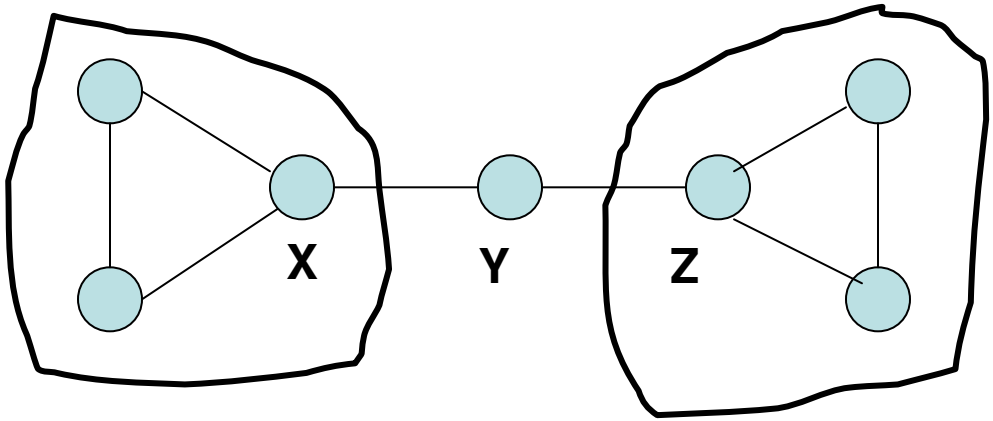
Centrality Metrics

Dr. Natarajan Meghanathan
Professor of Computer Science
Jackson State University
E-mail: natarajan.meghanathan@jsums.edu

Centrality

- Tells us which nodes are important in a network based on the topological structure of the network (instead of using the offline information about the nodes: e.g., popularity of nodes)
  - How influential a person is within a social network
  - Which genes play a crucial role in regulating systems and processes
  - Infrastructure networks: if the node is removed, it would critically impede the functioning of the network.



Nodes X and Z have higher Degree

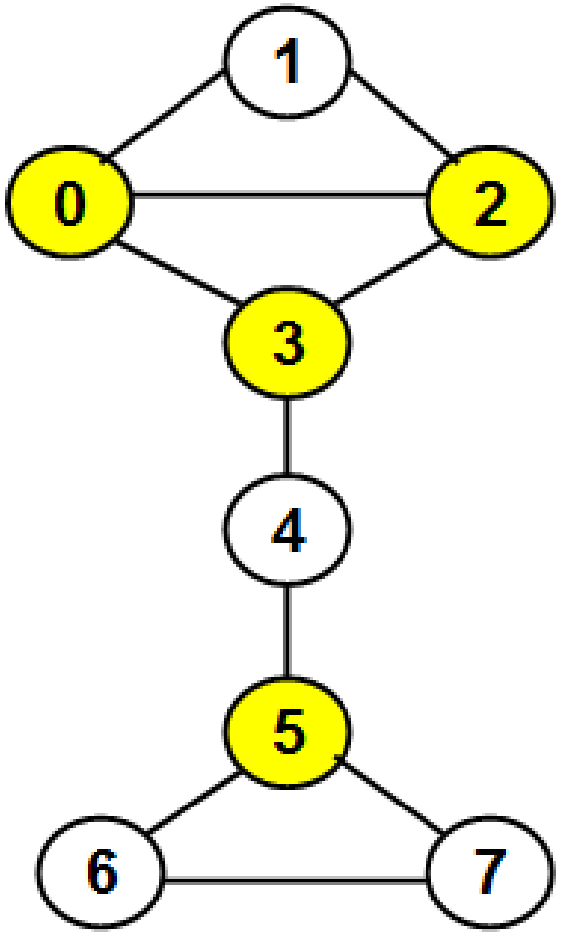
Node Y is more central from the point of view of Betweenness – to reach from one end to the other

Closeness – can reach every other vertex in the fewest number of hops

Centrality Metrics

- Degree-based Centrality Metrics
  - **Degree Centrality:** measure of the number of vertices adjacent to a vertex (degree)
  - **Eigenvector Centrality:** measure of the degree of the vertex as well as the degree of its neighbors
- Shortest-path based Centrality Metrics
  - Betweeness Centrality: measure of the number of shortest paths a node is part of
  - **Closeness Centrality:** measure of how close is a vertex to the other vertices [sum of the shortest path distances]
  - **Farness Centrality:** captures the variation of the shortest path distances of a vertex to every other vertex
- Hybrid Centrality Metrics
  - **Local Clustering Coefficient based Degree Centrality:** Nodes having a lower local clustering coefficient, but larger degree, lie on the shortest paths for several of their neighbor nodes.

Degree Centrality



	0	1	2	3	4	5	6	7
0	0	1	1	1	0	0	0	0
1	1	0	1	0	0	0	0	0
2	1	1	0	1	0	0	0	0
3	1	0	1	0	1	0	0	0
4	0	0	0	1	0	1	0	0
5	0	0	0	0	1	0	1	1
6	0	0	0	0	0	1	0	1
7	0	0	0	0	0	1	1	0

Adjacency Matrix

x

1
1
1
1
1
1
1
1

Column Vector

=

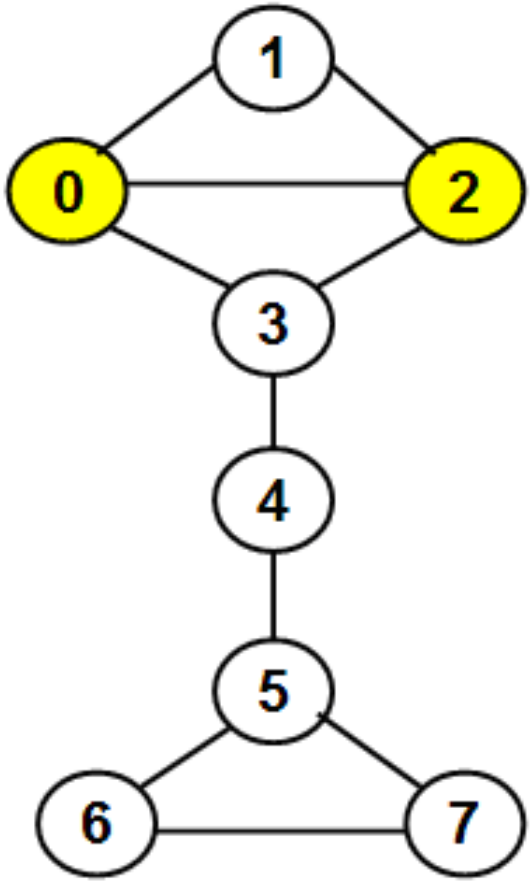
	Vertex IDs
3	0
2	1
3	2
3	3
2	4
3	5
2	6
2	7

Degree Centrality

Time Complexity: $\Theta(V^2)$

Weakness: Very likely that more than one vertex has the same degree and not possible to uniquely rank the vertices

Eigenvector Centrality (1)



Time Complexity: $\Theta(V^3)$

Power Iteration Method

Iteration 1

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \\ 3 \\ 3 \\ 2 \\ 3 \\ 2 \\ 2 \end{bmatrix} \equiv \begin{bmatrix} 0.416 \\ 0.277 \\ 0.416 \\ 0.416 \\ 0.277 \\ 0.416 \\ 0.277 \\ 0.277 \end{bmatrix}$$

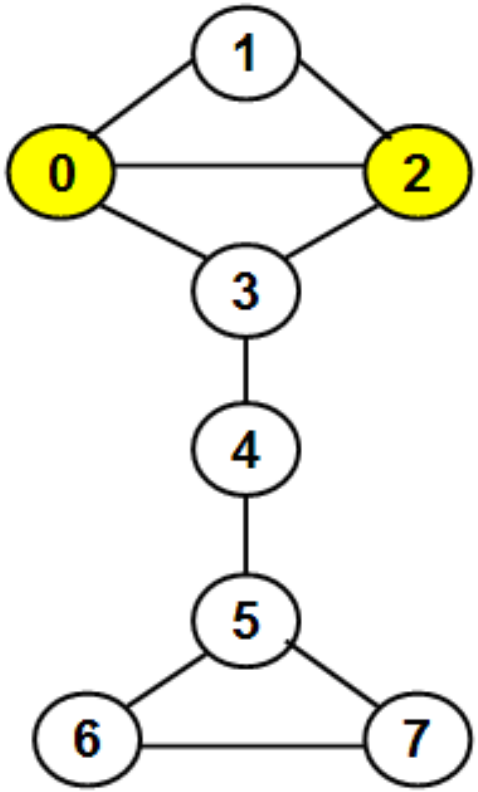
Normalized Value = 7.21

Iteration 2

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} 0.416 \\ 0.277 \\ 0.416 \\ 0.416 \\ 0.277 \\ 0.416 \\ 0.277 \\ 0.277 \end{bmatrix} = \begin{bmatrix} 1.109 \\ 0.832 \\ 1.109 \\ 1.109 \\ 0.832 \\ 0.831 \\ 0.693 \\ 0.693 \end{bmatrix} \equiv \begin{bmatrix} 0.428 \\ 0.321 \\ 0.428 \\ 0.428 \\ 0.321 \\ 0.321 \\ 0.268 \\ 0.268 \end{bmatrix}$$

Normalized Value = 2.59

Eigenvector Centrality (2)



Iteration 3

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} 0.428 \\ 0.321 \\ 0.428 \\ 0.428 \\ 0.321 \\ 0.321 \\ 0.268 \\ 0.268 \end{bmatrix} = \begin{bmatrix} 1.177 \\ 0.856 \\ 1.284 \\ 1.177 \\ 0.749 \\ 0.857 \\ 0.589 \\ 0.589 \end{bmatrix} \equiv \begin{bmatrix} 0.441 \\ 0.321 \\ 0.481 \\ 0.441 \\ 0.281 \\ 0.321 \\ 0.221 \\ 0.221 \end{bmatrix}$$

Normalized Value = 2.67

Iteration 4

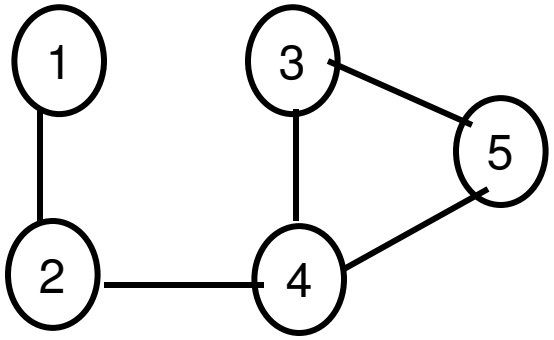
$$\begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} 0.441 \\ 0.321 \\ 0.481 \\ 0.441 \\ 0.281 \\ 0.321 \\ 0.221 \\ 0.221 \end{bmatrix} = \begin{bmatrix} 1.243 \\ 0.922 \\ 1.203 \\ 1.203 \\ 0.762 \\ 0.723 \\ 0.542 \\ 0.542 \end{bmatrix} \equiv \begin{bmatrix} 0.471 \\ 0.349 \\ 0.456 \\ 0.456 \\ 0.289 \\ 0.274 \\ 0.205 \\ 0.205 \end{bmatrix}$$

Normalized Value = 2.64

After 7 iterations

Vertex ID	0	1	2	3	4	5	6	7
Principal Eigenvector	0.489	0.364	0.489	0.467	0.264	0.232	0.155	0.155

EigenVector Centrality Example (1)



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

Let X0 = $\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$

Iteration 1

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 2 \\ 3 \\ 2 \end{bmatrix} \equiv \begin{bmatrix} 0.213 \\ 0.426 \\ 0.426 \\ 0.639 \\ 0.426 \end{bmatrix}$$

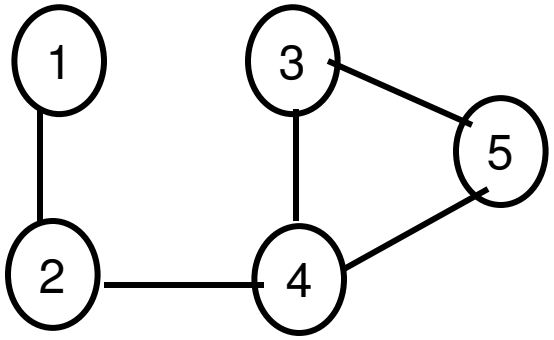
Normalized Value = 4.69

Iteration 2

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0.213 \\ 0.426 \\ 0.426 \\ 0.639 \\ 0.426 \end{bmatrix} = \begin{bmatrix} 0.426 \\ 0.852 \\ 1.065 \\ 1.278 \\ 1.065 \end{bmatrix} \equiv \begin{bmatrix} 0.195 \\ 0.389 \\ 0.486 \\ 0.584 \\ 0.486 \end{bmatrix}$$

Normalized Value = 2.19

EigenVector Centrality Example (1)



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

Let X0 = $\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$

Iteration 3

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0.195 \\ 0.389 \\ 0.486 \\ 0.584 \\ 0.486 \end{bmatrix} = \begin{bmatrix} 0.389 \\ 0.779 \\ 1.07 \\ 1.361 \\ 1.07 \end{bmatrix} \equiv \begin{bmatrix} 0.176 \\ 0.352 \\ 0.484 \\ 0.616 \\ 0.484 \end{bmatrix}$$

Normalized Value = 2.21

Iteration 4

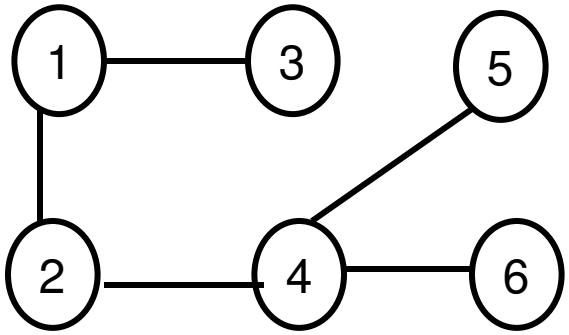
$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0.176 \\ 0.352 \\ 0.484 \\ 0.616 \\ 0.484 \end{bmatrix} = \begin{bmatrix} 0.352 \\ 0.792 \\ 1.100 \\ 1.320 \\ 1.100 \end{bmatrix}$$

Normalized Value = 2.21 converges

Eigen Vector Centrality

1	0.176
2	0.352
3	0.484
4	0.616
5	0.484

EigenVector Centrality Example (2)



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

Let X0 = $\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$

Iteration 1

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \\ 1 \\ 3 \\ 1 \\ 1 \end{bmatrix} \equiv \begin{bmatrix} 0.447 \\ 0.447 \\ 0.224 \\ 0.671 \\ 0.224 \\ 0.224 \end{bmatrix}$$

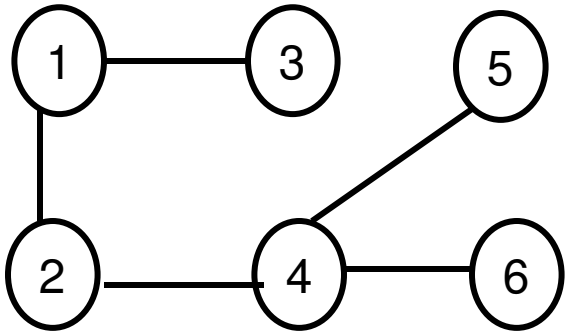
Normalized Value = 4.472

Iteration 2

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0.447 \\ 0.447 \\ 0.224 \\ 0.671 \\ 0.224 \\ 0.224 \end{bmatrix} = \begin{bmatrix} 0.671 \\ 0.671 \\ 0.447 \\ 0.895 \\ 0.671 \\ 0.671 \end{bmatrix} \equiv \begin{bmatrix} 0.401 \\ 0.401 \\ 0.267 \\ 0.535 \\ 0.401 \\ 0.401 \end{bmatrix}$$

Normalized Value = 1.674

EigenVector Centrality Example (2)



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

Let X0 = $\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$

Iteration 3

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0.401 \\ 0.401 \\ 0.267 \\ 0.535 \\ 0.401 \\ 0.401 \end{bmatrix} = \begin{bmatrix} 0.668 \\ 0.936 \\ 0.401 \\ 1.203 \\ 0.535 \\ 0.535 \end{bmatrix} \equiv \begin{bmatrix} 0.357 \\ 0.500 \\ 0.214 \\ 0.643 \\ 0.286 \\ 0.286 \end{bmatrix}$$

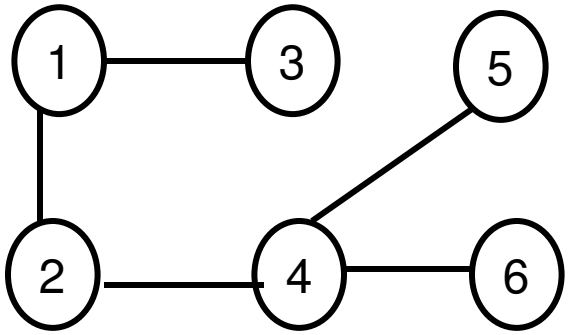
Normalized Value = 1.872

Iteration 4

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0.357 \\ 0.500 \\ 0.214 \\ 0.643 \\ 0.286 \\ 0.286 \end{bmatrix} = \begin{bmatrix} 0.714 \\ 1.000 \\ 0.357 \\ 1.072 \\ 0.643 \\ 0.643 \end{bmatrix} \equiv \begin{bmatrix} 0.376 \\ 0.526 \\ 0.188 \\ 0.564 \\ 0.338 \\ 0.338 \end{bmatrix}$$

Normalized Value = 1. 901

EigenVector Centrality Example (2)



Iteration 5

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0.376 \\ 0.526 \\ 0.188 \\ 0.564 \\ 0.338 \\ 0.338 \end{bmatrix} = \begin{bmatrix} 0.714 \\ 0.940 \\ 0.376 \\ 1.202 \\ 0.564 \\ 0.564 \end{bmatrix} \equiv \begin{bmatrix} 0.376 \\ 0.494 \\ 0.198 \\ 0.632 \\ 0.297 \\ 0.297 \end{bmatrix}$$

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

Normalized Value = 1. 901 converges

Let X0 = $\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$

EigenVector Centrality

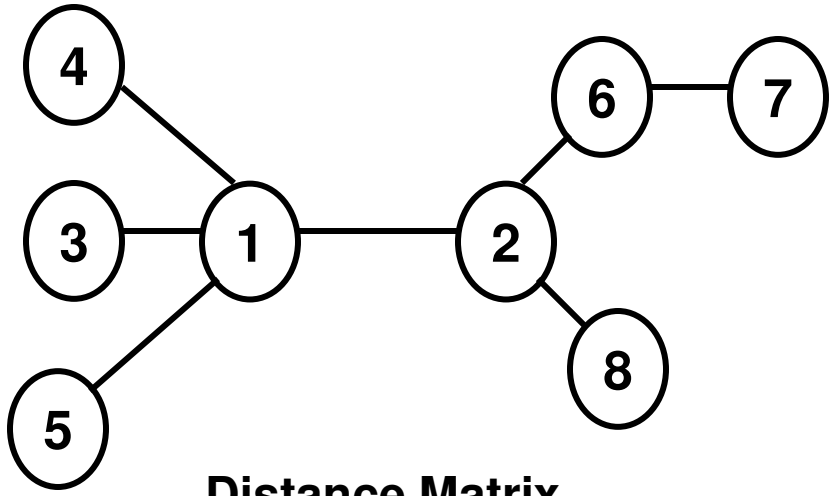
$$\begin{bmatrix} 0.376 \\ 0.494 \\ 0.198 \\ 0.632 \\ 0.297 \\ 0.297 \end{bmatrix}$$

Node Ranking

4
2
1
5
6
3

Note that we typically stop when the EigenVector values converge.
For exam purposes, we will Stop when the Normalized value converges.

Closeness and Farness Centrality



Distance Matrix

	1	2	3	4	5	6	7	8
1	**0**	**1**	**1**	**1**	**1**	**2**	**3**	**2**
2	**1**	**0**	**2**	**2**	**2**	**1**	**2**	**1**
3	**1**	**2**	**0**	**2**	**2**	**3**	**4**	**3**
4	**1**	**2**	**2**	**0**	**2**	**3**	**4**	**3**
5	**1**	**2**	**2**	**2**	**0**	**3**	**4**	**3**
6	**2**	**1**	**3**	**3**	**3**	**0**	**1**	**2**
7	**3**	**2**	**4**	**4**	**4**	**1**	**0**	**3**
8	**2**	**1**	**3**	**3**	**3**	**2**	**3**	**0**

Time Complexity: Θ(VE + V^2)

Sum of distances	**Closeness**
11	0.0909
11	0.0909
17	0.0588
17	0.0588
17	0.0588
15	0.0667
21	0.0476
17	0.0588

Farness

Principal Eigenvalue
η1 = 16.315

Principal Eigenvector
δ1 =
[0.2527
0.2518
0.3771
0.3771
0.3771
0.3278
0.4439
0.3763]

Ranking of Nodes

Score	Node ID
0.2518	2
0.2527	1
0.3278	6
0.3763	8
0.3771	3
0.3771	4
0.3771	5
0.4439	7

Betweeness Centrality

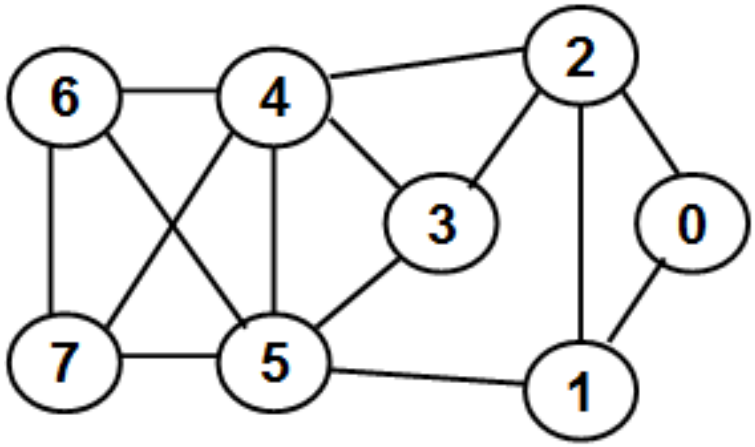
$$BWC(i) = \sum_{j \neq k \neq i} \frac{sp_{jk}(i)}{sp_{jk}}$$

(j < k for undirected graphs)

Time Complexity: Θ(VE + V^2)

- We will now discuss how to find the total number of shortest paths between any two vertices *j* and *k* as well as to find out how many of these shortest paths go through a vertex *i* ($j \neq k \neq i$).
- Use Breadth First Search (BFS) to find the shortest path tree from vertex j to every other vertex k
  - Root vertex j is at level 0
  - Vertices that are 1-hop away from j are at level 1; 2-hops away from j are at level 2, and so on.
  - The number of shortest paths from *j* to a vertex *k* at level *p* is the sum of the number of shortest paths from *j* to the neighbors of *k* in the original graph that are at level *p*-1
  - The number of shortest paths from *j* to *k* that go through vertex *i* is the maximum of the number of shortest paths from *j* to *i* and the number of shortest paths from *k* to *i*.

Betweenness Centrality Example



BWC for node 0: 0.0

BWC for node 1
Pair (0, 5): → 1 / 1
Pair (0, 6): → 1 / 2
Pair (0, 7): → 1 / 2
Pair (2, 5): → 1 / 3
BWC (1) = 2.333

BWC for node 2
Pair (0, 3): → 1 / 1
Pair (0, 4): → 1 / 1
Pair (0, 6): → 1 / 2
Pair (0, 7): → 1 / 2
Pair (1, 3): → 1 / 2
Pair (1, 4): → 1 / 2
BWC (2): 4.0

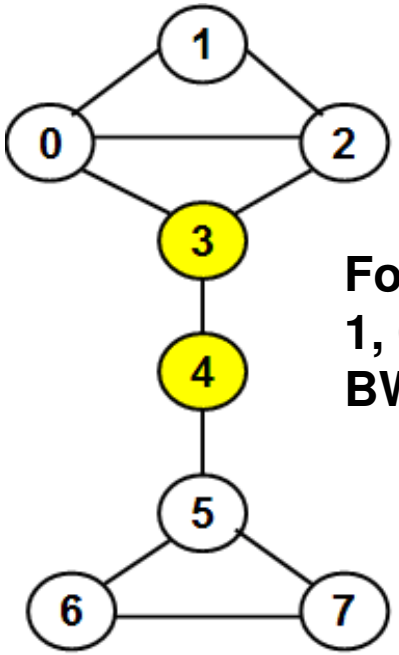
BWC for node 3
Pair (2, 5) → 1 / 3
BWC (3) = 0.333

BWC for node 4
Pair (0, 6) → 1 / 2
Pair (0, 7) → 1 / 2
Pair (2, 5) → 1 / 3
Pair (2, 6) → 1 / 1
Pair (2, 7) → 1 / 1
Pair (3, 6) → 1 / 2
Pair (3, 7) → 1 / 2
BWC (2) = 4.333

BWC for node 6: 0.0
BWC for node 7: 0.0

BWC for node 5
Pair (0, 6) → 1 / 2
Pair (0, 7) → 1 / 2
Pair (1, 3) → 1 / 2
Pair (1, 4) → 1 / 2
Pair (1, 6) → 1 / 1
Pair (1, 7) → 1 / 1
Pair (3, 6) → 1 / 2
Pair (3, 7) → 1 / 2
BWC (5) = 5.0

ID	BWC
0	0.0
1	2.333
2	4.0
3	0.333
4	4.333
5	5.0
6	0.0
7	0.0



For vertices
1, 6 and 7
BWC = 0

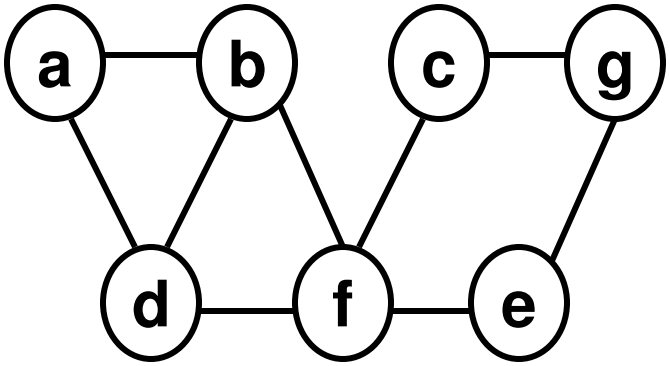
Betweeness for Vertex 0
Pair (3,1) --->1 / 2
Pair (4,1) --->1 / 2
Pair (5,1) --->1 / 2
Pair (6,1) --->1 / 2
Pair (7,1) --->1 / 2
Total of all Betweeness
(Vertex 0): 2.5

Betweeness for Vertex 2
Pair (3,1) --->1 / 2
Pair (4,1) --->1 / 2
Pair (5,1) --->1 / 2
Pair (6,1) --->1 / 2
Pair (7,1) --->1 / 2
Total of all Betweeness
(Vertex 2): 2.5

Betweeness for Vertex 5
Pair (6,0) --->1 / 1
Pair (7,0) --->1 / 1
Pair (6,1) --->2 / 2
Pair (7,1) --->2 / 2
Pair (6,2) --->1 / 1
Pair (7,2) --->1 / 1
Pair (6,3) --->1 / 1
Pair (7,3) --->1 / 1
Pair (6,4) --->1 / 1
Pair (7,4) --->1 / 1
Total of all Betweeness
(Vertex 5): 10

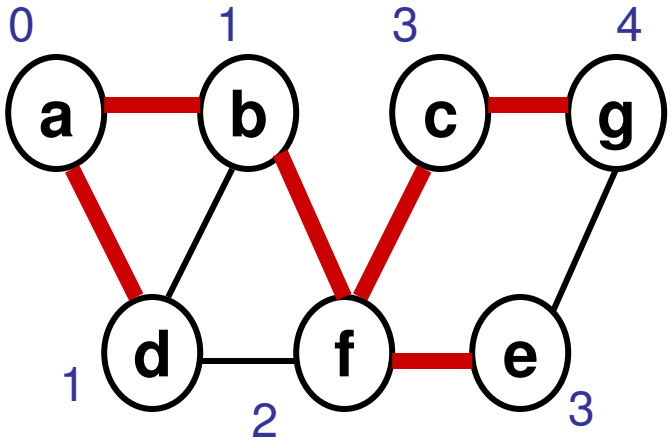
Betweeness for Vertex 3
Pair (4,0) --->1 / 1
Pair (5,0) --->1 / 1
Pair (6,0) --->1 / 1
Pair (7,0) --->1 / 1
Pair (4,1) --->2 / 2
Pair (5,1) --->2 / 2
Pair (6,1) --->2 / 2
Pair (7,1) --->2 / 2
Pair (4,2) --->1 / 1
Pair (5,2) --->1 / 1
Pair (6,2) --->1 / 1
Pair (7,2) --->1 / 1
Total of all Betweeness
(Vertex 3): 12

Betweeness for Vertex 4
Pair (5,0) --->1 / 1
Pair (6,0) --->1 / 1
Pair (7,0) --->1 / 1
Pair (5,1) --->2 / 2
Pair (6,1) --->2 / 2
Pair (7,1) --->2 / 2
Pair (5,2) --->1 / 1
Pair (6,2) --->1 / 1
Pair (7,2) --->1 / 1
Pair (5,3) --->1 / 1
Pair (6,3) --->1 / 1
Pair (7,3) --->1 / 1
Total of all Betweeness
(Vertex 4): 12

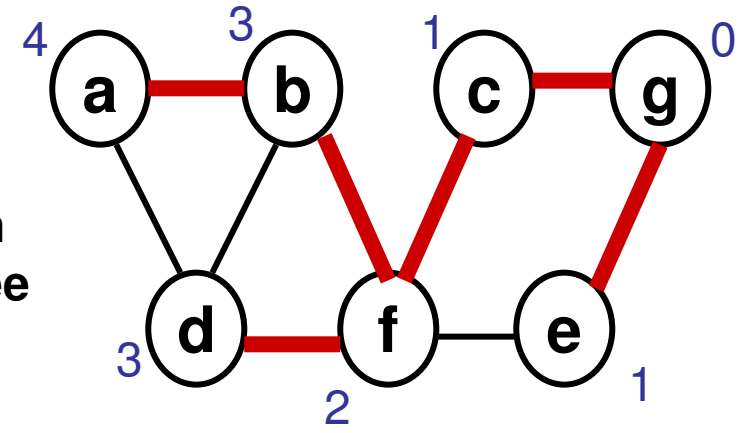


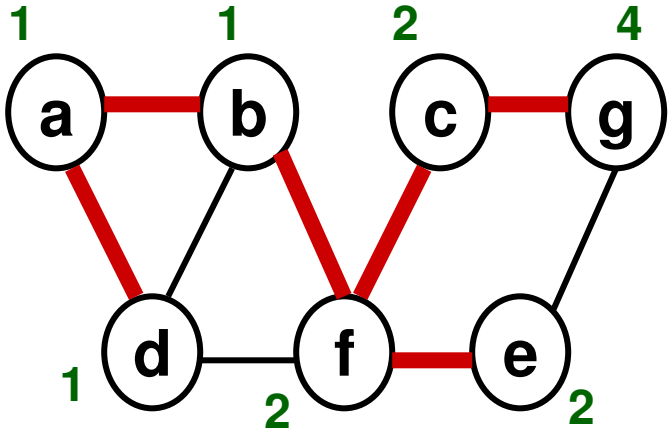
shortest paths from a to g that go through c
is the product(# shortest paths from a to c,
shortest paths from g to c)
= product (2, 1) = 2

BWC ('c' with respect to pair a-g) = 2/4

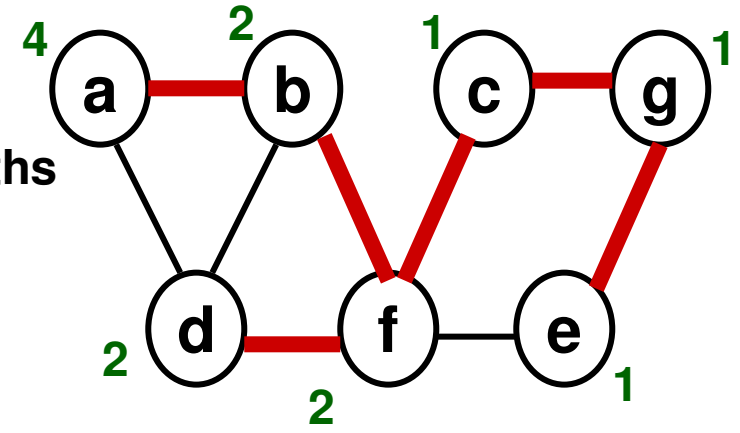


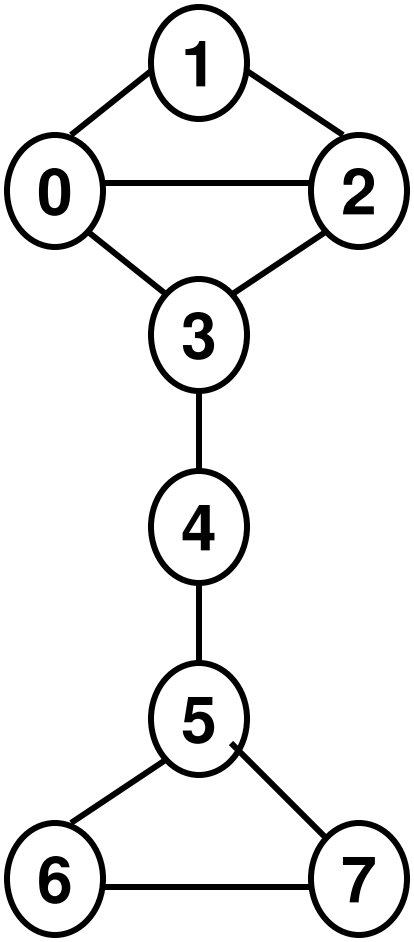
Levels of Vertices on the BFS tree

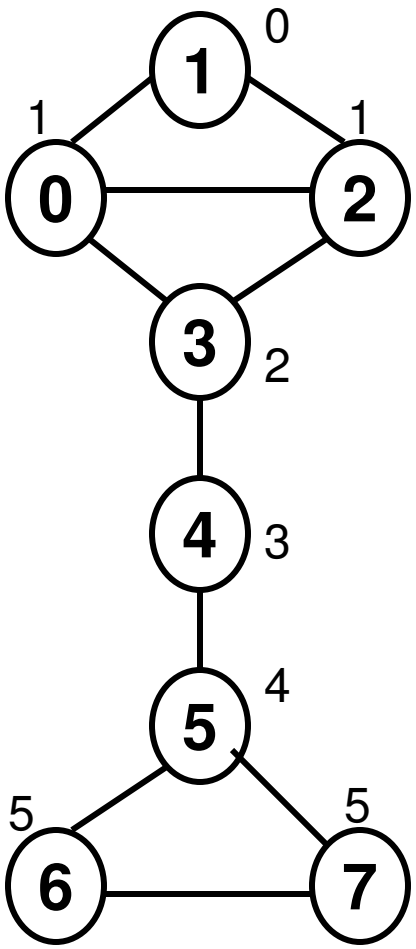


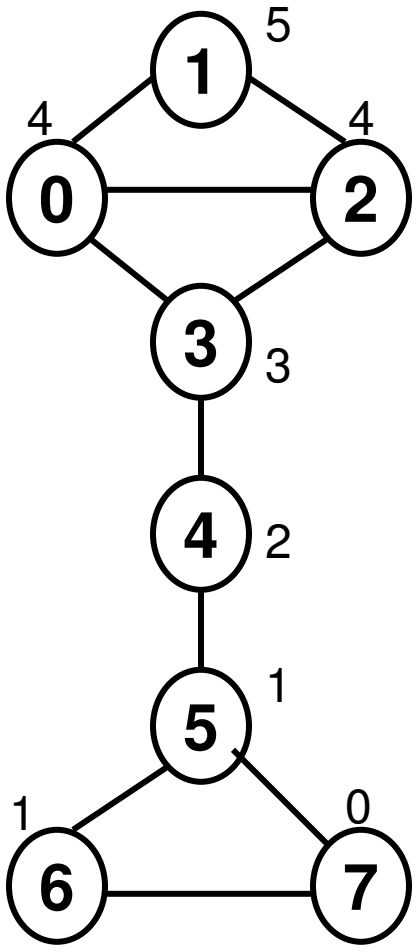


 # shortest paths from the root to the other vertices

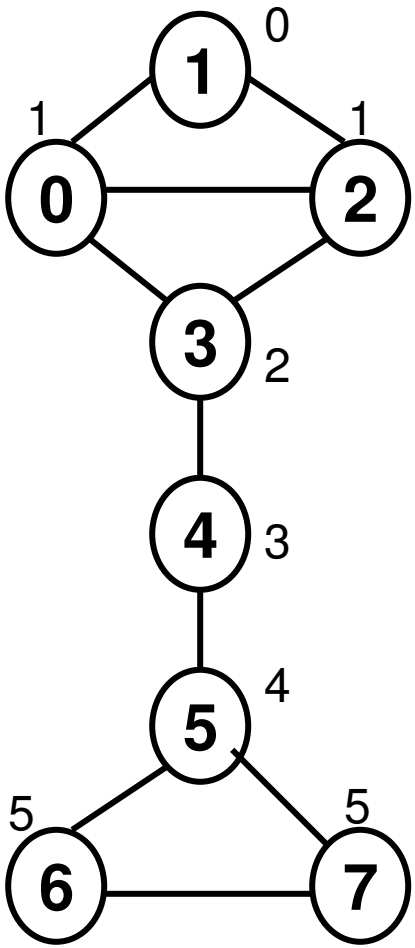




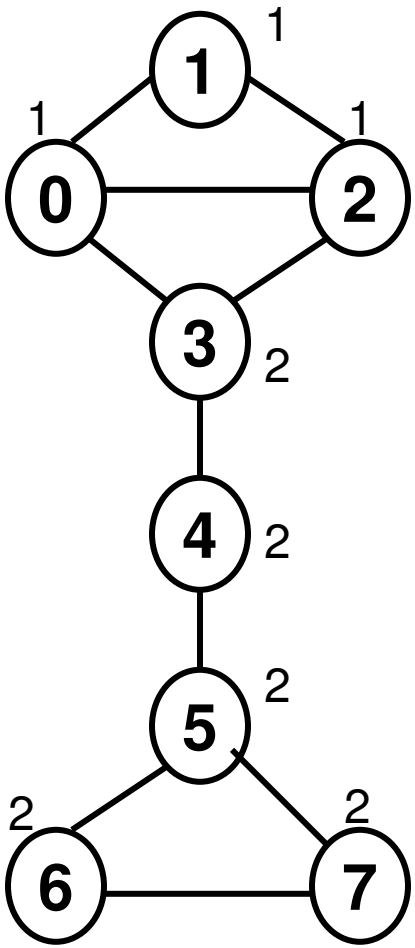




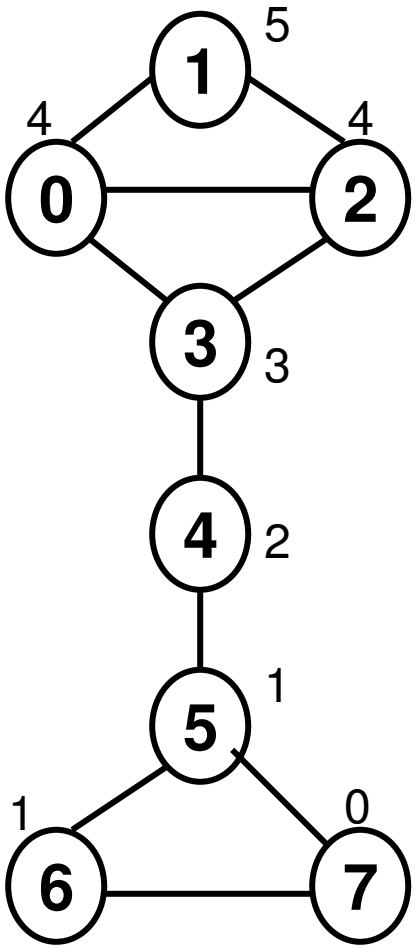
To determine how many Shortest paths from nodes 1 to 7 that go through node 4.



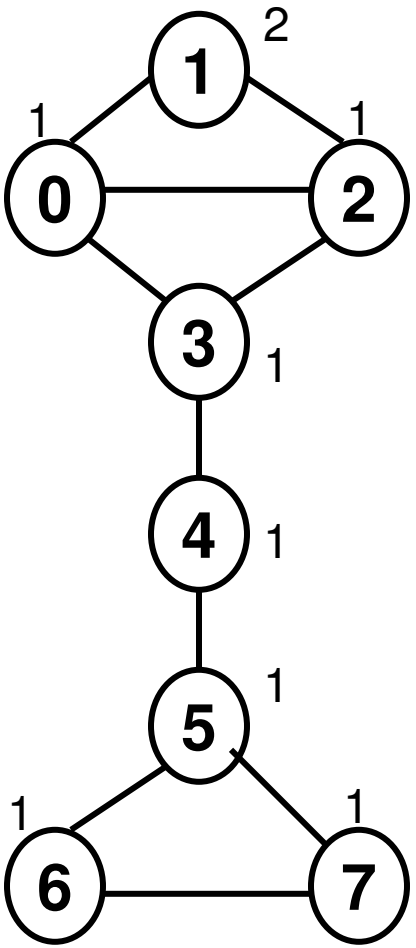
BFS Tree rooted at Vertex 1



 # shortest paths from vertex 1 to the other vertices



BFS Tree rooted at Vertex 7



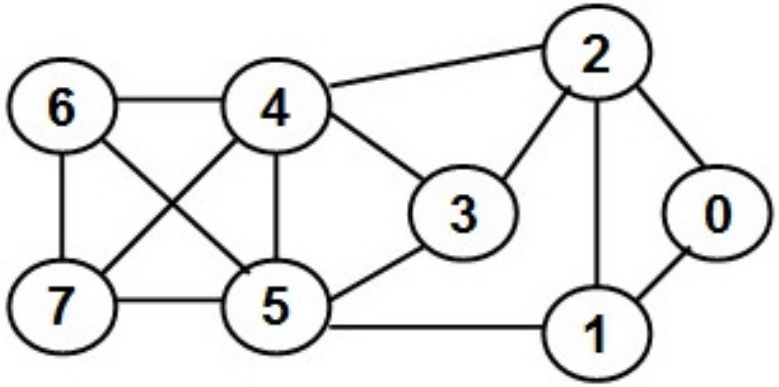
 # shortest paths from vertex 7 to the other vertices

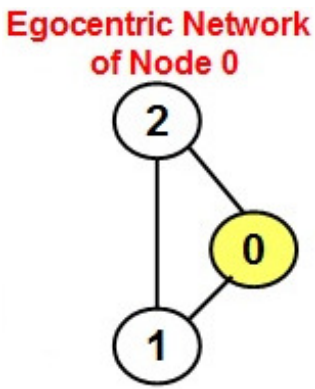
To determine how many Shortest paths from nodes 1 to 7 that go through node 4: = Product(2, 1) = 2

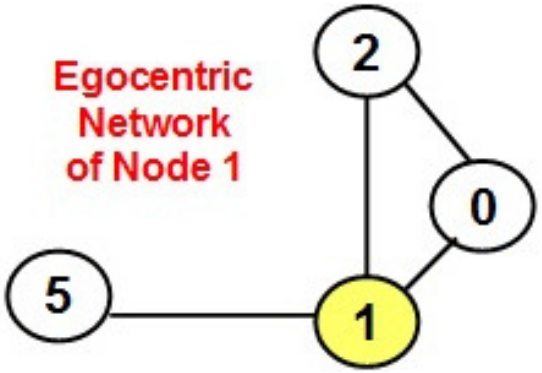
BWC(node 4 with respect to pair 1-7) = 2/2

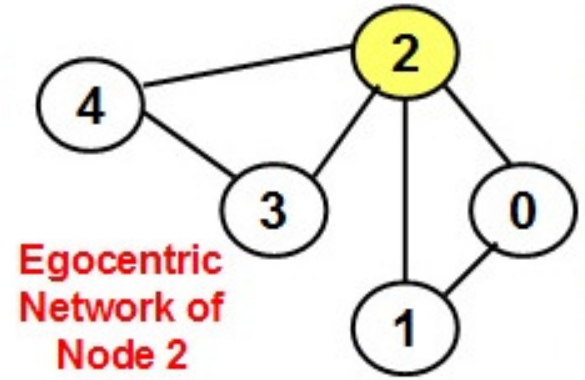
Ego Network

- The ego network is node-specific.
- The ego network for a node comprises of the node and its neighbors as vertices and the links connecting the node and/or its neighbors as edges.
- The BWC of a vertex computed on the entire graph is directly related to the LCC'DC of the vertex computed on its ego network (see next few slides).
  - Could be used to rank the vertices.





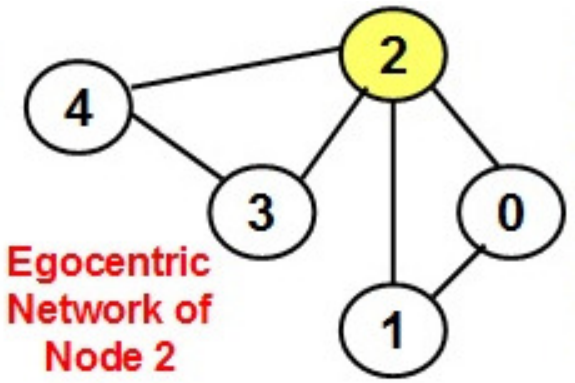




Local Clustering Coefficient-based Degree Centrality (LCC'DC)

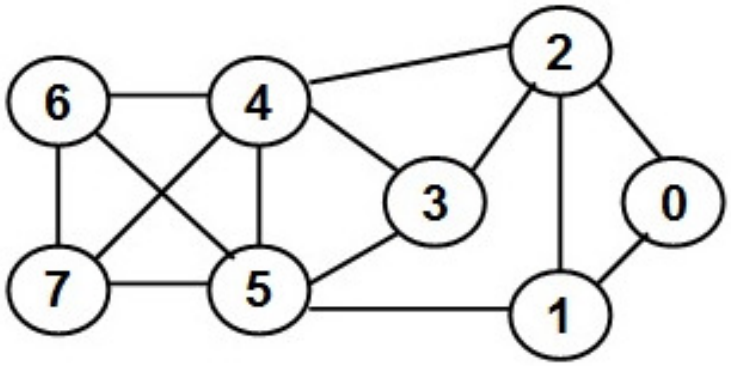
- Local Clustering Coefficient (LCC) of a node is the ratio of the number of edges connecting the neighbors of the node to that of the maximum number of edges between the neighbors of the node.
- A node having a lower LCC and a larger degree is more likely needed to connect its neighbor nodes on the shortest path, compared to that of a node having a larger LCC and a larger degree.
- LCC'DC (v) = (1 – LCC(v)) * degree(v)
- LCC'DC can be used as an alternate metric for ranking the vertices in a graph in lieu of BWC.
- The LCC'DC metric for a vertex can also be computed on the egocentric network graph of the vertex.

Note: LCC of a vertex with Degree 1 is 1.0



A maximum of 4(4-1)/2 = 6 edges is possible among the neighbors of node 2.
There are actually 2 edges among the neighbors.
LCC(2) = 2/6 = 1/3.
Degree(2) = 4.
LCC'DC(2) = [1 – LCC(2)] * Degree(2)
= [1 – 1/3] * 4 = 8/3 = 2.67.

LCC'DC Example



Vertex	LCC'DC	BWC
0	0.0	0.0
1	2.0	2.33
2	2.67	4.0
3	1.0	0.33
4	2.5	4.33
5	3.0	5.0
6	0.0	0.0
7	0.0	0.0

Vertex	Degree	Max. Edges among neighbs	Actual Edges among neighbs	LCC	LCC' 1 - LCC	LCC'DC
0	2	2(2-1)/2 = 1	1	1.0	0.0	0.0
1	3	3(3-1)/2 = 3	1	0.33	0.67	2.0
2	4	4(4-1)/2 = 6	2	0.33	0.67	2.67
3	3	3(3-1)/2 = 3	2	0.67	0.33	1.0
4	5	5(5-1)/2 = 10	5	0.5	0.5	2.5
5	5	5(5-1)/2 = 10	4	0.4	0.6	3.0
6	3	3(3-1)/2 = 3	3	1.0	0.0	0.0
7	3	3(3-1)/2 = 3	3	1.0	0.0	0.0

Spearman's Rank-based Correlation

- We could find the similarity of the ranking of the vertices in a graph with respect to two different centrality metrics using the Spearman's rank-based correlation measure.
- We follow the convention of assigning the rank values from 1 to *n* for a graph of *n* vertices, even though the vertex IDs range from 0 to *n*-1.
- To obtain the rank for a vertex based on the list of values for a centrality metric, we first sort the values (in ascending order).
  - If there is any tie, we break the tie in favor of the vertex with a lower ID; we will thus be able to arrive at a tentative, but unique, rank value for each vertex with respect to the centrality metric.
- We determine a final ranking of the vertices as follows:
  - For vertices with unique value of the centrality metric, the final ranking is the same as the tentative ranking.
  - For vertices with an identical value for the centrality metric, the final ranking is assigned to be the average of their tentative rankings.

Spearman's Rank-based Correlation

$$SCC(B,L) = 1 - \frac{6\sum_{i=1}^{n} d_i^{\,2}}{n(n^2-1)}$$

d_i is the difference in the final ranking for vertex i in the two datasets

Correlation between BWC and LCC'DC

Vertex	BWC (B)	Tentative Rank: B	Final Rank: b_i	LCC'DC (L)	Tentative Rank: L	Final Rank: l_i	Rank Difference (d_i): $b_i - l_i$	d_i^2
0	0	1	2	0	1	2	0	0
1	2.33	5	5	2	5	5	0	0
2	4	6	6	2.67	7	7	-1	1
3	0.33	4	4	1	4	4	0	0
4	4.33	7	7	2.5	6	6	1	1
5	5	8	8	3	8	8	0	0
6	0	2	2	0	2	2	0	0
7	0	3	2	0	3	2	0	0
							Sum	2

Correlation Coefficient = 1 - (6*2)/(8*(8^2-1)) = 0.98

Concordant, Discordant Pairs

- Let B and L be the two centrality metrics of interest.
- For any two vertices vi and vj:
  - vi and vj are concordant if:
    - B(vi) > B(vj) and L(vi) > L(vj)
    - B(vi) < B(vj) and L(vi) < L(vj)
    - B(vi) = B(vj) and L(vi) = L(vj)
  - vi and vj are discordant if:
    - B(vi) > B(vj) and L(vi) < L(vj)
    - B(vi) < B(vj) and L(vi) > L(vj)

Kendall's Concordance based Correlation Coefficient

$$KCC(B,L) = \frac{\#conc.pairs \; - \; \#disc.pairs}{\frac{1}{2}n(n-1)}$$

N. Meghanathan, "Correlation Coefficient Analysis of Centrality Metrics for Complex Network Graphs," Proceedings of the *4th Computer Science Online Conference*, (CSOC-2015), Intelligent Systems in Cybernetics and Automation Theory: Advances in Intelligent Systems and Computing, Vol. 348, pp. 11-20, April 27-30, 2015

Concordance-based Correlation

Vertex	LCC'DC	BWC
0	0.0	0.0
1	2.0	2.33
2	2.67	4.0
3	1.0	0.33
4	2.5	4.33
5	3.0	5.0
6	0.0	0.0
7	0.0	0.0

C – Concordant
D – Discordant

KCC = (27 – 1)/28
= 0.93

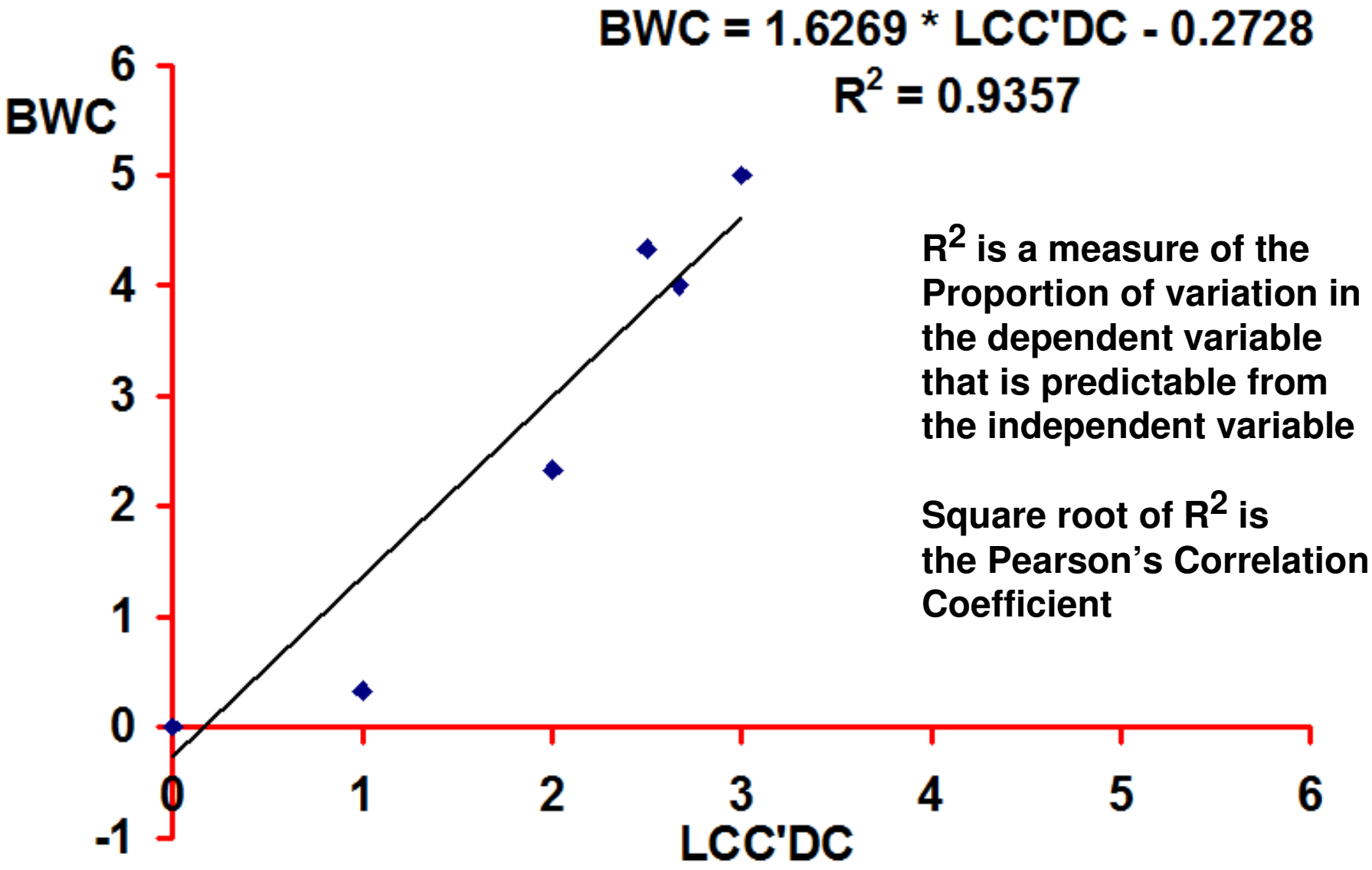
Pair Vi, Vj	LCC'DC (Vi, Vj)	BWC (Vi, Vj)	C/D
0, 1	(0, 2)	(0, 2.33)	C
0, 2	(0, 2.67)	(0, 4)	C
0, 3	(0, 1)	(0, 0.33)	C
0, 4	(0, 2.5)	(0, 4.33)	C
0, 5	(0, 3)	(0, 5)	C
0, 6	(0, 0)	(0, 0)	C
0, 7	(0, 0)	(0, 0)	C
1, 2	(2, 2.67)	(2.33, 4)	C
1, 3	(2, 1)	(2.33, 0.33)	C
1, 4	(2, 2.5)	(2.33, 4.33)	C
1, 5	(2, 3)	(2.33, 5)	C
1, 6	(2, 0)	(2.33, 0)	C
1, 7	(2, 0)	(2.33, 0)	C
2, 3	(2.67, 1)	(4, 0.33)	C
2, 4	(2.67, 2.5)	(4, 4.33)	D
2, 5	(2.67, 3)	(4, 5)	C
2, 6	(2.67, 0)	(4, 0)	C
2, 7	(2.67, 0)	(4, 0)	C
3, 4	(1, 2.5)	(0.33, 4.33)	C
3, 5	(1, 3)	(0.33, 5)	C
3, 6	(1, 0)	(0.33, 0)	C
3, 7	(1, 0)	(0.33, 0)	C
4, 5	(2.5, 3)	(4.33, 5)	C
4, 6	(2.5, 0)	(4.33, 0)	C
4, 7	(2.5, 0)	(4.33, 0)	C
5, 6	(3, 0)	(5, 0)	C
5, 7	(3, 0)	(5, 0)	C
6, 7	(0, 0)	(0, 0)	C

Pearson's Correlation

Node	LCC'DC	LCC'DC(i) - Avg(LCC'DC)	(LCC'DC(i) - Avg(LCC'DC))^2	BWC	BWC(i) - Avg(BWC)	(BWC(i) - Avg(BWC))^2	(LCC'DC(i) - Avg(LCC'DC)) * (BWC(i) - Avg(BWC))
0	0	-1.39	1.93	0	-1.99	3.960	2.766
1	2	0.61	0.37	2.33	0.34	0.116	0.207
2	2.67	1.28	1.64	4	2.01	4.040	2.573
3	1	-0.39	0.15	0.33	-1.66	2.756	0.647
4	2.5	1.11	1.23	4.33	2.34	5.476	2.597
5	3	1.61	2.59	5	3.01	9.060	4.846
6	0	-1.39	1.93	0	-1.99	3.960	2.766
7	0	-1.39	1.93	0	-1.99	3.960	2.766
	Avg = 1.39		Sum = 11.78	Avg = 1.99		Sum = 33.33	Sum = 19.17

$$r = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}} = \frac{19.17}{\text{Sqrt}(11.78)*\text{Sqrt}(33.33)} = 0.9675$$

Regression Analysis: LCC'DC vs. BWC



R^2 is a measure of the Proportion of variation in the dependent variable that is predictable from the independent variable

Square root of R^2 is the Pearson's Correlation Coefficient

Regression Analysis: LCC'DC vs. BWC

Predicted BWC = 1.6269 * Actual LCC'DC – 0.2728

Node	Actual LCC'DC	Actual BWC	Pred. BWC	Norm. Actual BWC	Norm. Pred BWC	Residual	Square of Residual
0	0	0	-0.27	0.0000	-0.0344	0.0344	0.00118
1	2	2.33	2.981	0.2884	0.3754	-0.0871	0.00758
2	2.67	4	4.071	0.4950	0.5127	-0.0177	0.00031
3	1	0.33	1.354	0.0408	0.1705	-0.1297	0.01682
4	2.5	4.33	3.794	0.5359	0.4779	0.0580	0.00336
5	3	5	4.608	0.6188	0.5803	0.0385	0.00148
6	0	0	-0.27	0.0000	-0.0344	0.0344	0.00118
7	0	0	-0.27	0.0000	-0.0344	0.0344	0.00118

We calculated the Residual as the difference between the normalized values for the actual and predicted BWC.

Standard Error of Residuals (SER) is the Square root of the Sum of the Squares of the Residuals of the normalized values

In the example above, SER = 0.18, which is reasonably small in a scale of 0 – 1, the range for the normalized values

References for LCC'DC and Correlation Analysis

N. Meghanathan, "A Computationally-Lightweight and Localized Centrality Metric in lieu of Betweenness Centrality for Complex Network Analysis," *Springer Vietnam Journal of Computer Science*, vol. 4, no. 1, pp. 23-38, February 2017. DOI: 10.1007/s40595-016-0073-1.

http://link.springer.com/journal/40595/4/1/page/1

N. Meghanathan and X. He, "Correlation and Regression Analysis for Node Betweenness Centrality," *International Journal of Foundations in Computer Science and Technology*, vol. 6, no. 6, pp. 1-20, November 2016.

http://wireilla.com/papers/ijfcst/V6N6/6616ijfcst01.pdf

Link Analysis-based Ranking

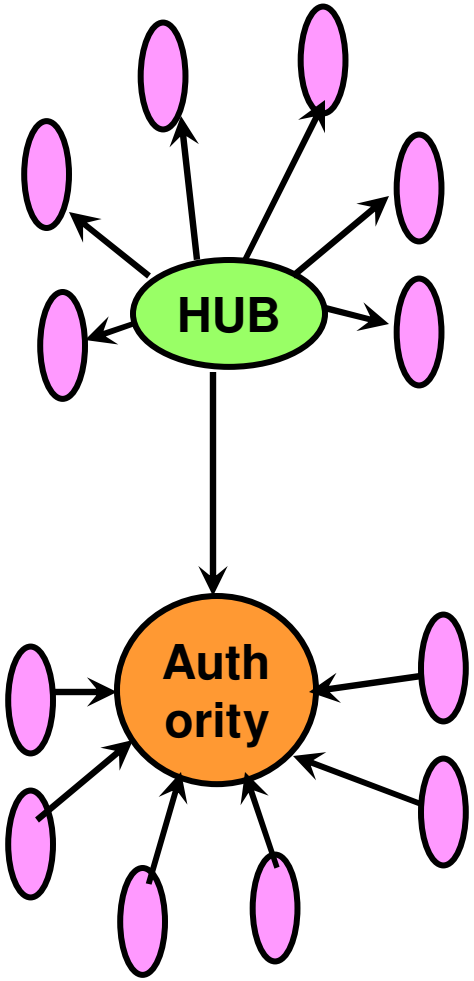
- We want to rank a node in a graph based on the number of edges pointing to it and/or leaving it as well as based on the rank of the nodes at the other end of these edges.

- Used primarily in web search
  - We model the web as a graph: the pages as nodes and the edges are directed edges – a page citing (having a link to) another page.

- Hubs and Authorities (HITS) algorithm
- PageRank algorithm

Hypertext Induced Topic Search (HITS) Algorithm

- **Hub:** Node that points to lots of pages
  - Yahoo like directory
- **Authority:** Node to which several other nodes point to
  - The larger the number of nodes pointing to a node, the more authoritative is the view presented by a node on a particular subject
- The HITS algorithm assigns **two scores for each page**:
  - **Authority:** an estimate of the value of the contents of the page
  - **Hub:** an estimate of the value of its links to other pages



- A page is considered to be **more authoritative** if it is referenced by many hub pages that are relevant to a search query
- A page is a **hub page** for a search query if it points to many authoritative pages for that query

- Good **authoritative** and **hub** pages reinforce one another.

A variant of HITS is used by Ask.com

Finding Pages for a Query in HITS

- **Initial Work**
- Step 1: Submit query *q* to a similarity-based engine and record a root set RS(q) pages.
- Step 2: Expand set RS(q) into the base set BS(q) to include pages pointed by RS(q) pages
- Step 3: Also include into BS(q), the pages pointing to RS(q) pages.
- **Run the HITS algorithm**
  - For each page pj, compute the authority and hub score of pj through a sequence of iterations.
- **After obtaining the final authority and hub scores** for each page, display the search results in the decreasing order of the authority scores. Pages having zero authority scores (nodes with no incoming links – strictly hubs) are listed in the decreasing order of their hub scores.
  - Note: nodes that are strictly hubs still contribute to the authority of the nodes that it points to.

HITS Algorithm

- Let E be the set of links in BS(q) and a link from page pi to pj is denoted by the pair (i, j).
- A: Authority Update Step

$$a(p_j) = \sum_{(i,j)\in E} h(p_i)$$

H: Hub Update Step

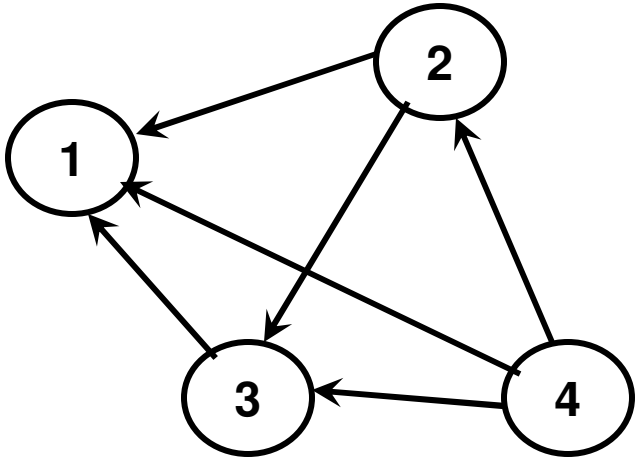
$$h(p_j) = \sum_{(j,k)\in E} a(p_k)$$

- After each iteration i, we scale the 'a' and 'h' values:

$$a^{(i)}(p_j) = \frac{a^{(i)}(p_j)}{\sqrt{\sum_k \left(a^{(i)}(p_k)\right)^2}}$$

$$h^{(i)}(p_j) = \frac{h^{(i)}(p_j)}{\sqrt{\sum_k \left(h^{(i)}(p_k)\right)^2}}$$

- As can be noted above, the two steps are interwined: one uses the values computed from the other.
  - In this course, we will follow the asynchronous mode of computation, according to which the authority values are updated first for a given iteration i and then the hub values are updated.
    - The hub values at iteration *i* are using the authority values just computed in iteration *i* (rather than iteration *i* – 1).



HITS Example (1)

Initial
a = [1 1 1 1] **h = [1 1 1 1]**

It # 1
a = [3 1 2 0] **h = [0 5 3 6]**
After Normalization,
a = [0.80 0.27 0.53 0] **h = [0 0.59 0.36 0.72]**

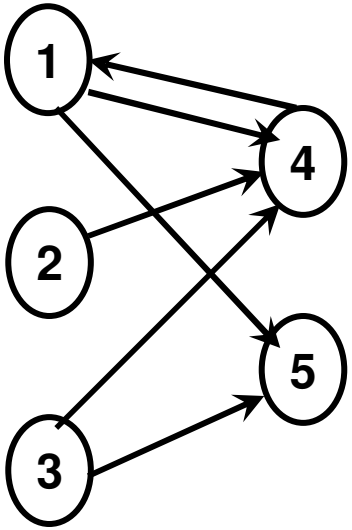
It # 2
a = [1.67 0.72 1.31 0] **h = [0 2.98 1.67 3.7]**
After Normalization,
a = [0.745 0.32 0.58 0] **h = [0 0.59 0.33 0.73]**

It #3
a = [1.65 0.73 1.32 0] **h = [0 2.97 1.65 3.7]**
After Normalization,
a = [0.74 0.32 0.59 0] **h = [0 0.59 0.33 0.73]**

Order Pages Listed after Search

1
3
2
4

HITS Example (2)



Initial
a = [1 1 1 1 1] **h = [1 1 1 1 1]**

It # 1
a = [1 0 0 3 2] **h = [5 3 5 1 0]**
After Normalization,
a = [0.26 0 0 0.80 0.53] **h = [0.64 0.38 0.64 0.12 0]**

It # 2
a = [0.12 0 0 1.66 1.28] **h = [2.94 1.66 2.94 0.12 0]**
After Normalization,
a = [0.057 0 0 0.79 0.61] **h = [0.66 0.37 0.66 0.027 0]**

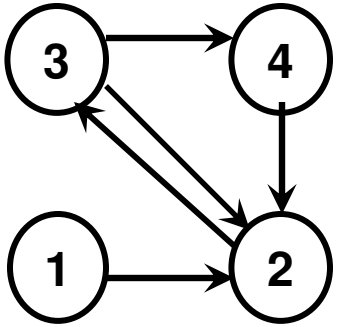
It # 3
a = [0.027 0 0 1.69 1.32] **h = [3.01 1.69 3.01 0.027 0]**
After Normalization,
a = [0.0126 0 0 0.79 0.61] **h = [0.66 0.37 0.66 0.006 0]**

It # 4
a = [0.006 0 0 1.69 1.32] **h = [3.01 1.69 3.01 0.006 0]**
After Normalization,
a = [0.003 0 0 0.79 0.61] **h = [0.66 0.37 0.66 0.001 0]**

Order Pages Listed after Search

4
5
1
3
2

HITS Example (3)



Initial
a = [1 1 1 1] h = [1 1 1 1]

It # 1
a = [0 3 1 1] h = [3 1 4 3]
After Normalization,
a = [0 0.91 0.30 0.30] h = [0.51 0.17 0.68 0.51]

It # 2
a = [0 1.70 0.17 0.68] h = [1.70 0.17 2.38 1.70]
After Normalization,
a = [0 0.92 0.09 0.37] h = [0.50 0.05 0.70 0.50]

It # 3
a = [0 1.70 0.05 0.70] h = [1.70 0.05 2.4 1.70]
After Normalization,
a = [0 0.92 0.027 0.38] h = [0.50 0.014 0.70 0.50]

It # 4
a = [0 1.70 0.014 0.70] h = [1.70 0.014 2.4 1.70]
After Normalization,
a = [0 0.92 0.008 0.38] h = [0.50 0.004 0.71 0.50]

Order Pages Listed after Search

2
4
3
1

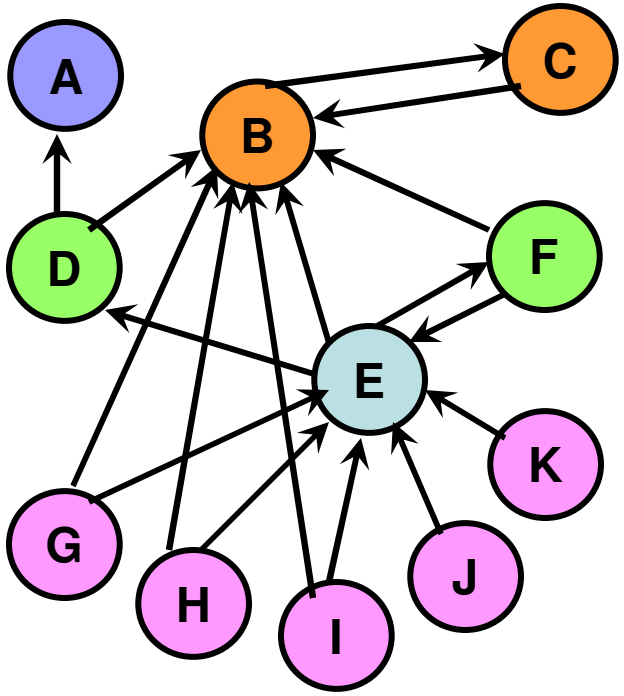
PageRank

- The basic idea is to analyze the link structure of the web to figure out which pages are more authoritative (important) in terms of quality.
- It is a content-independent scheme.
- If Page A has a hyperlink to Page B, it can be considered as a vote of A for B.
  - If multiple pages link to B, then page B is likely to be a good page.
- A page is likely to be good if several other good pages link to it (a bit of recursive definition).
  - Not all pages that link to B are of equal importance.
  - A single link from CNN or Yahoo may be worth several times
- The web pages are first searched based on the content. The retrieved web pages are then listed based on their rank (computed on the original web, unlike HITS that is run on a graph of the retrieved pages).
- The Page Rank of the web pages are indexed (recomputed) for every regular time period.

PageRank (Random Web Surfer)

- Web – graph of pages with the hyperlinks as directed edges.
- Analogy used to explain PageRank algorithm (Random Web Surfer)
- User starts browsing on a random page
- Picks a random out-going link listed in that page and goes there (with a probability ‘d’, also called damping factor)
  - Repeated forever
- The surfer jumps to a random page with probability 1-d.
  - Without this characteristic, there could be a possibility that someone could just end up oscillating between two pages B and C as in the traversing sequence below for the graph shown aside:

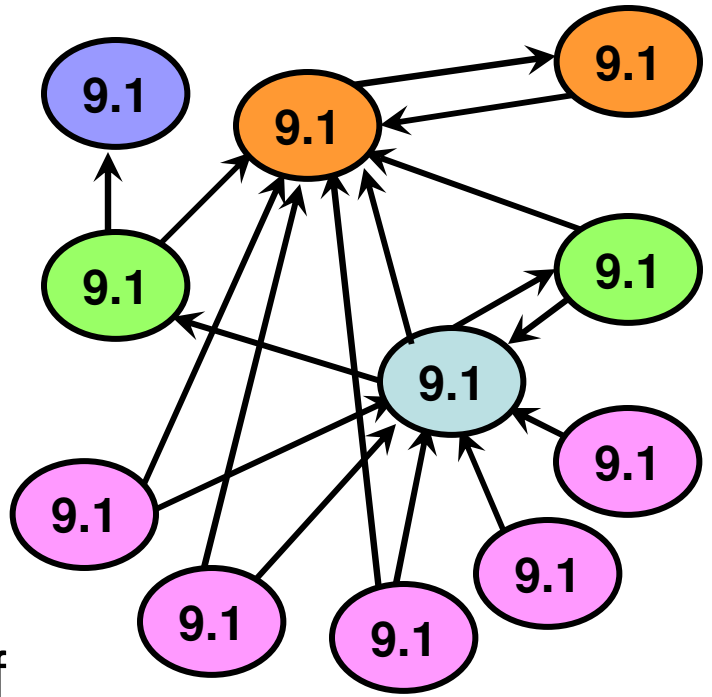
    G → E → F → E → D → B → C



Lets say d = 0.85.
To decide the next page to move, the surfer simply generates a random number, r. If r <= 0.85, then the surfer randomly chooses an out-going link from the existing page. Otherwise, jumps to a randomly chosen page among all the pages, including the current page.

PageRank Algorithm

- PageRank of Page X is the probability that the surfer is at page X at a randomly selected time.
  - Basically the proportion of time, the surfer would spend at page X.
- **PageRank Algorithm**
- **Initial:** Every node in the graph gets the same pagerank. PR(X) = 100% / N, where N is the number of nodes.
- At any time, at the end of each iteration, the page rank of all nodes add up to 100%.
- Actually, the initial pagerank value of a node is the pagerank at any time, if there are no edges in the graph. We have 100% / N chance of jumping to any node in the graph at any time.



Initial PageRank of Nodes

PageRank Algorithm

Page Rank of Node X

$$PR(x) = \frac{(1-d)*100}{N} + d\sum_{y->x}\frac{PR(y)}{Out(y)}$$

Assuming there are NO Sink nodes

- Page Rank of Node X is the probability of being at node X at the current time.
- How can we visit node X from where we are?
  - **(1-d) term: Random Jump:** The probability of ending up at node X because of a random jump from some node, including node X, is 1/N.
  - However, such a random jump itself could occur with a probability of (1-d).
  - This amounts to a probability of (1-d)/N to be at node X due to a random jump.

PageRank Algorithm

Page Rank of Node X

$$PR(x) = \frac{(1-d)*100}{N} + d\sum_{y->x}\frac{PR(y)}{Out(y)}$$

Assuming there are NO Sink nodes

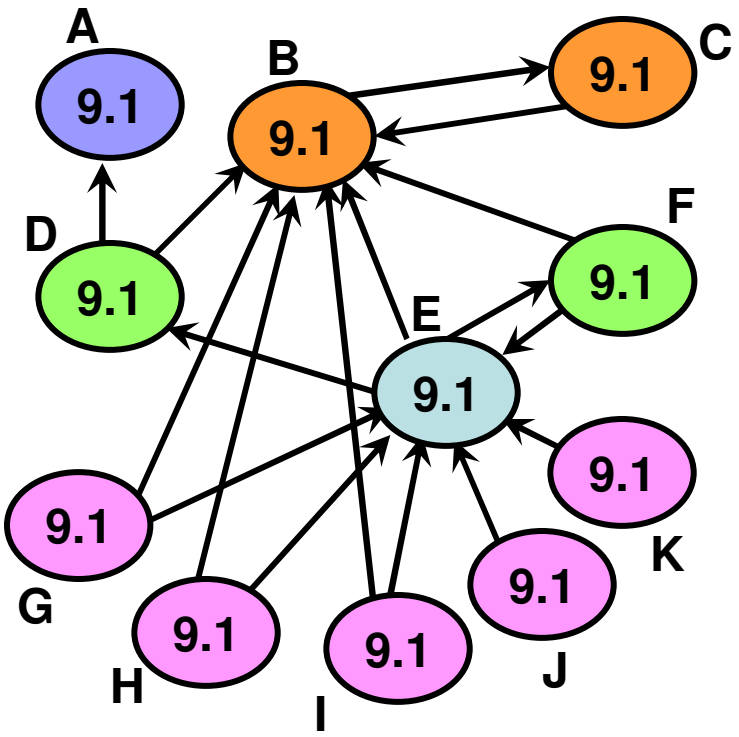
- Page Rank of Node X is the probability of being at node X at the current time.
- How can we visit node X from where we are?
  - **d term: Edge Traversal from a Neighbor:**
  - We could visit node X from one of the nodes that point to node X.
  - Lets say, we are at node Y in the previous iteration. The probability of being at node Y in the previous iteration is PR(Y). We can visit any of Y's neighbors.
  - The probability of visiting node X among the Out(Y) out-going links of node Y is PR(Y) * (1 / Out(Y)) = PR(Y) / Out(Y).
  - Likewise, we could visit X from any of its neighbors.
  - All the probabilities of visiting X from any of its neighbors have to be added, because visiting X from any of its neighbors is independent of the neighbors.
  - The whole event of visiting from a neighbor occurs with a prob. 'd'

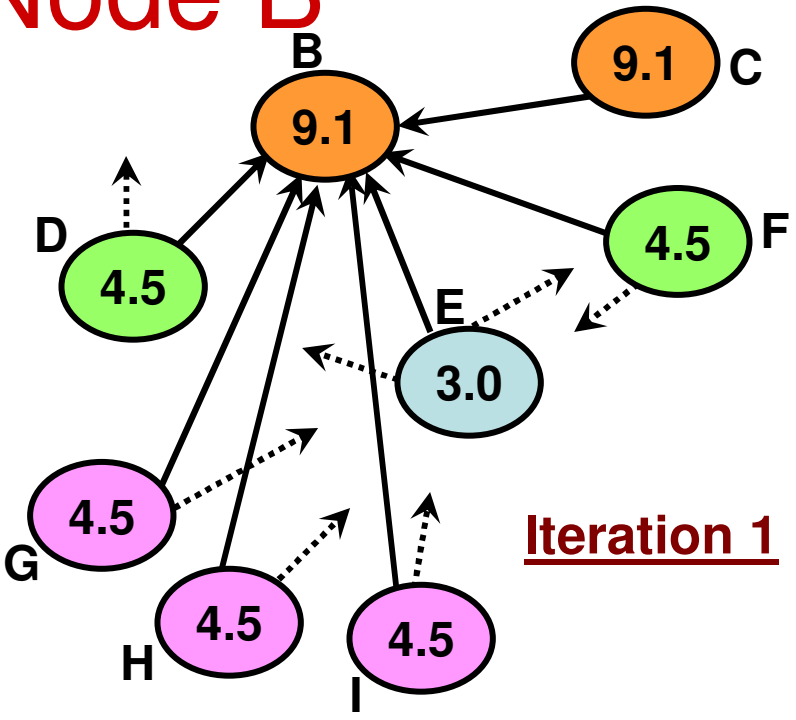
PageRank

- Since Page Rank PR(X) denotes the probability of being at node X at any time, the sum of the Page Ranks of all the nodes at any time should be equal to 1.
- We can also interpret the traversal from a node Y to node X as node Y contributing a part of its PR to node X (node Y equally shares its PR to the nodes connected to it through its out-going links).
- Implementation:
  - Note that (unlike HITS) we need to use the page rank values of the nodes from the previous iteration to update the page rank values of the nodes in the current iteration.
    - **Need to maintain two arrays at any time t: $PR^{(t-1)}$ and $PR^{(t)}$**

Calculating PageRank of Node B

Initial PageRank of Nodes





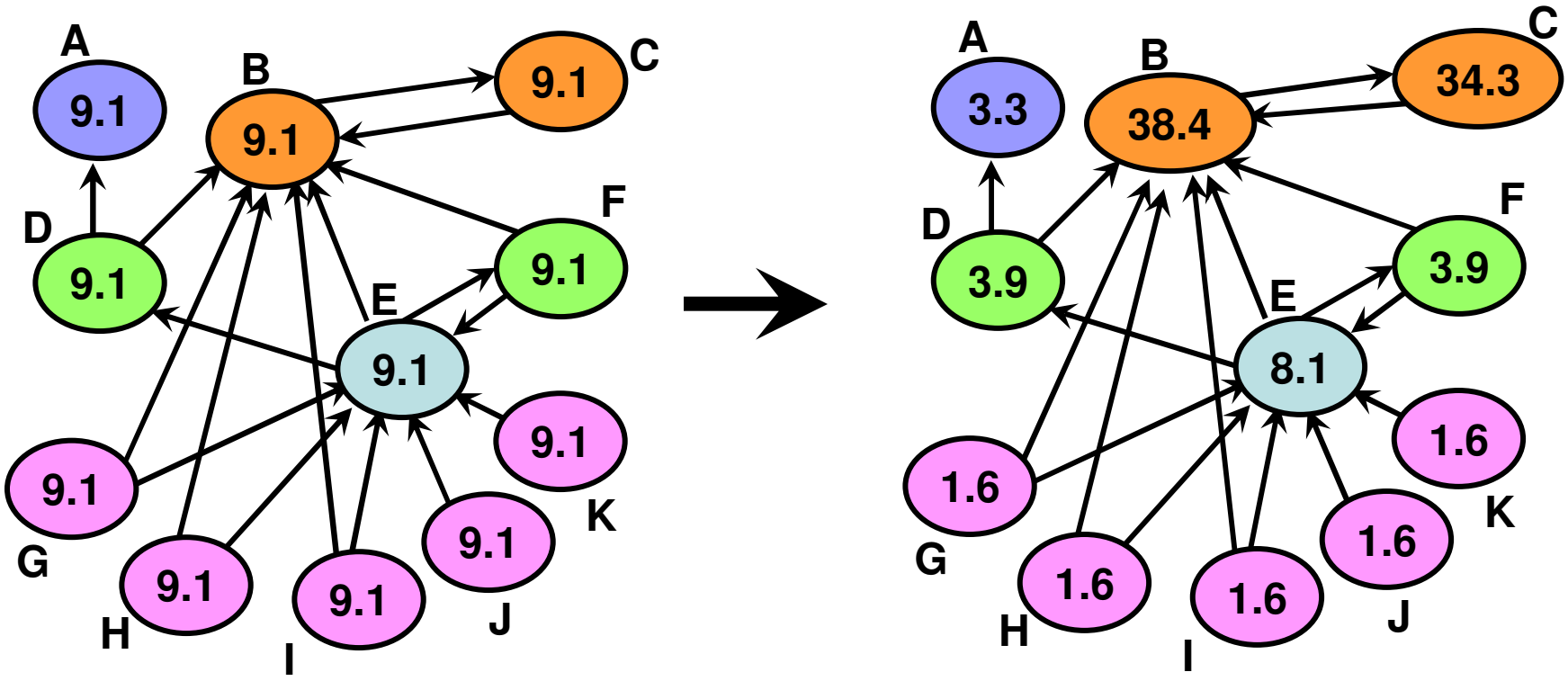
Assume the damping factor d = 0.85

For any iteration,

PR(B) = 0.15 * 9.1 +
0.85 * [PR(C) + ½ PR(D) +
⅓ PR(E) + ½ PR(F) +
½ PR (G) + ½ PR(H) + ½ PR(I)]

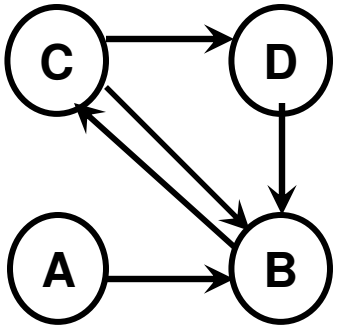
For Iteration 1,
Substituting the PR values of the nodes (initial values), we get PR(B) ≈ 31

Final PageRank Values for the Sample Graph



PageRank: More Observations

- Algorithm converges (few iterations sufficient)
- For an arbitrary graph, it is pretty difficult to figure out the final page rank values of the nodes.
- Certain inferences could be however made.
- For our sample graph:
  - For nodes that do not have any in-links pointing to them, the only way we will end up at these nodes is through a random jump: this happens with a probability (1-d)/N. In our case, it is (1-0.85)* 100/11 = 1.6%.
  - Two nodes with links from the same node (symmetric in-links) have the same PR. (nodes D and F) and it will be higher than those nodes without any in-links.
  - One in-link from a node with high PR value contributes significantly to the PR value of a node compared to the in-links from several low PR nodes.
    - In our sample graph, an in-link from node B contributes significantly for node C compared to the several in-links that node E gets from the low-PR nodes. So, the quality of the in-links matters more than the number of in-links.



Note that there are NO sink nodes (nodes without any out-going links)

Assume damping Factor d = 0.85

PR(A) = (1-d)*100/4
PR(B) = (1-d)*100/4 + d*[PR(A) + 1/2 * PR(C) + PR(D)]
PR(C) = (1-d)*100/4 + d*[PR(B)]
PR(D) = (1-d)*100/4 + d*[1/2*PR(C)]

Initial	It # 1	It # 2	It # 3	It # 4
PR(A) = 25	PR(A) = 3.75	PR(A) = 3.75	PR(A) = 3.75	PR(A) = 3.75
PR(B) = 25	PR(B) = 56.88	PR(B) = 29.79	PR(B) = 41.30	PR(B) = 41.29
PR(C) = 25	PR(C) = 25	PR(C) = 52.10	PR(C) = 29.07	PR(C) = 38.86
PR(D) = 25	PR(D) = 14.38	PR(D) = 14.38	PR(D) = 25.89	PR(D) = 16.10

It # 5	It # 6	It # 7	It # 8	It # 9
PR(A) = 3.75	PR(A) = 3.75	PR(A) = 3.75	PR(A) = 3.75	PR(A) = 3.75
PR(B) = 37.14	PR(B) = 40.68	PR(B) = 39.17	PR(B) = 39.17	PR(B) = 39.71
PR(C) = 38.85	PR(C) = 35.32	PR(C) = 38.33	PR(C) = 37.04	PR(C) = 37.04
PR(D) = 20.27	PR(D) = 20.26	PR(D) = 18.76	PR(D) = 20.04	PR(D) = 19.49

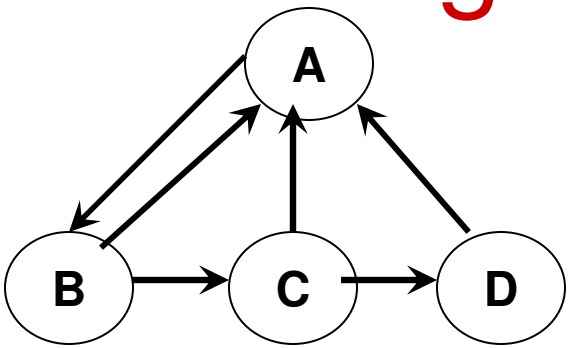
It # 10
PR(A) = 3.75
PR(B) = 39.25
PR(C) = 37.5
PR(D) = 19.49

Ranking

1. B
2. C
3. D
4. A

Page Rank Example (1)

Page Rank Example (2)



PR(A) = (1-d)*100/4 + d*[½*PR(B) + ½*PR(C) + PR(D)]
PR(B) = (1-d)*100/4 + d*[PR(A)]
PR(C) = (1-d)*100/4 + d*[½*PR(B)]
PR(D) = (1-d)*100/4 + d*[½*PR(C)]

Initial	
PR(A)	25
PR(B)	25
PR(C)	25
PR(D)	25

It # 1	
PR(A)	46.25
PR(B)	25
PR(C)	14.38
PR(D)	14.38

It # 2	
PR(A)	32.71
PR(B)	43.06
PR(C)	14.38
PR(D)	9.86

It # 3	
PR(A)	36.54
PR(B)	31.55
PR(C)	22.05
PR(D)	9.86

It # 4	
PR(A)	34.91
PR(B)	34.81
PR(C)	17.16
PR(D)	13.12

It # 5	
PR(A)	36.99
PR(B)	33.42
PR(C)	18.54
PR(D)	11.04

It # 6	
PR(A)	35.22
PR(B)	35.12
PR(C)	17.95
PR(D)	11.63

It # 7	
PR(A)	36.19
PR(B)	33.68
PR(C)	18.68
PR(D)	11.38

It # 8	
PR(A)	35.68
PR(B)	34.51
PR(C)	18.06
PR(D)	11.69

It # 9	
PR(A)	36.03
PR(B)	34.08
PR(C)	18.42
PR(D)	11.43

Node Ranking: A B C D

Page Rank: Graph with Sink Nodes Motivating Example

- Consider the graph: A → B
- Let d = 0.85
- PR(A) = 0.15*100/2 PR(B) = 0.15*100/2 + 0.85*PR(A)
- Initial: PR(A) = 50, PR(B) = 50
- Iteration 1:
  - PR(A) = 0.15*100/2 = 7.5
  - PR(B) = 0.15*100/2 + 0.85 * 50 = 50.0
  - PR(A) + PR(B) = 57.5
  - Note that the PR values do not add up to 100.
  - This is because, B is not giving back the PR that it receives from A to any other node in the graph. The (0.85*50 = 42.5) value of PR that B receives from A is basically lost.
  - Once we get to B, there is no way to get out of B other than random jump to A and this happens only with probability (1-d).

Page Rank: Sink Nodes (Solution)

- Assume implicitly that the sink node is connected to every node in the graph (including itself).
  - The sink node equally shares its PR with every node in the graph, including itself.
  - If *z* is a sink node, with the above scheme, out(z) = N, the number of nodes in the graph.
- The probability of getting to node X at a given time is still the two terms below:
  - Random jump from any node (probability, 1-d)
  - Visit from a node with in-link to node X (probability, d)

Page Rank of Node X

$$PR(x) = \frac{(1-d)*100}{N} + d\sum_{y->x}\frac{PR(y)}{Out(y)} + d\sum_{z->\varphi}\frac{PR(z)}{N}$$

Explicit out-going links to certain nodes

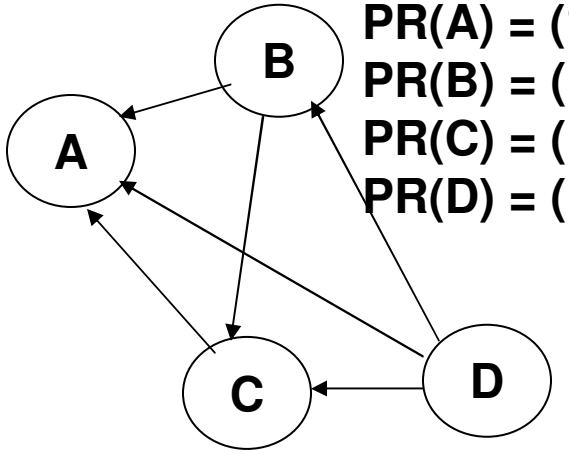
Implicit out-going links to all nodes (sink nodes)

the second term of the original Page Rank formula is now broken between that of nodes with explicit out-going links to one or more selected nodes and the sink nodes with implicit out-going links to all nodes.

Consolidated PageRank Formula

$$PR(x) = \frac{(1-d)*100}{N} + d\sum_{y->x}\frac{PR(y)}{Out(y)} + d\sum_{z->\varphi}\frac{PR(z)}{N}$$

Page Rank Example (3)



PR(A) = (1-d)*100/4 + d [PR(B)/2 + PR(C)/1 + PR(D)/3] + (d)*[PR(A)/4]
PR(B) = (1-d)*100/4 + d [PR(D)/3] + (d)*[PR(A)/4]
PR(C) = (1-d)*100/4 + d [PR(B)/2 + PR(D)/3] + (d)*[PR(A)/4]
PR(D) = (1-d)*100/4 + (d)*[PR(A)/4]

Node Ranking: A, C, B, D

Initial	
PR(A)	25
PR(B)	25
PR(C)	25
PR(D)	25

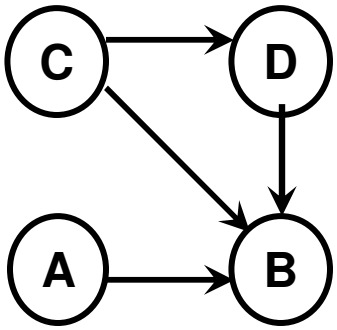
It # 1	
PR(A)	48.02
PR(B)	16.15
PR(C)	26.77
PR(D)	9.063

It # 2	
PR(A)	46.14
PR(B)	16.52
PR(C)	23.386
PR(D)	13.954

It # 3	
PR(A)	44.41
PR(B)	17.51
PR(C)	24.53
PR(D)	13.55

It # 4	
PR(A)	45.32
PR(B)	17.03
PR(C)	24.47
PR(D)	13.18

Page Rank Example (4)



PR(A) = 0.15*100/4 + 0.85*{PR(B)/4}
PR(B) = 0.15*100/4 + 0.85*{PR(A) + PR(C)/2 + PR(D)} +
0.85*{PR(B)/4}
PR(C) = 0.15*100/4 + 0.85*{PR(B)/4}
PR(D) = 0.15*100/4 + 0.85*{PR(C)/2} + 0.85*{PR(B)/4}

		Iterations							
	Initial	1	2	3	4	5	6	7	8
A	25	9.06	16.96	13.37	14.95	14.26	14.56	14.43	14.49
B	25	62.19	45.25	52.69	49.48	50.85	50.27	50.52	50.41
C	25	9.06	16.96	13.37	14.95	14.26	14.56	14.43	14.49
D	25	19.69	20.82	20.58	20.63	20.62	20.62	20.62	20.62

Rank
B
D
A, C

The example shows that though both A and C differ with respect to the number of outgoing edges, their Page Rank value is the same.

The Page Rank of a vertex is dependent only on the incoming edges and where they are from.