OBJECTIVES

Measures & Metrics

- Path
- Distance
- Diameter

PATH (1/4)

DEFINITION

Let $G = (V, E)$ be a graph. We say a (non-empty) sequence of vertices $\gamma = (v_1, v_2, ..., v_r, v_{r+1})$ in V is a path or walk if $(v_i, v_{i+1}) \in E$ for $1 \leq i \leq r$.

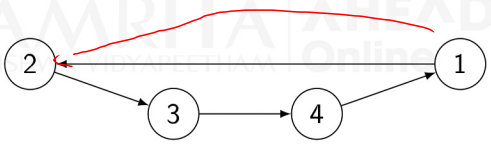
- The length of the path is $len(\gamma) := r$. We say v_1 is the start vertex and $v_r + 1$ is the end vertex of γ.
- If $v_i \neq v_j$ for $1 \leq i \neq j \leq r+1$, we say the path is simple.
- If $v_{r+1} = v_1$, we say γ is closed. If $\gamma = (v_1, ..., v_r, v_1)$ is a closed path with $v_i \neq v_j$ for $1 \leq i \neq j \leq r$, we say γ is a (simple) cycle or circuit.

PATH (2/4)

EXAMPLE

Let n > 2. A cycle graph of order n is a graph of the form $G = (V, E)$ where $V = \{v_1, v_2, ..., v_n\}$ and $E = \{\{v_1, v_2\}, \{v_2, v_3\}, ..., \{v_{n-1}, v_n\}, \{v_n, v_1\}\}$. Note that $\{v_i, v_j\}$ means an undirected edge, as opposed to (v_i, v_j).

- Here is a cycle graph of order 4.




```
G=nx.DiGraph()
G_Cycle=nx.cycle_graph(4, create_using=G)
nx.draw_networkx(G_Cycle)
```

Path (3/4)

Example

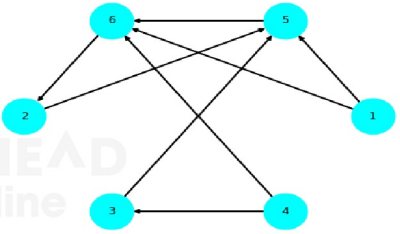
A linear graph (or path graph) of order n, is a graph of the form $G = (V, E)$ where $V = \{v_1, v_2, ..., v_n\}$ and $E = \{\{v_1, v_2\}, \{v_2, v_3\}, ..., \{v_{n-1}, v_n\}, \{v_n, v_1\}\}$. Here is a line graph on 4 vertices.


```
G1=nx.DiGraph()
G_Path=nx.path_graph(4, create_using=G1)
nx.draw_networkx(G_Path)
```

PATH (4/4)

Finding Paths

```
def findPaths(G,u,n):
    if n==0:
        return [[u]]
Stmt:1
paths=findPaths(G1,1,3)
print(paths)
[[1, 5, 6, 2], [1, 6, 2, 5]]
```




```
Stmt:1
paths = [[u]+path for neighbor in G.neighbors(u) for path in
findPaths(G,neighbor,n-1)  if u not in path]
```

DISTANCE, DIAMETER (1/6)

DEFINITION (DISTANCE)

Let $G = (V, E)$ be a graph. For $u, v \in V$, let $\Gamma(u, v)$ denote the set of paths from u to v. We define the distance $d(u, v)$ between u and v to be

$$d(u,v) = \begin{cases} \infty; \textit{ there is no path from } u \textit{ to } v; \\ min\{len(\gamma) : \gamma \in \Gamma(u,v)\}; \textit{ else} \end{cases}$$

- In other words, the distance between two vertices is the least number of steps (edges) it takes to get from one to the other.
- The vertices which are distance 1 from u are the neighbors of u.

Distance, Diameter (2/6)

Definition (Eccentricity)

Let $G = (V, E)$ be a graph (directed or not). The eccentricity of node v_1, denoted $ecc(G, v_1)$, is the maximum distance $d(v_1, v)$ for $v \in V$.

- The eccentricity of a node v is the maximum distance from v to all other nodes in G
- The eccentricity is finite if and only if G is connected and undirected, or G is strongly connected and directed.

Definition (Diameter)

Let $G = (V, E)$ be a graph (directed or not). The diameter of G, denoted $diam(G)$, is the maximum distance $d(u, v)$ for $u, v \in V$.

- The diameter is the maximum eccentricity
- The diameter is finite if and only if G is connected and undirected, or G is strongly connected and directed.

Distance, Diameter (3/6)

- The diameter provides an upper bound on the time it takes to get between two points in the graph. Thus smaller diameters indicate higher efficiencies for graphs.
- Diameter is a measure of *how connected* a graph is

Definition (Average Distance)

For a directed or undirected graph G = (V, E), define the average distance on G to be

$$d_{avg}(G) := \frac{1}{n(n-1)} \sum_{u \in V} \sum_{v \in V, u \neq v} d(u, v)$$

DISTANCE, DIAMETER (4/6)

EXAMPLE

- For the cyclic graph C n of order n, we have $diam(C_n) = \lfloor \frac{n}{2} \rfloor$
- The diameter of the directed cyclic graph of order n is $n-1$.
- The diameter of a linear graph of order n is $n-1$.
- The diameter of a directed linear graph is ∞.
- The diameter of the complete graph K_n is $diam(K_n) = 1$.

```
G = nx.DiGraph([(1, 2),(1, 3), (1, 4), (3, 4), (3, 5),
(4, 5),(5,1),(2,5),(3,5), (5,6), (6,2)])
print(nx.shortest_path(G, source=1, target=6))
print(nx.shortest_path_length(G, source=1, target=6))
[1, 3, 5, 6]
3
```

DISTANCE, DIAMETER (5/6)

```
G = nx.Graph([(1, 2), (1, 3), (1, 4), (3, 4), (3, 5), (4, 5)])
nx.eccentricity(G)
{1: 2, 2: 3, 3: 2, 4: 2, 5: 3}
nx.eccentricity(G, v=[1])
```

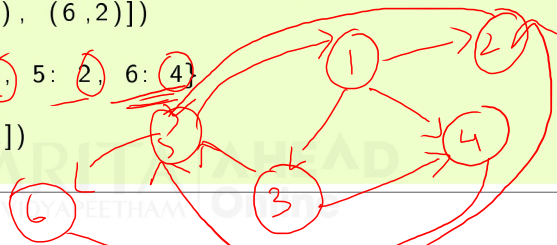
```
G_Cycle=nx.cycle_graph(25)
nx.diameter(G_Cycle)
12
```

```
G = nx.Graph([(1, 2), (1, 3), (1, 4), (3, 4), (3, 5), (4, 5)])
nx.diameter(G)
3
```

DISTANCE, DIAMETER (6/6)

```
G = nx.DiGraph([(1, 2),(1, 3),(1, 4),(3, 4),(3, 5),(4, 5),
(5,1),(2,5),(3,5), (5,6), (6,2)])
nx.eccentricity(G)
{1: 3, 2: 3, 3: 3, 4: 3, 5: 2, 6: 4}

nx.eccentricity(G, v=[1])
{1: 3}
```




```
G = nx.DiGraph([(1, 2),(1, 3),(1, 4),(3, 4),(3, 5),(4, 5),
(5,1),(2,5),(3,5), (5,6), (6,2)])
nx.diameter(G)
4
```

Summary

Path, Distance

- Finding Paths
- Shortest Path
- Diameter

References

Hagberg, A., Swart, P., and S Chult, D.
Exploring network structure, dynamics, and function using networkx.
Tech. rep., Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

Newman, M.
Networks.
Oxford university press, 2018.

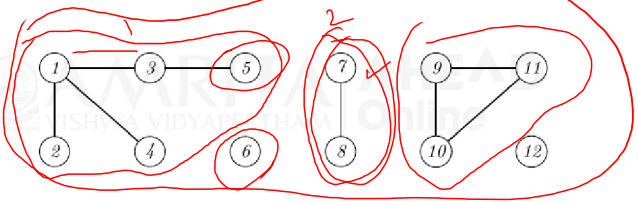
Objectives

- Measures & Metrics
  - Graph Connectedness
  - Graph Components

Connectedness

Definition (Connectedness)

Let $G = (V, E)$ be an undirected graph, and $v_0 \in V$. The connected component of v_0 is the set of all $v \in V$ such that there exists a path from v_0 to v. The connected components of G are the subsets of V which arise as connected components of some $v_0 \in V$.



- The connected component of 1 is the same as the connected component of 2, or 3, or 4, or 5.
- Similarly for 7 and 8, or 9, 10, and 11. Then the connected components of G are {1, 2, 3, 4, 5}, {6}, {7, 8}, {9, 10, 11}, and {12}.

Connectedness ..

Definition (Connectedness: Digraph)

Let $G = (V, E)$ be a directed graph, and let $G' = (V, E')$ be the associated undirected graph, i.e., let $E' = \{(u, v), (v, u) : (u, v) \in E\}$. The connected components of G are the connected components of G'

Definition of connected components is that it allows us to break up arbitrary graphs into smaller, and hopefuly bite-size, pieces.

- Let $G = (V, E)$ and $G' = (V', E')$ be graphs. If G is undirected, we assume G' is also undirected. We say $G' = (V', E')$ is a subgraph of G if $V' \subset V$ and $E' \subset E$.
- Let G be a graph. We say G is connected if G has exactly one connected component.

CONNECTEDNESS ..

Discussion

- We can get from vertex u to vertex v if and only if they are in the same connected component.
- In particular, we can get from any vertex u to any other vertex v if and only if G is connected.
- Consequently, being connected is one basic property we typically want in things like communication and transportation networks.

DEFINITION (STRONGLY CONNECTED)

Let $G = (V, E)$ be a directed or undirected graph, and $v_0 \in V$. The strongly connected component of v_0 is the set of all $v \in V$ such that there exists both a path from v_0 to v and a path from v to v_0 . The strongly connected components of G are the subsets of V which arise as strongly connected components of some $v_0 \in V$.

We say G is strongly connected if it has exactly one strongly connected component.

K-CONNECTEDNESS (1/5)

NETWORK DESIGN

- Connectedness in one primary requirement in network design
  - More link, more cost
  - Minimal edges: tree

DEFINITION (K-CONNECTED)

Let $G = (V, E)$ be a graph (possibly directed and non-simple) of order $n\,k$ with $n > 1$. We say G is **k-connected**, or **k-vertex-connected**, if the removal of any subset of $<$ k vertices (and involved edges) yields a connected subgraph. The vertex connectivity $k(G)$ of G is the maximal non-negative integer such that G is $k(G)$-connected. Alternatively, $k(G)$ is the minimal number of vertices one needs to remove to make G disconnected or have order 1.

K-CONNECTEDNESS (2/5)

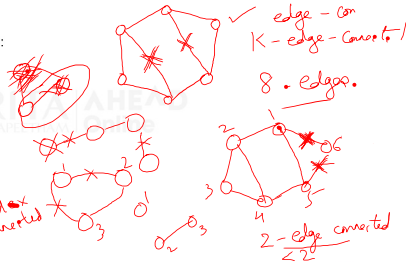
- A vertex cut is a set of vertices V_0 of V such that the subgraph $V - V_0$ is disconnected. Hence the minimal size of a vertex cut (when one exists) is $k(G)$.
- Partition tolerence in distributed system/cloud system.
- $k(G)$ tells us that if $< k(G)$ nodes of our network fail, the remainder of our network will still be functional (connected).
- G is 1-connected if and only if G is connected.
- Any k-connected graph is automatically (k - 1)-connected from the definition

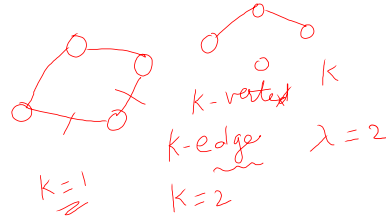
DEFINITION

Let $G = (V, E)$ be a graph (possibly directed and non-simple) of order $n \geq 2$. We say G is **k-edge-connected** if the removal of any subset of $<$ k edges (but no vertices) yields a connected subgraph. The edge connectivity $\lambda(G)$ is the maximal non-negative integer such that removing any subset of $< \lambda(G)$ edges (but no vertices) leaves G connected. Alternatively, $\lambda(G)$ is the minimal number of edges one needs to remove to make G

EXAMPLE

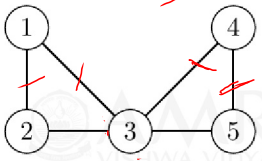
- Linear graph L_n of order $n \geq 2$:
  - $k(L_n) = \lambda(L_n) = 1$.
- Tree T of order $n \geq 2$:
  - $k(T) = \lambda(T) = 1$.
- Cycle graph C_n of order $n \geq 3$:
  - $k(C_n) = \lambda(C_n) = 2$.
- Complete graph K_n:
  - $k(K_n) = \lambda(K_n) = n - 1$.

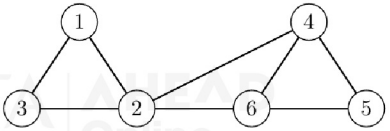




K-CONNECTEDNESS (4/5)

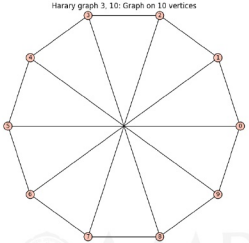
EXAMPLE

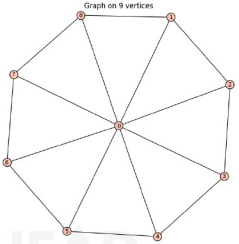




- $k(G) = 1$, $\{3\}$ is a vertex cut,
  $\lambda(G) = 2$, $\{\{1, 2\}, \{1, 3\}\}$
- $k(G) = 1$ and $\lambda(G) = 2$. (Two cycle graphs are connected using two edges)

K-CONNECTEDNESS (5/5)





EXAMPLE

Harary Graph ($H_{2n,3}$), Wheel Graph

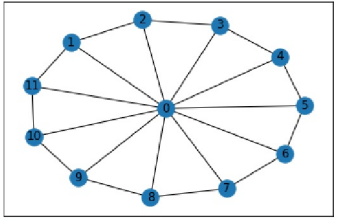
- Harary Graph, $H_{2n,3}$, is a 3-connected graph on 2n vertices with the min- imum possible number of edges, 3n.
  - $k(H_{2n,3}) = \lambda(H_{2n,3}) = 3$
- wheel graph $W(n)$ has diameter 2 and is 3-connected (and 3-edge-connected), with 2n-2 edges

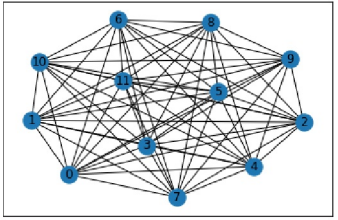
NetworkX Examples

- Plot a wheel graph and complete graph of order n, find $k(W(n))$ and $\lambda(W(n))$

```
G = nx.wheel_graph(12)
len(nx.minimum_edge_cut(G))
3
len(nx.minimum_node_cut(G))
3
```

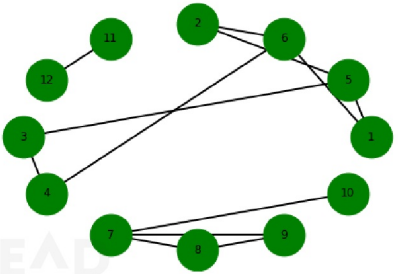
```
G = nx.complete_graph(12)
len(nx.minimum_edge_cut(G))
11
len(nx.minimum_node_cut(G))
11
```





In NetworkX

```
G7 = nx.Graph([(1,5),(1,6),(2,5),(11,12)])
G7.add_edges_from([(2,6),(3,5),(4,3),(4,6)])
G7.add_edges_from([(7,8),(8,9),(9,7),(7,10)
nx.draw (G7)
```




```
nx.is_connected(G7)
False
nx.number_connected_components(G7)
3
[len(c) nx.is_connected(G7)for c in sorted(nx.connected_components(G7),
key=len, reverse=True)]
[6, 4, 2]
```

Summary

Connectedness

- k-connectedness
- λ-connectedness

References

Hagberg, A., Swart, P., and S Chult, D.
Exploring network structure, dynamics, and function using networkx.
Tech. rep., Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

Newman, M.
Networks.
Oxford university press, 2018.

OBJECTIVES

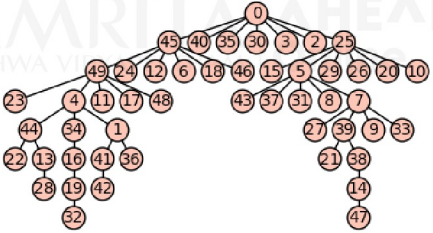
Measures & Metrics

- Tree, k-regular

TREE, K-REGULAR (1/2)

DEFINITION (TREE)

Let G be a (simple undirected) graph. We say G is a tree if it is connected and has no cycles of length > 2.

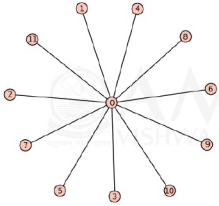


cycle of

TREE, K-REGULAR (2/2)

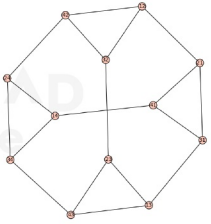
Tree

- Let $G = (V, E)$ a connected graph of order n. Then G is a tree if and only if $|E| = n - 1$.



deg = k

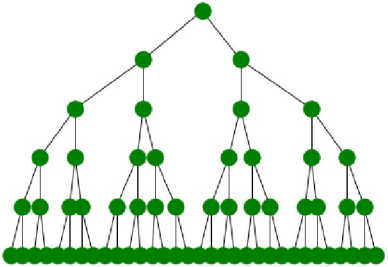
k - regular.

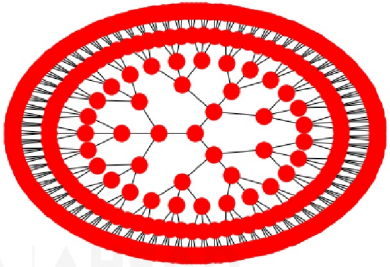


EXAMPLE (STAR GRAPH)

The star graph of order n is the undirected graph $G = (V, E)$ with $V = \{1, 2, ..., n\}$, E = {{1, 2}, {1, 3}, . . . {1, n}}. That is, vertex 1 is connected to all other vertices, and there are no other edges.

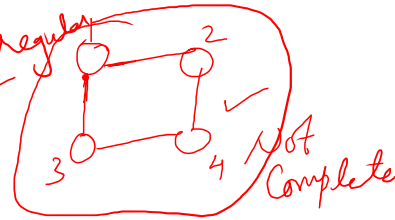
In NetworkX ..





```
import matplotlib.pyplot as plt
import networkx as nx
import pydot
from networkx.drawing.nx_pydot import graphviz_layout
T = nx.balanced_tree(b, 5) // b is number of branches
pos = graphviz_layout(T, prog="twopi")
nx.draw(T, pos, node_color="r")
is_tree(T)
```

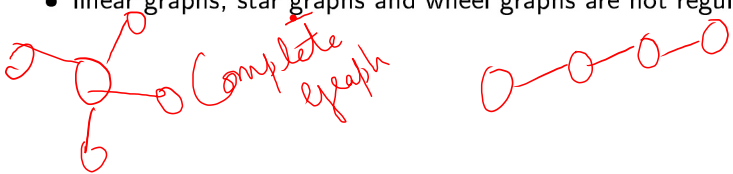
K-REGULAR



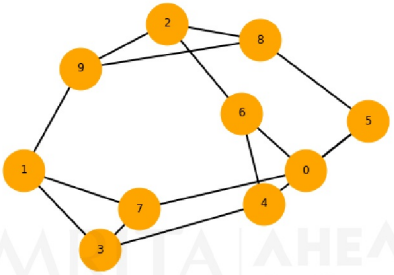
DEFINITION

Let G be a graph (simple or not, but undirected). We say G is k-regular if each vertex of G has degree k. We say G is regular if it is k-regular for some k.

- C_n , K_n and $H_{2n,3}$ are regular graphs
- linear graphs, star graphs and wheel graphs are not regular



In NetworkX ..




```
G8=nx.random_regular_graph(3,10,seed=None)
nx.draw (G8)
is_regular(G8)
True
is_k_regular(G8)
True
```

SUMMARY

- Tree
- k-regular

References

Hagberg, A., Swart, P., and S Chult, D.
Exploring network structure, dynamics, and function using networkx.
Tech. rep., Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

Newman, M.
Networks.
Oxford university press, 2018.

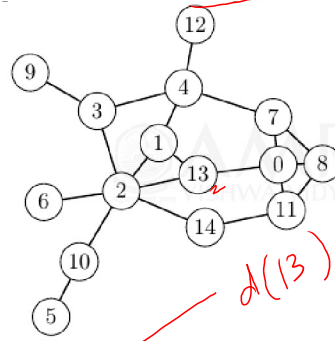
Objectives

Measures & Metrics

- Degree Centrality
- Degree Distribution
- Scale-free graph

Centrality Metrics

Florentine families graph



$d(13) = \frac{3}{15-1} = \frac{3}{14}$

Centrality

- Notion of Centrality
  - How Powerful a node is ?
  - How much influence a node has ?
  - Global and local centrality measures
- Degree centrality.

Degree Centrality

Node Centrality

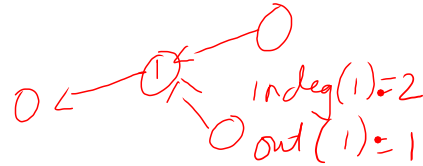
- Degree of a vertex is the number of edges connected to it, in terms of the adjacency matrix A, the degree for a node indexed by i in an undirected network is $k_i = \sum_j a_{ij}$
- Given a (simple undirected) graph G, the degree centrality of node u

$$d_u = \frac{deg(u)}{n-1} = \frac{k_u}{n-1}$$

n = no. of vertices/nodes

- Degree centrality values are normalized by dividing by the maximum possible degree, which is $n-1$.
- Maximum possible degree centrality 1, since maximum possible degree of a node is $n-1$.

Degree Centrality ..



In Directed Networks

- In directed networks, vertices have both an in-degree and an out-degree, $k_i^{\text{in}} = \sum_j a_{ij}$, $k_i^{\text{out}} = \sum_j a_{ji}$, respectively.
- In-degree centrality is

$$d_u^{\text{in}} = \frac{deg^{\text{in}}(u)}{n-1} = \frac{k_i^{\text{in}}}{n-1}$$

h-index
Top 2% Scientists.

Simple but Illuminating

- In social networks to find influential people
- In citation networks, the number of citations a paper receives could be used for judging the impact of scientific research

Degree Distribution (1/3)

Definition (Degree Distribution)

The degree distribution of G is the function $P : Z_{\geq 0} \rightarrow Z_{\geq 0}$ defined by P(d) is the proportion of nodes in V of degree d.
Note the degree distribution defines a probability function on $Z_{\geq 0}$ it satisfies $P(d) \geq 0$ for all d and the probabilities sum to one

- in degree distribution
- out degree distribution

Degree Distribution (2/3)

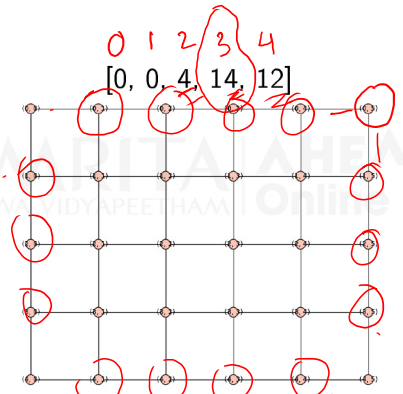
MLE

Power Law

```
G = graphs.Grid2dGraph(5,6);G
G.degree_histogram()
```

0 1 2 3 4

[0, 0, 4, 14, 12]



30 vertices

Vertices w' deg=0

0

Vertices w' deg=1

0

w deg=2

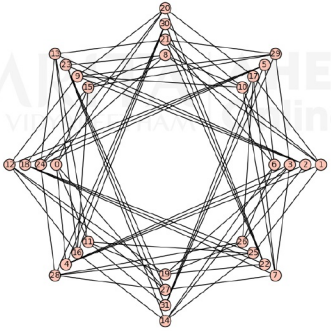
$P(d)$= [0, 0, 4/30, 14/30, 12/30]

Degree Distribution (3/3)

Power Law ..

```
g = graphs.WellsGraph(); g
g.degree_histogram()
```

[0, 0, 0, 0, 0, 32]



$P(d) = [0, 0, 0, 0, 0, 1]$

Summary

Node Degree

- Degree Centrality
- Distribution

References

Hagberg, A., Swart, P., and S Chult, D.
Exploring network structure, dynamics, and function using networkx.
Tech. rep., Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

Newman, M.
Networks.
Oxford university press, 2018.

Objectives

Measures & Metrics

- Betweenness Centrality
- Closeness Centrality

Betweenness Centrality

Basics

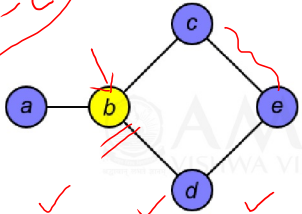
- This measures the number of times a given vertex u lies on a (shortest length) path between other vertices v_1 and v_2, and gives a higher score the more times u appears.

Betweenness Centrality

$$c(v) = \sum_{s \neq v \neq t} \frac{\#\{\text{shortest } st - \text{paths containing } v\}}{\#\{\text{shortest } st\text{paths}\}}$$

Betweeness Centrality: Example (1/2)

Find the centrality betweeness of node b



Betweeness Centrality

$$C_B(v) = \sum_{s \neq v \neq t} \frac{\#\{\text{shortest } st-\text{paths containing } v\}}{\#\{\text{shortest } st\text{paths}\}}$$

$$C_B(n) = \sum_{s \neq n \neq t} \frac{\sigma_{st}(n)}{\sigma_{st}},$$

$$C_B(n) = \frac{\sum_{s \neq n \neq t} \frac{\sigma_{st}(n)}{\sigma_{st}}}{\binom{n-1}{2}} \quad // \text{ normalizing}$$

$$C_B(b) = \frac{\sigma_{ac}(b)}{\sigma_{ac}} + \frac{\sigma_{ad}(b)}{\sigma_{ad}} + \frac{\sigma_{ae}(b)}{\sigma_{ae}} + \frac{\sigma_{cd}(b)}{\sigma_{cd}} + \frac{\sigma_{ce}(b)}{\sigma_{ce}} + \frac{\sigma_{de}(b)}{\sigma_{de}}$$

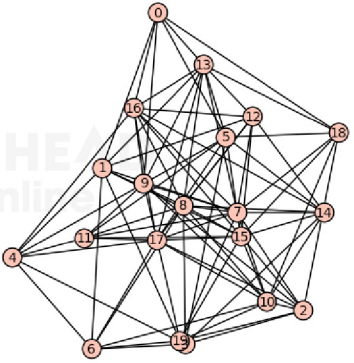
$$= \frac{1}{1} + \frac{1}{1} + \frac{2}{2} + \frac{1}{2} + \frac{0}{1} + \frac{0}{1} = 3.5$$

$$= \frac{3.5}{\binom{4}{2}} = \frac{3.5}{6}$$

Betweeness Centrality: Example (2/2)

In Sage

```
g = graphs.RandomGNP(20,.5)
g.centrality_betweenness()
{0: 1/24,
 1: 17/80,
 2: 7/80,
 3: 11/432,
 4: 0,
 5: 1/30,
 6: 1/36,
 7: 29/270,
 8: 43/180,
 9: 67/2160}
```



Closeness Centrality (1/2)

- Closeness centrality is a useful measure that estimates how fast the flow of information would be through a given node to other nodes.
- It is usually expressed as the normalised inverse of the sum of the topological distances in the graph
- Measure of how close a node is to other nodes in a graph
- The inverse distance is a measure of closeness centrality
- Inverse of farness

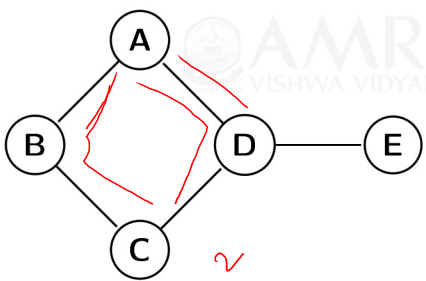
$$CC(i) = \frac{n-1}{\sum_{v \neq u} d(u,v)}$$

- $d(u,v)$ is the length of the shortest path between u and v

Closeness Centrality (2/2)

Calculation Using Distance Matrix

- A distance matrix is a table that shows the distance between pairs of nodes
- Distance matrix is square and symmetric
- Adding columns will result in a distance vector (sum of the distances)



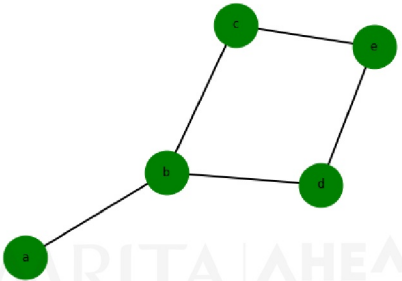
Dist. Matrix

	A	B	C	D	E		
A	0	1	2	1	2	6	$CC(A) = 4/6$
B	1	0	1	2	3	7	$CC(B) = 4/7$
C	2	1	0	1	2	6	$CC(C) = 4/6$
D	1	2	1	0	1	5	$CC(D) = 4/5$
E	2	3	2	1	0	8	$CC(E) = 4/8$

$n-1$

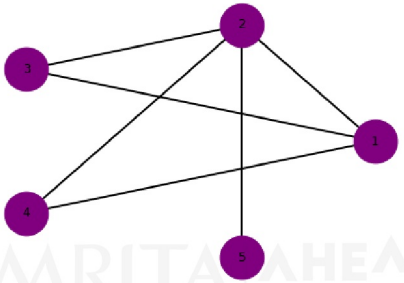
0.7 0.6 0.7 0.8 0.5

IN NETWORKX




```
G = nx.Graph([('b', 'c'), ('b', 'd'), ('c', 'e'), ('d', 'e'),('a', 'b')
betweenness_centrality(G)
{'b': 0.5833333333333333,
 'c': 0.16666666666666666,
 'd': 0.16666666666666666,
 'e': 0.08333333333333333,
 'a': 0.0}
```

In NetworkX




```
G1 = nx.Graph([(1, 2), (1, 3), (1, 4), (2, 3), (2, 4), (2,5)])
nx.closeness_centrality(G1)
{1: 0.8,
 2: 1.0,
 3: 0.6666666666666666,
 4: 0.6666666666666666,
 5: 0.5714285714285714}
```

Summary

- Centrality Betweenness
- Centrality Closeness

References

Hagberg, A., Swart, P., and S Chult, D.
Exploring network structure, dynamics, and function using networkx.
Tech. rep., Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

Newman, M.
Networks.
Oxford university press, 2018.