Week-3 ~ **Strong and Weak Relationships**

Worked on

- September 3rd, Saturday
- September 4th, Sunday

Module-01: **Introduction**

on 3rd September, 2021 ~ Friday

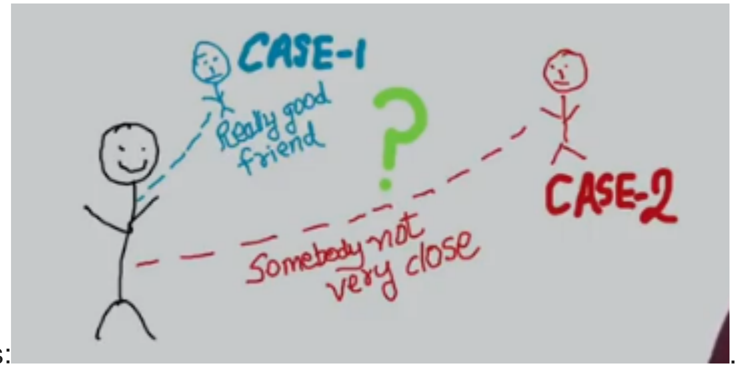
Started with a short clip from the movie: **Harry potter and the Philosopher's Stone (2001), Directed by: Chris Columbus *(Based on Harry Potter by J.K. Rowling)*** in which...

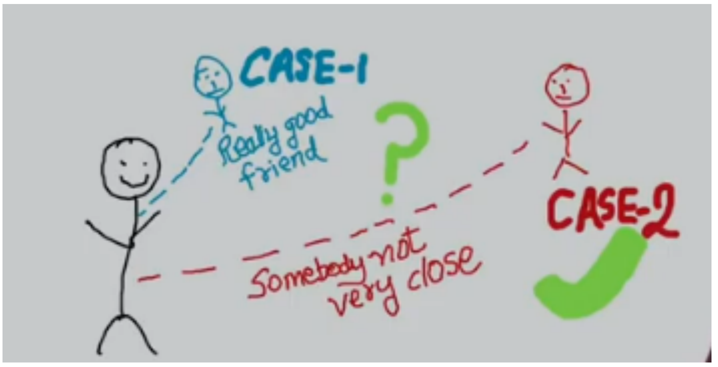
> - Hagrid comes to take Harry Potter. He is perplexed, he is not sure who is this 8foot long person -- who knocks the door off and enters.
> - And when he asked, Who are you?, Why are you here?
> - Hagrid replies: Iam here to take you Hokwortz -- He doesn't even know what it is.. Its the school of Witchcraft and Wizadry and you are invited to come and join..

To put the theme of usage of it in this week..

> Hagrid comes to meet Harry potter and take him. And harry potter doesn't know him.

Pause for a min... and think... Is this what happens in a real life.. **Does opportunites come from those, whom we are not close.. or a really good friend..??**

-- 2 cases:


.. Well study says, the case-2. i.e., Most pople get opportunites from which whom they are not close with.. -- something called as **Weak-Tie** (Tie means FriendShip), A Weak Friendship...


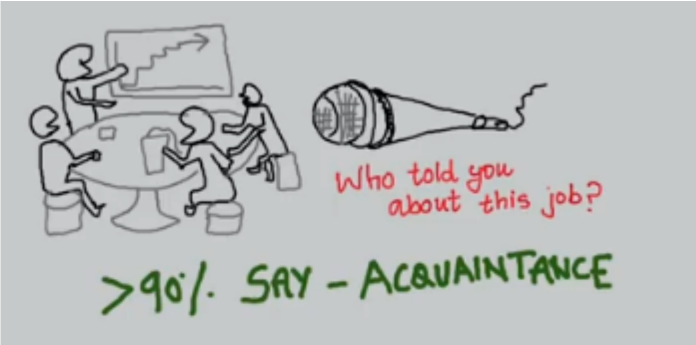
---

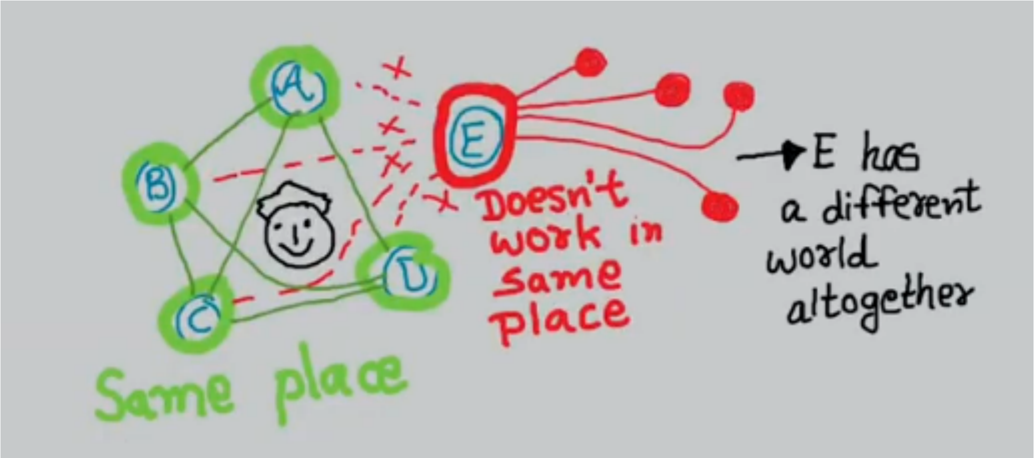
i.e., say if you have 100 friends, you'll get opprortunity from the one, who is not a close friend, but an acquaitance --- **How's that..??**, that's what makes this week, come let's explore..

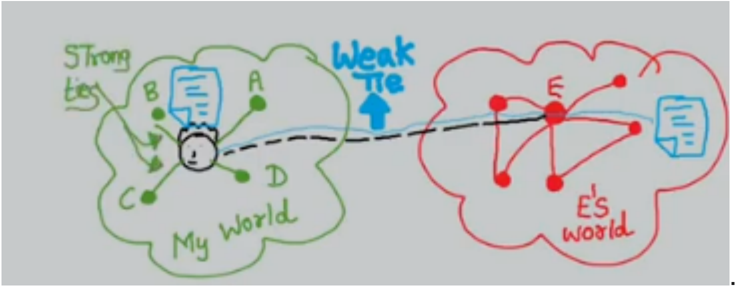
Module-2: **Granovetter's Strength of Weak ties**

Assume I went to 100 people and asked them randomly, **You are working in this office, who told you about this job opportunity?**

> More than 90% will say that... **This person's friend who is my friend's friend... or someone to whom I was not close to him, told about this job.. etc...**, very rarely we encounter like.. said by my father..

> 
>

- This might be counter-intuitive.... Is it like, our close friends don't want to say the job-opportunity??  ...Not really..!! the following reason, makes convincible..
- Assume a scneario, you have 5 friends, 4 are of them are very close to you and all those are from same world(i.e., all those work in a same place). Whatever they know, even you know most of them..Ofcourse, occasionally may tell a new information.
- Now see the E's role (5th friend) here.., he is from different world together. Whatever he knows, you may not be getting from close friends. 

. This E's information, to which you are slightly aware of. So, he is the one who'll tell the information from his world, which increases your sample space of information, which A, B, C, D can't.
- Given that E is a distant friend of you, probably a weaker friendship with him. That's called as **Weak Tie**, and with the A,B, C, D its called **Strong Tie** 

.
- Strong ties are weak when it comes to telling a new job related information. And, Weak ties are strong in-terms of telling the new job opportunity, which will be of worth-knowing. Hence..

> **Weak** ties are actually **Strong**, while the **Strong** ties are actually **weak**.

Historical Perspective....

- **Granovettor** in the late 1960s, conducted his experiment that: **From where do you got this job opportunity?**, and most of them replied **it was through Acquantainces**.
- That is when, he observed, something counter-intuitive is being heppening, and conducted experiments and concluded that **Strength of weak-ties is in play here** and tried publishing a paper in 1969 -- but the scientific community didn't accepted it, so got rejected.
- Later in 1973 people started accepting his vies , and it got accepted. Then after getting accepted, it became reslly really famous -- famous simply because, it is **Such an important concept yet very counter-intuitive**.

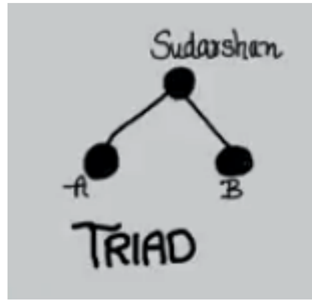
MORAL: Keep your acquantainces happy, you should not just treat your good friends. --*There is some fallacy here, if you started treating acaquaintances, then they become good friends and you may not get new information.*

So, the point is: You should have a big friend circle, your opportunites increase, but the new opprotunites come acquantinces for the above reasons listed.

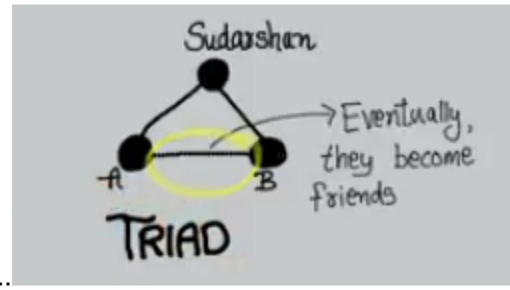
So, now let's get familiar with the slightly abstract mathematical concepts around these

Module-3: **Triads, Clustering coefficient & Neighborhood overlap**

Triads and Triadic closure...

-- Root-node is the person, and child-nodes are the people whom he is friend of. **NOTE:** Here child nodes doesn't know each other themselves directly. 

.

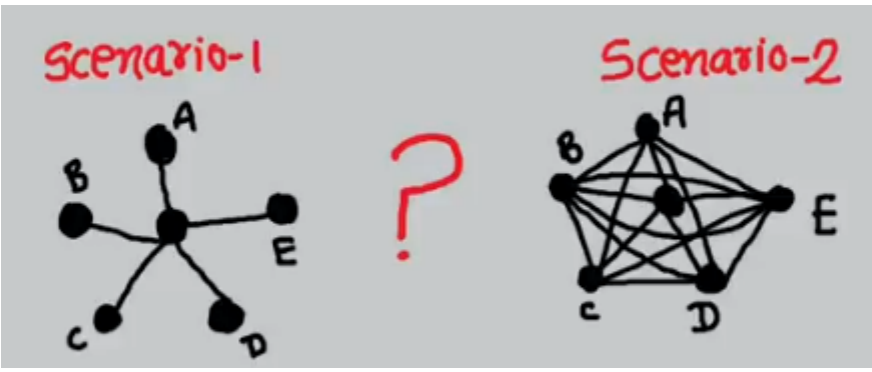
Eventually one can expect that, those become friends each other.. then .. 

.

- The structure is called a **Triad**, and the friendship that happens in this triad, making it a triangle -- this phenomenon is called the **Triadic closure**.
  - See, its Traid, becoems closed -- so Triadic Closure. 

Clustering Coefficient

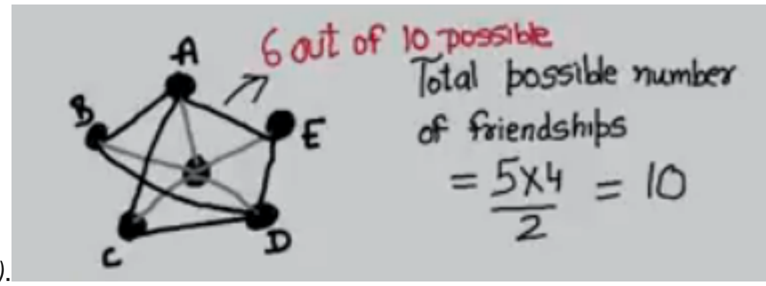
\begin{align} \textbf{Clustering coefficient} = \frac{\textit{Existing friendships}}{\textit{Total possible friendships that can exist}} \end{align}
Consider 5 friends in 2 scenarios.. firstly, all of them not know them-selves, second all know themselves. Which one do you prefer *(to be the center-node in the graph)*..??



-- mostly second one. Ofcourse, in some scenarios, where the 1st might be favourable.

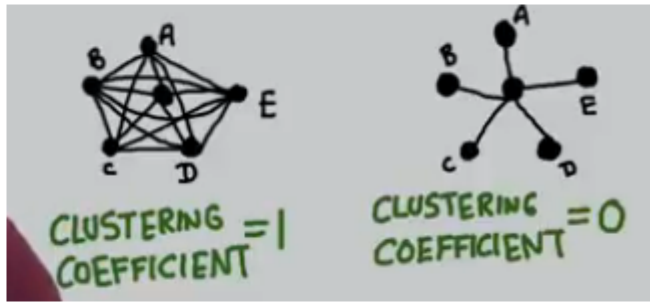
Let's try to quantify this matematically..!! there must be something in between these.. like some not being not-known to each...right..

> i.e.,.. Say you are friend with 5 people and there are some friendships between them selves. Now how many possible friendships can happen between these.. In 5 people, 4 chooses to be friends... how many ways one can do that..? 5 friends in 4 different ways: so $5 \cdot 4=20$ -- as we counted the SINGLE friendship TWICE(as from either-side), scale-down by half gives $ \frac {5 \cdot 4}{2}=\frac {20}{2}=10$.

- In the example, there are 6 friendships *(as represented in the figure, consider friendships among themselves, not including you[the center-node])*. 

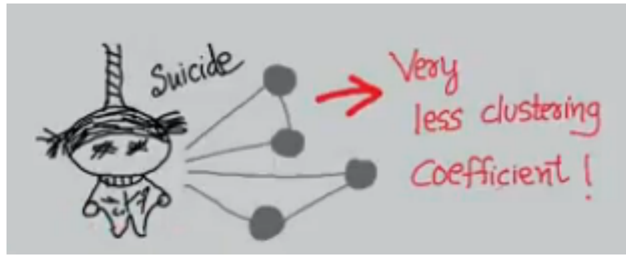
- The strength of friendships depends on the *friend ships between them*. The fraction $\frac{6}{10}$ denotes the strength of friendships with these 5 friends -- formally called as **Clustering Coefficient**.i.e.,

> In ${Nr}$ place the friendships that are existing between them, in ${Dr}$, place the total no. of friendships that can exist between them.

If it is 1, all know each other, if 0, none of them know each other. 

. In between, tells the strongness... nearer to 1, stronger, nearer to 0, weaker..

Why this is important..??

> It is observed that, people who've committed suicides, had very less clustering co-efficient. 

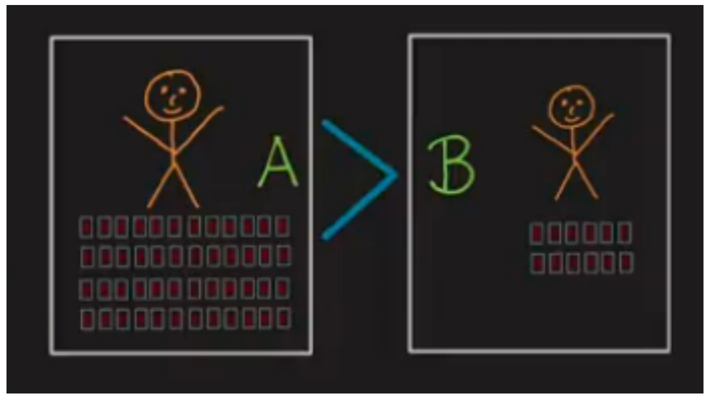
> .
> Its not that, those people had less friends, they had, but of different circles. Whenever they need to meet, they'll meet one and comes back.. not together as all those aren't known to themselves.

and in similar cases

Neighborhood Overlap

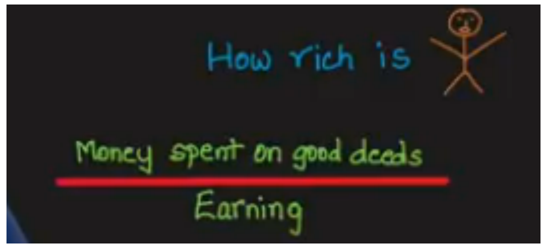
Term may seem bit complex, let's go with the right analogy...
How to judge a person..?? Is it based on.. *How much he earns..??*

- i.e., if A earns > than B, then A is richer than B.

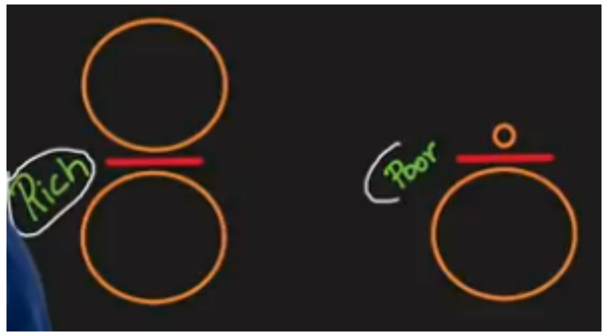


Let's try redifining it.. by ot looking at the bank balance.. instead.. as

- In Dr, the earnings of person
- In Nr, to where the money is spent on good deeds

> 
>
> Then, if he spends ofr noble-causes, then I call as richest.

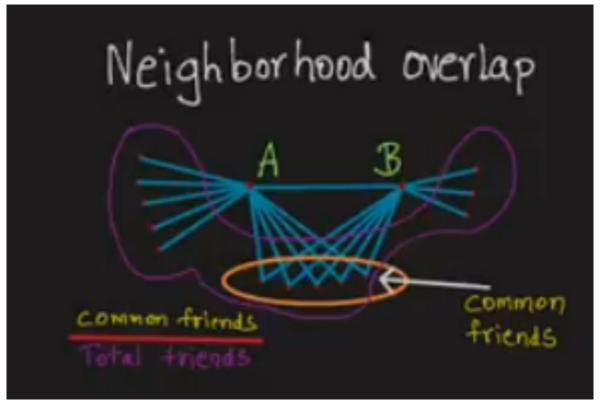
Now see.. the richness is got defined as.. **The proportion of money how well the money is spent divided by how much one ears**.
Why used fraction here....??
If the Dr is large and Nr is small - that gives a very less value *(closer to 0)* -- then I don't call as rich, if equal, then it would be closer to 1, then can call as rich.



Similarly.. **The strength of a friendship** can be defined as follows.... See the no.of common friends they had -- and designate it as their strength

or..

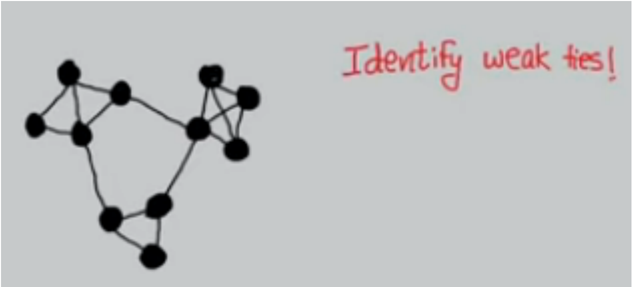
Look, what fraction of friends are common in between.

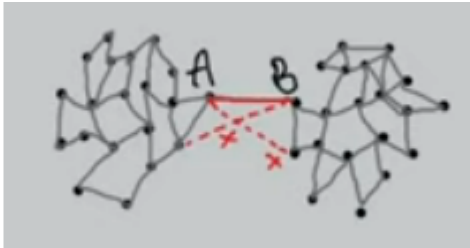


i.e., $$\textbf{Neighbourhood Overlap} = \frac{\text{# common friends}} {\text{# total friends}}$$. athematically speaking.. its.. $$\text{Neighbourhood Overlap} = \frac{A \cap B} {A \cup B}$$ If it is $\text{close to} 1 \text{Neighborhood Overlap is MAXimum}$, if $\text{close to} \text{Neighborhood Overlap is MINimum}$

Module-4: Structure of Weak ties, bridges and local bridges

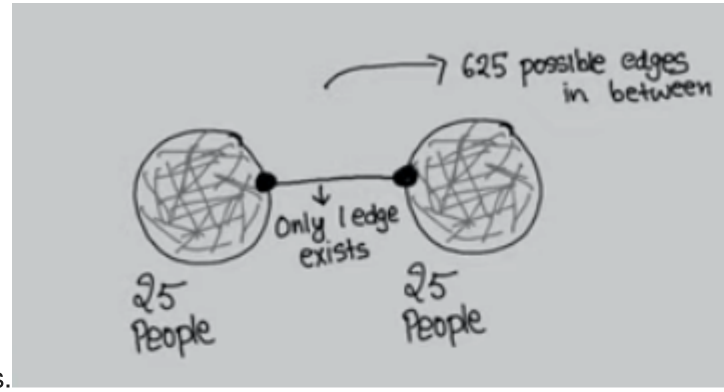
Assume you are given a graph G, and asked to tell **What are the weak ties?**, Can you tell or can't?

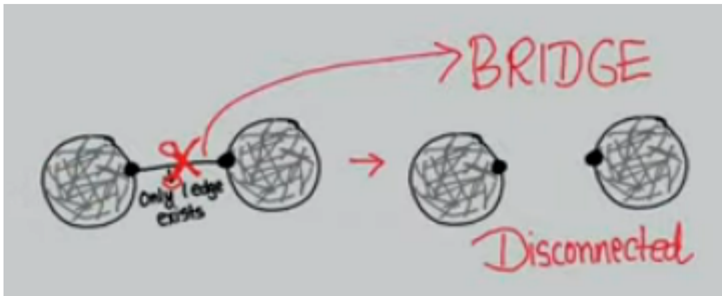


Say you are given this graph.. the one *(Recollect the graph of week-1: Connectedness..)*.. 

A and B are from two different islands, and those are friends, and they don't have any common friends in between..
Then by definition it is the *Weak tie..* -- May say that, they can also be good friends.. -- this isn't possible for the reason:

> If they both are good friends, then B should have more friends than this and same for A too (because of triadic closure)

So, given that, triadic closure is not there-- is itself and indication that, it's a Weak tie. So in this. 

-- that being only 1 edge is not possible in real life. So 

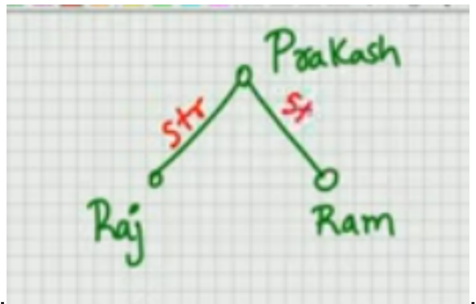
, removal of such edge, the graph becomes disconnected. It is called a **Bridge.**

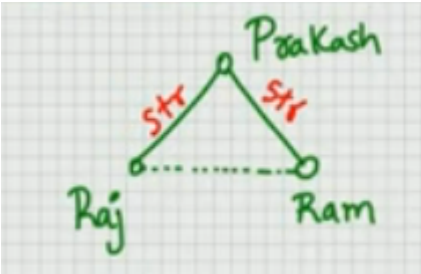
> A and B do not have any common friends, nor there is a path from A to B other than A to B.

Local Bridge *("Weak bridge" to remember intuitively)*
[DEF]

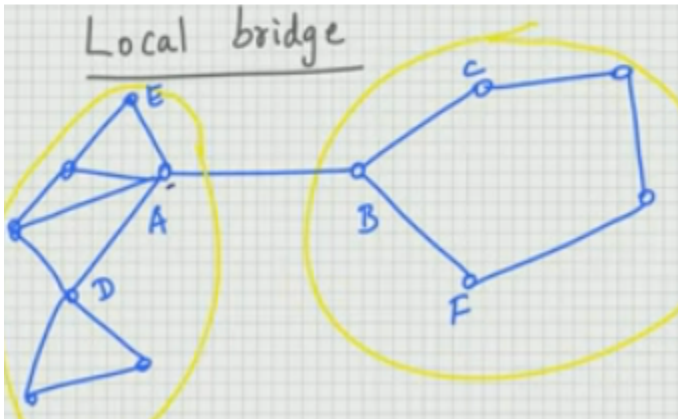
> Its a bridge(an edge), without having any triads on either side of the vertices.

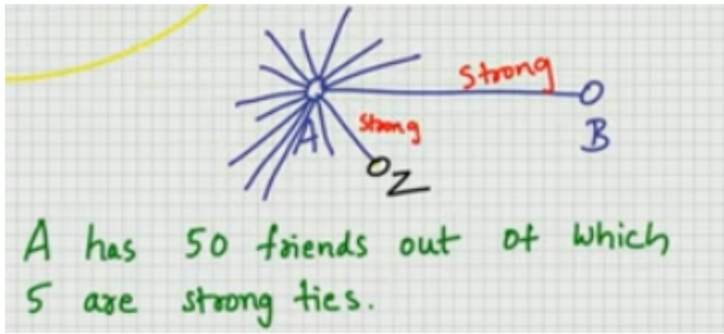
Strong Triadic Closure property

Assume a friendship like..

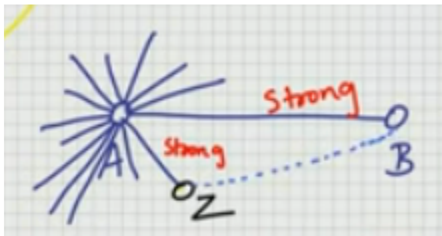
.. Its intuitive that, Even *Raj* and *Ram* becomes friends (May or may not be the strong tie, but a tie). So, this results in the triadic closure due to the strong tie 

-- (dotted line to denote, some tie) ---- which forms as **Strong triadic closure property**. pretty straight forward right...!!

Now (say) in the graph of 

if A has 50 friends 

.. and some of them are strong ties.

based of Strong triadic closure property, .. there exists some tie between A and B 

.

This results in: A and B not being in local bridge -- why, now there exists a common friend for both, thus forms a triadic closure. *(Even as per definition of local bridge, presence of a triad violates it to be a local bridge).*

To conclude..

> A local bridge is mostly a weak tie.

If there exists any triads *(between A and B)* due to strong ties of A, *(as like in above: via "Z")*, then that violates to be a local bridge. *(by definition and intuition).*

Module-5: **Validation of Granovetter's experiment weak tie using Cellphone data**

In the time of 1960s (the years in which Granovetter conducted his experiments on weak ties), there wasn't much data that time.
Now, the fact which is going to be revealed is **The local bridges turnout to be weak ties** which Granovetter couldn't show back then -- not clear.. read the below conducted experiment and read this again..

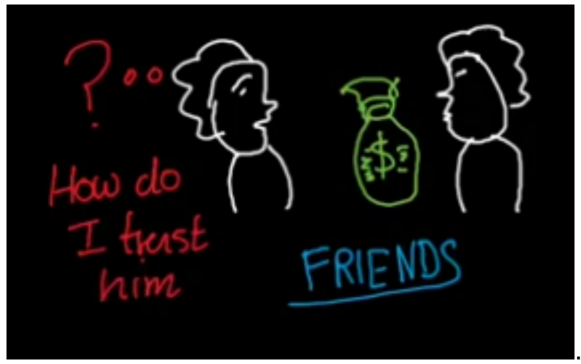
Now, due to this era, a lot of data exists. Im the year of 2007, there conducted an experiment using *Cell phone's data*, which is done s follows:

> - An edge is put between two people if they had a call. `![](resources/CellPhoneDataIntro.png)` . The also recorded the *amount of time* they talk for a period of 18 months(4+1/2 months), and the observed the big network. (Its mostly 85% connected).
>
> A proxy definition for a weak bridge is a *Neighborhood overlap* -- if lower: local bridge, if higher: Not a local bridge _(Here sir, termed reversly, but in next, said in this form.)_
>
> - High neighnorhood means, longer durations of call and vice versa for less neighborhood. `![](resources/CellPhoneData1.png)` i.e., `![](resources/CellPhoneData2.png)`.

- In the graph structure, edges that corresponds to smaller neighborhood overlap (that means, edges that are close to be the local bridges) -- we observe that, **cell phone talking duration is lesser** -- which means a weak tie. --- think, intuitive right..!!

Module-6: Embeddedness

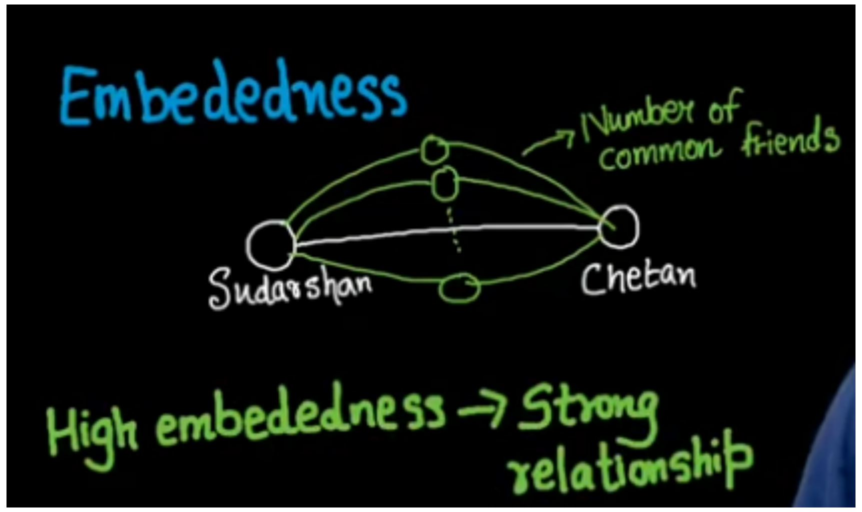
Imagine a situation.. (Say) on a street.

> - You've met a person, and talked.... and soom became friends. Now if he suddenly asks for money, what's your immediatie reaction?
> 
>
> ... Who is this person, we just became friends, and doesn't know really well -- how can I trust him..?? _(May be its a genuine request, but you don't)_
> - What if that person you've met turned out to be an old friend. Then you would have given him, because of duration which built trust.
> 
>
> -- there is still 1 more reason, which gets revealed in further discussion of lecture --- its the

Embeddedness

It can be defined as follows..

> Embeddedness between two friends is defined as the no. of common friends in betwee. *(Yes..!! We sae the siimilar ones in the past like neighborood overlap... but this is most pretty well studied in social-sciences literature).*



Where they observed that **If there is high embeddedness -- its a strong relationship**.

In the above example, (for the first person), the embeddedness(#common friends) was zero -- ie., no common friends.
Now, imagine that, suddenly in talk, he reminds your one of the good friend, and he is friend to hi -- then you probably would have trusted, and you may started to take the decision of giving money.

> See that, the trust increased by many folds due to the embeddedness(no. of common friends).

And say, if said 5 common friends, then you realize that he is not cheating you, and you feel some trust in him.

$$\text{Embeddedness} \propto \text{Trust}$$
let's see why this is true, with some examples..

> You wen to bank for a loan, then the bank manager expects for the guarantee.. If in case you become the defaulter, he can ask the other one.

Similar situation arises in friendship too..If had common friends in both, the chances of cheating are less -- why this is true..??
If in case the do, then the common friends come in between to resolve it. And atleast in fear of so many friends they may not cheat.
If had no common friends, then there are no one to pull in.

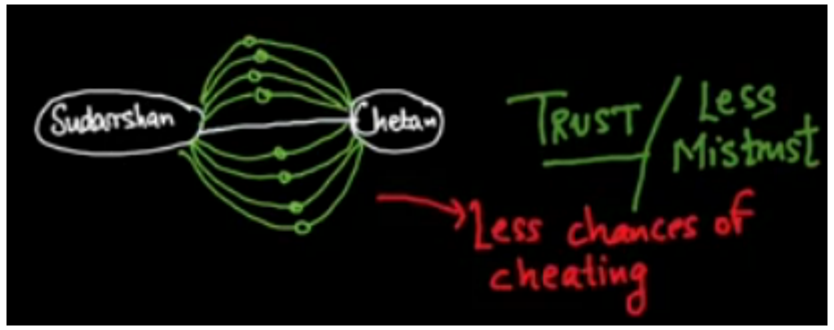
-- So all these psychological factors come in place.. having more common friends means more trust and vice-versa.

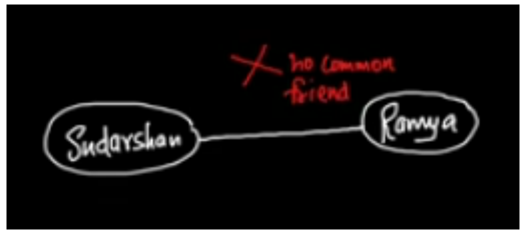
Do you think every single relationship should have *High embeddedness*? (**Embeddedness** means: Its not w.r.t. a person(a node), its a person's friend ship with another (an edge))

> Answer is: Yes and No.

Module-7: Structural Holes

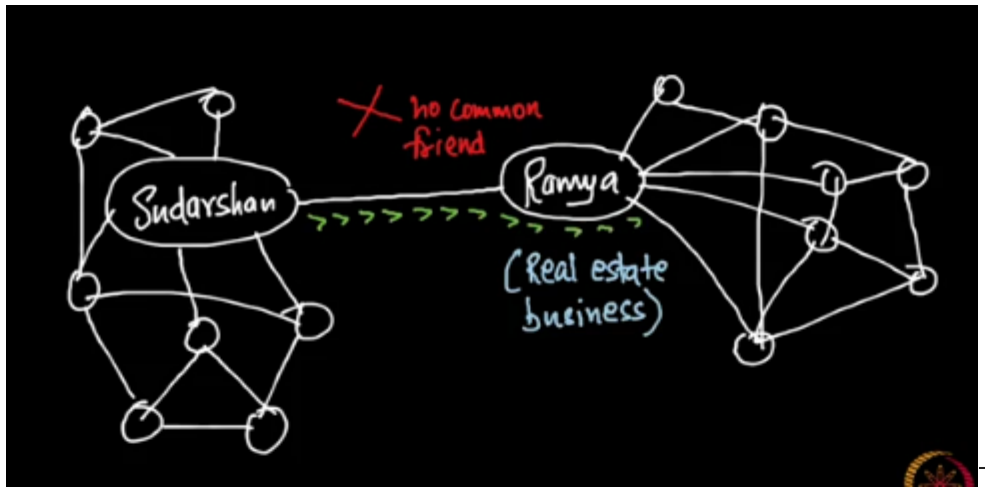
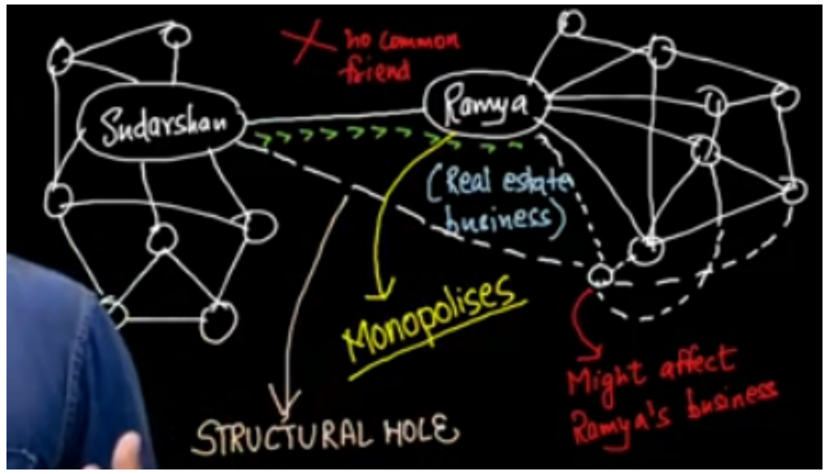
Say, there are two friends, and they have many common friends -- means more trust - means less chance of cheating..!



Now, you and other friend -- no common friends in between. 

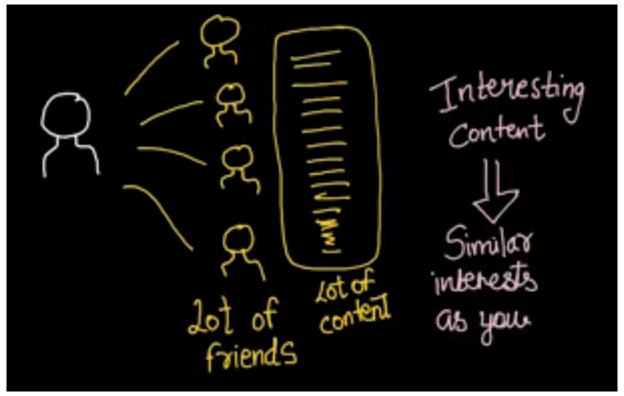
.. if present inany structures like Business, Money transaction... intuitively.. some danger may happend (may not be.. always true).

Will this be of any advantageous..?? -- Yes, Why..? look at the community..

> 
>
> Two communities are different. And if Sudarshan(sir) has to reach out to the other community, can only go via Ramya
> Now, if the sir would like to buy a house, then sir, need to knly consult Ramya, because she is the only known one to the community (Say that, sir doesn't know any other) -- here she ***Monopolises..*** due to Zero Embeddedness.
>
> Now, this becomes a huge advantage for Ramya, harnessing the fact that sir doesn't know anyone in that commnunity (If known, it affects her business)-- *this is similar to the Granovetter's Theory of Weak ties, she is the connection to a new world.*
> Have you noticed the **Structural hole** inthe graph..?? i.e., Any one to reach from L.H.S. community to R.H.S. need to pass only by long chain -- any friend of Sudarshan(sir) then to Ramya, then to other community.
> 
>

Module-8: **Social Captial**

Consider facebook, say it wants to become even more popular. So for a one to stick to to it..(ah..hh..!!), one's newsfeed should be filled with a lot of content and that should be interesting. When you get this..?? When had similar kind of interested people..or they had something excited to share..



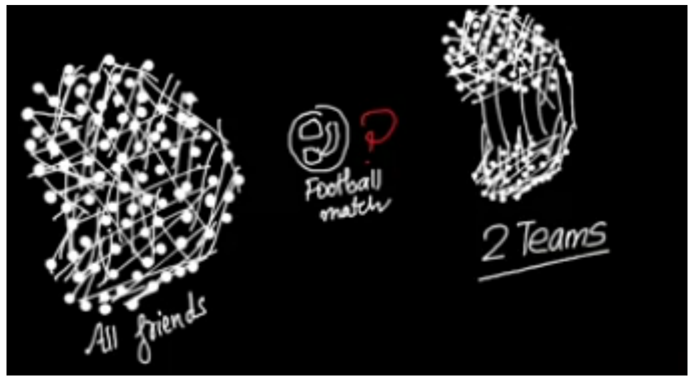
With this in mind, facebook wants to ensure that, you not only make friends randomly, but to those of similar interest. Every person can _maintain_ limited no. of friends, so they should be of similar interest.

Say, you have 100 nodes, and you need to sprinkle 500 edges. Where exactly will he put edges..?
This becomes Social Capital. You want to make a network and you get a atmost benefit from it.

. For the companies like FaceBook, its the lot of attention on their UI. **** Consider another example..
There are a set of 20 employees (and you are the CEO). There exists 2 pairs of people with some disturbances/fighting. As matter of resolving, you can try out like, give some money(budget) and tell them to go for a lunch and bond up.. or for a theme park, go and play and bond up.. -- so that you create captial out of it. (Here, you may expect like, all to be a good team)

Sometimes, the social network is itself the capital.
Nomination works well due to the complex structure in the social networks which is giving better productivity.

How to maximize it..?? -- is an open question. i.e., its not easy to tell that some protocol works...
As an example, there are bunch of friends (say 1000), all are friends with each other.. Is there any fun in playing football with groups of 2 teams..?? (If one person had two children and they are playin chess. Now to whom should the person cheer for..?? When loved equally, how can one cheer for one side.. to make other person lose) -- there has to be two communities right..(where a lot of love is within and not across)

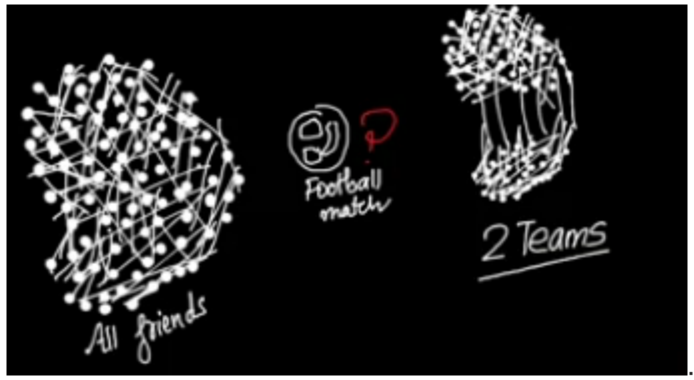


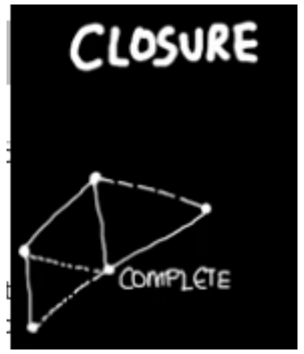
-- so that, can form two teams and play. -- So, 100% unity here is boring..

Here comes two terms:

Closure

A friend's friend should become friend.



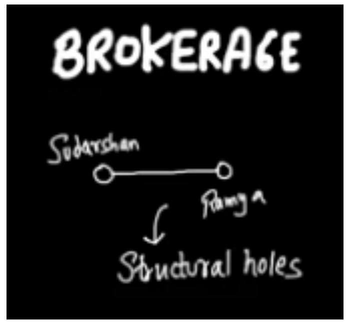
... This cascades, and every body is friend with another_ 

-- this way there is no fun

vs

Brokerage

There should be an edge (a sort of local bridge), like the above module's example. ---i.e,. There should be some strucutral holes too.



So, Closure is important and Brokerage is also important *to have the communication to other community*.

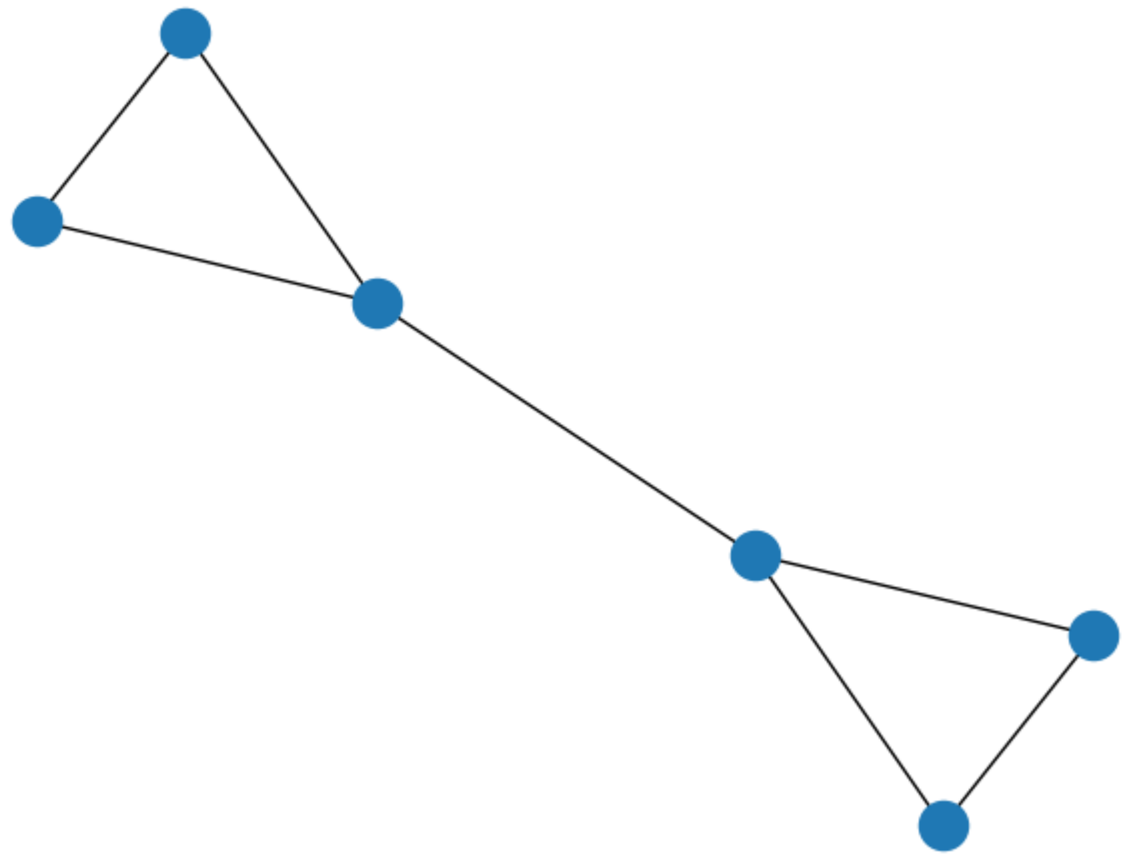
Module-9: **Finding Communities in a graph _(Brute Force Method)_**

In this module, we are going to divide the whole graph into two communities(groups) -- in a brute force manner, and check with of those divisions are best.
How do we check that.... as per the definition of communites..

The nodes which are part of a community, will have more no. of edges with other nodes in the same community and less with other community.

In [16]:
```
import networkx as nx
import matplotlib.pyplot as plt

nx.draw(nx.barbell_graph(3, 0))
plt.show()
```



In [17]:

```
nx.barbell_graph?
```

```
Signature: nx.barbell_graph(m1, m2, create_using=None)
Docstring:
Returns the Barbell Graph: two complete graphs connected by a path.

For $m1 > 1$ and $m2 >= 0$.

Two identical complete graphs $K_{m1}$ form the left and right bells,
and are connected by a path $P_{m2}$.

The `2*m1+m2`  nodes are numbered
    `0, ..., m1-1` for the left barbell,
    `m1, ..., m1+m2-1` for the path,
    and `m1+m2, ..., 2*m1+m2-1` for the right barbell.

The 3 subgraphs are joined via the edges `(m1-1, m1)` and
`(m1+m2-1, m1+m2)`. If `m2=0`, this is merely two complete
graphs joined together.

This graph is an extremal example in David Aldous
and Jim Fill's e-text on Random Walks on Graphs.
File:      ~/anaconda3/envs/Python-R/lib/python3.8/site-packages/networkx/generators/classic.py
Type:      function
```

In [11]:

```
G = nx.barbell_graph(4, 0)
nodes = G.nodes()
n = G.number_of_nodes()
```

Flow goes as: (Trying all possiblities is goes as below..)

> 1st community having 1 node, then 2nd -> n-1 and all its possiblites
>
> Then 1st having 2 nodes, then 2nd will have n-2.. so on.. -- *need to check all those possiblities*
>
> Here a P2N: If could find the nodes which fall in the 1st community, then the rest will be of 2nd community -- so, our aim goes like: Get all the combination of nodes that fall in the 1st community.

How do we get the combinations..??

> via the package `itertools` (look out some test below..)

In [3]:
```
import itertools
```

In [4]:
```
itertools.combinations?
```

```
Init signature: itertools.combinations(iterable, r)
Docstring:
Return successive r-length combinations of elements in the iterable.

combinations(range(4), 3) --> (0,1,2), (0,1,3), (0,2,3), (1,2,3)
Type:           type
Subclasses:
```

In [5]:
```
itertools.combinations([2, 3, 4, 5, 6, 7], 3)
```

Out[5]:
```
<itertools.combinations at 0x7fe7adeb3f90>
```

Tests

--tests--

It returns the object, which contains combinations as the tuples..

In [7]:
```
for combination in itertools.combinations([2, 3, 4, 5, 6, 7], 3):
    print(combination)
```

```
(2, 3, 4)
(2, 3, 5)
(2, 3, 6)
(2, 3, 7)
(2, 4, 5)
(2, 4, 6)
(2, 4, 7)
(2, 5, 6)
(2, 5, 7)
(2, 6, 7)
(3, 4, 5)
(3, 4, 6)
(3, 4, 7)
(3, 5, 6)
(3, 5, 7)
(3, 6, 7)
(4, 5, 6)
(4, 5, 7)
(4, 6, 7)
(5, 6, 7)
```

In [8]:
```
# To get as list..
for combination in itertools.combinations([2, 3, 4, 5, 6, 7], 3):
    print(list(combination))
```

```
[2, 3, 4]
[2, 3, 5]
[2, 3, 6]
[2, 3, 7]
[2, 4, 5]
[2, 4, 6]
[2, 4, 7]
[2, 5, 6]
[2, 5, 7]
[2, 6, 7]
[3, 4, 5]
[3, 4, 6]
[3, 4, 7]
[3, 5, 6]
[3, 5, 7]
[3, 6, 7]
[4, 5, 6]
[4, 5, 7]
[4, 6, 7]
[5, 6, 7]
```

---**tests**---

In [21]:
```
# lets get all the node combinations of the first community..
first_community = []
for i in range(1, n//2 + 1):  # No need of checking till """"n"""", 1st has k nodes, then 2nd will have n-k nodes, there is no need of checking for till n, as they will be the
    combination = [list(combi) for combi in itertools.combinations(nodes, i)]  # Getting all the possible combinations with "i" limit
    first_community.extend(combination)  # Used extend, so as to place the new combinations at end of list, not the list of combinations as one element(which happens when used
```

In [25]:
```
first_community
```

Out[25]:
```
[[0],
 [1],
 [2],
 [3],
 [4],
 [5],
 [6],
 [7],
 [0, 1],
 [0, 2],
 [0, 3],
 [0, 4],
 [0, 5],
 [0, 6],
 [0, 7],
 [1, 2],
 [1, 3],
 [1, 4],
 [1, 5],
 [1, 6],
 [1, 7],
 [2, 3],
 [2, 4],
 [2, 5],
 [2, 6],
```

```
    [2, 7],
    [3, 4],
    [3, 5],
    [3, 6],
    [3, 7],
    [4, 5],
    [4, 6],
    [4, 7],
    [5, 6],
    [5, 7],
    [6, 7],
    [0, 1, 2],
    [0, 1, 3],
    [0, 1, 4],
    [0, 1, 5],
    [0, 1, 6],
    [0, 1, 7],
    [0, 2, 3],
    [0, 2, 4],
    [0, 2, 5],
    [0, 2, 6],
    [0, 2, 7],
    [0, 3, 4],
    [0, 3, 5],
    [0, 3, 6],
    [0, 3, 7],
    [0, 4, 5],
    [0, 4, 6],
    [0, 4, 7],
    [0, 5, 6],
    [0, 5, 7],
    [0, 6, 7],
    [1, 2, 3],
    [1, 2, 4],
    [1, 2, 5],
    [1, 2, 6],
    [1, 2, 7],
    [1, 3, 4],
    [1, 3, 5],
    [1, 3, 6],
    [1, 3, 7],
    [1, 4, 5],
    [1, 4, 6],
    [1, 4, 7],
    [1, 5, 6],
    [1, 5, 7],
    [1, 6, 7],
    [2, 3, 4],
    [2, 3, 5],
    [2, 3, 6],
    [2, 3, 7],
    [2, 4, 5],
    [2, 4, 6],
    [2, 4, 7],
    [2, 5, 6],
    [2, 5, 7],
    [2, 6, 7],
    [3, 4, 5],
    [3, 4, 6],
    [3, 4, 7],
    [3, 5, 6],
    [3, 5, 7],
    [3, 6, 7],
```

```
 [4, 5, 6],
 [4, 5, 7],
 [4, 6, 7],
 [5, 6, 7],
 [0, 1, 2, 3],
 [0, 1, 2, 4],
 [0, 1, 2, 5],
 [0, 1, 2, 6],
 [0, 1, 2, 7],
 [0, 1, 3, 4],
 [0, 1, 3, 5],
 [0, 1, 3, 6],
 [0, 1, 3, 7],
 [0, 1, 4, 5],
 [0, 1, 4, 6],
 [0, 1, 4, 7],
 [0, 1, 5, 6],
 [0, 1, 5, 7],
 [0, 1, 6, 7],
 [0, 2, 3, 4],
 [0, 2, 3, 5],
 [0, 2, 3, 6],
 [0, 2, 3, 7],
 [0, 2, 4, 5],
 [0, 2, 4, 6],
 [0, 2, 4, 7],
 [0, 2, 5, 6],
 [0, 2, 5, 7],
 [0, 2, 6, 7],
 [0, 3, 4, 5],
 [0, 3, 4, 6],
 [0, 3, 4, 7],
 [0, 3, 5, 6],
 [0, 3, 5, 7],
 [0, 3, 6, 7],
 [0, 4, 5, 6],
 [0, 4, 5, 7],
 [0, 4, 6, 7],
 [0, 5, 6, 7],
 [1, 2, 3, 4],
 [1, 2, 3, 5],
 [1, 2, 3, 6],
 [1, 2, 3, 7],
 [1, 2, 4, 5],
 [1, 2, 4, 6],
 [1, 2, 4, 7],
 [1, 2, 5, 6],
 [1, 2, 5, 7],
 [1, 2, 6, 7],
 [1, 3, 4, 5],
 [1, 3, 4, 6],
 [1, 3, 4, 7],
 [1, 3, 5, 6],
 [1, 3, 5, 7],
 [1, 3, 6, 7],
 [1, 4, 5, 6],
 [1, 4, 5, 7],
 [1, 4, 6, 7],
 [1, 5, 6, 7],
 [2, 3, 4, 5],
 [2, 3, 4, 6],
 [2, 3, 4, 7],
 [2, 3, 5, 6],
```

```
 [2, 3, 5, 7],
 [2, 3, 6, 7],
 [2, 4, 5, 6],
 [2, 4, 5, 7],
 [2, 4, 6, 7],
 [2, 5, 6, 7],
 [3, 4, 5, 6],
 [3, 4, 5, 7],
 [3, 4, 6, 7],
 [3, 5, 6, 7],
 [4, 5, 6, 7]]
```

In [ ]:

```
# Now its the turn for second _community ... this is going to be simple.. as we need not find the combinations again, can just use those which are not in nodes,
second_community = []
# Subtraction can be easily applied via `set.. -- which inturn returns a set..
```

In [29]:

```
first_community[4]
```

Out[29]:

```
[4]
```

In [27]:

```
set(nodes) - set(first_community[2])  # See there will be no 4 in the result..
```

Out[27]:

```
{0, 1, 3, 4, 5, 6, 7}
```

In [30]:

```
# come let's do it over all the elements of the first_community...
for idx in  range(len(first_community)):
    second_community.append(list(set(nodes) - set(first_community[idx])))
```

Now we have the set of divisions ready..!! Its time to make a decision, which division among those, is best..

From the above addressed point..

Nodes which are part of 1 community, will have more no. of intra community edges (i.e., connections within the same community.)
and very less inter-community edges (i.e., connections across the community)

If the division is good, then the no. of **intra community edges should be HIGH** and **inter-community edges should be LESS**

In [31]:

```
# So, let's keep track of that ratio.. (described below..)
num_intra_edges1 = []
num_intra_edges2 = []
num_inter_edges = []
```

$$ \text{ratio} = \frac{\sum \text{ intra_community edges}}{\text{inter-community edges}} $$

In [32]:

```
# Let's keep track that every division..
ratio = []
```

Now we have set of nodes which fall in their respective communities.

Now how do we find the no. of edges amongst the nodes...??

> `subgraph()` function comes to handy.
> Which takes the list of nodes of a graph, and returns the sub-graph with that nodes and the edges that exist among them in the graph G.

In [ ]:

```

```

In [36]:

```
G.subgraph(first_community[10]).edges()
```

Out[36]:

```
EdgeView([(0, 3)])
```

In [37]:

```
# We need only the count of edges.. so let's do it for all the combinations of the nodes..
# for the first_community..
for idx in range(len(first_community)):
    num_intra_edges1.append(G.subgraph(first_community[idx]).number_of_edges()) # sotring no of intra-edges in the first_community (for each combination,,)

# for the second-community
for idx in range(len(second_community)):
    num_intra_edges2.append(G.subgraph(second_community[idx]).number_of_edges())
```

In [42]:

```
# Getting the number of inter edges...
total_edges = G.number_of_edges()
for idx in range(len(first_community)):
    num_inter_edges.append(total_edges - num_intra_edges1[i] - num_intra_edges2[i])
```

In [55]:

```
# Now we have values ready for finding the ratio..
for idx in range(len(first_community)):
    num_intra_edges = num_intra_edges1[i] + num_intra_edges2[i]
    ratio.append((num_intra_edges / num_inter_edges[i]))
```

In [58]:

```
# Its time to get the max_ratio.. which tells the max-benefit..
max_value = max(ratio)
max_index = ratio.index(max_value)  # We need the index at which there is max_division, so that we can pull out that max_division..
max_value, max_index
```

Out[58]:

```
(2.25, 0)
```

In [59]:

```
# Let's know the divisions..
first_community[max_index], second_community[max_index]
```

Out[59]:

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

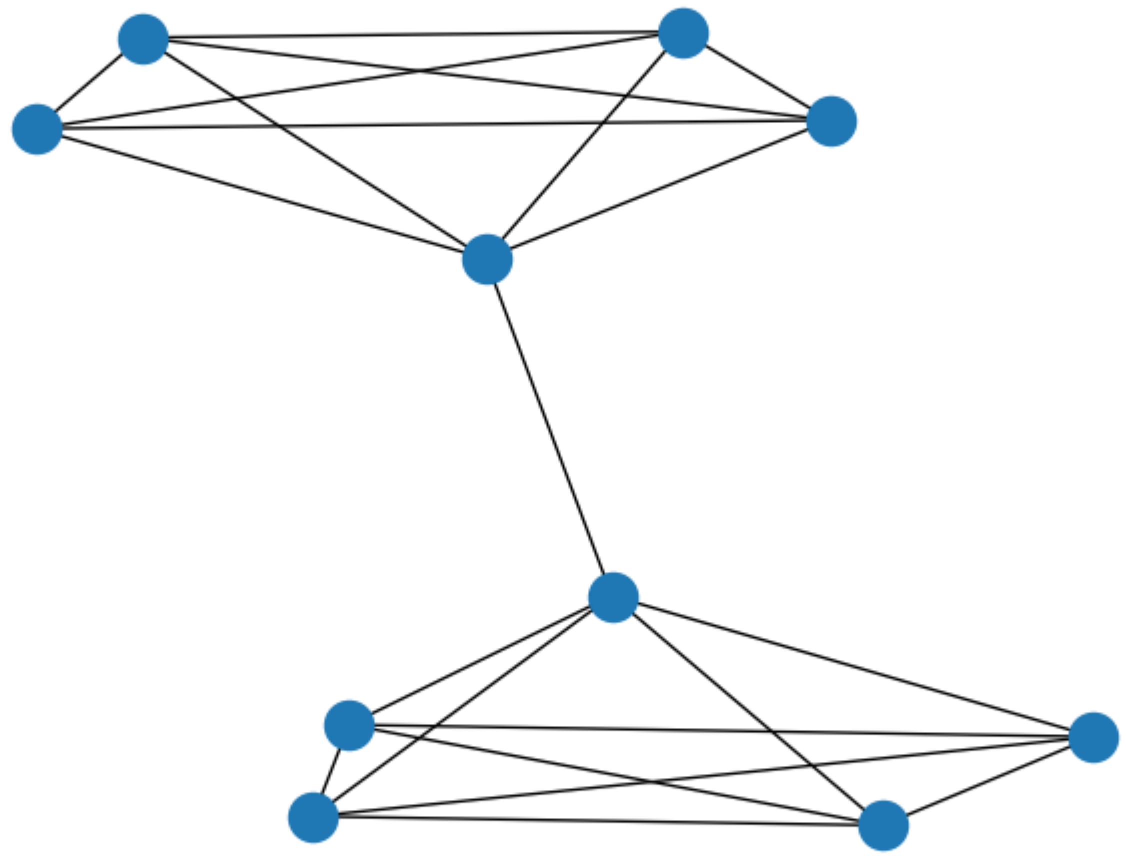
Now, all this packed in a script file named `FindingCommunitiesBruteForce.py` .. let's make some tests with it..

In [4]:

```
import FindingCommunitiesBruteForce as finder
import networkx as nx
```

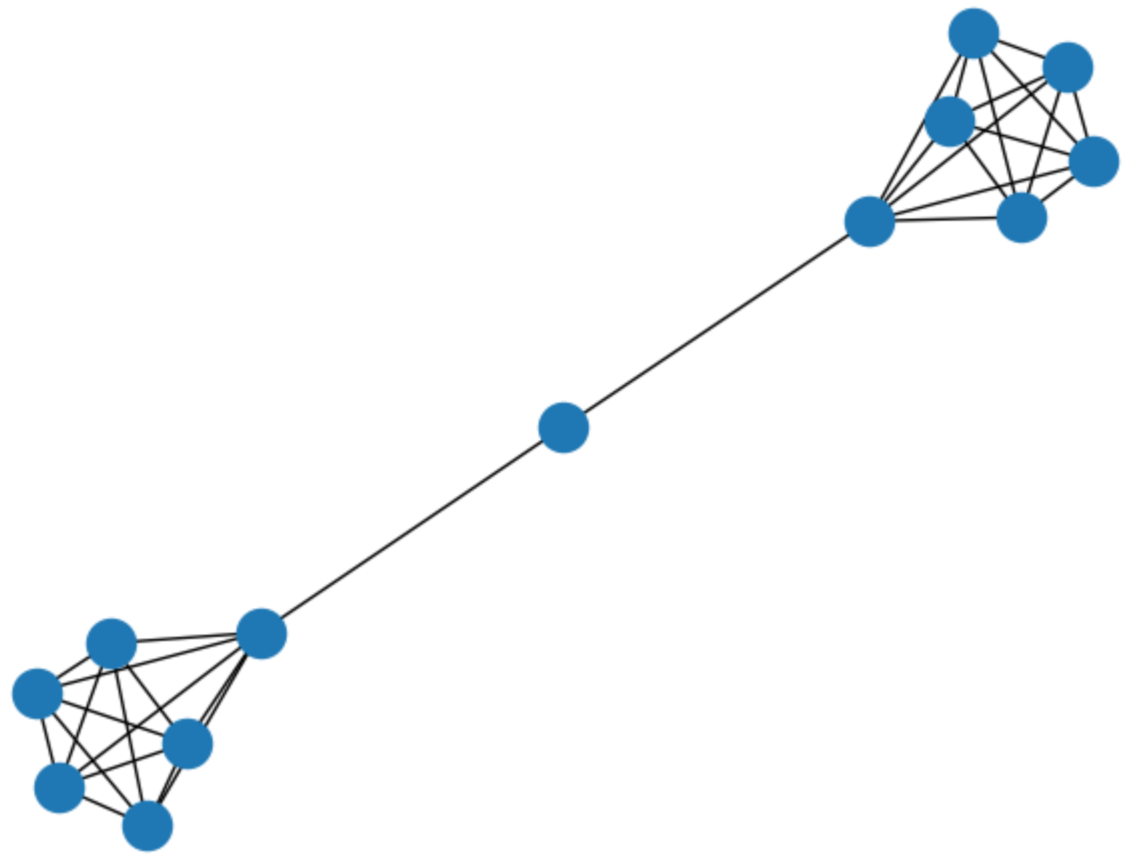
In [10]:

```
import matplotlib.pyplot as plt
G = nx.barbell_graph(5, 0)  # two sets of clusters of 5 nodes each, connected via 0 node in-between..
nx.draw(G)
plt.show()
```



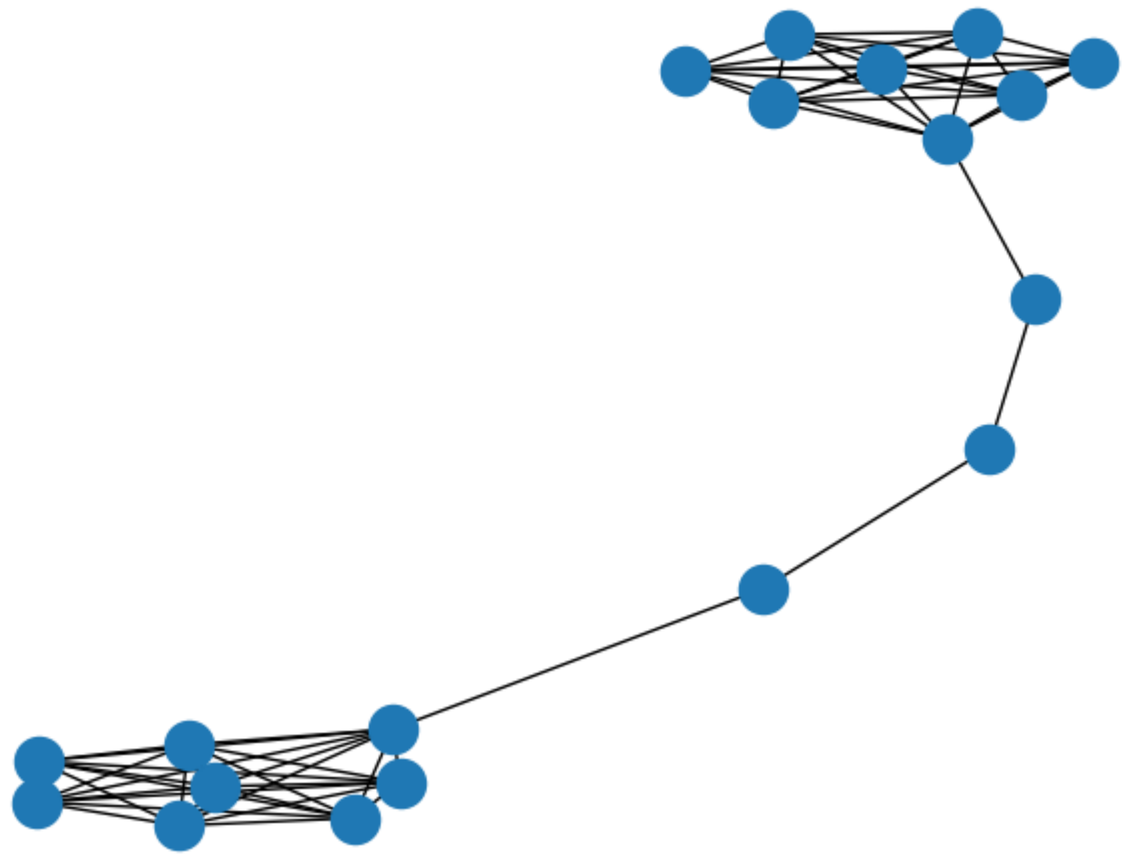
In [12]:

```
G = nx.barbell_graph(6, 1)  # two sets of clusters of 6 nodes each, connected via 1 node in-between..
nx.draw(G)
plt.show()
```



In [13]:

```
G = nx.barbell_graph(8, 3)  # two sets of clusters of 5 nodes each, connected via 0 node in-between..
nx.draw(G)
plt.show()
```



In [15]:
```
G = nx.barbell_graph(5, 0)  # two sets of clusters of 5 nodes each, connected via 0 node in-between..
finder.find_communities_via_bruteForce(G)
```

Out[15]:
```
([0], [1, 2, 3, 4, 5, 6, 7, 8, 9])
```

But madam got as ([0, 1,2, 3, 4], [8, 9, 5, 6, 7])

This bruteforce approach doesn't meet scaling -- if had graph with more no. of nodes, it takes huge time

There exists an algorithm named **Girvan Newmann Algorithm** let's explore it..

Module-11: Community Detection Using Girvan Newman Algorithm

This is based on the concept of **Edge betweenness** --

> The edges that are linking from one community to the other community, they tend to to have a *High value of betweenness* -- i.e., No. of shortest paths, that pass through these edges.
> So, if we could find the edge betweenness between these edges and we keep removing them, then it tends to become a community graph structure.

In [16]:
```
import networkx as nx
```

In [19]:
```
G = nx.barbell_graph(4,0)
```

In [20]:
```
# Let's find the no. of components in the given graph G
nx.connected_component_subgraphs(G) # It returns connected components as the subgraph.. So, if had 2 connected components, it returns 2 sub-graphs
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-20-ad1c14bb164c> in <module>
      1 # Let's find the no. of components in the given graph G
----> 2 nx.connected_component_subgraphs(G) # It returns connected components as the subgraph.. So, if had 2 connected components, it returns 2 sub-graphs

~/anaconda3/envs/Python-R/lib/python3.8/site-packages/networkx/__init__.py in __getattr__(name)
     49             "This message will be removed in NetworkX 3.0."
     50         )
---> 51     raise AttributeError(f"module {__name__} has no attribute {name}")
     52 
     53 

AttributeError: module networkx has no attribute connected_component_subgraphs
```

In [22]:

```
nx.connected_component_subgraphs?
```

```
Object `nx.connected_component_subgraphs` not found.
```

Oh..!! this is deprecated: See here -- *got re-directed from* Stack Overflow

Alternatives:

> **connected_component_subgraphs**(*G, copy=True*) [source]
>
> DEPRECATED: Use `(G.subgraph(c) for c in connected_components(G))`
>
> Or `(G.subgraph(c).copy() for c in connected_components(G))`
>
> Use `(G.subgraph(c) for c in connected_components(G))`
> Or `(G.subgraph(c).copy() for c in connected_components(G))`

In [26]:

```
components = [G.subgraph(component) for component in nx.connected_components(G)]
print("Connected components found are: ", len(components))
```

```
Connected components found are:  1
```

In [37]:

```
# Now we need to remove right.. so let's do that.. but wait..!! on which basis..?? -- based on betweenness.. So, lets first implement this..
def edge_to_be_removed(G):
    dic = nx.edge_betweenness_centrality(G) # It returns the dicitionary as: key being edge, and value being its centrality
    # But dictionary doesn't maintan order.. so let's keep in a list..
    list_of_tuples = list(dic.items())  # Returns as a tuple-format as: (key, value)
    # Now, lets sort it.. based on edge-betweenness
    list_of_tuples.sort(key = lambda x:x[1], reverse=True)  # need in DESC..
    # Return the edge which had highest edge-betweenness...
    return list_of_tuples[0][0]
# As sorted in DESC, first element will be a tuple which had highest-edge_betweenness, and get the edge in (edge, value)
```

In [38]:

```
# A test..
edge_to_be_removed(G)
```

Out[38]:

```
(3, 4)
```

In [ ]:
```
# Now, lets try removing the edge..
G.remove_edge(*edge_to_be_removed(G))  # will get as ((src, target)), but need (src, target) .. prefixing ""*""

# Now, find the no. of connected components.. again..
components = [G.subgraph(component) for component in nx.connected_components(G)]
print("Connected components found are: ", len(components))
```

We need to do this until we be with one connected component.. and stop as soon as we get >1 (say 2) component ...so, going for a loop..

In [52]:
```
components = [G.subgraph(component) for component in nx.connected_components(G)]
print("Connected components found are: ", len(components))

while len(components) == 1:  # Stop when connected components are != 1
    G.remove_edge(*edge_to_be_removed(G))  ## ((a, b)) -to-> (a, b)
    components = [G.subgraph(component) for component in nx.connected_components(G)]
    print("Connected components found are: ", len(components))
```

```
Connected components found are:  2
```

Putting the algorithm together...

In [53]:
```
def girvan_newman(G):
    # Finding the count of connected components..
    components = [G.subgraph(component) for component in nx.connected_components(G)]
    print("Connected components found are: ", len(components))

    while len(components) == 1:  # Stop when connected components are != 1
        G.remove_edge(*edge_to_be_removed(G))  ## ((a, b)) -to-> (a, b)
        components = [G.subgraph(component) for component in nx.connected_components(G)]
        print("Connected components found are: ", len(components))

    return components
```

In [54]:
```
# Let's test it..
G = nx.barbell_graph(5, 0)
components = girvan_newman(G)

# Let's get the nodes in each component..
for component in components:
    print(component.nodes())
```

```
Connected components found are:  1
Connected components found are:  2
[0, 1, 2, 3, 4]
[5, 6, 7, 8, 9]
```

In [66]:
```
# Let's check for Zachary karate club..
G = nx.karate_club_graph()
components = girvan_newman(G)

# Let's get the nodes in each component..
for component in components:
    print(component.nodes(), end=" --- length: ")
    print(len(component.nodes()))
```

```
Connected components found are:  1
Connected components found are:  1
Connected components found are:  1
```

```
Connected components found are:  1
Connected components found are:  1
Connected components found are:  1
Connected components found are:  1
Connected components found are:  1
Connected components found are:  1
Connected components found are:  1
Connected components found are:  1
Connected components found are:  2
[0, 1, 3, 4, 5, 6, 7, 10, 11, 12, 13, 16, 17, 19, 21] --- length: 15
[2, 8, 9, 14, 15, 18, 20, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33] --- length: 19
```

In [64]:

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

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-64-1e3ff2de76e8> in <module>
----> 1 nx.draw_networkx_labels(G)
      2 plt.show()

TypeError: draw_networkx_labels() missing 1 required positional argument: 'pos'
```

In [61]:

```
nx.draw?
```

```
Signature: nx.draw(G, pos=None, ax=None, **kwds)
Docstring:
Draw the graph G with Matplotlib.

Draw the graph as a simple representation with no node
labels or edge labels and using the full Matplotlib figure area
and no axis labels by default.  See draw_networkx() for more
full-featured drawing that allows title, axis labels etc.

Parameters
----------
G : graph
    A networkx graph

pos : dictionary, optional
    A dictionary with nodes as keys and positions as values.
    If not specified a spring layout positioning will be computed.
    See :py:mod:`networkx.drawing.layout` for functions that
    compute node positions.

ax : Matplotlib Axes object, optional
    Draw the graph in specified Matplotlib axes.

kwds : optional keywords
    See networkx.draw_networkx() for a description of optional keywords.

Examples
--------
>>> G = nx.dodecahedral_graph()
>>> nx.draw(G)
>>> nx.draw(G, pos=nx.spring_layout(G))  # use spring layout

See Also
--------
draw_networkx
draw_networkx_nodes
draw_networkx_edges
```

```
draw_networkx_labels
draw_networkx_edge_labels

Notes
-----
This function has the same name as pylab.draw and pyplot.draw
so beware when using `from networkx import *`

since you might overwrite the pylab.draw function.

With pyplot use

>>> import matplotlib.pyplot as plt
>>> G = nx.dodecahedral_graph()
>>> nx.draw(G)  # networkx draw()
>>> plt.draw()  # pyplot draw()

Also see the NetworkX drawing examples at
https://networkx.org/documentation/latest/auto_examples/index.html
File:      ~/anaconda3/envs/Python-R/lib/python3.8/site-packages/networkx/drawing/nx_pylab.py
Type:      function
```

In [ ]:

```

```

In [39]:

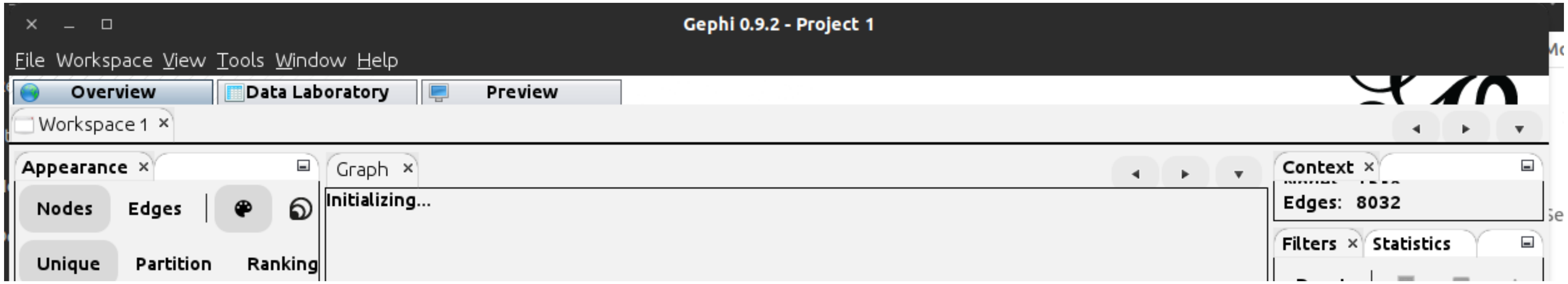
```
G.edges()
```

Out[39]:

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

Module-11: Visualising Communities using Gephi

Previously gephi worked for week-2 lectures, but now getting an error:



Errors that occured in 1st run of week-2.. and solutions..

- doing `export LIBGL_SOFTWARE_ALWAYS=1` and
- java version should also be 8.

from this reference.

this is even noted in the text file and saved in the Gephi's bin folder.

Solve this error...

End

Rough work..

$ \(f(x) = \sum_{i=0}^{n} \frac{a_i}{1+x} \)$

[f(x) = \sum_{i=0}^{n} \frac{a_i}{1+x}]$

In [3]:

```
%%latex
$ \( f(x) = \sum_{i=0}^{n} \frac{a_i}{1+x} \)$

\[ f(x) = \sum_{i=0}^{n} \frac{a_i}{1+x} \]
```

$ \(f(x) = \sum_{i=0}^{n} \frac{a_i}{1+x} \)$ \[f(x) = \sum_{i=0}^{n} \frac{a_i}{1+x} \]

\begin{align} H←120 + \frac{30(R−G)}{Vmax−Vmin} , if Vmax = B \end{align}