

NPTEL

NPTEL ONLINE CERTIFICATION COURSE

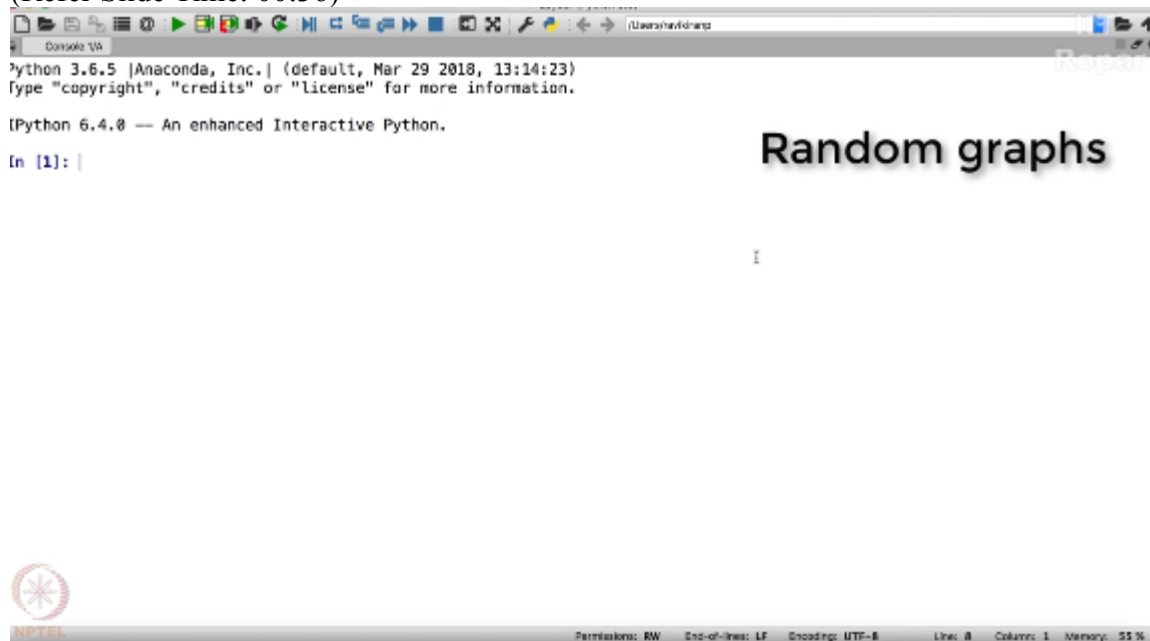
**Discrete Mathematics  
Graph Theory – 3 &  
Generating Functions**

**NetworkX - Random graphs**

**By  
Prof. S.R.S Iyengar  
Department of Computer Science  
IIT Ropar**

In the previous videos we have seen all the theoretical part of graph theory, we have seen several topics like Eulerian, Hamiltonian, Coloring, Planar graphs and so on, we have seen all the theory part of it, now we will be seeing how to implement it in Anaconda, the Spyder.

Now we will start with learning how to create a random graph,  
(Refer Slide Time: 00:36)



The screenshot shows a Jupyter Notebook interface. On the left, a console window displays the following text: "Python 3.6.5 [Anaconda, Inc.] (default, Mar 29 2018, 13:14:23) Type "copyright", "credits" or "license" for more information. [Python 6.4.0 -- An enhanced Interactive Python. In [1]: |". On the right, a slide titled "Random graphs" is visible, with a cursor pointing to the first line of text.

what do I mean by that? If I specify the number of vertices and the number of edges, a random graph should be generated, right, we will see how to do that, the first step is to import networkx as nx, now I have loaded networkx, what I am going to do is first create a random graph, let me show you how to do it, so I'll name the graph as G,  $G = nx.gnm$ , what do I mean by GNM? G stands for graph, NM stands for number of vertices and number of edges respectively, GNM random graph like this with underscore in middle random\_graph and in brackets do you see here what is shown? N,M, right, so N is the number of nodes, and M is the number of edges, so (Refer Slide Time: 01:42)

Type "copyright", "credits" or "license" for more information.

IIT  
Ropar

IPython 6.4.0 -- An enhanced Interactive Python.

In [1]: `import networkx as nx`

In [2]: `G=nx.gnm_random_graph(|`

```
Arguments
graph(n, n, seed=None, directed=False)
```

|



let me say some 10 vertices and some 15 edges I want, so if I close the bracket the graph has got loaded, the next would be to draw it, `nx.draw(G)`, do you see this graph here, the graph is disconnected.

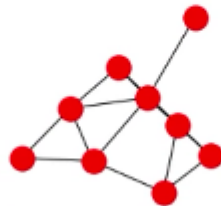
(Refer Slide Time: 02:04)

In [3]: `nx.draw(G)`

IIT  
Ropar



|



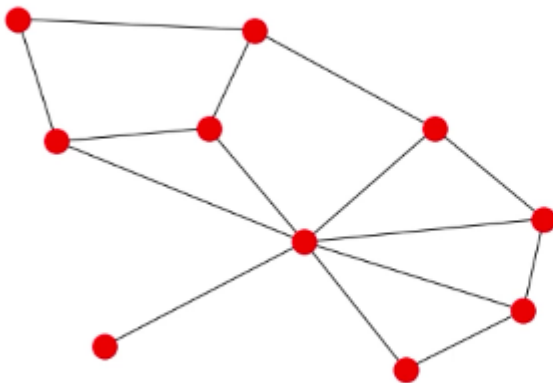
In [4]: |



Let us try once again and see if we get a connected graph, I'm again going to generate G itself here, and I'm going to give `nx.draw(G)` do you see that this time we have obtained a connected graph,

(Refer Slide Time: 02:22)

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



```
In [6]: |
```

I

let us check if the size of the graph is same as that we mentioned, yes, do you see that I had mentioned 15 here, and I have obtained 15, size stands for the number of edges.

Now we have seen how to create a random graph, by specifying the number of vertices and the number of edges, now here comes another cute aspect of graph theory, I can generate a random graph using the parameter probability, how do I do that? Instead of number of edges I specify the probability of an edge between 2 nodes, do not worry I'm not going to use any aspect of probability here, for those who do not know you can understand it this way, I take a coin and I toss it, in case I get a head I draw an edge between 2 vertices in case I get a tail I don't draw, I don't draw an edge between 2 vertices, and this holds true for all the pairs of vertices in the graph.

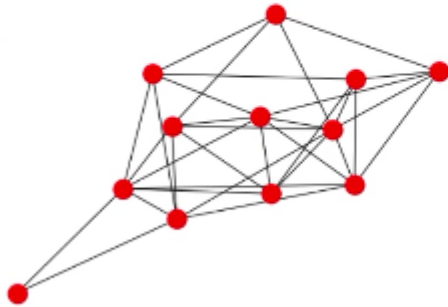
Now observe what I am going to do now, let me clear the screen and start a fresh, so I'm creating a new graph H here as `nx.gnp`, you see GNP stands for graph on N vertices and P stands for the probability, do not worry I am going to explain what are we going to do next, so random graph, now I'm going to specify the number of vertices first, right, let me say I'm going to draw a graph on let say 10 vertices or 12 vertices, and now among any 2 vertices of all this 12 vertices, if a head comes I'm going to draw an edge, if a tail comes I'm not going to put an edge, what does that mean? It means between any two vertices there is a 50% chance of putting an edge either you put it or you don't put it depending on whether you get a head or a tail, right, so the chance is 50% or 0.5 to be precise.

Now if I mention like 0.5 here and close the bracket you see the graph has got created, let us see I'm going to draw the graph `nx.draw`, `nx.draw(H)` you see we have obtained the graph here, (Refer Slide Time: 05:30)

```
TypeError: draw() missing 1 required positional argument: 'G'
```

```
In [10]:
```

```
In [10]: nx.draw(H)
```

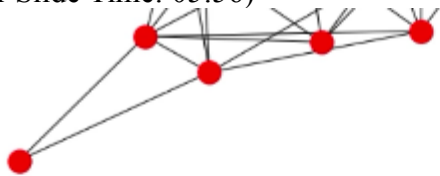


```
In [11]:
```

Permissions: RW End-of-lines: LF Encoding: U

it's a connected graph between 12 vertices with 0.5% chance of having an edge between any 2 vertices, what do we mean by that? Let us check the number of edges in this graph, so I have to give H.size you see 33 it says, there are 33 edges in this graph.

(Refer Slide Time: 05:56)



```
In [11]: H.size()
```

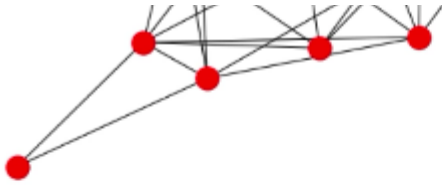
```
Out[11]: 33
```

```
In [12]: |
```



Now if in case it's a complete graph on 12 vertices, what should happen? What is the total number of edges possible in a graph, you must be knowing that the total number of edges possible in a graph is  $N$  choose 2, right,  $N$  choose 2 is here it is going to be 12 choose 2 because  $N$  is 12, so 12 choose 2 happens to be 12 into, let me calculate that, 12 x 11 and 132 divided by 2 I want, so it is 66, right,

(Refer Slide Time: 06:52)



```
In [11]: H.size()
Out[11]: 33
```

```
In [12]: 12*11
Out[12]: 132
```

```
In [13]: 132/2
Out[13]: 66.0
```

```
In [14]:
```



I

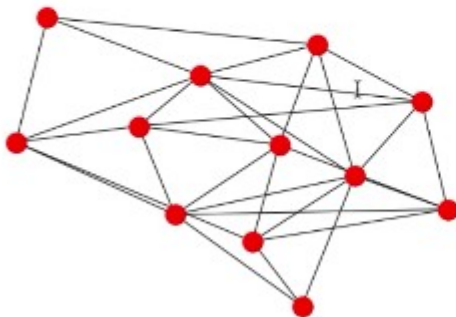
so if it's a complete graph on 12 vertices I will have 66 edges, but since there is a 50% chance of having an edge between any 2 vertices intuitively speaking you feel that 33 edges should be present in the graph, right, which is what we have obtained here, but trust me this is not going to happen always, we are not very sure if we'll get 33 edges or not.

Now if you've not understood what I have done, it is perfectly fine I'm going to repeat the entire process again, let me again take a  $H = nx.gnp\_random\_graph$ , so random graph will get created, I'm again going to take 12 vertices and with a 50% chance of having edges between any 2 nodes,  $nx.draw(H)$  so you see we have some edges  
(Refer Slide Time: 08:05)



```
In [19]: H=nx.gnp_random_graph(12,0.5)
```

```
In [20]: nx.draw(H)
```



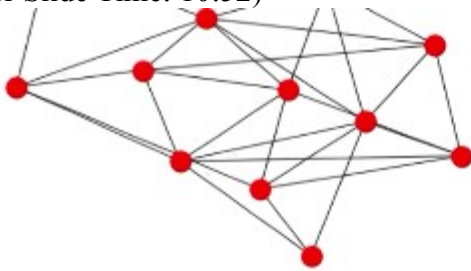
```
In [21]:
```

here let me check the size of this graph  $H.size$  is 31, did you observe last time we had got 33 edges that was perfect, right, because there was 50% chance, but I'm sure if you are trying it on

your machine you might really not end up having 33, because it's a random graph you might get some other graph and I am getting some other graph here, right, it is same as telling if I toss a coin say 10 times, it's not always obvious or it is not sure that we will get 5 times head, and 5 times tail, it might happen 7 and 3, or 8 and 2, or 9 and 1, or 10 and 0, we don't know, but I'm going to repeat this experiment several times, how do I do that? We'll see, I've drawn the graph here but each time I'm not going to draw the graph, how I'm going to do that? Let us see.

Let me name total, you just observe what I am going to do from now, it's not really essential that you understand each point, you can observe and at the end of this, end of next 5 to 10 minutes you will be able to grasp things yourself, so in order to tell that after I repeat this experiment several times I will end up having 33 edges in the graph, I'm going to run a short piece of court, so now I have given this total edges, now for I in range let me say 10, please be observing `H = nx.gnp_random_graph` on 12 nodes and 0.5 chance, `M = size` or `H.size`, `total_edges.append(m)`,

(Refer Slide Time: 10:52)



IT  
Repair

```
In [21]: H.size()
Out[21]: 31

In [22]: total_edges=[]    I

In [23]: for i in range(10):
...:     H=nx.gnp_random_graph(12,0.5)
...:     m=H.size()
...:     total_edges.append(m)
```

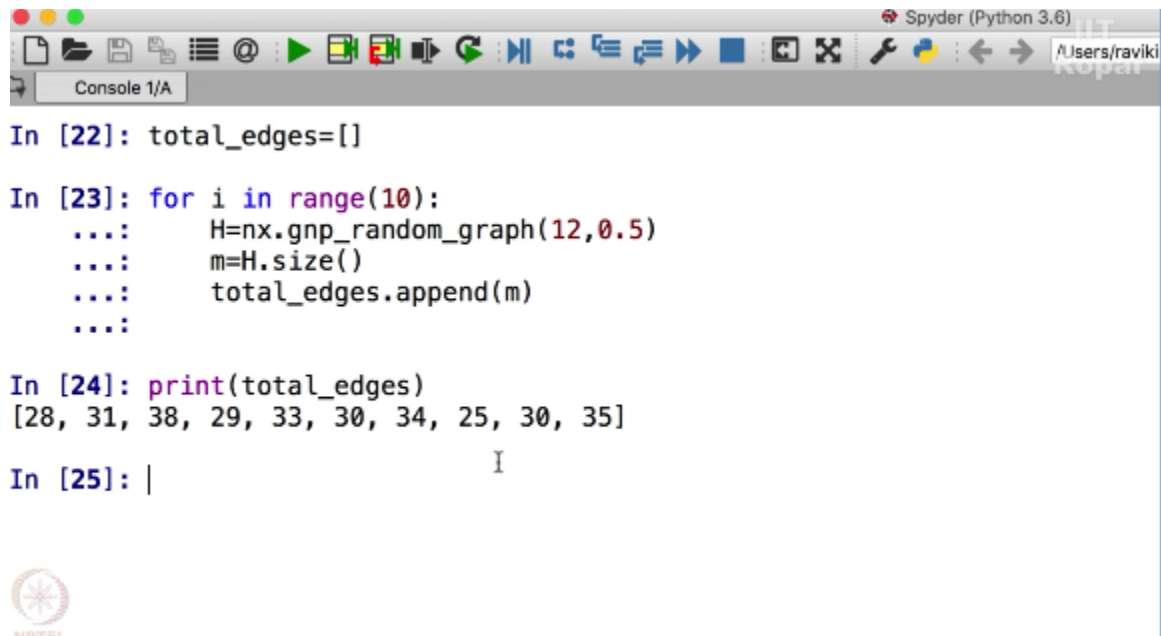


Permissions: RW End-of-lines: LF

now let me explain what I have done, you see I have created total edges here, I want to get the output in terms of total number of edges, now for I in range 10, what does this mean? I'm repeating this experiment 10 times, right, this entire thing which is inside this for will be repeated 10 times, what is it which is getting repeated? I've created a graph with 12 vertices, and probability 0.5, right, now once the graph is created I want to check the size, now once the size is created I'm going to append it, what is it? What do we mean by appending? Appending is not but keeping several things one the side other, right, append M means all the sizes after repeating the 10 times you will be getting 10 graphs, so the size of each graphs will just be written side by side, you can see that now.

I'm going to print total edges, I hope now you are able to understand why I did total edges, you see the 10 edges I have been just written one the side other,

(Refer Slide Time: 12:15)



```
Spyder (Python 3.6)
/Users/raviki
Console 1/A
In [22]: total_edges=[]
In [23]: for i in range(10):
...:     H=nx.gnp_random_graph(12,0.5)
...:     m=H.size()
...:     total_edges.append(m)
...:
In [24]: print(total_edges)
[28, 31, 38, 29, 33, 30, 34, 25, 30, 35]
In [25]: |
```

28, 31, 38, 29, 33, 30, 34, 25, 30, and 35, do you see we haven't always got 33 edges in the graph, so it's not very intuitive, right, now I want to take the average of all this, why? Because I say that when I repeat this experiment several times and when I take the average, I'm sure of getting the 33 edges which I should be getting in the graph, how I am going to do that? Import numpy as np, so for such operations as I had mentioned in the earlier video we will be using this library called numpy.

Now I'm going to take the average of all these edges, and I'm going to print that, print np.average of total edges, we want the average of the total number of edges, right which I've been appended here, how do we take the average of it? Np., we use the library numpy and we take the average, do you see we have not again got 33 as the average, it says 31.3, how do we obtain 33 then? Well, if I repeat this experiment for say 100 times I'm sure of getting 33, let us check. I'm going to change for I in range 10 to let say 100, rest everything remains the same, I wanted to create the graph H, take the size, appended, right, now I'm going to print the total edges, do you see the 100 edges here which I have got appended, (Refer Slide Time: 14:17)



```

In [27]: print(np.average(total_edges))
31.3

In [28]: for i in range(100):
...:     H=nx.gnp_random_graph(12,0.5)
...:     n=H.size()
...:     total_edges.append(m)
...:

In [29]: print(total_edges)
[28, 31, 38, 29, 33, 38, 34, 25, 38, 35, 24, 31, 42, 33, 31, 32, 37, 29, 26, 29, 29, 32, 28, 32, 32, 36, 42, 35, 34, 33, 31,
34, 36, 32, 28, 28, 29, 25, 25, 28, 37, 31, 29, 36, 33, 32, 35, 30, 38, 29, 32, 35, 28, 35, 38, 30, 33, 31, 32, 21, 25, 34,
31, 35, 33, 32, 31, 34, 38, 37, 28, 41, 31, 43, 38, 33, 35, 31, 33, 35, 35, 29, 36, 34, 32, 31, 26, 31, 38, 48, 38, 36, 34,
38, 37, 34, 36, 38, 29, 41, 29, 33, 36, 29, 36, 35, 29, 34, 42, 37]

In [30]: |

```

so now I'm sure obviously, manually calculating the average is a very tedious and cumbersome job and hence we are going to take, we need not import numpy again, clear, so np.average of total edges, do you see this? 32.45454545 it goes on, right, it's almost close to 33, but still we have not got 33 as the exact number of edges.

Now let us again repeat it for say 1000 times, now I'm going to repeat this experiment for 1000 times and I'm going to print the total edges, you see the total 1000 edges here is difficult to even redetect one shot right, (Refer Slide Time: 15:23)

```

41, 38, 31, 35, 31, 36, 35, 28, 33, 32, 36, 31, 38, 33, 39, 33, 39, 34, 27, 33, 38, 24, 37, 31, 31, 33, 29, 34, 35, 32, 26,
27, 31, 35, 35, 39, 35, 34, 38, 33, 29, 36, 34, 38, 30, 35, 40, 29, 33, 34, 32, 34, 31, 32, 32, 31, 25, 35, 36, 33, 36, 36,
42, 36, 34, 33, 29, 37, 38, 34, 40, 38, 32, 34, 28, 33, 37, 36, 40, 38, 38, 35, 31, 32, 34, 32, 34, 45, 32, 32, 32, 29, 36,
33, 31, 36, 39, 41, 31, 29, 34, 30, 36, 38, 35, 28, 27, 34, 29, 27, 28, 30, 36, 31, 33, 32, 34, 29, 33, 26, 27, 27, 35, 43,
28, 35, 36, 31, 38, 36, 33, 37, 33, 39, 32, 32, 37, 35, 44, 31, 28, 32, 33, 38, 31, 34, 26, 33, 33, 32, 27, 32, 38, 29, 28,
37, 34, 33, 32, 38, 35, 38, 36, 26, 48, 38, 28, 31, 37, 22, 34, 34, 37, 34, 33, 28, 33, 31, 35, 29, 33, 34, 33, 38, 37, 37,
32, 29, 38, 35, 34, 30, 35, 38, 30, 33, 31, 38, 38, 41, 38, 33, 34, 27, 29, 28, 37, 39, 31, 34, 27, 37, 37, 33, 30, 37, 32,
39, 37, 38, 38, 41, 35, 27, 32, 39, 31, 38, 27, 29, 33, 39, 37, 33, 32, 36, 32, 35, 38, 24, 38, 39, 33, 28, 27, 31, 21, 25,
34, 36, 33, 33, 33, 37, 30, 37, 37, 33, 32, 37, 33, 30, 34, 31, 39, 33, 33, 33, 34, 31, 31, 27, 35, 36, 38, 34, 30, 34, 34,
33, 32, 33, 30, 33, 34, 31, 38, 36, 33, 26, 34, 29, 37, 38, 37, 27, 35, 31, 37, 34, 38, 31, 30, 29, 37, 36, 33, 29, 29, 31,
27, 33, 36, 38, 33, 32, 41, 33, 42, 32, 32, 33, 37, 31, 38, 28, 29, 37, 34, 31, 32, 25, 32, 34, 37, 32, 38, 25, 39, 26, 48,
35, 28, 32, 34, 32, 28, 35, 28, 37, 37, 34, 33, 35, 38, 36, 42, 36, 25, 28, 39, 42, 37, 34, 33, 31, 35, 31, 26, 31, 36, 37,
33, 36, 27, 33, 34, 34, 33, 37, 39, 34, 48, 28, 31, 26, 38, 38, 37, 33, 28, 42, 37, 32, 37, 29, 38, 37, 26, 39, 30, 34, 32,
35, 37, 34, 33, 31, 27, 37, 27, 32, 38, 34, 31, 27, 36, 33, 32, 33, 48, 30, 36, 34, 32, 28, 36, 34, 27, 33, 38, 33, 35, 34,
24, 32, 37, 29, 31, 34, 38, 35, 35, 33, 32, 32, 29, 33, 25, 36, 32, 33, 37, 33, 29, 36, 38, 36, 28, 25, 31, 38, 32, 36, 37,
35, 38, 33, 33, 34, 38, 29, 33, 31, 44, 37, 34, 38, 33, 37, 38, 37, 35, 31, 38, 28, 31, 36, 38, 38, 29, 36, 27, 26, 29, 34,
37, 36, 35, 35, 33, 34, 38, 35, 30, 36, 25, 29, 33, 35, 38, 28, 32, 38, 40, 31, 32, 33, 28, 34, 38, 34, 37, 37, 35, 36, 29,
36, 35, 38, 37, 33, 25, 36, 38, 27, 28, 38, 37, 36, 35, 34, 31, 38, 38, 25, 34, 32, 40, 39, 35, 37, 38, 29, 31, 39, 31, 32,
32, 33, 38, 48, 35, 30, 33, 26, 31, 34, 27, 31, 26, 38, 33, 36, 38, 32, 35, 36, 23, 31, 41, 31, 38, 32, 27, 41, 31, 29,
31, 36, 28, 25, 38, 33, 35, 34, 31, 34, 33, 29, 34, 28, 32, 36, 34, 39, 33, 38, 39, 36, 33, 32, 32, 39, 32, 38, 29, 38,
38, 26, 36, 29, 34, 31, 41, 32, 36, 36, 38, 28, 35, 32, 35, 32, 35, 38, 30, 36, 31, 28, 34, 40, 33, 38, 34, 36, 28, 38, 35,
33, 32, 35, 36, 22, 36, 31, 36, 32, 34, 35, 26, 32, 28, 33, 33, 32, 34, 34, 32, 27, 29, 33, 31, 34, 42, 33, 35, 38, 33, 34,
33, 33, 28, 31, 35, 33, 36, 32, 35, 33, 33, 29, 37, 29, 36, 37, 28, 33, 35, 33, 41, 38, 37, 32, 38, 31, 38, 34, 32, 29, 36,
33, 34, 31, 36, 34, 28, 31, 32, 41, 25, 33, 28, 34, 28, 28, 26, 32, 43, 33, 34, 32, 33, 32, 36, 37, 33, 38, 35, 38, 27, 37,
34, 31, 41, 34, 38, 41, 34, 36, 37, 35, 42, 38, 33, 38, 38, 29, 33, 24, 35, 28, 34, 31, 42, 38, 33, 38, 28, 41, 40, 38, 29,
34, 34, 32, 35, 31, 34, 29, 32, 32, 37, 33, 36, 38, 26, 36, 36, 36, 33, 30, 31, 33, 27, 35, 26, 38, 26, 35, 37, 36, 37, 28,
33, 32, 38, 35, 32, 33, 36, 48, 34, 34, 35, 32, 33, 29, 34, 24, 31, 32, 48, 34, 35, 31, 35, 36, 28]

In [33]: |

```

now I'm going to take the average of these, you see we are very close 32.96121 so on, (Refer Slide Time: 15:30)



```
30, 33, 30, 37, 33, 29, 30, 30, 27, 20, 30, 37, 30, 33, 34, 31, 30, 30, 29,  
32, 33, 30, 40, 35, 30, 33, 26, 31, 34, 27, 31, 26, 30, 33, 33, 36, 38, 32,  
31, 36, 28, 25, 38, 33, 35, 34, 31, 34, 33, 29, 34, 28, 32, 36, 34, 30, 33,  
38, 26, 36, 29, 34, 31, 41, 32, 36, 36, 38, 28, 35, 32, 35, 32, 35, 38, 30,  
33, 32, 35, 36, 22, 36, 31, 36, 32, 34, 35, 26, 32, 28, 33, 33, 32, 34, 34,  
33, 33, 28, 31, 35, 33, 36, 32, 35, 31, 33, 29, 37, 29, 36, 37, 28, 33, 35,  
33, 34, 31, 36, 34, 28, 31, 32, 41, 25, 33, 28, 34, 28, 28, 26, 32, 43, 33,  
34, 31, 41, 34, 30, 41, 34, 36, 37, 35, 42, 38, 33, 38, 36, 29, 33, 24, 35,  
34, 34, 32, 35, 31, 34, 29, 32, 32, 37, 33, 36, 30, 26, 36, 36, 36, 33, 30,  
33, 32, 30, 35, 32, 33, 36, 40, 34, 34, 35, 32, 33, 29, 34, 24, 31, 32, 40,
```

```
In [33]: print(np.average(total_edges))
```

```
32.961261261261264
```

```
In [34]: |
```



Permissions: RW End

so the average says very close to 33 and hence we see that 33 edges are possible when we create a graph on 12 vertices with probability 0.5, when you run the piece of code for say 100 times, I'm sure most of you would have got 33, most of you wouldn't have got it, but still if you keep increasing the number of times you repeat it you will end up having 33 edges.

## IIT MADRAS PRODUCTION

Founded by  
Department of Higher Education  
Ministry of Human Resources Development  
Government of India

[www.nptel.iitm.ac.in](http://www.nptel.iitm.ac.in)

Copyrights Reserved