

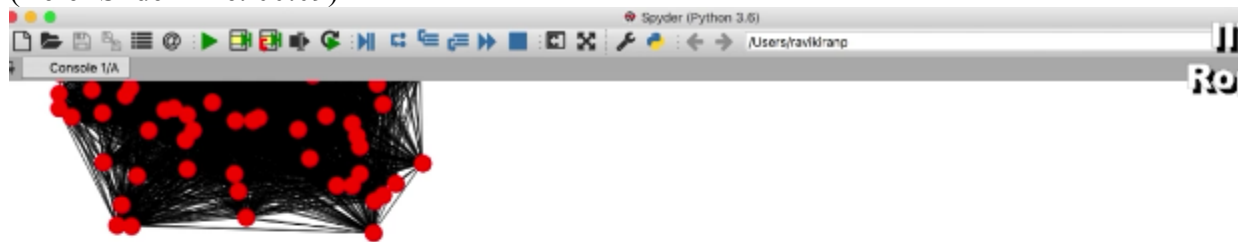
NPTEL
NPTEL ONLINE CERTIFICATION COURSE

Discrete Mathematics
Graph Theory - 1

Story so far – Using NetworkX

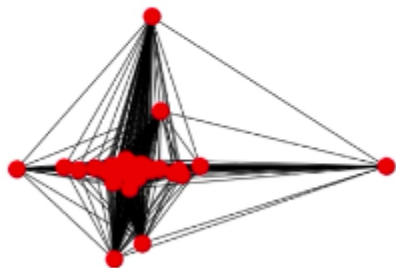
By
Prof. S.R.S Iyengar
Department of Computer Science
IIT Ropar

So we saw the various layouts for visualizing the graph,
(Refer Slide Time: 00:09)



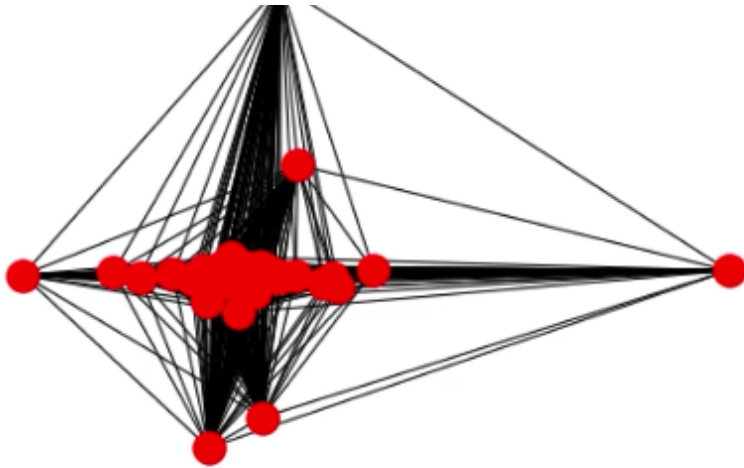
```
in [115]: nx.draw_spectral(M)
```

I



```
in [116]:
```

we had seen spectral, random, spring and so on. Now there is another nice command which will give `M.order` it says order, and then brackets like this, what is it going to do? By order we mean the number of vertices,
(Refer Slide Time: 00:33)

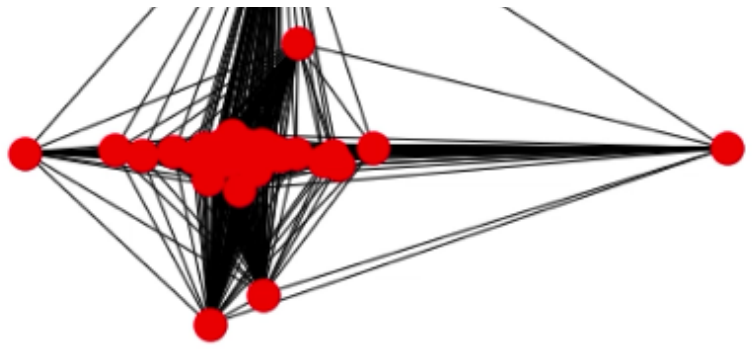


```
In [116]: M.order()  
Out[116]: 70
```

```
In [117]: |
```

see it's showing 70 initially I had chosen M to have 70 vertices, and now if I give M.order and brackets it's showing me 70 which says that there are 70 nodes in the graph.

Similarly M.size you must be guessing what size means, size means the number of edges which the graph will have, let me check oh it is having 2415 edges, this graph is quite big you see, (Refer Slide Time: 01:11)

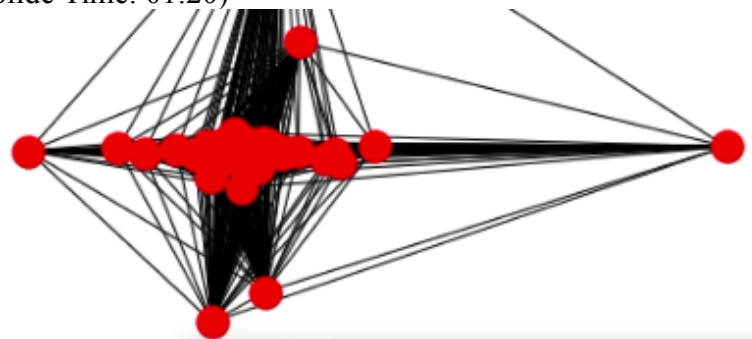


```
In [116]: M.order()
Out[116]: 70

In [117]: M.size()
Out[117]: 2415

In [118]: |
```

now there are several more things which we can do with nx. let me me check nx.tab if you give you can see here the various options
(Refer Slide Time: 01:20)



```
In [116]: M.o
Out[116]: 70

In [117]: M.s
Out[117]: 241

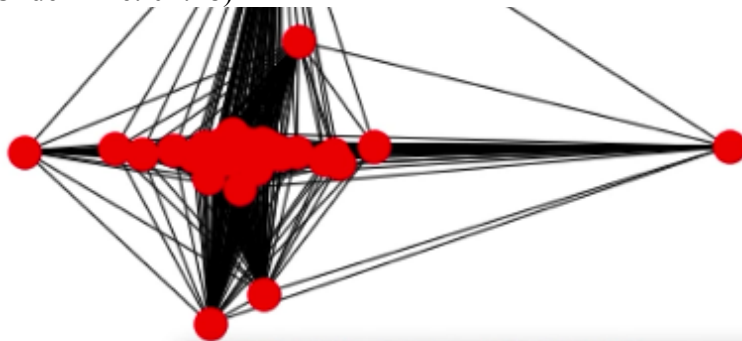
In [118]: nx.
```

- nx.ExceededMaxIterations
- nx.Graph
- nx.GraphMLReader
- nx.GraphMLWriter
- nx.HasACycle
- nx.LCF_graph
- nx.MultiDiGraph
- nx.MultiGraph

nx.graph, nx.digraph, multigraph, well you'll be learning all of these in the coming week.

For now let me check something which is very relevant to whatever we have learnt, okay, let me go to classes, circular, community, let me say G is something like, yeah, do you see something called graphical here

(Refer Slide Time: 01:48)

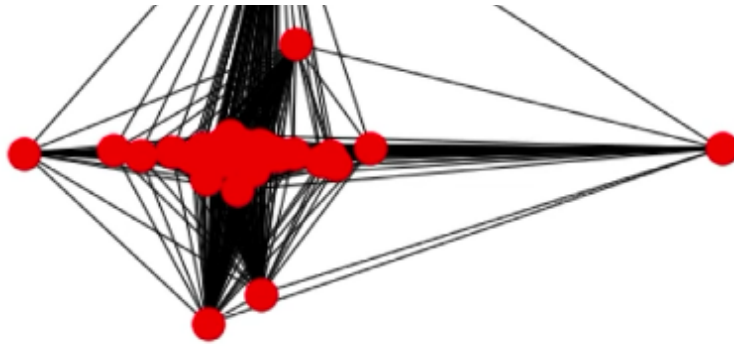


III
Re

```
nx.graph_edge_distance
nx.graph_number_of_cliques
nx.graphical
In [116]: M.o nx.graphmatrix
Out[116]: 70 nx.graphml
           nx.graphviews
In [117]: M.s nx.greedy_color
Out[117]: 241 nx.grid_2d_graph
In [118]: nx.g
```

now let us see how is it helpful for us, well I am going to do something like this now, nx.is_graphical and I am going to give a question mark

(Refer Slide Time: 02:02)



II
Ro

```
In [116]: M.order()
Out[116]: 70

In [117]: M.size()
Out[117]: 2415

In [118]: nx.is_graphical?
```

you see something here written here, true if the sequence is a valid degree sequence, and false if not,

(Refer Slide Time: 02:13)

```
spyder (Python 3.6)
/Users/ravikiran

Console I/A
sequence : list of iterable container
           A sequence of integer node degrees

method : "eg" | "hh"
         The method used to validate the degree sequence.
         "eg" corresponds to the Erdős-Gallai algorithm, and
         "hh" to the Havel-Hakimi algorithm.

Returns
-----
valid : bool
        True if the sequence is a valid degree sequence and False if not.

Examples
-----
>>> G = nx.path_graph(4)
>>> sequence = (d for n, d in G.degree())
>>> nx.is_graphical(sequence)
True

References
-----
Erdős-Gallai
  [EG1960]_, [choudum1986]_

Havel-Hakimi
  [havel1955]_, [hakimi1962]_, [CL1996]_
File:      ~/anaconda3/lib/python3.6/site-packages/networkx/algorithms/graphical.py
Type:      function

In [119]:
```

well, this is very much self-explanatory, if I give a sequence and if I ask if the sequence, if the degree sequence is graphic or not, we have used the word graphic earlier they are using the

word graphical well it means the same, you will know if a graph can be drawn on the given degree sequence or not, so it will give the value either true or false depending on how and what the sequence is.

(Refer Slide Time: 02:44)

```
>>> G = nx.path_graph(4)
>>> sequence = (d for n, d in G.degree())
>>> nx.is_graphical(sequence)
True
```

References

Erdős–Gallai

[EG1960]_, [choudum1986]_

Havel–Hakimi

[havel1955]_, [hakimi1962]_, [CL1996]_

File: ~/anaconda3/lib/python3.6/site-packages/net

Type: function

In [119]: 1|

So let me define my sequence L to be in square brackets 2, 2, 2, and now what I am going to check is it graphical? Well you had all been already seen that it's not nx.graphical, if I give this it's going to show up an error, so what I'm going to do is nx.is_graphical, so now and then in bracket I'm going to give the sequence L, yes, it is true,

(Refer Slide Time: 03:18)

References

Erdős-Gallai

[EG1960]_, [choudum1986]_

Havel-Hakimi

[havel1955]_, [hakimi1962]_, [CL1996]_

File: ~/anaconda3/lib/python3.6/site-packages/net

Type: function

In [119]: l=[2,2,2]

In [120]: nx.is_graphical(l)

Out[120]: True

In [121]:

if you remember we had already seen this in our earlier video, 2, 2, 2 will give a triangle or a C_3 , C_3 means a cycle with 3 nodes that is the graph on this degree sequence 2, 2, 2.

Now let me check for some other sequence L equals, I am going to use the same alphabet $L = 2, 2, 2, 1$

(Refer Slide Time: 00:09)

References

||
Ro

Erdős-Gallai

[EG1960]_, [choudum1986]_

Havel-Hakimi

[havel1955]_, [hakimi1962]_, [CL1996]_

File: ~/anaconda3/lib/python3.6/site-packages/net

Type: function

In [119]: l=[2,2,2]

In [120]: nx.is_graphical(l)

Out[120]: True

In [121]: l=[2,2,2,1]

(Refer Slide Time: 03:50)

let me check if this is graphical, what I'm going to do is NX is graphical you can always use your up arrows on your keyboard and to get back your previous command whatever is relevant there, nx.is_graphical L and it says false,

(Refer Slide Time: 04:06)


```
Havel-Hakimi
[havel1955]_, [hakimi1962]_, [CL1996]_
File:      ~/anaconda3/lib/python3.6/site-packages/netw
Type:      function

In [119]: l=[2,2,2]

In [120]: nx.is_graphical(l)
Out[120]: True

In [121]: l=[2,2,2,1]

In [122]: nx.is_graphical(l)
Out[122]: False

In [123]: |
```

now my question will be you must be guessing why is it false? You must know the reason because you have just 1 odd degree whereas it should be even number of odd degrees, right for a sequence to be graphic, and hence it is showing false here.

Now let us check with the sequence which we had earlier taken up that is 5, 5, 5, oh sorry it's 5, 5 and then 3, 3, 2, 2, 2,
(Refer Slide Time: 04:40)

```

HaveI-Hakimi
  [haveI1955]_, [hakimi1962]_, [CL1996]_
File:      ~/anaconda3/lib/python3.6/site-packages/net
Type:      function

In [119]: l=[2,2,2]

In [120]: nx.is_graphical(l)
Out[120]: True

In [121]: l=[2,2,2,1]

In [122]: nx.is_graphical(l)
Out[122]: False

In [123]: l=[5,5,3,3,2,2,2]

```

well the sequence we had already learnt it earlier, now let me check if its graphic, well, yes it is, (Refer Slide Time: 04:52)

```

In [119]: l=[2,2,2]

In [120]: nx.is_graphical(l)
Out[120]: True

In [121]: l=[2,2,2,1]

In [122]: nx.is_graphical(l)
Out[122]: False

In [123]: l=[5,5,3,3,2,2,2]

In [124]: nx.is_graphical(l)
Out[124]: True

In [125]: |

```

we are able to verify our results now.

And the last one let me take it as L equals, well you can always use some other alphabet I am just using L for convenience 5, 5, 5, 5, 2, 2, 2

(Refer Slide Time: 05:09)

```
In [119]: l=[2,2,2]
```

```
In [120]: nx.is_graphical(l)
```

```
Out[120]: True
```

```
In [121]: l=[2,2,2,1]
```

```
In [122]: nx.is_graphical(l)
```

```
Out[122]: False
```

```
In [123]: l=[5,5,3,3,2,2,2]
```

```
In [124]: nx.is_graphical(l)
```

```
Out[124]: True
```

```
In [125]: l=[5,5,5,5,2,2,2]
```

this is my sequence now, let me check if its graphical `nx.is_graphical` of this sequence gives me false,

(Refer Slide Time: 05:18)

```
In [121]: l=[2,2,2,1]
```

```
In [122]: nx.is_graphical(l)
```

```
Out[122]: False
```

```
In [123]: l=[5,5,3,3,2,2,2]
```

```
In [124]: nx.is_graphical(l)
```

```
Out[124]: True
```

```
In [125]: l=[5,5,5,5,2,2,2]
```

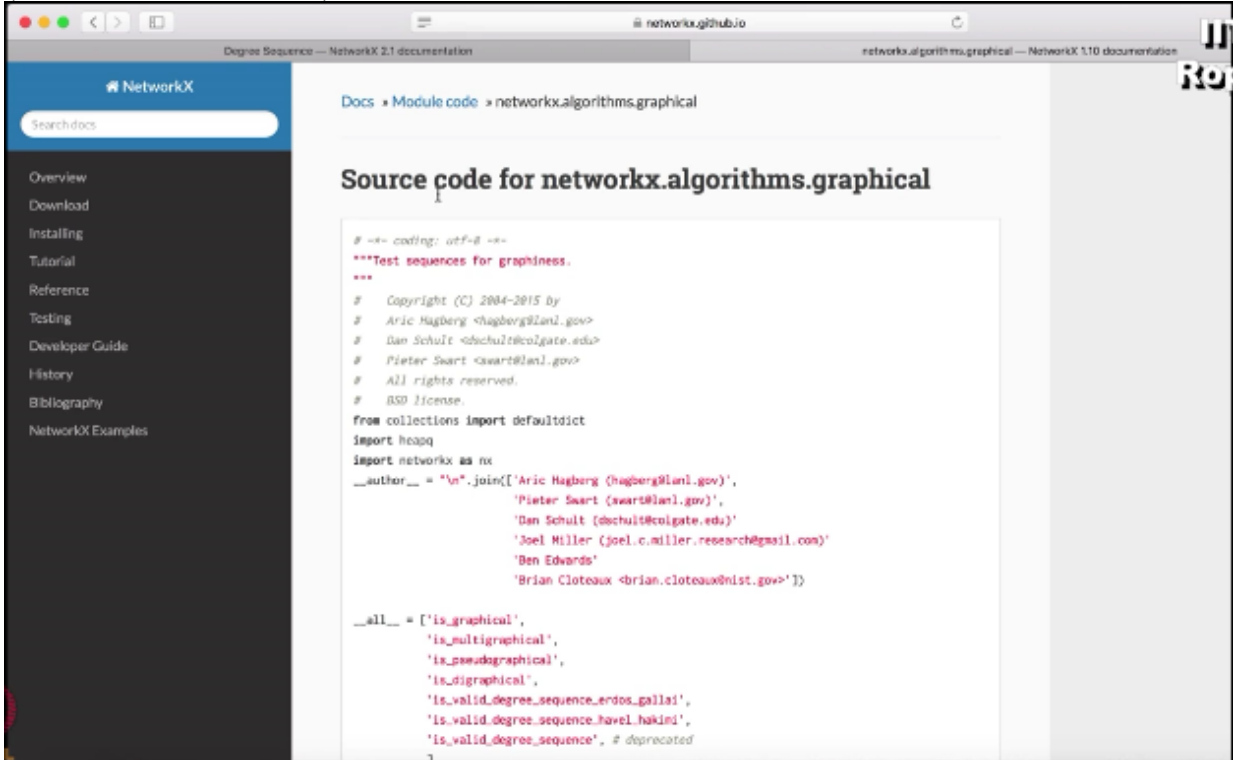
```
In [126]: nx.is_graphical(l)
```

```
Out[126]: False
```

```
In [127]: |
```

well, this was a challenge question for you guys if you remember in a video, and we had also shown that it is not graphical, we had seen a beautiful video where we had shown the exact reason as to why it is not graphical, if you remember we had seen that the last sequence had some negative numbers as degrees and hence it is not graphical, with just this one liner is graphical we could just play around and find out if the sequence is graphical or not.

Well do not assume that things can be this simple, several people are working behind the convenience of this one-liner, what is that? Let us see,
(Refer Slide Time: 00:09)

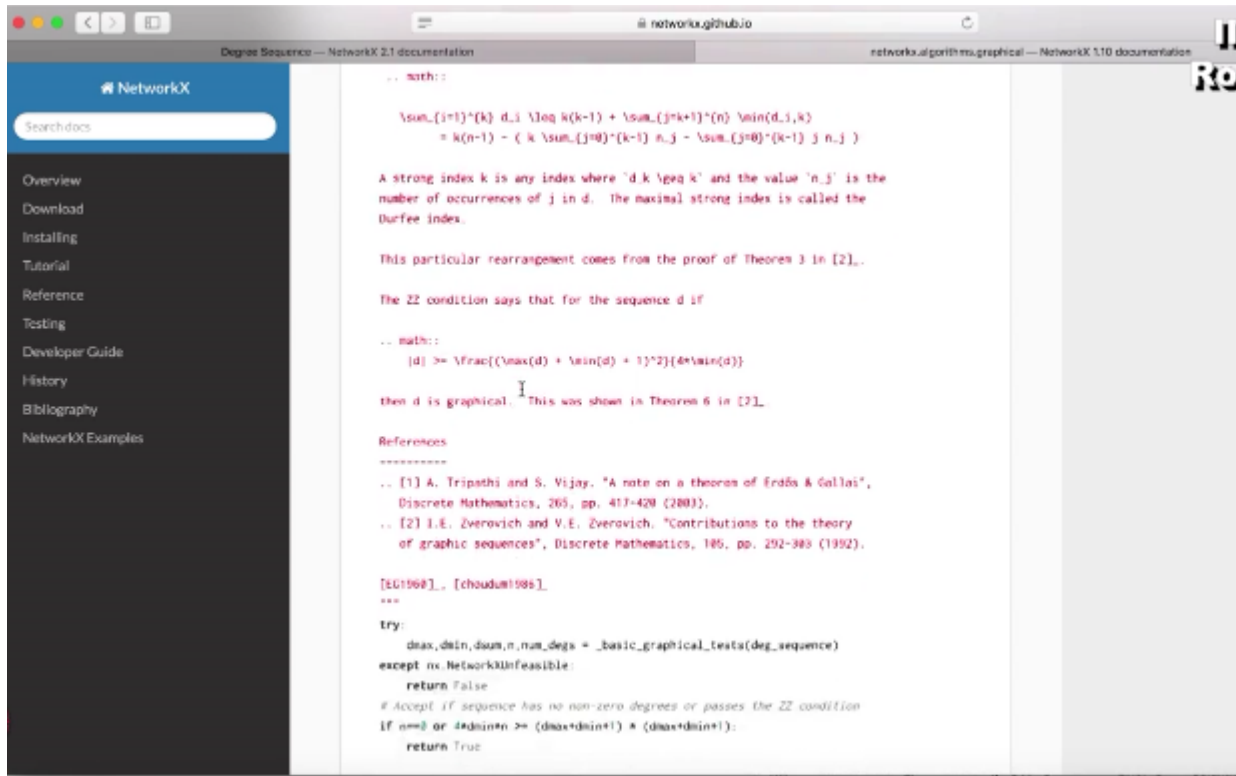


```
# -*- coding: utf-8 -*-
"""Test sequences for graphiness.
"""
# Copyright (C) 2004-2015 by
# Aric Hagberg <hagberg@lanl.gov>
# Dan Schult <dschult@colgate.edu>
# Pieter Swart <swart@lanl.gov>
# All rights reserved.
# BSD license.
from collections import defaultdict
import heapq
import networkx as nx
__author__ = "\n".join(['Aric Hagberg (hagberg@lanl.gov)',
                        'Pieter Swart (swart@lanl.gov)',
                        'Dan Schult (dschult@colgate.edu)',
                        'Joel Miller (joel.c.miller.research@gmail.com)',
                        'Ben Edwards',
                        'Brian Cloteaux <brian.cloteaux@nist.gov>'])

__all__ = ['is_graphical',
           'is_multigraphical',
           'is_pseudographical',
           'is_digraphical',
           'is_valid_degree_sequence_erdos_gallai',
           'is_valid_degree_sequence_havel_hakimi',
           'is_valid_degree_sequence', # deprecated
           ]
```

do you see this here, do you see this window? The source code for NetworkX algorithms, well behind this one-liner we have the entire program written and hence this is making our life very simple, this is the code written for finding if a degree sequence is graphical or not, do you see this,

(Refer Slide Time: 06:40)



well it's very huge, this is a documentation which people have done for this NetworkX you can see that every such code used in NetworkX has a huge source code like this, so we must be really happy and we must use the resources available to us in the package NetworkX.

Let us now go ahead and check a new graph, we have already seen what is a path graph, I named it as P here, let me see how we can draw a path graph, $P = nx.path_graph$ and as earlier we had given the number of vertices in brackets we should be mentioning the same here, path graph and let me say 5 vertices
(Refer Slide Time: 07:33)

```
In [124]: nx.is_graphical(l)
Out[124]: True
```

```
In [125]: l=[5,5,5,5,2,2,2]
```

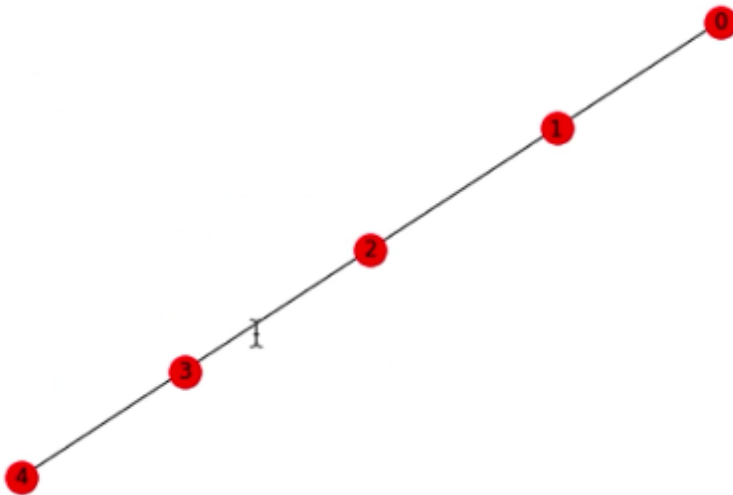
```
In [126]: nx.is_graphical(l)
Out[126]: False
```

```
In [127]: P=nx.path_graph(5)
```

```
In [128]:
```

and then we have to draw it, `nx.draw` I'm going to draw my graph `P` with labels, with labels as true,
(Refer Slide Time: 07:54)

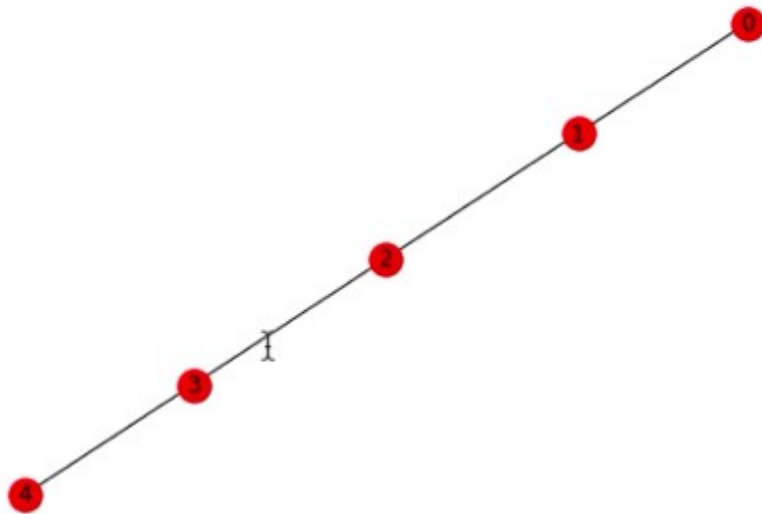
```
In [128]: nx.draw(P, with_labels=True)
```



```
In [129]:
```

do you see the graph here we've got a path graph on 5 vertices, this is just a random path graph.

Now we have already studied the concept of cut vertex and cut edge, if you remember a cut vertex is a vertex on whose removal the graph becomes disconnected, right well, now let us see how we can check that here, now let me make one point very clear as we have seen cut vertex it is also called as articulation points, now this command `nx.articulation_points` in the bracket the graph P,
(Refer Slide Time: 08:46)

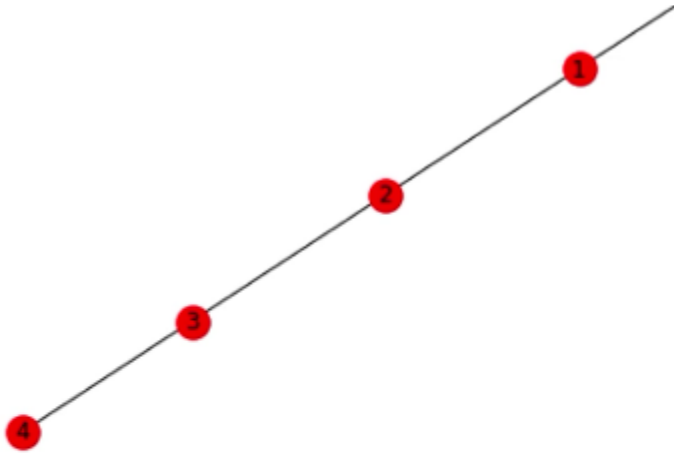


||
Re

```
In [129]: nx.articulation_points(P)
```

now this is going to give me the cut vertices let us see.

Well do you see self-command coming as your output here some line generator object articulation points act and some phrase,
(Refer Slide Time: 09:01)

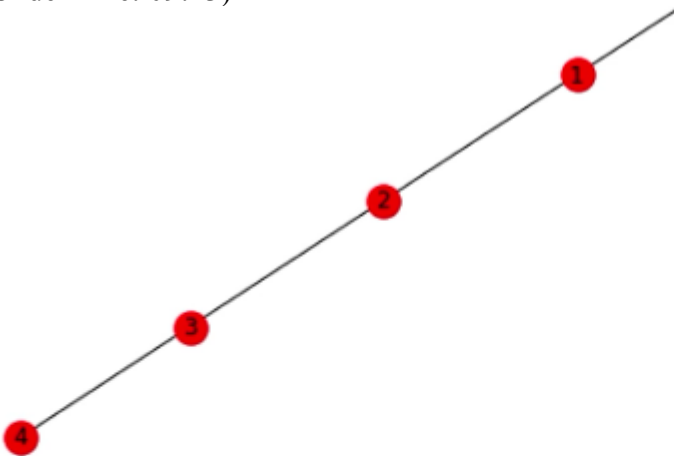


In [129]: `nx.articulation_points(P)`

Out[129]: <generator object articulation_points at 0xa1ba

In [130]:

well this is not giving us the list of all the cut vertices, so what we have to do right now is list and `nx.articulation_points` of P
(Refer Slide Time: 09:23)



In [129]: `nx.articulation_points(P)`

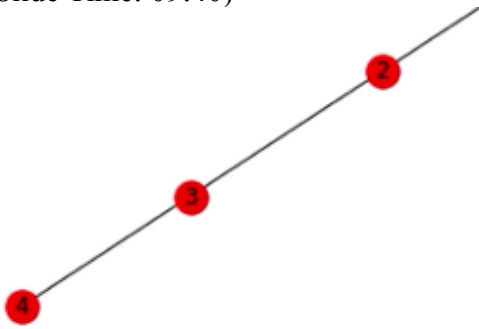
Out[129]: <generator object articulation_points at 0xa1ba

In [130]: `list(nx.articulation_points(P`

Arguments

please remember to give the syntax very accurately as a small error might lead up to you not getting your output, so `nx.articulation_points` let us see if this gives us the list,

(Refer Slide Time: 09:40)



```
In [129]: nx.articulation_points(P)
Out[129]: <generator object articulation_points at 0xa1ba...
```

```
In [130]: list(nx.articulation_points(P))
Out[130]: [3, 2, 1]
```

```
In [131]: |
```

well 3, 2, 1, it says, so the vertex 3, vertex 2, vertex 1, these are our cut vertices in this graph.

Now let me just create another one say $R = nx.path_graph$ or let me say some 12 vertices,
(Refer Slide Time: 10:04)

```
Spyder (Python 3.6)
/Users/harikiramp

Console 1/A
In [125]: l=[3,3,3,3,2,2,2]
In [126]: nx.is_graphical(l)
Out[126]: False
In [127]: P=nx.path_graph(5)
In [128]: nx.draw(P,with_labels=1)
In [129]: nx.articulation_points(P)
Out[129]: <generator object articulation_points at 0xa1bafaba0>
In [130]: list(nx.articulation_points(P))
Out[130]: [3, 2, 1]
In [131]: R=nx.path_graph(12)
```

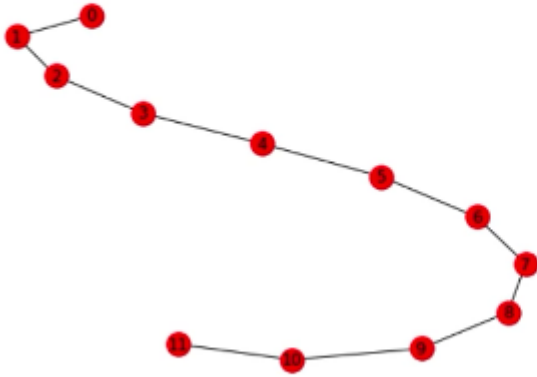
it's a path graph on 12 vertices, nx.draw R with labels as true, well, do you see this graph here (Refer Slide Time: 10:21)

```
In [130]: list(nx.articulation_points(P))
```

```
Out[130]: [3, 2, 1]
```

```
In [131]: R=nx.path_graph(12)
```

```
In [132]: nx.draw(R,with_labels=1)
```



```
In [133]:
```

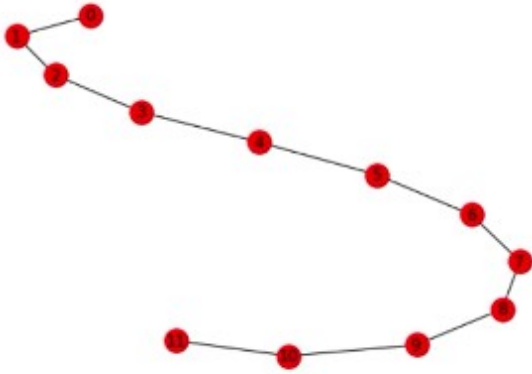
I

Permissions: RW End-of-line

it's a path graph on 12 vertices, let us just see what are the cut vertices, articulation points again you got the same phrase here, now list will solve all the problem, but please note it has to be list nx.articulation_points of this graph R, if you give P you're going to again get the same list, so your graph matters with whether it is P or R, so articulation points R will give me the list of all the vertices

(Refer Slide Time: 10:58)

```
In [132]: nx.draw(R,with_labels=1)
```



```
In [133]: nx.articulation_points(P)
```

```
Out[133]: <generator object articulation_points at 0x110192990>
```

```
In [134]: list(nx.articulation_points(R))
```

```
Out[134]: [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
```

```
In [135]:
```

I

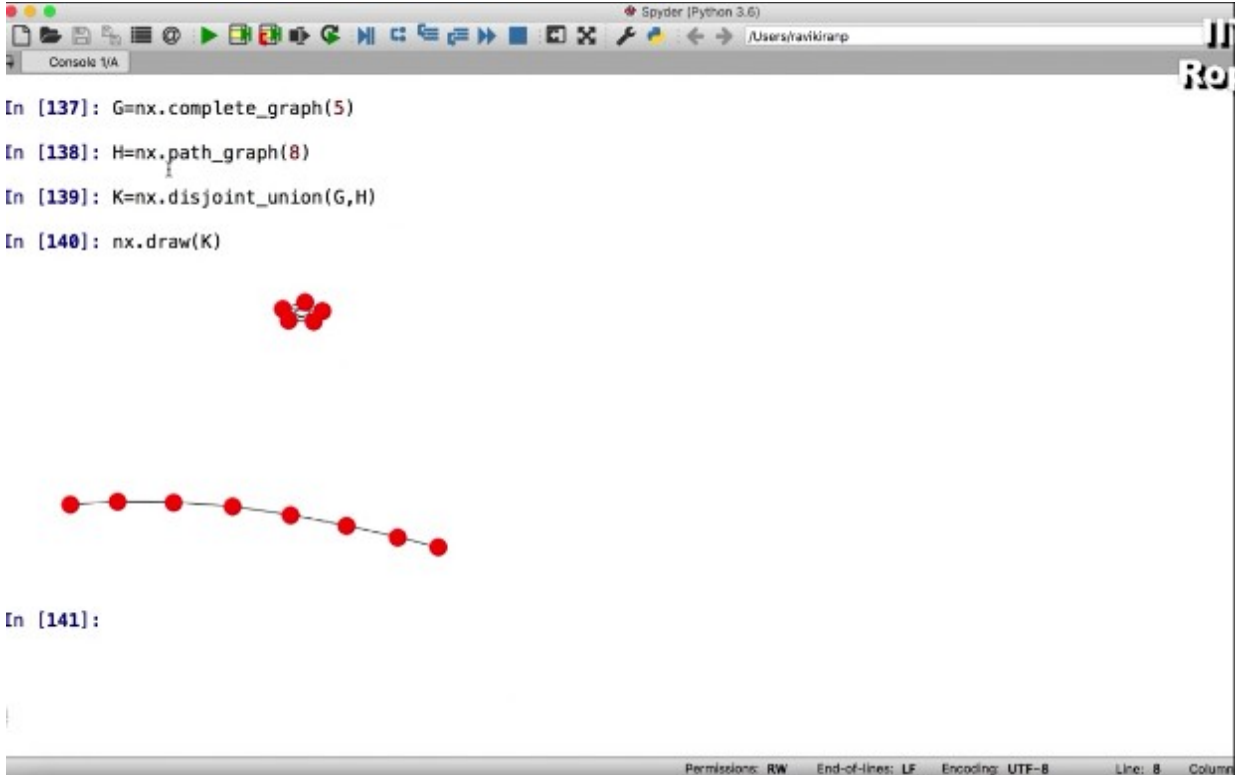
Permissions: RW End-of-line

which are cut vertices here.

Now what is the command for cut edges? It's a challenge for all of you to find it out all by yourself, let me create a graph now,
(Refer Slide Time: 11:29)

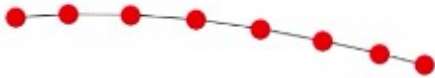
```
Spyder (Python 3.6)
/Users/ravikiranp
Console 1/4
In [137]: G=nx.complete_graph(5)
In [138]: H=nx.path_graph(8)
In [139]: K=nx.disjoint_union(G,H)
In [140]: nx.draw(K)

In [141]:
```



observe these graphs here which have created, now let us check if the graph is connected or not, if you remember we had showed that by connectedness there should be a path between every pair of vertices A and B, now let us check using this command `nx.is_connected`, at which graph? This graph,
(Refer Slide Time: 11:58)

```
In [137]: G=nx.complete_graph(5)
In [138]: H=nx.path_graph(8)
In [139]: K=nx.disjoint_union(G,H)
In [140]: nx.draw(K)
```



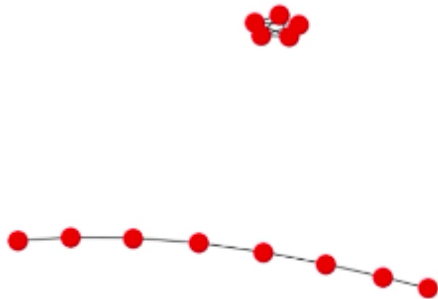
```
In [141]: nx.is_connected(K)
```

so you must check if this graph is connected or not, how do we do that? Is connected is the command for that, first you see this graph is not connected as we had seen for graphic is graphic it is the same for connected, the command S is connected.

Now let me show you a star graph here you had already seen what is a star graph is, I had earlier done it, a star graph looks something like this, don't worry much you will understand it nx.star_graph or let me say 11 vertices
(Refer Slide Time: 12:37)

```
In [137]: G=nx.complete_graph(5)
In [138]: H=nx.path_graph(8)
In [139]: K=nx.disjoint_union(G,H)
In [140]: nx.draw(K)
```

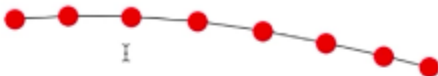
11
R9



```
In [141]: nx.is_connected(K)
Out[141]: False
In [142]: S=nx.star_graph(11)
```

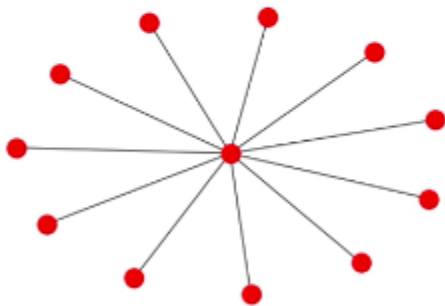
Permissions: RW End-of-line: LF Encoding: UTF-8

now nx.draw S,
(Refer Slide Time: 12:47)



11
R9

```
In [141]: nx.is_connected(K)
Out[141]: False
In [142]: S=nx.star_graph(11)
In [143]: nx.draw(S)
```



```
In [144]: |
```

Permissions: RW End-of-line: LF Encoding: UTF-8

you see this is called as a star graph, okay, we have seen such a graph earlier, now let me ask if it is connected, S, well you must jump and tell me that yes it will be true, and the command is verifying it,

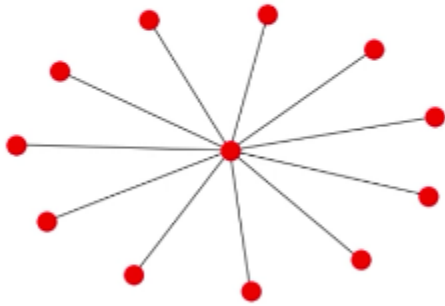
(Refer Slide Time: 13:08)

I
Re

```
In [141]: nx.is_connected(K)  
Out[141]: False
```

```
In [142]: S=nx.star_graph(11)
```

```
In [143]: nx.draw(S)
```



```
In [144]: nx.is_connected(S)  
Out[144]: True
```

```
In [145]: |
```

Permissions: RW End-of-lines: LF Encoding: UTF-8

it's a peculiar property of the star graph you see there are 11 vertices outside on the periphery and in the center is 1 vertex, whenever we give this command star graph in bracket the number of vertices in NetworkX it means that the number of vertices inside bracket is actually those which lie on the periphery these ones, this vertex which is in the center is not counted, and hence here 11 means all the 11 vertices which are lying here, so this is a star graph, we have seen that a star graph is connected.

IIT MADRAS PRODUCTION

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

www.nptel.iitm.ac.in

Copyrights Reserved