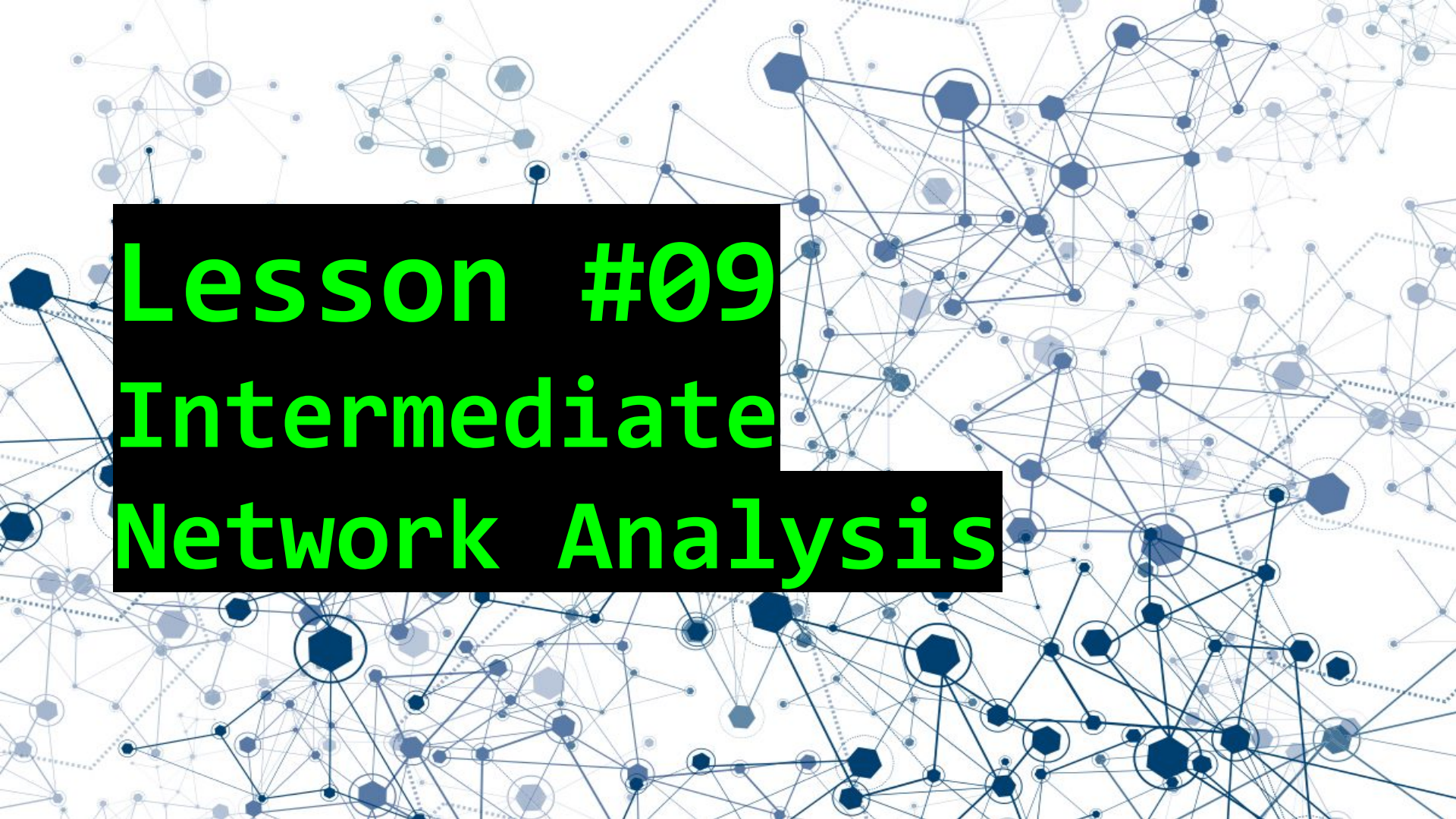
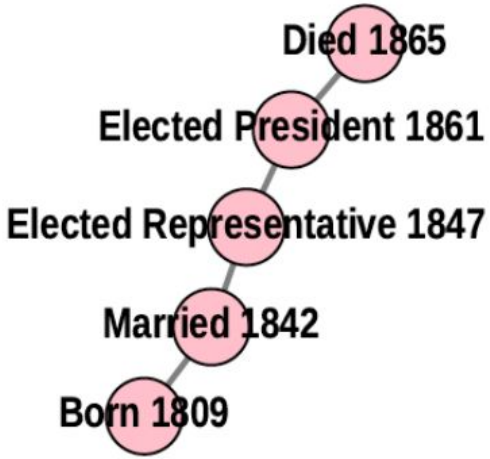

Lesson #09
Intermediate
Network Analysis

notas

- Create Networks from Adjacency and Incidence Matrices
- Generate Synthetic Networks
- Measuring Networks
  - Global measures
  - Explore neighborhoods
  - Think in terms of paths
  - Choose the right centralities

notas

Create Networks from Adjacency and Incidence Matrices



$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & -1 \end{bmatrix}$$

Incidence Matrix

```
0 1 0 0 0   0 1 0 0 0
0 0 1 0 0   0 0 1 0 0
0 0 0 1 0   0 0 0 1 0
0 0 0 0 1   0 0 0 0 1
0 0 0 0 0   1 0 0 0 0
```

Adjacency Matrix

Adjacency Matrix

- Python way
- Numpy way
- Pandas way

Network type	Adjacency matrix	Interpretation
Simple	Only 0s or 1s, and no 1s on the main diagonal	Absence/presence of an edge
Weighted	At least one floating-point number	Numbers are edge weights
With self-loops	At least one non-zero on the main diagonal	Same as above
Signed	At least one negative number	Same as above
Undirected	Symmetric	Same as above
Multigraph	Not possible	Cannot be represented as an adjacency matrix

Adjacency Matrix

- Python way

```
A = [[0, 1, 0, 0, 0],
     [0, 0, 1, 0, 0],
     [0, 0, 0, 1, 0],
     [0, 0, 0, 0, 1],
     [.1, 0, 0, 0, 0]]
```

```
# directed graph
G = nx.DiGraph()

# nested list comprehension
edges = [(i,j) for i,row in enumerate(A)
               for j,column in enumerate(row) if A[i][j]]

# add edges
G.add_edges_from(edges)
```

```
[(0, 1, {}), (1, 2, {}), (2, 3, {}), (3, 4, {}), (4, 0, {})]
```

Adjacency Matrix

- Python way

```
A = [[0, 1, 0, 0, 0],
     [0, 0, 1, 0, 0],
     [0, 0, 0, 1, 0],
     [0, 0, 0, 0, 1],
     [.1, 0, 0, 0, 0]]
```

```
# directed graph
G = nx.DiGraph()

# nested list comprehension
edges = [(i,j, {"weight": A[i][j]}) for i,row in enumerate(A)
         for j,column in enumerate(row) if A[i][j]]

# add edges
G.add_edges_from(edges)
```

```
[(0, 1, {'weight': 1}), (1, 2, {'weight': 1}), (2, 3, {'weight': 1}),
 (3, 4, {'weight': 1}), (4, 0, {'weight': 0.1})]
```

Adjacency Matrix

- Numpy way

```
A = [[0, 1, 0, 0, 0],
     [0, 0, 1, 0, 0],
     [0, 0, 0, 1, 0],
     [0, 0, 0, 0, 1],
     [.1, 0, 0, 0, 0]]
```

```
import numpy as np

A_mtx = np.matrix(A)
G = nx.from_numpy_matrix(A_mtx,
                         create_using=nx.DiGraph())
```

```
[(0, 1, {'weight': 1.0}), (1, 2, {'weight': 1.0}),
 (2, 3, {'weight': 1.0}), (3, 4, {'weight': 1.0}), (4, 0, {'weight': 0.1})]
```

Adjacency Matrix

- Pandas way

	Born	Married	Elected Rep	Elected Pres	Died
Born	0.0	1.0	0.0	0.0	0.0
Married	0.0	0.0	1.0	0.0	0.0
Elected Rep	0.0	0.0	0.0	1.0	0.0
Elected Pres	0.0	0.0	0.0	0.0	1.0
Died	0.1	0.0	0.0	0.0	0.0

```
labels = "Born", "Married", "Elected Rep", "Elected Pres", "Died"

# change nodes label inplace
nx.relabel_nodes(G, dict(enumerate(labels)), copy=False)

# create a dataframe from a networkx graph
df = nx.to_pandas_adjacency(G)
```

```
edges = nx.to_pandas_edgelist(G)
```

	source	target	weight
0	Born	Married	1.0
1	Married	Elected Rep	1.0
2	Elected Rep	Elected Pres	1.0
3	Elected Pres	Died	1.0
4	Died	Born	0.1

Adjacency Matrix

- Pandas way

```
G = nx.from_pandas_edgelist(edges,
                            *edges.columns,
                            create_using=nx.DiGraph())
```

Incidence Matrices


```
J = nx.incidence_matrix(G, oriented=True).todense()
```

```
matrix([[-1., -1.,  0.,  0.,  0.],
        [ 1.,  0., -1.,  0.,  0.],
        [ 0.,  0.,  0., -1., -1.],
        [ 0.,  1.,  0.,  1.,  0.],
        [ 0.,  0.,  1.,  0.,  1.]])
```

```
import sys
print(sys.getsizeof(nx.incidence_matrix(G, oriented=True)))
print(sys.getsizeof(nx.incidence_matrix(G, oriented=True).todense()))
```

```
56
136
```

- SNAP for C++
- SNAP for Python
- SNAP Datasets
- BIOSNAP Datasets
- What's new
- People
- Papers
- Projects
- Citing SNAP
- Links
- About
- Contact us

Open positions

Open research positions in **SNAP** group are available at **undergraduate, graduate** and **postdoctoral** levels.

Enron email network

Dataset information

Enron email communication network covers all the email communication within a dataset of around half million emails. This data was originally made public, and posted to the web, by the Federal Energy Regulatory Commission during its investigation. Nodes of the network are email addresses and if an address *i* sent at least one email to address *j*, the graph contains an undirected edge from *i* to *j*. Note that non-Enron email addresses act as sinks and sources in the network as we only observe their communication with the Enron email addresses.

The Enron email data was originally released by William Cohen at CMU.

Dataset statistics	
Nodes	36692
Edges	183831
Nodes in largest WCC	33696 (0.918)
Edges in largest WCC	180811 (0.984)
Nodes in largest SCC	33696 (0.918)
Edges in largest SCC	180811 (0.984)
Average clustering coefficient	0.4970
Number of triangles	727044
Fraction of closed triangles	0.03015
Diameter (longest shortest path)	11
90-percentile effective diameter	4.8

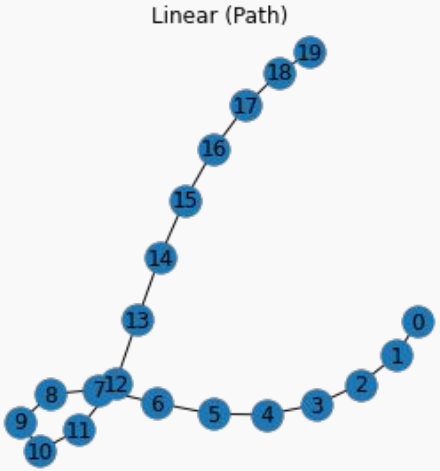
Exercise 1.1.3

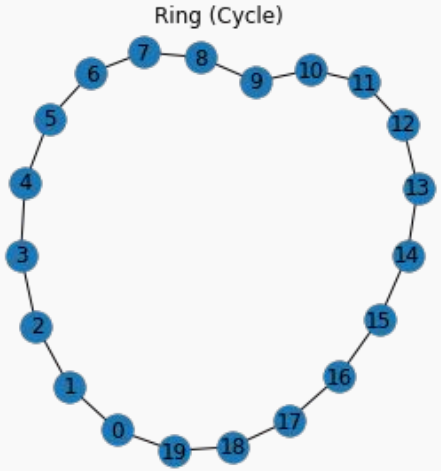
Source (citation)

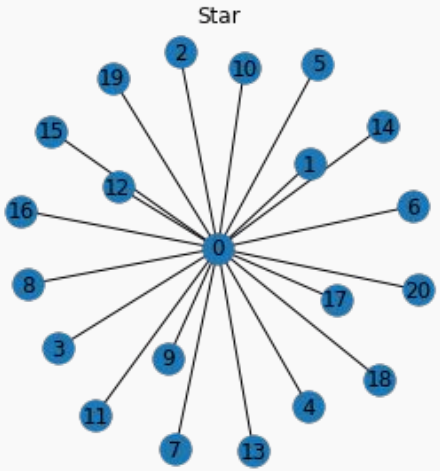
- J. Leskovec, K. Lang, A. Dasgupta, M. Mahoney. Community Structure in Large Networks: Natural Cluster Sizes and the Absence of Large Well-Defined Clusters. Internet Mathematics 6(1) 29--123, 2009.

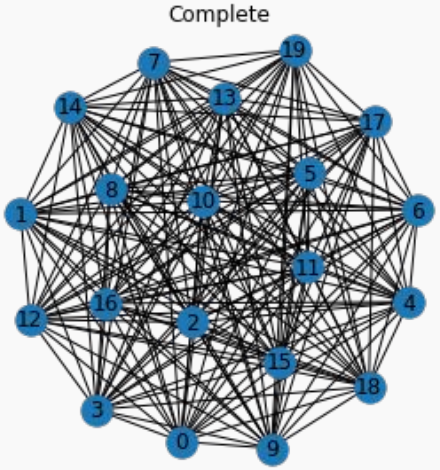
Generate Synthetic Networks

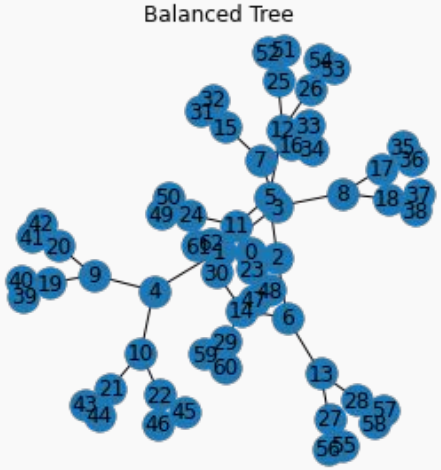
Classic Networks

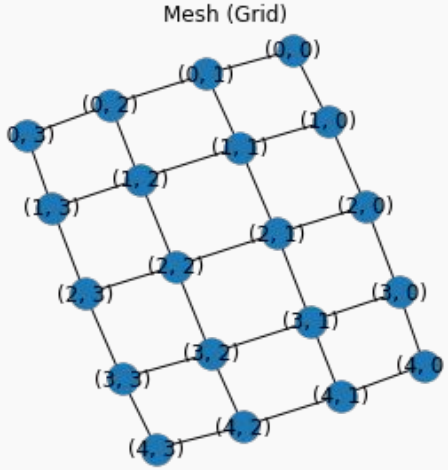












```
import matplotlib.pyplot as plt
import networkx as nx

fig, plot = plt.subplots(2, 3,figsize=(15,10))
subplots = plot.reshape(1, 6)[0]

# Generate and draw classic networks
G0 = nx.path_graph(20)
G1 = nx.cycle_graph(20)
G2 = nx.star_graph(20)
G3 = nx.complete_graph(20)
G4 = nx.balanced_tree(2, 5)
G5 = nx.grid_2d_graph(5, 4)

graphs = [G0,G1,G2,G3,G4,G5]

names = ["Linear (Path)", "Ring (Cycle)",
         "Star", "Complete", "Balanced Tree", "Mesh (Grid)"]

for graph,title,plot in zip(graphs,names,subplots):
  nx.draw_networkx(graph, ax=plot)
  plot.set_title(title)
  plot.axis("off")
```

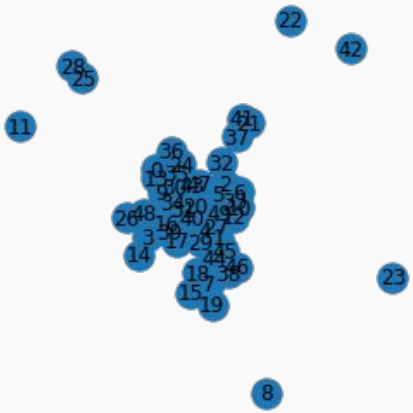
Random Graphs

- **Erdös–Rényi**
  a. "binomial graph", contains (N-1)/2 nodes, but each edges occur with a probability P.
- **Watts–Strogatz**
  a. "Six degree of separation", create a illusion of small world.
- **Barabási–Albert**
  a. principle of preferential attachment, "celebrity" nodes
- **Holme–Kim**
  a. the same as above, plus the probability of adding a triangle for each added edge, making the synthetic network even more clustered and lifelike.

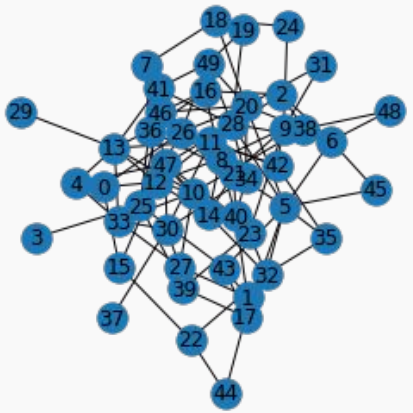
Erdos (n,p)

erdos(50,0)

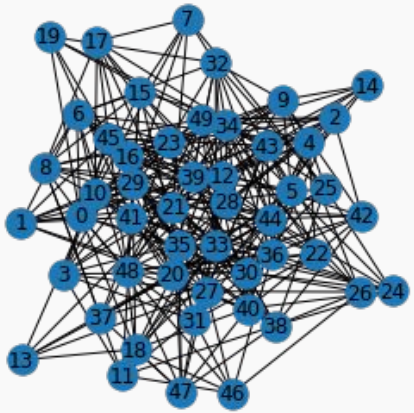
erdos(50,0.05)



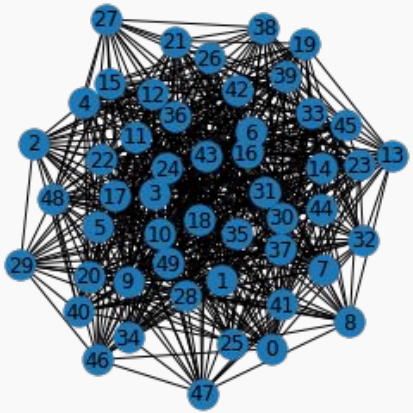
erdos(50,0.1)



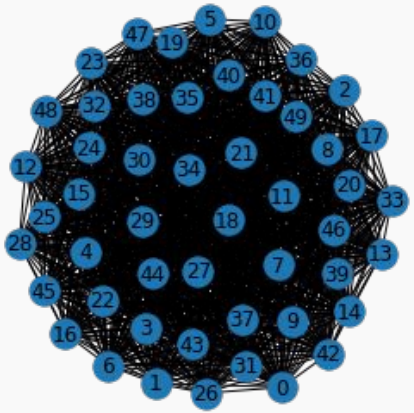
erdos(50,0.25)



erdos(50,0.5)



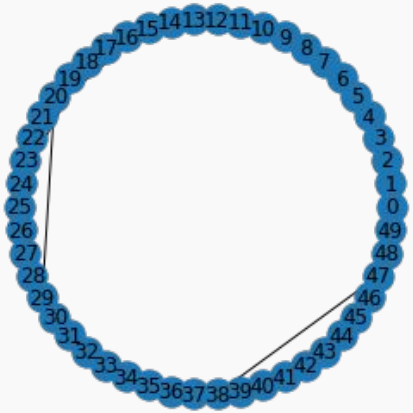
erdos(50,1)



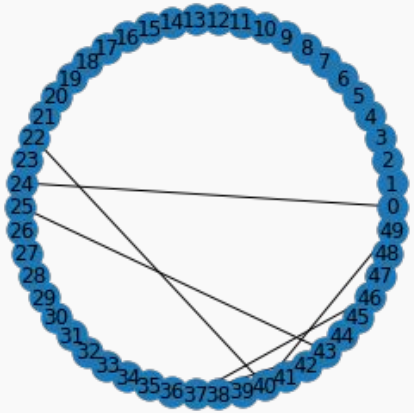
Watts Strogatz (n,k,p)

watts_strogatz(50,4,0)

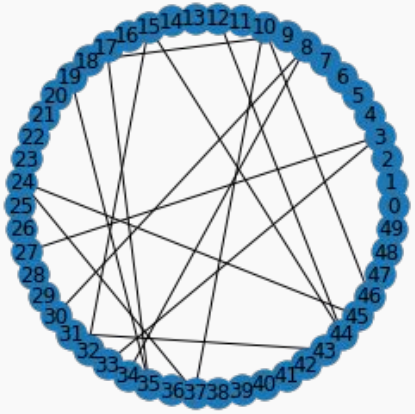
watts_strogatz(50,4,0.05)



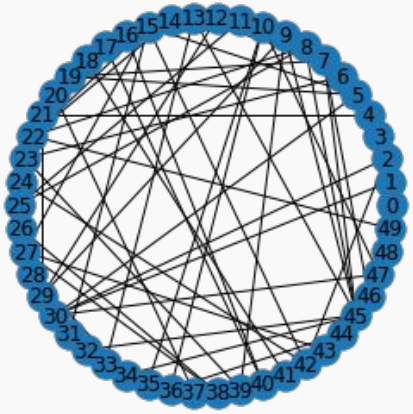
watts_strogatz(50,4,0.1)



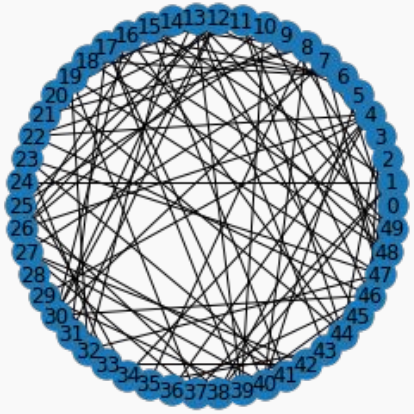
watts_strogatz(50,4,0.25)



watts_strogatz(50,4,0.5)

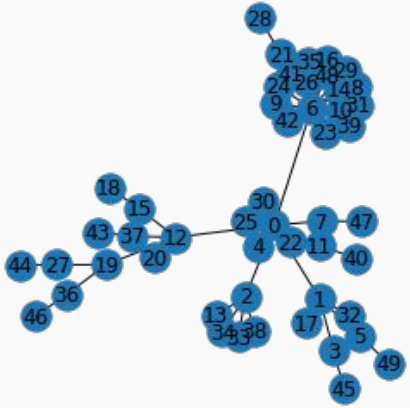


watts_strogatz(50,4,1)

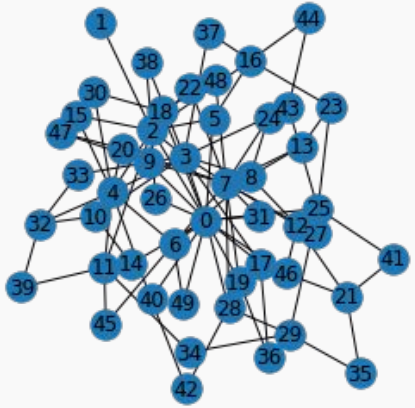


Barabasi Albert (n,k)

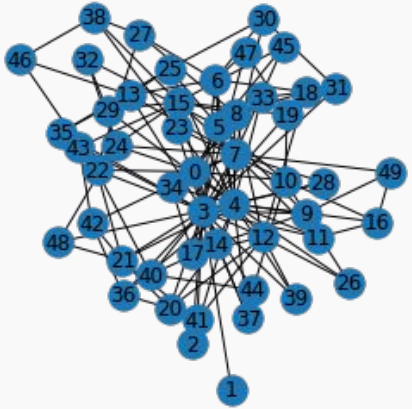
barabasi(50,1)



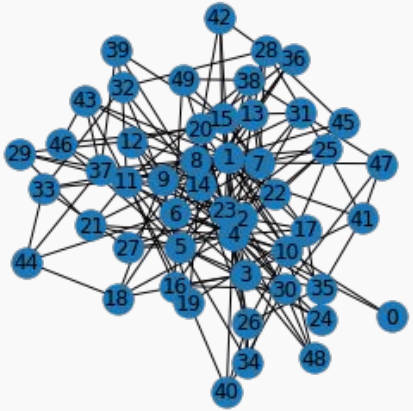
barabasi(50,2)



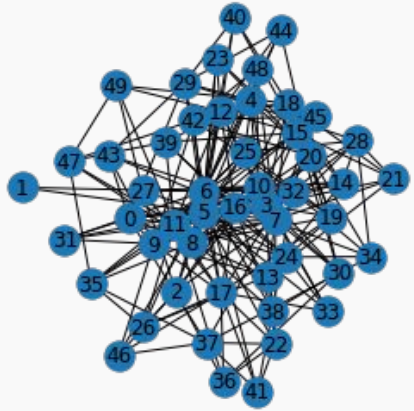
barabasi(50,3)



barabasi(50,4)



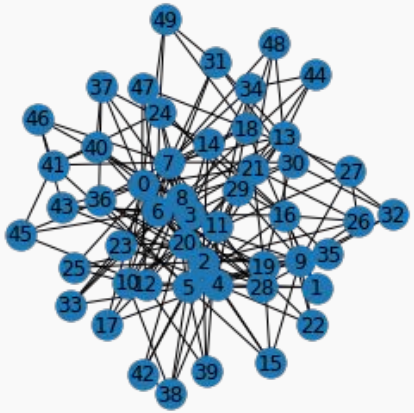
barabasi(50,5)



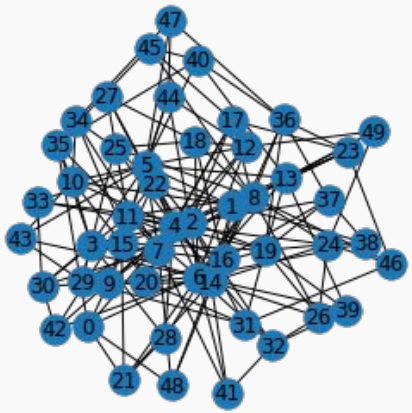
barabasi(50,10)

Holme Kim (n,k,p)

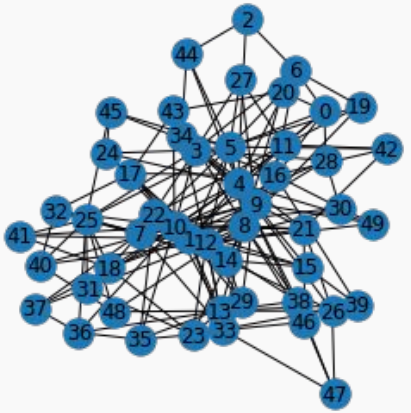
holme(50,4,0)



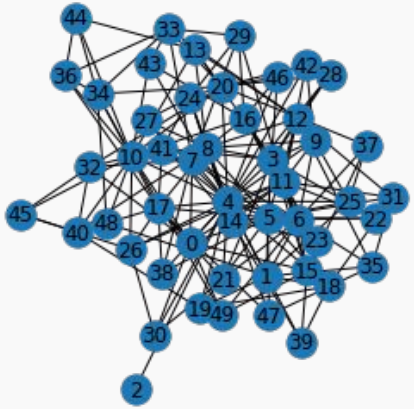
holme(50,4,0.05)



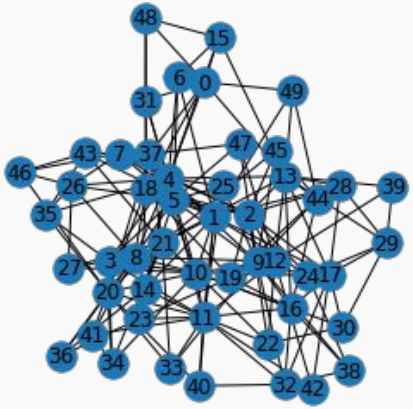
holme(50,4,0.1)



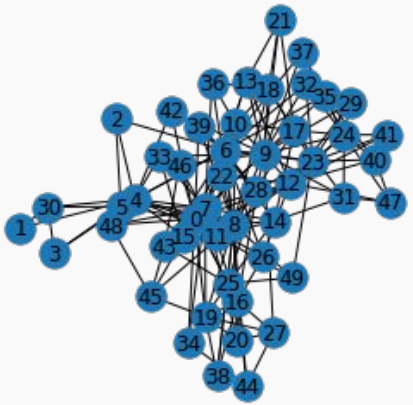
holme(50,4,0.25)



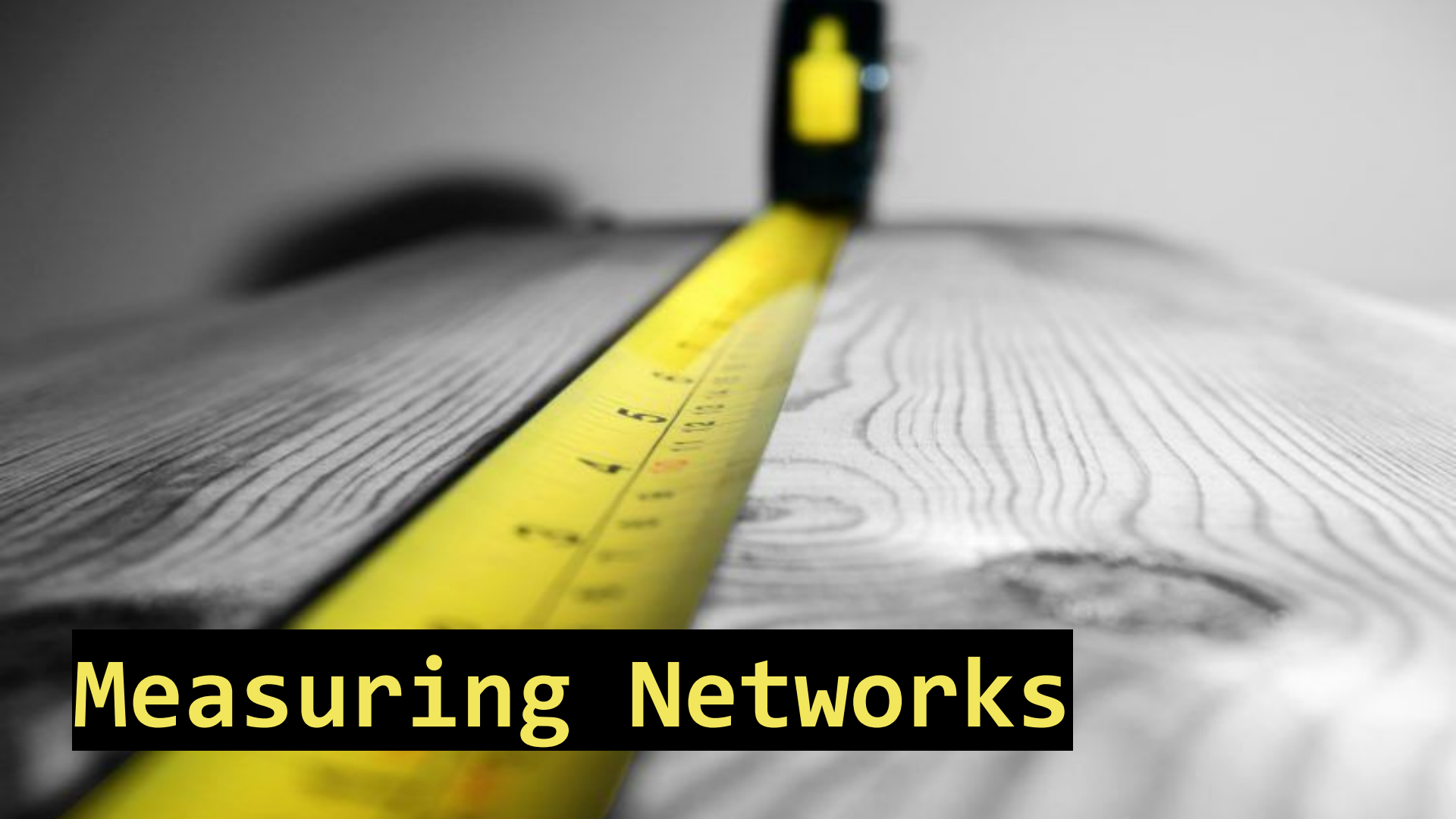
holme(50,4,0.5)



holme(50,4,1)



Experimentation: Section 1.2


Measuring Networks

Characterization of Complex Networks: A Survey of measurements

L. da F. Costa F. A. Rodrigues G. Travieso
P. R. Villas Boas
Instituto de Física de São Carlos, Universidade de São Paulo
Caixa Postal 369, 13560-970, São Carlos, SP, Brazil
luciano@ifsc.usp.br

February 2, 2008

Abstract

Each complex network (or class of networks) presents specific topological features which characterize its connectivity and highly influence the dynamics of processes executed on the network. The analysis, discrimination, and synthesis of complex networks therefore rely on the use of measurements capable of expressing the most relevant topological features. This article presents a survey of such measurements. It includes general considerations about complex network characterization, a brief review of the principal models, and the presentation of the main existing measurements. Important related issues covered in this work comprise the representation of the evolution of complex networks in terms of trajectories in several measurement spaces, the analysis of the correlations between some of the most traditional measurements, perturbation analysis, as well as the use of multivariate statistics for feature selection and network classification. Depending on the network

Start with global measures

```
import networkx as nx
G = nx.read_graphml("cna.graphml")

nx.number_of_nodes(G)
3252

nx.number_of_edges(G)
14394

nx.density(G)
0.0013614885456759828
```

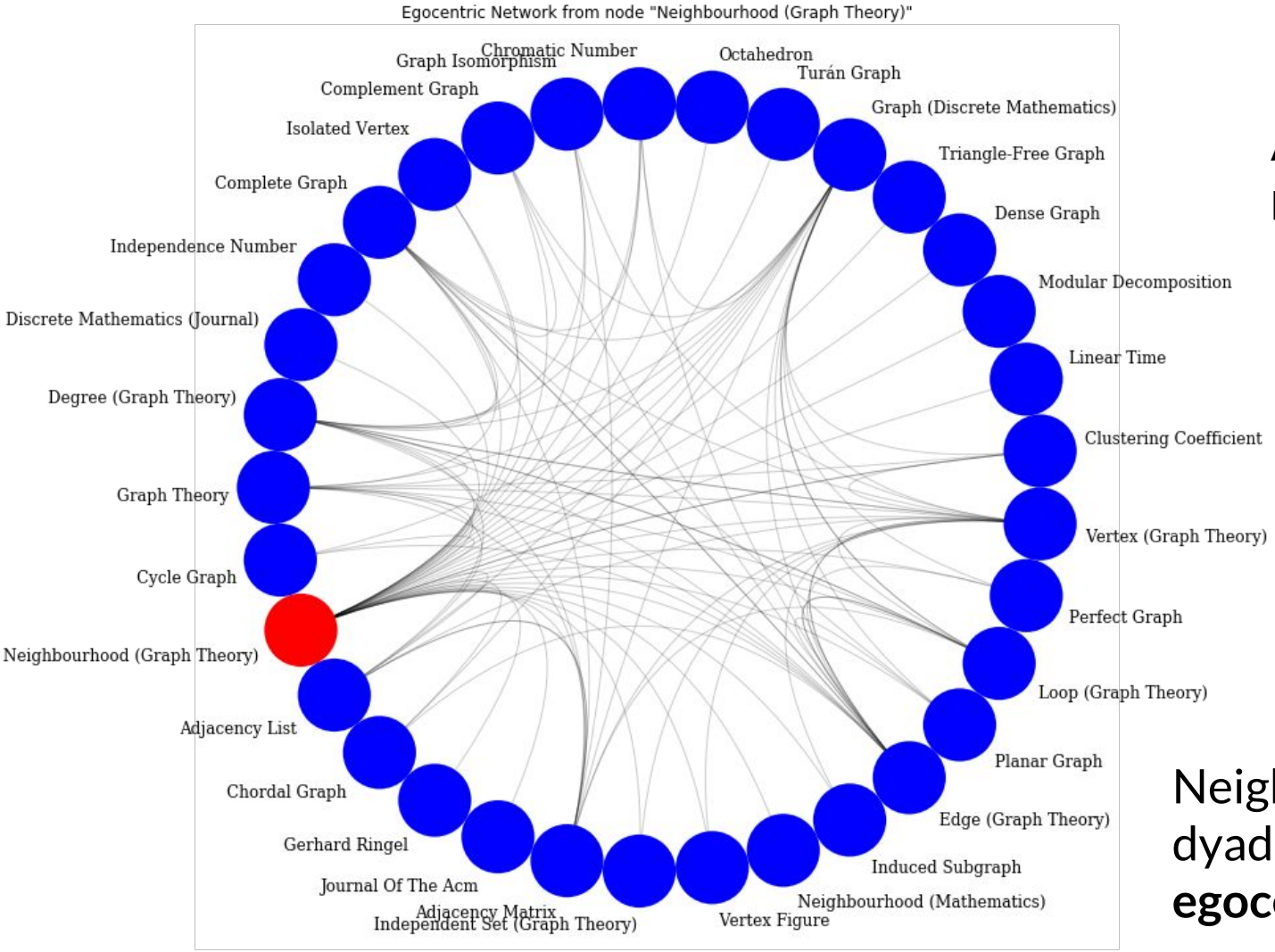
Explore neighborhoods

Node and edge counts and density are some of the **macroscopic network properties**.

Neighborhoods are responsible for the local properties of network graphs (**microscopic level**)

- Egocentric network
- Clustering coefficient

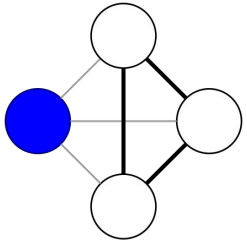
```
ego = "Neighbourhood (Graph Theory)"

# out nodes
alters_1 = G[ego] #32

alters_2 = list(nx.all_neighbors(G, ego)) #75
nx.degree(G,ego) #75
```



A social network of a particular individual.

Neighborhood as a dyadic structure: **egocentric network**

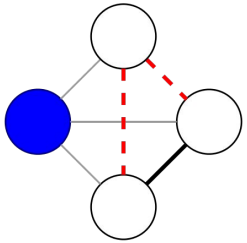
Clustering Coefficient



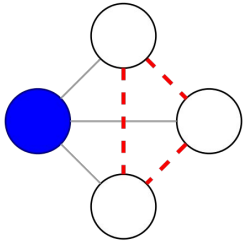
c = 1

Some social theories consider **triads** essential units of social network analysis.

- the **clustering coefficient** is the fraction of possible triangles that contain the ego
- Think of the clustering coefficient as a measure of "**stardom.**"



c = 1/3

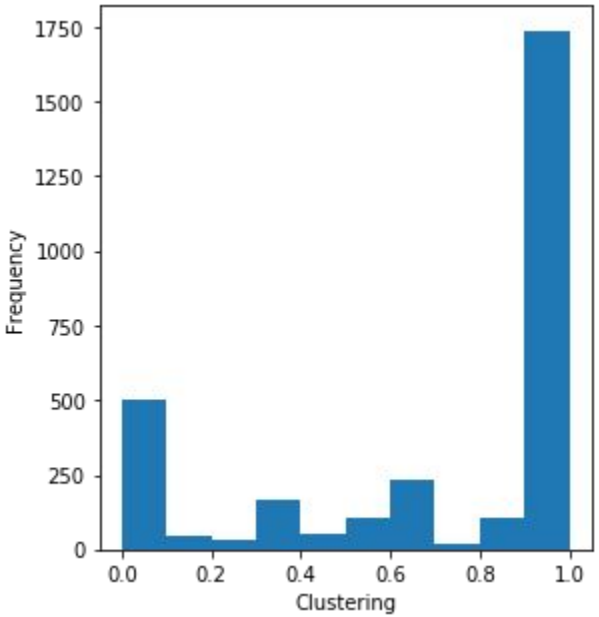


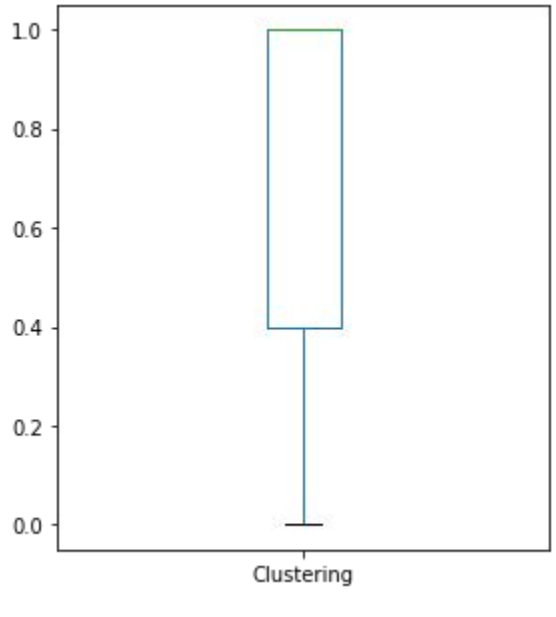
c = 0

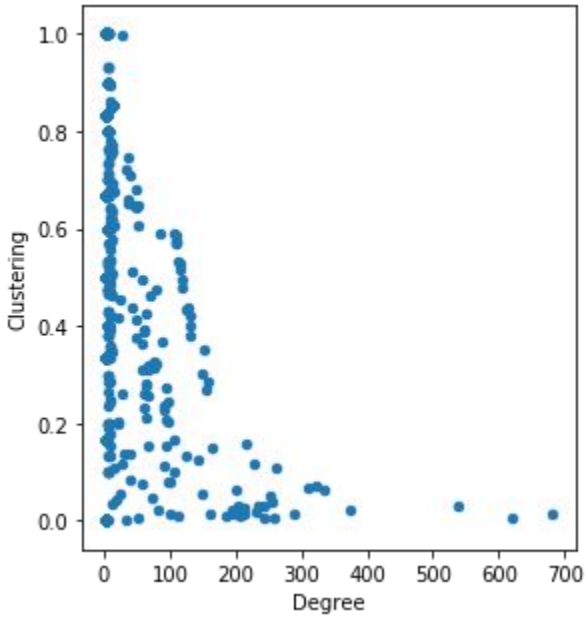
```
cc = nx.clustering(nx.Graph(G), ego)
0.4866818873668189
```

Clustering Coefficient

cna.graphml

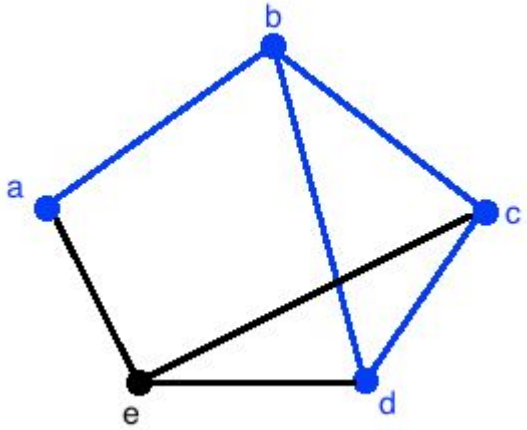




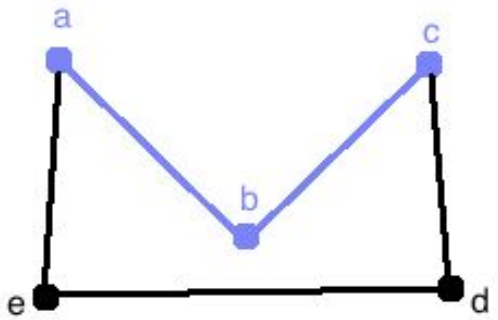


Exercise 2.1, 2.2

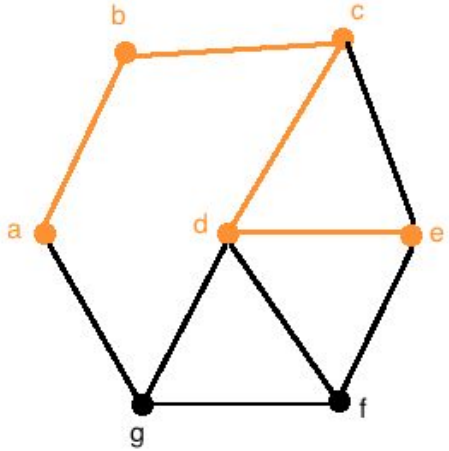
Think in terms of path



Walk
abcdb
abcdcbce
aecbde



Trail
abc
aedc
abcdea



Path
abcde
agdef

The shortest paths are called **geodesics**

```
path_gen = nx.shortest_simple_paths(G, ego, "Webgraph")
next(path_gen)

['Neighbourhood (Graph Theory)',
 'Clustering Coefficient',
 'Scale-Free Network',
 'Webgraph']
```

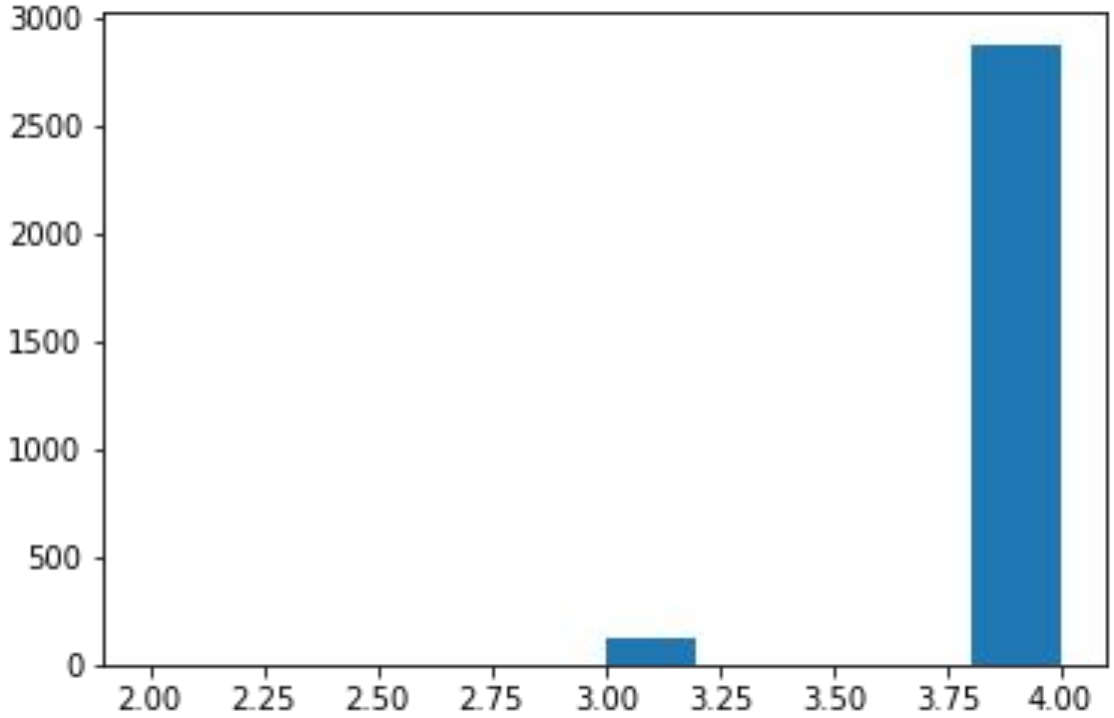
Network as a circle

The **eccentricity** is the maximum distance from a node to all other nodes in the network

```
ecc = nx.eccentricity(nx.Graph(G))
ecc[ego]  #3
```

```
# diameter = max(ecc)
ecc[max(ecc,key=ecc.get)] #4

# radius = min(ecc)
ecc[min(ecc,key=ecc.get)] #2

# center => set of nodes with ecc equal to radius
[i for i in ecc if ecc[i] == ecc[min(ecc,key=ecc.get)]]
['Complex Network']

# periphery => set of nodes with ecc equal to diameter
len([i for i in ecc if ecc[i] == ecc[max(ecc,key=ecc.get)]]) #2875
```

```
Counter({2: 1, 3: 119, 4: 2875})
```



Eccentricity

Choose the Right Centralities

One of the goals of social network analysis is to identify actors with outstanding properties (**most important**)

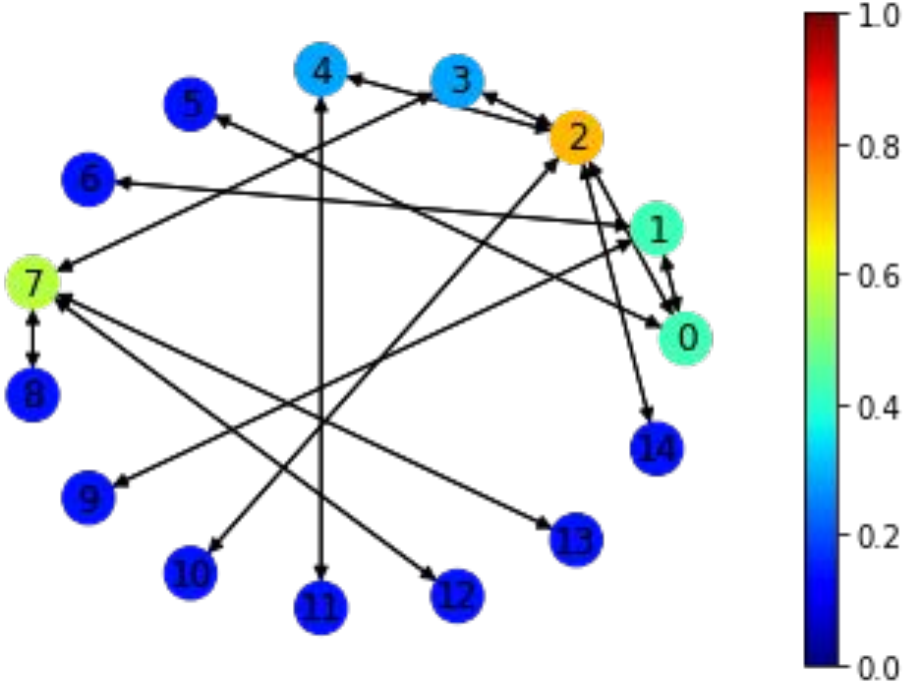
- Degree centrality
- Closeness Centrality
- Betweenness Centrality
- Eigenvector Centrality

Ulrik Brandes

The Space of All Centralities

https://www.youtube.com/watch?v=ND9b8QhkXxU

Degree Centrality



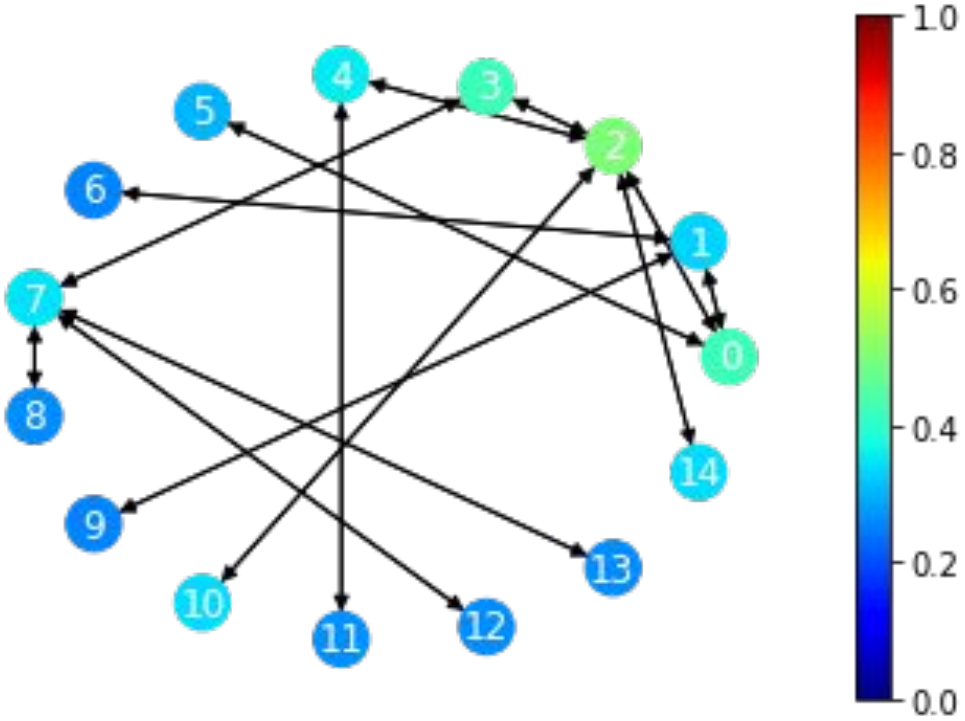
Degree is a simple centrality measure that counts how many neighbors a node has.

If the network is directed:

- in-degree
- out-degree

$$C(node) = \frac{N-1}{\sum_{1}^{N-1} D(node,v)}$$

Closeness Centrality



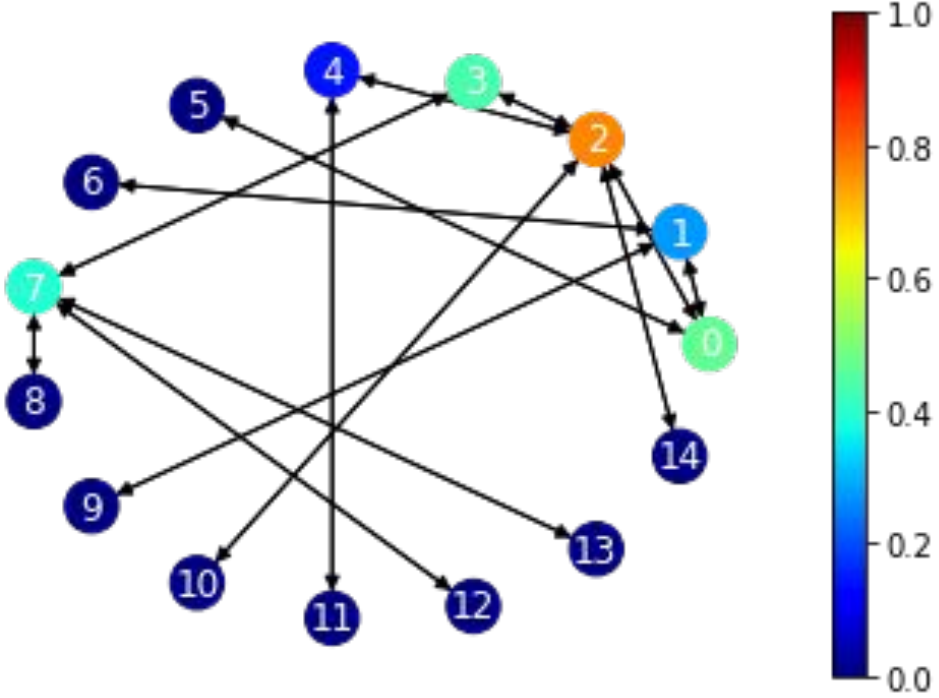
It shows how close the node is to the rest of the graph.

- 0 (the node has no neighbors)
- 1 (the node is the hub of the global star and is one hop away from any other node)

A node with a high closeness centrality may be capable of affecting the entire network.

Betweenness Centrality

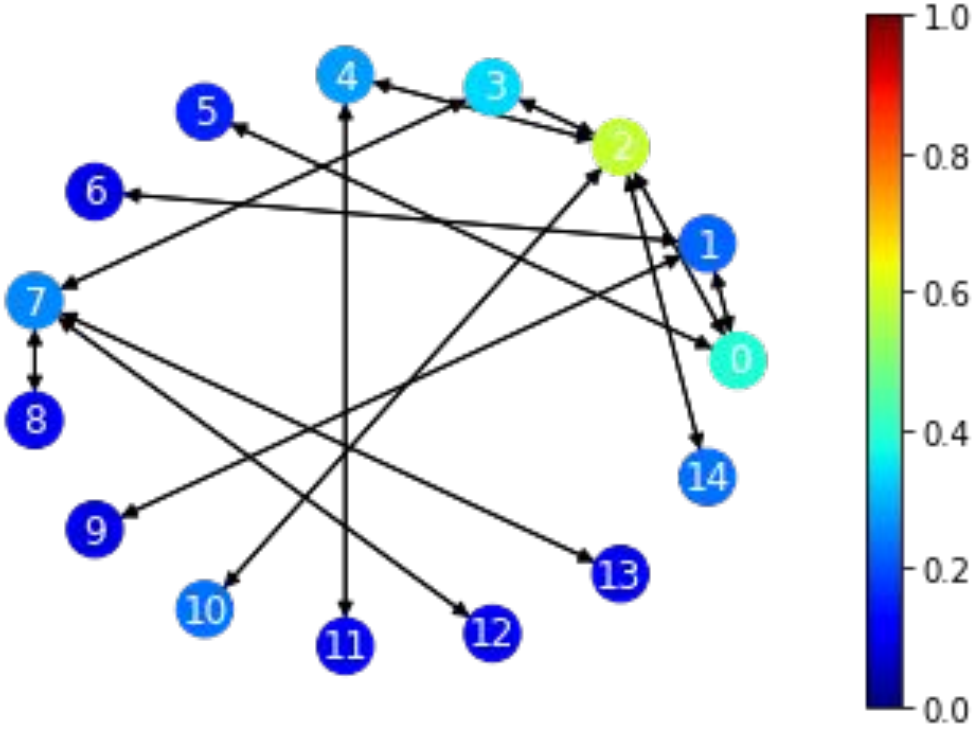
$$betweenness(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$$



It measures the fraction of all possible geodesics that pass through a node.

If the betweenness is high, the node is potentially a crucial go-between (thus the name) and has a **brokerage capability**.

Eigenvector Centrality

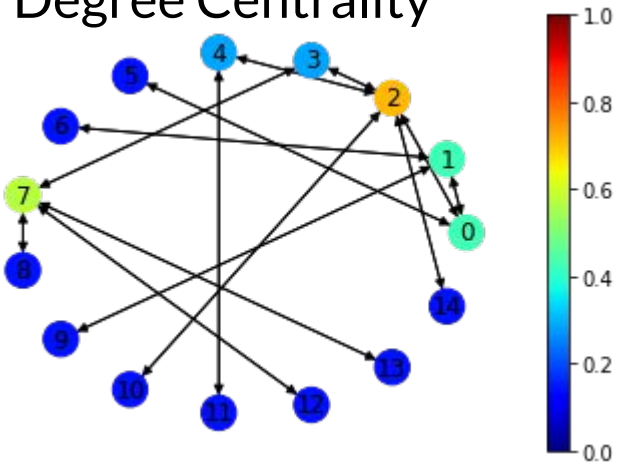


"Tell me who your friends are, and I will tell you who you are."

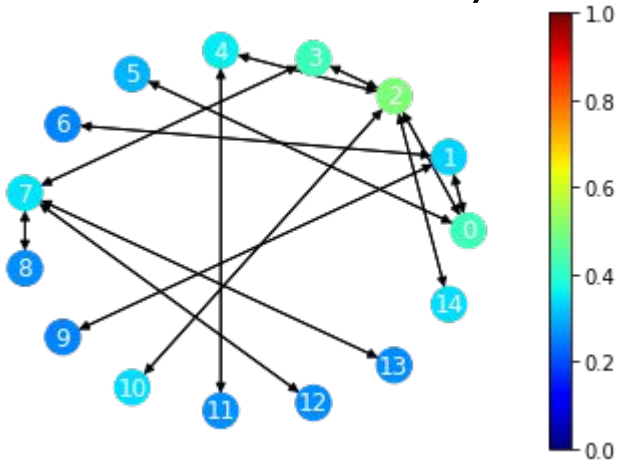
You can use this measure to locate groups of interconnected nodes with high prestige.

A natural extension of degree centrality is **eigenvector centrality**.

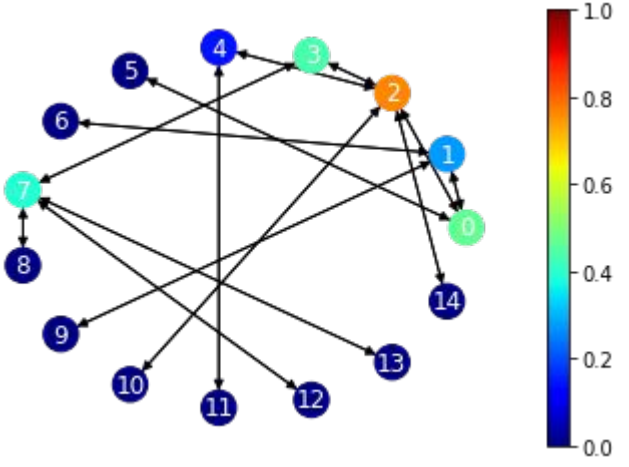
Degree Centrality



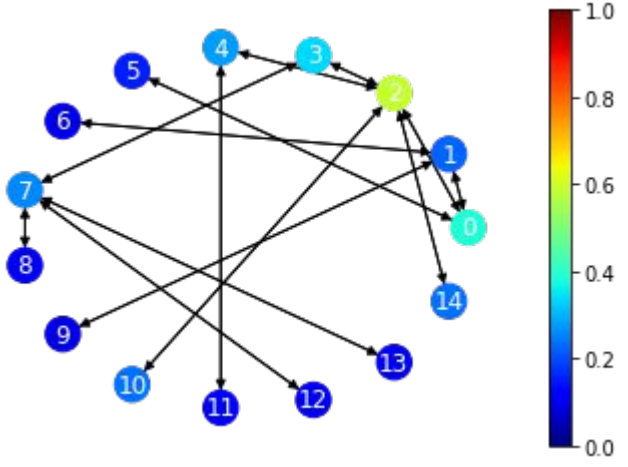
Closeness Centrality



Betweenness Centrality



Eigenvector Centrality



Exercise 2.3, 2.4