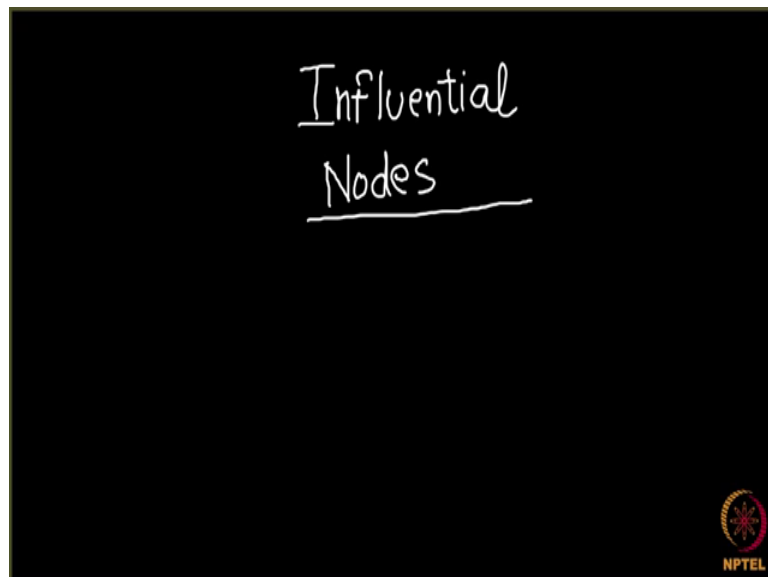


Social Networks
Prof. S. R. S. Iyengar
Department of Computer Science
Indian Institute of Technology, Ropar

How to go Viral on Web
Lecture - 164
Coding cascading Model

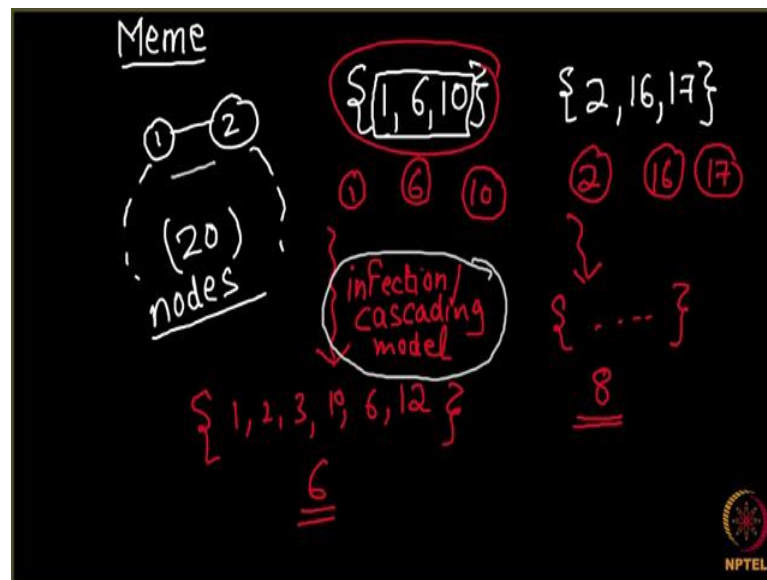
So, next what is our aim? Our aim is to determine the influential pair of nodes. For actual question is we want to see that which nodes in our network are the most influential.

(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)



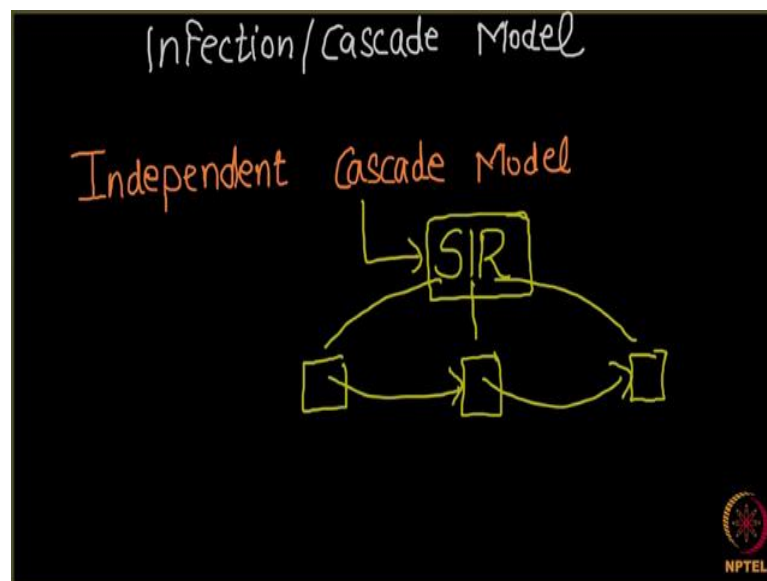
So, let us say there is a meme which wants to traverse on a network which is traversing on a network, let us say the network is something like this consisting of let us say some 20 nodes, 20 nodes some network. And now, we want to see out of these 20 nodes which nodes are the most influential? What we 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.

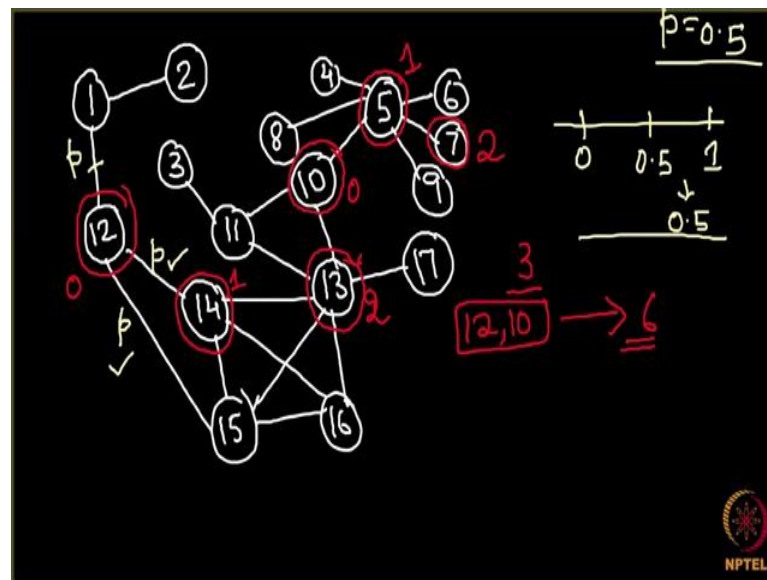
(Refer Slide Time: 03:23)



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 model 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 it 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 way 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.

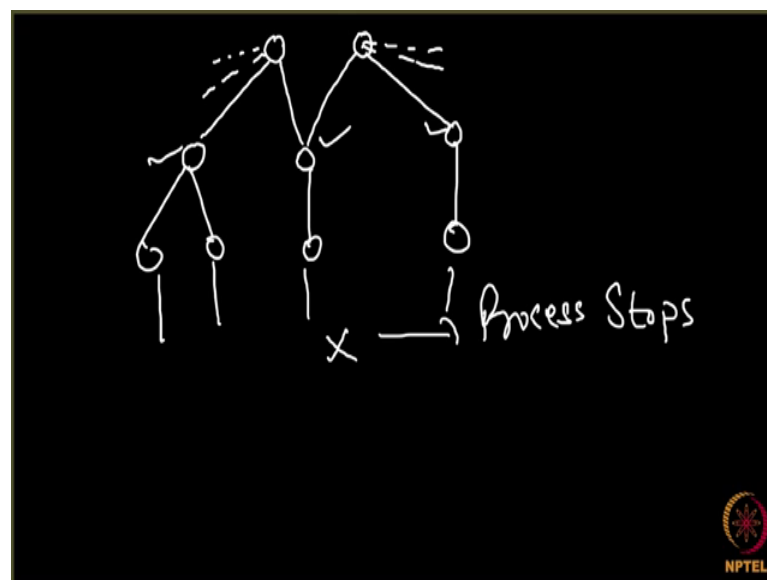
So, time iteration 1 node 13 gets infected. And in the same iteration this node 5 over here also tries to infect its remaining neighbors. So, there are 5 neighbors and let us say it succeeds in infecting this neighbor 7. So, at time iteration 2, so at time iteration 2 this node 13 over here and this 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 seven 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)



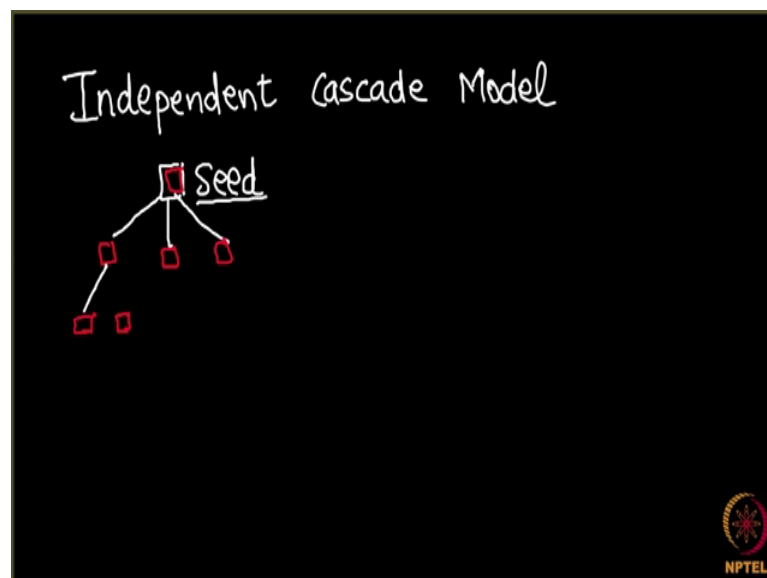
So, what is happening is initially there are certain nodes which are infected let us say these 2 nodes are infected. In the next iteration these nodes will look at their neighbors. So, this can be a possible case here right. So, these nodes will look at their neighbors and infect each of them with the 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.

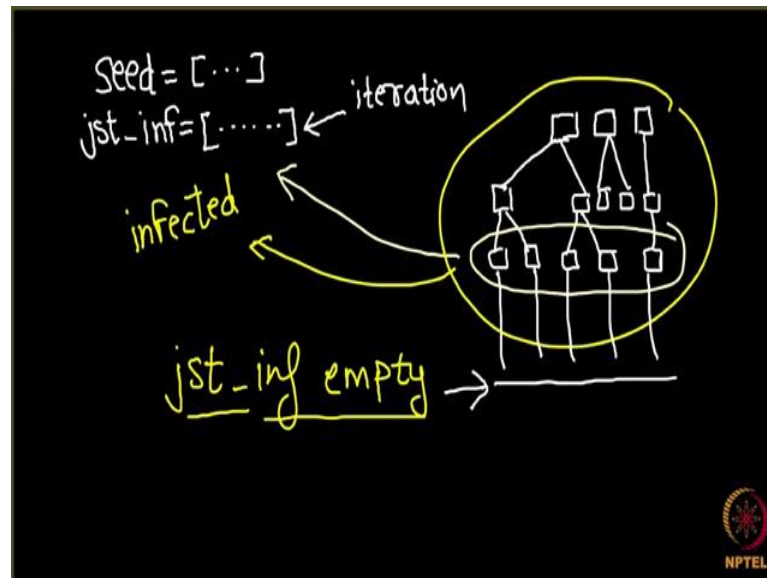
(Refer Slide Time: 11:07)



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.

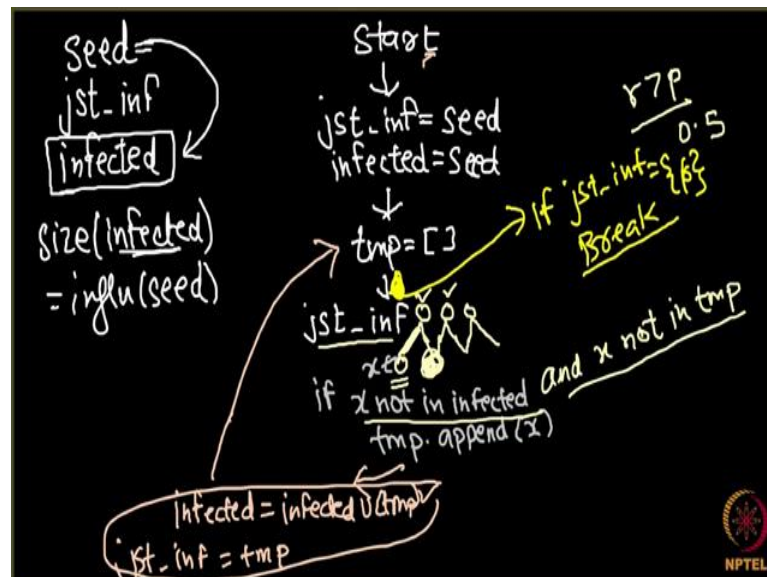
So, at this iteration this becomes the least just infected right and then we will also be using another list which is infected. And this list holds all the nodes over overall the process throughout the process till now whichever nodes are infected comes in this infected list. So, you see what is what 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 not in this list infected it is in list (Refer Time: 19:41). So, we might again append it to tmp. 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.

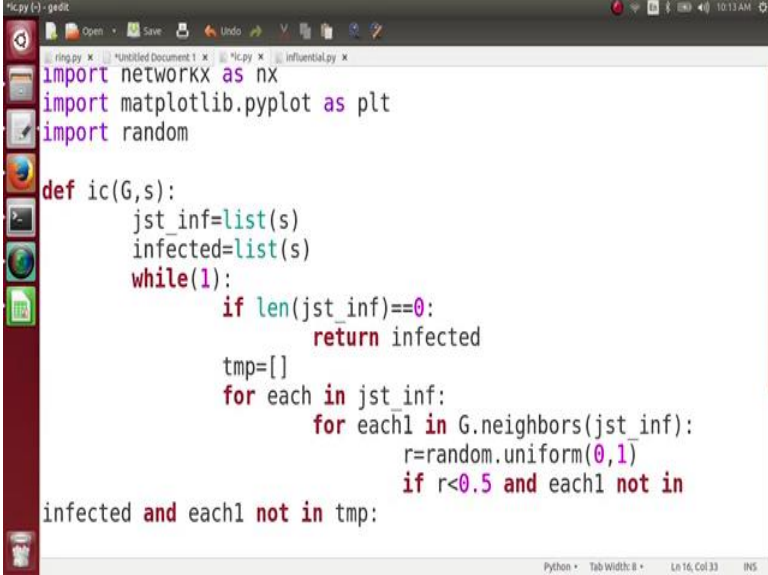
So, we will simply for all the nodes in tmp we set our list infected right. So, these all node should be infected. Infected becomes infected union tmp right. So, these all nodes become infected first of all then. Secondly, you notice that now my set of just infected should change right. So, previously 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 a 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)



```
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

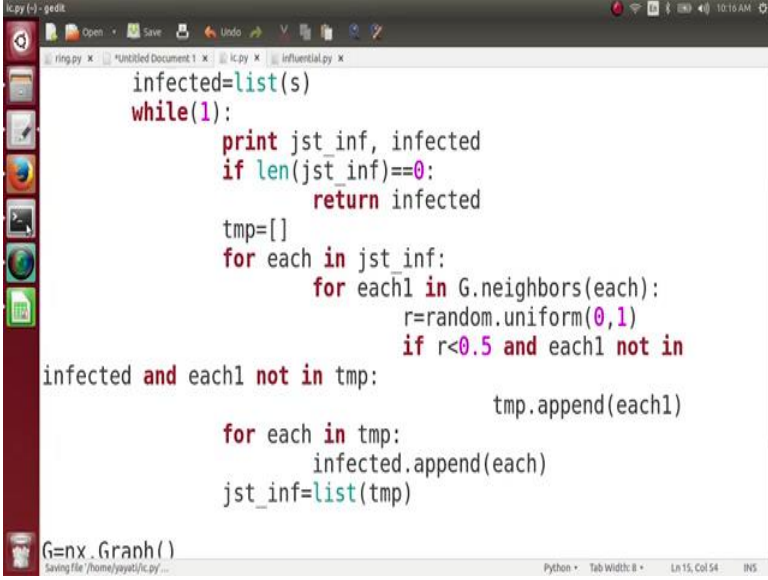
initially, both of these nodes have the same value as c . And then we have a loop while 1, while 1 what are we going to do, we know that what is the terminating condition for this loop. So, there should be some persons who are just infected. If nobody is just infected, we should break.

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)

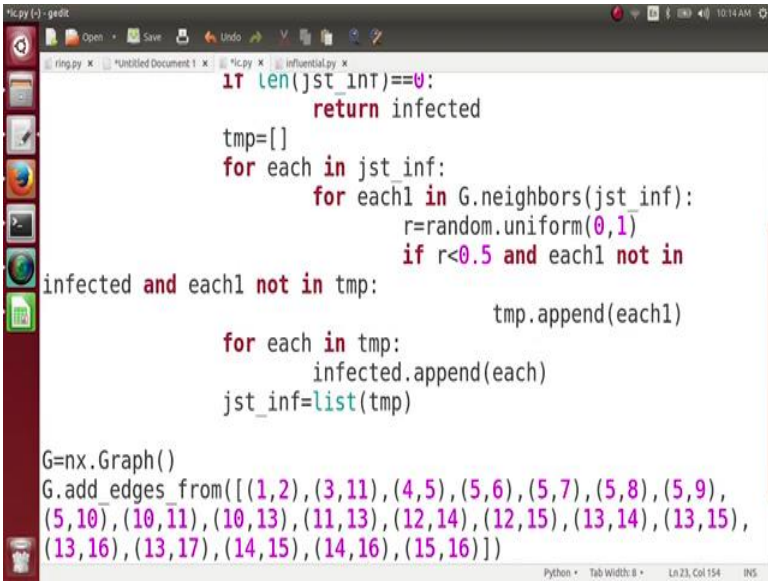


```
infected=list(s)
while(1):
    print jst_inf, infected
    if len(jst_inf)==0:
        return infected
    tmp=[]
    for each in jst_inf:
        for each1 in G.neighbors(each):
            r=random.uniform(0,1)
            if r<0.5 and each1 not in
infected and each1 not in tmp:
                tmp.append(each1)
    for each in tmp:
        infected.append(each)
    jst_inf=list(tmp)

G=nx.Graph()
```

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)

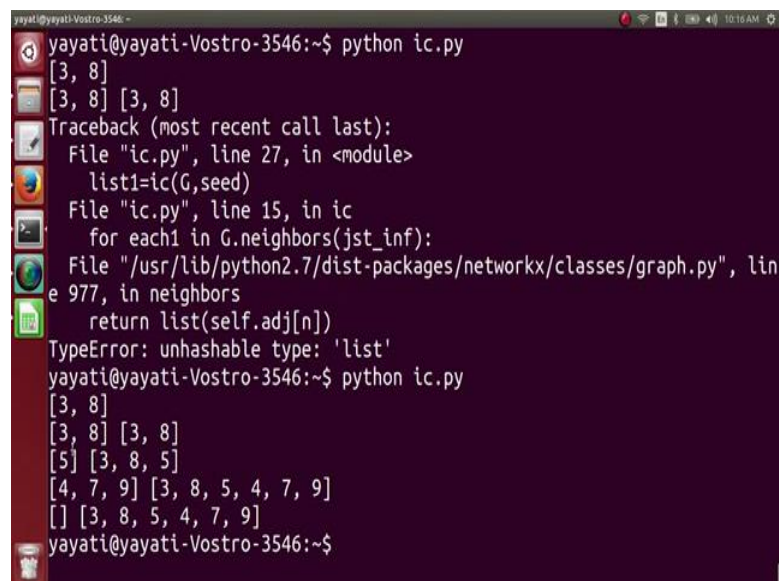


```
if len(jst_inf)==0:
    return infected
tmp=[]
for each in jst_inf:
    for each1 in G.neighbors(jst_inf):
        r=random.uniform(0,1)
        if r<0.5 and each1 not in
infected and each1 not in tmp:
            tmp.append(each1)
    for each in tmp:
        infected.append(each)
    jst_inf=list(tmp)

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 nodes 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)

A terminal window with a dark purple background and white text. The prompt is 'yayati@yayati-Vostro-3546:~\$'. The user runs 'python ic.py'. The output shows a list '[3, 8]' followed by '[3, 8] [3, 8]'. Then a 'Traceback (most recent call last):' error occurs. The traceback points to 'File "ic.py", line 27, in <module>' where 'list1=ic(G,seed)' is called, then to 'File "ic.py", line 15, in ic' where 'for each1 in G.neighbors(jst_inf):' is used. The error is 'TypeError: unhashable type: \'list\'' at line 977 in 'neighbors' where 'return list(self.adj[n])' is executed. The user runs 'python ic.py' again, and the output shows a sequence of lists: '[3, 8]', '[3, 8] [3, 8]', '[5] [3, 8, 5]', '[4, 7, 9] [3, 8, 5, 4, 7, 9]', and finally '[] [3, 8, 5, 4, 7, 9]'. The prompt returns to '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.