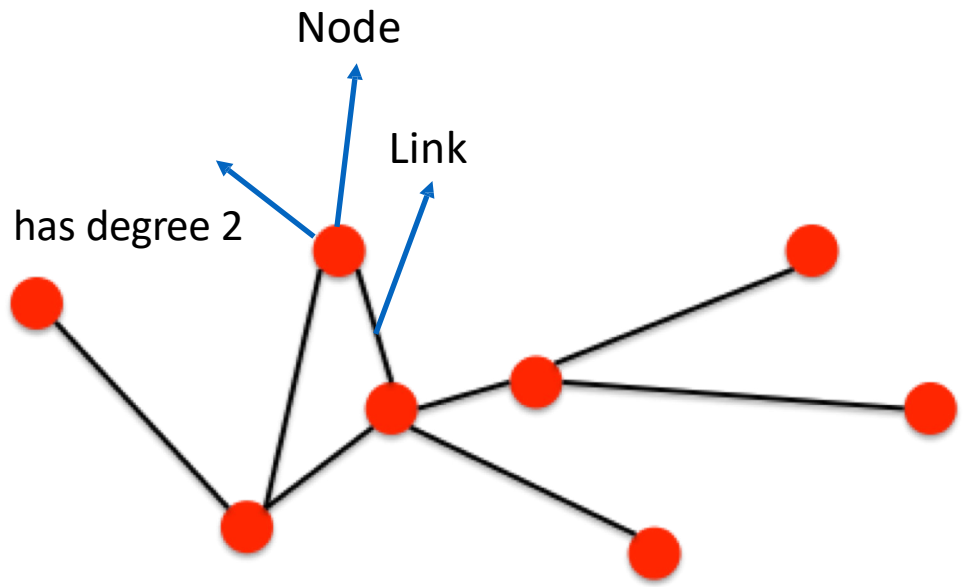
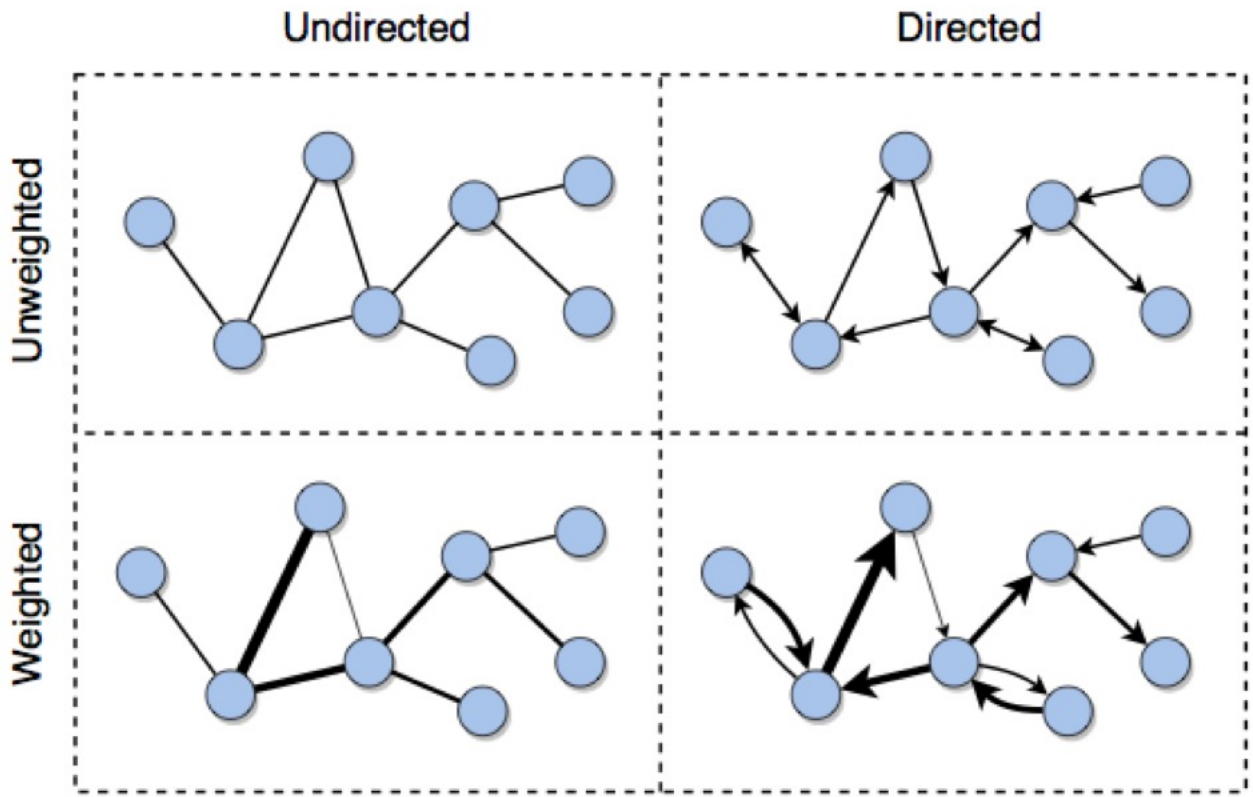
Lecture2: Network Elements

Definition

- A **network** (graph) is a structure used to model pairwise relations between objects. A network is made up of entities (aka **nodes**, vertices) and the relationships between them (aka **links**, edges)
- Number of edges of incident to the vertex is **the node degree**

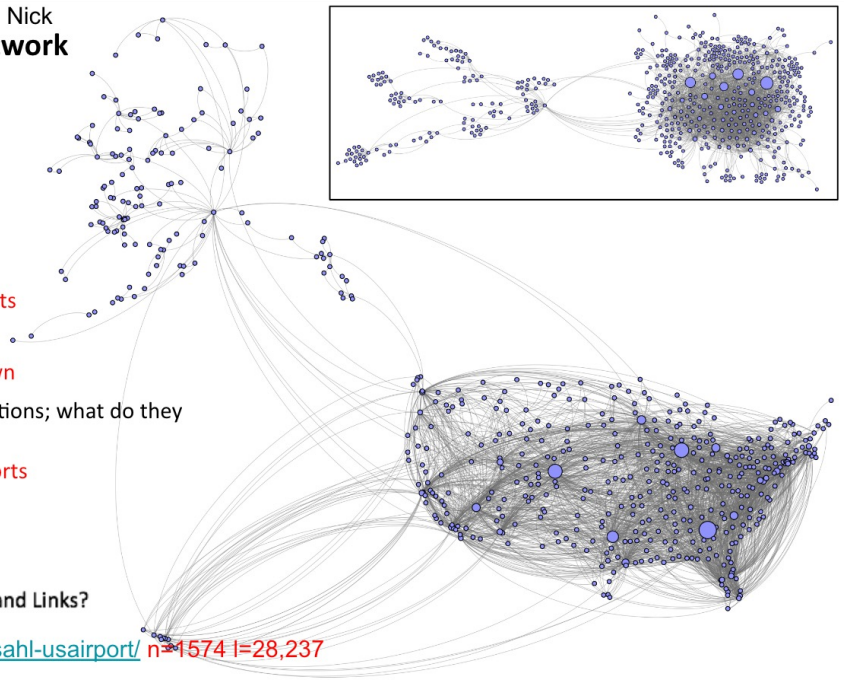




Can you think of a few examples in each of these categories?

Group 6: Alan, Jovon, Daniel, Nick

US air transportation network



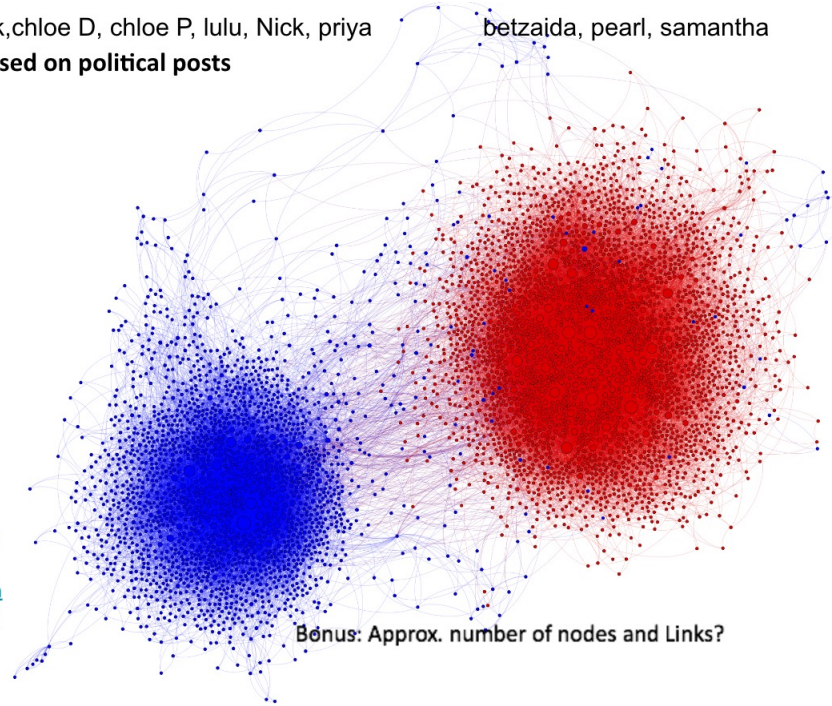
- What do nodes represent?
  - Airports
- What do links represent?
  - Flights from Point A to B
- Do links have direction?
  - Yes, based on the nature of flights
- Do links have weights?
  - How frequently the route is flown
- Larger nodes have more connections; what do they represent?
  - More flights to/from other airports
- What do the layouts represent?
  - The locations of airports

Bonus: Approx. number of nodes and Links?

http://konect.cc/networks/opsahl-usairport/ n=1574 l=28,237

Group 2 - Ice Cream Group - Isaak,chloe D, chloe P, lulu, Nick, priya betzaida, pearl, samantha

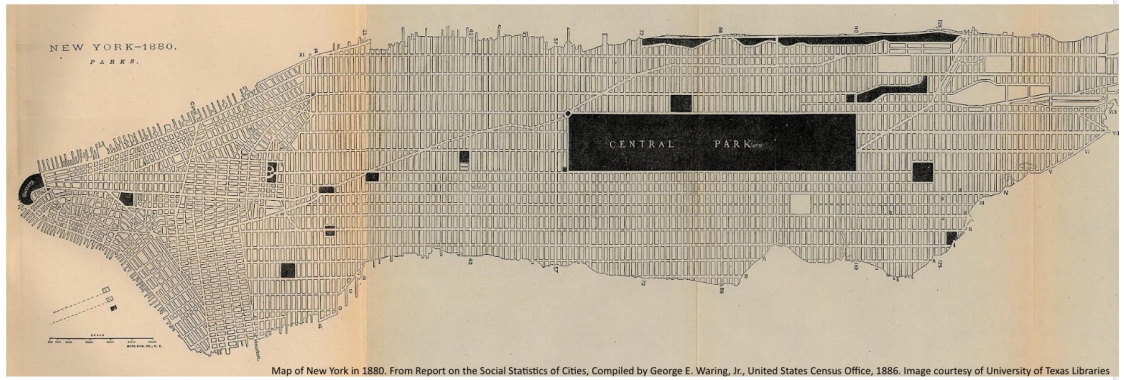
Retweet network on Twitter, based on political posts during 2010 US election



- What do nodes represent?
  - twitter users
- What do links represent?
  - A retweet
- Do links have direction?
  - Yes (i (origin) is retweeted by j(destination)
  - Do links have weights?
    - No
- Larger nodes have more connections; what does that mean?
  - users that received more retweets
- What do the clusters and colors represent?
  - Colors represent two political parties
  - Bonus: https://cnets.indiana.edu/groups/nan/truthy/visualizing-the-political-discourse-on-twitter/
    1000 users, 250,000 tweets , no information about retweets

Bonus: Approx. number of nodes and Links?

Street Networks Group 10 **Lulu, Yonghe, Nick, Clara, Geeta, Yara, Madison, Jocelyn**



Map of New York in 1880. From Report on the Social Statistics of Cities, Compiled by George E. Waring, Jr., United States Census Office, 1886. Image courtesy of University of Texas Libraries

- What do nodes represent?
  - Street intersections.
- What do links represent? Streets
- Do links have direction? What does it represent? Direction of traffic
- Do links have weights? What do they represent? Yes, how wide the streets are (number of lanes)

Bonus: Approx. number of nodes and Links? https://arxiv.org/abs/1308.1533

n=1047 l=4617

Remember: always consult Networkx https://networkx.github.io/

NetworkX

Stable (notes)

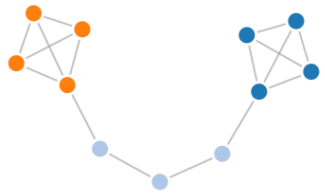
2.4 — October 2019
download | doc | pdf

Latest (notes)

2.5 development
github | doc | pdf

Software for complex networks

NetworkX is a Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks.



Network Elements Learning Objectives

1. Degree
2. Solution MyFirstNetwork_Exercise-Soln.ipynb
3. Clustering Coefficient
4. Shortest path lengths

For each of the 3 metrics you can calculate:

Average in the network, and histogram of the properties

Notebook Scripts for the Class

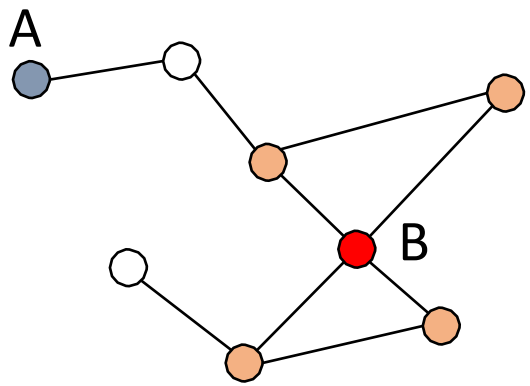
1. MyFirstNetwork_Exercise-Soln.ipynb

2. Example_ShortestPath_and_CC.ipynb

3. Reading_SocialNetwork.ipynb with data: SchoolEdges.csv

NODE DEGREES

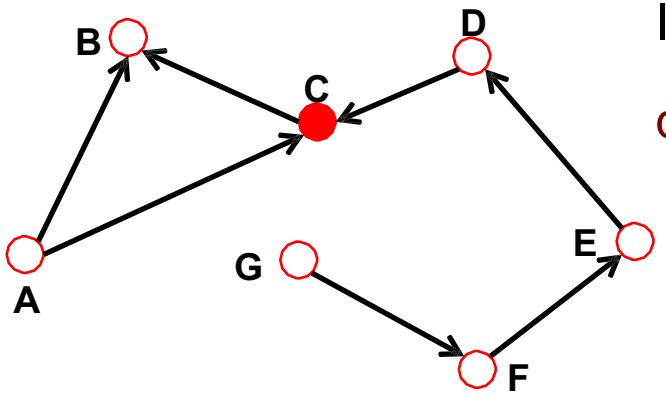
Undirected



Node degree: the number of links connected to the node.

$$k_A = 1 \qquad k_B = 4$$

Directed



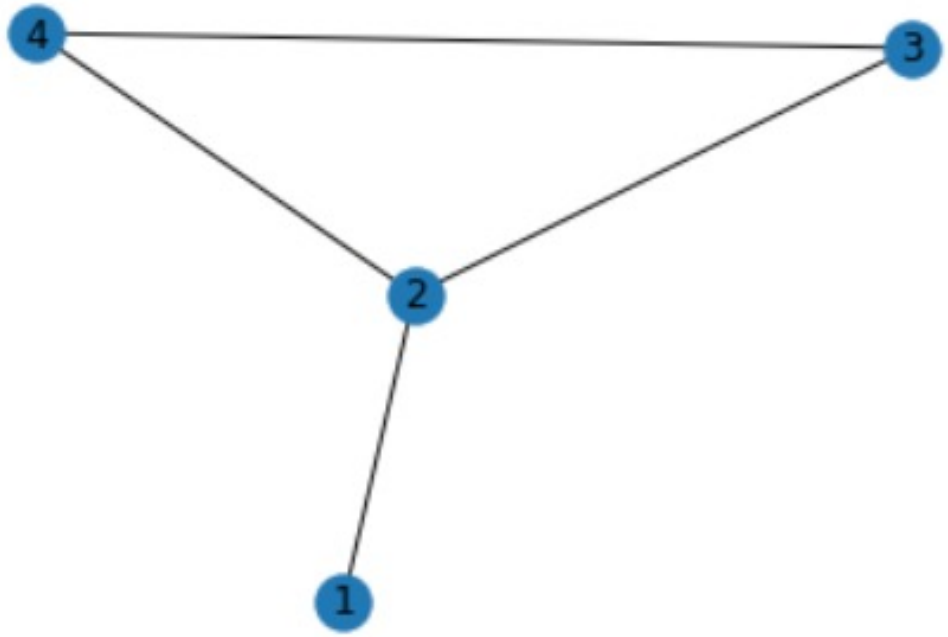
In *directed networks* we can define an in-degree and out-degree. The (total) degree is the sum of in- and out-degree.

$$k_C^{in} = 2 \qquad k_C^{out} = 1 \qquad k_C = 3$$

MyFirstNetwork_Exercise-Soln.ipynb

1) Nodes
2) Edges
3) Number of Nodes
4) Number of Edges
5) Number of Neighbors per node
6) Degree of each node
7) Average degree calculation

with number of links and nodes

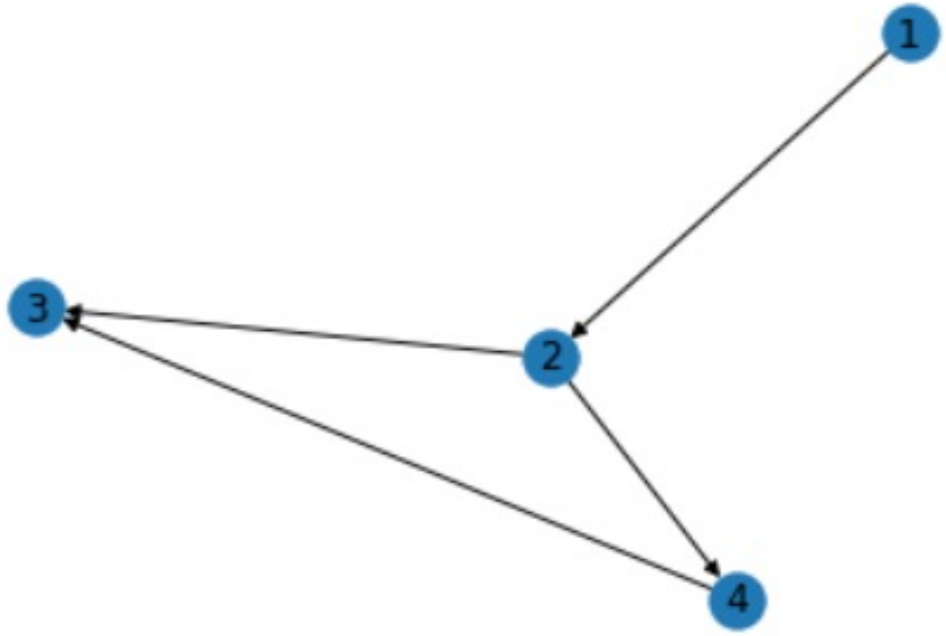


MyFirstNetwork_Exercise-Soln.ipynb

1) Nodes
2) Edges
3) Number of Nodes
4) Number of Edges
5) Number of Neighbors per node
6) Degree of each node

   In-degree of each node

   out-degree of each node

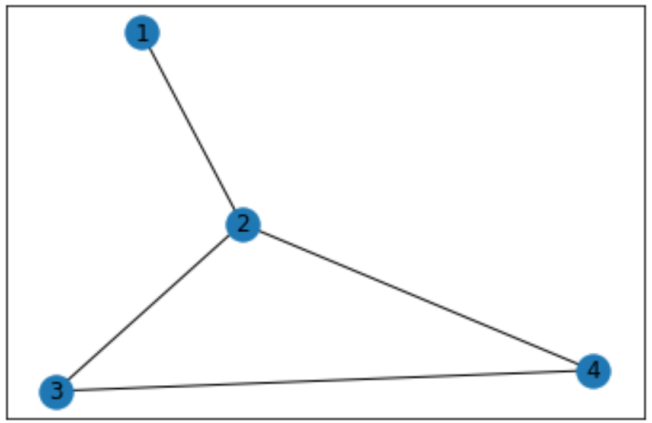


7) Average indegree outdegree and degree calculation with short equation
8) Histogram

Exercise: Create the 4 node network of Lecture 1 and respond the following questions (use networkx documentation or Google the questions)

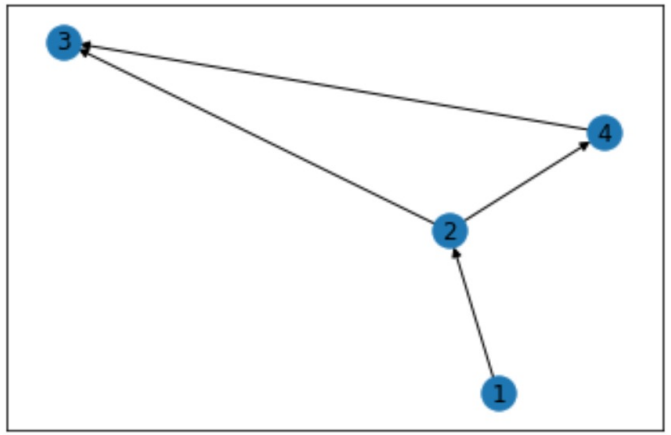
```
g = nx.Graph()  #Graph base class for undirected graphs. See also:DiGrap
g.add_edge('1','2',weight=1.0) #Nodes can be arbitrary (hashable) Python
g.add_edge('2','3',weight=1.0) #Edges are represented as links between n
g.add_edge('2','4',weight=1.0) #We create a network by adding 11 edges t
g.add_edge('4','3',weight=1.0)
```

```
nx.draw_networkx(g) #The draw_networkx function is called (check online
```




```
gd = nx.DiGraph()  #Graph base class for undirected graphs. See also:DiG
gd.add_edge('1','2',weight=1.0) #Nodes can be arbitrary (hashable) Pytho
gd.add_edge('2','3',weight=1.0) #Edges are represented as links between
gd.add_edge('2','4',weight=1.0) #We create a network by adding 11 edges
gd.add_edge('4','3',weight=1.0)
```

```
nx.draw_networkx(gd)
```



Note the Difference between Graph and DiGraph, they create Undirected and Dierected Graphfs respectively
Here the Weight of each link is the same

Question 1, Print Nodes

```
print("Nodes: ",list(gd.nodes))
```

```
Nodes:  ['1', '2', '3', '4']
```

Question 2, Print Edges

```
print("Edges: ",list(gd.edges))
```

```
Edges:  [('1', '2'), ('2', '3'), ('2', '4'), ('4', '3')]
```

Question 3, Number of Nodes

```
print("Number of Nodes: ",gd.number_of_nodes())
```

```
Number of Nodes:  4
```

Question 4, Number of Edges

```
print("Number of Edges: ",gd.number_of_edges())
```

```
Number of Edges:  4
```

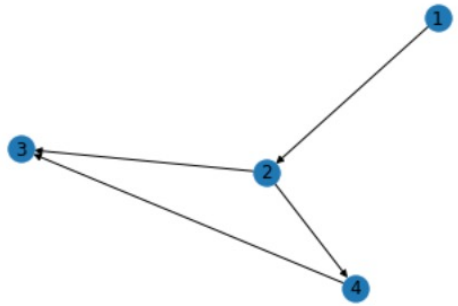
Question 5, Neighbors of each Node

```
for node in g.nodes():
    print ("Neighbors of ", node, " are : ", list(gd.neighbors(node)))
```

```
Neighbors of  1  are :  ['2']
Neighbors of  2  are :  ['3', '4']
Neighbors of  3  are :  []
Neighbors of  4  are :  ['3']
```

```
for node in g.nodes():
    print ("Predecessors of ", node, " are : ", list(gd.predecessors(node)))
```

```
Predecessors of  1  are :  []
Predecessors of  2  are :  ['1']
Predecessors of  3  are :  ['2', '4']
Predecessors of  4  are :  ['2']
```




```
for node in g.nodes():
    print ("Sucessors of ", node, " are : ", list(gd.successors(node)))
```

```
Sucessors of  1  are :  ['2']
Sucessors of  2  are :  ['3', '4']
Sucessors of  3  are :  []
Sucessors of  4  are :  ['3']
```

Note that in a directed graph neighbors are only the Out degree node but degree(node)=in_dgree(node)+out_degree(node)

Example extracting the degree, in_degree and out_degree of node 4 in Graph dg

```
gd.degree('4')
```

```
2
```

```
gd.out_degree('4')
```

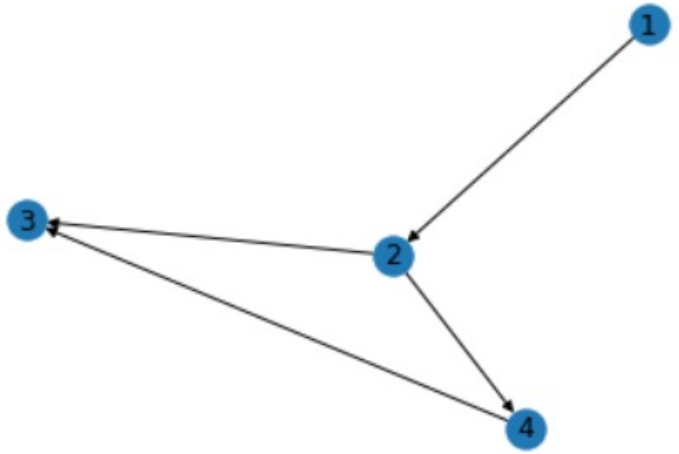
```
1
```

```
gd.in_degree('4')
```

```
1
```

```
for node in gd.nodes():
    print ("Degree of ", node, " is : ", gd.degree(node))
```

```
Degree of  1  is :  1
Degree of  2  is :  3
Degree of  3  is :  2
Degree of  4  is :  2
```




```
for node in gd.nodes():
    print ("In-Degree of ", node, " is : ", gd.in_degree(node))
```

```
In-Degree of  1  is :  0
In-Degree of  2  is :  1
In-Degree of  3  is :  2
In-Degree of  4  is :  1
```

```
for node in gd.nodes():
    print ("Out-degree of ", node, " is : ", gd.out_degree(node))
```

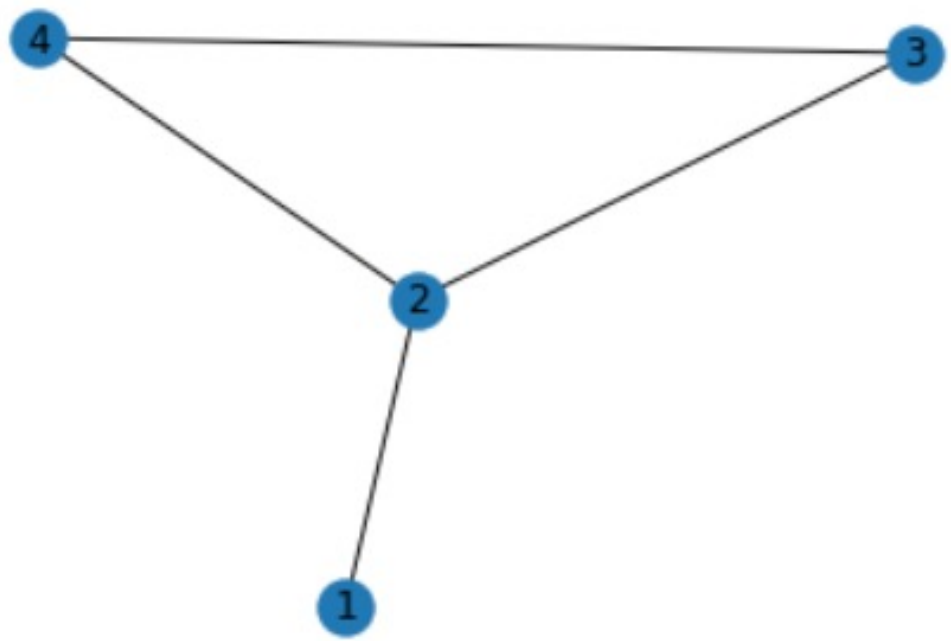
```
Out-degree of  1  is :  1
Out-degree of  2  is :  2
Out-degree of  3  is :  0
Out-degree of  4  is :  1
```

Note degree(node) = in_degree(node)+out_degree(node)

6) Iterate through nodes and print degree

```
for node in g.nodes():
    print ("Degree of ", node, " is : ", len(list(g.neighbors(node))),"or ",g.degree(node))
```

```
Degree of  1  is :  1 or  1
Degree of  2  is :  3 or  3
Degree of  3  is :  2 or  2
Degree of  4  is :  2 or  2
```

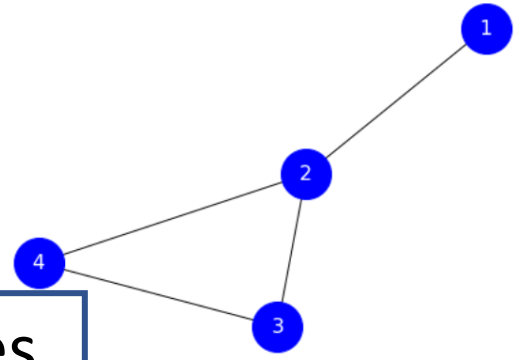


Formula of the Average Degree

Average Degree for an Undirected Graph

- It is the sum of the degree of each node divided by the number of nodes

<degree> = (1+3+2+2)/4 = 2



- It can be also written as the 2x(Number of Links)/#Nodes

<degree> = 2x(4)/4 = 2

```
undirected graph = ", np.mean(list(dict(g.degree()).values())), ", or: ", 2.0*g.number_of_edges()/g.number_of_nodes())
```

```
Average degree of  undirected graph =  2.0 , or:  2.0
```

```
np.mean(list(dict(g.degree()).values()))
```

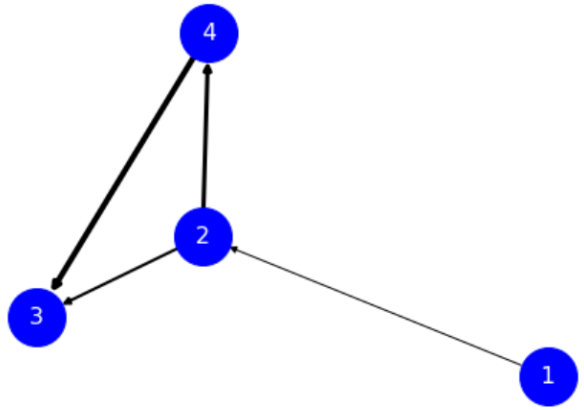
```
2.0
```

Average Degree for a Directed Graph

- It is the sum of the degree of each node divided by the number of nodes

  <in-degree> = (0+1+2+1)/4 = 1

  <out-degree> = (1+2+0+1)/4 = 1



- It can be also written as the (Number of Links)/#Nodes

  <in-degree> = (4)/4 = 1 <out-degree> = (4)/4 = 1

- Finally, the average degree is

  2x(Number of Links)/#Nodes or

  <degree>=<out-degree> + <in-degree> = 2

(Number of Links)/#Nodes

```
print ("Average out-degree of  directed graph = ",np.mean(list(dict(gd.out_degree()).values())),", or: ",gd.number_of_
```

```
Average out degree of  directed graph =  1.0 , or:  1.0
```

(Number of Links)/#Nodes

```
print ("Average in-degree of  directed graph = ",np.mean(list(dict(gd.in_degree()).values())),", or: ",gd.number_of_ed
```

```
Average in-degree of  directed graph =  1.0 , or:  1.0
```

```
print ("Average degree of  directed graph = ",np.mean(list(dict(gd.degree()).values())),", or: ",np.mean(list(dict(gd.
```

```
Average degree of  directed graph =  2.0 , or:  2.0
```

<degree>=<out-degree> + <in-degree> = 2

```
np.mean(list(dict(gd.in_degree()).values()))+np.mean(list(dict(gd.out_degree()).values()))
```

```
2.0
```

```
np.mean(list(dict(gd.degree()).values()))
```

```
2.0
```

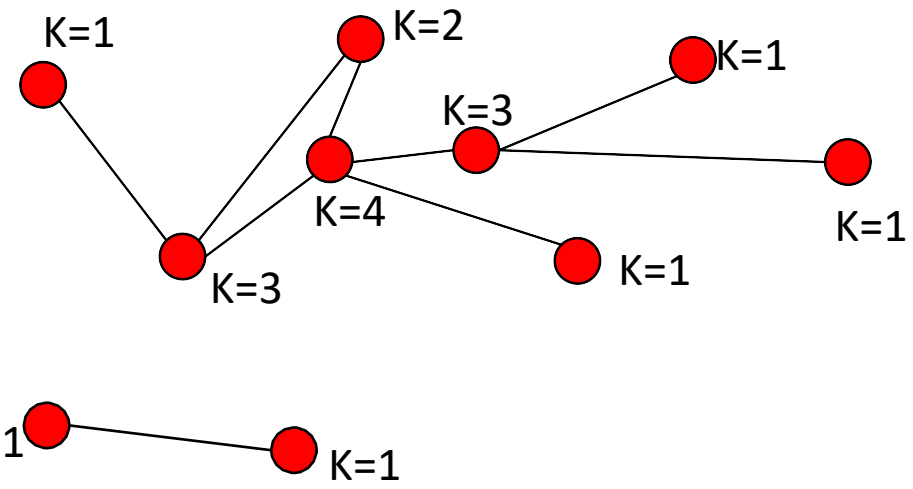
Building a Histogram of the properties of the nodes

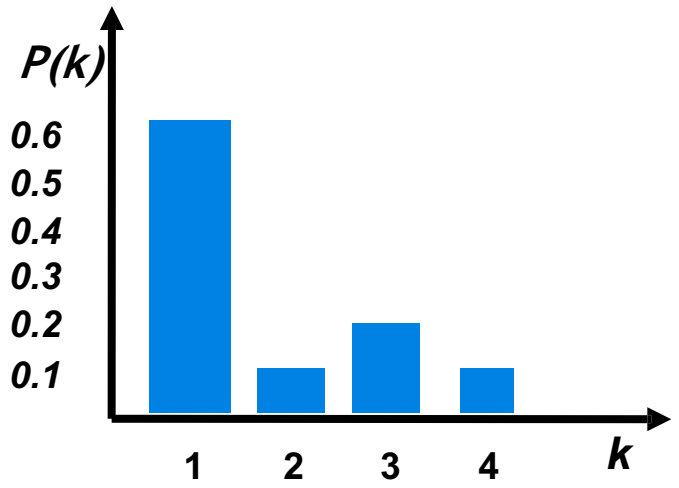
DEGREE DISTRIBUTION

Degree distribution P(k): probability that a randomly chosen vertex has degree k

N_k = # nodes with degree k

$P(k) = N_k / N$ → plot





K=1 N_1=6
K=2 N_2=1
K=3 N_3=2
K=4 N_4=1

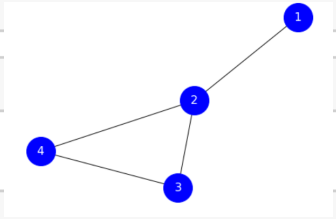
N = 10

The degree distribution is the normalized Histogram, it gives the probability to find a node with degree k

```
degree_list=list(dict(g.degree()).values())
```

```
degree_list
```

```
[1, 3, 2, 2]
```



Range=[min,max+2]

```
Bins = range(1,max(degree_list)+2) ##Note the range max +2
```

```
for n in Bins:
    print(n)
```

```
1
2
3
4
```

```
counts,k = np.histogram(degree_list, bins=Bins)
```

```
counts  #counts the occurrence of degree_list in Bins
```

```
array([1, 2, 1])
```

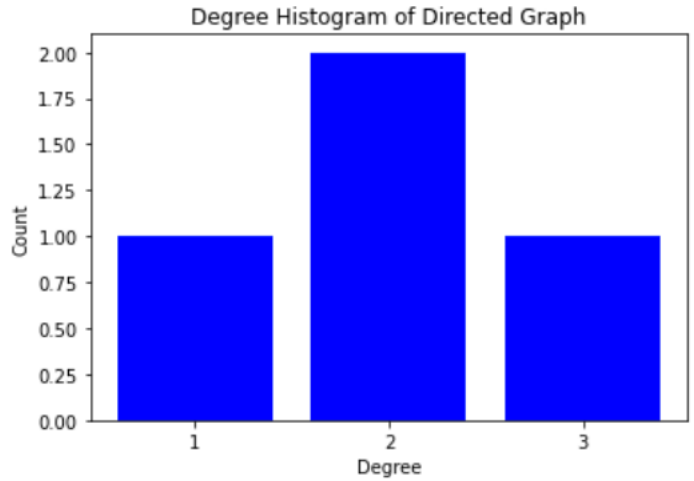
```
k #Note that k the vector is lager that count that is why we use k[:-1] in the bar plot
```

```
array([1, 2, 3, 4])
```

We need to plot

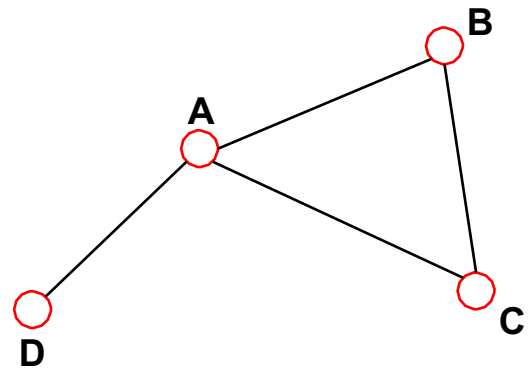
```
fig, ax = plt.subplots()
plt.bar(k[:-1],counts,width=0.80, color='b')
plt.title("Degree Histogram of Directed Graph")
plt.ylabel("Count")
plt.xlabel("Degree")
ax.set_xticks([d for d in k[:-1]])
ax.set_xticklabels(k[:-1])
```

```
[Text(1, 0, '1'), Text(2, 0, '2'), Text(3, 0, '3')]
```



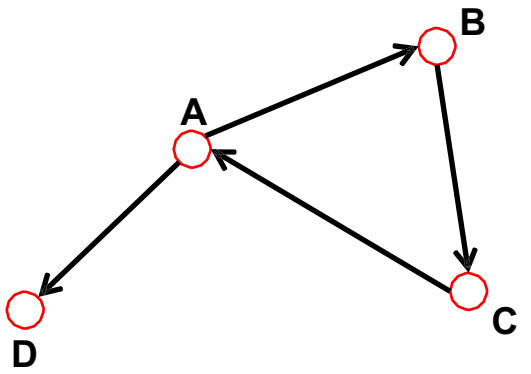
Shortest Path length

SHORTEST PATH LENGTH



The *distance (shortest path, geodesic path)* between two nodes is defined as the number of edges along the shortest path connecting them.

*If the two nodes are disconnected, the distance is infinity.

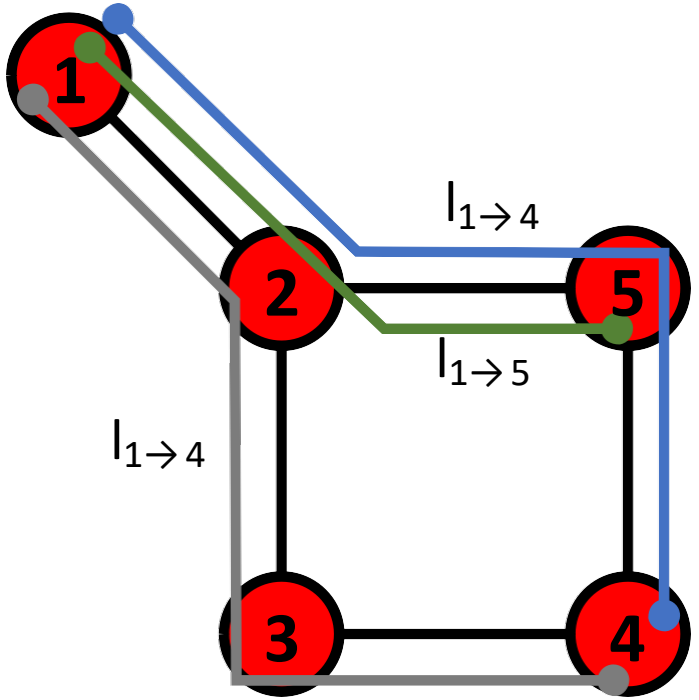


In directed graphs each path needs to follow the direction of the arrows.

Thus in a digraph the distance from node A to B (on an AB path) is generally different from the distance from node B to A (on a BCA path).

Shortest Path

The path with the shortest length between two nodes (distance).

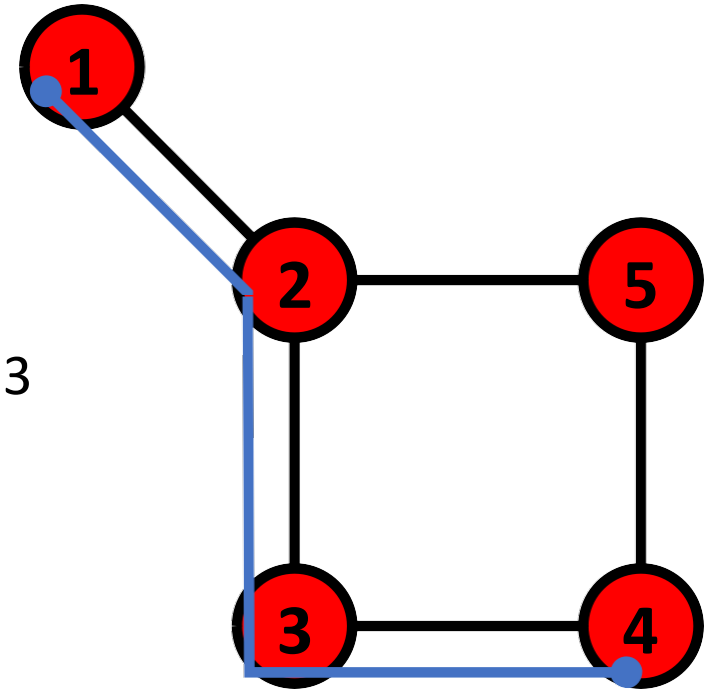


$l_{1\to 4} = 3$

$l_{1\to 5} = 2$

Average Shortest Path Length

$$(l_{1\to 2} + l_{1\to 3} + l_{1\to 4} + l_{1\to 5} + l_{2\to 3} + l_{2\to 4} + l_{2\to 5} + l_{3\to 4} + l_{3\to 5} + l_{4\to 5})/10 = 1.7$$



$l_{1\to 4} = 3$

The average of the shortest paths for all pairs of nodes.

Important Note: Numbers of pairs for a network of size N is N(N-1)

2*(1+2+3
+2+1+2
+1+1+2
+1)/20 = 1.6

N(N-1)=20

shortest_path(*G, source=None, target=None, weight=None*) [source]

Compute shortest paths in the graph.

Parameters:

- **G** (*NetworkX graph*)
- **source** (*node, optional*) – Starting node for path. If not specified, compute shortest paths for each possible starting node.
- **target** (*node, optional*) – Ending node for path. If not specified, compute shortest paths to all possible nodes.
- **weight** (*None or string, optional (default = None)*) – If None, every edge has weight/distance/cost 1. If a string, use this edge attribute as the edge weight. Any edge attribute not present defaults to 1.

Returns:

path – All returned paths include both the source and target in the path.

shortest_path_length(*G, source=None, target=None, weight=None*) [source]

Compute shortest path lengths in the graph.

Parameters:

- **G** (*NetworkX graph*)
- **source** (*node, optional*) – Starting node for path. If not specified, compute shortest path lengths using all nodes as source nodes.
- **target** (*node, optional*) – Ending node for path. If not specified, compute shortest path lengths using all nodes as target nodes.
- **weight** (*None or string, optional (default = None)*) – If None, every edge has weight/distance/cost 1. If a string, use this edge attribute as the edge weight. Any edge attribute not present defaults to 1.

Returns:

length – If the source and target are both specified, return the length of the shortest path from the source to the target.

average_shortest_path_length(*G, weight=None*) [source]

Return the average shortest path length.

The average shortest path length is

$$a = \sum_{s,t \in V} \frac{d(s,t)}{n(n-1)}$$

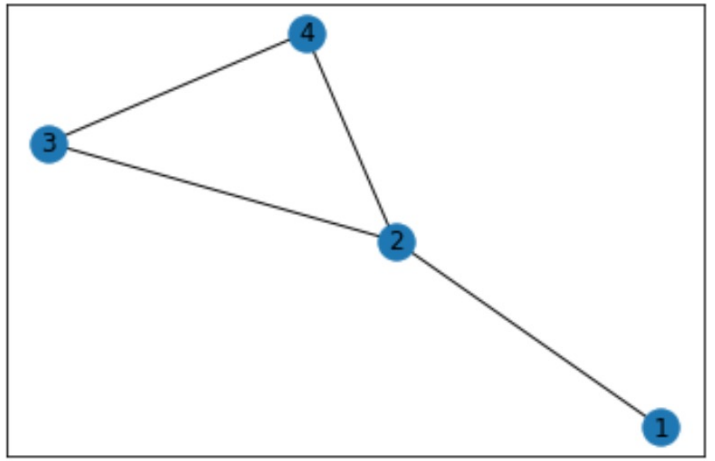
where `V` is the set of nodes in `G`, `d(s, t)` is the shortest path from `s` to `t`, and `n` is the number of nodes in `G`.

Example 1, Average Shortest Path

```
1 has  3 paths
['1', '2'] length  1
['1', '2', '3'] length  2
['1', '2', '4'] length  2
2 has  3 paths
['2', '1'] length  1
['2', '3'] length  1
['2', '4'] length  1

3 has  3 paths
['3', '2'] length  1
['3', '4'] length  1
['3', '2', '1'] length  2

4 has  3 paths
['4', '2'] length  1
['4', '3'] length  1
['4', '2', '1'] length  2
```



<L>=16/12

Number of Paths in Undirected Graph: nodes(nodes-1)

See: Example_ShortestPath_and_CC.ipynb

In the social sciences, the word **"clique”** is used to describe a group (averaging 5 or 6) "persons who interact with each other more regularly and intensely than others in the same setting."

Definition of Clustering Coefficient

clustering(*G, nodes=None, weight=None*) [source]

Compute the clustering coefficient for nodes.

For unweighted graphs, the clustering of a node u is the fraction of possible triangles through that node that exist,

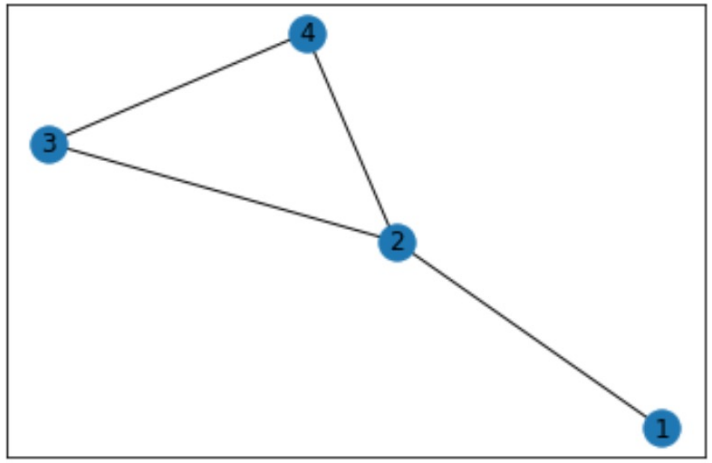
$$c_u = \frac{2T(u)}{deg(u)(deg(u)-1)},$$

where $T(u)$ is the number of triangles through node u and $deg(u)$ is the degree of u.

Note there are extensions of the definitions for Weighted and Directed Graphs

Next class we study the calculations of Paths and clustering coefficient in ipynb and cover Routing in Networks

Example 1 Average Clustering Coefficient




```
nx.clustering(g)
```

```
{'1': 0, '2': 0.3333333333333333, '3': 1.0, '4': 1.0}
```

```
nx.average_clustering(g)
```

```
0.5833333333333333
```

```
(0+0.333333+1+1)/4
```

```
0.58333325
```

T(1) = 0 2*0/(1)(0) =0

T(2)=1 2*1/(3)(2)=0.333

T(3)=1 2*1/(2)(1)=1

T(4)=1 2*1/(2)(1)=1

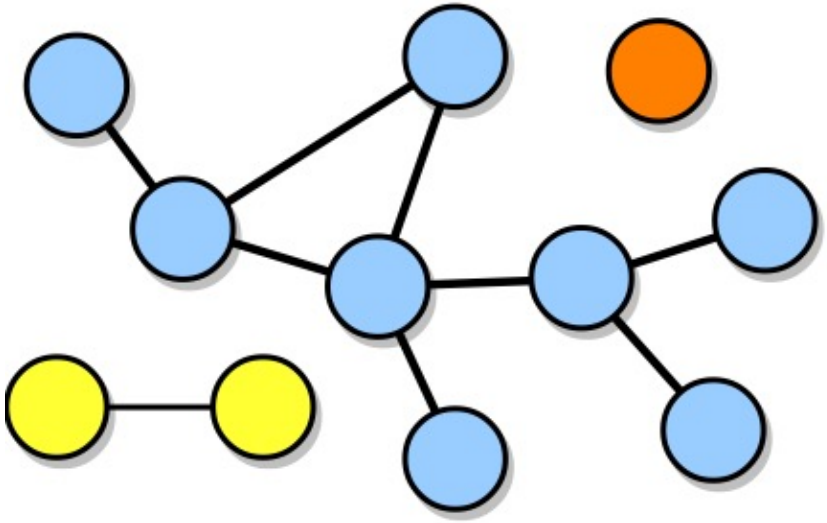
$$\frac{2T(u)}{deg(u)(deg(u)-1)}$$

<C>=0.5833

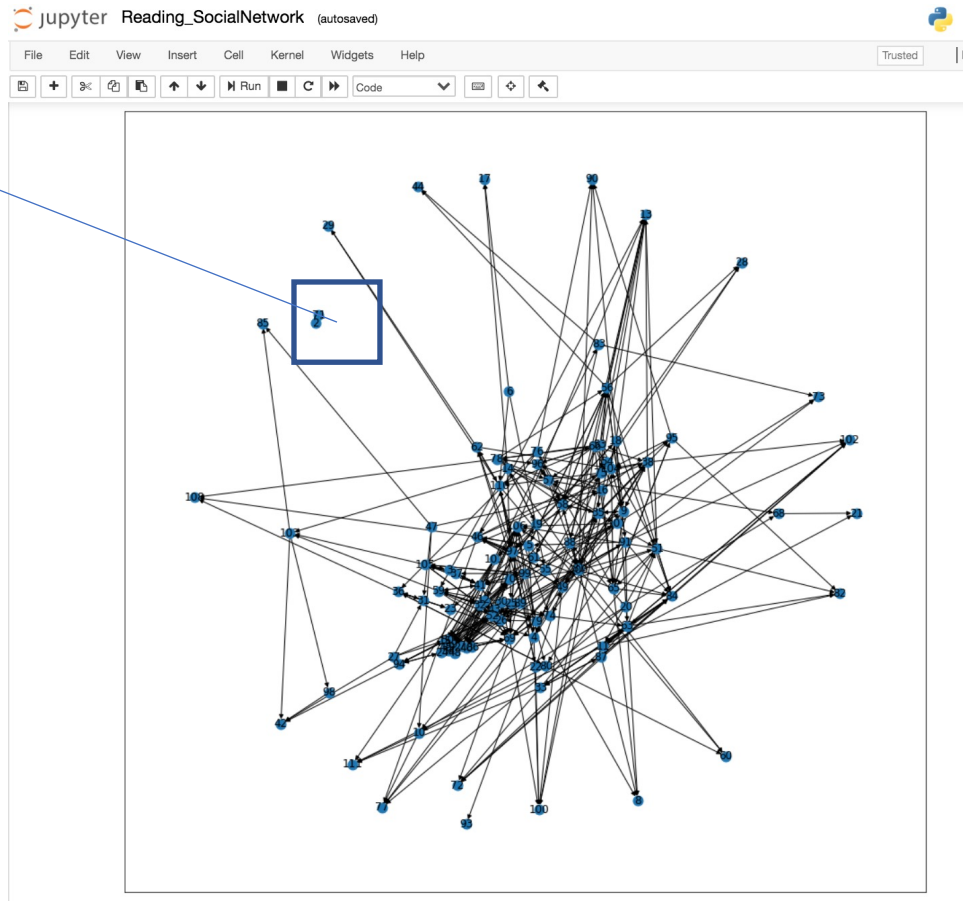
See: Example_ShortestPath_and_CC.ipynb

Connectedness and components

- A network is **connected** if there is a path between any two nodes
- If a network is not connected, it is **disconnected** and has multiple connected components
- A **connected component** is a connected subnetwork
  - The largest one is called **giant component**; it often includes a substantial portion of the network
  - A singleton is the smallest-possible connected component



Disconnected
Component
Of size 2



See: Reading_SocialNetwork.ipynb

Extract Largest Component

```
In [111]:
Gcc = sorted(nx.weakly_connected_components(g), key=len, reverse=True) # identify largest connected component of
```

```
In [112]:
len(Gcc) ## Note we have 2 components
```

```
Out[112]: 2
```

```
In [114]:
Gcc[1]
```

```
Out[114]: {2, 71}
```

```
In [115]:
Gcc[0]
```

```
Out[115]: {3,
 4,
 5,
 6,
 7,
 8,
 9,
 10,
 11,
 12,
 13,
 14,
 15,
 16,
 17,
 18,
 19,
 20,
 21,
```

```
In [116]:
G = g.subgraph(Gcc[0])
```

See: Reading_SocialNetwork.ipynb

Important, this step create a Graph G with all the nodes in the largest Connected Component of the dataset

Note you need to use nx.connected_components(g) if the graph is undirected

Class Summary

- Definitions of Degree, Clustering Coefficient, shortest path, shortest path length, and average shortest path length

- We need to extract the largest connected component if the network is disconnected

- Three python notebook scripts: MyFirstNetwork_ExerciseSoln.ipynb,

Example_ShortestPath_and_CC.ipynb and Reading_SocialNetwork.ipynb

For Lecture 3

- Do the Assignment 1_part 1, and bring questions to lecture

- Watch the Six Degree documentary: https://www.youtube.com/watch?v=2rzxAyY7D7k and participate here: https://docs.google.com/presentation/d/1gZnjE7jvMaQ39Bos6jB0iT5nz-fYeBujsAi3El3Z6Vo/edit?usp=sharing (Class participation 10%)

- Read the article **Collective dynamics of 'small-world' networks uploaded in the folder Lecture 3,** we'll go through the Participation slides of the Documentary

Office Hours

- Marta: Tuesdays 4:00-5:30 PM @Room 406c and @ Zoom

  Thursdays 9:00-11:00 by appointment via email (20 min)
  https://berkeley.zoom.us/j/4124372482

- Giuseppe: Wednesdays 11:10-1pm, @Davis 305A or
  link: https://berkeley.zoom.us/my/perona

- Martin: Thursdays 1-3pm, @Davis 305A or

https://berkeley.zoom.us/j/7628895377

- Register to the piazza site:
  https://piazza.com/berkeley/spring2023/ce88cp88

CE 88/CP 88

Assignment 1 part 1 (10 pts)

Spring 2022

Network Elements

1) Go to the data repository provided by the NetSci book by Barabasi, here: http://networksciencebook.com/translations/en/resources/data.html and select 1 dataset to create their graphs based on their lists of edges and nodes. Describe what the nodes and the links represent (0.5 pt) Should it be read as directed or undirected graph? (0.5 pt)

2) Plot the histogram of degrees of the network (0.5 pt)

3) What is the average clustering coefficient in the network? (0.5 pt)

4) What is the average shortest path in the network? (1pt)

5) What is the largest clustering coefficient in the network? (1pt)

6) List the nodes with the largest clustering coefficient (2 pts)

7) What is the value of the longest shortest path in the network? (2 pts)

8) Write the nodes in longest shortest paths of the network? (2 pts)

Properties Learned Today

In ReadingSocialNetwork.ipynb we analyze the Results of a School Survey

In ReadingSocialNetwork.ipynb we analyze the Results of a School Survey

SchoolEdges.csv

```
Source Target Type Id Label Weight
2 71 Directed 2126  1.0
3 37 Directed 2127  1.0
3 41 Directed 2128  1.0
3 105 Directed 2129  1.0
4 35 Directed 2130  2.0
4 62 Directed 2131  1.0
4 100 Directed 2132  4.0
4 110 Directed 2133  2.0
5 4 Directed 2134  5.0
5 23 Directed 2135  5.0
5 46 Directed 2136  5.0
5 49 Directed 2137  5.0
5 56 Directed 2138  3.0
6 17 Directed 2139  2.0
6 55 Directed 2140  2.0
6 110 Directed 2141  3.0
7 24 Directed 2142  2.0
7 34 Directed 2143  1.0
7 39 Directed 2144  2.0
7 40 Directed 2145  2.0
7 45 Directed 2146  2.0
7 48 Directed 2147  1.0
7 69 Directed 2148  1.0
7 92 Directed 2149  1.0
9 35 Directed 2150  3.0
9 38 Directed 2151  3.0
9 65 Directed 2152  2.0
```

Reading_SocialNetwork.ipynb

Define Graph and Read the Edges

In [2]:
```
g = nx.DiGraph()
```

In [3]:
```
fl=pd.read_csv('SchoolEdges.csv',delimiter=' ')
fl.columns = list(map(str.lower, fl.columns)) #change to lower case
```

In [4]:
```
fl.columns
```

Out[4]:
```
Index(['source', 'target', 'type', 'id', 'label', 'weight'], dtype='object')
```

In [5]:
```
fl
```

Out[5]:

	source	target	type	id	label	weight
0	2	71	Directed	2126	NaN	1.0
1	3	37	Directed	2127	NaN	1.0
2	3	41	Directed	2128	NaN	1.0
3	3	105	Directed	2129	NaN	1.0
4	4	35	Directed	2130	NaN	2.0
...	...	...	...	...	...	...
450	110	29	Directed	2579	NaN	5.0
451	110	60	Directed	2580	NaN	3.0
452	110	62	Directed	2581	NaN	5.0
453	110	80	Directed	2582	NaN	3.0
454	110	97	Directed	2583	NaN	6.0

455 rows × 6 columns

In [6]:
```
for L in fl.index.values:
        g.add_edge(fl['source'][L],fl['target'][L],weight=fl['weight'][L])
```

In [7]:
```
g.in_degree()
```

Out[7]:
```
InDegreeView({2: 1, 71: 1, 3: 3, 37: 4, 41: 7, 105: 4, 4: 4, 35: 5, 62: 2, 100: 6, 110: 6, 5: 3, 23: 3, 46: 5, 49: 5,
56: 5, 6: 0, 17: 2, 55: 2, 7: 17, 24: 12, 34: 8, 39: 9, 40: 18, 45: 14, 48: 11, 69: 12, 92: 12, 9: 6, 38: 6, 65: 6, 1
04: 7, 10: 3, 77: 5, 11: 2, 21: 2, 63: 5, 82: 4, 84: 10, 87: 3, 102: 6, 111: 4, 12: 1, 19: 2, 50: 6, 70: 2, 14: 1, 2
5: 1, 59: 4, 68: 2, 81: 2, 103: 3, 107: 2, 15: 7, 16: 2, 57: 5, 95: 3, 96: 4, 18: 2, 8: 4, 13: 10, 28: 3, 90: 4, 97:
7, 101: 1, 20: 0, 88: 2, 22: 1, 36: 4, 94: 4, 26: 11, 43: 3, 27: 0, 31: 3, 42: 4, 52: 7, 30: 0, 32: 7, 74: 4, 79: 4,
106: 9, 33: 2, 51: 6, 72: 5, 58: 9, 109: 4, 80: 3, 108: 3, 75: 6, 99: 2, 47: 0, 85: 2, 91: 3, 53: 1, 54: 1, 44: 2, 6
1: 1, 29: 2, 66: 1, 78: 2, 76: 0, 60: 3, 83: 1, 73: 3, 86: 0, 93: 1, 89: 0, 98: 1})
```

Lab (Group Exercise)

1- Calculate the Clustering Coefficient of each node and of the School friendship network

2- Calculate the Clustering Coefficient of of the School friendship network

3-Calculate the average shortest path of the School friendship network

Use Example_ShortestPath_and_CC.ipynb and Reading_SocialNetwork.ipynb

Add your name and lastname and answer here:
https://docs.google.com/presentation/d/10VYRXs5Xg44CfxBK3x42Lsf-vpBfQteu4S50nm33FaE/edit#slide=id.g1fdb202be52_0_0