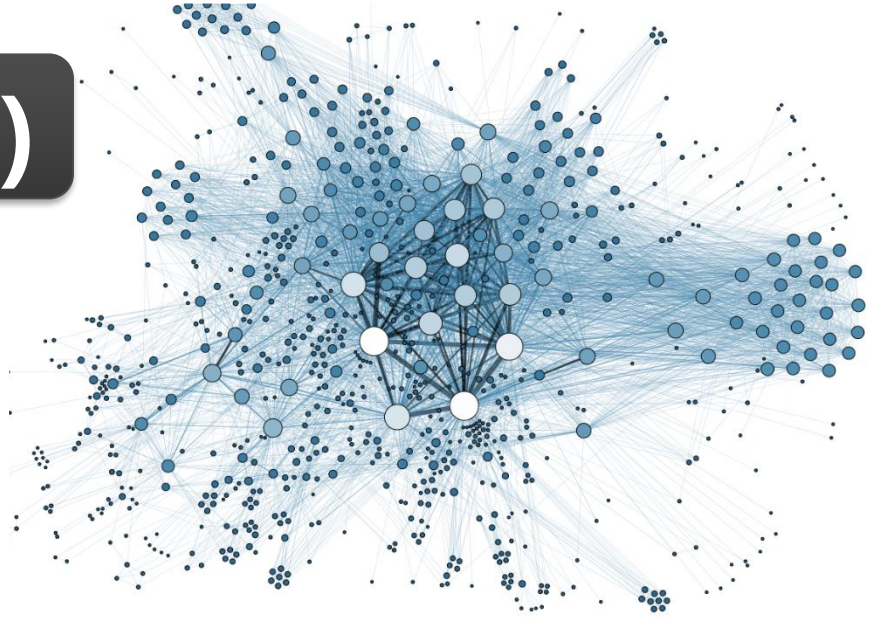
Social Media Analytics

INFS7450

Node Measures (II)



Prof. Hongzhi Yin

School of EECS

The University of Queensland

Spectral Centrality Measures

- Eigenvector Centrality
- Kat's Index
- PageRank
- Hits

PageRank

- Problem with Katz and Eigenvector Centralities:
  - In directed graphs, once a node becomes an authority (high centrality), it passes **all** its centrality to each of its neighbors
  - This is less desirable since not everyone known by a well-known person is well-known, e.g., a fan of a well-known actor

- **Solution?**
  - We can divide the value of passed centrality by the number of outgoing links, i.e., out-degree of that node
  - Each connected neighbor gets a fraction of the source node's centrality

PageRank, cont.

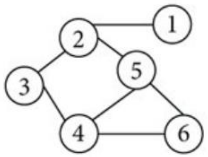
$$C_p(v_i) = \alpha \sum_{j=1}^{n} A_{j,i} \frac{C_p(v_j)}{d_j^{\text{out}}} + \beta$$

What if the degree is zero?

$$\begin{cases} d_j^{out} > 0 \\ D = diag(d_1^{\text{out}}, d_2^{\text{out}}, \ldots, d_n^{\text{out}}) \end{cases}$$

$$\mathbf{C}_p = \alpha A^T D^{-1} \mathbf{C}_p + \beta \mathbf{1}$$

$$\mathbf{C}_p = \beta (\mathbf{I} - \alpha A^T D^{-1})^{-1} \cdot \mathbf{1}$$

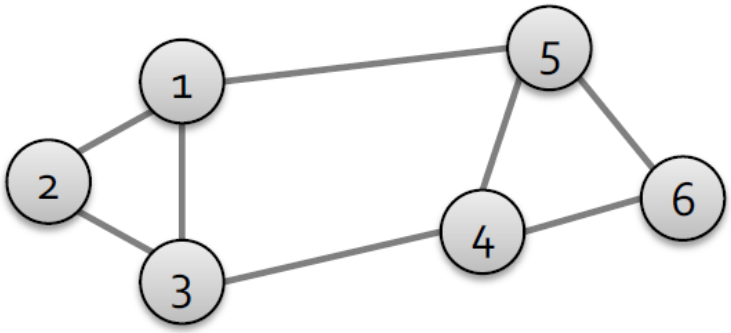


$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

Similar to Katz Centrality, $\alpha < 1/\lambda$, where λ is the largest eigenvalue of $A^T D^{-1}$.

Degree Matrix

- **Degree matrix (D):**
  - $n \times n$ diagonal matrix
  - $D=[d_{ii}]$, d_{ii} = degree of node i



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

What is the inverse of this degree matrix?

Degree Matrix

The inverse of matrix D will also be a diagonal $n \times n$ matrix in the following form:

$$D^{-1} = \begin{bmatrix} \frac{1}{d_1} & 0 & \cdots & 0 \\ 0 & \frac{1}{d_2} & 0 & 0 \\ \vdots & 0 & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{d_n} \end{bmatrix}$$

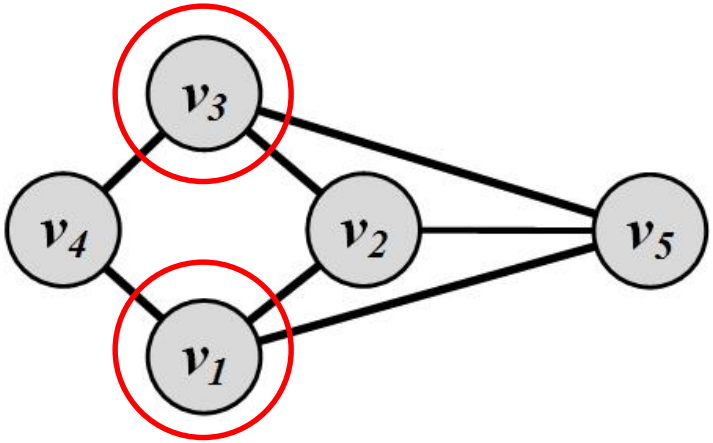
What if the out-degree is zero?

When $d_j^{\text{out}} = 0$, we know that since the out-degree is zero, $\forall i, A_{j,i} = 0$. This makes the term inside the summation $\frac{0}{0}$. We can fix this problem by setting $d_j^{\text{out}} = 1$ since the node will not contribute any centrality to any other nodes.

$$C_p(v_i) = \alpha \sum_{j=1}^{n} A_{j,i} \frac{C_p(v_j)}{d_j^{\text{out}}} + \beta$$

PageRank Example

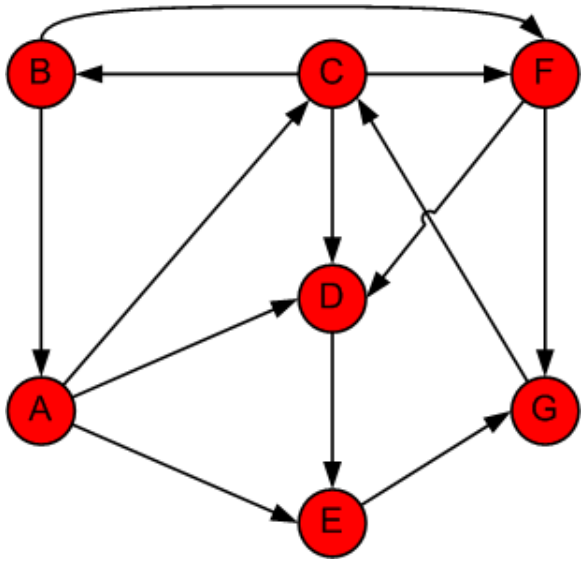
- We assume α=0.95 < 1 and and $\beta = 0.1$



$$A = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \end{bmatrix} \quad D = \begin{bmatrix} 3 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 3 \end{bmatrix}$$

$$\mathbf{C}_p = \beta(\mathbf{I} - \alpha A^T D^{-1})^{-1} \cdot \mathbf{1} = \begin{bmatrix} 2.14 \\ 2.13 \\ 2.14 \\ 1.45 \\ 2.13 \end{bmatrix}$$

PageRank Example



α=1 and β =0

Using Power Iteration Method

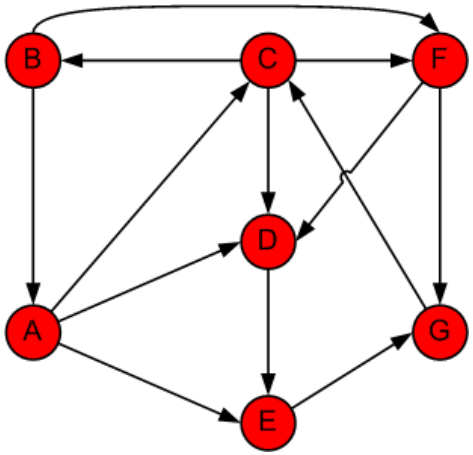
Referring to the power iteration algorithm for Kats Centrality. Just replace the adjacency matrix by $A^T D^{-1}$

Step	A	B	C	D	E	F	G
0	1/7	1/7	1/7	1/7	1/7	1/7	1/7
	B/2	C/3	A/3 + G	A/3 + C/3 + F/2	A/3 + D	C/3 + B/2	F/2 + E
	0.071	0.048	0.190	0.167	0.190	0.119	0.214

PageRank: Example

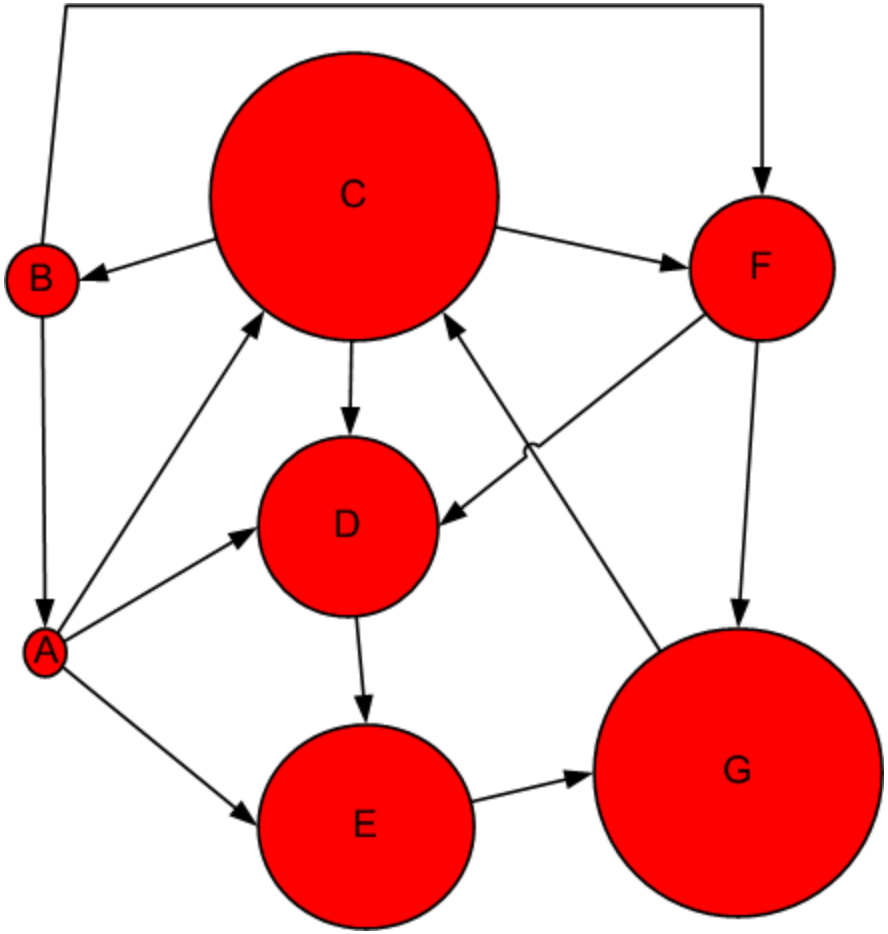
Step	A	B	C	D	E	F	G	Sum
1	0.143	0.143	0.143	0.143	0.143	0.143	0.143	1.000
2	0.071	0.048	0.190	0.167	0.190	0.119	0.214	1.000
3	0.024	0.063	0.238	0.147	0.190	0.087	0.250	1.000
4	0.032	0.079	0.258	0.131	0.155	0.111	0.234	1.000
5	0.040	0.086	0.245	0.152	0.142	0.126	0.210	1.000
6	0.043	0.082	0.224	0.158	0.165	0.125	0.204	1.000
7	0.041	0.075	0.219	0.151	0.172	0.115	0.228	1.000
8	0.037	0.073	0.241	0.144	0.165	0.110	0.230	1.000
9	0.036	0.080	0.242	0.148	0.157	0.117	0.220	1.000
10	0.040	0.081	0.232	0.151	0.160	0.121	0.215	1.000
11	0.040	0.077	0.228	0.151	0.165	0.118	0.220	1.000
12	0.039	0.076	0.234	0.148	0.165	0.115	0.223	1.000
13	0.038	0.078	0.236	0.148	0.161	0.116	0.222	1.000
14	0.039	0.079	0.235	0.149	0.161	0.118	0.219	1.000
15	0.039	0.078	0.232	0.150	0.162	0.118	0.220	1.000
Rank	**7**	**6**	**1**	**4**	**3**	**5**	**2**	

Effect of PageRank



PageRank

Node	Rank
A	**7**
B	**6**
C	**1**
D	**4**
E	**3**
F	**5**
G	**2**



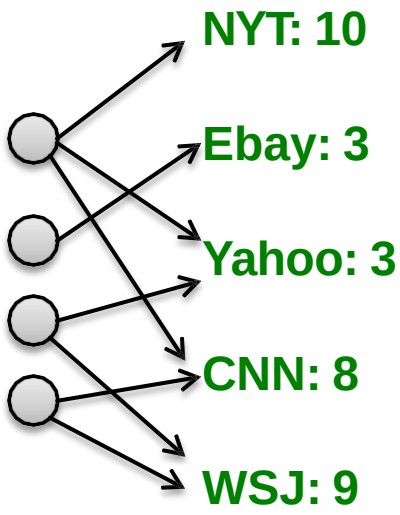
HITS Centrality

- **Spectral measures**
  - **HITS Centrality**: In a directed graph, a node is more important if it has more links
    - In-coming links? Out-going links?

- **Hubs and Authorities**

Each node has **2** scores:

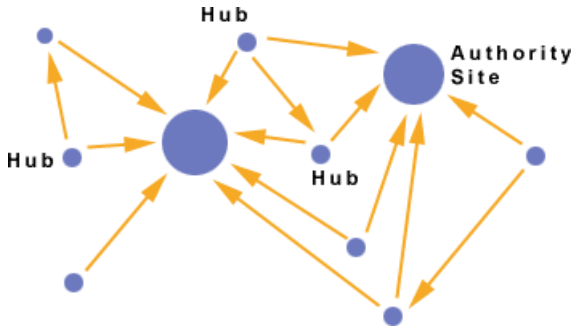
- **Quality as an expert (hub):**
  - Total sum of votes (authority scores) of nodes that it points to (out-going links)
- **Quality as a content provider (authority):**
  - Total sum of votes (hub scores) from nodes that point to it (in-coming links)



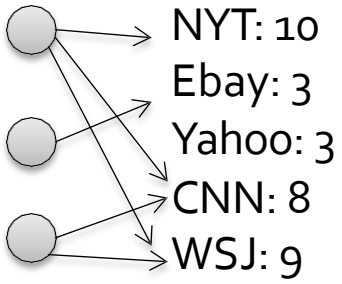
Hubs and Authorities

Two types of important web pages:

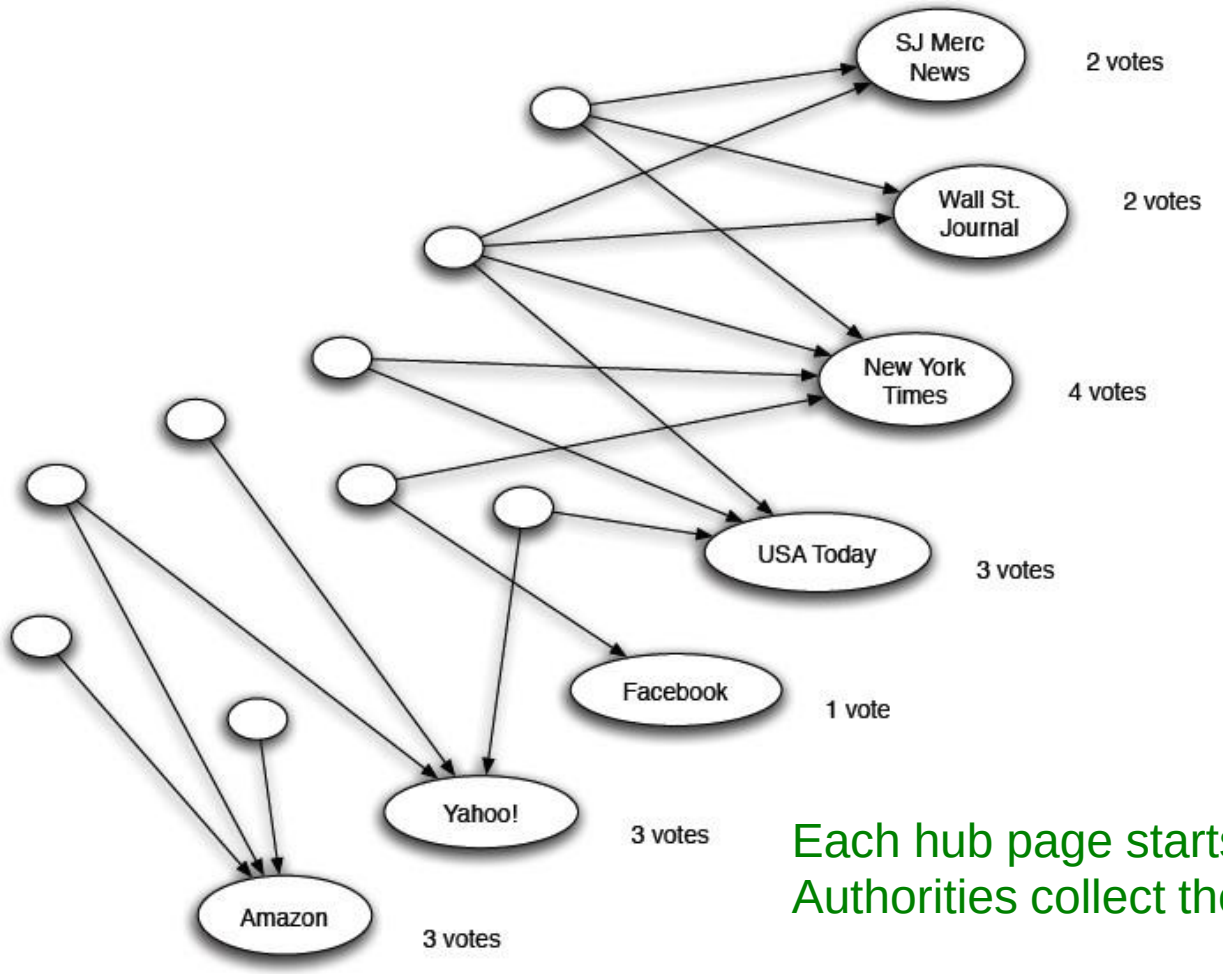
1. **Authorities** are nodes containing useful information
   - Newspaper home pages
   - Course home pages
   - Home pages of properties



2. **Hubs** are nodes that link to authorities
   - Yahoo Directory
   - Hao123.com
   - List of universities (e.g., QS World University Rankings)



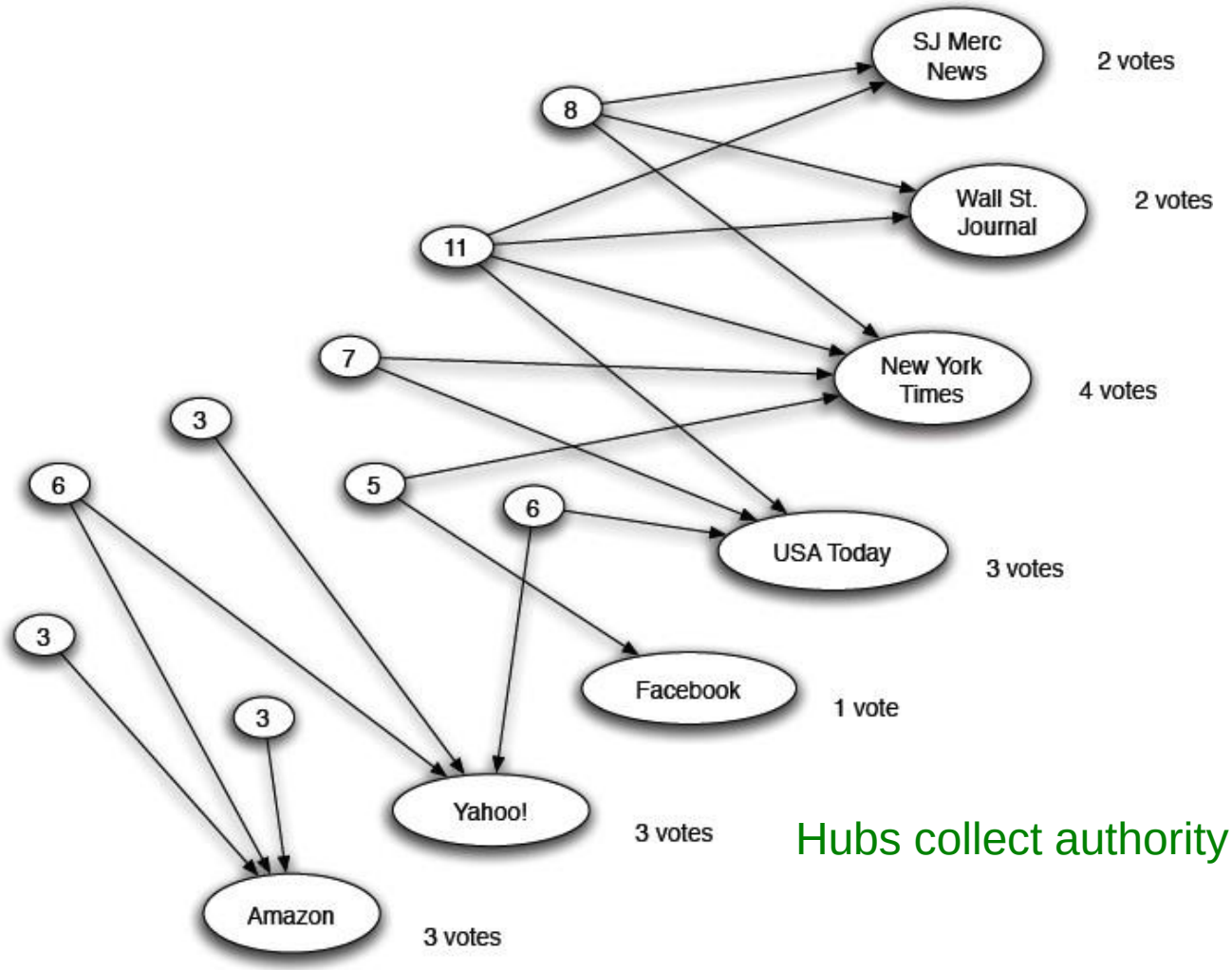
Counting In-links: Authority



Each hub page starts with **hub score 1**
Authorities collect their votes

(Note this is an idealized example. In reality web graph is not bipartite and each page has both the hub and the authority score)

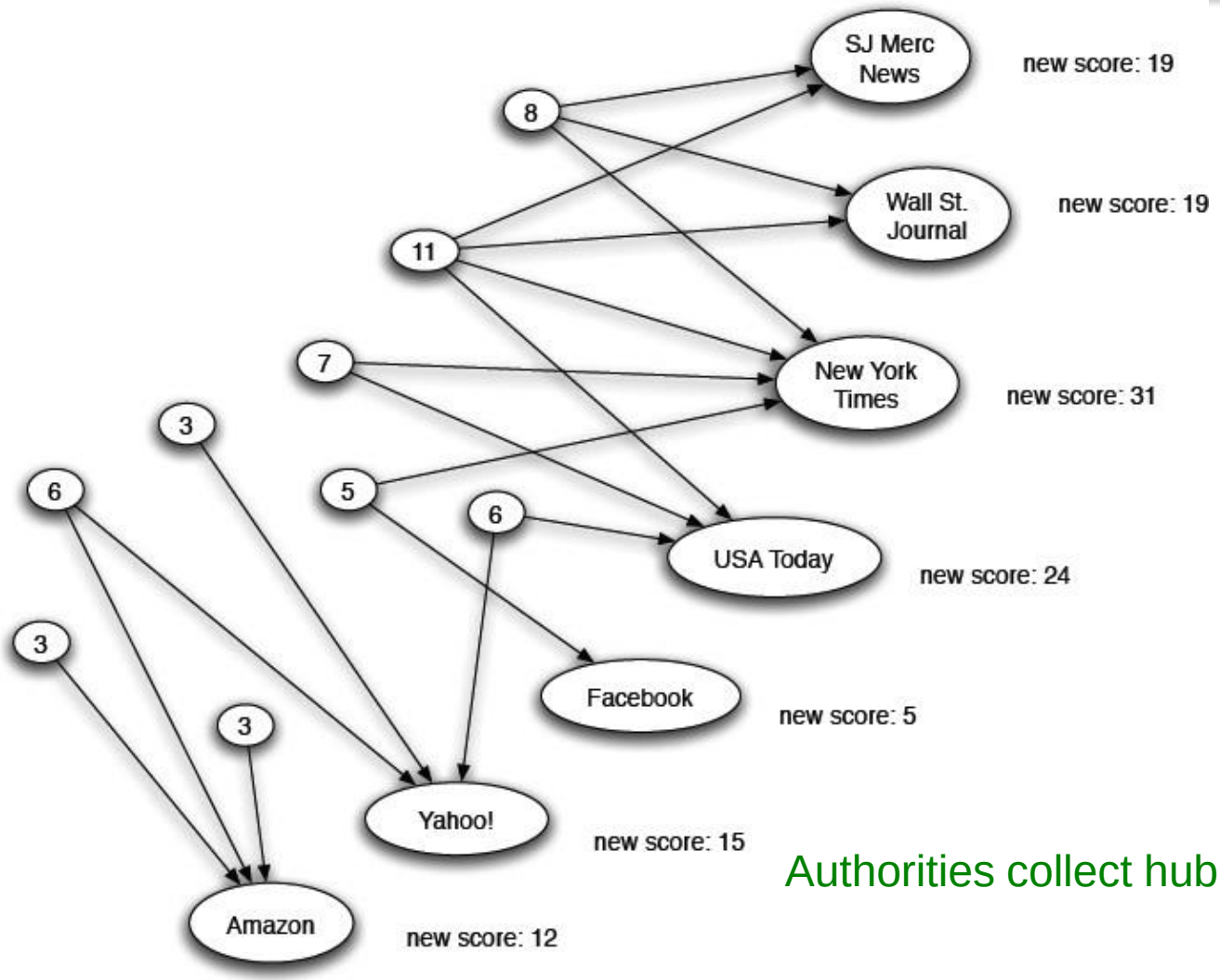
Expert Quality: Hub



Hubs collect authority scores

(Note this is an idealized example. In reality graph is not bipartite and each page has both the hub and authority score)

Update Authorities



Authorities collect hub scores

(Note this is an idealized example. In reality graph is not bipartite and each page has both the hub and authority score)

Mutually Recursive Definition

- **A good hub links to many good authorities**
- **A good authority is linked from many good hubs**
  - Note a self-reinforcing recursive definition
- **Model using two scores for each node:**
  - **Hub** score and **Authority** score
  - Represented as vectors $\boldsymbol{h}$ and $\boldsymbol{a}$, where the i-th element is the hub/authority score of the i-th node

HITS Centrality

- **Spectral measures**
  - **HITS Centrality**: mutual reinforcement
    - A page is authoritative if it is pointed by many good hubs (pages which contain good list of authoritative pages),
    - A hub is good if it points to authoritative pages.

$$c_{\text{aut}}(x) = \sum_{y \to x} c_{\text{hub}}(y)$$

$$c_{\text{hub}}(x) = \sum_{x \to y} c_{\text{aut}}(y)$$

Hubs and Authorities

- **Each page *i* has 2 scores:**
  - Authority score: a_i
  - Hub score: h_i

Convergence criteria:

$$\sum_i \left(h_i^{(t)} - h_i^{(t+1)}\right)^2 < \varepsilon$$

$$\sum_i \left(a_i^{(t)} - a_i^{(t+1)}\right)^2 < \varepsilon$$

HITS algorithm:

- Initialize: $a_j^{(0)} = 1/\sqrt{n}, \quad h_j^{(0)} = 1/\sqrt{n}$
- Then keep iterating until **convergence**:
  - $\forall i$: Authority: $a_i^{(t+1)} = \sum_{j \to i} h_j^{(t)}$
  - $\forall i$: Hub: $h_i^{(t+1)} = \sum_{i \to j} a_j^{(t)}$
  - $\forall i$: Normalize:

$$a_i^{(t+1)} = a_i^{(t+1)} / \sqrt{\sum_i \left(a_i^{(t+1)}\right)^2}$$

$$h_i^{(t+1)} = h_i^{(t+1)} / \sqrt{\sum_j \left(h_j^{(t+1)}\right)^2}$$

Hubs and Authorities

- **Hits in the vector notation:**
  - Vector $\boldsymbol{a} = (\boldsymbol{a_1} \ldots, \boldsymbol{a_n}), \quad \boldsymbol{h} = (\boldsymbol{h_1} \ldots, \boldsymbol{h_n})$
  - Adjacency matrix $\boldsymbol{A}$ (n x n): $\boldsymbol{A_{ij}} = \boldsymbol{1}$ if $\boldsymbol{i \rightarrow j}$
- **Can rewrite** $\boldsymbol{h_i} = \sum_{\boldsymbol{i \rightarrow j}} \boldsymbol{a_j}$ **as** $\boldsymbol{h_i} = \sum_{\boldsymbol{j}} \boldsymbol{A_{ij}} \cdot \boldsymbol{a_j}$
- **So:** $\boldsymbol{h} = \boldsymbol{A} \cdot \boldsymbol{a}$ **And similarly:** $\boldsymbol{a} = \boldsymbol{A^T} \cdot \boldsymbol{h}$
- **Repeat until convergence:**
  - $h^{(t+1)} = A \cdot a^{(t)}$
  - $a^{(t+1)} = A^T \cdot h^{(t)}$
  - Normalize $a^{(t+1)}$ and $h^{(t+1)}$

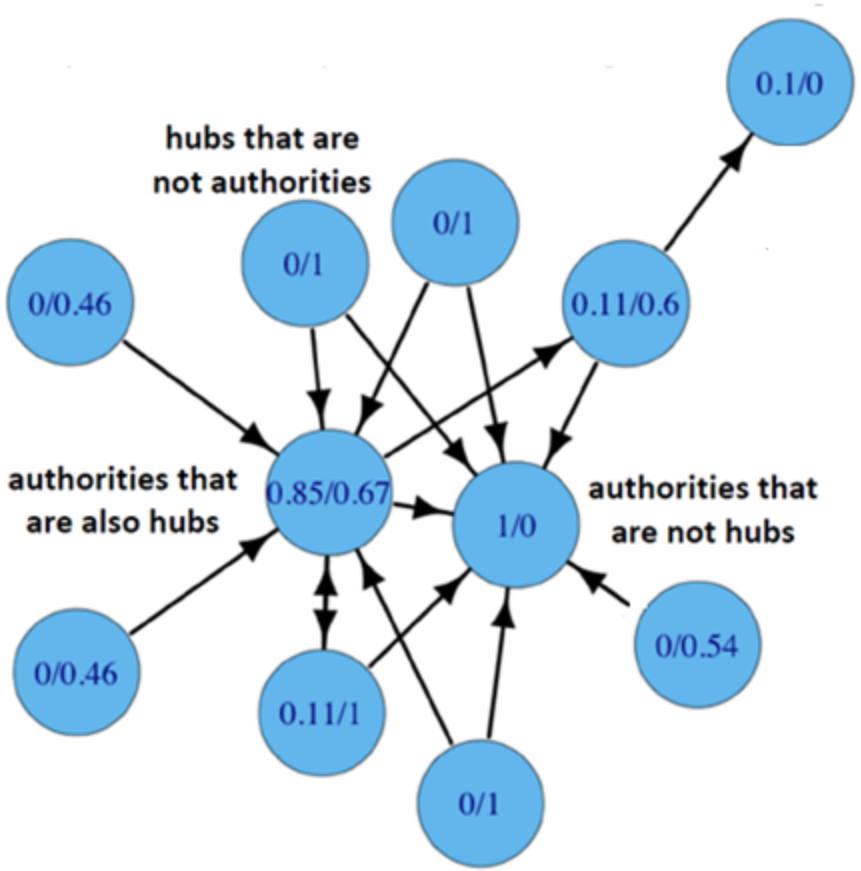
Convergence criteria:

$$\sum_i \left(h_i^{(t)} - h_i^{(t+1)}\right)^2 < \varepsilon$$

$$\sum_i \left(a_i^{(t)} - a_i^{(t+1)}\right)^2 < \varepsilon$$

HITS Centrality Example

- A directed graph with nodes labelled with their authority followed by their hub centralities

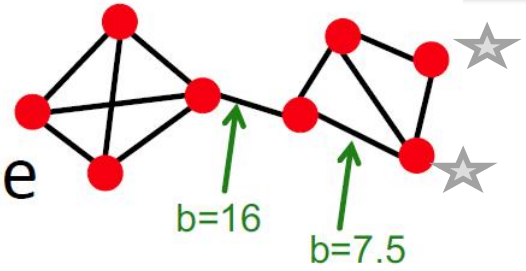


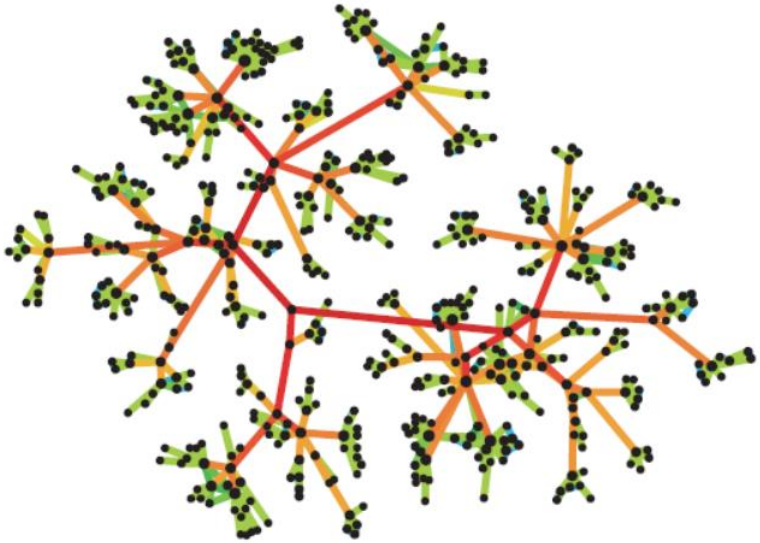
Path-based Measures of Centrality

- Edge Betweenness
- Node Betweenness

Network Centrality for Edge

- **Edge betweenness:** Number of shortest paths passing over the edge
- **Intuition:**





Edge betweenness in a real network

We assume the number of shortest paths between any pair of nodes is 1. If there are multiple shortest paths, count them fractionally.

How to Compute Edge Betweenness

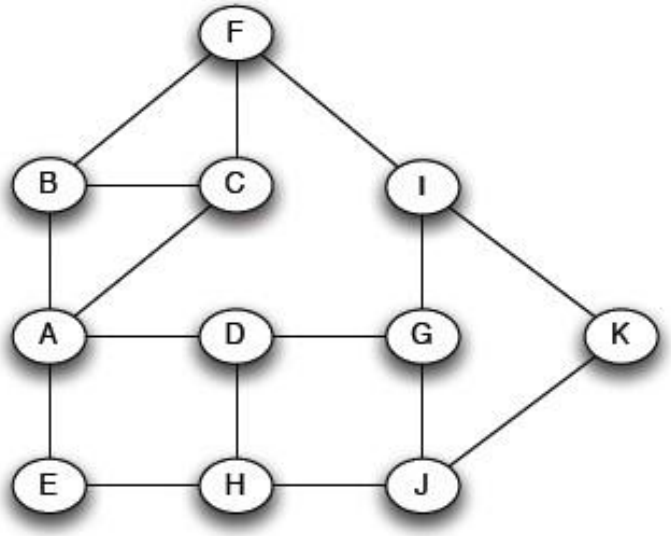
- **Divide-and-conquer idea**
- $\boldsymbol{S = S_A \cup S_B \ldots}$
- **Edge Betweenness**: Number of shortest paths passing over the edge
  - Subtask 1: compute the number of shortest paths starting from A and passing over the edge
  - Subtask 2: compute the number of shortest paths starting from B and passing over the edge

  ….

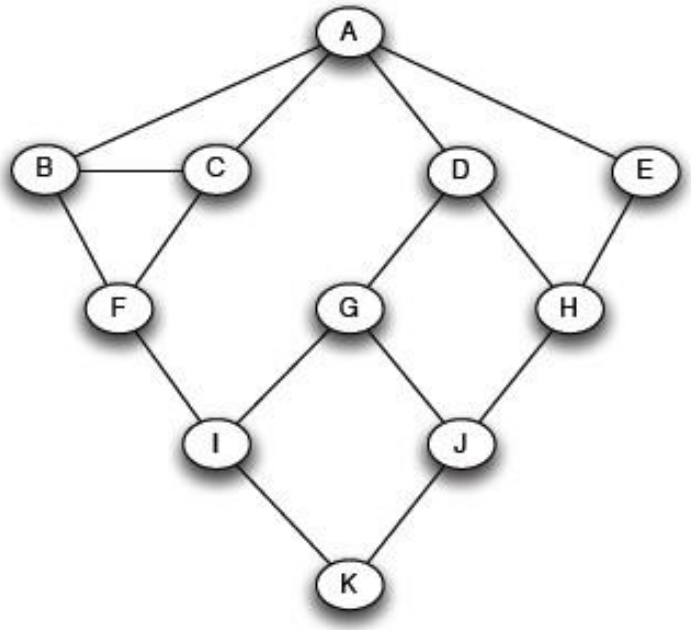
  Add the results of each sub-task.

How to Compute Edge Betweenness

- **Want to compute betweenness contributions of shortest paths starting from each node (e.g., A)**

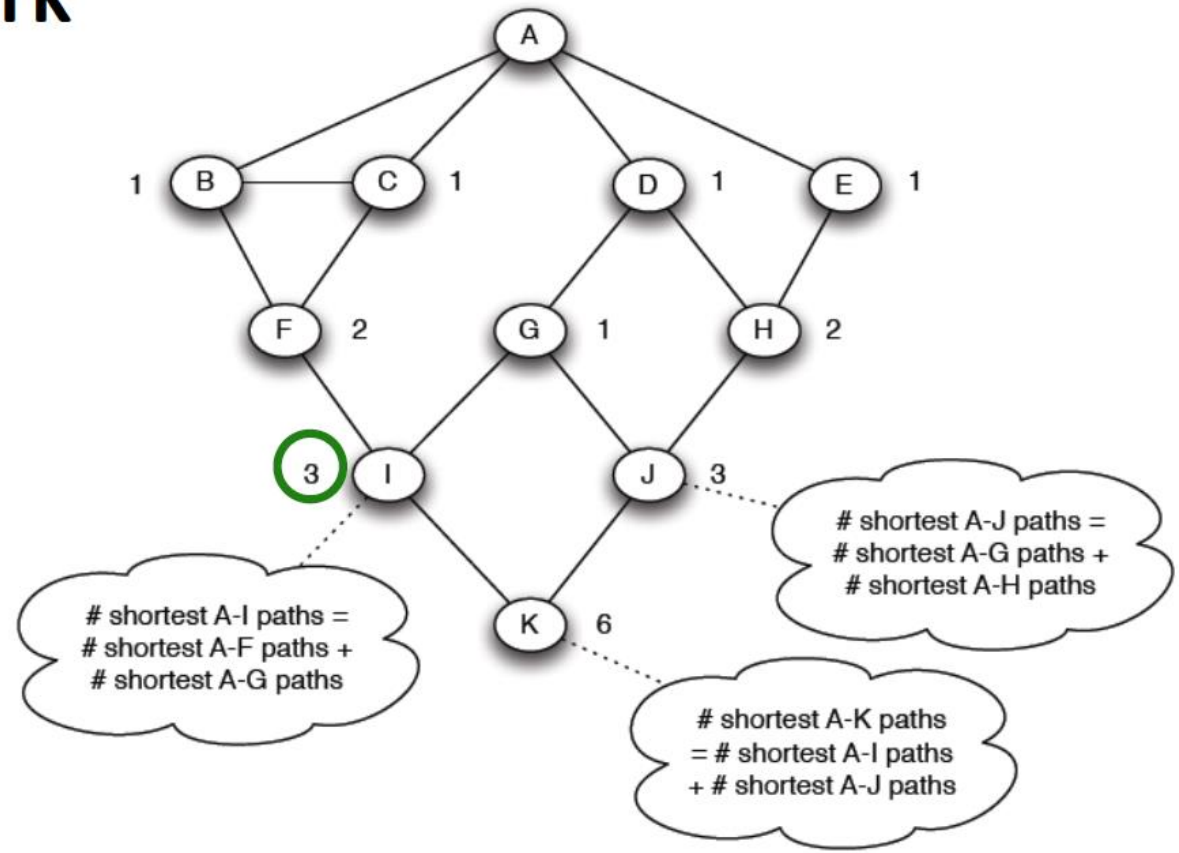


- **Breadth first search starting from A**



How to Compute Edge Betweenness

- **Forward step:** Count the number of shortest paths σ_{Ai} from A to all other nodes i of the network



How to Compute Edge Betweenness

- **Backward step: Compute betweenness:** If there are multiple paths count them fractionally

Normalization-Step

(1) We assume that the number of shortest path from node A to any node X is 1.

(2) Node flow (X)= the normalized number of shortest paths from node A to node X + the normalized number of shortest paths that pass through node X;

(3) Edge flow (e) = the normalized number of shortest paths that pass through the edge e.

-- node flow =
1+∑child edge flow

How to Compute Edge Betweenness

- **Backward step: Compute betweenness:** If there are multiple paths count them fractionally

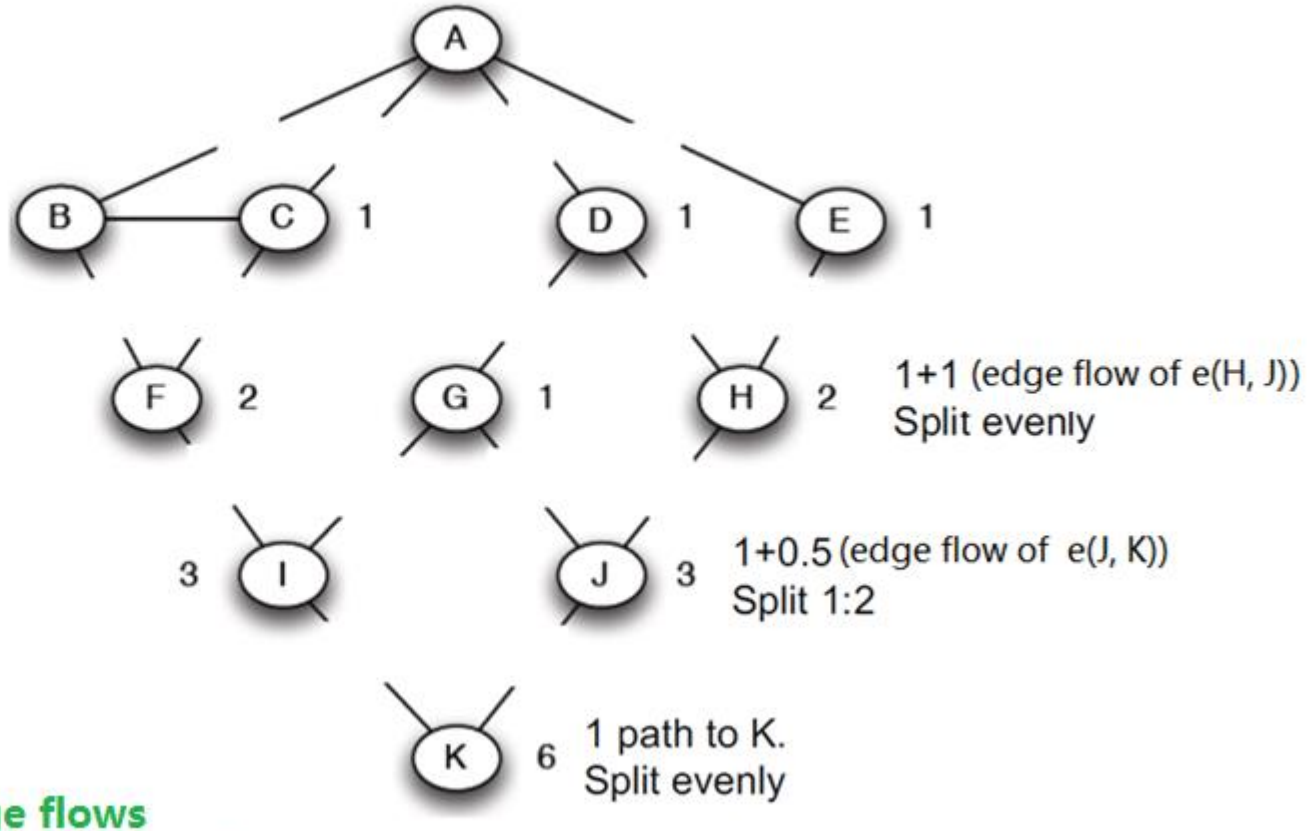
The algorithm:

- **Compute** edge **flows**:

-- node flow = 1+∑child edge flow

-- split the flow up based on the parent value

- Repeat the procedure for each starting node



For each edge, add all edge flows computed with different starting nodes

How to Compute Edge Betweenness

- **Backward step: Compute betweenness:** If there are multiple paths count them fractionally

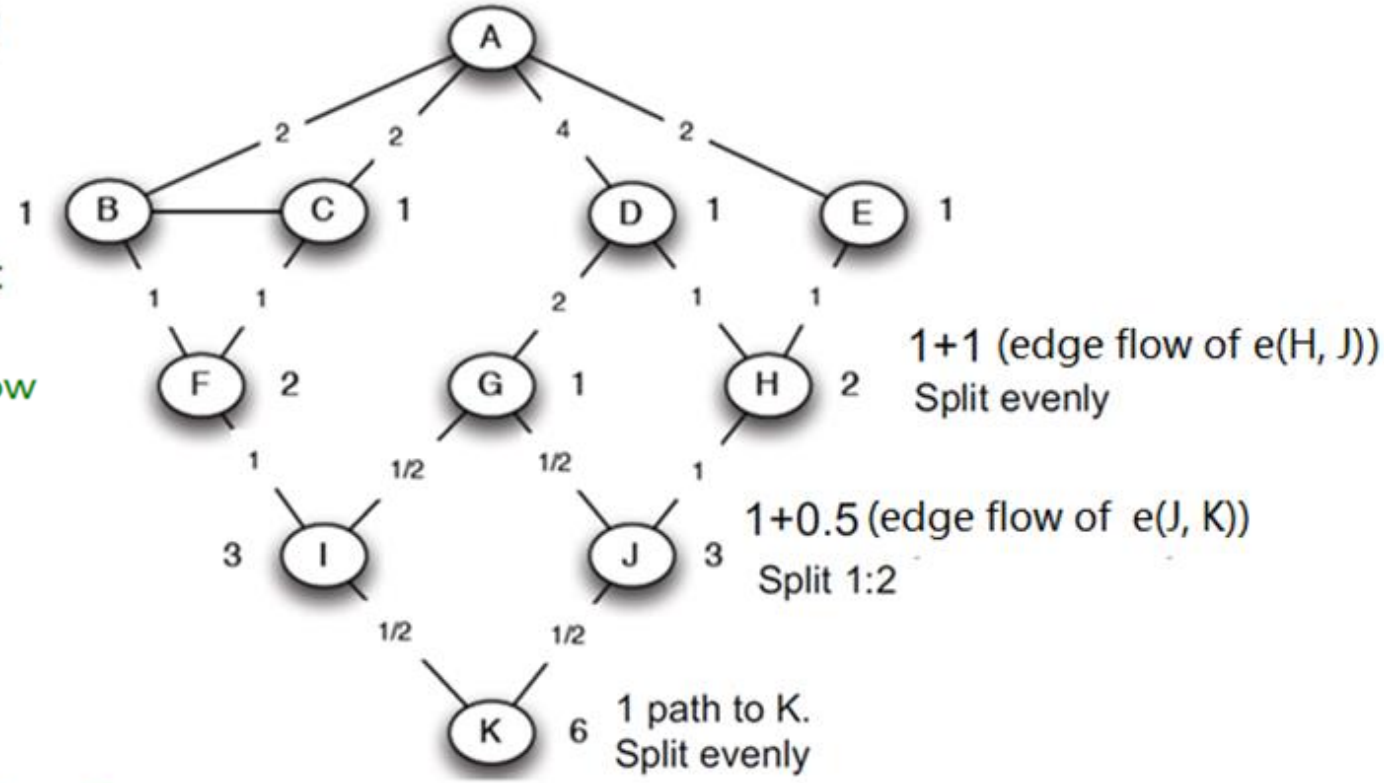
The algorithm:

- **Compute** edge **flows**:

-- node flow = 1+∑child edge flow

-- split the flow up based on the parent value

- Repeat the procedure for each starting node



For each edge, add all edge flows computed with different starting nodes

Node Betweenness Centrality

Another way of looking at centrality is by considering how important nodes are in connecting other nodes;

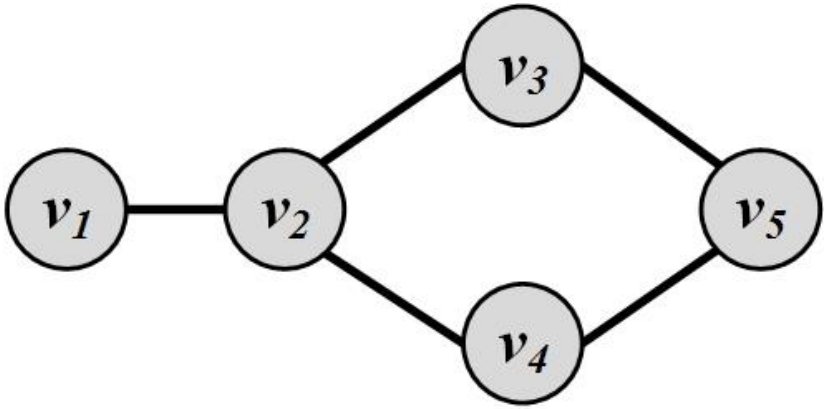
- **The betweenness centrality for node x is the probability that a shortest path passes through x**

$$C_b(v_i) = \sum_{s \neq t \neq v_i} \frac{\sigma_{st}(v_i)}{\sigma_{st}}$$

σ_{st} The number of shortest paths from vertex s to t – *a.k.a.* information pathways

$\sigma_{st}(v_i)$ The number of **shortest paths** from s to t that pass through v_i

Node Betweenness Centrality: Example 1



0/1 0/1

$$C_b(v_2) = 2 \times (\underbrace{(1/1)}_{s=v_1,t=v_3} + \underbrace{(1/1)}_{s=v_1,t=v_4} + \underbrace{(2/2)}_{s=v_1,t=v_5} + \underbrace{(1/2)}_{s=v_3,t=v_4} + \underbrace{0}_{s=v_3,t=v_5} + \underbrace{0}_{s=v_4,t=v_5})$$

$$= 2 \times 3.5 = 7,$$

$$C_b(v_3) = 2 \times (\underbrace{0}_{s=v_1,t=v_2} + \underbrace{0}_{s=v_1,t=v_4} + \underbrace{(1/2)}_{s=v_1,t=v_5} + \underbrace{0}_{s=v_2,t=v_4} + \underbrace{(1/2)}_{s=v_2,t=v_5} + \underbrace{0}_{s=v_4,t=v_5})$$

$$= 2 \times 1.0 = 2,$$

$$C_b(v_4) = C_b(v_3) = 2 \times 1.0 = 2,$$

$$C_b(v_5) = 2 \times (\underbrace{0}_{s=v_1,t=v_2} + \underbrace{0}_{s=v_1,t=v_3} + \underbrace{0}_{s=v_1,t=v_4} + \underbrace{0}_{s=v_2,t=v_3} + \underbrace{0}_{s=v_2,t=v_4} + \underbrace{(1/2)}_{s=v_3,t=v_4})$$

$$= 2 \times 0.5 = 1,$$

Computing Node Betweenness

- In betweenness centrality, we compute shortest paths between all pairs of nodes to compute the betweenness value.

- **Trivial Solution**:
  - Use Dijkstra and run it $O(n)$ times
  - We get an $O(n^3)$ solution

- Better Solution?

Brandes Algorithm [2001]

$$C_b(v_i) = \sum_{s \neq t \neq v_i} \frac{\sigma_{st}(v_i)}{\sigma_{st}} = \sum_{s \neq v_i} \delta_s(v_i)$$

$$\delta_s(v_i) = \sum_{t \neq v_i} \frac{\sigma_{st}(v_i)}{\sigma_{st}}$$

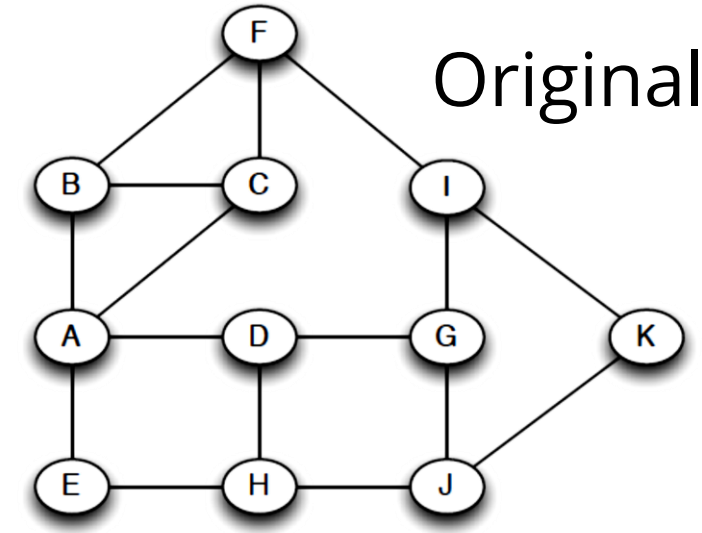
The dependence of s *to* v_i

There exists a recurrence equation that can help us determine $\delta_s(v_i)$

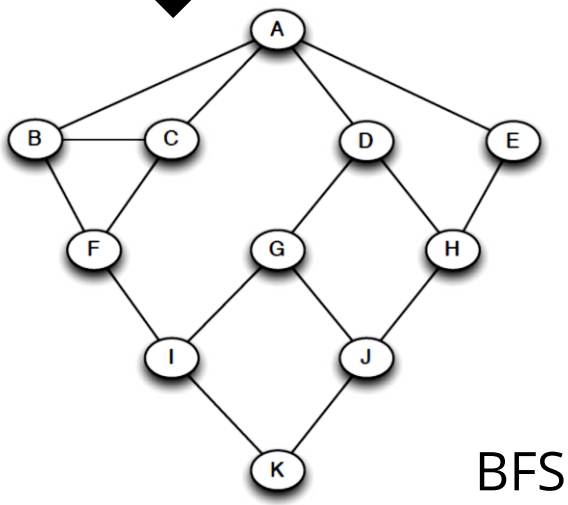
$$\delta_s(v_i) = \sum_{w \in \text{Children}(\ v_i\ ) \text{ w.r.t BFS (A)}} \frac{\sigma_{sv_i}}{\sigma_{sw}} (1 + \delta_s(w))$$

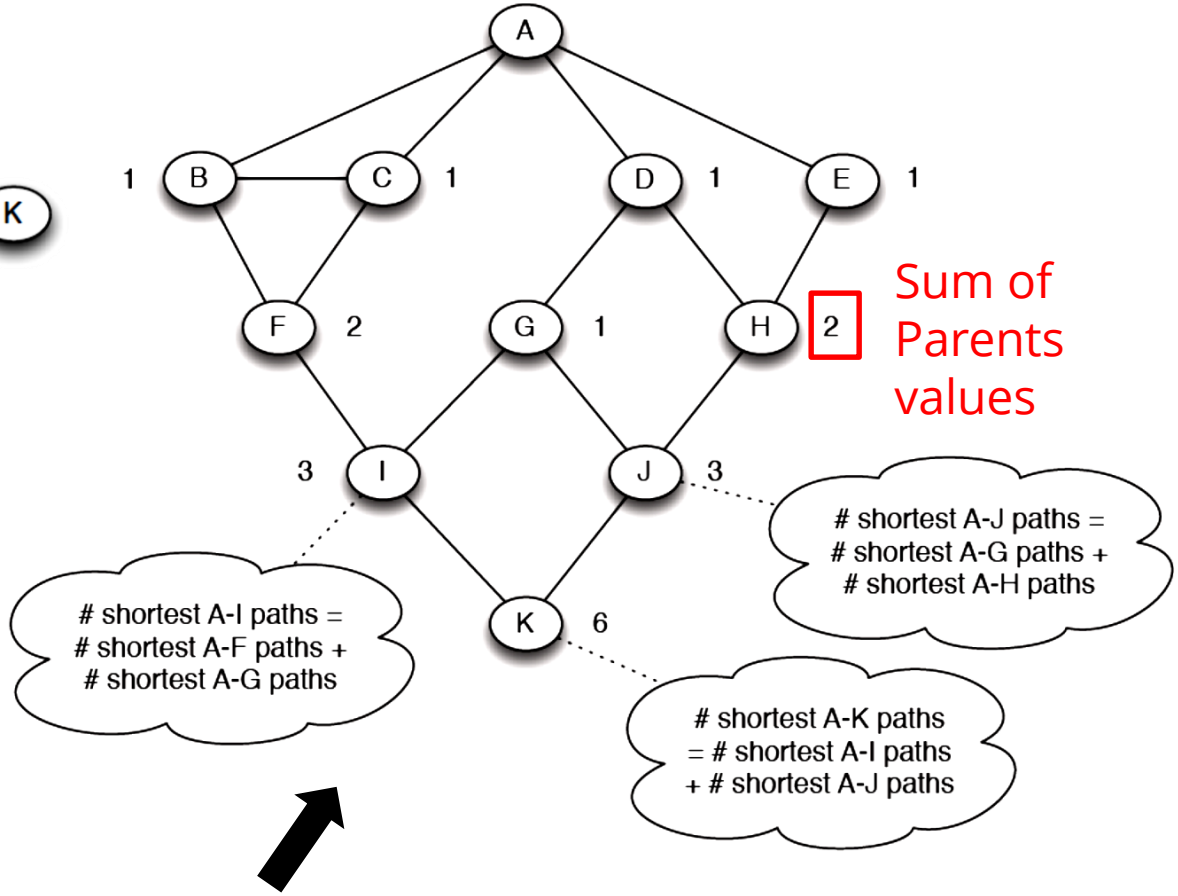
- *w is the child node of v_i w.r.t. BFS starting from s*
- If *w* is not v_i's child node, it can be ignored

How to compute σ_{st} (Forward Step)



Original Network

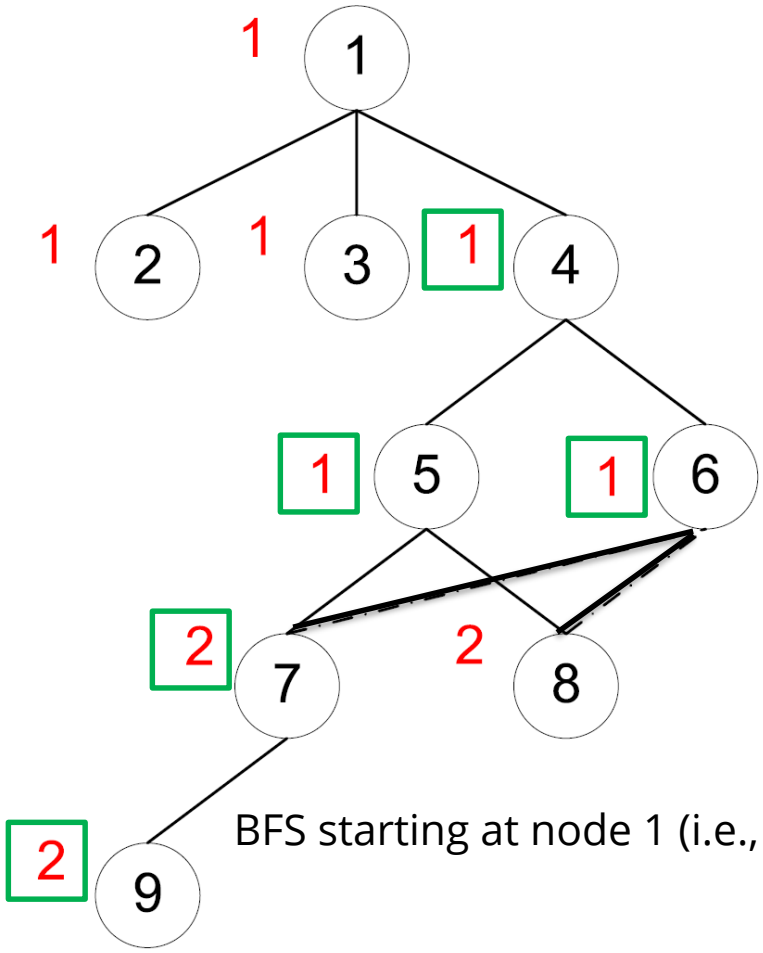




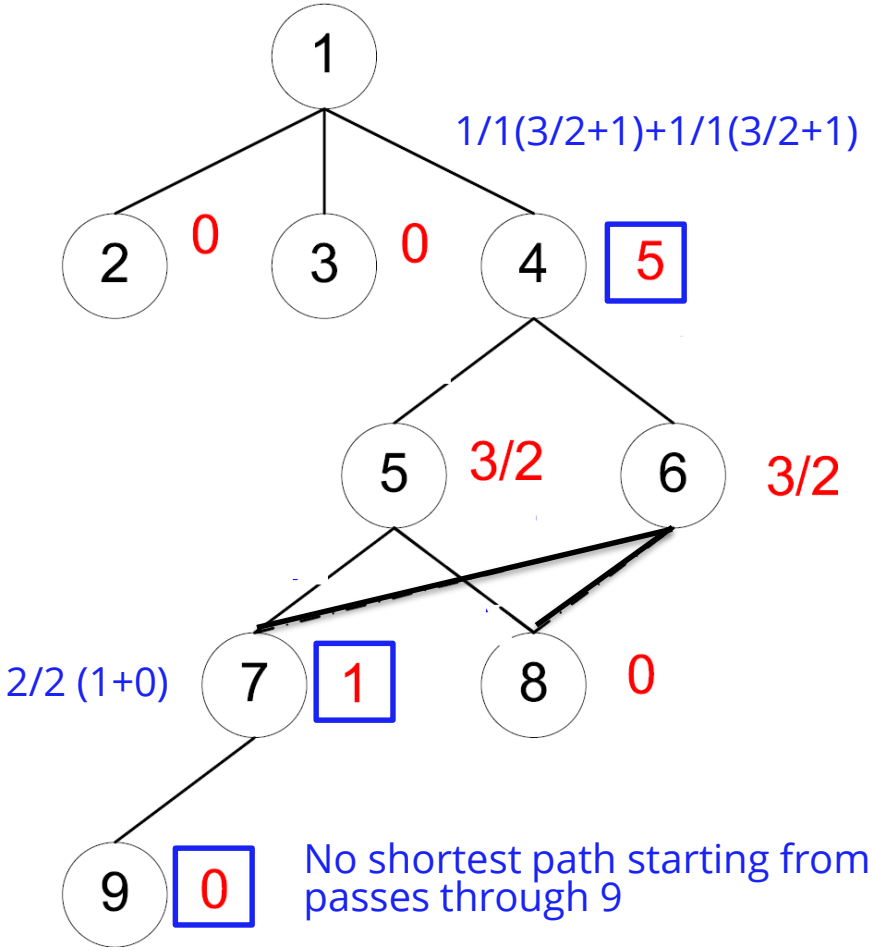
BFS starting at A (i.e., s)

Source: Networks, Crowds, and Markets: Reasoning about a Highly Connected World. By David Easley and Jon Kleinberg

How do you compute $\delta_s(v_i)$ (Backward Step)



BFS starting at node 1 (i.e., s)



No shortest path starting from 1 passes through 9

$$\delta_s(v_i) = \sum_{w \in \text{Children}(\ v_i\ ) \text{ w.r.t BFS (A)}} \frac{\sigma_{sv_i}}{\sigma_{sw}} (1 + \delta_s(w))$$

Brandes Algorithm [2001]

- So far, we have computed the dependence of the starting node s (i.e., 1) to each node v_i

$$\delta_s(v_i)$$

- Then, we need to repeat the above forward-backward steps for all the other nodes (i.e., treating each node as the starting node).

- Finally, for each node v_i, its betweenness value is

$$C_b(v_i) = \sum_{s \neq v_i} \delta_s(v_i)$$

Example of Betweenness Centrality

https://www.linkedin.com/pulse/wtf-do-you-actually-know-who-influencers-walter-pike

- The betweenness centrality of a node V is defined as the proportion of shortest paths between all pairs of nodes that go through V.
- Here: the red nodes have high betweenness centrality.

Brandes Algorithm [2001]

- Iterate over all vertices s in V
  - Calculate $\delta_s(v)$ for all $v \in V$ in two phases:
    1. Breadth-first search, calculating distances and shortest path counts from *s*, push all vertices onto stack as they're visited.
    2. Visit all vertices in reverse order (pop off stack), aggregating dependencies according to equation.

References

- R. Zafarani, M. A. Abbasi, and H. Liu, Social Media Mining: An Introduction, Cambridge University Press, 2014.
- http://socialmediamining.info/
- Stanford CS224W Analysis of Networks