 : cascade incomplete
0 7 : cascade incomplete
0 8 : cascade incomplete
0 9 : cascade complete
1 2 : cascade incomplete
1 3 : cascade incomplete
1 4 : cascade incomplete
1 5 : cascade incomplete
1 6 : cascade incomplete
1 7 : cascade incomplete
1 8 : cascade incomplete
1 9 : cascade incomplete
2 3 : cascade incomplete
```

So, only these 3 sets they result in a complete cascade. All the other sets they result in an incomplete cascade. This shows us this that yes, there are certain key people, certain key nodes in this network and if you start your cascade from there. It tends to become a complete cascade. Proving our second idea that yes, there are key people in this network.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

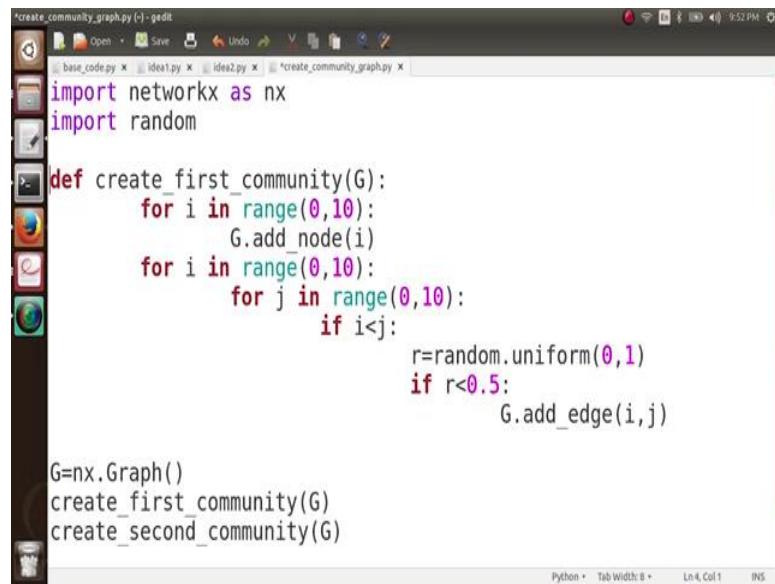
**Cascading Behavior in Networks**  
**Lecture - 99**  
**Coding the Third Big Idea- Impact of Communities on Cascades**

(Refer Slide Time: 00:05)



Now, we are going to talk about the third idea which is very cute which says that if you have many communities in your network. So, for the sake of simplicity, we will take two communities and we show that if our cascade starts from one community; then it actually even if it gets into this entire community, it is difficult for this cascade to get into another community. So, we will be coding and now we will be looking at this aspect.

(Refer Slide Time: 00:31)



```
create_community_graph.py (~) - gedit
base_code.py x idea1.py x idea2.py x create_community_graph.py x
import networkx as nx
import random

def create_first_community(G):
    for i in range(0,10):
        G.add_node(i)
    for i in range(0,10):
        for j in range(0,10):
            if i<j:
                r=random.uniform(0,1)
                if r<0.5:
                    G.add_edge(i,j)

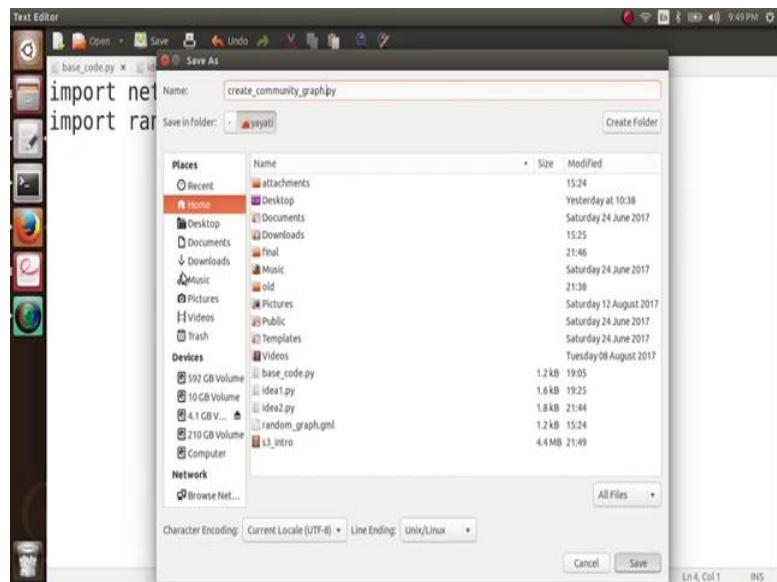
G=nx.Graph()
create_first_community(G)
create_second_community(G)

Python • Tab Width: 8 • Ln: 4 Col: 1 INS
```

So, for implementing this, we need a graph which consist of two communities: two dense communities. So, what will we do? What will we do? We will create an artificial graph which will be consisting of two communities. How do we create this artificial graph is we will create a graph having 20 nodes? So, first 10 nodes will be connected to each other and second 10 nodes means nodes 0 to 9 will be connected to each other and node 10 to 19 will be connected to each other and there will be just one link between these two communities.

Let us just quickly do it. So, I will just import networkx as nx and I will also need the random functions. So, a import random as well.

(Refer Slide Time: 01:24)



Let me save this as this file as create community graph.py ok. So, I import it random and then what I will do is I create a graph  $G = nx.Graph$ . After this I create the first community in this graph. So, I call the function create first community and what does this function do is, it just creates a random links between the people I will just show you. So, define, create, first community and we pass the graph here first community  $G$  and what you do here is ok. So, first I add 10 nodes in this community and no edge is. So, for  $i$  in range 0 to 10, what do I do is  $G.add\_node(i)$ . So, I have added 10 nodes in this graph, next I want to add some links.

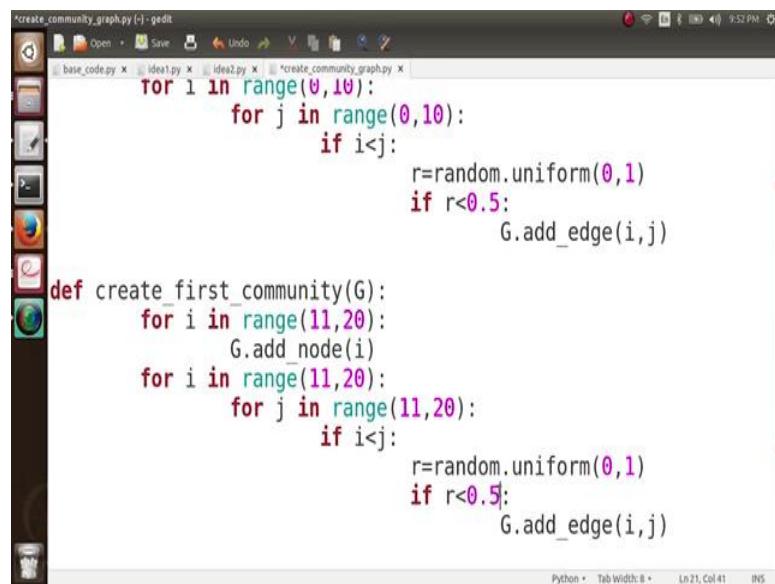
So, how do I add some links? For as we know these are 10 nodes and there are 10 chose to possible edges between these nodes. So, I will take every edge and I will put every edge with the probability of 0.5. So, I am doing exactly what happens in an Erdos-Renyi graph, but I am just coding it manually. So, for  $i$  in range 0 to 0 so, I will put first iterator over all of these nodes and then for  $j$  in range 0 to 10, I put a second iterator here and again to avoid duplication and the same number if  $i < j$  so, that I get every pair only once.

What do I do? I will want to put an edge here with the probability of 0.5 and I am quite sure that you know what I am going to do next because we have seen it previously. We have coded it also previously. If I want to do some event with a probability, I create a random number; random uniform number real number from 0 to 1. So, I will get a real

number from 0 to 1.  $r = \text{random.uniform}$ ,  $\text{random.uniform}(0, 1)$  and then if  $r$  is less than 0.5. What I do is  $G.\text{add\_edge}$  form  $i$  to  $j$ .

So, I have added every possible edge between these 10 nodes with the probability of 0.5. So, this is my first community. Similarly, I will create my second community. Create second community and I am sure that you know that the code is going to be same except for the numbering of nodes.

(Refer Slide Time: 04:45)



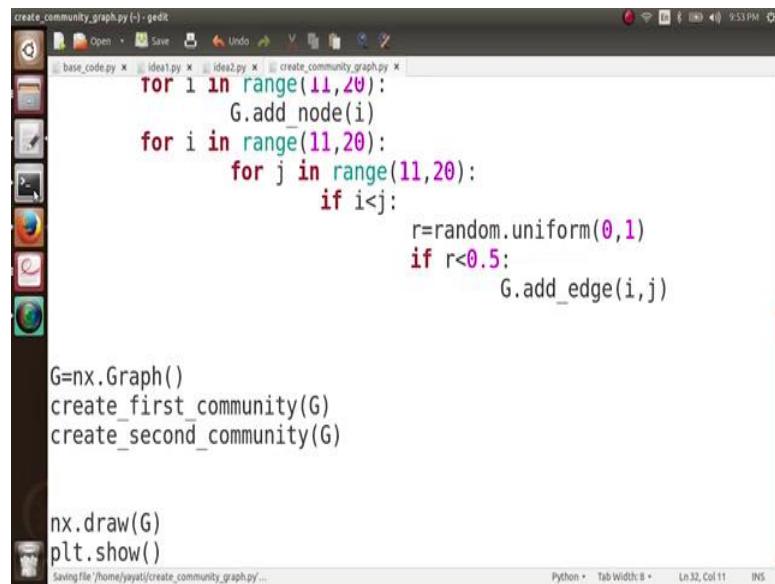
```
create_community_graph.py (-) -gedit
base_code.py x idea1.py x idea2.py x *create_community_graph.py x
for i in range(0,10):
    for j in range(0,10):
        if i<j:
            r=random.uniform(0,1)
            if r<0.5:
                G.add_edge(i,j)

def create_first_community(G):
    for i in range(11,20):
        G.add_node(i)
    for i in range(11,20):
        for j in range(11,20):
            if i<j:
                r=random.uniform(0,1)
                if r<0.5:
                    G.add_edge(i,j)

Python * Tab Width: 8 * Ln 21, Col 41 INS
```

So, we can just copy paste this code here and what I can do is I know that my nodes are now going to run from 11 to 20 right. So, from 11 to 20 and here also 11 to 20 and here also 11 to 20, everything is done. So, 10 nodes I created here in the; similar way 10 nodes, I created here and between these 10 nodes. I will again put an edge with the probability of 0.5.

(Refer Slide Time: 05:11)



```
create_community_graph.py (~) - gedit
base_code.py x idea1.py x idea2.py x create_community_graph.py x
for i in range(11,20):
    G.add_node(i)
for i in range(11,20):
    for j in range(11,20):
        if i<j:
            r=random.uniform(0,1)
            if r<0.5:
                G.add_edge(i,j)

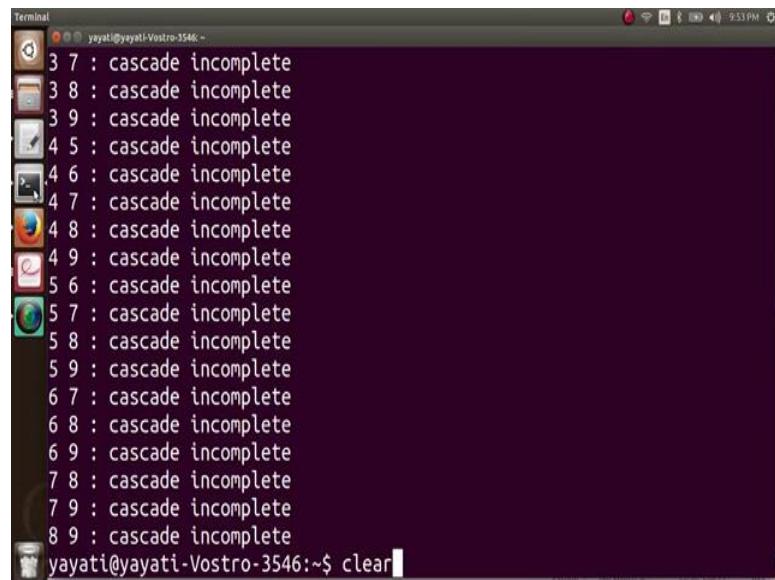
G=nx.Graph()
create_first_community(G)
create_second_community(G)

nx.draw(G)
plt.show()

Saving file /home/yayati/create_community_graph.py...
Python Tab Width: 8 Line 32, Col 11 INS
```

So, what am I going to have now in my network, there will be two separate communities, or you can say two components in this graph? So, what currently we have made is a disconnected graph. I can also actually show you this graph. So, let me show you this graph nx.draw (G), then we have a plt.show right and let us run.

(Refer Slide Time: 05:48)



```
Terminal yayati@yayati-Vostro-3546: ~
3 7 : cascade incomplete
3 8 : cascade incomplete
3 9 : cascade incomplete
4 5 : cascade incomplete
4 6 : cascade incomplete
4 7 : cascade incomplete
4 8 : cascade incomplete
4 9 : cascade incomplete
5 6 : cascade incomplete
5 7 : cascade incomplete
5 8 : cascade incomplete
5 9 : cascade incomplete
6 7 : cascade incomplete
6 8 : cascade incomplete
6 9 : cascade incomplete
7 8 : cascade incomplete
7 9 : cascade incomplete
8 9 : cascade incomplete
yayati@yayati-Vostro-3546:~$ clear
```

It creates community graph.

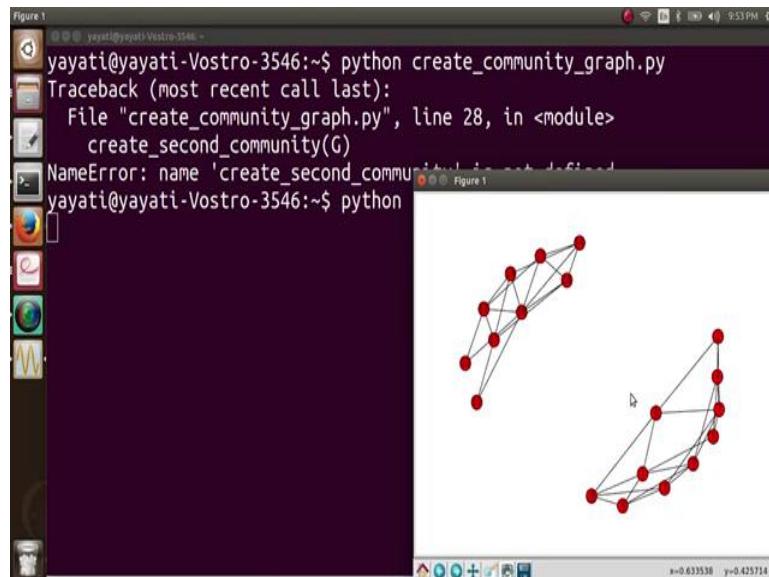
(Refer Slide Time: 05:52)



```
yayati@yayati-Vostro-3546:~$ python create_community_graph.py
Traceback (most recent call last):
  File "create_community_graph.py", line 28, in <module>
    create_second_community(G)
NameError: name 'create_second_community' is not defined
yayati@yayati-Vostro-3546:~$ python create_community_graph.py
yayati@yayati-Vostro-3546:~$ python create_community_graph.py
yayati@yayati-Vostro-3546:~$ python create_community_graph.py
yayati@yayati-Vostro-3546:~$ python idea3.
```

I am just create.

(Refer Slide Time: 06:01)

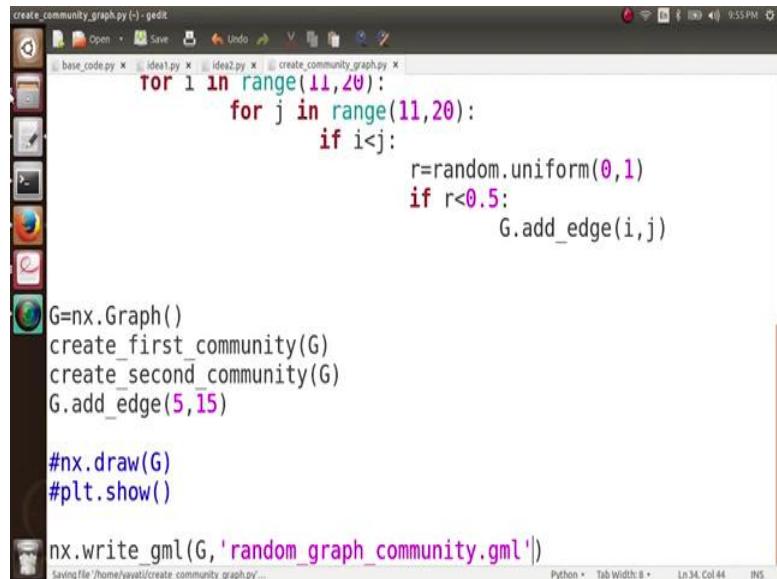


We forgot to change the name of function here. So, it is create second community. So, you can see this graph here right. We have one community here having which is random graph with the 10 nodes and 0.5 probability and one community here.

Now, these are components we cannot say these are communities also, but they are components; we want a connected graph. So, what I am going to put next is I am going

to put just one edge between these two communities. So, that the communities in my graph are very well defined; I put only 1 edge.

(Refer Slide Time: 06:42)



```
base_code.py x idea1.py x idea2.py x create_community_graph.py x
for i in range(11,20):
    for j in range(11,20):
        if i<j:
            r=random.uniform(0,1)
            if r<0.5:
                G.add_edge(i,j)

G=nx.Graph()
create_first_community(G)
create_second_community(G)
G.add_edge(5,15)

#nx.draw(G)
#plt.show()

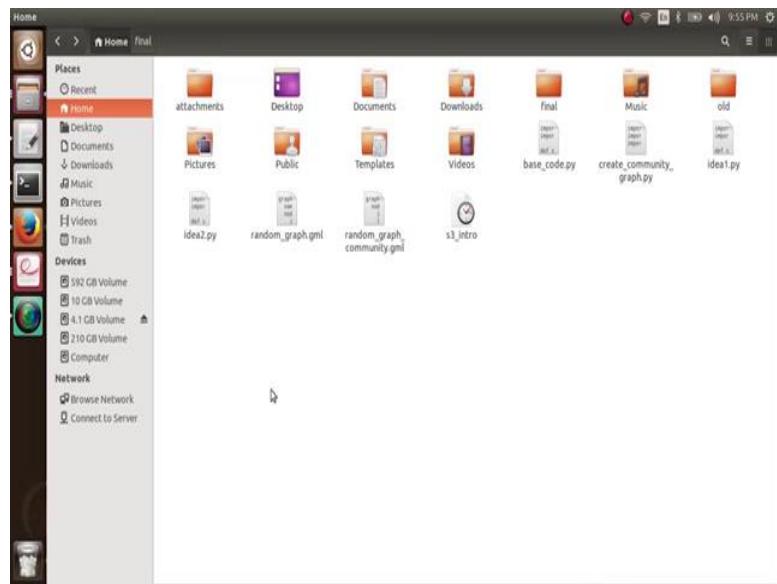
nx.write_gml(G, 'random_graph_community.gml')
Saving file '/home/yayati/create_community_graph.py...'
```

So, what do I do is `G.add_edge` and I add an edge.

Student: (Refer Time: 06:48).

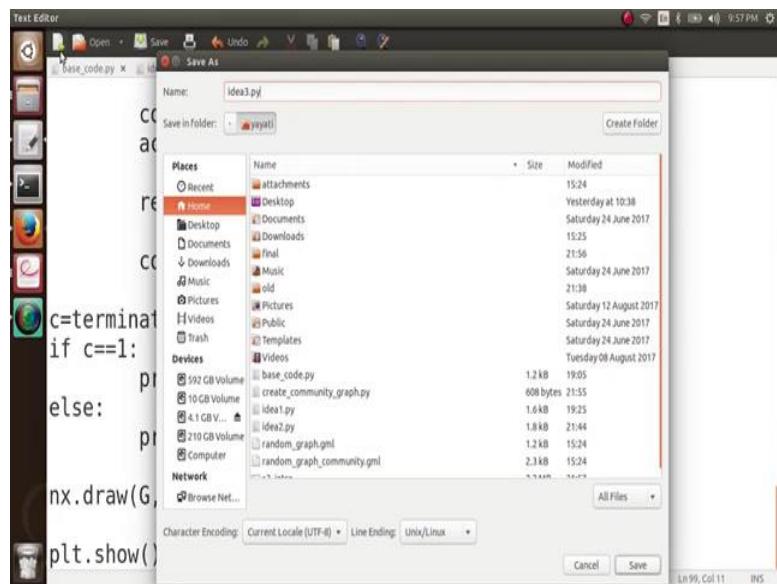
Let us say from node 5 to 15. So, let us run it. So, you can see here now this is my graph having two communities. We are going to work with this graph for a third idea, for the implementation of the third idea. What I will do is, I store this graph `nx.write_gml` and I store this graph as let me name it as random graph with community. So, random graph community.sorry.gml write\_gml; I run it.

(Refer Slide Time: 07:41)



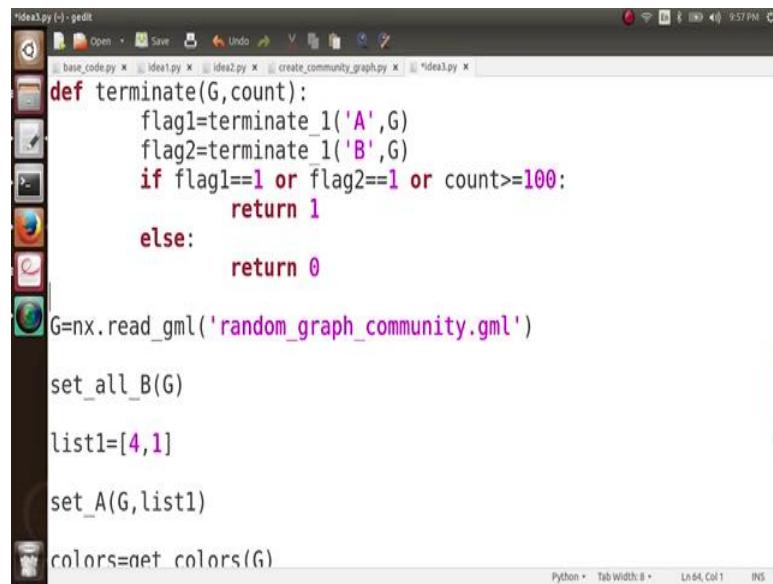
So, you can see a graph here random graph community.gml ok. So, now, we are ready to see. So, we have a graph with community, and we are ready to see the impact of cascading on such a network.

(Refer Slide Time: 08:04)



So, I will actually take my code from here and Take my code from here and let us save it here. We will do all the changes here. Let us call it idea 3.py.

(Refer Slide Time: 08:13)



```
idea3.py (-) - gedit
base_code.py x idea1.py x idea2.py x create_community_graph.py x idea3.py x
def terminate(G,count):
    flag1=terminate_1('A',G)
    flag2=terminate_1('B',G)
    if flag1==1 or flag2==1 or count>=100:
        return 1
    else:
        return 0

G=nx.read_gml('random_graph_community.gml')

set_all_B(G)

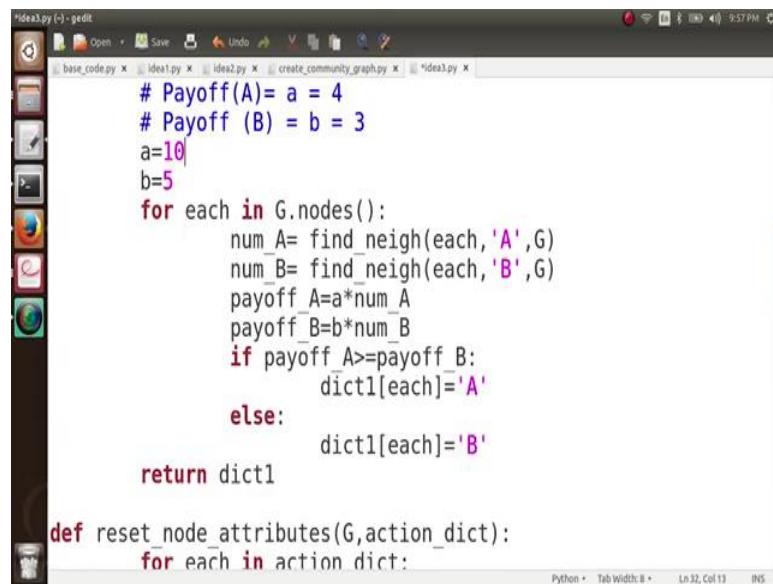
list1=[4,1]

set_A(G,list1)

colors=get_colors(G)
Python • Tab Width: 8 • Ln 64, Col 1 INS
```

So, first of all which is the graph we are going to work with this random underscore graph underscore community one thing and then let us change the payoffs little bit or let us keep it the say.

(Refer Slide Time: 08:26)

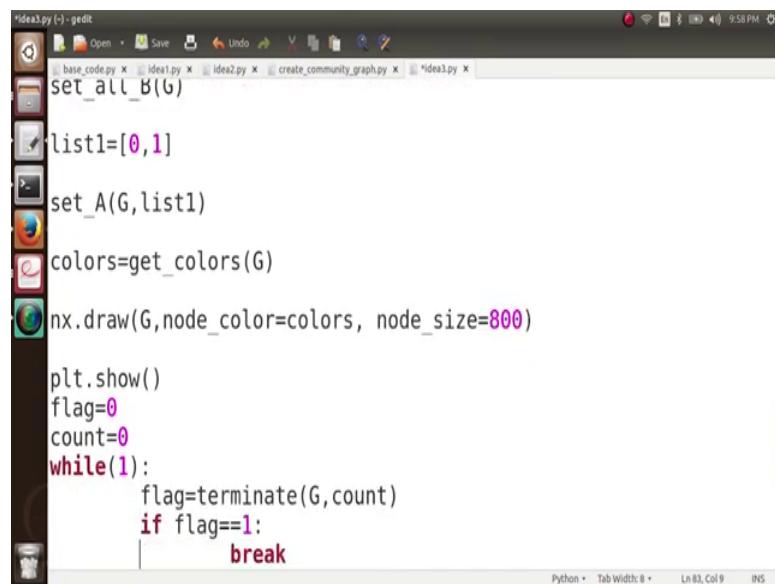


```
# Payoff(A)= a = 4
# Payoff (B) = b = 3
a=10
b=5
for each in G.nodes():
    num_A= find_neigh(each,'A',G)
    num_B= find_neigh(each,'B',G)
    payoff_A=a*num_A
    payoff_B=b*num_B
    if payoff_A>=payoff_B:
        dict1[each]='A'
    else:
        dict1[each]='B'
return dict1

def reset_node_attributes(G,action_dict):
    for each in action_dict:
Python • Tab Width: 8 • Ln 32, Col 13 INS
```

So, let the payoffs be 10 and 5 and let my starting nodes be 0 and 1.

(Refer Slide Time: 08:36)



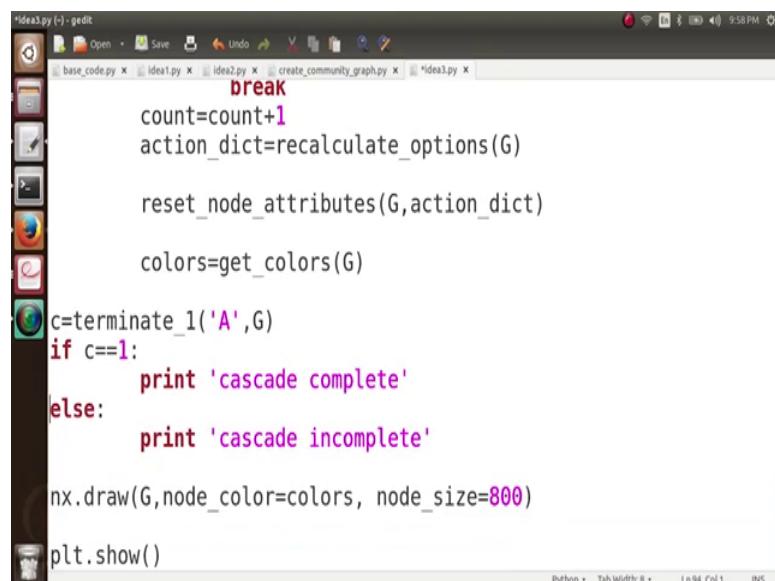
```
idea3.py (~) - gedit
base_code.py x idea1.py x idea2.py x create_community_graph.py x idea3.py x
set_all_B(G)

list1=[0,1]
set_A(G,list1)
colors=get_colors(G)
nx.draw(G,node_color=colors, node_size=800)

plt.show()
flag=0
count=0
while(1):
    flag=terminate(G,count)
    if flag==1:
        break
```

And what I am interested in is looking at how will this cascade going to occur.

(Refer Slide Time: 08:42)



```
break
count=count+1
action_dict=recalculate_options(G)

reset_node_attributes(G,action_dict)

colors=get_colors(G)

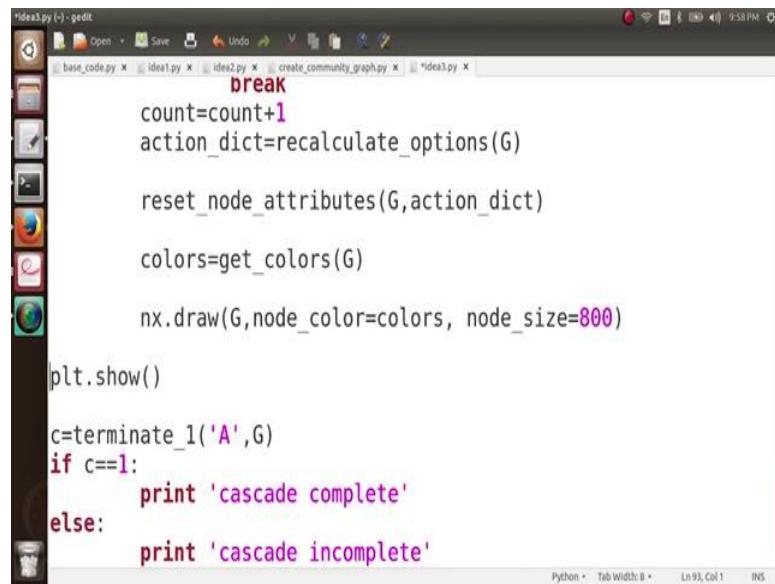
c=terminate_1('A',G)
if c==1:
    print 'cascade complete'
else:
    print 'cascade incomplete'

nx.draw(G,node_color=colors, node_size=800)

plt.show()
```

And let me look at this in every step.

(Refer Slide Time: 08:48)



```
idea3.py (-) - gedit
base_code.py x idea1.py x idea2.py x create_community_graph.py x idea3.py x
break
count=count+1
action_dict=recalculate_options(G)

reset_node_attributes(G,action_dict)

colors=get_colors(G)

nx.draw(G,node_color=colors, node_size=800)

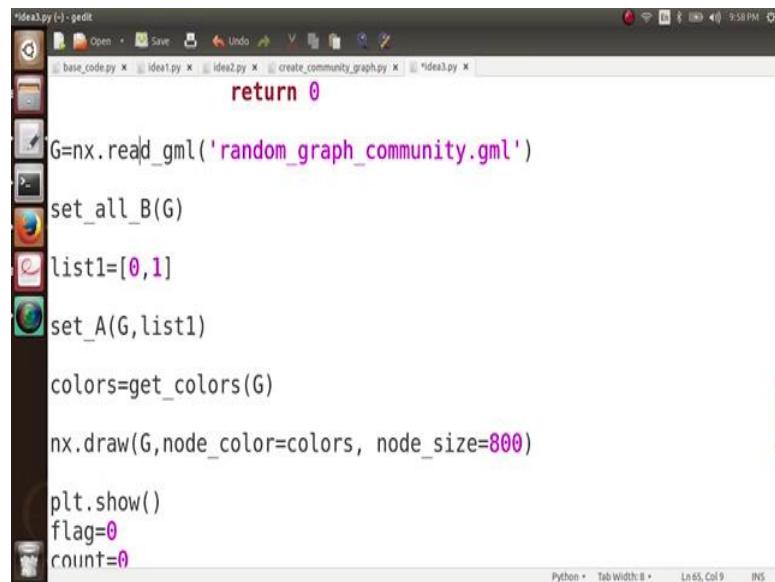
plt.show()

c=terminate_1('A',G)
if c==1:
    print 'cascade complete'
else:
    print 'cascade incomplete'

Python Tab Width: 8 Ln 93, Col 1 INS
```

I want to see the graph at every step. So, you can see I will just show you the code. It is the same code which we have written previously.

(Refer Slide Time: 08:58)



```
idea3.py (-) - gedit
base_code.py x idea1.py x idea2.py x create_community_graph.py x idea3.py x
return 0

G=nx.read_gml('random_graph_community.gml')

set_all_B(G)

list1=[0,1]

set_A(G,list1)

colors=get_colors(G)

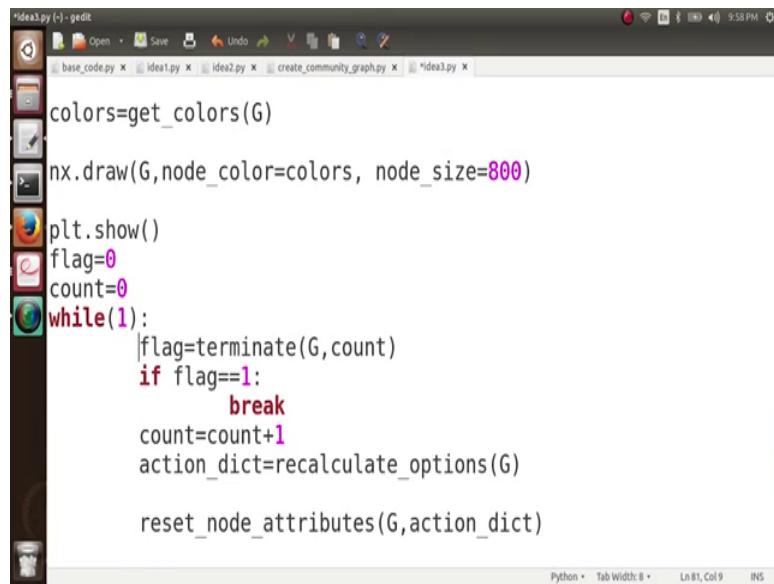
nx.draw(G,node_color=colors, node_size=800)

plt.show()
flag=0
count=0

Python Tab Width: 8 Ln 65, Col 9 INS
```

So, here we load a graph. Here we set all the nodes to have the action B, here we choose the initial adopters. We set the initial adopters, we draw the graph.

(Refer Slide Time: 09:09)

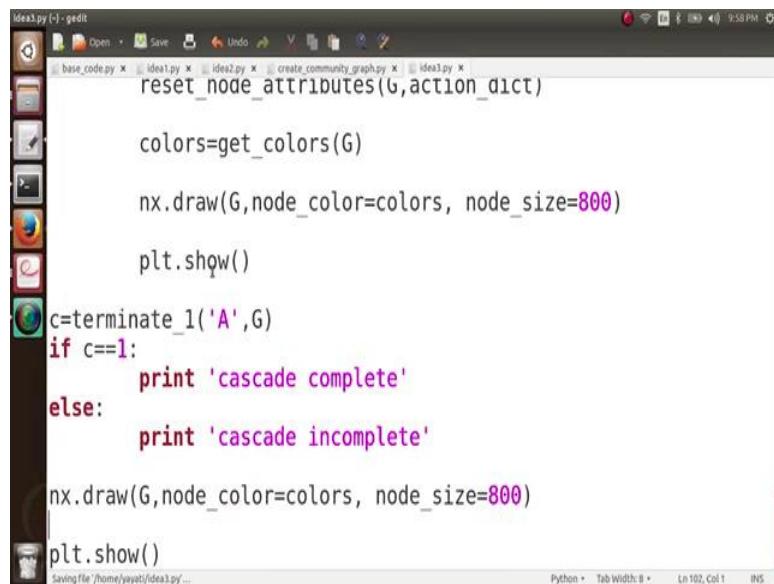


```
idea3.py (-) - gedit
base_code.py x idea1.py x idea2.py x create_community_graph.py x idea3.py x
colors=get_colors(G)
nx.draw(G,node_color=colors, node_size=800)
plt.show()
flag=0
count=0
while(1):
    flag=terminate(G,count)
    if flag==1:
        break
    count=count+1
    action_dict=recalculate_options(G)
    reset_node_attributes(G,action_dict)

Python • Tab Width: 8 • Ln 81, Col 9 INS
```

Next, this is just a while loop in which the process runs again and again. Terminating condition, as we have discussed before action dictionary, we calculate the next snapshot and then we upload the next snapshot exactly what we want to do.

(Refer Slide Time: 09:25)



```
idea3.py (-) - gedit
reset_node_attributes(G,action_dict)

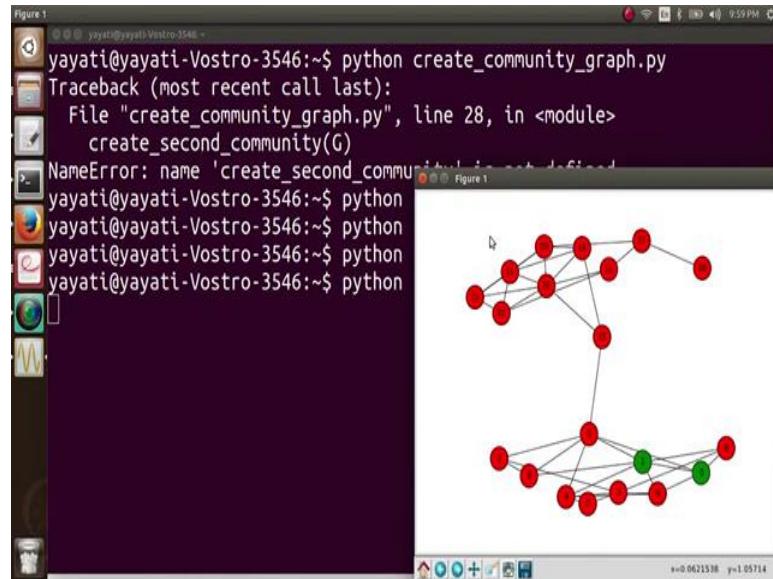
colors=get_colors(G)
nx.draw(G,node_color=colors, node_size=800)
plt.show()
c=terminate_1('A',G)
if c==1:
    print 'cascade complete'
else:
    print 'cascade incomplete'
nx.draw(G,node_color=colors, node_size=800)
plt.show()

Saving file /home/yayati/idea3.py...
Python • Tab Width: 8 • Ln 102, Col 1 INS
```

So, we just the code is very simple, we just have to run this code and see what happens.

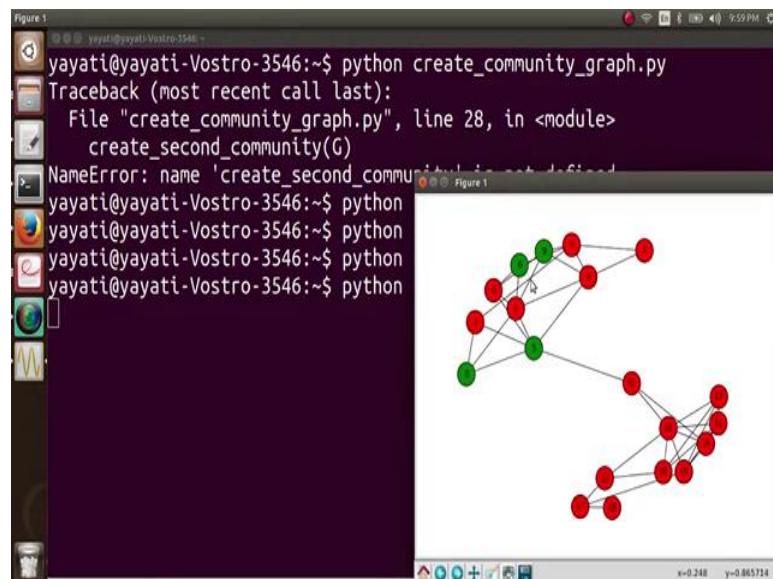
So, let us implement it let us execute it python idea 3.py ok.

(Refer Slide Time: 09:37)



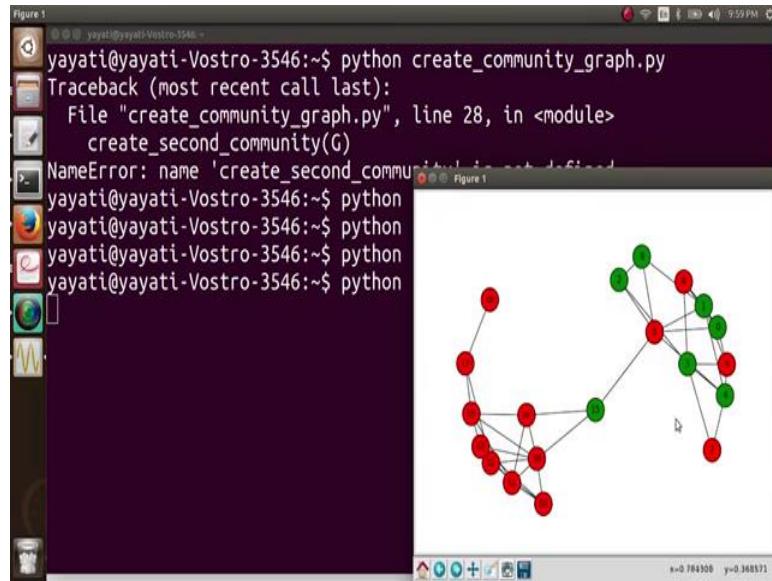
So, this is your initial graph and the nodes 0 and 1 are infected here.

(Refer Slide Time: 09:44)



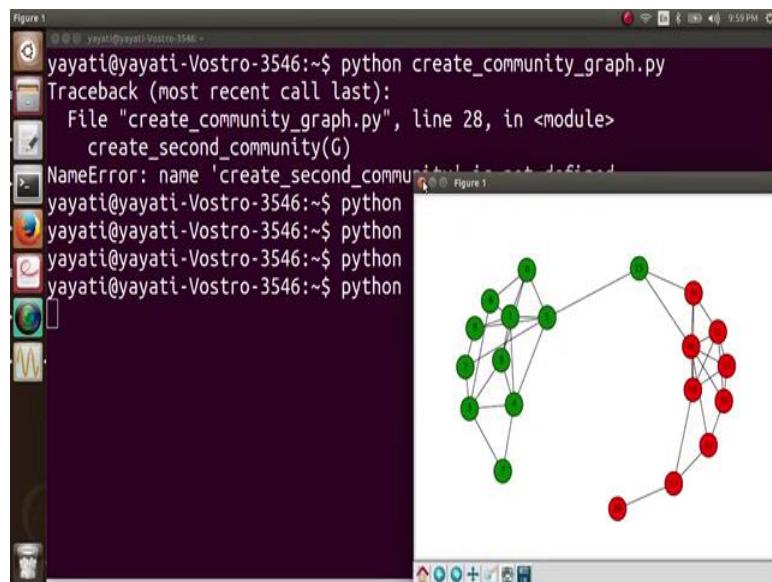
I have adopted the idea and then you can see that 0 and 1 have adopted back the behavior B, but these other four nodes in this community have adopted this behavior A.

(Refer Slide Time: 09:56)



And then you see more nodes have adopted this behavior A and this one node in this community have also adopted this behavior A and then you can see that most of the nodes in this first community where the cascade started have adopted the behavior A.

(Refer Slide Time: 10:12)



And then you see, one community has completely adopted the behavior A while in other communities still everybody has adopted just the behavior B. And you can keep running it again and again and you can see that this behavior you not passing on to the other

community. It remains stuck to this first community only. We can see that this cascade it keeps strapped in the first community itself.

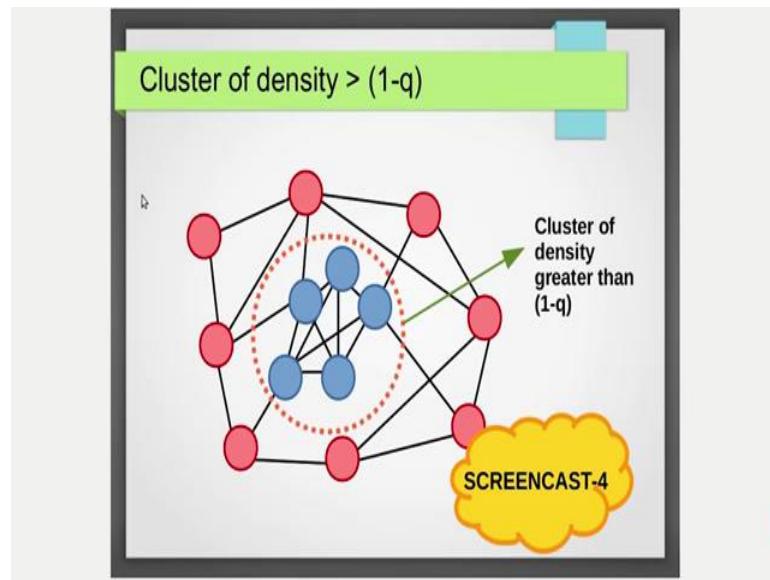
So, this graph is not going to change, it will just keep running for hundred iterations like this and your cascade has actually come to an end. So, if you remember during the lecture we discussed that there is this, there are three stability conditions: first is everybody in this network adopts A, second is everybody in this network adopts B and third is some nodes in this network adopt A while others adopt B. So, you can see that is the third condition here.

Some nodes in this network have adopted A, other nodes in this network have adopted B and then we have looked at what should the network look like for the third condition to happen and that was the presence of communities and that is exactly what is happening here. Here are two communities one community has adopted one behavior and the second community has adopted a different behavior.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

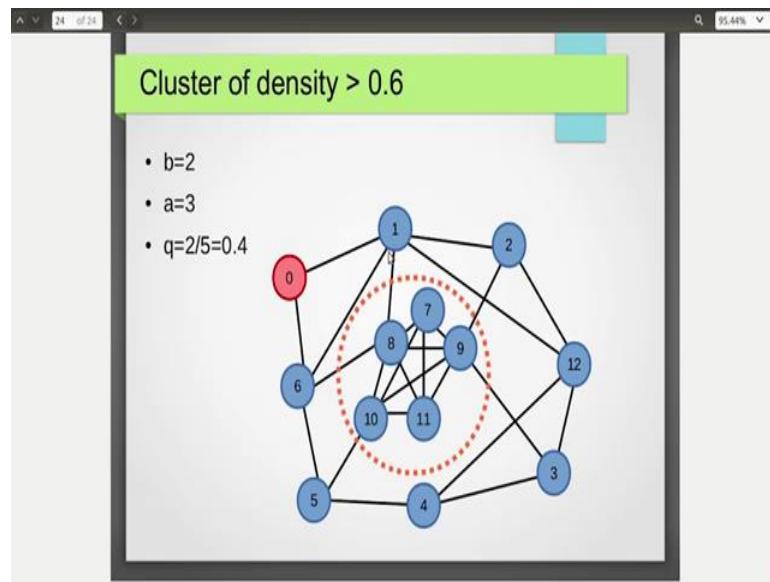
**Cascading Behavior in Networks**  
**Lecture - 100**  
**Coding the Fourth Big Idea – Cascades and Clusters**

(Refer Slide Time: 00:05)



Let us now move on to the last programming screen cast where we are going to validate this theorem experimentally that if there is a cluster of density greater than  $1 - q$  in your network, then you can never achieve a complete cascade. What we are going to do for this? We need certain parameters.

(Refer Slide Time: 00:25)



So, these are the parameters which we take. Let the payoff associated with your behavior  $b$  which was your initial behavior on the network is 2 that the payoff associated with a new behavior which is coming up in your network  $b$  3. So, the value of  $q = 2/5$ .

So, you remember how did we calculate this  $q$ ? So, you can go back and watch it the value of  $q$  is as we calculated was  $b/(a + b)$  which is  $2/5$  here which is 0.4. What does it mean? It means that in this network where initially everybody has adopted this behavior  $b$  and now some people have adopted this behavior  $a$ . Now every node will look at their neighbors and if 40 percent of their neighbors have adopted the behavior  $a$  that node will also adopt the behavior  $a$ . And then we are going to work with a graph. So, this is the graph we are going to work with.

So, as you can see that in this network, here is a cluster and the density of this cluster is greater than 0.6. So, you can also validate it density greater than 0.6 means if you look at every node in the cluster whether it be node 7, 8, 9, 10 or 11; 60 percent of their neighbors should be inside this cluster only. So, for example, this node 8 here has how many neighbors? 1, 2, 3, 4, 5, 6; it has 6 neighbors. And out of the 6 neighbors, 4 belongs inside this community.

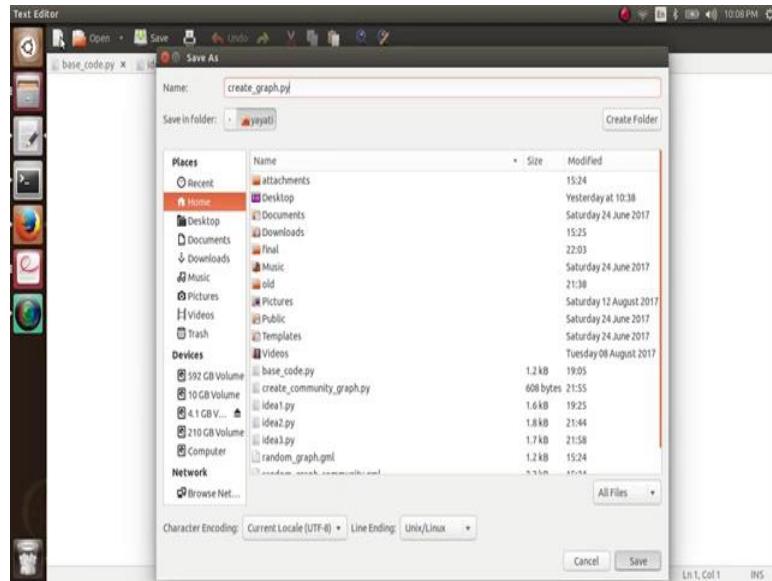
So, 4 by 6 is 2 by 3 and inside this it is own community which means greater than 60 percent. Node 7 has all its nodes inside the same cluster, 11 has all its nodes neighbors

inside the same cluster and nodes 9 and 10 also have 60 percent of their neighbors inside this cluster.

So, we are going to take this network and after taking this network, we will you experimentally validate that no matter from which ever nodes you start your cascade outside this cluster. So, your cascade should start outside this cluster will start our cascade from some set of nodes which lie outside this cluster. And then we will see that you can never achieve a complete cascade. Your cascade will not be able to impact any of these nodes number from 7 to 11. So, we now code it and see.

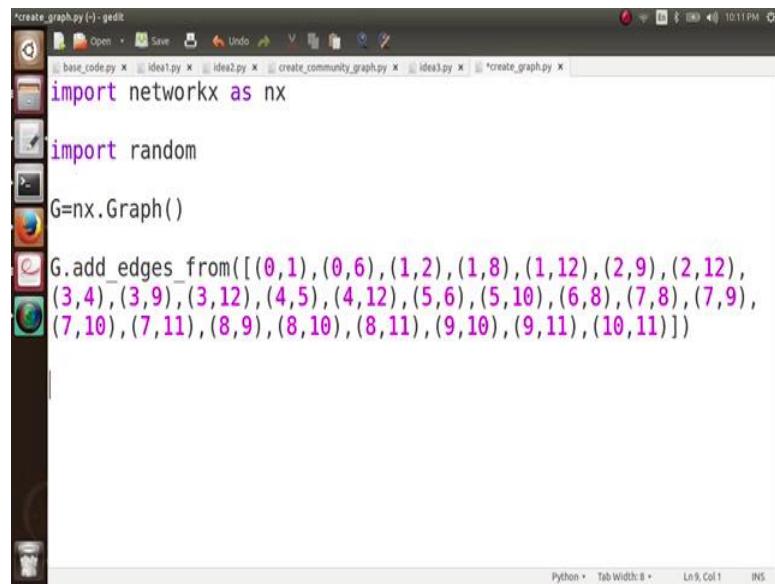
So, for coding we are going to make this graph manually. We want to work with this graph because here we clearly know that we have this cluster. This is the set this set as a density greater than 0.6.

(Refer Slide Time: 03:06)



So, let us make a graph first. So, I will create a file here let say create underscore graph dot py.

(Refer Slide Time: 03:19)

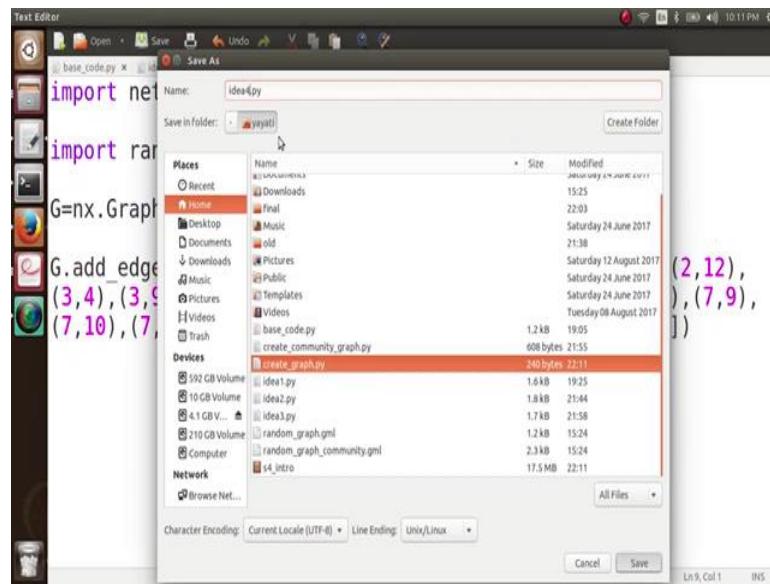


```
create_graph.py (-) - gedit
base_code.py x idea1.py x idea2.py x create_community_graph.py x idea3.py x create_graph.py x
import networkx as nx
import random
G=nx.Graph()
G.add_edges_from([(0,1),(0,6),(1,2),(1,8),(1,12),(2,9),(2,12),
(3,4),(3,9),(3,12),(4,5),(4,12),(5,6),(5,10),(6,8),(7,8),(7,9),
(7,10),(7,11),(8,9),(8,10),(8,11),(9,10),(9,11),(10,11)])
```

And you can whatever edges I am going to type here, you can actually go back and match it with the figure which we have seen with the graph which we have seen previously. So, import networkx as nx and then import random and then G = nx.graph and then I am going to put here edges and these edges are going to be absolutely the edges which where there in the figure.

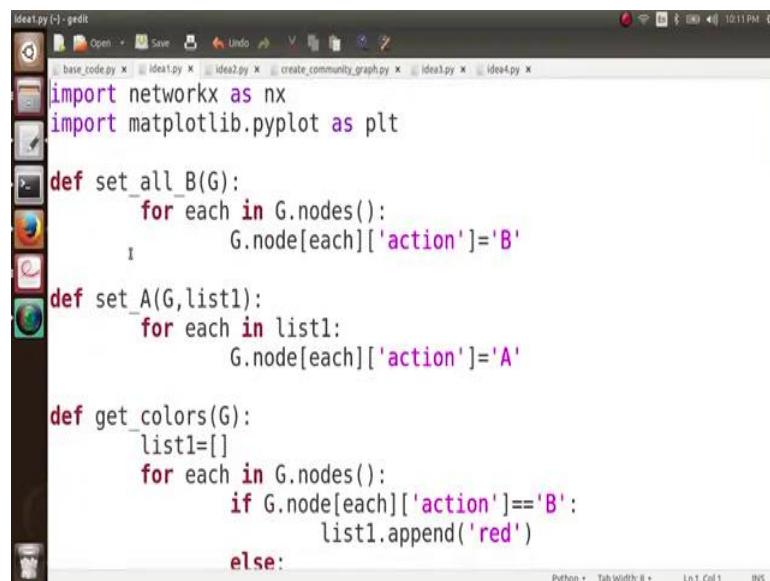
So, these are the edges which we have added in this graph you can go back and match it with the figure which we have been previously. So, here we are getting this graph, we have this graph. Here only in this file only we are actually going to implement a third idea.

(Refer Slide Time: 04:02)



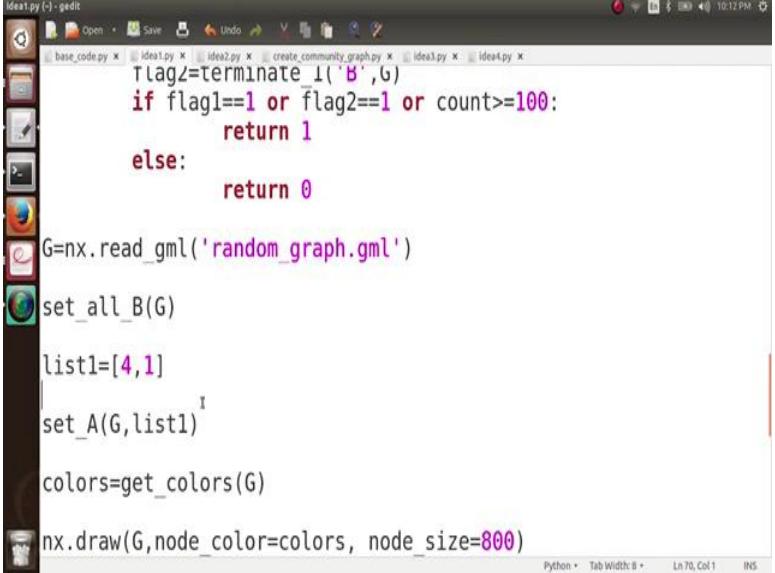
So, let us actually save it as from create graph instead we should save it as let us say idea4.py. We are going to a forth idea here. So, we have the graph which we wanted.

(Refer Slide Time: 04:22)



And next what we want to do is our all the functions are actually going to be the same which were here. So, I might consider copy pasting these functions here. So, just give me 1 minute. These are all the basic functions which we have implemented previously ok.

(Refer Slide Time: 04:55)



```
idea4.py (-) - gedit
base_code.py  idea1.py  idea2.py  create_community_graph.py  idea3.py  idea4.py
Tflag2=terminate_1('B',6)
if flag1==1 or flag2==1 or count>=100:
    return 1
else:
    return 0

G=nx.read_gml('random_graph.gml')

set_all_B(G)

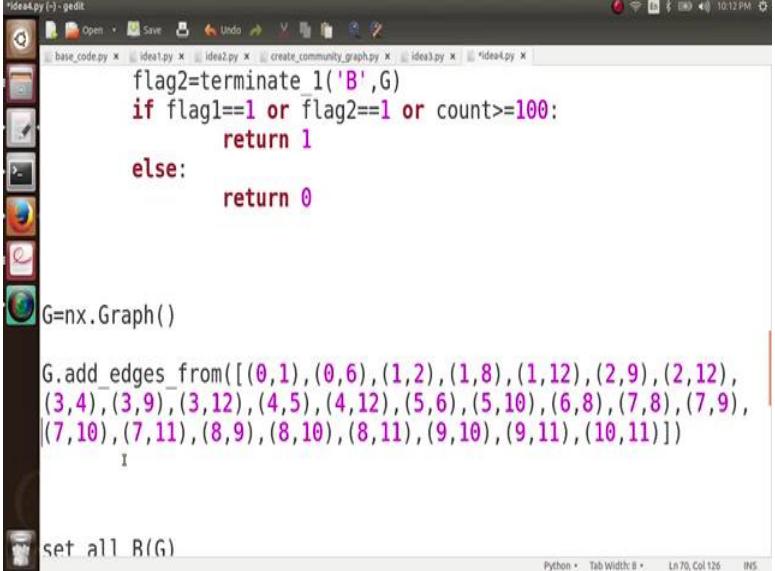
list1=[4,1]
set_A(G,list1)

colors=get_colors(G)

nx.draw(G,node_color=colors, node_size=800)
Python Tab Width: 8 Ln 70, Col 1 INS
```

And what else do we need from this first code is let us take this code also. It is going to help us, and we have this code here.

(Refer Slide Time: 05:11)



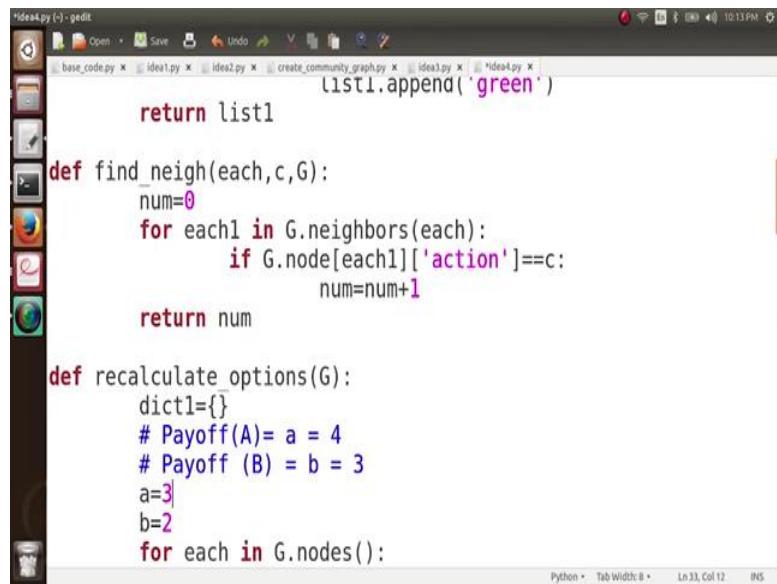
```
idea4.py (-) - gedit
base_code.py  idea1.py  idea2.py  create_community_graph.py  idea3.py  idea4.py
flag2=terminate_1('B',6)
if flag1==1 or flag2==1 or count>=100:
    return 1
else:
    return 0

G=nx.Graph()

G.add_edges_from([(0,1),(0,6),(1,2),(1,8),(1,12),(2,9),(2,12),
(3,4),(3,9),(3,12),(4,5),(4,12),(5,6),(5,10),(6,8),(7,8),(7,9),
(7,10),(7,11),(8,9),(8,10),(8,11),(9,10),(9,11),(10,11)])
set_all_B(G)
Python Tab Width: 8 Ln 70, Col 126 INS
```

So, here we have this graph and we have all the necessary functions here. What we have to do now is as we decided we know that the payoff associated with this action b.

(Refer Slide Time: 05:32)



```
ideas4py (~) - gedit
base_code.py  idea1.py  idea2.py  create_community_graph.py  ideas.py  *idea4py.py
list1.append('green')
return list1

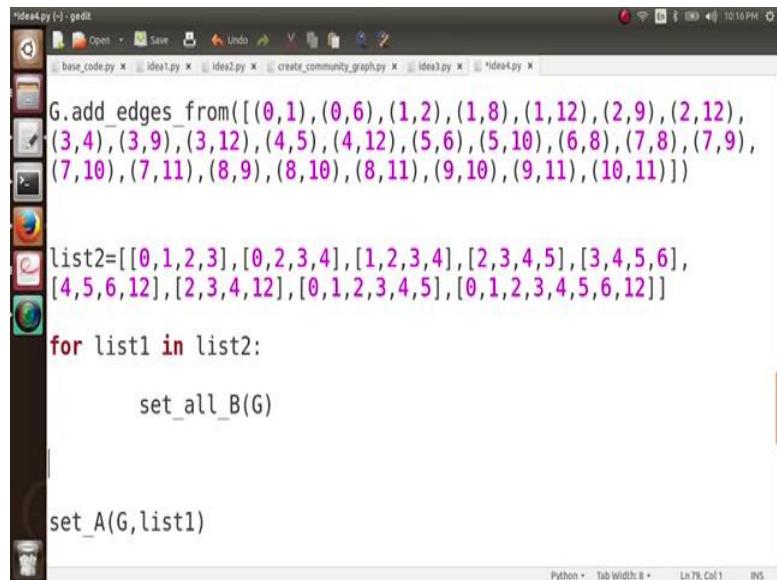
def find_neigh(each,c,G):
    num=0
    for each1 in G.neighbors(each):
        if G.node[each1]['action']==c:
            num=num+1
    return num

def recalculate_options(G):
    dict1={}
    # Payoff(A)= a = 4
    # Payoff (B) = b = 3
    a=3
    b=2
    for each in G.nodes():
        if G.node[each]['action'] == 'A':
            dict1[each] = a
        else:
            dict1[each] = b
    return dict1

Python  Tab Width: 8  Ln 33, Col 12  INS
```

Here is going to be 2 and a here is going to be 3.

(Refer Slide Time: 05:41)



```
ideas4py (~) - gedit
base_code.py  idea1.py  idea2.py  create_community_graph.py  ideas.py  *idea4py.py
G.add.edges_from([(0,1),(0,6),(1,2),(1,8),(1,12),(2,9),(2,12),
(3,4),(3,9),(3,12),(4,5),(4,12),(5,6),(5,10),(6,8),(7,8),(7,9),
(7,10),(7,11),(8,9),(8,10),(8,11),(9,10),(9,11),(10,11)])

list2=[[0,1,2,3],[0,2,3,4],[1,2,3,4],[2,3,4,5],[3,4,5,6],
[4,5,6,12],[2,3,4,12],[0,1,2,3,4,5],[0,1,2,3,4,5,6,12]]

for list1 in list2:
    set_all_B(G)

set_A(G,list1)

Python  Tab Width: 8  Ln 79, Col 1  INS
```

And then so, there were how many nodes in the network? If you remember there was 0 to 12 nodes. So, total 13 nodes in the network and out of this 13 networks denotes inside our cluster were nodes from 7 to 11. So, we can start our cascade from any node from any set of nodes except the node 7 to 11.

What I am going to do is there are actually a lot of possible sets from this list from 0 to 6 and then node number 12. So, how many nodes? 0 to 6; 7 plus 8; so, there are 8 nodes.

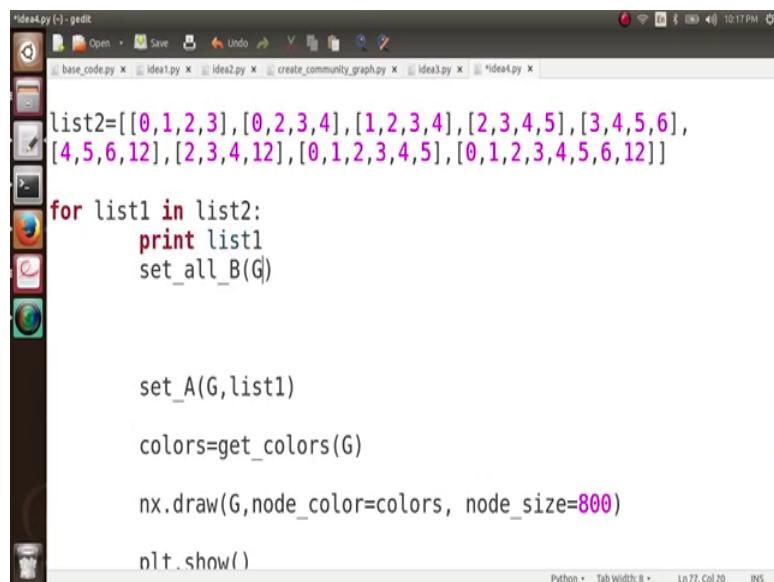
So, there are how many possible sets on in 8 nodes is  $2^8$ . So, I am not actually going to take all the 2 raise to the  $2^8$  sets and show you the result. I will take some sets here and show you the result. So, I create a list here let us say list 2 and what is this list 2? This list 2 is the list of lists. So, I will having some lists here.

So, this first list here is the first my set of initial adopter for the first experiment. This list is the set of initial adopters for the second experiment and I show you that for all these sets. So, there are different initial adopters. So, for all of these sets, your cascade is not complete. It is unable to hit any of the node inside the cluster.

So, let us make here list let say that the first list is 0, 1, 2, 3. So, I can take any number of elements here outs any number of elements here except the elements from 7 to 11 which are inside my cluster. And let us take 0, 2, 3, 4 and let us also take sorry 1, 2, 3, and 4 and 2, 3, 4, 5. Let us take many sets 3, 4, 5, 6 4, 5, 6, 12. Let say 2, 3, 4, 12. Let us take many nodes, let us take 6 node 0, 1, 2, 3, 4, 5. It seems that if you start your cascade from so many nodes from 6 node the cascade should be complete. And rather let us take extreme case, let us take all the nodes except the nodes in the cluster 0, 1, 2, 3, 4, 5, 6 and 12.

So, except all the nodes in the cluster, we are talking about I am taking all the nodes here and we see that even in this case your cascade is unable to become complete. So, now, what we are going to do for list 1 in list 2, we are going to repeat this experiment ok.

(Refer Slide Time: 09:30)



```
idea4.py -eedit
base_code.py X idea1.py X idea2.py X create_community_graph.py X idea3.py X *idea4.py X
list2=[[0,1,2,3],[0,2,3,4],[1,2,3,4],[2,3,4,5],[3,4,5,6],
[4,5,6,12],[2,3,4,12],[0,1,2,3,4,5],[0,1,2,3,4,5,6,12]]

for list1 in list2:
    print list1
    set_all_B(G)

    set_A(G,list1)
    colors=get_colors(G)
    nx.draw(G,node_color=colors, node_size=800)
    plt.show()

Python Tab Width: 8 Ln 77, Col 20 INS
```

So, for every possible set of initial adopters here, what will be seeing is that list and then will be seeing the final graph and then will be seen whether the cascade is complete or not. So, there is nothing much we just need to execute it and see what is happening here.

(Refer Slide Time: 09:48)



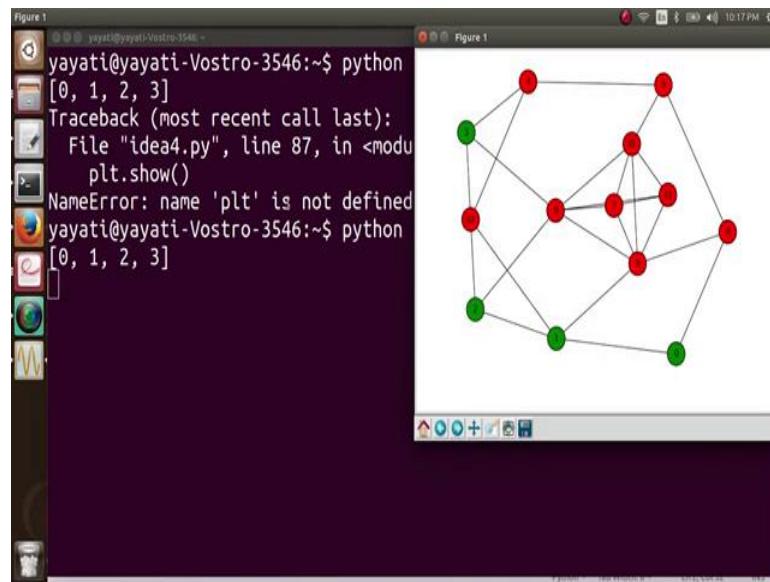
python idea4.py.

(Refer Slide Time: 10:02)

A screenshot of a code editor window titled "idea4.py - gedit". The file contains Python code for a social network simulation. It includes imports for networkx, matplotlib.pyplot, and random. Three functions are defined: "set\_all\_B" which sets all nodes to action 'B', "set\_A" which sets nodes in list1 to action 'A', and "get\_colors" which returns a list of 'red' nodes. The code uses a for loop to iterate over nodes and an if statement to check their action. The editor has multiple tabs open in the background.

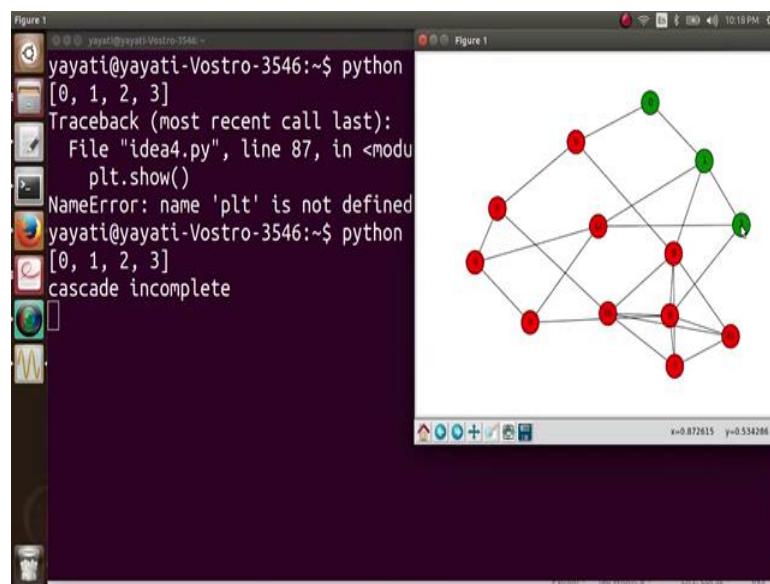
So we have to import matplotlib.pyplot as plt.

(Refer Slide Time: 10:18)



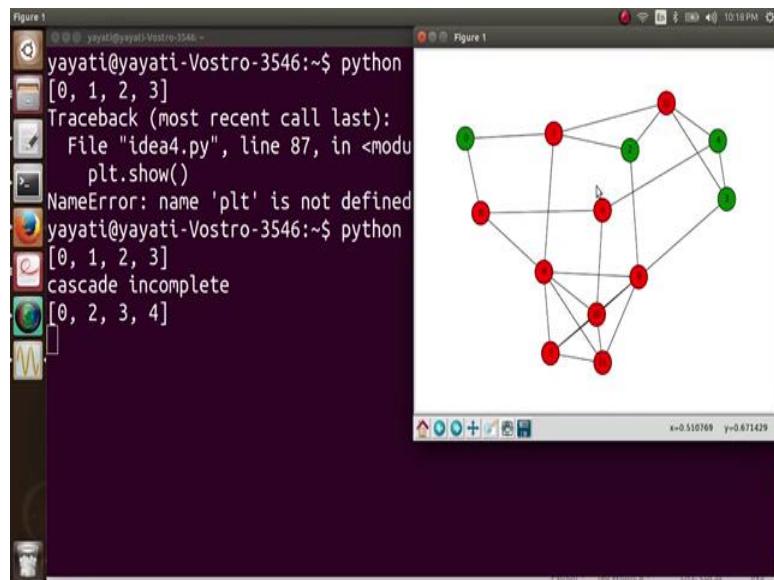
So, here is a case when our initial adopter are 0, 1, 2, 3. So, this is the cluster we were talking about. So, this is a cluster having a density greater than 60 percent and you will see that in none of the cases this cascade will be able to touch this cluster.

(Refer Slide Time: 10:37)



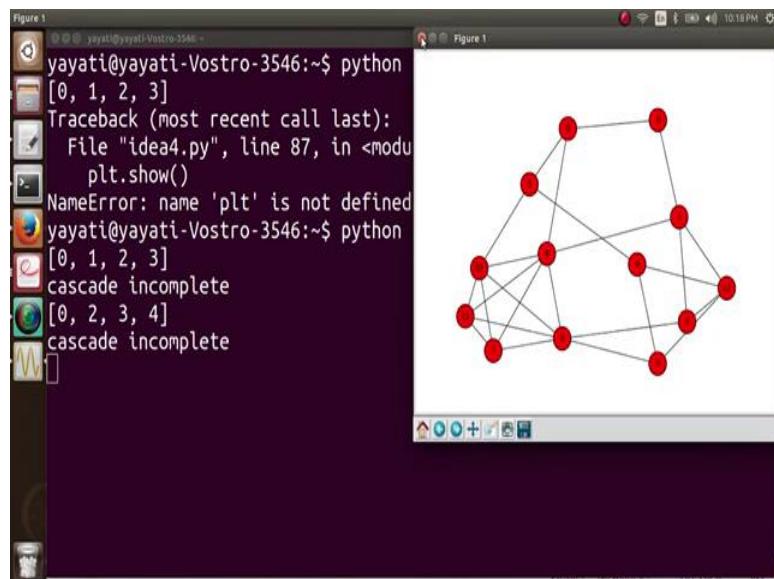
So, in this case when our cascade starts from 0, 1, 2, 3, this is the final cascade 0, 1, 2 and it remains incomplete, rather outside cluster also there are a number of nodes which remain unaffected.

(Refer Slide Time: 10:47)



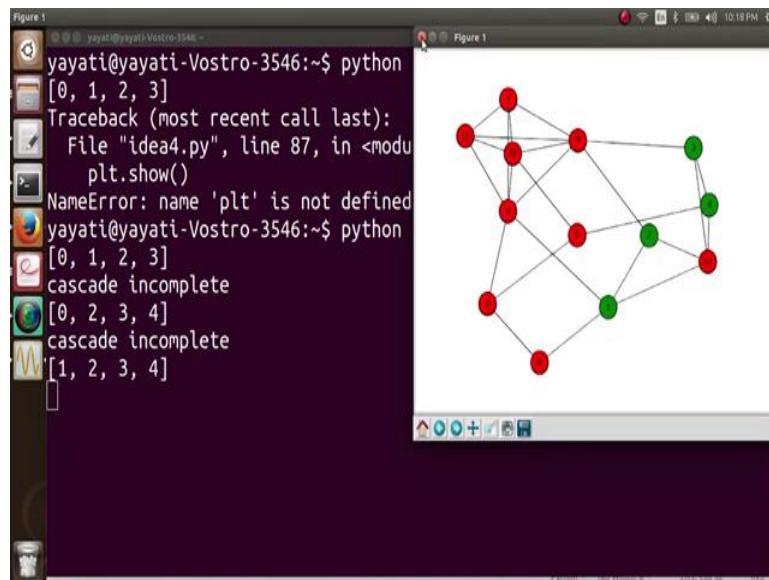
And then we start from 0, 2, 3, 4.

(Refer Slide Time: 10:50)



And then, you see that cascade is incomplete rather the cascade dies away with time.

(Refer Slide Time: 10:57)



And let say we start with 1, 2, 3, 4. And then again the cascade dies away with time 2, 3, 4, 5 dies away 3, 4, 5, 6 and sub having some node infected, but still it is unable to touch any of the nodes in our cluster from 7 to 11. And then let us start from 4, 5, 6, 12 when we start from 4, 5, 6, 12. So, this is the final graph and we see that the cascade remains incomplete then, we start from 2, 3, 4 and 12 and then many nodes are inserted, but again and cluster remains unaffected and then we start from almost 6 node seems like everybody should be in affected at the end, but you can still see that the cluster remains unaffected. And then this was the extreme case we were talking about.

So, we start. So, these many nodes almost 1, 2, 3, 4, 5, 6, 7, 8 nodes in the graph, I have adopted the behavior a. Even then our cascade remains incomplete and our cluster remains intact the cascade is unable to enter this cluster. So, we have validated that if there is a cluster of density greater than  $1 - q$  in the network of the cascade can never become complete.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Link Analysis (Continued)**  
**Lecture - 101**  
**Introduction to Hubs and Authorities (A Story)**

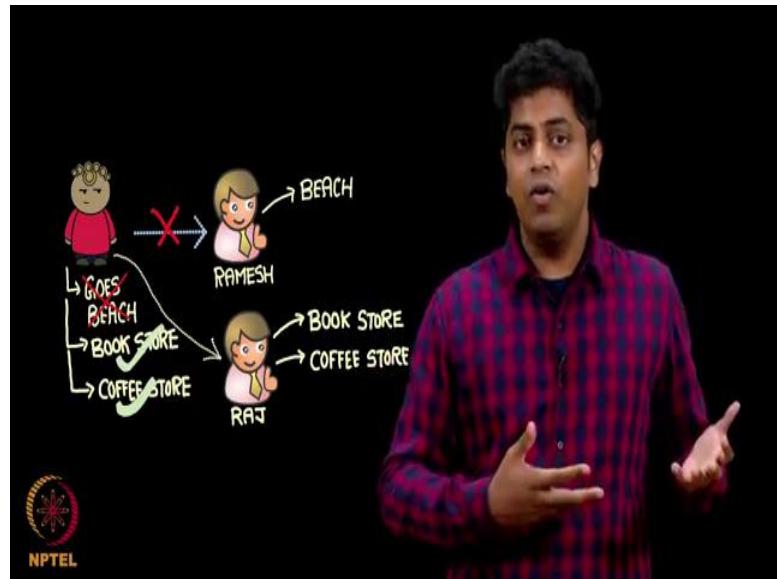
It has been customary that we start the chapter with an example. Now telling you everything about the chapter, we are going to follow the tradition. In this chapter too, I am going to start with an example and slowly after the example I will ask you few questions and then start telling you what this chapter is all about.

(Refer Slide Time: 00:26)



So, assume I went to a new city. I have no clue what is the city all about and if I type in what are the best things to, what are the nice tourist spots in this place. There will be some ten hits online Wikipedia or some tourist information and things like that. That may not really interest me. We need it is very personality dependent. So, what do we generally do? We ask the good friend of ours.

(Refer Slide Time: 00:56)



Let me assume I ask this friend I go to the city in the first place and then I ask I ring this good friend of mine who stays in the same city his name is let us say Ramesh. I ask Ramesh. Ramesh, can you please tell me a nice place to visit. I am done with my I came here for a business trip I am done with my business whatever was the official work, now I am free to go around the city and then see the city.

Tell me what all I can do in your city. Ramesh will say my city has beautiful beach which is very close by to where you are staying. You should go and explore it I ask him where exactly it is he tells me the place and exactly route and I go there. So, Ramesh suggested that I go to this beach on the seaside. I see the place a lot of eat outs nice things to shop at things like that. But now the person that I am I do not really enjoy let us say I did not enjoy going to this seaside and eating something which is considered sea food and things like that.

I did not enjoyed I come back, then next day I have more time to explore the city. I will probably not ask Ramesh, why? I will infer that Ramesh recommendation that he gave he does not suit my personality type. If I ask him once again, he will give a similar suggestion. So, I will not now talk to Ramesh I will ring another friend of mine by name Raj, I call up Raj and Raj says Sudarshan, there is a beautiful place near by a couple of kilometers from where you stay it is a beautiful book stores you can sit there read

whatever book you want and buy a book if you wish that is a nice place with the great ambience.

You must visit there I say fine let us let me try over Raj's suggestion. So, Ramesh just said the beach I did not like I did not ask Ramesh anything further. Now Raj comes and he suggests go and see this book centre. I got the book centre at it is plane awesome. I really loved the book centre. It is a great experience to go through the popular science books and the fiction side in the general reading side, great stationary stuff there a something that really a gives me a high. I thoroughly enjoyed visiting this place I comeback and I message Raj I tell him; dude thank you very much. This is one of the best things I can ever do in a new city.

I was extremely happy. What do I do? What just happened, I asked Ramesh. He said something, I did not like. So, I did not ask Ramesh. And then I rang Raj, Raj said go to a book centre I loved going to this book centre and then I now like Raj's recommendation. For the next day, I again ring him ring, whom? Raj and ask him to come on give me another suggestion as good as what you did yesterday.

And raj says go to this coffee store which is famous. Go with a notebook and pen and you can we can scribe something there are or write something there. It is a nice place; we can even read a magazine or two sipping the cities best possible freshly brewed coffee.

Fine, I go there and again I enjoy the ambience. It is a fantastic place; I drink my favorite espresso and I am on caffeine high I start writing on what is the next NPTEL course I must offer and I am all very happy. I come back and I now have high regards for Raj right.

(Refer Slide Time: 04:55).



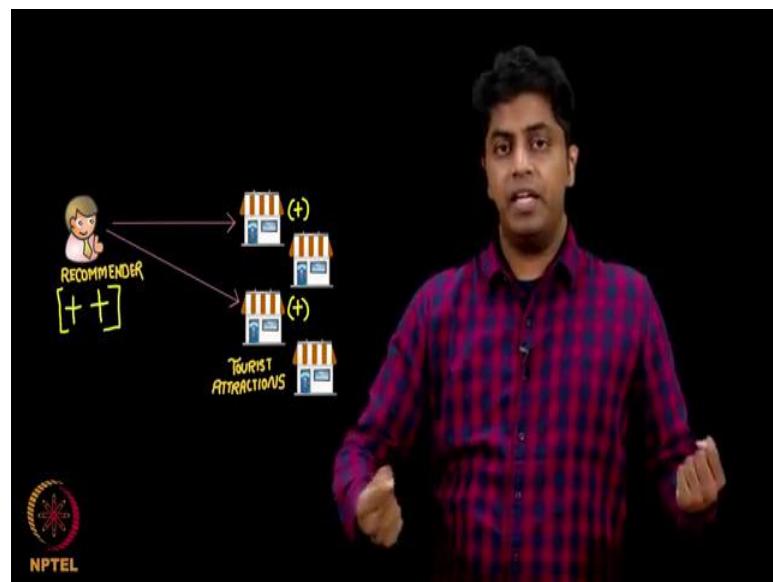
Let us see what just happened. Here is a person who recommends me a place is the place is good, you give some star rating to the place not only to the place, you also give some rating to the person who recommends this place. The place is good, the person gets more points. The person has more points whatever recommendation he says will be valued by me. Let us think of this slowly, patiently and observe what just happened.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Link Analysis (Continued)**  
**Lecture - 102**  
**Principle of Repeated Improvement (A story)**

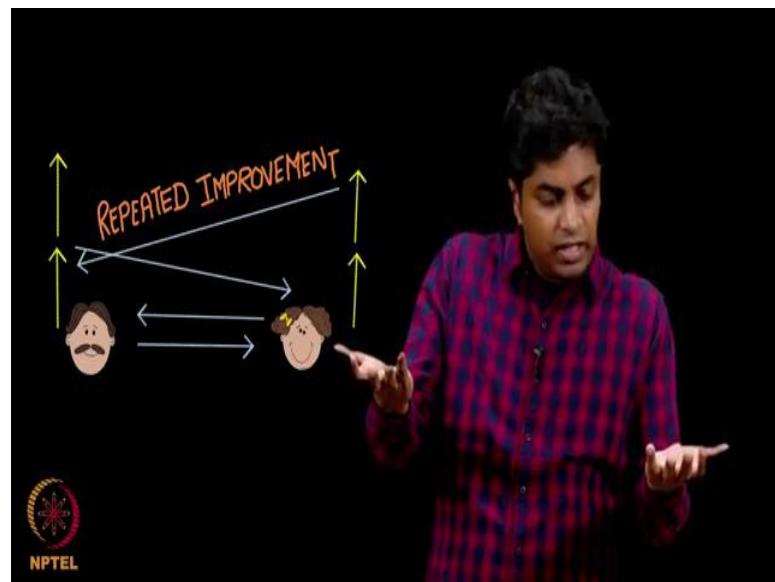
Let us try to go beyond the simple example; that I gave you what role did Raj take here?

(Refer Slide Time: 00:10)



He was a recommender, what are the places here? The places to visit, tourist attraction right, Raj; was a pointer, to a nice place. Now I will trust Raj, if he points me to a nice place. If the place is nice, I will more trust Raj. When I all the more trust raj, his points increase, correct; he becomes more trustworthy, he points to a place, the place is good, the place gets better rating ok. Do you see; what exactly is happening here?

(Refer Slide Time: 00:57)



I give you another example; assume you and your spouse. If you treat your spouse well. If you are good, you will treat your spouse well. Let us say you are in the best of your moods; you are mentally very fit; very fit. So, or you are physically, and you have all comforts of your life, you are good; your spouse is good. If your spouse is good, you get support, you become better, you become better, your spouse gets good support, and she becomes he or she becomes better so on and so forth, right; this is called the concept of repeated improvement. Because of, the um mother; the daughter does really well in her life?

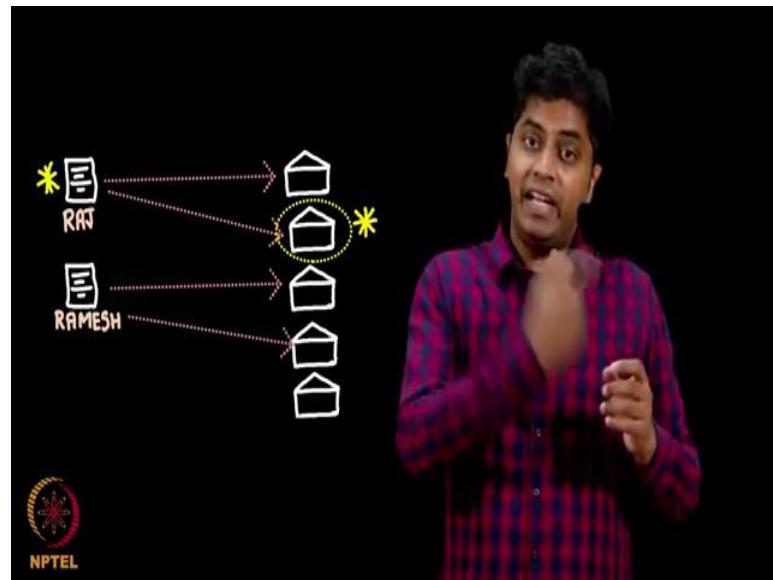
Because the daughter; mother is very happy she does better, she does better; she does better; she does better, he or she does better, he or she does better this goes on improving right. This is true with, happiness is a nice quote in English which says, divide your sorrow, multiply your; multiply your happiness and divide your sorrow, which means, if you are sad you probably should divide it to your friends, everyone should take an equal share so, that the, an amount of sorrow per person is less.

(Refer Slide Time: 02:14)



If it is happiness, you should probably multiply it. the more you give happiness, the many folds it becomes. So, the point that I am trying to make here is happiness yields happiness and this happiness yields more happiness this side and so on and so forth ok, all right. So, this is very true with Raj and his recommendations.

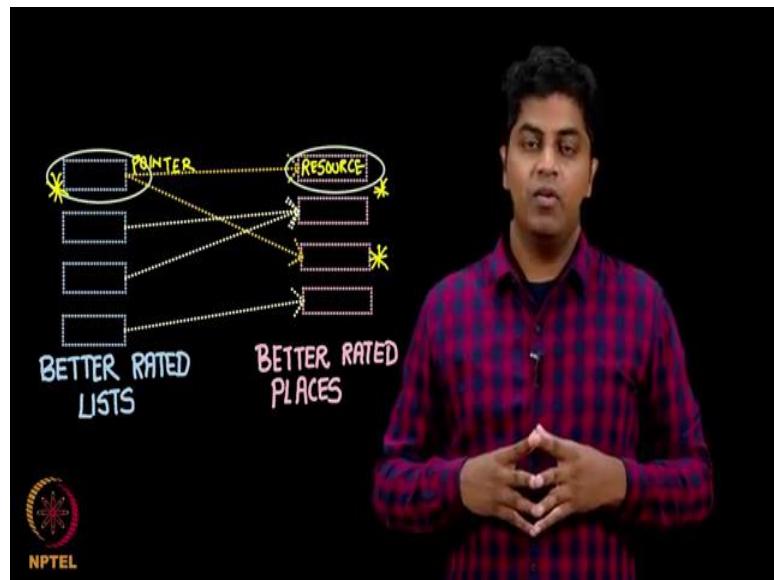
(Refer Slide Time: 02:32)



Raj role is what is called, a list, a list of recommendations. Assume there are plenty many such lists available for tourists to come to the city, Raj is basically a bold; he says go to this place, this place, this place, this place.

Ramesh is another bold; basically it is a sheet of paper, which says go to this place, this place, this place, this place. And people come read this list, and go to the place. If the place is good they give some rating for the place, and come back and give some rating for the person, for the list right.

(Refer Slide Time: 03:12)



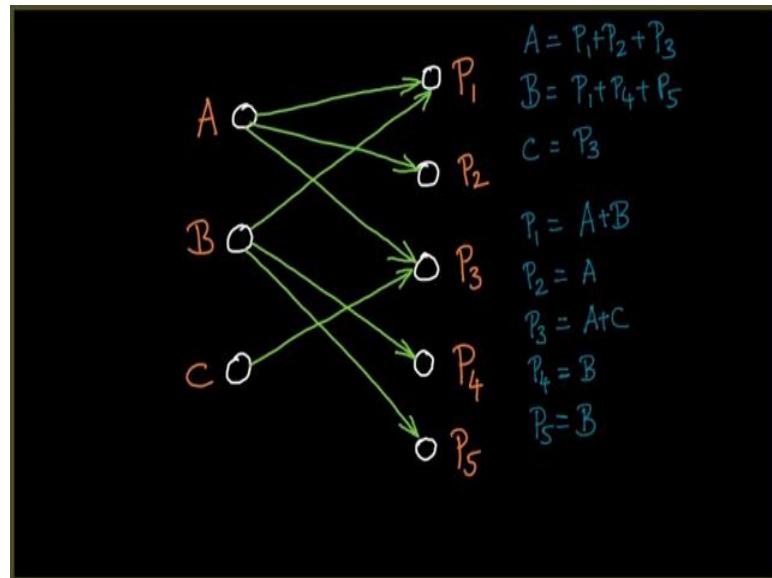
Now, better rated lists; should point to better rated places, and better rated places should be pointed by better rated lists, correct very intuitive.

If you did not get the intuition, you do not worry, as I proceed with this example, you will get good intuition. So, all I am trying to say here, is that; the pointer to a resource, and the resource itself, pointer and the resource. If the resource is good, the pointer get some credit, if the pointer has good credit, whatever he points to; also is a good resource, and resource gets good credit to; is all I am that I am saying. What has this to do; with our subject, what has this to do with let us say networks.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Link Analysis (Continued)**  
**Lecture - 103**  
**Principle of Repeated Improvement (An example)**

(Refer Slide Time: 00:05)



Let me consider 3 people on the left and five places on the right: place 1, place 2 and so on. And there on the left you have people A B C, recommending, these five places to you. Let us say A recommends P 1, A also recommends P 2 so on A recommends P 3 so on and so forth, right assume this is the given graph.

So, what do you think is A 1 here? A 1 will be P 1 plus P 2 plus P 3 as I was saying, the principle of repeated improvement is what this is called as, I will assign the score P 1 + P 2 + P 3 to A. I will start P 1, P 2, P 3, P 4, P 5, A B and C with 1 1 1 point. So, I go and doing this; B is so much, C is so much; may be you may want to pause, and take a look at how this is be it true? I am writing this now so on, so on, so on and that is it.

So, what I am trying to do right now is I (Refer Time: 01:16) am going to open an excel sheet, spreadsheet rather any spreadsheet; we will do. I am going to open a Google spreadsheet, and I am going to try these values there; and I am going to see, whether it

will converge or not. It will help, if you can make a note of this graph and also this, A B C, P 1, P 2, P 3, P 4, P 5 here.

So that, you know what are the formulas which you can try punching and on the Google search.

(Refer Slide Time: 01:42)

A	B	C	P1	P2	P3	P4	P5				
1	1	1	1	1	1	1	1				
2	1	1	1	1	1	1	1				
3	3	3	2	1	2	1	1				
4	0.4285714286	0.4285714286	0.4285714286	0.2857142857	0.1428571429	0.2857142857	0.1428571429	0.1428571429			
5	0.5714285714	0.5714285714	0.2857142857	0.8571428571	0.4285714286	0.5714285714	0.4285714286	0.4285714286			
6	0.4545454545	0.3536363636	0.1818181818	0.3157947373	0.1579473668	0.2105263158	0.1579473668	0.1579473668			
7	0.684105263	0.6315786474	0.2105263158	0.8181818182	0.4545454545	0.6363636364	0.3636363636	0.3636363636			
8	0.4482758621	0.4137931034	0.1379310345	0.3103446276	0.1724173731	0.2417373103	0.1379310345	0.1379310345			
9	0.724137931	0.5862069666	0.2413793103	0.8620689655	0.4492758621	0.5862068966	0.4137931034	0.4137931034			
10	0.4666666667	0.3777777778	0.1555555556	0.3164556962	0.164556962	0.2151698734	0.1516987342	0.1516987342			
11	0.6962025316	0.6202531646	0.2151698734	0.8444444444	0.4606666667	0.6222222222	0.3777777778	0.3777777778			
12	0.4545454545	0.4049586777	0.1404958678	0.310495868	0.17355719	0.2310495867	0.1404958678	0.1404958678			
13	0.7190082645	0.5950413223	0.2314049587	0.8595041322	0.4545454545	0.5950413223	0.4049586777	0.4049586777			
14	0.4652406417	0.385026738	0.1497326203	0.3161094225	0.1671732523	0.2188449864	0.1489361703	0.1489361702			
15	0.7021276599	0.6139017829	0.2188449848	0.8562673797	0.4652406417	0.614973262	0.385026738	0.385026738			
16	0.4574257428	0.4	0.1425742374	0.3148514851	0.1722772277	0.2277227723	0.1425742574	0.1425742574			
17	0.7148514851	0.6	0.2277227723	0.8574257428	0.4574257426	0.6	0.4	0.4			
18	0.4634146341	0.388962054	0.1476251605	0.3158278629	0.1664949132	0.221006564	0.1473377097	0.1473377097			
19	0.7053245605	0.610503282	0.2210065646	0.8523748395	0.4634146341	0.6110397946	0.388962054	0.388962054			
20	0.4560463692	0.397247271	0.1436063598	0.3151400995	0.1713336497	0.2299136213	0.1438063598	0.1438063598			
21	0.7123872805	0.602752729	0.2299136213	0.8561936492	0.4589463692	0.602752729	0.397247271	0.397247271			
22	0.4657798673	0.191107741	0.1466084848	0.1146608479	0.1466084848	0.999999999	0.166666669	0.166666669			

I have now come to the Google spreadsheet; as you can see, you have, A B C D, A B C, P 1, P 2, P 3, P 4, P 5 here, let me try making it bold and the underline it, perfect. And then, I start with the value; A is 1, B is 1, C is 1, P 1, P 2, P 3, P 4, P 5 are all 1s.

And the value of a as you can see, is P 1 + P 2, because D 2, E 2 and F 2, P 1 + P 2 + P 3. You can see, I have punched in all the values of A B and C. Since B is, what is B? B is P 1 I have asked it a write down the graph, I am show you have written it down. If you can cross check, you will see that; B is P 1, P 4 and P 5. So, it will be, 1 + 1 + 1, which is three, so on and so forth; C is simply you are picking. So, I go on like this, I write down everything that is this row.

And then the next row; is I, normalize it; what do I do? I take this 3, 3, 1 which is 3 + 3 is 6 + 1 is 7 and I divide it 3 by 7. The 3 is divided as you can see; the entry of A 3 is divided by the entry of A 3 + B 3 + C 3. For a change, I am using a spreadsheet, they are instances where spreadsheets easier than, programming; all the programming is a, very

powerful thing to do. Sometimes spreadsheet is a quick reference for you to compute and see, what is happening.

So, you see; that I am just trying to normalize it here. Similarly for P 1, P 2, P 3, P 4, P 5; I have normalized this, which is 2 plus 1, 3, 4, 5, 6. I am dividing everything by this sum; I will draw formula here. So, what I do is, when I keep repeating this, repeating what? These assignments, I do this, what is called, principle of repeated improvement. I add the corresponding entities, and then keep normalizing it.

Then I do it in the second iteration I get, so much. So, what is this? This will obviously be, the sum of; as you can see, D E F. Which is this one, this one and this one; this P 1 plus P 2 plus P 3. Just above it D 4, E 4, F 4 is simply this, this and this; values added together is this so on and so forth is automatic gets added. That is the power of the spreadsheet, when you pull it the automatic values are being flooded.

And then, as and always; as I told you, what I did here, similarly the same thing is repeating here I am normalizing it. Now you see the value are changing here 0.42 became 0.45. Let me do this one more iteration; so, this is what is important for us right; this value. So, let me make this, bold; bold. This is what we get after normalizing it, go back.

So, you see it sort of getting closer and closer may not be here this is going here, via this is going somewhere else; 0.14, 0.18, 0.13. So, I do not think, it is very, very in anyway close to converge it; maybe I may have to do this again let me try continuing this, maybe up to 20 right.

So, what do I see? I see that 0.45, 0.46 0.45 still not converging right. So, I will do is, I will try pulling this till 100. When I keep doing this as I keep doing this, maybe even 200 I will go until, mean around 200 iterations, and I stop it here it is flooding, yeah, its flooded.

(Refer Slide Time: 06:00)

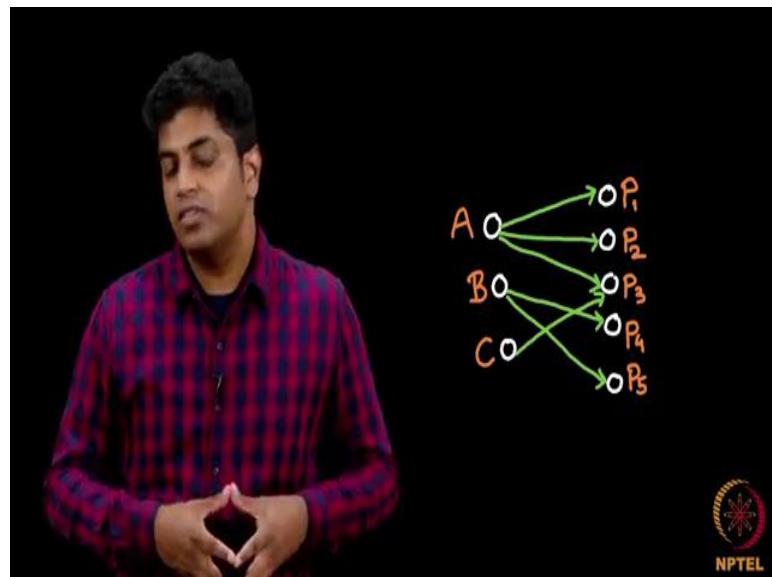
	A	B	C	D	E	F	G	H	I	J	K
190	0.4608111272	0.3938265525	0.1453623203	0.3154488009	0.1700864866	0.2237400659	0.1453623203	0.1453623203			
191	0.7092753594	0.6061734475	0.2237400659	8.8546376797	0.4608111272	0.6061734475	0.938265525	0.938265525			
192	0.4608111272	0.3938265525	0.1453623203	0.3154488009	0.1700864866	0.2237400659	0.1453623203	0.1453623203			
193	0.7092753594	0.6061734475	0.2237400659	8.8546376797	0.4608111272	0.6061734475	0.938265525	0.938265525			
194	0.4608111272	0.3938265525	0.1453623203	0.3154488009	0.1700864866	0.2237400659	0.1453623203	0.1453623203			
195	0.7092753594	0.6061734475	0.2237400659	8.8546376797	0.4608111272	0.6061734475	0.938265525	0.938265525			
196	0.4608111272	0.3938265525	0.1453623203	0.3154488009	0.1700864866	0.2237400659	0.1453623203	0.1453623203			
197	0.7092753594	0.6061734475	0.2237400659	8.8546376797	0.4608111272	0.6061734475	0.938265525	0.938265525			
198	0.4608111272	0.3938265525	0.1453623203	0.3154488009	0.1700864866	0.2237400659	0.1453623203	0.1453623203			
199	0.7092753594	0.6061734475	0.2237400659	8.8546376797	0.4608111272	0.6061734475	0.938265525	0.938265525			
200	0.4608111272	0.3938265525	0.1453623203	0.3154488009	0.1700864866	0.2237400659	0.1453623203	0.1453623203			
201											
202											
203											
204											
205											
206											
207											
208											
209											
210											
211											

Now, you see something start line, it is converging. So, the values 0.4608111272 that the node A had remains the same, right. Look at this, this and this is the same. So, this is the same. It is exactly the same, this is called the principle of iterated improvement, where in you observe that, when you keep I am come to the first line, when you assign all the nodes to have 0.1, and I and you keep doing this point assignment thinking, you will see that the values converge. We will use this in a, fourth coming lectures to explain what exactly one can infer from this.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

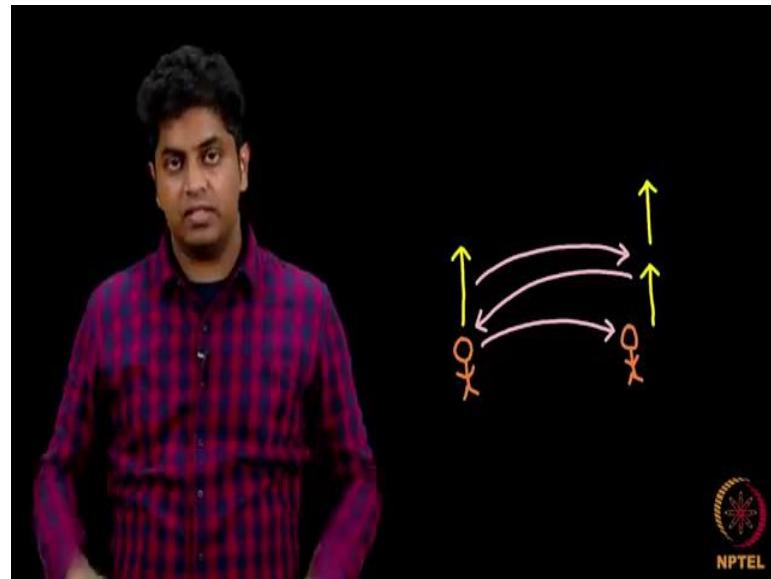
**Link Analysis (Continued)**  
**Lecture - 104**  
**Hubs and Authorities**

(Refer Slide Time: 00:08)



So, as we saw; there is A B C this side; P 1, P 2, P 3, P 4, P 5 this side, someone if a recommends a person this side, he gets a point, and A gets the point, of the person. That he is recommending to whether place that he is recommending B, gets some points based on the places that B recommends. And the places get the points based on the points that B has, correct.

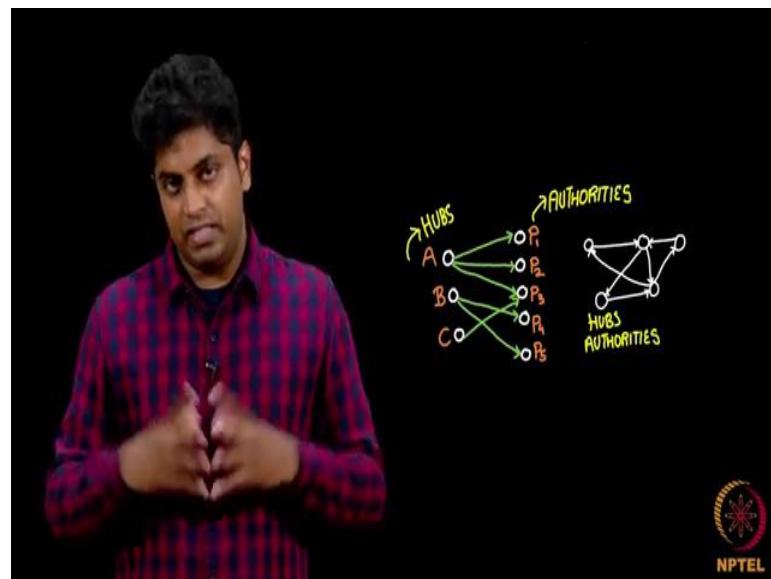
(Refer Slide Time: 00:39)



So, it sounds very weird you see, I am famous and if I say you are famous you become famous, you become famous and you say I am famous, and I become famous. And I am famous, and I say you are famous you become famous and this sort of goes on I have been using this animation from the beginning of this chapter right.

So, because you are pointing to this, he gets your reputation, you are pointing to this person. So, you get his reputation, so, on and so on and so, on. You climb up like this; correct this is called the principle of repeated improvement.

(Refer Slide Time: 01:11)

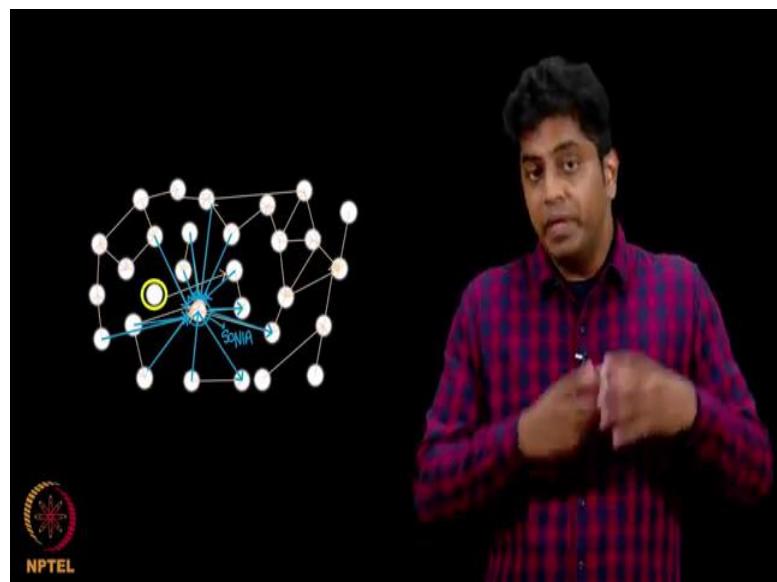


Now the question is in a complicated graph, a graph which is  $G$  of  $V$  comma  $E$ , every single node is adjacent to a few nodes. Let us first consider that, a node acts as both, the left side like A B C as well as on the right side, let us say P 1, P 2, P 3, P 4, P 5.

So, what do I mean by this? This is a recommender; this is a place. So, a recommender is let us call it by a name, the standard name for it. Let us call it hubs, hub means a place where you get information. And the place itself P 1, P 2, P 3, P 4, P 5, we will call it authorities, ok; if you want some advise, you go and ask this A B Cs. They point you to P 1, P 2, P 3, P 4, P 5, where you have your solution. In the previous example, it was people recommend my friends recommending places to visit, it can be anything for that matter.

So, given a graph  $G$ , all I am saying is a node can be given points as hubs. It can even be given points as authorities; what do I mean by this? So, let us take a nice example and try explaining this concept.

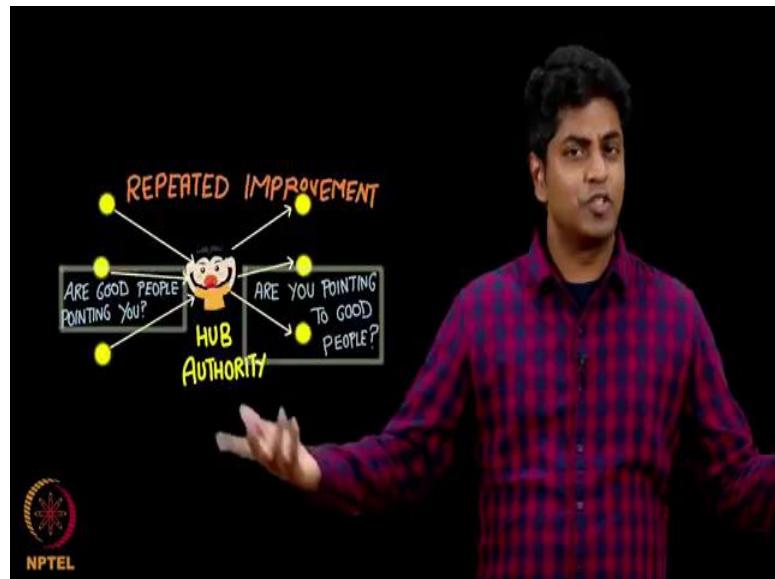
(Refer Slide Time: 02:40)



You have, let us say 30 people in a class, in a room, you go to some person randomly; and ask, can you help me with a solution to this problem? And that person says, I cannot help you, but I can point you to someone who can help. That person points to someone; and who points to whom is given as a directed graph, for example, Sonia is a node; there are 30 nodes 30 people in this room Sonia is one of them and she has arrows going to some three people.

So, she will point to these 3 people. If I ask her for solution, there are people who point to Sonia also, maybe 10 people point to Sonia ok.

(Refer Slide Time: 03:29)



Who points to you and whom do you point, are two different things are you pointing to good people? Is one way of measuring a person or good people pointing to you, or good number of people pointing to you; is another measure. So, every person has this two parameters, are you pointing to good resources: good people or good people pointing to you right.

This is exactly similar to the previous case, where people who are recommending and people who and the places or people who are advising that there is a solution this side and you go and find the solution this side, these were two different shells. Now I can combine them, I am trying to combine them, and I am give trying to you in easy piece. That given a node, you try to give both the values stored. How well the node is being pointed to is one value to the node, how well is the node pointing to is another value of the node.

So, as you saw there is a hubs score and authorities score for a node. There are two parameters that decide how important a node is. One is what does it point to, second one is what is it pointed to, who are pointing to him, him to a particular node? As you can see that definition itself is slightly recursive, you see a node is good if it is pointing to people; and people are good, if they are being pointed at by good people. And how do

you achieve this? You achieve this by repeated improvement. where are our earth is this useful?

But then, in the late 90s; it was a requirement for people to maintain pages, which had the list of all possible news material right.

(Refer Slide Time: 05:20)



And they were called the news lists; how do you trust a news list If it is good or not. If it has some good articles; point us to good articles. Now how do you know an article is good or not? You would click on it go there and see that it is good, and then you would say this news list is good. Because it points me to the right articles, just the Raj, Ramesh example I gave you right. So, this was the way in which you could rate a bunch of bolts, which would point you to nice news articles.

(Refer Slide Time: 05:53)



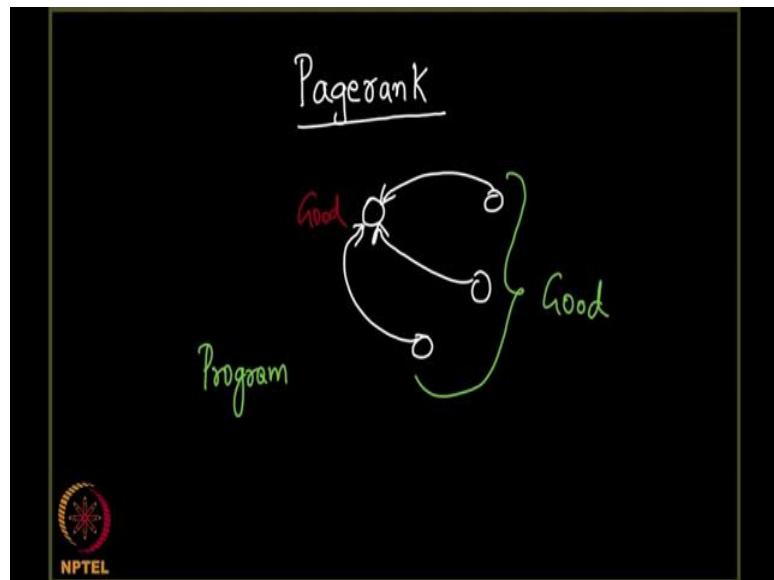
And that was a birth of this concept called hubs and authorities. Mathematically speaking, in fact, one can show that; whenever you take a graph and start with this hubs and authorities repeated improvement process. You will always converge to a point, what do I mean by converging to a point? You see this repeated improvement, results in points getting accumulated in a node, at the end after a long process.

Let us say you tried several iterations; and you will observe that the proportion of allocation, the proportion of points; that each node takes with respect to other nodes this always remains the same. For example, a node A and node B, if node A accumulates half as much as node B it will continue to accumulate; half as much as node B in the future, that is called the convergence state. So, given any graph, the hub and authorities score, as you keep doing this principle of repeated improvement it converges.

**Social Networks**  
**Prof. S.R.S Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Link Analysis (Continued)**  
**Lecture - 105**  
**PageRank Revisited – An example**

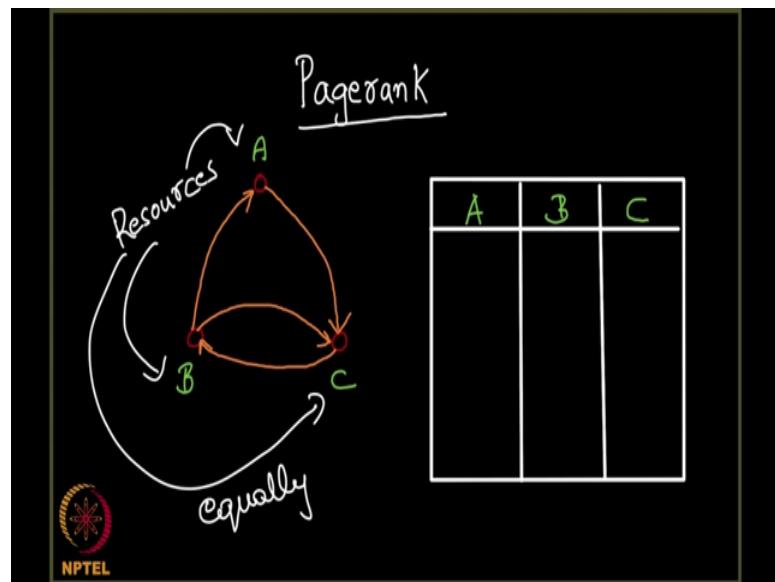
(Refer Slide Time: 00:10)



We have been discussing about pagerank all this while we will see it in a brand-new light. If you can recollect, I told you what exactly pagerank does, it is all about what you are is decided by who your friends are. If good people point to you it means that you are good. So, you are good if all these people who are pointing it to you are good.

Now, that is a very vague way to put it, can we quantify it nicely? We saw of program a whole lot of programming we are programming screen casts you saw where we exactly show you how pagerank is computed. And why exactly its done that way is what will be our focus right now in our discussion for this lecture and the forthcoming lectures ok.

(Refer Slide Time: 01:09)

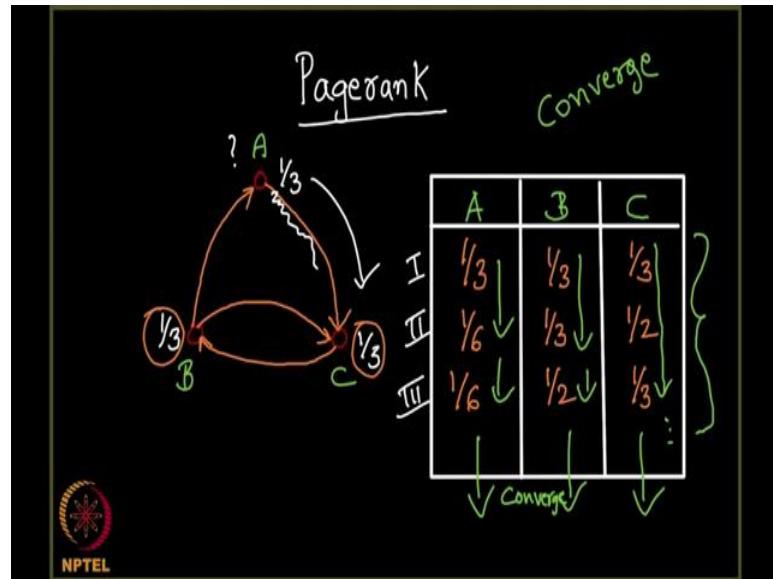


So, let us start with a nice example, let me try writing a graph for you all a very simple three node graph. So, here is the graph here are three nodes let me name it A, B and C we put edges from B does a edge from B to C and there is an edge from C to B and there is one edge from B to A and one from A to C.

So, what do you observe? You observe that there is this edge only one edge comes from A to C while this one edge coming from C to B, but there are 2 edges going from B right one to A and one to C correct ok.

So now, let me do the following let me write down table and make a note of what is happening here I need three columns let me write that down; one for A, one for B and one for C ok. So, here we are. So, I am going to write the values of A and B and C and what we are going to do with it I am going to start with some value for A, B and C let me write that down a here B here and C here. Let us start with the value of a being so, assume we had resources; we had resources allocated to A, B and C equally, what do I mean by that?

(Refer Slide Time: 03:24)



By that I mean I take A, B and C and I give one-third my resource to A and one-third of all the resources to C and the remaining one-third to B, which means I start with one-thirds to A one-thirds to B and one-thirds to C. What happens next? If you remember the programming screencast all that we did there was we took these values A is  $1/3$ , B is  $1/3$ , C is  $1/3$  and we told them and we observed and we told you all that A is going to give  $1/3$  to C correct ok. So, which means the new value of C, the new value of C is going to come from A correct. So, C will be  $1/3$  exactly the value that it had before all right ok.

So, what would be the value of B let us see, what is going to be the value of B? B takes the value from C as you can observe, which means B will be whatever C was  $1/3$  right. Let me write that down  $1/3$  and what we A, B? A what we A B? A is taking whatever B had, but B has two edges going which means it will divide its  $1/3$  equally to A and C, giving A half of  $1/3$  which is  $1/6$  and C gets half of  $1/3$  which is  $1/6$  once again.

So, let us see what happens to A; A gets  $1/6$  let me write that down  $1/6$ , but then again as you can see B gives  $1/6$  to C correct. So, what should I do? I should increments C here by  $1/6$ . So, this is the second iteration value. So, let me write this down what is  $1/3$  plus  $1/6$  here this is nothing else, but  $1/2$  correct a simple calculation tells me its  $1/2$ , let me write it down  $1/2$  well this was my first iteration this was my second iteration, let me go ahead and do the third iteration just for clarity sake.

So, once again what is what will be the value of let us say C? C gets all that A had ok, A this value was  $1/6$ . So, C gets whatever A had. So, C becomes  $1/6$  correct. So, please note this value  $1/3$ ,  $1/3$  and  $1/3$  is what we started from correct and that need not necessarily continue to be the values, the values change. You saw we start with  $1/3$ ,  $1/3$ ,  $1/3$  and it became  $1/6$   $1/3$  and  $1/2$  and now we are computing the third iteration right ok.

So, let us see what happens in the third iteration um. So, let me remove all the annotations that I did and let me compute the third iteration. So, what is the third iteration? It is very clear that we wrote C as  $1/6$  because C takes the value of whatever was the value of A so, it became  $1/6$  correct. Now, what would be the value of B? B simply takes whatever was the value of C. What is the value of C? The value of C was  $1/2$ .

So, which means my B will be  $1/2$  right let me write that down  $1/2$  perfect so far so good. So, what is the value of A? Now here is a small problem as you know A takes half of B right, which means B was  $1/3$  and A takes half of this as you can see A takes half of what B was? B was  $1/3$  as you can see. So, A becomes  $1$  over  $6$  which is  $1/2$  of B,  $1$  over  $6$  correct perfect.

Now, now as you can observe B also gives  $1/2$  of its existing value to C which means B was  $1/3$  and half of that is  $1/6$  and C gets  $1/6$  more correct plus  $1/6$ . What happens to this? This becomes  $2/6$  correct which is  $1/3$ . So, let me replace this by  $1/3$  and it goes on like this.

So, my question for you all is the following, if this process keeps continuing like this where will it reach? So, let me remove all these things. So, my question is this. If this keeps continuing like this goes on like this where will it reach, will it at all will it converge will this process, converge? So, what do I mean by that? By that I mean we will just go on like this randomly  $1/3$  becoming  $1/6$  and then  $1/6$   $1/3$  becoming  $1/3$  and  $1/2$ ,  $1/3$  becoming  $1/2$  and then  $1/3$  and so on. What will happen, will it ever converge, or will it keep continuing with its random values? So, let us try verifying what exactly happens in our next lecture.

**Social Networks**  
**Prof. S.R.S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Link Analysis (Continued)**  
**Lecture - 106**  
**PageRank Revisited – Convergence in the Example**

(Refer Slide Time: 00:05)

	A	B	C	D	E	F	G	H	I	J	K
1											
2	A	B	C								
3	0.3333333333	0.3333333333	0.3333333333								
4	0.1666666667	0.3333333333	0.5								
5	0.1666666667	0.5	0.3333333333								
6	0.25	0.3333333333	0.4166666667								
7	0.1666666667	0.4166666667	0.4166666667								
8	0.2083333333	0.4166666667	0.375								
9	0.2083333333	0.375	0.4166666667								
10	0.1875	0.4166666667	0.3958333333								
11	0.2083333333	0.3958333333	0.3958333333								
12	0.1979166667	0.3958333333	0.40625								
13	0.1979166667	0.40625	0.3958333333								
14											
15											
16											
17											
18											
19											
20											
21											
22											

We just now saw graph with three vertices A B and C and now going to use a spreadsheet to see the convergence question that I post in the previous lecture. So, if you recollect A was the node on top, B was the node on the left and C was the node on the right. So, the value of C completely the value of all these three vertices started off with  $1/3$ ,  $1/3$  and  $1/3$  correct. And the value of C if you remember if you do not, maybe you should watch the previous lecture and make note of the graph there right the value of C completely dependent on the value of A.

So, I will say is equal to A as value which is A 3 that is how that is how you use a spreadsheet. So, I am sort of bored using python interpreter for such easy and trivial things so, I am using spreadsheet. That is one reason one reason is because I find it very easy that is why I am using a spreadsheet, the second reason is that when you want to observe convergence it is very easy to use a excel sheet and then see the convergence, especially when it comes to topic such as this.

So, getting back C's value will be same as A 3 correct C was the bottom right node and the value of B if you remember was simply the value of C right, but then the value of A was half has let me just write this down was equal to what is that this I am sorry B's value was simply the value of C fine. And the value of A was half of the value of B, I write B3; B3 stands for this the column B and then row3. So, 0.5 times B 3 was the value of A, but then the value of C also had 0.5 times the value of B 3 correct.

Please note what I am doing here the formulas are visible here 0.5 times B3, the value of A is half of B3. And then the value of B is exactly the value of C 3 in the previous iteration whatever that was. And the value of C turns out to be the value of A3, if you remember from A it takes everything and half of what B 3 had B had which is this much right.

So, this is perfect so what I do is the facility that a spreadsheet gives is, when I just select this and come down and paste it. It will do the same thing that you did here for the next step. For example, A was equal to 0.5 as you can see A was equal to 0.5 times B3 right this should be 0.5 times B 4 you will observe it will automatically, see look at this it became B 4.

When you copy paste it excel gives you that liberty to automatically repeat the pattern of formula. So, B 5 happens to be C 4 and C 5 as you can see happens to be whatever was in A 4 plus half of whatever was in B 4. The best part now is if you just select these things and then pull this down it will populate the rest of the values. Now, you see the values are changing tremendously of a let us see when you look at A it is so on, so on, so on and you see the sort of values are same here, values are same here, it looks like it is converging right. So, let us look at it, let us continue this formula for more and then see whether it converges.

(Refer Slide Time: 04:01)

A	B	C	D	E	F	G	H	I	J	K
27	0.20000325521	0.399998724	0.399998724							
28	0.199991862	0.399998724	0.4000244141	0.399998724						
29	0.199991862	0.4000244141	0.399998724							
30	0.200012207	0.399998724	0.400004098							
31	0.199991862	0.400004098	0.400004098							
32	0.200002045	0.400004098	0.3999939965							
33	0.200002045	0.3999939965	0.400004098							
34	0.19999949482	0.400004098	0.3999989827							
35	0.200002045	0.3999989827	0.3999989827							
36	0.1999994914	0.3999989827	0.4000015259							
37	0.1999994914	0.4000015259	0.3999989827							
38	0.2000007629	0.3999989827	0.4000025453							
39	0.1999994914	0.4000025453	0.4000025453							
40	0.2000001272	0.400002543	0.399999185							
41	0.2000001272	0.399999185	0.400002543							
42										
43										
44										
45										
46										
47										
48										
49										
50										
51										
52										
53										
54										
55										
56										
57										
58										

Well, look at this it is almost 0.2, 0.39, 0.4 the values are not changing. Let me continue it even further right, its getting more and more sort of refined and even further let me go even further keep going, keep going, please note excel this just Google spreadsheet approximates the values here right.

(Refer Slide Time: 04:33)

A	B	C	D	E	F	G	H	I	J	K
37	0.1999994914	0.4000015259	0.39999989827							
38	0.2000007629	0.3999989827	0.4000025453							
39	0.1999994914	0.4000025453	0.4000025453							
40	0.2000001272	0.400002543	0.399999185							
41	0.2000001272	0.399999185	0.400002543							
42	0.1999998093	0.400002543	0.399999364							
43	0.2000001272	0.399999364	0.399999364							
44	0.1999999568	0.399999364	0.400000954							
45	0.1999999568	0.400000954	0.3999999364							
46	0.2000000477	0.3999999364	0.4000000159							
47	0.1999999568	0.4000000159	0.4000000159							
48	0.2000000079	0.4000000159	0.399999762							
49	0.2000000079	0.399999762	0.4000000159							
50	0.1999999881	0.4000000159	0.399999998							
51	0.2000000079	0.399999998	0.399999998							
52	0.199999998	0.399999998	0.400000008							
53	0.199999998	0.400000008	0.399999998							
54	0.2000000003	0.399999998	0.400000001							
55	0.199999998	0.400000001	0.400000001							
56										
57										
58										

Eventually, let us see what happens not much of a change. You still have 0.2 0.4 0.4 happening, but then you see what is happening here this is very, very close to 0.2 very

close to 0.4,, very close to 0.4 A's I keep continuing it you will be surprised to see that it will converge to the exact value.

(Refer Slide Time: 04:49)

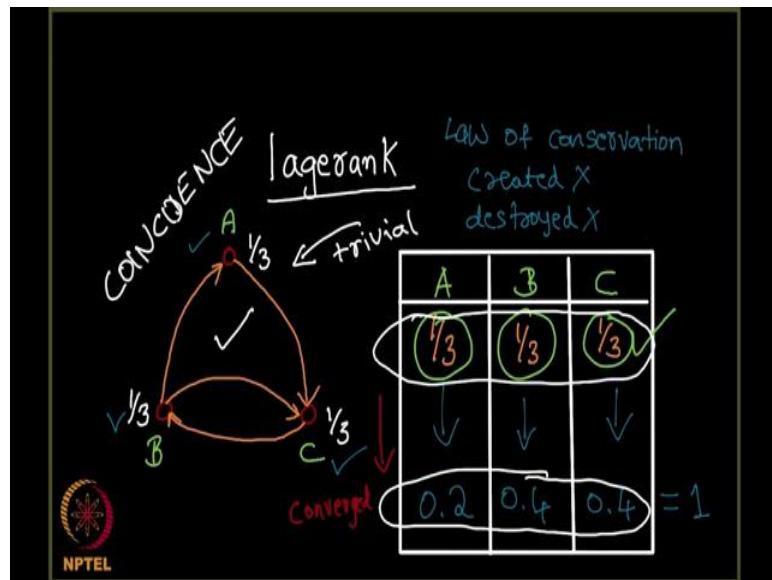
	A	B	C	D	E	F	G	H	I	J	K
57	0.200000005	0.399999985	0.400000001								
58	0.199999995	0.400000001	0.399999998								
59	0.200000005	0.399999998	0.399999998								
60	0.199999999	0.399999998	0.400000004								
61	0.199999999	0.400000004	0.399999998								
62	0.200000002	0.399999998	0.400000001								
63	0.199999995	0.400000001	0.400000001								
64	0.2	0.400000001	0.399999999								
65	0.2	0.399999999	0.400000001								
66	0.2	0.400000001	0.4								
67	0.2	0.4	0.4 CONVERGES!								
68	0.2	0.4	0.4								
69	0.2	0.4	0.4								
70	0.2	0.4	0.4								
71	0.2	0.4	0.4								
72											
73											
74											
75											
76											
77											
78											

Now look at this hip; hip hooray there is 0.2 here, 0.4 here, and 0.4 here. And this is the place where I would say it converges not just here; here itself right converges that is right. So, what just happened? Let us switch back to the screen cast, in my next lecture and then see what exactly we did here and what did we observe. Please note the values A turned out to be 0.2, B turned out to be 0.4 and C turned out to be 0.4.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Link Analysis (Continued)**  
**Lecture - 107**  
**PageRank Revisited - Conservation and Convergence**

(Refer Slide Time: 00:05)



So, let us look at what exactly happened, we took Google sheets and tried seeing what happens as we repeat this process alright. What is the process? Initially you give  $1/3$ ,  $1/3$  and  $1/3$  to all the notes correct you start with the same values, it is the resources that you have, we have used the example of gold coins if you remember I repeat.

We have used the example of gold coins I am saying resources, from now onwards I will say just an assignment right it is the value that I assigned and every time in every iteration what you do is you the value of let us say A is transferred to C, the value of C is transferred to B the value of B is transferred equally to A and C right we understand this so far so good.

Now when I observed where it converges, we observed that it indeed converges do you remember the values, the values were  $0.2$ ,  $0.4$  and  $0.4$  you see that the total is equal to  $1$ , why is that? That is very easy to see the resources that you have distributed to these nodes do not go out anywhere, they just remain in the network itself. This is like the law

if you remember let me write this down this is like the law of conservation of energy correct, the law of conservation of energy says energy can neither be created, it cannot be created nor can it be destroyed. What is happening here is exactly analogous to the law of conservation of energy.

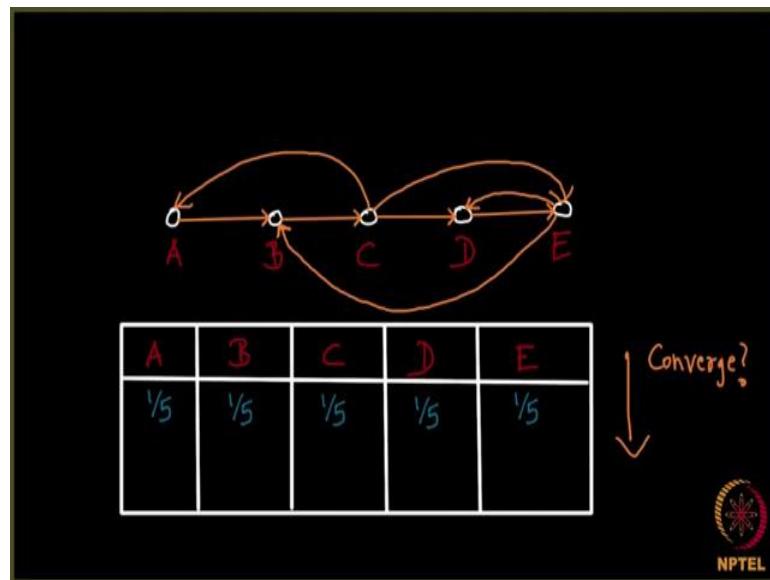
The sum total was 1 that is how you started off with right and it remains 1 until the end, but what just happened this process has actually converged that is what you need to understand. How did it converge? I suspect this was a coincidence that it converged. It may not so happen for all the graph; will it happen for any random graph? If I take and then see assign values to it and I will observe that it converges, may not be true.

Let us check for another graph to understand this. In this case the graph, that I considered here look slightly trivial, by trivial I mean what is it just 3 vertices and some 4 edges maybe it looked very symmetric and that is why the convergence happened. Maybe it may not converge, just in case I took some other graph; let us take some other graph and then see whether it converges or not.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Link Analysis (Continued)**  
**Lecture - 108**  
**PageRank, conservation, and convergence - Another example**

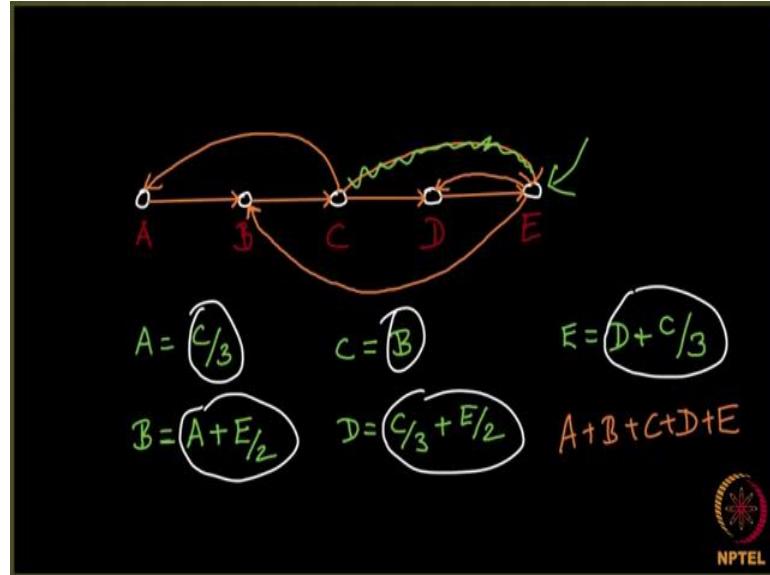
(Refer Slide Time: 00:09)



Let me consider a new graph with 5 vertices, let me call them A B C D and E and then let me put some random edges here and there. So, that every edge has some incoming links and some outgoing links. So, this is my graph, fine. And now let me create a table for this A B C D values and then start off with some values for A B C D E; I divided my resources equally to A B C D E resources or gold coins as in the previous example that we have seen in our lectures. So, what will that do? It will be  $1/5$   $1/5$   $1/5$  and  $1/5$  correct.

So, if I continue like this, we observed that it converged for 3 nodes. Now, I have taken 5 nodes not so, easy looking graph will it converge here as well is my big question, will it converge here or not. Let us go to the excel sheet and then the Google spreadsheet and then try doing the same thing that we did before, and we will see whether it converges or not. So, let us be prepared to go and edit there, let us observe what is happening here so, that the punching in of the equation will be easier for it.

(Refer Slide Time: 01:28)



Now, look at this  $A$  is actually equal to  $C$  by 3 correct, why is that? From  $C$  you see that there are 3 edges going out. So, the value of  $C$  will be divided between these 3 people. So, I will say  $A = C/3$ , good next:  $B = A + E/2$ . You can check why, maybe you should pause and then check for each value. I have written it down; I will just go through it is for you to check whether its right or not.  $B$  gets all the values from  $A$ ; it also gets half the value from  $E$  because it have 2 edges going from  $E$  as you can observe ok.

So,  $C$  will be  $B$  because there is only one thing that is coming inside  $C$  and  $B$  sends only 1 edge to see; I mean from emanating from  $B$  there are only 1 there is only 1 edge. And, then  $D$  is  $C/3 + E/2$  you can check why and finally,  $E = D$ . So, we will keep these things and then try to go to our spreadsheet and then try to code it. So, what I will do is I will directly go there to the spreadsheet, finish this assignments of  $A B C D E$ , start with  $1/5$   $1/5$   $1/5$  and bring you people there. And, then show you as I pull down the values on excel, on Google spreadsheet you will observe that I am populating everything, and we will see whether it converges or not.

So, that I forgot something here. So, let me just write that down  $E$  as you saw has incoming link from  $C$  as well not just from  $D$ . So, you have an incoming link from  $C$ . So, let me make a note of this  $E$  is not just equal to  $D$ , it is plus  $C/3$ ; a good way to see if all of them add up to 1 is to see if these things add up to, if these 5 things add up to, if you add these things you must get; what should you get guess. You must get  $A + B + C + D$

+ E, see if you are getting this and you just think of how and why exactly do I say they should add to A + B + C + D + E; it is a simple quiz question.

(Refer Slide Time: 04:08)

A	B	C	D	E
0.2	0.2	0.2	0.2	0.2
0.0666666667	0.3	0.2	0.1666666667	0.2666666667
0.0666666667	0.2	0.3	0.2	0.2333333333
0.1	0.1833333333	0.2	0.2166666667	0.3
0.0666666667	0.25	0.1833333333	0.2166666667	0.2833333333
0.0611111111	0.2083333333	0.25	0.2027777778	0.2777777778
0.0833333333	0.2	0.2083333333	0.2222222222	0.2861111111
0.0694444444	0.2263888889	0.2	0.2125	0.2916666667
0.0666666667	0.2152777778	0.2263888889	0.2125	0.2791666667
0.0754262963	0.20628	0.2152777778	0.2150462963	0.28792693
0.0717592592	0.2194444444	0.20925	0.215707407	0.2889055956
0.06875	0.215182037	0.2194444444	0.2121527778	0.2844907407
0.07314814815	0.2109953704	0.215182037	0.2153935185	0.2853029259
0.071720267901	0.2157986111	0.2109953704	0.214371142	0.2871141975
0.07033179012	0.2152777778	0.2157986111	0.2138888889	0.2847029321
0.07193287037	0.2126832562	0.2152777778	0.2142843364	0.288821793

I have now opened the Google spreadsheet and the values of A B C D and E as you would have guessed starts from 1/5. So, I am going to populate it with 1/5 throughout and as we computed in our previous lecture the value of A is going to be C divided by 3. I think you all know how Google spreadsheet works, as I told you earlier you just need to take the value of this which is C column and the third row. You should in fact, punch in is equal to C 3/3 ok. And then what was B? B was simply same as A plus what was in E divided by 2, is not it ok. What was my C?

My C was nothing else but, B you may want to look at my previous lecture if you are not following what I am doing. It is a pretty straightforward process my D is going to be D was slightly complicated. It was C/3 C/3 + E/2, E/2 that is it. And what was E? E was equal to my D + C/3, a good way to check whether it is right or not is just to see if it is equal to the entire row, is this equal to A 4 + B 4 plus; what is that A 4 + B 4 + C 4 + D 4 + E 4. It should be 1 perfect, which means the values are properly put; let me delete this now ok.

So, now what I do is I copy this and paste this and this creates a new iteration, as you see the value of this changes, but this value remains the same because it is simply again 0.2/3

C/3 and so on and so forth. Let me just pull this and see whether it converges or not, it is no way converging maybe it does not converge I do not know.

(Refer Slide Time: 06:35)

Let me go ahead and see well I see something promising; it looks like it is converging. You see 0.07 and 0.07 here 0.21 and 0.21 although the third decimal digit is not change, it is something else. I sort of see a ray of hope here maybe it will converge. Let me go on, let me go on the point is to keep going until you get the same values. It looks like it is converging, you see the last 3 the 3 decimal digits after the point looks like the same.

But I am not really happy I want to see the exact converged values. Let me go on, like go on and on I am still not happy, let me pull this too let us say as you can see, I am in 76th row. I have repeated this 76 times, if I go on to let us say 200 or let us say little over 100 I see that the values are exactly the same right. It looks like it is exactly the same. When it is exactly the same no matter where, until what point I go, I am going to 250 it will still be the same. You see these 2 cells it was so much and it still remains the same. Let us make a note of this, we are observing that the graphs here that the graph given graph here with 5 notes A B C D E; when you start with  $1/5 \ 1/5 \ 1/5$  it is converging right, it is converging to these values.

So, let us go to the next lecture and see what can one say about this, why exactly this is happening.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Link Analysis (Continued)**  
**Lecture – 109**  
**Matrix Multiplication (Pre-requisite 1)**

(Refer Slide Time: 00:05)



So, let us look at a few Pre-requisites before going further. These are nothing, but elementary let us say high school level mathematics precisely speaking it is matrix multiplication and we will understand what happens when you multiply a matrix. Although you would have understood what matrix multiplication is. You would not have seen this part that I am explaining next. Observe carefully, it is an interesting observation you can make about matrices ok.

(Refer Slide Time: 00:36)

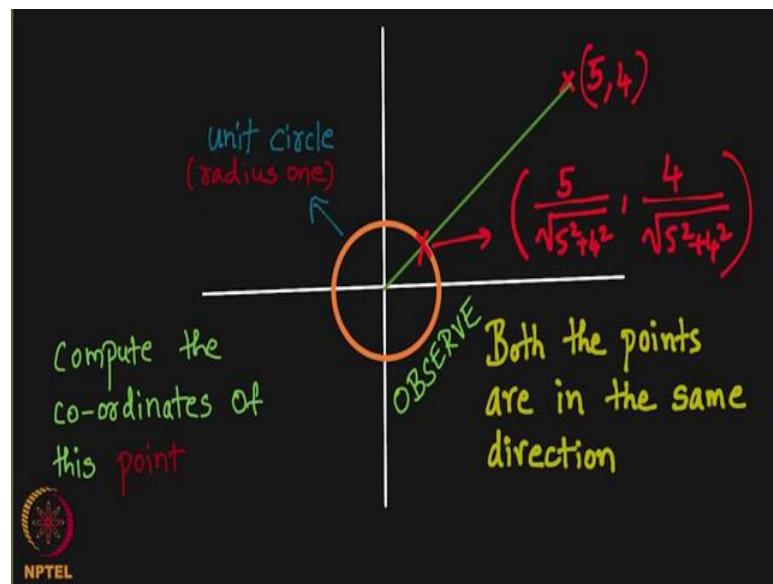
### Elementary Matrix Question

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1+2 \\ 3+4 \end{pmatrix} = \begin{pmatrix} 3 \\ 7 \end{pmatrix}$$


NPTEL

So, let us look at this very elementary matrix question you take a matrix  $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$  all right. We are going to play around this matrix only a simple  $2 \times 2$  matrix right. Multiply it with this vector  $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$ , what do you get? You simply get  $1 + 2$  and then  $3 + 4$  right fine. We all understand this much of matrices at least right next. This is equal to  $\begin{pmatrix} 3 \\ 7 \end{pmatrix}$ , so,  $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$ , is giving rise to  $\begin{pmatrix} 3 \\ 7 \end{pmatrix}$ , right good and then.

(Refer Slide Time: 01:15)



Let me see how one can represent points on this two-dimensional plane, two-dimensional coordinate system ok. So, we got  $\left(\frac{3}{7}\right)$ , right. Now let me take a point  $(5, 4)$  somewhere in this coordinate axis. I will tell you what 1 means by pulling this to a circle ok. You will understand it very soon it is a quite easy concept. So, look at the vector of this typically by this point  $(5, 4)$ . It is simply the line joining the origin and  $(5, 4)$  right and now what I do? I draw a unit circle. What do you mean by unit circle? By this you mean what is the unit circle? By a unit circle we mean, a circle with radius 1 with center as the origin all right.

So, what happens next? Here is a quiz question for you all. Can you compute what is this point? This very point here. What is this point? Pause the video and then compute this. Please note I have given you a point  $(5, 4)$  and I am giving you a unit circle and I am asking you where does this line joining the origin and  $(5, 4)$  intersect the unit circle right. This is some basic thing. You compute the coordinates of this point extremely basic right. What you do is you first write 5 and then divide this by the square root of the sum of the squares right.

We all have seen this. It takes a minutes pause and then observe this if you unable to see this. It is an extremely high school level coordinate system mathematics nothing beyond that. And then you have similarly  $4/\sqrt{5^2 + 4^2}$  and this is your point on the unit circle.

Given any point  $(a, b)$ , you can write the corresponding point intersecting the unit circle in the same direction as  $a/\sqrt{a^2 + b^2}$ . Here  $a$  is 5 and  $b$  is 4. So,  $4/\sqrt{5^2 + 4^2}$  all right? Particularly I showed it for the example  $(5, 4)$  so far so good you are understand what has happened so far.

Let us observe something an obvious observation is that both the points are in the same direction right. They are both in the same direction correct. From origin if you can see, these two vectors  $(5, 4)$  and  $5/\sqrt{5^2 + 4^2}$  both are in the same direction. Now let me look at what just happened before.

(Refer Slide Time: 04:08)

*Elementary Matrix Question*

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1+2 \\ 3+4 \end{pmatrix} = \begin{pmatrix} 3 \\ 7 \end{pmatrix}$$
$$\begin{pmatrix} 3 \\ 7 \end{pmatrix} \rightarrow \begin{pmatrix} 3/\sqrt{3^2+7^2} \\ 7/\sqrt{3^2+7^2} \end{pmatrix} = \begin{pmatrix} 0.39 \\ 0.91 \end{pmatrix}$$


NPTEL

We took let me repeat this, it was an elementary matrix question and we took matrix  $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$  and multiplied with  $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$ . We got  $\begin{pmatrix} 3 \\ 7 \end{pmatrix}$ , so, right.

Now, what do I do? I take this  $\begin{pmatrix} 3 \\ 7 \end{pmatrix}$ , and then try to see what the point on the unit circle in the same direction is. What do I mean by this? Just now we did that before right. Before the previous video that I showed you whatever we did here, I am going to ask you to do the same thing for the next slide. 3 by 7 when it is pulled to the unit circle will give you so much right. Again,  $a/\sqrt{a^2 + b^2}$  and  $b/\sqrt{a^2 + b^2}$  is the corresponding point on the unit circle.

What is this equal to? Let me compute I use my calculator and I saw this comes out to be  $\begin{pmatrix} 0.39 \\ 0.91 \end{pmatrix}$ , good. What next? What now? I multiplied  $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$  with  $\begin{pmatrix} 1 \\ 2 \end{pmatrix}$ . I am sorry  $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$  and I got  $\begin{pmatrix} 3 \\ 7 \end{pmatrix}$  and I sort of this is called normalization. I normalized it; I got this value on the unit circle.

(Refer Slide Time: 05:32)

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1+2 \\ 3+4 \end{pmatrix} = \begin{pmatrix} 3 \\ 7 \end{pmatrix}$$
$$\begin{pmatrix} 3 \\ 7 \end{pmatrix} \rightarrow \begin{pmatrix} 3/\sqrt{3^2+7^2} \\ 7/\sqrt{3^2+7^2} \end{pmatrix} = \begin{pmatrix} 0.39 \\ 0.91 \end{pmatrix}$$


NPTEL

What if I continue doing this process right? I got this  $\begin{pmatrix} 3 \\ 7 \end{pmatrix}$ . I continue this I pull it to the unit circle I got so much. What if I take these two things look at this from the previous slide, I simply take this  $\begin{pmatrix} 0.39 \\ 0.91 \end{pmatrix}$ .

(Refer Slide Time: 05:46)

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{pmatrix} 0.39 \\ 0.91 \end{pmatrix}$$


NPTEL

And I apply  $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$  matrix on it ok.

(Refer Slide Time: 05:54)

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{pmatrix} 0.39 \\ 0.91 \end{pmatrix} = \begin{pmatrix} 2.21 \\ 4.81 \end{pmatrix}$$

$\begin{pmatrix} 2.21 \\ 4.81 \end{pmatrix} \rightarrow$  normalize &  
repeat this process.



What do I get? I get my calculator says I get  $\begin{pmatrix} 2.21 \\ 4.81 \end{pmatrix}$ . Again, I repeat this process, I normalize this, and I see as I keep repeating this process, what exactly do I get.

(Refer Slide Time: 06:13)

$$M \begin{bmatrix} \text{vec}^1 \end{bmatrix} = \begin{bmatrix} \text{vec}^2 \\ \text{vec}^3 \\ \text{vec}^4 \end{bmatrix}$$

$\downarrow$        $\downarrow$        $\downarrow$   
 $y_3$      $y_3$      $y_3$   
 $\downarrow$   
Converged

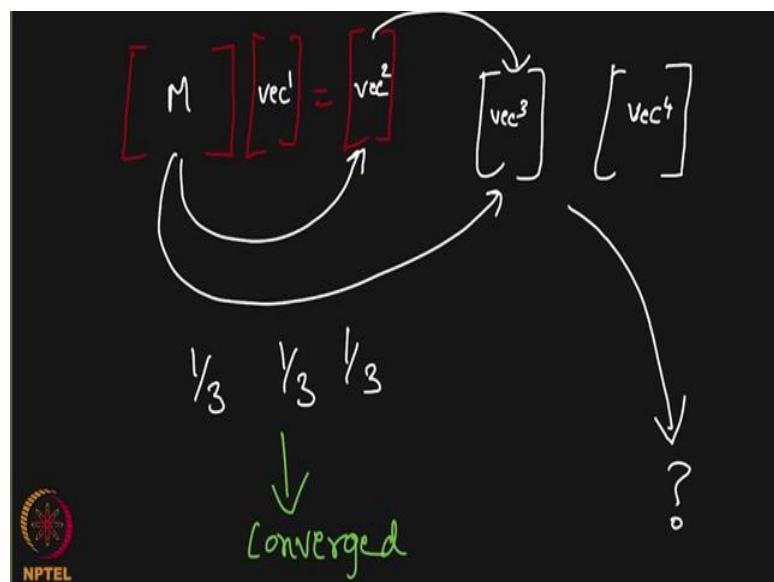


So, what I see I took a matrix, I took a matrix applied it to a vector; I got another vector. So, a vector a matrix I got another vector. What I do is I take this matrix and apply it to the same vector. Now before that I sort of, I normalized it normalizing is just to ensure that you do not get bigger values. All you care about is the direction I will tell you what I am trying to do, and I get a new vector another vector. Let us say this was vector 1, this is

vector 2. Now you get a vector 3 all right and then again apply this matrix on this vector, you get a vector 4. At every instance you are normalizing by normalizing I mean; you pull it to the circle, and you see where exactly this is going where exactly this is going.

Somehow, we seem to be interested in repeating the same process again and again and we see what happens. We saw what happened with the iterated value of this matrix. Remember 1/3, 1/3, 1/3 not of the matrix that of the graph. We took a graph with 3 nodes, we are saying the values 1/3, 1/3, 1/3 and we saw that it converged. And now we are asking a matrix question; it converged right now we are asking a matrix question right.

(Refer Slide Time: 07:48)



When you apply a matrix repeatedly on a vector, what happens to it? Will it also converge or not? Obviously when you apply a matrix like  $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$  on a vector the values become bigger and bigger and that is why I am trying to pull it back to the unit circle right. I am normalizing it, what I will do is, I will switch to screen cast mode and I will try to understand this matrix multiplication process nicely.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Link Analysis (Continued)**  
**Lecture – 110**  
**Convergence in Repeated Matrix Multiplication**  
**(Pre-requisite 1)**

(Refer Slide Time: 00:05)

```
Sudarshans-Air:~ sudarshaniyengar$ ipython
Python 2.7.13 |Anaconda 4.4.0 (x86_64)| (default, Dec 20 2016, 23
:05:08)
Type "copyright", "credits" or "license" for more information.

IPython 5.3.0 -- An enhanced Interactive Python.
?           --> Introduction and overview of IPython's features.
%quickref --> Quick reference.
help        --> Python's own help system.
object?    --> Details about 'object', use 'object??' for extra det
ails.

In [1]: import numpy
In [2]: A=numpy.mat('1 2;3 4')
In [3]: v=numpy.mat('1;1')
In [4]: A*
```

Let me now open our ipython shell and let me try coding whatever I observed just now . So, before which I will try to help you understand a few commands, there is something called the numpy library which is a storehouse of a lot of matrix operations functions which help you meddle with matrices. So, how do you declare a matrix? You simply say A equals numpy.mat and then you write down your matrix in the form of the 1st row followed by a semicolon and then the 2nd row.

If you remember it right from a previous lecture we are taking this matrix  $\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$  and then there is a new vector v that I defined which is if you remember [1, 1], that is defined in python as 1 space 1, no that is not right 1 semicolon and then 1 isn't that right. So, what is it denote, I am sorry what is it denote? 1 is the first row, 1 is the second row right. So, when I say A times v now A is a matrix  $\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$  and v is the matrix (1 1).

(Refer Slide Time: 01:16)

```
IPython 5.3.0 -- An enhanced Interactive Python.
?          --> Introduction and overview of IPython's features.
%quickref --> Quick reference.
help       --> Python's own help system.
object?    --> Details about 'object', use 'object??' for extra details.

In [1]: import numpy

In [2]: A=numpy.mat('1 2;3 4')

In [3]: v=numpy.mat('1;1')

In [4]: A*v
Out[4]:
matrix([[3,
         [7]])

In [5]: edit mat.py
```

I get [[3], [7]] which is  $1 + 2$  and then  $3 + 4$  correct that is the way in which you multiply a matrix with the vector. So, let me now open a new shell ipython shell and then type a piece of code, to observe what happens when you keep repeatedly applying A on the vector v of course, you keep normalizing it every now and then ok.

(Refer Slide Time: 01:50)

```
1 import numpy
2
3 A=numpy.mat('1 2;3 4')
4 v=numpy.mat('1;1')
5 print v
6 print "# #####"
7 for i in range(10):
8     z=A*v
9     z=z/numpy.linalg.norm(z)
10    print z
11    print "*****"
~
~
~
~
~
~
~
neocomplete requires Vim 7.3....later with Lua support ("+lua").
```

So, let me edit mat.py and what I do is I say import numpy and then I will declare my matrix which is  $A = \text{numpy.mat} ('1;2 ; 3 4')$  the close and then my initial vector is going to be  $\text{numpy.mat} ('1; 1')$ . And now what I am doing is, for  $i$  in range 10 you know what I

am doing here; I say  $z = A * v$  right and then I need to normalize this vector right. Do you remember  $\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$  application on  $[1, 1]$  gave us some value and we pulled that to the unit circle? That pulling to the unit circle is done by the following operation, what you do is  $z = z/\text{numpy.linalg}$  this is a linear algebra package that is available within numpy you say  $\text{norm}(z)$ .

So, this one is going to be a real number which is basically that what you get. So, 1 plus 2 is  $3 + 4$  is 7 this becomes  $[[3], [7]]$  when you apply this matrix on  $[1, 1]$ . And,  $[[3], [7]]$  you normalize it what do you do 3 divided by 3, 7 divided by square root of 3 square + 7 squared right. Remember in the previous lecture I am just doing the same thing here.

After doing this what I do is, I print my  $z$  and I print some stars here. So, that it looks neat it separates two things and then I go back here, and I simply display my  $v$  for clarity sake; just to tell you from where we started from correct. And, this is just beautification do not worry much this is just for us to read the output clearly nothing else ok.

So, I am going to say this if you want let us go through the code once import numpy that imports all the library functions, that numpy has and I declare a matrix  $\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ . And, then the next line is I declare this column matrix  $[1, 1]$ . I simply display what is this matrix  $[1, 1]$  here. And, then I repeat this loop 10 times what do I do, I apply my matrix  $A$  on  $v$  and call it  $z$  and then I normalize  $z$ . What do you mean by normalizing  $z$ ? You pull it to the unit circle right, that is what this does linear algebra norm precisely does that square root of sum of squares nothing right. So, now let us say you print  $z$ .

(Refer Slide Time: 04:47)

```
ails.

In [1]: import numpy
In [2]: A=numpy.mat('1 2;3 4')
In [3]: v=numpy.mat('1;1')
In [4]: A*v
Out[4]:
matrix([[3,
       [7]])

In [5]: edit mat.py
Editing... /Users/sudarshaniyengar/anaconda/lib/python2.7/site-pa
ckages/IPython/core/magics/code.py:714: UserWarning: Could not op
en editor
  warn('Could not open editor')

In [6]: run mat.py
```

Let us see what the output of this is, forget the warning message that you keep getting like this that generally happens because of the it is relation problems. And, you would not output the write editor and stuff like that anyway let us ignore it.

(Refer Slide Time: 05:03)

```
In [6]: run mat.py
[[1]
 [1]]
#####
[[ 0.3939193 ]
 [ 0.91914503]]
*****
[[ 0.3939193 ]
 [ 0.91914503]]
*****
[[ 0.3939193 ]
 [ 0.91914503]]
*****
[[ 0.3939193 ]
 [ 0.91914503]]
*****
[[ 0.3939193 ]
 [ 0.91914503]]
```

So, what I do is run mat.py and I get some answer here yes. So, I start with [1, 1], it is application whose application? The matrix  $\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$  application on [1, 1] gives me [[3], [7]]. When I normalize [[3], [7]], if you remember from my previous lecture, I indeed got

0.39 and 0.91 there. The question there was if I repeatedly apply what do I get? I am getting the same values as you can see.

(Refer Slide Time: 05:32)

```
[[ 0.3939193 ]
 [ 0.91914503 ]]
*****
[[ 0.3939193 ]
 [ 0.91914503 ]]
*****
[[ 0.3939193 ]
 [ 0.91914503 ]]
*****
[[ 0.3939193 ]
 [ 0.91914503 ]]
*****
```

In [7]:

It is simply 0.3939193 and 0.91914503 I hope there is no error with the code, let me just check the code.

(Refer Slide Time: 05:38)

```
1 import numpy
2
3 A=numpy.mat('1 2;3 4')
4 v=numpy.mat('1;1')
5 print v
6 print "#####"
7 for i in range(10):
8     z=A*v
9     z=z/numpy.linalg.norm(z)
10    print z
11    v=z
12    print "*****"
~
```

So, there is a problem here indeed this is the reason why I think it is good for us to code together and if there is an error I will not redo the coding I will correct it here; so, that you will realize what kind of mistakes one can do. So, I have made a very silly mistake here, I

am simply taking z and then applying assigning A of v we do it right. What I should do is, if I want repeated application after displaying z I should say  $v = z$ .

So, that when I come here, I am sorry when I come here the next z will be A applied to my previous z which was v right. So, the output that I got previously was incorrect let us redo this. So, let me say run mat.py and I get some values here, let me see what this is 0.39 0.91 as expected and then the next one is something else. You see 4175 something the next one is 4158 something, 4159 something, 415973, but this was 415980 so on and so forth you will observe that oh my goodness it is converging. 0.41597356 here the next iteration the 10th iteration was 41597356 precisely the same as the previous iteration, you will even observe it with this.

So, these two column vectors sort of are the same, maybe this was this converged. What if, let me do something what if what was that mat.py and so, edit mat.py and I will go here and I will try to vary this [1, 1] to let us say 1 comma let us say let us say some 3 comma sometime 11 alright. Now, let us see what happens to this, I come out the code remains the same I come out and I execute this; you see what happens looks like I am getting the same values how is that.

(Refer Slide Time: 07:45)

```
In [9]: edit mul.py
Editing...
In [10]: edit mat.py
Editing...
In [11]: run mat.py
[[ 3]
 [11]]
#####
[[ 0.42661868]
 [ 0.90443159]]
*****
[[ 0.41523533]
 [ 0.90971403]]
*****
[[ 0.41602471]
 [ 0.90935331]]
*****
[[ 0.41597001]
 [ 0.90937833]]
*****
```

So, [[3], [11]] gave me this value after normalizing and application of the matrix A on this gives me this. An application of my matrix A on this gives me this right you see that finally; I am getting 41597356 and 90937671 this looks familiar. Why? Let me scroll up

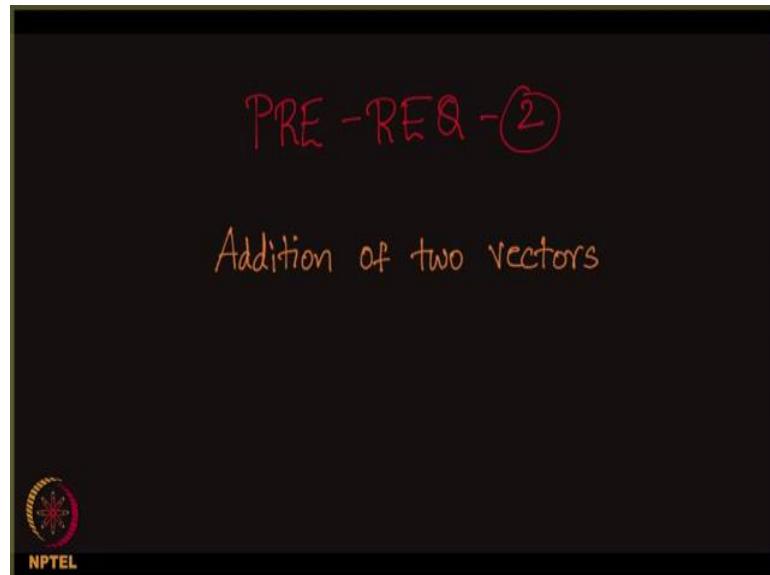
and see do you see that this is precisely the value that we got before, 41599093 come down 41599093 it is exactly the same.

What do we conclude here? No, matter where you start from what vector you start from, when you start applying the same matrix  $\begin{pmatrix} 1 & 2 & 3 & 4 \end{pmatrix}$  on any vector, you get a constant vector. It converges to a constant vector.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**  
**Link Analysis (Continued)**

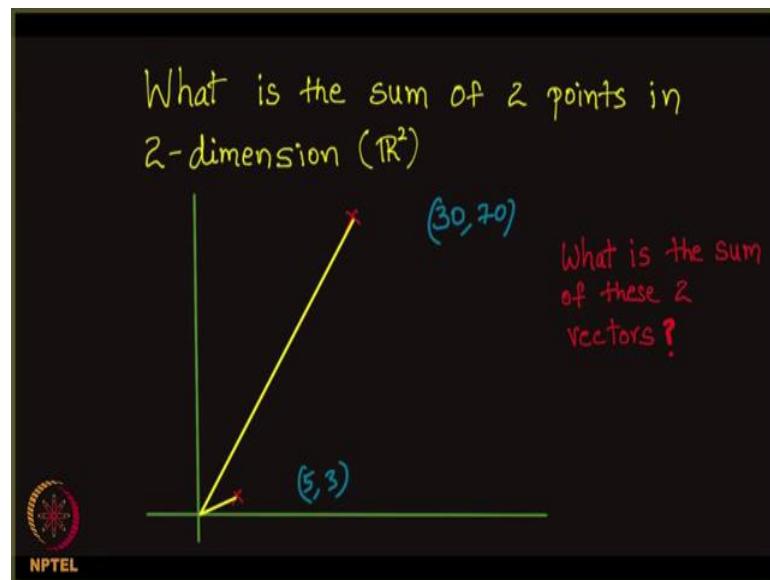
**Lecture – 111**  
**Addition of Two Vectors (Pre-requisite 2)**

(Refer Slide Time: 00:05)



Let us start with another prerequisite right now. I spoke about the first prerequisite which is the fact that the matrix multiplication on a vector leads it to convergence state. And now I am talking about the second prerequisite which is which is actually a very straightforward observation. It is about addition of two vectors. We have studied the notion of scalars and vectors in our high school days; I am just going to revise that and make some subtle observation about them ok. What happens when you add two vectors? Is a very straightforward question we all have added  $(a, b)$  vector,  $(a, b)$  added with  $(c, d)$  gives you  $a + c$  and  $b + d$  correct, but we have not observed something important here. Let us observe that now.

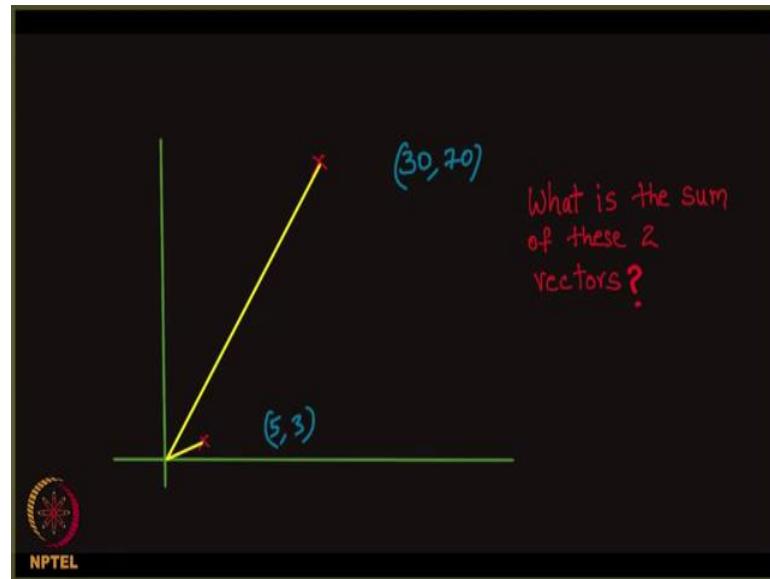
(Refer Slide Time: 00:55)



So, here is my question. What is the sum of two points in two dimension? Let us say this is my two-dimension plane; and this is my first point. I will give it some name; let us say at the point (30, 70) all right. And another point sort of relatively close to origin and this point is (5, 3) all right. So, let me join the corresponding vector. What do you mean by that? You just join the, origin and the point by a line this is called the position vectors right.

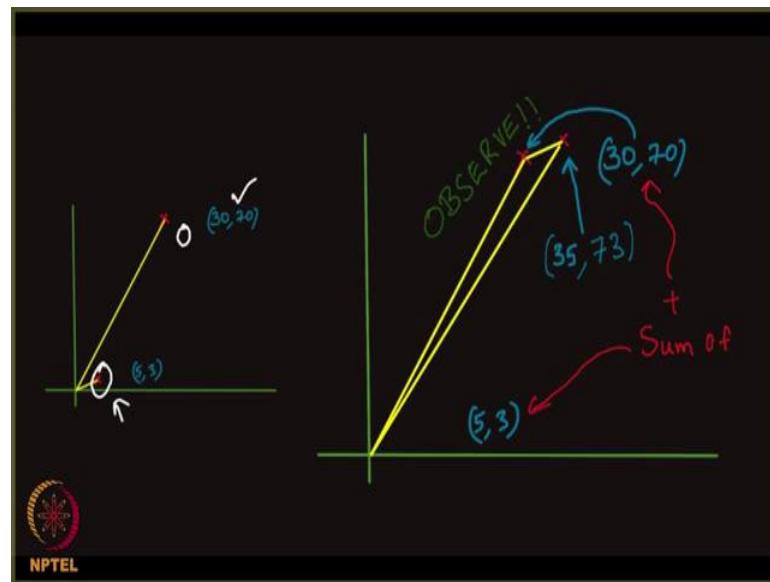
As you saw I did that for both ok this one as well as this one; Next what is the sum of these two vectors? What can you say about the sum of these two vectors? This is the right time for you to pause the video and compute the sum. And look at where exactly the sum comes. What is the sum 30 plus 5, 70 plus 3, (35, 73)? Where exactly is this point? What is its direction? My direction I mean when you join the origin and this point. Is it close to the point (30, 70) or is it close to the point (5, 3) is a question in hand? Let us see that in detail now.

(Refer Slide Time: 02:13)



So, let me write that once again. And my question is what is the sum of these two vectors right and I will try to find the sum very; obviously, the sum is simply this point that is the sum of  $(30, 70)$  and  $(5, 3)$  right.

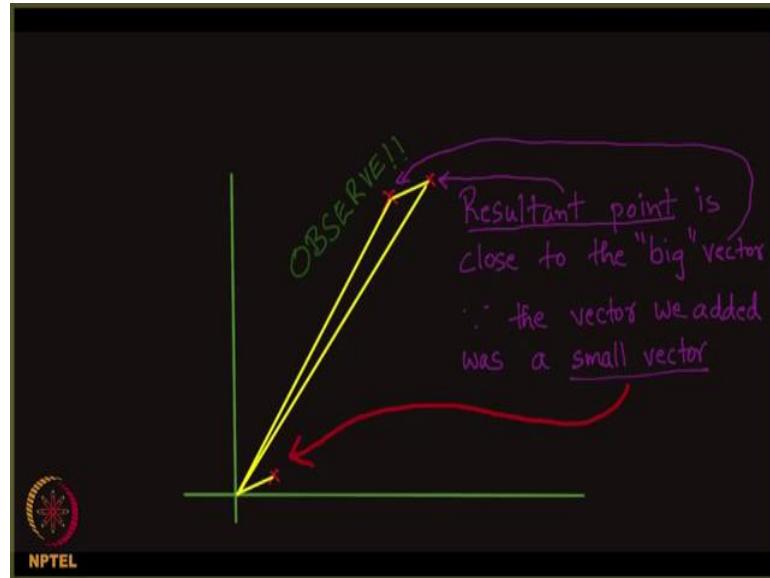
(Refer Slide Time: 02:34)



And I did not mean to exaggerate this concept. It is a very straightforward concept, but there is the observation is not so straightforward. The observation comes next. This  $(35, 73)$  is close to  $(30, 70)$  than  $(5, 3)$ .

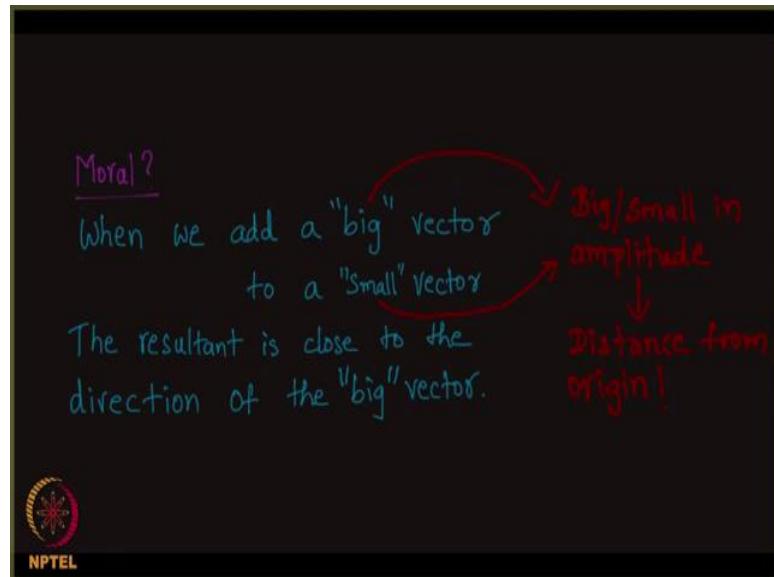
This vector as you can see (5, 3). The resultant vector is somewhere (35, 73) is somewhere close to this vector than to this vector; did you observe that? Yeah and I am sure you know the reason as well ok. We observed this next.

(Refer Slide Time: 03:10)



So, let us see what exactly this means. The resultant point is close, when you add the resultant point is close to the big vector. Whatever I mean the big vector, the vector that is big in amplitude. This is the big vector I am talking about; the resultant point which is this is close to the big vector. Because the vector we added was a small vector, please note if two vectors which are of the same amplitude when you add, this may not happen. This will happen only when you take a big vector and add it to the small vector. What do you mean by a big vector? A vector with a big amplitude is called a big vector. And this is a small vector because its the amplitude is small right ok.

(Refer Slide Time: 04:09)

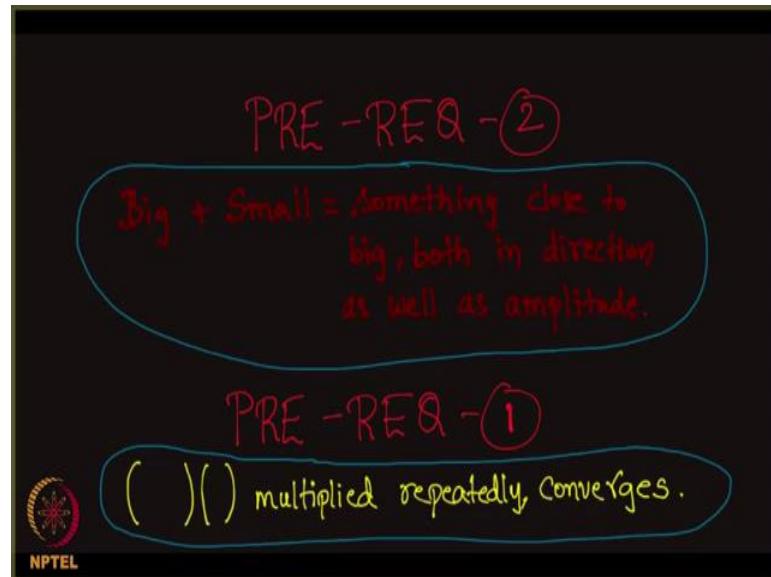


So, this is a small vector right ok. What is a model that we learned just now? What do we observed just now? So, you can pause and think about what we just now said. And then continue the video. When we add a big vector to a small vector, the resultant vector is close to the direction of the big vector big in codes right; perfect although pretty obvious you will even be wondering why I even had, a lecture on this such a simple concept.

This is the displays a key role in understanding the convergence of the matrix that we discussed previously right. So, it is important for us to spend some time on these seemingly trivial concepts. Most of the concepts are trivial, but their significance is very high. So, it is important for us to understand although appealingly trivial example. And it is important to us to understand these concepts one level deep. Because they will come in not so obvious places; and we may not be able to interpret them properly.

So, big and small by big and small I mean in the amplitude as I told you people right, perfect. So, this is the distance from the origin and its revising whatever. So, simply the moral is very clear to you people when you add a big vector to a small vector, you get a vector close to the big vector both in direction as well as amplitude ok.

(Refer Slide Time: 05:41)



So, Big plus small is equal to something close to big both in direction, as well as amplitude.

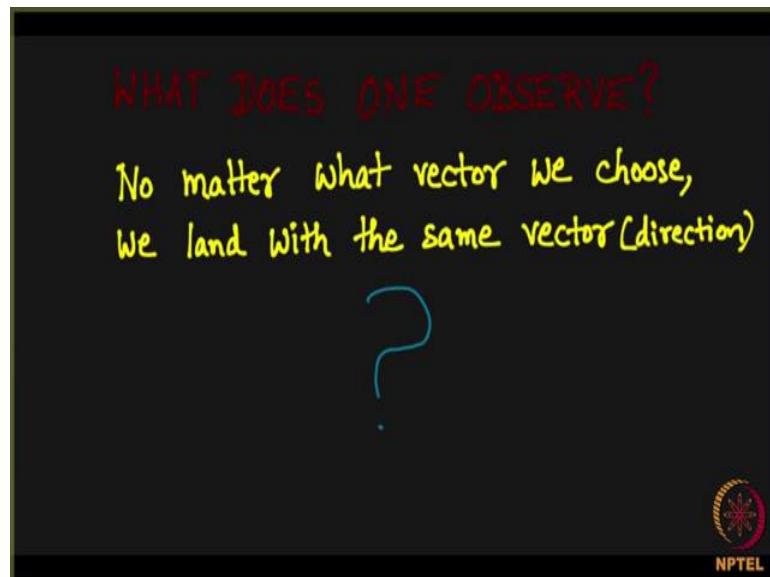
So, this was our prerequisite 2. And it will help if we can recollect prerequisite 1. Can you all recollect? What is prerequisite 1? Was this prerequisite 1 was simply the fact that when you multiply a matrix to a vector repeatedly it converges. So, please ensure that you do not go further without understanding this prerequisite 1 and 2.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Link Analysis (Continued)**  
**Lecture - 112**  
**Convergence in Repeated Matrix Multiplication- The Details**

So, we saw a couple of prerequisites and we are all set to go ahead and then make our required observations. Let us go very slowly. I will help you all recollect what has happened so far and then try to connect different pieces of the puzzle.

(Refer Slide Time: 00:22)



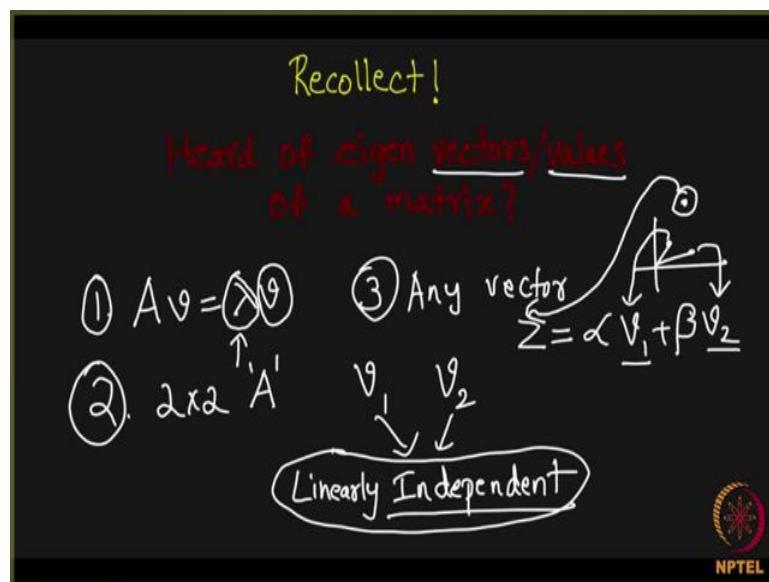
So, what does one observe of this matrix multiplication process right? So, we observe that no matter what vector we choose, irrespective of what vector we choose remember the screen cast that I did of the python programming code. I took different vectors. No matter what vector I chose, I landed up with the same vector right.

We all converged we observe that every single vector, no matter where you start from the repeated application of the matrix results in the very same vector. By very same vector, I mean the same direction right. This we observe. Why did this happen? What makes this process of applying the matrix repeatedly on a vector and scaling it down of course, scaling it down is to ensure that the numbers do not become big that is all, nothing else. As you

keep applying the matrix on any given random vector, it always goes to the same point in  $\mathbb{R}^2$  plane;  $\mathbb{R}^2$  is that two-dimensional plane correct.

Why is this happening? What is the physics behind it? What is the logic behind it? Let us unravel that slowly.

(Refer Slide Time: 01:42)



So, let us recollect some basics from our again high school mathematics. Throughout our discussion we are not going to use anything HiFi. All we are going to use is some very basic matrix theory.

I am sure you all heard of eigen values and eigenvectors right. So, let us recollect that. So, given a matrix we know there is something called an eigenvector and eigen value. Now, what is that? Let me help you recollect it and eigenvector is something of a matrix  $A$  eigenvector  $v$  is defined as something that simply gets scaled up by a lambda factor. Then lambda is called an eigen value and the  $v$  is called an eigenvector correct ok.

So, this is a right time for you to open let us say Wikipedia or any other online reference and then refresh your basics of eigenvalues and eigenvectors. You only need this definition that  $Av = \lambda v$ . This is first thing that we need to know and second thing that we need to know is for a  $2 \times 2$  matrix, let us say  $A$  a  $2 \times 2$  matrix  $A$ ; there are 2 eigenvectors not always, but mostly you will always have 2 eigenvectors ok. And these eigenvectors are

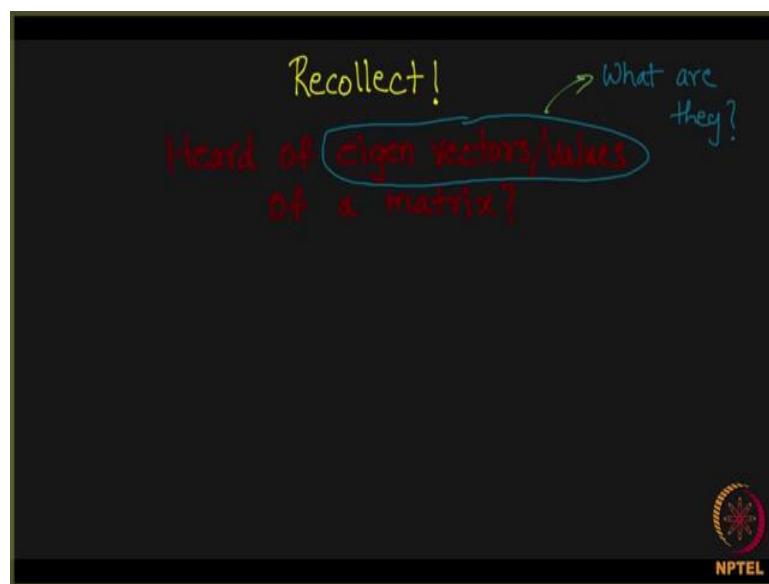
independent. They are independent. What do I mean by that? By that I mean you take any vector any vector of your choice.

Let us say any vector  $Z$  you can always write  $Z$  as a linear combination of  $v_1$  plus some  $\beta * v_2$  because that they are what is called linearly independent you can always write any vector as the linear combination of  $v_1$  and  $v_2$ . If you do not know these things, you probably should brush up your basics. So, this is the third one.

First one is the definition of Eigen values and eigenvectors, second one is given  $2 \times 2$  matrices is matrix. There is always 2 eigenvectors and they are linearly independent. What do you mean by linearly independent? Two vectors that are linearly independent in  $\mathbb{R}^2$  given any point in  $\mathbb{R}^2$  that point can be written as the linear combination of these 2 eigenvectors.

So, this point is  $Z$  you can always write  $Z$  as  $\alpha * v_1 + \beta * v_2$ . This is the basics of matrix theory, I am sure all of you are familiar, if not please take a pause and look at it. You need not know the reasoning behind all these things, you just need to recollect these things that should be enough ok.

(Refer Slide Time: 04:35)



So, let us go further now ok. So, please revise now before going any further. You should know what eigenvectors and eigenvalues are, I am not going to apply them in any not so obvious manner, every single application of this concept will be pretty straightforward.

(Refer Slide Time: 04:57)

So, I hope you have recollected what is eigenvector and what is an eigenvalue and I proceed further ok. Now how and why of eigenvectors and eigenvalues here? So, what did we do in our programming screen cast? What did we observe of this matrix multiplication? Whenever a matrix  $A$  acts on any vector  $v$  any vector  $v$  please observe that  $A$  is a matrix  $v$  is some random vector  $v$ , then you can always write  $v$  as a linear combination of  $\lambda_1 * v_1 + \lambda_2 * v_2$ . This is always possible right. We just now saw in that in our previous prerequisite. This is always possible; let us note this ok.

Next so, this  $v_1, v_2$  are eigenvectors and  $\lambda_1, \lambda_2$  are eigenvalues. We observed it already ok, I am just helping you recollect it by saying it once more so far so good. So, all I am saying here is any given vector  $v$  if a matrix  $A$  is applied on it, then you can always see it as  $A$  being applied on a linear combination of eigenvectors  $v_1$  and  $v_2$  no matter what  $v$  you choose. Now what is this equal to? This is equal to  $A$  is  $A$  matrix. Its application on a scalar  $\lambda_1 * v_1$ , you can always pull out the scalar here as; you can see you can pull out the scalar here correct.

So, you can write it as  $\lambda_1 * A(v_1) + \lambda_2 * A(v_2)$ . But then, observe carefully, what are  $v_1$  and  $v_2$ ?  $v_1$  and  $v_2$  are eigenvectors and so, what if they are eigenvectors? If they are eigenvectors, you can further write this  $A(v_1)$  as  $\lambda_1 * v_1$  right;  $A(v_1)$  is  $\lambda_1 * v_1$ ,  $A(v_2)$  is  $\lambda_2 * v_2$  and you finally, get this correct.

(Refer Slide Time: 07:00)

$$A^k(v) = \lambda_1^k v_1 + \lambda_2^k v_2 \quad (\text{Say } \lambda_1 > \lambda_2)$$

Amplitude      Direction

NPTEL

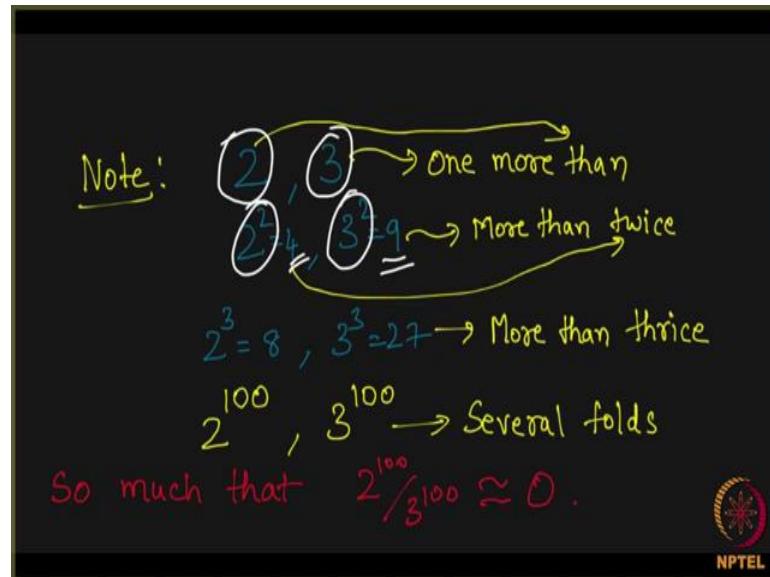
As I continue this process, look at my previous slide; as I continue this process, I apply A again on this, I continue to apply A gain on this. What do I get? I repeatedly apply A that is equivalent to me applying  $A^k(v)$ . So, I am talking a lot of things sort of very quickly. I suggest that you take a pause and then look at what I am saying all right. So, I am applying  $A^k(v)$  which gives me  $\lambda_1^k * v_1$ , why? Pretty obvious look at the previous slide if you are confused look at this, understand this carefully and you will understand what I am doing here.

This gives me  $\lambda_1^k * v_1 + \lambda_2^k * v_2$  correct perfect so far so good absolutely no confusion so far right. Carefully observe this  $\lambda_1$ . Let me assume is greater than  $\lambda_2$  all right. This always holds good. Eigenvalues are most of the times distinct; when they are distinct one of them is greater than the other one.

So, when one of them is greater than the other one, what happens?  $\lambda_1$  is basically the amplitude right  $\lambda_1^k$ ; if  $\lambda_1$  is some let us say 2 a, number greater than 1, then  $\lambda_1^k$  for a huge k will be a huge value right. This is what we call as amplitude for the vector  $v_1$ . When you multiply  $\lambda_1^k$  to  $v_1$  it sort of scales  $v_1$  up,  $\lambda_2^k$  being a big number when multiplied to a vector  $v_1$  it just makes this vector shoot away from origin right. We have discussed this already.

So, think about it for a minute.  $v_1$  simply signifies the direction and this product tells us the final vector which is extremely scaled right. The value the existing  $v_1$  is getting pushed by this value  $\lambda_1^k$  that is what this means ok.

(Refer Slide Time: 09:10).



Let us note something here. Let us observe this carefully. Let us take these two numbers 2 and 3 right just plain simple 2 number 2 number 3 and look at this 3 is one more than 2 correct as simple as that 3 is 1 more than 1 right which is like saying 3 is 50 percent more than 2 correct.

But then when you square it  $2^2$  gives you 4,  $3^2$  gives you 9 and then what happens? This 9 is more than twice of 4. When you take 2 numbers A and B, if  $B > A$ , if you look at their proportion by what factor it is greater, you observe that this number is 50 percent greater than this number. But when you square them, you observe that it turns out to be twice as much as this number. As you continue this way you will observe that look at this what happened.

If you cube it, you get 8 and 27; now that is surprisingly more than 3 times. So, this is like saying let me give you a nice fictitious example. Look at your bank balance look at my bank balance. Assume your bank balance is 2 lakhs and my bank balance are 3 lakhs. Let make you feel happy by making you rich. Assume your bank balance is 3 lakhs and my bank balance is 2 lakhs all right which is like you are just 1 lakh richer than me.

So, assume God comes and cubes our bank balance, he cubes. So, my bank balance was 2 he makes it to cube and your bank balance was 3 lakhs, he makes it 3 cube. So, initially you were just 50 percent richer than me, but now you become more than thrice richer than me right. So, this although God came and cubed me as well as you, he did this he gave the

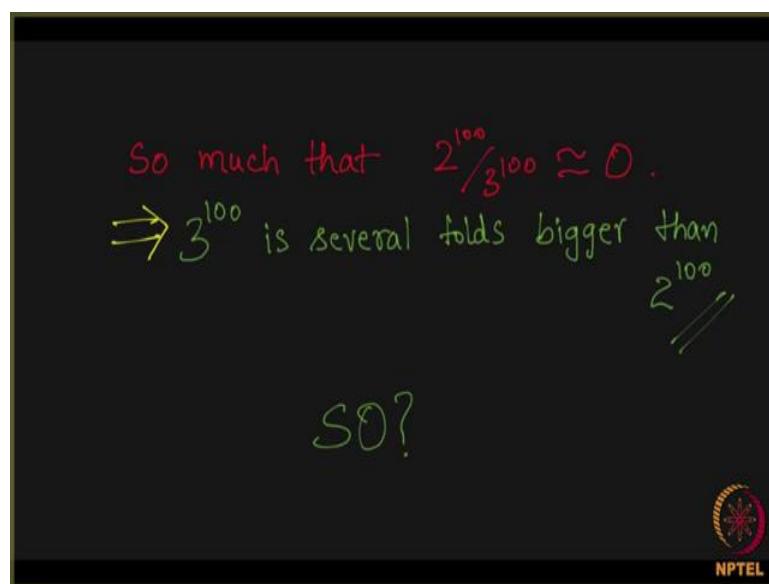
same gift worth to me and you depending upon what was the number that we had, we ended up having a bigger number. A person who had more, now has a lot more I think you got the intuition.

So, now as we keep going on further, this was more than thrice. We observed we keep doing this. Let us say we empower it by 100 then, we observe something startling. This  $3^{100}$  is several forces, bigger than  $2^{100}$ . So, big that let us observe what happens. So, big it several force big that so much more than this that if you look at the ratio it is close to 0, why?  $2^{100}$  divided by  $3^{100}$  as you can see is  $(2/3)^{100}$ .  $2/3$  is a number smaller than 1 and you are empowering it to the number 100 which is a very big number you take as number less than 1 and keep multiplying it to itself it will quickly go to 0 you see, that is what is happening here. Take a minutes pause and observe what exactly we explained in this slide look at the previous slide right.

We are talking about the multiplication of a big number namely  $\lambda_1^k$  and  $\lambda_1 > \lambda_2$ . What is the connection between this and what we discussed here? What is the connection between this slide and this slide? Take a minute, I repeat stare at this slide see what is happening here, stare at this slide, what did we just now say and collectively we can say something in the next slide. This is the right time to pause and think about and guarantee yourself that you understand this slide as well as this slide.

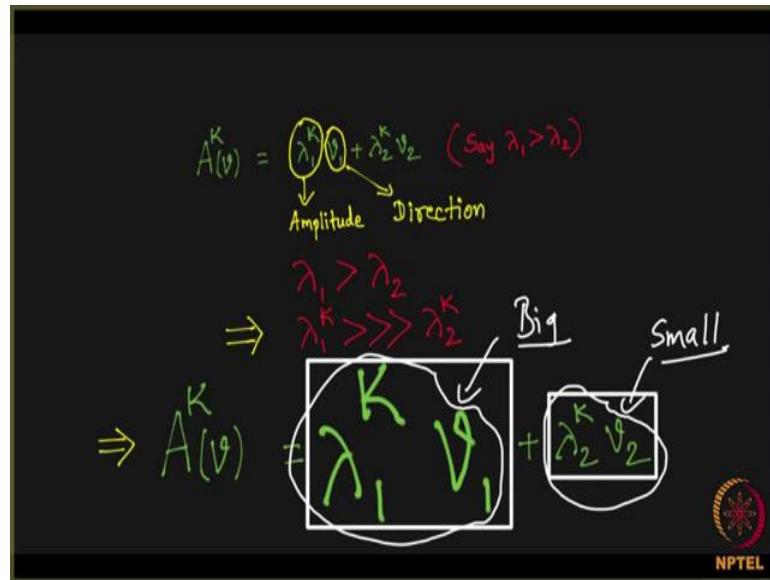
Now, let us go to the next slide if you are through with these 2 slides.

(Refer Slide Time: 12:49).



It is so big that if the ratio is 0, we saw that which implies that 3 to the 100 is several folds bigger than 2 to the 100 ok. I am just stating the same thing repeatedly. So, what? So, we can make a big inference now.

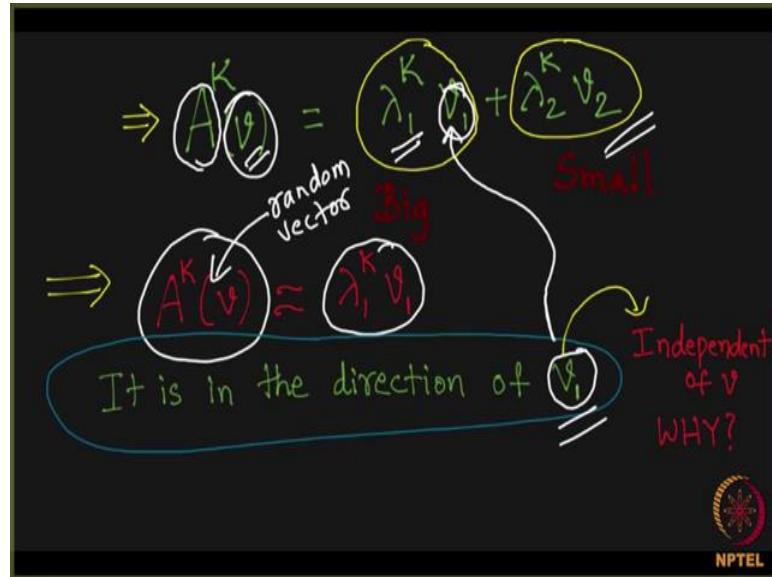
(Refer Slide Time: 13:10)



We observe that when you empower A when you repeatedly apply A on any random vector v, you can always write this as  $\lambda_1^k * v_1 + \lambda_2^k * v_2$ . What happens? This is amplitude and this is the direction we discussed that  $\lambda_1 > \lambda_2$  which implies lambda 1 to the k is very greater than  $\lambda_2^k$ ; if k is big. We saw 2 and 3 example and k was 100. It was huge, huge so much that the bigger one simply completely dominates or the smaller one so much that the smaller one is negligible in front of the bigger one so much so, that the ratio goes to 0 correct ok.

So, now this implies that  $A^k(v)$  is  $\lambda_1^k * v_1$  is a big quantity, why? That is because  $\lambda_1 > \lambda_2$  and this results in  $\lambda_1^k$  being very greater than  $\lambda_2^k$ . This is a bigger entity, this is too big, this is small. Small compared to what? Small in comparison to the big thing that is sitting here I am sorry it is not the  $v_1$  which is way; it is this entire this entire thing that is big is what I mean here when I say big. So, entire thing is small, small in amplitude is all I am saying ok. Let us go next.

(Refer Slide Time: 14:37).



So, we saw that  $A^k(v)$  is so much and this one is really huge. This one is small. When I say small, it is comparatively small right ok. So, now, when you take a big vector and add it to the small vector, what do you get? Do you recollect? Do you see the bells ringing in your mind? We saw prerequisite right. We saw that whenever a big vector is added to a small vector, it will simply be in the direction of the bigger vector right which means my  $A^k(v)$  will result in  $\lambda_1^k * v_1$  and you can simply ignore this factor here ok, perfect.

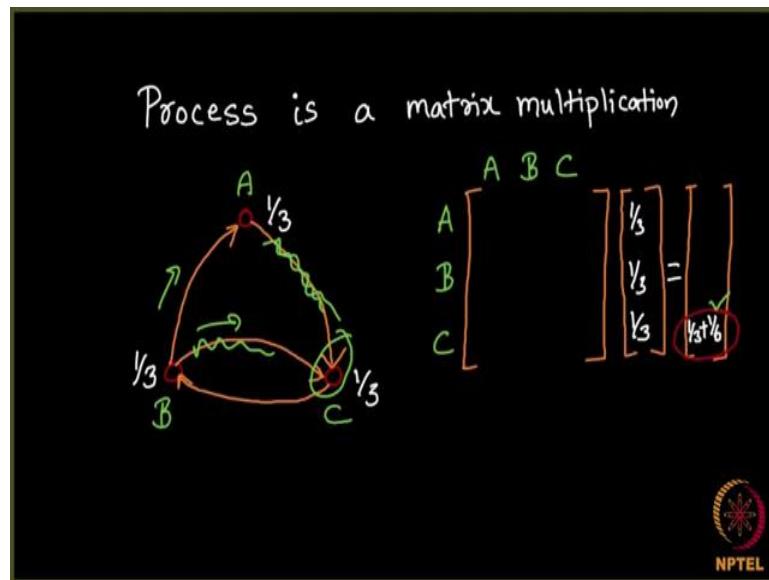
So, what do we conclude? We conclude, it is in the direction of  $v_1$  and please note this  $v$  was a random vector. Let me write that down. This is a most important observation, it was a random vector and no matter what you chose for  $v$ , no matter what you choose for  $v$  it was a random vector, no matter what we chose for  $v$ ; you always ended up with  $v_1$ . What is  $v_1$ ?  $v_1$  is the eigenvector corresponding to the highest Eigen value. If  $\lambda_1 > \lambda_2$ , then I take that corresponding Eigen vector, and this is independent of this  $v$  and it is something to do with the matrix that you take.

So, whatever vector you take, you always end up with the eigenvector. Do you see why? I just gave the explanation. Again this is the right time to pause and then understand what, I just now said and that completes the proof for the fact that whenever you take a matrix as screen cast if you can recollect, we did a screen cast of our programming where we took a matrix  $\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$  and applied it on different vectors. It was going to the same vector, why? This is the reason.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Link Analysis (Continued)**  
**Lecture - 113**  
**PageRank as a Matrix Operation**

(Refer Slide Time: 00:05)



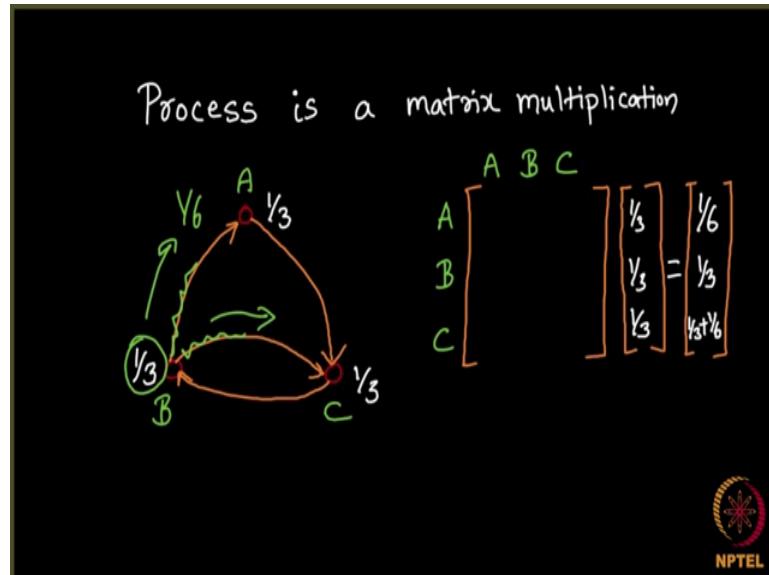
So, we saw a graph on three nodes right. I would like to see, if this process can be better understood by writing it as a matrix operation. So, that is my motive. This process is a matrix multiplication and we will see how exactly one can see this as matrix multiplication good.

So, now, let me try writing 3 cross 3 matrix like this. You will soon understand what the motivation is behind writing it as a matrix, but please bear with me as of now try to see what I am trying to do here ok. It is a very straightforward process. So, I will write down a matrix like this and then I will write down 1/3, 1/3, 1/3 and that is how we start that is where we start from as you can see it is 1/3 here, 1/3 here and 1/3 here correct.

So, we start as 1/3, 1/3, 1/3 and what I need is the next vector whatever I am going to get. And what is that? As you all know the value of A B C will respectively be so much, we start with C being 1/3 + 1/6. Why is that? That is because you have this C here, C has 2

incoming edges and from B C gets B gives half to A and half to C and we get 1/6 from C gets 1/6 from B and it gets 1/3 from A correct.

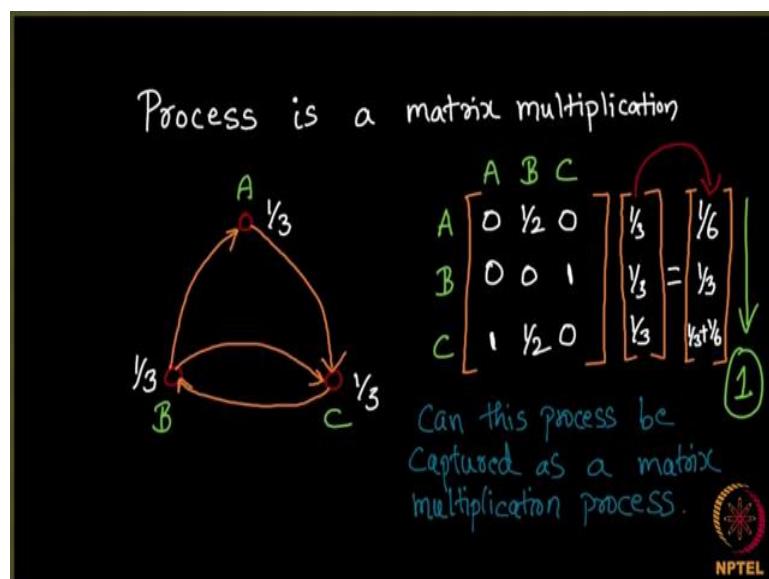
(Refer Slide Time: 01:59)



Now, what next? B gets 1/3 right; how is that? B gets 1/3 from C everything of what C has B gets

So, I am going to write 1/3 here. Next A gets 1/6, why is that? That is because from B you have to go just going and half is given to A and the other half is given to C and half is 1/3 happens to be 1/6. So, it is going to be 1 over 6 as you can see.

(Refer Slide Time: 02:39)



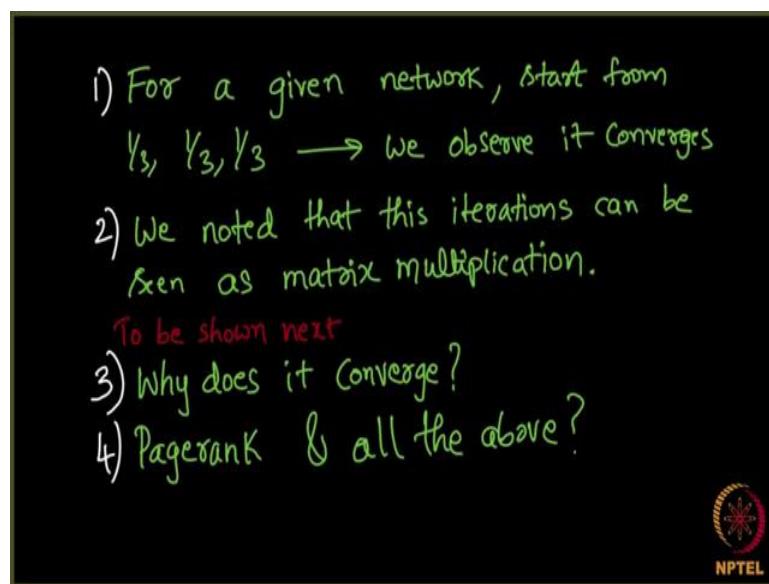
Next you observe that this converges to want this, this adds up to 1; not converges this adds up to 1 right. And my big question right now is; what is that matrix which when multiplied with this vector gives rise to this vector ok.

What is that matrix? Let us investigate it right. So now, I will write a matrix which will give me  $\frac{1}{3} \frac{1}{3} \frac{1}{3}$  as  $\frac{1}{6} \frac{1}{3}$  and  $\frac{1}{2}$ ;  $\frac{1}{3} + \frac{1}{6}$  as you can see is  $\frac{1}{2}$  all right. So, can this process be captured as a matrix multiplication process? That is my question all right. Yes, it can be is what you will observe very soon that is because this is the matrix which will give rise to this vector when you multiply this vector and this is my matrix, you can verify it.

So, take a minutes pause and see how exactly this matrix is giving rise to the next iteration. Let me go through it slowly so that you all understand. So, when you multiply 0 half and 0 to  $\frac{1}{3} \frac{1}{3}$  and  $\frac{1}{3}$ , basically this half gets multiplied which is  $\frac{1}{3}$  giving you 0.  $\frac{1}{2}$  gets multiplied with  $\frac{1}{3}$  giving you  $\frac{1}{6}$  and 0 gets multiplied with  $\frac{1}{3}$  giving you 0 right ok.

So, as you would have observed, this is precisely the process that is happening here correct that is precisely that is happening here. So, I am just capturing it as a matrix. It is a straightforward thing, you can observe what is happening here; take a couple of minutes and then observe it we will go to the next slide and we will see what is this to do with a page rank.

(Refer Slide Time: 04:43)



Now, what have we done so far? For a given network, we start from  $\frac{1}{3} \frac{1}{3} \frac{1}{3}$  and see if it converges or not. We observe that it converges. This is the first point. The second point is that we note that these iterations, these iterations that we are doing right now be matrix multiplication process correct; we just saw it right now. Coming next, we are going to show this next that why does it converge. Whatever we saw in the Google spreadsheet if you remember, it converges right. Why exactly does it converge? What is the process here right? And the fourth point is what is page rank to do with all these things ok.

We will see this one by one.

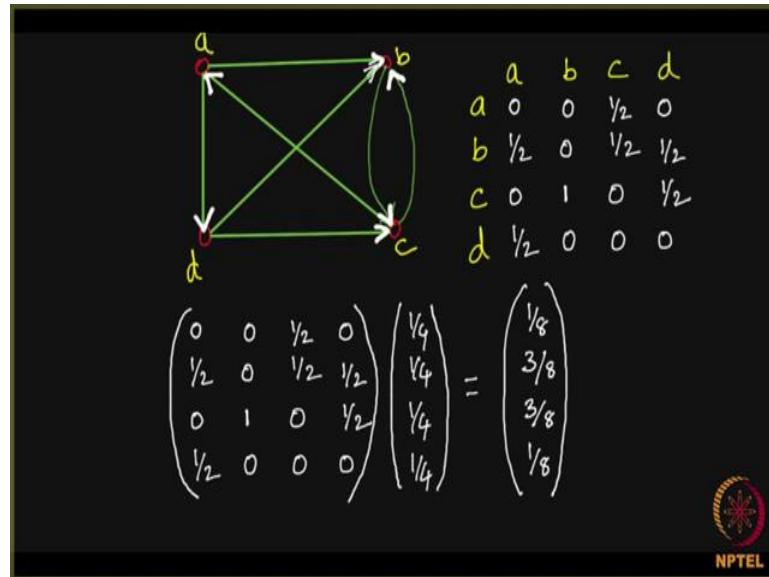
**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Link Analysis (Continued)**  
**Lecture - 114**  
**PageRank Explained**

So we are nearing the conclusion right now. As you can see, let us recollect the big question. What is the question? When you assign values, gold coins or resources whatever you call it to all the nodes and as you start sharing them, respecting the edges that leave and leave a vertex and then enter a vertex right a directed graph, you will observe that the values do indeed converge, right.

So, let us take it another example and then see it. We have already seen an example on 3 nodes, another example on 5 nodes. We will see it another example for clarity sake.

(Refer Slide Time: 00:49)



So, let us consider this graph on 4 nodes. This is the corresponding matrix; we have discussed this before right. Take a minutes' time pause the video and then check this is actually the matrix.

Now, let me write down this matrix as the matrix, then I start off with  $\frac{1}{4}$   $\frac{1}{4}$   $\frac{1}{4}$  and  $\frac{1}{4}$  and then I see where it takes this.  $\frac{1}{4}$   $\frac{1}{4}$   $\frac{1}{4}$   $\frac{1}{4}$  simply becomes  $\frac{1}{8}$ ,  $\frac{3}{8}$ ,  $\frac{3}{8}$  and  $\frac{1}{8}$  right.

(Refer Slide Time: 01:30)

$$\begin{pmatrix} 0 & 0 & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 1 & 0 & \frac{1}{2} \\ \frac{1}{2} & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{4} \\ \frac{1}{4} \\ \frac{1}{4} \\ \frac{1}{4} \end{pmatrix} = \begin{pmatrix} \frac{1}{8} \\ \frac{3}{8} \\ \frac{3}{8} \\ \frac{1}{8} \end{pmatrix}$$

a	$\frac{1}{4}$	$\frac{1}{8}$	.	.	.	.	$\frac{19}{100}$
b	$\frac{1}{4}$	$\frac{3}{8}$	.	.	.	.	$\frac{1}{3}$
c	$\frac{1}{4}$	$\frac{3}{8}$	.	.	.	.	$\frac{19}{50}$
d	$\frac{1}{4}$	$\frac{1}{8}$	.	.	.	.	$\frac{9}{100}$

QUIZ?

So, we know this right we saw this. Now let me write a table and then make a note of all the values here a b c d and then I started off with  $\frac{1}{4}$   $\frac{1}{4}$  and then I got this what is the next value. Where does it go like this? Where exactly does it converge?

So, here is a quiz question for you all. Take a minutes pause and then see that it indeed converges to  $\frac{19}{100}$   $\frac{1}{3}$   $\frac{19}{50}$   $\frac{9}{100}$ . In fact, you need not even take a look at these values; you should verify that you actually arrive at these values as you proceed with this matrix operation all right. Quick quiz for you all, it is good that you check it.

Now, as this goes further you know, this assignment is now what is called as stable assignment because it does not change anymore right. You are saying these to the nodes a b c and d, you assign it to a b c and d. Assign what? Assign these values and you will observe that there is no change anymore after this ok.

(Refer Slide Time: 02:44)

$$\begin{pmatrix} 0 & 0 & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 1 & 0 & \frac{1}{2} \\ \frac{1}{2} & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{4} \\ \frac{1}{4} \\ \frac{1}{4} \\ \frac{1}{4} \end{pmatrix} = \begin{pmatrix} \frac{1}{8} \\ \frac{3}{8} \\ \frac{3}{8} \\ \frac{1}{8} \end{pmatrix}$$

Observe  
 $A\alpha = \alpha$

Same Values

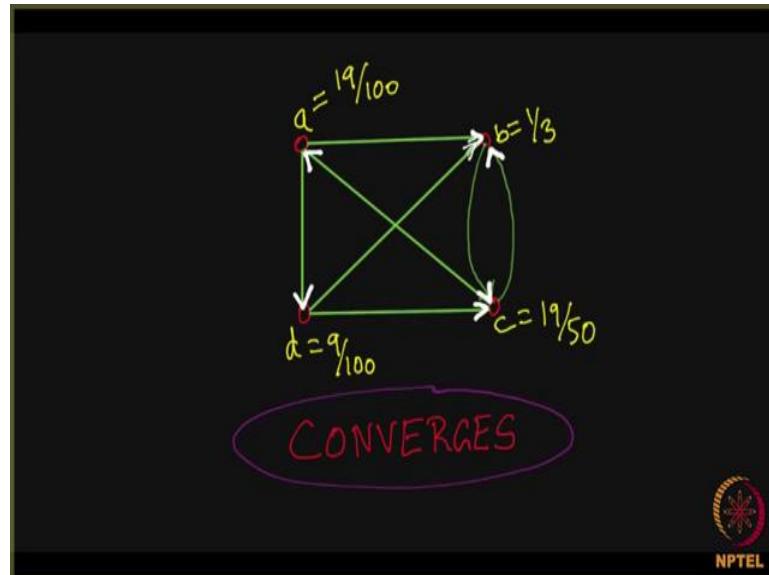
a	$\frac{1}{4}$	$\frac{1}{8}$	.	.	.	.	$\frac{19}{100}$
b	$\frac{1}{4}$	$\frac{3}{8}$	.	.	.	.	$\frac{1}{3}$
c	$\frac{1}{4}$	$\frac{3}{8}$	.	.	.	.	$\frac{19}{50}$
d	$\frac{1}{4}$	$\frac{1}{8}$	.	.	.	.	$\frac{9}{100}$

QUIZ?



So, it remains the same correct. So, note something. Here is the situation where  $\alpha$  becomes  $\alpha$  equal to  $\alpha$  and that is why the change does not happen or rather the change does not happen because,  $\alpha$  equals  $\alpha$ . It is all the same right. So, this is the matrix way of saying it right ok.

(Refer Slide Time: 03:09)



So, consider the graph. Once again and the assignment, you will observe that this is the assignment where values do not change. You can try doing it when  $a$  is assigned  $19/100$   $b$  is assigned  $1/3$  and  $c$  is assigned  $19/50$  and  $d$  is assigned  $9/100$ . The values do not change

after that and such a value such as assignment is unique. And we all have seen it already; I am sort of free iterating it right now in this lecture video.

(Refer Slide Time: 03:43)

Why is it converging?

Note

Such a matrix is called a Markov matrix.

Every column sums to 1

$$\begin{pmatrix} 0 & 0 & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 1 & 0 & \frac{1}{2} \\ \frac{1}{2} & 0 & 0 & 0 \end{pmatrix}$$

NPTEL

So, why does it converge? Why exactly is it converging? Now is not that obvious we have seen that enough. Take the matrix note that what is this, what do you understand by this, have you seen this matrix sometime before. If you are an engineering student, I do not think you would have passed by your semesters without noticing this matrix right.

So, any student who has finished his first four semesters of engineering would have definitely passed through a phase where he would have encountered this matrix. It is actually called the Markovian matrix right. So, observe the property here, every column sums to 1. Such a matrix is called a Markovian matrix or simply a Markov matrix.

(Refer Slide Time: 04:36)

Such a matrix  
is called a  
Markov matrix.

**Note**

Every column  
sums to 1

$\begin{pmatrix} 0 & 0 & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 1 & 0 & \frac{1}{2} \\ \frac{1}{2} & 0 & 0 & 0 \end{pmatrix}$

Has a very interesting property

Highest Eigen value is 1



So, this Markov matrix is a very interesting property. Its highest Eigen value is simply one which means every other Eigen value is less than 1 alright ok.

(Refer Slide Time: 04:50)

$$\begin{pmatrix} 0 & 0 & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 1 & 0 & \frac{1}{2} \\ \frac{1}{2} & 0 & 0 & 0 \end{pmatrix} \quad \begin{array}{l} \text{Has a very interesting property} \\ \text{Highest Eigen value is 1} \end{array}$$

$\Rightarrow A^k v = \lambda_1^k v_1 + \lambda_2^k v_2 + \dots + \lambda_n^k v_n$

Less than 1

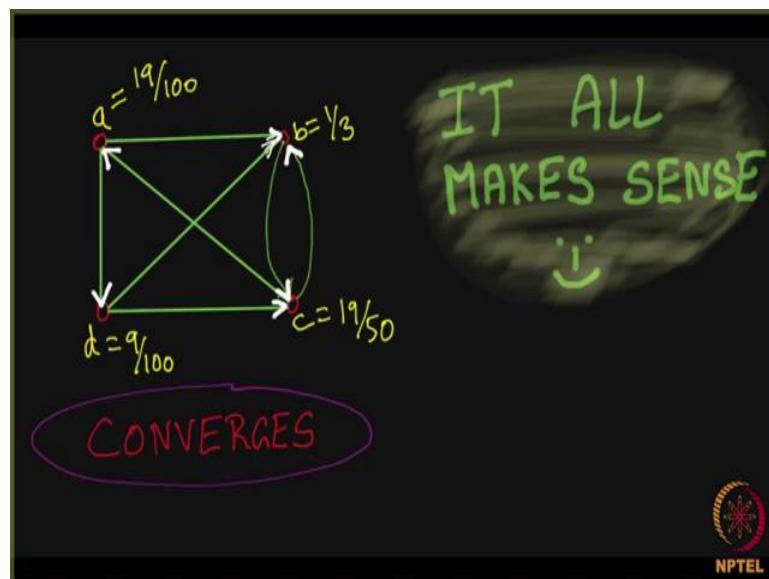
$\Rightarrow A^k v = v$ ,  $\Rightarrow$  Eigen vector corresponding to 1

So, do you see what happens here? Let us go slowly.  $A^k(v)$  of 3 is simply this where  $v_1, v_2, v_n$  are all eigenvectors and  $\lambda_1$  etcetera are all the corresponding Eigen values good. So, we can always write A's action on v can be captured with this. So, we have seen this before I am repeating it ok.

Next  $\lambda_1$  is 1 here because that is the highest Eigen value. As I told you that is the property of such a matrix and the other Eigen values are less than this. Now, what does it mean? Now you would have observed a number which is  $1^k$  will make it one itself right. Other numbers are less than 1 here. All these are less than 1 here and when you raise it to the power of k, all of them become 0, right. All of them variably become 0. So,  $A^k$  will simply be equal to  $\lambda_1^k * v_1$  mind you  $\lambda_1$  is 1. So, this is the eigenvector corresponding to the highest Eigen value which is 1 and let me just remove  $\lambda_1$  here because it is equal to 1.

So, what do we observe? We observe  $A^k(v)$  converges to the Eigen vector corresponding to 1, perfect.

(Refer Slide Time: 06:33)



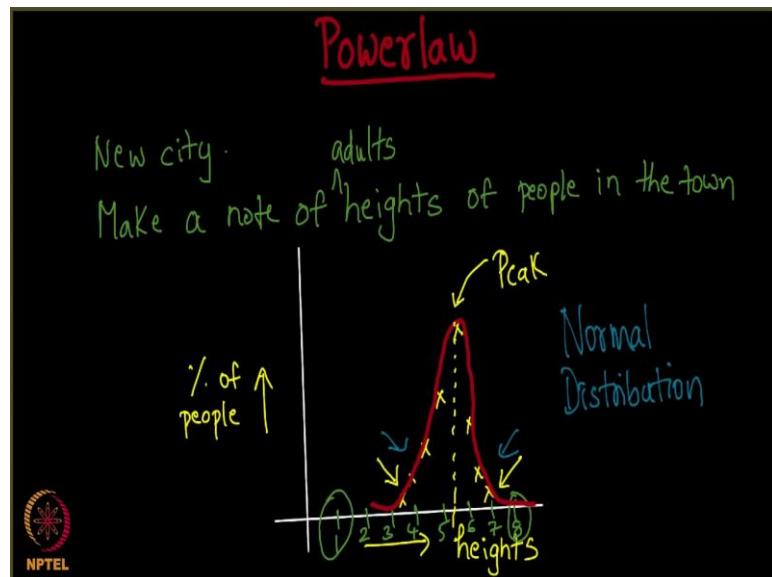
That is pretty much here we have concluded. So, this was the graph that you took. You saw this was a sort of a stable assignment which means it converges at this value and now it all makes sense as you can see, right. So, such a value assignment is unique and no matter where you start from you always converges to this. So, that is the best part.

So, that sort of concludes good amount of discussion on page rank. We have revisited page rank from our previous lectures about some 20 lectures ago, a couple of weeks ago roughly let us say 3 to 4 hours videos before we saw page rank very at a very conceptual level, we did not see the math part of it, we have now seen the math part of it. I hope it is clear to all of you. It is important for you all to look at it may be re-watch the videos because it takes time to sink in especially the mathematics component of a page rank.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**  
**Rich Get Richer Phenomenon**

**Lecture – 115**  
**Introduction to Power law**

(Refer Slide Time: 00:08)



So, we are now starting a brand-new chapter called the Power Law. So, it is difficult to; difficult on your minds if I start talking about power law directly. So, as and always it has been customary, that we start with an example right with a nice story rather ok. So, what I will do is I will now ask you all this question. Assume if I go to a new city and I make a note of everybody's heights of everyone in the town. I make a note of every ones height of people in the town. I come back and then I look at this data. And I try plotting the following very simple graph. The graph will be this all my x axis, I will have heights; on my y axis I will have percentage of people with that height.

Now, let me illustrate this with a numeric value. So, what I do is let us say this is 1, this is 2, this is 3; 1, 2, 3, 4, 5, 6, 7. So, I forgot to mention that I am making note of only height of adults in the town ok. As you would have guessed they there is no reason why we should take one foot 1 foot or let us say 8 and above, we do not know of anyone whose of this height adults especially.

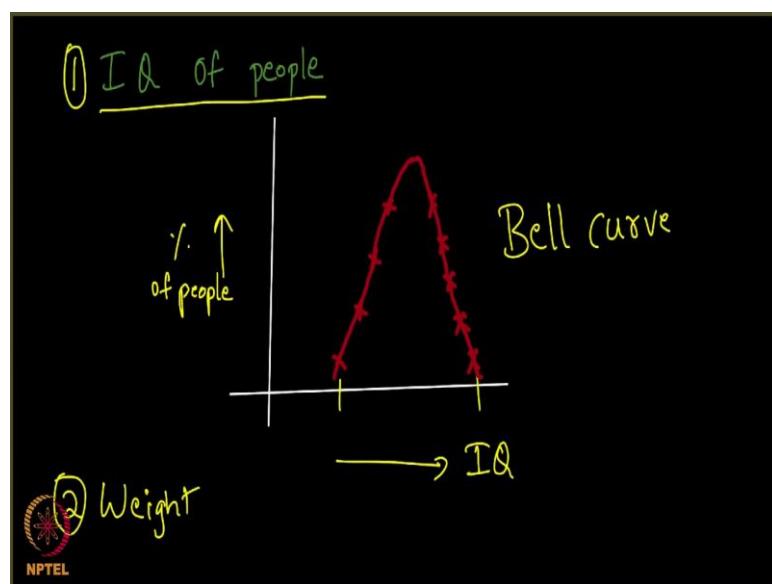
So, what I do is this plot mainly goes like this. I look at the percentage of people who are 5 feet tall, percentage of people who are 6 feet tall. And most of the people as you would have guessed will be between 5 and 6. So, around 5.3, 5.4 will be the peak. There will be very less people with 6 and over very very few people who will be 7 feet tall, very unlikely in many countries although you will find tall people; it is very country specific. You might find all people in different countries, but most the countries will not have very tall people above 7 feet.

So, they are going to be very very less. And people just above four are going to be less; people who are 4 feet are very very less. And this is how your plot will look like as you would have guessed ok. It goes up and comes down like this. So, what do we observe? We observe that there are very few people with very less height. And very few people with who are very very tall and most the people are between this ok.

Let us say between 5 and 6 feet correct. This is varied peaks; such a distribution is called a normal distribution; this is called a normal distribution. What do you? What do you mean by normal distribution? The distribution where it is less on the left and right, and it peaks somewhere in the middle right ok.

So, we can see many things in life which has normal distribution things.

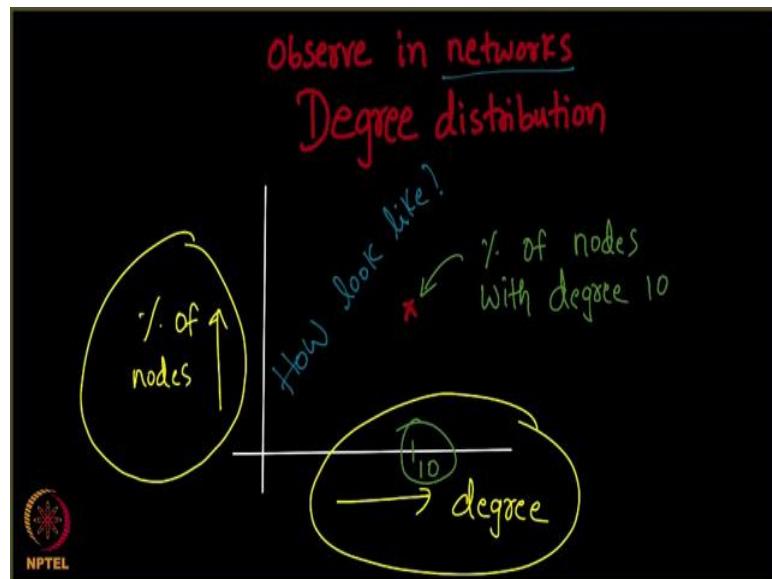
(Refer Slide Time: 03:57)



For example, let us say IQ of people; IQ of people in a classroom; you will see you will see if I were to plot on the x and y axis. As and always as I told you this is going to be my in this example IQ of people. And y axis will be percentage of people with that IQ right. So, let us say this is slightly on the lower side; this is slightly on the higher side; you will find very few people with very very low IQ; meaning below average very few people with extremely high IQ. And most of them will sort of lie in the middle, and if you actually try plotting values for everything; you will indeed get a curve like this. And this curve in statistics is called the Bell curve. It resembles the shape of the bell that's why it is called the bell curve. In fact, it is easy to capture it mathematically as well although we will not discuss it right now.

So, IQ of people forms a bell curve you can think of many such examples right; weight of people in a town, weight is another such example, IQ being one such example, height being another example right. So, let us look at an example which is very specific to our work to our subject and that is going to be next.

(Refer Slide Time: 05:35)

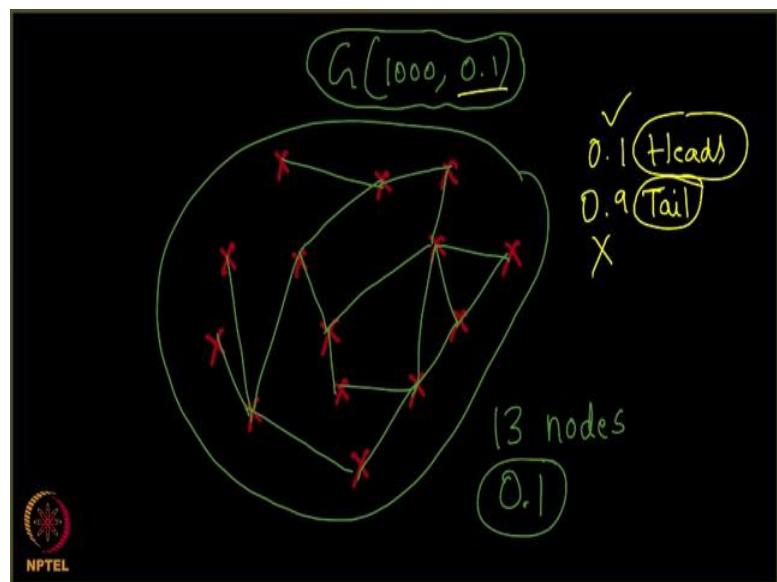


So, if I what do we observe, what do we observe? If we were to take a network and try to ask the degree distribution in the network; degree distribution what do you even mean by this? By degree distribution I mean again x and y axis ok. This is my x axis and my x axis. How long my x axis will be degree of nodes. And along my y axis will be percentage of nodes with that degree; with that I mean if you have something like 10

here; just this point will denote the let me just write the downs its seems into your minds. This denotes the percentage of nodes with degree 10, degree 10.

And big question right now is, how does this plot look like? How does this plot look like? For that we should first take a network, now here is a good the chance for you all to write a piece of code and observe the following.

(Refer Slide Time: 07:03)



Let me take you all should probably consider a graph with 1000 nodes. And with probability let us say point 1 you put edges between them. What do you mean by this? By this you mean let us say we had 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13 nodes something like this.

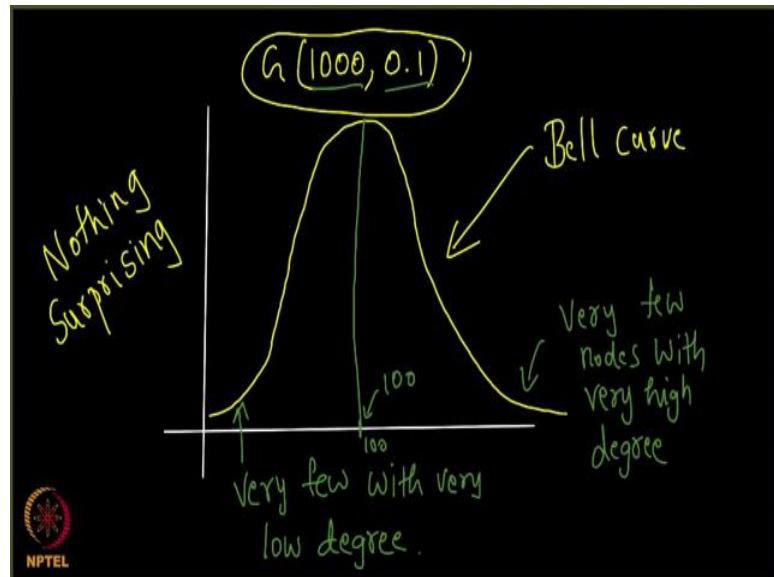
What you do is you will put edges by considering the probability 0.1. What do what do I mean by probability 0.1? We have discussed the set length before, but I will just recalculate. You toss a coin with head and tail probability 0.1; heads probability and 0.9 will be your tail probability right.

If you get a head, you put an edge; if you get a tail, you do not put an edge that is how I am continuing this right. Assume I got a network something like this right. This will be my final network. And this is my graph G with probability, how many how many nodes are here? 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13 nodes, this is graph 13 nodes and edge

probability is 0.1 which means I toss a coin with probability of it being heads is 0.1 and tail being 0.9, I put an edge if I get a head I do not put an edge; if it is a tail.

I repeat this process for every possible edge on this 13 node graph ok. I showed you a simple example, but what you should be doing is do it for a 1000 node graph right.

(Refer Slide Time: 08:56)



Do it for a 1000 node graph. Once you finish programming, you get 1000 node graph. And put edges with probability 0.1. This is called a  $G(1000, 0.1)$ . And what should you do? As you would have guessed compute it is degree distribution. We discussed in our, previous slide; what is the degree distribution. You put degree on the x axis and you put the percentage of nodes on the y axis, and see how this plots looks like.

You will be started to see, that this plot will resemble a bell curve once again with very few nodes, very few nodes with very high degree with very high degree. At the same time, you will see very few nodes, very few nodes with very low degree. And again, it will peak somewhere in the middle.

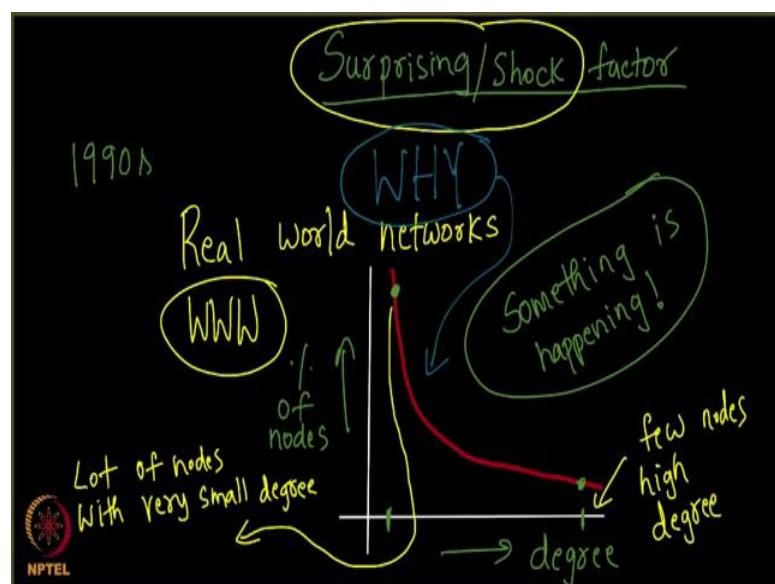
If you can guess, it peaks around 100 for x being 100. It peaks you can probably tell me why it is the product of 1000 and 0.1. It is lift as a it is left as a quiz for you all to say why does it peak at 100. Anyways, the moral the story is simple. When we draw out a graph with 1000 nodes and 0.1 edge probability, we will see again a bell curve. You

probably you would tell me yeah, we saw it in the previous cases, we saw many examples right it is not at all surprising for us right.

So, IQ of people you saw, you saw, that the degree of nodes follow. What is called the normal distribution? And everything else looks similar right. If you look at weights of people, height of people, IQ of people all of them seem to follow this bell curve only nothing surprising about it right. So, let me write that down there is nothing surprising it nothing surprising.

Then in the late 1990s scientist observed something that is actually very surprising.

(Refer Slide Time: 11:31)



So, what do you mean by the very word surprise? Surprise is something that happens unexpectedly. You think that this is how it should be, but nature has something else in store. You see something else happening where what you thought should have happened is not what is happening there. And that is what creates a surprising or what is called a shock factor. In the late 1990's network scientist observed a very nice phenomenon. They took at the, they looked at several real-world networks. So, until 1990's people did not have big networks. That was because we did not have computational ability back then. It is only in the 1880's and 1990's that we started collecting a lot of data and we had data sets we had computers which would crunch the data sets.

So, let us say they if you consider the network of the World Wide Web. It is a few millions of nodes. People ask the same question. What will be the distribution of degree on these networks? And they all expected it will be a bell curve. And as you would have guessed, there was a surprise stroke shock factor there and the fact was the following. They observed that it was no way close to the bell curve. The curve looked something like this.

There was a drop; it was not a bell like this. It was not like this; it was a drop like this. So, once again what is the x and y axis? x axis is the degree and y axis is the percentage of nodes that have so much degree. So, people observed that there is a drop here. Why is it happening right? So, if let us say you try drinking milk tomorrow morning and milk taste salty let us say; you will actually start wondering what went wrong, why is it why is my milk tasting, salty of all things right; something should I gone wrong right, something should be there is some background information that you are missing.

So, similarly when scientist observed that drop of the curve like this which is nowhere close to the bell curve, people thought something must be happening here that we do not seem to understand. So, let we write that now something is happening here that we do not understand. So, what could be happening why exactly is this observed. What is this even mean?

You see there are a lot of nodes with very low degree and very few nodes right; very few nodes with very high degree that is what it means right. Few nodes with high degree and then let me write this here lot of nodes with very small degree. And this startle the community of network scientist. They started wondering why this a property is being seen, what is making this property emerge.

Now, is the right time for you all to pause the video and think about the question. Why is this happening the big why? Why is it that we are not able to see a normal distribution here? Why are we seeing a drop in the curve like this?

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**  
**Rich Get Richer Phenomenon**

**Lecture – 116**  
**Why do Normal Distributions Appear?**

(Refer Slide Time: 00:11)



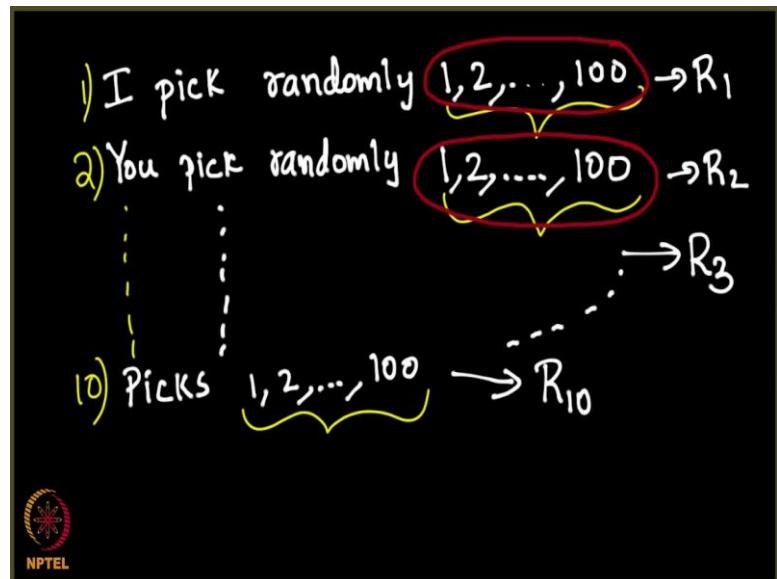
So, we saw two things. One was the normal distribution right. We saw basically two plots. So, far one is that I told you people there is this normal distribution, the inverted curve and then a drop like this ok. So, the definition the example heights of people in a town and this happens to be the degree distribution of nodes in a real-world network you know real world; we see this ok.

Now, before asking why do we see a very different kind of a plot here? Why in most of the situations, we see this kind of a plot. Before asking this question, we would like to ask individually, why and how does this come about? This one same time why and how of this as well.

So, to begin with let me ask the why and how of this normal distribution. How does normal distribution come by ok? Let us take a look at a nice example. And try to motivate the reason behind why the normal distribution appears in nature ok. Here is an

example you should invite you all to think a little about it ok. It is not very difficult to understand it is pretty straight forward ok.

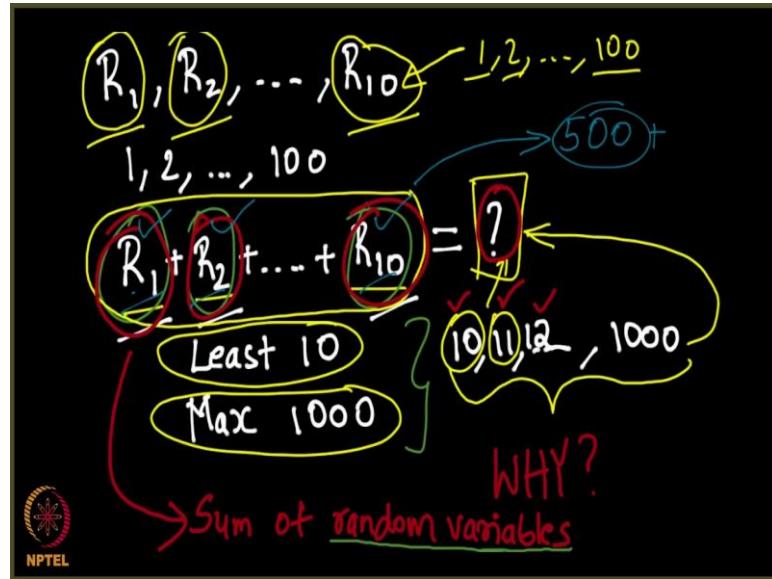
(Refer Slide Time: 02:01)



So, assume I pick a random number; randomly I pick randomly a number from 1 to 100. One of these numbers I pick someone number ok. And then you as another person, you also pick randomly some number from again from the same set 1 to 100. Some number you will pick randomly here all right good. Now see when I pick a number randomly here from 1 to 100, I can pick any number; it could be a 1 it could be a 2, it could be a 3, it could be a 50, 60, 73 89 100. Whatever I want when you pick you as well will have the liberty to pick any number.

Let us assume 10 such people pick. I am the first person; you are second person so on and so forth; some 10 people pick like this ok. Tenth person also picks a number from 1, 2, 100 right. So, what I, now what I am now going to do is let us say the number here that I pick let me call it  $R_1$ . The number you pick I call it  $R_2$ , the third person picks a number I am going to call it  $R_3$  and so on. And the 10 person who picks the number that number is  $R_{10}$  ok.

(Refer Slide Time: 03:52)

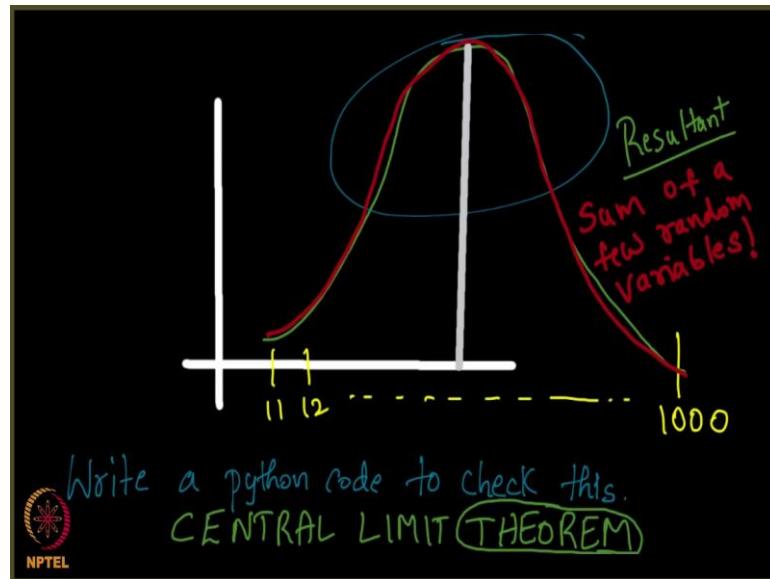


So, let me go to the next slide, let me write down. There are 10 random numbers each one between 1, 2, and 100. Each of these random numbers are from 1, 2, 100. Here is an important question while each of these random numbers can be anything from 1, to 100; just in case I added them look at this just in case I added all these, what will this be? See each one of these can be a number from 1 to 100 which means the sum will be at the least it will be 1 plus 1 plus 1 10 times. Assume all them shows a 1, 1, 1 each then at least it will be 10 and a maximum of how much.

If each one of them picked 100 it will be 100, 100, 100, 10 times. It will be 1000. So, this is the least and this is the maximum the minimum and maximum, but then, but then observe here that any number 10, 11, 12 up to 1000 can appear for what for the sum correct, but here is the climax. While whatever numbers you pick can be any number between 1, 2, and 100 when you take these ten numbers and add them, what you get as an answer is not uniformly at random from these numbers which means 10 and 11 do not appear uniformly at random here. While let us say in R<sub>1</sub> and 2 and 3 and what not up to 100 appears randomly. Each one of them is has the same chance of appearing here, but then the chances of 10 appearing here is less than the chances of 11 appearing here is less than the chances of let us say 12 appearing here, here in this place. And why is that?

If you can think; you will conclude that this is indeed once again a bell curve right.

(Refer Slide Time: 06:25)



It is going to be if I were to plot the values; let us say the least what 11 12 so, on up to it will go on and on and up to let us say 1000 is the maximum. You will observe that uniform. So, let me go back to the previous slide. So, each one of them each one of R 1 can be on an average let us say anything from 1 to 100 so on and so forth. So, this will be roughly; so, this will be like this the plot will look like this right. It is good to write a programming and then check.

Let me assign that to you the assignment to you all to write is to write, a piece of python code; write a python code to check this. You will be surprised to see that it is actually a very beautiful curve like this ok, to check this all right. So, why it is this happen that is very obvious. It is very unlikely that you pick a 1 and a 1 and a 1 for all of them correct. It is equally unlikely that you pick a 100 and a 100 and a 100 for all of them correct.

So, what is more probable as a sum  $i$  if you if you have guessed it right it is roughly around 500, 550 but roughly around 500 plus right. That will go, that will be free; that is the peak here, rapped that the peak here is 550 and completely the peak here is going to be 550 that is the peak ok. Anyway, you do not worry much about it. All you need to know is that you will see a normal distribution here. Now why is that? Why does it why what exactly is the moral of the story?

The moral of the story is the sum of random variables that you see here. The sum of random variables they are basically a bunch of a sum entities that are random. They are

called the random variables, sum of random variables. And this is slightly it is the term that would not have really heard of well unless you have done a course in statistics, but just see random variables as simply random entities which can take any values all right. Now whenever, the resultant this; a resultant whenever the resultant is a sum of several random variables; whenever the resultant is a sum of a few random variables several few random entities random variables, then you will see a normal distribution like this.

You will see this distribution whenever the resultant is a sum of few random variables. This goes by the name this is actually a very well-known theorem in statistics rather probability. It is called the of central limit theorem. So, I have stated in a very in a really easiest possible way, you do not worry much about the word theorem here. It is sort of scary for many of you people who do not like math much, but the all I am saying here is that the sum of random variables right.

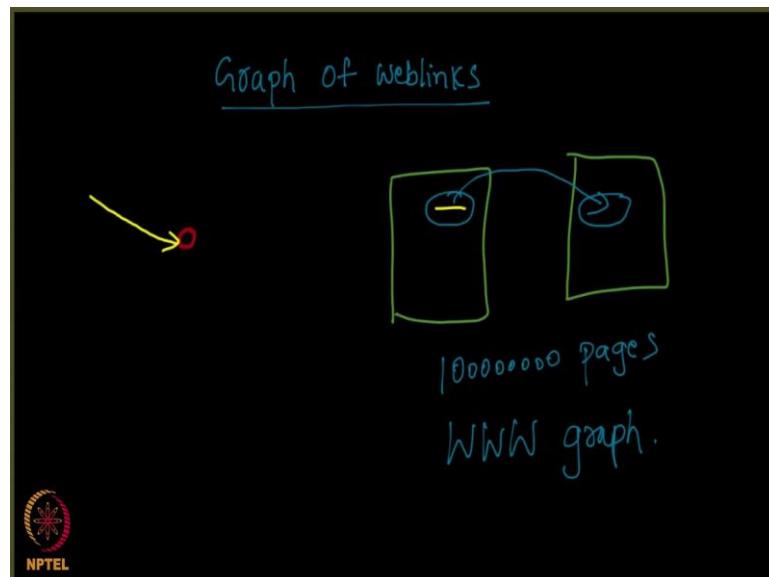
This sum of random variables is this resultant and that resultant is generally a bell curve if its sum of few random variables. And that is how, this emerges. This always emerges when the resultant is a sum of random variables this. In fact, true in nature whenever you see an experiment. The sum will basically the resultant that you are plotting will basically be a bunch of random events put together right ok.

So, let me go to the new slide and let me ask the next question that the next question is in the first slide itself. that I question is this. How does this emerge? What is the reason behind the emergence of this thing? We gave some reason for the left plot this one. So, what is the reason for this one is what we will be covering in the next lecture.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**  
**Rich Get Richer Phenomenon**

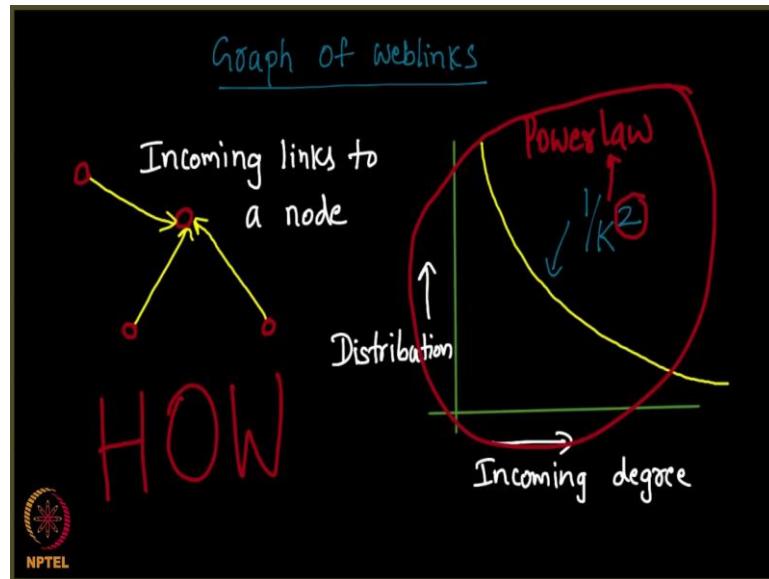
**Lecture – 117**  
**Power Law emerges in WWW graphs**

(Refer Slide Time: 00:05)



So, remember that the graphs that we discussed before. If you do not remember, let me revise it quickly for you all. A web graph is basically a graph with a node ok. This node represents let us say a page like this a page pointing to another page right. So, here is a page which has a hyperlink to another page. You click on this, you are taken to a new page right and I and internet has millions and millions of such pages right; several such a page, a whole lot of such pages correct. And then I look at the underlying graph and that is called the web graph, the graph of the World Wide Web.

(Refer Slide Time: 01:03)

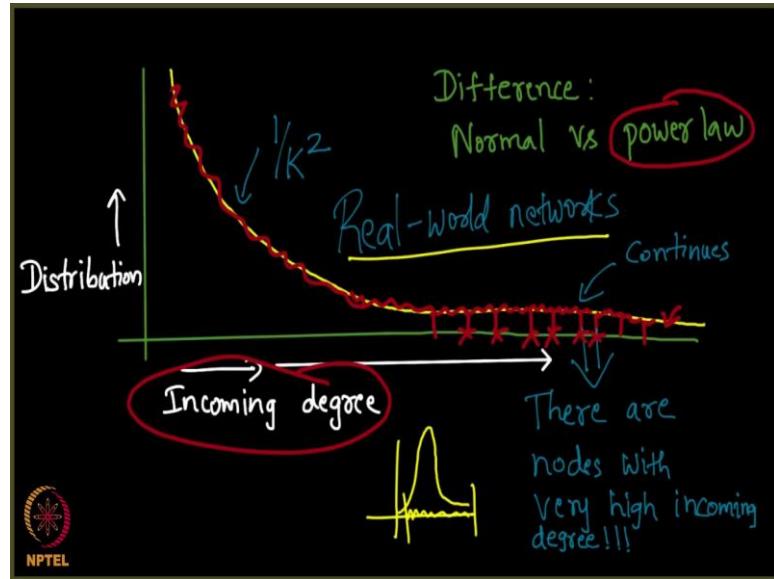


So, I am going to consider that right now ok. So, typically a node represents a page, this represents a page. And an incoming link to it represents a link that is pointing to this page as simple as that right ok. So, there may be many such incoming links to a webpage from some other page. By that time in if I put a link to your home page from my home page then, I put an edge from my home page to your home page. Home pages are all nodes here and if there is a link from one page to the other, you put a directed edge right. These are all the web pages from where there is a link put to this page.

So, these are the incoming links to a given right. Here the incoming link is incoming degree is 3 right; that is called the degree. Now, let me plot on x and y axis the following. On x axis, I will write the incoming degree and on y axis I will draw the distribution right and I will observe what is happening here. I observe that there is a drop like this. Unlike in the previous case, we do not see a normal distribution here and in fact, a closer observation even tells me that this plot very much resembles  $1/K^2$  where K is the x axis. It resembles this right. And this is called the power law right; the law which states that some distributions do not exhibit normal behaviour. You do not see a bell curve there; you see a drop like this ok.

So, the exponent here is to, it can be anything for what we now it can be anything. But in this case in the case of the web links, it turns out to be  $K^2$  ok.

(Refer Slide Time: 03:09)



So, what are we observed so far? When you plotted the incoming degree versus the distribution right, you got this for web graphs. But then when you would you this is 1 over K square, but then there is a difference. What is the difference? Normal versus power law.

What is the difference? See, initially you saw we plotted two graphs. One was normal, one was power law right. What is the huge difference between these two things? What does power law tell us which normal distribution did not? We see this in fact, this actually extends right. The this is the yellow line here, let me show that you once again; this actually extends, it goes beyond; for a for a long time you will see a curve like this ok.

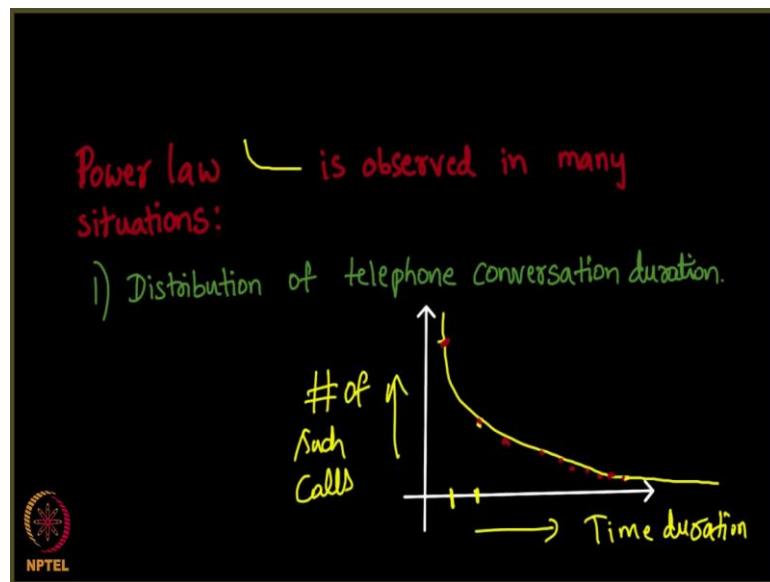
So, let me just extend this green line also. And what is this signify? What is this band here? What is this band here? What is it denote? Please look at the x and y axis, its rather self explanatory. What does it denote? It denotes something very significant, observe carefully. It denotes that this continues this goes on; goes on means what? It means, there are nodes with very high incoming degree Do not you think that is what this means? Look at this side this x axis, x axis means what?

We just note it, the incoming degree. The incoming degree is very high here as it goes on the right side of the x axis and the yellow line denotes that there are nodes; there are nodes with very high degree. That was not the case in normal distribution as you

remember correct. The power laws simply say that the curve goes a extends beyond a stage. And it sort of is close to the x axis although it does not become equal to x axis; it goes on which means the it shows of shows us of the existence of nodes that have very high incoming degree right, fine. Here is the question. Why do you observe this drop of the curve? And secondly, how come there are so many nodes with higher and higher degree is observed in real world networks?

I hope you remember the in the previous lecture. One of the previous two lectures, we observed that if you plot it for let us say random graph which is a GNP model, with some 1000 nodes and probability 0.1. You observe that it was a quick drop. How was the graph? Remember the graph was something like this that is all. After this there was no extension, before this there is no value no node which satisfies this. It is all it all lies between this and this, but in case of real-world networks we see an its sort of extends. Firstly, it drops it drops and extends right. Both funny and strange; why is this happening will be the crux of this chapter; and the crux of discussions in the forthcoming lectures. Let us go ahead ok.

(Refer Slide Time: 06:39).



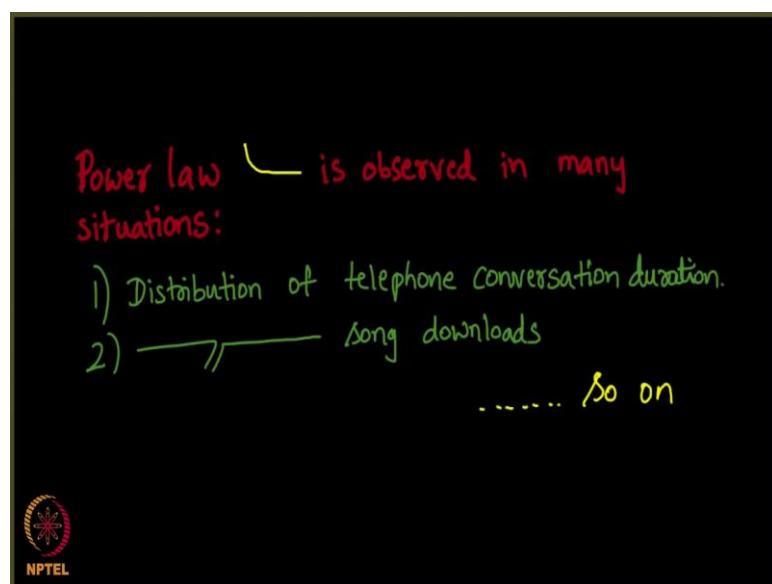
So, it has a very high degrees what we observe remember that. There are many nodes with very high degree it goes on and on. This is called the power law. Power law is basically the drop of the plot the distributional drops ok. And that, that resembles a  $1/k^2$  or  $k^3$  or in general  $1/k^\alpha$ . Let us say all right.

So, the power law is a drop, it is observed in many situations, not just in the in. What did we observed in the previous case? We saw the distribution of let us say the incoming nodes of web graph. And we observe it is a power law then, immediate next question you may want to ask is where exactly I see power law apart from these web graphs. In fact, you see them in many situations. You will see them in telephonic conversations let me go back so, that I can show them to you one by one. So, you will see this distribution, this kind of distribution is observed in the telephonic conversation.

For example, what does this means? This means you again plot x and y axis. On x axis you plot the time duration of phone calls and on y axis you plot the number of such phone calls or percentage of such phone calls. Here both are one and the same. And you will observe that it is again I drop like this ok. So, what do I mean by this? How many phone calls have been of 5 minutes duration of whole lot of them? How many of them have been 10-minute duration of whole lot? But relatively less compared to what used to happen for a 5-minute phone call duration. That is a whole lot as you can see 10 minutes is less 15 minutes is a lot less so on and so forth right.

And you do find this extension; you do find there are phone calls which are of a high time duration right. This distribution is also a power law very surprisingly ok. Where else do we see power law? We see that in many places. In fact, this data sets are available online, you can just check.

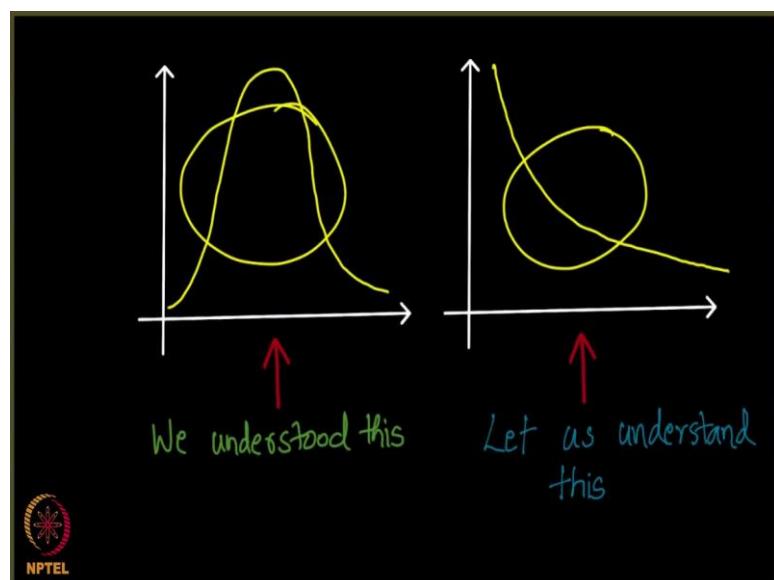
(Refer Slide Time: 09:04)



So, where again do we see the distribution of song downloads right? What are the numbers of songs? What are the songs which are downloaded more than 10000 times from our website? There are some official a song downloads website right.

So, if you go and look at this even these shows a power law ok. This was a exponential drop lines ok. So, which means power law is observed in many places. While we think, it should have been a bell curve in place of bell curve. There is a drop like this ok. So, there are many such networks you can take a look at it. In fact, there are hundreds known till date or may be even more right. I know at least couple of dozens of a very well-known network that exhibit power law. And people are even looked at why exactly they are exhibit power law which will be the focus of our next lecture.

(Refer Slide Time: 10:02)



So, now we observe that there is a normal distribution. There is a power law distribution right. So, this is this we completely understood right. We understood it completely why was it? Remember I took random variables and I showed you that the sum of random variables is what results in a curve like this right. There is nothing much to explain there and physicists, chemists, biologists you name it people are observed this normal distribution extensively. In statistics this makes the crux of a many observations and they analyse it that is all very nicely neatly and the math of it is completely understood and it is also very nice and elegant. You may have to look at it in case you are interested ok.

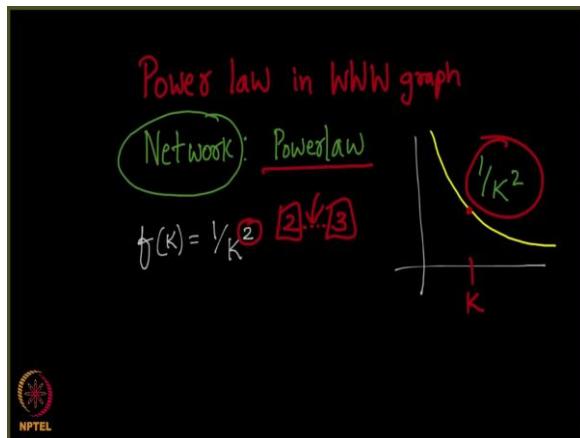
And what we need to understand is we understood this; what we need to understand is this now, right. How does this combined? Let us understand this.

So, our main focus that is coming next is we look at this power law networks which is the web graphs and we will ask this question how did this happen and why did this come here, what exactly made this curve appear like this. While we were actually expecting, what were we expected? You are expecting something like this right; something like this we were expecting, we ended up getting something like this ok. We will see more of it in the next lecture.

Social Networks  
Prof. S. R. S. Iyengar  
Department of Computer Science  
Indian Institute of Technology, Ropar

Rich Get Richer Phenomenon  
Lecture - 118  
Detecting the Presence of Powerlaw

(Refer Slide Time: 00:06)



So, we saw in the previous lecture that Powerlaw emerges right. We observe powerlaw in the network of weblets right called the WWW graph the web graph, we observe this we saw this ok. Now, assume you have a network alright. So, before going any further ~~ah~~ I am going to address this question on how to detect power law in given network. Assume you had a network you are given a network and you have this question in mind. The question is in is this network exhibiting powerlaw right. If so, then what exactly is the exponent. So, x and y axis and if you can see something like this what exactly is the exponent of this.

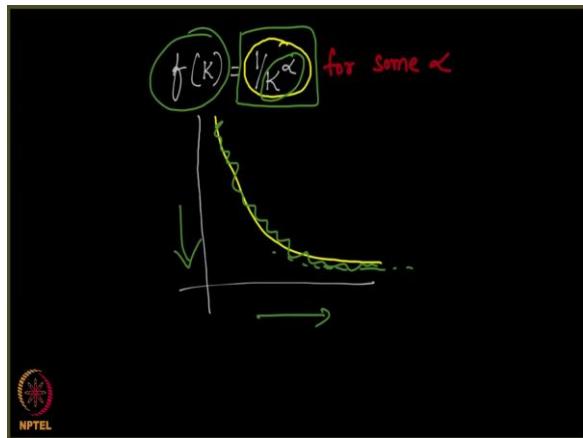
I told you that the exponent in case of the web graph happens to be 1 over K square 1/K<sup>2</sup> right. So, how will you find out whether your network that you are looking at exhibits powerlaw or not. So, here is a very easy litmus test that you all can quickly observe and sort of node your head saying it is nothing, but obvious. This is basically as I told you this is your f(x) rather f of Kf(K), if this is K this becomes f of Kf(K) which is 1 over

Formatted: Superscript

$K^{-\alpha}$ . So, let me note that down and say  $f_{\text{of}} K f(K)$  is equal to  $1/K^{\alpha}$ .

By the way this exponent here needed necessarily be 2, it is observed to be between 3 and 3, in case of some networks it is 3, in case of some networks it is 2 and so on right. In some networks it is greater than 2 and less than 3, it is somewhere in the middle right in many cases. I would like to find out if a network is exhibiting powerlaw or not.

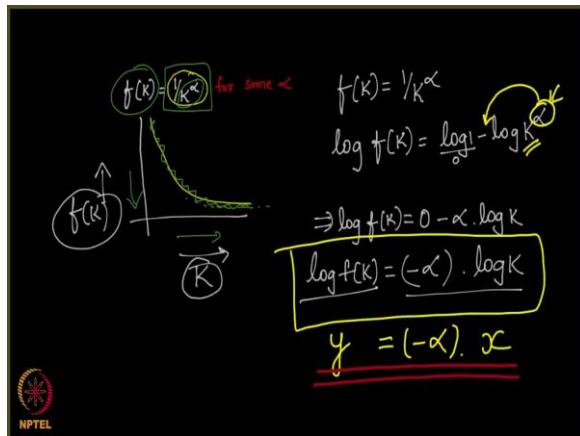
(Refer Slide Time: 02:28)



So, how do I do that? It is fairly straight forward a process. If  $f_{\text{of}} K f(K)$  is equal to  $1/K^{\alpha}$  for some  $\alpha$ , I do not know what that  $\alpha$  is for some  $\alpha$ . Let us say some alpha how do I find this. So, I have a plot given remember you can plot the distribution and you observe that it resembles something like this. And, you know it is something like  $1/K^{\alpha}$ , but you do not know for sure whether it is a reciprocal of a polynomial like this  $K^{\alpha}$  or something.

So, what you do is you, what does this curve observe this curve this curve is nothing else, but your  $f_{\text{of}} K f(K)$ . And, you ask the question is it of the form  $1/K^{\alpha}$ . As you see as the denominator increases the value sort of decreases that is why it becomes small right ok. How do I see this? I write so, let me make some space from myself here how do I see this?

(Refer Slide Time: 03:35)

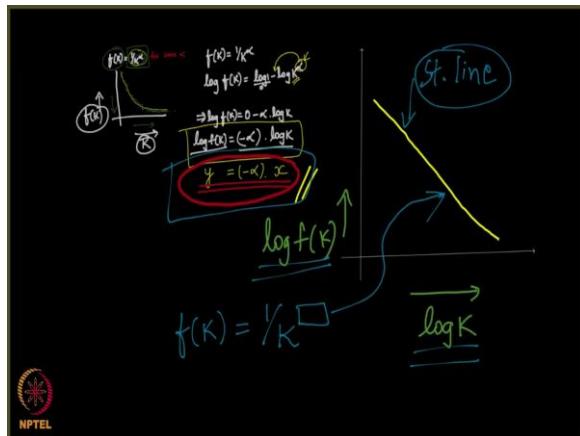


Let me write on this side. So, how do I see this? Given that f of Kf(K) is equal to 1/K<sup>a</sup>, over K to the alpha wWhat I do is I apply log on both sides very straight forward math, I am sure I am not using anything beyond let us say high school level math which is equal to log 1 minus\_ log K<sup>a</sup> K to the alpha. So, basics of logarithms I am sure all of you are familiar this simply becomes log of f of Kf(K) is equal to we all know log of 1 is 0 0 right.

This becomes 0 minus\_ log K<sup>a</sup> K to the alpha can be brought here that is the property of logarithms, I am sure you all remember minus alpha times log of K so, look at this; look at this. So, what does this mean? Log of f of Kf(K) is equal to minus\_a alpha times\_ log of K, as you know the x axis is your K and y axis was your f of Kf(K). Now, you see if you were to plot the log of it ok. So, what you will do is you will plot instead of K you will plot log K, instead of f of Kf(K) you will plot log of f of Kf(K) ok.

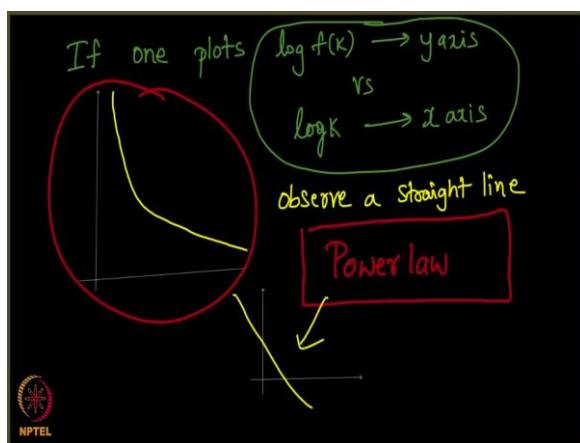
If you observe it, what do you see here? You see that look this is a variable that we call it some y is equal to minus\_alpha, please note this is a constant minus alpha is a constant because it is the power of K right. K is the variable here right times log of K which I am going to replace by let us say variable x. So, as you can see you are getting equation of a straight line straight-line y equals minus alpha times x. So, what does it mean this simply means that when you are trying to plot the log of the graph; the log log of the graph-graph?

(Refer Slide Time: 06:02)



Let us say you are not plotting the typical  $K$  and  $f(K)$ . You are plotting be clear here I am writing a plot here y axis this is my x axis and if I am trying to plot instead of  $K$ , I plot here  $\log K$  and here I plot  $\log f(x)$ . So, what I will trying to say look at this look at this, I am trying to say that this will actually be a straight line a straight line right. So, actually it should pass through a origin here, but do not worry I am just giving you an example. So, this will be a straight line with slope let us say minus alpha ok, if get a straight line with slope minus alpha.

(Refer Slide Time: 07:43)



So, let me write down a straight line. If you getting a straight line which means that log K versus log  $f(-K)$  rather log  $f(-K)$  versus log K that you wrote was of this form, which means  $f \propto K^{\alpha}$  was indeed of the form  $1/K^{\alpha}$  to the power of something and that something happens to be the slope of this line. So, all in all the litmus test is the following it might sound very technical, but all I am saying is the following let me write that down neatly.

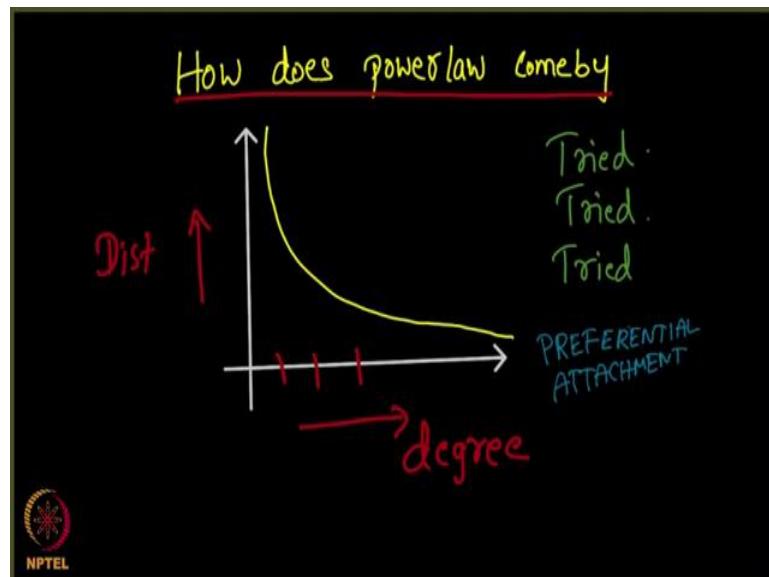
If one plots log of  $f \propto K^{\alpha}$  in the y axis; in the y axis versus log K in my x axis, just in case after plotting this if you observe a straight line then you can possibly conclude that there is powerlaw happening here. Now, you might ask me this question when you are given a plot like this let us say like this why ~~I cannot~~ observe this plot and then simply say there is powerlaw happening here. ~~But~~ then this particular plot can be anything it need not necessarily be  $1/K^{\alpha}$  over K to the alpha it can be anything right.

So, you never know if it is really of the form  $1/K^{\alpha}$  divided by a variable to the power of a constant let us say  $\alpha$ . You would not know if it is like this only right. So, what you do is you instead do not take chances when you have when you get a curve like this you try to draw the log log plot; you try to draw the log log plot. This is called a log log plot y axis make it log x axis, make it log values and then try to see you will get a straight line. When you get a straight line, it means that this is powerlaw happening here perfect.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Rich Get Richer Phenomenon**  
**Lecture - 119**  
**Rich Get Richer Phenomenon**

(Refer Slide Time: 00:12)

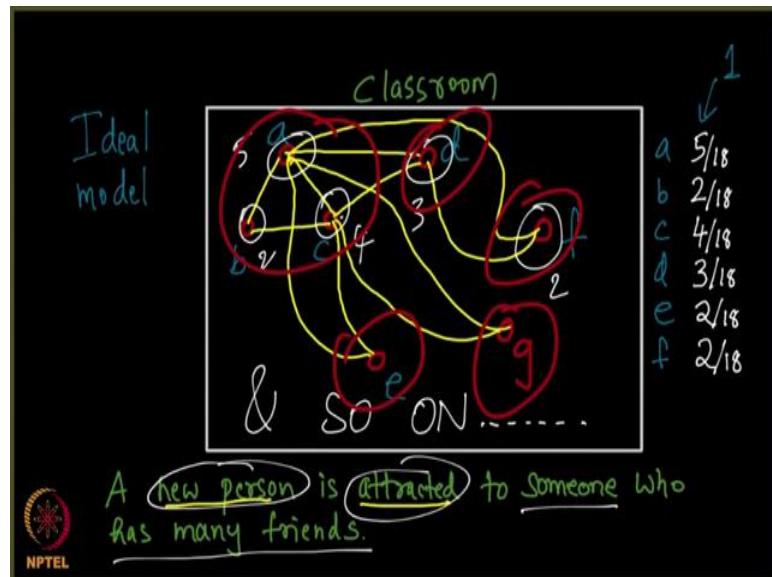


So, let us now make an attempt to answer the big question that we have been asking from quite some time. How does power law come by? What results in this power law right.right? So, the interesting question just to just for the sake of revision let me rephrase it. So, when you try to draw the degree let say on the x axis. So, how many vertices are there of degree 1, of degree 2 of, degree 3 if you plot that basically it is the distribution on the y axis you will observe drop like this. The big question is how does this drop this drops combine, why does it come in place of something like this we observe something like this why is this even this is even happening right.

So, many scientists have tried and tried to understand this, and they have come with their own set of hypothesis and one of the most convincing hypothesis is what is called the preferential attachment hypothesis. It is a hypothesis people are said that there is something called preferential attachment that is happening because of which one observes power law. Now, let us see what exactly is this

preferential attachment this preferential attachment is. So, assume there is a classroom alright.

(Refer Slide Time: 01:59)



So, this is my classroom and there is a person who is sitting here another person another person. So, these 3 people are mutually friends with each other ok. This is a classroom let me write that down, this is a classroom and there only 3 people with only 3 people to begin with let say 1 2 and 3. So, a ~~fourth~~-<sup>4th</sup> person comes; a ~~fourth~~-<sup>4th</sup> person comes and he tries making friends with 2 of the existing people. So, which means he makes friends with let say 1 and 3. Now, wait a minute observe the degree of different vertices here his degree is 2 right and the degree of this is 3, this degree is 2 the degree of the node 1 is 3 correct. So, let me put this is a oval so, that you know that I am talking about degree here right fine.

Now, when a person comes when a person comes look at the story that I am telling you might question why is this story this story is true, for the time being you please assume with the story happens to be true alright. So, this there is this person who come<sub>2</sub> and he observes that there are 1 2 3 4 there are 4 people here of different degrees correct and he prefers to make friends with. So, let me do one thing I think will remove 1 2 3 4 here for clarity say let me instead call it a b c ok, let me call this a and let me remove 2 here and 3 here and 4 here. Because, it is confusing when I call when I name the nodes as a numbers and then also talk about their degree ok.

Let me call it a b c d and now there is a new fellow e, he looks at these people a b c d and e and looks at their degrees; a has degree 3, b has degree 2, c has degree 3, d has degree 2. And, e must make friends with let say 2 people every person who comes inside makes friends with 2 people, just the way d made friends. He also makess friends with 2 people and who will he choose the story goes like this he will be attracted to people.

So, let me write that down this is a sort of important he will be attracted a new person, a new person is attracted is attracted to someone is attracted to someone who has many who has many friends who has many friends. So, what do I mean by this? It does not means that he is not attracted to people who are less friends. The point I am trying to make is he will be attracted to you know what it is self explanatory has I has I do this small exercise 3 plus 2 is 5 plus 3 is 8 plus 2 is 10.

So, now what I will do is I will write 3 by 10 probability 2 by 10 probability 3 by 10 probability 2 by 10 probability, you will see the sum of all this is actually 1. And, this will be the probability with which e will be attracted to b c d or let say a ok; this is a probability with which you will be attracted. So, e will be attracted to a with probability 3 by 10, b with probability 2 by 10, c with probability 3 by 10, d with probability 2 by 10.

So, what does it even mean? It means that I when I enter classroom.s I will be attracted to people proportionately to they with respecting their degree. I will be attracted to c with probability 3 by 10 because the degree is c and then the total number of degree is 10. So, I will be attracted with degree 3 by 10.

So, you probably are confused you are you are not understanding what has this this has to do with degree, what has friendship to do with degree. So, the point is simple when you talk to people a person who has a lot of friends will quickly come to the limelight, he is visible to you, he or she is visible to you and you would want to be friends with her more than anyone else, the person who is very visible.

So, the probability of making a new person having a friend is directly proportional to the number of friends that they already have alright. So, let me assume that e actually makes friends ends up making friends with c and in fact, a which means what does it means. I should now update the degree of c it is no more 3 it will be 1 2 3 4 right,right; the degree of c is 4 and the degree of a also happens to b 4 degree of a is also 4 ok.

So, this requires me to erase the degree of a here and the degree of c also is to be removed right. And, then the degree of a happens to be 4 and the degree of c happens to be 4 and as you would have guessed I may want to remove all the denominators here. So, it is no more 10 you see what else is it done, it is going to be some of the degrees which is it 12 correct it is going to be 12 here. So, I am going to write 12 12 12 and 12 e also has 2 degree. So, e contributes again  $2 \frac{1}{12}$  ok. So, let me count the total degree right now e is  $2 \frac{1}{2} + 2 \frac{1}{4} + 2 \frac{1}{4} + 2 \frac{1}{2}$  which is oh no, it is not 12 it is 14 you see it is 14. So, let me remove 12 and make it 14 it is not 12 it is 14 good.

So, 14 so, now you are getting what I am doing right. So, 14 for c denominator and d will be again 14, the entire process is actually quite self explanatory, e is also 14. So, entire thing is very self explanatory, why that is because you are just following a simple rule that whenever a new person comes, he becomes attracted to someone who has many friends. It will become very clear one side I writing a new node let say f.

Now, the story becomes slightly complex, but it is easy for you to understand. So, what happens think about it, let me just remove these whatever the degrees here, any way we have a degrees here. So, that is should be enough I am just removing them. So, now tell me where exactly will f become friends with, f has now several options a b c d e.

But let us assume f become friends with only 2 people. Now, now you might me why is this restriction to see, it does not look like a real life scenario. What do you mean every person has only 2 friends? This is not have it happens in real life, but all I am trying to say is let us for a moment consider an ideal model right. Not all of us will make only 2 friends , but we all know that our friendship budget is sort of the same. We cannot make a lot of friends; we cannot make no friends also.

There is some average number and let us have a ideal world here let say each person makes friends with only 2 people ok; f comes here and he needs to makes make friends with 2 people. So, he makes friends with a with probability  $\frac{1}{14}$ , b with probability  $\frac{1}{14}$ , c with probability  $\frac{1}{14}$ , d with  $\frac{1}{14}$  and e with  $\frac{1}{14}$  anything can happen. And when he so, this is sort of tossing a coin and then deciding, but you see there is higher probability for f to become friends with a and c right.

So, the problem probably becomes friends with a and with let say d, see I just said these 2 are high probability. Probability does not mean that you always will hit on nodes with

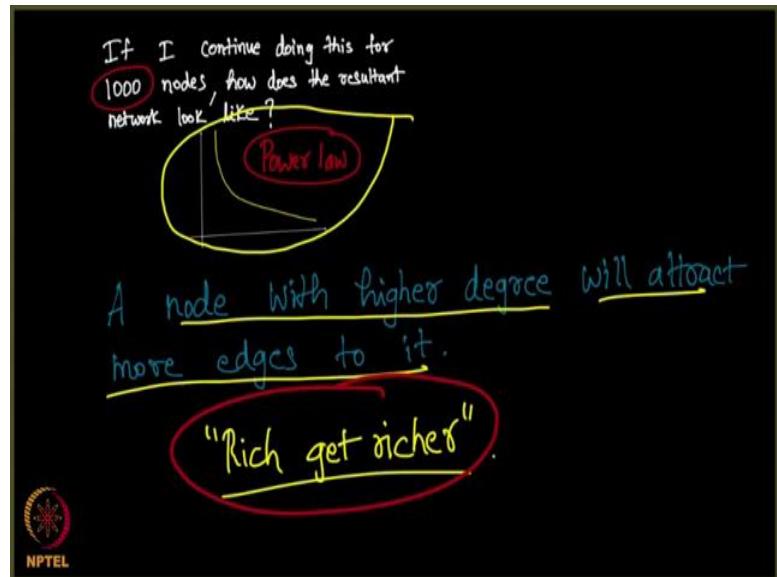
the same with a with a high probability, you might become friends with anyone here right. You might become friends with anybody here good. So, now let us recompute this probability since see what happens ok. So, here goes my next question why should II should recomputed because, there will be new nodes that is coming namely g ok.

For g to become friends with 2 people g needs to know the revised probability. So, let me remove it and let me again see what is the total degree here 2 plus the total degree is here 2 + 3 2 plus + 3 because 1 2 3 4 4 plus + 2 plus + 1 2 3 4 5 for a ok. So, let me write it down 5 for a, 2 for b and 4 for c and 3 for d and 2 for e and 2 for f.

So, what is the sum? The sum is 5 6 7 10 14 15 16 17 18 is the sum. So, the denominator will become 18 right now ok. I hope you are getting a hang of it right now, as I am doing this repeatedly. I am spending sometimes doing this manually. So, that you are understand what happening and later on we are going to write a piece of code and show you what exactly is happening here.

So, now g has come g has come, g must become friends with couple of people and he will respect this probabilities and become friends with 2 people. And, as you can see there are roughly one thirds probability that he will be friends with a right. And, good properly that he will be friends with c as well correct and also yeah 3 1 18 is 1 norm 6 right with d as well. So, g ends up becoming friends with let us say a and let say c and so on and so on dot-dot dot ok. As I keep doing this the question that I would like to ask you all is the following, let me write that doubts that it seems into your mind.

(Refer Slide Time: 13:47)



If I continue doing this process this for 1000 nodes let say, starting from the first 3 nodes as you know; let me go to the previous slide we start with this first 3 nodes. And, we went on adding more and more and more nodes correct right. As I keep doing this as I keep doing this if I keep doing this for 1000 nodes how does the graph look like, how does the resultant network look like.

So, what is happening let me go back and see what is happening. A new node always makes a 2 friends right, a new node makes friends with 2 people always and the next node that comes make friends with the existing people respecting the degree right. Any node that is coming becomes friends with the existing node by respecting the existing degree correct ok.

So, look like how does it look like? It looks like you will be started to observe that it looks like a network, where the degree distribution indeed follows powerlaw. Now that is the climax of a discussion, it indeed follows powerlaw. Such a network which grows by respecting the degrees of the existing people and then grow slowly node by node it follows powerlaw ok.

And, let us go back and then see what happened here, you see people nodes who already had higher probability had the propensity to attract new nodes right. If this continuous for 1000 nodes, you will observe that higher degree nodes will become more and more

rather it will become the degree will become higher and higher for nodes the whose degrees is already higher ok.

I repeat that I repeat that is that it is clear to you ok. So, let me is write that here a node with higher degree higher degree will attract more edges to it, why is that why is that isn't that obvious that is what we have been discussed. Because, a new node that comes that is comes will become friends with the existing a node that comes will become friends to the existing node that already has higher degree. Because, that is are that is the protocol that is the algorithm right. So, he will have many other options, but he will choose that node with higher degree already correct; that is what we discussed here right correct. So, a node with higher degree will attract more nodes to it.

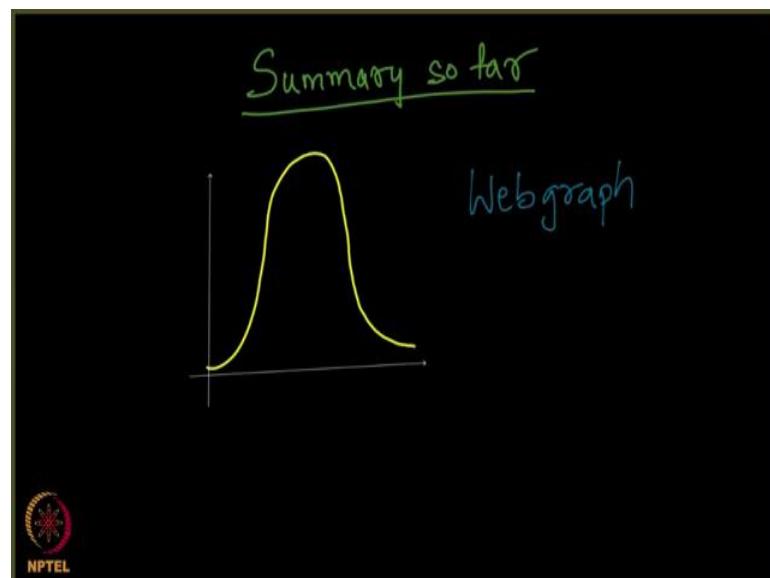
Now, this process this process is called rich get richer process; I am sure it makes sense the very terminology make sense to you. What you mean by rich get richer? People who already have high degree are getting more and more degree because, that is what algorithm does you see. And surprisingly such an algorithms such of protocols such of mechanism of developing networks results in the emergence of powerlaw. It also a node with higher degree will attract more adjust to it. This phenomena is called the rich get richer phenomena and we observe powerlaw in such networks.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Richer Get Richer Phenomenon**

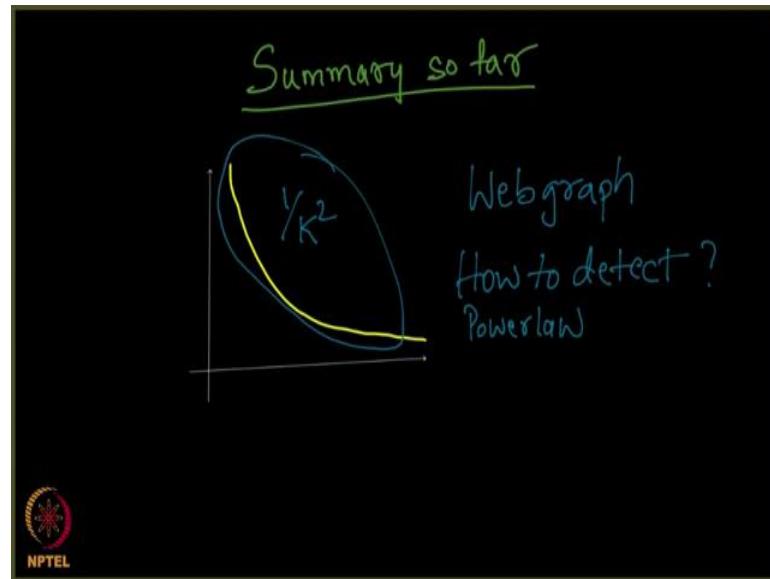
**Lecture – 120**  
**Summary So Far**

(Refer Slide Time: 00:07)



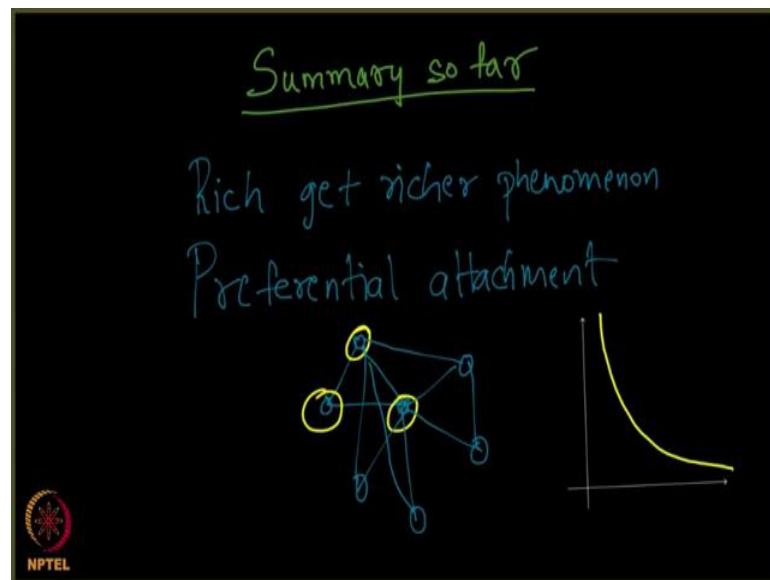
So, what are we seen so far, let us summarize it. The summary so, far is the following: we observed the emergence of what is called the normal distribution right. We saw that it is always a curve called the bell curve, the height example of a town right. In a town if you take the heights of different people you will observe this, but then we saw that it is in fact, not the case if you were to be seeing what is called the web graph or the (Refer Time: 00:42) graph, www graph right.

(Refer Slide Time: 00:49)



What do you observe? We told you that we observe a drop like this and even I told you that the drop is basically a function  $1/K^2$  right. And, we even saw how to detect such a thing let us such a thing happening in a network, how did detected right. How you detected power law that was the question we saw, next right perfect and then we switched gears and we saw the process of rich getting richer right.

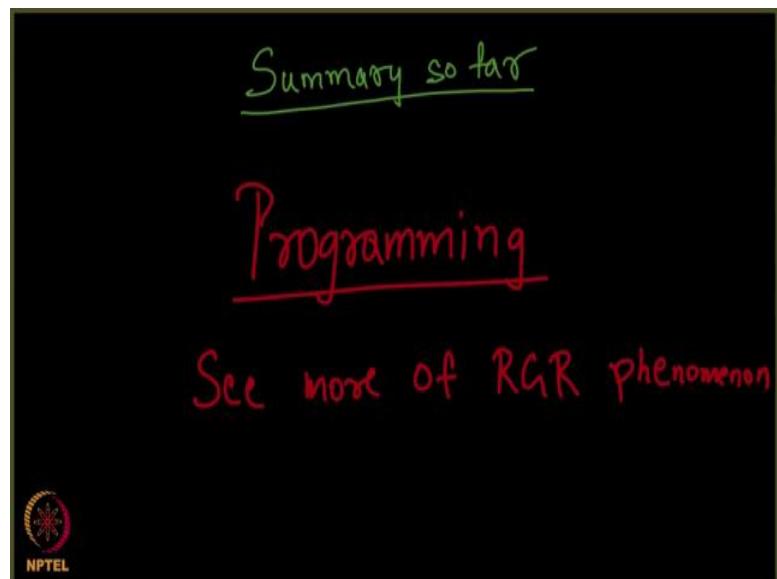
(Refer Slide Time: 01:24)



Rich get richer phenomenon and what was that, if you can recollect it was what is called the preferential attachment, preferential attachment. What we say? We said that given

node we start with let say 3 nodes, just as an example we start with 3 nodes and then and a new nodes comes. It respects the degree of the existing nodes and based on the degree proportionately it becomes friends with a couple of people alright. And, as we continue this process as we continue this process, we observe that when we plot the degree distribution of such a graph we observe the power law alright. There is a summary so for and now what we will do is you will switch gears and we will try looking at some programming.

(Refer Slide Time: 02:31)

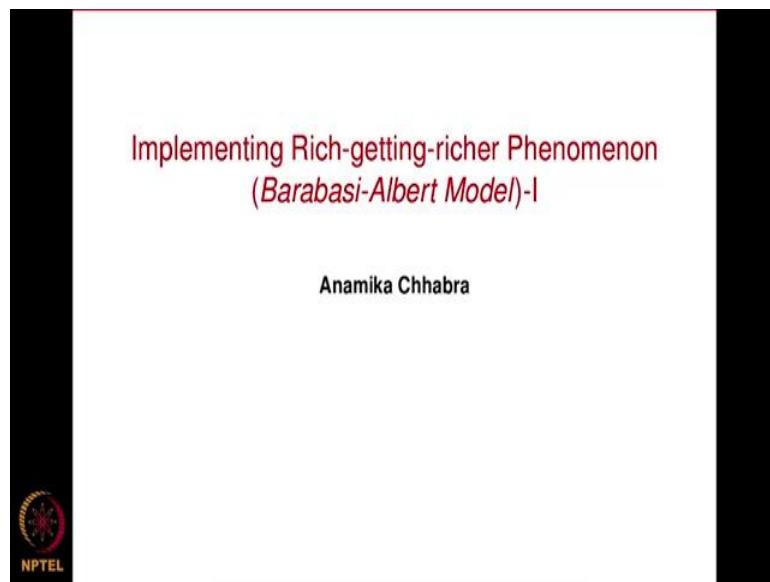


We will try programming whatever we have learnt so far and try to see if we are able to make the same observations that I have hypothesize so far. So, after programming we will get back and then see more of rich getting richer phenomenon. So, now let switch to Python and then see some really cool programming observations.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Prof. Anamika Chhabra**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Rich Get Richer Phenomenon**  
**Lecture - 121**  
**Implementing Rich-getting-richer Phenomenon**  
**(Barabasi-Albert Model)-1**

(Refer Slide Time: 00:05)



Hey everyone, in this video sequence we are going to implement an interesting phenomenon that is called Rich getting richer Phenomenon. It is also called Matthew effect and it is also called Preferential Attachment. Now, there is a particular model that is called Barabasi Albert model which is precisely based on this method that is called preferential attachment. So, we are going to implement Barabasi Albert model in these subsequent videos and that be same as implementing rich getting richer phenomena as well. Before we get into the steps of implementation, let us look at the main idea behind this concept.

(Refer Slide Time: 00:40)

Rich-get-richer phenomenon (i.e. Barabasi-Albert Model)- The Main Idea

- ➊ We start with  $m_0$  nodes, the links between which are chosen arbitrarily, as long as each node has at least one link.
- ➋ The network develops following two steps:
  - ➌ **Growth:** At each timestep we add a new node with  $m$  ( $\leq m_0$ ) links that connect the new node to  $m$  nodes already in the network.
  - ➍ **Preferential attachment:** The probability that a link of the new node connects to node  $i$  depends on the degree of node  $i$ .

NPTEL

So, the main idea is that we are going to start with some  $m_0$  number of nodes. And, these nodes are going to have some edges across each other. The only condition is that every node should have at least one edge; this means that we start with an existing network of  $m_0$  nodes. And, the new node and the new notes get attached to this existing network one by one. Now, how do they get attached? There are two properties that every node follows, and the first thing is that every node gets attached to some  $m$  number of existing nodes. So, for example, say  $m_0$  is equal to say 5 and  $m$  has to be less than equal to  $m_0$ .

So, assume  $m_0$  is equal to 5 and  $m$  is equal to say 3. So, we will start with an existing network of 5 nodes and 1 new node will come and it will get attached to some 3 existing nodes. And, then second node will come that will also get attached to these existing 3 nodes and so on so, this keeps happening. Now, how do we decide which 3 existing nodes should get attached to the new node, that is decided by a preferential attachment. Now, what is preferential attachment? This means that the more connected a node is, the more likely it is to receive links with a new node.

In other words, more degree an existing node has more probability it will have of connecting to the new node. So, this is how we will keep adding new nodes to the existing network. This was the main idea behind rich getting richer phenomenon. Now, let us look at this steps that we are going to follow for the implementation.

(Refer Slide Time: 02:26)



## Steps for Implementation

- Take  $n$ , i.e. total number of nodes, from the user.
- Either take  $m$ , i.e. the number of edges to be connected to the new node, from the user, or decide it yourself based on  $n$ . In our implementation we will decide it ourselves as follows:
  - ( $m_0$  is the initial number of nodes that should have atleast one link.  $m$  should be less than or equal to  $m_0$ ).
  - We will take  $m_0$  to be any random number between 2 to  $n/5$ . (You can use any measure.)
  - We will take  $m$  to be one less than  $m_0$ .
- Add the rest  $n-m_0$  nodes. Add edges to these  $n-m_0$  nodes based on 'preferential attachment'.

So, I am going to required value of  $n$ ,  $n$  that is the total number of nodes in the network. We will take the value of  $n$  from the user, the other thing that we need is the value of  $m$  that is equal to the number of edges that every new node is going to get attached to; basically the number of edges that every node will introduced into the network. So, we will take value of  $m$  from the user or we can have our code decide the value of  $m$  automatically.

In our code we will decide the value of  $m$  based on  $n$ , it is up to you can do whatever you feel like ok. The other thing that we need is  $m_0$ , as I told you  $m_0$  is the number of nodes in the existing network, in the initial network. The only condition is that  $m \leq m_0$ . So, we need  $n$ , we need  $m$  and we need  $m_0$ ; these 3 values we need, and the only condition is that  $m$  should be less than equal to  $m_0$ .

So, in our code we are going to have  $m = m_0 - 1$ , you can just fix it on any a new value that you wish to have. So, in our code we will have  $m_0$  to be any random number between 2 to  $n/5$ , this is this is just an assumption we are taking; you can go ahead with any value of  $m_0$ . You can fix on a value  $m_0 = 4$ . So, every time you will start with an initial network of 4 nodes and  $m$ , we will take be  $m_0 - 1$  you can take any value.

Now, we have a network of  $m_0$  nodes, the next step is to keep adding new nodes to this existing network. So, the number of nodes that are going to be attached is equal to  $n - m_0$  because,  $n$  is the total number of nodes. So, these  $n$  minus  $m_0$  nodes are going to be

attached to existing network based on preferential attachment as I explained you. Now, let us see how we are going to implement preferential attachment ok.

(Refer Slide Time: 04:38)

### Steps to be followed for Preferential Attachment

For all  $n - m_0$  nodes, repeat the following:

- Add the node.
- To add the edges, do the following:
  - Preprocessing:
    - Get a dictionary of degrees (since preferential attachment will happen based on degrees).
    - Maintain a dictionary of probabilities. (The probabilities have to be assigned based on degrees. More the degree, more the probability. Precisely,  $\text{probability}[i] = \text{Degree}[i]/\text{Sum}(\text{degrees of all the nodes})$ ).
    - Maintain a list of lists for maintaining cumulative node probabilities.(This is for choosing a node based on probabilities)  
For Example,  
 $\text{Probabilities} = [0.2, 0.3, 0.5]$   
 $\rightarrow \text{Cumulative Probabilities} = [0.2, 0.5, 1.0]$ .
  - While  $\text{edges\_added}$  are not equal to  $m$ :
    - Choose a random number from 0 to 1.
    - Whichever node has cumulative probability more than this random number, the edge will be connected to that node, if not already added.  
If  $r = 0.4$   
 $\rightarrow \text{Node 2 will be chose out nodes 1, 2 and 3.}$

So, we have to add  $n - m_0$  nodes one by one. So firstly, we will add the node to the network. The next step is to add  $m$  edges to this node to the existing nodes in the network. Now, how are we going to decide the nodes that are going to be attached to this existing node? As I told you we need we need the probabilities and the probabilities are going to be decided based on the degrees that the nodes have, more the degree more the probability right. So, for that pattern we need to keep a track of the degrees of the nodes. So, we will do some pre processing for this keeping track. We will maintain a dictionary of degrees for every node because, we the degrees basically going to decide the probabilities.

Second thing is we will maintain a dictionary of probabilities. So, probability of a node getting attached to a new node will be equal to the degree of that node divided by the sum of the degrees of all the nodes right. So, more the degree more the probability so, we are going to maintain a track of the probabilities in a dictionary. Now, apart from that we are going to maintain a third thing, we will basically maintain a list of list because we want to keep an order of the things and order of nodes dictionary does not let you maintain an order. So, we are preferring to keep a list of lists for maintaining cumulative node probabilities ok.

Now, let me tell you why we need that. So, whenever you have to choose a node based on a probability this is a standard technique that we use, although you can device some other method is well. We are going to follow this method of making use of cumulative probabilities. Now, let me explain you what that is using an example. Assume there are 3 nodes and they have the properties 0.2 0.3 and 0.5, we are going to maintain a list that is cumulative probabilities.

So, the first node we have cumulative probability 0.2, the second we will have 0.2 plus 0.3 that is 0.5, the third will have 0.5 plus 0.5 that is 1.0. So, this is the less that we are going to maintain, I will show you how we are going to use this list. So, when new node has come, we have to connect this new node to existing m nodes for that we will choose random number between 0 and 1 ok. And, then whichever node has cumulative probability more than this random number is the node which is going to be connected to the new node.

Let me show that one example. So, assume  $r$  comes out to be 0.5 0.4 let us look at the cumulative probabilities list that we just created. So, first nodes cumulative probabilities 0.2 which is less than 0.4; so, we will go ahead we will check the second one, 0.5 is the second value, 0.5 is more than 0.4. So, we will stop there since  $r$  is less than or equal to 0.5 the second node is basically the node which is going to get connected to the new node. You would have understood this method, but do you understand the main idea behind it.

So, the main idea is that more probability node has more likely it will get connected to the node, this is what we have to implement. Now, try to understand how this particular method is helping us implement the same thing. So, basically more probability a node has more window it will have in the cumulative probabilities list. Now, what do I mean by that? So, as you can see just take a look at the cumulative probabilities list. So, we have first nodes cumulative probability to be 0.2. So, its window will be 0 to 0.2, the window for the second node is 0.2 to 0.5 ok, the window for the third node is 0.5 to 1.0 ok.

So, now you can see that the window for the third node is more than the window for the second node and the window for the second node is more than the window for the first node. So, wider a window is more likely it is that the random number is going to fall into

that window ok. So, assume a node has probability 0.1, it is window will be 0.1 big, if a node has probability 0.6 it is window will be 0.6 wide and more likely it is that the random number is going to fall into that window ok.

And this precisely we are implementing here. So, if a node has more probability the random number is going to fall into that window more likely and that is how we will connect that node to the new node. So, that is the main idea. This is how we are going to add the edges to the new node, and this is how we are going to add new nodes up to the point that we get total n nodes in the network. So, this is the main idea.

In fact, not the main idea this is detailed steps of implementation that I have discussed with you. I would suggest you since I have gone to detail into the steps of implementation, it will be nice if you just start of yourself, coding yourself and implement yourself and do not follow the subsequent videos line by line. And, just compare the results with the results of the subsequent videos that will be a nice practice. So, let us get started with the coding part.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Rich Get Richer Phenomenon**  
**Lecture – 122**  
**Implementing Rich-getting-richer Phenomenon (Barabasi-Albert Model)-2**

(Refer Slide Time: 00:05)

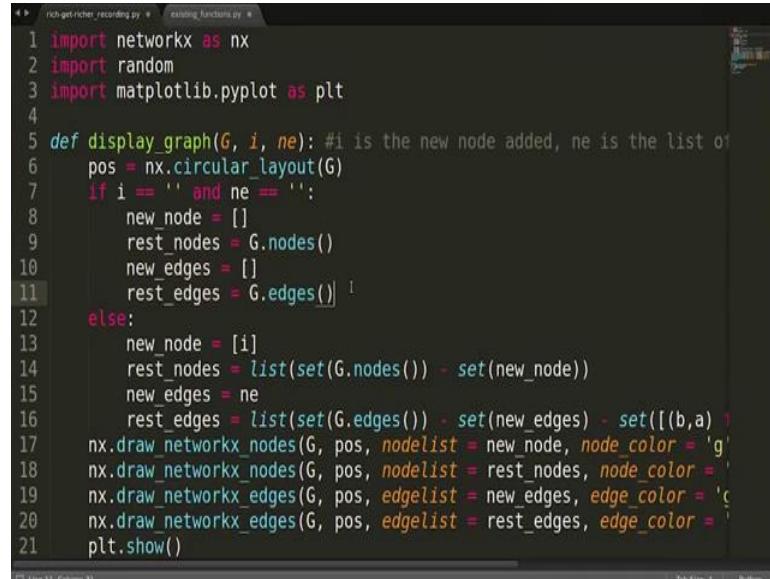
---

 **Implementing Rich-getting-richer Phenomenon**  
**(Barabasi-Albert Model)-2**

**Anamika Chhabra**

In the previous video, we discuss the main idea behind Rich-getting-richer Phenomena. And we also discuss the steps that we are going to follow for implementation. In this video we are going to start off with implementation.

(Refer Slide Time: 00:19)



```
1 import networkx as nx
2 import random
3 import matplotlib.pyplot as plt
4
5 def display_graph(G, i, ne): #i is the new node added, ne is the list of
6     pos = nx.circular_layout(G)
7     if i == '' and ne == '':
8         new_node = []
9         rest_nodes = G.nodes()
10        new_edges = []
11        rest_edges = G.edges()
12    else:
13        new_node = [i]
14        rest_nodes = list(set(G.nodes()) - set(new_node))
15        new_edges = ne
16        rest_edges = list(set(G.edges()) - set(new_edges) - set([(b,a)]))
17    nx.draw_networkx_nodes(G, pos, nodelist = new_node, node_color = 'g')
18    nx.draw_networkx_nodes(G, pos, nodelist = rest_nodes, node_color = 'b')
19    nx.draw_networkx_edges(G, pos, edgelist = new_edges, edge_color = 'g')
20    nx.draw_networkx_edges(G, pos, edgelist = rest_edges, edge_color = 'b')
21    plt.show()
```

So, let me import some packages that we are going to need, we are going to need the network. We are going to take random number, so we will meet random package. We are also going to display the networks, so let us import matplotlib ok.

So, let us start our main function ok. So, if you remember from the previous video, if you remember the outline if you have not seen the video, you may go back and see the steps at we are going to follow here. So, we require the values of n, m and m0, n is the total number of users, m is the number of edges that every new node will get attached to, m0 is the number of nodes in the initial network that we will start with.

Let us take the value of n from the user. So, we can use the function raw input and in case you are using Python 3.x you can use input function instead of raw input. So, let us use this function, raw input and as the user entered the value of n all right. Next, we need the value of m0 as I told you can initialize m0 to be some 3, 4, 5 whatever you want. Let me take m0 based on n that the user will pass. So, I plan to take a random value between 2 is that is the minimum number of nodes that should be there in the initial network and n divided by 5; to n divided by 5 I will choose the random number between 2 to n/5. So, assume n the user entered n to be 100 then I will take a random value between 2 and 20 that is how I will start the initial network.

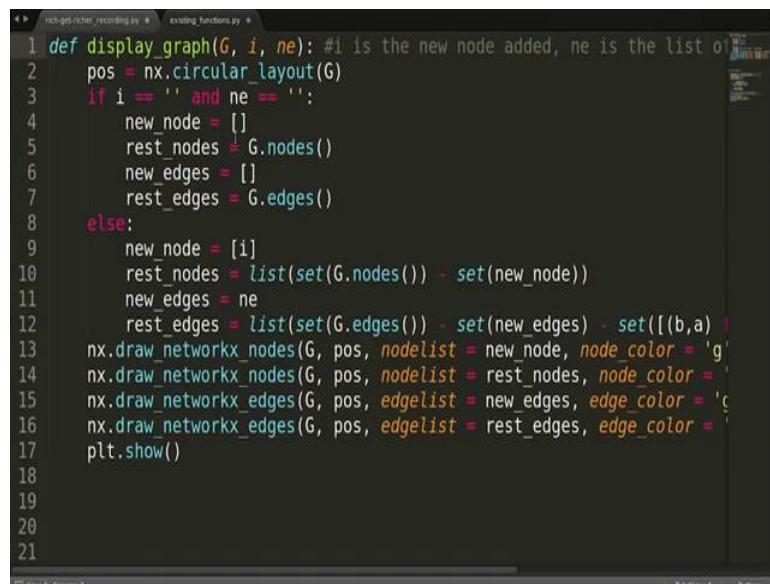
In fact, if the user enters a very high value of n that this might not be a good idea, you can it is it is actually better to fix up a value of n0 to be 3, 4 or something. For example,

user enters n to be 1000 then your initial networks will be between a random number between 2 and 200 right. It is fine you can take any value of m0. And if you remember the only thing is that m should be less than equal to m0. So, here we will take m to be m0 - 1. m = m0 - 1. And we have to start with an initial network with m0 nodes.

So, as of now we have not to means started the graph. So, let us start the initial graph with m0 nodes. And the condition is that every node should have at least one edge. So, I will have a path graph because, there every node has at least one edge you can do any other thing like, you can add the edges one by one just making sure that an every node has one link at least. You can manually add the edges, so it is up to you. I will take a path graph, I will write nx.path graph of m0 nodes ok.

Now, to start off I wish to see how the graph looks like. So, that we can see the changes that that keep happening to the graph at every time stamp, I am going to make use of a function display graph that we created in one of the previous videos. So, I am going to just take the function from there and I will briefly explain it here, I will not write the function line by line here.

(Refer Slide Time: 04:37)



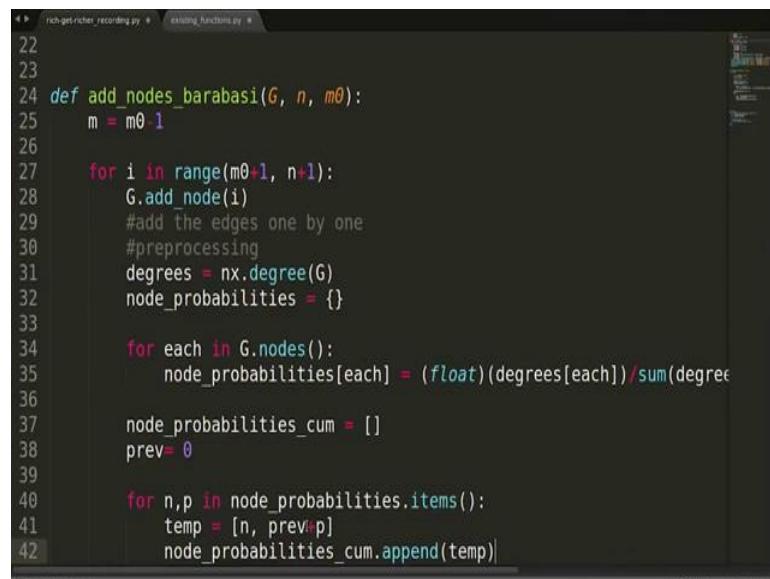
```
1 def display_graph(G, i, ne): #i is the new node added, ne is the list of edges
2     pos = nx.circular_layout(G)
3     if i == '' and ne == '':
4         new_node = []
5         rest_nodes = G.nodes()
6         new_edges = []
7         rest_edges = G.edges()
8     else:
9         new_node = [i]
10        rest_nodes = list(set(G.nodes()) - set(new_node))
11        new_edges = ne
12        rest_edges = list(set(G.edges()) - set(new_edges) - set([(b,a) for (a,b) in new_edges]))
13    nx.draw_networkx_nodes(G, pos, nodelist = new_node, node_color = 'g')
14    nx.draw_networkx_nodes(G, pos, nodelist = rest_nodes, node_color = 'r')
15    nx.draw_networkx_edges(G, pos, edgelist = new_edges, edge_color = 'g')
16    nx.draw_networkx_edges(G, pos, edgelist = rest_edges, edge_color = 'r')
17    plt.show()
18
19
20
21
```

So, let me call that function display graph before that let me copy it. So, this is the function that we created one of the previous videos, I am copying it and pasting it here, I brief out for the once you are not see in the previous videos.

So, what this function is doing? We are passing the graph G that has to be displayed. We are passing i, i is the new node to be added and ne is the list of new edges that are going to be added. So, basically at every iteration I plan to call this function display graph and I also want to display the new edge that got added in this iteration, what did I say; I want to display the new node that got added, I and I also want to display the new edge is that got added. So, I am going to change the colours of this nodes and edges. So, that its nice to see how the nodes are getting added for that only I have created this customized function.

I am going to use a circular layout ok. So, I will use this function nx.circular\_layout(G). And since, I am going since i need to display the colours of the new node and new edges differently. So, what I have to do here is that. Firstly, I am putting this condition ok. So, initially there will be no new node no new edge. So, i and ne will be empty that is what will happened at the first before any iteration takes place initially right. So, new node will be. So, we will maintain a list of new nodes and we will maintain a list of new edges. Initially, they will be empty and rest of the nodes which is all the nodes in the graph, rest of the edges which is all the edges in the graph. So, we require these 4 list previous nodes, new node previous edges, new edges, these 4 list, we need ok..

(Refer Slide Time: 06:43)



```

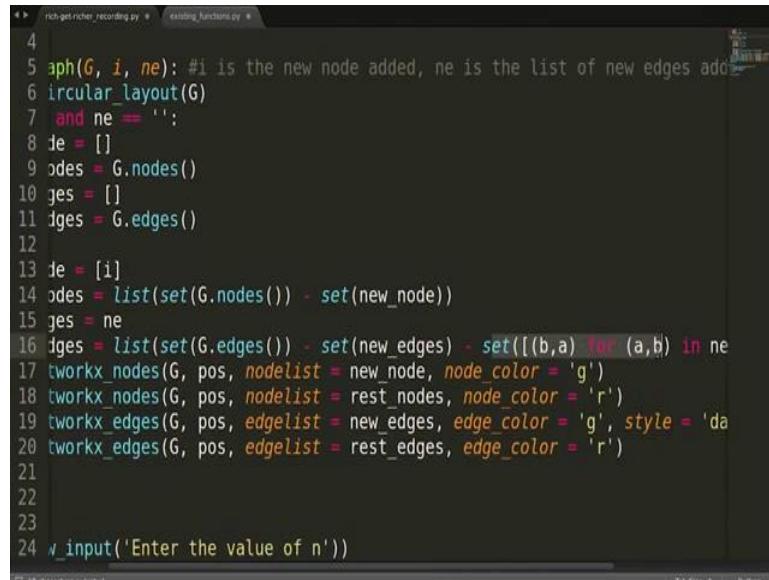
22
23
24 def add_nodes_barabasi(G, n, m0):
25     m = m0-1
26
27     for i in range(m0+1, n+1):
28         G.add_node(i)
29         #add the edges one by one
30         #preprocessing
31         degrees = nx.degree(G)
32         node_probabilities = {}
33
34         for each in G.nodes():
35             node_probabilities[each] = (float)(degrees[each])/sum(degrees)
36
37         node_probabilities_cum = []
38         prev= 0
39
40         for n,p in node_probabilities.items():
41             temp = [n, prev+p]
42             node_probabilities_cum.append(temp)

```

In case i and ne are not empty, there are some values in them, what are we going to do is, new node list is going to have i new edges list is going to have ne, which we

passed as a parameter. Rest of the edges will be all the edges, I am sorry rest of the nodes will be all the nodes minus the new node, rest of the edges will be all the edges minus new edges minus the intersection right; minus the common edges basically, let me explain.

(Refer Slide Time: 07:17)



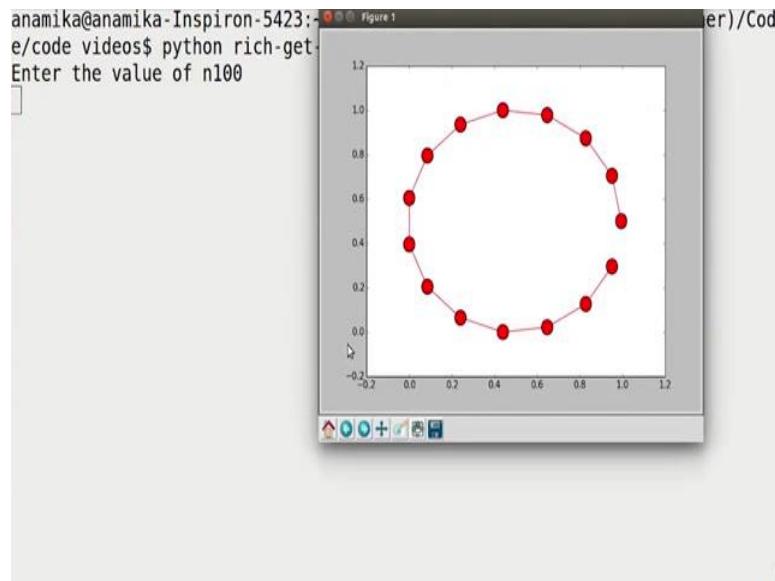
```
4
5 def aph(G, i, ne): #i is the new node added, ne is the list of new edges added
6     circular_layout(G)
7     if ne == '':
8         de = []
9     nodes = G.nodes()
10    ges = []
11    dges = G.edges()
12
13    de = [i]
14    nodes = list(set(G.nodes()) - set(new_node))
15    ges = ne
16    dges = list(set(G.edges()) - set(new_edges) - set([(b,a) for (a,b) in ne]))
17    networkx_nodes(G, pos, nodelist = new_node, node_color = 'g')
18    networkx_nodes(G, pos, nodelist = rest_nodes, node_color = 'r')
19    networkx_edges(G, pos, edgelist = new_edges, edge_color = 'g', style = 'dotted')
20    networkx_edges(G, pos, edgelist = rest_edges, edge_color = 'r')
21
22
23
24 v_input('Enter the value of n')
```

So, it should take b a and a b to be the same edges right. So, that is why we are deducting that those edges ok. So, then we are displaying. So, since you want the graph to be customized having different nodes having different colours different edges different having different colours, you will have to make use of this function draw networkx nodes and draw networkx edges. So, separately you are calling them.

So, first function will be to display the existing nodes, second function will be to display the new nodes or vice versa. First function will be to display the new edges and second function is to display the existing edges ok. So, look at the parameters quickly we will pass the graph G and then the position that we already created and the node list that has to be given this particular colour. And we are specifying the node colour to be green here and the existing nodes, we are displaying red, it is new edges, we are display in green and you can also change the style and displaying the new edges to be dotted so that, we can precisely differentiate. And we are existing edges, we are displaying that. So, this was about the displaying part, now we are calling this function display graph and ok.

So, we are calling this function, we need 3 parameters G i and ne. So, let us pass G here and i initially, there is nothing, there is no node and the edges also will be empty ok. So, let us call this function let us call main to check the functioning of our existing code ok.

(Refer Slide Time: 09:33)



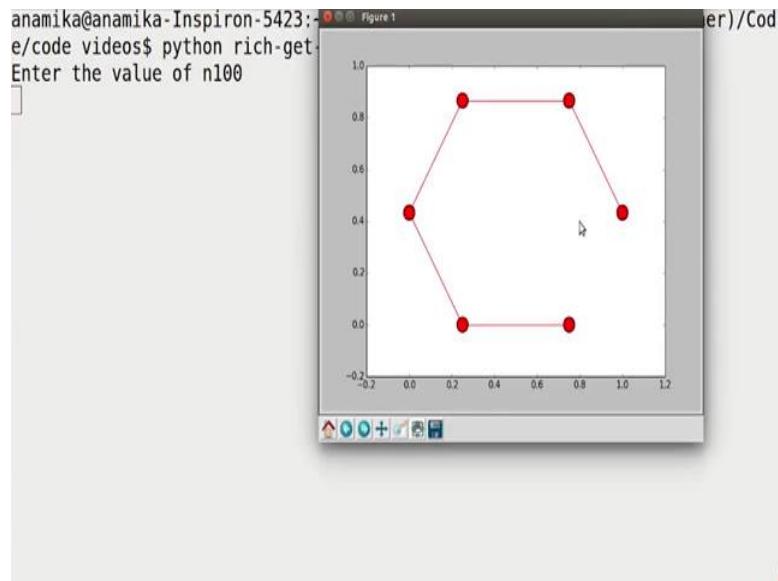
Let us call this file, enter the value of n, I am going to get 100 ok. So, this is the initial graph this is the initial graph as you can see, we had chosen m0 to be random. So, this is the number of nodes that the code has chosen. This is the initial graph and we are going to add the nodes one by one to this graph.

(Refer Slide Time: 09:56)

```
Enter the value of n100
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_7_(Rich get richer)/Code/code videos$ python rich-get-richer_recording.py
Enter the value of n100
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_7_(Rich get richer)/Code/code videos$ 100
100: command not found
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_7_(Rich get richer)/Code/code videos$ python rich-get-richer_recording.py
Enter the value of n100
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_7_(Rich get richer)/Code/code videos$ python rich-get-richer_recording.py
Enter the value of n100
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_7_(Rich get richer)/Code/code videos$ python rich-get-richer_recording.py
Enter the value of n100
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_7_(Rich get richer)/Code/code videos$ python rich-get-richer_recording.py
Enter the value of n100
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_7_(Rich get richer)/Code/code videos$ python rich-get-richer_recording.py
Enter the value of n100
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_7_(Rich get richer)/Code/code videos$ clear
```

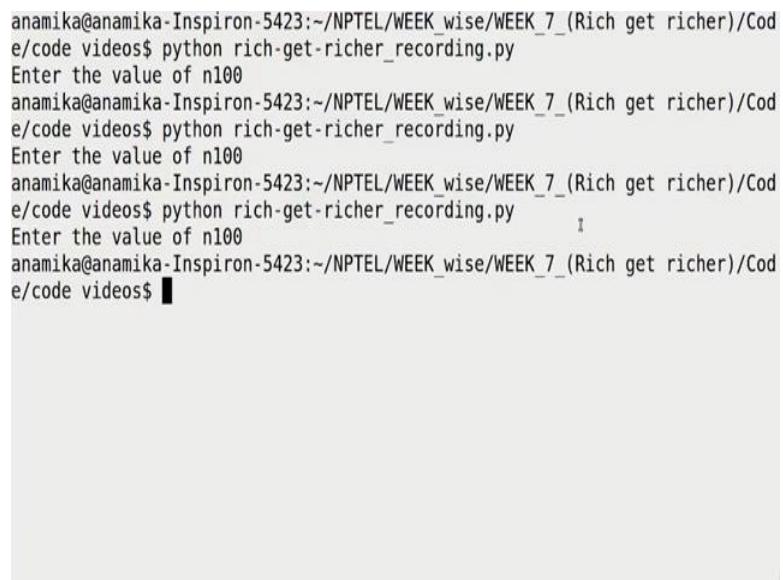
I can execute it again to see and this is taking these many nodes initially, also taking quite good number of nodes initially, I am sorry; I am sorry, second good number of nodes by chance ok.

(Refer Slide Time: 10:27)



I am just trying to see whether it case. So now, it took 1, 2, 3, 4, 5, 6 nodes initially.

(Refer Slide Time: 10:32)



So, it always keeps changing you can fix it or you can have the good design. So, this is the initial graph and we are going to add the nodes to this graph.

Now for adding the nodes let us create a function let us create a function add nodes since, we are going to follow Barabasi Albert model, let me name that accordingly Barabasi ok. We should pass the graph there we should pass the value of n the total number of nodes, we should pass the value of m that is the number of edges and it should written the graph that is what we are going to we will be displaying ok.

So, we have to create this function, now you see there are  $m_0$  nodes into the existing graph already ok. So, we need the value of  $m_0$  here right because  $m_0$  is the number of nodes already. So, I should better pass  $m_0$  over here and  $m_0$  over here and let me decide the value of m in the function itself. So, I am removing this and I am passing G, n and  $m_0$ . Let me decide the value of m here, I will write m is equal to  $m_0$  minus 1 ok. So, we have  $m_0$  nodes and we have to add  $n - m_0$  nodes more.

So, I will start a loop for every node for  $i$  in range  $m_0 + 1$  is the next node that is going to be added and we have to stop at  $n$ . So, since it is a range function, I will put  $n$  plus 1 because, it excludes last value. So, this is a loop to add nodes one by one. So, we will add the first node `G.add_node`, the node is going to be  $i$  now we are added the nodes now, we have to add the edges. If you remember from the previous video, we have to do some pre processing. So, let me see that and under pre processing before that let me also write add the edges one by one ok.

Now, what is the pre processing? We need to choose the nodes based on the probabilities, which are based on the degrees. So, we need to maintain the degrees. So, let me take dictionary `degrees = nx.degree(G)` this written as a dictionary of dictionary, where the key is the node and the value is the degree ok. Now we also need to maintain the probabilities, which are which will be based on the degrees. So, let me create dictionary and node probabilities inside probabilities no probabilities is equal to empty dictionary this is what we are going to create, now how do we create it?

As I told you the probabilities probability of node  $i$  is equal to the degree of  $i$  divided by the sum of the degree of all the nodes. So, we should get a start loop here to assign the values to this dictionary. So, I will write for each in `G.nodes` node probabilities for each is going to be degree the I am sorry the degrees are there in these dictionary degrees of each, we will give us the degree of that node and that should be divided by the sum of all the degrees. Now where are all the degrees? `degrees.values`, if you write `degrees` or

values you get a list of all the degrees, we are going to sum them up and since we are dividing let me do the (Refer Time: 14:59) casting float ok, we should work.

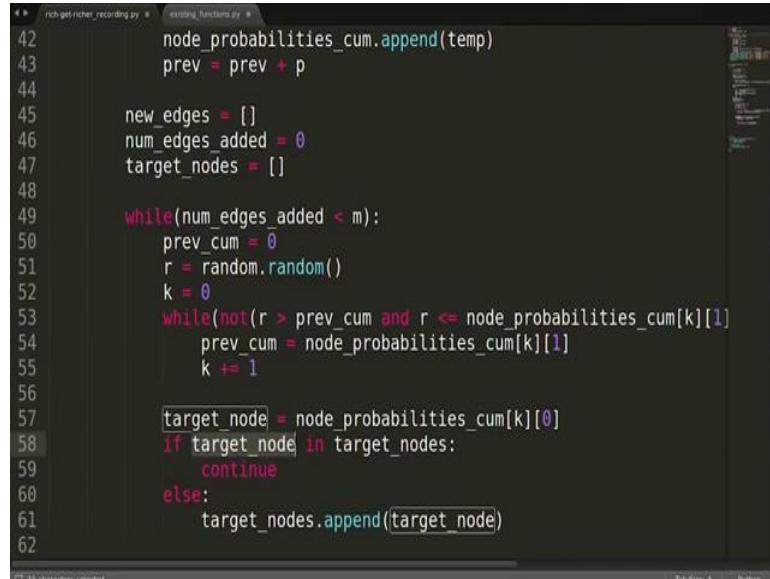
So, we have now created the dictionary node probabilities ok if you remember from the previous video. Now, the next step is to create list that is cumulative node probabilities since, we want then, to be ordered I got create list of list and not the dictionary. Let me show, you how will do that node probabilities cumulative is equal to a list ok. So, we have to take the probabilities from these dictionary node probabilities one by one and based on that we will compute the cumulative probabilities. So, at this point let me start loop, I will write for n comma p in node probabilities.items.

Now, what do we get here? Node probabilities is a dictionary.items will give us a list of toples and topple we will have 2 things n , p that is the node comma the probability. So, we are caption that in n, p here and we have to store the cumulative values in the in the list of lists that is node probabilities cumulative here. So, let me initialize the probability to be initial probability to be 0 you will understand as I go on, I am plating a list temporary list, let me it temp that we will appending that we will appending to this list ok..

So initially, this list has to have node comma the cumulative probability. So, the cumulative probability for the first node is going to be basically, 0 plus it is own probability, if you remember from the previous example, we took in the previous video. So, the cumulative probability for the first node is going to be it is probability itself. So, we have going to add 0 it, and the cumulative probability for the second node is going to be the cumulative probability for the previous node plus the probability for the current node. So, we need to maintain this variable previous.

Now, this is going to be the list for the first node and we will append it to the node probabilities cumulative.append(temp) right..

(Refer Slide Time: 17:44)



```
rich_get_richer_recording.py # existing_functions.py
42     node_probabilities_cum.append(temp)
43     prev = prev + p
44
45     new_edges = []
46     num_edges_added = 0
47     target_nodes = []
48
49     while(num_edges_added < m):
50         prev_cum = 0
51         r = random.random()
52         k = 0
53         while(not(r > prev_cum and r <= node_probabilities_cum[k][1]):
54             prev_cum = node_probabilities_cum[k][1]
55             k += 1
56
57         target_node = node_probabilities_cum[k][0]
58         if target_node in target_nodes:
59             continue
60         else:
61             target_nodes.append(target_node)
62
```

And the previous should now be updated. So, I will write previous is equal to previous plus plus p, because p was the probability for the current node that p appended. In order to add the edges, we will now create another loop that we make use of this cumulative probabilities list. One thing that we have to take care of is an edge that has already been added should not be added again ok.

So, we need to keep a track of the number of edges added that is one thing, second thing is we also have to keep track of the edges that I have been added. So, we need to maintain a list of the edges that we have added for a given node so, that we do not repeat the edges. So, in case and if you do not keep a track of that that is if you do not keep a track of the edges that we have already added, we might end up adding the same edge again. So, in that case it might happen that some nodes are adding lesser edges than n ok.

So, let me create a list new edges is empty; here, we will store the new edges and number of edges added number of edges added for this particular node right initialise to 0. And in order to check whether the same existing node is not going to be connected again, I am going to create list target nodes initialise to 0, I will show you how these list are going to be add going to be used.

Now, we have to add the m edges to the new node. So, I will start loop here why? Number of edges added is less than m what do we do? We have to chose an existing node based on its probability for that we need to random number. So, I will take r is

equal to `random.uniform`, which is going to give me a random value between 0 and 1 you can also use `random` node `random`, you can use that this one I think just check, what is the difference between these 2 `random.random` gives value between 0 and 1 right ok.

So now so, if you remember the only difference between `random` and `uniform` is that in `random` you do not need to pass any parameter and it gives you a float value between 0 and 1 and `uniform`. You can pass the parameters `a` comma `b` where the floating point value that the function will return will be between `a` and `b`. So, if you want a random floating point between 0 and 1, you can use `uniform` bit parameter 0 and 1 that was about the probabilistic of these functions.

Now, we have to choose an existing node based on this `r`. So, if you remember there is going be your window which is cumulative probability list will provide us. So, we have to check the first nodes cumulative probability, if `r` is less than equal to cumulative probability of the first node, we will stop there otherwise; we will go to the next node, if `r` is less than equal to the cumulative probability of the second node we will consider that else we will go on. So, let me start loop here why not `r` is greater than we need this starting point as well. So, let me let me take a variable here previous cumulative is equal to 0. So, this is just create the windows we need to keep track of the previous nodes cumulative probability as well in order to find the window.

So, I will write while `r` is greater than previous cumulative, and `r` is less than equal to node probabilities cumulative of the first node ok. So, we need keep a track of the first as well `k` is equal to 0, first node `k`. And if you remember node probabilities cumulative is a list of list, where every member list has first element as the node and the second element as the cumulative probability. So, we will write I am sorry one here right. So, while this does not happen that `r` is not into that window, we will keep going on with the next element.

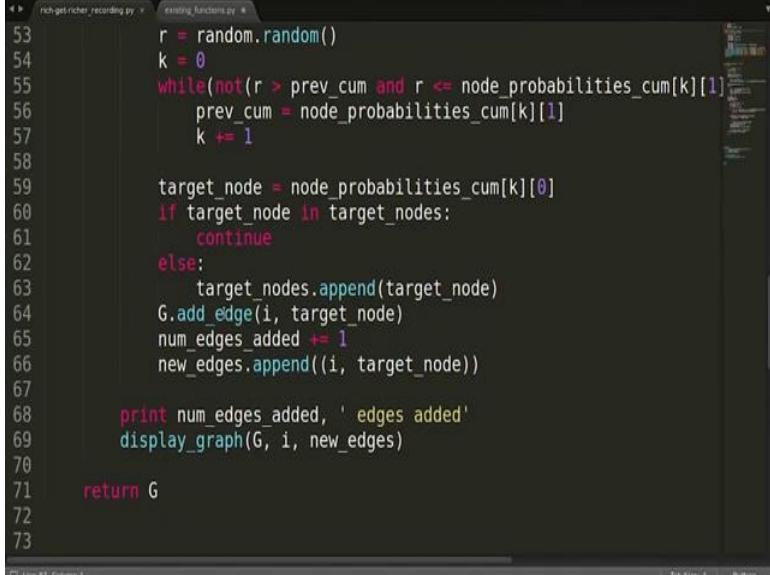
So, in our process previous cumulative is going to be updated with the cumulative probability of the current node and `k` will be implemented ok. Now whenever it finds a node, whose cumulative probability is more than this random number it will come out of the loop and when it comes out of the loop, it has found the node that is the target node. So, target node we are taking a variable `target node`, sorry `target node` is equal to `now node` cumulative probabilities is a list of lists and every list has first

element as a node and second implement as cumulative probability. Now we need the element here the node here. So, we will write k and here 0. So, we will get the first element that is the node that is the node.

Now, we have to check whether this node has already been connected to the new node or not? So, for that wait there is a spelling mistake target write e. So, we have already maintain this target nodes list, which will contain the all the target nodes. So, if though new node which we have chosen is already in target nodes, we will continue finding another target node if the if the target node is not there in this list, we will added there ok. So, we will write it is target node in this list target nodes, what will do? We will continue ok; we will find another target node otherwise this target node is not there in target nodes list. So, we will appended there target nodes.append this new node target node.

So basically, if the node that we have chosen is already in the list of nodes that is the target nodes. We will continue means we will go back here, and m is not implemented.

(Refer Slide Time: 25:24)



```
53     r = random.random()
54     k = 0
55     while(not(r > prev_cum and r <= node_probabilities_cum[k][1])
56         prev_cum = node_probabilities_cum[k][1]
57         k += 1
58
59     target_node = node_probabilities_cum[k][0]
60     if target_node in target_nodes:
61         continue
62     else:
63         target_nodes.append(target_node)
64         G.add_edge(i, target_node)
65         num_edges_added += 1
66         new_edges.append((i, target_node))
67
68     print num_edges_added, 'edges added'
69     display_graph(G, i, new_edges)
70
71 return G
72
73
```

So, we will repeat the whole process up to this point and in case the target node is not there in this list, we will do the rest of the processing that is connecting to this node by using the function G.add\_edge. So, we are going to add an edge from r node that is i to the target node right. So, the number of edges added will be implemented number of edges added plus equal to 1. And we also need to keep track of the new edges added

because, we are going to change the colour. So, we early maintain a list new is large sorry new edges new edges.append, we will add this edge and the edges basically i comma private node ok..

So, this is the new edge that is added. So, we can also keep track of the number of edges added. So, that we can check whether at every step we are adding same number of edges or not. Let us check print number of edges added. So, we will just keep a track of the number of edges added and after that we can display. So, this is one iteration over that is 1 node added with m edges. So, at this point you would like to display the graph take parameters, 3 parameters G comma the new node added is i the new edges added is where you keeping the this is new edges all right.

So, this should return the graph. So, I will return G. So, I think this function is done and we can start executing it ok.

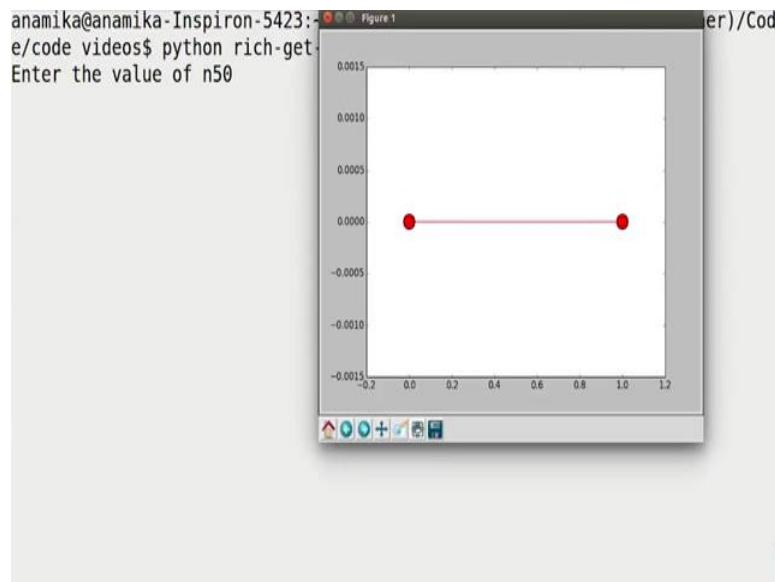
(Refer Slide Time: 27:17)

```
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_7_(Rich get richer)/Code
e/code videos$ python rich-get-richer_recording.py
Enter the value of n50
```

So, this should return the graph. So, I will return G. So, I think this function is done and we can start executing it ok.

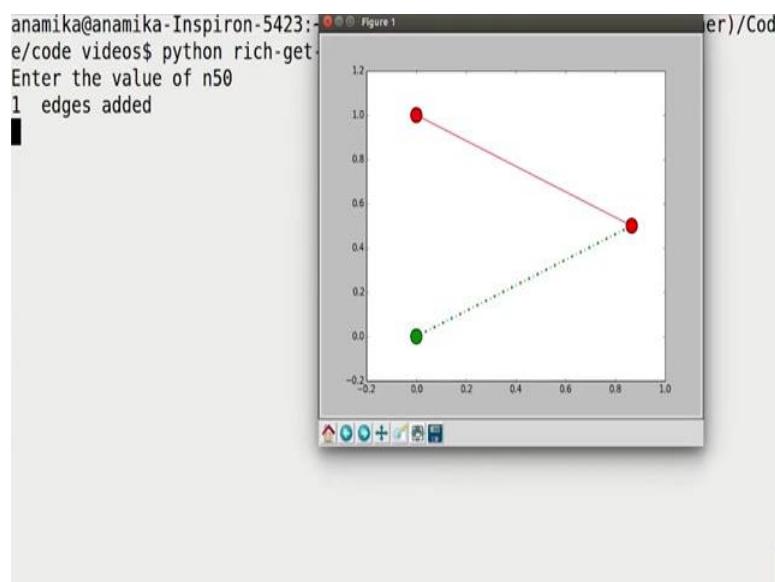
So, that is check it enter the value of n let me enter 50.

(Refer Slide Time: 27:23)



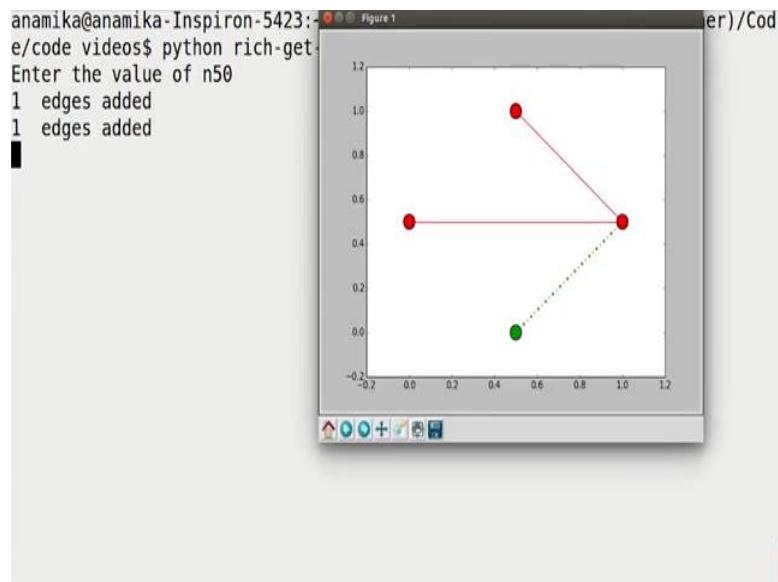
So this is the initial graph, it has chosen  $m_0$  to be true.

(Refer Slide Time: 27:32)



I will close this graph and see this is what I get in the first iteration, the new node is getting attached to one existing node. So, green is a new node and this dotted line indicates the new edges added.

(Refer Slide Time: 27:45)

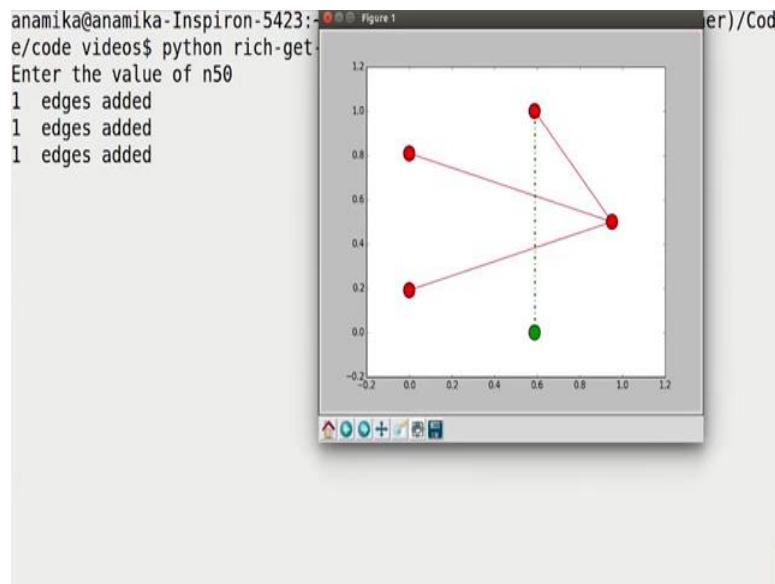


So, I am closing it and I see the second iteration the next new node is getting attached to the one of the existing edges.

So here, you can see there are 3 existing nodes, which are red and 2 of the nodes have degree 1 and 1 of the nodes has degree 2. So, the node which has degree 2 had more probability of getting attached to the new node, which is precisely what is happening having said that you cannot always be showing that always node, which has high degree will always big connecting it just a probabilistic phenomena right because, the referential attachment is probabilistic mechanism..

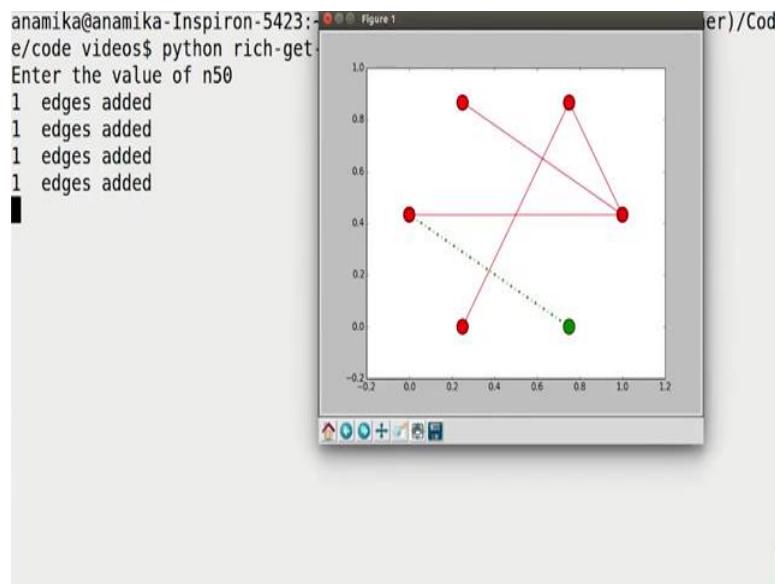
The new node is free to connect to any node in the network whether it has high degree or it has just single edge. However, if new node has a choice between degree 2 and a degree 4 node then degree 4 node will be twice as likely to be connecting to the new node as compare to the node with degree 2. So, that is what happens.

(Refer Slide Time: 28:54)



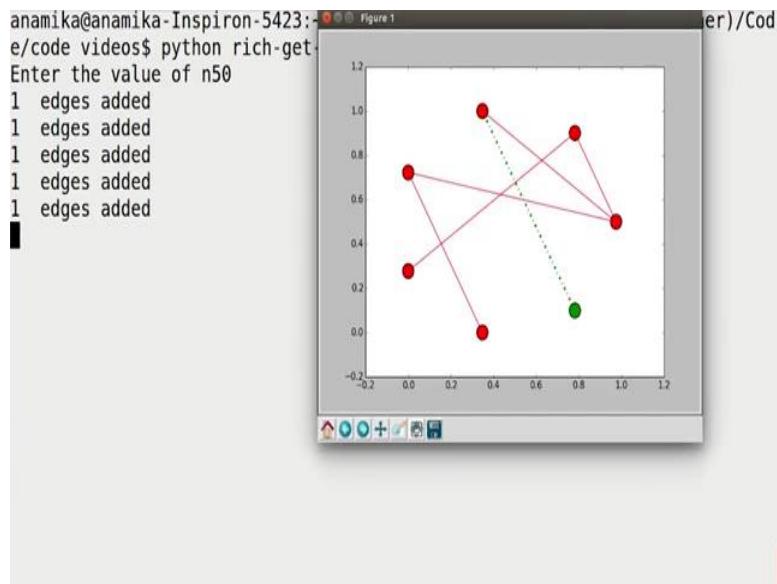
Now let us take the next iteration ok. So, the new node is getting attached to a node, which has degree 1. So, by chance it got connected to a node with less degree because, this is probabilistic. Let us check the next one.

(Refer Slide Time: 29:09)



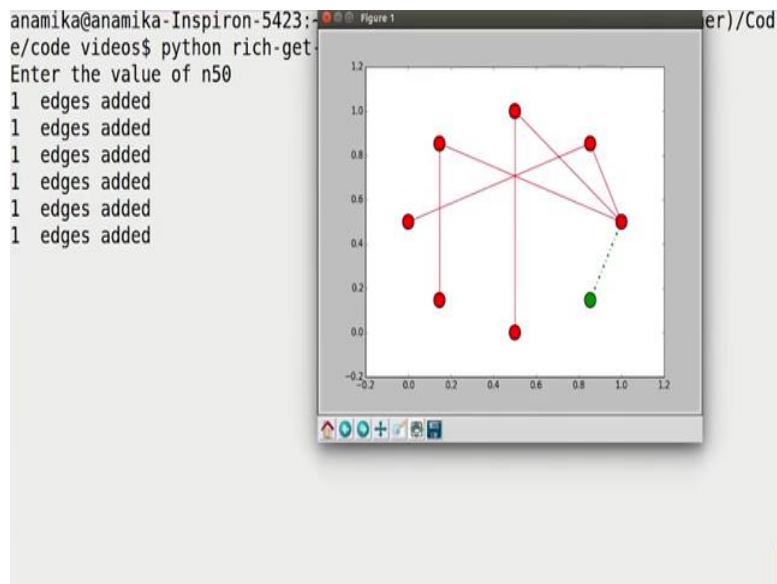
So, again the node is getting attached to a node with degree 1 because, there are few nodes.

(Refer Slide Time: 29:14)



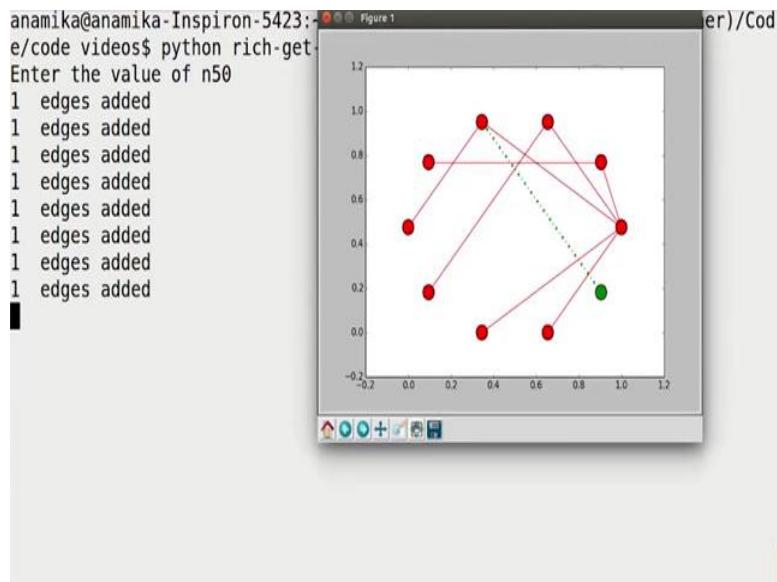
Now again the node is getting attached to two by chance a node with less degree. Let us see the next one.

(Refer Slide Time: 29:20)



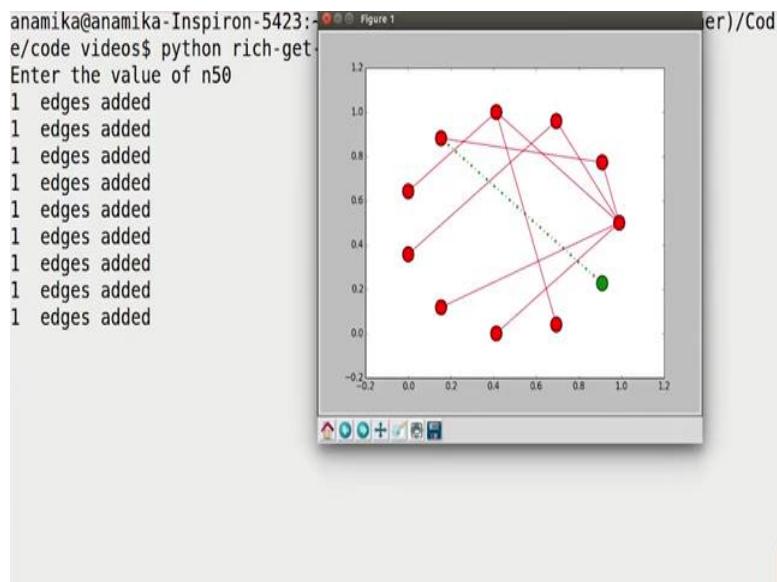
So now, this node has chosen an existing node with high degree. Let us check the next one. Again, it has to choose the node with high degree.

(Refer Slide Time: 29:30)



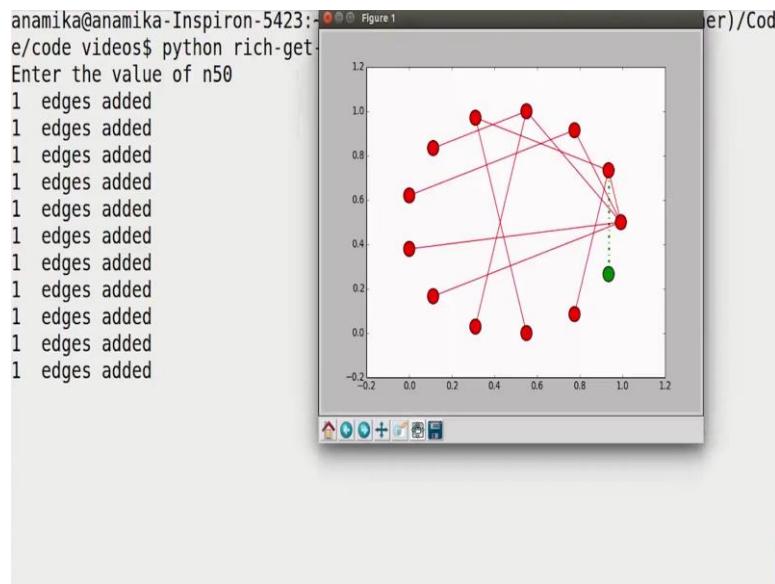
Now it has chosen our node with degree 2.

(Refer Slide Time: 29:34)



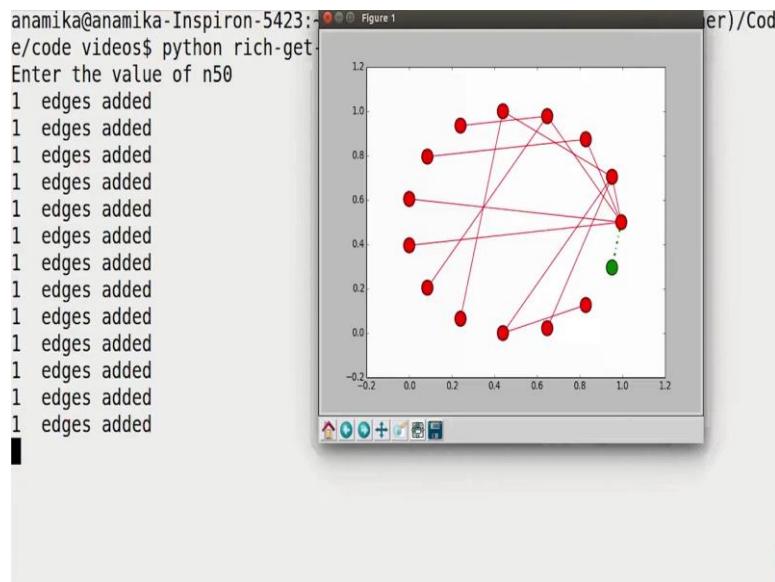
So, that keeps happening.

(Refer Slide Time: 29:36)



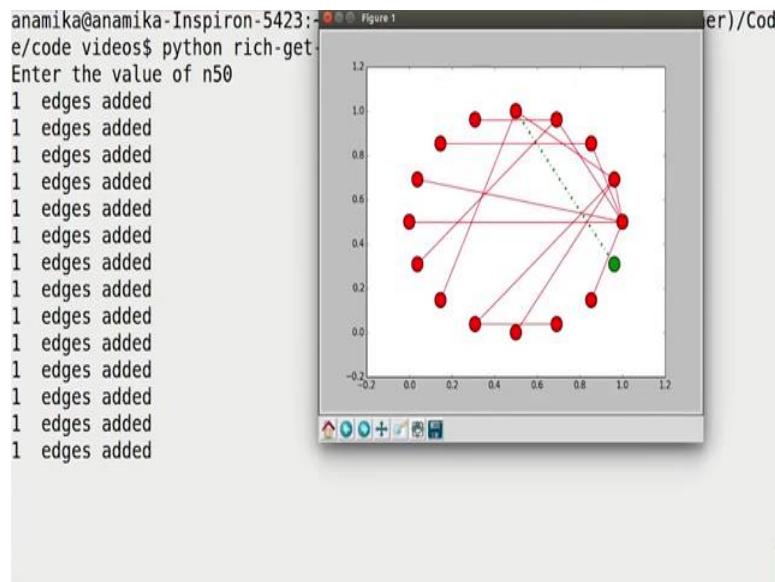
But overall, the nodes which have high degree will be connecting more to the new nodes.

(Refer Slide Time: 29:42)



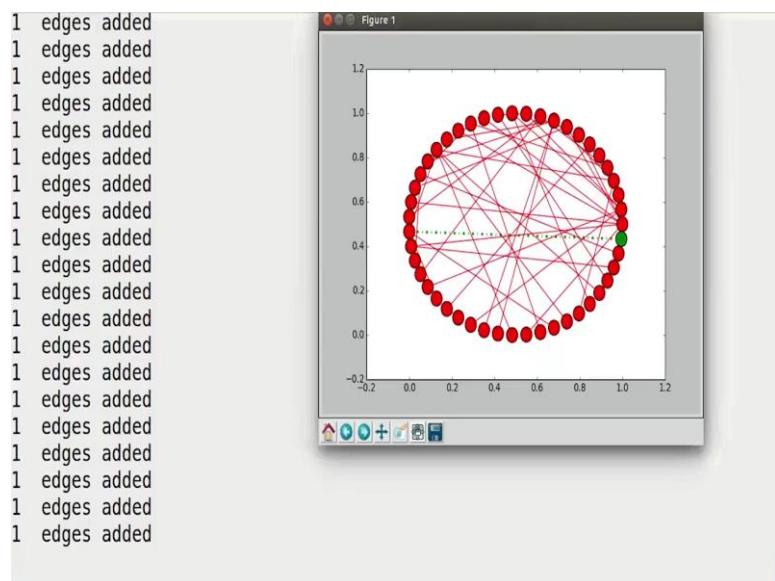
So, you can see here again happening there. So, it is again getting the connected to the node with high degree.

(Refer Slide Time: 29:47)



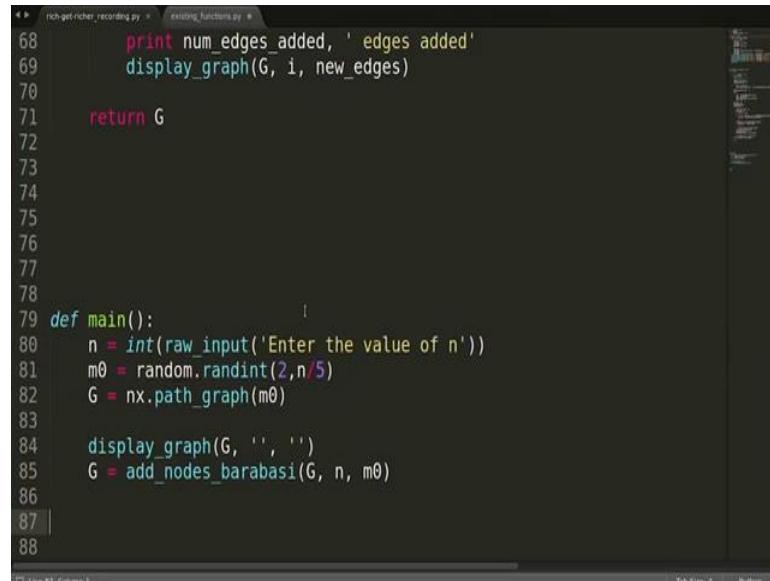
So, at each times and the network will not be symmetrical there will be more edges towards one towards some bunch of nodes, if that is not happening this network would have been symmetrical which is not so. So, I will have to keep closing this symbol 50 times. So, let me quickly do that you can keep observe with behavior ok. I should have taken the smaller network anyway.

(Refer Slide Time: 30:25)



So, this is the kind of structure that you are getting. So, a code is working fine alright.

(Refer Slide Time: 30:35)

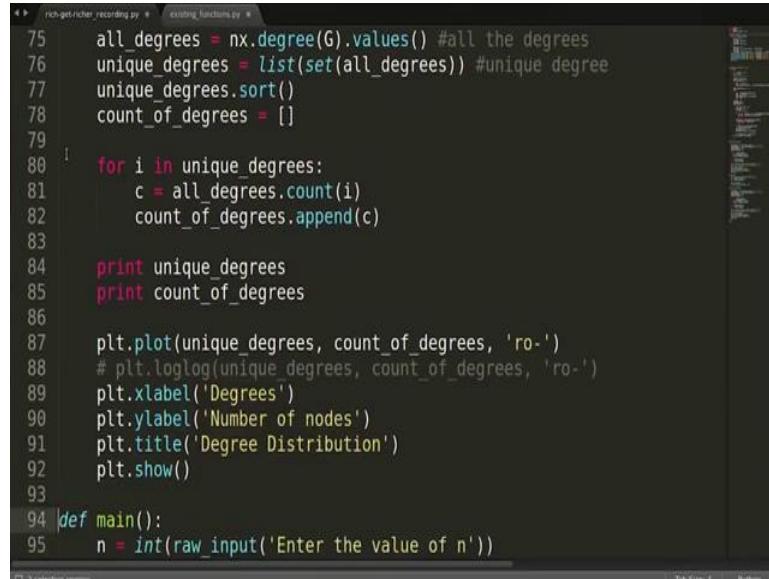


```
rich-get-richer_recording.py | existing_functions.py |
68     print num_edges_added, ' edges added'
69     display_graph(G, i, new_edges)
70
71 return G
72
73
74
75
76
77
78
79 def main():
80     n = int(raw_input('Enter the value of n'))
81     m0 = random.randint(2,n/5)
82     G = nx.path_graph(m0)
83
84     display_graph(G, '', '')
85     G = add_nodes_barabasi(G, n, m0)
86
87
88
```

Now, an important property of the network start follows Barabasi Albert model or preferential attachment or Rich-getting-richer phenomena is that the degree distribution follows power law ok. That means, that the number of nodes, which less get less degree are way to high number and the number of nodes with very high degree are very less in number. So, that is the sort of property that is networks follow.

Let us try to get the degree distribution for this network, we had already created this function for plotting the degree distribution in one of the previous videos, I am going to just copy paste from there and I will briefly explain that. So, I will copy paste it, I will quickly explain this ok.

(Refer Slide Time: 31:24)



```
75 all_degrees = nx.degree(G).values() #all the degrees
76 unique_degrees = list(set(all_degrees)) #unique degree
77 unique_degrees.sort()
78 count_of_degrees = []
79
80 for i in unique_degrees:
81     c = all_degrees.count(i)
82     count_of_degrees.append(c)
83
84 print unique_degrees
85 print count_of_degrees
86
87 plt.plot(unique_degrees, count_of_degrees, 'ro-')
88 # plt.loglog(unique_degrees, count_of_degrees, 'ro-')
89 plt.xlabel('Degrees')
90 plt.ylabel('Number of nodes')
91 plt.title('Degree Distribution')
92 plt.show()
93
94 def main():
95     n = int(raw_input('Enter the value of n'))
```

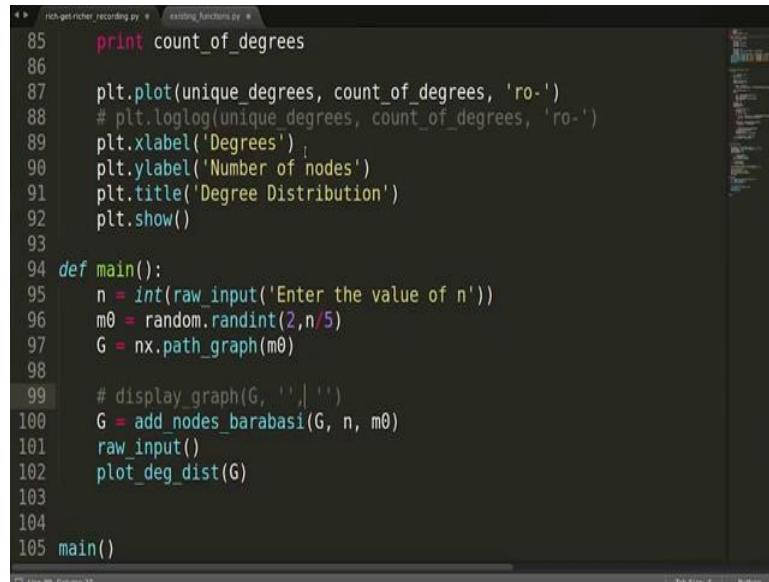
So, what you are doing here is since, you want to plot the degrees the distribution of the degrees, we need to get all the degrees. So, we are using the function `nx.degree(G)`, which returns a dictionary we are taking all the values from that dictionary that is the all the degrees, we are store in this list.

Now, out of that list we are maintaining list of unique degrees because, what we have to check we have to check the number of nodes having a particular degree. So, we should know what the unique degrees are basically, the all possible degrees that the nodes have in that network. So, that is what will be kept in this list and then we are sorting this list because, we have to plot them. And we will keep a track of the number of nodes with degree 1, degree 2, degree 3 and so on, where the x axis will have all possible degrees in sorted order..

So, we have to maintain a count of degrees for all possible degree in the network. So, we have created this, list now in order to populate this list we have started this loop for `i` unique degrees, we will take first element of unique degree. And we will check in all degrees how many elements that is how may nodes have that particular degree; we will make use of a function `count`. So, we writing `all_degrees.count(i)`, this will return as the number of nodes having the degree(`i`) and we are appending that to this list count of degrees and then we are simply plotting them.

So, x axis will have unique degrees, all possible degrees sorted, and y axis will have the count of degrees that is count of nodes having that particular degree. I am get displaying them in their color and this is for labeling title that is what is the function about ok.

(Refer Slide Time: 33:29)



```
rich-get_richer_recording.py | existing_functions.py
85     print count_of_degrees
86
87     plt.plot(unique_degrees, count_of_degrees, 'ro-')
88     # plt.loglog(unique_degrees, count_of_degrees, 'ro-')
89     plt.xlabel('Degrees')
90     plt.ylabel('Number of nodes')
91     plt.title('Degree Distribution')
92     plt.show()
93
94 def main():
95     n = int(raw_input('Enter the value of n'))
96     m0 = random.randint(2,n/5)
97     G = nx.path_graph(m0)
98
99     # display_graph(G, ' ', ' ')
100    G = add_nodes_barabasi(G, n, m0)
101    raw_input()
102    plot_deg_dist(G)
103
104
105 main()
```

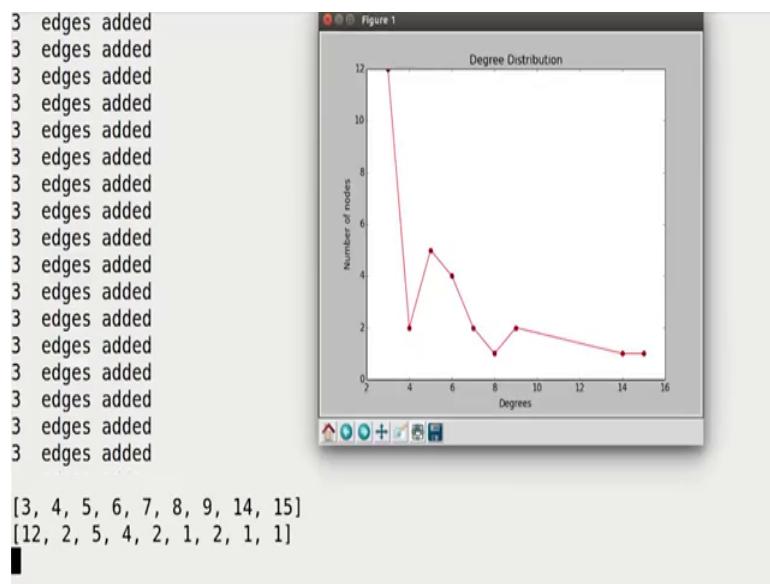
Let us call this function here i am sorry here in the end in name we are calling this function plot degree distribution G ok. So, let me let me ask the user to plus enter after getting all the plots after getting all the graphs.

(Refer Slide Time: 33:54)

```
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_7_(Rich get richer)/Code/code videos$ python rich-get-richer_recording.py
Enter the value of n
```

Let us check the functionality I will get. So, this is initial graph, and these are the nodes getting added I have to press this 30 times and after that we should get the distribution get everything works fine ok.

(Refer Slide Time: 34:16)



I press enter now this is the distribution that we are getting. So, this is sort power law in the sense that the nodes with less degree are very high number and the nodes with very high degree are very less in number. So, power law is basically the sort of plot since a number of nodes is less. We are not getting it very nicely calling power law, but nevertheless it is following the that law.

If you take good number of nodes say some 100 or say 1000, you will get a nice distribution you can also check that press check that. So, let us check let us execute this code for some higher value of n. So that, we can see the distribution nicely so in order to that I need not press the I need not close the graph multiple times, I am going to remove this displaying of the graph from all the places that is all that is that is only we will calling this display graph function. So, I have commented that.

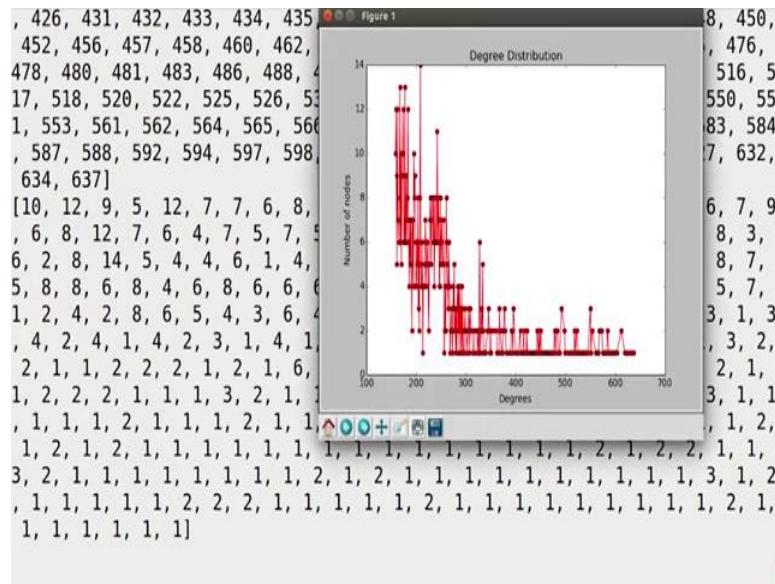
(Refer Slide Time: 35:27)

And let us call this with a high value of n say 1000.

(Refer Slide Time: 35:33)

So, 158 edges are getting added at every movement I think it is a very high number, I think, I should fix it on a smaller number there this is going on and on because, it has to go on for 1000 times. So, it is going on and on it is taking some chain let us see it is just finished, I am sorry.

(Refer Slide Time: 35:53)



So, I will press enter and I got the distribution. So, this is fall in solve power law as you can see right if you take a log plot of this, we will get this straight line. So, that is the whole idea about this.

So, as I told you the networks which follow preferential attachment follow power law as well and that is what we have observe there is well. So, that was the whole idea behind Rich-getting-richer phenomena. Here, we were adding the edges based on the degrees right, more degree more probability of getting connected. On the other hand, if you add the edges randomly what should happen? in that case, you will not get this power law distribution because, there is no preference to words getting a attach to a particular node, we are just randomly getting connected to the existing nodes.

So, in that case you will not the getting power law. In fact, you will getting normal distribution. So, in the next video we are going to implement random networks and we will see the kind of distribution that we obtain.

**Social Network**  
**Prof. S. R. S. Iyengar**  
**Prof. Anamika Chhabra**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Rich Get Richer Phenomenon**  
**Lecture - 123**  
**Implementing a Random Graph (Erdos– Renyi Model) - 1**

(Refer Slide Time: 00:05)

**Implementing a Random Graph**  
**(Erdos-Renyi Model)-I**

Anamika Chhabra



Hey everyone, in the previous video we implemented Barabasi-Albert model which follows preferential attachment that is also called rich getting richer phenomena. In this video sequence we are going to implement Erdos Renyi model which is used to create random networks. In this model each edge has a fixed probability of being present or absent independent of the other edges.

The idea is that we will start with the network with some n nodes with no edges amongst them. Now there are going to be  ${}^n C_2$  pairs of nodes in the network. We have to decide whether which pair of nodes should be added between the nodes of each of these pairs or not.

Now, let me explain that method to you using example. So, we will take the first pair of nodes and we will take a coin we will toss it. If we get a head, we will put an edge between this pair of nodes, if we get a tail, we will not put an edge between this pair of

nodes. Then we will take the next pair of nodes and we will toss the coin and we will do the same.

This we will keep doing for each pair of nodes in the network which is  ${}^nC_2$ . So, in the end we would have added some number of edges in the network. To generalize and to understand how it actually happens in the network in the model, we basically take some value of  $p$ . So, when we toss a coin which is not biased  $p$  is equal to 0.5 that is you get the head and tail with equal probability.

However, in the model we take the value of  $p$  from the user or we fix it this value, this can be any value between 0 and 1. So, for example, assume  $p = 0.3$ . In that case if you correlate with the example that I gave, the coin will be biased with the probability 0.3 that is head will appear with probability 0.3 and tail will appear with probability 0.7 ok.

So, that's the whole idea and you can imagine that if the value of  $p$  is high, the more number of edges will be added to the network and if the value of  $p$  is less than less number of edges will be added in the network, if value of  $p = 0$  no edges will get added. If the value of  $p = 1$ , then all the edges will be added to the network and it will be a complete graph. So, that is the whole idea behind this Erdos Renyi model. Let us look at this steps that we are going to follow for the implementation.

(Refer Slide Time: 02:47)

### Steps for Implementation

- 1 Take  $n$ , i.e. total number of nodes, from the user.
- 2 Take  $p$ , i.e. the value of probability from the user.
- 3 Create an empty graph. Add  $n$  nodes to it.
- 4 Add edges to the graph *randomly*.



So, there are 2 parameters that are required for implementing this model that is n and p, where n is the total number of nodes in the network and p is the value of probability as I told you, this will be input to the model and based on that only the edges will be added. So, we are going to take value of n from the user and then we will be taking the value of p from the user.

So, initially the network has only the nodes. So, we will take an empty graph and we will add n nodes to it and there are no edges in the network as of now. So, edges will be added to the network randomly Now, let me explain you how we are going to randomly add the edges.

(Refer Slide Time: 03:34)

### Steps to be followed for adding edges Randomly

- ① Take a pair of nodes.
- ② Get a random number  $r$ .
- ③ If  $r$  is less than  $p$ : Add this edge, else ignore.
- ④ Repeat step 1 for all possible pair of nodes.



So, we are going to take a pair of nodes ok. Basically we have to process all the pairs of nodes which is  ${}^nC_2$  pairs, we will take the first pair and we will get a random number r between 0 and 1. If the value of  $r \leq p$  we will add an edge between this pair of nodes and if value of  $r > p$  we will not add an edge between this pair of nodes and we will check the next one. We will keep repeating this step 1; step 1 2 3 actually ok.

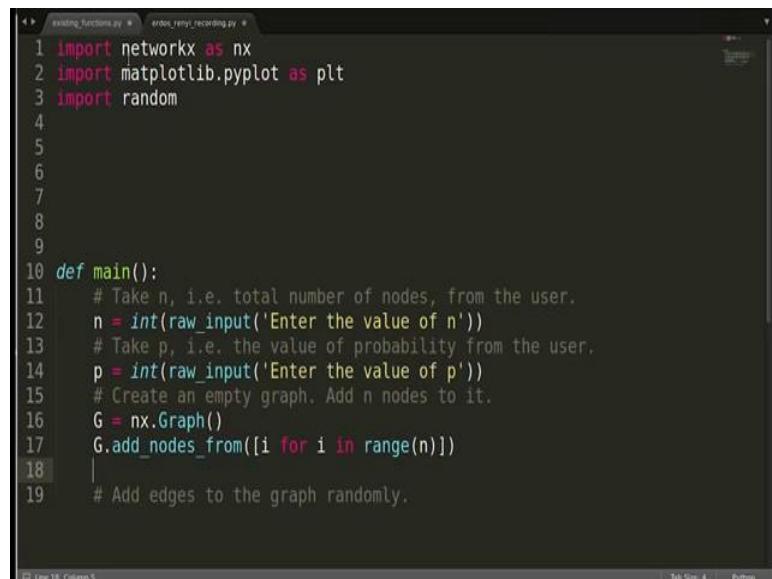
We will keep repeating the first 3 steps for all possible pair of nodes and by the end after we have processed all the pairs, we would have added certain number of edges to the network. So, this is the whole idea that we are going to follow for the implementation in the next video and we will also see the degree distribution that this kind of network follows.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Prof. Anamika Chhabra**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Rich Get Richer Phenomenon**  
**Lecture - 124**  
**Implementing a Random Graph (Erdos- Renyi Model)-2**

In the previous video we discussed Erdos Renyi model to create Random networks. We also discussed the sequence of steps that we will be following for the implementation. In this video we will do the implementation.

(Refer Slide Time: 00:21)



A screenshot of a code editor window titled "erdos\_renyi\_recording.py". The code is as follows:

```
1 import networkx as nx
2 import matplotlib.pyplot as plt
3 import random
4
5
6
7
8
9
10 def main():
11     # Take n, i.e. total number of nodes, from the user.
12     n = int(raw_input('Enter the value of n'))
13     # Take p, i.e. the value of probability from the user.
14     p = int(raw_input('Enter the value of p'))
15     # Create an empty graph. Add n nodes to it.
16     G = nx.Graph()
17     G.add_nodes_from([i for i in range(n)])
18
19     # Add edges to the graph randomly.
```

I have pasted the sequence of steps that we discussed in the previous video over here and we will start implementing them. To start with let me import the packages that we are going to need. We will be needing networkx, I am sorry we need to display the graph, so we will need matplotlib. We also need random package ok. Let us start the main. Let us take these steps 1 by 1. So, we have to take the value of n from the user and also we have to take the value of p that is probability from the user.

We can use the function draw input for that. So, we can write n is equal to draw input returns in string format. So, we need to convert it to int. Similarly, we will take the value of probability ok. The next step is to create an empty graph and add n nodes to it. Will

create an empty graph, we have to now add nodes to it n nodes to it, we will use the function G.add\_nodes from here we can pass a list. So, we must add n nodes, what we can do is will write i from i for i in range n is the total number of nodes.

So, 0 to n - 1 nodes will be added to the graph. At this point we can also display the graph we had already created this (Refer Time: 02:49) graph function in the previous video. So, we are going to reuse it. Let me copy paste that function from the previous videos content and we will use it as it is.

(Refer Slide Time: 03:05)



```
existing_functions.py * erdos_renyi_recording.py *
22
23 def erdos_renyi(G, p):
24     for i in G.nodes():
25         for j in G.nodes():
26             if i != j:
27                 r = random.random()
28                 if r <= p:
29                     G.add_edge(i,j)
30                     ne = (i,j)
31                     display_graph()
32                 else:
33                     continue
34
35
36
37 def main():
38     # Take n, i.e. total number of nodes, from the user.
39     n = int(raw_input('Enter the value of n'))
40     # Take p, i.e. the value of probability from the user.
41     p = int(raw_input('Enter the value of p'))
42     # Create an empty graph. Add n nodes to it.
```

So, initially when we display the graph it be all empty only the nodes will be there. The next step is to add the edges into this graph which only has nodes as of now, we will create a function for that.

We have to add the edges randomly as per Erdos Renyi model. So, let us create that function. Will pass the graph and the other thing that must be passed is p. So, in Erdos Renyi basically we pass 2 parameters n and p. We pass n here or we can compute the number of nodes from the graph itself. So, I am going to pass only these 2 parameters.

Let us create this function, now, if you remember from the previous video, we have to add the edges randomly. So, basically, we will take one pair of nodes and based on the probability p we will add an edge, or we will skip it.

So, let us take all pair of nodes, how can we do that? We will start we will use 2 for loops write for i in G.nodes. We got all the nodes here as the first element in the pair and we will write for j in G.nodes. We do not have to add the edges to from we do not have to add the self edges from one node to itself.

So, we will put a condition here, if  $i \neq j$  only then we will add the edge. We have to take a random number here and will compare that random number with probability p. So, we will take  $r = \text{random.random}$ , which will return us a value between 0 and 1.

So, if  $r \leq p$ , we will add that edge. So, we will write G.add\_edge. The edge is going to be i comma j and in case r is greater than p we will not add that edge. So, we will just write continue. So, these are the main things to add the edges. Now we also want to display the graph and we also want to keep track of the edges which are getting added at each iteration. So, let me do one more thing here.

So, we will keep track of the edges which are being added into this variable i comma j ok. And then I would like to display the graph, so let me call that function display graph which we have already added. Display graph has two parameter 3 parameters the graph as you can see here, the graph and the node which is being added and the edge which is being added.

(Refer Slide Time: 05:58)

```
1  #!/usr/bin/python
2  import matplotlib.pyplot as plt
3  import random
4
5  def display_graph(G, i, ne): #i is the new node added, ne is the list of edges
6      pos = nx.circular_layout(G)
7      if i == '' and ne == '':
8          new_node = []
9          rest_nodes = G.nodes()
10         new_edges = []
11         rest_edges = G.edges()
12     elif i != '':
13         # new_node = [i]
14         # rest_nodes = list(set(G.nodes()) - set(new_node))
15         rest_nodes = G.nodes()
16         new_edges = ne
17         rest_edges = list(set(G.edges()) - set(new_edges) - set([(b,a)]))
18     # nx.draw_networkx_nodes(G, pos, nodelist = new_node, node_color = 'red')
19     nx.draw_networkx_nodes(G, pos, nodelist = rest_nodes, node_color = 'blue')
20     nx.draw_networkx_edges(G, pos, edgelist = new_edges, edge_color = 'red')
21     nx.draw_networkx_edges(G, pos, edgelist = rest_edges, edge_color = 'black')
22     plt.show()
```

So, here nodes are not being added because, all the nodes are added to the graph initially only the edges are getting added at each iteration. So, if you note here and if you remember ne is a list of edges right and i is a single node.

(Refer Slide Time: 06:24)

```
existing_functions.py * erdos_renyi_recording.py *
25     for j in G.nodes():
26         if i != j:
27             r = random.random()
28             if r <= p:
29                 G.add_edge(i,j)
30                 ne = [(i,j)]
31                 display_graph(G, )
32             else:
33                 continue
34
35
36
37 def main():
38     # Take n, i.e. total number of nodes, from the user.
39     n = int(raw_input('Enter the value of n'))
40     # Take p, i.e. the value of probability from the user.
41     p = int(raw_input('Enter the value of p'))
42     # Create an empty graph. Add n nodes to it.
43     G = nx.Graph()
44     G.add_nodes_from([i for i in range(n)])
45
```

So, we have to keep we have to take care of that while we are passing the things here. Now we do not want to change the colour of the nodes, we will just pass an empty string here and we have to pass the edge which is being added in this iteration as a list. So, let me convert this into list and will what I will be passing.

And in case we are not adding any edge we want to display the graph at that moment as well because, we want to see whether in this particular iteration any edge was added or not, we want to display the graph there also. So, let me copy this in that case there will not be any new edge. So, we will keep this empty and everything else should remain same. Now let us go back to main and see if we can start executing this code.

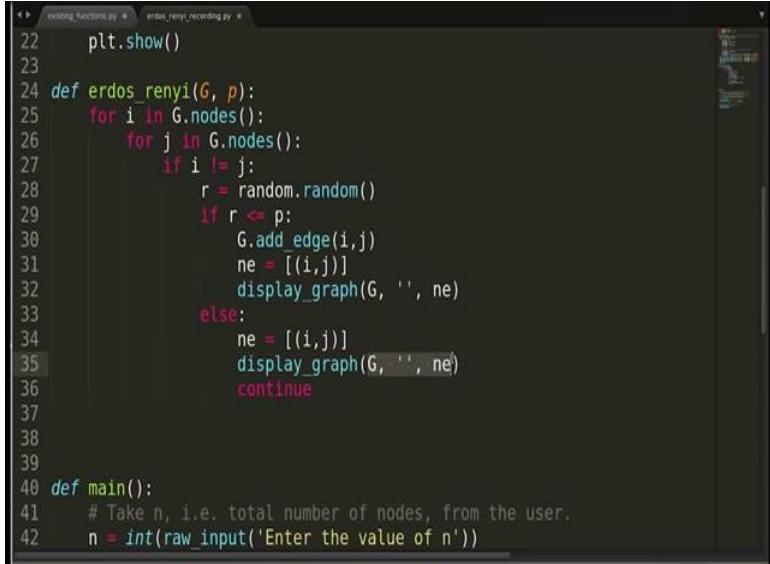
(Refer Slide Time: 07:19)



```
existing_functions.py * erdos_renyi_recording.py *
34     display_graph(G, '', ne)
35     continue
36
37
38
39 def main():
40     # Take n, i.e. total number of nodes, from the user.
41     n = int(raw_input('Enter the value of n'))
42     # Take p, i.e. the value of probability from the user.
43     p = int(raw_input('Enter the value of p'))
44     # Create an empty graph. Add n nodes to it.
45     G = nx.Graph()
46     G.add_nodes_from([i for i in range(n)])
47
48     # Add edges to the graph randomly.
49     display_graph(G)
50     erdos_renyi(G, p)
```

We are calling this function Erdos Renyi, we are calling display graph. There are 3 parameters to this function and in main we are passing only 1 parameter. So, let me pass the rest 2 parameters. So, this is the initial displaying of the graph where there are no new nodes, no new edges. Let us pass the empty strings there. So, the initial graph should be created.

(Refer Slide Time: 07:46)



```
existing_functions.py * erdos_renyi_recording.py *
22     plt.show()
23
24 def erdos_renyi(G, p):
25     for i in G.nodes():
26         for j in G.nodes():
27             if i != j:
28                 r = random.random()
29                 if r <= p:
30                     G.add_edge(i,j)
31                     ne = [(i,j)]
32                     display_graph(G, '', ne)
33                 else:
34                     ne = [(i,j)]
35                     display_graph(G, '', ne)
36                     continue
37
38
39
40 def main():
41     # Take n, i.e. total number of nodes, from the user.
42     n = int(raw_input('Enter the value of n'))
```

And when we are calling display graph from this function, we are passing G comma, we passing G and we passing empty string and a list ok. Let us see whether this function

were surprise for that. So, the initial condition this condition cannot be used because, both of these are not empty and if you go to the else part or i is empty here.

So, we see that we need to make some changes into this function in order to be able to use this for our set of conditions in this particular model. So, maybe we can write elif i = empty because this is the case that will be followed when we call display graph from Erdos Renyi function. Now we basically do not know want to change the colours of nodes because, every node is added initially. So, maybe we can let go off this thing ok.

Only the edges coloured has to be changed. So, this is fine, and this is fine and this should be fine let me see, we want to display the nodes right. So, let me remove the new nodes statement from here and the rest of the nodes will be displayed in red colour. Rest of the nodes is what we have to define here.

So, I will write rest and I will just copy from here ok. I am just playing around you will be able to do it yourself as well or you can write the function once again. I will just explain after I am done with this. I think they should do ok.

So, in our case when we will calling display graph function from the set of Renyi function, we are passing 3 things and in all the cases G is there and empty string is passed as i and the third one is the list. In case we are not adding the edge, this list is going to be empty.

So, let us see which condition will be followed. This condition will not be followed because, ne is not empty. We will be going into this and rest of the nodes is are going to be all the nodes and new edges yes we are passing a list there, yes this should work.

(Refer Slide Time: 10:27)

```
existing_functions.py  erdos_renyi_recording.py
38
39
40 def main():
41     # Take n, i.e. total number of nodes, from the user.
42     n = int(raw_input('Enter the value of n'))
43     # Take p, i.e. the value of probability from the user.
44     p = float(raw_input('Enter the value of p'))
45     # Create an empty graph. Add n nodes to it.
46     G = nx.Graph()
47     G.add_nodes_from([i for i in range(n)])
48
49     # Add edges to the graph randomly.
50     display_graph(G, '', '')
51     erdos_renyi(G, p)
52
53 main()
54
```

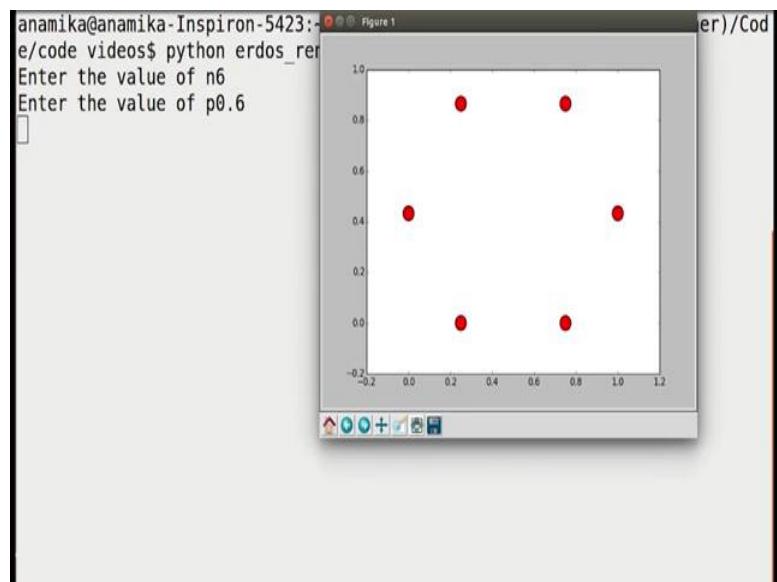
Let us go to the main and the value of n is going to be integer right and the value of probability is going to be between 0 and 1. So, that is actually no going to be integer. So, let us add a float there. Now let us call this main function. Now we can now check the functionality of this code.

(Refer Slide Time: 10:53)

```
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_7_(Rich get richer)/Code/code videos$ python erdos_renyi_recording.py
Enter the value of n6
Enter the value of p0.6
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_7_(Rich get richer)/Code/code videos$
```

So, let us check how it works. Enter the value of n, so let me pass a small value so that we can check the graphs at every iteration let me pass 6 for example. The value of p let me add 0.6 here we can pass any value between 0 and 1.

(Refer Slide Time: 11:15)



So, this is the initial graph where there are only nodes, 6 nodes are there, there are no edges. Now I am closing this and in the next iteration we are getting this edge between these 2 nodes and in the next iteration again we are getting this edge, we are getting an edge and in the 4th iterations as you can see we are not getting any edge. We got this edge and so in some iterations as you can see, we are getting some edges one edge and in some iterations, we are not getting.

So, I think I have to do this for 6 into 6, 36 times right. So, as you can see 36 times this loop will run and in some of the iterations i might be equal to j, so no edge might be added. In some of the iterations the r might be greater than p, so h might not be added. So, this was the functioning of Erdős-Rényi model that we created. I also told you that the degree distribution that Erdős-Rényi model follows is not power law rather it is normal distribution. We had created a function for plotting the degree distribution in one of previous video.

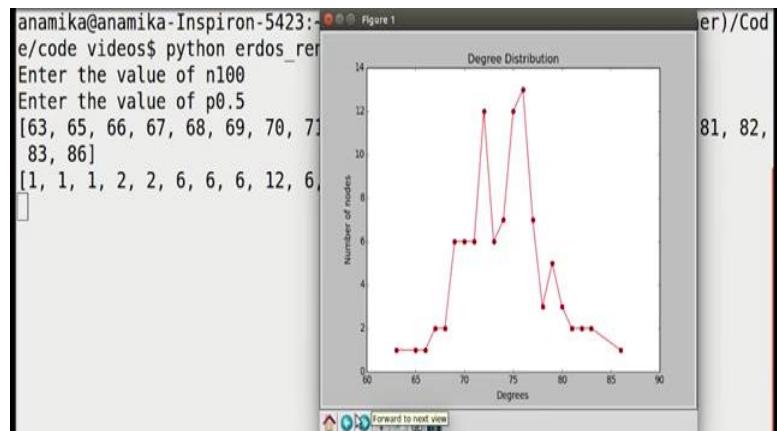
So, I am going to just use that function as it is, let me copy that and we will use this function plot, degree distribution, so let us use it here. In order to show you the degree distribution I will take some good number of nodes so that we can see a nice plot with a lot of values. So, I will remove the displaying of the graph so that, I do not need to press all tab 4 again and again. So, I am just commenting this displaying of graph here. Now let us check the functionality of this code.

(Refer Slide Time: 13:23)

```
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_7_(Rich get richer)/Code/videos$ python erdos_renyi_recording.py
Enter the value of n100
Enter the value of p0.5
[63, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82,
83, 86]
[1, 1, 1, 2, 2, 6, 6, 12, 6, 7, 12, 13, 7, 3, 5, 3, 2, 2, 2, 1]
```

So, we have to observe the degree distribution for an Erdos Renyi network. Let me add 100 nodes and let me add the probability to be 0.5.

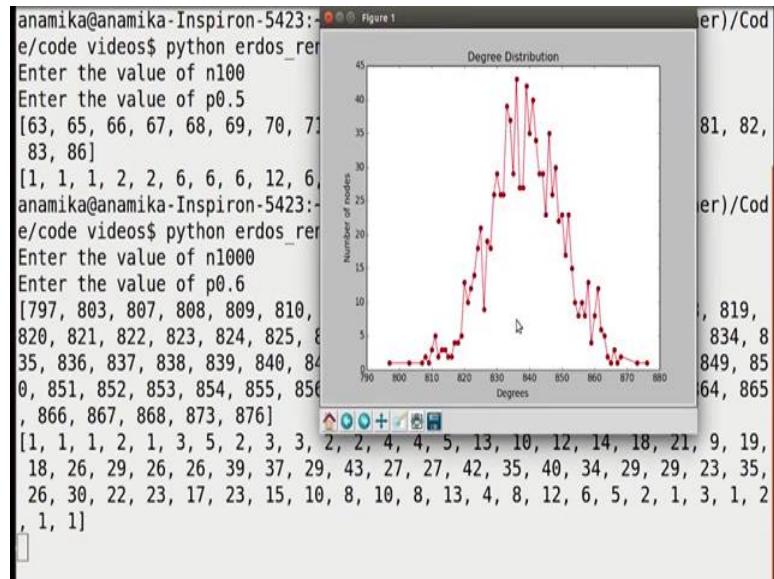
(Refer Slide Time: 13:33)



So, this is the sort of curve that we obtain. So, you can see since we were adding the edges randomly, there were no preferential attachment. We can see that there are very less nodes with very less degree and there are very less nodes with very high degree and there are very large number of nodes with medium degree. So, that is a sort of

distribution that we obtain, sort of normal distribution that we get in the case of a random network.

(Refer Slide Time: 14:08)



Let me try it once again for a greater number of nodes. Maybe I will add some 1000 nodes and the probability to be 0.6. So, again this is nice curve that we obtain. So, as you can see as you keep increasing the number of nodes, you can see the nice distribution that it takes ok. So, that was about the Erdos Renyi network to Erdos Renyi model to create the random networks and it follows the normal distribution.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Prof. Anamika Chhabra**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Rich Get Richer Phenomenon**  
**Lecture - 125**  
**Forced Versus Random Removal of Nodes (Attack Survivability)**

In this video we are going to implement Forced Versus Random Failure of Nodes which is also called Attack Survivability. We are going to take a network and we will be removing nodes from this network and we will observe the effect of removal of nodes from the network. We will be using two strategies to remove the nodes. In the first strategy we will remove the nodes randomly, and in the second strategy we will remove the nodes which are highly connected.

We will observe that when we remove the nodes which are highly connected, it requires very less iterations for the graph to become disconnected. This sort of information can be used by some agent for example, if he has the information of who is more connected and who is not. He, if in case he wants to damage the network for example, by spreading some sort of virus, he will not be attacking a random nodes rather, he will attack the nodes that are more connected, so that the network becomes infected to a greater extent.

So, there is one thing we will observe and implement, and we will also see one more thing in the context of the kinds of graphs. For example, the first thing that we checked on real world networks, real world networks sort of follow preferential attachment where there are there is existence of hubs in the network which are responsible for connection of most part of the network. So, when you remove then the graph becomes, becomes disconnected.

We are going to observe the same thing that is the removal of nodes on random networks as well. And we will observe that it does not make much difference on a random network when we remove the nodes randomly as compared to when we remove the nodes selectively; that is we remove the nodes which are having high degree.

That is because there is there are no hubs in random networks which are responsible for the connection of most part of the network; so the number of iterations that it will require

in the first case that is random removal and the second case that is selective removal. In the case of random networks will not make much difference. We are going to implement that and we will observe the number iterations that are required. Let us get started with implementation.

(Refer Slide Time: 02:23)

```
forced_vs_random_recording.py
1 import networkx as nx
2 import random
3
4 def remove_a_random_node(G):
5     nodes = G.nodes()
6     r = random.choice(nodes)
7     G.remove_node(r)
8     return G
9
10 def remove_a_high_deg_node(G):
11     degrees = G.degree()
12     nodes = sorted(degrees.items(), key = lambda x:x[1], reverse = True)
13     #where the tuple is having two things: (node, degree)
14     G.remove_node(nodes[0][0])
15     return G
16
17 def main():
18     G = nx.read_edgelist('email.txt')
19     print nx.info(G)
20
21 raw_input()
```

Let me import the packages first, we might also need random package, let us define our main. I already have downloaded a real-world network that is email network and that is an edge list format, we will be making use of that. This network is basically undirected. So, there is link from a node a to node b, if they exchanged any email with each other. So, that sort of network is what we will be using.

Let me read that network n x.read\_edgelist email.txt. Ok I have kept this in the same folder where I have get this file and then let me print some details about this network. So, I will write n x.info, so that I get the basic statistics about the graph. Let me call this function let us now check whether the graph has been read or not it has been converted into graph object or not.

(Refer Slide Time: 03:46)

```
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_7_(Rich get richer)/Code/videos$ python forced_vs_random_recording.py
Name:
Type: Graph
Number of nodes: 1133
Number of edges: 5451
Average degree: 9.6222
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_7_(Rich get richer)/Code/videos$
```

So, this is the basic statistics about the graph. It has more than 1000 nodes and more than 5000 edges and the average degree is there. So, let us get back here. So, we are going to remove the nodes from this network in 2 ways: let me print the first page random removal ok. We are going to randomly remove the nodes. Let me create a copy of the given graph because, we are going to remove nodes in 2 ways.

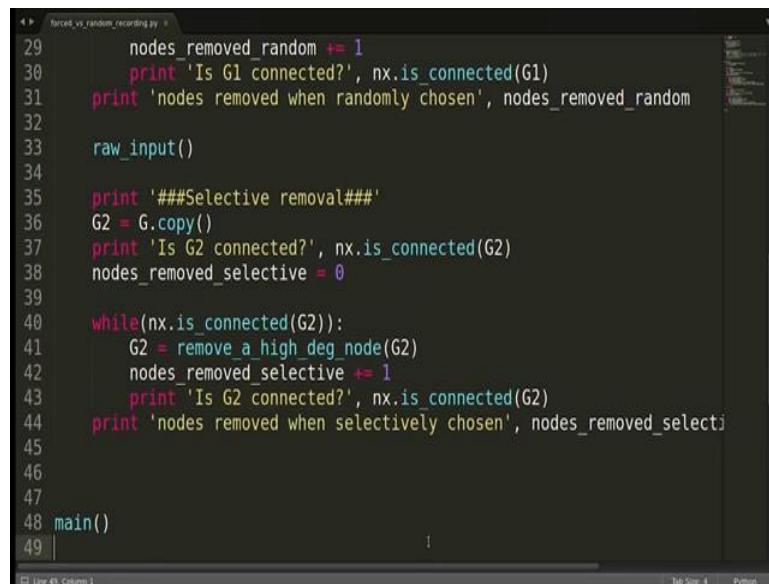
So, we will work on both the copies. So, this is the first copy G1 from which we will remove the nodes randomly. First let us check whether the graph is connected or not, is G1 connected ok, I will write `nx.is_connected(G1)` ok. We have to keep a track of how many nodes have to be removed to make the graph disconnected when we randomly remove.

So, let us create variable nodes removed when we randomly removed initialise to 0. Now while the network becomes disconnected, we have to keep removing the nodes. So, we will write `nx.is_connected(G1)`, we have to keep removing. So, we should better create a function for removing a node I create that function in a while. I will write G1 is equal to remove a random node.

So, this function I will just create, I am going to pass the graph here ok. And as you keep removing we will increment the counter for nodes remove random plus is equal to one and in every iteration we will keep checking whether graph is connected or not we will actually print that or let me just copy paste this sorry ok.

So, we at every iteration we are going to check whether graph is disconnected or not and in the end when we are done we will come out this loop and the graph becomes disconnected and at that point of time will print the number of nodes that needed to be removed to make the graph disconnected. Randomly chosen we will print nodes remove random ok. We will lastly used to press an enter here, as well as here ok. We are going to remove the in the second strategy, we are going to remove the nodes selectively based on the degree.

(Refer Slide Time: 07:06)



```

 29     nodes_removed_random += 1
30     print 'Is G1 connected?', nx.is_connected(G1)
31     print 'nodes removed when randomly chosen', nodes_removed_random
32
33     raw_input()
34
35     print '###Selective removal##'
36     G2 = G.copy()
37     print 'Is G2 connected?', nx.is_connected(G2)
38     nodes_removed_selective = 0
39
40     while(nx.is_connected(G2)):
41         G2 = remove_a_high_deg_node(G2)
42         nodes_removed_selective += 1
43         print 'Is G2 connected?', nx.is_connected(G2)
44     print 'nodes removed when selectively chosen', nodes_removed_selective
45
46
47
48 main()
49

```

So, let me just copy paste this entire thing and I will just made changes into that. So, in the second strategy, we are removing selectively, selective removal we will create another copy G2 of the graph. We will see whether G2 is connected or not. So, I am just making changes there. Nodes removed when selective removal was there. Let me write selectively, selective and again G2 and here also G2 and nodes remove the selective is G2 connected.

Actually, should I have written G only here, anyway nodes removed when random when selectively chosen. Nodes removed selectively right; I think there is a mistake here. Ok the only thing left is now to create this function that is 2, these 2 functions we have very minute, we have to remove the node selectively ok. So, remove node remove high degree node may be ok. So, we are yet to create these 2 functions, the first function is remove a random node and the second function is remove a high degree node.

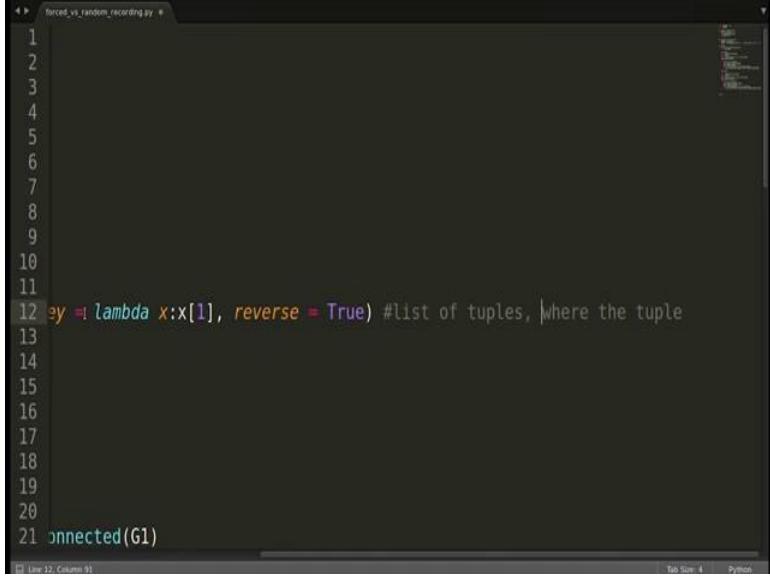
Let us create these functions ok. This function we have to create. So, out of all the nodes we have to choose 1 node randomly and we have to remove that. Let me get a list of all the nodes. nodes is equal to G.nodes, I have to randomly remove a node out of this while use random or choice out of nodes and we have to remove it. So, I will write G.remove node, which node to be removed or has to be removed and at this point we have remove the nodes from the graph and we will return it ok. Let us make a generalized just keep G here.

So, we pass the G and we return the G. So, whichever graph was input 1 more was removed, and it was returned. So, that function is being used here, everything seems fine, this syntax and return type, everything is fine. Next thing is to create this function remove a high degree node. Let us get this function now ok. What we have to do here? We have to see the degrees of the nodes and we have to sort them accordingly and the node which has highest degree will be removed.

How do we do this? Firstly, let us get dictionary having the degrees of all the nodes. So, G.degree ok, now, now out of this dictionary we have to get a sorted list of the key value pairs right. So, what we can do is a these are basically not nodes these are edges because, when we sort a dictionary we get list of tuples where every tuple is every tuple is actually key and value where key is node and the value is degree, so it make sense to call it nodes.

So, what we can do is we can use sorted function and we will pass degree degrees degree might be key or because, degree is the name of the function, so let us call it degrees. deg.items\_key = lambda the same old method that we used to get a sorted dictionary. And then X, we want to sort based on the value and we want to sort in descending order reverse = true, this should work I am sorry.

(Refer Slide Time: 11:32)



```
1
2
3
4
5
6
7
8
9
10
11
12 ey = lambda x:x[1], reverse = True) #list of tuples, where the tuple
13
14
15
16
17
18
19
20
21 onnected(G1)
```

So, what is nodes? Nodes basically is list of tuples where the tuple (Refer Time: 11:41) percent is here where the tuple is having 2 things node and the yes node and the degree right and it is sorted. So, the first node will be having the highest degree and that is what we should remove.

So, I will write `G.remove_node`, which node we have to remove out of the nodes list we have to take the first entry that be a tuple and out of that tuple we have to take the first value that is going to be the node that has to be removed right. So now, let us return `G` ok. This should work it is it no mistake. So, we are calling that function here `G2` is equal to remove a high degree node and we are incrementing the counter and we are connecting, that is all. It should work let us check let us check.

(Refer Slide Time: 12:43)

```
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_7_(Rich get richer)/Code/code videos$ python forced_vs_random_recording.py
Name:
Type: Graph
Number of nodes: 1133
Number of edges: 5451
Average degree: 9.6222

###random removal###
Is G1 connected? True
Is G1 connected? False
nodes removed when randomly chosen 1

###Selective removal###
Is G2 connected? True
Is G2 connected? False
nodes removed when selectively chosen 1
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_7_(Rich get richer)/Code/code videos$ python forced_vs_random_recording.py
```

So, this is a basic statistics as we saw earlier too, I am pressing enter ok. So, in this case we are randomly removing, and we have to remove only 1 node to make the graph disconnected. Now I am pressing enter and we are now selectively removing and again we need here only one node to make a graph disconnected.

So, by chance in this case as you see whatever node we removed randomly that was by chance a high degree node that is that is how the graph becomes disconnected. Let me run this function again and let me see again it is taking one it is surprising.

(Refer Slide Time: 13:26)

```
e/code videos$ python forced_vs_random_recording.py
Name:
Type: Graph
Number of nodes: 1133
Number of edges: 5451
Average degree: 9.6222

###random removal###
Is G1 connected? True
Is G1 connected? False
nodes removed when randomly chosen 1

###Selective removal###
Is G2 connected? True
Is G2 connected? False
nodes removed when selectively chosen 1
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_7_(Rich get richer)/Code/code videos$
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_7_(Rich get richer)/Code/code videos$ python forced_vs_random_recording.py
```

And again let me run this again.

(Refer Slide Time: 13:27)

```
###random removal###
Is G1 connected? True
Is G1 connected? False
nodes removed when randomly chosen 16
```

So, in this case you see randomly removal took 16 iterations and the selective removal took only 1.

(Refer Slide Time: 13:37)

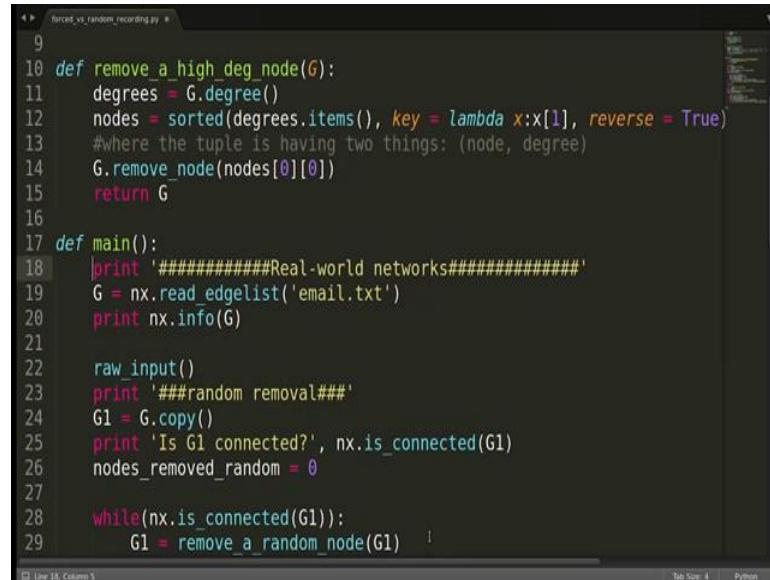
```
Number of nodes: 1133
Number of edges: 5451
Average degree: 9.6222

###random removal###
Is G1 connected? True
Is G1 connected? False
nodes removed when randomly chosen 13
```

Let me run this again in random removal, it took 13 and in then I in the selective one it took 1, I am just going to run it last time. So, again in randomly it took 17 and in selective it took 1. You can just keep checking that ok. So, that is one thing. In case of

real-world networks this sort of thing happens. Now, we are going to check the same thing on a random network ok.

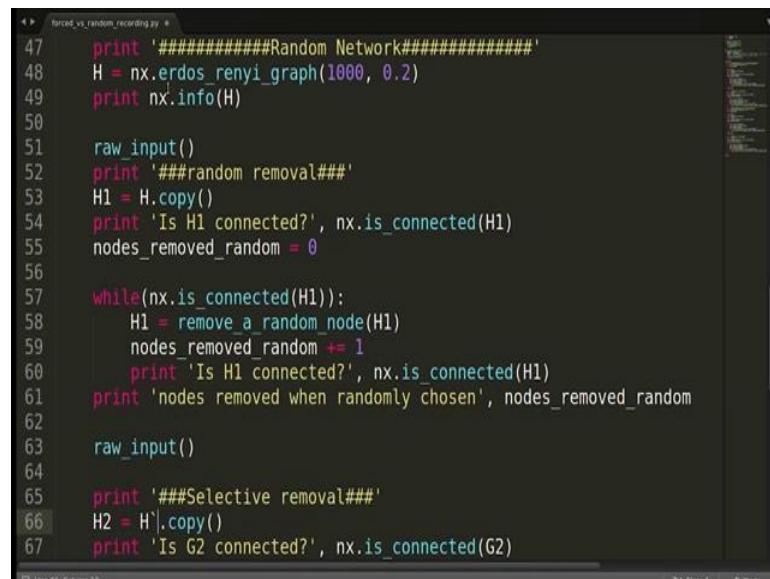
(Refer Slide Time: 14:10)



```
4 forced_vs_random_removal.py
9
10 def remove_a_high_deg_node(G):
11     degrees = G.degree()
12     nodes = sorted(degrees.items(), key = lambda x:x[1], reverse = True)
13     #where the tuple is having two things: (node, degree)
14     G.remove_node(nodes[0][0])
15     return G
16
17 def main():
18     print '#####Real-world networks#####'
19     G = nx.read_edgelist('email.txt')
20     print nx.info(G)
21
22     raw_input()
23     print '###random removal###'
24     G1 = G.copy()
25     print 'Is G1 connected?', nx.is_connected(G1)
26     nodes_removed_random = 0
27
28     while(nx.is_connected(G1)):
29         G1 = remove_a_random_node(G1)
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
```

So, let me let me write here that the all this was for all this was for real world network or a network which basically follows power law distribution and preferential attachment, real world networks right. This does not look good, let me write this does not, let me write this. So, this was for real world networks. This looks very bad anyway, let us go ahead.

(Refer Slide Time: 14:49)



```
4 forced_vs_random_removal.py
47     print '#####Random Network#####'
48     H = nx.erdos_renyi_graph(1000, 0.2)
49     print nx.info(H)
50
51     raw_input()
52     print '###random removal###'
53     H1 = H.copy()
54     print 'Is H1 connected?', nx.is_connected(H1)
55     nodes_removed_random = 0
56
57     while(nx.is_connected(H1)):
58         H1 = remove_a_random_node(H1)
59         nodes_removed_random += 1
60         print 'Is H1 connected?', nx.is_connected(H1)
61     print 'nodes removed when randomly chosen', nodes_removed_random
62
63     raw_input()
64
65     print '###Selective removal###'
66     H2 = H1.copy()
67     print 'Is G2 connected?', nx.is_connected(G2)
```

So, next we are going to do the same things for a random network. Now how can we create a random network? Ok in the previous videos we created an Erdos Renyi network. We can use that code to create a network and we can also use a function from network x where we have a function where we pass the values of n and p and we get a random network.

So, in order to tell you the function, I will use that function only over here. Let me call the graph H. I wrote nx.erdos\_renyi network. So, this is a I am sorry, graph. So, this is the name of the function and let me pass some n, the total number of nodes and the value of t the probability. This should give us random network. Let me print the basic information of H.

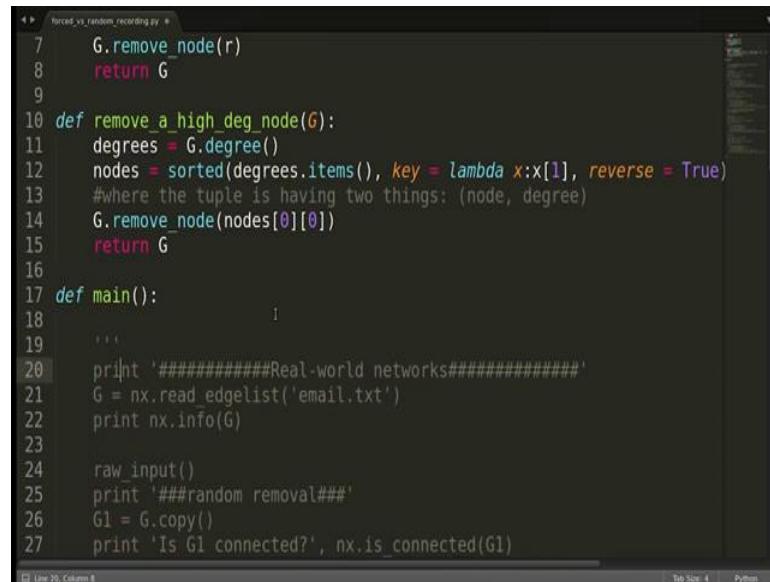
After that we want to remove the nodes as the same way we did earlier. Let me copy paste the entire code here and we will make changes into that to this right. So, I am just pasting the entire thing here, random network starting here only. Raw input and then random removal 1 the graph name is H, what to write to ok. We are making copy of H and then we are checking whether H1 is connected or not and nodes removed is find and then nx is this connected H, (Refer Time: 16:47) I could have actually use here replace like replace function here, it is ok. So, everything else remains the same, I am just changing the graph here ok.

(Refer Slide Time: 17:00)

```
forced_vs_random_removal.py
57     while(nx.is_connected(H1)):
58         H1 = remove_a_random_node(H1)
59         nodes_removed_random += 1
60         print 'Is H1 connected?', nx.is_connected(H1)
61     print 'nodes removed when randomly chosen', nodes_removed_random
62
63     raw_input()
64
65     print '###Selective removal###'
66     H2 = H.copy()
67     print 'Is H2 connected?', nx.is_connected(H2)
68     nodes_removed_selective = 0
69
70     while(nx.is_connected(H2)):
71         H2 = remove_a_high_deg_node(H2)
72         nodes_removed_selective += 1
73         print 'Is H2 connected?', nx.is_connected(H2)
74     print 'nodes removed when selectively chosen', nodes_removed_selective
75
76
77 main()
```

I actually should have used replace method, but I think we are done we will almost done now, check it last one alright. So, I think we have good to go. We have just taken random network now and we are applying random removal as well as selective remove in this case. And we are continuing the number of iterations that are required in both the cases ok.

(Refer Slide Time: 17:39)



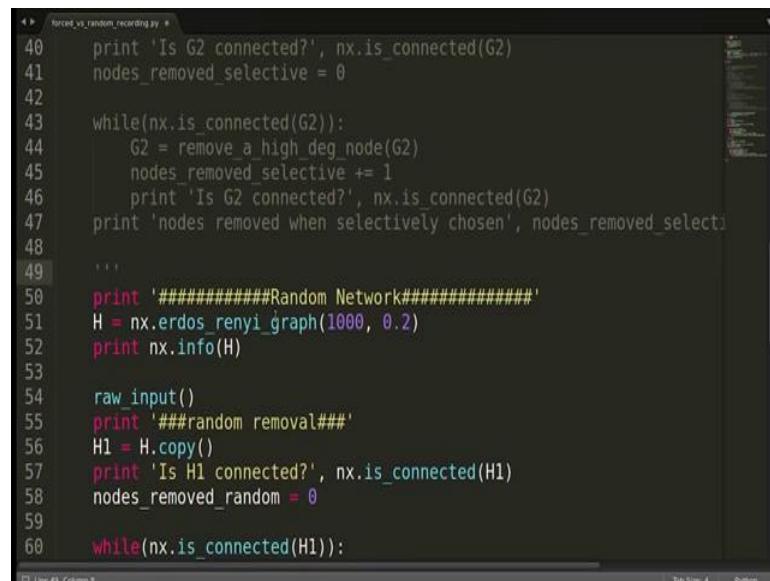
```

4 h forced_vs_random_removal.py *
7     G.remove_node(r)
8     return G
9
10 def remove_a_high_deg_node(G):
11     degrees = G.degree()
12     nodes = sorted(degrees.items(), key = lambda x:x[1], reverse = True)
13     #where the tuple is having two things: (node, degree)
14     G.remove_node(nodes[0][0])
15     return G
16
17 def main():
18     """
19     """
20     print '#####Real-world networks#####'
21     G = nx.read_edgelist('email.txt')
22     print nx.info(G)
23
24     raw_input()
25     print '###random removal###'
26     G1 = G.copy()
27     print 'Is G1 connected?', nx.is_connected(G1)

```

Since, we have already observed the behaviour on a real-world network, let me comment that part. So, that we are able to concentrate on only real-world network.

(Refer Slide Time: 17:45)



```

40     print 'Is G2 connected?', nx.is_connected(G2)
41     nodes_removed_selective = 0
42
43     while(nx.is_connected(G2)):
44         G2 = remove_a_high_deg_node(G2)
45         nodes_removed_selective += 1
46         print 'Is G2 connected?', nx.is_connected(G2)
47     print 'nodes removed when selectively chosen', nodes_removed_selective
48
49     """
50     print '#####Random Network#####'
51     H = nx.erdos_renyi_graph(1000, 0.2)
52     print nx.info(H)
53
54     raw_input()
55     print '###random removal###'
56     H1 = H.copy()
57     print 'Is H1 connected?', nx.is_connected(H1)
58     nodes_removed_random = 0
59
60     while(nx.is_connected(H1)):

```

So, I am commenting this part up to which point this point ok.

(Refer Slide Time: 17:50)

```
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_7_(Rich get richer)/Code/videos$ python forced_vs_random_recording.py
#####
#Random Network#####
Name: gnp_random_graph(1000,0.2)
Type: Graph
Number of nodes: 1000
Number of edges: 100063
Average degree: 200.1260

###random removal##
```

So, let us check our code. So, this is for random network we have 1000 nodes and these many edges. I am pressing enter, random removal is taking place.

(Refer Slide Time: 18:01)

```
Is H2 connected? True
Is H2 connected? False
nodes removed when selectively chosen 928
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_7_(Rich get richer)/Code/videos$
```

So, you see this is taking a lot of time it is taking a (Refer Time: 18:07) number of time. It is actually going on and on as you can see it is going on and on still going on ok. So, the number of nodes we had to remove from a random network when we randomly choose them was 985. So, (Refer Time: 18:28) 985.

I am pressing enter here. Now we are selectively removing that is we are choosing the nodes with high degree and then we are removing them. It is again going on and on surprisingly it is going on and on; let us go on and on 928 as you can see. So, in the case of random network when we randomly remove the nodes, it took some 938 nodes to make the graph disconnected, and in case when we remove the nodes based on their degrees that is we remove the high degree nodes and every iteration, it again took as 928 nodes to make the graph disconnected.

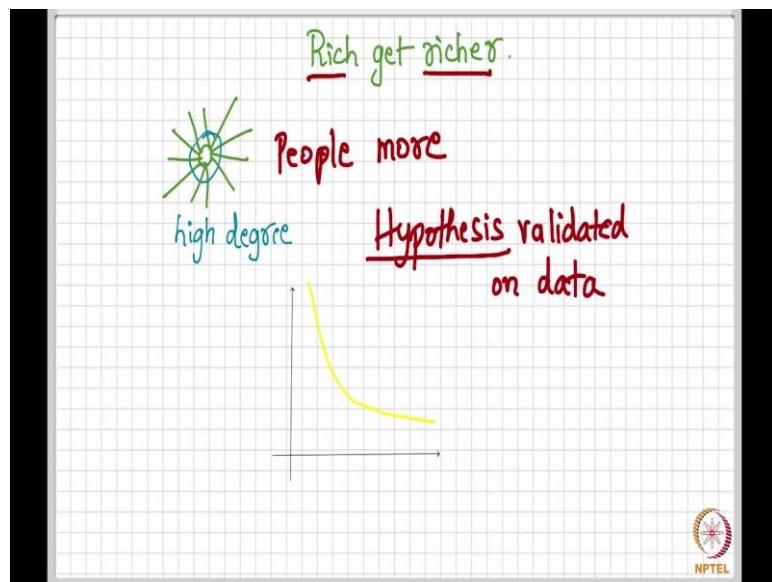
Now that is happening because, there are no hubs in random networks which are connecting a lot of nodes. So, all the nodes are sort of equally likely because, equally likely in the sense of the number of nodes that they are connected to. So, there is no preference because, the edges were added randomly and there is no preference towards addition of nodes to some particular nodes in this.

So, that is a sort of difference that we observe in case of real-world networks as well as random networks, because real world networks are not random and they actually exhibit some sort of preference towards the nodes.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Rich Get Richer Phenomenon – 2**  
**Lecture - 126**  
**Rich Get Richer - A Possible Reason**

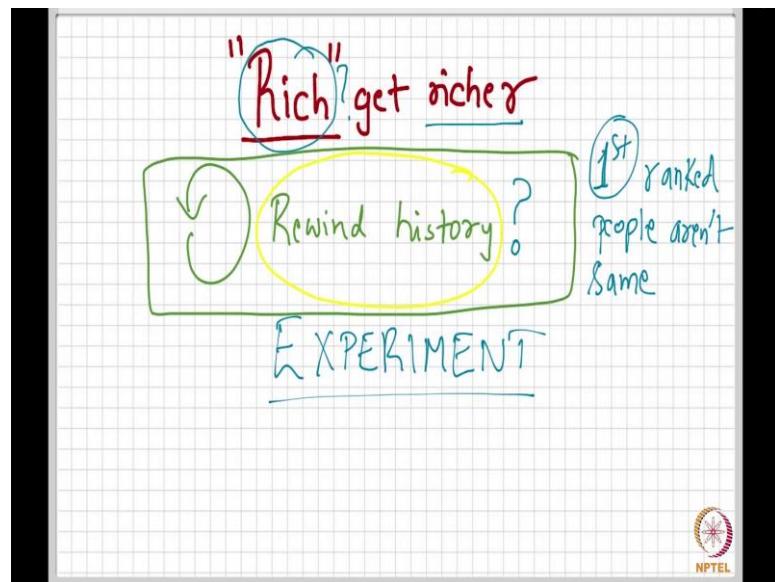
(Refer Slide Time: 00:10)



We saw in the previous lectures; the paradigm called the Rich Get Richer. It was a rich get richer. So, what was happening there? People with high degree they can do attract people more people who have high degree. If they have high degree, then they have more propensity to attract more people towards them. So, a person who has more friends will attract more friends more friends. Since, he has more friends he will attract more friends and so on and so forth. This is a sort of a hypothesis; this is a hypothesis validated on several data; validated on data. That does not make this hypothesis true, but it is highly suspected that this phenomena is indeed happening.

What phenomena? The phenomena that the rich get richer because of which we observe the power law happening right, the power law as in the drop in the curve if you remember the drop in the curve correct, fine. So, now, we will try to ask this question; this question slightly unrelated from networks, but related of course, and quite an intriguing question.

(Refer Slide Time: 01:44)



When we say rich get richer right, how did this people become rich in the first place to begin with right? I mean you say that people who have more friends attract more friends. How it they start getting this so called more friends, right?

So, let us take an example. If I were to rewind history, do you think Alice in Wonderland still would be a very popular story ever written? Do you think JK Rowling's Harry Potter would still be a bestseller right? There are many such authors who wrote many novels and stories which did not make it to the best seller status just the way Harry Potter did. If I were to re rewind history that is my question, if I have to rewind it do you think successful people will continue to be successful? Well, how do you even rewind history, how do you even mimic such a experiment, how do we try to sort of re do a stuff and then see if popular people continue to be popular?

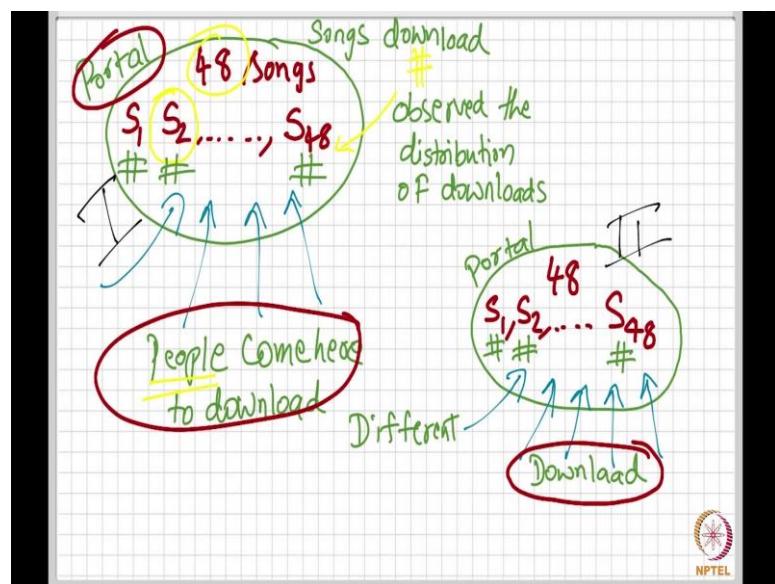
If history where to remind itself and then play once again, do you think Britney Spears would still be the best? Do you think Obama will again become the president of the United States? With this is very tough to answer right. So, probably, one should think of an experiment which sort of gives the tries to attempt an answer to this question. So, what kind of experiment can one do; is my question. What experiment can you think of?

So, I hope my question is clear to you all. The question is people who are rich tend to become richer right. Who are these rich people? Everybody start a fresh with 0 money right. So, who are these people who become famous? JK Rowling before her first work

was not known to people right and she got the people got to know of her through her Harry Potter's work, the philosophers stone which is the part 1 of Harry Potter.

And then she became really famous and whatever she wrote sold. Of course, the work was good too but then still it gave her an edge because she was successful with her first attempt right. So, is this true in general that if you are good you will make it or is it because by luck you make it the first time and you become rich and you become richer because you were rich and so on. There was a nice experiment conducted which I am going to describe right now.

(Refer Slide Time: 04:49)



People took some music songs for download, they took 48 songs ok; song 1, song 2 up to 48 songs. And a bunch of people were asked to come to this portal, this is a portal basically, this is a portal. Let me write that down. This is a portal for songs download, portal for songs for the downloading of the songs ok; songs down download people come here to download songs.

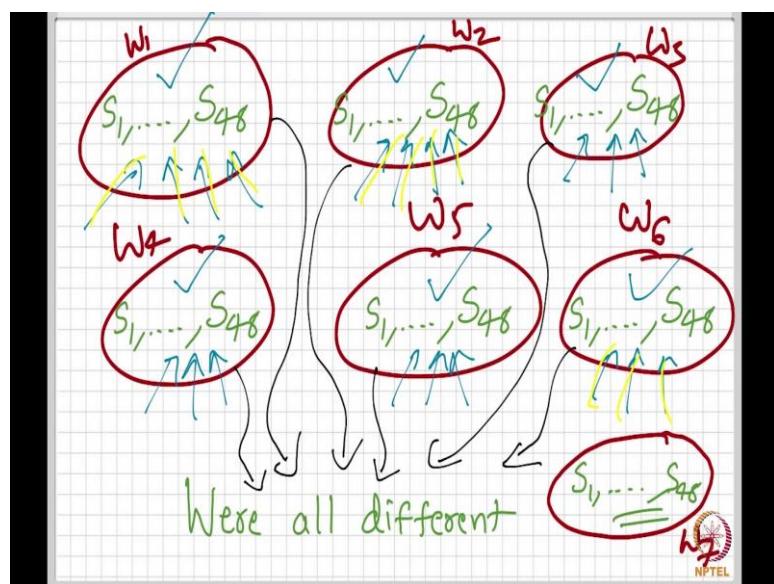
Now, with time people observe that song 2 was being downloaded a whole lot. People do get to see what song is downloaded how many times then the number of downloads is visible and people did see this and eventually it was observed that songs that were downloaded more often became more and more popular ok.

People observed the distribution of downloads, they observed the distribution of downloads. Some of them were really popular and some of them were not very popular ok. Now, here goes our question. If I were to take another world ok, a parallel world as I call it and then take the same 48 songs as I took here and display them for downloading on a different portal. Here was a portal ok, this is a different portal another portal ok. And then again I ask I let some new bunch of people to come here random bunch of people to see and then they download the songs as usual as in the previous case and now I see the distribution of how much how many times song 1 was downloaded, how many the number of times song 2 was downloaded, the number of times song 48 was download and so on, all the songs.

I have made note of this as well right in the previous case ok. So, now, what do you expect? Let me be clear. The people here are different they are different from people here to download the songs.

This is one world this one is world number 1 this one is world number 2 ok.

(Refer Slide Time: 08:019)



So, if I want to create many such worlds let us say world 1, world 2, world 3, let say world 4, 5 and 6; I have six different worlds and then all these worlds I have the same bunch of songs; song 1 to song 48 I had 48 songs ok, same thing everywhere alright. So, let me just copy paste and I have the same thing here and then the same thing here in w3 and w4 I have the same bunch of songs and assume I have this 5 what is call the parallel

universe, where different bunch of people are allowed to come and download the songs. Here people are different; here people are different and so on.

So, please note this people do not know about other portals. There are six portals and different people assigned to different places. So, people were assigned here are different from people of same here and people were same here and so on. So, it was observed that the distribution; the distribution of w1 and the distribution of w2 and that of w3 and that of w4 and that of w5 and that of w6 were all different.

What do I mean by that? By that I mean winner here the most downloaded song here was different from most downloaded song here was different from what it was here different from here different from here and different from here.

Now, pause for a minute and think what is happening? You have 6 different worlds where different people were assigned the task of downloading music and rating the music and they it was observed that the number of downloads of a song was different in different portals. Now, why is this? What just happened here?

(Refer Slide Time: 10:39)



The reason here is precisely what we were discussing before, the rich getting rich richer phenomenon; rich get richer phenomena. The fact that you displayed the download information or let us say rating information right, you displayed the download information the download information not download information download

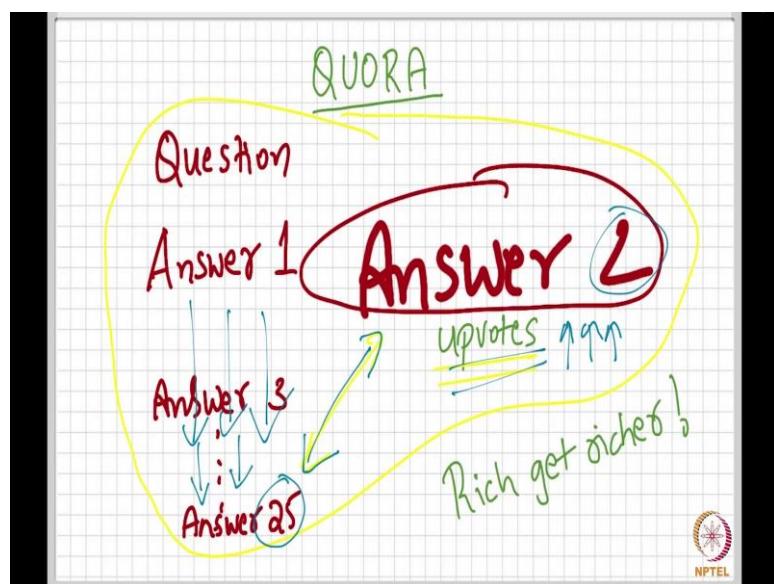
information and people saw that and sort of became biased right. So, highly downloaded songs it was displayed there of course, this are the highly downloaded songs biased new people new people to download these songs and hence they downloaded and that added to the download count once again so on and so forth ok.

So, there was an there was an experiment which involved another world with the same songs where no information of the download was display and yet again the distribution was very different here ok. So, this was the world let say 7th world alright ok. So, what does it signified?

Signifies that the fact that something is good results in that something becoming further good right. Good gets good becomes good even further rather the right world is better. Good gets better and better gets let say more better sounds like a wrong English grammar, but there is there is no other way for [laughing] me to explain this; good gets better, better becomes more better and more better becomes more and more better.

Why? That is because if something is very good. People tend to get attracted to it and they added to the popularity of the stuff right. So, this is called the rich get richer phenomena. This is observed in almost all the places. I will give one instance where you can relate this phenomenon very well and that is your Quora portal.

(Refer Slide Time: 13:02)



If you did not know Quora is a portal where people ask questions. There is a question and there is an answer. There is not just an answer; there is an answer 1, answer 2, answer 3 and so on. So, what happens in Quora is just in case answer let say answer 2 becomes really really popular ok; answer 2 becomes really popular; you know what happened then ok? The sort of dominates this question thread, answer 2 becomes popular and it gets more what is called upvotes and more people see answer 2 and they will upvote it all the more.

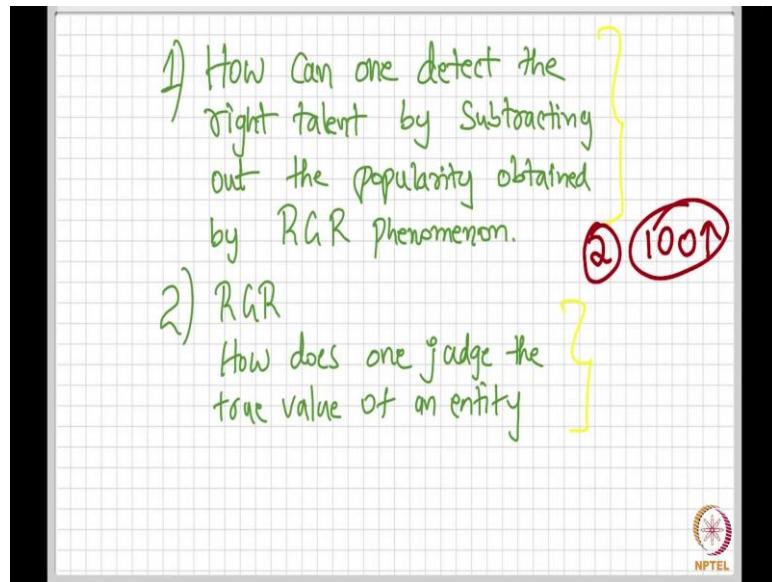
And just in case a new answer comes, let us say answer 25, which is much better than this answer; nobody at all even if it is much better than answer 2. Assume answer 25 is way better than answer 2, nobody will see it. It will just be flushed down somewhere because Quora displays answers based on upvotes right.

At least so far, the main logic behind displaying the answers first is based on the sorted order of their upvotes. So, more the upvotes, more people will see it and more will they upvote the answer and answer become popular and no answer that is below answer 2 or lower will ever be displayed. So, do you see the rich get richer phenomena happening here.

Now, one can ask several questions here. So, getting back to a previous question let me go back slide where we were discussing JK Rowling's popularity etcetera. So, if I write to rewind history right, it is most probably true that people the first ranked people are not the same. First rank does not first few rank people. Although people who do really well will continue to do really well, people who do not do well will continue to do if continue not to do very well, but the first rank, but the top few ranks will actually shuffle.

By this I mean, Obama may not become a president if history were to rewind, JK Rowling may not become the bestselling author if history were to rewind and our experiment sort of gives a nice ideas to why the our hypothesis is true that the person who starts of afresh might become rich out of luck to begin with and then rich get richer and he or she becomes richer and richer and richer and attracts more um popularity and becomes all the more richer ok.

(Refer Slide Time: 16:40)



A good and nice question to ask is how do you how can one detect the right talent by subtracting out subtracting out the popularity obtained by this rich getting richer phenomenon. So, how can one do this? This is this is one nice question that one can ask this is one thing.

And second thing is if something has had this something has exhibited this rich getter getting rich phenomena how does one judge the true value of an entity which is grown because of rich getting richer phenomena right. So, both sound sort of similar, but there is certain difference between them. The first one being something that is over already how would you know who is the right person by subtracting out the popularity, the second one is you should not just rank them, but you should understand that true value.

For example, in Quora upvotes if the worlds are beyond 100 should you really take it seriously right? A questions answers is beyond 100, but was, but as an answer 2, second answer to the question; one should probably subtract out a few votes here to sort of normalize and to understand the exact popularity of the answer right.

So, it this allows still open questions in the domain where people have not really understood. Firstly, we do not understand if rich get getting richer phenomena is the reason why power law emerges, first one right.

(Refer Slide Time: 18:58)

1) RGR the reason for  
emergence power law.

2) RGR adds noise  
to popularity!  
How to remove.



Is rich getting richer the reason? The reason for the emergence of power law this we have not understood it well it is just to hypothesis. Secondly, rich getting richer phenomena adds noise to popularity. How to remove this noise right? How does one go about removing it, how to remove this noise? So, that the true color, the true picture, the true popularity of a person is revealed.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Rich Get Richer Phenomenon – 2**  
**Lecture - 127**  
**Rich Get Richer – The Long Tail**

We have seen power law on networks. There is a different kind of power law that is in play in nature right and we are going to study more about it in this chapter in this particular lecture.

(Refer Slide Time: 00:22)



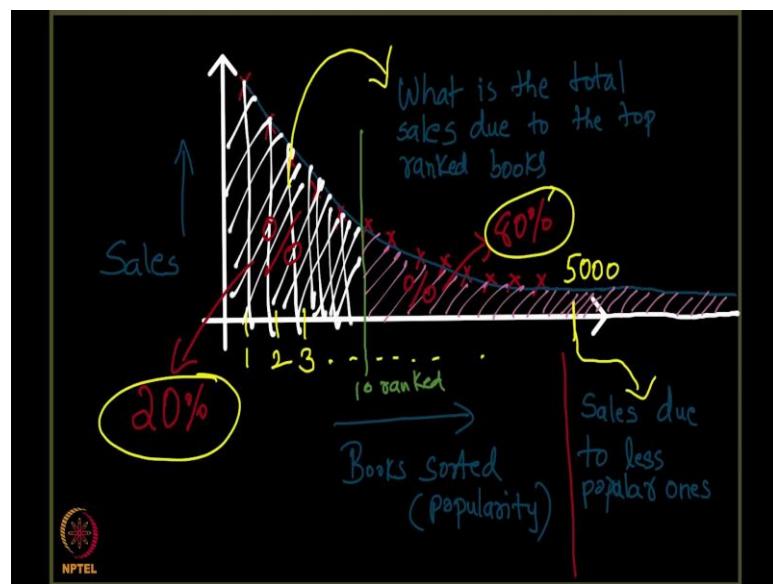
So, let me write more power law. So, this is mostly outside the realms of graphs and networks. So, let us go to a bookstore. So, assume you go to a bookstores and you browse through some books right, go to a bookstore and you see a book 1 and there are several books in the shelf as you know.

I am not good at drawing; let me make some attempts to write books. So, here is a bookstore with a lot of books alright, books. There are different types of books, different genres ok. Now, there is a huge collection generally, as you would have observed there is generally huge collection of books and a question is why would a bookstore have such huge collections right.

There firstly, there are different genres of books as a huge collection. Why would anyone have something like this right? If I will were to run a bookstores, I would simply have the best sellers right. I would simply have the best sellers. So, roughly let me assume that 10 percent of the books that are there in the bookstores are best sellers, best sellers. So, why not just keep the 10 percent of it and discard the remaining 90 percent; it only makes sense you see right.

If I were the book shop owner, I would simply keep the best sellers and discard the rest why then do bookstores any big bookstore that you visit they have a whole lot of books, why cannot they just keep the best sellers? Now, what is what is the obvious answer to this question? The obvious answer to this question is that we should probably ask this question in a different way.

(Refer Slide Time: 02:52)



I should plot this graph ok, let me just make it clear to you people slightly involved.

I will see the most popular the first most popular book and the second most popular, the third most popular book, so on and so forth. Assume, I have some 5000 such books. So, what is my x axis? My x axis is going to be popularity rank of the book. These are these are books sorted according to popularity, according to popularity fine. What is on the y axis? It is the sales of this books, the sales of this books. Obviously, the most first rank book will obviously, have very high sales, second rank book will have slightly less sales, third rank that is what you mean by ranking you say, correct.

Evidently there will be this power law like distribution correct, correct. So, on let me join these points and as you can see it will be a curve like this. Now, this looks like the same old theory of power law, but power law is slightly different as you know, it is about the distribution of the node degree correct. So, but the curve here resembles that. Now, let us get back to our question, let me go to the previous slide and let me help you remind let me see the question that I asked.

Why not have only the best sellers and discard the not so best sellers ok? I want to discard this and only retain this. Why because at my profit if my profit is mainly based on a bestseller, why then have this the remaining stuff 90 percent of the books are not best sellers. Let me just discard them. Now, what is so, wrong about this hypothesis of my bookstore business that I am going to start very soon of only the best sellers?

Well, that is where we need science rather smart accountancy or smart economics understanding of economics the situation. What we should do is think for a minute, what one should look at it is what is the amount of sales of the top few. Let say let me let me just take the first 10 which is let us say here; let us say first 5 or first 10 whatever that is. So, let us say first 10 ranked books; top 10 rank, the first 10 books I will see what the sales of this is.

A very small a very whatever I mean, it does not take a lot of time for you think. What is the answer for this. Let me just write that down. So, that it seems to our minds. My question is what is the total profit that I might possibly make profit due to the top ranked books. What do I mean by this? I just mean what exactly I mean it is thought will tell you all I am trying to say here is what is this plus, this plus, this plus, this plus, this plus, this plus this. So, basically it is the area below this curve right.

I mean whatever is the area that will denote what will that denote that will denote the total profit total sales rather not necessarily profit I think I should remove the word profit here, it is not profit it is the sales. What is the total sales due to the top ranked books? It is going to be this curve correct and definitely this is going to tell me what is the sales due to the top ranked books and the remaining stuff which is this will tell me what is my sales, what is my sales so on; this will go on as you know right.

This will extend, this does not stop here it just extends to all 5000 books. What does this denote? This denotes this denotes what? Sales, sales due to a line here; sales due to due

to less popular books, less popular books. So, what I am trying to say here? All I am trying to say here is I am looking at my sales due to my popular products versus sales because of my niche product. If these sales is very high; now please note these are the sales because of best sellers. Just because they are best sellers and just because they sell in big numbers does not mean that they make most of my complete sales.

So, a good question for you all to ask which I am sure this is running in your mind already is what percentage of my sales is this and what percentage is this. To get your clear understanding I think you people should rather not see the curve this way which is slightly misleading you must know that the curve here the blue line extends like this, so does the white line here, the y x axis and then the area below the curve also extends like this.

So, good question to for one to ask here is what percentage is this ok, what percentage is this and what percentage is this right. Just in case, this was only 20 percent; you know what I am going to where I am getting at right now. This will be 80 percent. Now, you know very well that I cannot have my bookstores with just the popular products, I need the niche products. So, niche means, the less popular ones.

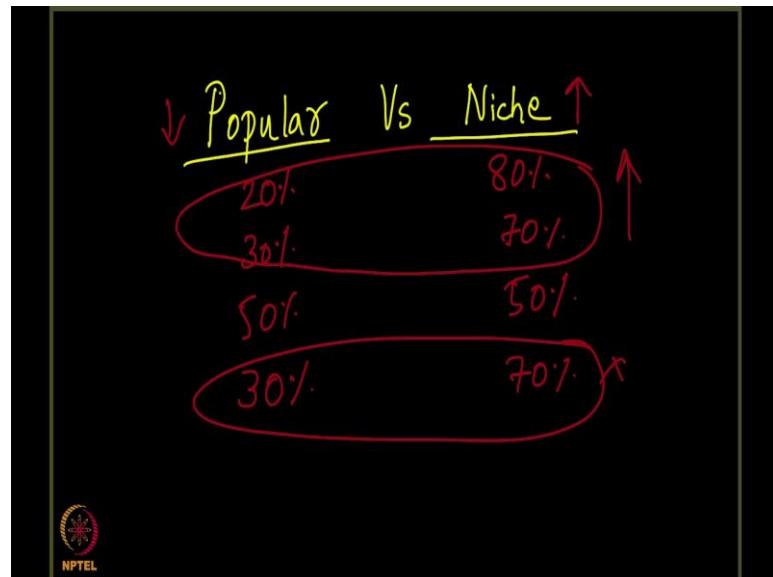
So, what are all here? As you can see let us say the past best sellers might be here, the best seller will be a best seller for a few months and then it will come here right or may be less popular books. Science books unfortunately are less popular because not many people read it. So, more scientific books are all here or technical books all here, novels and all the masala stuff is here, although you see that this may not be a big chunk of your sales, it can just be 20 percent and this can just be 80 percent.

So, the point here is you may want to concentrate on this more than this although your sales is mainly because of the bestselling book. So, you think if you write a curve like this and then see you will realize that it may not be. So, one thing that you almost observed is this. Do you think a book here continues to stay here? It will probably come somewhere here very soon right. Every best seller has its shelf life after while it will stop being a best seller and it will become a niche product; it will become a classic right.

So, it is very interesting to see how this curve continuous to stay like this where more popular stuff keeps coming here and less popular stuff with time goes to this part of the

curve. And what is more interesting is what exactly is the percentage? 20 percent, 80 percent is what I am saying it could be different.

(Refer Slide Time: 11:36)

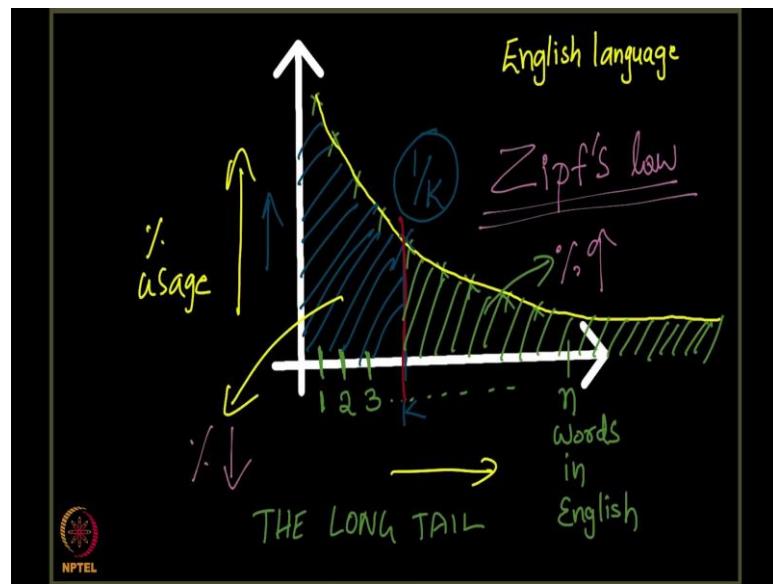


So, in your business you may have to see what is the popular products, popular versus the niche ones. You should have a good understanding of this ratio. It could be 80 percent; I am sorry just a minute whatever I said 20 percent and 80 percent or this could be 30 percent and 70 percent or it could be 50 percent and 50 percent it is always observed that something like this never happens, something like this never happens.

It is slightly on the higher side; the niche is on the higher side, popular is slightly on the lower side right. So, this speaks in volumes on why when you go to a bookstore you have all sorts of products. You cannot just keep the best selling once right. This is generally true no matter what business you are thinking of.

Let us say the kind of video that people watch on YouTube, YouTube cannot afford to keep only the most watched video saying that people who watch this video less rather I mean videos which are being watched less should just be removed right. So, the big chunk of what people watch is probably indeed the niche ones is probably the niche ones and less so of the popular once ok.

(Refer Slide Time: 13:22)



So, this has been there historically, in sciences people have observed this property in many many works of science.

One such thing is a very interesting fact which is let me try plotting and tell you what is this. If you look at the word frequency, let us not get lost. I just started off with this bookstore example and then I showed you this phenomena that happens and then I told you that popular sales, sales of popular products or sort of less in ratio as compared to the niche products and I also observed we also observed that a product that is there as a popular product might become a niche product.

It is interesting to see how this curve is preserved throughout in sales right ok. So, this is not this very similar to what happens in general with nodes as well. A node that is popular may not continue to be popular with time. It might become less popular and it might go this side right. What kind of questions do you have in mind? In fact, one can ask several research questions on this topic. So, I just right to me over email in case you want some cool questions on this topic. There are several nice very nice questions that one can attempt that the world does not know rather ok.

So, as trying to plot something very interesting about English language; about your English language, ok. If you take up the most popular word in English which is the most frequently used word in English call it the first ranked word and the second most

frequently used word in English call it the second ranked word and the third most frequently used English used word in English call it third rank and so go on.

And write down all the words let us say there are  $n$  words in English, you write  $n$  units on x axis English and then you see what is the number of times what is the proportion of times that you use this word. The first ranked word how many times, what percentage is it is usage. For example, in a document, if you think the word t h e the is most frequently used I will ask what percentage of the document is it frequently used. So, the plot looks again similar to this is and then if you join this points you will get a power law like that is not a good way to write a curve ok.

And you will again get the curve like this and this curve is known to be known to follow this function  $1/k$ . So, if this is  $K$ , this will be  $1/k$  and of course, I have used scale y axis. So, this might appear to be like a this distance seems the same as this distance, but do not mind my drawing the fact is this turns out to be  $1/k$ , if you look at the distribution of words in English language based on their ranking sorted based on their ranking and this is popularly called the Zipf's law. You can take a look at it on the internet.

It is a very popular law found several decades ago and this again says that the amount of frequently used words in English, same old theory back again if you see. That are there are there is this part of the curve which talks about the frequently used words in English, these are the words that are very frequently used rather popular words and this side is the sort of less popular words, but if you see this tail goes on like this ok; this is a big percentage this a big percentage.

I mean the area here is way bigger compared to the area here; this is a smaller percentage. Again, this is observed in words to know. So, coming for coming getting back to where we started from, it is not just true in books bookstores like business. It is also true in English language know that is surprising. It is observed in many places right and this is called the fat tail phenomena rather the long tail not the fat it is called the long tail. What is the long tail mean here? I am sure you all of guessed what I am trying to say here.

By this I mean this part of the curve this part of the curve is it goes on like this. It is start from here; it goes on like this given that this tail is longer do not have this part of the

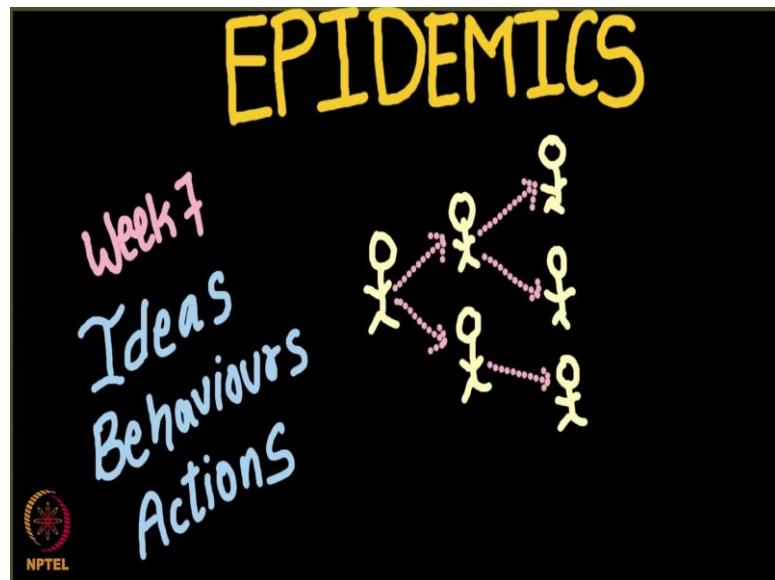
curve looks resembles the tail of an animal that is probably the reason why they call it the long tail.

Since, it is long, it acts to a whole lot of the area as compared to the popular part of the curve right. So, that is why this is called the long tail phenomena. What is long tail phenomena? Long tail phenomena is nothing but the fact that; so, the popular products are less in number although they are popular and less popular products are more in number although they are less popular the sort of dominate the space.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

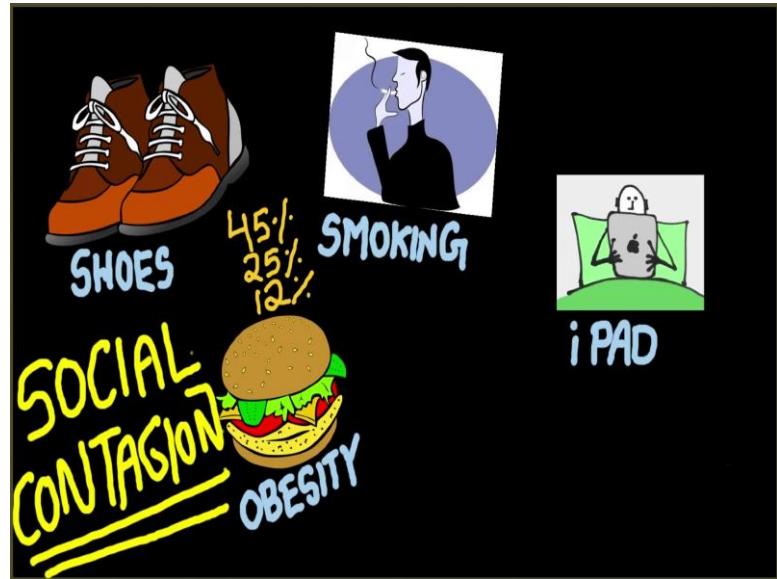
**Rich Get Richer Phenomenon – 2**  
**Lecture - 128**  
**Epidemics- An Introduction**

(Refer Slide Time: 00:08)



This week we are going to start off with a new chapter and the name of the chapter is Epidemics. So, before starting off with this chapter, I would like to recall what we studied in week 7 the cascading behavior in networks. So, we have looked at how ideas, behaviors and actions spread on a network. We have looked at people who are connected with the help of a network and then will look at how these ideas, behaviors and actions spread between these people and we have talked about a number of ideas and behaviors.

(Refer Slide Time: 00:47)



So, let me recall some of the examples. We talked about the sports shoes example where one of my friend who is in a healthcare industry and he had come up with his own brand new technology of sports shoes and he wanted me to advertise his sports shoes, which I advertised to some people and some people means some of my friends and they advertised it to some of their friends and so on, there was this cascade of sports shoes on these networks.

And then we have looked that smoking as a behavior is contagious. One of my friend could be smoking and when I look at this and when I look at my friend I could feel like smoking is cool and I might also start smoking which leads to a cascade of smoking on the network.

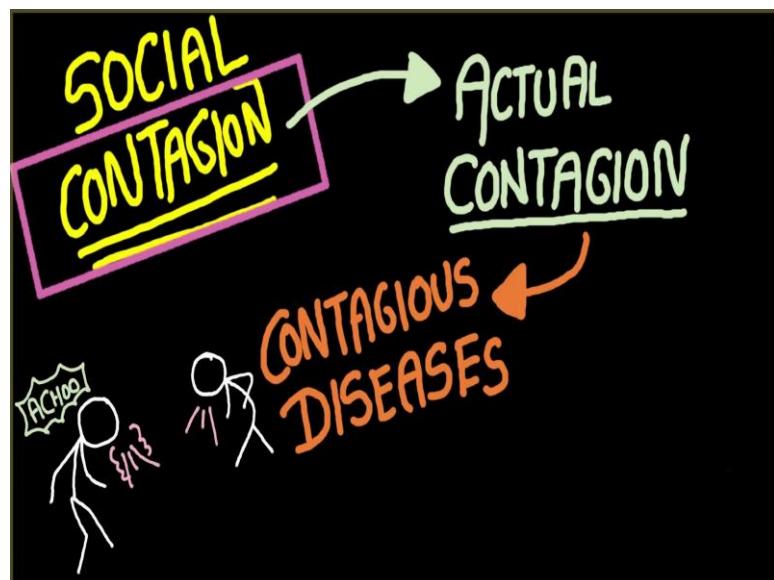
And then obesity; obesity is contagious. We have also studied this in the preliminary video. It does not appear much like obesity is contagious. Obesity seems to be quite an individualistic behavior, but actually not. There is a very interesting piece of research which says that if your friend is obese, your chances of becoming obese increases by 45 percent. If your friend's friend is obese still your chances of becoming obese increases by 25 percent; if your friend is obese still your chances of becoming obese increases by 12 percent and it's only when you reach to a friend that there is no longer of a correlation between that guy's body weight and your body weight.

All in all obesity is also a behavior which spread on a network similar to obesity depression and happiness are also the behavior which spread on a network and yet another product which can spread on a network is iPad. I basically wanted to recap that there are these ideas, behaviors and actions which can spread on a social networks and we have studied all of these in a week 7 which is cascading behavior in networks and most of these behaviors which are spreading are known as social contagion social contagion.

Why social contagion? Contagion is a word which is generally used for the spreading of a contagious disease. Let us say flu, measles, chickenpox, etcetera.

But here, contagion refers to something its spreads from person to person and what is spreading here is a social behavior, social norm; hence, the term social contagion.

(Refer Slide Time: 03:32)



So, we have talked in detail about social contagion in week 7. What we are going to do in this chapter is we are removing this word social and we are going to top just about contagion that is an actual contagion, a real contagion. And what is an actual and real contagion is nothing, but a contagious diseases; a contagious disease like flu. So, in this chapter we are going to see how we can model the spreading of contagious diseases on a networks.

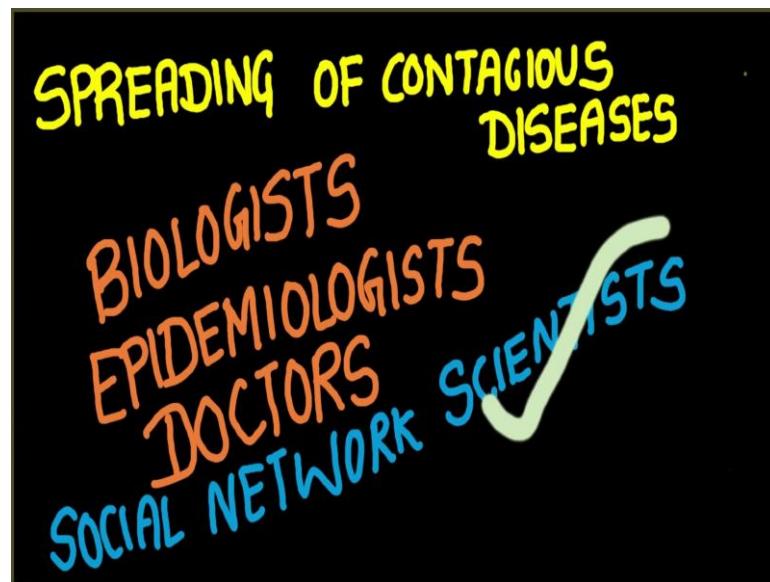
(Refer Slide Time: 04:02)



Now, one question is why is it even important, why the modeling of contagious diseases is important? And the answer have actually flashed it on the screen and the answer is it helps us in fighting epidemics. Whenever a contagion is spreading on a network and it spreads way too much; way too much means, it infects a lot of people almost everybody in the world, we say that it has become an epidemic.

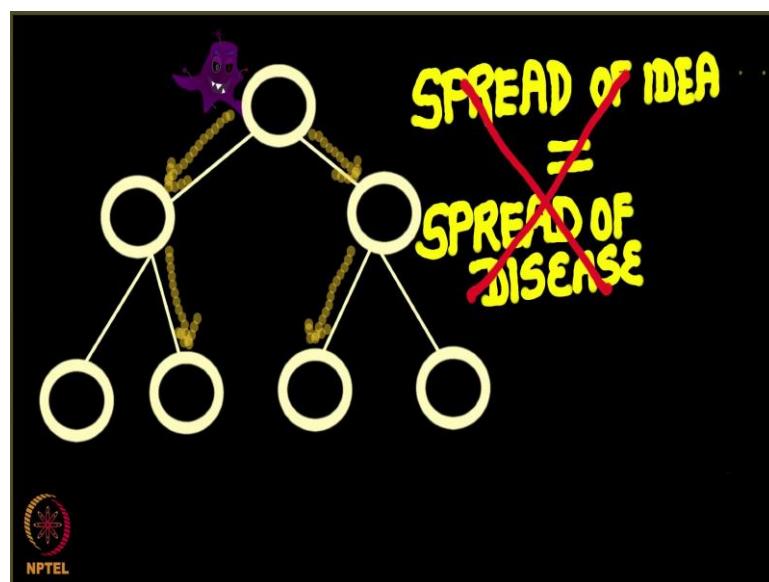
So, these contagious diseases can become epidemics overtime and such epidemics the need to be controlled, we cannot afford to have epidemics. World has suffered from dangerous epidemic from time to time and all of us know about various deadly epidemics. For example, Ebola; the most recent one, Swine flu and then Black death and there is a large number you can search on Google or Wikipedia. The world has suffered from a lot of epidemics throughout the history.

(Refer Slide Time: 05:12)



So, we are going to see in this chapter how can we model this is spreading of contagious diseases. But, now again a question should not it be biologists who should be studying how this contagious diseases spread or should not it be epidemiologists or rather medical doctors. Though social network scientists have much to offer there what will social network scientist have to what they will be having to say about the spreading of contagious diseases, can be even work in this direction? And the answer is a big yes. Why the answer is a big yes is actually very simple.

(Refer Slide Time: 05:53)



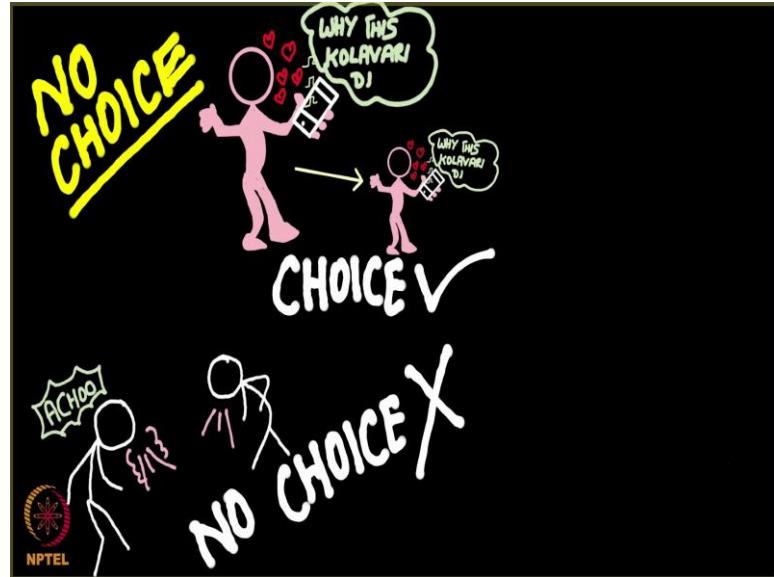
The answer is like anything this contagious diseases they spread on a network. So, as you can see, here I have a network and here is this one node and this node can infect this node which can infect this node and so on.

So, our world can be modeled like this network and our contagious disease is actually spreading on this network which is shown here. So, let us see a contagious disease starts spreading from here. So, this person gets infected the first and then it makes its way through this network. But wait now, does not do you notice something; is not it exactly like this spread of an idea? An idea also spreads the similar way.

There is a node from where a new idea starts and then it looks at its neighbors and some of its neighbors adopts this idea and similarly, this idea travels or rather diffuses through this network. So, it seems like spreading of a disease and spreading of an idea they are the same things; looks like spreading of an idea is same as the spread of a disease.

But the answer is no, they are two different processes. Why two different processes?

(Refer Slide Time: 07:12)

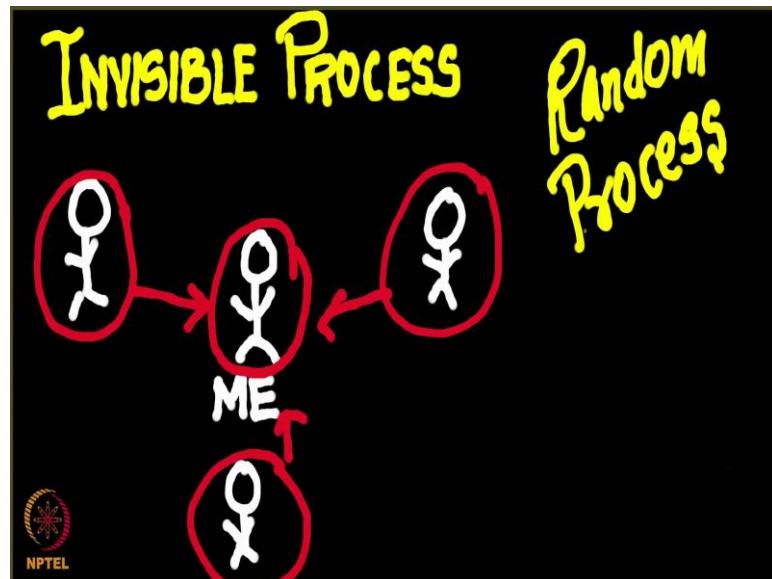


The first reason is no choice. When we talk about an idea which is spreading; let us say there is a song and my friend hear this song why this Kolavari Di and he likes it very much. And next day he tells me about it, I also listen to the song like it very much, I adopt the song do you see there is a choice hear my friend told me about this song I like this song and I adopted this song.

So, well there is a choice when we talk about the spreading of an idea. But, let us I am talking about the spreading of a disease. This time instead of a song my friend has come to me with a flu. So, he has this flu and he is sneezes on my face and now flu virus come and its entering my body. Can I first of all I notes know this virus is entering in my body? But still, let us say I know this. So, can I say that I do not like this flu virus, I am not going to take it?

While in the case of a song, you have the choice to say this is a bad song, I do not like it, but you are you cannot, you cannot do the do this in the case of a disease. So, there is no choice. If the disease has to infective, it will infective there is no way out. So, you have less of a control over the spreading of a disease as compared to the spreading of an idea.

(Refer Slide Time: 08:40)



This second difference is the invisibility of the process. What is invisibility of the process? I will tell you a small anecdote; this was when I was in my fourth standard. So, here is me and I am in my fourth standard and there are three people in my family; my mom, my dad and my brother and all of these three people they suffered from eye flu. So, eye flu was spreading that time and all of these three family members of mine they suffered from this eye flu.

And now, I can catch flu from any of these three right and the very next day I also catch eye flu. So, I have got this eye flu here. But I do not know which is the person out of these three who has infected me with it eye flu; till today, I do not know. So, the

spreading of a disease is quite an invisible process as compared to the spreading of an idea where you might have a very clear clue of which is the person from whom you have adopted the idea from, but it is very difficult to say which is the person from, whom you have adopted a disease.

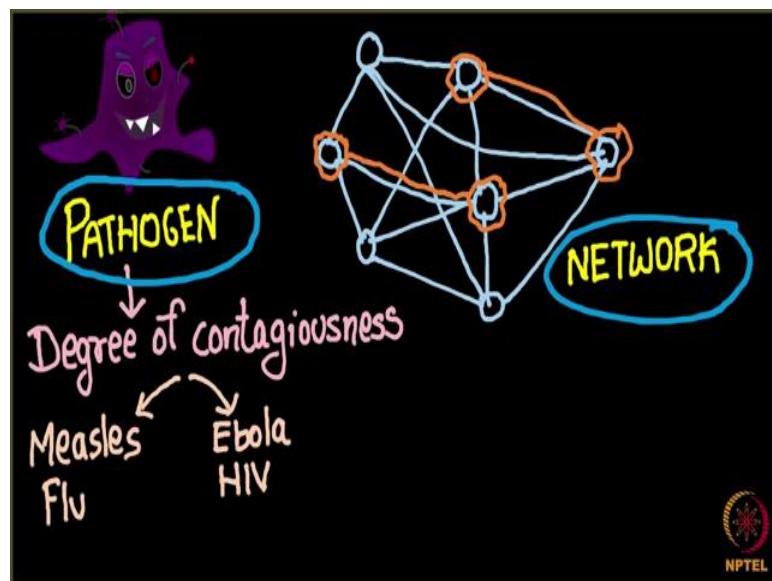
Hence, we need different model for the spreading of an idea, which we have looked at before and now we need a different model when we want to talk about the spreading of disease because the spreading of a disease is more of a random process as compared to the spreading of an idea and will explore it further.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Rich Get Richer Phenomenon – 2**  
**Lecture - 129**  
**Introduction to epidemics (continued...)**

So, now, before starting with the question or before looking into how an epidemic spread lets us look at 2 things, 2 important things. So, I would like you to ask this question from yourself which 2 factors do you think are the most important which 2 factors are required when we want to model a disease spreading on a network? Assume there is this classroom and there are some people, some students in this classroom and one of the student catches a flu there and this flu is spreading through the network. So, which 2 things will you require if you want to model the spreading of this flu? And here goes the answer.

(Refer Slide Time: 00:50)



So, the first one is the pathogen itself about the flu itself. So, what is spreading on this network is important. For example, it is degree of contagiousness matter, how contagious this flu is. Taking an analogy with these spreading of an idea as we have seen, something like a piece of code spreads less quickly as compared to a juicy piece of gossip.

Similarly, in the case of diseases, there are certain diseases which is spread more quickly as compared to other diseases. If you talk about diseases like measles and flu, they spread quite quickly as compared to the diseases like Ebola and HIV.

So, first of all what is important is the pathogen. We need to know how contagious this pathogen is and the second which is the obvious for a network scientist of course, is the network, yes we need the network on which this pathogen is spreading. Does the network really matter? Yes it matters. So, if you look at the network which is shown to in this figure, you can see that there are quite a blessed number of edges in this network.

The network is less dense rather we call that this network is (Refer Time: 02:06) So, you put any disease over there it will slowly move to this network; rather it can die away quickly, but if your network is say something like this and I put a lot of edges there between these people, so you put any disease on this network and this disease will quickly spread on this network because, your network is dense.

So, yes one thing is the density of the network or let us say the sparsity of the network it affects how your contagion is going to spread on the network rather, there is one thing very interesting to note here is do you see that this pathogen has something to do with the network? So, let us say I want to model these spreading of flu and I can be having a network like this and the network is going to be quite dense and actually network is dense for the spreading of something like a flu because, even if you come in because flu spreads even if you coming close proximity with someone, so even if you just stand and talk to a person for 5 minutes you can catch flu right.

And even people do not need to come in close contact together, let us say that you have worked on a piece of code on your keyboard and then you go away and then your sister comes and she also works on the same keyboard, even this thing can make a flu to spread. So, this common flu, common cold flu, common cold virus, it can spread even through keyboard.

So, the network is going to be very dense in the case of such a contagion, if the contagion is a flu, but let us say I do not want to model the spreading of a flu, I want to model spreading of let us say HIV. So, in the modeling of a HIV will the same contact network work? And the answer is no, in the case of HIV since it is spreads with the help

of sexual contact, there will be quiet less number of edges which will be which will be counted in this contact network.

So, the network in the case of HIV is going to be very sparse. So, whether your network is dense or your network is spars also depends upon what kind of a pathogen we are talking about. If this is the pathogen like flu, then the network is going to be quite dense and if the pathogen is something like let us say HIV then this network is going to be very sparse.

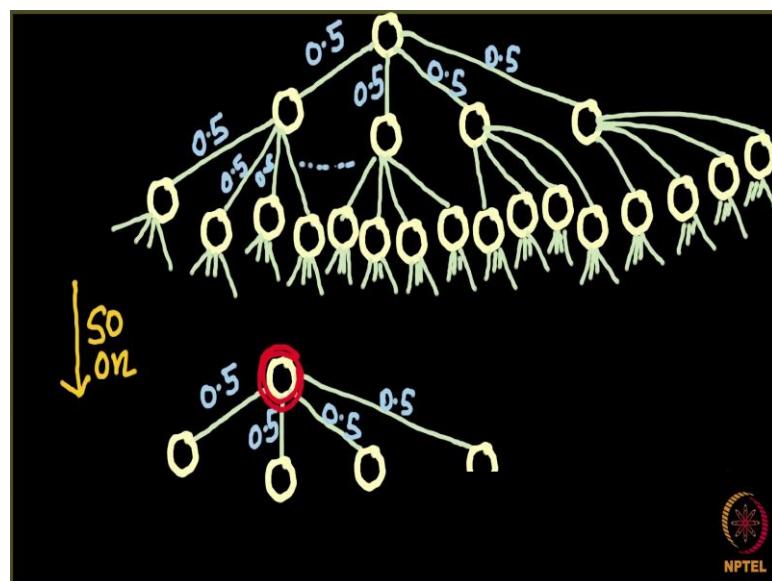
So, here while your blue I just repeat your contact network for the spreading of flu orange edges or the orange nodes here, this is the contact network for the spreading of HIV which is quite sparse. So, these are the two things which are most required for modeling the spreading of a disease; the first one is the pathogen how contagious it is and second one is obviously, the network structure, so will be using just both of these 2 things to model the spread of diseases on our social networks.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Rich Get Richer Phenomenon – 2**  
**Lecture - 130**  
**Simple Branching Process for Modelling Epidemics**

So, now we have covered all the prerequisites, so the prerequisites were start. So till now we have covered all the prerequisites which we were required for understanding the modelling of epidemics and we have looked at the two most important factors there. The first one is the pathogen and second one is the network. So we are now all set to go ahead and model this spreading of a disease. So let us say how we can go about it.

(Refer Slide Time: 00:35)



So, let us say here is a node, here is a person and now this person over here it has let us say four friends here as shown in this figure; so there is this person and this person has four friends here.

And then these four friends each of these has each of these have again four friends each. So you see we are taking a very simple example for modelling; so we will start from a very simple example and we will move towards a complex network. So we are taking the most simple network that is which is in the form of a tree; obviously, we know that social networks are not in the form of tree; because there will be triads on this network

and these nodes will be connected to each other. But only for the sake of simplicity for the time being we are taking the network to be a tree network, so one node having four children each of these four children again having four children.

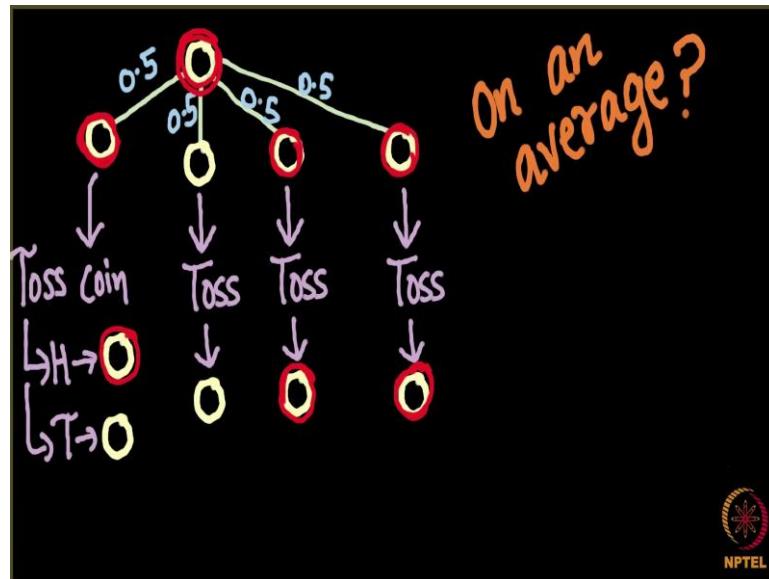
And then which of these again having four children each. So this is the kind of network on which we will be modelling a disease. So this is the network here now let us say this if you see this edge over there so I write a number on this 0.5. So as we have discussed previously there were two factors one was the network and another was the pathogen so the network we have seen here is in the form of a tree over here and second thing is the contagiousness of the pathogen. So we model the contagiousness of the pathogen in terms of probability.

For example, I have given this edge a probability of 0.5; what does it mean? It means that if this node here is infected then there is a 50 percent chance that this node will be able to infect this node. So with the 50 percent probability 1 by 2 probability the infection will transmit across this edge and if this node is infected it is this particular child will be infected. So I will say that the probability of infection across this edge is 0.5 and again for the sake of simplicity, I say that the probability of infection across all the edges is 0.5.

So, again if this node is infected it infects this node with the probability of 0.5; similarly, across this edge similarly across this edge and similarly across all of these edges. So every edge has a probability 0.5 of infection; what can we say from here? So let me just take the first level of this tree so I redraw this network here I have this node which is the same node here. So I take this node here and then this node has four children and it can infect each of these it infects each of this these children with the probability of 0.5.

Now if this node is infected so I show it with the red circle this node is infected. How many children do you think it will infect here? So each of its child it infects with the probability of 0.5 and when it infects these how many children do you think will become infected.

(Refer Slide Time: 03:53)



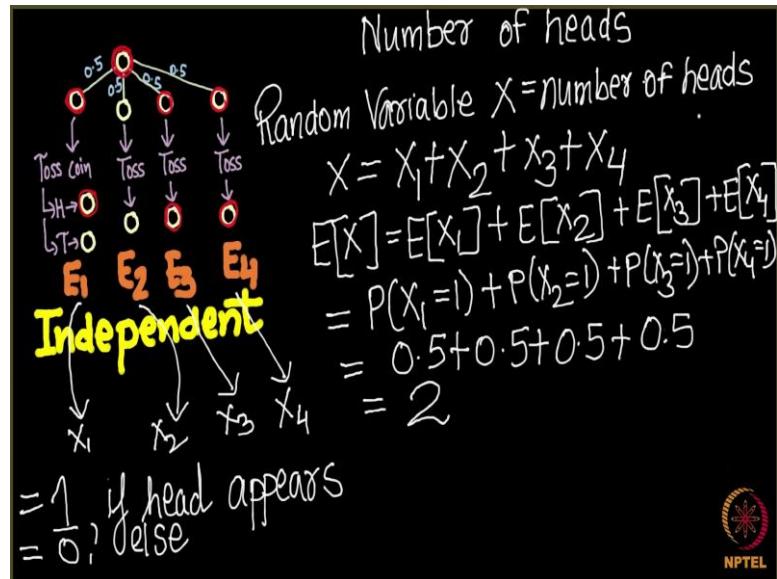
So, I redraw this network here just for the sake of simplicity and my question is how many of its children will it infect ok. So if we see here this node has a 0.5 probability of infecting this node what does that mean.

That means that if I toss a fair coin here and unbiased coin here I toss a coin I get head with the probability of 0.5 right. So if head turns up I will mark its the particular child to be infecting and if there is a tail I will say that this child is not infected. So how do I model this? If the probability of infection is 0.5, I will take a fair die toss it I get a head it means this node is infected else node.

And I do this for all of its children, so I do it for second child and let us say I get a tail and if I get a tail this node remains uninfected. And then I do it for the third child and let us say I get a head and I infect this and I do it for the fourth child let us say I again get a head.

So, I infect this fourth child also so three children are infected, so this happens for this particular case what do you so this three guys got infected over here. So what do you think happens on an average, on an expectation so what is the expected number of its children which will get infected here.

(Refer Slide Time: 05:26)



So, let us quickly see that; so I again redraw it so here I have four cases; in this case I am tossing the coin here, I am tossing the coin here, here, here. So four of these are different experiments, which are independent of each other so I have four experiments here.

I toss one coin for this, one coin for this, one coin for this and one coin for this and all these coin tosses are independent of each other. So I say that these all experiments are independent one has nothing to do with another; all of these are independent. And now our aim is to find the number of heads expected number of heads rather so if I the number of heads I get is equal to the number of nodes infected; because if I get a head I infect that particular node so our aim is to find the expected number of heads.

So whenever we have to find the expected number of a variable we modulating the form of a random variable. So I take a random variable  $X$  here and the value of this random variable  $X$  is equal to the number of heads I am going to get. And how do I find the value  $X$  over here it is with the help of these four things so what I do is, I take four more random variables. One random variable for each experiment so for experiment 1 I have a random variable  $X_1$  rather we call it indicator random variable. So indicator random variable is a random variable, which takes the value only 1 or 0.

And then for  $E_2$  I have another indicator random variable, for  $E_3$  I have 1 for  $E_4$  I have 1. Indicator random variable is something very simple what is happening in experiment 1 and either getting a head or I am not getting a head. So I say that the value of  $X_1$  is 1 if I

get a head and 0 if I do not get a head. So this is what do I mean by an indicator random variable it is 1 for success and 0 for failure. So I have  $X_1$  which is 1 if head appears, 0 if tail appears similarly  $X_2$  is 1 if head appears 0 if tail appears and same for  $X_3$  and  $X_4$ .

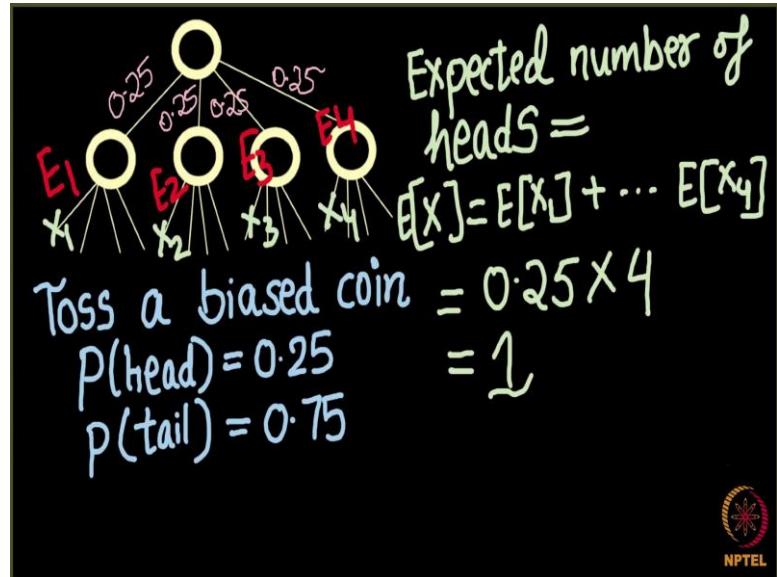
So, now can you write this  $X$  in terms of  $X_1$ ,  $X_2$ ,  $X_3$  and  $X_4$  so what is my random variable  $X$ ; it is the number of heads. So I know that  $X_1$  is what the number of heads I get from experiment 1 right. If it is a success, I get 1 head if it is a failure I get 0. Similarly,  $X_2$  is a number of heads I get from experiment 2 which again can be either 1 or 0 similarly I have  $X_3$  and then I have  $X_4$ . So what do I can say is  $X$  equals to  $X_1$  plus  $X_2$  plus  $X_3$  plus  $X_4$ . So wherever I get a head out of these experiments I can add that add that value and it becomes equal to  $X$  ok.

What is our aim? Our aim is to find the expected value of  $X$  so I will take an expectation on both the sides; left hand side and right hand side and I get expected value of  $X$  is expected value of whatever is there in the right hand side and I can actually distribute this expectation. So I can write expected value of this complete term  $X_1$  plus  $X_2$  plus  $X_3$  plus  $X_4$  as something like this. So this is called linearity of expectations I will not go in much detail so I get expected value of  $X$  is this and then again here is a you can take it as a black box. So just understand this if I am talking about the expected value of an indicator random variable.

So, here  $X_1$ ,  $X_2$ ,  $X_3$ ,  $X_4$  are indicator random variable either 0 or 1 so if I talk about the expected value of an indicator random variable it is nothing, but the probability that this variable is 1. So, I can write it like this expected value of  $X$  is probability that  $X_1$  equals to 1 plus probability that  $X_2$  equals to 1 plus probability that  $X_3$  equals to 1 and so on. And then we know what is the probability that  $X_1$  is 1; when is  $X_1$  1?  $X_1$  is 1 when I get a head in the first coin toss and the probability of that happening is nothing, but 0.5 and same is for  $X_2$ ,  $X_3$  and  $X_4$ .

So, overall I get the value 0.5 into 4 which is 2. So I have done it rigorously you even if you did not understand it is perfectly it comes to (Refer Time: 10:31) also that. If let us say I toss a coin four time so on an average how many heads will you get and intuitively your mind tells you that the answer is 2; 0.5 into 4 so even if you did not understand this calculation of expectations it is perfectly fine ok.

(Refer Slide Time: 10:51)



Now, let us take another example; again I have a node here and then this node has four neighbours and then each of them has four neighbours and then let us say the probability of infection across every edge this time is 0.25 instead of 5. So in the previous case we have tossed a coin so the probability was 0.5 we tossed a coin we looked at whether it is a head or it is a tail and accordingly we did what we have to do. Now what I do in this case can I toss a coin in this case; I do not know so coin gives me head with probability 0.5 tail with probability 0.5; what do I do in this case?

So what we can do in this case is we toss a biased coin and my biased coin is such that it gives me head with the probability of 0.25 and gives me tail with the probability of 0.75. So biased coin is something like it is not fair coin, fair coin gives you head and tail with equal probability. So a biased coin can be something like if you remember the movie the movie Sholay; so there was a coin which had head on both the sides so that was a biased coin in that case probability of head was 1 probability of tail was 0.

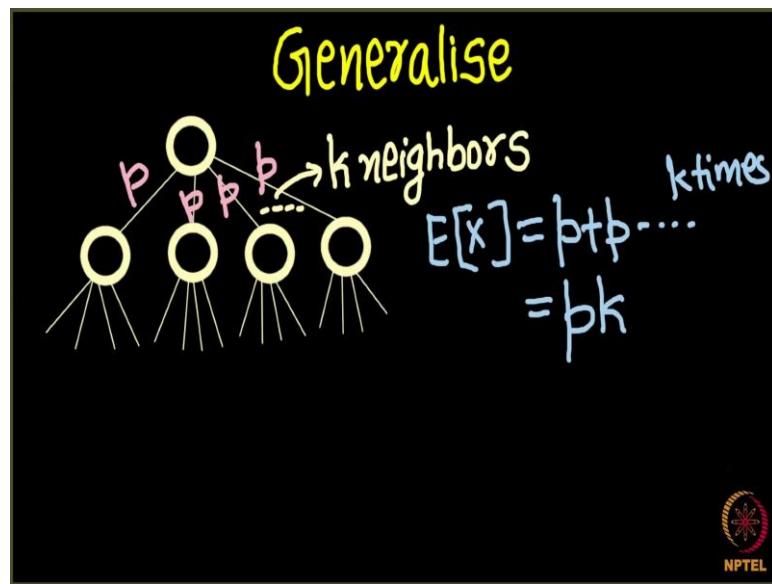
No matter whatever you do you are going to get a head every time; that was a biased coin. Here I am using another form of biased coin which does not give you head always. It gives you head 25 percent of the times and gives you tail 75 percent of the times. And again what I am going to do is if I then again I will have these four experiments here for each experiment, I am going to toss a coin and then if I get an head I infect the particular

node, if I get a tail I do not infect the particular node. And here what do you think is going to be the expected number of heads.

So, it is again very simple its almost the same what we have done previously so again we take four indicator random variables here  $X_1 X_2 X_3$  and  $X_4$  and we know that expected value of  $X$  is expected value of  $X_1 + X_2 + X_3 + X_4$  which can be written like this. And we have looked at what is the expected value of a indicator random variable is nothing, but the probability of 1 which is probability of head which is 0.25 in this case. So I am going quiet I am going quiet quickly because it is similar to what we have done previously you can pause the video here and see what is actually happening.

So, we get here 0.25 into four which is nothing, but 1 so the expected number of heads here is 1 which means that if this guy here infects each of these nodes with the probability 0.5 on an expectation on an average it will infect one of its children ok.

(Refer Slide Time: 13:53)



Now, what we are going to do is we are going to generalize what we have done till now. We have looked at two examples and now we are going to generalise whatever we have done.

And it is very easy how do you generalise it I take a network here and now I say instead of having let us say 4 children so instead of every node having four children, I say that every node is having  $k$  children or  $k$  naught neighbours. Every node is having  $k$  children

so the node here it will have four children and then the node here it will have its own four children node here four children and so on I am very sorry not four it will be k.

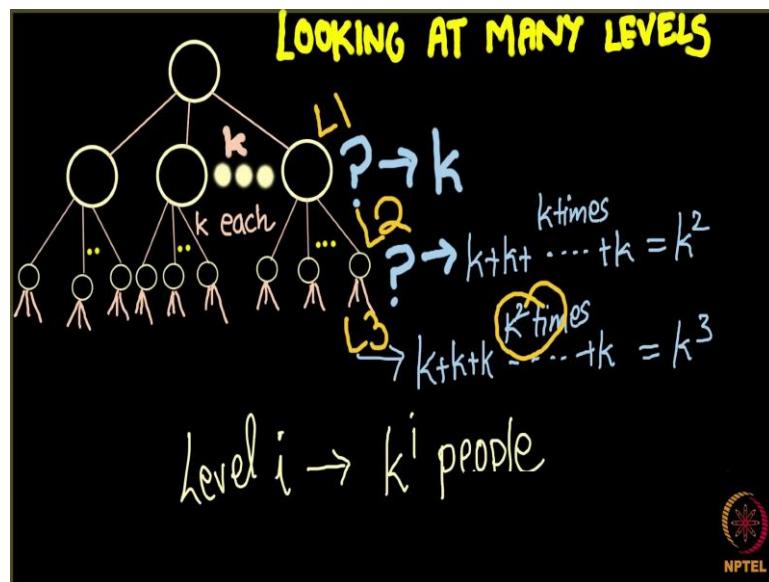
So, this very sorry this node here will be having k children this node here will be having k children this node here will be having k children and so on. And let us say the probability of infection is p so we have taken 0.5 in the first example 0.25 in the second example here we generalise it and take this probability to be p. Now what do you think will be the expected number of infected people, so it will be simply be so you can write each of these.

So how many experiments are we doing here? We are doing k experiments here because there are k children and what is the probability of success in every experiment is p. So the expected value of X that is expected value of you know infected people here is going to turn out to be  $p + p + p k$  times which is going to be equal to  $p * k$ .

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Rich Get Richer Phenomenon – 2**  
**Lecture - 131**  
**Simple Branching Process for Modeling Epidemics (Continued)**

(Refer Slide Time: 00:09)



Now, what we are going to do is, we are going to look at more levels, many levels. How do we, what do I mean by looking at many levels? So till now we have looked at if there was one person who is infected and this person has some children how the disease is going to spread in this case according to what is the number of children this person has and how is the pathogen, what is the contagiousness of the pathogen we have looked at that in terms of probability.

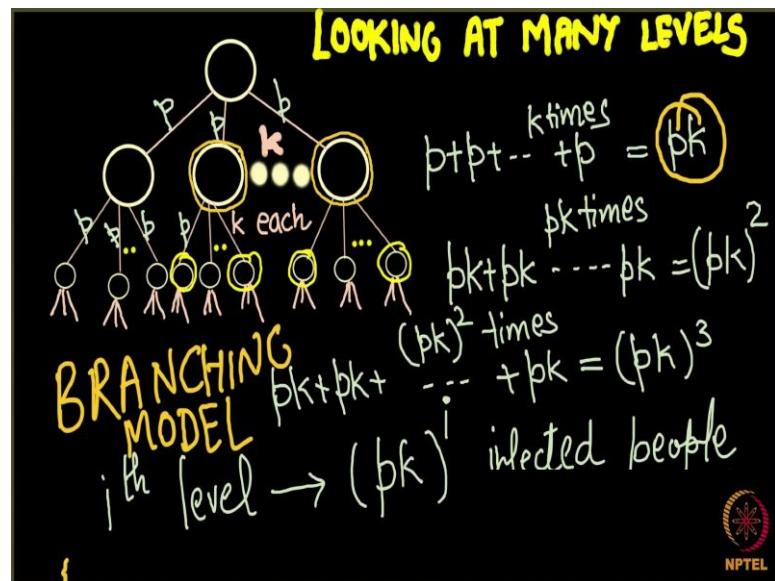
Now, what I am going to do is, we are going to look at this network in terms of many level. So this is the first node here and let us say this node here has got  $k$  children  $k$  children and each of these  $k$  children is again having some  $k$  children each of these children has these  $k$  children and so on. Each of these  $k$  children are again having some  $k$  children; what do you think? So let us call this particular level to be level 1 so this is your level 1.

So, what do you think are the number of people present at level 1 and we know it very clearly this is  $k$  because our node here it is having  $k$  people? So the number of people at this level are going to be  $k$  people now what do you think is the number of people at the second level. So this is our level 2 so what is the number of people here at level 2 so a little bit of maths tells us what is the number of people. So in the previous level we were having  $k$  people and each of these  $k$  people have  $k$  children.

So, how many people here we have;  $k + k + k$   $k$  times which is equal to  $k^2$  similarly let us look at level 3. So how many people will be here at level 3 so at level 2 we were having here at level 2 we were having  $k^2$  people and each of these  $k^2$  people now are having  $k$  children. Hence at level 3 we have how many people  $k + k + k$   $k^2$  times because each of these  $k^2$  people are having  $k$  children each.

So, we get an answer  $k^3$  so on and so forth so if I ask you the question what is the number of people at level  $i$ ; so what it will be at level 1 we had  $k$  people  $k^1$  at level 2 we had  $k^2$  people; at level 3 we had  $k^3$ . Similarly, at level  $i$  we are going to have  $k^i$  people. So, we have looked that how the different levels how do the number of people increase; what we want to do next is.

(Refer Slide Time: 03:06)



Let us look at how the contingency spreads over here and again looking at many levels and here we have a node. So this node is here having  $k$  children each of these  $k$  children is again having  $k$  children and each of those is again having  $k$  children.

The probability of infection here is  $p$ ; I know that there are  $k$  people at this level, but what are we interested in now is out of these  $k$  people on an average how many people are infected here. And we know the answer right so if there is a person here and he has  $k$  neighbors each of the edge each of the edge has a probability  $p$  of infecting. So we have looked at it previously that there are going to be  $p + p + p k$  times, that is on an average  $p * k$  people are going to be infected at this level.

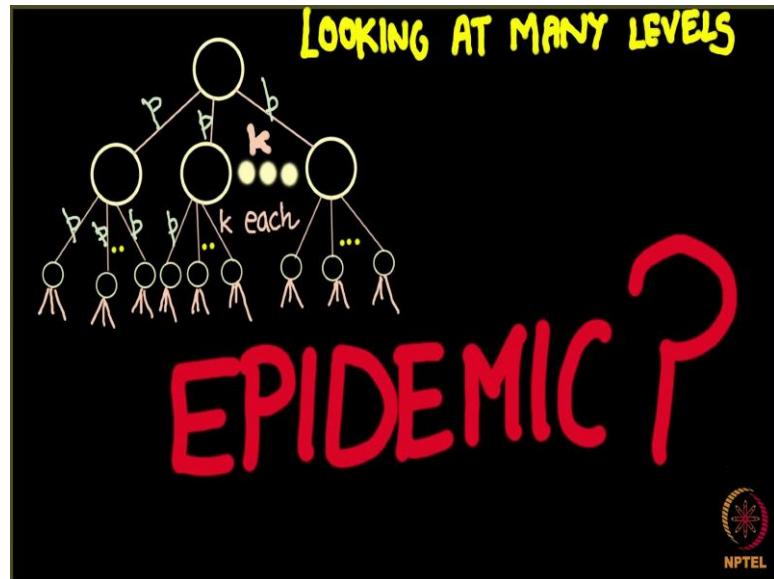
What about the next level how many people are now going to be infected the next level; can you pause this video for 2 minutes and then try to figure it out by your own and then you can resume the video. Now here I am having  $p * k$  number of infected people let us say these are those infected people; so, one is here, one is here, so here are  $p * k$  number of infected people. What happens next? This person here how many people is it going to infect in the next level again the question is the same.

One node is infected here. And this node is having  $k$  neighbors each of the neighbors gets infected with probability  $p$  as before on an average it creates  $p * k$  number of infected people. Similarly, this node here creates  $p * k$  number of infected people and how many such nodes are there  $p * k$  such nodes are there. So how many people get infected at this level is  $p * k + p * k + p * k$  times which is  $p * k^2$ . What happens at the third level is again similar.

So, let us say these are those people are those  $p k^2$  number of people who are infected. So now, on the next level how many people will be infected? So each of these guys over here will infect on an average  $p * k$  number of people so in the next iteration  $p k + p k + p k$  and there are  $p^2$  people so we add it  $p * k^2$  times and we get  $p * k^3$ . So in the third level  $p * k^3$  number of people gets infected.

Similarly, what happens at the  $i$ th level is  $p * k^i$  people get infected. I think it is clear now so in this tree kind of a network at  $i$ th level we have how many people we have  $k^i$  people. And then if we talk about the number of infected people there are  $p * k^i$  number of infected people. This is all about the basic epidemic modeling and this model which we have discussed about this is known as the branching model. So, this model it is known as the branching model.

(Refer Slide Time: 06:59)



We have looked at this example; always keep this example in your mind so one person having  $k$  children each of these having  $k$  children and then we looked at the branching model here we found out the expected number of people infected at the  $i$ th iteration is  $p * k^i$ . What I said in the beginning, why are we studying these why are we studying this contagion, how does a contagion spread? And the answer was to understand whether something goes epidemic or not.

When will I say something has become epidemic on this network shown over here I go I keep going down in this network keep going down in this network and when I go lot of down let us say through some infinite level and I see a person infected here. I can say that this disease has become an epidemic. If it was not to become an epidemic it would have died somewhere here, but this disease starts from here infect these people, infect some of the people at this level we keeps we keep going down and it even infect some of these people at too much bottom or let us say at the infinite level.

Then I say that this disease has become an epidemic now from all the information I have given you that there are some probability  $p$  and number of neighbors  $k$   $p * k^i$  people infected at the  $i$ th level. Can you tell me whether this disease is going to be an epidemic or not? So based on the values of  $p$  and  $k$  can you comment about whether contagion is going to be epidemic or not. Is it even possible to say it and the answer is yes and in next lecture we see how do we do that? How just by looking at the value of  $p$  the probability

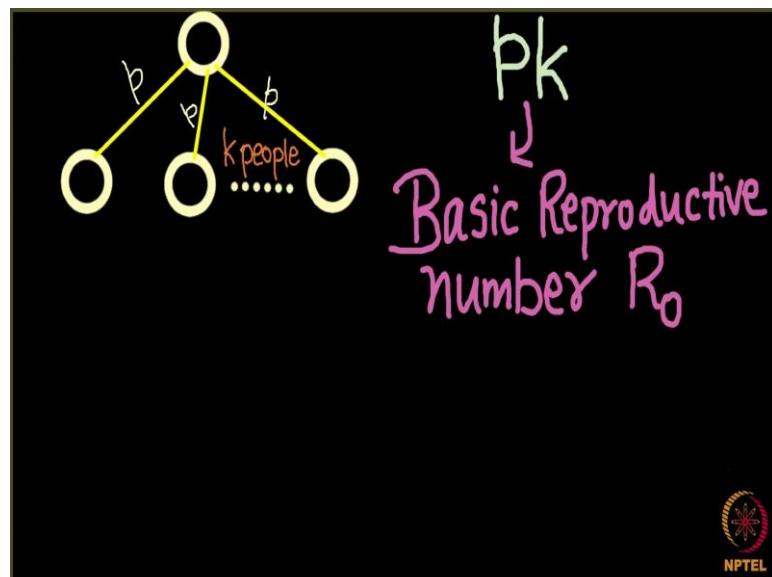
and  $k$ , the number of children every node has, we can comment about whether this contagion is going to become an epidemic or not.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Rich Get Richer Phenomenon – 2**  
**Lecture – 132**  
**Basic Reproductive Number**

In the last lecture, we said that just based on the probability  $p$  that is the contagiousness of the pathogen and number  $k$  by just looking at the number of children every node has, we can comment on whether a disease is going to become an epidemic or not. How do we do that?

(Refer Slide Time: 00:26)

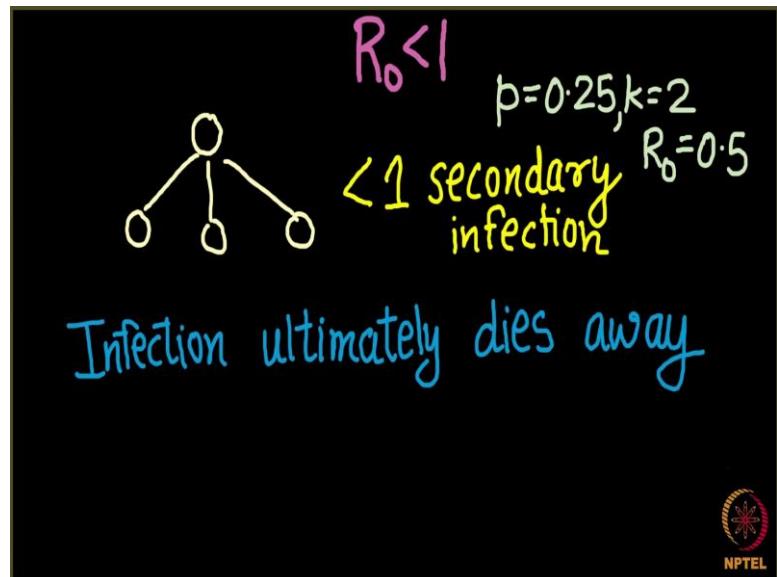


Let us look at that. Here is a node this node has some  $k$  people, or this node has some  $k$  children. Each of these children gets infected with the probability  $p$ . This is what we have studied till now. We looked at this number  $p * k$  here, what was this number  $p * k$ ? This number  $p * k$  was the expected number of people who are going to be infected here.

If you can see what is this expected number of people who are going to be infected here. You can see if this node is infected  $p * k$  tells us the number of secondary infections produced by one person; primary infection here and it creates some number of secondary infections and that number the expected value of that that number is  $p * k$ . This number  $p * k$  is important. It is called the basic reproductive number denoted by  $R_0$  and it is very

very important, because this number is going to reveal a lot of secrets. This number itself is going to tell us whether a disease is going to become an epidemic or not.

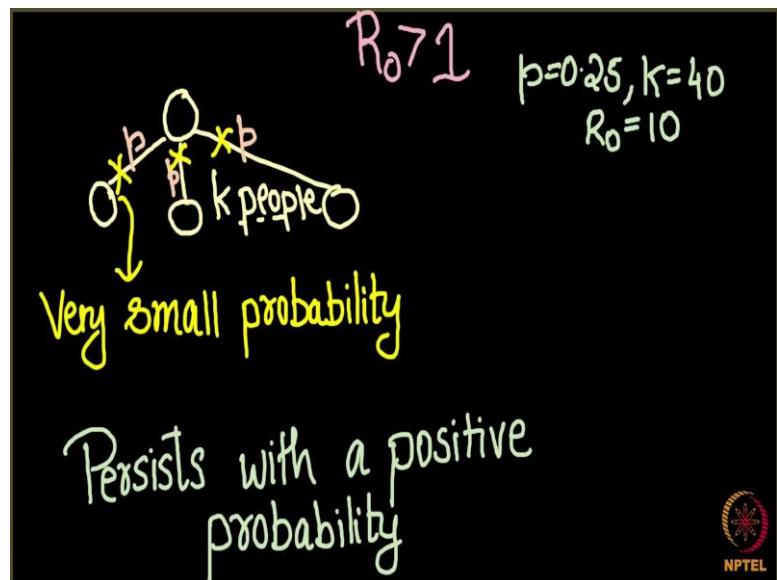
(Refer Slide Time: 01:40)



And I tell you here that if this number  $R_0 < 1$ . Let us see what will happen. So, this number  $R_0 < 1$ . Recall what is  $R_0$ ?  $R_0$  is nothing, but  $p * k$  that is the number of secondary infections. If  $R_0 < 1$ , what happens? There are less than 1 secondary infections. If you look at this node over here, it creates less than 1 secondary infection what does it mean? It is unable to infect even one person here. You see that disease dies over here itself and let us say by hook or crook, the disease somehow survives here. But then in the next iteration, it has a very high probability of dying out. First of all, it dies away here only, even if here it somehow survives is able to infect one node. In the next iteration, it will not be able to infect even one node.

If  $R_0 < 1$ , what is going to happen is let us take this example first. Let us say  $p$  is 0.25 that is the probability of infection is 0.25 and every node is having two neighbors, then a basic reproductive number  $R_0$  is going to be 0.5. That is this node here on an expectation, it is going to infect half of its child, half of its children. Half of its children says that it is unable to infect even one person here and the disease dies away in this case. What happens? So, when  $R_0 < 1$ , your infection ultimately dies away which means that this contagion is not going to become an epidemic.

(Refer Slide Time: 03:25)



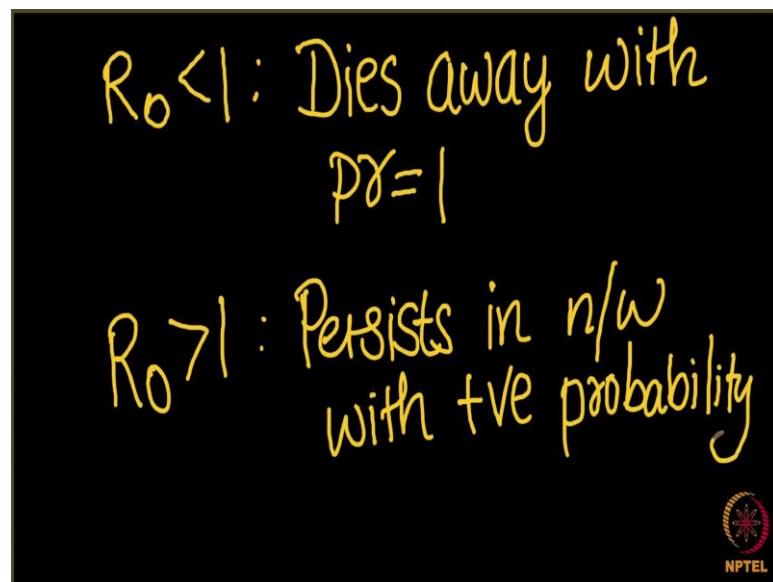
What happens in the other case,  $R_0 > 1$ ? If  $R_0 > 1$ , we again take the same network  $k$  people probability of infection  $p$  across every edge and let us see, let us say  $p$  is 0.25,  $k$  is 40. There are 40 people and each person gets infected with the probability of 0.25. The basic reproductive number is 10. So, you would see what will happen here. This node here will infect 10 people here and then those 10 people will infect almost 100 people and those 100 will infect some 1000 people.

So, you see how this disease is going to become an epidemic. So, in this case when a very sorry so, in this case when your  $R_0 > 1$ , your disease becomes an epidemic. There is a slight difference between both of which is very important. Previously here when your  $R_0 < 1$ , we said that infection ultimately dies away, and it dies away with probability 1. It is for sure that this infection will die away. I am not going to tell you now, exactly how, I will be covering that in the advance part of this lecture, but you take my words in this case infection will ultimately die away for sure that is with probability 1 infection will vanish from this network.

But in this case here what is going to happen is, your infection yes it persists in the network it is an epidemic, with the high probability it is an epidemic, but still here is a chance that it will die away. So, in this case when  $R_0 > 1$ , your infection your contagion is an epidemic, but not with the 100 percent certainty. There is still a very small probability that it can die away. How it can die away is let us say at the first level itself.

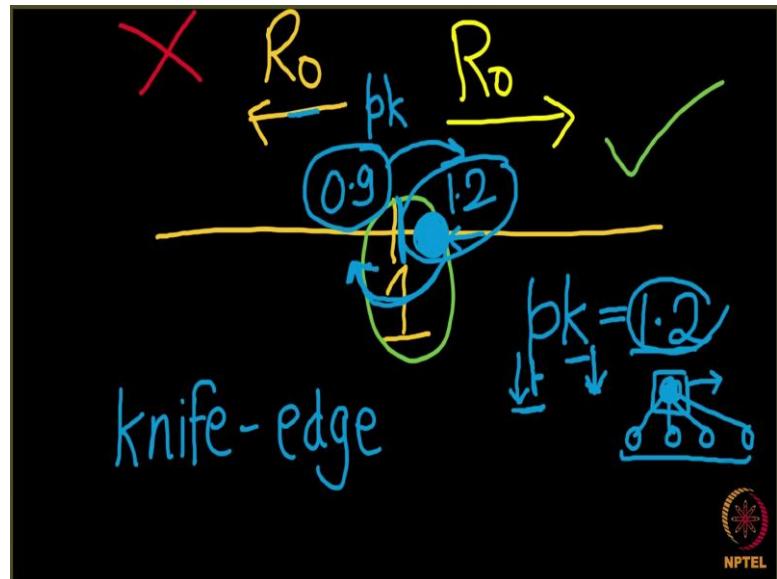
All of these links fail and the probability of happening that is very small. There are 40 links and out of this 40 links each of these links fail, very less probability. But still it is positive; there is some positive probability that all of these connections fail. And if all of these edges fail will very sorry; if all of these edges fail in transmitting a disease, it means that your infection is died away. So, when  $R_0 > 1$  with some positive probability which is high or infection persists become an epidemic, but with a very small probability it can die away also. So, persists with a positive probability.

(Refer Slide Time: 06:25)



While write this statement here for you if your  $R_0 < 1$ , then your disease your contagion dies away certainly, dies away with probability equals to 1. But if your  $R_0$  it is greater than 1, then it persists in the network. It becomes an epidemic; your infection persists in the network with positive probability. I do not say that this probability is 1, but with some positive probability your infection persists in the network. So, we have looked at the importance of the value of  $R_0$ . If  $R_0 < 1$ , the diseases for sure going to die away; if  $R_0 > 1$ , the disease becomes an epidemic. Do you see something very interesting there? The importance of this value  $R_0$ .

(Refer Slide Time: 07:26)



So, here let us see here is the value 1 and we say that if  $R_0$  it is less than 1, your infection dies away; infection no longer exists. But if your  $R_0$  it is greater than 1,  $R_0$  is on this side then your infection is going to become an epidemic. So, this value of 1 here it is becoming so very important. Why so very important? Let us say you have a disease, a contagion is spreading on a network and let us say the value of  $R_0 > 1$  right and let us say this value lies somewhere here.

So, here is your value of  $R_0$  let us say it is some 1.2 and it means that your disease is going to become an epidemic. Now you see what is happening. If somehow, I can do something here and bring this value a little bit down from 1 let us say from 1.2, I make this value to be 0.9. Do you see the disease suddenly dies away? So, this disease, which was an epidemic here, this disease was an epidemic here; I played little bit with its value of  $R_0$  and suddenly this disease will die from this network. Is not it so very interesting and so very important also? We have looked at  $R_0$ , what was  $R_0$ ? It was  $p * k$  so,  $p * k$  is 1.2 here and I get you know play a little bit here.

I can maybe decrease the value of  $p$  a little bit or maybe  $k$  a little bit. How can I decrease  $p$  a little bit? Now what is  $p$ ?  $p$  is the probability that the disease will attack you. So, I can decrease that probability. So, let us say a contagion is a spreading and epidemic disease is spreading, viral disease is spreading; I can put an advertisement on TV, I can ask people to take better health measures to have proper hygiene and this will you know

decrease my value of  $p$ . So, I can either decrease my value of  $p$ . Decreasing the value of  $k$  is also very easy. What was  $k$ ?  $k$  was the number of people with this infected person was in contact with.

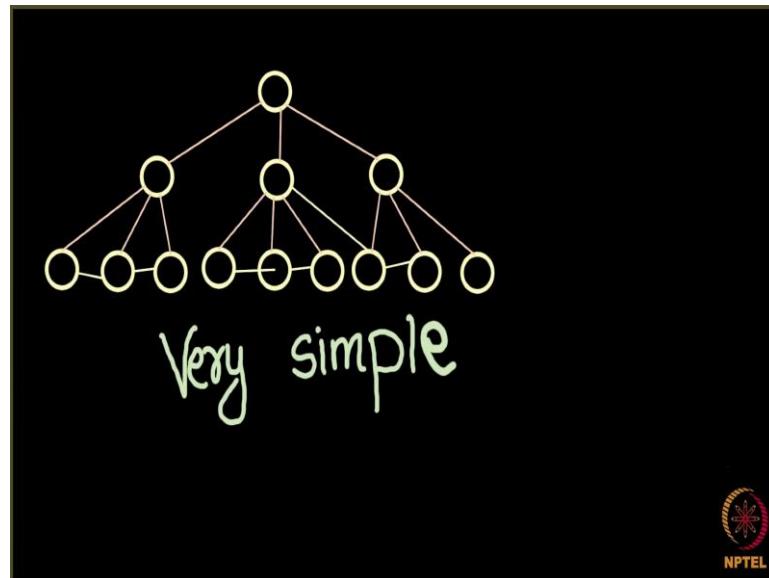
So, if we can decrease this number of people who are encountering this person. Basically, what I can do? I can for some time I can ask this person not to come out of his room that is called quarantining. So, I can isolate this person. So, as soon as I isolate this person, the value of  $k$  comes down. Either  $p$  comes down or  $k$  comes down, this value 1.2 even if it gets a little bit reduced; it will come on this side of the network and the disease will vanish. And the same thing happens here, here is a disease and the disease is not an epidemic but let us say the value of  $R_0$  is 0.9, a little bit increase in  $p$  or a little bit increase in  $k$  can now cause this disease to become an epidemic.

So, this particular property of this  $R_0$  is called knife-edge nice property. It is like the edge of an knife right, edge of an knife here. So, little bit on the side you come the disease will vanish from the network and little bit if it if there is an increase here a little bit, the disease will become an epidemic. So, that is the importance of this value, the basic reproductive number. It very well defines whether your disease is becoming an epidemic or not.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Rich Get Richer Phenomenon – 2**  
**Lecture - 133**  
**Modeling epidemics on complex networks**

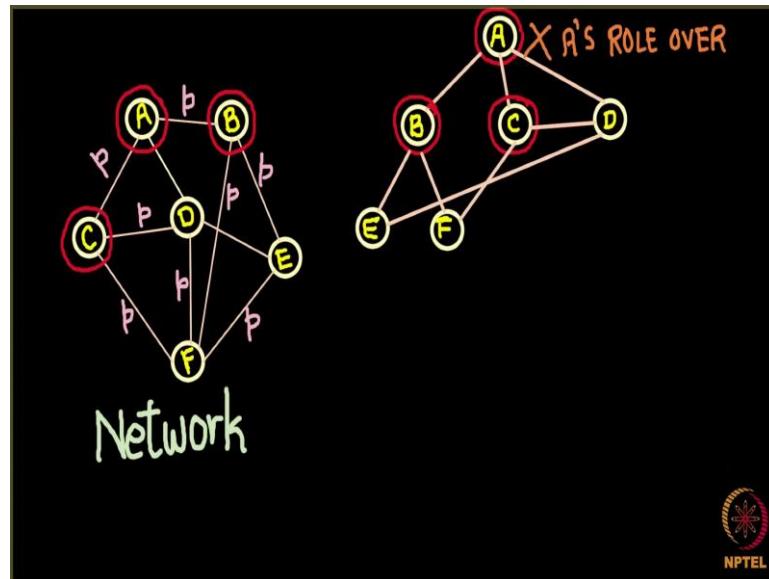
(Refer Slide Time: 00:11)



Till now we have looked at the simplest epidemic model, we will be considered the simplest network and the network was in the form of a tree something like this simple. Now, we know that a real world network do not look like this, a real world networks they look complicated and rather if you do remember the lectures from the previous weeks you will say that a networking never look like this, there should be some triadic closure here right.

So, if this guy over here is a friend with both of these there is a good chance, that both of these are friends, both of these are friends and maybe the this person here can be a friend with this one So, networks actually look very complicated something like this. How can we modify a most simple branching process what we did before for a real-world networks which are not trees? That we are going to see in this chapter.

(Refer Slide Time: 01:07)



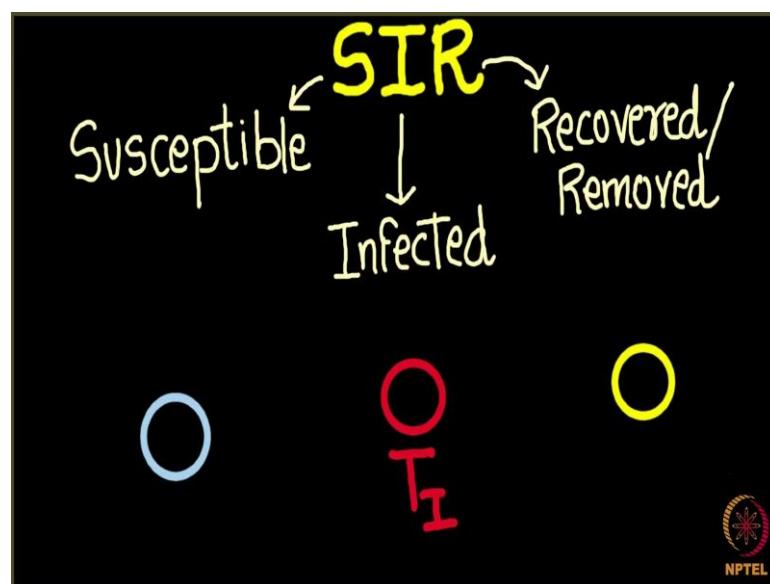
Let us say that we have a complicated network, not a tree. By complicate it what I mean is not a tree which looks something like this I have some nodes A, B, C, D, E, F here and every edge has a probability  $p$  of transmitting the infection as before. So, everything is same every edge is having a same probability. The only differences, in the layout of the network instead of the tree we have taken a proper network.

Let us say A is connected to D also here and the network look something like this. Now, I want to simulate the spreading of a contagion here how do we do that? Say initially A is infected and now A has three neighbors B, C and D and it tries to infect each of these again independently with the probability  $p$ . So, again here are three independent processes, we can flip coin for the edge A to C and if head comes, we infect C. Similarly, we can flip a coin for the edge A to D and if a head comes, we mark D as infect it and same thing we can repeat for B. So, maybe A is able to infect B and C.

How can I layout? Layout in the form of a tree, what is happening here? Let us see here is A, and A was having three neighbors B C and D and A was able to infect B and C. And now you see as role is over. I will discuss more about it. What I mean here in a branching model also at this step A's role was over A go to one chance to infect and A infected B and C. Now, it is B and C's turn now B and C will go ahead and see which of their neighbors they can infect; why so?

It should be so right because if A keeps getting a chance rather if every node keeps getting a chance to infect their neighbors at ever time period then this entire network will get infected. And that is not what happens in a real world also right you get infected from some disease and then you remain infected for some period of time. During this time you keep infecting other people and after that you might recover from the disease. So, we need a better model here let us make this picture clearer. So, what will be doing now is we will be looking at a very realistic diseases spreading model branching process was something very theoretical and very simple.

(Refer Slide Time: 03:52)



Now, we are moving little bit more towards what happens in the real world and we are going to study a brand-new model not for the tree networks, but for any kind of networks and the model is SIR epidemic model. Why is it called SIR epidemic model is because it tells you, it talks about three life cycles of an infection.

So, whenever you get infected from a disease, you mainly pass through three phases. The first one is the susceptible, susceptible is the one where you are happy living your life you are not infected from any disease, you are not infected from the disease. And the second stage is infected, so you come in contact with somebody who is infected and you get infected and the last one is recovered or removed.

Recovered or removed is something that needs to be taken very seriously here, something which is very important. What do we mean by the recovered and removed

here is what makes this model the, what makes this model spatial this SIR model it is used for two kinds of diseases, first kind of disease is something which is the terminal illness. What happens in the case of a terminal illness is patient ultimately dies away. You are susceptible and then you get infected and finally, you will be killed by the disease and when you get killed by the disease you will be removed from the network.

Or a second case; second case is the diseases something like that which can occur to you only once in your lifetime. Something like measles, you are susceptible and then you get infected from measles. Now what will happen when you get infected from measles is your body will develop a lifetime immunity against this disease and you are never going to catch this disease again that time, we say that you are recovered. You are never going to go from infected back to the susceptible this disease is going to happen to you only once in your lifetime. So, either you will gain a permanent immunity, or you will be killed by this disease that is our SIR epidemic model.

In our next lecture we will see how can we simulate this model on an actual network; we are going to do that the forgoing. So, we are having. So, a node can here exist in three states as we saw and exactly one state at a time you cannot be susceptible and infected at the same time. And infected and recovered both at the same time you can be only in one phase at one time.

If you are susceptible I am going to show you by a blue circle, if you are infected I am going to show you by an infected circle I will tell you what this  $T_I$  is and if you are recovered or removed I am going to show you by yellow circle. So, yellow circles can be ignored from a network because they have recovered and removed, they have nothing to do with the spreading of the disease now.

What is this  $T_I$  here?  $T_I$  here is again something very logical what is  $T_I$ ? When you get infected there is some time period for which you remain infected it can be one day, two-day, one week, two weeks, during this time only you are going to infect other people. So,  $T_I$  define the time for which you are going to remain infected. So, maybe for the flu this time can be let us say three days. So, during these three days you will be infected with flu and you will keep infecting other people.

If you look at this figure here, where I said that here is A, and then A infected these two people B and C. And then we said that as role is over and these are B and C were going

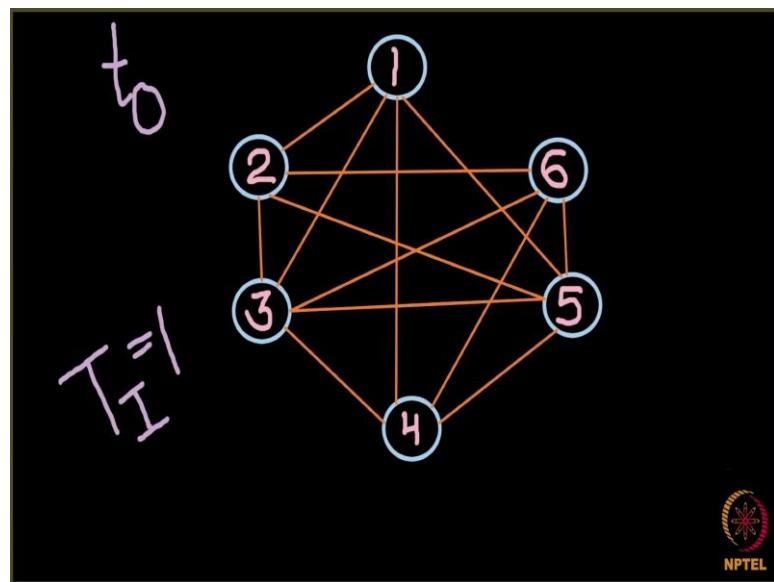
to further infect other people. Can you tell me what is the value of  $T_I$  here? So,  $T_I$  the time for which you are infecting here is 1 right because it was 2, after A infected B and C A will again get a chance to infect once more, we will look at it in detail just wanted to tell you the use of  $T_I$ . So, this is it this is an SIR model and in our next lecture we will see that how we can simulate this SIR model on a given network.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Rich Get Richer Phenomenon – 2**  
**Lecture - 134**  
**SIR and SIS Spreading Models**

So, we are now going to see how we can simulate the spreading of this SIR model we just discussed on a network. So, we will be modeling this is spreading in the form of number of iterations or number of days. So, you can imagine a situation where there is this measles is spreading on network. We will be looking at what happens at day 1, then what happens at day 2, what happens at day 3 and so on.

(Refer Slide Time: 00:32)

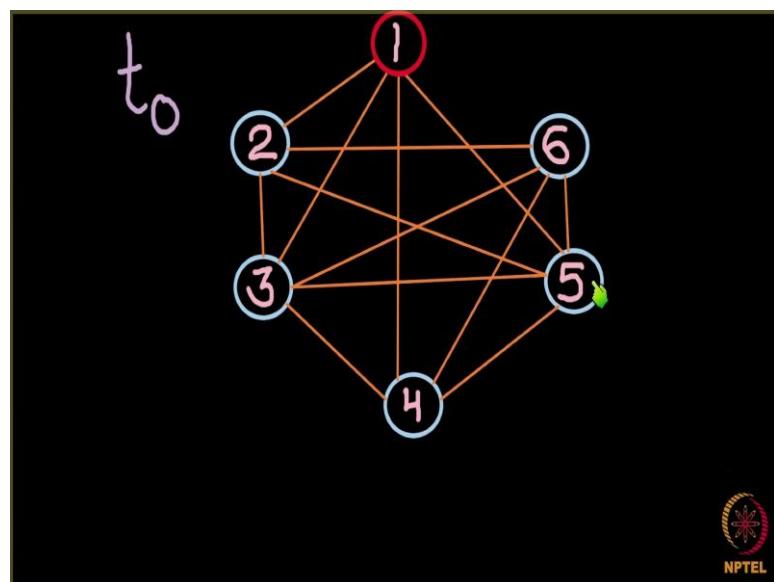


so, let us talk about day 0.  $T_I$  is for time; so, you are talking about day 0. At day 0, the lets say the network looks something like this. So, you see there are 6 nodes in this network. 6 people in this network and the blue color is for susceptible. So currently, each of these people is susceptible. None of them has acquired the disease and let us say measles. So, none of these people are suffering from measles.

And these are edges across which your infection can transmit and let us say the probability same across all the edges. Probability of infection is  $p$ . So, the pathogen for measles it is having a contagiousness value which is  $p$ . Now, let us see what will

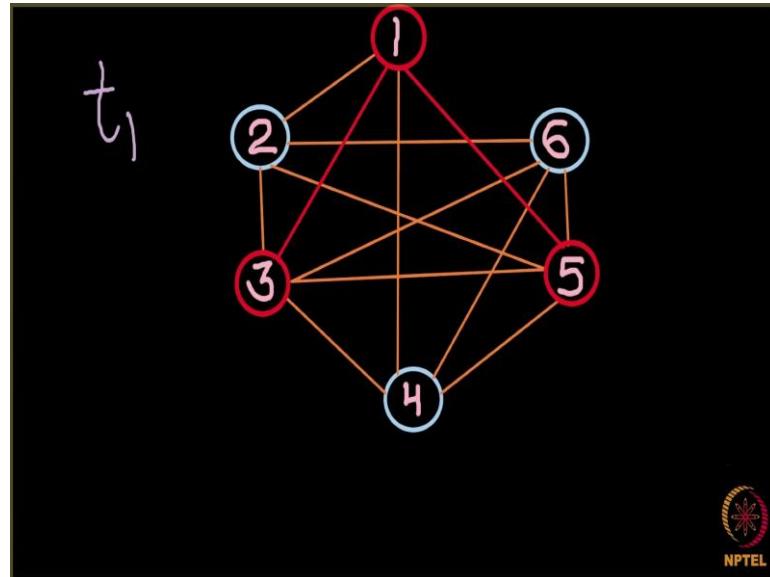
happen? Assume that and yeah one more thing, the value of TI we are going to be taken to 1. So, all though it might not sound realistic, but in the case of measles we cannot say that we remain infected for just one day and after that only you just acquire lifetime immunity. You remain infective for a long period of time, but for just for the sake of simplicity, let us take for the time being TI equals to 1. In the next example, we increase this value and see what happens, but for this example we take the value of TI to be 1. So, here is a new kind of measles, let us say. So, this measles keep you infected just for one day and after this one day you acquire lifetime immunity.

(Refer Slide Time: 02:08)



So, this is a network at day 0 and at day 0 this node 1 gets infected so, this node 1 gets measles the 0th day. What will happen on the 1st day now? Now this node 1 has 4 neighbors 2 and then 3 and then 4 and then 5. So, it has 4 neighbors and it tries each of these neighbors it tries infecting each of these neighbors with the probability of  $p$  and let us say, it is able to infect node 3 and node 5.

(Refer Slide Time: 02:38)



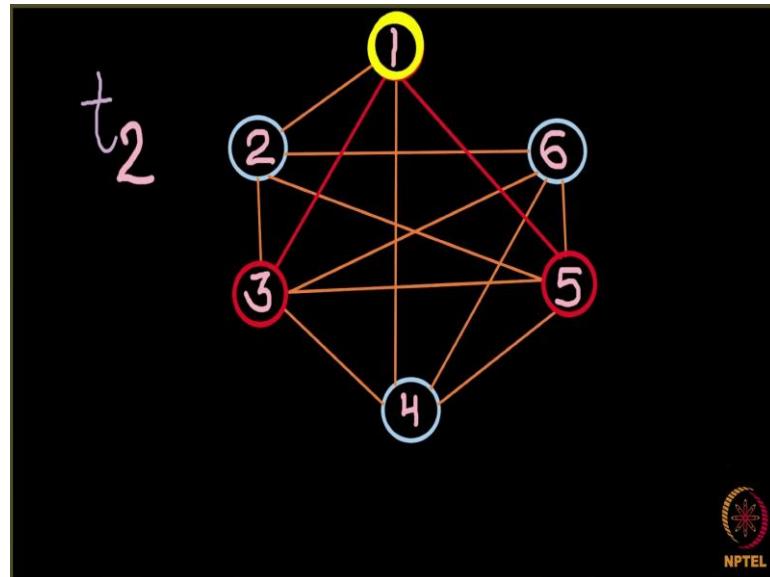
The 1st day, node 1 infects node 3 and node 5. Now you might want to ask, you might ask me a question here. How do I decide which are the nodes which node 1 is infecting? Sorry.

How do I decide it is going to be node 3 and node 5 only? And I will tell you that this is a very random process as we discussed before. So, I am going to toss a coin for each of these edges. So, if the probability of infection is 0.5, I toss up here coin for this edge, for this edge, for this edge and for this edge and here and I get a tail here, I got a head here, I got a tail here and I got a head here and I did it. And you see now, so what will happen. One simulation of this model can go like this. Where node 1 has infected node 3 and node 5, but if you simulated the second time, you again start from this graph and you again toss the coins; you might get some other result. So, it is a random process unlike in the case where we discussed how ideas and behaviors spread on a network, everything there was deterministic.

If you were given the snapshot of a network at a particular time, you would for sure with very certainty you can tell what would happen at the end. But here in this case, there is less certainty. It is more of a random process. So, let us see node 1 here, infected node 3 and node 5. Now, see node 1. Node 1 was infected at day 0 and now it is day 1. So, what is going to happen at the end of day 1? At the end of the day 1, our node 1 it is going to

be recovered. So, node 1 is discovered towards the end of day 1, but it has infected these two nodes in the network, node 3 and node 5.

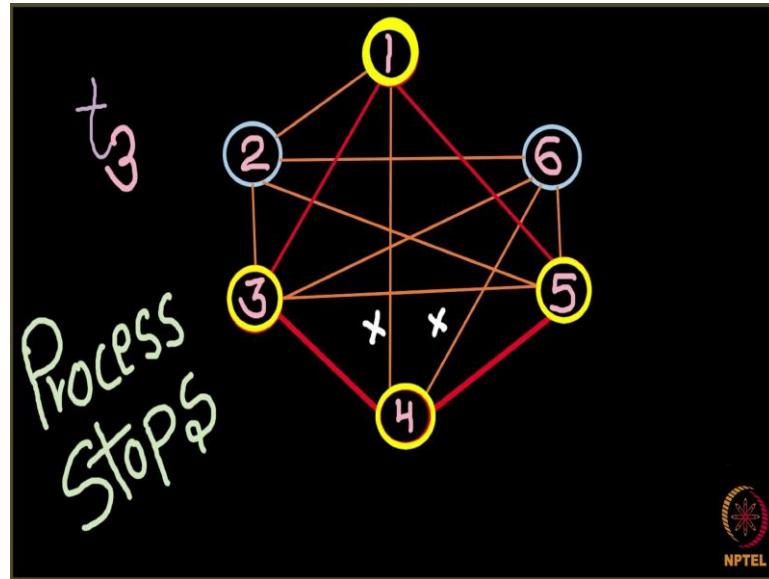
(Refer Slide Time: 04:34)



And now on day 2, node 1 do not get infected anybody, but and nodes 3 and nodes 5; node 3 and node 5 start doing their work and let us say, node 5 end up infected node 4. So, node 3 here is unable to infect any of its neighbors and node 5 here infect node 4. Node 3 here, it's an able to infect node 2 and node 6 and node 4 as well. And you see, node 3 can for sure not infect node 1.

Because node 1 it has recovered from the network. Overall node 3 gets nobody it could infect, but this person node 5 here, infects node 4. What happens next towards the end of the day 2? So, these node 3 and node 5 they were suffering from day 1. They suffered at the end of day 1, right? And now, they have passed one day of infection. So, towards the end of node 2 both of these nodes will become recovered.

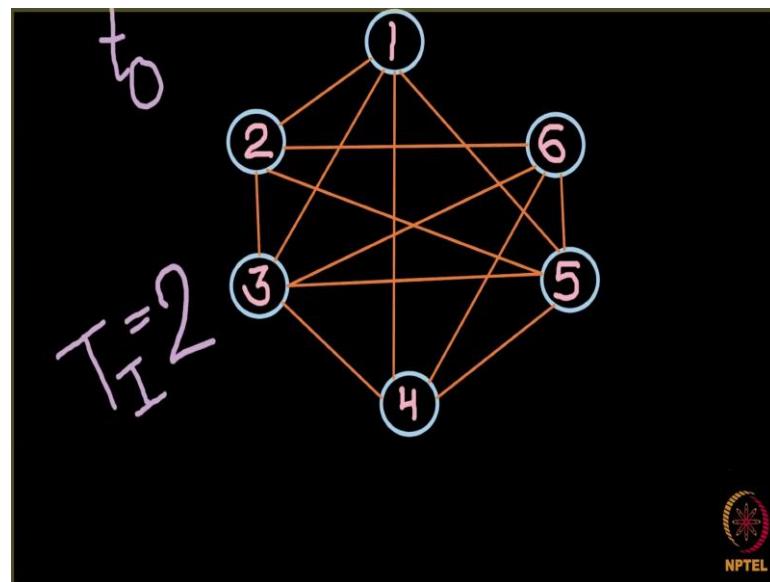
(Refer Slide Time: 05:40)



And on day 3 it is node 4 which goes around, and start infecting the people and let us say, node 4 is unable to infect anybody. Node 4 obviously cannot infect node 3, node 1 and node 5 now because they have recovered. It can infect node 6 but let us say it is unable to infect node 6 and towards the end of day 3, node 4 also gets recovered.

What will happen on day 4 now? On day 4, there are these 6 people out of which 4 have recovered towards a susceptible infection gets over because for this process to go further, we need a node, we need a red node here. We need a node which is infected, but now there is no node here which is infected, no red node here. So, what will happen? The process will stop. So, this was one simulation we did where we took the value of TI that is a infection period to be 1. Every node remains infected for one day.

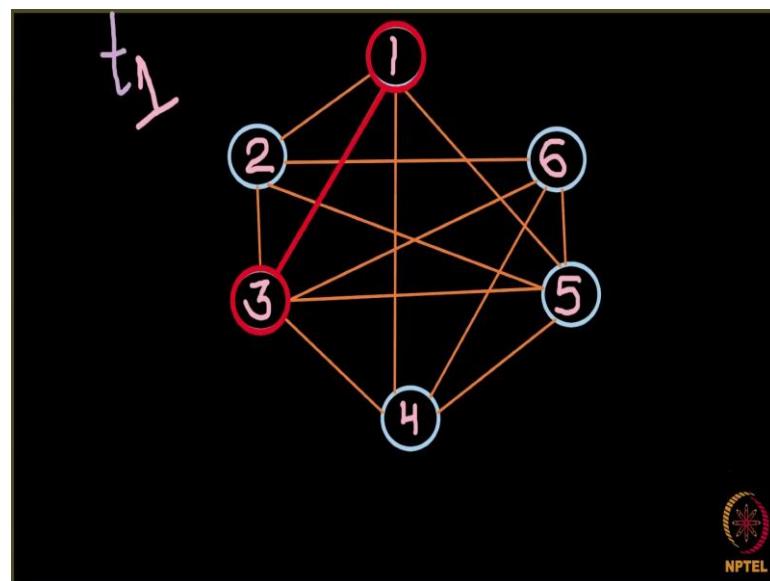
(Refer Slide Time: 06:41)



Let us now look at another example and let us say I have changed this infection period to 2. Now you remain infected with measles for 2 days and after this 2 days, you recover from the disease and again this is node 1 which catches the infection at day 0.

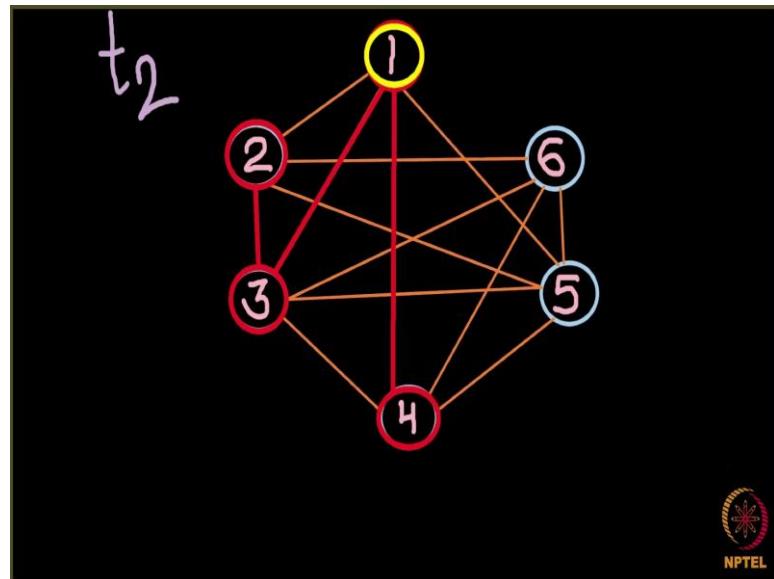
So, now, you see get 2 days. Days 1 and day 1 and day 2 to infects its neighbor and it will become it will recover at the end of day 2.

(Refer Slide Time: 07:11)



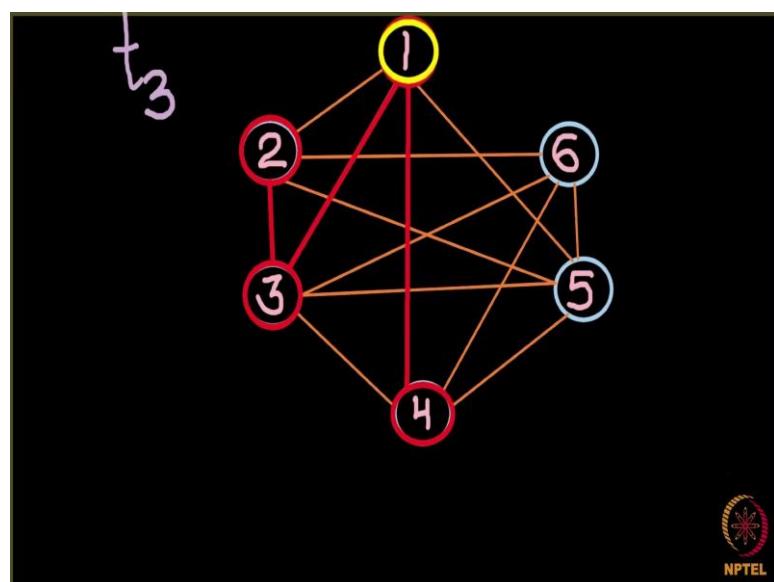
So, node 1 here, let us say day 1 it infects node 3, just node 3.

(Refer Slide Time: 07:18)



And then comes day 2. Now at day 2, both of these nodes will be infecting people. Node 1 as well as node 3. So, let us say day 2, node 3 infected node 2, node 1 is unable to infect anybody at day 2 or let us say rather node 3 infected node 2 and node 1 is able to infect node 4 at day 2. So, node 1 got day 1 to infect people and it is also getting now day 2 to infect people because it remains infected for 2 days, and towards the end of the second day, node 1 recovers.

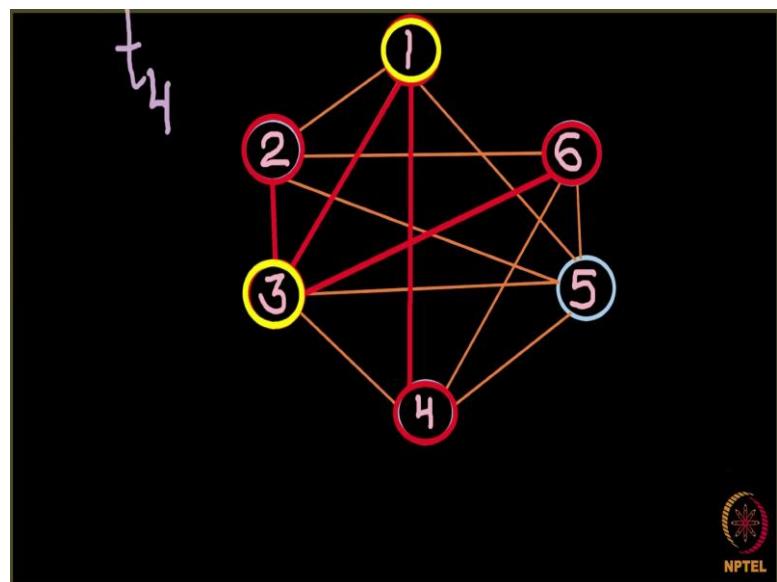
(Refer Slide Time: 07:56)



Now, during the third day, all of these three nodes, 2, 3 and 4 get infect their neighbors and let us say node 3 infects node 6.

If you look back at this network, so let me show you. So, this node 3, it was infected at day 1 and then day 2 it was infected node 2 and again at day 3 infected node 6. And at the end of day 3, what will happen? It will recover.

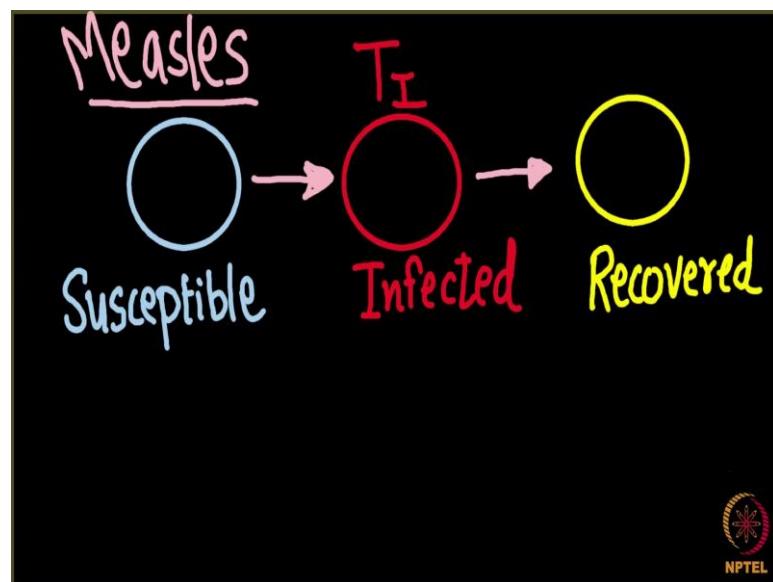
(Refer Slide Time: 08:26)



And now what will happen in day 4? Nodes 2, 4 and 6 get to infect people and let us say node 2 infects node 5 and what will happen towards the end of this day? Nodes 2 and 4 will recover.

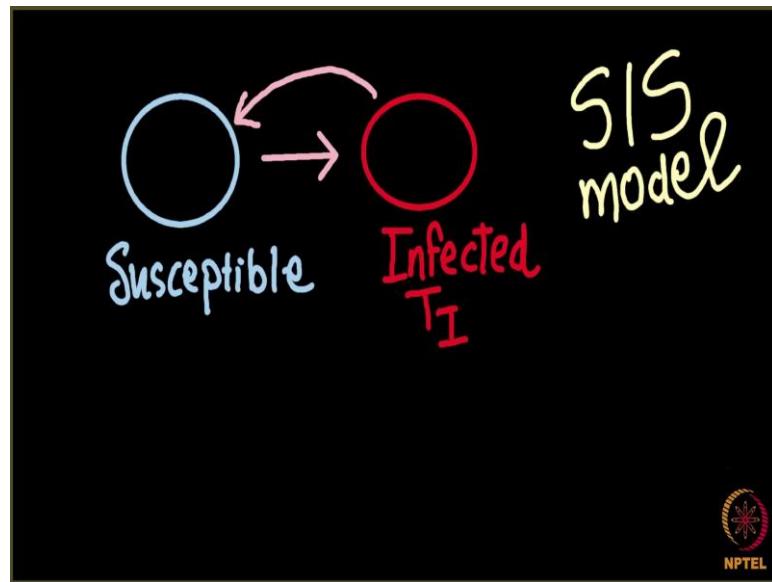
You can easily guess what will happen next, towards day 5. So, towards day 5 also, node 6 and 5 here will be infected and towards day 6 both of these will be recovered; and towards day 6 day 6 both of these will be recovered. So, at the end all the nodes in this network will recover. So, this is how we could have stimulated it for the value TI equals to 2. So, please note that it is one possible simulation. Every time you simulated it, this network will go, and it will be infected in a different way.

(Refer Slide Time: 09:23)



So here what we talked about lets quickly recap. It was a node, so the node can be either susceptible or a node can be infected and it remains infected for  $T_I$  time during which it infects other people and finally, the node gets recovered something like in the case of measles. Can I use the same model if I want to simulate the spread of a common flu? What will happen? let us say common cold. So, this infection might not hold good there, right. So, in common cold yes, you become susceptible, you are susceptible and then you become infected after coming in contact with an infected person. Then will you be permanently recovered from common cold? It never happens. What happens is, after recovering from common cold, you enjoy for some days and or let us say some months. So, let us say some years, but after that again that common cold infects you.

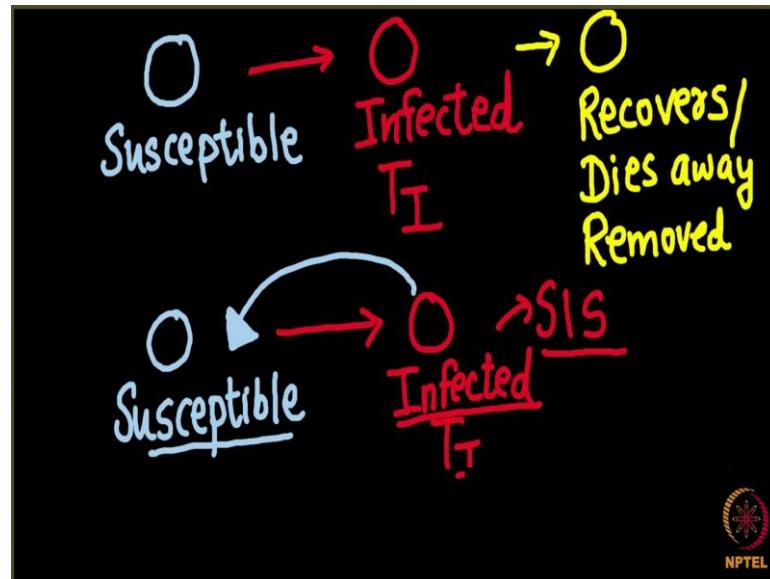
(Refer Slide Time: 10:30)



So, what happens is instead of going to this recovered state you go back to the susceptible state. So, susceptible infected and after remaining infected for  $T_I$  time period, you go back to the susceptible state and this model is called the SIS model because S refers here for susceptible, I is for infected and S is again for susceptible. So, you become susceptible, you are becoming infected and then you again become susceptible and this is used, this model is used for something like a common cold, which disease which can infect you again and again.

And I am quite sure that you can now create a simulation of SIS model on a network on your own, but still in the next lecture, I will show you how we can simulate a SIS model. How can we simulate an SIS model on a network? So now, we are going to look at 1 simulation of the SIS model. We have looked at the difference between the SIR model and SIS model. Let me quickly recap it.

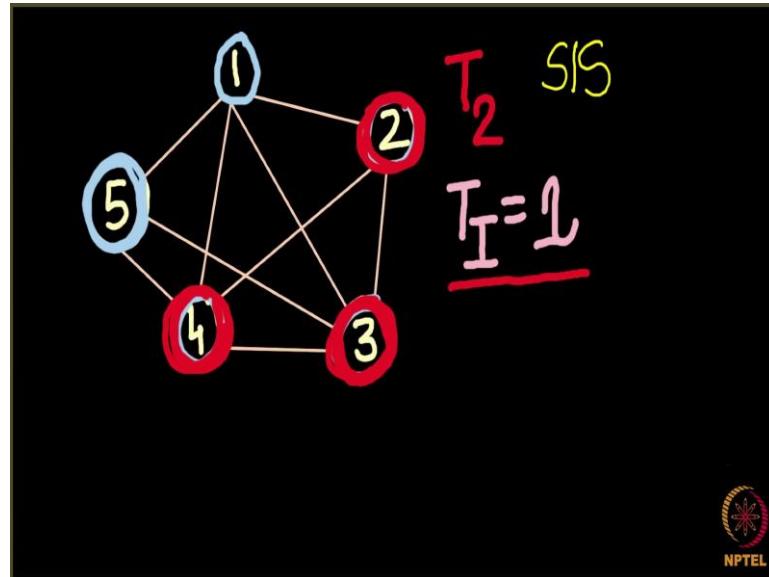
(Refer Slide Time: 11:39)



So, in SIR model here we were having a node and let us say this node is susceptible. That is, it has now got the infection and then we have seen that from susceptible, this node can go and move into an infected state. So, this happens after this node comes in contact with an infected person and this node becomes infected and it remains infected for  $T_I$  period of time. During which it keeps infecting other people and finally, what happens, this node it either recovers. That is, it develops a lifetime immunity against that, against the disease, or it leaves the network. That is, let us say this node dies away in which case we say that this node has been removed.

So, this was your SIR model and SIS model was a little bit different. It was for a disease which can infect us time and again such as a common flu, cold, common cold. So here again, we have a node which can be initially susceptible and then when it comes in contact with an infected person, this node it becomes infected and it remains infected for  $T_I$  period of time. And after this  $T_I$  period of time, what happens is it again goes back to the susceptible state. So, there is no recovered state. After some time, this node becomes susceptible again then it might come in contact with an infected person and again become infected and so on. Now, how will this simulation of such a model look like? So, this is our SIS model and we are going to see how will the simulation of such a model look like. So, let us see.

(Refer Slide Time: 13:47)



So, let us take a network again. So, I take a network here. Let us say there are 5 nodes in the network. 1 and then 2, 3, 4 and 5 and I will put some edges between these nodes. You put some edges, let us say from 1 to 5. From 1 to 2, 1 to 4, 1 to 3 let us say. And then let us say, from 2 to 3, 2 to 4, 3 to 4, 3 to 5, 4 to 5 that is it.

Now, let us say these many it's a network consisting of 5 nodes and let us see how an epidemic or how a contagion will spread here and here we are talking about the SIS spreading model, SIS. Initially, all of these nodes shown in this network they are susceptible. So we will showing susceptibility by blue color. So, let us make all these nodes to be susceptible. So, 1 is susceptible and 2 is susceptible 3, 4 and 5. All these nodes are susceptible.

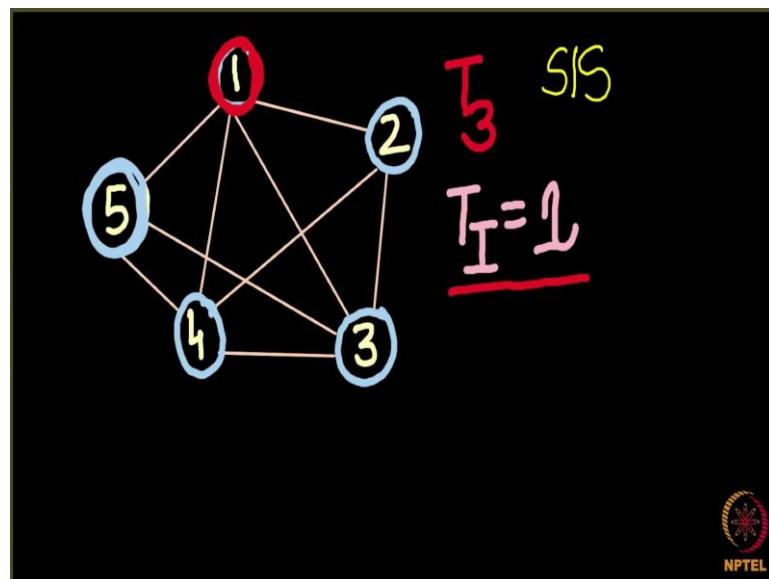
Now, this is T<sub>0</sub> and let us say, at T<sub>0</sub>, an infection counts and infects 1. So, a node 1 is now infecting from common cold and our value of T<sub>I</sub>, that is, time period during which a node remains infected let it be 1. So, one node remains infected for one day. Now, let us see what will happen during the second day. So, we are modeling what happens during, i am i am sorry, during the first day. So, initially there was a 0th day, now comes the first day. So now first day, this node 1 it will look at all of its neighbors and try to infect some of them.

So, let us say node 1 become successful in infecting this node 2 here and node 4 here. So, during day 1, node 1 infected this node 2 and node 4, and by the end of day 1 what will

happen? Node 1 will recover because at  $T_I$  it is equal to 1. Towards the end of this day, node 1 will recover. After recovering it will not go back to the recovered state, it will go back to the susceptible state as we have discussed. So, it loses its common cold and come back to the susceptible state; come back to the susceptible state.

First day over; let us now see what happens during the second day. So now is the second day. Second day node 2 and 4, we look at their neighbors and we will try infecting them and let us say they become successful in infecting node 3. So now at day 2, on day 2, node 3 catches the infection and again towards the end of day 2, node 2 and node 4 will go back to the susceptible state. So, node 2 and node 4, now go back to the susceptible states. So, this is node 2 here and node 4 here.

(Refer Slide Time: 17:45)

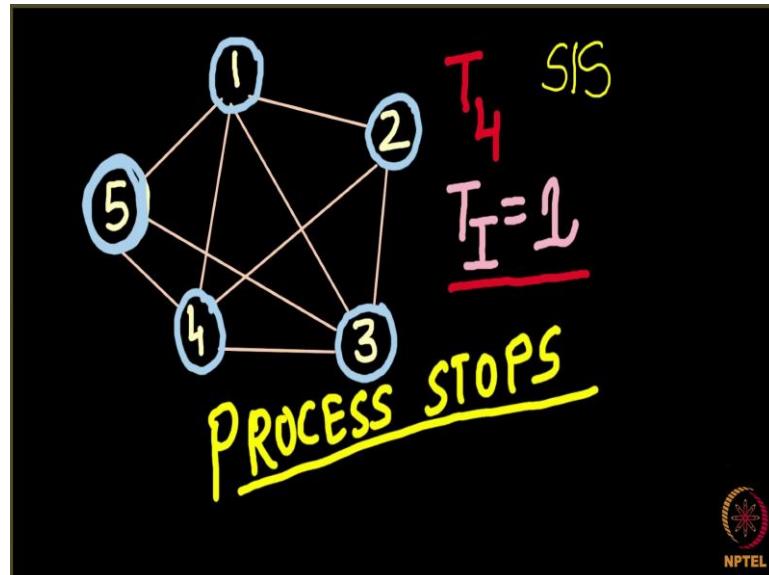


And then, what can happen during the third day let us say; now comes the third day. So, now, in the third day this node 3 is node 3 is infected. So, what node 3 will do?

Let us say it infects node 1. Do you see what happened just now? It infects node 1. So, node 1 as previously also became infected and it is infected certain people it becomes susceptible, but now this node 1 it again gets infected. So, this node 1 it gets infected here and towards the end of day 3 this node 3 now, it comes back to the susceptible state and this process keeps on going; and this process keep on going. So, node 1 may then infect certain nodes and then those nodes will infect certain nodes but suppose at this

iteration what happens is node 1 is unable to infect anybody. So, we are talking about day 4.

(Refer Slide Time: 18:41)



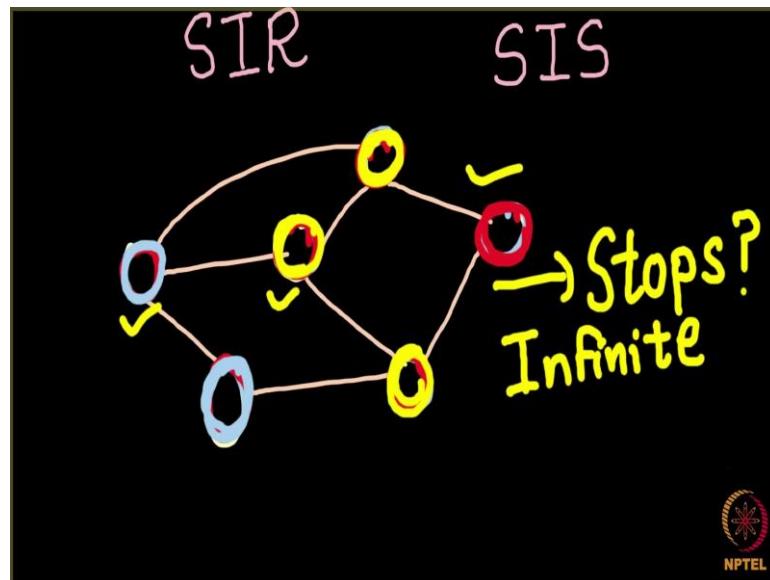
So let us say, at day 4 node 1 is enable to infect anybody and what will happen towards the end of this day; towards the end of day 4?

Node 1 will again go back to the susceptible state. And do you see now what is happened in this network? Everybody now comes back to this susceptible state there is nobody who is infected; and as soon as this happens in this network, there is not even a single node which is infected everybody is susceptible what will happen is the process stops. Now this was the simulation of SIS model on a network.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Rich Get Richer Phenomenon – 2**  
**Lecture - 135**  
**Comparison between SIR and SIS Spreading Models**

(Refer Slide Time: 00:10)



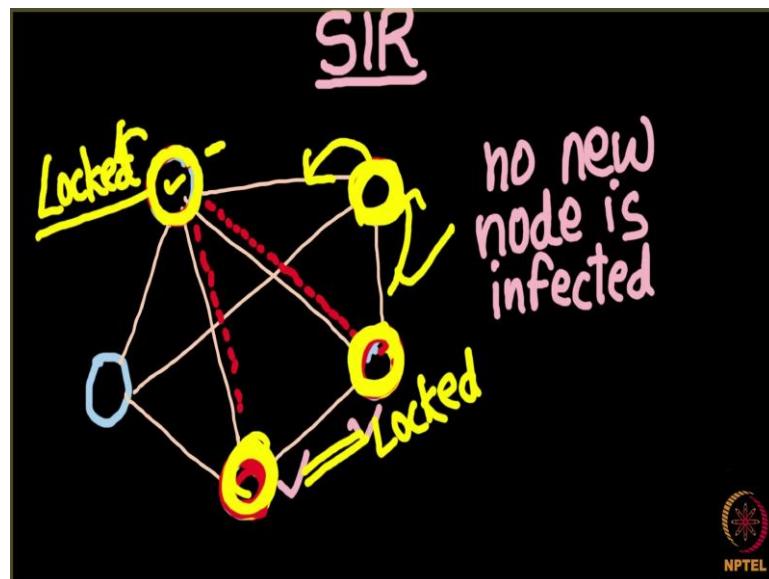
Till now we have talked about two epidemic spreading models, two disease spreading models. One was the SIR model and another was the SIS model. What was happening in both of these models was there were certain nodes in so, we were given a network and there were certain nodes in this network. And these nodes were connecting with the help of some edges, something like this.

And what was happening is, these nodes were changing their colors with time, changing their faces with time. So, initially all of these nodes were susceptible and then, some of the nodes in this network have become infected. And these infected nodes are started infecting the other people and while this was happening some nodes in the network were getting recovered as well and some nodes were turning out from infected, were going from infected state back to the susceptible state so something like this was happening.

These nodes are changing their colors with time and these nodes are changing their colors with time, shifting from one face to another. Now my question to you is this

process where these nodes are changing their colors, somebody is getting infected, and then somebody is getting susceptible, somebody is getting recovered, do you think that this process stops after some time or do this process keeps running for an infinite period of time? What do you think is the answer? You might want to pause the video two minutes and think about it.

(Refer Slide Time: 01:55)



Let us first talk about the SIR model, so let us see what happens in the SIR model. In SIR model initially every node is susceptible. So, I will take a network of five nodes here and initially every node is susceptible, let me put some edges between these nodes. All these are susceptible nodes and we have some edges between these nodes these are the edges.

And then what happens is we need at least one infected person in this network to spread infection. And let us say this is the node here which is infected and this node starts the infection. What is going to happen afterwards is, this node might infect some more nodes so let's say this node passes on the infection to this node and this node here gets infected.

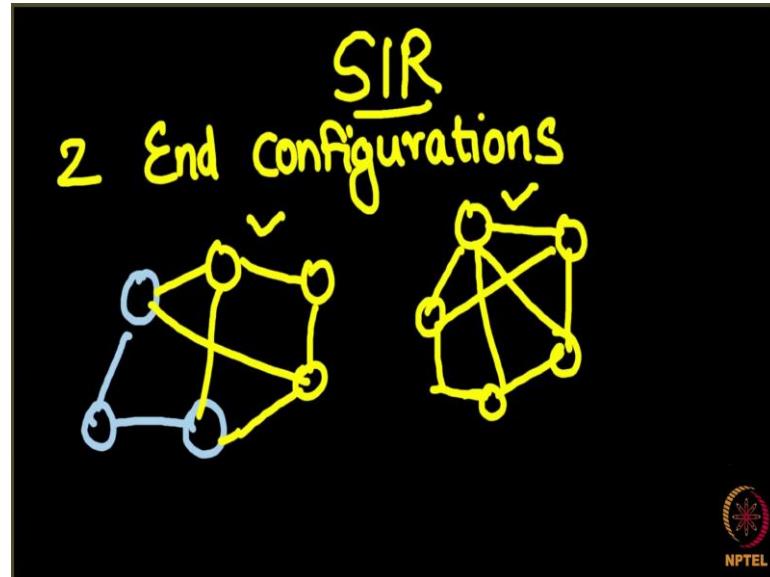
And let's say this node passes over the infection to this node also and this node also gets infected here. But what will happen after certain period of time? After some time this node will go back to the recovered state right or the removed state. And as soon as this node gets goes into the recovered state or removed state, you kind of see that it is locked, it is locked means it is kind of removed from a network. Now we can do nothing with this node it is out of question right.

And then we had these two infected nodes over here, and then these two infected nodes might infect some of the nodes or might not infect some of the nodes and get recovered. So, these nodes are also going to be recovered after some time. So, this is one possible configuration which can occur once the process stops. So, you see here the process will definitely stop in this case. So, let's say that this infection was spreading here. And what has happened there is there comes an iteration where no node is infected. So, this is an iteration where what happens is no new node is infected.

So, we have seen that when no new node is infected the process comes to an end. No new node was infected and with time both of these nodes they also went into the recovered state. Now this network is done, if the process stops nothing is going to happen. But let us say the process does not stop here. So, these nodes were red here these nodes were infected here and the process does not stop here, these nodes keep infecting another nodes even then don't you think that after some period of time this process will come to an end. Because there is a fix supply of nodes after some time both of these nodes are also going to become recovered and as soon as these two nodes become recovered they are also locked and so on.

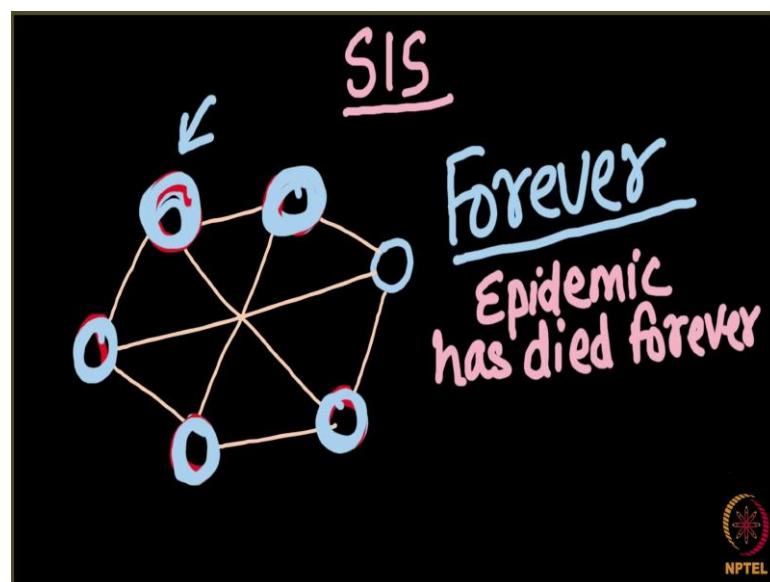
The process keeps going, keeps infecting, and the nodes starts being locked more and more nodes become locked and the time will compare every other node in the network is locked. So, this process will come to an end eventually. So, what can be the ending configuration for this process?

(Refer Slide Time: 05:14)



If we are talking about SIR model there are two end configurations, two end configurations. And the one ending configuration is there are certain nodes in the network which are still susceptible. And then the rest of the nodes in this network are recovered is the one possible configuration there are some edges between them. And second possible configuration is every node in this network has become recovered. And whenever the SIR process runs on any network it will end here or it will end here, it cannot keep running for an indefinite period of time.

(Refer Slide Time: 06:05)

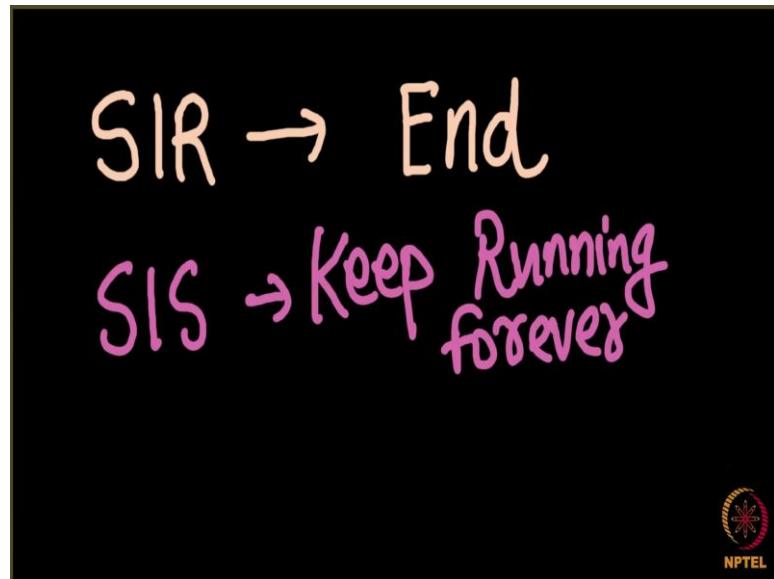


Let us now see what happens in the case of a SIS model. So, here I am talking about SIS model, let us see whether it will come to an end or not. So, there are certain susceptible nodes here I draw some susceptible nodes and these are connected with some edges here we make some edges and then the infection will start from some node. So, let us say this is the node from where the infection starts. And then what will happen with time again this node is going to infect some other nodes. Let us say this node infects these two nodes and what will happen is, this node will now go back to the susceptible state.

So, you see it is not locked here it goes back to the susceptible state and now these two nodes they will infect some more nodes let us say these and these and now these two nodes they go to the susceptible state and then these two again two infected nodes these two infected nodes they can again infect some nodes. So, let us say they infect this node over here. So, this node gets infected and you see it can be a cycle, it can be an indefinite cycle, rather the simplest form of cycle let me tell you.

So, this node over here let us say this node infects this node and it then it becomes susceptible. After some time, this node infects this node so this node is infected and this node becomes susceptible and this process can keep running for forever, this process can keep running forever. So, there is no guarantee that a SIS process will ever stop, but once it stops suppose that these two nodes are also recovered and in the next iteration this node over here it is unable to infect any other node and then it will also turn into susceptible. In this case you see what happens, in this case your epidemic or your contagion has died forever.

(Refer Slide Time: 08:27)



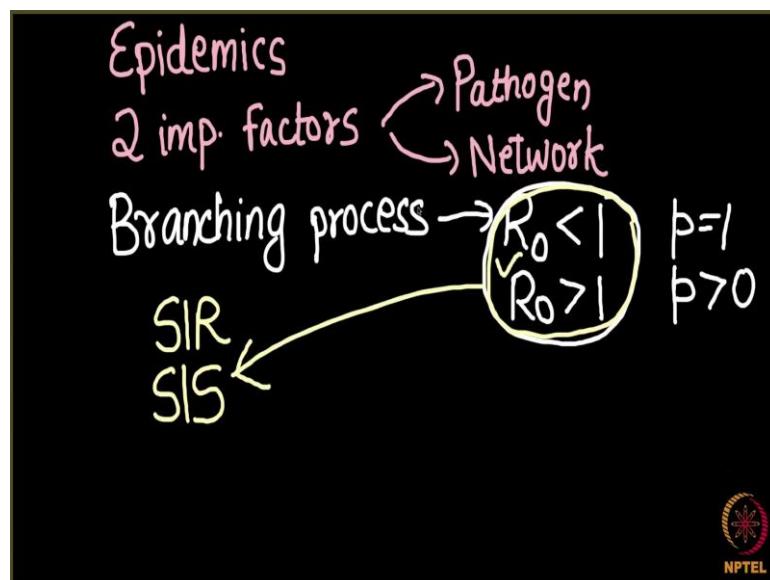
So, what did we see? We see that in the case of in this lecture we saw that, SIR model will definitely come to an end. But if we talk about the SIS model, it can keep running forever, it can keep running forever.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Rich Get Richer Phenomenon - 2**  
**Lecture - 136**  
**Basic Reproductive Number Revisited for Complex Networks**

Before going further let us quickly recap what all we have done in this chapter.

(Refer Slide Time: 00:12)



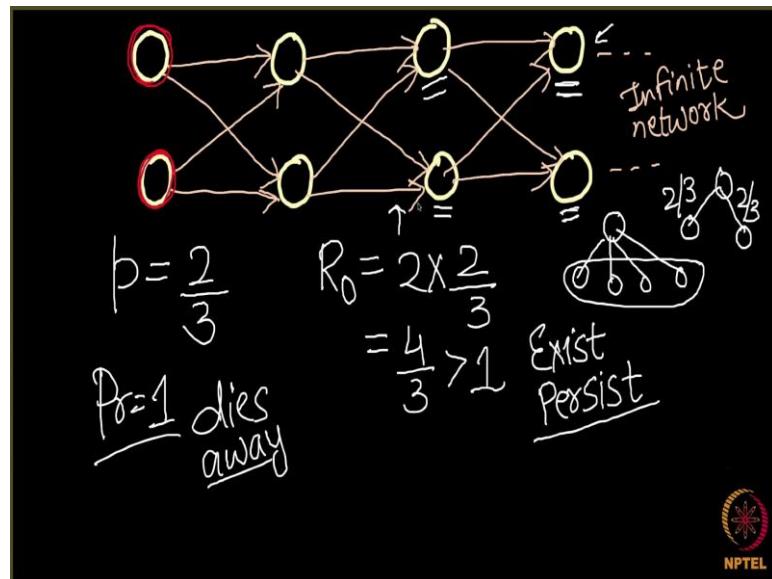
First of all we have looked at what are epidemics and how network scientists can study epidemics. Then we have looked at 2 important factors which are required for the modeling of a contagious disease. And what were those factors? The first one was a pathogen which was spreading and the second one was a network, network on which our contagion is spreading. After that we looked at a very simple process for modeling this spreading of a contagion, which is a branching model, branching process. And, one of the very interesting concept we looked in the branching process was the reproductive number  $R_0$ .

We have seen that when  $R_0$  was less than 1 a disease it spreads sorry a disease it dies away with the probability equals to 1. And, when our  $R_0$  it is greater than 1 our disease persists in the network with some positive probability and  $R_0$  was responsible was very

helpful in telling us whether a disease is going to be an epidemic or not. And, then we looked at 2 little bit complicated spreading models SIR model and SIS model.

Now, again I will pose a question to you and the question is like here we have seen that there is a reproductive number, which if less than 1 it means that disease for sure dies away. If it is greater than 1 then, then it is quite sure that the not completely sure, but quite sure that the disease will persist in the network. This theory which we did for the branching process, do you think that it will hold here, and the answer can be no, the answer is no. Why this theory does not hold here?

(Refer Slide Time: 02:14)



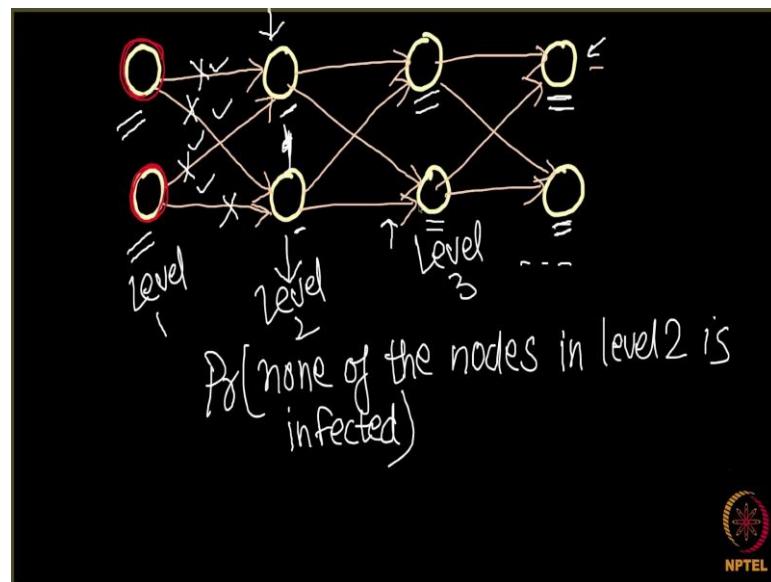
Let us take a small example and try to figure it out. Say we have a network here and let us say network is in this form I will give you a network and we adjust somewhat like this. So, this node is connected to 2 nodes on the next level and this node is connected to both the nodes on the next level. And similarly, this node here, this node here and this and this network goes on like this. So, it is an infinite network. And here I tell you that initially this node and these nodes. So, these 2 nodes are infected initially and then I tell you that the probability of your infection is spreading is  $2/3$ . So, let us try to use the concept of basic reproductive number.

What was a basic reproductive number? What did  $R_0$  define?  $R_0$  told us the number of secondary infections. So, if a node is here and it has some neighbors here so, on an average how many of the people in this level will be infected, that was what  $R_0$  told us.

So, if we look at this network what is happening is every node. So, if we look at this node here this holds for any node in this network, if we look at this node here it is connected to this node and this node. So, 2 nodes in the next level, if I look at this node it is again connected to 2 nodes in the next level.

So, every node is having 2 children and each of these can be infected with the probability of 2 by 3. So, what is my basic reproductive number? It turns out to be 2 into 2 by 3 which is 4 by 3 which is greater than 1, which means that there should be a quite a high probability that my disease you know should exist on this network. There should be a high probability that my disease it should persist in the network, but it is not so, it does not happen like this. Rather, I show you that on this network this disease it for sure with a actually with a probability equals to 1 the disease dies away, which means that our old theory of basic reproductive number is not working here in the case of SIS and SIR model. So, let us see when will this disease die away. So, we will be taking this network.

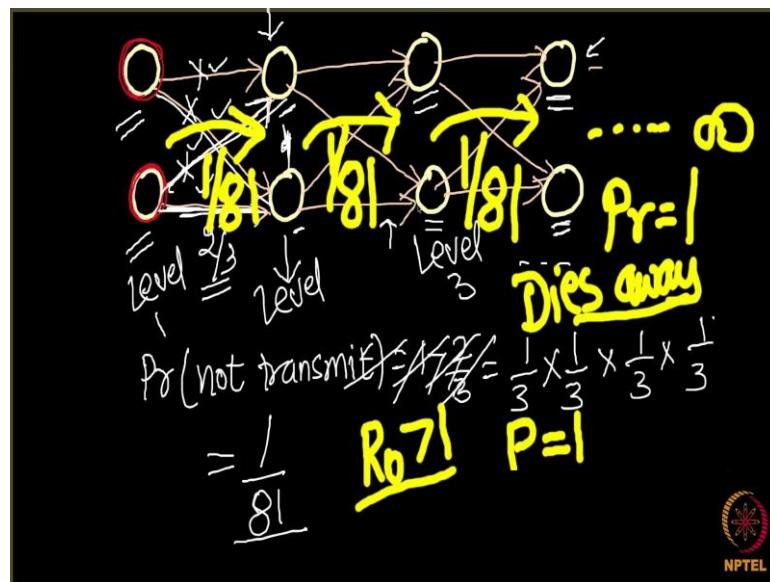
(Refer Slide Time: 05:05)



So, here is the here is our network, initially these 2 nodes are infected, let us look what is the probability that your infection does not come to this level. So, let us say that this is level 1 here, this is level 2, this is level 3 and so on. We are interested in looking at the probability that none of the nodes at level 2 is infected. What is the probability that none of the nodes in level 2 is infected and it is actually simple to calculate. So, when will be a node here infected?

If any of these four links workout then either this node will be infected, or this node will be infected right. So, when we none of these nodes infected when this link also fails, this link also fails, this link also fails and this link also fails, even if one of these links work my infection will come here to the second level. So, none of these links should work. What is the probability that none of these links will work? Let us see.

(Refer Slide Time: 06:27)



So, probability of infection here was  $2/3$  for every edge right,  $2/3$  is the probability that my infection will transmit across this edge. So, what is the probability that my infection will not transmit through this edge is nothing, but  $1 - 2/3$  which is equal to  $1/3$  right. And, then what is a probability that my disease does not get transmitted from here is again  $1/3$ . So, I am going to multiply this, I am going to multiply this  $1/3$  and again  $1/3$  for the third edge and again  $1/3$  for the fourth edge.

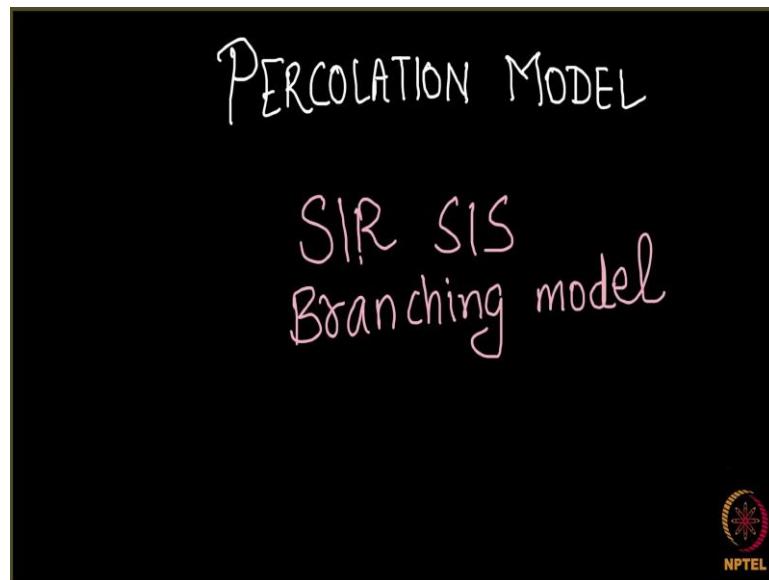
So, it turns out to be  $1/81$ , this  $1/81$  is the probability that my infection it does not get transmitted from my level 1 to level 2.  $1/81$  is the probability that my infection does not get transmitted from this level to this level, which is actually a high probability which is on a higher side. When you say there is a high probability that my infection will die away here, even if 1 of the node gets infected here again for going so, from here to here again the disease dies away with the probability  $1/81$ . Again, from here to here the disease dies away with the probability  $1/81$  and since it is an infinite network with the probability equals to 1, your disease it dies away from the network.

So, even if you see your basic reproductive number was greater than 1, what is happening is counterintuitive, what is happening is with the probability equals to 1 your disease is dying away from this network.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

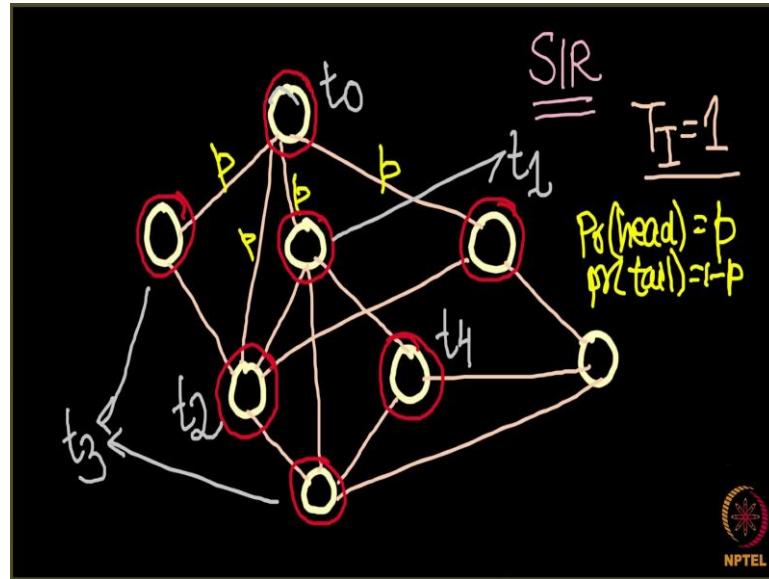
**Rich Get Richer Phenomenon – 2**  
**Lecture - 137**  
**Percolation model**

(Refer Slide Time: 00:10)



In this lecture we are going to now study a very interesting model which is known as the Percolation model. And this percolation model it is actually not a new model for you know modeling the spreading of your disease is rather it is a new angle, new way of looking at an already existing model. SIR model SIS model and for a matter of an event for your branching model, simple branching model; we are going to look at these models from a different angle. I will tell you what is the different angle.

(Refer Slide Time: 00:49)



So, let us quickly see we will simulate we will quickly simulate SIR model on a network and then we will see how can we look at it in a new manner. So, let us take a network here. So, I will just quickly make a network over here, I will take some nodes and we will take some edges, (Refer Time: 01:28) edges. So, I think this should be enough ok. So, we take this network over here and now we simulate a SIR model on it. And, let us say our  $T_I = 1$ , that is every node remains infected for 1 time period and after this 1 time period or 1 day that node stops infecting other nodes and let us say the infection starts from this node. So, this is our node which is infected at time 0, the 0 th day this node is infected.

Then now this node has 4 neighbors and it gets a chance to infect each of these neighbors. So, I am sure that you know how this process is occurring. So, each of the edges over here has a probability  $p$  of infection. So, what we are going to do is toss a coin for this particular edge and this will be a biased coin, biased coin such that probability of head on that coin will be  $p$  and probability of tail on that coin will be 1 minus  $p$  right. So, we toss a coin over here and see whether it is a head or tail, if its head we will infect this node else leave it. Similarly, for this edge, for this edge and for this edge and let us say after doing this 4 coin tosses this node over here and this node over here gets infected.

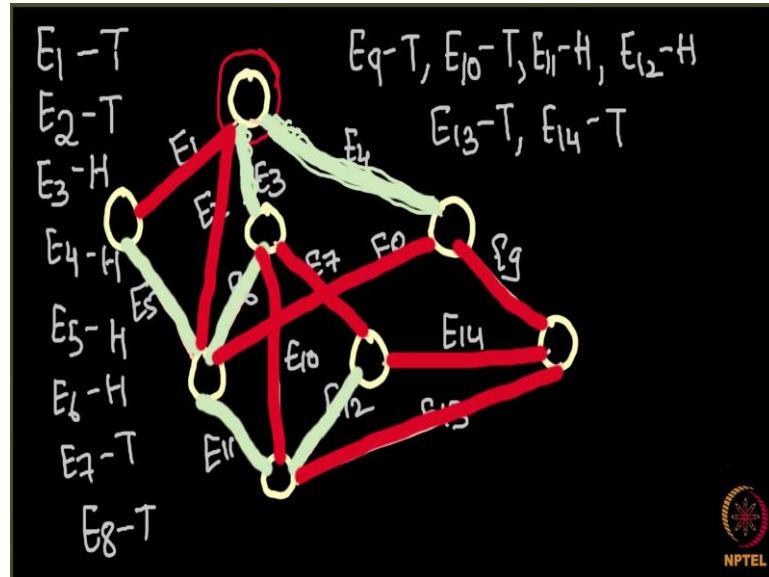
So, both of these nodes they get infected at time  $t_1$ . Now, in the next iteration both of this particular node it recovers. and it is no longer able to infect other notes while, these 2 nodes over here they now go ahead and they infect the network. And, let us say that in the next time step that is at time 2 this node over here gets infected so, this node gets infected at time  $t_2$ . And, then again both of these nodes recover and then this node over here it sees, and it tries infecting other nodes. And, let us say it ends up infecting this node over here and this node over here.

So, both of these nodes they get infected at time  $t_3$  and then this node has a chance for these 2 nodes now infect further and let us say that in the next iteration this node over here gets infected. So, this happens at time  $t_4$ . So, how do we view this process? We are viewing this process as; we are viewing this process as there was this time 0, when this node gets infected day 0. Then it gets a chance to infect its neighbors and that at time 1 at day 1 these 2 nodes get infected and then time 2 these 2 nodes get infected and so on. So, we look at this process in terms of time.

Now, percolation model asks you the question that can you remove this notion of time from here. It will seem to me to be a very dynamic process everything is running dynamically time 1, time 2, times 3, time 4 can you make static. So, let us say I want to know I was having this initial network over there ok. So, initially my network was like this and then this wait and this node over here, this node was infected. And, let us say now I am interested in knowing when this process will run whether this node over here will get infected or not.

And, then what we did for knowing whether this node will get infected or not we ran the simulation in terms of time. We looked at what will happen at time 0, what will happen that time 1, what will happen at time 2 and so on. And finally, we see that yes this node over here is infected. There is a rather of very new way a novel way to do this a very interesting one, where this process will not run in terms of time. We look at it as a very static process and that is what is called the percolation model. So, how do I make this entire process static is so, I take this network. So, this is my network and I want to see how I can make everything static over here. This is the network; this is the network I am having and no wait 1 second ok.

(Refer Slide Time: 06:30)



This is the network we are having and then we know that at the beginning this node over here is infected. And, we are interested to know whether this node will get infected or not and we know all this is a complete game of tossing a coin only. At time 1 we have tossed a coin for this, whatever gets infected; in the second iteration we toss a coin for that particular node whatever gets infected in third iteration we toss a coin for that particular node and so on.

Now, see what we will do is, first I will just quickly label all of these edges. So, let us say this is E1 E2 E3 E4 E5 E6 E7 E8 E9 10 11 12 13. So, there are 14 edges in this network, what I am going to do is instead of running this process time wise I am now going to flip 14 coins 1 for each edge. So, first I flip a coin here for this edge E1. So, I flip a coin for E1 and let us say I see a tail and then I flip a coin for E2 and I see a tail, E3 let us say I get a head. E4 let us say I again get a head, E5 let us say head, E6 let us say head we are getting so, many heads.

E7 I got a tail, E8 I got a tail and then let me toss it for E9 I get a tail. E10 tail, E11 let us say head, E12 again a head, E13 tail, E14 tail ok. So, see what I am doing is I have flipped a coin for all the edges at the beginning itself, I am not running this process time wise. Now, what do I do in this graph wherever I see a head. So, I imagine this graph in the form of let us say pipes. So, here there are the nodes, and these are the pipes through which let us say some water can flow and actually here some contagion is flowing.

So, these are the pipes through which water flows across these nodes and whenever a head occurs so, I say that a pipe is open and whenever a tail occurs I say that the pipe is closed. So, let me show let us have opened pipes in green color. So, let me open some pipes. So, E 3 is a head and then E 4 is a head, E 4 is a head 1 3, E 3 is a head, E 4 is a head and then E 5 and then E 6, E 7 I am sorry not E 7 E 6 and then we have E 11, where is E 11 E 11 and we have E 12 ok.

So, all these are the open pipes and remaining whenever I get tail I say that these are these are all the closed pipes. So, this is a closed pipe, this is a closed pipe and then closed, closed, closed, closed and these are the closed pipes. So, we can now see these nodes as connected in the form of pipes, where the green pipes are the open and red ones are the closed; now what it has to do with telling us whether some node was infected or not. So, we started our infection over here and we were interested in the question whether towards the end of the process this node over here will get infected or not.

And now, I tell you the answer; the answer says that imagine here is this water flowing from this node and we see that some of the pipes here are closed and some of the pipes here are open. So, I say that if water is able to flow from this node to this node, yes this node will get infected as this node will not get infected. And, you can match it with what we have done previously. So, whichever nodes get got infected are the ones you which you can reach with the help of these open pipes. So, if you want to see whether some node is infected or not let us say I want to know whether this node is infected or not, I will see whether there is a path of open pipes to this node from this node.

So, let us say here and then I can go here, here and here I see that there is a path. So, yes this node will get infected at the end and let us say I want to talk about this node over here. So, I see that there is no path of open pipes from this node over here to this node. So, this node will not be infected towards the end of the process. And, you see we have seen that this entire process of the infection of different nodes in the network in an SIR or let us say SIS model without using any notion of time. So, this is called the percolation model and it is very useful in the analysis, in the theoretical analysis of such processes.

And, I think that by now it would be clear why use the term percolation. We say the term percolation because it is like this water is percolating through this network. And, we can use it in the modeling of the epidemic diseases.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

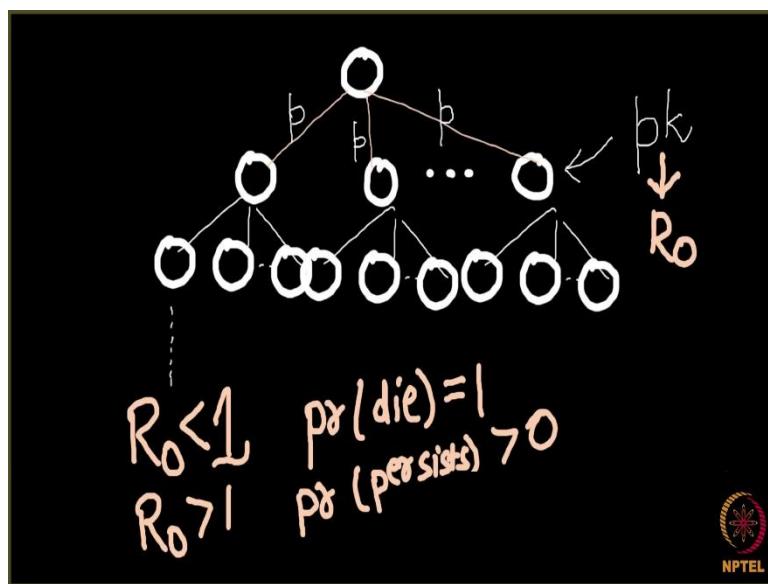
**Rich Get Richer Phenomenon – 2**

**Lecture - 138**

**Analysis of basic reproductive number in branching model (The problem statement)**

In this lecture we are going to do the Analysis of basic reproductive number.

(Refer Slide Time: 00:18)



So, if you do remember when we talked about a branching model, so branching model where there is a node here and then this node is having  $k$  neighbors. So, this node here it is having  $k$  neighbors right and then each of these  $k$  neighbors, each of these  $k$  neighbors are again having some  $k$  neighbors. So, something like this, each of these  $k$  neighbors are again having  $k$  neighbors (Refer Time: 00:52) little bit of overlapping this and so on right.

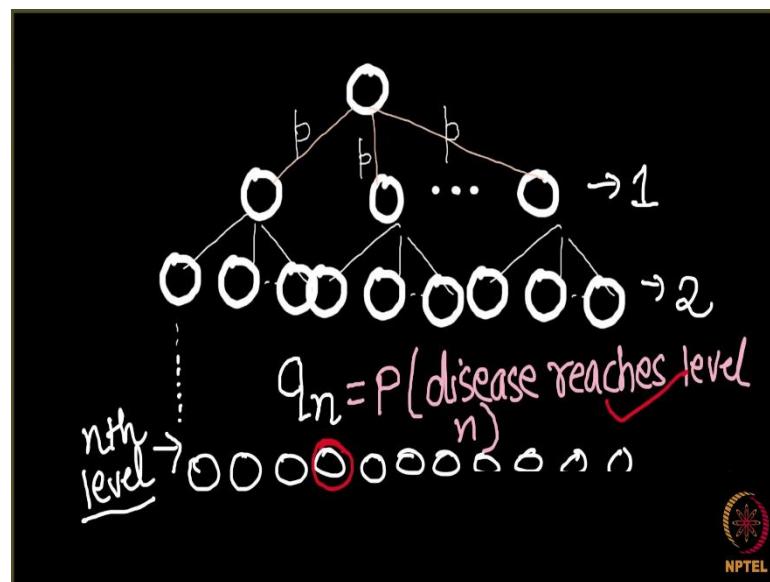
So, this was our basic branching model and then we have looked at the number of expected people who get infected here was given as  $p$  into  $k$ , where  $p$  is the probability that this disease get transmitted across this edge. So, here we were having a number  $p$  into  $k$  and we call this number as basic reproductive number  $R_0$ .

And then we made a claim and the claim was we said that if  $R_0 < 1$  then the probability that this disease die, the probability that this disease dies equals to 1. That is if  $R_0 < 1$  for

sure this disease dies away and if  $R_0$  is greater than 1 then with the positive probability the disease persists in the network.

So, the probability that this disease persists in the network is greater than 0, we said this. And we will look at how intuitively this is true, but we have not looked at a rigorous proof. So, in this lecture we will be doing a rigorous proof for this and the proof is actually very interesting. Let us see how we can do it. So, let us model it a little bit. So, what do we do is, we have this network over here ok.

(Refer Slide Time: 02:32)



We have this network here right. First of all, let us see at when do I call this disease to become epidemic. So, let us see this is level 1, this is level 2 and somewhere down here is some level which is the nth level.

So, this is my nth level right and then I take a variable I take a notation let us say  $q_n$ . And what is  $q_n$ ?  $q_n$  is the probability that the disease reaches level n. Reaches level n means at least 1 person at this level n should be infected. So, if these are the people at level n say, so at least 1 person in this level should be infected. And if 1 person at this level is infected, I say that my disease reaches level n. So,  $q_n$  becomes a probability that my disease reaches level n ok.

(Refer Slide Time: 03:45)

$$\begin{aligned} q_n &= P(\text{disease reaches level } n) \\ &= P(\text{disease persists level } n) \end{aligned}$$

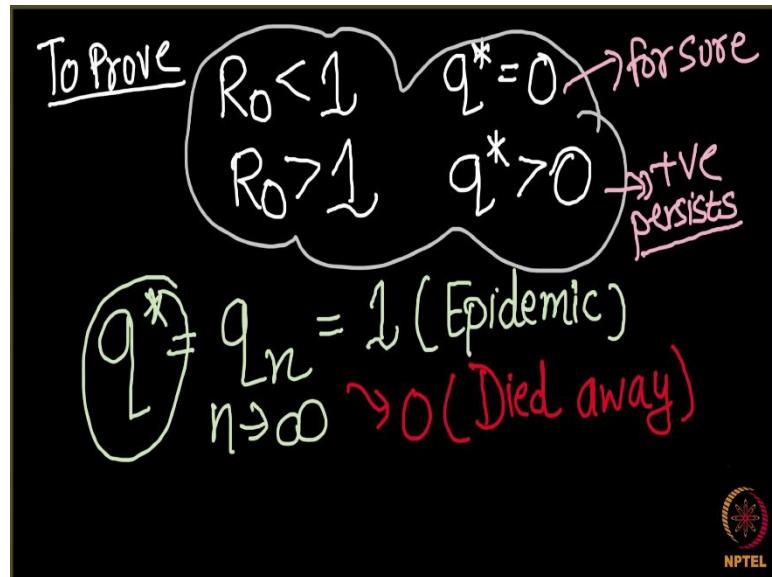
$q^*$  if  $q_n = 1$  (Epidemic)  
 $q_n = 0$  (Died away)



Let me write it down. So,  $q_n$  means the probability that disease reaches level  $n$  or I can also say that it is the probability that my disease persists still level  $n$ , so far so good. Now let me look at what does this value represent. I take here  $q^*$  and I define  $q^*$  as  $q_n$  as  $n \rightarrow \infty$ . What does it mean? It means that I keep going down, down, down, down, down and I go to the infinite level and I see there what is the probability that this disease is still persisting at that infinite level, that is my  $q^*$ .

And you see what if, what does it mean if  $q^*$  is 1? It means that is even if I go to the infinite level my disease is existing, it means that my disease is epidemic. If my  $q^*$  is 1, I say that my disease is epidemic and if my  $q^*$  is 0, I say that my disease has died away ok. So, now what do we have to prove? We have to prove that if our basic reproductive number  $R > 1$ .

(Refer Slide Time: 05:23)



So, what do we have to prove? We have to prove that if our basic reproductive number  $R_0$ , if it is less than 1, then we say that the disease for sure dies away, which means that then  $q^*$  should be 0 right. What is  $q^*$ ? It is 0 when the disease dies away,  $q^*$  is 0, when the disease dies away. So, when  $R_0 < 1$ , the probability that your disease exists till your infinite level should be equal to 0; that is your  $q^*$  should be equal to 0 and second thing when your  $R_0 > 1$ , then  $q^*$  should be greater than 0 right.

So,  $q^*$  if it is greater than 0, it means that if the probability that your disease persists when you go to the infinite level in your network that probability is something positive. So, this is how we write our problem statement rigorously. It is essentially the same thing which we stated before that if  $R_0 < 1$ , then our disease dies away for sure that is  $q^*$  equals to 0. If  $R_0 > 1$ , then with a positive probability our disease it persists in the network. So, we are going to do a proof which is a little bit lengthy.

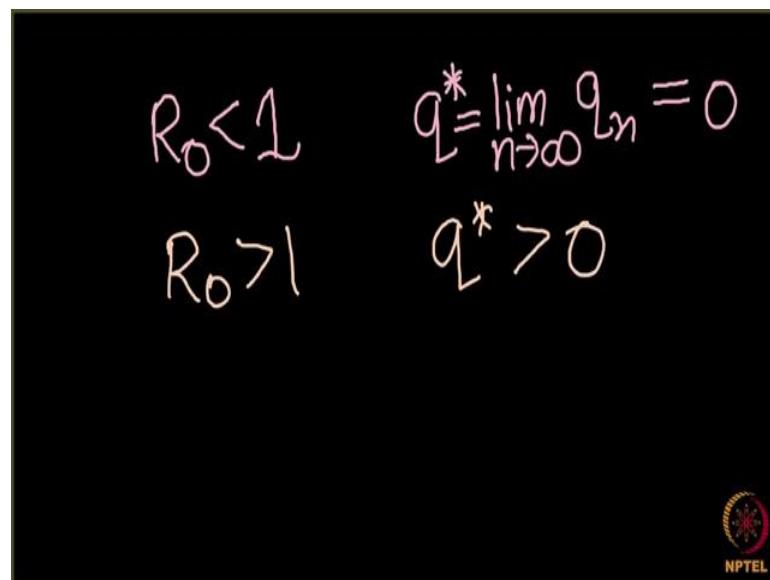
So, at times you may want to look back and forth and see what is happening, although I try my best to keep it as clear as I can ok. This is what we have to prove. And we will continue in the next lecture. So, just look back and see how we have modeled this problem, what is  $q^*$ , what is  $q_n$  and so on. And, we will continue in the next lecture.

**Social Network**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Rich Get Richer Phenomenon - 2**  
**Lecture - 139**  
**Analyzing basic reproductive number 2**

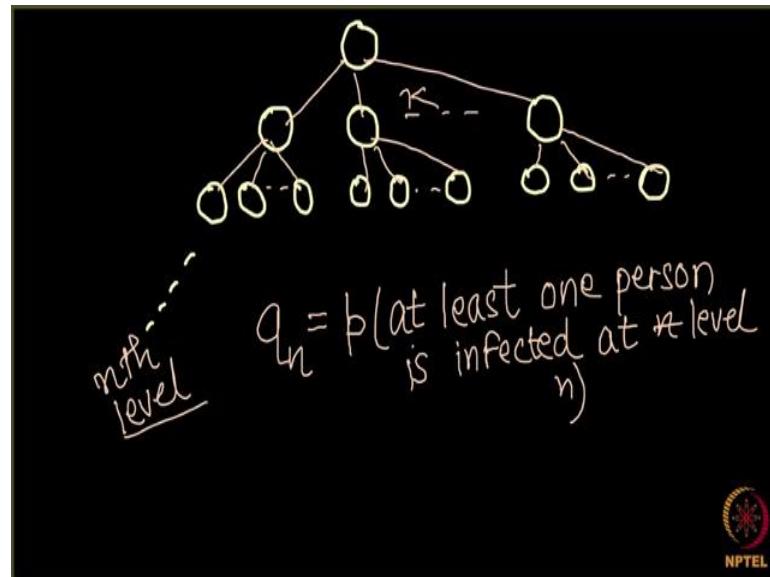
So, we have formulated the problem statement. I will again revise it.

(Refer Slide Time: 00:20)


$$R_0 < 1 \quad q^* = \lim_{n \rightarrow \infty} q_n = 0$$
$$R_0 > 1 \quad q^* > 0$$

I keep revising things again and again so that we should not get confused whether we are in proof. At problem statement was simply we have to prove that if the value of  $R_0$  basic reproductive number is less than 1, then our  $q^*$ ,  $q^*$  which is nothing but  $\lim_{n \rightarrow \infty} q_n = 0$  and if our  $R_0$  it is greater than 1 then the same  $q^* > 0$ . And you remember what is  $q_n$ ; let us quickly see what is  $q_n$ .

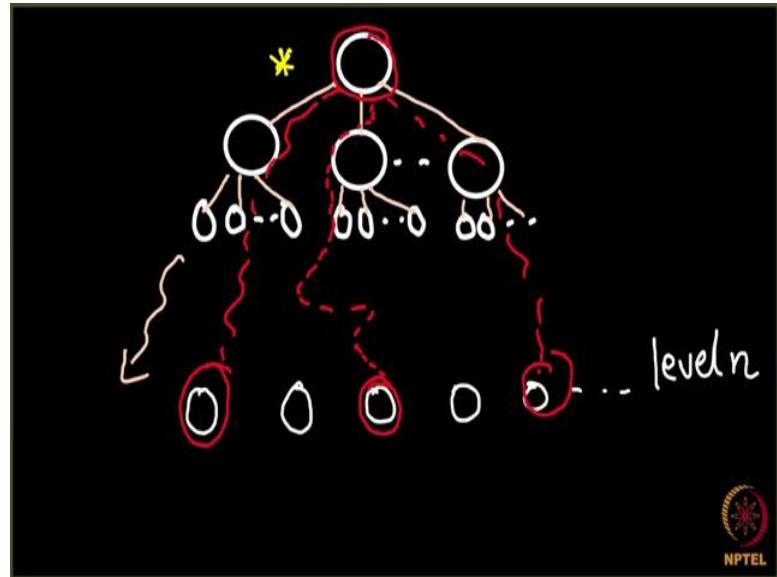
(Refer Slide Time: 00:58)



So, if I have a network over here let us say I have a network here, this node has  $k$  neighbors and each of these again have some  $k$  neighbors and so on. And it we make which is here and these are the edges  $k$  neighbors here. And this has  $k$  neighbors here,  $k$  neighbors here  $k$  neighbors here and so on.

And we got up to the  $n$ th level  $n$ th level. So, what was  $q_n$ ?  $q_n$  was the probability that at least at least one person is infected at the  $n$ th level. One person is infected at level  $n$ . Let us now see how we can find out this value of  $q_n$ . So, I make a clearer network over here; so, I need some time.

(Refer Slide Time: 02:02)



And here I can be a little big and here are the edges fine. This is my network right and somewhere down is our nth level. So, let us say this is our level n ok. Now what do we do? I define an event here. Now carefully watch, I define an event here and the event I say that it is a star event, star event.

What do I mean by a star event? We know that one of the node here should be infected right and for this node here to be infected there should be a path right, there should be a path from this node which was initially infected. So, this node here is obviously, infected. For some of the nodes level end to be infected there should be a path from this node to this node to somewhere here to somewhere here some path should be here right.

So, now what does even star means that that path can come from here, that path can come from here, and in fact some node here or that path can come from here, and in fact some node here. But there is at least some path and this path should run through should go through at least one of these edges either through this either through this or either through this right. So, my star event says that it is the event that your infection reaches this level n through a path which involves this edge.

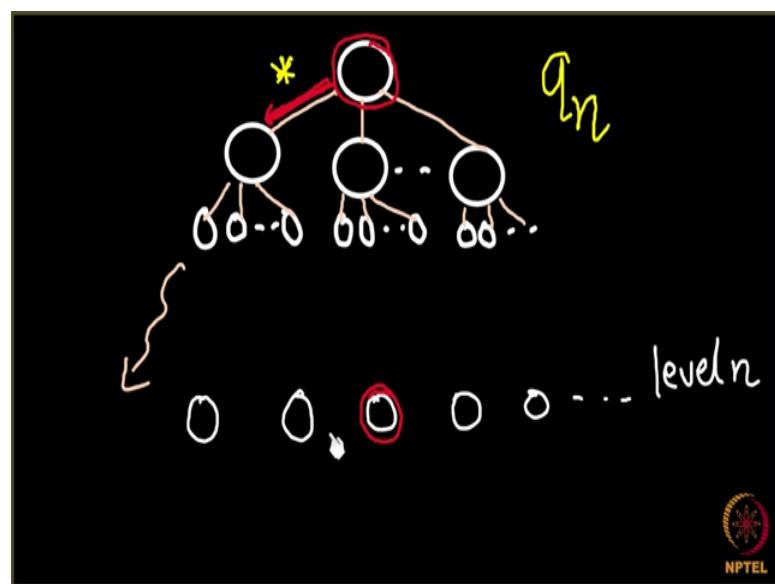
So, infection goes from this edge after this edge I do not know what happens, but it reaches level n, but this edge is for sure is infected. So, star event is the event that your infection reaches level n involving this edge and something happens after this edge any path. So, after this edge infection can go through this edge and have some path, can go through this

edge and have some path can go through this edge and have some path, but for sure it comes from here.

That is my event star ok. What do you think is the probability of this event star? What is the probability that your infection reaches level  $n$  through this path, through some of the path here? It is easy how will infection reach here. So, we know that first for infection to reach here may take back all my paths ok.

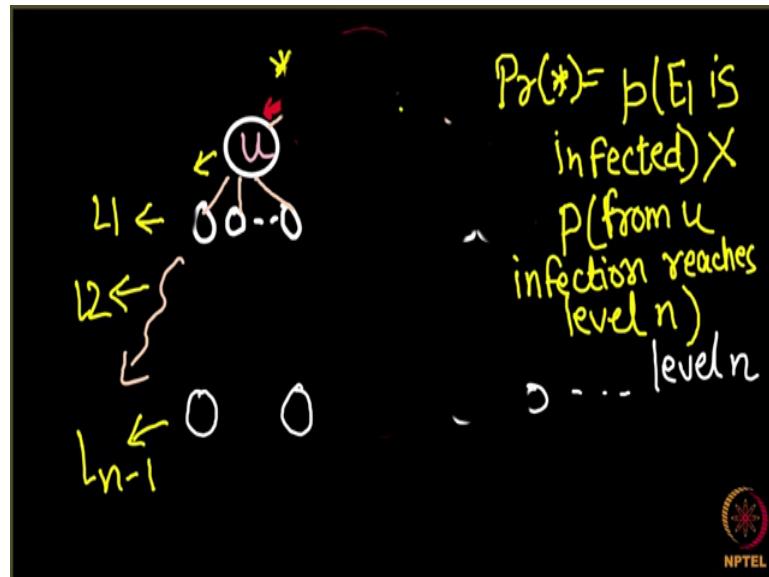
So, for my infection to come through this path first of all this thing should be infected right. This edge should be infected and then something happens afterwards.

(Refer Slide Time: 05:28)



Now, see what was  $q_n$ ?  $q_n$  was the probability that starting from here your infection reaches this level right that is  $q_n$ . You start from this level from this node and your infection reaches this right ok.

(Refer Slide Time: 05:52)



Now, tell me can I write the probability of event star as probability that let us call this edges E1 and probability that E1 is infected. The infection passes through E1 multiplied by, what should come after multiplication? After multiplication is the probability that and let us say that this node over here is u. So, into probability that from u infection reaches level n.

Probability that E1 is infected into probability that from u infection reaches level n. What is this probability that from u infection reaches level n, do you see? If you start from u it is the same as if you as if you are starting from here, just one level has reduced. What is what was  $q_n$ ? You start from here and your infection should reach nth level. What is  $q_{n-1}$ ? That you start from here and your infection reaches level n - 1.

So, if you look at this node u over here, so assume that this node is not here, all these nodes are not here, I will even remove this, we will take them back wait; we will just rub all these portion. Let us say all this is not here all this is not here; I forget about all these and what I want to know is that you start from u and infection reaches level n.

So, this was here level n, what is this level for u? So, if I start from u, so for u this is level 1, next will be level 2 and this will be level n - 1. So, this thing, probability that your infection starts from u and reaches level n is nothing, but  $q_{n-1}$  ok.

So, I will just take everything back right. So, how can I write my probability of event star, what is the probability that E1 is infected? The probability that E1 is infected let us take a new slide and write it there, I will quickly draw all these things here.

(Refer Slide Time: 08:59)

$$P_{\text{star}}(*) = P(E_1 \text{ gets infected})$$

$$\times q_{n-1}$$

$$P_{\text{star}}(*) = pq_{n-1}$$

$$q_n = 1 - P_{\text{star}}(\text{All } *) \text{ events fail}$$

$$= 1 - (1 - pq_{n-1}) \times (1 - pq_{n-1}) \cdots [k \text{ times}]$$

$$q_n = 1 - (1 - pq_{n-1})^k$$

NPTEL

So, we have this node and it has some  $k$  neighbors which has some  $k$  neighbors again and so on. And here is your level  $n$  and then recall what was your event star? Event star was you reach level  $n$  through this path and we said that the probability of event star equals to probability that the edge  $E_1$  that the edge  $E_1$ . So, if this is  $E_1$  probability that  $E_1$  gets infected multiplied by  $q_{n-1}$  right. And what is the probability that  $E_1$  gets infected is nothing but  $p$ .

So, it is  $pq_{n-1}$  is the probability of event star so far so good. Now what were we looking at? We start our infection from here and it should reach level  $n$ . For reaching level  $n$  how many star events are possible, it can go from here, it can go from here, it can go from here. So, there is a star event here, there is a star event here and there is a star event here.

So, there are  $k$  star events right and your infection can go through the happening of this star event or the happening of this star event or the happening this star event. If one of these at least one, so even if one of this star event succeed your infection will reach level  $n$  right because, what is this star event that your infection reaches level  $n$  through this path. So, for your infection node to reach level  $n$  all the star event should fail right. So, I can write  $q_n$ .  $q_n$  was my probability that my infection reaches level  $n$ .

And for reaching level  $n$  it must go through one of the star events. So, can I write  $q_n$  as 1 minus probability that all-star events fail?  $q_n$  is nothing, but the probability that all the star events fail ok. Now what is the probability that all the star events fail; 1 minus. What is the probability that the first star event fail? So, probability that first star event succeeds is this, probability that first star events will be  $E_1$  minus this, which is  $1 - (1 - pq_{n-1})$ .

What is the probability that second star event fail all of them should fail? So, I put a multiplication here and what is the probability that second star event fail it is a same. So, it is  $1 - (1 - pq_{n-1})$  and I keep doing it how many times? How many star events are there that many times, so it is  $k$  times right?

So, I can write  $q_n$  as  $1 - (1 - pq_{n-1})^k$  ok. So, I will quickly put back everything in place whatever we have done till now, we will do it at the end of every small lecture which is this proof is running So, let us quickly see what we have done.

(Refer Slide Time: 12:50)

$$\begin{aligned} \text{If } R_0 < 1 \quad q^* &= 0 \\ R_0 > 1 \quad q^* &> 0 \\ q_n &= 1 - (1 - p q_{n-1})^k \\ q^* &= \lim_{n \rightarrow \infty} q_n \end{aligned}$$

So, first what was our problem statement? Our problem statement was to prove that if  $R_0 < 1$  then  $q^* = 0$ , if  $R_0 > 1$  then  $q^* > 0$ , this was our problem statement.

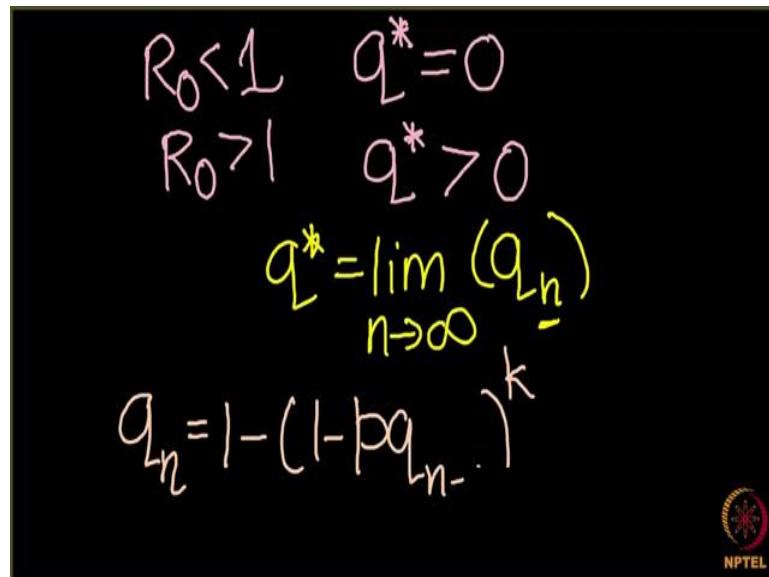
Next in this lecture what we did? In this lecture we derived a formula for  $q_n$  and the formula is  $1 - (1 - pq_{n-1})^k$ . And we also know what is  $q^*$  from our first lecture:  $q^* = \lim_{n \rightarrow \infty} q_n$ . So, we will start from here in our next lecture.

**Social Networks Rich**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Rich Get Richer Phenomenon – 2**  
**Lecture - 140**  
**Analyzing basic reproductive number – 3**

So, now in this lecture we are going to start from the same place where we left off in the last lecture. So, what did we see in the last lecture, first of all our problem statement.

(Refer Slide Time: 00:16)



$R_0 < 1 \quad q^* = 0$

$R_0 > 1 \quad q^* > 0$

$q^* = \lim_{n \rightarrow \infty} q_n$

$q_n = 1 - (1 - pq_{n-1})^k$

NPTEL

What is the problem statement? First, if  $R_0 < 1$  then we have to prove that the value of  $q^*$  equals to 0 and if  $R_0$  is greater than 1 then the value of  $q^*$  is something which is greater than 0. This thing we have to prove and then we know what is  $q^*$  right.

So, what is  $q^*$ ?  $q^*$  is nothing but  $\lim_{n \rightarrow \infty} q_n$  where  $q_n$  is the probability that your infection persist till the nth level; that is at least 1 person at the nth level is infected. And in the last lecture we have derived a formula for  $q_n$  and the formula was  $1 - (1 - pq_{n-1})^k$ . What we are going to do in this lecture is now we are going to analyze this formula further.

(Refer Slide Time: 01:16)

$$q_n = 1 - (1 - pq_{n-1})^k$$

$$q_0 = 1$$

$$q_1 = 1 - (1 - pq_0)^k$$

$$q_2 = 1 - (1 - pq_1)^k \quad P \rightarrow n \text{ level}$$

So, let me write this formula here and we will analyzing it  $q_n = 1 - (1 - pq_{n-1})^k$  ok. Now what is  $q_0$ ? Let us look at  $q_0, q_1, q_2, q_3$  and so on. What you think is  $q_0$ ? You will not get it from this formula. So,  $q_0$  is what, the probability that your infection persists till the 0th level and what is a probability.

So, what was a problem statement? This was a guy who was here having  $k$  neighbors and this person was again having  $k$  neighbors. This person was again having  $k$  neighbors and so on. And your infection started from here and the question was what is the probability that infection reaches here to the  $n$ th level was our question.

What is level 0 here? This is level 0 and what is the probability that at least 1 person at this level is infected? It is obviously, 1 right for sure 1 this guy must be infected here. If this guy was not infected here, there was no problem we would be solving right. So, entire problem is because this person at the 0th level is infected it is infected with the probability 1. So,  $q_0 = 1$ . What is  $q_1$ ? Finding  $q_1$  is easy. Put the value in this formula what is  $q_1, 1 - (1 - pq_0)^k$ , right.

What is  $q_2$ ?  $q_2$  is  $(1 - pq_1)^k$  and what is our aim? Our aim is to find  $q^*$  which will come after we will keep repeating this formula. So, we have  $q_0$ , from  $q_0$  we can find  $q_1$ , from  $q_1$  we can find  $q_2$  and then we have to do this process infinite number of times and finally, we will find the value of  $q^*$ . We can do it infinite number of times. So, let us see how do we find out this value.

(Refer Slide Time: 03:22)

$$q_0 = 1$$

$$q_1 = 1 - (1 - pq_0)^k$$

$$q_2 = 1 - (1 - pq_1)^k$$

$$\vdots$$

$$q_n = 1 - (1 - pq_{n-1})^k$$

$$q^* = f(f(f(\dots(1))))^{\infty \text{ times}}$$

$$y = f(x) = 1 - (1 - px)^k$$

$$q_1 = f(q_0)$$

$$q_2 = f(q_1) = f(f(q_0))$$

$$q_3 = f''(q_0) \quad \vdots \text{---} \infty \text{ times}$$

$$q^* = f(f(f(\dots(q_0)))$$



So, now we have all this we know  $q_0, q_1, q_2$  and we know that what is  $q_n, q_n$  is nothing, but  $(1 - pq_{n-1})^k$ . I try to write it down in the form of a function. It is already in form of a function. If I just take a function  $y = f(x)$  we define this function as  $1 - (1 - px)^k$ .

Now in terms of this function can we see what is  $q_1$ ,  $q_0$  obviously is 1, what is  $q_1$  according to this function? So, if we see what is  $q_1$  here  $(1 - pq_0)^k$ . This and this are the same. I have just written this in the form of a function here. So, we can write  $q_1$  as nothing, but 1 minus 1 minus oh sorry. So, we can simply write  $q_1$  as; what  $f(q_0)$  right. What is  $f(q_0)$ ?  $(1 - pq_0)^k$  which is same as this right. What is  $q_2$  now?  $q_2$  is  $f(q_1)$ ,  $(1 - pq_1)^k$  which is nothing, but I put the value of  $q_1$  here. It becomes  $f''$  means  $f$  of  $f$  of so, to  $f(f(f(q_0)))$ .

Similarly, what is  $q_3$ ?  $q_3$  is nothing, but  $f'''(q_0)$ . And similarly, you see what is  $q^*$  going to be?  $q^*$  is going to be  $f$  dash dash dash dash infinite times means  $f(f(f(f(\dots(q_0))))$ ). And we also know that the value of  $q_0$  is 1. So, I can write it down as  $q^* = f(f(f(f(\dots(1))))$  right.

(Refer Slide Time: 05:37)

$$y = f(x) = 1 - (1 - px)^k$$
$$q^* = \underbrace{f(f(f(\dots^{00\text{times}}(1))))}_{R_0}$$


So, what is our overall aim now? Our aim now is we have a function  $y = f(x)$  which we define which is  $1 - (1 - px)^k$  and I know now the value  $q^*$  which I have to find is nothing but  $f(f(f(f(\dots(1))))).$

And how do I find out this value? Once I find out this value my task is done, our aim is to find out the value of  $q^*$ , obviously given the value of  $R_0$  that is to come in the next lecture. Our overall aim is just to find this value of  $q^*$  and we will be finding it in next lectures. Probably just 1 or 2 lecture more we will be finding what is  $q^*$ .

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Rich Get Richer Phenomenon - 2**  
**Lecture - 141**  
**Analyzing basic reproductive number – 4**

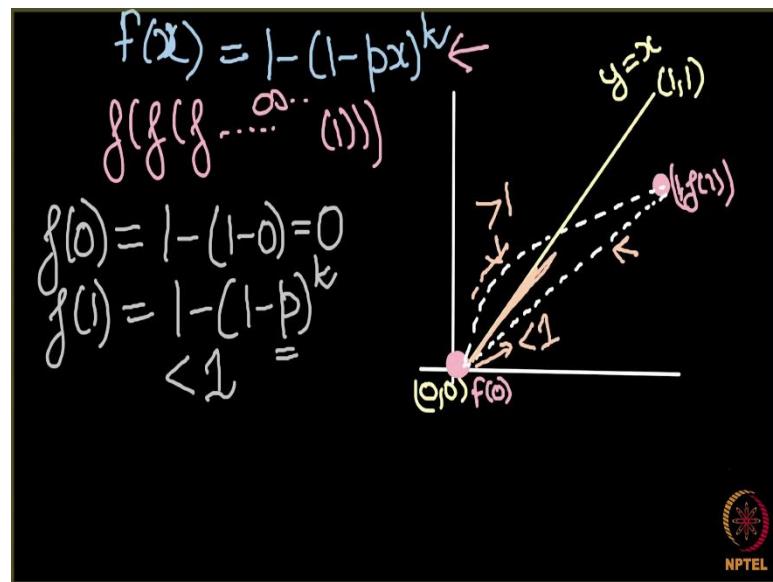
(Refer Slide Time: 00:11)

$$\begin{aligned}
 R_0 < 1 \rightarrow q^* = 0 \\
 R_0 > 1 \rightarrow q^* > 0 \\
 q_n = 1 - (1 - p q_{n-1})^k \\
 f(x) = 1 - (1 - px)^k \\
 q^* = \underline{f(f(f(\dots(1))))}^{\text{infinitely}}
 \end{aligned}$$



So, where were we? We will quickly recap from the beginning; our aim to show that if  $R_0 < 1$   $q^* = 0$ . And, to show that if  $R_0 > 1$  then  $q^* > 0$ , this was our aim. And, then we found out a formula for  $q_n = 1 - (1 - p q_{n-1})^k$ , but our aim is to find  $q^*$  right. And, we have defined the function, we have looked at a function and the function were  $f(x) = 1 - (1 - px)^k$ . And, we have looked at that this  $q^*$  is nothing, but  $f(f(f(f(\dots(1)))))$  and this is the value which we are after. We need to find out what is this value, let us see how we find this value.

(Refer Slide Time: 01:12)



So, what all we are having  $f(x) = 1 - (1 - px)^k$ , this is our  $f(x)$  and our aim is to find  $f(f(f \dots (1)))$ . How do we find it? First, let us analyze what is this function, how does this function look. Let us see how this function look will. So, let us say, let us try to find  $f(0)$ . What will  $f(0)$  be? It will be  $(1 - (1 - 0))$  right which will be nothing, but 0. So,  $f(0)$  is 0 and let us look at what is  $f(1)$ ;  $f(1) = 1 - (1 - p)^k$  and we know that  $p$  is something positive.

So, this entire value is going to be less than 1. So,  $f(1)$  is something which is less than 1  $f(0)$  is 0 and  $f(1)$  is something which is less than 1; let us try to see how this plot will look like. So, I draw a plot here let us say y axis and x axis and here it is 0, here it  $(0, 0)$  and I first of all I draw a line also here and this sorry and this is a line I say what is this line you all know. So, this line is  $y$  equals to  $x$  and this point is 1 comma 1 right. And, let us see how a function  $f(0)$  look will like, a function  $f(0)$  is first of all having a point here for sure because at 0  $f(0)$  is 0.

And we know that  $f(1)$  is something less than 1. So, this point is  $f(0)$  and  $f(1)$  is going to lie somewhere here which is less than 1. So, this point is  $f(1)$ ,  $(1, f(1))$ . So, this point is  $(1, f(1))$  ok. We have seen the starting point of this function and the ending point. What happens during middle? What happens here? How does this function look like? And, that is also easy to calculate, let us see how we can calculate that. Let us try to find

out the slope of this function at this point, let us try to find out the slope of this function at this point.

So, what is this slope? This slope we know is 1 right, this slope is 1 which is your 45 degree; tan of 45 degree 1. So, if at this point so, now see this function  $f(x)$  can either go like this or go like this. When does this function go like this? When this slope of this function here is less than 1 right, if the slope is so, because this is slope equal to 1; so, if this is slope equal to 1, if slope less than 1 then this function will go from downwards of this function  $y = x$ . And, when will this function go like? This it will go like this when the slope of this function is greater than 1. So, let us first try to find out the slope of this function. So, what do we do to find out the slope of the function is to differentiate?

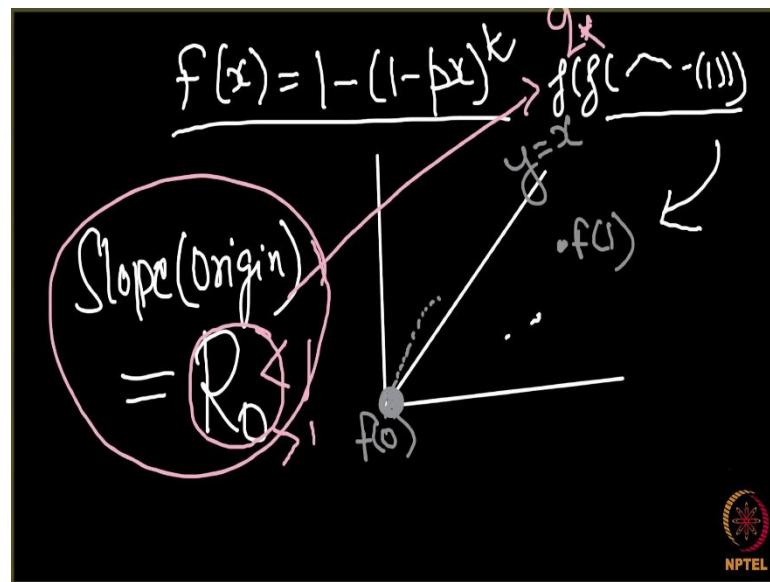
(Refer Slide Time: 04:53)

$$\begin{aligned}
 f(x) &= 1 - (1 - px)^k \\
 f'(x) &= -k(1 - px)^{k-1}(-p) \\
 &= pk(1 - px)^{k-1} \\
 \cancel{x \neq 0} \quad f'(0) &= pk(1 - 0)^{k-1} \\
 &= \cancel{pk} \\
 R_0 \leftarrow &
 \end{aligned}$$


So, we have the function  $f(x) = 1 - (1 - px)^k$ , let us differentiate it. So,  $f'(x) = -k(1 - px)^{k-1}$  right multiplied by differentiation of  $(1 - px)$  which is  $-p$ . So, this becomes equal to  $pk(1 - px)^{k-1}$  right. So, this is the slope of this function and we are interested to find the slope of this function at  $x = 0$  at origin.

What is  $f'(0)$ ?  $f'(0) = pk(1 - p * 0)^{k-1}$  which is nothing, but  $p * k$ ; it is amazing. The slope of this function is  $p * k$ . What is  $p * k$ ? If you remember  $p * k$  is nothing, but your basic reproductive number  $R_0$ , from where we started and where we reached.

(Refer Slide Time: 06:09)



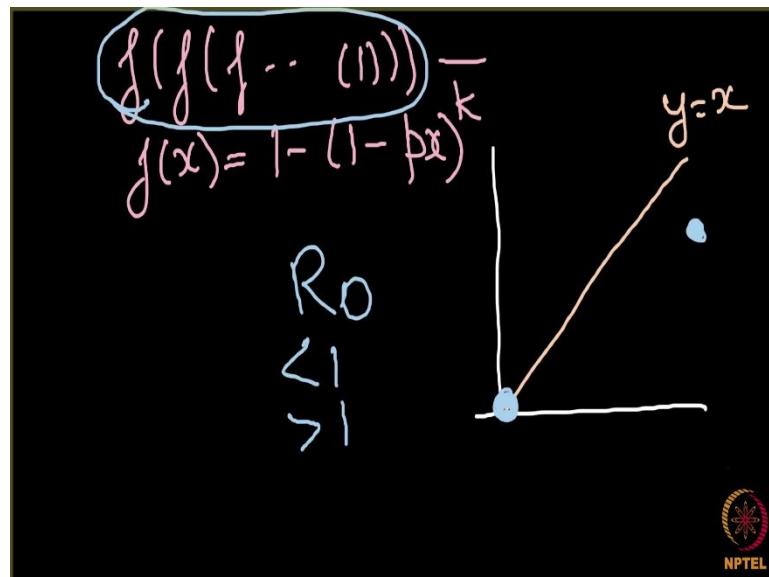
So, you see we started off with this function  $f(x) = 1 - (1 - px)^k$  and then we try to plot this function. And, this is the line  $y$ , and this is the line  $y$  equals to (Refer Time: 06:34) 1 second and this is the line  $y = x$ . This is the line  $y = x$  and then we saw that here will be our point  $f(0)$  and here will be our point  $f(1)$ . And, then we were interested in finding the slope of this function at this point and it turned out that this slope of this function at origin is nothing, but the value of the basic reproductive number.

So, just remember all these we have this  $f(x)$  here and the function look something like this and we wanted to find out  $f(f(f(f(\dots(1))))$  and then we tried to look at this function. The function turned out something like this and then we saw something amazing; we saw that the slope of this function at origin is nothing, but  $R_0$ . So, now we will see for different values of  $R_0$ , that is  $R_0$  can be less than 1 or greater than 1 how does this value which is nothing, but  $q^*$  turn out to be and that will be the end of this proof which will which will do in the coming up lecture.

**Social Network**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Rich Get Richer Phenomenon – 2**  
**Lecture - 142**  
**Analyzing basic reproductive number – 5**

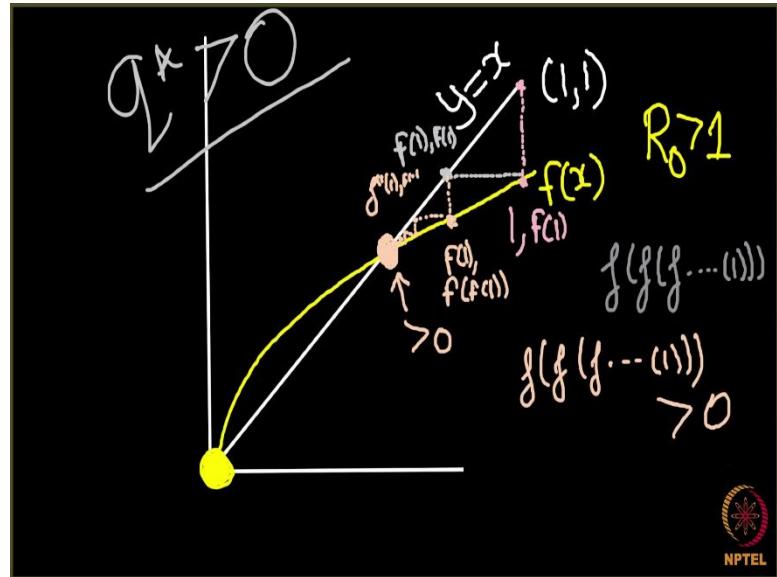
(Refer Slide Time: 00:17)



Now, coming to the final part of this theorem proving; what we were doing, our aim was to find  $f(f(f(\dots(1))))$ , where  $f(x)$  was a function which was  $(1 - (1 - px)^k)$  and we have looked at this function. So, I will just roughly make it here. So, we have looked at this function, if these are our axis and let us say this is not this let us say, this is our line  $y = x$  we saw that the first point of this function was here at origin.

And, second point was something which is less than 1 and then when we saw that the slope of this function at this origin is nothing, but a ray basic reproductive number  $R_0$ . Now, let us see how we are going to find out this value of  $q^*$  which is  $f(f(f(\dots(1))))$ , when our  $R_0 < 1$  and when our  $R_0 > 1$  ok.

(Refer Slide Time: 01:33)



So, let me draw the graph neatly here. This is the graphs and say this is the line, which is  $y$ , this is the line  $y = x$  and let us say this is the point. What will this point be?  $(1, 1)$  and for our function we know that 1 point is here and then 1 point is somewhere here, which is less than 1. And, then let us look at the first case. Let us say the value of  $R_0 > 1$ , when the value of  $R_0 > 1$  we know that our function its slope is going to be greater than 1. So, this line  $y = x$  is corresponding to the slope equals to 1. So, slope greater than 1 will go something like this. So, I draw something like this.

So, our function look something like this ok. So, this is our function and I write that this function is my  $f(x)$  and you see what is my aim, my aim is to find  $f(f(f(\dots(1))))$ ; let us see how do we do it. So, I just make a little bit change to this figure ok, I want some shell points over here yes. This is the point  $(1, 1)$ , now see something interesting is going to happen. I know that this point over here is  $(1, 1)$  right, let us say I draw a line vertically here. I draw a line vertical here, what do you think is this point over here. So, you see the value of  $x$  axis remains the same which is 1 and what is the value of  $y$  axis over here, this is the function  $y$  equals to  $f(x)$ . So, the value of  $y$  over here is  $f(1)$  is not it. So, this is this point is  $(1, f(1))$  ok. Now, let me draw a line horizontally over here. When I draw a line horizontally over here, what do you think is this point over here.

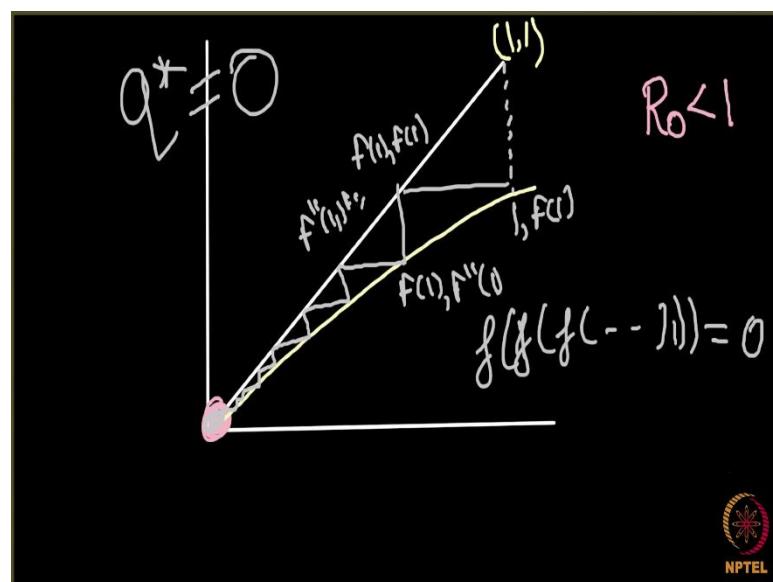
So, this point over here now, you see the value of  $x$  changes and what is the value of  $x$  becomes; this line is  $y$  equals to  $x$ . So, the value of  $x$  here should becomes  $x$  equal to the

value of  $y$  and what is the value of  $y$ ?  $f(1)$ . So, I get a point here  $f(1)$  comma  $f(1)$ . So, far so good and I think that you are now getting an idea what I am doing. We again draw a line vertically over here and what do you think is now this point over here, this point over here. Again, the value of  $x$  axis remains the same which was  $f(1)$  and what does the  $y$  axis becomes,  $y$  here is  $f(x)$ . So,  $y$  axis here is  $f(f(1))$ .

So, you see how we are moving from 1 to  $f(1)$  to  $f(f(1))$  and then I go horizontally over here, and it will be nothing, but  $f''(1), f''(1)$ . And, then I can come here, and it will be up triple dash of 1 and when I do keep doing it infinite times, what will happen, when will it converge. And, see what will happen at this point now, at this point both of these curves meet right. So, when I am going to do these process infinite times this process will converge here because, after this I cannot draw horizontally or vertically because, both of the curves have met. So, when I do these infinite times I converge at a point and you see what this point is.

This point is greater than 0, which means that your  $f(f(f(\dots(1)))$ ) turns out to be greater than 0 which means that the value of  $q^*$  in this case is greater than 0. So, we are done with the first part, what happens when  $R_0 > 1$ . What happens when  $R_0$  is less than 1, is entirely the same completely similar.

(Refer Slide Time: 06:26)



So, again here was our curve and here was the line let us say  $y = x$  and then 1 point of our plot was here and another point of our plot was here. And, we know here  $R_0 < 1$ , if  $R_0 <$

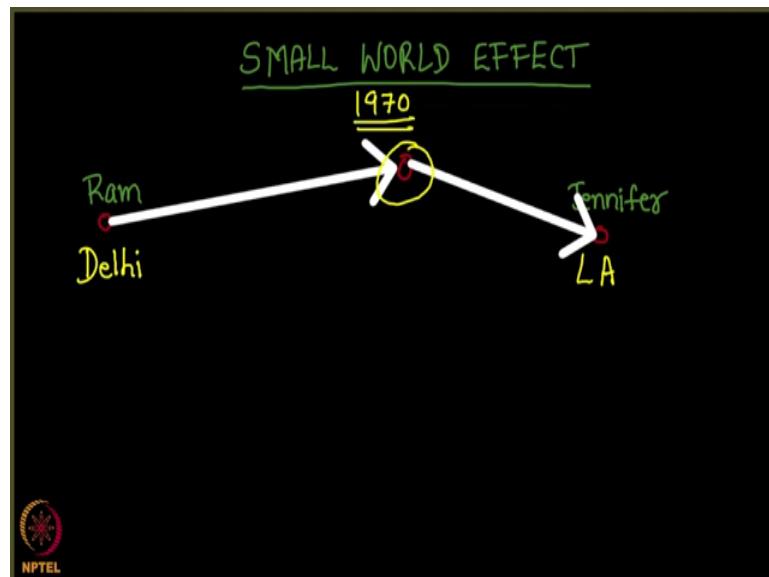
1 how will my curve look like. It will go something like this, right something like this. And now I can repeat the same procedure. So, we can repeat the same procedure.

This point here is  $(1, 1)$  right, now again drop this here and I get, do I get  $(1, f(1))$  and then I go horizontally here, and I get  $(f(1), f(1))$ . And, then I come here, and I get  $(f(1), f''(1))$  and then I go here I get  $(f''(1), f''(1))$  and I keep doing. So, where is this curve going to converge is at 0. So, when you do it infinite times the curve converges at 0. So, in this case  $f(f(f(f(f(1))))))$  turns out to be 0, which means that in this case  $q^* = 0$ .

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**The Small World Effect**  
**Lecture - 143**  
**Small World Effect – An Introduction**

(Refer Slide Time: 00:07)

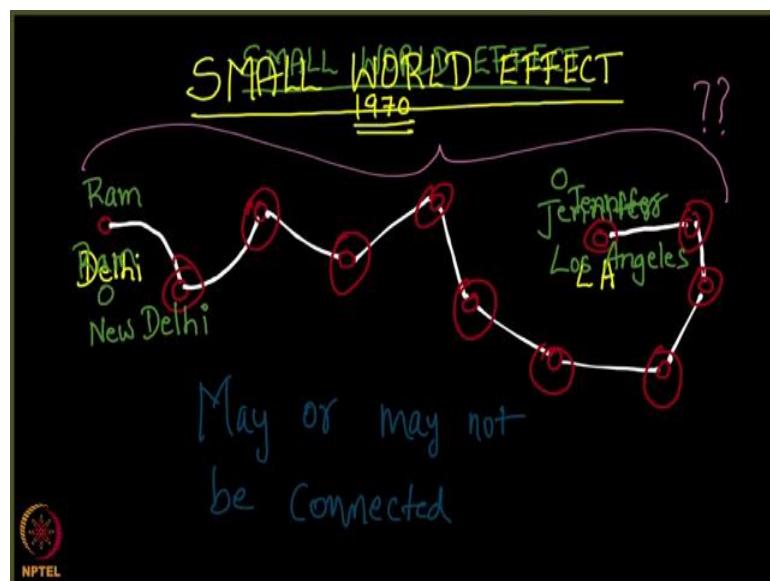


Let us start with the brand-new chapter by name the Small World Effect, you would have heard of this or experienced this more than many times in your life although it comes quite counterintuitive a fact. It is indeed quite experienced by many of us, you will indeed you will in fact, realise that all that I am going to talk right now in this chapter is nothing, but obvious. But then the looking at the science part of it is actually less obvious although, it is obvious in the sense that it is intuited.

So, let us start with a nice question as and always; assume you are let us say in India in some part of let us say Delhi and you have this question let us say your name is Ram R a m Ram and the question is there is someone at Los Angeles by name Jennifer, whom you would like to contact right. So, how will you contact this person called Jennifer? Assume, it was back in let us say in 1970 where you would not make use of something called the internet, you would know nothing about Jennifer who is at Los Angeles.

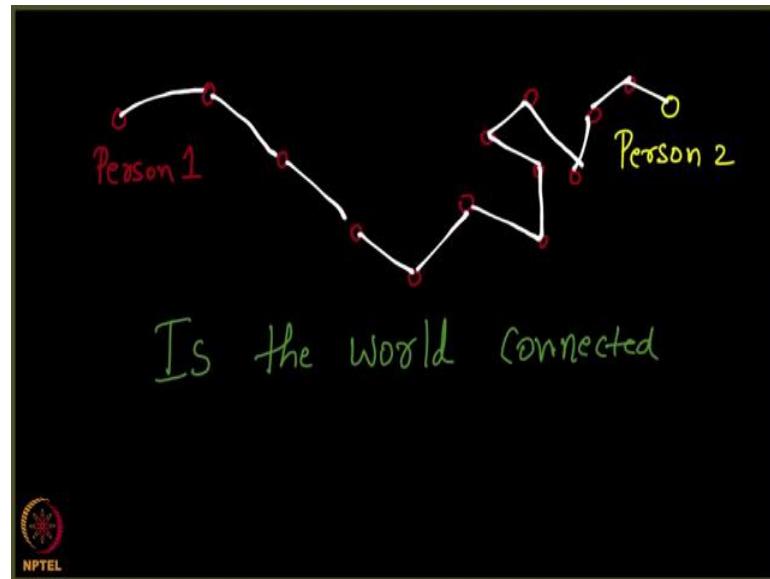
So, what you do you badly want to contact Jennifer for some business requirement and Jennifer cannot be contacted directly. So, what you will do is you will try to contact Jennifer through someone ok, you will find someone who possibly might know Jennifer. And, you will request this someone to put in touch with you, put in touch Jennifer with you right. Now, how feasible is this? It all boils down to you knowing someone who knows someone who knows Jennifer; this may or may not be the case, but surely something in me tells me that there possibly is someone.

(Refer Slide Time: 02:40)



Let us say there is someone that Ram knows and that someone might know someone else and that someone might know someone else and someone else so on and so forth. A long chain like this might know Jennifer. Let me put the ovals here which represents the nodes which represent people here. So, the idea is Ram might know someone who knows someone who knows someone who knows someone and so on and finally Jennifer. Now, this sounds a little funny to me, it is funny that these 2 people are indeed connected which 2 people Ram and Jennifer, are they even connected is my big question. They may or may not be connected right, they may or may not be connected; may or may not be connected. So, now let us try doing this experiment and see whether the world is connected or not. What do I mean by that?

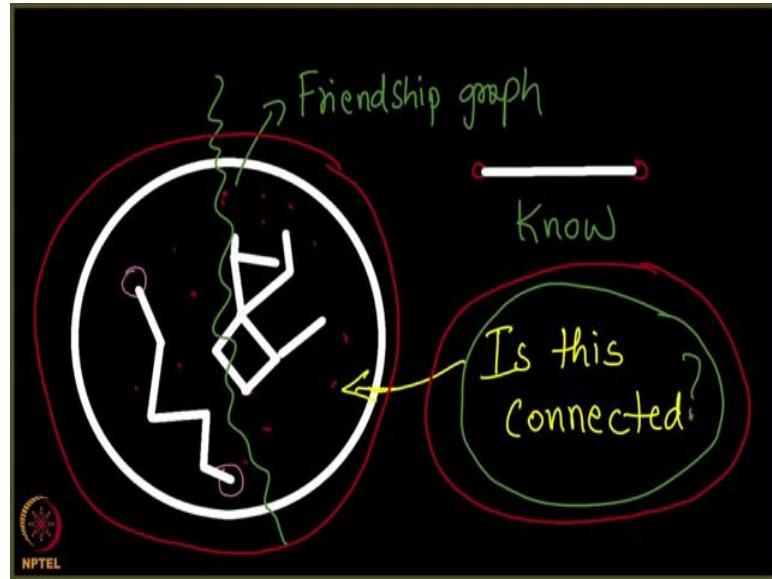
(Refer Slide Time: 04:09)



By that I mean in case I take any 2 random people in the world, assuming that I take 2 people someone from here someone from here, some place in the world. Some place of some rather a person some person 1 from this part of the world and some other person. Let me let me use some other colour here some other person 2 in some other part of the world. So, let us ask this question do you think this person knows someone, who knows someone who knows someone so on and so forth ok.

Who knows person 2, is this true that let me just connect these dots? Do you think the world is connected; is the world connected like this? What I am trying to ask. In other words, denoted in terms of graph theory, all I am asking is if I were to take the entire world right.

(Refer Slide Time: 05:11)

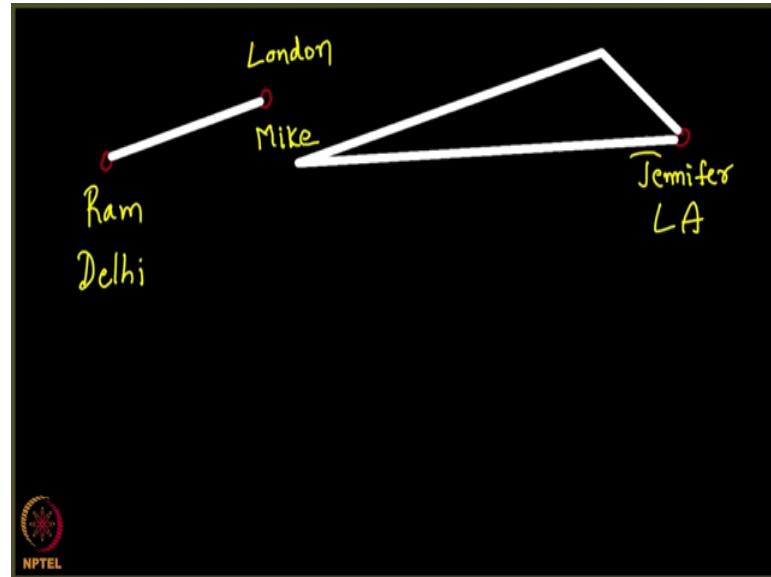


And if I were to consider all possible people over here and where between any 2 people let us say person 1 and person 2, you put an edge; you put an edge. If this person if they know each other if they know each other. Now, if you keep putting such edges between people here alright so on and so forth; so, on and so forth alright.

What kind of a network will you get? Is this network connected? If this is connected then between any 2 people that you might possibly pick there exists a path maybe right, there exists a path like this. But, in case it is disconnected there may not be a path between any 2 vertices. Now, the question that I am trying to ask for fun say is the graph, is the friendship graph of the world connected right. This is the friendship graph of the world friendship graph is this connected.

Now, of what use this such a question right in the first place; why anyone would bother to ask or answer such a question. So, let us get back to our question of Jennifer you would trying to know someone some person 1.

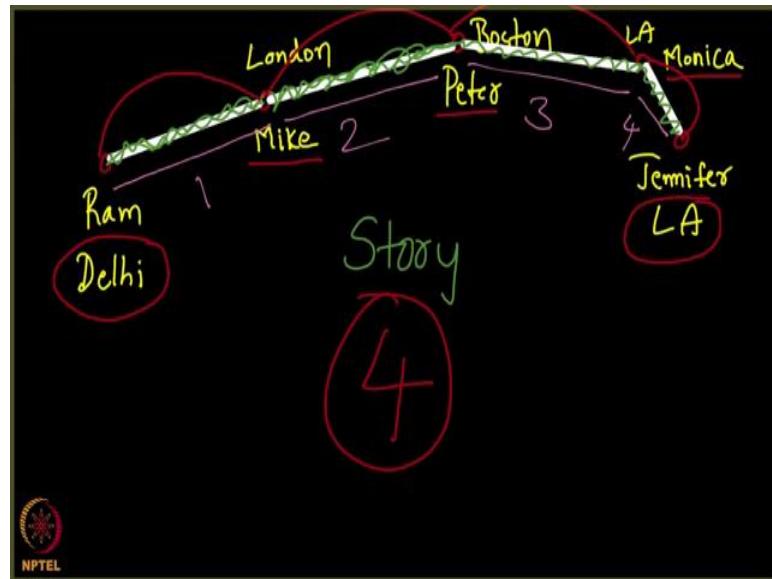
(Refer Slide Time: 06:58)



Let us say I will call him Ram here in from New Delhi trying to contact someone by name Jennifer in Los Angeles right. So, how would he even try contacting someone who might know Jennifer, what Ram will do is Ram is from Delhi he has a friend in let us say London ok. So, this friend of his is in London someone by name Mo someone by name Mike is in London.

And Ram knows him first-hand and he request Mike, Mike can you please put me to someone called Jennifer from Los Angeles. Now, here is a point to think. Ram thinks Mike might know someone who knows Jennifer, it is too much of a coincidence if Mike knows Jennifer first-hand right that that may not really be possible. But Ram still wants to explore the possibility of contacting Mike and requesting him to get him in touch with Jennifer. He knows for sure that Mike may not know Jennifer right.

(Refer Slide Time: 08:35)



But still he would he would you like to give it a try and ask Mike, Mike do you know someone in Los Angeles who might know Jennifer and then Mike says well dude I do not know anyone in of course, Ram knows Mike and asks Mike do you know someone in Los Angeles by name Jennifer. And, Mike says I do not know anybody in Los Angeles, but I know someone in Boston ok. I know someone in Boston ok, this place is Boston by name by name Peter ok.

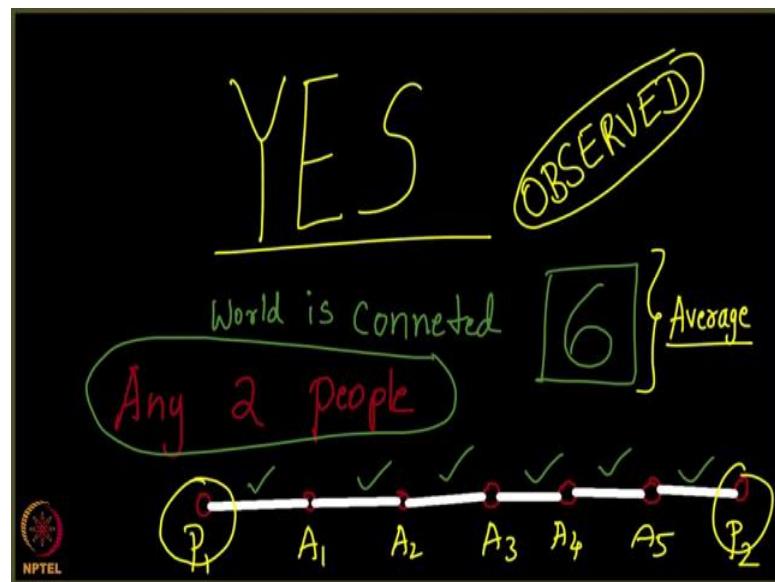
Someone called Peter alright and he says this Peter is actually from Los Angeles although he stays in Boston he might know Jennifer ok. And, Mike does in fact, contact Peter. So, initially Ram contacts Mike and Mike contacts Peter and asks Peter could you please help me get in touch with someone called Jennifer in Los Angeles who works for so and so company. And, Peter says although I am from Los Angeles I do not know anyone by name Jennifer in so and so area, but I know someone in that area who might know this person called Jennifer and that person's name is Monica and she is actually from Los Angeles ok.

And, Peter contacts Monica and Monica it so, happens that Monica indeed stays very close to Jennifer's place and she happens to be Jennifer's friend. And, Monica says yeah I do know Jennifer and this is how the connection from Jennifer to Monica to Peter to Mike to Ram has ok. It is surprising that is the connectivity only 1 from here to here, from here to here was 2, from here to here was 3, from here to here was 4. I told you a

story, this is actually a story this may not be true, a story of someone by name Ram in Delhi who wants to contact someone by name Jennifer in Los Angeles and gets through by asking a friend called Mike who asks a friend called Peter, who asks a friend called Monica. Right, is this always true that we can find anyone whom we want in a few hops like this, let us say here is 4 hops.

By hops I mean 1 friend and then second friend, third friend and then the fourth friend; is it always true that we can find someone through this quick single digit hops in this case it was 4. Is it always a small number as you know in the previous slide, we indeed asked this question? Is it at all connected, if it is connected, we can ask for distance between 2 people it is may not be connected? Now, here goes the big question the friendship network that we were discussing is it connected, if it is connected what the distance between 2 people on an average is. So, this is the big question. The answer for this is a big yes.

(Refer Slide Time: 11:49)



Yes, we are indeed connected, yes, we are what is connected the world is connected the world is connected; connected in what sense? Connected in a very surprising sense, that any 2 people of your choice if you take in the world. Let us say a person 1 and person 2 it is known that these 2 people are separated by 1 2 3 4 let me push this fellow this side 5 and 6 a in just 6 hops. You know which means a person here let me call him  $P_1$  person here  $P_1$  knows this person  $P_2$  through intermediate people let us say  $A_1 A_2 A_3 A_4 A_5$  and

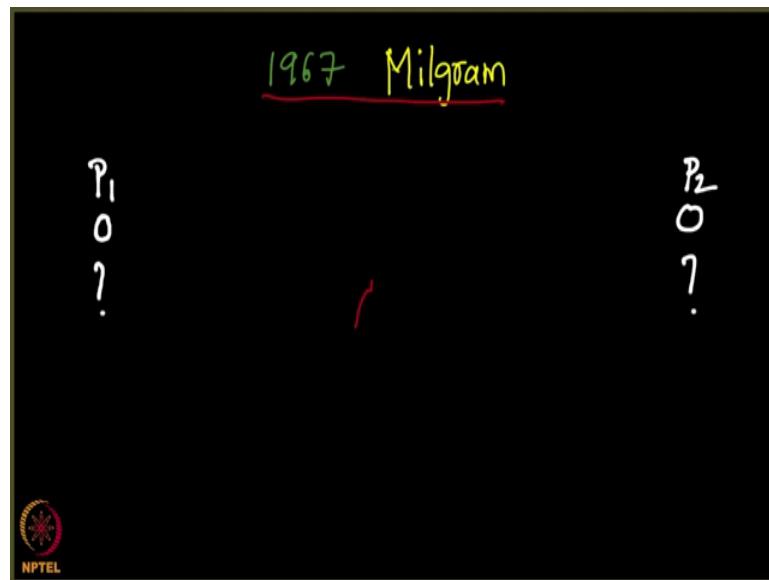
the sixth person is the person you are you are. So, 1 this is a 1 hop, this is 1 hop, this is second hop, and this is the third hop, fourth hop, fifth hop, sixth hop. So, 1 2 3 4 5 6, in 6 connections any 2 people in the world are connected. So, this is not actually exactly 6 by 6, I mean on an average it is 6 between any 2 people and this is what scientist have observed, note this they have observed.

Now, this is observed word that I said is slightly absurd you see. How can anyone even conduct such an experiment and conclude that there are 6 intermediary people between any 2 people on an average; does not sound true you see even if it is true how would you go about checking it correct. So, if I say the average number of friends per person in the world is let us say 200 ok, it could be a ridiculous statement just like that made up right. How would you substantially validate it correct. Similarly, if I say any 2 people in the world are connected by near 6 hops on an average so, this is average; how will anybody validate such a claim. So, we will see in our next lecture how exactly the scientist validated this claim and why it is actually true.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

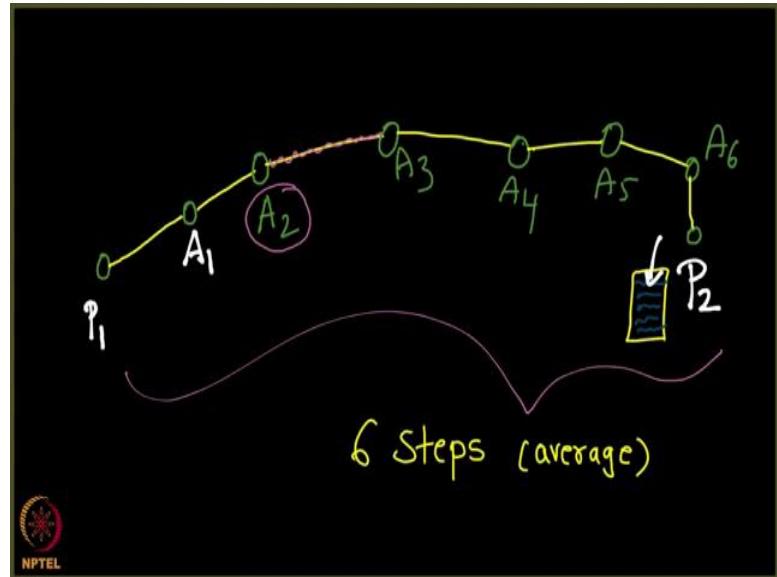
**The Small World Effect**  
**Lecture - 144**  
**Milgram's Experiments**

(Refer Slide Time: 00:08)



So, back in 1967, a scientist by name Milgram took a bold step to try executing this experiment of finding out whether the friendship network in the world is connected at all. If you take two people let us say  $P_1$  and  $P_2$  in the world, are they connected through several people in between or is the world of friendship networks disconnected? So, he took a bold step and conducted a very neat experiment.

(Refer Slide Time: 00:52)



All that he did was he choose a person  $P_1$  and gave a letter to this person  $P_1$  had a letter which the which with this person  $P_1$  was supposed to pass this on to a person  $P_2$  in some other part of the world. And  $P_1$  and  $P_2$ ; obviously, did not know each other's. They were pick uniformly at random let us say.

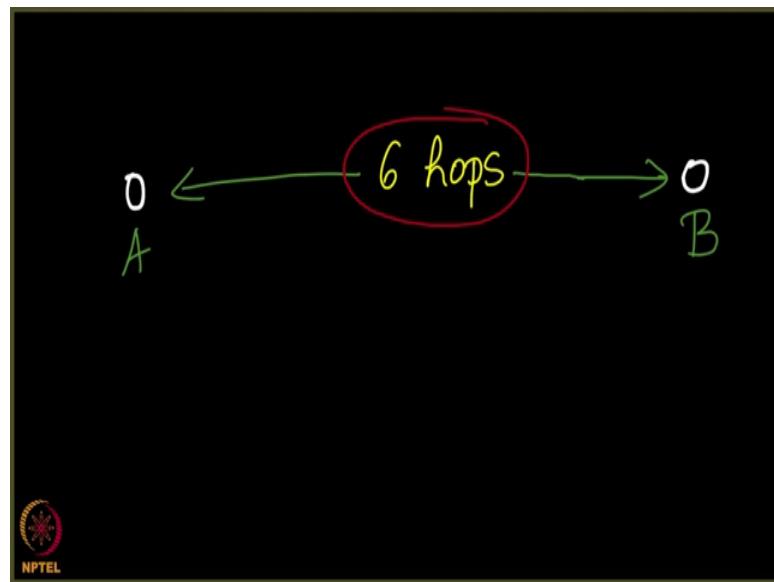
And  $P_1$  was supposed to deliver this letter to  $P_2$ ; now given that  $P_1$  did not know  $P_2$ . Now there is another rule here the address of  $P_2$  is specified in the letter, but  $P_1$  cannot post it to  $P_2$  directly. He cannot post it to be directly, instead he is asked to pass it on to someone whom he might know who might in tern know  $P_2$  or he may not know  $P_2$ . All that  $P_1$  should do is pass it on to some other persons let us say  $A_1$  whom  $P_1$  might know, whom  $P_1$  suspects that this  $A_1$  might know  $P_2$  and this  $A_1$  should then pass it on to  $P_2$ . Again, the same story, he may not know  $P_2$ . So, what she does is he chooses someone. This particular  $A_1$  chooses someone exactly the way  $P_1$  did and sends this letter to  $A_2$  asking him to pass this letter to the destination which is  $P_2$ .

This goes on and on; this goes on and on, he might suspect that it might indeed take a reverse direction and it might reach somewhere else. Now can this happen. I do not think this can happen because when  $A_2$  tried giving the letter to  $A_3$ . He gave it in such a way that he chooses  $A_3$  in such a way that  $A_3$  was someone who sort of was close to  $P_2$  if not new  $P_2$  personally, was locationally region wise close to  $P_2$ .

If you remember the previous example, people passed on the letter to the next person they knew who might possibly be closer to P<sub>2</sub> than what they are. So, this reversal thing may not happen mostly does not happen right. So, what does A<sub>3</sub> do here? A<sub>3</sub> intern sends it to another friend of his namely A<sub>4</sub> and A<sub>4</sub> sends it to some other friends let us say A<sub>5</sub> and A<sub>5</sub> sends it to some other friend namely A<sub>6</sub> and A<sub>6</sub> might know P<sub>2</sub> and A<sub>6</sub> will send it to P<sub>2</sub>.

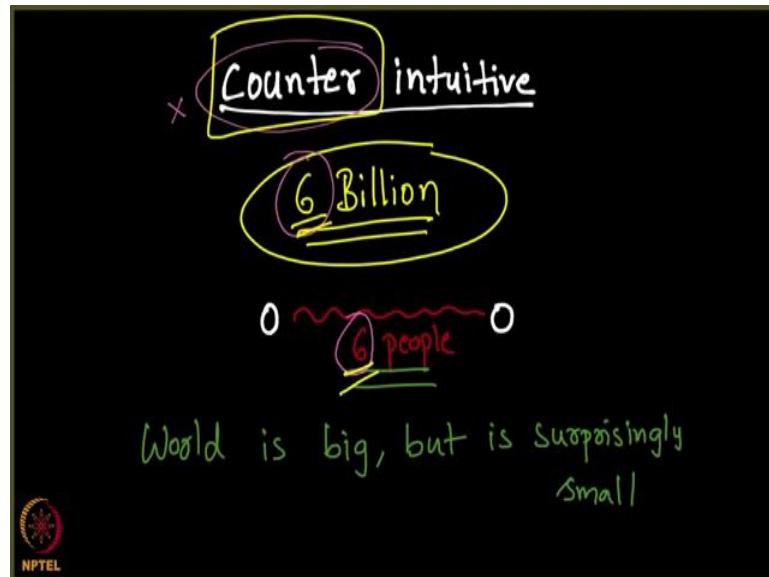
Milgram observe that surprisingly most of the letters that he sent they started with of course, the source like this and then reached the destination. In this entire thing happened in 6 steps on an average. I repeat these 6 steps are on an average and Milgram observed this, and he published this article in a magazine called psychology today and this magazine did; then this article did attract a whole lot of attention back then.

(Refer Slide Time: 04:49)



And people did not believe that this was true that any two people in the world are simply separated by a mere 6 hops. By 6 hops I mean with 6 friends, they can always connect any two people; absolutely any two people can be connected. Pick any person A another person another person B, you will observe that there are 6 people separating them and again as I told you, this is on an average 6 hops. And Milgram did this daring experiment and published these results. Now, what was this to say? What is this obvious or is it counter intuitive ok?

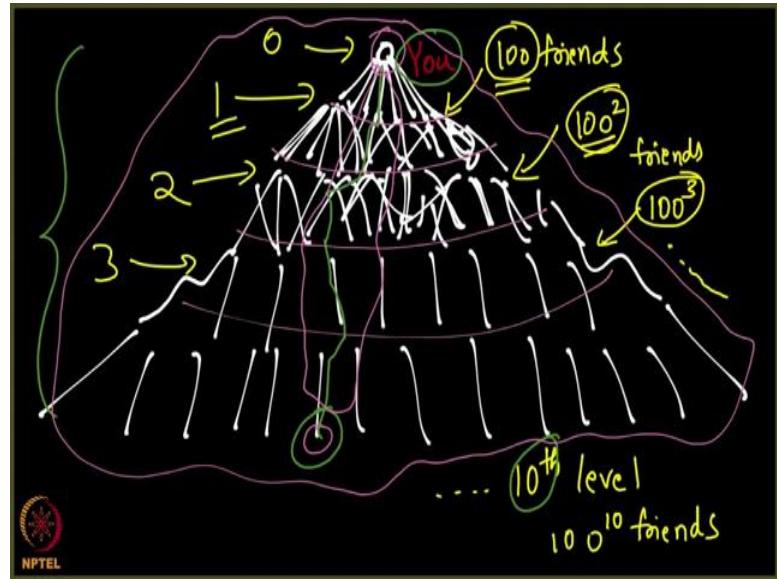
(Refer Slide Time: 05:35)



It is indeed counter intuitive, why is it counter intuitive? Let us see the counter intuitive part first. This is counter intuitive because the world as you know has roughly it say what is the best what you know, the population of world is more than 6 billion people. Out of this 6 billion people all that we are saying is any 2 people are connected by just 6 hops that is unbelievable right. Just 6 people separate each other. That is unbelievable because this looks like two huge a number and this looks like too small a number right.

So, what is happening here? We this is the counter intuitive part of it. The world is big the world is big, but it is surprisingly, but is surprisingly small as well at the same time; surprisingly small by small we mean of course, the world is big because it is 6 billion people. But then the world is small because it is 6 people who separate any two people right. So, if you want to know someone in the world, it is not difficult for you can indeed know them through a friend's friend's friend's friend's friend right. Let us see why it is actually not counter intuitive? It is not so counter intuitive. In fact, let us see what is behind this idea, how exactly this happens.

(Refer Slide Time: 07:16)



So, assume this is you and you have some say 100 friends. So, you have some say 100 friends and then each one of your friends they intern have 100 friends each, 100 friends each, 100 friends each so on. As you node the next step when each person has 100 friends, it turns out to be 100 into 100 which is 10,000 friends. And as this keeps going on and on so, each of your 10,000 friends will have 100 friends each and so on and so forth right. It each level has so, what is the number of friends at this level. It is what was the number of friends the second level, let me try writing it square instead, 100 square friends.

The next level will be the 100 square each one of them will have 100 friends. So, it will be 100 cube and so on. Let us say at the 10th level like this at the 10th level, you will have 100. So, as you can see this let us call this level 1, let us call this level 1. So, level 1 has 100 friends, level 2 has 200  $100^2$  friends, level 3 has  $100^3$  friends and so on; 10th level you have  $100^{10}$  friends that is a huge number.

And as you can see, if you want someone from let us say that this person, you want know someone in the world right you can should just sort of slight down the slope of people. And the height of this as you know as I told you, 10th level is simply 10 hops down. So, you a for people will know every single person in the world in this in a span of this let us say, 10 steps down right. It is not counter intuitive 10 is a small number as you can see

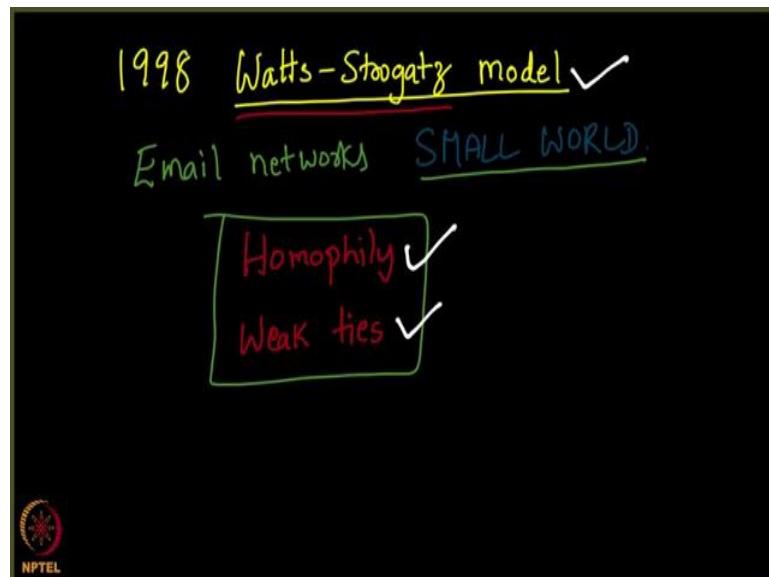
right. This here sort of grows exponentially. It comes down exponentially right. So, that is the reason why you will be exhausting the entire world's population.

If you look at your friends, your friend's friends, your friend's friends and so on. So, we saw that it is indeed counter intuitive given that it is 6 billion, but any 2 people are separated by 6 hops said Milgram right; Milgram said this and this is indeed counter intuitive. But when we see it properly in a different lens, we observed that is not all that counter intuitive. It is indeed sort of obvious.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**The Small World Effect**  
**Lecture - 145**  
**The Reason**

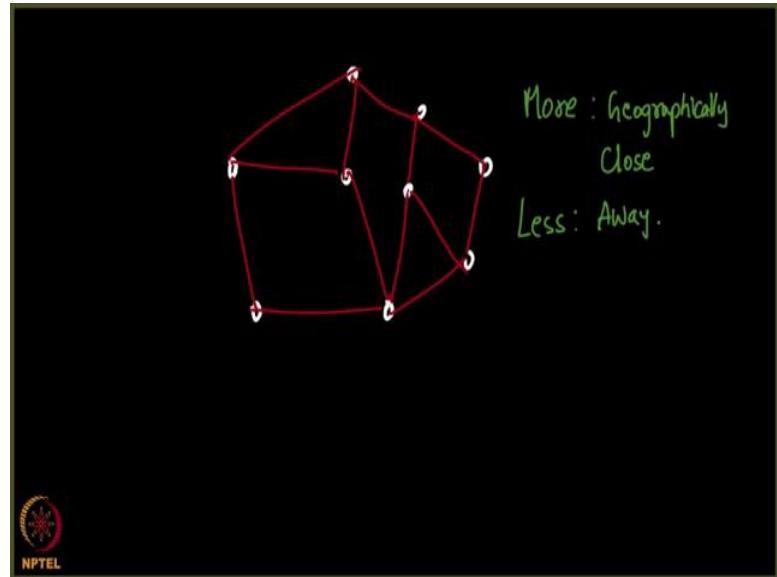
(Refer Slide Time: 00:07)



In the year 1998, this experiment was redone, and it saw a fresh look by a couple of scientists by name Watts and Strogatz. It is in fact, popularly called the Watts Strogatz model which will be seeing very soon. So, they reconducted this experiment on email networks, email networks and observed the small world phenomena on email networks as well, they observed the small world phenomena. It was not way to surprising for them and they went ahead and proposed that their possible reason why this is happening is because of two concepts which we indeed saw already which is homophily and the concept of weak ties.

The first nice explanation was indeed given by Watts and Strogatz they said the reason why the small world is being observed is mainly because of two concepts; homophily and weak ties. Let us dwell deep into their explanations, what exactly was the reasoning that.

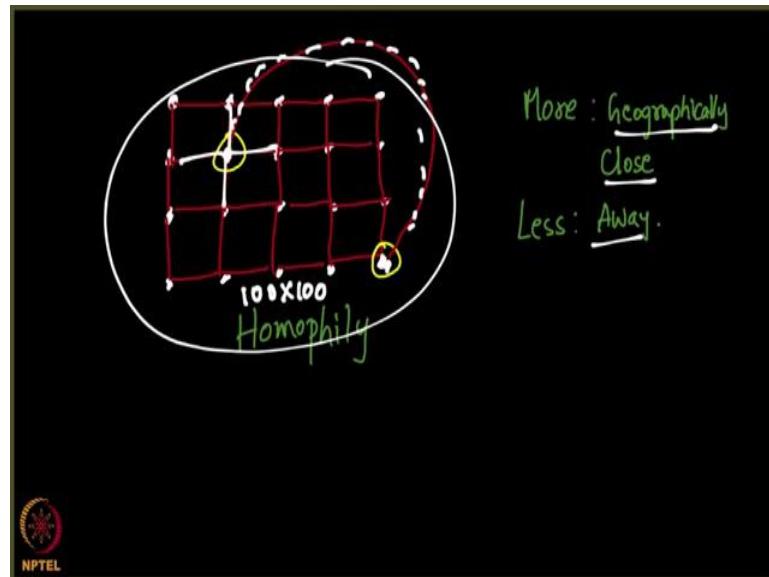
(Refer Slide Time: 01:40)



They supplied they said in a given network of course there are many nodes, there are several edges connecting them, correct. So, they proposed that these edges are actually sort of grid like structures right, the grid like structures.

Then this is like our locality in your place. So, you live in a locality and these are the friendships that you have made; obviously, you would have made more friends, more friends who are geographically close to you, more who are geographically close geographically close right and less people who are geographically away from you correct. So, instead of seeing it as a graph, let us modulate as a grid. So, what do I mean by that?

(Refer Slide Time: 02:47)



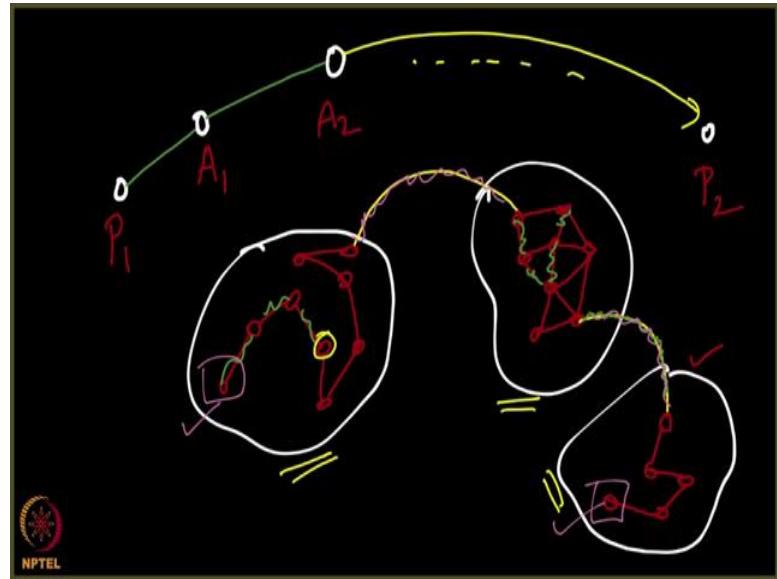
Let me just write down that this is my locality. These are let us say the houses or my the way this could be a school or college, an office or whatever, so, all I am trying to say is let us have a very simplistic model of representing people as white dots and their friendships as edges and each person has some friends in his neighbourhood ok. He makes friends in his neighbourhood because of homophily right. People who stay together in the same locality probably share some characteristics and they become friends with their immediate neighbours right. So, now, such a thing probably happens because of homophily as we saw.

But then, but then a person who is here might make friends with a person who is here. So, this person might make friends with a person here. So, let me draw a red line here right.

So, imagine this was a huge let us say 100 cross 100 grid 100 cross 100 grid and some two extremal points are becoming friends like this. This is very much possible right, you might no friends only in your locality, but you do know some friends away from you as well right.

So, Watts and Strogatz proposed that it is because of the homophily and weak ties that in a grid like this while most of your friends are your neighbours on the grid you tend to some of us tend to make friends with far away people as well and this is what is leading to the small world phenomena as observed by us and what is that let me explain.

(Refer Slide Time: 04:55)



So, when a person is asked to deliver a letter to another person,  $P_1$  is asked to deliver letter to another person  $P_2$ ,  $P_1$  will indeed give that letter to his someone of his neighbours, but then he does not give it to any random neighbour. He gives it to that neighbour he thinks is close to the destination right. He gives it to that neighbour who is close to the destination possibly right and this keeps happening right. But then 1 might know someone who is not his neighbour but is someone who is staying far away from him who might be very close to  $P_2$  alright.

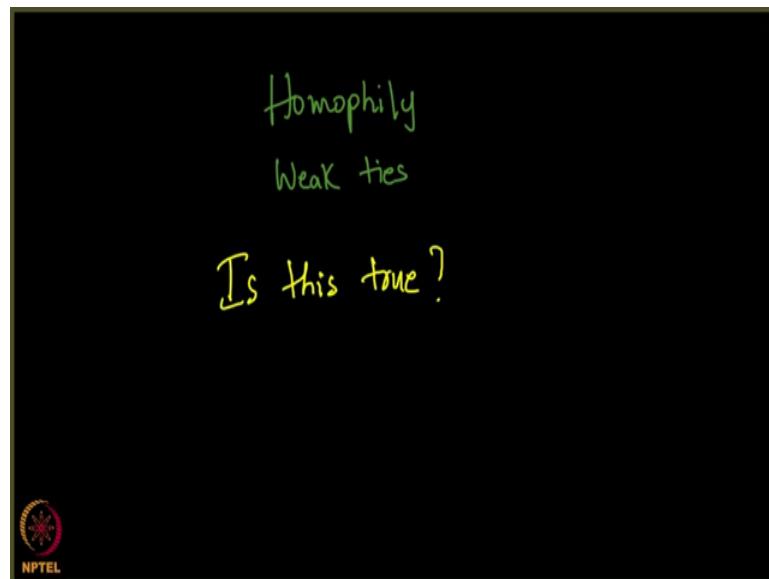
So, this is what happens. So, when let us say this is one region this is another region this is another region. When you are trying to send a piece of information or let us say a letter or whatever you call it through your people whom you are close to you probably will stay in this shell, you probably will stay in this city. But some people do have links to outside the city and they will pass on the letter to people outside the city and again once it goes outside the city it probably will stay there because of homophily.

But then some one person might know someone from this world rather this country or this continent and the letter will indeed reach this particular place which initially was not the case as you saw. We were just roaming around here and then we came here and then we roamed around here a little and then we saw that there was a path to the next country and we entered here and then slowly it reaches the desired destination ok, from the

source it reaches the desired destination. Mainly, because of two things as I told you which is homophily and the notion of weak ties.

By weak ties I mean the at edges such as this we have studied already right. It has these two things that lead to let me write that down once again.

(Refer Slide Time: 07:26)

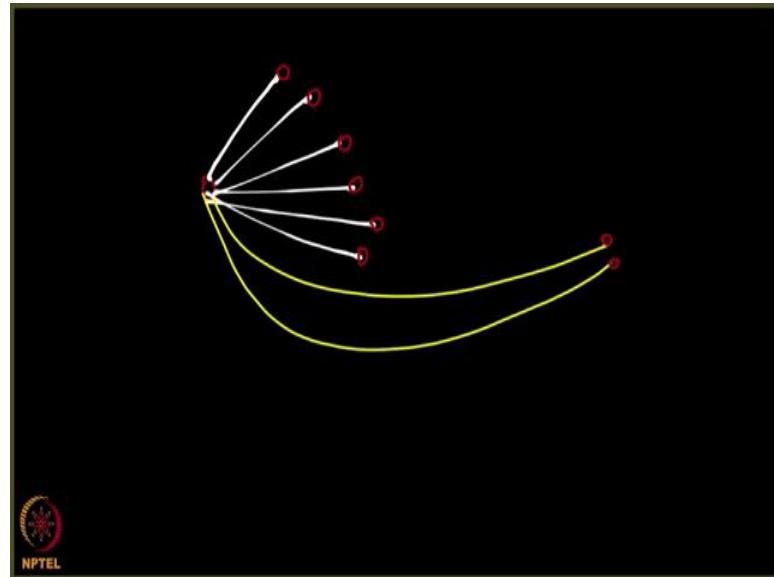


Homophily and it is not just homophily that leads to small world phenomena, homophily coupled with weak ties that result in such a phenomenon from occurring.

Now, anybody can hypothesis something like this. How do we know this is true, is this true, is this true? By let us been observed in many networks that this is indeed true that several networks have what is called these clusters like this right and nodes exhibit a homophily, they make friends with many people who are like them, but then there are friends who are outside the territory as well.

So, let me illustrate it well.

(Refer Slide Time: 08:15)



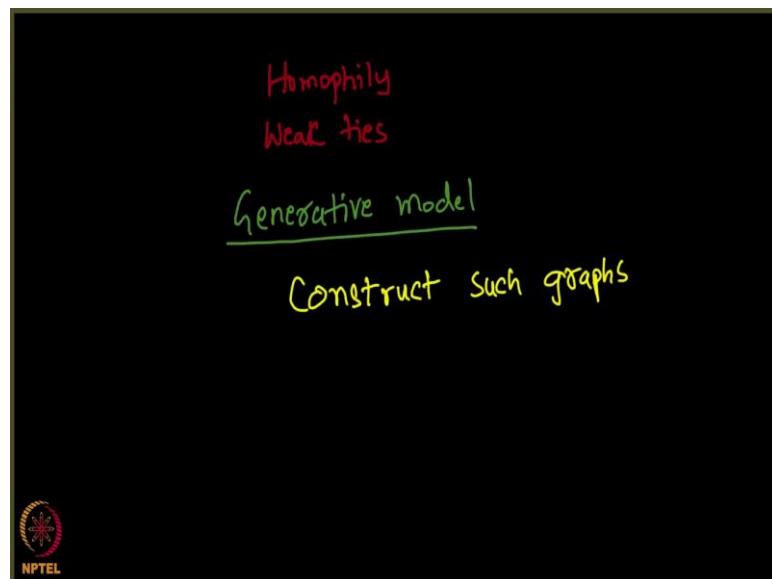
This is you, you will have many friends who are like you inside your territory, but a few of them will be outside the territory right, outside the territory. Some of them will be, most of them will be inside the territory. By territory I mean your zone homophily and a couple of them will be outside your territory. It is this couple of them couple of those friends that each one of us have would results in the small world phenomena right that makes the world really closely connected.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**The Small World Effect**  
**Lecture - 146**  
**The Generative Model**

Scientific theories like any other theories are sometimes hypothesized. By hypothesis we make something that is proposed to be true proposed as true for which we may not have complete scientific evidence. So, this was one such proposal where Watts and Strogatz observed it and said homophily and weak ties could be the reason why this is happening right.

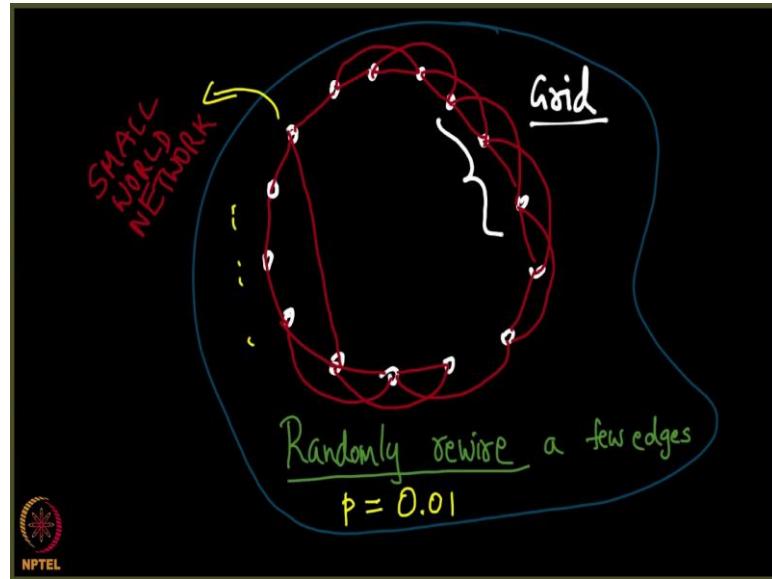
(Refer Slide Time: 00:28)



Then they said we will now give a generative model for this. By generative model we mean the following. A generative model stands for a way in which you can construct such graphs, an algorithm or a method through which you can construct such graphs. What do we mean by such graphs?

Graphs with the property that we are discussing graphs with small world properties. So, there proposal was this.

(Refer Slide Time: 01:11)



First consider a cycle ok, put all the nodes on a circle like this cycle like this whatever, so, how many wire you want. And then start drawing edges to their nearest neighbors. Let us say something like this. So, everybody is friends with let us say the next two people adjacent to them. So, let me try explaining it. This chap will be a friends with him and him, this chap will be a friends with him, this chap will be a friends with him and so on and so on, right; so on and so forth.

So, now what what since Strogatz model suggested is that start with this is very similar to a grid that we saw right. Start with a grid where you know people who are geographically close to you right and then and then there is a just head that you should randomly re wire; sounds little technical, but let me explain in a minute what; that means. Randomly re wire a few edges by that we mean taken edge here of your choice whatever you want. Let us say I will remove this edge as you saw I remove this edge and I am going to randomly put it back as well ok, I will randomly put it back, it can be between any two edges, I do this kind of re wiring.

So, there is a probability that we can associate with that kind of re wiring. I will say I will do some 1 percent of the edges that I have, I will randomly rewiring re wire them; that is called the random re wiring probability. It was observed that whenever we did something like this, the resultant network was indeed a small world network. What do I mean by that? As I have been explaining this is the kind of network where you will

observe that any two nodes are in fact, very close to each other. You take a billion nodes here and then repeat this and you will observe that give a negative nodes there is a very short distance between them; this is called the Watts Strogatz generative model.

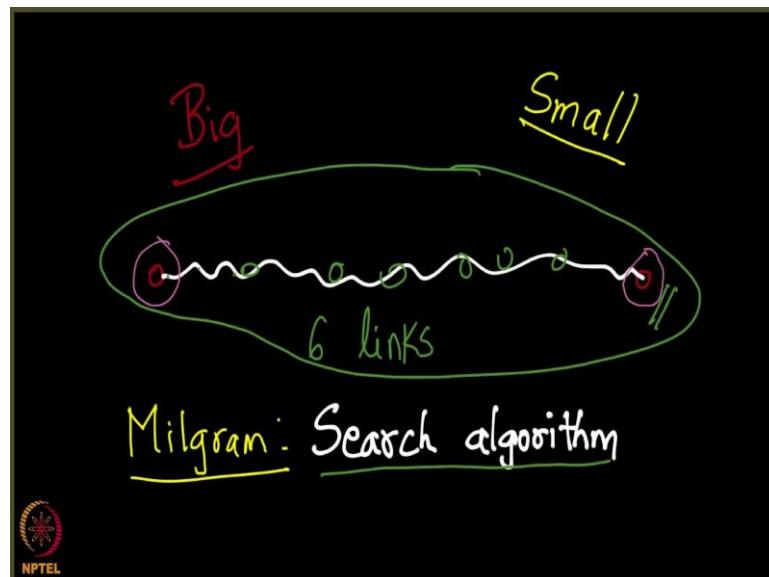
You can probably try writing a code for this and then check whether it really works. The way it is shown here, you will be surprised to see that it indeed works and that is the motivation for why the world is big yet too small.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**The small World Effect**  
**Lecture - 147**  
**Decentralized Search – I**

Fine, we saw how the world is connected.

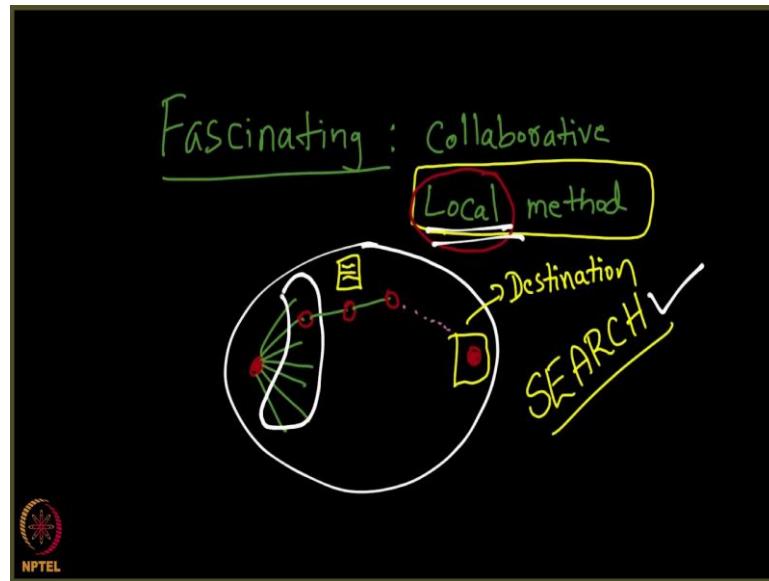
(Refer Slide Time: 00:13)



The world is way to big on one side and we saw it is actually also small in a different sense right: big in one sense, small in another sense. What is more important here; this experiment let us we collect. So, there were two people, anywhere in the world, they are connected by some 6 links right: 6 links. I am sure you people now see the point and you sort of appreciated, but what is more worthy of being appreciated is not the fact that given any two people in the world they are connected by near 6 links right 1, 2, 3, 4 links a more interesting fact it is the following.

In the Milgram experiment if you observe, it is actually a search algorithm. It was the search algorithm, why so? You see that a person is holding a letter here and does not know who is here and he supposed to find a path on the friendship graph of the world right. There are so many people over here; he needs to find how to get to the reach this person here right. It is indeed a search algorithm.

(Refer Slide Time: 01:55)

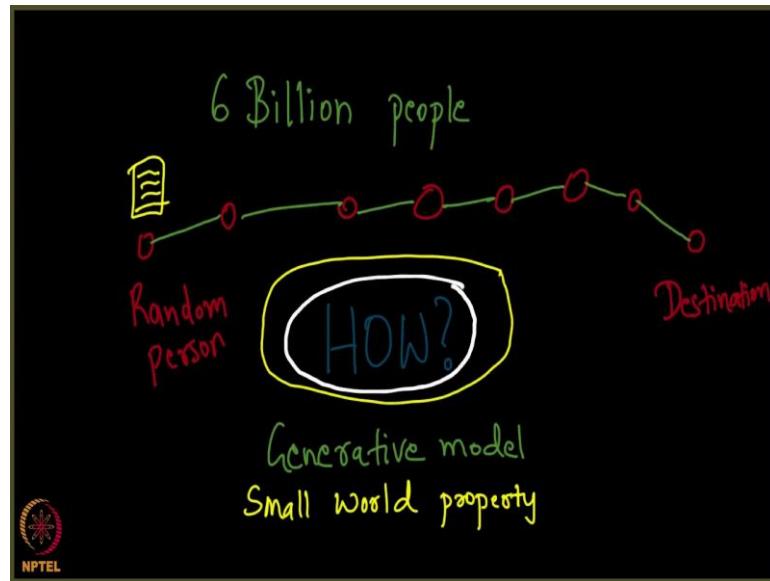


What is more fascinating? What is more fascinating about the Milgram's experiment is the fact that it is a very collaborative attempt. People use a sort of a very local approach, a local algorithm by a local method. We mean they do not know of the global picture in the graph right. So, in the few friendship network: a person here only knows who his friends are and has no clue of who could be the friends of this destination person. He has local this person has local he has only a few friends here. Now he done very intelligently passes on the letter to someone of his friends hoping that it will reach the destination and they someone friend sends it to another friend hoping that it will reach the destination.

So, a point to note here is this is indeed a search algorithm. And what is so fascinating about this is that it uses a local method, he does not know the entire graph given here. Nobody knows entire graph, they only know one's friends and they just pass on the information to one of his friends, right. And then surprisingly Milgram notes that it finally, reaches the destination. It reaches the destination and the approach is completely local.

I hope you understand what I mean by the word local. By local I mean a person who has the let us say the letter right now, only things about passing it on the most eligible friend of his; hoping that this chain will continue. And finally, help the letter reach the destination. It is a very local algorithm; it is indeed very fascinating to see how this works.

(Refer Slide Time: 04:17)

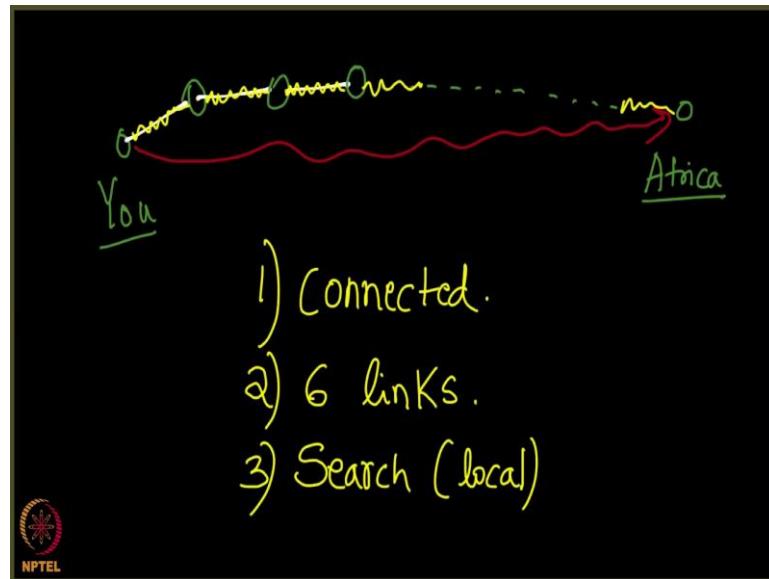


Imagine there are 6 billion people as I was telling the population of the world is more than 6 billion people in this world. And some random person when you take and give a letter to him and ask him to make this letter reach, you take a random person and ask him to help this letter reach the destination the destination. Although, he has no clue how to do this, he somehow manages to give it the impetus. He gives a push to it and every single person does the same thing right; every single person does the same thing. And finally, it was observed that in roughly 6 hops, you reach the destination in that a sort of misty. Mystifying that without anybody knowing the entire graph, they all are cooperating and helping the letter reach the destination.

Now, our question is how this happens. How? Is it even it looks unbelievable, does it even happen right? So, if we went to write a computer simulation, what would we write? What coding would we do to see that this indeed is true? You remember the generative model that we discuss in the previous session, the generative model called the Watts Strogatz generative model right. We showed you how to generate such graphs which have what is called the small world property, right. It is called the small world property which means any two people are connected by surprisingly a smaller number of links.

Now, let us slowly understand how exactly this search algorithm gets executed and how exactly it is so local. Nobody knows the global picture at they ensure that within 6 links, the letter reaches the destination.

(Refer Slide Time: 06:45)



So, assume let us take as an example a real world scenario where this is you. And this is let us say some someone in some part of let us say Africa and you badly want to contact this person for some business, and you have no clue how to contact him.

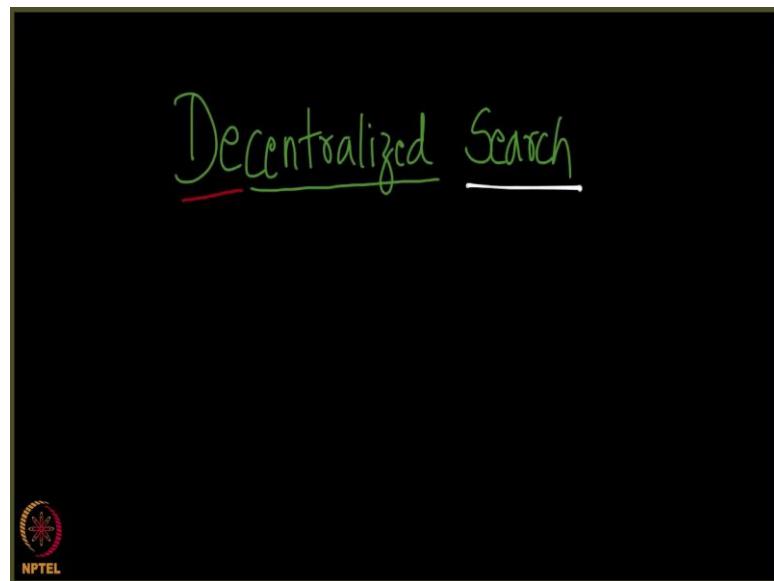
Again, an assumption is we are in the let's say 60s and 70s where there was no internet. Nowadays it is quite easy. Although the same phenomena happen; it is not as difficult as it sounds in this example right. If you were want to contact someone in Africa, how would you go about? You will ask someone, again the same story and ask them to help you with someone who might possibly know this person in Africa who might know someone who might know someone in this place called whatever in Africa and so on. And surprisingly as I have been repeating, this helps you reach the destination. This way I mean this has this has sort of this has been one of the most counterintuitive facts in graph theory ever that not only is the world connected look at this.

First point is not only is the world connected, it is surprisingly connected in the sense that in 6 links we can go reach the destination; how awesome is that. And the third and the most fascinating fact is that the search algorithm that people use to go find the destination is a local method. By local method I mean I repeat by a local method, I mean nobody has the global picture; nobody knows who is friends with whom, but they just do their local job. All that the person needs to do is to ensure that he sends it to the

immediate neighbour and this every person follows the same protocol. And finally, it goes and reaches the definition.

Whereas now see the how of this algorithm, how exactly this works. This goes by the name decentralized search and we will see more of it right now.

(Refer Slide Time: 09:03)



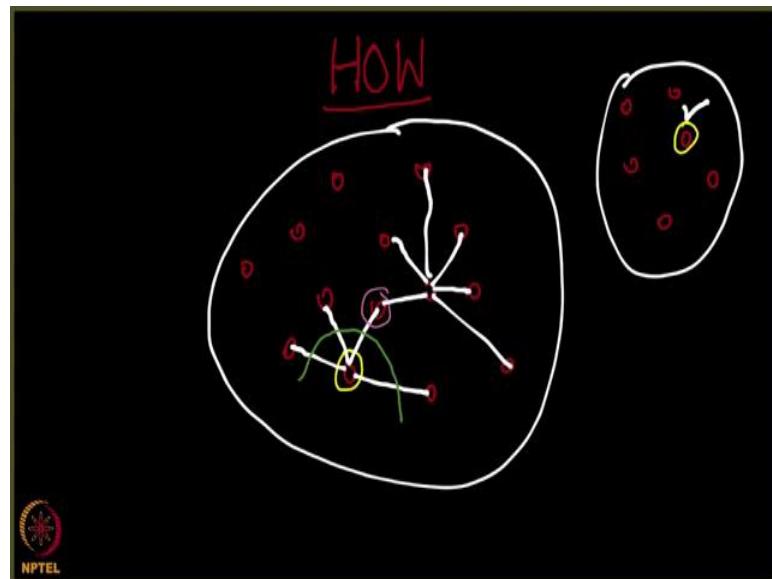
Decentralised search, it is actually self explanatory as you would have realised. Centralised means there is central authority; decentralized means there is no central authority still you are able to search.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**The Small World Effect**  
**Lecture - 148**  
**Decentralized Search – II**

So, we asked a question in the previous lecture, the question is how is this happening?

(Refer Slide Time: 00:11)



How is the search happening on the small world network? Right, it is very intriguing you see I mean how can someone just pass on a piece of information to someone else and in 6 hops you can reach the destination anywhere in the world, the reason actually is pretty simple. In fact, we gave a intuition of it in one of the previous lectures; we are going to go in detail with it right now. As you know in the real-world network you know how its it happens how a real-world network is created, we saw the Watts-Strogatz model right.

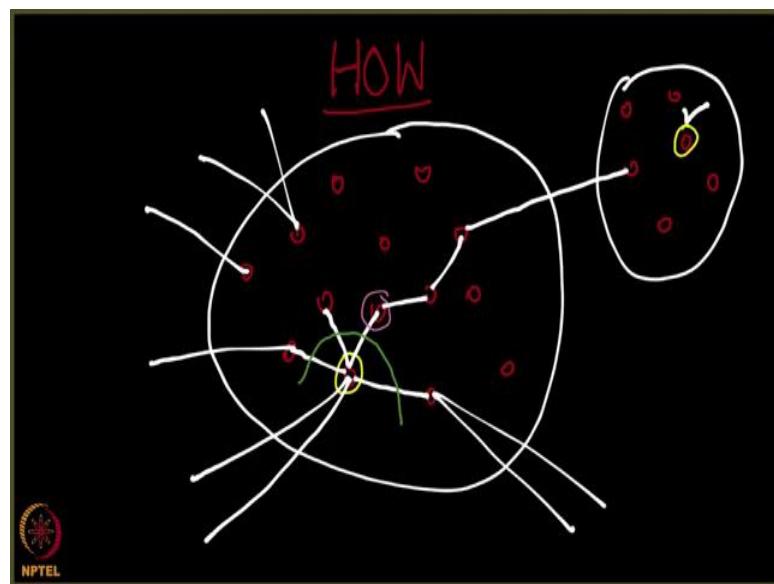
There are several nodes and several links. How do links form? Assume this is you, how will you make friends? You will basically be friends with your immediate neighbors, your classmate, your schoolmate maybe your gym partner, you're the person whom you know in your painting club or may be in your biking club and so on and so forth. You know the local people really well, but then as you know there are some people who

might know someone outside their locality right. So, whenever you are asked to send a piece of information let us say to some other part of the globe.

All you will do is you will choose one of those friends of yours who probably is closer to this destination, who is possibly closer to this destination. And that person in turn chooses another person, another person and this person might see you see how and why, how and why probably he was chosen. Why did this fellow chose this fellow? Probably because this fellow here this guy might actually know this guy or someone in this particular region that is why this fellow chose this fellow, while he actually knew maybe several people.

He purposefully chose him of all people to send right he knew several people, but right he knew several people. But he chose to send it to only this person that is because he suspects this person might really know someone from this region.

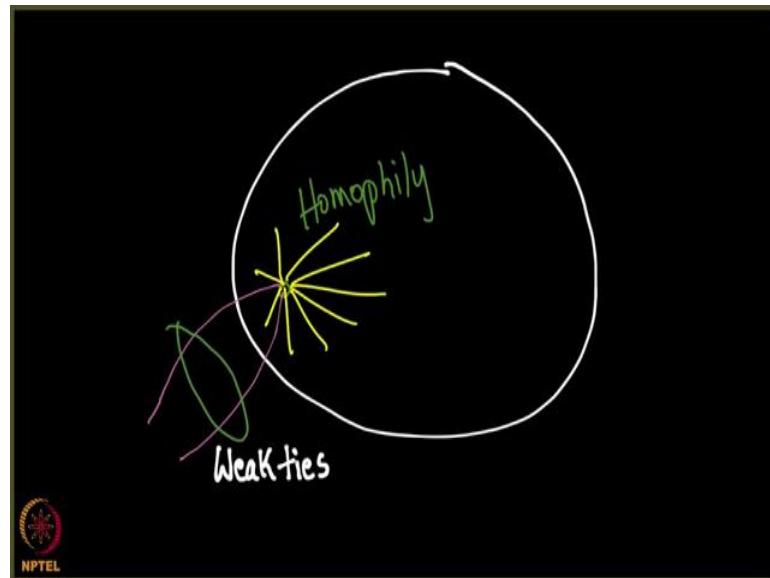
(Refer Slide Time: 02:51)



Did you think this happens, if you want to send a piece of parcel to someone in Afghanistan or let us say someone in Dubai or someone in some Arab country. You probably will know a friend of yours in your college who is from one of these countries right. And, you may want to send it to this friend and ask him to send it to someone in let us say Dubai which will help the packet reach the destination, which is probably in one of those Arabian countries right.

So, that is the whole idea behind this local algorithm, where a person starts from here and then intelligently tries to reduce the distance on the destination. We all have a sense of what is the destination, how close we are getting, and we try to achieve it. So, now how do we simulate and see this?

(Refer Slide Time: 03:49)



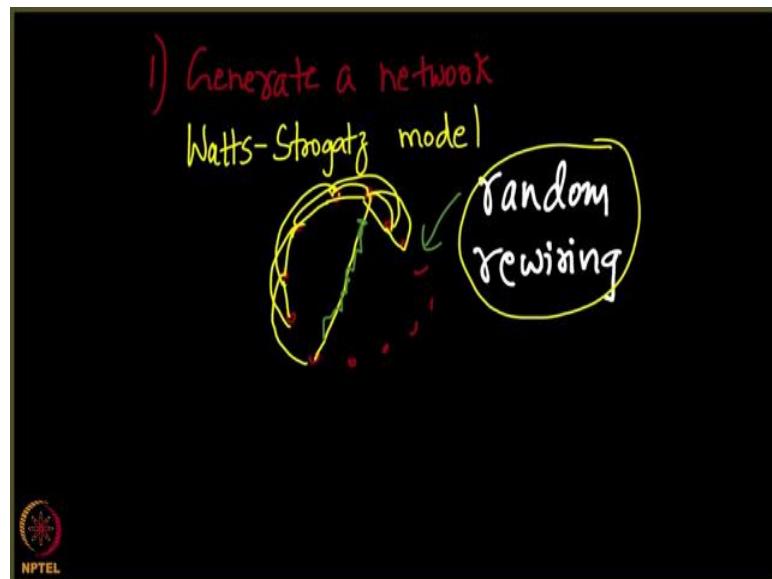
We first take a Watts-Strogatz model. Now, before anything let us go back to the previous slide and then see. So, this kind of edges actually common in the Watts-Strogatz generative model that I told you which depicts the real world networks; you see basically we make friends with our localities in our localities people who are around us. But, then couple of links will actually be to outside places example this fellow although is friends from, is friends with people who are close to him right next to him; he might also have a couple of friends outside. This fellow might have couple of friends outside, this person might have someone person outside, 2 friends outside, someone outside this always happens you see.

So, let us say if we have a person here with some links inside a lot of links will; obviously, be in his locality itself, but a few links will actually be outside. Isn't that true? We know a lot of people in our country, but we also do you know let us say 1 or 2 percent of our friends will be outside our country. They form as you all know what is called the weak ties, they form the weak ties and how would we simulate this algorithm

on such a network? Let me let me revise what has happened so far. All I have stated is there is homophile because, of which we are friends with people who are right next us.

There is also the weak ties that we have because, it is only intuitive that 1 or 2 percent of us will always have some friends outside the country, outside our geographic location correct and that is what makes passing of this packet becomes very easy. How do we simulate and see this? I will just be, just to complete I will write this as these are this constitutes homophile ok. How do we program this and see it? What we should initially do is, I will give this as an exercise problem for you all you should be able to do it. In case you are unable to do it please let us know we will help you out.

(Refer Slide Time: 06:17)



So, first thing is we need to generate a network with small world property, we know how to generate that. This is called the Watts-Strogatz model, where if you remember you take nodes right, you take nodes and you put edges to your immediate neighbours, who are geographical closer to you correct. So, this goes on like this other if it is closer to you this happens for on every single node here so on and so on and so on because, these are all connected like this so on. And, you do what is called some random rewiring, isn't it? You do what is called the random rewiring.

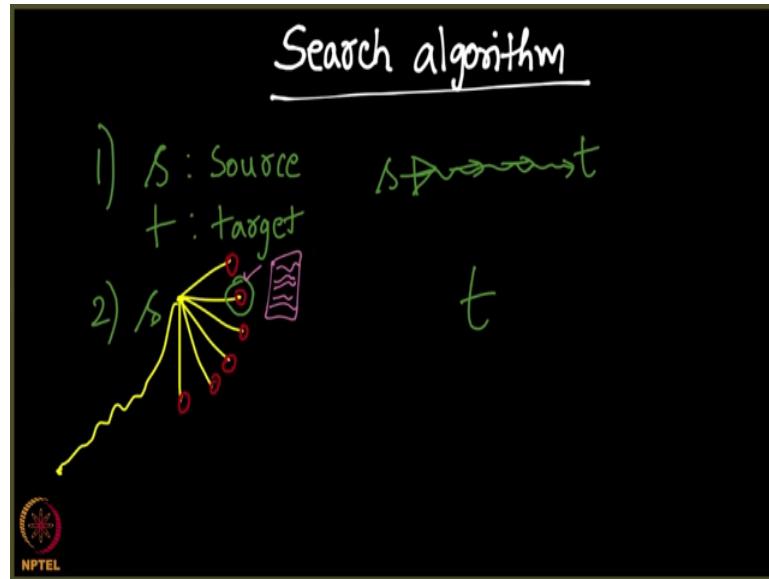
So, what you do is everybody's friends with their locality people only their immediate neighbors, but a few people random rewiring means you take some 1 percent of the friendships and shift them randomly; you resort of relocate them randomly right we have

discussed this before. So, if there was a node from let us say here to here you basically remove it and then put it somewhere from here to here. That way this becomes what is called let us say a teleportation channel; a person who is here might only know people in his local neighborhood, but then by this random rewiring what you do is you connect him to a person who is otherwise geographically far away from him.

And that is what the Watts-Strogatz model tells you, for you to get this small world property in such a random rewiring should happen. What do you mean by, what do you mean by random rewiring should happen? Who makes it happen? The point here this is how the real-world network functions. So, you might ask me who does random rewiring, nobody does random rewiring. This is how it you know things are I told you the fact that you have a couple of people right outside your geographical location, while most of your friends are, we within your geographical location.

You might have a couple of friends outside your country, outside your continent right and that is equivalent of the random rewiring here right. When such a thing happens, we can actually try implementing our own search algorithm.

(Refer Slide Time: 09:13)



So, what we do is the following, we are first you are asked to go from a source node  $s$ , let us say a source node  $s$  and a target node  $t$  and a target node  $t$  ok. And, then what you do is I will write the second point you should go from  $s$  to  $t$ ,  $s$  to  $t$  you have to go to  $t$  and what you do is you will go to  $t$ , this direction, this direction, this direction. So, what you

do is you pass on you find that you find all nodes of s, you see s is of course, adjacent to many nodes. But you are on the lookout for that node which might be closer to t. As I have stated before as I have stated before there is this always this couple of people who are connected outside couple of your friends who are outside your geographical location.

So, you might find you may not find, but certainly amongst these friends of s that you have, some one of them would probably be close to t right. Someone would possibly be close to t and will pick that person who is close to t close to t and pass on the letter to him as simple as that, will pass on the letter to this very person right. And, as you know this sort of continuous and goes on. Now, let us think about this algorithm for a minute. What if this s is you ok, you are this s and t is in Australia, but you have friends see as you are in India and you need to send something to Australia? How does search algorithm work?

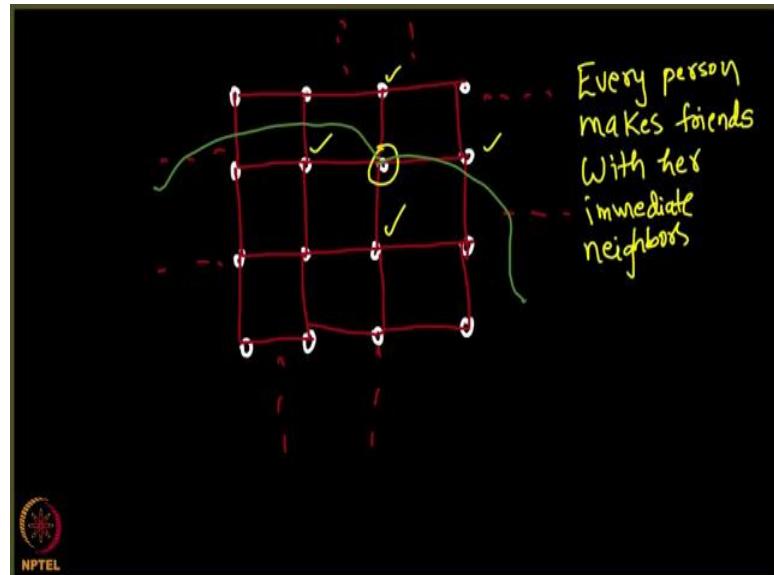
Let me try to confuse you people with a question here. What if s has all his friends in India with a couple of friends outside and that friend who is outside is let us say one from Indonesia and one from United States of America right. If you were to pass any of these; if you were to pass the letter to any of the people within India either you are not sure it will go to Australia correct. But, at the same time if you try passing it to Indonesia or we should type United States you will be you will go further away from it right.

So, like going further away from t if you were to pass it to some someone else right. So, this random rewiring may not; this random rewiring may not necessarily guarantee you teleportation points, as I was telling you to nodes of your choice. So, that you can send the letter easily correct; these weak ties may not really connect you to the desired destination. They may not take you close right? they may not take you close. So, how does search still work? Right, it still works right we then in that case you know we just pass on the information to anyone here who is anyone here, who is close to t; he could be a fellow from India itself.

Or if the closest is actually someone from Indonesia to t we of course, do pass it to that person right. But, the question here is may be it would get complicated all the more complicated and you may not reach the destination at all, you might go away and away

from the destination well and fact is this never happens and the reason follows. So, what we will do is let us take a grid ok.

(Refer Slide Time: 13:37)

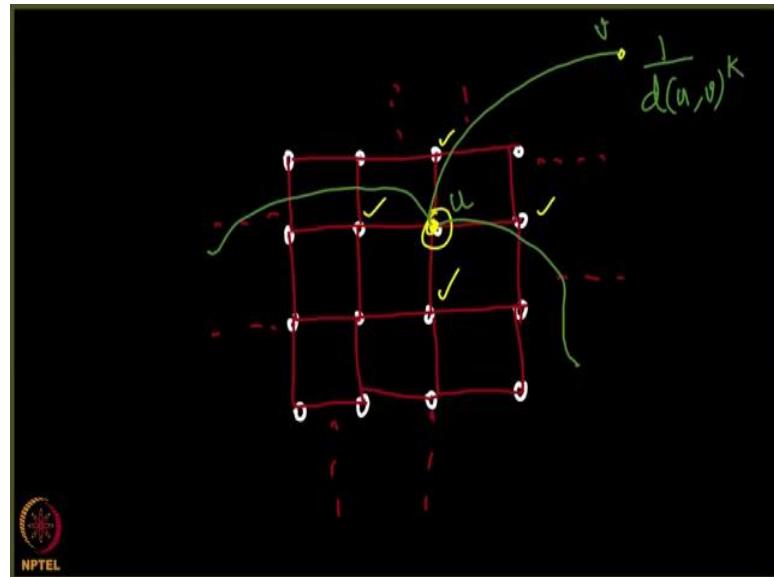


Just take a 4 cross 4 grid like this right and then complete the grid. So, every node has so many friends and as you can see a person has 4 neighbors and assume they are people on all the sides; it is a huge grid it is a huge grid. And, this represents that you are friends with; you are friends with these 4 people 1 2 3 not the one I am sorry yeah you are friends with these 4 people. But, then you are also friends with a couple of people away from you something like that right. So, assume every person let me write that down every person makes friends, I am writing it slowly because it is important. This better sink into our minds every person makes friends with his or her immediate neighbors right.

It will be necessarily be the 4 neighbors which are these, he might make friends with the nearest neighbors who are let us say 2 distance away from him. So, he might be friends with him 2 distance away, 2 distance away, 2 distance away etcetera. In his vicinity he makes friends, a couple of links is always way outside his vicinity which is far away ok. Let us assume this is the friendship network in 2 dimension and I will even write a code let us say to define what exactly are these green lines here. What exactly are these green lines? They are links to far away nodes. Let us come out with the way in which we

define these far away nodes ok. Let us come out with a way in which we define these far away nodes. So, how we define it?

(Refer Slide Time: 16:15)

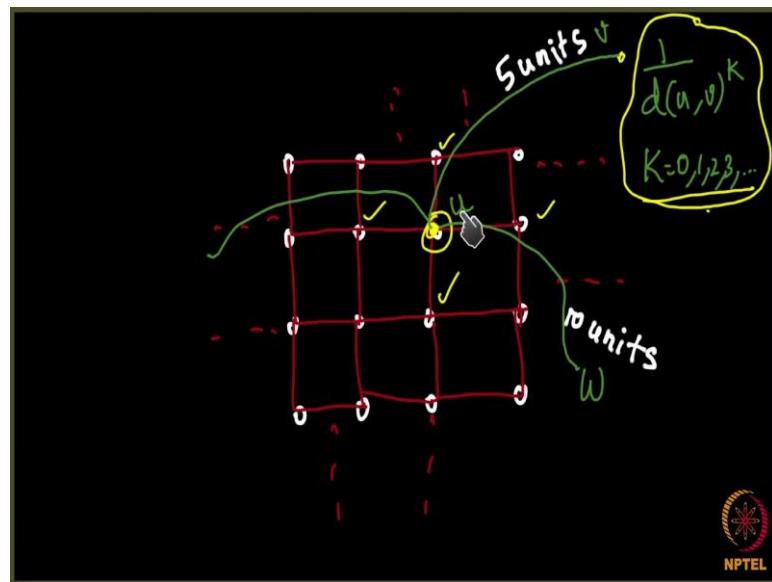


What I do is I pick a node far away from this node and then I put an edge with probability. Let us say 1 over distance between this is u and this is v, distance between u comma v to the power of k.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**The Small World Effect**  
**Lecture - 149**  
**Decentralized Search – III**

(Refer Slide Time: 00:05)



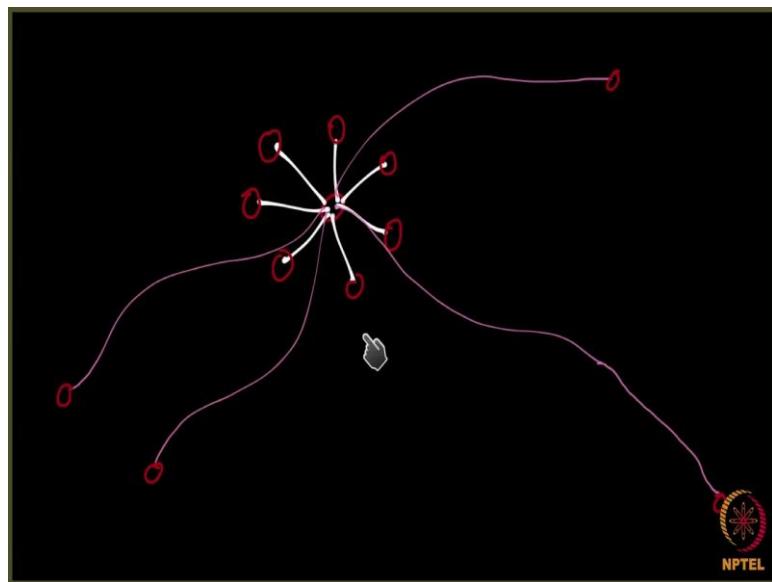
So, K here equals 1 2 0 1 2 3 here denotes that you are rewiring, basically your friendship that u maintains with people who are not in his neighbourhood. The friendship that u maintains is sort of it is based on the distance of that node from u; further away the node less probable is a use u being friends with them. For example, let us say the this is a node which is some 10 units 10 distance away from u and this one is some 20 distance away from u.

So, this is less probable, and this is more probable ok, that is what this distance function captures and for a fixed K. So, what do you mean by this? If the distance is let us say 5 here let me let me write that down. So, if this distance is let us say 5 units and this one is 10 units, the probability of u being friends with v is more than the probability of u being friends with this person ok. Let me give him some name, let me call him w this person let its call let us say this node is called w ok.

The probability of  $u$  being friends with  $v$  is a lot more because, it is closer just 5 unit away; 5 units I mean 5 edges away; it is not just 1 edge ok. In case  $u$  and  $v$  are 5 edges away then you put an edge between  $u$  and  $v$  based on a probability function ok. What is the probability function here? Nothing it is very simple you judge whether you want to put an edge or not by looking at the distance from  $u$ . This is far away from  $u$  this is 10 units far away from  $u$ . This is 5 units closer to  $u$ , this is more probable to become a neighbour of  $u$ ,  $v$  is more probable to become neighbour of  $u$  than  $w$ .

What decides this? It is this equation that decides this. So, 1 over if you let us say assume  $I K$  equals 2 then  $1/5^2$  is the probability of  $u$  being adjacent to  $v$ . And,  $1/10^2$  is the probability of  $u$  being adjacent to  $w$  that is if  $K$  equals 2. If  $K$  equals 1 then  $1/5$  is a probability of  $v$  being adjacent to  $u$  and  $1$  over  $10$  is a probability of  $w$  being adjacent to  $u$  ok. Let us not worry much about this fact, let us only observe the following all I am saying here is there is a node here. And he is of course, friends with his immediate neighbours, a few of his immediate neighbours alright; very clear till here.

(Refer Slide Time: 03:09)

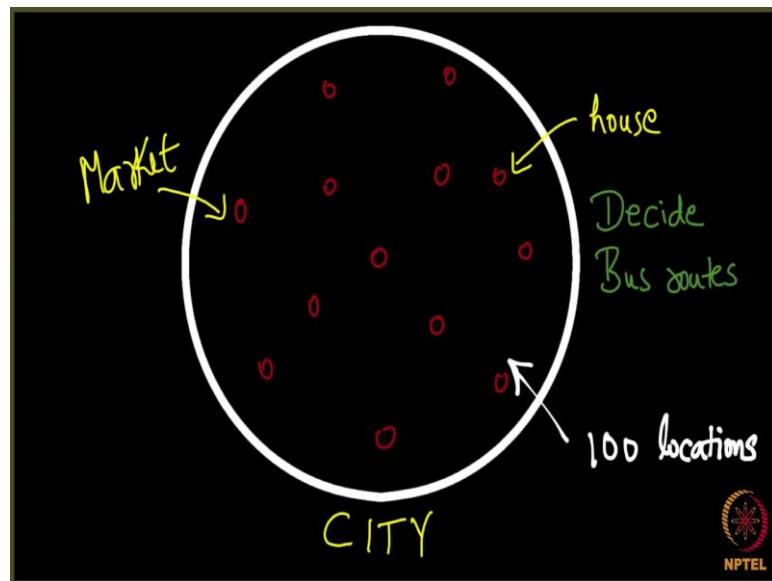


And all that I was saying before is that he also takes part in being friends with far away people, far away people let us say a few of them. Few friends are far away people for him, but then that friendship simply depends on the distance ok. By that I mean further away they are less probable that this person is friends with him. Who decides it? I am

going to decide it based on the inverse of the distance. So, if let us say in plain English language if this is some 5 kilometres away.

This is 20 kilometres away, the probability of friendship here is rare probability of friendship here is a more probable ok. So, I once I give you a nice example you will realise what is the happening here, what is the motivation for this distance function alright. So, here goes the right motivation; assume there is a city you are a city planner.

(Refer Slide Time: 04:27)



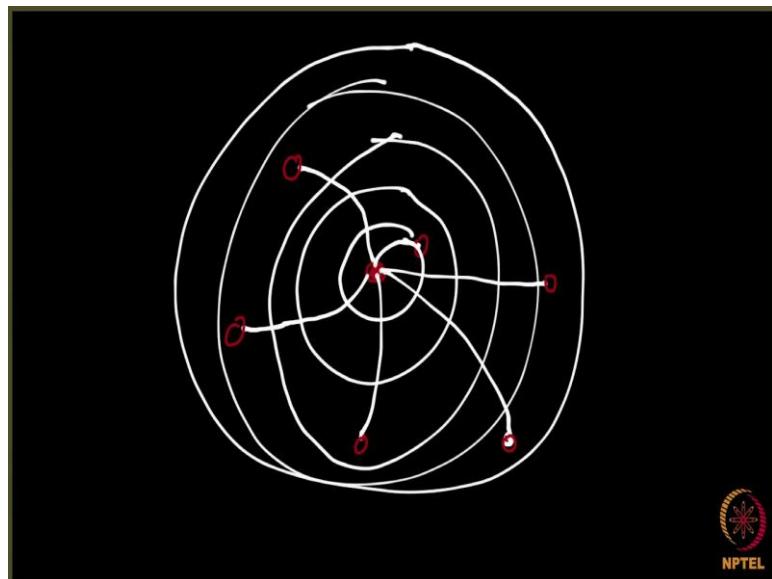
Let us assume you are a city plan, and this is the city and here is the centre of the city alright and you may want to go to different places here. These are all different locations, for the time being just forget the graph here. We are not talking about a graph; we are just talking about let us say we are planning a newly built city. The city is newly built, I am planning some road networks here, public transport here to be precise; planning let us plan some public transport here. So, now assume this is your house and this is some market a supermarket or a shopping complex. So, whatever you want, now you want to go from here from your house, you would like to go to market.

Of course, you can take your car and then go, but the point is you would like to use public transport. There are so many locations here, as you can see there are let us say 100's of locations here. You cannot basically put a bus a public transport bus from any node to any other node here; from your home to market you may not have a bus. So, let

us say you may want to go from your home to let us say some other place and to some other place and then reach the market. So, buses do not ply between any 2 locations.

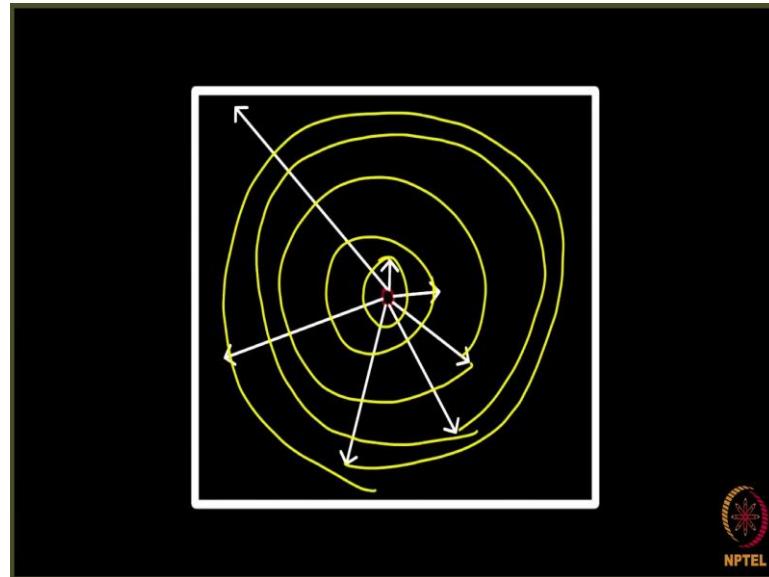
Now, here is the question if you were to decide what should be the bus routes, if you were to decide the bus routes what will be the locations from which you will um deploy a bus right. Assume every single location has the same amount of people and the requirement for them to travel is also the same, what exactly will you do? How will you plan the bus routes? Right, assume you have some 100 such locations here, there are 100 such red points whatever you are seeing here 100 such locations. So, whatever points you are seeing here assume there are 100 such points and you cannot simply put buses from any point to every other point right that is not feasible that is not practical.

(Refer Slide Time: 07:09)



So, what you do is this; assume this is the city all you do is you there is a central bus stand, let us say this is the central bus stand right. And, then from here you will have a bus go this place, this place, this place, this place, this place and this place right. These are the several locations where your buses go right and then, do you see something here? Do you see that? There are concentric circles here and you are trying to ensure that you have a bus for every possible concentric circle correct. So, what do I mean by this? By this I mean, if you let us let us now take a grid which is easy on the mind.

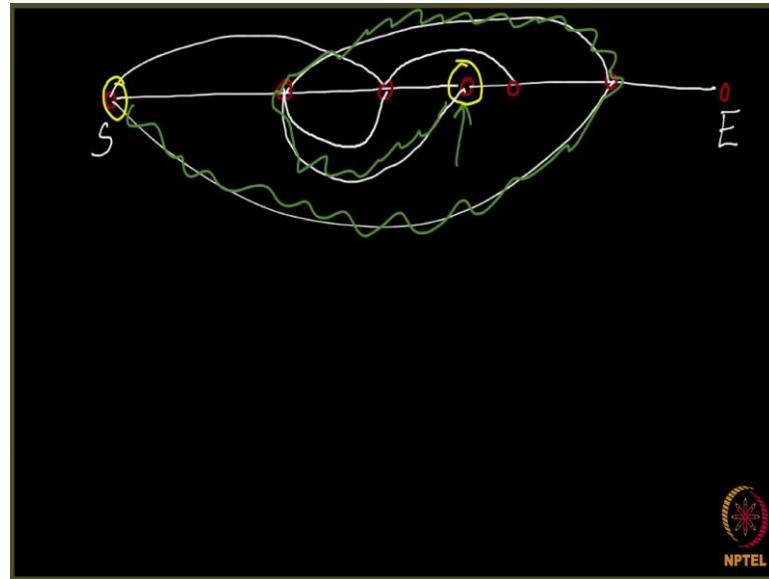
(Refer Slide Time: 08:01)



Assume there are some locations here, you would not sure that there is a um bus that goes to concentric circles to concentric circles. So, what do I mean by that? By that I mean from this centre let us say from the centre there are buses which go just to the next location, next to next location, next to next location, next to next location so on, so on and so forth ok. You will try to cover all possible locations from here correct, all possible locations. I am not drawing it well, but you basically you are getting the right intuition right.

So, that you are you are basically covering all possible areas in the place and the bus will go there and from there you may want to take another bus.

(Refer Slide Time: 09:03)

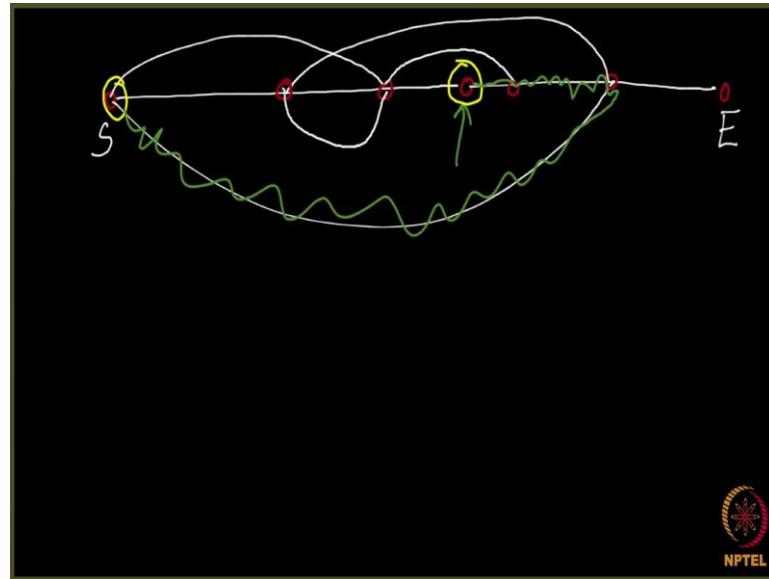


Simply very simply stated, assume there is a metro service where you need to go to let us say this place ok. Metro starts from this is the start and this is the end, let us say it is linear it is just it is a straight like this and you need to go here. You need to go here, but a metro train goes to this location and let us say it goes to this location ok, these 2 locations let us assume. But then you may want to get down here and then take another metro to another station, but that station goes a little further away and from here you can always come back to another station.

And that station probably takes you to further away place, there are probably isn't any metro from here to here. So, as you can see you need to reach this place starting from this place, but the metros are very weirdly distributed. Some superfast metros you know that they do not they do not stop they do not stop; from here you directly take you here. And, then from here you can only come here and from here maybe there is train to this place and there is not a train to this place. See it is complicated, how would you come then?

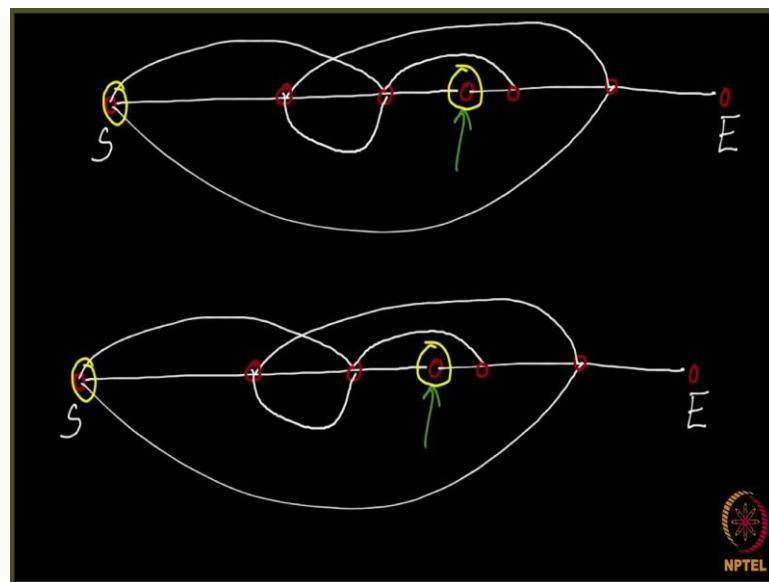
So, then you would probably take a metro to this place and then travel to this place and then travel to this is place. You know how to take a detour here; you see this is how things work.

(Refer Slide Time: 11:05)



That is because if your reach is high, by that I mean see if you have long routes people may want to come back from at ok. If you only have shorter routes let us see what happens.

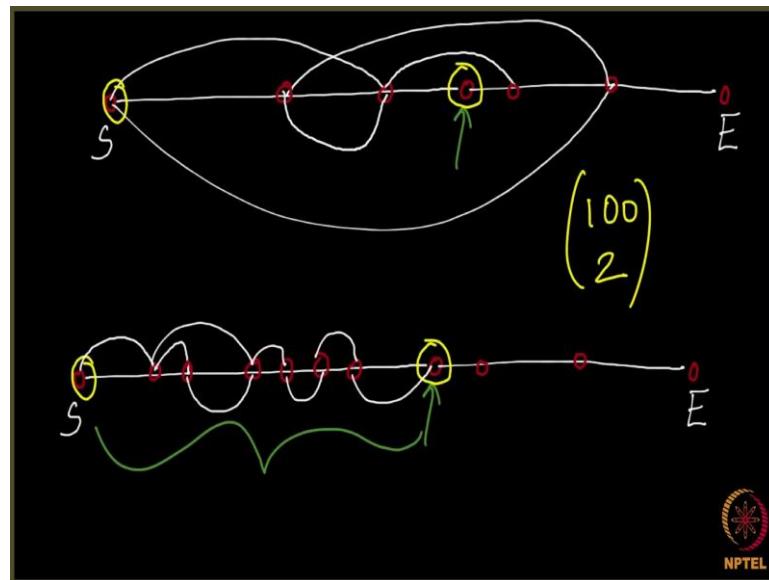
(Refer Slide Time: 11:15)



If there is a shorter route from S to E exactly the same thing ok, let me just that replicate this and then put it here so, that it is easy for us to work. I will copy paste this and let me give you another scenario where every single route is fairly short ok, it is fairly short. So, what we do in that case? That that comes with another baggage of problems right; let me

once again copy paste this in this and then pasting and then I am going to tell you that, what if this were not the paths let me remove these paths now. And, I need to go these to same destination right, I need to come and reach here right. I am going to start here, but then the metros are all like this now, there is there is a metro to this place and then another metro to this place.

(Refer Slide Time: 12:15)

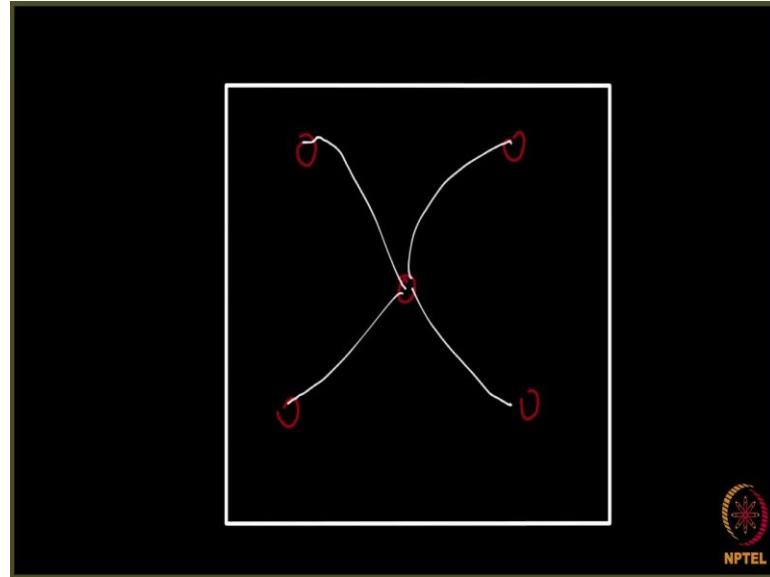


And then from here there is another metro this place so on and so forth. Here to here and then again from here to here alright. You know how to do a whole lot of toggling; you see a whole a hell lot of toggling you may have to do from here and then to here and then there is a metro to this place right. So, you may have to do a lot of jumps here, it this is a very strenuous process right. This is a very strenuous process, as you can see. Why is that, there is a no there is no quick way of coming from here to here right. If there are some 100 locations here you cannot put 100 choose to 100 c 2, let me write that down you cannot put 100 c 2 number of um trains or buses in this on this road. You can only put a few buses or trains between stations correct.

This the above one has a problem that, if you put superfast trains you may have to go ahead and then come back while, here you may have to do a lot of hops it is a lot of waste of time; you see you need to get in to a bus and then another bus and then another bus and so on, you need to do a lot of hops. What is the ideal way to construct um metro service here right um? So, what we will do is, if you observe this problem in this is the 1

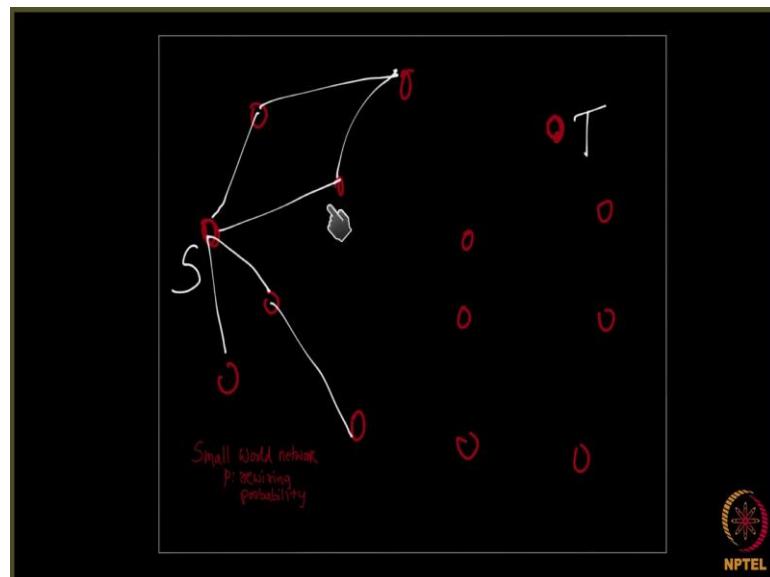
dimensional, you see starting from here and going to here. If you see the same problem in 2 dimensions, if you see the same problem let us say assume you write a square or a circle you represent a city right.

(Refer Slide Time: 14:05)



We have exactly the same problem that when you want to put buses from one locality to the other right, it is a huge issue from where to where will you put from where to where will you put. So, let us do one thing assume you were to write program you accomplish this. So, what will you do? You will first generate a small world network.

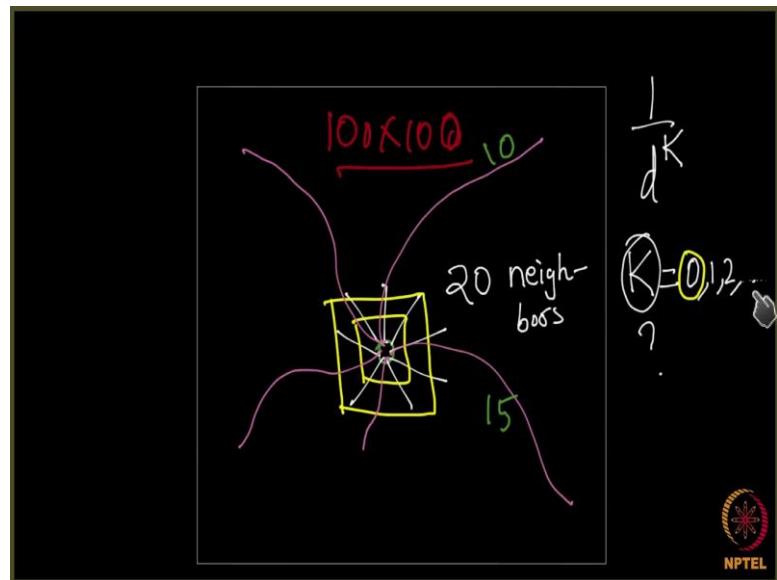
(Refer Slide Time: 14:55)



Let us say a small world network with some rewiring probability ok;  $p$  is the rewiring probability; you will indeed get some graph here right you will get some graph here right. So, let me let us write that down I get some graph here right, I get some graph here. Let us say a huge graph here correct and then I can go from any node to any node right node. But I must ensure that I can go from any node here to any other node here, from any source to any target I should be able to go from this source target that should be able to go. And, I want my edges to facilitate this that the whole point edges to facilitate this.

So, how does a small world network look like? It will have a lot of edges to its immediate neighbours geographically ok. Look at this is this is a city and a closer the vertices more are the possibility of edges, but then here and there I do put edges to further away nodes to ok. Now, what I will do is let me assume that I put I put um 10 I take 100 cross 100 grid right, 100 cross 100 grid whatever this was, if you remember a huge grid.

(Refer Slide Time: 16:33)



I will take here 100 cross 100 grid, then what I do is I um I basically will do this I will let me write that down; it is it is a slightly involved. You will understand it better if you get the next 2 minutes of my explanation. So, this is a 100 cross 100 grid let us say right and every single node here, let us say every single node here is adjacent to some let me say 10 nodes 10 nearest nodes; whatever they are on the grid alright 10 nearest neighbours.

What I mean by that? So, all these people are maybe at 2 levels ok, let us say some a 1 2 3 4 5 6 7 8, let us say some 20 neighbours 20 nearest neighbours ok.

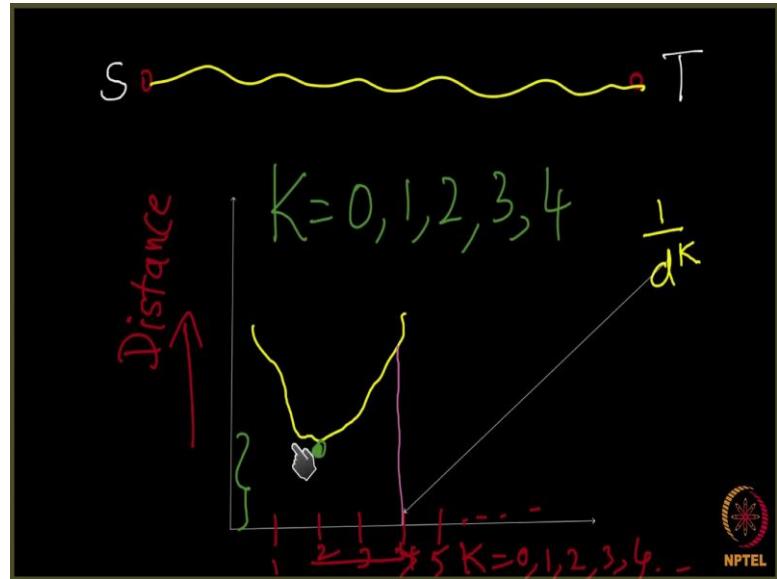
This is the homophile part 20 neighbours and then it also is adjacent to a few nodes further away and this is decided based on the probability function 1 over to the K, where K can be 0 1 2 etcetera as I told you right. What should be the ideal value of K? If I were to run this experiment on a 100 cross 100 grid right and I will choose first K equals 0, what will happen? When K is 0, this entire thing will be 1 which means it will be the same probability with which you will be picking vertices further away; its uniform distribution that can be dangerous you see.

Why? You will randomly put roads between 2 nodes then you may have to go longer distances and then come back into if you want to come to the destinations alright; it may not really serve your purpose. If you put K equals 1 then node that is 10 units away and the node that is 20 units away 10 units is more probable because, it will become 1 over 10 probability 20 units with will be a lot improbable, because it be  $1/20$  ok. Now, instead I take K equals 2 this be then it becomes let us say assume this was 10 units and 15 units so, 20 units whatever anything you want.

Then if you pick K is equal 2 this becomes, the probability of this being friends will be  $1/10^2$ , the probability of this being friends will be  $1/15^2$ . But, then again there is a problem there would not be very further away nodes because,  $1/15^2$  is  $1/225$  and that is a very small number, which means a node that is 15 units away by units as you all know, I mean the number of edges the shortest path length. If a node is 15 distance away then the probability of that being friends is  $1/15^2$ , but a node that is 10 units away is  $1/10^2$  which is  $1/100$ .

This is more probable; this is very less probable than what it is used to be when K equals to 1. As you can say if K is equals to 3 or more nodes that are far away becomes very in probable and navigation becomes difficult for you to go from a source to a target. So, here goes our question we were talking about simulation right continuing our discussion.

(Refer Slide Time: 20:43)



If you were to go from a given source to a target node  $T$  and you are considering different  $K$ .  $K$  equals to 0,  $K$  equals 1,  $K$  equals 2, 3, 4 and assume you draw this plot ok. Assume you draw this plot of this  $y$  axis versus the  $x$  axis right and then you will plot, what are the path lengths. If you put on  $x$  axis you put  $K$  which is basically 0 1 2 3 4 etcetera ok, 1 2 3 4 5 and so on. And, on  $y$  axis you put the let us say the distance, the distance covered when you take the check the typical such algorithm. When you when you go to the next node and then you check how far is the target and you go to the next node that is closest to the target and so on.

When you do this, you will observe that when  $K$  equals 0,  $K$  equals 1,  $K$  equals 2 etcetera the plot looks like this ok. It let me write nicely, the plot has the observation is at 2 it says something. The distance is minimum now what, but do I mean by distance? By distance I mean you need to the destination you can go and reach quickly. If the rewiring probability is  $1/d^K$ , where  $K$  is 2 when  $K$  is let us say 3 the distance is slightly more as you can see. When  $K$  is let us say 4, then the distance is a lot more as you can see ok. I hope you are understanding what I mean by this distance.

By this distance I mean the path length of the process, when you want to go from the source to the target it is a longer path if you were to choose  $K$  as 4. If you were to choose  $K$  as 2 the shortest the path from  $S$  to  $T$ , what do you mean by what do I mean by path from  $S$  to  $T$ ? The same way Milgram experiment, when you take a node and you go to

the next node which is closest to be on the grid ok; of all the neighbours you choose that node that is closest to T on the grid. That is what we have been discussing all this while right. And this particular algorithm will tell you, this particular programming exercise will tell you that a K equals 2 is the best possible value for K.

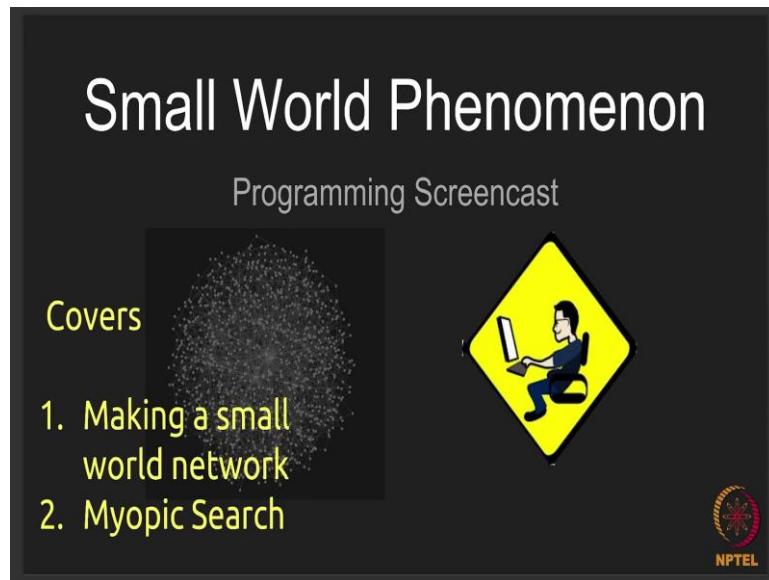
Whenever, the rewiring probability that you are considering is given as 1 over d to the K right. So, if this is sort of a hand waving that I have done here. In fact, that is a very beautiful proof for this saying that K equals 2 for grid and K equals 1 for 1 dimension. And textbook describes it, it is very nicely described with some very basic probability you can understand it completely. Let me know if you have not understood yet, I will help you out understanding it. But, this will be out of your own interest; this will not be part of the course examination.

But, the idea here is simple idea here is that you use K to sort of decide on whether you will make friends with nodes that are further away right. And, when you make friends with nodes that further away a lot more then you are in trouble; a lot less you are in trouble. The perfect balance is when you choose K equals 2 ok. In fact, we will be showing you programming exercise; we will be explaining this in detail. It will become probably clear to you, then. As of now just know that this is curve is how it looks like and for 2 the distance is the shortest.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**How to go Viral on Web**  
**Lecture - 150**  
**Programming illustration- Small world networks: Introduction**

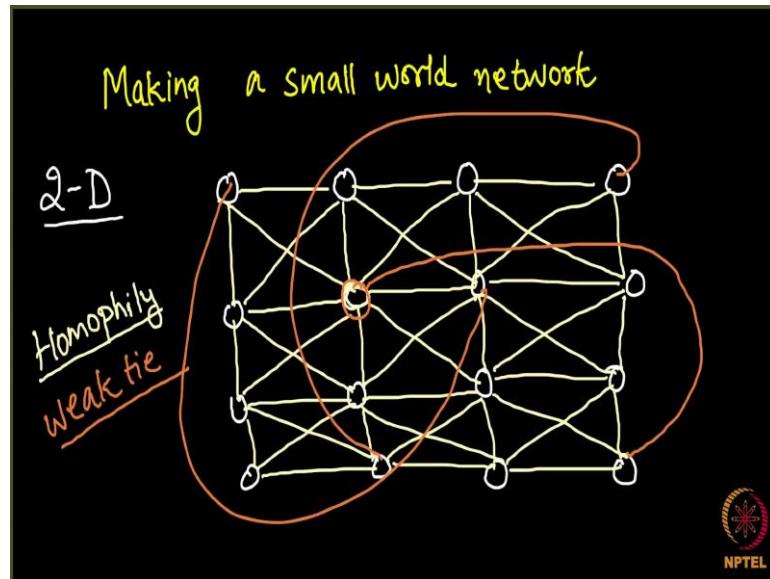
(Refer Slide Time: 00:05)



In the previous week, we have looked at the Small World Phenomenon. This week we are starting off with a programming screencast based on small world phenomenon and then we will take up another concept. The first starting off with a programming screencast of small world phenomenon, we are going to do two things there. First of all we are going to look at how do we make a small world network. So, I am sure all of you remember how we make a small world network, I will quickly recap it. And secondly, will do how do one does a myopic search on this network, how do one does a decentralised search on this network.

Now, before starting off with the programming screencast, I will quickly recap both of these concepts; how do we make a small world network and how do we do a myopic search on it ok.

(Refer Slide Time: 00:59)



Let us look at the first one. So, the first one was you want to locate how do we code, how do we make a small world network; so making a small world network making a small world network. So, here let us talk about the model which we have looked in the lecture. So, you will remember, how do we modulate in two dimensional in a two-dimensional space. We have certain nodes here. So, let us put some nodes here.

So, let us say I put some 16 nodes over here. So, I have some 16 nodes over here and then we have looked at, how do we make the connections between these two nodes. So, connections between these two nodes were based on two concepts. The first one was homophily. So, in homophily what we have done is, every node was connected to the nodes which are geographically closer to it. For example, this node here it can be connected to this node graphically closer this node, this node and this node.

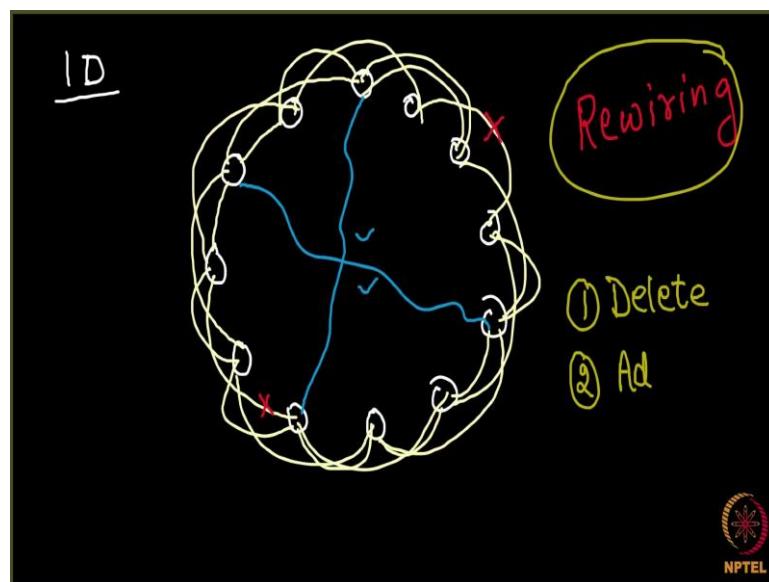
So, one on it is top bottom right and left. And similarly, for all of these nodes we can do the same thing and all of these links are made because of homophily. So, we get a grid something like this and what even we can do next is we can even connect these nodes to their diagonally opposite partners, because they are also geographically closer. So, we can make a grid like this as well. So, the grid which we had previously, this one is also correct and this one is also correct where we link the diagonally opposite nodes as well.

So, all these links are because of homophily and then we have look that what makes this world small is the presence of weak ties. So, the second reason the second concept which

gave rise to the small world network was weak ties where every node for example, this node will be randomly connected to some of the node which is quite far away from this. So, this node might be connected to somebody here, this node might be connected to somebody here, this node might be connected to somebody here.

So, these weak ties make our world small. So, this was how we can portray our world. So, gets small in two dimensions and rather then we looked at we can portray the small world network with the model with model which got.

(Refer Slide Time: 03:25)



So, gets give in one dimension which is rather more interesting. So, in one dimension, we can assume that the nodes in the network they are in the form of a ring. So, here we have some nodes in this network.

So, these are the nodes and again we have certain connections because of homophily. So, because of homophily the nodes are connected to other nodes which are geographically closer to them. So, we can take let us say 1 on the right-hand side and another on the left-hand side or rather we can take 2 on the right hand side and 2 on the left hand side. And again, this node here is connected to 2 on the right-hand side and 2 on the left hand side. And similarly, all the nodes connected to 2 on the right-hand side and 2 on the left hand side. So, let me make this connection quickly.

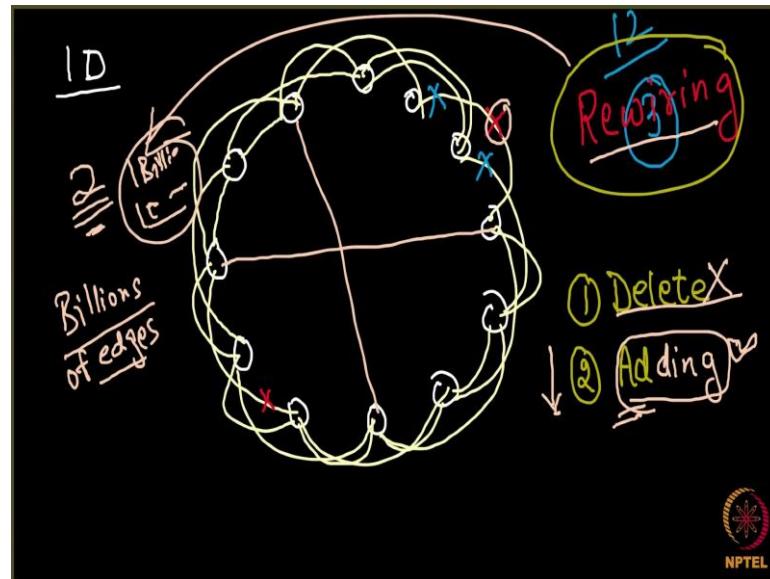
So, something like this. Every node is connected to 2 nodes on its left side and 2 nodes on its right side. So, let us say something like this and so on. So, these are the ties which exists because of homophily and again to make this world small, we should have somewhat long range ties weak ties. So, what we have looked previously in the chapter in the previous week was the concept of rewiring was the concept of rewiring. What we did in rewiring was we deleted of some of the edges from here. So, let us say this edge can be deleted and instead of this edge, we randomly put an edge somewhere else in the network.

Similarly, this edge can be deleted, and we can randomly put an edge at some other place in the network and so on. I make a for our coding for our programming screencast, I will make a small change here. And the change which I am doing here is instead of rewiring so, by the basic model which what so, gets has given it is actually rewiring where what you do it consist of two steps. So, the first is step is first of all you delete an edge from the network, you delete a random edge from the network and second you add a random edge in the network.

So, delete one edge and add one edge, but for our programming screencast I am doing a small change to this. What I am going to do is instead of this rewiring so, here instead of this rewiring, what am I going to do is, I do not rewire the edges here rather like in the previous model how did we see I am just simply going to add some long range contacts. And it will not make a as you will see; it will not make much change to the number of edges as you will see just on the addition of two edges, we will have did use the diameter of this network.

So so, let me elaborate it know what i want to say. What we are interested in is looking at the diameter of the network. Let us say we have this network somewhat like this and here can you tell me what the diameter of this network is. So, assume that how many nodes are there, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 say 12 nodes are here.

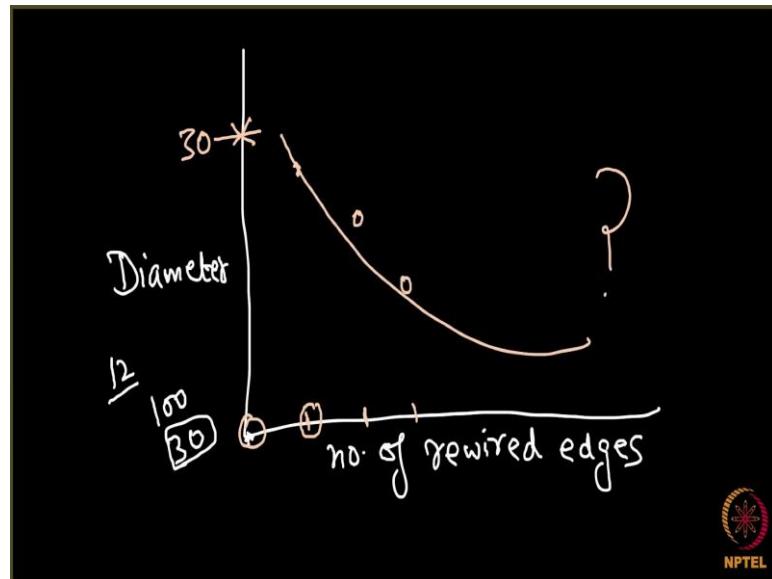
(Refer Slide Time: 06:55)



So, when there are 12 nodes, we can see that the diameter of this network is 6 or a node 6 actually it is going to be little bit less than 6. So, so, let us say from these are the two nodes which are at the maximum distance. So, the diameter will be 1, 2 and 3, right. So, the diameter of this network is 3. And what will happen when you are actually going to rewire some edges? So, let us say you rewire this edge and you put an edge here and then you delete this edge and you put an edge here, what it will do it will reduce your diameter.

So, now if you want to arrive from this node to let us say this node, the distance instead of 3 is just 2. So, you can jump from this node to here and you can jump from here. So, the diameter is reduced to 2. So, more the wiring you do, the diameter decreases accordingly.

(Refer Slide Time: 07:59)



What is our aim our aim is to look at if I plot a. So, if I plot a graph let us say, I plot a graph and on the x axis I have the number of rewired edges. So, I rewire 1 edge, I can rewire 2 edge and so on. Here are my number of rewired edges is and on the y axis I have the diameter.

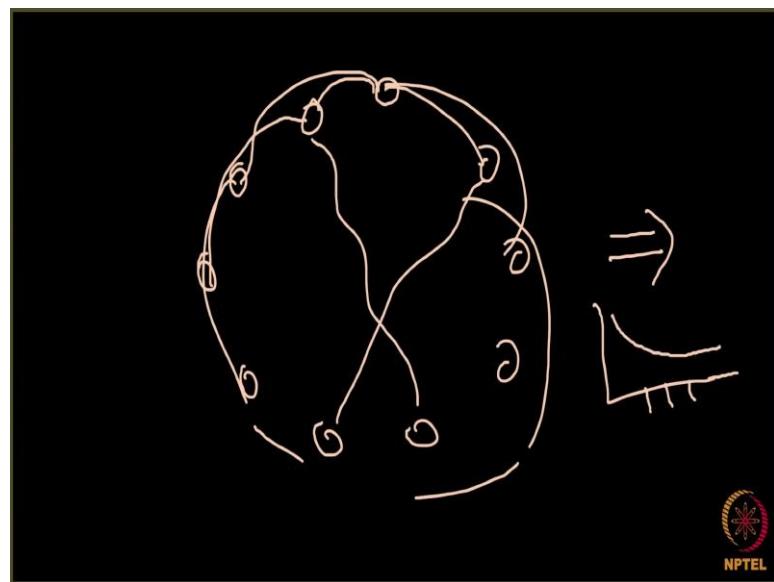
How will this plot look like? So, initially we know let us say instead of 12, let us say our network consisted of some 100 nodes and what will its diameter be? Its diameter let us say was something like 30. So, initially the diameter of my network was 30. So, here when I have 0 number of rewired edges; here when I have 0 number of rewired edges, the diameter of my network was 30. And then I rewire 1 edge and when I rewire 1 edge, what will happen? My diameter will reduce maybe somewhat here and then I rewire another edge and then again my diameter can reduce and again I rewire my diameter can reduce.

How will this curve look like? We want to study. So, as we do more and more rewiring of edges, how does the diameter of a network reduce is what we aim at is what we aim it studying. And while doing this as I discussed with make a small difference to a network over here and what is a difference is instead of deleting an edge here and adding an edge here we remove this step of deletion and we just add some rewired edges. And as you will see, the number of edges will not change much. So, you will see that on just the addition of let us say 2 or 3 edges.

On just the addition of 2 or 3 edges, the diameter will go down drastically and hence this whether you do rewiring or whether you do simply addition of an edge, these 2 are not these 2 are not very different concepts. The number of edges in the final network is actually going to be almost the same. So, let us say that you take a very big network on let us say some millions of nodes and billions of edges.

And on this network having billions of edges whether you rewire when you require 2 edges so, the number of edges will remain 1 billion only. According to the rewiring principle the number of edges will remain, 1 billion only, but if you add 2 edges, the number of edges will be 1 billion plus 2; 1 billion and 2 which are almost the same. So, here is a little change that we make to this network to this is framework of building your small world network.

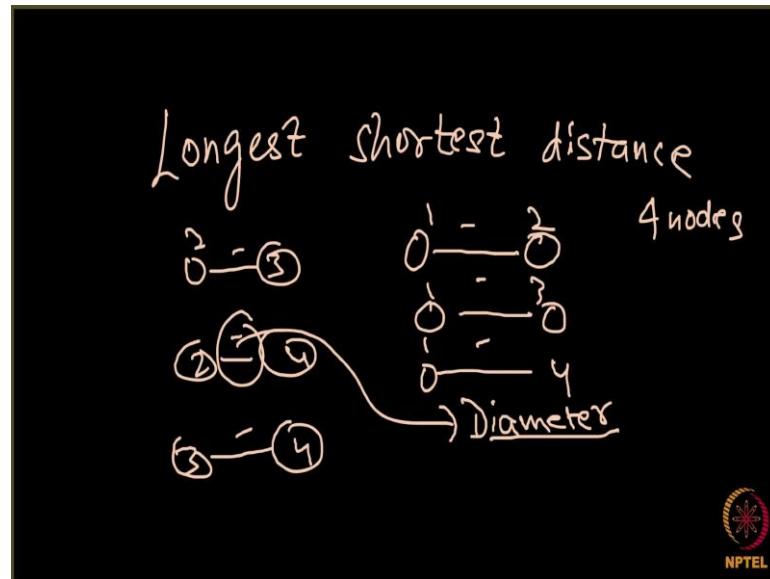
(Refer Slide Time: 10:37)



So, we have this ring network. And on this ring network, every node is connected to 1 node on the left side sorry 2 nodes on the left side and 2 nodes on the right side and so on. So, every node is following the same principle and at the end what we are going to do is we will add some extra edges there. And we will look at as we add these extra edges over here, how does the diameter of this network reduce.

And I hope will you remember; what is the diameter of a network.

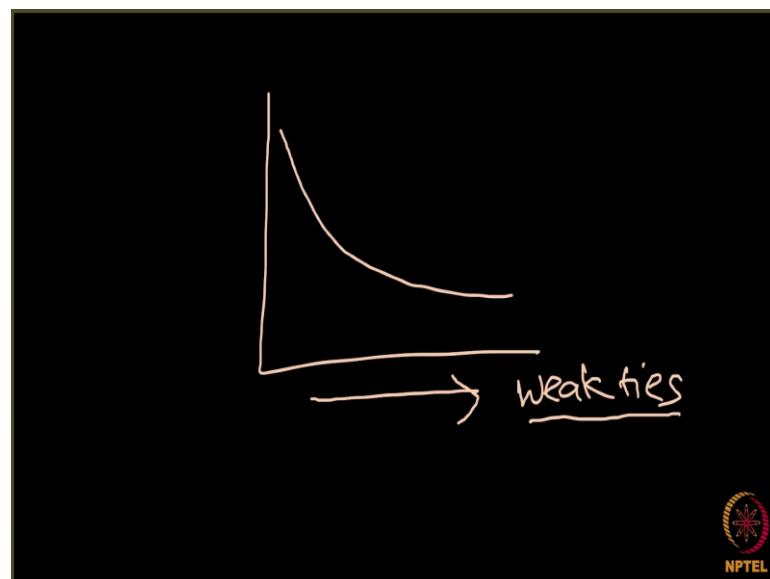
(Refer Slide Time: 11:09)



Diameter of a network is nothing but the longest, shortest, distance; longest shortest distance. So, you take every possible pair of nodes in the network, you find the distance between them distance between them. So, let us say there are some 4 nodes in this network. So, you take 1, 2, 1 3, 1 4 and then and then you can take 2 3, 2 4 and then you have 3 4. And you look at their distance right the shortest distance between these two nodes and the maximum over here is your diameter.

So, we have discussed it previously, I will not going to the details of this.

(Refer Slide Time: 11:57)



So, overall the first part of this programming screencast is going to deal with when we make a small world network as we increase the number of extra edges; as we increase the number of weak ties or the short range or the long range contacts how is this diameter going to reduce.

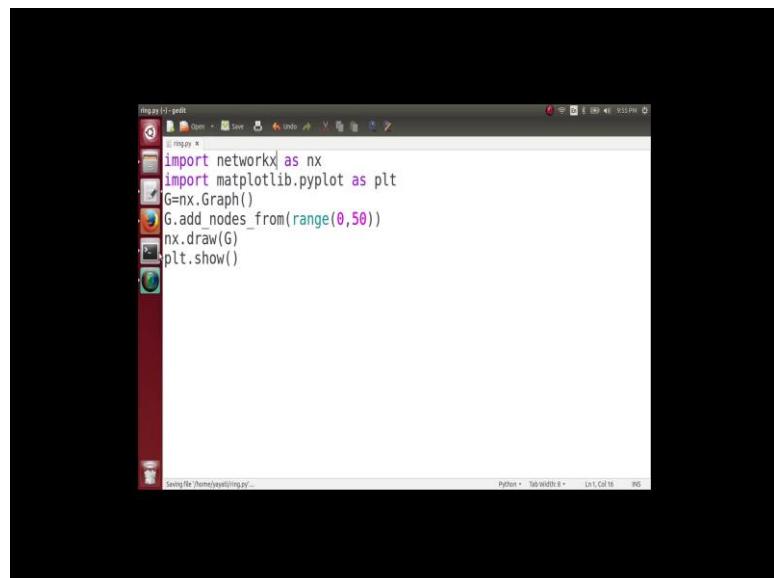
**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**How to go Viral on Web**  
**Lecture - 151**  
**Base Code**

So, now we are staring off with the programming screen guard for creating a small world network. What we are going to do is as we discussed we are going to create a one dimensional small world network in the form of a ring. Here every node is connected to 2 nodes on the left side and 2 nodes on the right side and there are certain weak ties in between. And then we will look at how does the diameter of this network reduces as we add more and more numbers of weak ties more and more numbers of long range contacts.

So, first of all what do we need is we need a network having 50 nodes.

(Refer Slide Time: 00:43)



A screenshot of a terminal window titled "ipython3-0" showing Python code. The code imports networkx as nx and matplotlib.pyplot as plt, creates a graph G, adds nodes from range(0,50), and draws the graph. The terminal window has a dark background and light-colored text. The code is as follows:

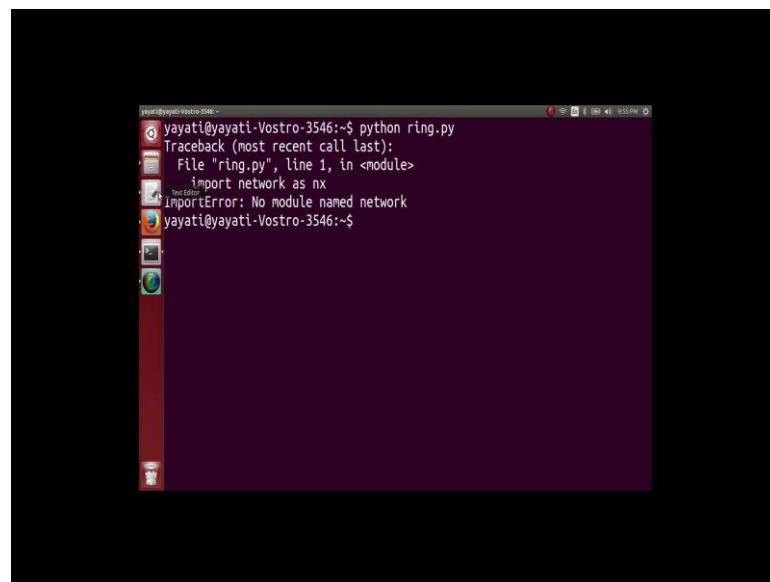
```
ipython3-0:~$ ipython3
In [1]: import networkx as nx
In [2]: import matplotlib.pyplot as plt
In [3]: G=nx.Graph()
In [4]: G.add_nodes_from(range(0,50))
In [5]: nx.draw(G)
In [6]: plt.show()
```

So, we know the code for that is very simple import networkx as nx after importing the module. So, we will like to visualize this graph so, we import matplotlib.pyplot as plt and then G = nx.Graph for creating a graph.

And then what we want to do is we want to add 50 nodes to this network. So, what do we do G.add\_nodes\_from and inside this we pass a list having numbers from 0 to 49? So, we have added the nodes, and then we can simply visualize this graph and nx.draw(G) and then plt. show.

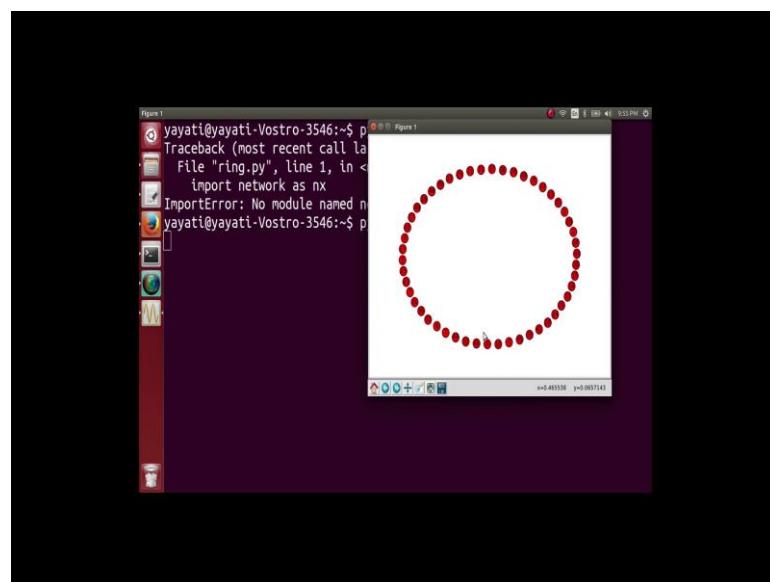
So, I am going to name this file as ring. py and let us execute and see.

(Refer Slide Time: 01:41)



So, python ring.py ok. So, import network x as nx.

(Refer Slide Time: 01:57)

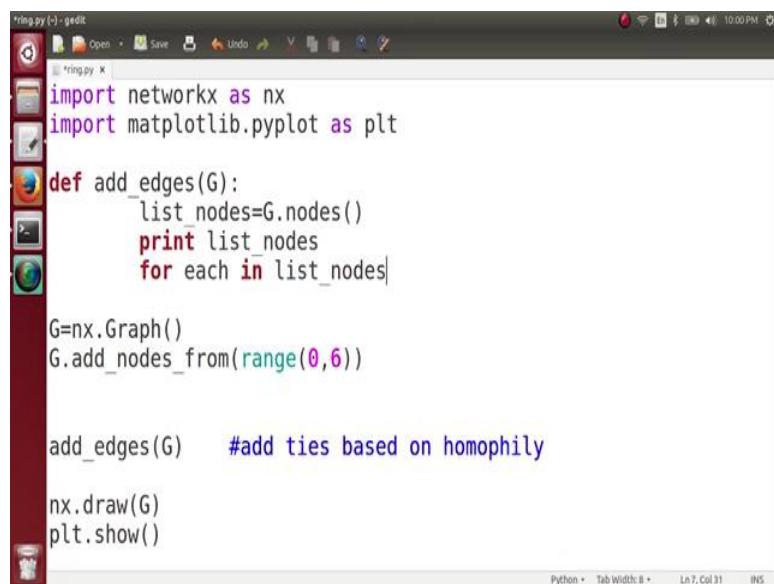


And then you can see that here, we have got a simple graph having 50 nodes label from 0 to I think that the maximum should be 49; labeled from 0 to 49 here. And you can see that the nodes here are noting a particular order. So, this is a graph which we are getting.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**How to go Viral on Web**  
**Lecture - 152**  
**Making homophily based edges**

(Refer Slide Time: 00:07)



```
string.py (~) - gedit
import networkx as nx
import matplotlib.pyplot as plt

def add_edges(G):
    list_nodes=G.nodes()
    print list_nodes
    for each in list_nodes:

        G=nx.Graph()
        G.add_nodes_from(range(0,6))

        add_edges(G)      #add ties based on homophily

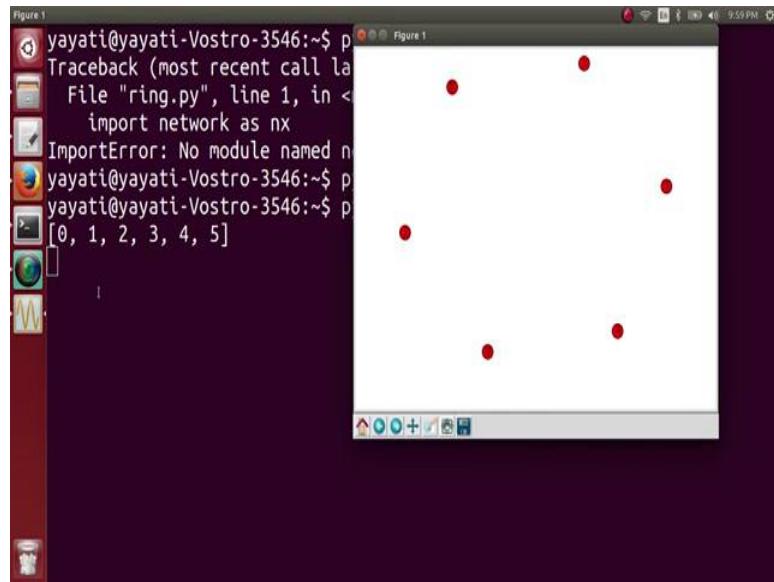
        nx.draw(G)
        plt.show()
```

Next what we are going to do is, we want to add the edges corresponding to the homophily on this network and we have seen that according to the homophily, we are going to connect each node to 2 nodes towards its left and 2 nodes towards its right. And, we have also seen that when we were portraying this network, drawing this network the nodes were not in any particular order. So, for the time being to test the validity of the code I will simply change the number of nodes to 6.

So, we will be having 6 nodes number from 0 to 5 and next what I am going to do is I have to add homophily based edges to it. So, I call a function `add_edges(G)` here. So, what this function is going to do is add ties based on homophily to this network and how do I do that. So, that is simple. So, what I have to do? I have to connect node 0 to 2 nodes towards its right which are 1 and 2 and 2 nodes towards its left you can see that those 2 nodes are node minus 1 and minus 2, rather goes nodes are this node number 6 here and that node number 5. So, how do we do that?

So, what we can simply do is we obtain a list of a. So, let me define the function here, define add edges and I will pass the graph G here and what I do is I obtain a list of nodes here; list of nodes equals to G.node which are simply the list\_nodes equals to G.nodes which are simply the nodes in this graph G. And, we can actually print these list\_nodes and see what it is giving to us.

(Refer Slide Time: 02:03)



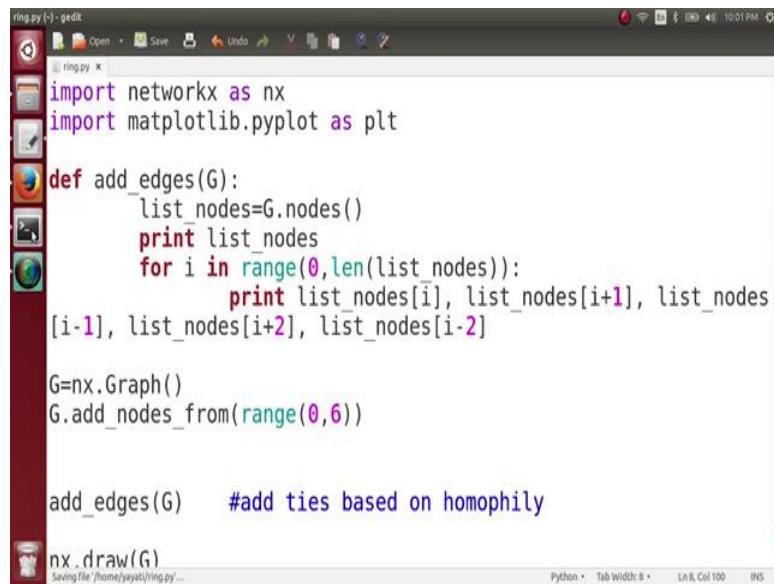
The screenshot shows a terminal window titled 'Figure 1' with a dark red background. On the left, there is a vertical docked panel containing icons for various applications like a file manager, terminal, and browser. The main terminal area displays the following text:

```
yayati@yayati-Vostro-3546:~$ p
Traceback (most recent call last):
  File "ring.py", line 1, in <
    import networkx as nx
ImportError: No module named networkx
yayati@yayati-Vostro-3546:~$ p
yayati@yayati-Vostro-3546:~$ p
[0, 1, 2, 3, 4, 5]
```

To the right of the terminal, a small window titled 'Figure 1' is open, displaying a visualization of five red circular nodes arranged in a ring-like pattern on a white background. Below the terminal, the desktop environment's taskbar is visible with several icons.

So, when I execute this code, we can see that we got here 5 nodes 0, 1, 2, 3, 4, 5 well and fine. Now, what we have to do is ok, we have to connect each node. So, we have to take every element in this list\_nodes and every element is to be connected to 2 elements towards its left and 2 elements towards its right.

(Refer Slide Time: 02:39)



```
ring.py (~) - gedit
import networkx as nx
import matplotlib.pyplot as plt

def add_edges(G):
    list_nodes=G.nodes()
    print list_nodes
    for i in range(0,len(list_nodes)):
        print list_nodes[i], list_nodes[i+1], list_nodes[i-1], list_nodes[i+2], list_nodes[i-2]

G=nx.Graph()
G.add_nodes_from(range(0,6))

add_edges(G)      #add ties based on homophily

nx.draw(G)
Saving file /home/yayati/ring.py...
```

So, how do we do that is for each in list\_nodes, what do I want is I will print each and rather now each in list.node for i a for i in range. So, we need to index in this is here for i in range 0 to length of list\_nodes ok. We are going to print, this we are going to print list\_nodes[i] right and where this list\_nodes[i] should be connected to is list\_nodes[i + 1], list\_nodes[i – 1], list\_nodes[i + 2] and list\_nodes[i – 2] right. So, this thing is very clear. So, we have seen previously whenever the in this is of a list goes in minus it starts back from 0. So, let us see whether it is working or not.

(Refer Slide Time: 03:47)



```
yayati@yayati-Vostro-3546: ~
import network as nx
ImportError: No module named network
yayati@yayati-Vostro-3546: ~$ python ring.py
yayati@yayati-Vostro-3546: ~$ python ring.py
[0, 1, 2, 3, 4, 5]
yayati@yayati-Vostro-3546: ~$ python ring.py
[0, 1, 2, 3, 4, 5]
0 1 5 2 4
1 2 0 3 5
2 3 1 4 0
3 4 2 5 1
4 5 3
Traceback (most recent call last):
  File "ring.py", line 14, in <module>
    add_edges(G)          #add ties based on homophily
  File "ring.py", line 8, in add_edges
    print list_nodes[i], list_nodes[i+1], list_nodes[i-1], list_nodes[i+2], list_nodes[i-2]
IndexError: list index out of range
yayati@yayati-Vostro-3546: ~$
```

So, let us see what is it printing. So, you can see here 0 gets connected to 1 5 2 and 4, 1 gets connected to 2 0 3 and 5, 2 gets connected to 3 4 1 and 0, 3 gets connected to 4 5 2 and 1. Little bit problem with 4, it gets connected to 5 3 and then 4 gets connected to 5 and then 3 and then 5 3 and then 4 plus 2 is 6 ok. So, you so, you note the problem here. So, here when the index of a list whenever the index of a list goes in minus there is no problem, it starts up from the last element, but there is no element named as `list_nodes[i + 2]`.

So, what we can do here is little bit of manipulation is required here. So, `list_nodes[i]` we can use and there can be a little bit problem with this `i + 1` and `i + 2`, `i - 1` and `i - 2` are perfectly fine. So, with `i + 1` and `+ 2` there is a little bit of problem. So, we will just resolve that ok. How do we resolve that? When will `i + 1` first problem, you can say that `i + 1` will create problem when you go to this element.

(Refer Slide Time: 05:31)

```

string.py (-) - gedit
File Open Save Undo Redo Find Replace Print
string.py x
import networkx as nx
import matplotlib.pyplot as plt

def add_edges(G):
    list_nodes=G.nodes()
    print list_nodes
    for i in range(0,len(list_nodes)):
        print list_nodes[i], list_nodes[i+1], list_nodes[i-1], list_nodes[i+2], list_nodes[i-2]
        # i+1-> n, we need to set this value to |

G=nx.Graph()
G.add_nodes_from(range(0,6))

add_edges(G)      #add ties based on homophily

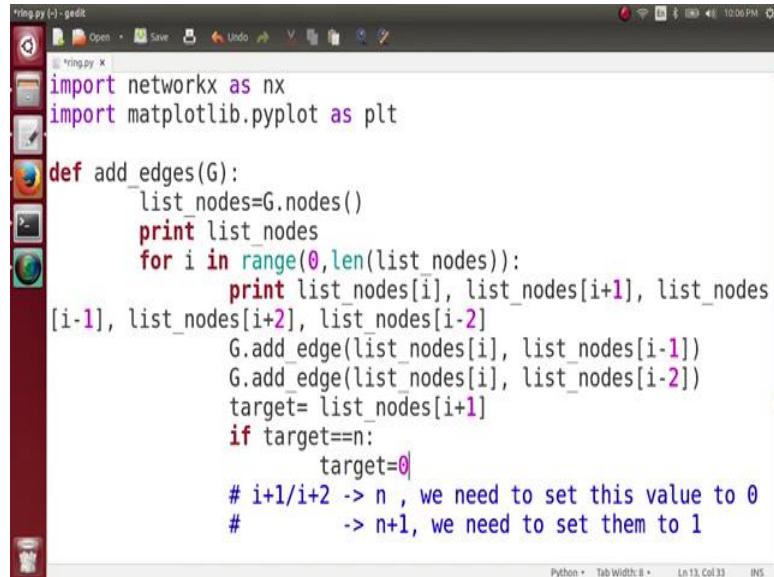
```

So, the last element of my list will be `n - 1`, where `n` is the number of nodes. The last element is `n` minus 1 and it will create problem when it has to connect to `list_nodes[5 + 1]` which is `n`. And, there is no list notes of `i + 1` because the length of the list is total length. So, we cannot access this element here. So, we can simply put a loop here.

So, what we are going to do is whenever are and then also now this one thing here before doing that `list_nodes[i]` is actually nothing, but `i` right `list_nodes` of 0 is 0 `list_nodes[1]` is 1 and so on. So, whenever my `list_nodes[i + 2]`. So, whenever `i + 1` has a danger,

whenever  $i + 1$  has a danger of becoming  $n$ , whenever  $i + 1$  has a danger of becoming  $n$  we need to set this value to what we go back from the starting.

(Refer Slide Time: 06:57)



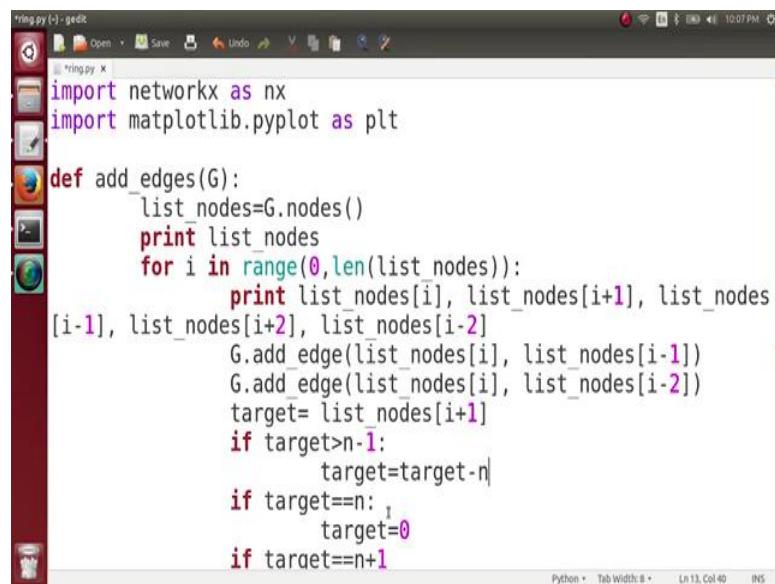
```
ring.py (~) - gedit
import networkx as nx
import matplotlib.pyplot as plt

def add_edges(G):
    list_nodes=G.nodes()
    print list_nodes
    for i in range(0,len(list_nodes)):
        print list_nodes[i], list_nodes[i+1], list_nodes[i-1], list_nodes[i+2], list_nodes[i-2]
        G.add_edge(list_nodes[i], list_nodes[i+1])
        G.add_edge(list_nodes[i], list_nodes[i-1])
        target= list_nodes[i+1]
        if target==n:
            target=0
        # i+1/i+2 -> n , we need to set this value to 0
        #           -> n+1, we need to set them to 1
```

So, whenever it is  $n$  it has to be set back to 0 and whenever  $i + 2$  it tends to become ok, it can be  $n$  or whenever  $i + 1$  or it can also be  $i + 2$  whenever they tend to become  $n$  we tend to set that value to 0. And, whenever these two values they tend to become  $n + 1$  we need to set them to 1, who let us quickly see how we can set it. So, we can know that this node  $list\_nodes[i]$  has to be connected to  $list\_nodes[i + 1]$ ,  $[i + 2]$ ,  $[i - 1]$  and  $[i - 2]$ ,  $[i - 1]$  and  $[i - 2]$  are perfectly fine.

So, there can be a problem when we are connecting them to  $i + 1$  and  $i + 2$ . So, what we are going to do is, what we have to actually do is we have to  $G.add\_edge$ . And, from where to we have to  $add\_edge(list\_nodes[i], list\_nodes[i - 1])$  can be simply added and  $list\_nodes[i]$  2 nodes of  $i - 2$  can be simply added. And, now what we are going to do is let us say target list node of  $i + 1$  and what we are going to do is, if target equals to equals to  $n$  what we will do is target equals to 0 and rather ok.

(Refer Slide Time: 09:13)



```
ring.py (-) - gedit
import networkx as nx
import matplotlib.pyplot as plt

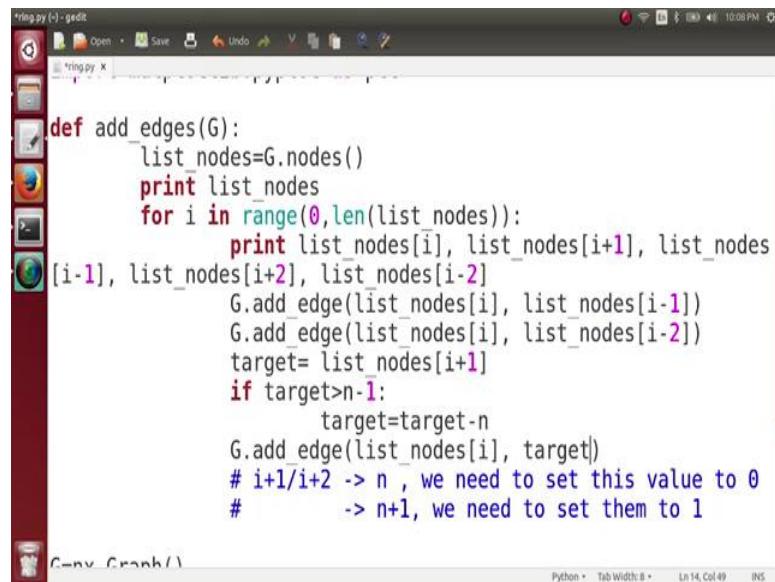
def add_edges(G):
    list_nodes=G.nodes()
    print list_nodes
    for i in range(0,len(list_nodes)):
        print list_nodes[i], list_nodes[i+1], list_nodes[i-1], list_nodes[i+2], list_nodes[i-2]
        G.add_edge(list_nodes[i], list_nodes[i+1])
        G.add_edge(list_nodes[i], list_nodes[i-2])
        target= list_nodes[i+1]
        if target>n-1:
            target=target-n
        if target==n:
            target=0
        if target==n+1
```

The screenshot shows a terminal window titled "ring.py (-) - gedit". The code in the window is for generating a ring graph. It imports networkx and matplotlib.pyplot, defines a function add\_edges that takes a graph G, prints its nodes, and then iterates through them. For each node i, it prints five nodes: i, i+1, i-1, i+2, and i-2. It then adds edges between node i and i+1, and between node i and i-2. The variable target is set to list\_nodes[i+1]. If target is greater than n-1, it is adjusted by subtracting n. If target is equal to n, it is set to 0. If target is equal to n+1, it is set to 1. The status bar at the bottom right indicates "Python • Tab Width: 8 • Ln 13, Col 40 INS".

And then if target equals to equals to  $n + 1$ , what we are going to do is target equals to 1; we can simply it. So, we can write both of these statements as 1 statement; what can that be? We can write if target is greater than  $n - 1$ , what we are going to set the target to be? So, target is going to be set to, then I write target equals to  $n - target$ . So, if target is greater than  $n - 1$ , now if target is  $n$  it should be set to 0 and if target is  $n + 1$  it should be set to 1.

So, target is nothing, but target minus  $n$ . So, can I do target equals to target minus  $n$ , what will we do if target is  $n$  it will set target is 0 and if target is  $n + 1$  target will be set to 1 ok. So, target equals to  $list\_nodes[i + 1]$  and then this happens.

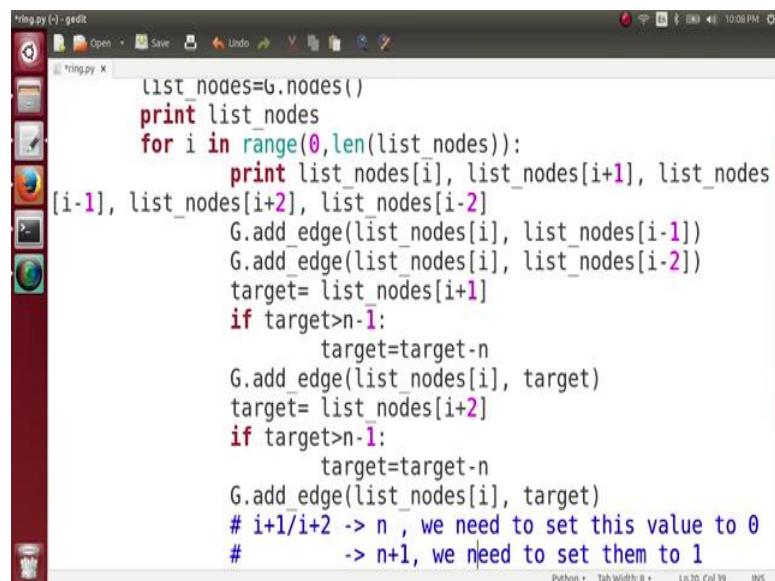
(Refer Slide Time: 10:27)



```
ring.py (-) - gedit
def add_edges(G):
    list_nodes=G.nodes()
    print list_nodes
    for i in range(0,len(list_nodes)):
        print list_nodes[i], list_nodes[i+1], list_nodes[i-1], list_nodes[i+2], list_nodes[i-2]
        G.add_edge(list_nodes[i], list_nodes[i-1])
        G.add_edge(list_nodes[i], list_nodes[i-2])
        target= list_nodes[i+1]
        if target>n-1:
            target=target-n
        G.add_edge(list_nodes[i], target)
        # i+1/i+2 -> n , we need to set this value to 0
        #           -> n+1, we need to set them to 1
C-ny_Craph()
```

And then what I can simply do is `G.add_edge` from `list_nodes[i]` to `target` right and now this target can be `list_nodes[i + 1]` and this target is also going to be `list_nodes[i + 2]`.

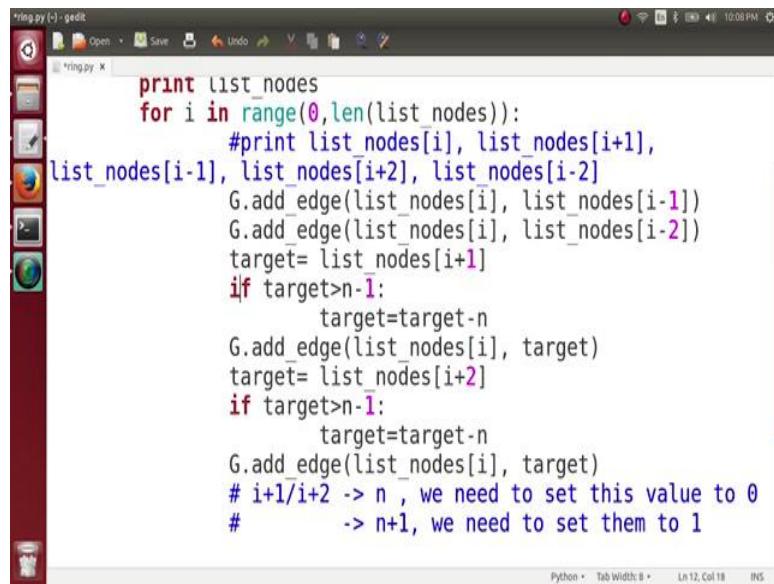
(Refer Slide Time: 10:53)



```
ring.py (-) - gedit
list_nodes=G.nodes()
print list_nodes
for i in range(0,len(list_nodes)):
    print list_nodes[i], list_nodes[i+1], list_nodes[i-1], list_nodes[i+2], list_nodes[i-2]
    G.add_edge(list_nodes[i], list_nodes[i-1])
    G.add_edge(list_nodes[i], list_nodes[i-2])
    target= list_nodes[i+1]
    if target>n-1:
        target=target-n
    G.add_edge(list_nodes[i], target)
    target= list_nodes[i+2]
    if target>n-1:
        target=target-n
    G.add_edge(list_nodes[i], target)
    # i+1/i+2 -> n , we need to set this value to 0
    #           -> n+1, we need to set them to 1
C-ny_Craph()
```

So, essentially going to replace this whole thing with `if target equals to list_node[i] 2` and then this code remains the same and, then `G.add_edge(list_nodes[i], target)` right and then we can comment this statement.

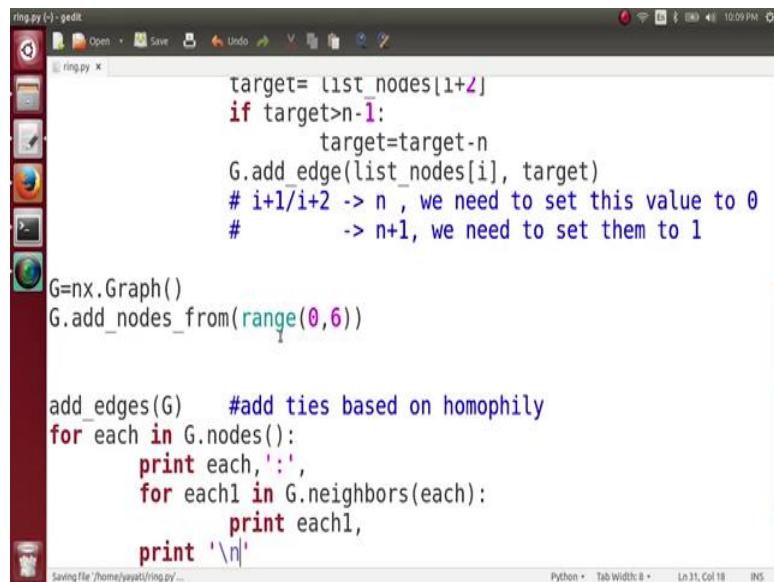
(Refer Slide Time: 11:09)



```
ring.py (-) - gedit
print list_nodes
for i in range(0,len(list_nodes)):
    #print list_nodes[i], list_nodes[i+1],
list_nodes[i-1], list_nodes[i+2], list_nodes[i-2]
    G.add_edge(list_nodes[i], list_nodes[i-1])
    G.add_edge(list_nodes[i], list_nodes[i-2])
    target= list_nodes[i+1]
    if target>n-1:
        target=target-n
    G.add_edge(list_nodes[i], target)
    target= list_nodes[i+2]
    if target>n-1:
        target=target-n
    G.add_edge(list_nodes[i], target)
# i+1/i+2 -> n , we need to set this value to 0
#           -> n+1, we need to set them to 1
```

Now, to see whether this code is working fine or not, what I am going to do here is I am going to look at the neighbors of each nodes.

(Refer Slide Time: 11:13)



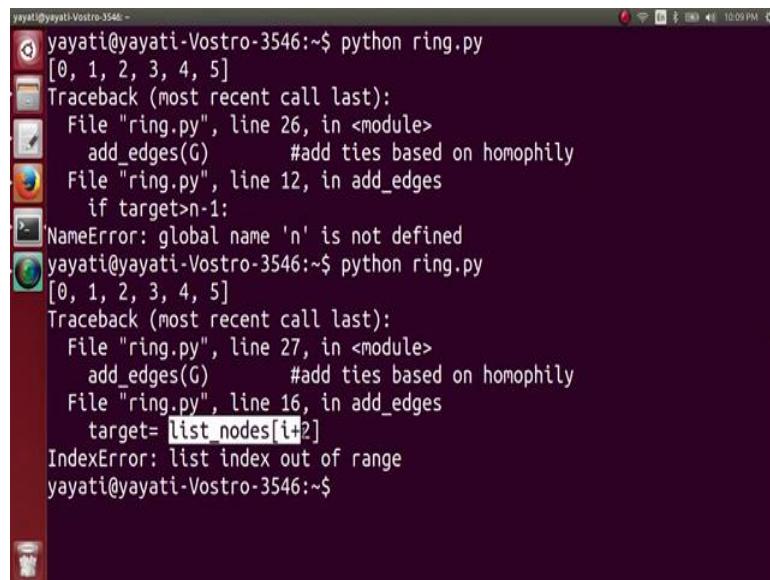
```
ring.py (-) - gedit
target= list_nodes[1+2]
if target>n-1:
    target=target-n
G.add_edge(list_nodes[i], target)
# i+1/i+2 -> n , we need to set this value to 0
#           -> n+1, we need to set them to 1

G=nx.Graph()
G.add_nodes_from(range(0,6))

add_edges(G)      #add ties based on homophily
for each in G.nodes():
    print each,':',
    for each1 in G.neighbors(each):
        print each1,
    print '\n'
```

So, for each in G.nodes, I am going to print each comma and then for each 1 in G.neighbors I am going to look at all the neighbors of this node each here. And, I am going to print this neighbors one by one, print each1 then I print the new line here; let us now execute this code and see ok.

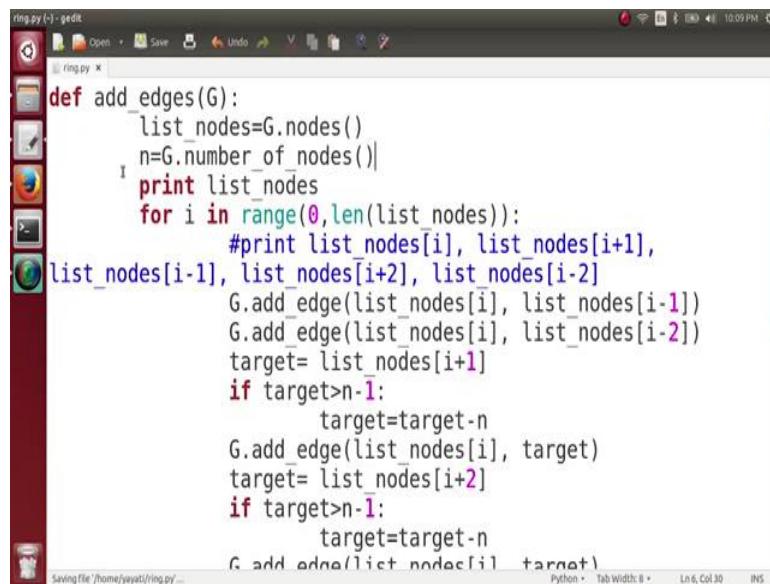
(Refer Slide Time: 11:53)



```
yayati@yayati-Vostro-3546:~$ python ring.py
[0, 1, 2, 3, 4, 5]
Traceback (most recent call last):
  File "ring.py", line 26, in <module>
    add_edges(G)          #add ties based on homophily
  File "ring.py", line 12, in add_edges
    if target>n-1:
NameError: global name 'n' is not defined
yayati@yayati-Vostro-3546:~$ python ring.py
[0, 1, 2, 3, 4, 5]
Traceback (most recent call last):
  File "ring.py", line 27, in <module>
    add_edges(G)          #add ties based on homophily
  File "ring.py", line 16, in add_edges
    target= list_nodes[i+2]
IndexError: list index out of range
yayati@yayati-Vostro-3546:~$
```

So, we get an error name n is not define. So, we have again and again used n here and we have not define what is n. So, it is easy what is n as we have discuss, n is nothing but the number of nodes in the network.

(Refer Slide Time: 12:09)



```
ring.py - gedit
def add_edges(G):
    list_nodes=G.nodes()
    n=G.number_of_nodes()
    print list_nodes
    for i in range(0,len(list_nodes)):
        #print list_nodes[i], list_nodes[i+1],
list_nodes[i-1], list_nodes[i+2], list_nodes[i-2]
        G.add_edge(list_nodes[i], list_nodes[i-1])
        G.add_edge(list_nodes[i], list_nodes[i-2])
        target= list_nodes[i+1]
        if target>n-1:
            target=target-n
        G.add_edge(list_nodes[i], target)
        target= list_nodes[i+2]
        if target>n-1:
            target=target-n
        G.add_edge(list_nodes[i], target)

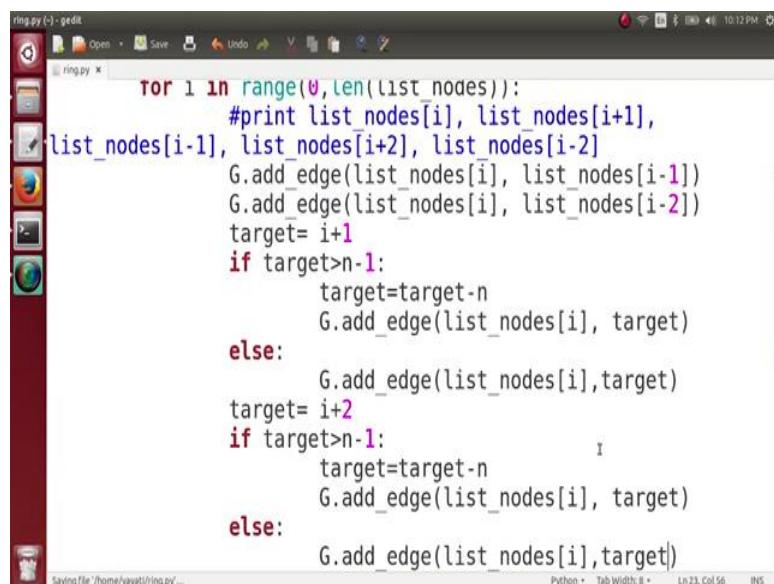
Saving file /home/yayati/ring.py ...
Python Tab Width: 8 Ln 6 Col 30 INS
```

So, n is G.number\_of\_nodes ok, here is again some problem target equals to list\_nodes[i + 2] line number 16 ok. So, what is going to happen here is when we obtain the target equals to list\_node[i + 1] just simply showing as an error here. Now, what can we do

here is will be here is remover. So, we will node add list\_nodes[i] is namely nothing, but i until and unless this value here becomes in negative.

So, list\_nodes[i - 1] did not be  $i - 1$ , it can go back and start from the last element in list, but list\_nodes[i] is essentially going to be  $i$ . Now, what I can do here is instead of list\_nodes[i + 1]. So, the maximum index in this list is  $n$  sorry  $n - 1$ . So, if it the here becomes the value  $n$  so, it cannot determine the value for list underscore nodes  $n$ . So, what I am going to do here is target =  $i + 1$ .

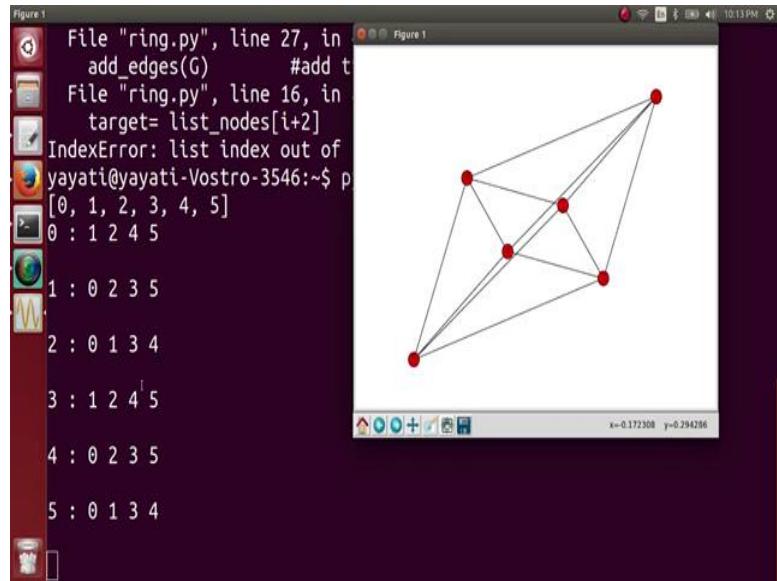
(Refer Slide Time: 13:39)



```
ring.py (~) - gedit
ring.py x
for i in range(0, len(list_nodes)):
    #print list_nodes[i], list_nodes[i+1],
    list_nodes[i-1], list_nodes[i+2], list_nodes[i-2]
    G.add_edge(list_nodes[i], list_nodes[i-1])
    G.add_edge(list_nodes[i], list_nodes[i-2])
    target= i+1
    if target>n-1:
        target=target-n
        G.add_edge(list_nodes[i], target)
    else:
        G.add_edge(list_nodes[i],target)
    target= i+2
    if target>n-1:
        target=target-n
        G.add_edge(list_nodes[i], target)
    else:
        G.add_edge(list_nodes[i],target)
Saving file /home/yayati/ring.py ...
Python Tab Width: 8 Line 23, Col 56 INS
```

So, if target =  $i + 1$ . So, now, if target is greater than  $n - 1$  we are going to change the target to target minus  $n$  and then we are going to add the edge. Otherwise what we are going to do is, else G.add\_edge otherwise there was no problem and we can simply add an edge from list\_nodes[i] to target. And, similarly here so, what I am going to do here is target =  $i + 2$  and if target is greater than  $n - 1$ , this things happens, else we can simply put an edge from list\_nodes[i] to target without changing target. So, that has now run this code and see ok.

(Refer Slide Time: 14:51)



So, we say here we got a graph here 0 is connected to 1 2 and then back towards 4 and 5, 1 is connected to 2 3 0 and 5, 2 is connected to 0 1 3 4, 3 is connected to 1 4 3 is connected to 1 2 4 5, 4 is connected to ok; little bit problem with 4 here. 4 should be connected to 3 2 0 and 5 right, yeah 4 is connected to 3 2 0 and 5 perfectly fine. And, 5 should be connected to 2 towards its left which are 3 and 4, 2 towards 6 right which are 0 and 1. So, this is perfectly fine. So, we have created the homophily based links.

(Refer Slide Time: 15:43)

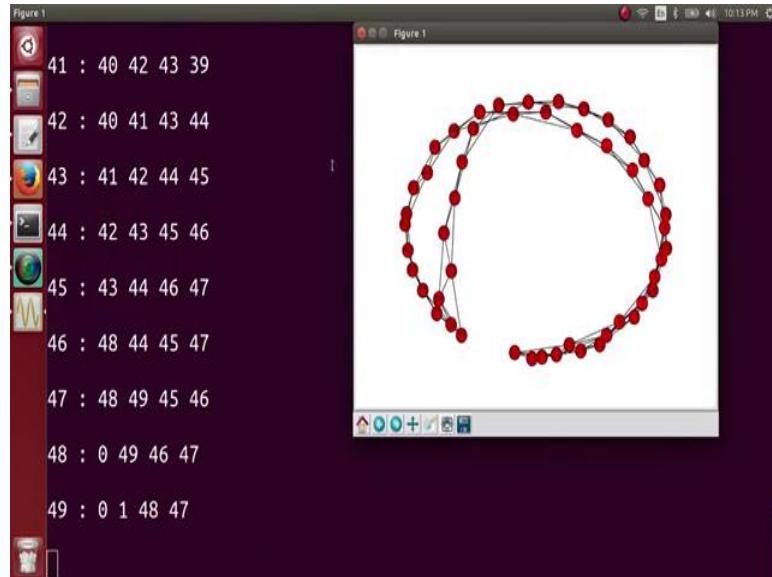
The figure shows a gedit editor window with the file 'ring.py' open. The code is as follows:

```
G=nx.Graph()
G.add_nodes_from(range(0,50))

add_edges(G)      #add ties based on homophily
for each in G.nodes():
    print each,':',
    for each1 in G.neighbors(each):
        print each1,
    print '\n'
```

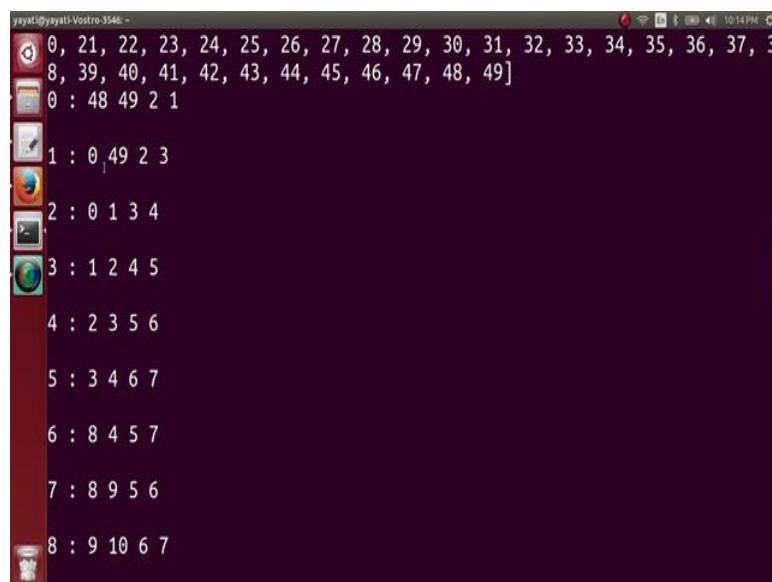
And, what we can do is we wanted a graph here for 0 to, let us say we want 50 nodes here. So, we I will make it 50 here and then we can execute it and see.

(Refer Slide Time: 15:47)



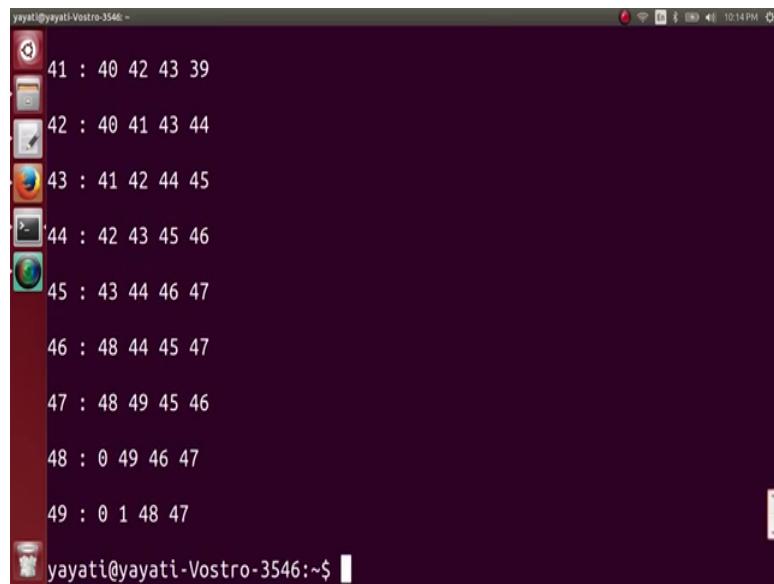
It might look here that this graph is ok, you cannot see a complete cycle here, but you can test the edges here and ensure that they all are correct.

(Refer Slide Time: 15:57)



So, you can see that 0, 1 is 0 is connected to 1 to 48 and 49, 1 is connected to 2 3 0 and 49, 2 is connected to 0 1 3 4.

(Refer Slide Time: 16:11)



A screenshot of a Linux terminal window titled "yayati@yayati-Vostro-3546:~". The window displays a list of nodes (41 to 49) and their corresponding connections. The connections are as follows:

- 41 : 40 42 43 39
- 42 : 40 41 43 44
- 43 : 41 42 44 45
- 44 : 42 43 45 46
- 45 : 43 44 46 47
- 46 : 48 44 45 47
- 47 : 48 49 45 46
- 48 : 0 49 46 47
- 49 : 0 1 48 47

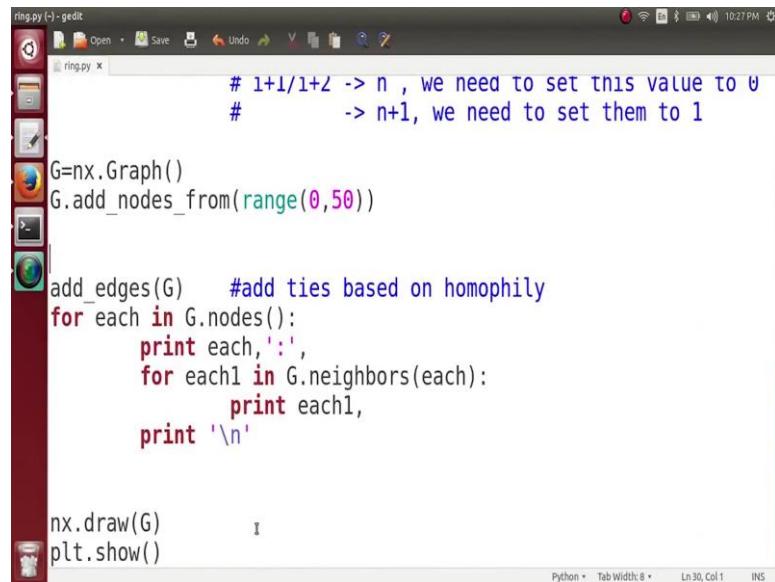
The terminal prompt at the bottom is "yayati@yayati-Vostro-3546:~\$".

So, you can verify all these edges. So, verifying all these edges you conform that we are getting a perfect cycle here, where every node is connected to 2 nodes towards its left and 2 nodes towards its right.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**How to Go Viral on Web**  
**Lecture - 153**  
**Adding Weak Ties**

(Refer Slide Time: 00:05)



The screenshot shows a terminal window titled "ring.py (~) - gedit". The code is as follows:

```
# 1+1/1+2 -> n , we need to set this value to 0
#           -> n+1, we need to set them to 1

G=nx.Graph()
G.add_nodes_from(range(0,50))

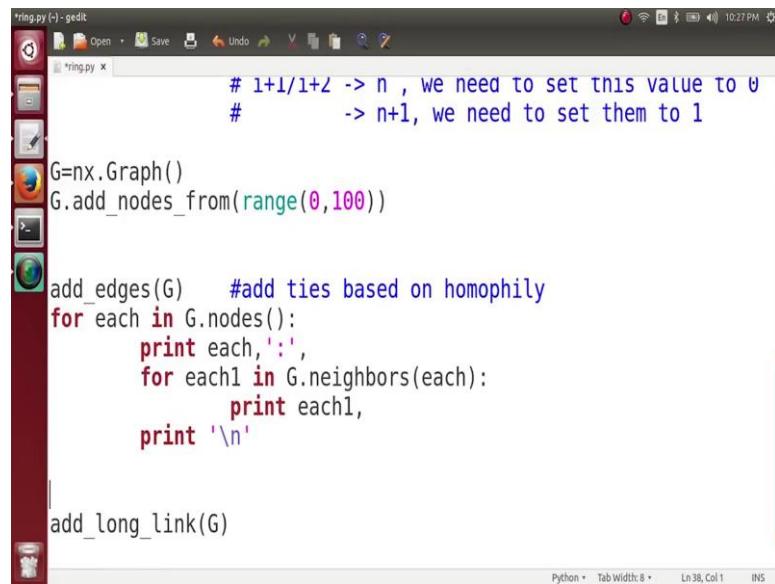
add_edges(G)      #add ties based on homophily
for each in G.nodes():
    print each,':',
    for each1 in G.neighbors(each):
        print each1,
    print '\n'

nx.draw(G)
plt.show()
```

The terminal window has a dark theme. The status bar at the bottom right shows "Python" and "Tab Width: 8".

So, now we have a network which is having 50 nodes, and which is having homophily based contexts where every node is connecting to 2 nodes towards its left and 2 nodes towards its right ok.

(Refer Slide Time: 00:23)



```
# 1+1/1+2 -> n , we need to set this value to 0
#           -> n+1, we need to set them to 1

G=nx.Graph()
G.add_nodes_from(range(0,100))

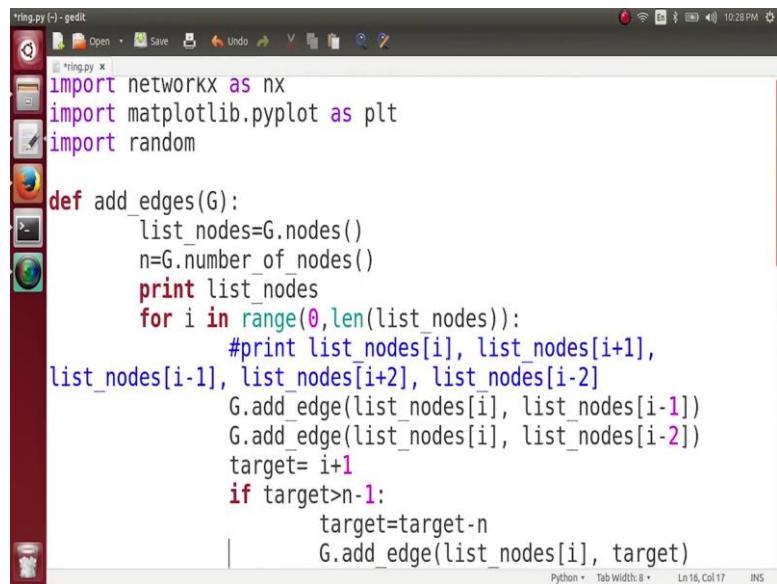
add_edges(G)      #add ties based on homophily
for each in G.nodes():
    print each,':',
    for each1 in G.neighbors(each):
        print each1,
    print '\n'

add_long_link(G)
```

Let us take 100 node instead of 50. Next what we want to do is I am going to create a simple function which is going to create a weak tie. So, when we call this function ones it creates one weak tie, we are going only to add these weak ties. And we know that this is very easy, we have done it many times before from going to take a function here let us say add long link its basically add a long range link in G and it is going to be pretty easy, isn't it?

So, what we have to do? We have to randomly pick 2 nodes in this method and put an edge between them.

(Refer Slide Time: 01:03)



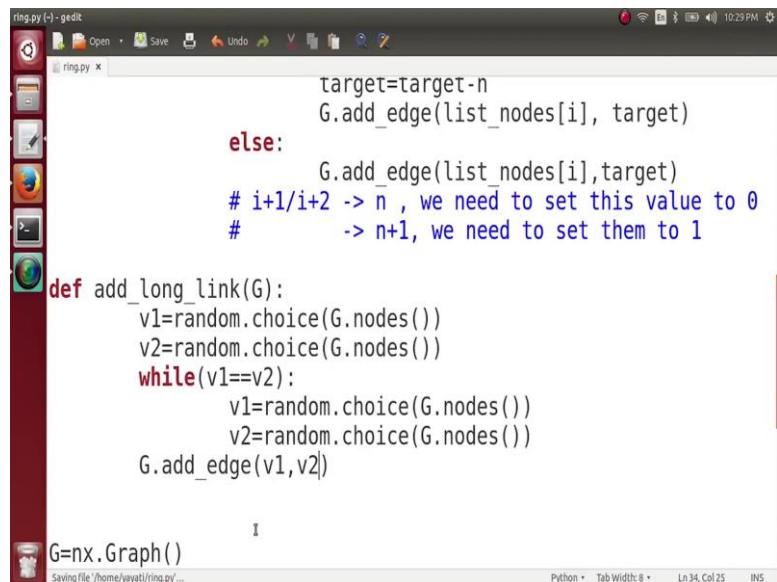
```
ring.py (-) - gedit
ring.py x
import networkx as nx
import matplotlib.pyplot as plt
import random

def add_edges(G):
    list_nodes=G.nodes()
    n=G.number_of_nodes()
    print list_nodes
    for i in range(0,len(list_nodes)):
        #print list_nodes[i], list_nodes[i+1],
        list_nodes[i-1], list_nodes[i+2], list_nodes[i-2]
        G.add_edge(list_nodes[i], list_nodes[i-1])
        G.add_edge(list_nodes[i], list_nodes[i-2])
        target= i+1
        if target>n-1:
            target=target-n
        G.add_edge(list_nodes[i], target)

Python • Tab Width: 8 • Lin 16, Col 17 INS
```

So, we can actually import the function random here.

(Refer Slide Time: 01:09)



```
ring.py (-) - gedit
ring.py x
        target=target-n
        G.add_edge(list_nodes[i], target)
    else:
        G.add_edge(list_nodes[i],target)
# i+1/i+2 -> n , we need to set this value to 0
#           -> n+1, we need to set them to 1

def add_long_link(G):
    v1=random.choice(G.nodes())
    v2=random.choice(G.nodes())
    while(v1==v2):
        v1=random.choice(G.nodes())
        v2=random.choice(G.nodes())
    G.add_edge(v1,v2)

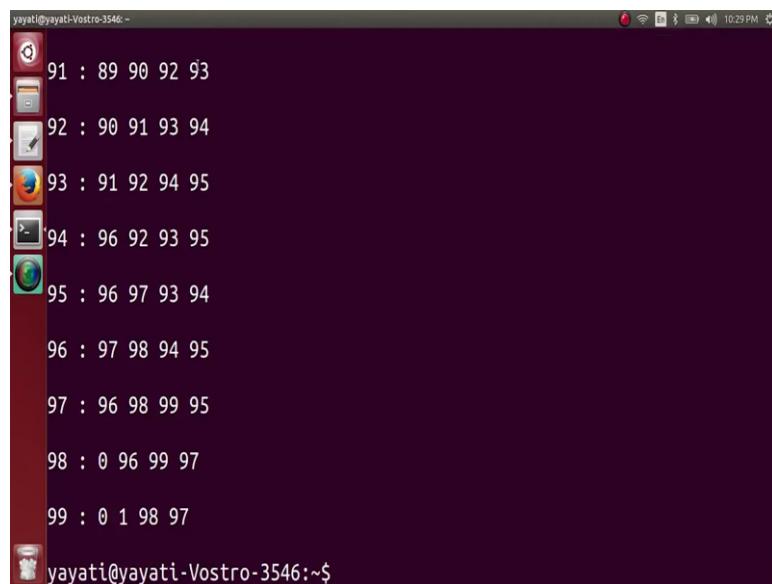
G=nx.Graph()
Saving file '/home/yayati/ring.py'...
```

And then what we have to do is define add long link G. And what we are going to do is let us take the first vertex it is nothing but random node choice for random element from the list G.nodes, and then v2, it also equal to random node choice G.nodes. And now, these two nodes can be same. We are going to put an edge between them only then these two nodes are different.

So, what I am going to do for that is why  $v_1 == v_2$ . So, if  $v$  by chance  $v_1$  become equal to  $v_2$  what I am going to do is I am going to repeat the same process again till I get to different values for  $v_1$  and  $v_2$ . And as soon as I get to different values for  $v_1$  and  $v_2$ , I put an edge between them  $G.add\_edge(v_1, v_2)$ .

So, when I call this function once add long link it adds two edges it adds one edge between 2 randomly picked word this is in my network.

(Refer Slide Time: 02:33)



The screenshot shows a terminal window on a Linux desktop. The terminal output lists nodes from 91 to 99, each followed by a colon and a list of its neighbors. The desktop environment includes a dock with icons for various applications like a browser and file manager, and a system tray at the top.

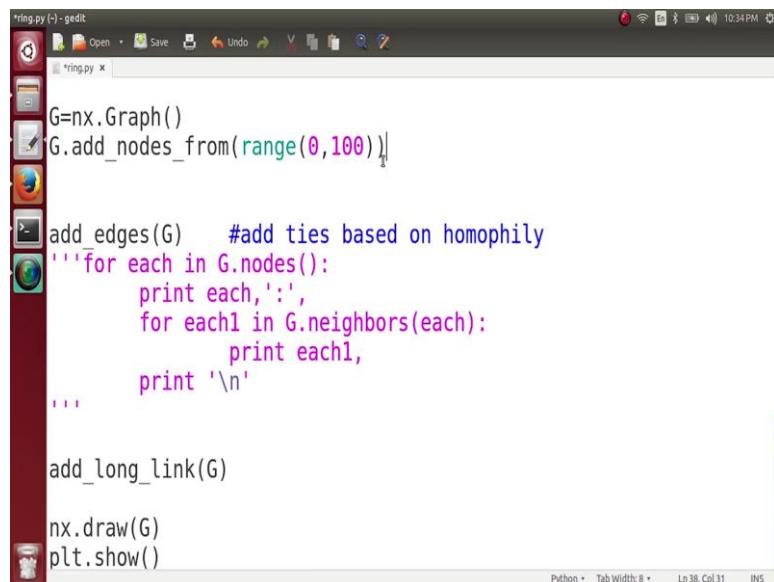
```
yayati@yayati-Vostro-3546:~$  
91 : 89 90 92 93  
92 : 90 91 93 94  
93 : 91 92 94 95  
94 : 96 92 93 95  
95 : 96 97 93 94  
96 : 97 98 94 95  
97 : 96 98 99 95  
98 : 0 96 99 97  
99 : 0 1 98 97  
yayati@yayati-Vostro-3546:~$
```

And we just check whether this code running fine or not, there is no error. So, this code is running fine.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**How to go Viral on Web**  
**Lecture - 154**  
**Plotting changes in diameter**

(Refer Slide Time: 00:05)



The screenshot shows a Gedit text editor window titled "ring.py". The code inside the editor is as follows:

```
G=nx.Graph()
G.add_nodes_from(range(0,100))

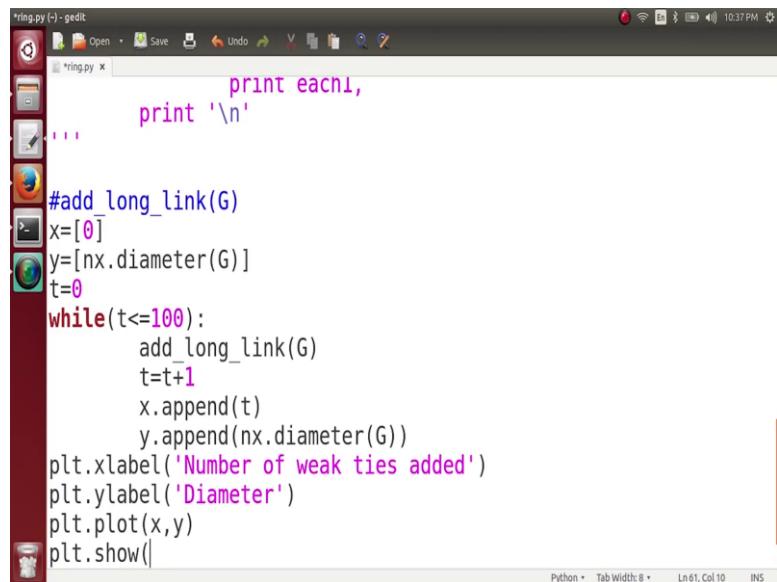
def add_edges(G)      #add ties based on homophily
    '''for each in G.nodes():
        print each,':',
        for each1 in G.neighbors(each):
            print each1,
        print '\n'
    ...

def add_long_link(G)
    nx.draw(G)
    plt.show()
```

So, what we have done till now is we have created a network having 100 nodes and, on this network, we have created edges based on homophily, that is what we can say is strong ties. Next, we want to look at we have created a function which adds 1, when we call this function once we get 1 long range link in this network. Now, what we want to do is we want to see as we add more and more number of long range ties in this network, how does the diameter of this network reduces.

And, that is easy what we want is on the x axis we want the number of links, number of long range links which have been added to the network and on the y axis we want to plot a diameter of the network. And, please note since we are only adding the edges and initially this network is connected. So, this network is always going to be connected and hence, the diameter will be a positive number which is not infinity ok.

(Refer Slide Time: 01:15)



```
tring.py (-) - gedit
tring.py X
print each1,
print '\n'

#add_long_link(G)
x=[0]
y=[nx.diameter(G)]
t=0
while(t<=100):
    add_long_link(G)
    t=t+1
    x.append(t)
    y.append(nx.diameter(G))
plt.xlabel('Number of weak ties added')
plt.ylabel('Diameter')
plt.plot(x,y)
plt.show()

Python Tab Width: 8 Ln 61, Col 10 INS
```

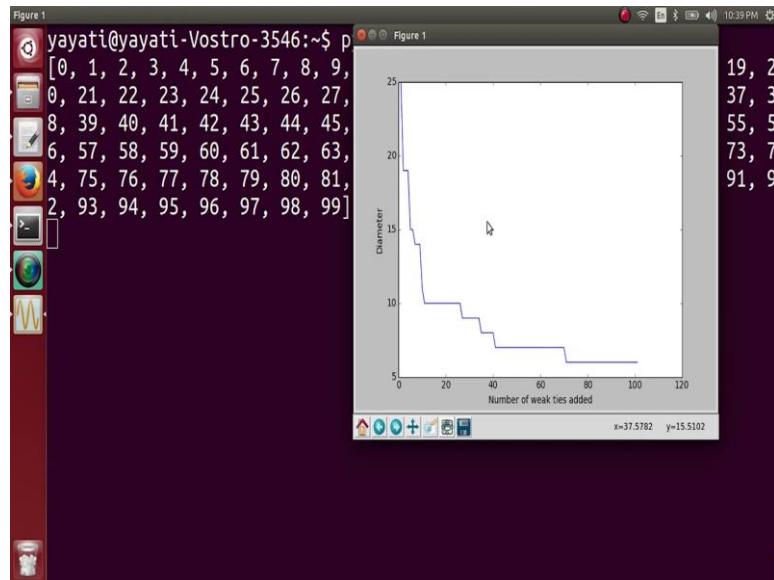
So, what I am going to do now is we need 2 axis 2 let 2 lists; one for the x axis and another for the y axis. So, this is the list for the x axis and we know that the first value for x axis will be 0; x axis shows the number of long range links added. So, when x is 0 that is when there is no long range link added in the network we know that at that time what is the diameter of my network. So, I am going to command this over here, we have added no long range tie as of now and here the value which will be appended to the y axis is nothing, but the diameter of the network. So, this is the function which tells me the diameter of the graph G.

So, we got 1 1 value for x and y and now we starts from times equals to 0 and at every time step I am going to add 1 long rang link to this network and let say we repeat this process for 100 times. So, when t is less than 100 less than or equals to 100. So, for 100 times since I am going to do this process, what I am going do is I am going to add 1 long link in this network. And, as soon as I add 1 long link in this network what I am going to do is, I am going to increment t by 1. And, then I am going to append t to y sorry t to the list x which means that 1, which means which shows me how many long-range links have been added.

So, at this time when t is 1, 1 long range tie has been added here and simply we know what has to append to the y axis is nothing, but nx.diameter(G). So, we can get here x axis and y axis and we can simply plot them. So, let me set the labels so, plt.xlabel is

nothing, but what we are having on x axis is the number of weak ties added. And, what we are having on y axis is the diameter of the network and then we are simply going to plot plt.plot(x, y) and then plt.show ok.

(Refer Slide Time: 03:49)



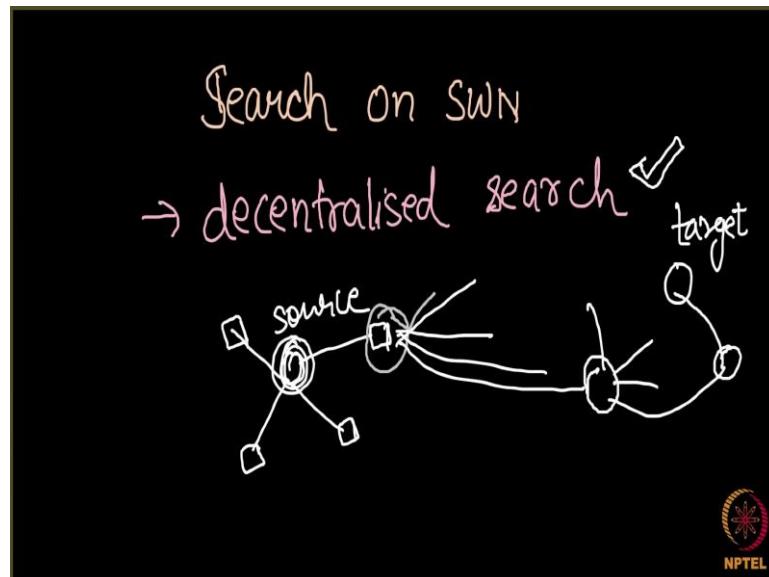
Let execute it and see ok. So, we can see here what is happening the initially you know the diameter of this network was 25. We know it is going to be 25 because, there are 100 nodes and every node is connecting to 2 nodes towards the left and 2 nodes towards the right. So, we can make a jump of 2 in we can make a jump of 2 hopes in 1 hope. So, in 25 jumps we can reach from one end of the network to another and you see that just on the addition of 1; just on the addition of 1 or 2 weak ties the diameter drastically reduces to 18. And, then and we can see that as we add more and more long range ties the diameter reduces drastically, it is not something linear. It reduces very quickly right and then the diameter of the network finally, reaches some 4 or 3 and we have added some 80 weak ties. So, this is how the curve which we wanted to look that looks like.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**How to go Viral on Web**  
**Lecture - 155**  
**Programming illustration- Myopic Search: Introduction**

The next part of our programming screencast is we will see how we do search on the small world networks, search on the small world networks.

(Refer Slide Time: 00:11)



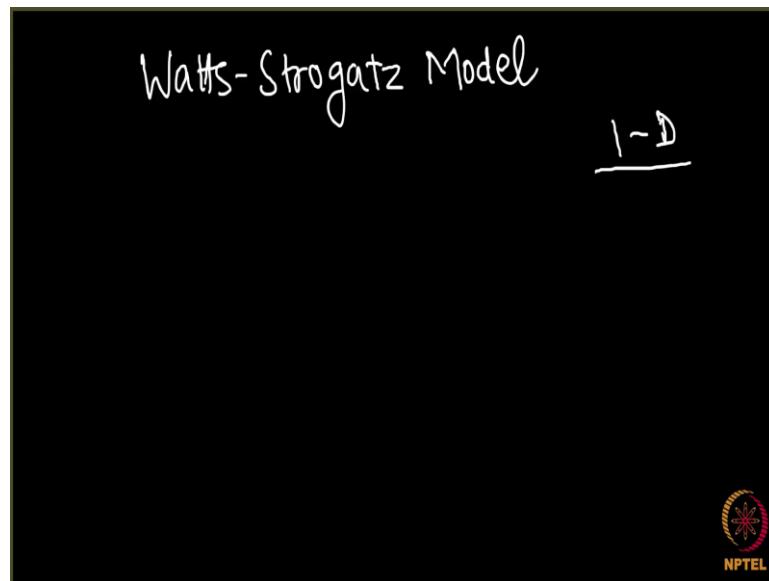
And, as we have discussed before the search on the small world networks is known as a decentralised search. Decentralised search because, it is a collaborative effort, there is no central authority no central node in the network which help you searching. And, we are have we have briefly looked at how do we do this decentralised search right.

So, how do we do this decentralised search was something like here is a node in this network and let us say this is the source node and it wants to find a path to some node, let us say the target node, let us say it wants to pass on some later do this target node. So, how this source was going to work is, it was going to look at all of its neighbours and for all of its neighbours it has an estimate of the distance of its neighbours from the target. So, the source looks at what is the distance of this neighbour from the target, what is the distance of this neighbour from this target this neighbour and this neighbour.

And it might find out that this neighbour here is closest to the target. So, it chooses this neighbour and this neighbour again here this neighbour, again here looks at all of its neighbours and chooses the one which is closest to the target. And, this one looks at all its neighbours and chooses the one which is closest to the target.

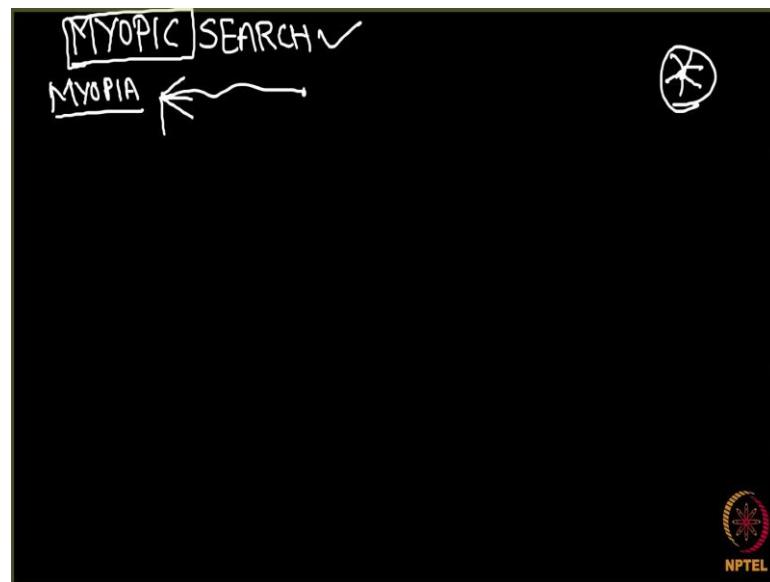
And, hence the letter in this process the letter reaches the target. Now, what we are going to do this? We are going to look at how do we do this decentralised search on a small world network, on a Watts-Strogatz model which we have studied before.

(Refer Slide Time: 01:53)



So, given a Watts-Strogatz model how do you perform a decentralised search on this model. So, let us say the model which we are talking about is that the is the ring model the one-dimensional Watts-Strogatz model and we want to see how do we perform a decentralised search over there.

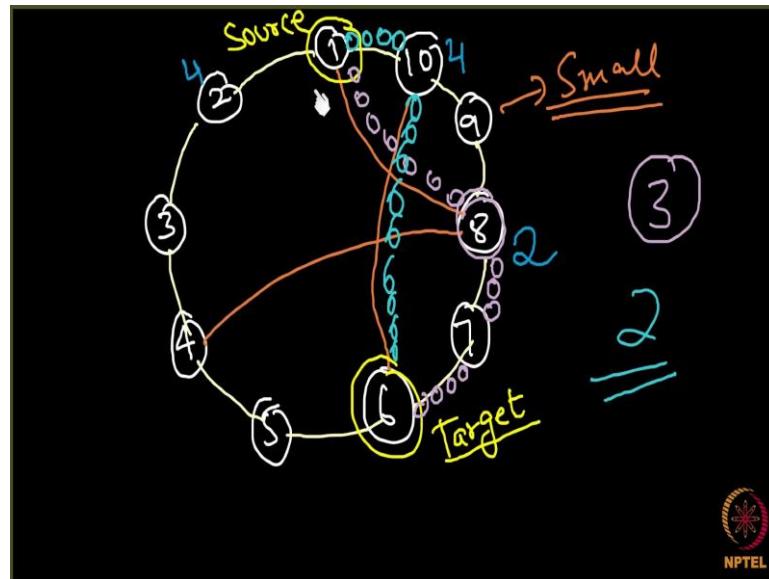
(Refer Slide Time: 02:19)



So, let me tell you this kind of search is known as the myopic search as well. So, we call this decentralised search as myopic search. Why myopic? Do you remember myopia we are studied it when we were in like 6th or 7th or 9th class, I do not remember? So, myopia; myopia is nothing, but the short sightedness; short sightedness means that you can only look at a small distance you cannot look at a long distance.

And, that is what happens in the decentralised search right, you do not know about the entire network, you know only about your locality. And, based on your locality you decide where you want to pass a letter and hence myopia. And, hence this search is known as the myopic search. Let us quickly see how we perform a myopic search before starting with the screencast. So, let me take a 1-dimensional Watts-Strogatz model here.

(Refer Slide Time: 03:13)



And, let us say I take here ten nodes arranged in a ring 1 2 3 4 5 6 7 8 9 and 10. Now, here write it 10 nodes which are arranged in the form of a ring and for the sake of simplicity I am going to connect each node. So, the homophily links each node 1 edge towards its left and 1 edge towards its right; now these are the homophily links in this network in this homophor ring and next my aim is to lay some weak ties in this network. So, I am simply instead of rewiring as we have discussed before I am simply going to add some edges here.

So, let us say I add an edge from 10 to 6 and then I add an edge from 1 to 8 let us say and let us say one from 4 to 8. And, these are be 3 weak ties in this network which makes this world small, which makes it a small world network. Now, how do we do a decentralised search over here? So, let us say our node 1 here is the source which has the letter and let us say the node 6 here is the target. We want to pass on the letter to this target which is the node 6. Now, how do our myopic search, how do our decentralized search work here.

We have seen before how does decentralised search work every node is going to look towards its neighbours and choose the neighbour which is closest to the target node 1 here. So, node 1 here it is supposed to pass on the letter to the node 6. So, according to our myopic search decentralised search it will first look at node 10 and then look at the distance of this node 10 from the target. Now, here I would like to tell you when you

look at the distance of your neighbours from the target you do not have information about the weak ties which your neighbours is connected to.

So, I might know my neighbour and I would have an estimate of the distance to the target. I will not be knowing the weak range the weak range ties of my neighbours the weak ties I will not know, I am very sorry long range contacts and a weak ties of my neighbours I will not know. I will know my weak ties, my long range contacts, but not the long range contacts of my neighbour. So, the node 1 here knows that he can pass on the letter to the node 10 and he can pass on the letter to node 8 or to the node 2.

And, then he calculates the distance for node 2 what is the distance from the target, it is 1 2 3 and 4, the distance of node 2 from the target is 4. And, then he looks at the node 1, looks at node 10 and what is the distance of node 10 from the target. Since, it does not know the long range contacts of node 10 so, we calculate the distance of node 10 to be 1 2 3 4. So, the distance of node 10 from the target is 4 and then node 1 also no more 8 whose distance from the target is 2.

So, what will happen according to decentralised search or the myopic search, the node 1 will pass on the letter to this node 8 over here. And, now this node 8 over here has the letter and then this node 8 looks at all of its neighbours node 9; whose distance from the target is 3. And, then node 1 whose distance from the target is 5 and then node 4, whose distance from the target is 2 and then node 7, whose distance from the target is 1 and it will simply pass on the letter to node 7.

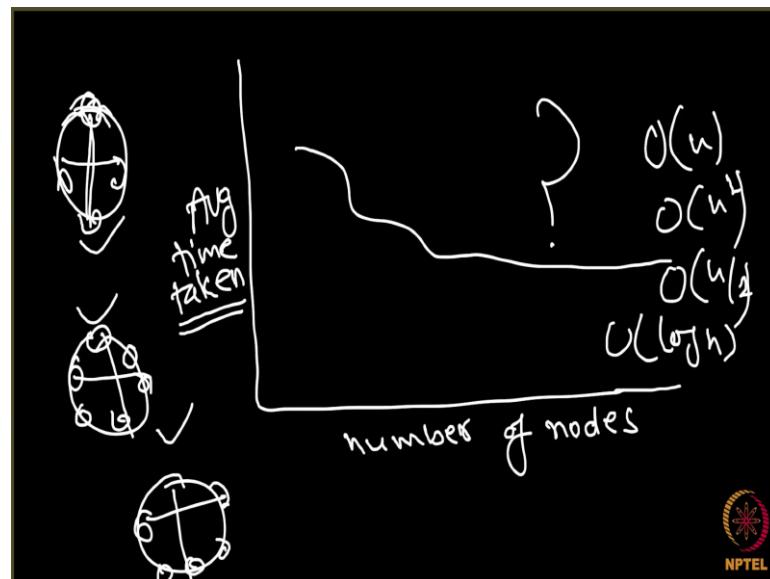
And then node 7 will look at all of its neighbours and the target is its neighbour. So, it will pass on the letter to the target. So, we can see here that this is the path which our decentralised search will find, and the length of this path is simply 3. My next question to you, now we understand what decentralised search is, what is myopic search. Do you think this search policy is optimal? What do I mean by optimal? Optimal means do you think that it has found out the best path, can you find a path from this node 1 to node 6 in this graph path from node 1 to node 6; whose distance is 3 and its actually possible.

So, if you looked at this path over here you start from node 1 and then you come to node 10 and from 10 you can directly go on to this node 6 over here. So, there is a path of length 2 from the source to the target in this network, but our myopic search has found out the path of length 3. So, it confuses that our myopic search our decentralised search

is not optimal it is not optimal, but still it gives us a path of a quite less distance with a less knowledge. So, one we were working out the optimal path from node 1 to node 6 we have assumed that we have the global knowledge about this network.

So, anybody who is knowing this entire network can find out this path, but if you know only the information about your locality, about your neighbours which is few in the real world which is difficult to get an optimal path. So, you follow this greedy approach which is called the decentralized search and find out this path. So, we will be doing this and what we will be looking at in this programming screencast is let us say here are the number of nodes in our network.

(Refer Slide Time: 09:09)



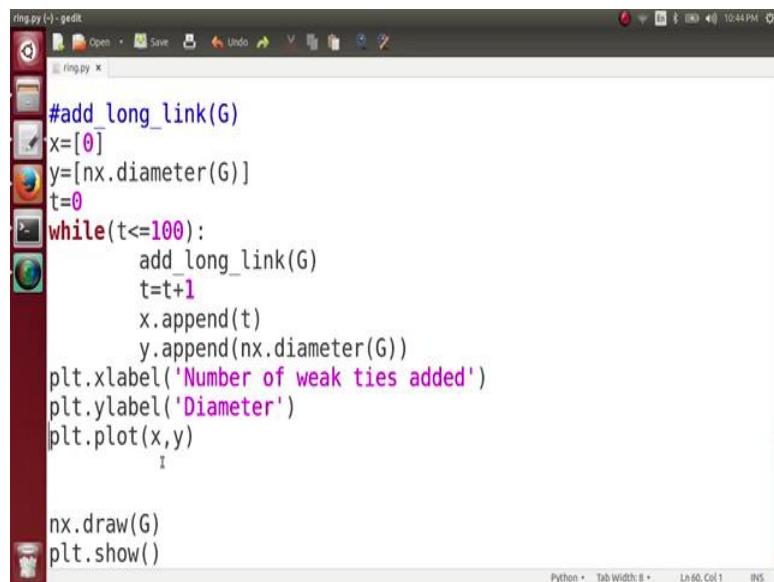
Say here are the number of nodes, mainly you can make a ring having 100 nodes, you can make a ring having 200 nodes, you can make a ring having 300 nodes and so on. So, different sized rings we are going to take of course, with the weak ties over here. So, different small world networks one dimensional small world networks we are going to take. And, then we are going look at we are going to do decentralised search on these networks and will be looking at what is the average time taken.

What is the average time taken to find the distance to find the path from the source to the target? How many steps are taken and will see at how does this plot look like, whether it is order of  $n$  whether it is order of  $n$  square or order of  $n$  by 2 or order of  $\log n$  or order of  $\log^2 n$ ; what is it you will be looking at this in the second programming screencast.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**How to go Viral on Web**  
**Lecture - 156**  
**Myopic Search**

(Refer Slide Time: 00:07)



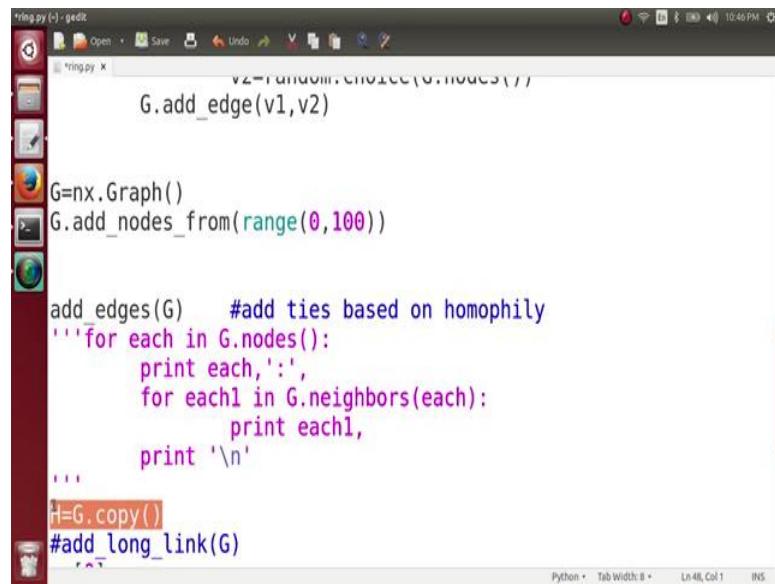
```
#add_long_link(G)
x=[0]
y=[nx.diameter(G)]
t=0
while(t<=100):
    add_long_link(G)
    t=t+1
    x.append(t)
    y.append(nx.diameter(G))
plt.xlabel('Number of weak ties added')
plt.ylabel('Diameter')
plt.plot(x,y)

nx.draw(G)
plt.show()
```

So, we have seen that how we can write a code for making a small world network in 1 dimension. What is the next aim is, we want to implement the decentralized search or myopic search on this network and look at how does this search confirm. So, what I am going to do in this code is, we look at how do we do myopic search on this network. So, for doing myopic search we remember what we have to do is we take a node and then we look at all of its neighbors. And, then we look at the distance of all these neighbors from the target and do you remember when we look at this distance, we assume that we have no knowledge about the long range context of these neighbors.

So, we have to look at the distance of these nodes across the cycle. So, while looking at that distance we assume that there is no long link in the network and then we find their distance.

(Refer Slide Time: 01:07)



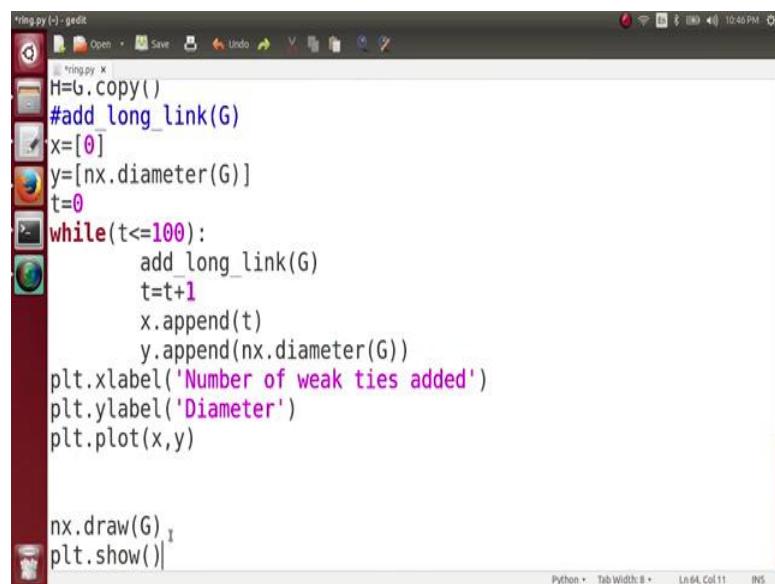
```
trring.py (-) - gedit
trring.py x
v2=random.choice(G.nodes())
G.add_edge(v1,v2)

G=nx.Graph()
G.add_nodes_from(range(0,100))

add.edges(G)      #add ties based on homophily
'''for each in G.nodes():
    print each,':',
    for each1 in G.neighbors(each):
        print each1,
    print '\n'
...
H=G.copy()
#add long_link(G)
```

So, for that purpose what I am going to do is the graph G here and then I have added the edges here. So, before adding any long link in this network I will take a copy of this graph here  $H = G.copy$ . And, why I have taken this copy here is actually very simple, the reason is very clear what is the reason when I am going to do myopic search. So, in that myopic search I have to find the distance of my neighbors from the target and by finding the distance, I do not have to see the long range ties in the network.

(Refer Slide Time: 01:45)

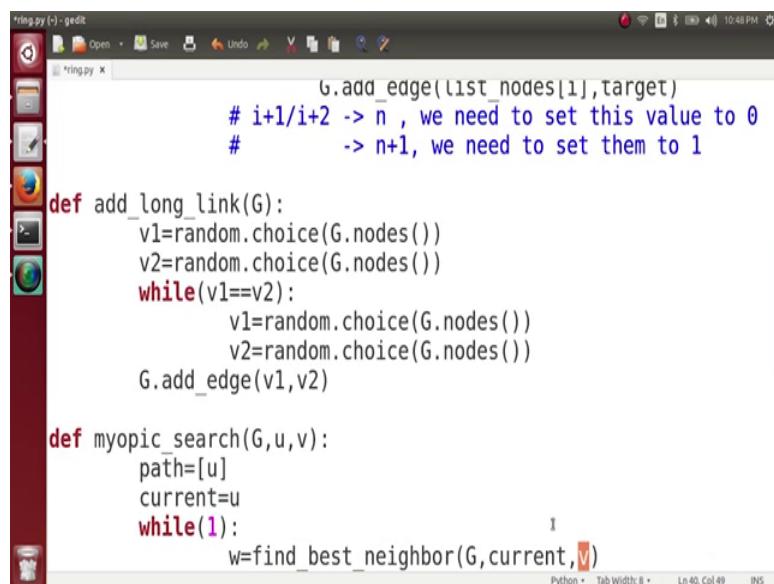


```
trring.py (-) - gedit
trring.py x
H=G.copy()
#add long_link(G)
x=[0]
y=[nx.diameter(G)]
t=0
while(t<=100):
    add_long_link(G)
    t=t+1
    x.append(t)
    y.append(nx.diameter(G))
plt.xlabel('Number of weak ties added')
plt.ylabel('Diameter')
plt.plot(x,y)

nx.draw(G)
plt.show()
```

So, when I have to find the distance, I look in this network edge, this network edge is this network edge. This network edge here is simply my cycle, the cycle which we have formed which consists of only the ties based on homophily and has no long range link. And, then this network G over here, G is the network which consists of the homophily based links as well as the weak ties. So, whenever I do my myopic search, I will be performing the myopic search on my network G, but when finding the distance of my neighbors from the target; I will be using this network edge over here ok. So, how do we proceed is I define a function myopic search over here.

(Refer Slide Time: 02:37)



```

trring.py (-) - gedit
10:48 PM
trring.py X
G.add_edge(list_nodes[1],target)
# i+1/i+2 -> n , we need to set this value to 0
#           -> n+1, we need to set them to 1

def add_long_link(G):
    v1=random.choice(G.nodes())
    v2=random.choice(G.nodes())
    while(v1==v2):
        v1=random.choice(G.nodes())
        v2=random.choice(G.nodes())
    G.add_edge(v1,v2)

def myopic_search(G,u,v):
    path=[u]
    current=u
    while(1):
        w=find_best_neighbor(G,current,v)

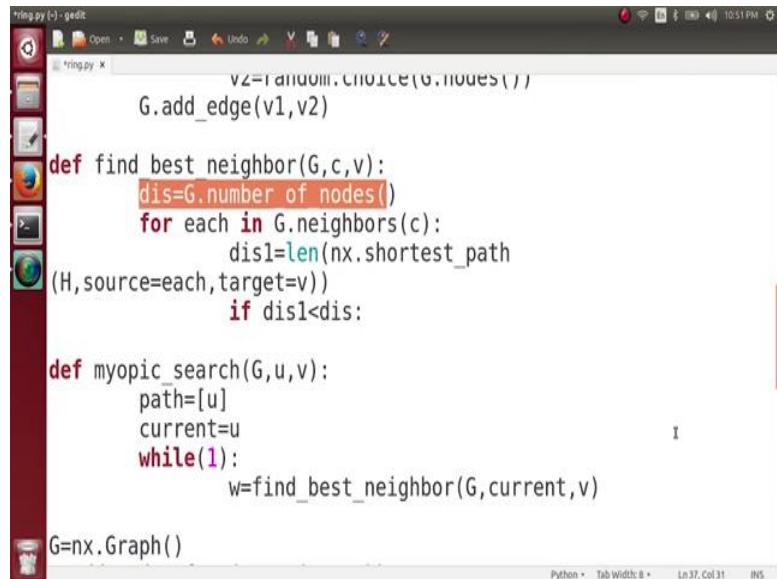
```

Define myopic search, I am these needs are graph the source vertex and a target vertex. So, u is my source and v is my target and we have to find the path from source and target. I take this path as a list and this path initially consists of only the node u, which is the source, current is my variable which takes care of where I am standing currently on the network. So, currently we are standing on u and while 1 what we are going to do is I take a loop over here, what we do is we have to find the next best neighbor where we have to move on.

So, I call a function here find best neighbor find best neighbor G current where, have to find the best neighbor from current and for finding the best neighbor I will be needing the target. So, what I am going to do is for finding the best neighbor this current node will be looking at all of its neighbors and we choose the one which is closest to the target

in the graph H, not in the graph G. In the graph H where there are no long-range ties. So, before proceeding further let me define this function here.

(Refer Slide Time: 04:01)



```
ring.py (~) - gedit
v2=random.choice(G.nodes())
G.add_edge(v1,v2)

def find_best_neighbor(G,c,v):
    dis=G.number_of_nodes()
    for each in G.neighbors(c):
        dis1=len(nx.shortest_path(H,source=each,target=v))
        if dis1<dis:

def myopic_search(G,u,v):
    path=[u]
    current=u
    while(1):
        w=find_best_neighbor(G,current,v)
        path.append(w)
        current=w
        if current==v:
            break
    return path

G=nx.Graph()
```

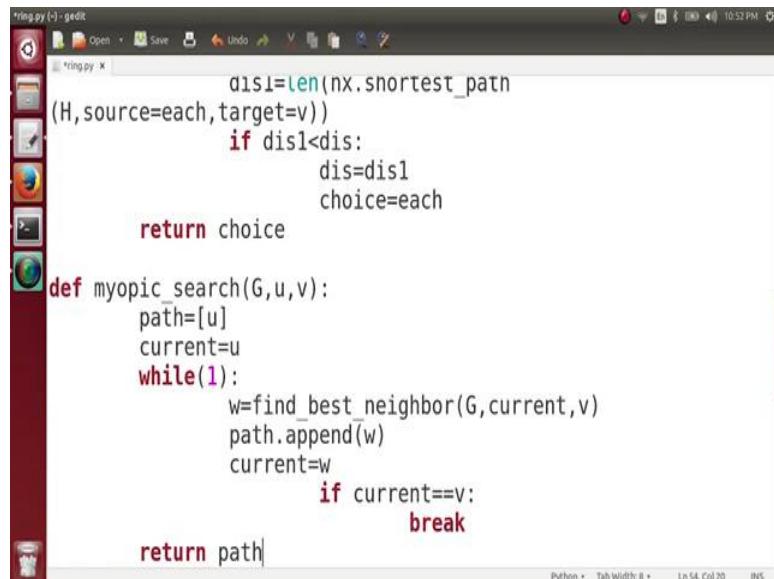
Define find best neighbor, define find best neighbor G current node and the target. So, I get a variable over here distance equals to G.number\_of\_nodes, you will soon understand why I have taken why I am taking this variable over here. So, what do I do next is I look at each of the neighbors of this node current here? So, for each in G.neighbors(c) so, for each in G.neighbors(c) what I am going to do is, I am going to calculate again a distance, distance 1 which is the distance of this neighbor from the target. So, for finding a distance of the neighbor from the target we can use the function nx.shortest\_path which gives us the shortest path between 2 nodes in a graph; nx.shortest\_path and the graph H.

So, you understand why we are taking H here because, we have to look at the distance across the cycle not across the complete graph, we have to ignore the long range ties. So, H and what is my source. So, here this is the source for my shortest path. So, the shortest path function requires three parameters the graph, the source that is the first node and the target which is the second node; approximate I am finding the distance. Please do not confuse it with a source and target over for our myopic search, the source and target for myopic search are different and these source and target are different. So, when we were doing the myopic search, we were at a node current and now this node current is

interested in finding the best neighbor and for all of its neighbors it is calculating their distance from the target.

So, here the source here is nothing, but the neighbor of my node current. So, the source is what? Source here is each which is the neighbor and target here is v. So, the target is actually the same the source changes, now what I am going to do is, if this distance is less than the previous value of distance which was quite high. So, we want this value to be quite high and if my distance 1 it is less than distance; what I am going to do is distance equals to distance 1.

(Refer Slide Time: 06:41)



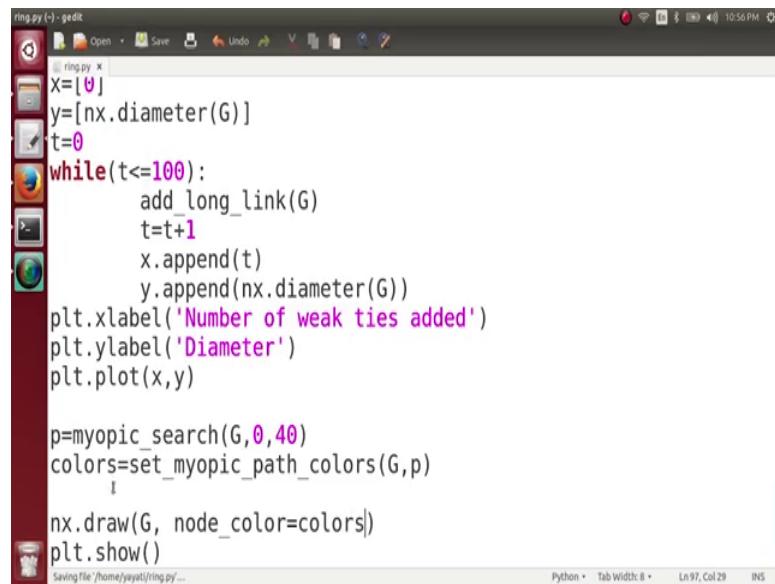
```
string.py (~) - gedit
dis1=len(nx.shortest_path
(H,source=each,target=v))
if dis1<dis:
    dis=dis1
    choice=each
return choice

def myopic_search(G,u,v):
    path=[u]
    current=u
    while(1):
        w=find_best_neighbor(G,current,v)
        path.append(w)
        current=w
        if current==v:
            break
    return path
```

So, this is a simple method in coding which we used to find out the maximum value or the minimum value. And, I set the value of choice the best neighbor to be equal to each and at the end I return choice. So, I think that this is pretty clear and then I return back to my code for myopic search. Here I have find out the best neighbor for my node current and after finding the best neighbor I append this best neighbor to my list to my path.

And, then the value of current becomes equals to this neighbor which I have found and also by doing this if the value of current becomes equals to v which is the target; then what we have to do is simply come out of this while loop and then we can return the so, we can return path h v. So, we return path here ok. So, this is the code for our myopic search and we return path here ok. How do we execute it?

(Refer Slide Time: 07:53)



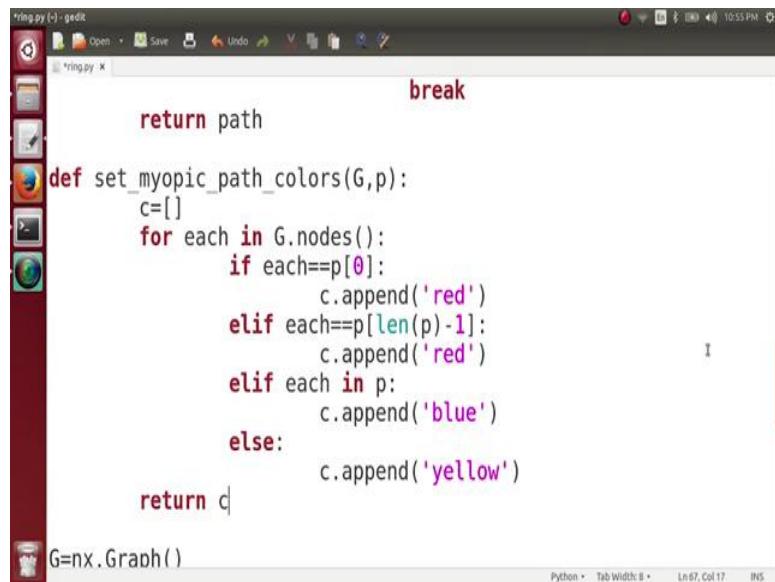
```
ring.py (~) - gedit
X=[0]
y=[nx.diameter(G)]
t=0
while(t<=100):
    add_long_link(G)
    t=t+1
    x.append(t)
    y.append(nx.diameter(G))
plt.xlabel('Number of weak ties added')
plt.ylabel('Diameter')
plt.plot(x,y)

p=myopic_search(G,0,40)
colors=set_myopic_path_colors(G,p)
nx.draw(G, node_color=colors)
plt.show()
Saving file '/home/yayati/ring.py' ...
Python Tab Width: 8 Ln 97, Col 29 INS
```

So, let us say p equals to the path which I wanted to find out and myopic search and let us say I have performed the myopic search, let us say from the node 0 to node 40. And, I wanted to see how this path looks like; I am going to play around with colors little bit here what I am going to do is, I am going to call a function here set myopic path colors why no.

So, I guess you know what I am going to do here. So, when we are going from this node from 0 to 40 I want to see how does this path look like. So, what I am going to do is I am going to define a function here, function for set myopic path colors. And, here I have to pass my graph G and the path p and it gives me an array called colors and let us define this function here.

(Refer Slide Time: 09:05)



```
tryng.py (-) - gedit
break
return path

def set_myopic_path_colors(G,p):
    c=[]
    for each in G.nodes():
        if each==p[0]:
            c.append('red')
        elif each==p[len(p)-1]:
            c.append('red')
        elif each in p:
            c.append('blue')
        else:
            c.append('yellow')
    return c

G=nx.Graph()
```

Define set myopic path colors and what the function is going to do is for I have a array here c, which is for colors for each in G.nodes. What I am going to do is, if each equals to let us say p 0 which is the starting node of our path; what I am going to do is for each in 0th node, if each is a starting node of our path and going to append a let us say red color here. And, if each equals to equals to the last element of my path then also I am going append red. What I am going to do is the source and the target here are going to be red.

The source and target here are going to be red and are going to be red we call it is actually not else if, it is elif and elif each in p ok. So, if it is the starting node of this path p, we append a red here. If it is the end path of this list p of this path p we append a red here, if it is neither starting node and but exists in p. So, what I am going to append there is let us say blue and what do I do for rest of the nodes I make them yellow.

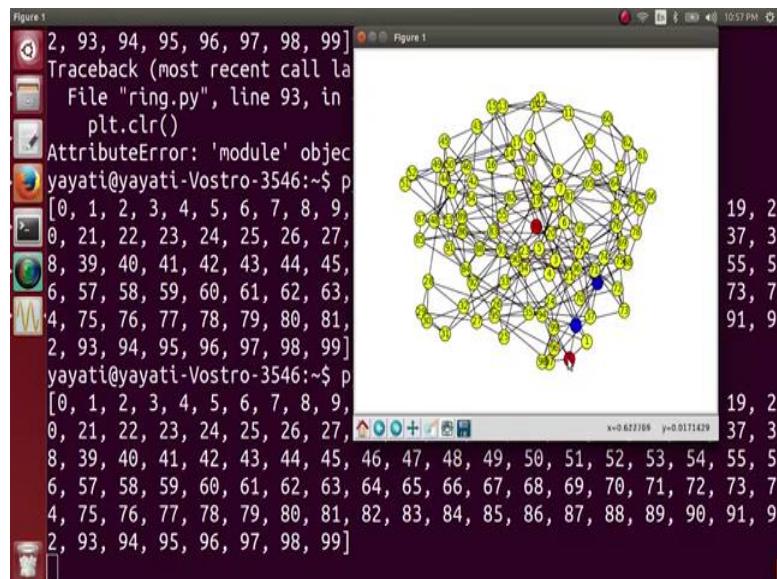
So, all the nodes in my graph are now going to be displayed in yellow, the starting and the ending nodes are going to be red and the rest of the paths on my; rest of the nodes on my path are going to be blue. And, then I return c here you will soon understand why I am using colors here and then we got colors here nx.draw(G) node underscore color equals to colors.

(Refer Slide Time: 11:35)

```
yayati@yayati-Vostro-3546:~$ python ring.py
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 2
0, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 3
8, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 5
6, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 7
4, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 9
2, 93, 94, 95, 96, 97, 98, 99]
yayati@yayati-Vostro-3546:~$ python ring.py
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 2
0, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 3
8, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 5
6, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 7
4, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 9
2, 93, 94, 95, 96, 97, 98, 99]
yayati@yayati-Vostro-3546:~$ python ring.py
  File "ring.py", line 52
    if current==v:
      ^
IndentationError: unexpected indent
yayati@yayati-Vostro-3546:~$
```

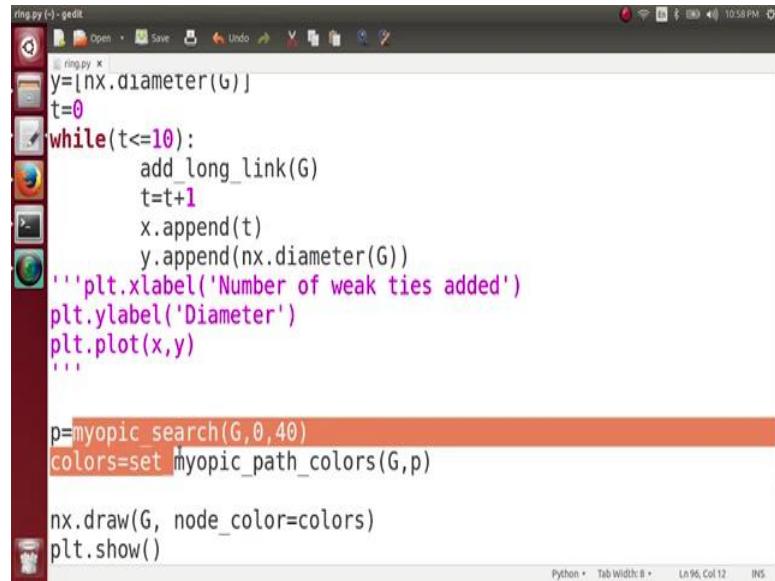
And, let us now see execute it and see line 52 shows an error. So, I have commented this portion over here and let us now execute it and see.

(Refer Slide Time: 11:47)



You can see that when we have to find a path from this node 0 here to 40. So, we can go from 0 to 2, 2 to 38 and 38 to 40. So, this is the path which my myopic search gives you and the network here looks quite random. So, we know why it looks random over here because, we have added a lot of long-range links to this network. So, what I going to do is, I am going to change this value of t here to let us say 10.

(Refer Slide Time: 12:13)



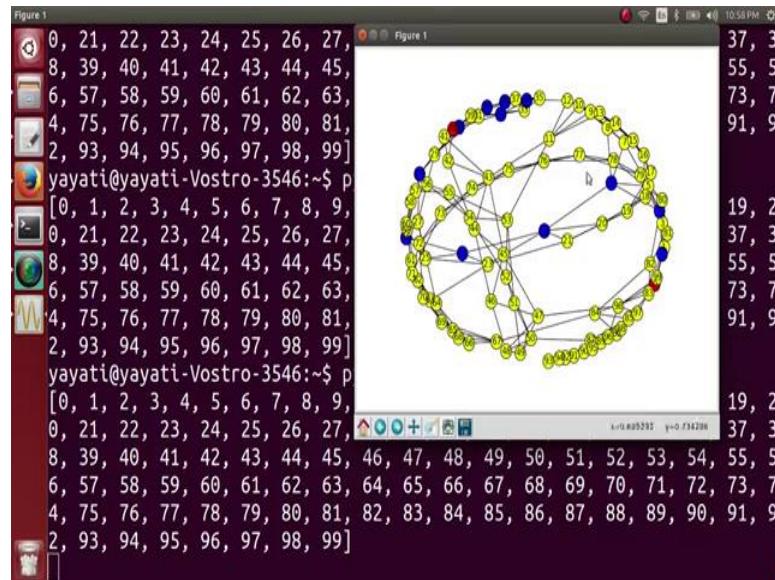
```
ring.py (~) - gedit
y=[nx.diameter(G)]
t=0
while(t<=10):
    add_long_link(G)
    t=t+1
    x.append(t)
    y.append(nx.diameter(G))
'''plt.xlabel('Number of weak ties added')
plt.ylabel('Diameter')
plt.plot(x,y)
'''

p=myopic_search(G,0,40)
colors=set myopic_path_colors(G,p)

nx.draw(G, node_color=colors)
plt.show()
Python  Tab Width: 8  Ln 96, Col 12  INS
```

So, let us add only 10 long range links over here and we know that what was the initial diameter of this network was nothing, but 25. From that 25 diameter after that I have added 10 long range links and then I find my path from the node 0 to 40 using my myopic search and let us see ok.

(Refer Slide Time: 12:39)

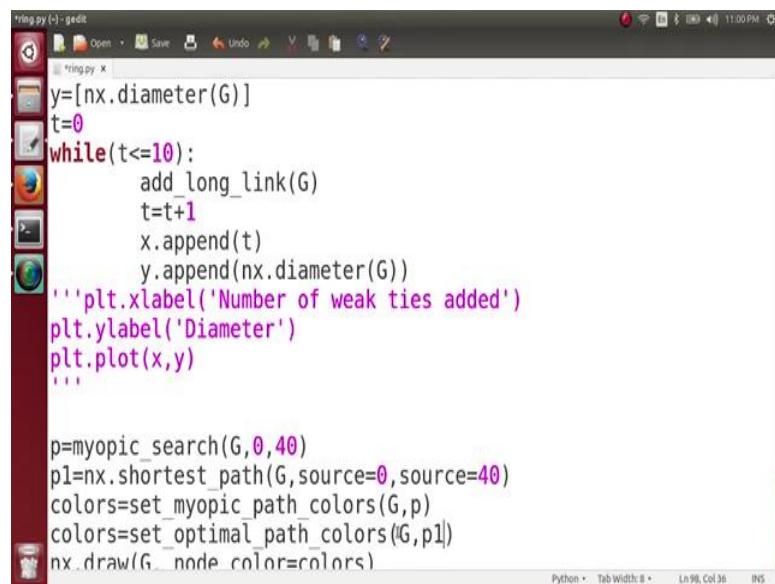


Now you see the path length has increased quite a lot and this is my ring network although it does not appear here quite like a ring. So, it starts here from node 0 and there

are 1 2 3 4 5 6 7 8 9 10 11 12 nodes in between. So, within so, the path length here is 12. So, the myopic search here finds a path of length 12.

What next? I am going to do is we have seen that the myopic search is not an optimal search. Let us now compare the myopic search to my optimal search and what we are going to do now, what we are going to do now is we will find the optimal distance let us say from this node 0 to 40 only.

(Refer Slide Time: 13:27)



```
ring.py (~) - gedit
y=[nx.diameter(G)]
t=0
while(t<=10):
    add_long_link(G)
    t=t+1
    x.append(t)
    y.append(nx.diameter(G))
'''plt.xlabel('Number of weak ties added')
plt.ylabel('Diameter')
plt.plot(x,y)
'''

p=myopic_search(G,0,40)
p1=nx.shortest_path(G,source=0,source=40)
colors=set_myopic_path_colors(G,p)
colors=set_optimal_path_colors(G,p1)
nx.draw(G, node_color=colors)

Python Tab Width: 8 Ln 98, Col 36 INS
```

So, what I am going to do is I am going to find another path and this path is nothing, but the shortest path, the optimal shortest path on my graph G from node 0 to 40. So, you see here I am applying a shortest path on the graph G.

So, this graph G here is consisting of the long range link also and then I find p 1 and what actually we can do here is I want to see a different path also. So, here I am going to, what I am going to do is set optimal path color. So, I am also going to set optimal path colors here based on the path obtained from here from G to p 1 and then this is going to be quite easy.

(Refer Slide Time: 14:21)

```
ring.py (-) - gedit
return path

def set_myopic_path_colors(G,p):
    c=[]
    for each in G.nodes():
        if each==p[0]:
            c.append('red')
        elif each==p[len(p)-1]:
            c.append('red')
        elif each in p:
            c.append('blue')
        else:
            c.append('yellow')
    return c

G=nx.Graph()
G.add_nodes_from(range(0,100))
Python Tab Width: 8 Ln 56, Col 1 INS
```

So, we are going to use the same code which is here.

(Refer Slide Time: 14:29)

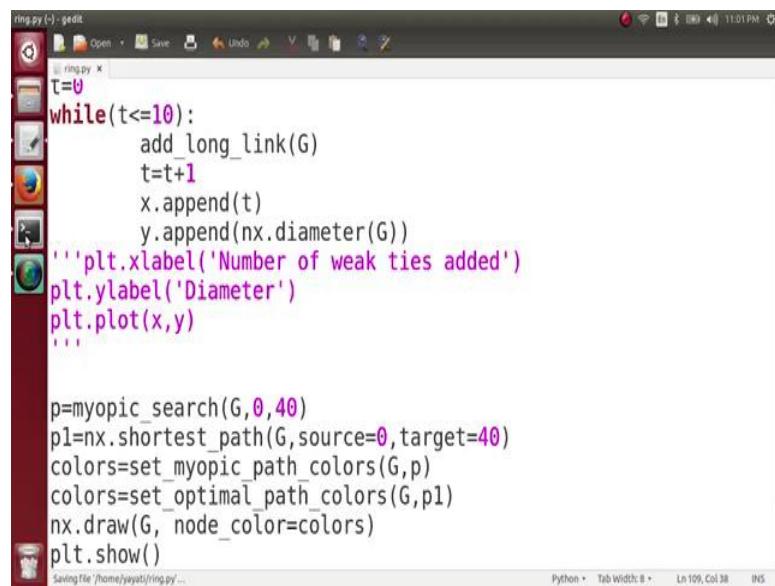
```
ring.py (-) - gedit
return c

def set_myopic_path_colors(G,p1):
    c=[]
    for each in G.nodes():
        if each==p1[0]:
            c.append('red')
        elif each==p1[len(p1)-1]:
            c.append('red')
        elif each in p1:
            c.append('green')
        else:
            c.append('yellow')
    return c

G=nx.Graph()
G.add_nodes_from(range(0,100))
Python Tab Width: 8 Ln 77, Col 40 INS
```

And, let us say so, here it is ( $G, p1$ ) and then  $p1, p1, p1, p1$  and what I am going to do is the starting node is red the ending the target node is also red. The middle nodes on this optimal paths let us color them green and the remaining nodes in the network are yellow. And then we can see and let us now execute this code and see the difference between both the paths.

(Refer Slide Time: 14:59)



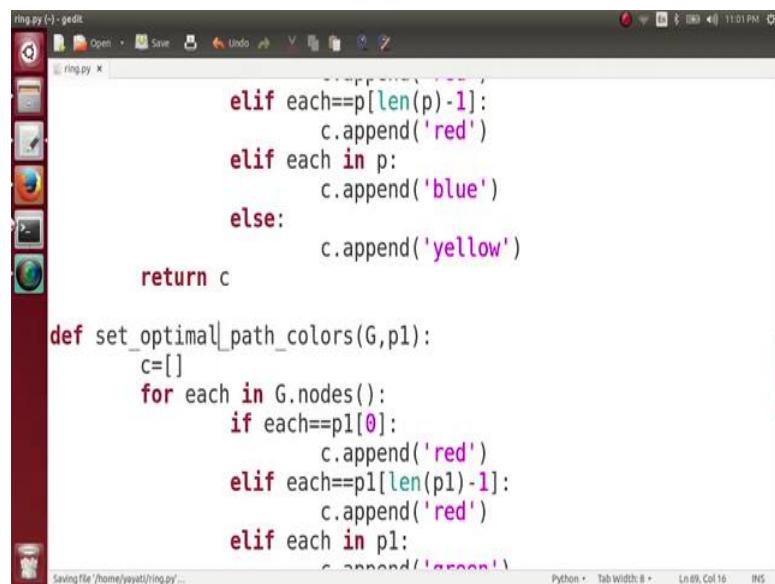
```
ring.py (~) - gedit
ring.py X
t=0
while(t<=10):
    add_long_link(G)
    t=t+1
    x.append(t)
    y.append(nx.diameter(G))
'''plt.xlabel('Number of weak ties added')
plt.ylabel('Diameter')
plt.plot(x,y)
'''

p=myopic_search(G,0,40)
p1=nx.shortest_path(G,source=0,target=40)
colors=set_myopic_path_colors(G,p)
colors=set_optimal_path_colors(G,p1)
nx.draw(G, node_color=colors)
plt.show()
Saving file '/home/yayati/ring.py' ...

Python Tab Width: 8 Ln 109, Col 38 IHS
```

Source I am very sorry source equals to 0 and this is target equals to 40, let us execute it and see set optimal path colors.

(Refer Slide Time: 15:19)



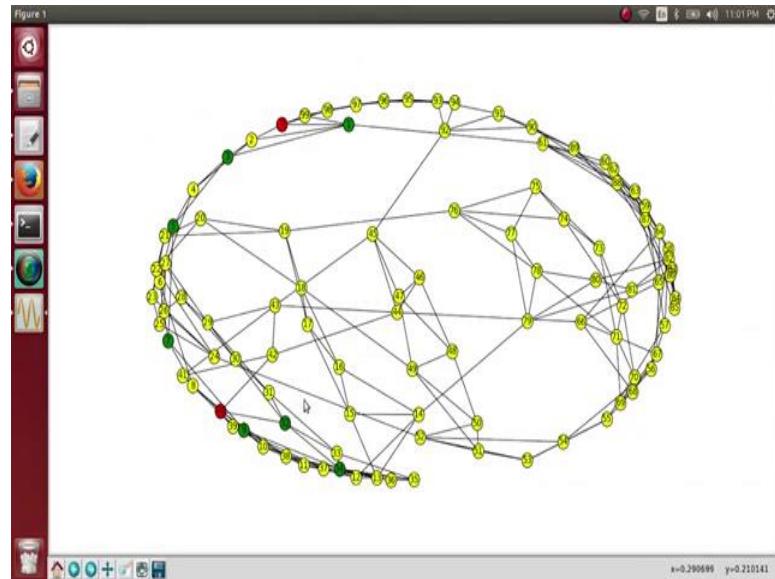
```
ring.py (~) - gedit
ring.py X
        elif each==p[len(p)-1]:
            c.append('red')
        elif each in p:
            c.append('blue')
        else:
            c.append('yellow')
    return c

def set_optimal_path_colors(G,p1):
    c=[]
    for each in G.nodes():
        if each==p1[0]:
            c.append('red')
        elif each==p1[len(p1)-1]:
            c.append('red')
        elif each in p1:
            c.append('green')
    return c
Saving file '/home/yayati/ring.py' ...

Python Tab Width: 8 Ln 69, Col 16 IHS
```

This is set optimal path colors.

(Refer Slide Time: 15:23)



Now, you see this network over here ok. So, here is an error. So, what is the error? When I am doing the set optimal paths colors, I have recolor the nodes, which I have found in the myopic path. So, we have to be a little bit careful so, what we will do is here when we have to find colors, we will actually have to combine both these functions.

(Refer Slide Time: 15:55)

```
t=0
while(t<=10):
    add_long_link(G)
    t=t+1
    x.append(t)
    y.append(nx.diameter(G))
'''plt.xlabel('Number of weak ties added')
plt.ylabel('Diameter')
plt.plot(x,y)
'''

p=myopic_search(G,0,40)
p1=nx.shortest_path(G,source=0,target=40)
colors=set_myopic_path_colors(G,p)
colors=set_path_colors(G,p,p1)
nx.draw(G, node_color=colors)
plt.show()
```

Let us say set path colors ( $G, p$ ) and  $p1$  ok,  $(G, p)$  and  $(p1)$  and then we need only one function here.

(Refer Slide Time: 16:05)

```
def set_myopic_path_colors(G,p):
    c=[]
    for each in G.nodes():
        if each==p[0]:
            c.append('red')
        elif each==p[len(p)-1]:
            c.append('red')
        elif each in p:
            c.append('blue')
        else:
            c.append('yellow')
    return c

def set_optimal_path_colors(G,p1):
    c=[]
    for each in G.nodes():
```

Set path colors.

(Refer Slide Time: 16:09)



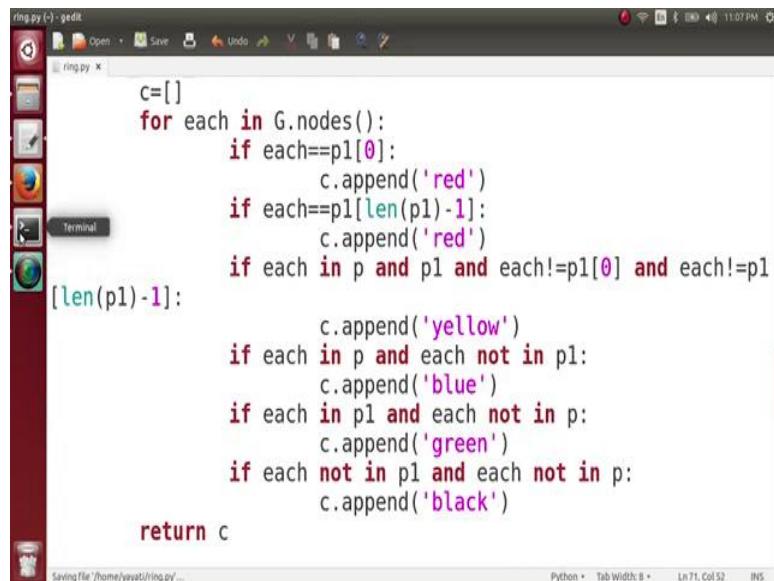
```
def set_path_colors(G,p,p1):
    c=[]
    for each in G.nodes():
        if each==p[0]:
            c.append('red')
        elif each==p1[len(p1)-1]:
            c.append('red')
        elif each in p and p1:
            c.append('yellow')
        elif each in p:
            c.append('blue')
        elif each in p1:
            c.append('green')
        else:
            c.append('black')
```

$(G, p, p_1)$  and what I am going to do is; obviously, what is if each is  $p_1[0]$  it is  $p[0]$  also because, the source node is same red and then here it is red. And, it is each in  $p_1$  when we are making it green and you see here one node can be both in  $p$  and  $p_1$  right. So, if there is a node which is present in both  $p$  as well as  $p_1$  and it is neither the source node nor the target node we are coloring that ok.

Please be careful with the colors here, we are coloring that node to be yellow ok. So, yellow is a node which occurs in both these paths  $p$  and  $p_1$  elif each in  $p$ ,  $p$  is the path for myopic search what we are going to append here is; for myopic search we are going to use blue. And, elif each in  $p_1$  it is then we are going to append green over here and else it is in none, we are going to append a black over here. So, please see all the nodes in this network are going to be black, the source node is red the target node is red.

The nodes which are in myopic in the path for myopic search are blue, nodes which are in path for optimal search are in green and the nodes which are in the paths for both the myopic search and the optimal search are in yellow. So, we are going to make it a little bit more rigorous and more understandable, what we are going to do is ok.

(Refer Slide Time: 17:55)



```

ring.py (~) - gedit
c=[]
for each in G.nodes():
    if each==p1[0]:
        c.append('red')
    if each==p1[len(p1)-1]:
        c.append('red')
    if each in p and p1 and each!=p1[0] and each!=p1
[len(p1)-1]:
        c.append('yellow')
    if each in p and each not in p1:
        c.append('blue')
    if each in p1 and each not in p:
        c.append('green')
    if each not in p1 and each not in p:
        c.append('black')
return c

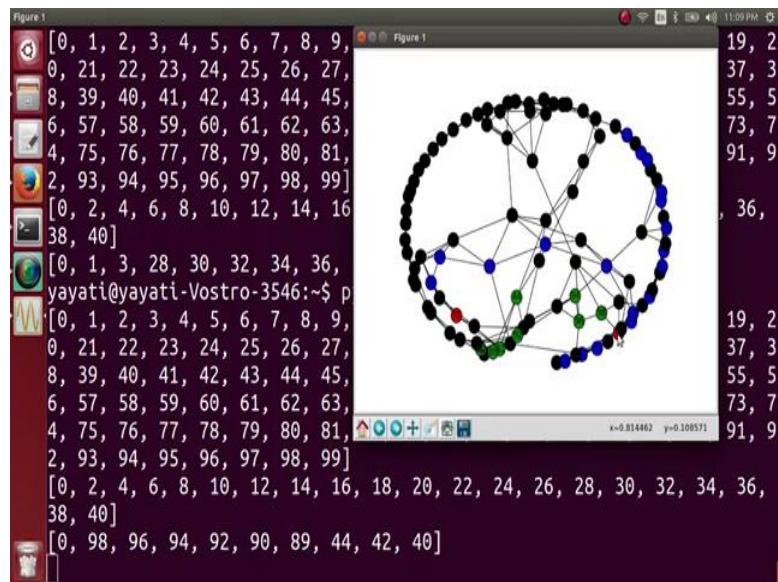
```

Saving file /home/yashas/ring.py ...

Python • Tab Width: 8 • Ln 71, Col 52 • INS

If each in  $p$  and  $p_1$  and each is not equals to  $p_1[0]$  and each is not equals to  $p_1[len(p1) - 1]$  which clearly says that if each in  $p$  and  $p_1$  it is in both the paths. But its neither equal to the source vertex not equal to the target vertex then we are going to append our yellow here, if each in  $p$  and each not in  $p_1$  then we are going to make it blue. And, if each in  $p_1$  and let us say each not in  $p$  then we are going to make it green. So, instead of an elif here we can use a if here, here also we can use a if and at last if each not in  $p_1$  and each not in  $p$  we are going to make it black.

(Refer Slide Time: 19:09)



Now, let us execute it and see, from here you see there is no node common between both of the paths. If you look at the myopic search its start from 0 and close to 40, it takes this path 0 to 2 to 4 ok; it is actually little bit difficult to make out where this path is going. So, before 30 there is 28 ok. So, this path goes something like this, the ending path is here ok. Let us see from 0 it goes to 2, 2 is somewhere here, from 2 it goes to 4 which is here and then to 6 8.

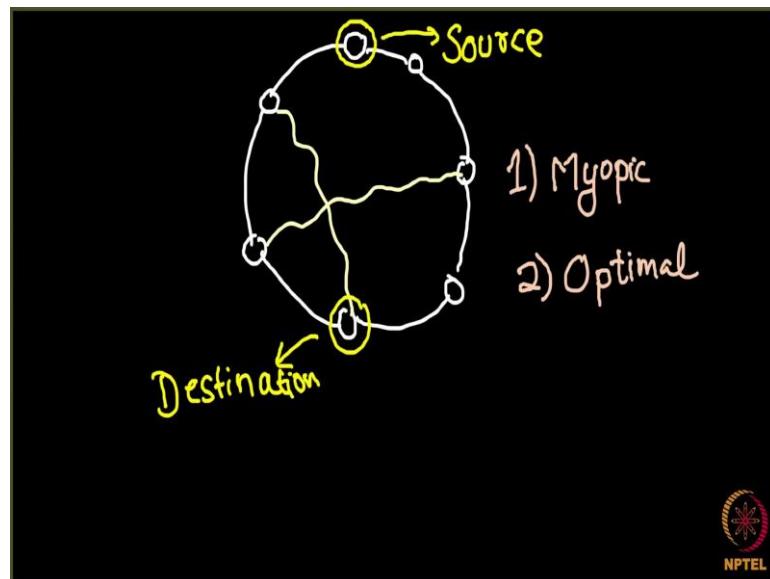
So, it goes all the way here and then it comes here, here and here and reaches 40 and why an optimal path starts from 0 and goes like this. So, we can see that optimal path is much shorter than what are myopic search is finding out, which clearly shows that a myopic search is not the optimal search.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**How to go Viral on Web**  
**Lecture - 157**  
**Myopic Search comparison to optical search**

So, what we are going to do in the first programming screencast is, we are going to take a small world network in 1 dimension which we have discussed already what is that, that is a first of all a ring.

(Refer Slide Time: 00:15)



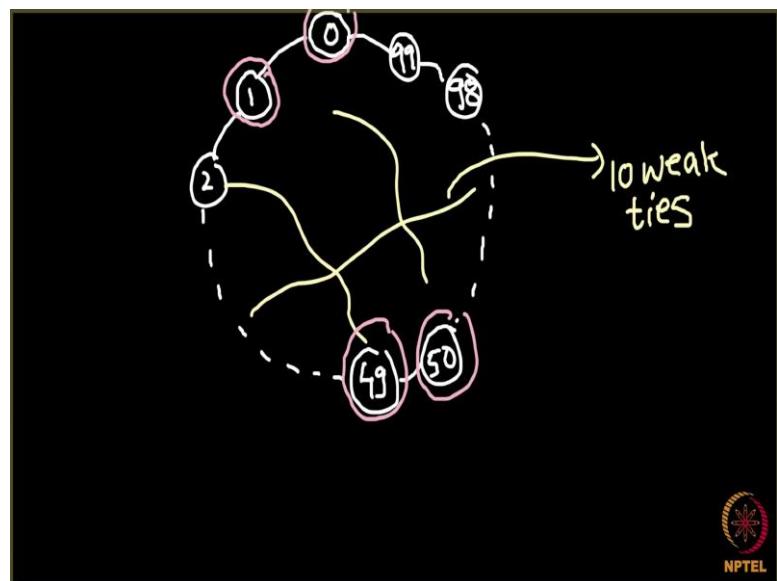
So, I am going to take a simple ring, a simple ring I mean 1 node on this ring for example, this node here is connected to only 1 node toward its left and 1 towards its right. So, I am going to take this simple ring and then what we are going to do is, we are going to perform 2 kind of search on this network. So, first of all by the side of these ties which are homophily base there will be weak ties something like this.

So, let me put 2 weak ties in this network. So, these yellow lines are the weak ties here and these are the homophily base ties. What we are going to do next is we are going to choose 2 nodes. So, I choose here 2 nodes let say this and I call this node as source and then I choose let say this node and then I call this node as destination. So, for going from

source to destination here we have 2 kind of searches which we have discussed before. So, we are going to implement those 2 kind of searches.

The first one is the myopic search and the second one is the optimal search; from our previous discussion we know that myopic search is not always optimal. In this particular case yes the myopic search is optimal. So, here myopic search will also give you the result we go from here and the, you moved here, and optimal search is also going to give you the same result. However, there can be cases where their results differ as we have seen an example previously. What we are going to do in this programming screencast is we are going to take a small world network one dimensional ring on 100 nodes.

(Refer Slide Time: 02:13)

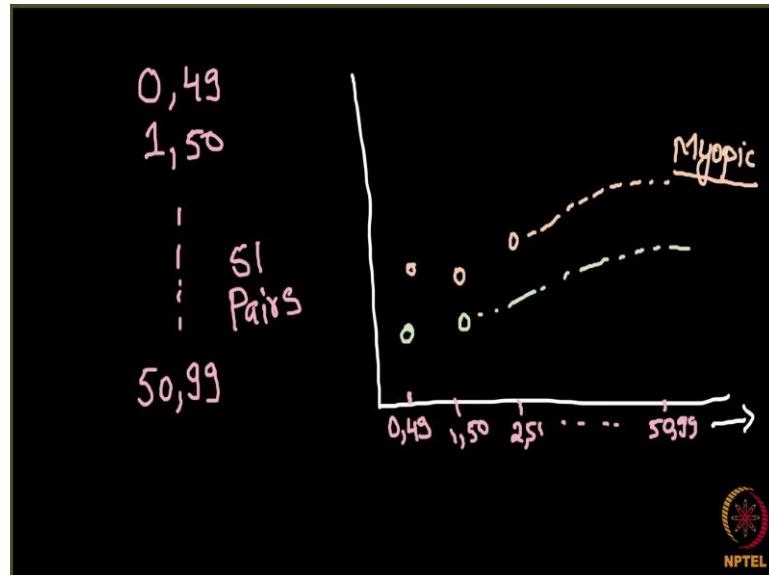


So, let say while label start from 0 so, 0 1 2 so on and something like 49 here and then 50 and then there comes 98 99. So, I am going to take this ring first of all. So, these are my homophily base ties and we are going to put across 10 weak ties here. So, I put 10 weak ties here 10, 10 long range context 10 weak ties. Next what is our aim? Our aim is to compare the myopic search with the optimal search for various different pairs.

So, which pair do you choose? So, we want many pairs for source and destination, and we are going to check for all these pairs; what is the difference between the path taken by the myopic search and the path taken by the optimal search. We think that the best pairs for doing this are the pairs which are diametrically opposite. Why? Because, along this

ring they are at the highest distance from each other; so, we can take 0 and 49 and then 1 and 50. So, these kind of pairs we are going to take. So, what we will be doing.

(Refer Slide Time: 03:31)

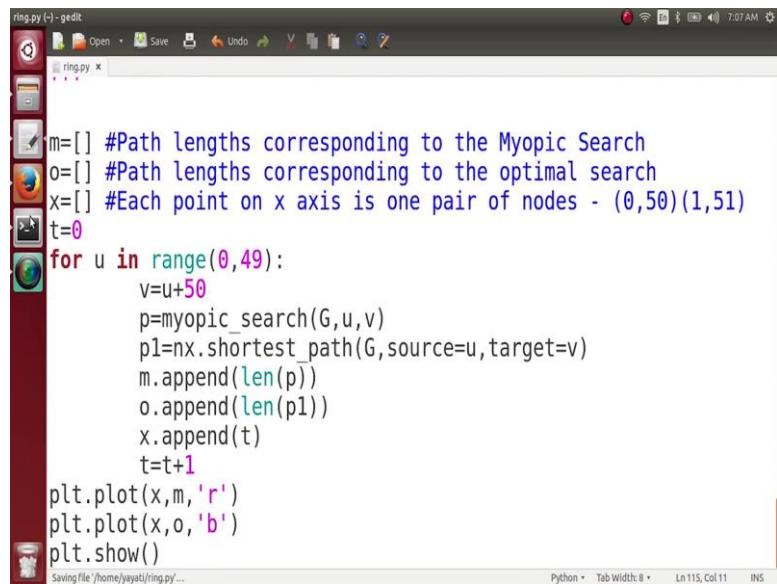


We will be taking the pairs 0 comma 49 1 comma 50 so on and so forth up to 50 comma 99. So, these are going to be I guess 51 pairs. So, we are going to take this 51 pairs and then what we are going to do is we are going to draw a plot. What is this plot? On the x axis here are my pairs. So, this point here can correspond to my pairs 0 comma 49, this point can be 1 comma 50, 2 comma 51 so on and so forth, 50 comma 99.

So, the x axis denotes these pairs and across the y axis we are going to draw 2 results. So, the first is corresponding to the myopic search. So, we will take this pairs 0 and 49 and, on this ring, we will calculate what will be the length of the path taken by myopic search. So, if myopic search goes from 0 to 1, 1 to 2 and 2 to 49; in this case the length of the path will be 3 right. So, we will see what is the length of the path taken by myopic search and plot that somewhere here and then 1 comma 50, 2 comma 51 so on and so forth.

So, we are going to see what answer a myopic search gives us and we are also going to draw a plot corresponding to the optimal search on the same plot. So, for 0 comma 0 comma 49, what was the path of the length given by optimal search for 1 comma 50 so on and so forth. And, we will be observing how this plot looks like in reality ok.

(Refer Slide Time: 05:17)



```
ring.py (~) - gedit
m=[]
o=[]
x=[]
t=0
for u in range(0,49):
    v=u+50
    p=myopic_search(G,u,v)
    p1=nx.shortest_path(G,source=u,target=v)
    m.append(len(p))
    o.append(len(p1))
    x.append(t)
    t=t+1
plt.plot(x,m,'r')
plt.plot(x,o,'b')
plt.show()
Saving file /home/jayati/ring.py...
```

So, what I am going to do now is I will be taking 1 array. So, this array is for my path length for a corresponding to the myopic searching, path lengths corresponding to the myopic search. And, I will be having an array o and this array o is for the path lengths path lengths corresponding to the optimal search, corresponding to the optimal search. And, then I will be having an array x and this x array simply for the x axis.

And, as we have seen previously each point on x axis is represent each point on x axis is nothing, but 1 pair of nodes and this 1 pair of nodes is something like it will be 0 and 49 or it will be 1 and 50 so on and so forth. So, it will be a pair of diametrically opposite nodes. So, that is my x axis, what I will do next is I will take a new array 0, sorry not array appoint t here; t is initially equal to 0.

And, next what I am going to do is the use of t array come to now soon, what I will do next is I know my first node here. So, we are going to change. So, previously we have what we have done was, we have looked at the distance from 0 to 40. Now, we have to look the distance between all the diametrically opposite nodes.

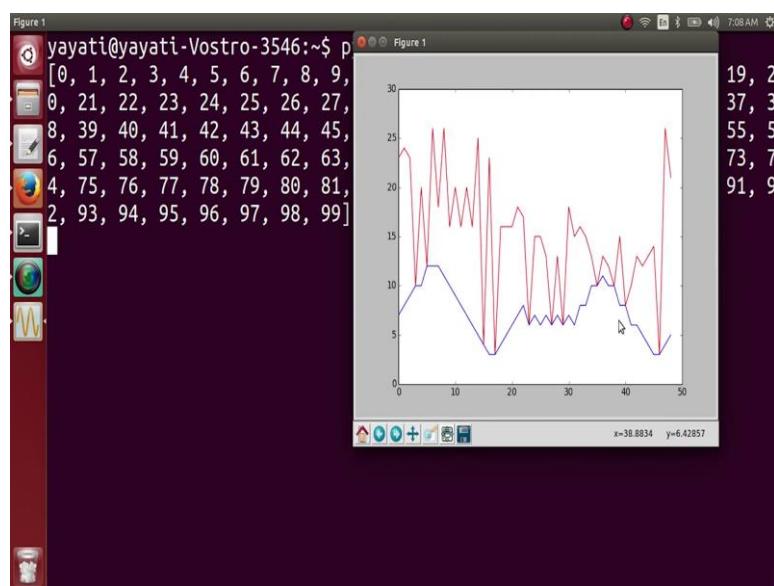
So, I will say my first node is u, for u in range and this range is on 0 to 49, 0 to 49 and for this, what I am going to do is my second which is a target node. So, u is the source node and target node is the diametrically opposite node and we can find it out by adding 50 to this u. So, how the pair will look like corresponding to 0 it will be 50. So, little bit changes here corresponding to 0 it will be 50, corresponding to 1 it will be 51 so on and

so forth. And, next what I will be doing is first I will perform a myopic search here, myopic search node from 0 to 40, but from u to v. And, next I am going to do an optimal search here which is again going to be from u to v. So, I am got the path according to my myopic search and I get path according to me optimal search and we do not need all these lines here; so, I delete them ok.

Next what I will do in my array m, which was corresponding to the myopic search I will append the length of the path which I got from myopic search. And, in my array o, I will append the length of the path which I got from my optimal search which is len(p1) and across the x axis I will simply append the t because, we need some numbers across the x axis and I increment this t by 1 and next what I have to do is at the simply plot this.

So, I will first I plot upload plt.plot I plot myopic search the path length corresponding to the myopic search on the x axis. So, x m and let say it should appear in red color. So, my myopic search of p1 is in red color on this plot and plt.plot and then x. And, optimal the path length corresponding to the optimal search occurs in blue color. And, then I simply visualize this plot; let us run this code and see ring of py.

(Refer Slide Time: 09:31)



So, you can see here, here red colors are corresponding to the myopic search. So, there are we have taken 50 around 50 pairs of diametrically opposite nodes and we can always see that most of the times the myopic search takes quiet large number of steps as compared to the optimal algorithm.

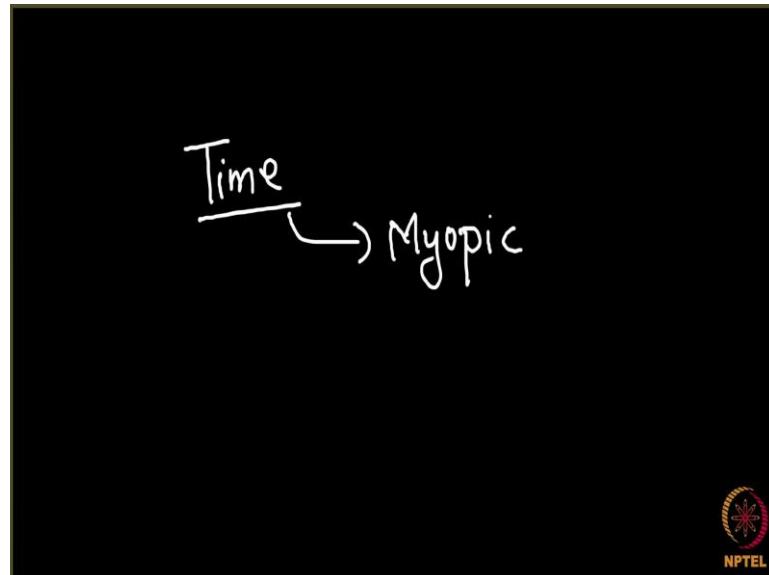
However, there are many points where myopic search actually performs optimally. So, out of these 50 points there are 11 points on which are myopic search it also performs optimally. So, this was the plot we wanted looked at.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**How to go Viral on Web**  
**Lecture - 158**  
**Time Taken by Myopic Search**

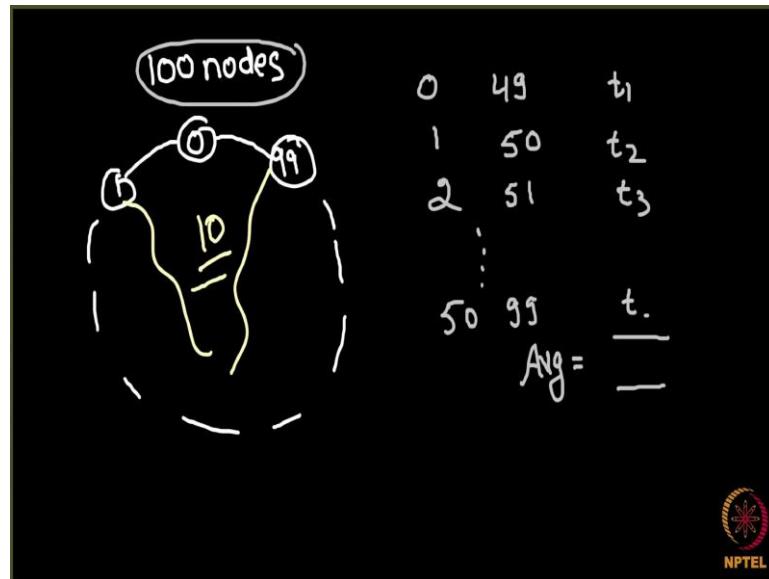
In the previous programming screencast, we compare the myopic search with optimal search.

(Refer Slide Time: 00:13)



Now next what we are interested in looking at is the Time Taken by the Myopic Search. So, if I perform my myopic search on a network of let us say 100 nodes what is the time taken? On a network of 200 nodes what is the time taken so on and so forth; is this time linear in the number of nodes or logarithmic in the number of nodes or what. So, for doing this what we are going to do is first of all let us take a network having 100 nodes.

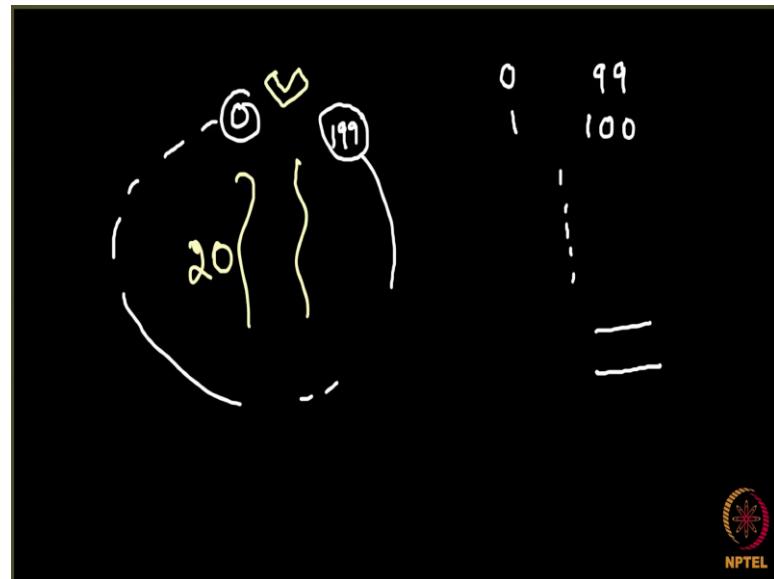
(Refer Slide Time: 00:41)



So, we take a network having 100 nodes as previous. So, we have nodes here 0 1 so on and so forth and here is 99 and we are going to do the same procedure what we did before. We will take the diametrically opposite points and look at how much time does myopic search take to work on this diametrically opposite points. So, we will be taking the point let us say 0 comma 49 and then we will see the time taken by the myopic search. Then for 1 comma 50 what is the time taken, for 2 comma 51 what is the time taken so on and so forth till 50 comma 99 what is the time taken.

Now, what will do next is we will take the average across all these times, that will give us on an average if we take 2 diametrically opposite points on this network; on an average what is the time taken by the myopic search. So, we calculate the average here. Now, this we have done for a network having 100 nodes, next what we will do we will repeat the same procedure on a network having 200 nodes. So, we will be taking a network having 200 nodes.

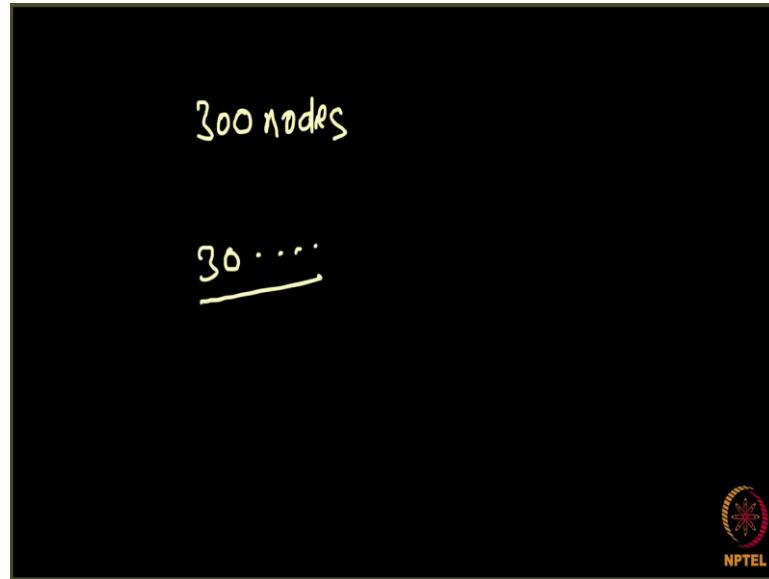
(Refer Slide Time: 01:57)



So, it will go from 0 to 199 and then we will again take the diametrically opposite points which here will be I will say 0 comma 99 and then 1 comma 100 so on and so forth. And, again we will take the average here and that will give us the time taken by myopic search on this network ok, while doing this we also take care of a small thing. So, when we have a ring a network over here right as shown you here the homophily based links and we know that there are going to be some weak ties as well.

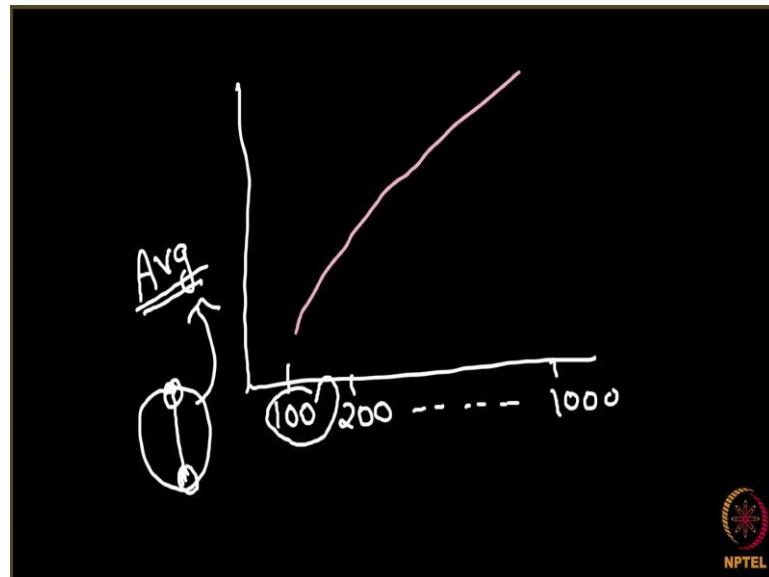
So, previously in this network we have taken 10 weak ties right, we have taken 10 weak ties. Now, if in this network is big right. So, what we are we will be going to assume in a programming screen cast is that the number of weak tie is 10 percent of the number of nodes in the network. So, for here we have 10 weak ties, here we will be making 20 weak ties in the next network which will be having 300 nodes.

(Refer Slide Time: 03:05)



We will be having 30 weak ties so on and so forth and then we will do this process. And, we will find the average over here and at the end what we will be plotting is, on the x axis we will be plotting the size of the network that is the number of nodes in the network 100 200 so on and so forth, let us say up to 1000.

(Refer Slide Time: 03:17)

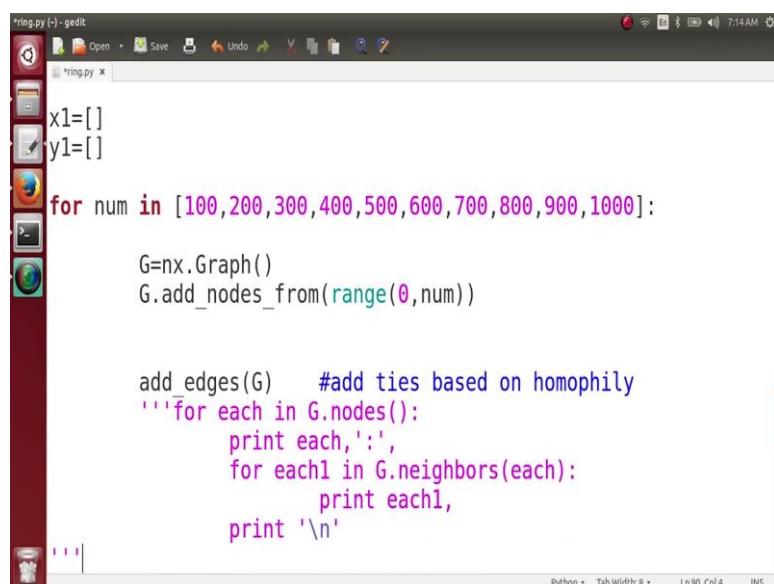


And on the y axis we will be plotting the average we have calculated. So, what is this average? It tells us that for one particular network, if we take the diametrically opposite points on this network. So, if we take this diametrically opposite points on this network

what is the average time taken by a myopic search and we will look at how does this plot look like is it linear, is it logarithmic or what. So, next what we want to do is now, we want to find out the time complexity of myopic search time which are myopic search takes for different size of networks.

So, till now we have considered only 1 network which was having 100 nodes. Now, we are going to do an experiment, we are going to build small world networks of different sizes 100 200 300 up to 1000 and we will do a myopic search there and we will see that how much time on an average does this take. So, at the end we want upload were on the x axis is the number of nodes and on the y axis is the average time taken by my myopic search. So, I make some arrays here.

(Refer Slide Time: 04:43)



```
ring.py (-) - gedit
x1=[]
y1=[]

for num in [100,200,300,400,500,600,700,800,900,1000]:
    G=nx.Graph()
    G.add_nodes_from(range(0,num))

    add edges(G)      #add ties based on homophily
    '''for each in G.nodes():
        print each,':',
        for each1 in G.neighbors(each):
            print each1,
        print '\n'
```

So, I make an array x1 for the x axis which will be my number of nodes and an array y1 for the y axis which will be the time which my myopic search is checking. And what I have to do is now I have to create networks of different size and do this entire process for all these networks. So, I have to basically put all these code in a loop. So, what I am going to do is for num in and I am going to take a list here consisting of the values 100, 200, 300, 400, 500 6 up to 1000. And, then what I am going to do is the graph, I am going to create here is going to be from range 0 to num. So, first of all I will get a network on 100 nodes then 200 nodes so on and so forth and I will put everything inside this loop.

(Refer Slide Time: 05:49)

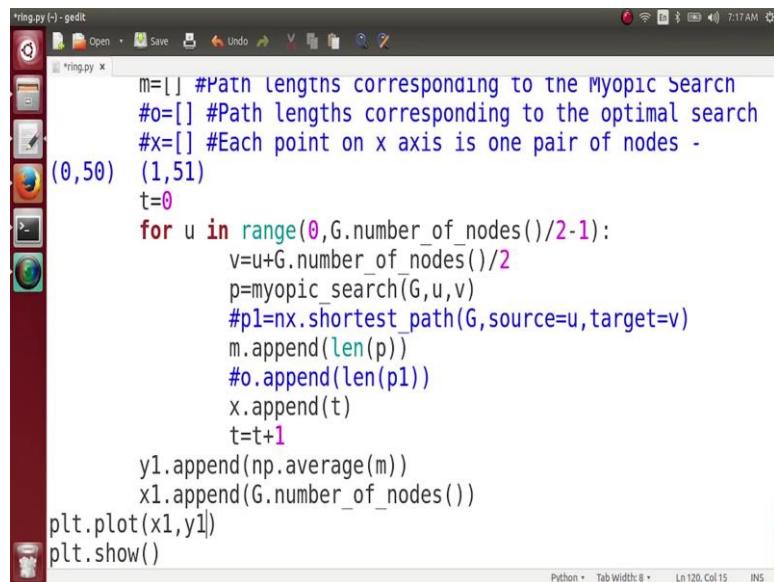
```
#ring.py (-) - gedit
ringpy X
H=G.copy()
#add_long_link(G)
x=[0]
y=[nx.diameter(G)]
t=0
while(t<=G.number_of_nodes()/10):
    add_long_link(G)
    t=t+1
    x.append(t)
    y.append(nx.diameter(G))
'''plt.xlabel('Number of weak ties added')
plt.ylabel('Diameter')
plt.plot(x,y)
'''

m=[] #Path lengths corresponding to the Myopic Search
```

And, now we are one thing to be noticed is `t` here `t` here it is telling us the number of weak ties which we have to add. So, the number of weak ties should change according to the number of nodes in the network. So, instead of 10 what I am going to do here is, I am going to make it here while `t <= G.number_of_nodes/10`; that is I am going to have 10 percent of the edge is in the network to be weak ties.

Whatever are the number of a nodes in the network so, if there are 100 nodes so, the number of nodes so, if there are 100 nodes I will be having 10 weak ties; if there are 200 nodes I will be having 20 weak ties so on and so forth. And, then I am going to add these long range links and rest of the things remains the same.

(Refer Slide Time: 06:59)

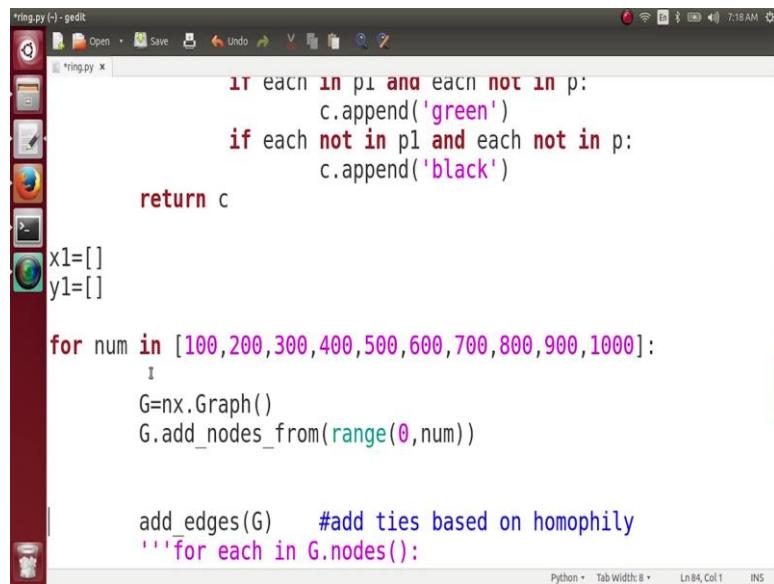


```
m=[] #Path lengths corresponding to the Myopic Search
#o=[] #Path lengths corresponding to the optimal search
#x=[] #Each point on x axis is one pair of nodes -
(0,50) (1,51)
t=0
for u in range(0,G.number_of_nodes()/2-1):
    v=u+G.number_of_nodes()/2
    p=myopic_search(G,u,v)
    #p1=nx.shortest_path(G,source=u,target=v)
    m.append(len(p))
    #o.append(len(p1))
    x.append(t)
    t=t+1
y1.append(np.average(m))
x1.append(G.number_of_nodes())
plt.plot(x1,y1)
plt.show()
```

Now, mine this value  $u$  will range what is on 0 to 49, it will range from 0 to  $G.\text{number\_of\_nodes}/2$  right. 0 to  $G.\text{number\_of\_nodes}/2 - 1$  and  $v$  is going to be  $u$  plus  $G.\text{number\_of\_nodes}/2$ . And, then we are going to apply a myopic search here, find out the path we do not, no we are not doing an optimal search over here and then I am wrote append length of  $p$  and here is an array. So, so you can see that  $m$  is an array over here which holds the path length corresponding to the myopic search. So, you see what is happening over here, for some small world network here we are performing a myopic search. So, we have a small world network, we have done this myopic search.

And we do it for many pairs of nodes and at the end what we want to append to our  $y$  axis is nothing, but the average of everything. So, what is going to come in our  $y$  axis is I will come out of this loop, what we are going to append in our  $y$  axis is the average which is  $\text{np.average}$ . And this is numpy,  $\text{numpy.average}$  of  $m$  right and what is going to be coming across  $x$  axis is nothing, but the number of nodes  $G.\text{number\_of\_nodes}$ . And at the end what will be, what we will be plotting is  $x1$  against  $y1$ ; I hope that this code is clear. What we have done is I will quickly recap it.

(Refer Slide Time: 09:39)



```
tring.py (-) - gedit
tring.py x
    if each in p1 and each not in p:
        c.append('green')
    if each not in p1 and each not in p:
        c.append('black')
    return c

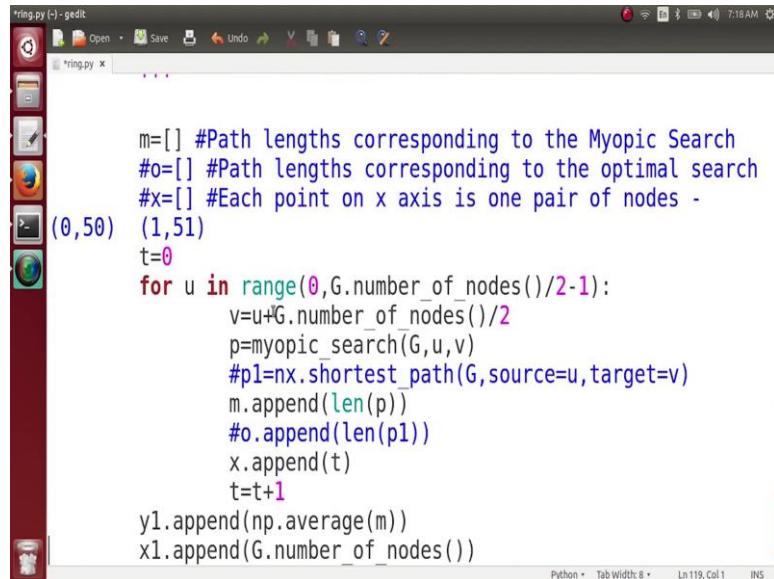
x1=[]
y1=[]

for num in [100,200,300,400,500,600,700,800,900,1000]:
    G=nx.Graph()
    G.add_nodes_from(range(0,num))

    add_edges(G)      #add ties based on homophily
    '''for each in G.nodes():
```

So, here are my 2 arrays x1 and y1, where x1 is the different. So, x1 is nothing, but thing here this complete will be x1 which is the number of nodes in different networks. And, y1 will hold the average path length which myopic search takes to connect a diametrically opposite points and then we make the small world network.

(Refer Slide Time: 10:05)

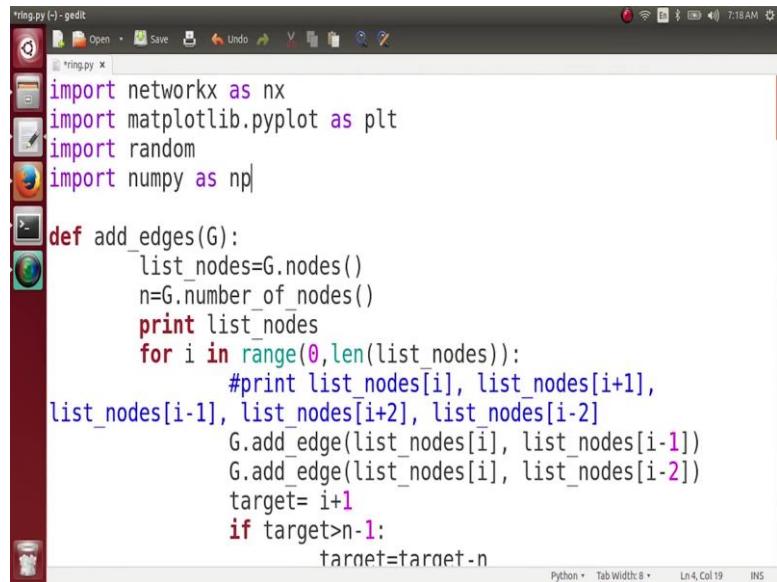


```
m=[] #Path lengths corresponding to the Myopic Search
#o=[] #Path lengths corresponding to the optimal search
#x=[] #Each point on x axis is one pair of nodes -
(0,50) (1,51)
t=0
for u in range(0,G.number_of_nodes()/2-1):
    v=u+G.number_of_nodes()/2
    p=myopic_search(G,u,v)
    #p1=nx.shortest_path(G,source=u,target=v)
    m.append(len(p))
    #o.append(len(p1))
    x.append(t)
    t=t+1
y1.append(np.average(m))
x1.append(G.number_of_nodes())
```

After making the small world network we take an array m be here and then we perform this myopic search for all the diametrically opposite points and keep a pending the path lengths in this array m. And at the end we take the average of this m and append it in y1

and this average of this m be append in y1 and x1 has nothing, but the number of nodes and we have using module numpy. So, that needs to be imported.

(Refer Slide Time: 10:35)

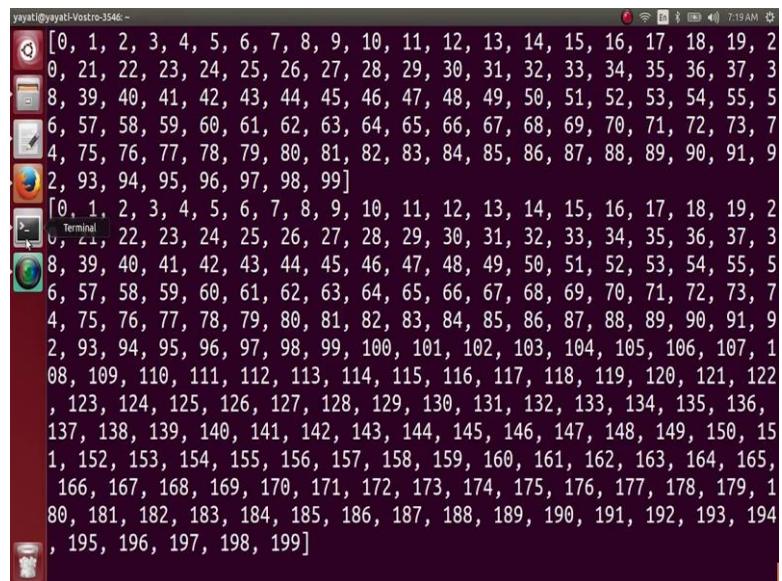


```
ring.py (~) - gedit
ring.py
import networkx as nx
import matplotlib.pyplot as plt
import random
import numpy as np

def add_edges(G):
    list_nodes=G.nodes()
    n=G.number_of_nodes()
    print list_nodes
    for i in range(0,len(list_nodes)):
        #print list_nodes[i], list_nodes[i+1],
        list_nodes[i-1], list_nodes[i+2], list_nodes[i-2]
        G.add_edge(list_nodes[i], list_nodes[i-1])
        G.add_edge(list_nodes[i], list_nodes[i-2])
        target= i+1
        if target>n-1:
            target=target-n
```

So, I import numpy as np means executes each.

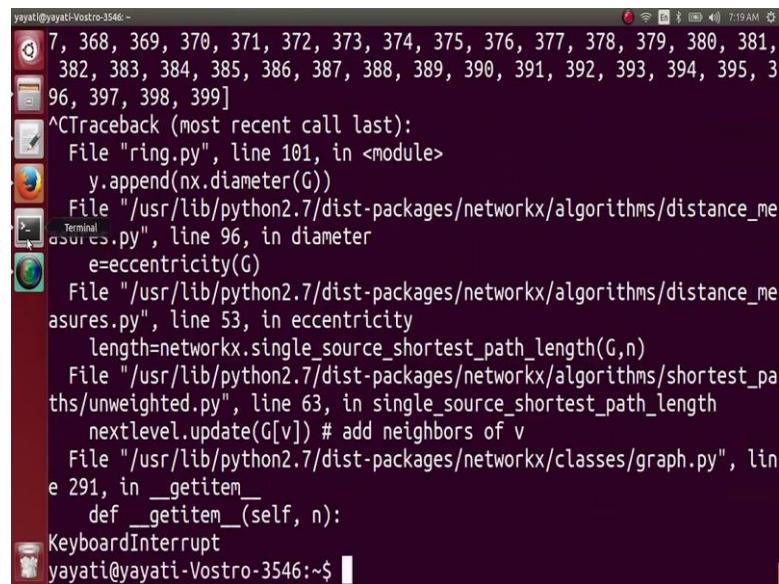
(Refer Slide Time: 10:41)



```
yayati@yayati-Vostro-3546:~$ [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99] [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199]
```

So, it will take some time.

(Refer Slide Time: 10:49)

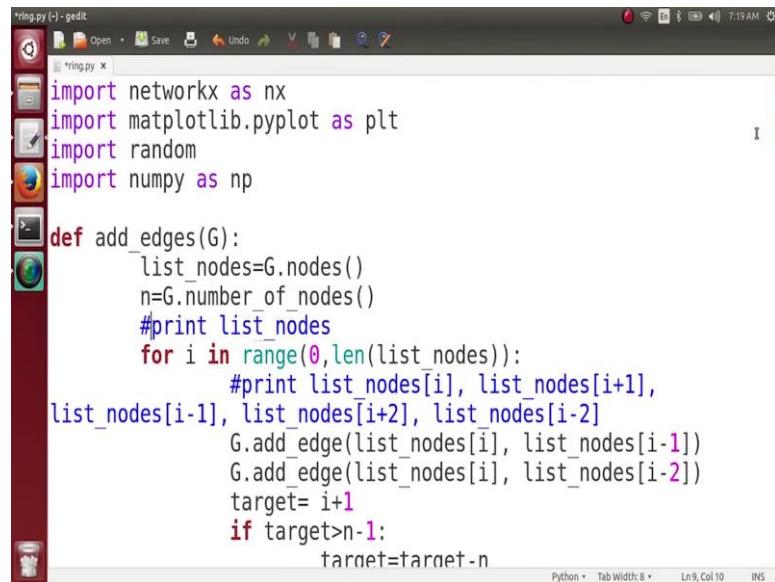


```
yayati@yayati-Vostro-3546:~
```

```
 7, 368, 369, 370, 371, 372, 373, 374, 375, 376, 377, 378, 379, 380, 381,
 382, 383, 384, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 395, 3
96, 397, 398, 399]
^CTraceback (most recent call last):
  File "ring.py", line 101, in <module>
    y.append(nx.diameter(G))
  File "/usr/lib/python2.7/dist-packages/networkx/algorithms/distance_me
asures.py", line 96, in diameter
    e=eccentricity(G)
  File "/usr/lib/python2.7/dist-packages/networkx/algorithms/distance_me
asures.py", line 53, in eccentricity
    length=networkx.single_source_shortest_path_length(G,n)
  File "/usr/lib/python2.7/dist-packages/networkx/algorithms/shortest_pa
ths/unweighted.py", line 63, in single_source_shortest_path_length
    nextlevel.update(G[v]) # add neighbors of v
  File "/usr/lib/python2.7/dist-packages/networkx/classes/graph.py", lin
e 291, in __getitem__
    def __getitem__(self, n):
KeyboardInterrupt
yayati@yayati-Vostro-3546:~$
```

For the sake of clarity where there is code is running correctly or not, but we are going to do is where is my first print statement; we will remove that print.

(Refer Slide Time: 11:07)

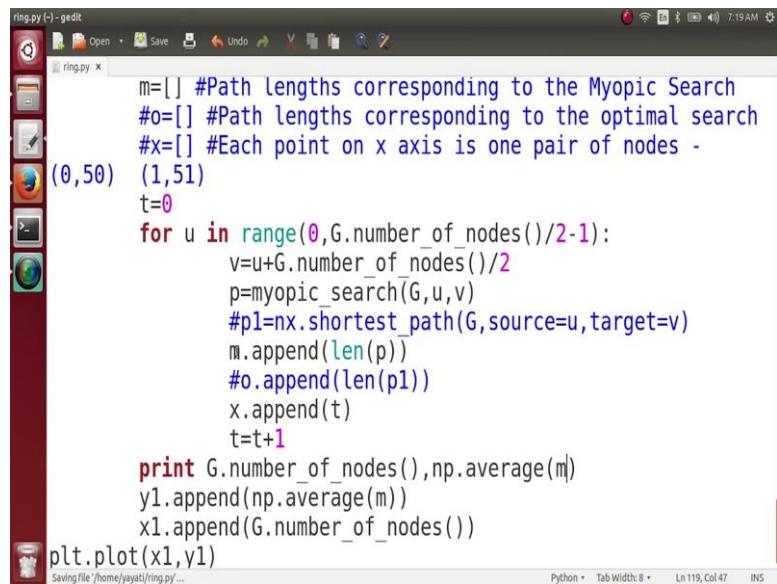


```
*ring.py (-) - gedit
import networkx as nx
import matplotlib.pyplot as plt
import random
import numpy as np

def add_edges(G):
    list_nodes=G.nodes()
    n=G.number_of_nodes()
    #print list_nodes
    for i in range(0,len(list_nodes)):
        #print list_nodes[i], list_nodes[i+1],
        list_nodes[i-1], list_nodes[i+2], list_nodes[i-2]
        G.add_edge(list_nodes[i], list_nodes[i-1])
        G.add_edge(list_nodes[i], list_nodes[i-2])
        target= i+1
        if target>n-1:
            target=target-n
```

We do not print the list of nodes here, instead what we do here whatever we are appending we print here.

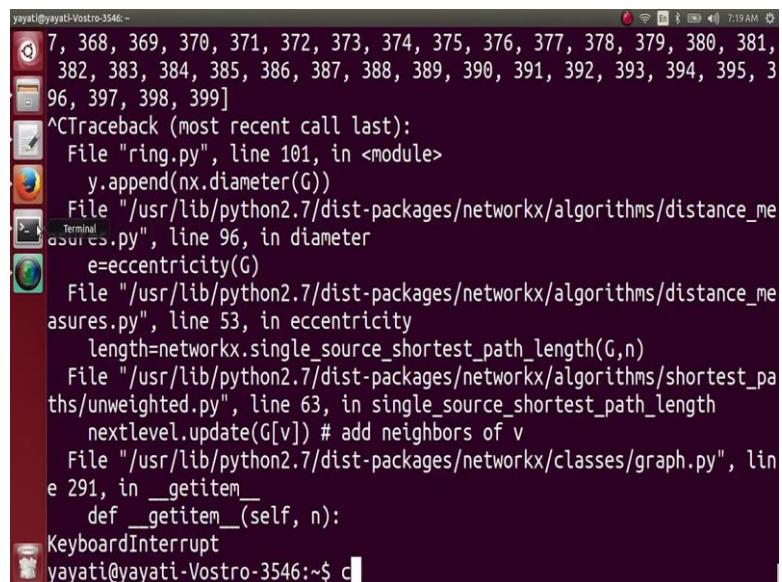
(Refer Slide Time: 11:15)



```
m=[] #Path lengths corresponding to the Myopic Search
#o=[] #Path lengths corresponding to the optimal search
#x=[] #Each point on x axis is one pair of nodes -
(0,50) (1,51)
t=0
for u in range(0,G.number_of_nodes()/2-1):
    v=u+G.number_of_nodes()/2
    p=myopic_search(G,u,v)
    #p1=nx.shortest_path(G,source=u,target=v)
    m.append(len(p))
    #o.append(len(p1))
    x.append(t)
    t=t+1
print G.number_of_nodes(),np.average(m)
y1.append(np.average(m))
x1.append(G.number_of_nodes())
plt.plot(x1,y1)
Saving file '/home/yayati/ring.py'...
```

So, we print here whatever we have appended in the x axis G.number\_of\_nodes and we print here np.average(m).

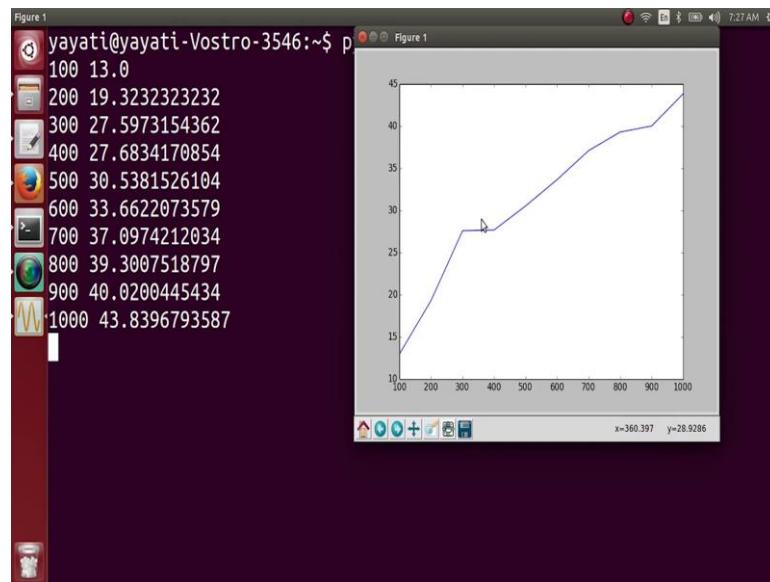
(Refer Slide Time: 11:29)



```
yayati@yayati-Vostro-3546:~$ 
7, 368, 369, 370, 371, 372, 373, 374, 375, 376, 377, 378, 379, 380, 381,
382, 383, 384, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 395, 3
96, 397, 398, 399]
^CTraceback (most recent call last):
  File "ring.py", line 101, in <module>
    y.append(nx.diameter(G))
  File "/usr/lib/python2.7/dist-packages/networkx/algorithms/distance_me
asures.py", line 96, in diameter
    e=eccentricity(G)
  File "/usr/lib/python2.7/dist-packages/networkx/algorithms/distance_me
asures.py", line 53, in eccentricity
    length=networkx.single_source_shortest_path_length(G,n)
  File "/usr/lib/python2.7/dist-packages/networkx/algorithms/shortest_pa
ths/unweighted.py", line 63, in single_source_shortest_path_length
    nextlevel.update(G[v]) # add neighbors of v
  File "/usr/lib/python2.7/dist-packages/networkx/classes/graph.py", lin
e 291, in __getitem__
    def __getitem__(self, n):
KeyboardInterrupt
yayati@yayati-Vostro-3546:~$ c
```

So, now.

(Refer Slide Time: 11:33)



So, when I have a network was of a was consisting of 100 nodes it took 13 steps 200 nodes 19 steps so on and so forth. It will be time to run and let us see the final output and you can actually do this process for a greater number of nodes. So, when you are code instead of 100 200 300, you can take 100 150 200 250 to get more data points and have a better plot ok. So, here you see plot and what if you will plot it for more data points you see the plot more clear and you can actually observe that this is not a linear plot rather it is a lower plot. So, as you increase the number of nodes, the time which myopic search takes to execute increase is logarithmically not linearly.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**How to Go Viral on Web**  
**Lecture - 159**  
**Pseudo Cores: Introduction**

(Refer Slide Time: 00:07)



We have studied about cascades, we have seen how people follow each other and this following can be in terms of behavior, can be in terms of adopting an idea, can be in terms of adopting a product. And, we looked at an entire chapter based on this concept of following each other and cascading and we also talked about epidemics.

(Refer Slide Time: 00:29)



So, how do diseases spread from person to person and we have looked at various models for how we can model this like SIS model, SIR model etcetera. And, we have also talked about the subtle differences between the spread of an idea and the spread of a disease.

(Refer Slide Time: 00:47)



So, we have seen that when a disease spreads there is no choice, when an idea spreads there is a choice. So, if an idea comes to you, you make a choice if you do not like the idea, you have the liberty to say no.

(Refer Slide Time: 01:05)



I do not like this idea, but if it is a disease and somebody infected person comes to you and the contagion is happening, you cannot say I do not like this disease I am not going to take it. So, there you do not have any liberty. So, in this way the spread of a disease and the spread of an idea are different. Spread of an idea has become a complete research direction in its own; there are a lot of things we can study on the spreading of an idea.

(Refer Slide Time: 01:39)



And today ideas spreads like anything unlike the past days. Why? Because today all the ideas they spread on internet. So, whether it is an image or it is an video or it is an text or

let us say it is about a riot that people want to do. Everything today spreads with the help of internet through Facebook, through twitter, through WhatsApp and all these platforms. So, first of all I would like to tell you about a term internet meme, most of you might have heard about it also.

So, internet meme is actually not a technical word, it is not anything, it is just anything which spreads through internet is called an internet meme. It can be a photo an image or video or text or anything whatever spreads through internet is called an internet meme. And, today this internet memes they carry the capability of changing your life unlike the past days; if you want to become popular you do not need to find a right platform anywhere else, you can just upload a video on internet and become popular.

(Refer Slide Time: 02:53)



So, do you know the story of Justin Bieber? How Justin Bieber became Justin Bieber, the story is very interesting. So, initially in his when he was like 11 or 12 years old, he and his mom they used to record his videos. So, he used to sing at his school level or at his school levels and whenever he used to have a nice concert or nice singing they will record his video and they will post it on YouTube. And, when they post it on YouTube slowly the number of likes number of views started increasing.

And, then there was his producer or director which had his eyes on what is happening there. He looked there is a small, there is a talent a very small boy having this big talent and then he got and grab Justin Bieber and brought him to the singing industry. And,

today we do not also need a mentor like that to pull you out; you can mainly go and keep uploading your videos on internet. Your videos, pics or anything on internet and as it gains a greater number of likes you will become popular. So, we do not need anything else.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**How to go Viral on Web**  
**Lecture - 160**  
**How to be Viral**

Although, I told you very easy way to become popular seems to be easy, do you think it is actually easy.

(Refer Slide Time: 00:21)



A population of 1 billion people everybody trying to appear on internet and getting popular, how easy is it for you to make your mark and go popular there; actually very difficult. There are lot of internet memes today and people have limited time. So, we cannot see all of those memes we just see the ones which may be our friends recommenders or which are really very very interesting and then go back to our work. So, in such a culture, in such a scenario, in such a situation it is actually very difficult to make your meme popular or rather we say it viral.

So, like diseases become viral some memes become viral and they are adopted by almost entire population. For example, Gangnam style meme is viral, Nyan cat meme is viral. So, how do you make your meme viral? How do you come to know that picture you updated on Facebook the last night whether it is going to become viral or not.

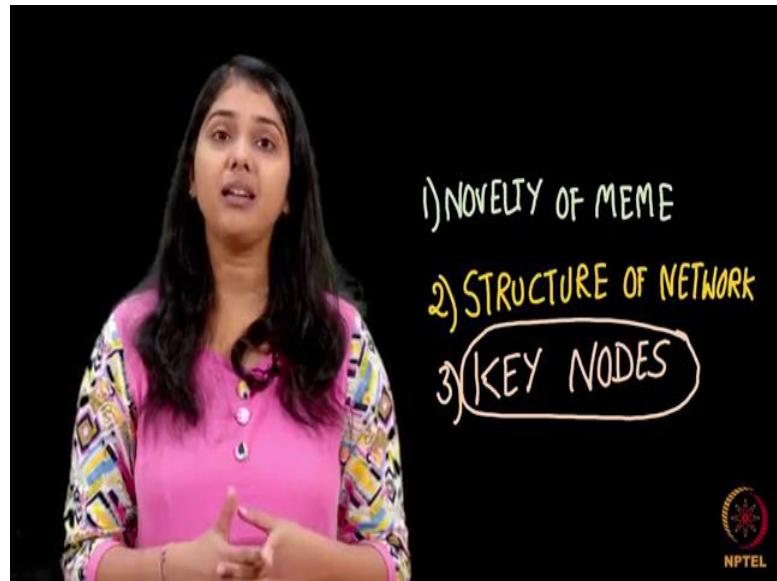
(Refer Slide Time: 01:23)



And, again we came back to the same question which we discussed in the chapter cascades, where we were concerned about the sport shoes my friends just now developed. So, he wants to know whether his sports shoes will become popular and will everybody buy them, or it will the cascade will quickly die out after some steps.

Now, how do we come to know that and like there the answer is again to 2 folds. So, one it is about the information you are sharing, it is about the quality of the sports shoes. So, it is about the quality of your meme. The novelty in your meme, do you have a nice background music or do you are you bringing something new to the platform; obviously, that matters what is inside your meme, what is inside your video or your picture your are releasing.

(Refer Slide Time: 02:17)



And, second, we looked at the structure of the network. So, some networks they are designed in such a way that they make everything popular and some networks they are designed in such a way that they do not let anything become popular.

But we cannot play here with the structure of the network structure of the network is same for everybody. So, the structure of the Facebook, structure of YouTube network, structure of Twitter network; structure means how the people in this network are connected. So, people are the nodes and the edges between these people that is the friendships and the followships are the ties. So, the structure is the same for every people and the structure is such that some of the memes become viral, some of the memes do not become viral. So, we cannot meddle much with the structure here. So, what do we do?

Do you remember the third idea which was particularly interesting, and the idea was about key nodes? So, if you want to get that cool bike it should be your dad whom you should be approaching not your spouse or your mother. In the similar way, if you want your information to become viral you probably make a key node, or a popular person talk about it. So, it is like if you convince Amitabh Bachchan or let us say Narendra Modi to re-tweet your post or to comment on your let us say your Facebook post no doubt it will become viral. No doubt more and more people will start liking it, will start sharing it, looking that somebody popular is liking your tweet.

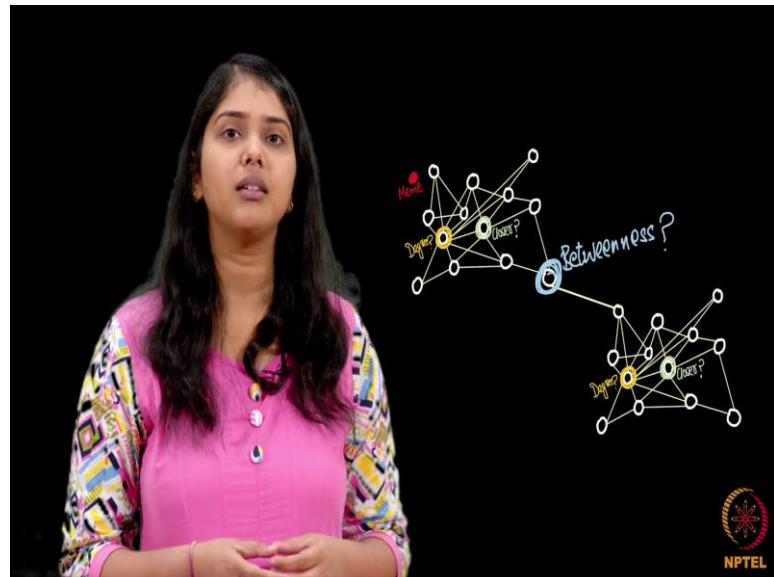
So, that is the third way of doing; that is about choosing the right people for sharing your meme, infecting the right people with your meme.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**How to Go Viral on Web**  
**Lecture - 161**  
**Who are the right key nodes?**

So, we now know that these should be the key people which we should be infecting, but how do we identify this key people. Who are these key people we should infect; assume I give you a network as shown here?

(Refer Slide Time: 00:19)



Can you look at this network and tell me which is the node here, which I should infect; so, that this meme which is travelling on this network which is rather diffusing on this network becomes viral, which should be the node here? And so, there are many possible criteria which you can look at, is it the node which is having maximum number of friends, having the highest degree we say. Should it be that node which we should look at and attack or should it be a node which is structurally closer to all the nodes in the network. Closer in the sense, if you look at its distance from every other node in this network the distance is small.

So, it can quickly reach all the nodes in this network that is actually called close the centrality and we will be discussed in another lecture. But, just for the intuition it just

says that this node is closer to all other nodes in the network or should it be a node having high betweenness. Betweenness we have looked at before, tells you how good you are in connecting people.

So, if you look at this node, here if we have these two components and there is this node in between. So, it has a high betweenness because, for going from this component to the other component you need to go through this node. So, should it be this node which I should be infecting or there should be some other criteria. So, how shall I find such a node in a given network; so, that if I infected with an idea or with my internet meme perhaps my meme should become viral.

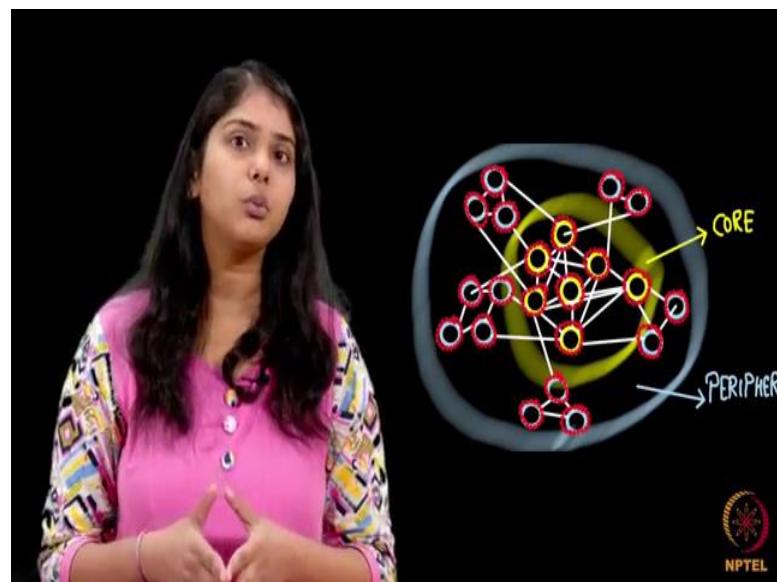
**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**How to go Viral on Web**  
**Lecture - 162**  
**Finding the right key nodes (the core)**

So, here I tell you the answer how should you choose these key people. Do you remember we talked about this structure in one of our chapters in a small world phenomenon? We talked about this structure and it is called a core periphery structure.

So, we have looked that there are certain people in this world some small fraction of people who are rich, who have access to better resources, who can travel throughout the world and such people can manage more connections and rather connections to different-different parts of the world and they lie at the centre of a network something like this.

(Refer Slide Time: 00:45)

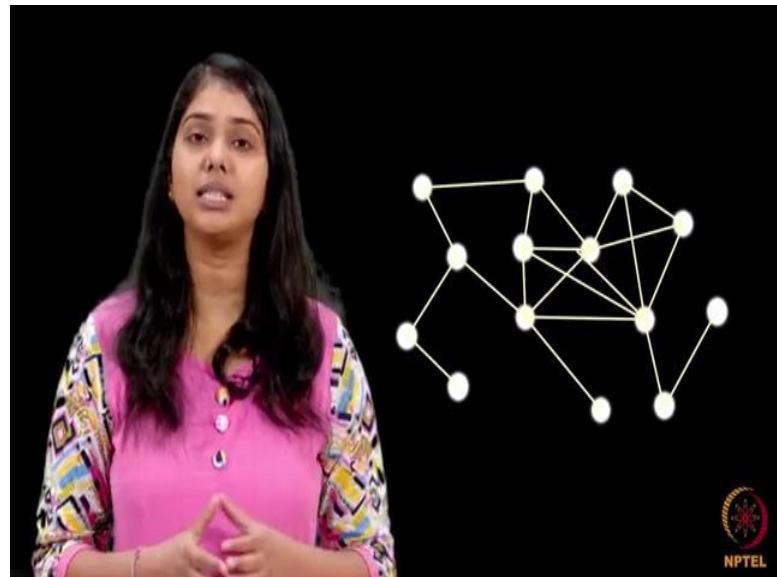


And rest of the people like a periphery are towards the periphery of this network they are towards the boundary. So, we talked about this structure before. This structure has a very big role to play here. So, it has been shown that if you infect one of these core nodes here you see what happens. So, if you closely look at this figure, you will see that these core nodes here they are very densely connected to the other core nodes.

So, if you infect one of these core nodes, it will quickly tell another core node, which will quickly tell another core node and soon this entire core will come to know of your idea. And we know that this entire core is now very nicely connected to all the people in the world to all the periphery. So, once this component is infected, your mean will go and reach all the periphery nodes as well. So, this should be the core nodes which you should be infecting ok.

Till now, well and fine we come to know, we talk about the core periphery structure and we say that if this is the core which is infect my mean is going to become viral. Now I give you a network here can you tell me what is the core here?

(Refer Slide Time: 02:05)



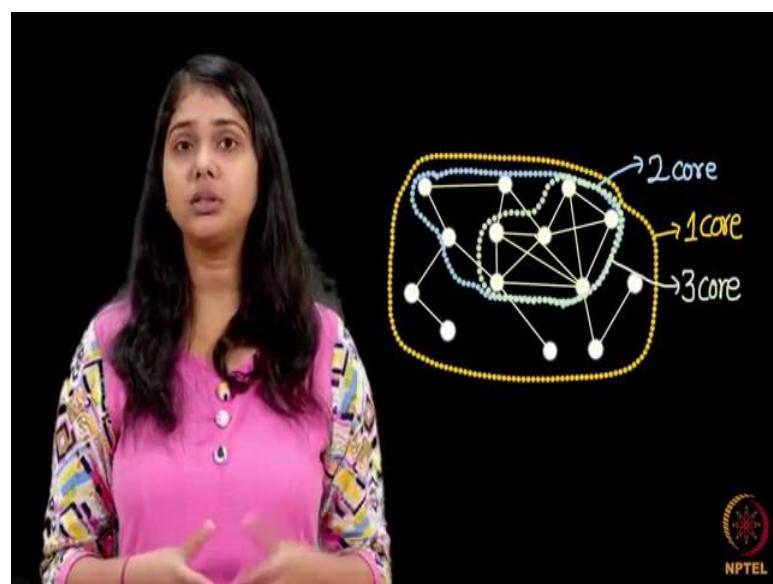
How do we even come to know what is core of a network, what lies in the core of a network? And then people have tried answering this question. And I will now discuss one of the most popular approach which you can use to identify these score nodes, identify these score nodes in this network. So, for that we will just switch to a definition a mathematical definition, but very simple and the definition is that of a K core.

(Refer Slide Time: 02:37)



So, given a graph let us say we are given a graph here, we say a node or we say not a node we take a sub graph from this graph and we say that the sub graph is 1 core. 1 core if every single node in the sub graph has the degree of at least 1. And similarly, you say a sub graph to be off 2 cores to be 2 cores if you look at every single node in this sub graph and every node has a degree of at least 2. So, for and so forth, so, you have 1 core, 2 cores, 3 cores, 4 core and so on. So, that is what defines a K core. You take a sub graph and every node in this sub graph should have a degree at least K ok.

(Refer Slide Time: 03:29)



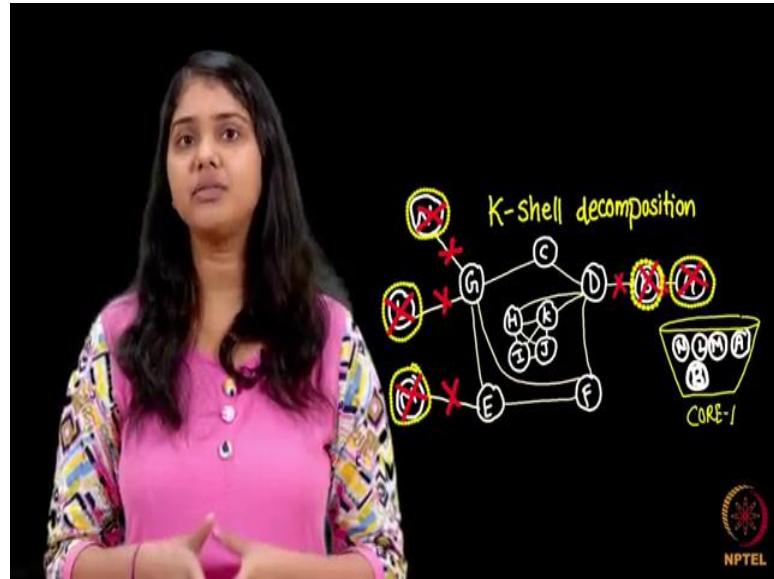
So, here is a graph. So, we can quickly see and try to figure out what is 1 core here, what is 2 cores, what is 3 cores. So, if we look at it and introspect it introspect on it a little bit we come to know that yes fine this is 1 core, this is 2 cores and this is 3 cores, but now what I am going to do? This is a small network. It was easy for you to look at it and then tell me what 1 core is, what is 2 cores, what is 3 cores.

(Refer Slide Time: 04:03)



I will give you a network having billions of nodes like Facebook or Twitter, I give you the network here and now can you tell me what is 1 core here, what is 2 cores, what is 3 cores and so on. It becomes very difficult. So, there is a very nice algorithm to do it. It is actually intuitive if you little bit think about it, you can converge to this algorithm on your own. And the algorithm is very interesting as well So, this algorithm determines what goes in 1 core, what goes in 2 cores, what goes in 3 cores and so on and it is known as K shell decomposition algorithm.

(Refer Slide Time: 04:39)



K shell decomposition algorithm because, it decomposes your network into multiple shells. Now, let us see how does this algorithm work. So, you look at your network. So, here is your network. And let us say your network is some instead of nodes I use the word ball here. So, your network has many balls and these balls are connected to each other.

Then just for some time I am using the word ball instead of the word node. And then I come with a bucket and I label this bucket as 1 core and put this bucket here. And then I look at this network, I look at all the balls, all the nodes which are having degree 1; which means which have only 1 friend, we also call them pendant vertices if you know graph theory otherwise you can just leave it.

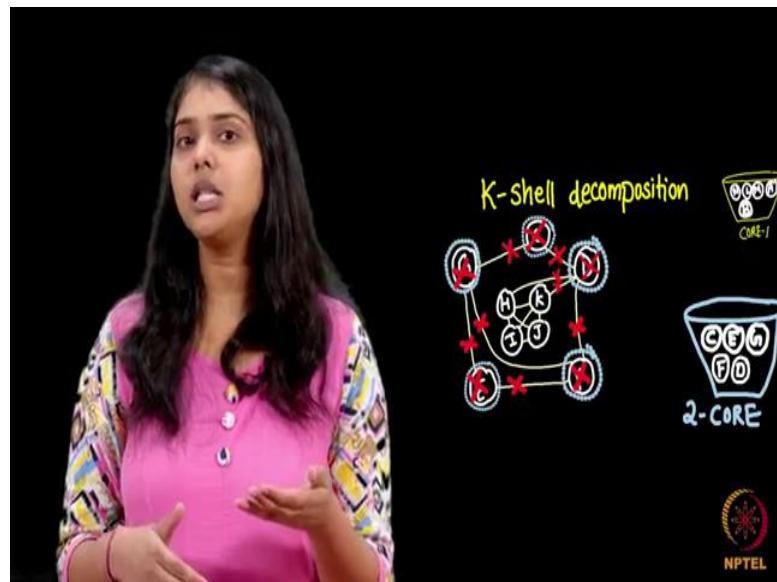
So, the just have degree 1 they are kind of attached to the network with just 1 edge. I take these nodes I plug them out and put them in this bucket. So, when I plug a node out please note that all the edges which are outgoing all the edges which are incident on this node also gets plugged out.

So, when I plug out a node, I cannot have a hanging edge in this network right. So, that edge is remove and I put that ball in this bucket. And then I keep doing it, I look at another ball which is having degree 1, put it in the bucket, I keep doing with all the balls and I look at all the balls having degree 1 and put it in the bucket.

Did you notice something here? When I pull a ball having degree 1 another ball which was previously having degree 2 can now become a ball having degree 1. So, that is when you keep removing these balls with degree 1, new balls keep immersing in your network which has degree 1. So, if you look at this node A here, when I plug this node A out, keep it here, you see this node B, it has now become of degree 1.

So, what do you do? You also remove node B and put it in the basket. So, mainly you keep doing this business of plugging out the nodes and keeping them in the basket till there is no degree 1 ball in your network. Till all the nodes here they have degree 2 or a higher degree. And once you achieve such a graph which has nodes of degree 2 or higher degree you stop and you put your bucket aside then you bring a fresh bucket and then we label this bucket as 2 cores.

(Refer Slide Time: 07:21)



And you now can guess what I am going to do is. So, I put this bucket having label 2 cores here and I again locate all these balls in my network. And now I look at the balls which have degree which have degree 2 that is with 2 edges they are connected to this network.

So, for example, the node C here, you can see that it is connected with degree 2 to this network with 2 edges. So, I plug C out and keep it in the basket. I do it for all the balls which are having degree 2. I take all those balls and put those balls in this basket. And again did you notice the same thing happens when you plug a ball out which is having a

degree 2, a new ball having degree 2 might immerge and rather a new ball having degree 1 might immerge.

So, you see here when I remove this ball having this when I remove this ball E from this network having degree 2, this node F immerge is which has just degree 1. What I do, what should I do now with this degree one node? Shall I keep it in my previous bucket or shall I keep it in this bucket? So, I will tell you I keep it in this bucket. This market means the bucket label 2 cores. So, I keep removing the balls having degree 2 or lesser. So, in this iteration what all goes in this bucket 2 all the nodes having degree 2 or lesser.

(Refer Slide Time: 09:09)



And you keep plugging them out and new balls keep immerging which might have a degree 2 or lesser you plug those balls also out and put them in this same bucket and you keep doing; so till there are no nodes of degree 2 or lesser in the network. So now, in this network there will only be nodes having degree 3 or a higher degree.

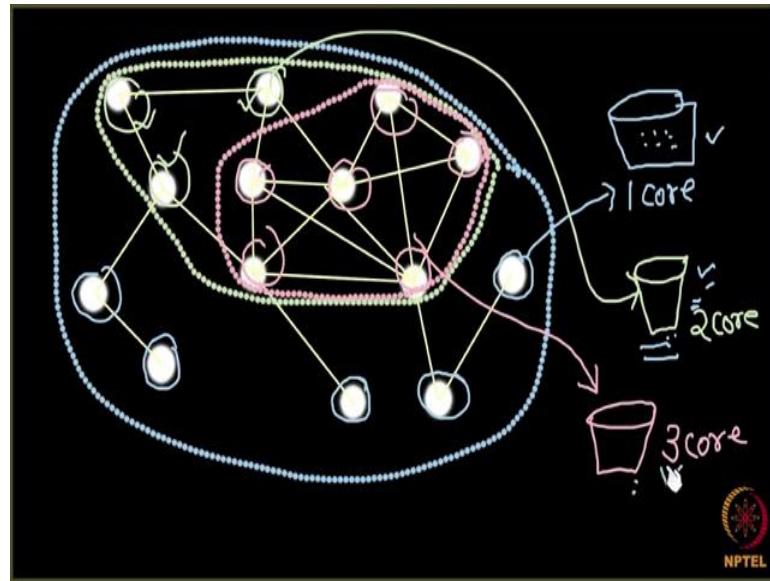
And when this is done, I pack this bucket and keep it aside and I bring the third bucket in picture, I label it as 3 cores and I start repeating the same procedure again. Plug all the nodes having degree 3 from this network and new balls might immerge which might be having a degree 3 or lesser and I keep all of them in the same bucket and I keep doing it. And we keep doing this procedure till my network becomes empty.

So, we are plugging nodes at every step. So, when we keep plucking nodes obviously one state should come when my network becomes empty and that might be at some K core at some K-th bucket, we do not know it can be a 10th bucket or it can be the 9th bucket.

Now, we have K cores. What we talk previously was about core and periphery; we were interested in finding the core of the network. So, what is the core of this network? So, when we do this K shell decomposition to find 1 core, 2 cores and 3 cores and so on. The nodes which are removed at the last step the most let me see the innermost core of the network, innermost shell of the network that is called the core.

So, in this network where we were having 1 core, 2 cores, 3 cores, the core of this network is this the 3 cores. And when we do k shell decomposition whatever goes in the last bucket, whichever nodes go in the last bucket form the core of that network. I want to tell something important related to K core, 2 cores and K shell decomposition.

(Refer Slide Time: 11:07)



So, we have looked at this network and, in this network,, we have looked at which node comprise of which nodes coming 1 core, which nodes coming 2 cores and which nodes coming 3 cores. So, we have seen that in this network the sub graph in which all the nodes have degree greater than or equals to 1 come in come in 1 core.

So, this entire network used to come in 1 core. So, this is the 1 core of this network. And then we have looked at 2 cores in which all the nodes have degree greater than or equals

to 2. So, this particular portion in this network is the 2 cores. And then similarly we have looked at that this particular portion in this network is the 3 coress. And then next we will discuss the techniques to determine the 1 core, 2 cores and 3 coress in the network which was called K shell decomposition. Now what is important here is if you remember that in K shell decomposition, we made several buckets right. So, there was a bucket corresponding to 1 core.

So, let us say this was a bucket corresponding to 1 core and then we had a bucket corresponding to 2 cores and 3 coress also. So, in this bucket corresponding to 1 core, which all were the nodes in K shell decomposition and if you will apply K shell decomposition over this network, you will note that these nodes over here were the ones which were present in this bucket corresponding to 1 core.

Similarly, if we look at the bucket corresponding to 2 cores then we saw that these nodes over here 1, 2, 3 1, 2, 3, so these 3 nodes were in bucket of 2 cores. And then similarly we had a bucket of 3 cores over here So, this is a bucket for 3 cores and all the nodes, so 1, 2, 3, 4, 5 and 6, the 6 nodes over here were in the bucket for 3 cores So, we had these 3 buckets.

Now, what is important here is please note that all the nodes which are present over here in the bucket for 1 core. So, if you are asked over this network what is the 1 core of this network?

(Refer Slide Time: 13:31)

$$\begin{aligned}
 1\text{-core} &= B_1 \cup B_2 \cup B_3 \\
 2\text{-core} &= B_2 \cup B_3 \\
 3\text{-core} &= B_3 \\
 \\
 k\text{-core} &= B(k) \cup B(k+1) \cup B(k+2) \dots \\
 &= \bigcup_{j \geq k} B(j)
 \end{aligned}$$



So, 1 core of this network is according to K shell decomposition 1 core of this network is the nodes which are present in bucket corresponding to 1 core. Let us say that bucket 1 union bucket corresponding to bucket corresponding to 2 cores which is B 2 union bucket corresponding to 3 cores which is B 3 right.

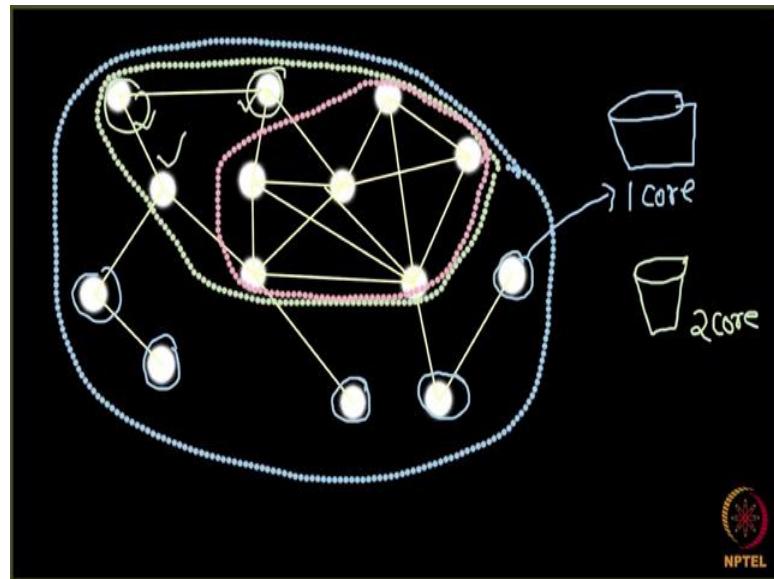
So, in this network if you see over here what is 1 core, so the 1 core in this network is all the nodes over here, union all the nodes over here, union all the nodes over here because, whatever is in this bucket for 2 cores is definitely present in the bucket for 1 core. So, here we have all the nodes which have degree greater than or equals to 2, so automatically this will be having degree greater than or equals to 1.

So, whatever is in this bucket is automatically in 1 core, whatever is there in this market is automatically in 1 core. Similarly, whatever is there in this bucket is automatically in 2 cores. So, the 2 cores of the network is, so if I want to write 2 cores in this network is so 2 cores is bucket 2 union bucket 3.

And if I want to write 3 cores in this network is 3 cores is my simply bucket 3. If you want to generalize this formula and write; if you want to generalize the formula and write it down we can say that when we try to determine the K core in a network with the help of K shell decomposition, we can say that K core of the network is, what is K core of the network is? It is bucket k, let us write it like this, bucket k union the nodes in bucket k plus 1 union the nodes in bucket k + 2 so on and so forth; that it is the  $\bigcup_{j \geq k} B(j)$ .

So, that is how would how you can determine the K core in network with the help of K shell decomposition.

(Refer Slide Time: 15:41)

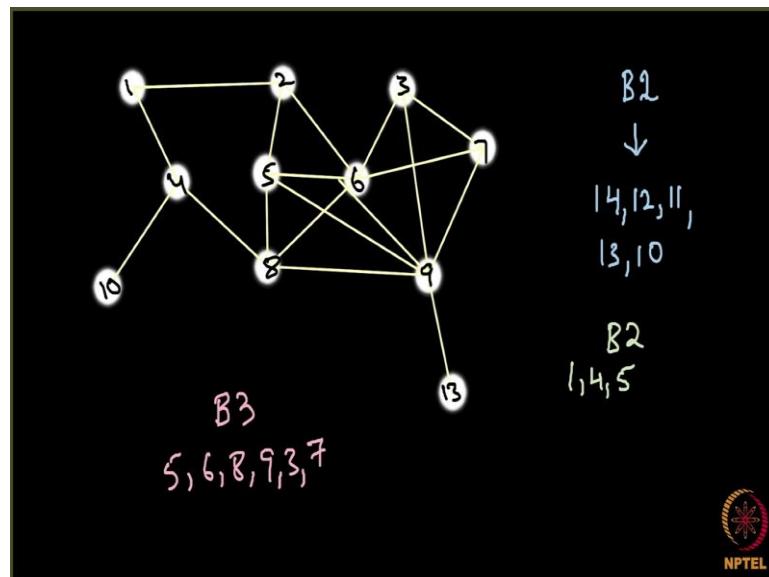


So, the overall motive of this video was to tell you that the buckets here do not correspond to the actual core numbers here; so, if you actual K core here. So, if you want to determine 1 core in this network, you cannot simply take the nodes which are present in bucket 1, you have to union it with the nodes in the bucket 2 and bucket 3 so on and so forth till the maximum bucket.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**How to go Viral on Web**  
**Lecture - 163**  
**Coding K-Shell Decomposition**

(Refer Slide Time: 00:05)



In the next programming screen cast we are going to implement facial Decomposition. So, I hope you remember you remember what was the algorithm for facial decomposition. First of all, we take bucket 1 and then after taking bucket 1 we start removing the nodes having degree 1.

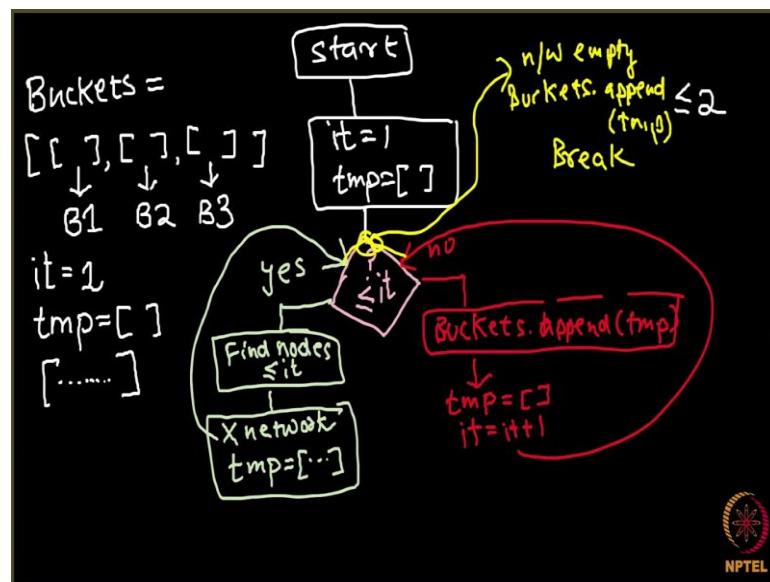
So, we see that this node over here these nodes have degree 1, this node has degree 2. So, these 3 nodes have degree 1, so what we are going to do, we are going to remove these 3 nodes and put them in bucket 1. So, let me number these nodes also. So, what I am going to do is let me number these nodes first ok. So, we are going to take bucket 1 and in bucket 1 will recursively put the nodes who have degree 1.

So, first of all nodes 14, 12 and 11 go to bucket 1 because, they have degree 1 and then we delete these nodes from this network. So, we cut node 14, 12 and 11 from this network and we put them. And next again we see that there is degree 1 in this network and then we recursively remove these nodes also from the network and put them in

bucket 1; similarly, we determine bucket 2 and bucket 3 in this network. So, bucket 2 will comprise of the nodes 1, 4 and 5 and bucket 3 will comprise of the nodes 5, 6, 8, 9, 3 and 7.

So, how do we write a code for this? So, before writing the code here I am going to tell you the structure of the code, so that it will be easy for you. Decode is a little bit involved, so it is important to understand its structure before writing the piece of code ok. So, let us take this network and we do not even need this network ok.

(Refer Slide Time: 02:15)



So, what we will be doing is let us say that here a good start. So, this is starting. What we are going to do is we know that we have to recursively remove the nodes having degree 1 or less than 1 till there are no degree 1 nodes in the network. Next, we have to recursively remove the nodes having degree less than or equal to 2 from the network till there are no nodes having degree less than or equal to 2 in the network so on and so forth.

So, in the beginning I am going to take some variables. So, I have a first of all I will take a list here and I name this list as buckets. What is this list bucket? This list buckets is going to be a lists of lists. So, it will be further consisting of many lists and this will be the list corresponding to my bucket 1. This will be my list corresponding to bucket 2, bucket 1 here means bucket corresponding to pour 1, bucket corresponding to pour 2, and bucket corresponding to pour 3.

So, this is my array buckets and the aim of our code is to fill this array buckets. And then I am going to take a variable let us say it which is initially 1. What this variable tells me? This variable tells me at a particular time in a code which degree nodes I am removing. So, if the value of it is k let us say it means that I am removing the nodes having degree less than or equals to k.

So, that is my variable it. And then we have a array or we have a list tmp. What this list tmp is going to hold is, so we are going to determine 1 of bucket at a time. So, I might be filling this bucket at a particular time, I might be filling this bucket or this bucket. So, t is going to temporarily hold this bucket. So, initially the value of tmp is going to be initially, so tmp will start with empty and then whatever values will put in tmp will be the values in B1.

So, we will fill tmp, we will keep filling tmp and a point will come where all the nodes have been degree less than or equals to 1 will be removed from this network. At that time we are going to close tmp and we are saying that we will say that this tmp is nothing but bucket 1 and we will append tmp here, it will be further clear ok.

So, what we are going to do is we have a start here. After start we are going to check a condition. So, initially we have it equals to 1 and tmp is nothing, but empty ok. Next what we are going to do? We are going to check a condition and what is this condition? This condition is whether there are nodes of degree greater than sorry, whether there are nodes of degree less than or equal to it; in this network whether, there are such nodes in the network and the answer can be yes or no.

So, let us say here the answer is yes and here the answer is no. So, if the answer is yes what we are going to do? If the answer is yes it means that, still there are nodes of degree less than or equals to it in this network. So, we have to prune those nodes from the network. So, what we will be doing is what we will do is find those nodes.

So, we find these nodes which have the degree less than or equals to it and what do we do? We remove these nodes from our network. So, we remove these nodes from our network and we append these nodes in our array tmp, in our list tmp which is the currently being filled bucket.

So, we append these nodes in tmp. That is what we do and after doing this we return back to here. So, when the value of it was 1 here what do you do, you find the nodes which have degree less than or equals to 1, you delete those nodes from the network and put them in your bucket and after that you again see whether you; I am very sorry, you come here you come here ok.

So after that, after removing these nodes having degree 1, you again see whether still there are nodes having degree less than or equal to 1 in your network and again if the answer is yes you remove those nodes also and you keep doing this till there are no nodes having degree less than or equals to 1 in the network. So, if the answer is no what will you do? It says that your bucket is over.

The bucket which you are currently filling is over, so what you do? At this point if the answer is no you append your tmp to your buckets. So, buckets.append(tmp) that is your current bucket is full, so you do this and what else you do. Now we have to start filling a new bucket. So, we set back tmp to empty and also, we have removed all the nodes having degree it. Now we have to start removing the nodes having degree it plus 1. And after doing this we go back to these things.

So, I hope the code structure is clear now. So, this loop will keep running. So, it will check for the value of it equals less than or equal it will be 1. So, we check whether there are nodes of degree less than or equals to 1. We keep doing this, we keep coming back, keep removing nodes having degree less than or equals to 1 till there are no such nodes in the network. And when there are no such nodes in the network, we change it to 2 and then we check the nodes having degree less than or equals to 2, we will keep doing it.

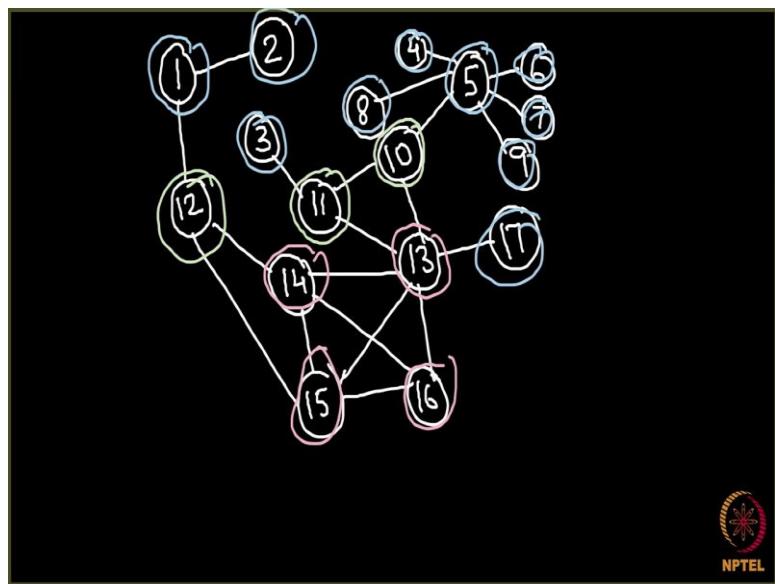
Now do you see this code will keep running forever, there is some problem with this code. At one time all the nodes from this network will be removed and that is the time where we should stop. So, what we are actually going to do is before coming to this step, before coming to this step and before checking whether there are such nodes in the network we can actually put a small condition here.

So, before doing this we can actually put a small condition there and this condition will check whether my network is empty, whether my network has become empty. Whether I have removed all the nodes from this network and if I have removed all the nodes from this network what I will do? I will simply append my temp to buckets.

So, see what is going to happen is, let us say that 5 is the maximum degree in this network and you are removing the nodes having degrees less than or equals to 5. And you remove these networks and at a particular time when you remove a node the network becomes empty, but your tmp will be holding the last nodes present, last nodes which you have removed from this network are still present in tmp, So, that bucket we still have to append.

So, at this time we will append tmp to buckets buckets.append(temp) and you break. Now we will be implementing this and I am sure this will be further clear when you look at the code, you might want to keep this in front of you while writing the code which will make your process easier ok. So, next we are going to write the next we are going to see the code for facial decomposition and please note we need a little bit more complicated network.

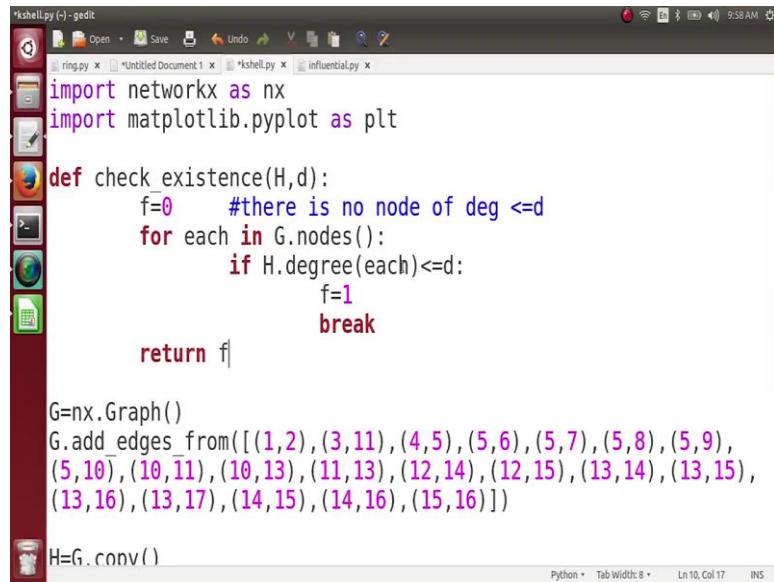
(Refer Slide Time: 10:05)



So, for doing facial decomposition and for all the further codes we will be going to use this particular network which is the slightly a little bit complicated. So, here if you look at the nodes present in nodes which will be removed in the bucket 1, so in bucket 1 these are the nodes which should be removed. So, first you remove this node 2 and 2 and all these nodes 4, 8, 6, 7, 9 and 17 over here and 3 over here and in the next iteration you will remove one and five as well and then your bucket 1 will be over.

And then in your bucket 2 will be going to remove these nodes 11 over here and 12 over here. And then once you remove 11, you will have to remove this node 10 over here also. So, these nodes should be in your bucket 2 and finally, your bucket 3 should comprise of these 4 nodes over here. So, we will be using this network for our programming screen casts. So, now, we are going to write the code for facial decomposition.

(Refer Slide Time: 11:13)



The screenshot shows a terminal window titled 'kshell.py (-) - gedit'. It contains the following Python code:

```
import networkx as nx
import matplotlib.pyplot as plt

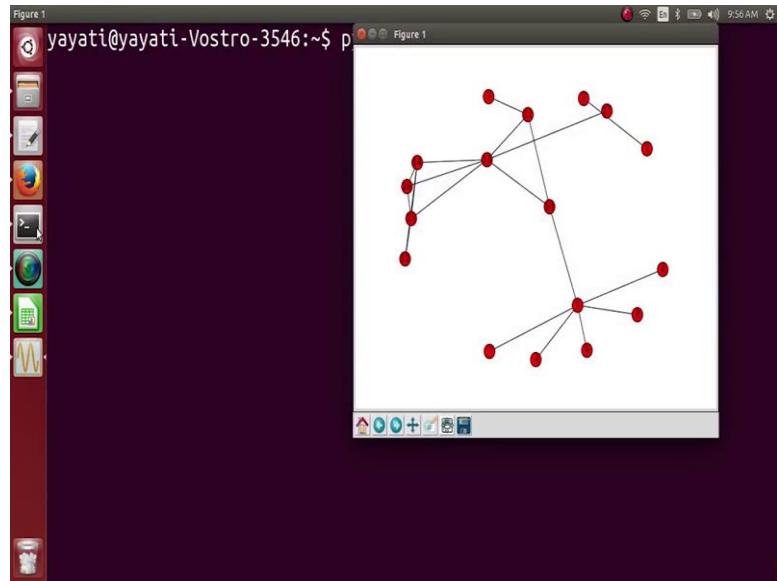
def check_existence(H,d):
    f=0      #there is no node of deg <=d
    for each in G.nodes():
        if H.degree(each)<=d:
            f=1
            break
    return f

G=nx.Graph()
G.add_edges_from([(1,2),(3,11),(4,5),(5,6),(5,7),(5,8),(5,9),
(5,10),(10,11),(10,13),(11,13),(12,14),(12,15),(13,14),(13,15),
(13,16),(13,17),(14,15),(14,16),(15,16)])
```

At the bottom of the code, there is a partially typed command: 'H=G.conv()'.

So, as usual we import network x as nx and we import matplotlib pyplot as plt. What do we do next is we create a graph  $G = \text{nx graph}$  and we add some edges to this graph? So, this graph which I am going to take is the same graph which we have seen previously in this video. So, the graph which I am going to take is this. You can actually verify and see that it is the same graph. So, we can also draw it and see `nx.draw(G)` and then `plt.show` and then we can execute and see it.

(Refer Slide Time: 12:02)



So, this is the network over here and we have to look at various shells in this network ok. What we do? Next is we have already discussed what we are going to do is first of all we are going to create a copy of this network  $H = G.\text{copy}$  and whatever we are going to do? We are going to do it with  $H$  and then we take our variable iteration corresponding to the degree which is initially 1. And then we have a list  $\text{tmp}$  corresponding to the bucket which we are filling currently.

So, it is corresponding to the let me write it down here. It is for the bucket being filled currently ok. And then we have our overall added bucket which is going to be a list of lists and each list here is a bucket ok. Next what we are going to do is we are going to run a while loop while 1.

So, this loop will keep running till a terminating condition comes and Now, we put a check condition whether there are nodes of degree less than or equals to it in this network. So, for that I use a function and let us call this function as `check existence`. So, flag equals to `check existence` and what do I pass here a graph  $h$  and the degree. And let us define this function first here. Define `check existence`, where I have passed the network  $H$  and the particular degree and how do I define it? It is very simple.

So, what do I do? Initially I said  $f$  equals to 0 which means that  $f$  is going to be 0 till the end of this function if in the case there is no node of degree less than or equals to  $d$  in the network right. And then what we are going to do for each in  $G.\text{nodes}$  we check each and

every node in this network and as soon as we find a node in this network whose degree is less than or equals to d, it means that still there is a node having degree less than or equals to d in the network immediately we set f to 1 and break this loop right and then we return f.

So, this f is going to be 0 till the end of this code if there is no node of degree less than or equals to d and it is 1 if there is even 1 node of degree less than or equals to d ok.

(Refer Slide Time: 14:53)

```

nxshell.py (-) - gedit
Q Open Save Undo V
ring.py *Untitled Document1 * nxshell.py * influential.py
((13,16),(13,17),(14,15),(14,16),(15,16))

H=G.copy()
it=1
tmp=[] #for the bucket being filled currently
buckets=[]#list of lists(buckets)
while(1):
    flag=check_existence(H,it)
    if flag==0:
        it = it +1
        buckets.append(tmp)
        tmp=[] #start with a fresh bucket
    if flag==1:
        node_set=find(H,it)
    |
nx.draw(G)
plt.show()

```

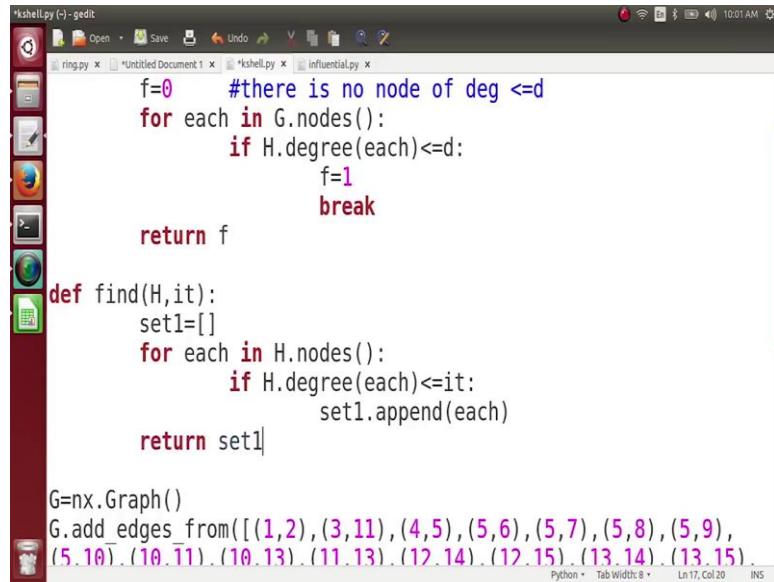
So, flag equals to check existence h comma it. And next what we have seen, flag can be 0 or 1. If flag is 0, it means that we have looked at all the nodes having degree less than or equals to it, even recursively we have seen we have pruned again and again and there is no node of degree less than or equals to it. In that case what we have to do? We have to mark our bucket as complete.

So, if flag is equals to 0, what do we do? We move on to the next degree, we set it equals to it plus 1 and we say that the current bucket is complete. So, we append our temp to buckets, mark it as complete and now we start with a fresh bucket.

So, this is nothing, but start with a fresh bucket right ok. And in another case if flag was 1, what we are going to do? If flag is one it means that we have to keep pruning the nodes having degree less than or equals to it. In that case first of all we should find out such nodes in the current network we have to find the nodes which are having degree

less than or equal to it. Let us call a function for that and let us name this function as find and so in the graph H finally, nodes having degree less than or equals to it and we can quickly write code for this function define find h comma it and how do I define it.

(Refer Slide Time: 16:26)



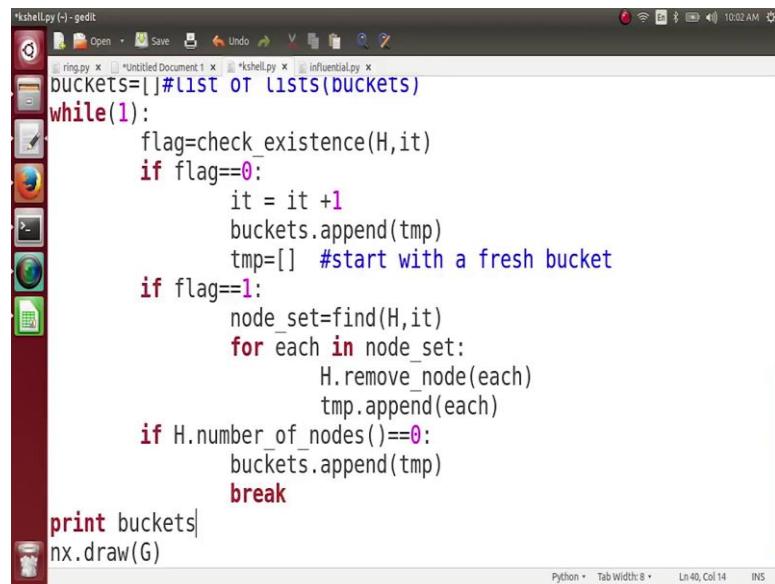
```
f=0      #there is no node of deg <=d
for each in G.nodes():
    if H.degree(each)<=d:
        f=1
        break
return f

def find(H,it):
    set1=[]
    for each in H.nodes():
        if H.degree(each)<=it:
            set1.append(each)
    return set1

G=nx.Graph()
G.add_edges_from([(1,2),(3,11),(4,5),(5,6),(5,7),(5,8),(5,9),
(5,10),(10,11),(10,13),(11,13),(12,14),(12,15),(13,14),(13,15)]).
```

So, initially this set is empty, and it is simple. For each in H both nodes if H.degree(each), if H.degree of H is less than or equals to it what we are going to do is set 1.append each and then return set 1 right. So, we have find out we have found out all the nodes having degree less than or equals to it. Next, we have to prune these nodes on the network and add them to our bucket. So, what do we do? Prune these nodes from our network and add them to the bucket.

(Refer Slide Time: 17:23)

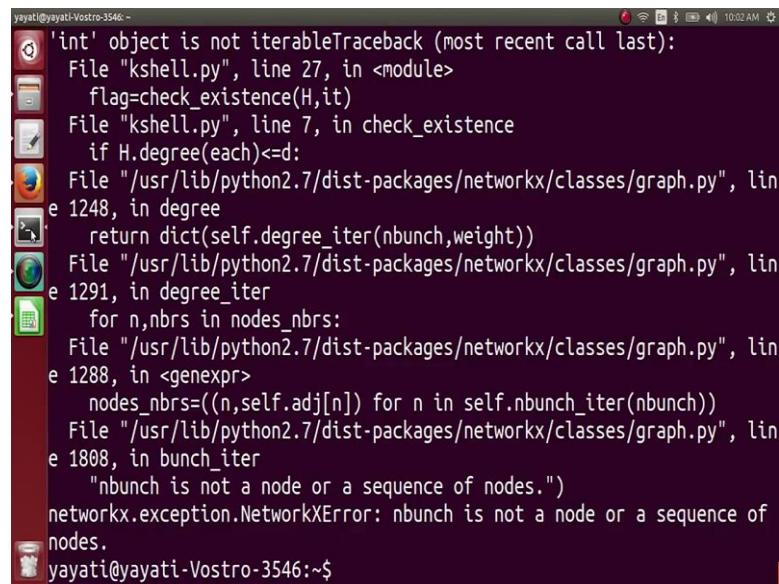


```
nkshell.py (-) - gedit
nipy.py * "Untitled Document 1" * nkshell.py * influential.py *
buckets=[] #LIST OF LISTS(BUCKETS)
while(1):
    flag=check_existence(H,it)
    if flag==0:
        it = it +1
        buckets.append(tmp)
        tmp=[] #start with a fresh bucket
    if flag==1:
        node_set=find(H,it)
        for each in node_set:
            H.remove_node(each)
            tmp.append(each)
        if H.number_of_nodes()==0:
            buckets.append(tmp)
            break
print buckets
nx.draw(G)
Python ▾ Tab Width: 8 ▾ Ln 40, Col 14 INS
```

So, for each in node underscore set what we are going to do is `H.remove_node(each)`, we have removed it from `H` and we have to put it in our current bucket which is being filled which is nothing, but `tmp.append(each)` right. In this case we will keep pruning the nodes and the last part of the code is a terminating condition. So, if by pruning these nodes our network becomes empty what we have to do?

So, if network becomes empty is if `H.number_of_nodes` if it becomes equals to 0 what we have to do? What we have to mark the current bucket as filled. So, `buckets.append(tmp)` and then `break` ok. And now we want to see whether our buckets are correct or not. So, for that what we can do is we can simply print `buckets`. It tells us the different buckets in our network.

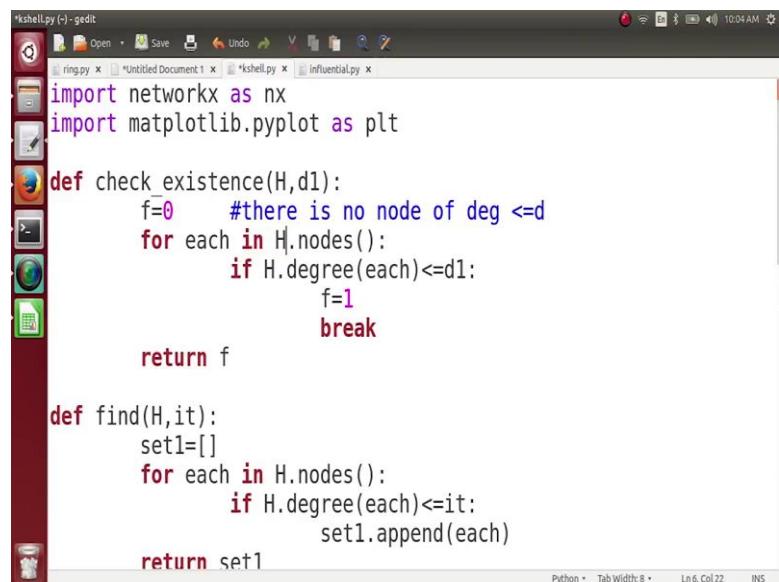
(Refer Slide Time: 18:44)



```
yayati@yayati-Vostro-3546:~$ kshell.py
'int' object is not iterable
Traceback (most recent call last):
  File "kshell.py", line 27, in <module>
    flag=check_existence(H,it)
  File "kshell.py", line 7, in check_existence
    if H.degree(each)<=d:
  File "/usr/lib/python2.7/dist-packages/networkx/classes/graph.py", line 1248, in degree
    return dict(self.degree_iter(nbunch,weight))
  File "/usr/lib/python2.7/dist-packages/networkx/classes/graph.py", line 1291, in degree_iter
    for n,nbrs in nodes_nbrs:
  File "/usr/lib/python2.7/dist-packages/networkx/classes/graph.py", line 1288, in <genexpr>
    nodes_nbrs=((n,self.adj[n]) for n in self.nbunch_iter(nbunch))
  File "/usr/lib/python2.7/dist-packages/networkx/classes/graph.py", line 1808, in bunch_iter
    "nbunch is not a node or a sequence of nodes."
networkx.exception.NetworkXError: nbunch is not a node or a sequence of
nodes.
yayati@yayati-Vostro-3546:~$
```

So, let us execute this code and see. An adder at line number 27. Check underscore existence H comma it ok. If it was 0 for each in G.nodes, so if I do not think this (Refer Time: 19:20) ok. Some error let us try to figure it out, some very small error. So, it what is it here? It is a number right a particular degree d for each in let us use some other variable here d1 ok.

(Refer Slide Time: 20:11)



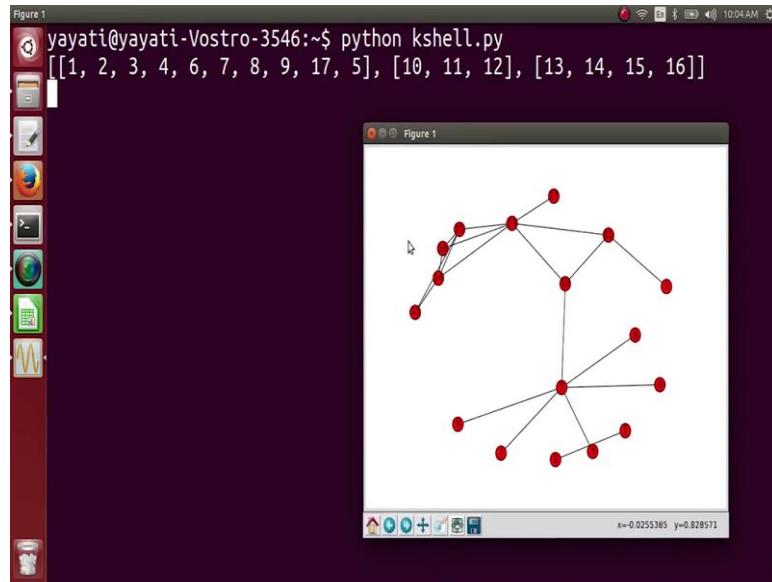
```
rkshell.py (~)-gedit
import networkx as nx
import matplotlib.pyplot as plt

def check_existence(H,d1):
    f=0      #there is no node of deg <=d
    for each in H.nodes():
        if H.degree(each)<=d1:
            f=1
            break
    return f

def find(H,it):
    set1=[]
    for each in H.nodes():
        if H.degree(each)<=it:
            set1.append(each)
    return set1
```

See a mistake here; here we have used G.nodes, so here we have to use H.nodes because, if we are using G.nodes here we might here be talking about a node which is not there in H, H has less nodes than G So, it should be H.nodes here and let us execute it ok.

(Refer Slide Time: 20:32)



So, you see it here, your 1 core is the first bucket is from 1, 2, 3, 4, 6, 7, 8, 9, 17, second is this and third is this and that is the result.

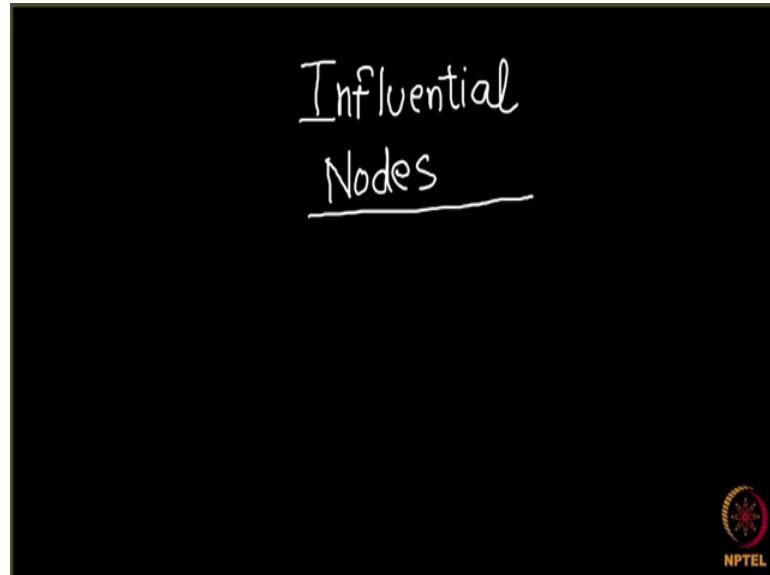
So, we have implemented facial decomposition, if you want you can also color these nodes with different colors to see the differentials and so on.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**How to go Viral on Web**  
**Lecture - 164**  
**Coding cascading Model**

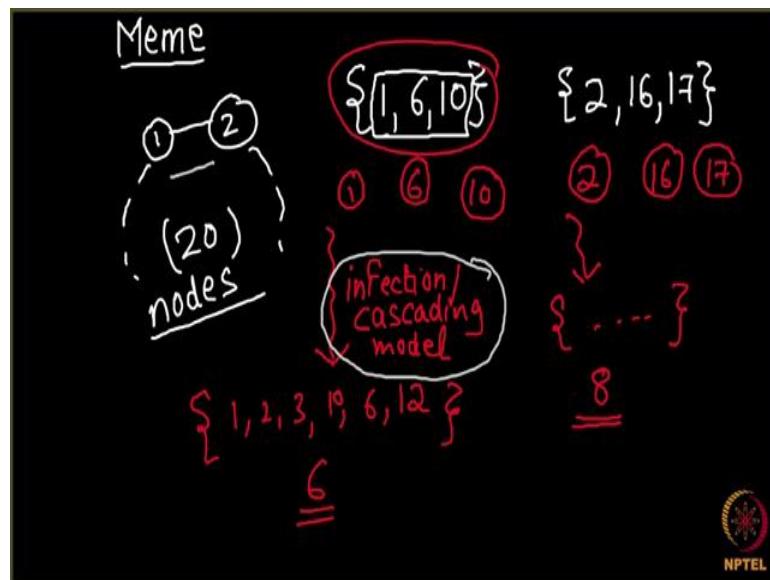
So, next what is our aim? Our aim is to determine the influential pair of nodes. For actual question is we want to see that which nodes in our network are the most influential.

(Refer Slide Time: 00:18)



So, we want to determine the influential nodes in a network, but how do we know which nodes are influential. And what does it mean when I say that these bunch of nodes are influential. What we are going to do that is; experimentally speaking, programmatically speaking, what we are going to do is we can have certain set of (Refer Time: 00:45) nodes from where our infection will start.

(Refer Slide Time: 00:47)



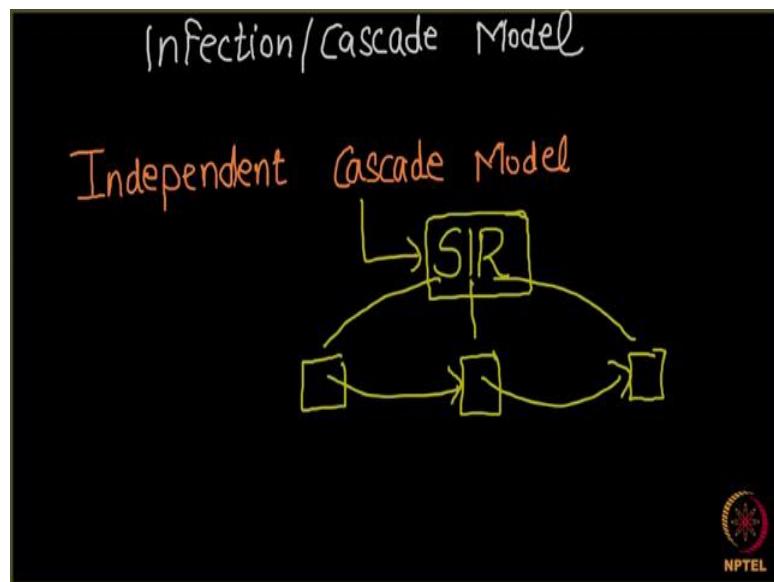
So, let us say there is a meme which wants to traverse on a network which is traversing on a network, let us say the network is something like this consisting of let us say some 20 nodes, 20 nodes some network. And now, we want to see out of these 20 nodes which nodes are the most influential? What we can do for that is if I want to compare let us say 2 sets of nodes, I have a set 1, 6 and 10, so these 3 nodes and I have another set let us say 2, 16 and 17. And I want to see that which out of these 2 sets is most influential.

What I am going to do is initially in this network I infect these 3 nodes with my main, so I will say that node 1 is infected with the main, node six is infected and node ten is infected. And then I will run an infection model, I run cascading model infection or a cascading model. And I will see when this model stops, when this process stops how many nodes in whole are infected. So, let us say that this is the set of nodes which are finally infected, let us say these are the set this is a set of nodes which is finally infected.

So, there are 6 nodes in this set. So, I can say that the influential power of this set is 6. And then to determine the influential power of this another set I can again do the same process. I will initially, in fact these 3 nodes to 16 and 17 in the network, I run my cascading model. And will see at the end how many nodes are infected and let us say 8 nodes are infected; it means that the influential power of this bunch of nodes this set of nodes is 8 ok.

So, this is the way we are going to find which set of nodes is most influential. So, to find which is most influential, we probably will take all possible set of nodes on these 20 nodes network and then determine their influential power and then see that which one creates the maximum cascade. And that is actually very time consuming process ok, but here we talked about the infection or cascading model. So, what is this infection or cascading model, let us talk in a little bit of detail.

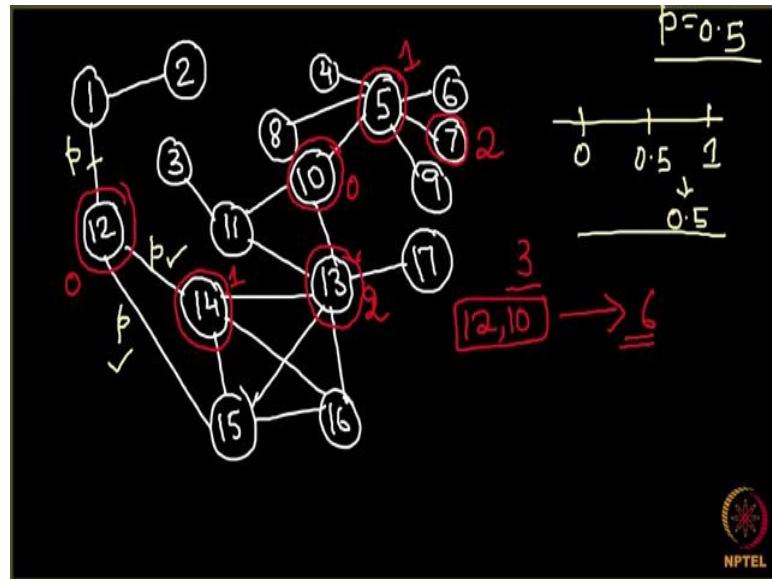
(Refer Slide Time: 03:23)



So, I need an infection or cascade model. And what is this infection or cascade model how do I code it and how does it work. So, that we are going to implement in our next programming screencast. And we are going to look at a cascading model which is known as the independent cascade model, independent cascade model. And this independent cascade model is actually very similar to the SIR model we discussed in the chapter epidemics, very similar to the SIR model.

Do you remember what was SIR model? There were 3 kind of nodes in the network susceptible, infected and recovered and nodes used to pass from this chamber to this chamber infected chamber and then to the recovered chamber. So, our model is very similar to the sir model, the node exactly similar. Let us look at what is independent cascade model now.

(Refer Slide Time: 04:25)



Assume that this is a network given to you here. So, I give you a network here and I want to say that let say the seed nodes in this network are 12 and 10. So, the infection in this network starts from these 2 nodes 12 and 10. And now I want you to implement independent cascade model on this network, how will you do that? So, at time equals to 0, these 2 nodes are infected, nodes 12 and 10 are infected. How the modal proceeds is very simple. What is going to happen at the next time stamp is this node 12 is going to look at all of its neighbors.

So, node 1, node 14 and node 15 and it will infect each of its neighbors independently with a probability  $p$ . And let us say  $p$  is 0.5, we can set the value of  $2p$  to be anything. So, it infects each of its nodes, each of its neighbors with the probability of  $p$  and we have done it previously right. How can we simulate this? We can actually toss a coin for this edge if we get an head then 12 infects 1 else node toss a coin for this edges well and toss a coin for this edges well. That is that was how we can implement these and in terms of programming also it is very simple.

So, what we can do is we can generate a random number from 0 to 1 and then we look whether this random number is greater than 0.5 or not. So, what is the probability that this random number is greater than 0.5, it is nothing, but 0.5. So, this is how we can do it in our coding. So, we are we generate a random number from 0 to 1 and if the number is greater than 0.5, we will in fact, the corresponding node. Ok that apart.

So, at time iteration 1, this node 12 over here we try to infect each of its neighbors with the probability  $p$  and let us say it succeeds in infecting this node 14 over here. And then node 10 is also doing the same, it will look at all of its neighbors and infect them with the probability  $p$ .

So, it again has 3 neighbors and let us say it succeeds in infecting the node 5 over here. So, at time iteration 1 this node 14 over here and this node 5 over here gets infected ok. What's now going to happen in next iteration is node 12 and node 10 have done their job. So, they were given 1 chance to infect their neighbors and their chance is now over. So, these 2 nodes it remains infected they are infected, but they do not get any further chance of infection.

So, it is very similar to a case where in a school some students are getting infected. So, initially at the first day the students who are infected, they infect the people around, but during the second day at second day at second day they do not get any chance to infect anybody. So, nodes 12 and 10; now we will not infect anybody and the just infected nodes, which are the just infected nodes, so it is time iteration 1. Just infected nodes are 14 and 5.

So, 14 and 5 now gets a chance get a chance to infect their neighbors. So, this node 14 over here looks at all of its neighbors and tries to infect each of them with a probability  $p$  except for the node 12 because, node 12 is already infected. So, we cannot infect it further. So, for remaining neighbors which are 3 in number, so for each of these neighbors it sees whether it can infect them or not and let us say node 14 succeeds in infecting this node 13 over here.

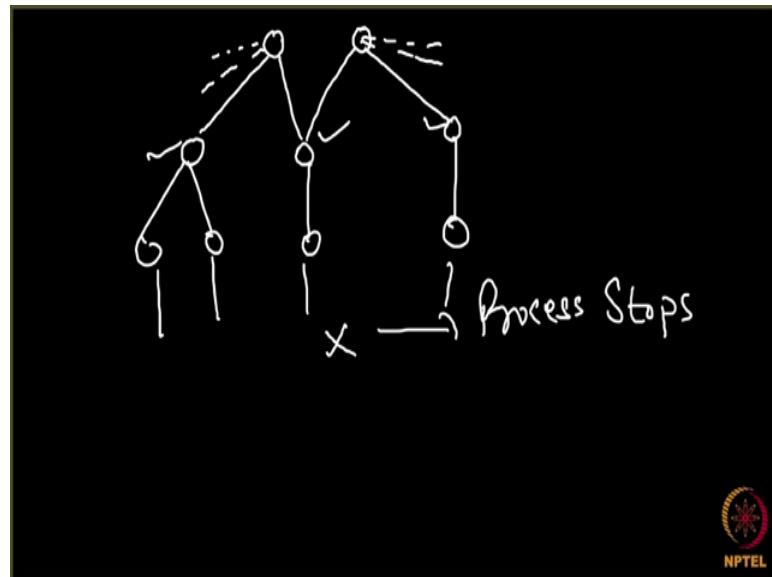
So, time iteration 1 node 13 gets infected. And in the same iteration this node 5 over here also tries to infect its remaining neighbors. So, there are 5 neighbors and let us say it succeeds in infecting this neighbor 7. So, at time iteration 2, so a time iteration 2 this node 13 over here and this node 7 over here gets infected and again in the next iteration these node 7 and 13 look at their neighbors and try infecting them and what will happen then, let us say that this node 13 is unable to infect anybody and this node 7 over here is also unable to infect anybody right.

So, nobody gets infected in iteration 3. What will happen next? So, these will be just now infected nodes which were infecting their neighbors. 13 and 7 have also got their due

chance of infecting their neighbors, but they could not infect anybody. So, now, there is nobody in the network which could look at their neighbors and infect them and hence the process stops.

So, do you see where did the process stop? The process stopped when we reach a iteration, we reach an iteration where nobody is infected. And in this particular case you saw that we started with these nodes 12 and 10 over here which were initially infected and the size of the cascade is 6. So, the influential power of node is set 12 and 10, we can say 6. So, this is how an independent cascade model is going to work. Let me revise it quickly.

(Refer Slide Time: 09:44)



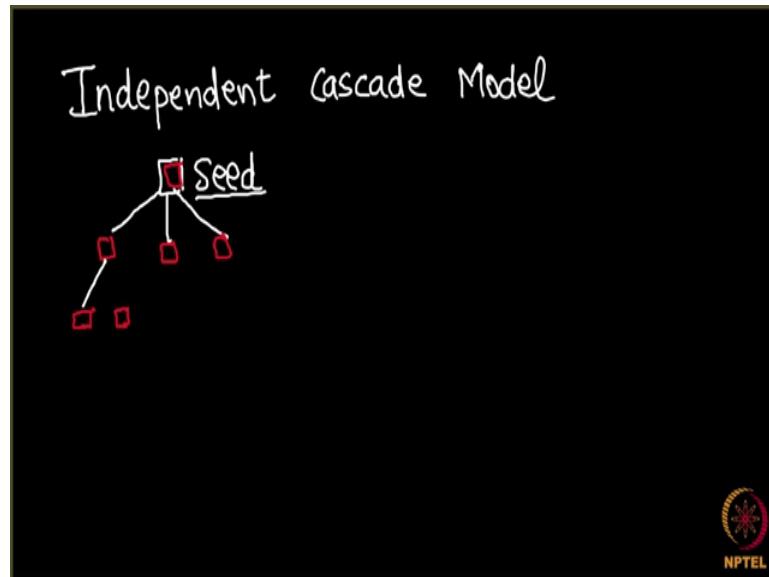
So, what is happening is initially there are certain nodes which are infected let us say these 2 nodes are infected. In the next iteration these nodes will look at their neighbors. So, this can be a possible case here right. So, these nodes will look at their neighbors and infect each of them with the probability  $p$  and let us say that these 3 neighbors got infected.

So, they might have other neighbors as well and let us see out of all these neighbors only these 3 are the ones which got infected. Then these three nodes over here will repeat the process and in fact, all of their neighbors with probability  $p$  and they might succeed in infecting some people and let us say here for more people go to infected. And let us see in the next iteration nobody goes to infected. So, in the next iteration nobody goes to

infected. So, what will happen is your process will stop here. So, this is what is known as an independent cascade model.

And why is it called an independent cascade model is actually very clear whether this edge over here get passes the infection or node is independent of whether this says over here passes the infection or not. So, hence it is known as the independent cascade model. And next we look at how can we code this independent cascade model. So, let us now see how we can implement this independent cascade model. So, before writing the code for independent cascade model let us look at the structure of the code.

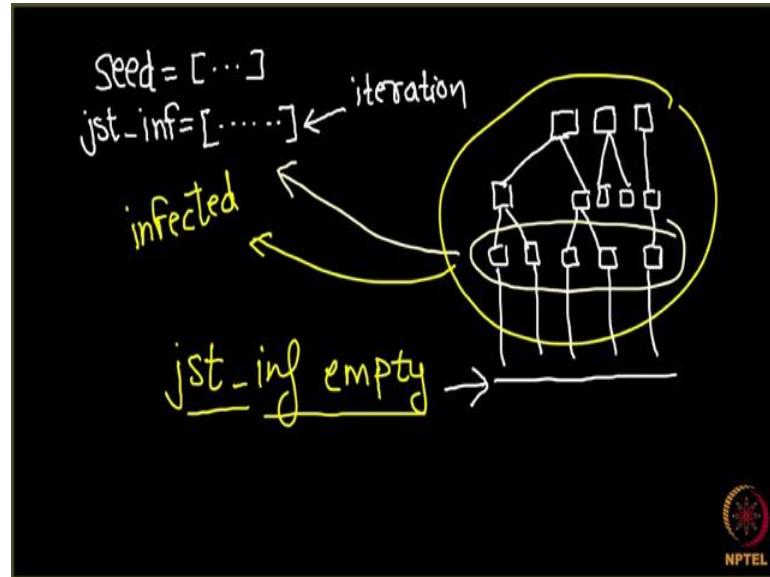
(Refer Slide Time: 11:07)



So, we are going to see how can we implement independent cascade model, independent cascade model. And what we are going to do is we know that we know the entire procedure right, we know that there is a set of nodes which are our seed nodes, which are our initially infected nodes. And the seed nodes are then going to look around all of their neighbors and ending up infecting few of its neighbors and ok.

So, this is a C and they end up infecting few of their neighbors and then all of these neighbors are going to look at their neighbors and then they are going to infect few of their neighbors. So, these are the neighbors which are not already infected. So, some new nodes over here are infected and this process will stop as soon as we reach an iteration where no new node is infected. So, how can we write a code for this? While writing a code for this we are going to use 3 lists.

(Refer Slide Time: 12:06)



So, let me tell you one list is obviously the seed nodes, which holds which of the nodes are initially infected, so this list is seen. Another list which we will be considering is let recall it just infected. So, another list is just infected. What does this list hold is, this list holds the moves which are most recently infected in any iteration?

So, so this particular list it iteration dependent. If you look at this list seed, it is always going to be the same. So, these are the nodes which were initially infected. It is just infected list it is going to change with time. So, in iteration 1, when C is going to infect certain nodes then this list will change, and whichever nodes will be infected at the first iteration will come in just infected. So, let us say again I will make the same diagram the seed is here and then this let us say there are 3 nodes in seed set and these 3 nodes they end up infecting some more loads. So, these are some more nodes which got infected life. So, at this particular iteration this will be my set of just infected nodes.

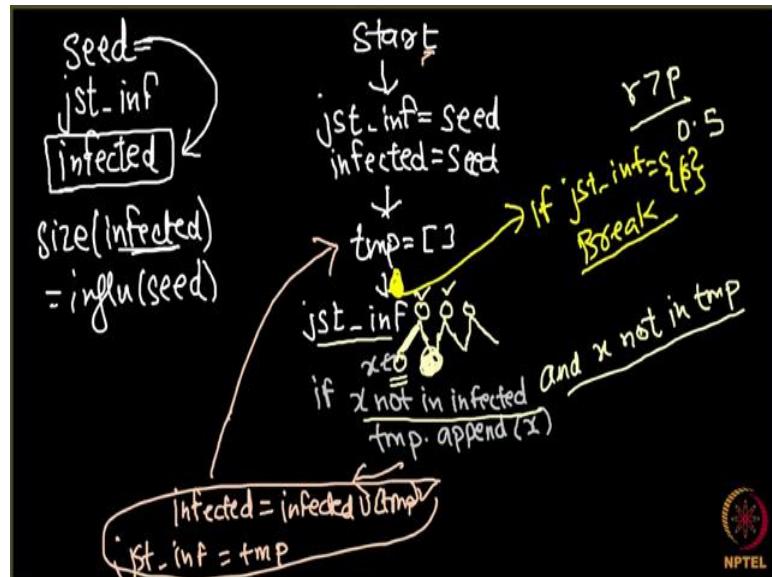
So, at this iteration this becomes the least just infected right and then we will also be using another list which is infected. And this list holds all the nodes over overall the process throughout the process till now whichever nodes are infected comes in this infected list. So, you see what is what is infected currently. So, this entire these all the nodes are infected right.

So, this is my list infected, this is my list infected. What will happen in the next iteration? What will happen over next iteration is let us say so, these whole nodes over

here they might end up infecting some more nodes. So, let us say in iteration 2 these are the nodes which are infected. Now what is just infected? So, these nodes over here these nodes are my just infected and then this entire set of nodes is my infected right.

So, we will be using these 3 lists, and now can you tell me when will this process end? This process will end at an iteration when this list just infected becomes empty. So, as soon as this just infected becomes empty will break the loop and come out. So, after this particular iteration over here, let us say that in the next iteration none of the nodes gets infected. So, just infected fact, it is empty here and the process will come to an end ok.

(Refer Slide Time: 15:01)



So, how we are going to implement it let us see. To implement it as we saw before we have 3 lists. So, 1 is corresponding to seed, one is the nodes which are just infected and at the end we have all the nodes which are infected. So, this particular thing we are going to return at the end. These are the overall nodes which are infected throughout the process. And the size of this list infected is what we called the influential power of this set seed.

We will be starting our infection with seed and at the end certain nodes will be infected. And the size of these infected nodes is nothing but the influential power over seed ok. And this is what we are going to return. Now what we will be doing is to implement our independent cascade model in the very beginning, so let me put a start over here. So, here we start, after starting I sit the value of just infected to be seed right, when nothing

has happened, so these are the seed nodes which are just infected and which have to create my further infections and the value of infected is also equal to seed because, when nothing has happened these are only the seed nodes which are infected.

What will happen next is we will enter a loop and what will happen in this loop is these all nodes which are in just infected. So, we will look at this just infected set and let us say ok. So, before doing this a small step and it take a temporary list. And let me call it tmp and you will understand why I am taking this empty list over here. So, this is my tmp. What will happen is I will take all the nodes which are in just infected. So, let us say that these are all the nodes which are in just infected and then all these nodes will look at their neighbors right, all the nodes will look at their neighbors. And then they will infect each of their neighbors with a probability of p.

So, you can do this by how do you infect them with a probability p, we have discussed it earlier. We are going to generate a random number from 0 to 1, a real number and if the number turns out to be greater than p, if  $r > p$  we are going to insert the node over here else node right. So, when p (Refer Time: 17:32) 0.5 will generate a number from 0 to 1 and if this number is greater than 0.5 it means that this node over here should be infected ok.

While infecting it, but few things we have to take care is, first of all this node should not already be infected right. So, we put the desired loops here. Overall what we have to do at this step is we have to look at this list just infected and then for each of its neighbors over here for each of the neighbors, so first we take this node over here and then for each of its neighbor we generate this random numbers and then we see whether this node over here should be infected or not.

And if this node is not already infected that is this node should not belong to a set infected, then we are going to append it to tmp. So, let us say that let me write it down, if a particular neighbor, so let me call this node over here as x, if x node in infected right. So what I am going to do is tmp.append(x) right and I am going to do this for each of the nodes over here. So, first I will do it for this x then I will do it for this x, then i will do it for this x, this x and so on.

And one thing you might note here now it seems it is a graph, it is not a tree like structure what can happen is let us say this node over here it is the neighbor of both the

node, it is the neighbor of this node as well as this node. And let us say when we came here this node over here got infected, so we have appended it to tmp right.

So, when we were looking at the neighbors of this node, we saw whether it should be infected and then we see that yes this node should be infected it is not already infected. So, we append it to a list tmp, but then when this node is looking at its neighbors and it might also see that it should be infected right, it is not till now it is not infected because, it is note in this list infected it is in list (Refer Time: 19:41). So, we might again appended to temp. So, to avoid this, so that sort of duplication what we can do it if x noting infected and x noting tmp. This is not a very important step, but just to avoid duplication we can write it here.

So, first of all this node over here, it should not be previously infected, second it shouldn't be in tmp. In that case we append x to tmp. So, we get this list tmp at the end. What is this list tmp telling us? tmp is telling us that in the current iteration these are the nodes which got infected. So, what we do next is we go to our set of nodes in tmp, which should be infected next.

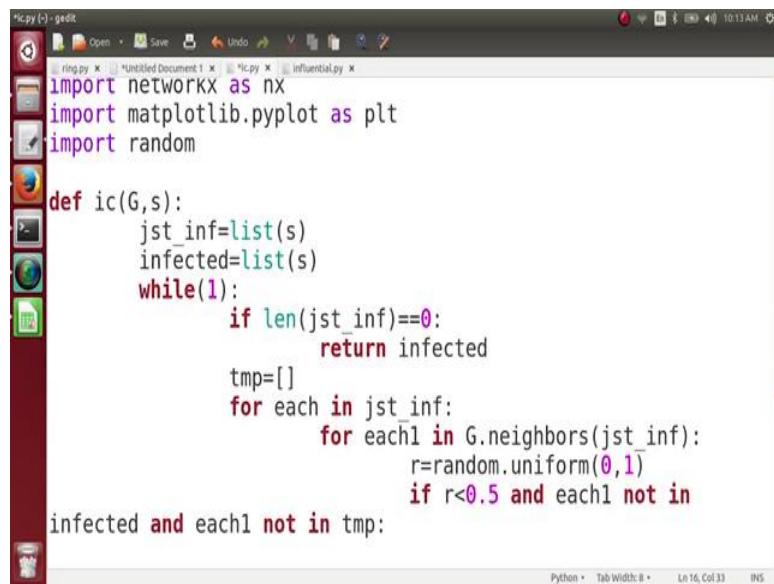
So, we will simply for all the nodes in tmp we set our list infected right. So, these all node should be infected. Infected becomes infected union tmp right. So, these all nodes become infected first of all then. Secondly, you notice that now my set of just infected should change right. So, previously my just infected were the node which were infected in that iteration, next what should become just infected now? Just infected should become equal to tmp right and then we jump back.

So, after doing both of these steps we jumped back here ok. So, you see what is happening, we start from here, we start from here and then we set just infected to be seen infected to be seen, we take a temporary list. For initially just infected is seed and then we seeded which node should get infected in the current iteration. We append them to the list infected and then we said just infected equals to tmp, then we come here. Then for all the nodes which were infected in the previous iteration we keep repeating this process ok. Now it has become an infinite loop right, it will keep running keep running keep running.

So, one should this process come to an end when this list just infected becomes empty. So, we can put as small terminating condition over here. Over here we can put

terminating condition that if your list just infected is 9, it has no element than you break right. So, this is the terminating condition over here. So, this is my entire function for implementing independent cascade model. And at the end we can return the list infected which holds all the nodes which are infected during this entire process ok.

(Refer Slide Time: 22:27)



```
nicpy (-) - gedit
ring.py * Untitled Document 1 * nic.py * influential.py *
import networkX as nx
import matplotlib.pyplot as plt
import random

def ic(G,s):
    jst_inf=list(s)
    infected=list(s)
    while(1):
        if len(jst_inf)==0:
            return infected
        tmp=[]
        for each in jst_inf:
            for each1 in G.neighbors(jst_inf):
                r=random.uniform(0,1)
                if r<0.5 and each1 not in
infected and each1 not in tmp:

```

So, now we are going to write a code for login independent cascade model. So, already made a file here and we are going to use the same graph which we have used previously. So, some part of the code I have written very basic part. And next let us look at how can we implement independent cascade model here. So, you should have some seed in this network.

So, we can randomly define some seed and let us say my seed is node 3 and let us say node 8. And next what I want to do is, I want to call my independent a function independent cascade I see which runs the independent cascade model on this graph G taking as input the seed c and gives me as output a list. And this is the list of the people who are infected finally, at the end.

So, that is list 1 ok. How do we implement it? So, define independent cascade G comma seed ok. What we are going to do is, we have to so, we have 1 list seed, another list we know we had a list called just infected and which is nothing which is initially going to be the same as the seed node, let us call it as here. It is same as the seed node s and we know that the people who are infected currently are also nothing, but the seed s right

initially, both of these nodes have the same value as c. And then we have a loop while 1, while 1 what are we going to do, we know that what is the terminating condition for this loop. So, there should be some persons who are just infected. If nobody is just infected, we should break.

So, if length of just infected is equals to 0, we are going to what we will do, we will return our list infected that is all the infected people right, that is sufficient we will return ok. That is the end of this function otherwise, what we have to do? We have to find out the new people who will get infected. So, we have a temporary array tmp for that and how do we find these people? We look at everybody for each in this list just infected; we are going to look at their neighbors.

So, for each 1 in G dot neighbors of just infected people what are we going to do? Now, so here is an edge from 1 person here to 1 person here and we have to do. We have to see whether the person will get infected or not that depends upon the probability of infection right. And we have seen before how do we model the probability of infection was with the help of random variable.

So, we import random here and what do we do is we generate a random number r, random real number r from 0 to 1 and what do we do next is if, so when is the infection going to occur when r is less than 0.5 let us say the probability of infection is 0.5 and what is the other condition, this node each 1 should not be previously infected. Each 1 not in infected and as we discussed previously to avoid any duplication in tmp, 1 not getting infected by the 2 nodes which are in just infected we do for each 1 node in tmp also, in tmp we do not need any duplication. So, what we do here, if this happens is the tmp dot append each one.

(Refer Slide Time: 26:59)

```
ic.py (-) - gedit
infected=list(s)
while(1):
    print jst_inf, infected
    if len(jst_inf)==0:
        return infected
    tmp=[]
    for each in jst_inf:
        for each1 in G.neighbors(each):
            r=random.uniform(0,1)
            if r<0.5 and each1 not in
                infected and each1 not in tmp:
                    tmp.append(each1)
    for each in tmp:
        infected.append(each)
    jst_inf=list(tmp)

G=nx.Graph()
Saving file '/home/yayati/ic.py'...
```

So, this each 1 is the node which is going to be infected in this iteration ok. So, we have found out all the nodes which get infected in the current iteration then, what we have to do for each in tmp. These nodes which got infected just now, what are we going to do infected dot append. So, these nodes are now infected. So, we append them to this list infected, infected dot append each and our list just infected now get tends to tmp and that is done. So, this is a very simple code and right. So, we want to see how does this code run.

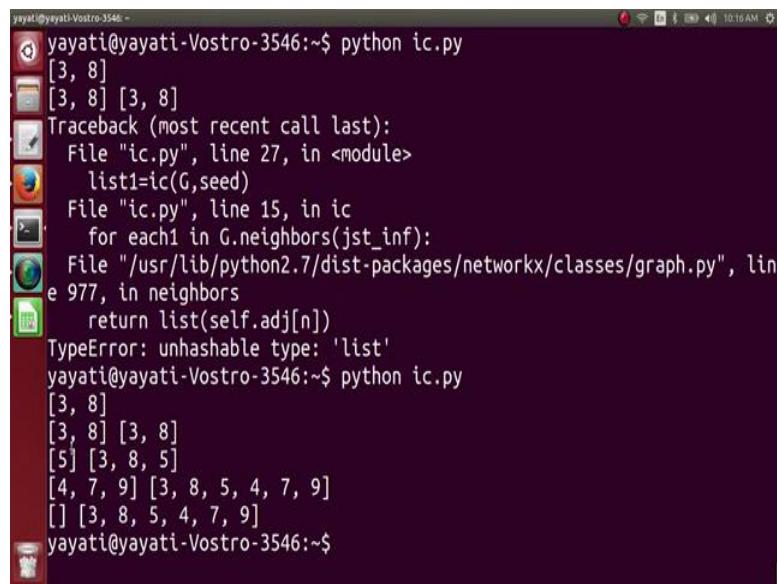
(Refer Slide Time: 27:54)

```
ic.py (-) - gedit
if len(jst_inf)==0:
    return infected
tmp=[]
for each in jst_inf:
    for each1 in G.neighbors(jst_inf):
        r=random.uniform(0,1)
        if r<0.5 and each1 not in
            infected and each1 not in tmp:
                tmp.append(each1)
    for each in tmp:
        infected.append(each)
    jst_inf=list(tmp)

G=nx.Graph()
G.add.edges_from([(1,2),(3,11),(4,5),(5,6),(5,7),(5,8),(5,9),
(5,10),(10,11),(10,13),(11,13),(12,14),(12,15),(13,14),(13,15),
(13,16),(13,17),(14,15),(14,16),(15,16)])
```

So, for that what I am going to do is let us print the value of just infected at every iteration and see how this is going. So, we print both the nodes, let us say we print just infected first and then let us print infected the total notes infected and at the beginning let us print s right ok, s just infected, infected and here before returning infected, here also let us print infected. Actually, there is no need for these leave it as it is ok. Let us run this code and see ok.

(Refer Slide Time: 28:42)



```
yayati@yayati-Vostro-3546:~$ python ic.py
[3, 8]
[3, 8] [3, 8]
Traceback (most recent call last):
  File "ic.py", line 27, in <module>
    list1=ic(G,seed)
  File "ic.py", line 15, in ic
    for each1 in G.neighbors(jst_inf):
  File "/usr/lib/python2.7/dist-packages/networkx/classes/graph.py", line 977, in neighbors
    return list(self.adj[n])
TypeError: unhashable type: 'list'
yayati@yayati-Vostro-3546:~$ python ic.py
[3, 8]
[3, 8] [3, 8]
[5] [3, 8, 5]
[4, 7, 9] [3, 8, 5, 4, 7, 9]
[] [3, 8, 5, 4, 7, 9]
yayati@yayati-Vostro-3546:~$
```

We see a problem here just infected is a list here right. So, we have done here, for each in G dot neighbors it should be each here right, for each in just infected for each 1 in G dot neighbors each right. (Refer Time: 29:27).

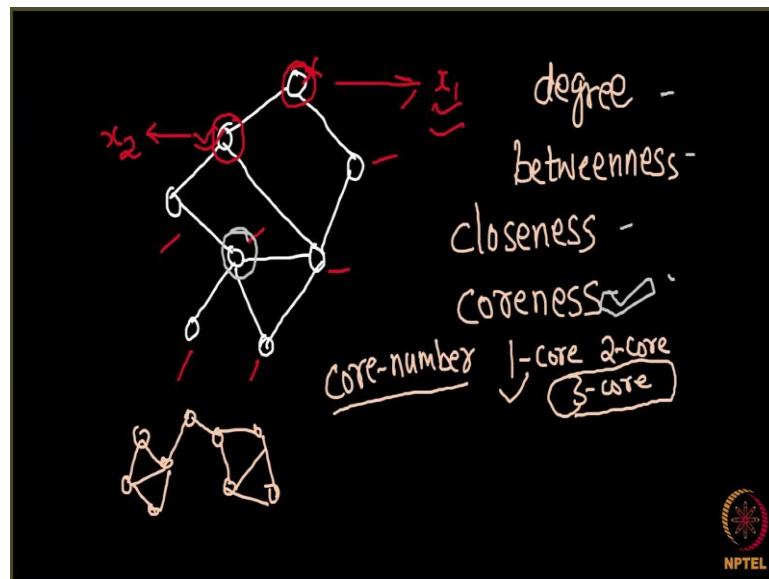
And then you can see here initially 3 and 8 were seed nodes 3 and 8 were the seed nodes, so just infected are also 3 and 8 and infected and 3 and 8 then one of these infected 5 and 5 also became infected. Then 5 infected 3 more nodes 4, 7 and 9 which got I did and 4, 7, 9 could not infect anybody. And hence, these were the finally infected node and the process finally stopped.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**How to go Viral on Web**  
**Lecture - 165**  
**Coding the importance of core nodes in cascading**

So, next we are going to do an interesting experiment. What we are going to do is we are interested in looking at which are the most influential nodes in a network. And we have our answer also right.

(Refer Slide Time: 00:19)



So, given a network over here we saw that to find that let us say in this network over here this network over here I want to see which nodes over here are the most influential. So, what can I do is we know that how do we define influence for a seed set; given a seed we will start our independent cascade model from there and we will see that at the end how many nodes got infected.

So, to say that which nodes over here are most influential what we can do is first of all we say that this is my seed. This is my seed and then I perform the independent cascade model here and I see that at the end how many people got infected and let us say I found a value  $x \cdot 1$ . Then I want to see the influential power of this node. So, I said the seed to be this now this is no longer the seed.

So, this over here is the seed only 1 node and then I will repeat the independent cascade model and I will see towards the end how many people got infected. I will do for this, I will do for this, do for this, do for this and at the end we will see that which of the  $x$  is maximum. And, whichever  $x$  is maximum that particular node is the node, which is the most influential in this network, but doing this kind of an experiment it is a very time consuming process right.

So, taking every node to be seed node and again and again repeating the independent cascade model and then finding out influential node is a time-consuming process. Here we have another very interesting question; can you look at this network and then guess which is the node most influential over here? I have a feeling for example, this node over here it has 1, 2, 3, 4 neighbors and no other node or maybe this node over here.

So, they have 4 neighbors. So, probably might be they are most influential, since they have more number of neighbors they can infect more people right. So, it might be the case that these nodes are most influential. So, that is an easy way of doing this same process right. So, taking every node over here and then running independent cascade model as opposed to just looking at this network and looking at its structure, looking at the properties of certain nodes and then saying that: yes, this node is influential is quite easy.

So, now the question is which nodes are the most influential over here. So, we have various centrality measures ok. What is centrality measure? It is just a complicated term. So, centrality measure is we can measure the values of notes over here based on certain things. For example, we can look at the degree of each node right.

So, is it the degree can I say that the nodes which are having the maximum degree, they are the most influential nodes or we have another measure let us say betweenness, I hope you remember what was betweenness. So, betweenness tells me if I look at a node, if I look at a node how important is it in connecting to different parts of the network?

So, let me use another example. Let us say there is a community structure, this is one community, and this is 1 community right and there is a node over here. So, this node over here is very instrumental in connecting these 2 portions of the graph. So, this node has a very high betweenness over here right. So, are these the nodes which are having a

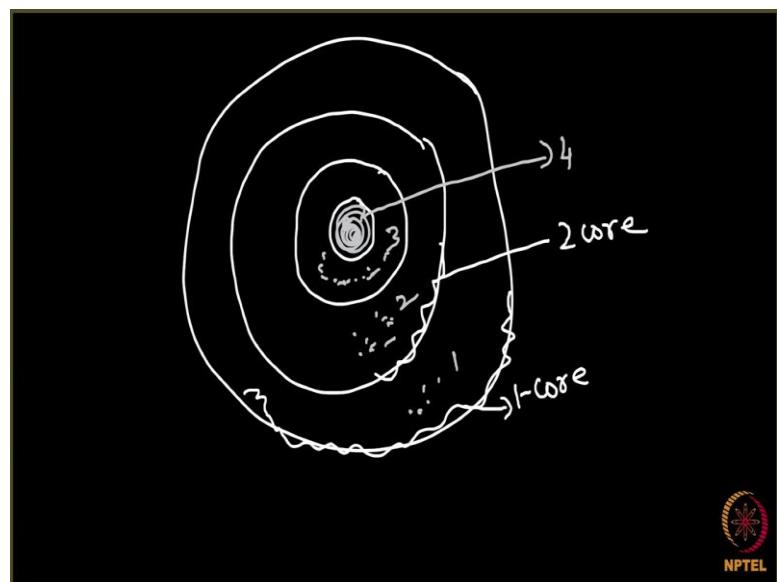
very high betweenness, which are the most influential and we will create the biggest cascade or are these the nodes having high closeness.

So, closeness is something like I take this node and I see that how close is it to other nodes in the network. For example, this node is very close to these 5 nodes and then still closer to this node and then far away from these nodes. So, you can look it up there is a very nice way of calculating this closeness centrality, but the intuition is simple. We see that how closer I am to the entire network that is closeness.

So, are these the nodes which are having a high value of closeness, which are the most influential oh we have a last measure which we discussed coreness. So, are these the nodes which are having maximum coreness, which are most influential? What do I mean by maximum coreness? Maximum coreness is we know that all the nodes in this network are in 1 core right. We define a value which is called core number, what is core number? It is nothing but the.

So, see a node can be part of 1 core, a node can be a part of 2 core, 3 core right. Whatever, is in 3 core, whatever node is in 3 core that is automatically in 1 core So, let us say there is a node which belong to three core I say that its core number is 3, right call number is 3 So, the inner you are in the network, the innermost core you are towards oh see let me put it like this ok.

(Refer Slide Time: 05:37)

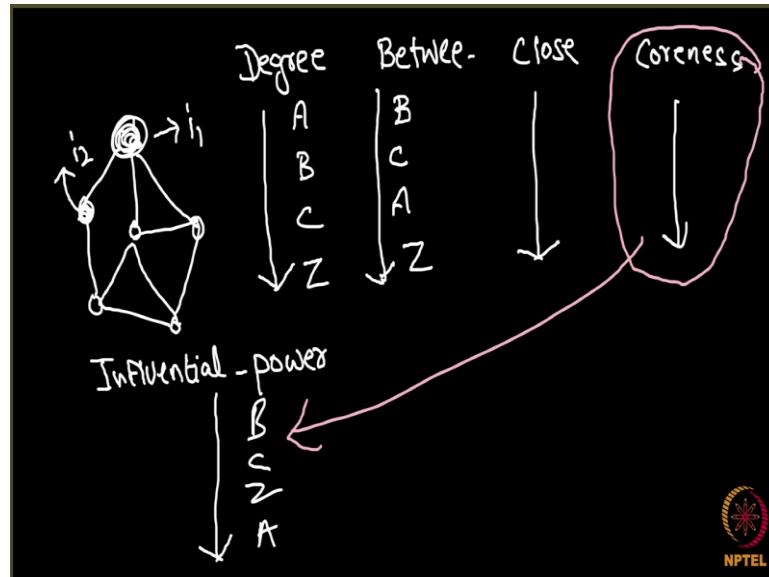


Let me say that this is the network, and then in this network, this is the so, this in this is 1 core right. So, whatever is inside this particular circle is in 1 core, whatever is inside this particular circle is in 2 core so on, and so we have 4 cores in this network. Now we know that the node which is present over here right, a node which is present over here it is a part of 1 core as well as 2 core as well as 3 core as well as 4 core right, but it is the innermost.

So, I say that the core number of this node over here is 4, the core number of all the nodes which are present over here is 3, the core number of all the nodes which are present here is 2 and which are present here is one. So, are these the nodes which lie over here which are the most influential? Has the influential power something to do with the coreness of a node which is the core number of a node?

So, this thing is what we want to look at. So, how we are going to do it is we are going to take a network. So, I take a network over here.

(Refer Slide Time: 06:47)



We will be doing it next in our next programming screen cast. So, I will take a network over here, what we are going to do? We are going to take degree of these nodes and we are going to arrange these nodes in descending order of their degree.

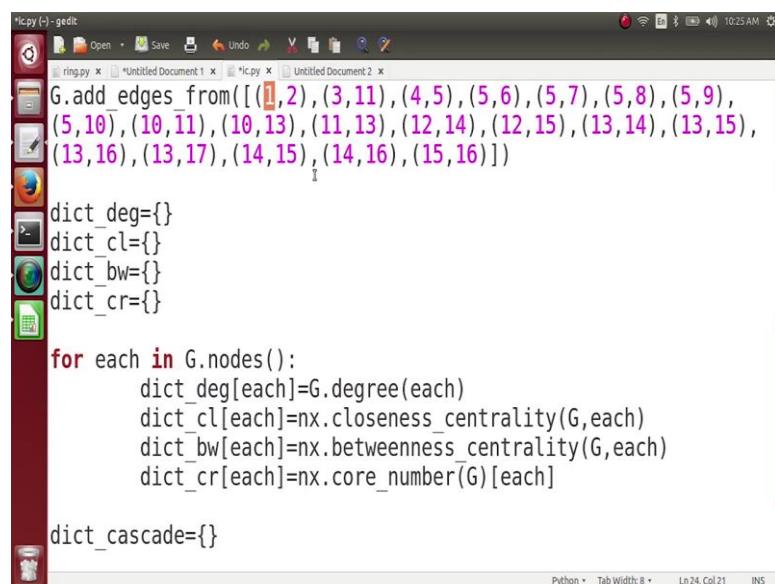
So, node which has the maximum degree comes over here and the nodes which has a minimum degree comes over here; let us say something like this. And similarly, we are

going to look at betweenness and we are going to arrange these nodes, nodes having the highest betweenness over here, least betweenness over here, similarly we are going to do for closeness, and we are going to do for coolness. And that aim is to see which out of these four measures is most related to the influential power of a node.

So, we will take a new list corresponding to the influential power. An influential power will actually take nodes and determine their influential power. So, I will take this node over here and look run independent cascade model from it and then see what is its influential power say;  $i_1$  going to do this for this same node  $i_2$  and then here i will arrange the nodes in the descending order of their influential power. So, let us say be caesar day right.

So, arrange the node here according to their influential power and then we will compare these lists. So, we will see at whether these are the nodes having highest degree which have the highest influential power or these are the nodes having highest betweenness which have highest influential power and to a surprise not surprised cannot call it supplies, but we see that actually these are the nodes having the highest value of coreness which best defines which are the most influential nodes in a network.

(Refer Slide Time: 08:39)



```
ring.py - gedit
10:25 AM
ring.py x *Untitled Document 1 x *ic.py x Untitled Document 2 x
G.add_edges_from([(1,2),(3,11),(4,5),(5,6),(5,7),(5,8),(5,9),
(5,10),(10,11),(10,13),(11,13),(12,14),(12,15),(13,14),(13,15),
(13,16),(13,17),(14,15),(14,16),(15,16)])
dict_deg={}
dict_cl={}
dict_bw={}
dict_cr={}
for each in G.nodes():
    dict_deg[each]=G.degree(each)
    dict_cl[each]=nx.closeness_centrality(G,each)
    dict_bw[each]=nx.betweenness_centrality(G,each)
    dict_cr[each]=nx.core_number(G)[each]
dict_cascade={}

Python Tab Width: 8 Ln 24, Col 21 INS
```

So, what we are going to do now is we are going to look at different-different nodes, arrange them according to certain degree closeness, betweenness, coreness and their cascading power and then see which of the centrality is matched the most with influential

power of a node. Where influential power is over noticed, if you select that particular node as seed for your infection, how many nodes are infected by the end of your process.

So, what we are going to do here now is we are going to use the same code which we have written previously for the independent cascade model ok. So, we are going to make 4 dictionaries now for holding different values. A dictionary for holding the degrees of nodes, a dictionary for holding the closeness of nodes; let us call it cl for simplicity and then dictionary for holding the betweenness of nodes. Let us call it bw and then a node a dictionary for holding the core numbers of nodes which are found by a shell decomposition and let us call that coreness denoted by cr ok.

So, we have these 4 dictionaries and now we are going to flood these dictionaries. How do we put in the values for these dictionaries for each in G.nodes what we are going to do is dictionary for degree the value corresponding to the key each which is the node e is the degree of that node, G.degree(each) right. And then dictionary corresponding to the closeness each, so there is a function in networkx you can directly calculate the closeness of a node and it is called nx.closeness\_centrality you can look it up and then stored closeness centrality.

And then we are going to put G comma each here and there is a similar function for betweenness bw is nx and this is simply called betweenness centrality. So, betweenness and a surprise for you I hope you did not know it earlier otherwise you would not have heard the previous programming screen cast. You can actually get the core number of a node directly with the help of networkx with the help of the function known as core number, which performs k shell decomposition on a network and gives you the core number for a v<sub>1</sub> node right.

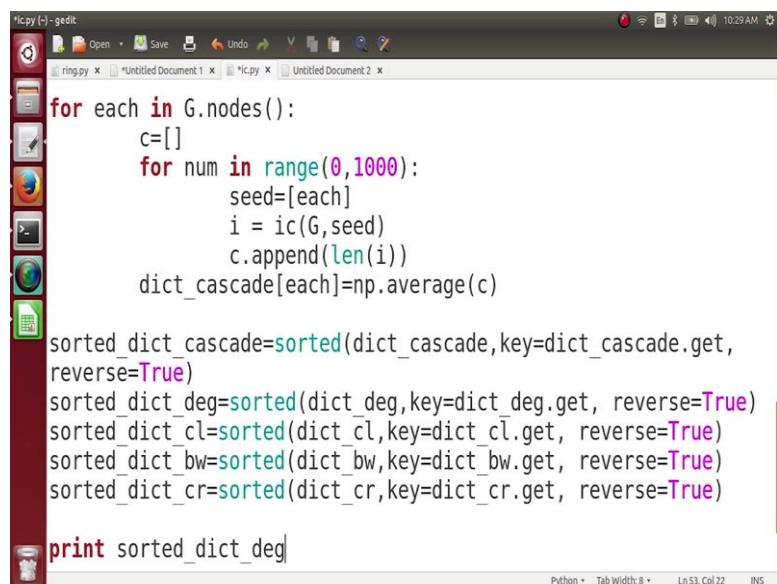
And here you see that the notation is a little bit different why. So, when you use nx.core\_number(G) is nothing but a dictionary. In that dictionary we want, so these are these are the values. So, these are the functions which returns you the values. nx.core\_number is a function. So, this is a function which returns your dictionary and from that dictionary you want the value corresponding to the node each.

So, that is why we have wrote it in this manner. So, we got the dictionaries corresponding to all these force and centralities. And at the end what we need we are creating our own dictionary which holds the cascading power of node. So, we have a

dictionary for that and how do we flood these dictionary? We have to flood this dictionary using our independent cascade model. So, how do we do that? So, in this 1 by 1 we will select every node in this network as seed. So, we will take 1s seed, 2s seed and what we will do to as we know when we run this independent cascade model, it is quite a random process right. So, we will run it some 1000 times.

So, we will take 1 s seed a run independent of cascade models from this 1000 times and take the average number of nodes which are getting infected.

(Refer Slide Time: 12:57)



```

icipy (-) - gedit
ring.py x *Untitled Document 1 x icipy x Untitled Document 2 x
for each in G.nodes():
    c=[]
    for num in range(0,1000):
        seed=[each]
        i = ic(G,seed)
        c.append(len(i))
    dict_cascade[each]=np.average(c)

sorted_dict_cascade=sorted(dict_cascade,key=dict_cascade.get,
reverse=True)
sorted_dict_deg=sorted(dict_deg,key=dict_deg.get, reverse=True)
sorted_dict_cl=sorted(dict_cl,key=dict_cl.get, reverse=True)
sorted_dict_bw=sorted(dict_bw,key=dict_bw.get, reverse=True)
sorted_dict_cr=sorted(dict_cr,key=dict_cr.get, reverse=True)

print sorted_dict_deg

```

So, what do we do? For each in G.nodes what we are going to do is c is currently empty. And then we take a loop which runs 1000 times for numbing range to 0 to 1000 and what we are going to do we said the seed of the process to be equal to each and then we call our independent cascade model with graph G and seeds.

So, now this value i over here is nothing but a list which holds the final people infected when the seed was the 1 node each here. So, what do we do? We append this length of i inner array c right. So, some 1000 times we get different-different people who are infected for different-different processes and for all these 1000 times we append the number of people who are infected in this array c.

So, this is array c we will be consisting of 1000 entries and each entry will correspond to 1 process where this node each was infected, and some people were infected by the end

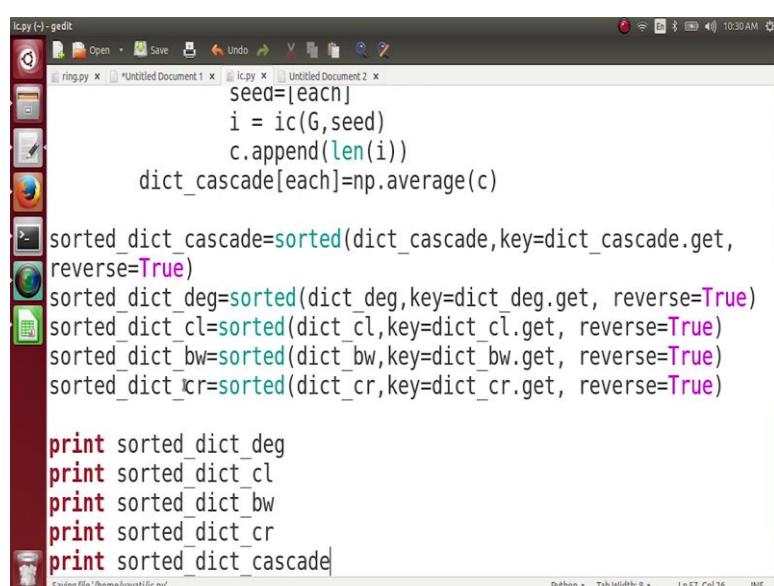
of the process. And now what I am going to append in my dictionary cascade dig the cascade each is the average, average from c, that is on an average how many people got infected, if I started my infection from this node each right.

And then what I am going to do now is I have find out I found out 5 dictionaries, 1 for degree closeness betweenness cornice and 1 for cascading and now I want to see which out of these 4 correlates the most with this 1, which is our degree for cascade.

So, what I am going to do is I am going to sort all of these dictionaries. So, sorting is easy with 1 command, so I will let first sort this dictionary corresponding to cascade So, there is a function sorted which you can call on a dictionary. So, you pass your dictionary cascade here right and we want to sort according to the keys are going to be nothing, but dictionary cascade.get and we want to sort them in a reverse order. So, we said reverse equals to true and we do this with all our dictionaries with all the remaining 4 dictionaries.

So, this cascade thing we can core degree we will replace it with degree and for closeness we replace it with cl and for betweenness we replace it with bw, bw and here also bw right and four corners we replace it with cr. And we can actually print all of these and see what is happening.

(Refer Slide Time: 16:29)



The screenshot shows a Gedit text editor window with several tabs open. The current tab contains the following Python code:

```
ic.py (-) - gedit
ring.py x Untitled Document 1 x ic.py x Untitled Document 2 x
seed=[each]
i = ic(G,seed)
c.append(len(i))
dict_cascade[each]=np.average(c)

sorted_dict_cascade=sorted(dict_cascade,key=dict_cascade.get,
reverse=True)
sorted_dict_deg=sorted(dict_deg,key=dict_deg.get, reverse=True)
sorted_dict_cl=sorted(dict_cl,key=dict_cl.get, reverse=True)
sorted_dict_bw=sorted(dict_bw,key=dict_bw.get, reverse=True)
sorted_dict_cr=sorted(dict_cr,key=dict_cr.get, reverse=True)

print sorted_dict_deg
print sorted_dict_cl
print sorted_dict_bw
print sorted_dict_cr
print sorted_dict_cascade
```

The code defines a list `seed` containing `[each]`. It then initializes `i` using `ic(G, seed)`, calculates its length with `len(i)`, and stores the result in `c`. The average of `c` is stored in `dict_cascade[each]` using `np.average(c)`.

Below this, the code uses the `sorted` function to sort four dictionaries (`dict_cascade`, `dict_deg`, `dict_cl`, `dict_bw`, and `dict_cr`) based on their values. The sorted dictionaries are then printed.

Print let us print degree first and then let us print degree and then let us print closeness and then let us print betweenness, and then coreness, and then cascade right. Let us implement it and see no ok.

(Refer Slide Time: 16:55)

```
yayati@yayati-Vostro-3546: ~
[2] [1, 2]
[] [1, 2]
[1]
[1] [1]
[] [1]
[1]
[1] [1]
[] [1]
[1]
[1] [1]
[] [1]
[1]
[1] [1]
[1]
[1] [1]
[2] [1, 2]
[] [1, 2]
Traceback (most recent call last):
  File "ic.py", line 45, in <module>
    dict.Cascade[each]=np.average(c)
NameError: name 'np' is not defined
yayati@yayati-Vostro-3546:~$
```

So, we have used this numpy module for finding the average we need to import it.

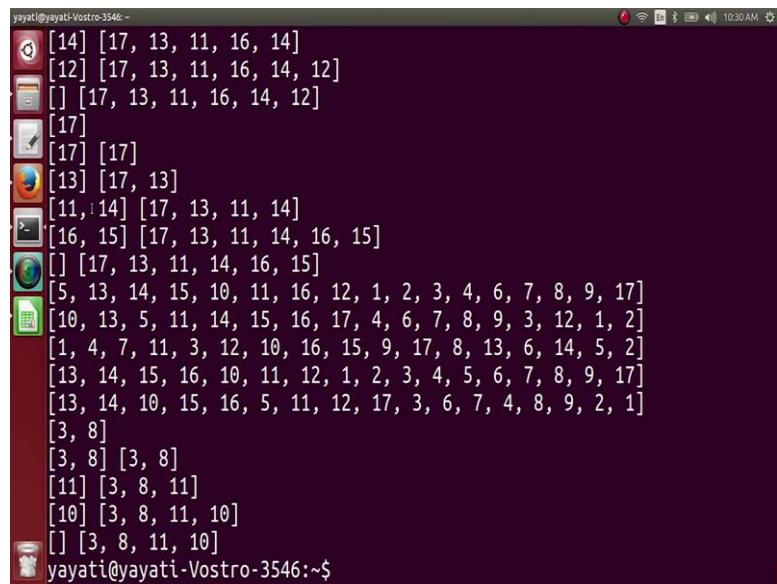
(Refer Slide Time: 17:05)

```
ic.py (~) - gedit
ring.py x *Untitled Document 1 x *ic.py x Untitled Document 2 x
import networkx as nx
import matplotlib.pyplot as plt
import random
import numpy as np

def ic(G,s):
    print s
    jst_inf=list(s)
    infected=[]
    while(1):
        print jst_inf, infected
        if len(jst_inf)==0:
            return infected
        tmp=[]
        for each in jst_inf:
            for each1 in G.neighbors(each):
                r=random.uniform(0,1)
                if r<=0.1:
```

So, import numpy as np and actually we were printing some extra values where we print some extra values not sure, we are actually printing ok.

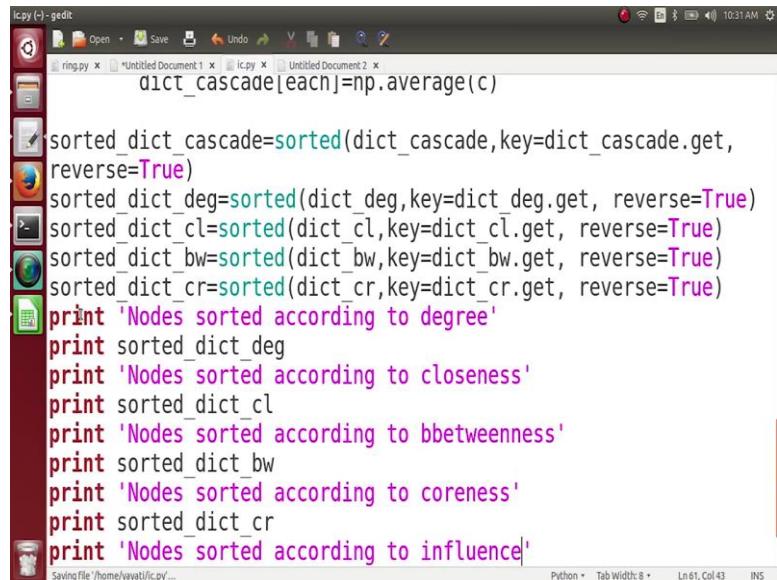
(Refer Slide Time: 17:31)



```
yayati@yayati-Vostro-3546:~ [14] [17, 13, 11, 16, 14] [12] [17, 13, 11, 16, 14, 12] [] [17, 13, 11, 16, 14, 12] [17] [17] [17] [13] [17, 13] [11, 14] [17, 13, 11, 14] [16, 15] [17, 13, 11, 14, 16, 15] [] [17, 13, 11, 14, 16, 15] [5, 13, 14, 15, 10, 11, 16, 12, 1, 2, 3, 4, 6, 7, 8, 9, 17] [10, 13, 5, 11, 14, 15, 16, 17, 4, 6, 7, 8, 9, 3, 12, 1, 2] [1, 4, 7, 11, 3, 12, 10, 16, 15, 9, 17, 8, 13, 6, 14, 5, 2] [13, 14, 15, 16, 10, 11, 12, 1, 2, 3, 4, 5, 6, 7, 8, 9, 17] [13, 14, 10, 15, 16, 5, 11, 12, 17, 3, 6, 7, 4, 8, 9, 2, 1] [3, 8] [3, 8] [3, 8] [11] [3, 8, 11] [10] [3, 8, 11, 10] [] [3, 8, 11, 10] yayati@yayati-Vostro-3546:~$
```

So, we do not see all these values because, so many times our process have run ok. So, we can actually if we do not need this portion of the core and let us now run it and see. And we do not know whether where one particular list is ending. So, let us say here print, so take degrees.

(Refer Slide Time: 17:59)



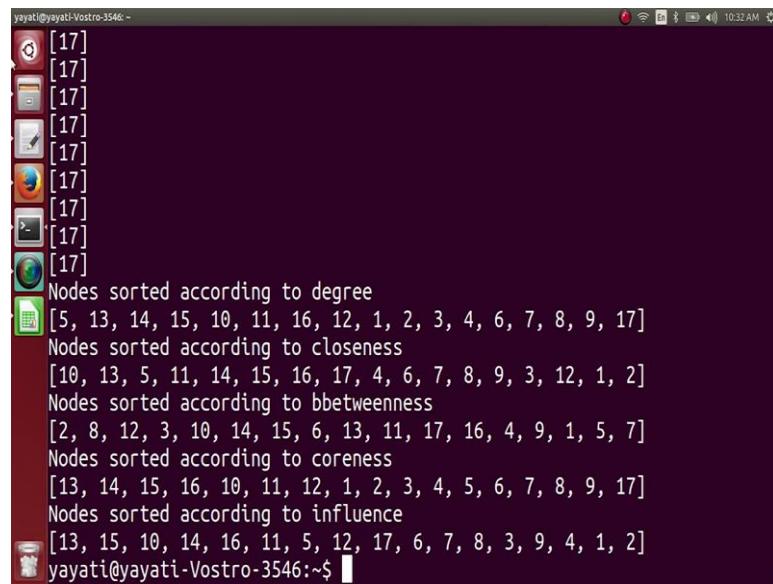
```
lc.py (-) - gedit
ringspy x *Untitled Document 1 x lc.py x Untitled Document 2 x
    dict_cascade[each]=np.average(c)

sorted_dict_cascade=sorted(dict_cascade,key=dict_cascade.get,
reverse=True)
sorted_dict_deg=sorted(dict_deg,key=dict_deg.get, reverse=True)
sorted_dict_cl=sorted(dict_cl,key=dict_cl.get, reverse=True)
sorted_dict_bw=sorted(dict_bw,key=dict_bw.get, reverse=True)
sorted_dict_cr=sorted(dict_cr,key=dict_cr.get, reverse=True)
print 'Nodes sorted according to degree'
print sorted_dict_deg
print 'Nodes sorted according to closeness'
print sorted_dict_cl
print 'Nodes sorted according to betweenness'
print sorted_dict_bw
print 'Nodes sorted according to coreness'
print sorted_dict_cr
print 'Nodes sorted according to influence'

Saving file /home/yayati/lc.py...
Python Tab Width: 8 Ln 61, Col 43 INS
```

So, these are your nodes sorted according to degree, according to closeness, according to betweenness, according to coreness and according to we can call it nothing but the influence fine.

(Refer Slide Time: 18:45)



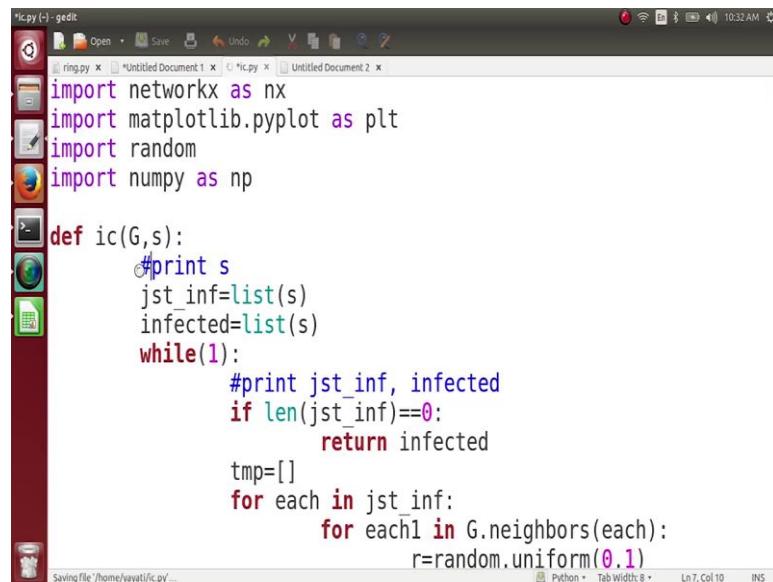
yayati@yayati-Vostro-3546:~

[17]  
[17]  
[17]  
[17]  
[17]  
[17]  
[17]  
[17]  
[17]  
Nodes sorted according to degree  
[5, 13, 14, 15, 10, 11, 16, 12, 1, 2, 3, 4, 6, 7, 8, 9, 17]  
Nodes sorted according to closeness  
[10, 13, 5, 11, 14, 15, 16, 17, 4, 6, 7, 8, 9, 3, 12, 1, 2]  
Nodes sorted according to betweenness  
[2, 8, 12, 3, 10, 14, 15, 6, 13, 11, 17, 16, 4, 9, 1, 5, 7]  
Nodes sorted according to coreness  
[13, 14, 15, 16, 10, 11, 12, 1, 2, 3, 4, 5, 6, 7, 8, 9, 17]  
Nodes sorted according to influence  
[13, 15, 10, 14, 16, 11, 5, 12, 17, 6, 7, 8, 3, 9, 4, 1, 2]

yayati@yayati-Vostro-3546:~\$

So, we have many extra things printed here which we actually do not want. Somewhere this print statement is coming ok, so it is coming from here we can actually comment it and run it again. Still something is sprinting, s is sprinting here, we comment it also see here.

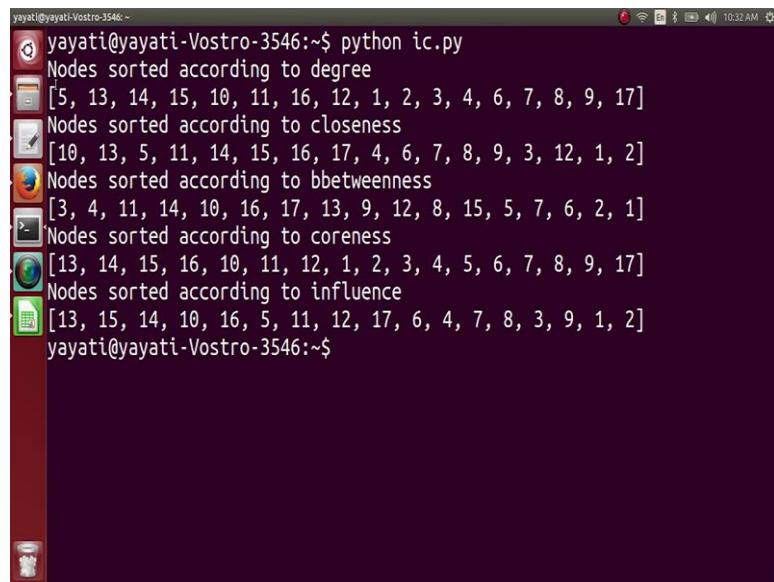
(Refer Slide Time: 19:07)



```
*ic.py (-) -gedit
import networkx as nx
import matplotlib.pyplot as plt
import random
import numpy as np

def ic(G,s):
    #print s
    jst_inf=list(s)
    infected=list(s)
    while(1):
        #print jst_inf, infected
        if len(jst_inf)==0:
            return infected
        tmp=[]
        for each in jst_inf:
            for each1 in G.neighbors(each):
                r=random.uniform(0.1)
                if r < 0.5:
```

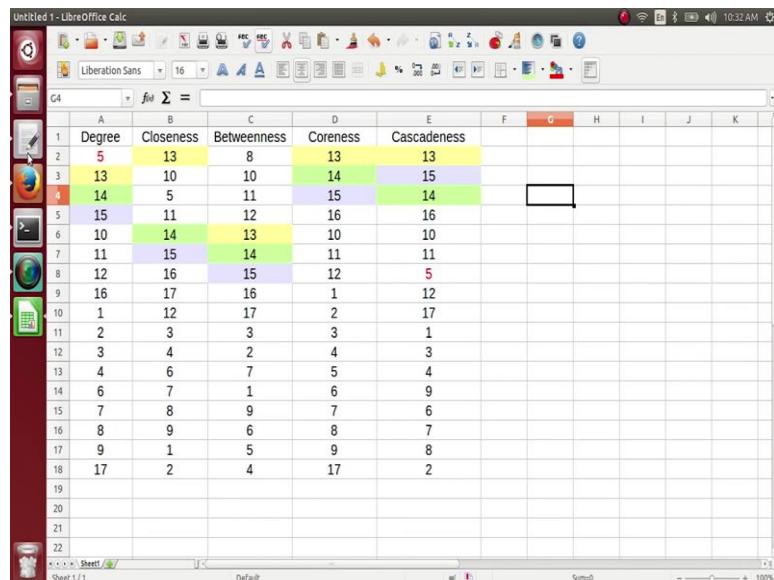
(Refer Slide Time: 19:11)



```
yayati@yayati-Vostro-3546:~$ python ic.py
Nodes sorted according to degree
[5, 13, 14, 15, 10, 11, 16, 12, 1, 2, 3, 4, 6, 7, 8, 9, 17]
Nodes sorted according to closeness
[10, 13, 5, 11, 14, 15, 16, 17, 4, 6, 7, 8, 9, 3, 12, 1, 2]
Nodes sorted according to betweenness
[3, 4, 11, 14, 10, 16, 17, 13, 9, 12, 8, 15, 5, 7, 6, 2, 1]
Nodes sorted according to coreness
[13, 14, 15, 16, 10, 11, 12, 1, 2, 3, 4, 5, 6, 7, 8, 9, 17]
Nodes sorted according to influence
[13, 15, 14, 10, 16, 5, 11, 12, 17, 6, 4, 7, 8, 3, 9, 1, 2]
yayati@yayati-Vostro-3546:~$
```

So, these are your nodes sorted according to your degrees, node sorted according to closeness between this coreness and influence. So, what I have done? I have actually run this process and I have kept these values here. So, these values might not be exactly the same.

(Refer Slide Time: 19:35)



	A	B	C	D	E	F	G	H	I	J	K
1	Degree	Closeness	Betweenness	Coreness	Cascadeness						
2	5	13	8	13	13						
3	13	10	10	14	15						
4	14	5	11	15	14						
5	15	11	12	16	16						
6	10	14	13	10	10						
7	11	15	14	11	11						
8	12	16	15	12	5						
9	16	17	16	1	12						
10	1	12	17	2	17						
11	2	3	3	3	1						
12	3	4	2	4	3						
13	4	6	7	5	4						
14	6	7	1	6	9						
15	7	8	9	7	6						
16	8	9	6	8	7						
17	9	1	5	9	8						
18	17	2	4	17	2						
19											
20											
21											
22											

So, for degree and closeness these are going to be the exactly the same. So, if you see a 5, 13, 14, 15 it is going to be the same 5, 13, 14, 15. For finding out the betweenness we use a random process, the algorithm uses a random process.

So, the value here might not be very same as what is here and for the coreness it is going to be the same. And for influence it is mostly going to be the same because, we have run this process a 1000 times. So, the results which we are going to get will be the same 13, 14, 15, 10 and you can see 13, 14, 15, 10 and so on. Ok let us analyze what's happening. So, here I have taken a spreadsheet, here I have put nodes according to a descending order of degree, descending order of closeness and so on.

Now I want to show you here something. Do you see here this cascade ness? It is actually nothing but the influence, so it is the influence here ok. So, if you see here you see the node which is having the highest influence is the one having the highest coreness. And actually the highest closeness it means that maybe closeness and coreness relates to influence, but if you look at the second highest node here which created the second highest cascade it as a high coreness, but it does not have a high closeness.

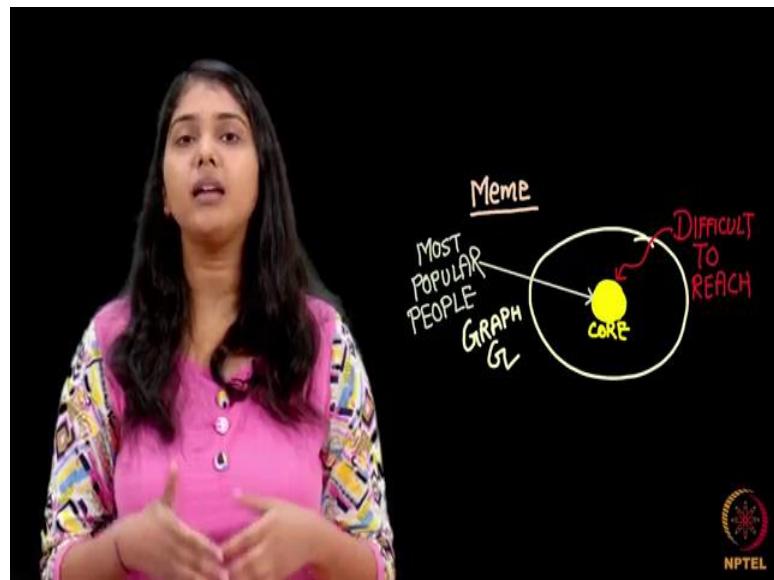
So, overall, you can see that this influence over here is mostly related to the coreness and degree. So, closeness if you see nodes 10, 5 and 11 here this goes quite down here and then this node edge here comes over here. So, the influence is maximally correlated with the coreness and degree here and out of coreness and degree although you can see that it is mostly related to coreness. This node 5 over here has a very high degree, but it has a very less influential power. So, this result it validates it is that: if you want to look at the maximum influential power of a node you should look at the nodes which are having the maximum coreness.

So, I have done this experiment over here for 1 network. You can actually replicate this experiment for various different networks rather you can use some real-world data sets also and verify this.

**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**How to go Viral on Web**  
**Lecture - 166**  
**Pseudo core**

(Refer Slide Time: 00:09)



I have this internet mean which I really want to become popular. I look at my network, I identify the core nodes and I try and convince them to adopt my idea. But, is that so easy? So, generally in this core are the most popular people of a network like the public figures, Bollywood celebrities let us say Amitabh Bachchan and Shahrukh Khan kind of people. Can I go to them and really ask them to share my meme further? Why will they listen to me? We know that it is very difficult to reach such people.

So, how do we use this strategy? So, one idea is to obviously, pay them a lot of money like this different advertising companies do. So, if you want to let us say your next soap to become popular, you try to give some money to a say of Bollywood heroine and ask her to act in your shoot and so on. But, can we use a smarter approach? I might not have that much amount of money, how do I meet these core nodes when I do not have enough amount of money to give them, I do not know.

So, reaching these core nodes is actually very difficult and even sometimes with your money they might also not be ready to adopt your idea. That is mostly because these people are inside, they form the crux of this network they are in the innermost part of this network. So, it is really very difficult to infect such people but see what now next we are going to do, we looked at this (Refer Time: 01:46) decomposition algorithm.

So, we have seen that there were multiple buckets. So, we had this bucket number 1, 1 core, bucket number 2, 2 core, bucket number 3, 3 core. What does this bucket actually tell us? These buckets actually tell us the different influence levels in this network.

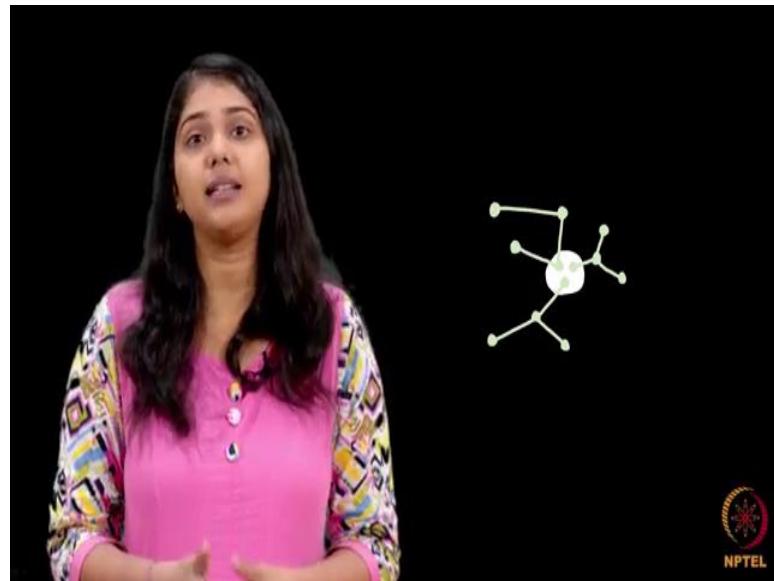
(Refer Slide Time: 02:08)



So, if you look at this bucket 12; let us say that twelve is the last bucket we formed. If you look at this bucket 12, it has the most influential people of the network the innermost core. If you look at bucket 1, it consists of the least influential people in the network. And if you look at let us say bucket 7, then it is somewhere in middle based on its influence.

So, we can actually look at this network in this way where, based on your buckets your network gets divided into multiple shells, outermost shell corresponds to bucket 1, innermost shell corresponds to bucket 12 and then here are our different buckets. So, we have here this network divided into multiple shells. And then we have seen that it is very difficult to convince these people in the innermost core.

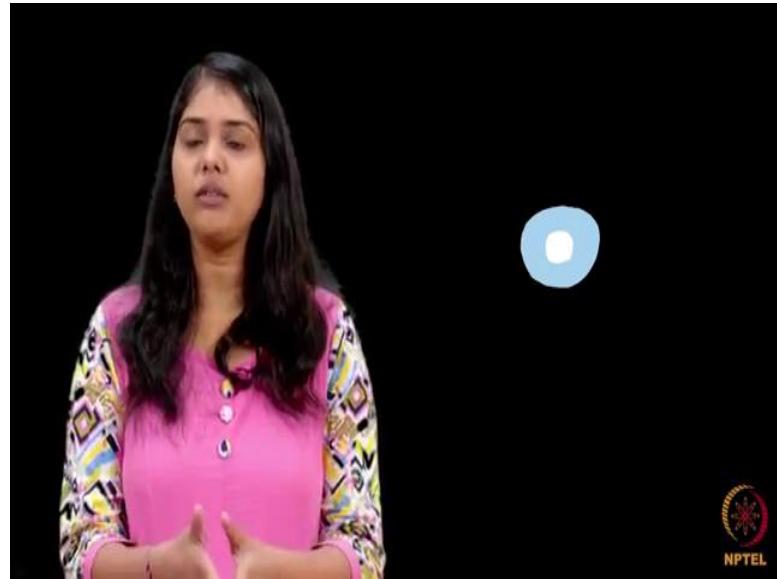
(Refer Slide Time: 03:05)



What next we will try to do if what we will do if we will look at these different, different shells and we will examine the influential power of all shells, what do I mean by that?

So, we have done this the done that experiment right where we choose some c nodes which are having the highest degree and then looked at how many people go to infected at the end or we looked at the nodes having highest closeness. And then looked at how many people got infected at the end. We do the same procedure here. We first of all choose nodes from the innermost shell; perform our cascade and look at how many nodes get infected at the end.

(Refer Slide Time: 03:46)



Then we take this second shell here. So, let us say the innermost shell is shell 12, we took shell 12 look at the how many people go to infected towards the end. Then we take the shell number 11, again take some nodes from here infect the network and then look at how many people got infected and we term this just a term.

So, this final number of nodes which got infected here, we call it as the cascade capacity of a shell, of a shell. Cascade capacity means if you start infecting the network from the shell choosing the seed nodes from this shell, what is the final amount of network that you have infected.

And we do it for all the shells. What do you think is the result, should it be if i start the infection from the outermost shell, shell number 1, very less people get infected towards the end, cascade capacity is very less? If I start from this second shell cascade capacity is a little bit more, third shell a little bit more and then it keeps increasing.

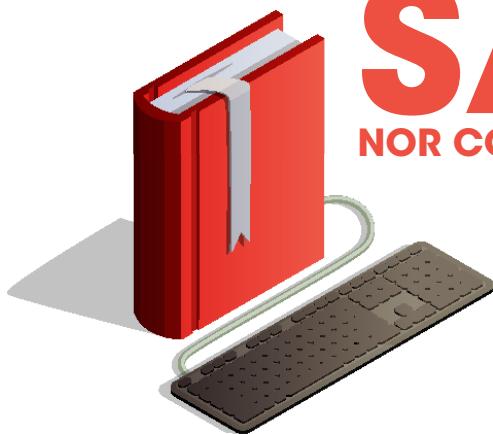
We expect the result should be something like that but when you do it, so what we did is we took some we can actually take some real world networks we can take some synthetic networks also, but since we are talking about the internet meme propagation in real world social networks, it is better to take our network which are which correspond to the online social networks.

So, we can take; so all these networks as we have seen before in our chapter data sets are now available online. You can get a graph for Facebook, you can get a graph for google plus, you can get a graph for twitter and almost every other online social network. So, you choose some of these social networks and then applying (Refer Time: 05:41) decomposition is very easy here. And then do this experiment. Take the nodes from different, different shells and look at the final cascade capacity.

And when you plot this graph, plot this graph on the x axis is the shell number, on the y axis is the cascade capacity what you observe? You see that it is not linear like what we expected. We will see what is happening here; It starts increasing starts increasing and add some shell it reaches its maximum and then it becomes constant. What does that mean? That means that to infect this network to make your meme go viral, it is not always necessary for you to go to the innermost core and convince these people.

So, if you look at some of these shells outer from this core shell, some of these they have actually the same cascade capacity as the core nodes. So, instead of these core nodes we have found a lot more people here which are actually we can called pseudo course, we call these people as Pseudo course. So, there is a big chunk of the pseudo core people, whom you can actually infect to make your meme go viral.

**THIS BOOK  
IS NOT FOR  
SALE  
NOR COMMERCIAL USE**



(044) 2257 5905/08



nptel.ac.in



swayam.gov.in