Centrality measures in graphs or social networks

V.S. Subrahmanian
University of Maryland
vs@cs.umd.edu

PageRank

- Examples of vertices:
  - Web pages
  - Twitter ids
  - nodes on a computer network
- Examples of edges:
  - Web page *u* has a hyperlink to web page *v*
  - Twitterid *u* follows Twitterid *v*
  - Nodes *u* and *v* are connected [undirected]

PageRank: Basic philosophy

- The importance of a web is based on
  - Number of web pages hyper-linking to that page
  - Importance of those web pages.
- The importance of a Twitter user is based on:
  - Number of followers and
  - Importance of those followers.

PageRank: Math

- Pretty simple.
- $PR(v) = \frac{1-d}{N} + d * \sum_{u\ is\ a\ pred.of\ v} \frac{PR(u)}{out-degree(u)}$

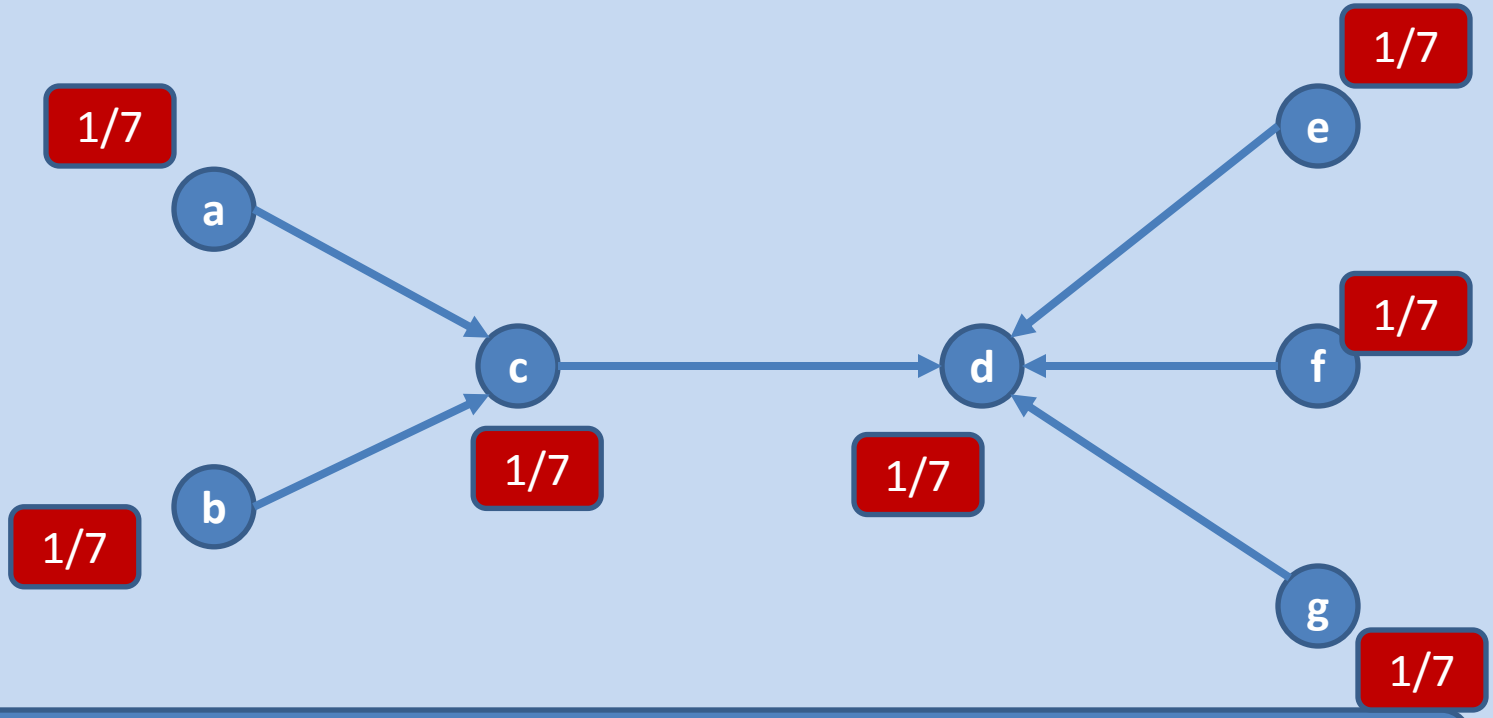
Where

- Out-Degree(v) = out degree of v.
- N = total number of nodes
- D = damping factor, the prob that a user will continue clicking, usually set to 0.85.
- Alternatively, 1-d is the probability that a visitor reaches a page directly without following hyperlinks.

Page Rank Algorithm

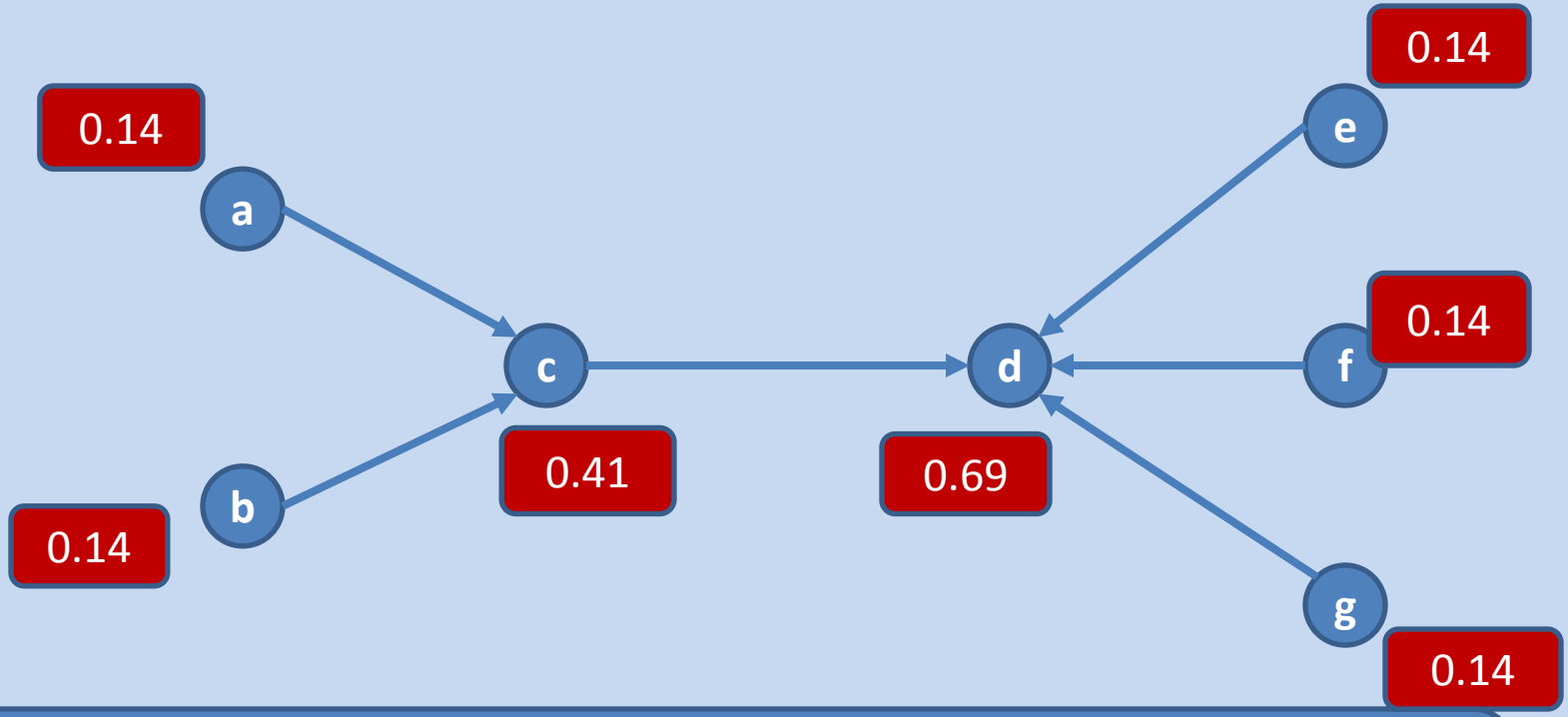
- Initially, set OLDPR(v) = 1/N for all vertices.
- Change = true;
- While *change* do
    - OLDPR(v) = NEWPR(v)
    - For each vertex *v*:
        - $NEWPR(v) = \frac{1-d}{N} + d * \sum_{u \text{ is a pred.of } v} \frac{OLDPR(u)}{out-degree(u)}$
    - Evaluate *change*
- Return NEWPR.
- Change is usually reset based on one of three factors:
    - The "while" loop has been executed K times for some fixed K determined by the user/app or
    - The difference of PR from the previous iteration to the current iteration is below some threshold for all nodes or
    - Very few nodes change

Example



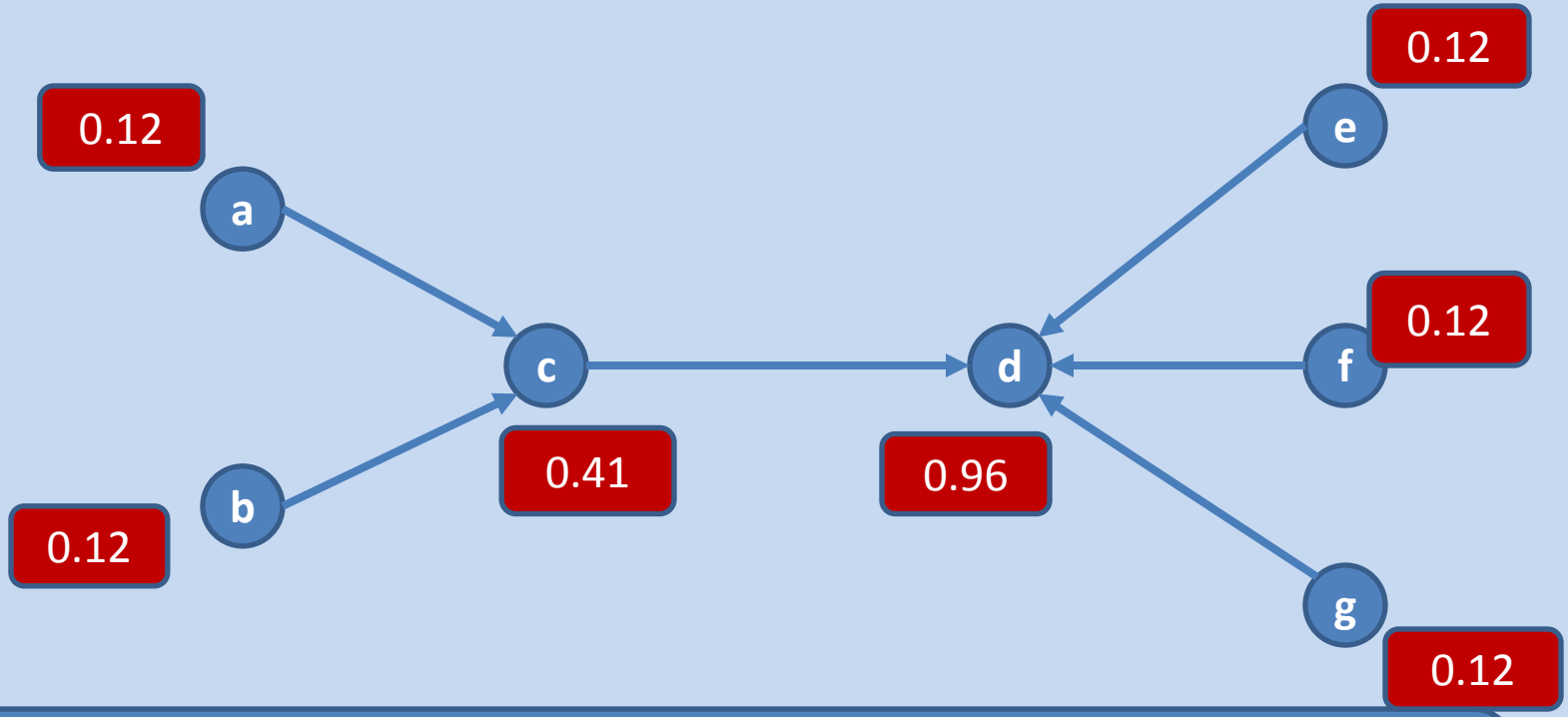
Initially, PR(v)=1/7 for all vertices. Let d=0 (for smplicity).

Example – Iteration 1



PR(a) = (1-1/7)/7 = (6/7)/7 = 6/49 = 0.12.
Same for PR(b), e, f, g.
PR(c) = (1-1/7)/7 + (1/7)/1 + (1/7)/1 = 6/49 + 1/7 + 1/7 = 0.41
PR(d) = (1-1/7)/7 + (1/7)/1 + (1/7)/1 + (1/7)/1 + (1/7)/1 = 6/49 + 4/7 = 0.69

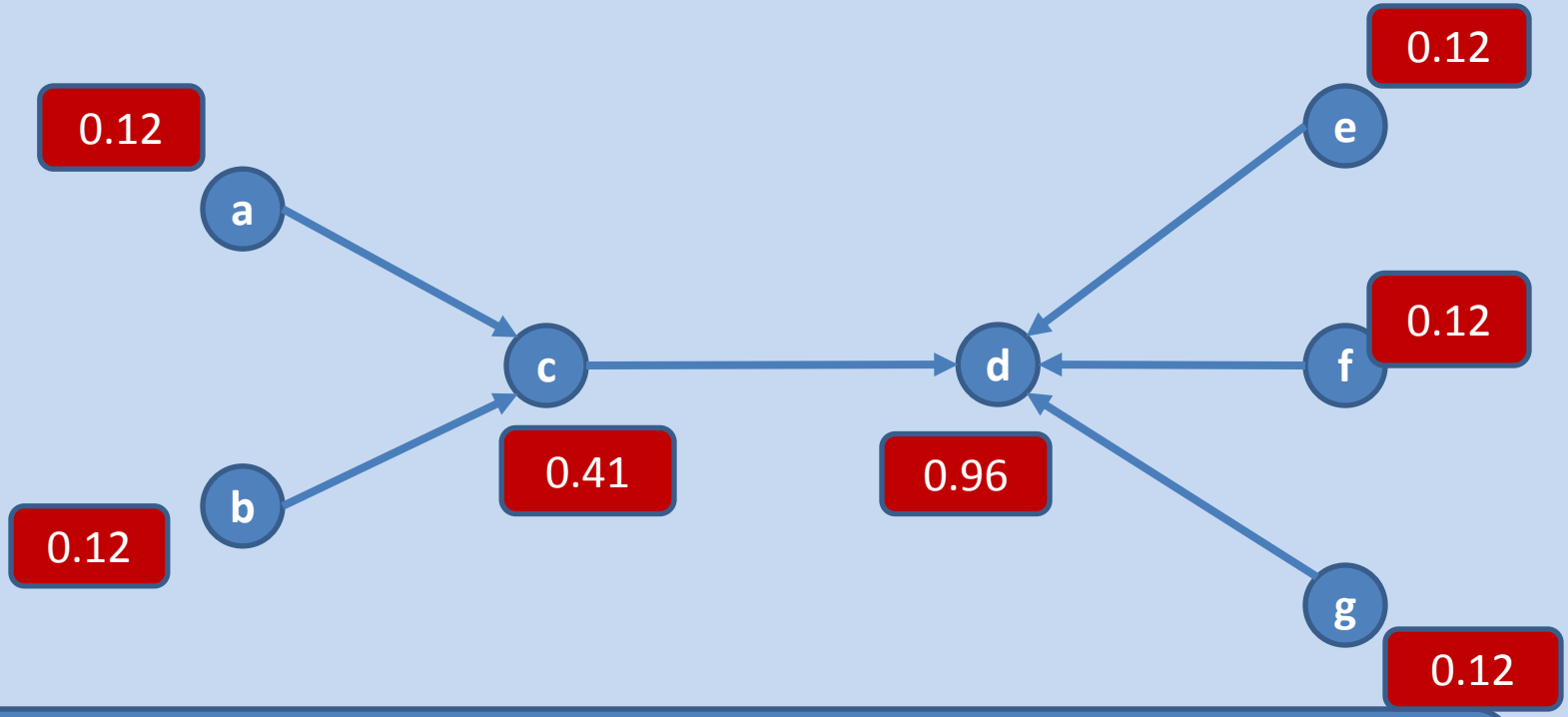
Example – Iteration 2



PR for a,b,e,f,g,c are unchanged. Why?

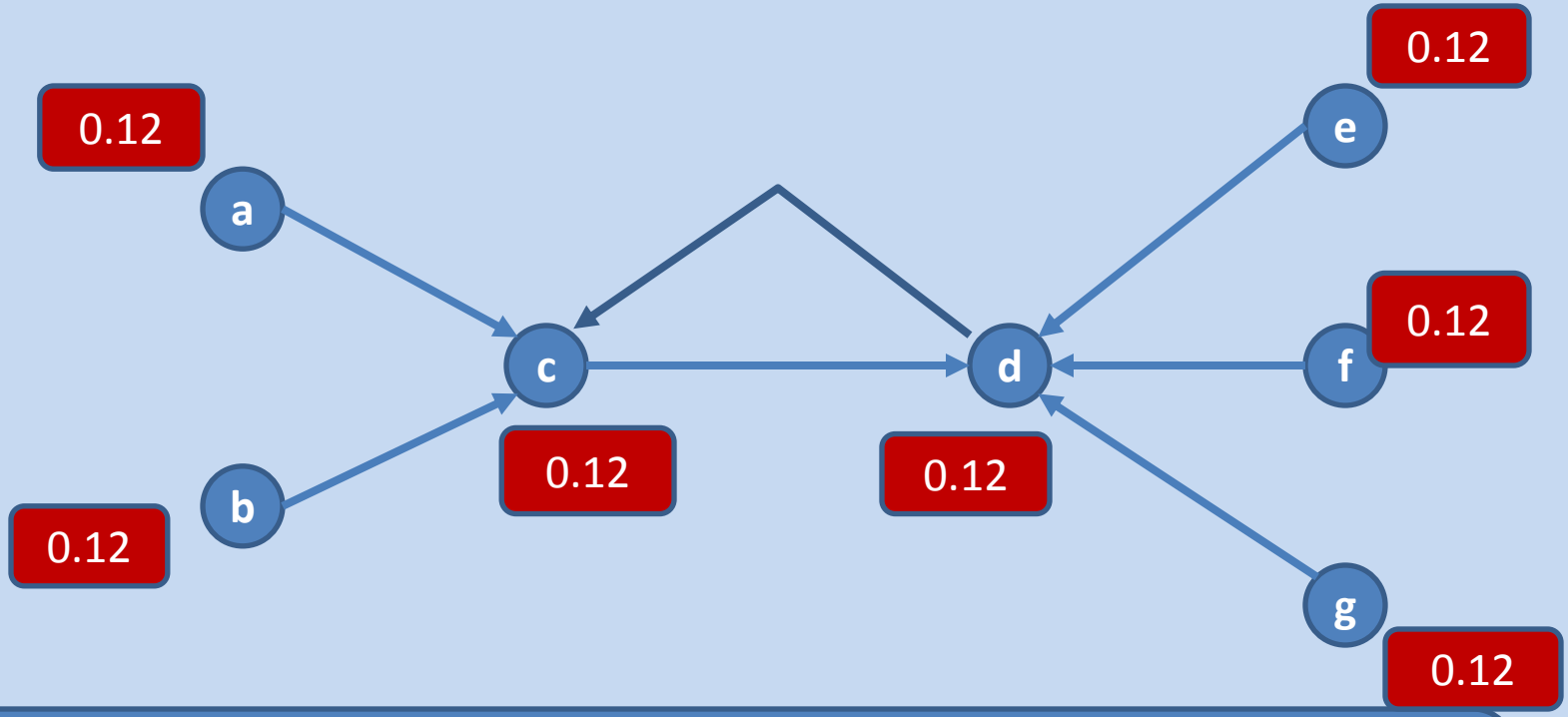
PR(d) = (1-1/7)/7 + (1/7)/1 + (1/7)/1 + (1/7)/1 + 0.41/1 = 6/49 + 3/7 + 0.41 = 0.96

Example – Iteration 2



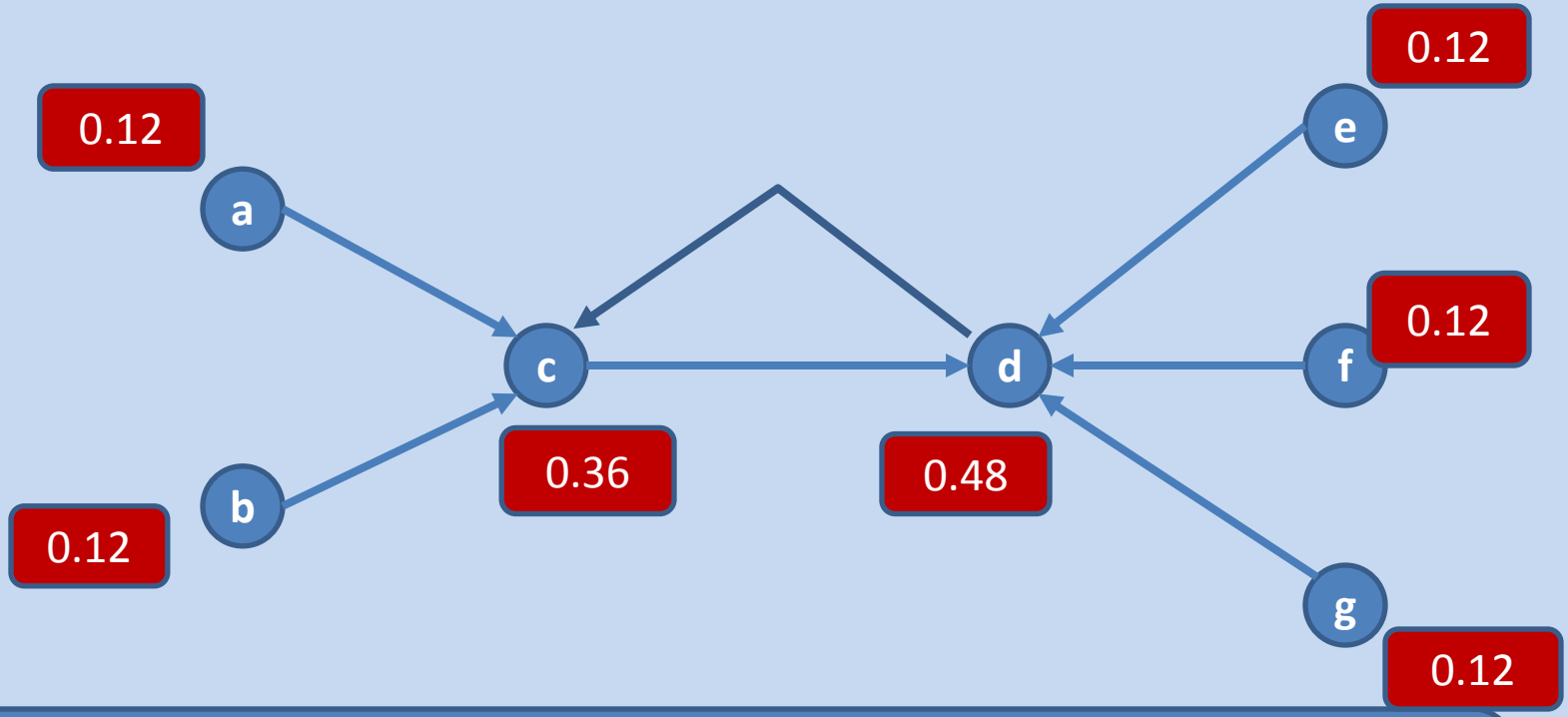
No change. We see that the most important nodes come out on top!

Example 2



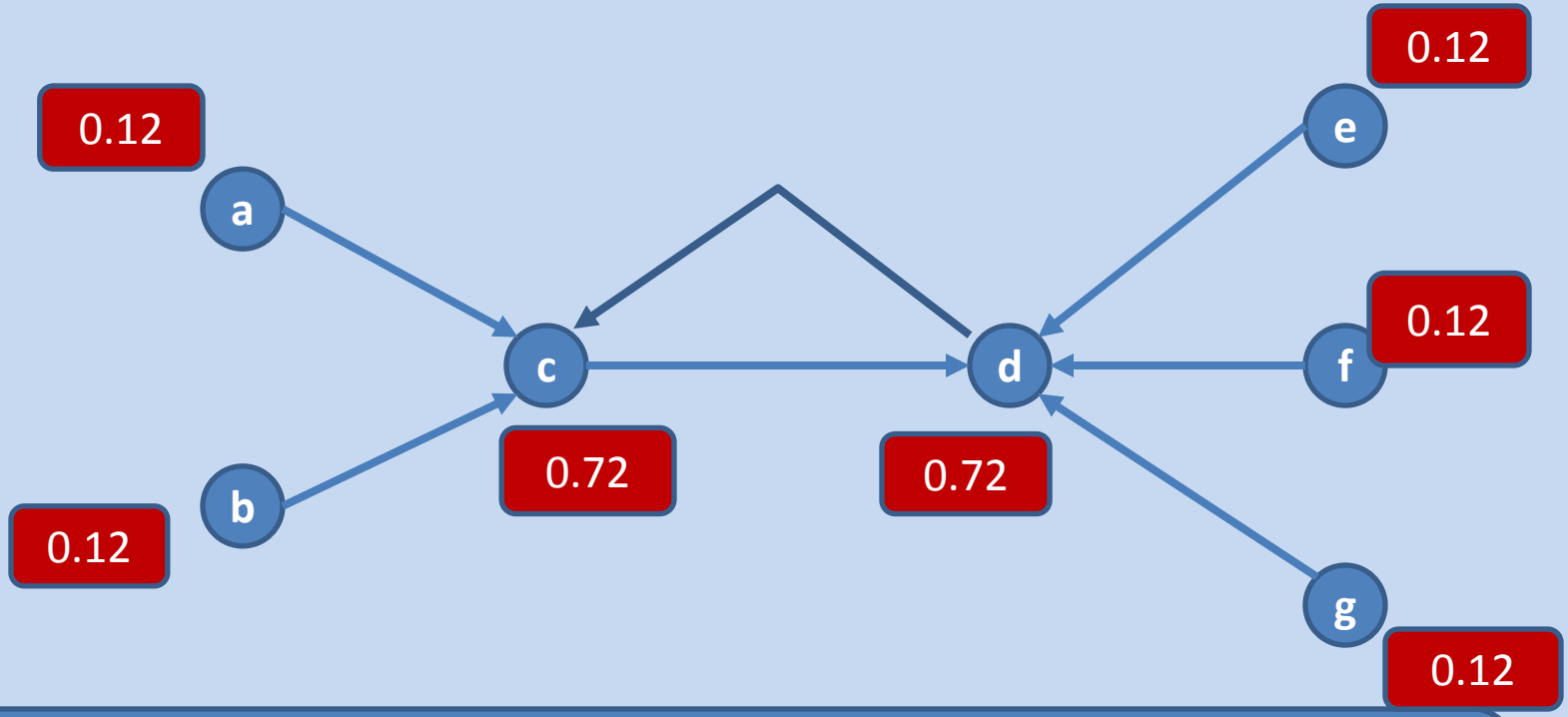
Let's change things slightly. This time, c and d follow each other (or reference each other).
Initially PR = 1/7 = 0.4 for everyone.

Example 2 – First Iteration



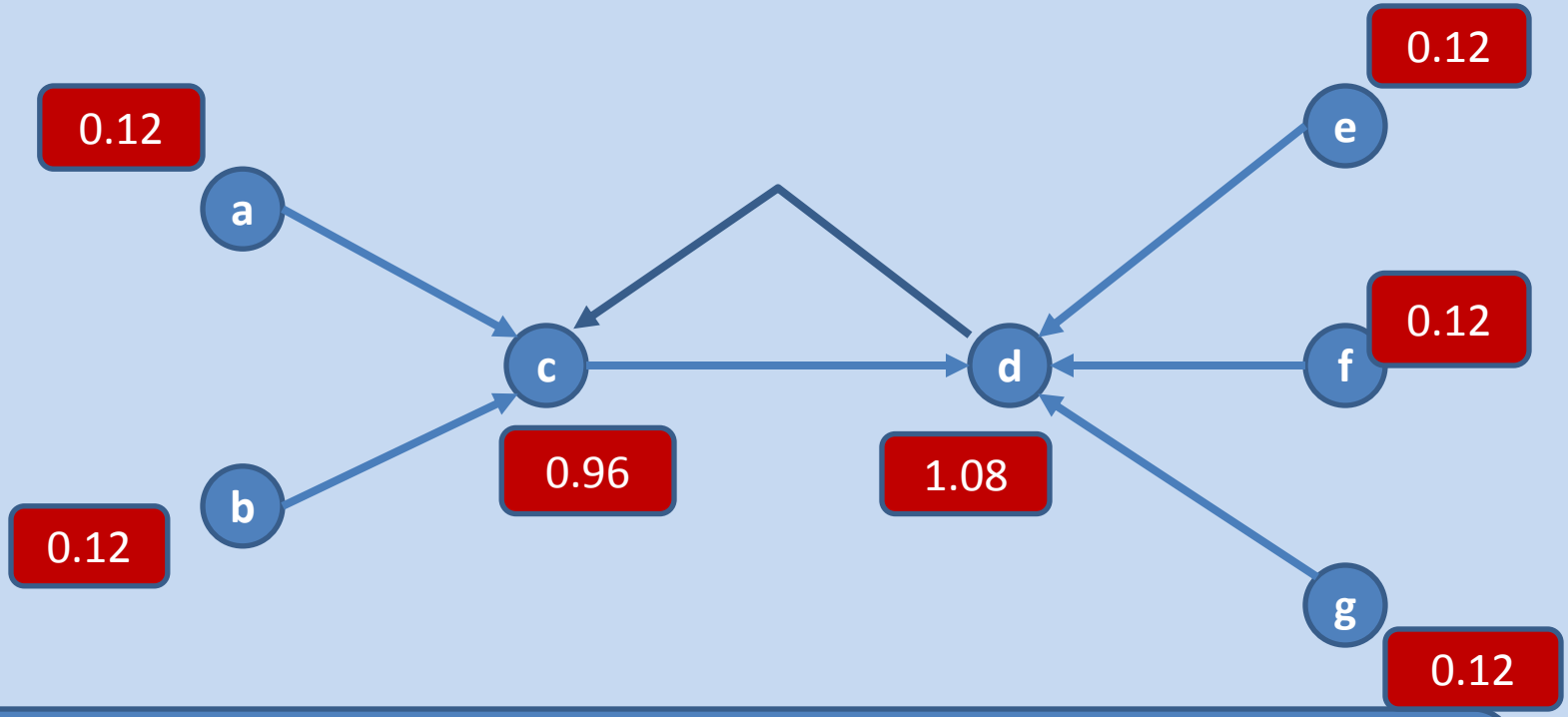
PR does not change for a, b, e, f, g.
PR(c) = 0.12/1 + 0.12/1 + 0.12/1 = 0.36
PR(d) = 0.12/1 + 0.12/1 + 0.12/1 + 0.12/1 = 0.48

Example 2 – Second Iteration



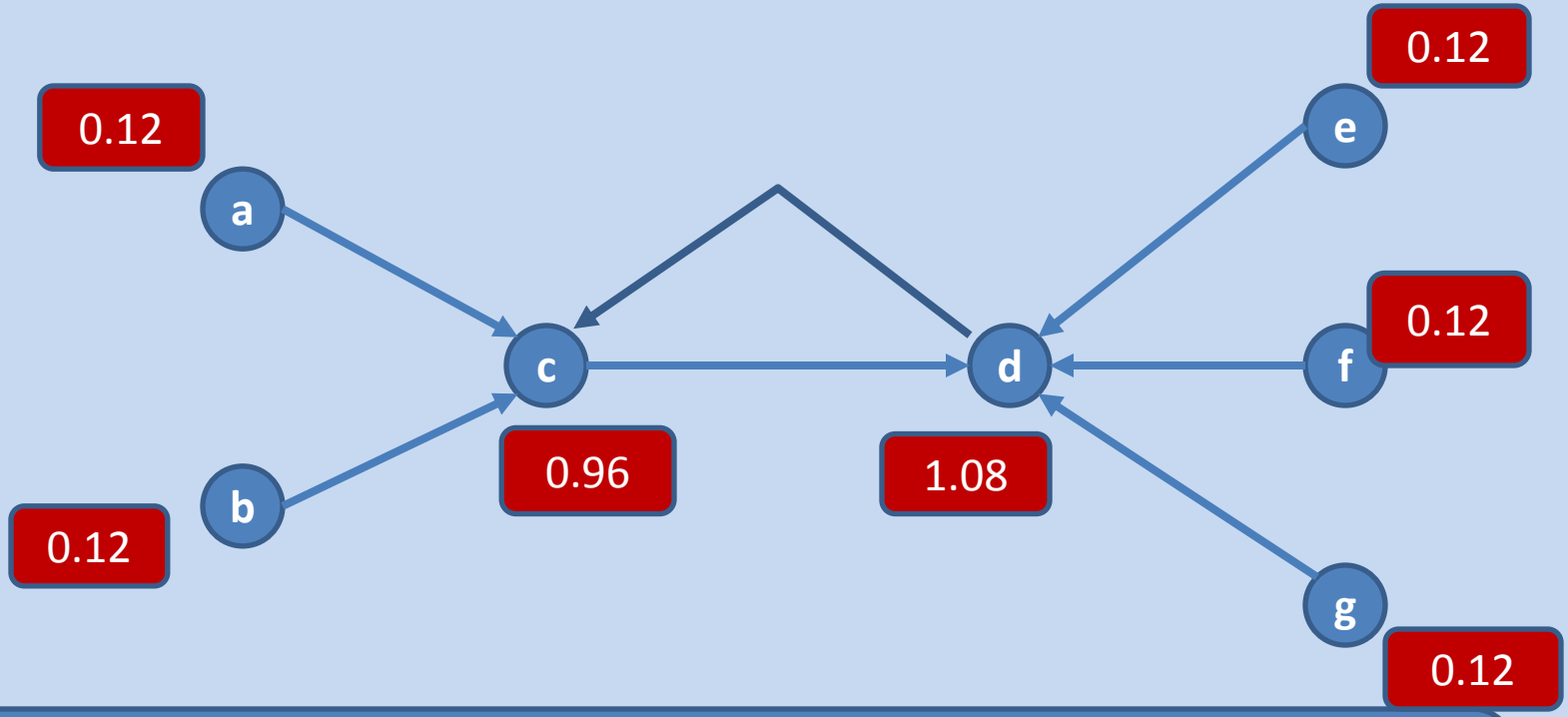
PR does not change for a, b, e, f, g.
PR(c) = 0.12/1 + 0.12/1 + 0.48/1 = 0.72
PR(d) = 0.12/1 + 0.12/1 + 0.12/1 + 0.36/1 = 0.72

Example 2 – Third Iteration



PR does not change for a, b, e, f, g.
PR(c) = 0.12/1 + 0.12/1 + 0.72/1 = 0.96
PR(d) = 0.12/1 + 0.12/1 + 0.12/1 + 0.72/1 = 1.08

Example 2 – Fourth Iteration



PR does not change for a, b, e, f, g.
PR(c) = 0.12/1 + 0.12/1 + 1.08/1 = 1.32
PR(d) = 0.12/1 + 0.12/1 + 0.12/1 + 0.96/1 = 1.32

Between-ness Centrality

- This part describes material in 2 papers:

  U. Brandes. On variants of shortest-path betweenness centrality and their generic computation, *Social Networks,* Vol. 30, pages 136-145, 2008

  U. Brandes. *A Faster Algorithm for Betweenness Centrality*, J. of Math. Sociology, Vol. 25, 2, pages 163-117, 2001.

Between-ness centrality

- Suppose *v* is a node.
- Let $\sigma(s,t)$ be the number of shortest paths between s and t.
- Let $\sigma(s,t|v)$ be the number of shortest paths between s and t that pass through *v*.
- Convention: $\sigma(s,s)=1$; $\sigma(s,t|v)=0$ if v = s or v=t. Also, 0/0 = 0.
- Between-ness centrality of *v* is given by:

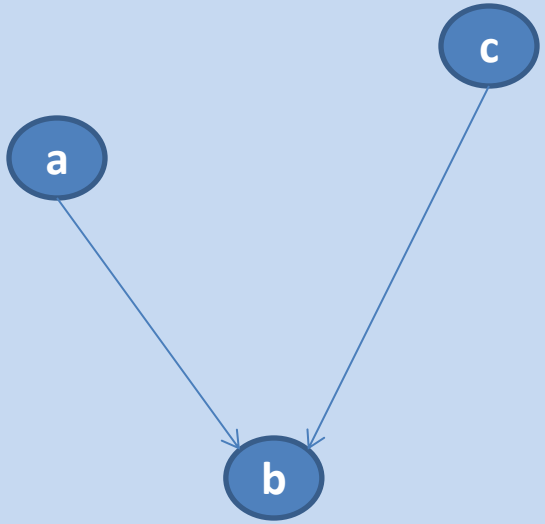
$$c_B(v) = \sum_{s,t \in V} \frac{\sigma(s,t|v)}{\sigma(s,t)}$$

- A small variant definition requires that s≠t in the above summation, i.e.

$$c_B(v) = \sum_{s \neq v \neq t \neq s,t \in V} \frac{\sigma(s,t|v)}{\sigma(s,t)}$$

Example BC

- Let's do a very simple calculation of BC for the vertices a in the graph on the right.



$$BC(a) = \frac{\sigma(a,a|a)}{\sigma(a,a)} + \frac{\sigma(a,b|a)}{\sigma(a,b)} + \frac{\sigma(a,c|a)}{\sigma(a,c)} + \frac{\sigma(b,a|a)}{\sigma(b,a)} + \frac{\sigma(b,b|a)}{\sigma(b,b))} + \frac{\sigma(b,c|a)}{\sigma(b,c)} + \frac{\sigma(c,a|a)}{\sigma(c,a)} + \frac{\sigma(c,b|a)}{\sigma(c,b)} + \frac{\sigma(c,c|a)}{\sigma(c,c)} = 0 + 1 + 1 + 0 + 0 + 0 + 0 + 0 = 2.$$

Some Math

- The *dependency* of s,t on *v* is:

$$\delta(s,t|v) = \frac{\sigma(s,t|v)}{\sigma(s,t)}$$

Specifies ratio of shortest paths between s,t that go through *v*.

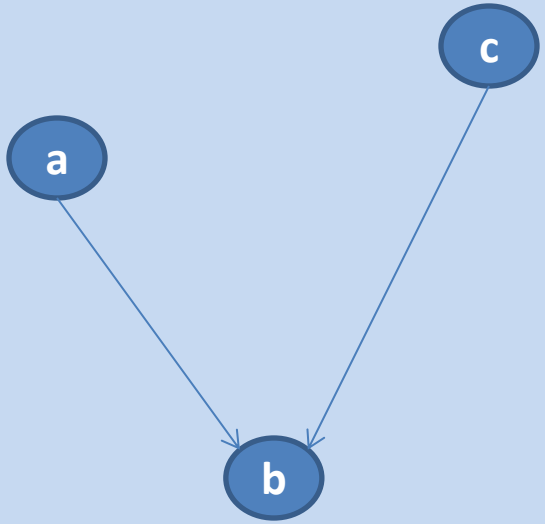
- The *dependency* of s on *v* is:

$$\delta(s|v) = \sum_{t \in V} \delta(s,t|v)$$

Dependency of s on *v* is just the sum of dependencies of s,t on v for all possible targets t.

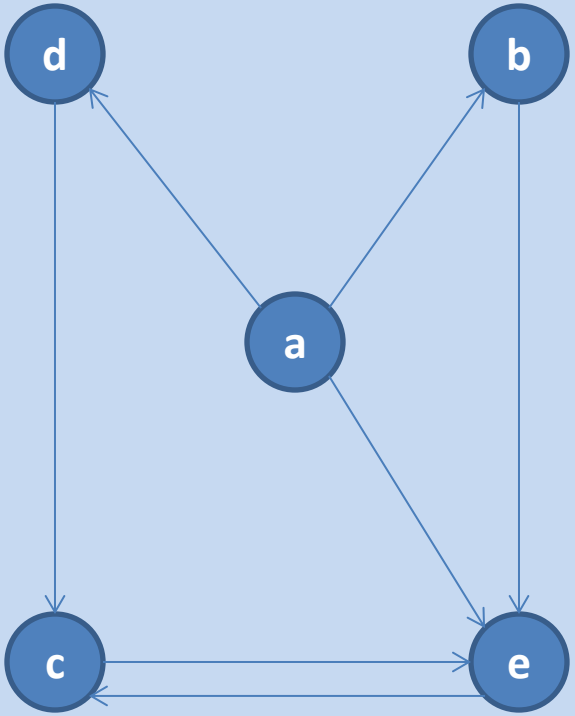
Example BC

- 6 pairs of nodes
  - a,b: SP = a,b
  - a,c: SP = a,b,c
  - b,c: SP = b,c
  - b,a: no SP
  - c,a: no SP
  - c,b: no SP
- $\delta(a, c|b) = 1.$
- $\delta(a \mid b) = \delta(a, a|b) = \delta(a, b|b) + \delta(a, c|b) = 0 + \frac{1}{1} + \frac{1}{1} = 2.$



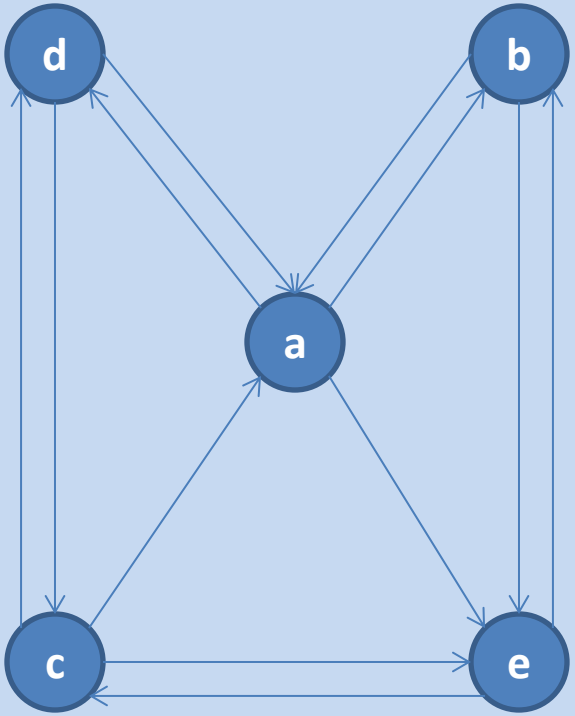
Example

- $\sigma(a,c) = 2$ because there are 2 shortests paths (a-d-c, a-e-c).
- $\sigma(a,c|e) = 1$ because one of these shortest paths goes through e.



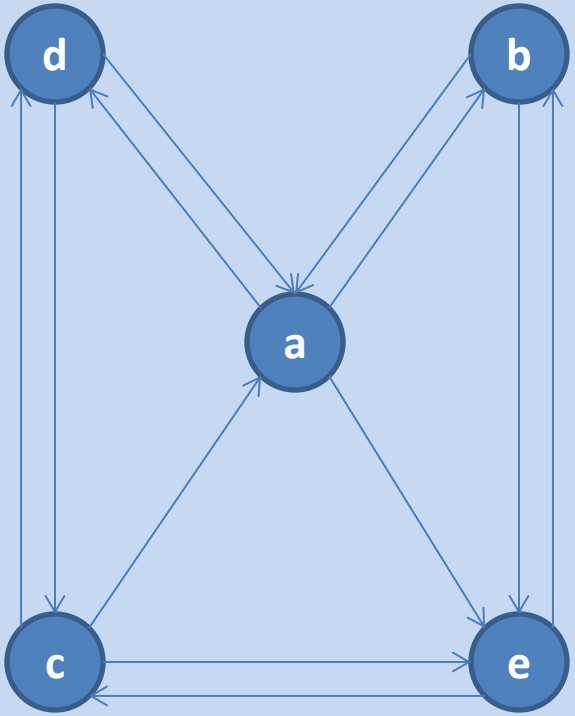
Example

- $\sigma(c,e) = 2$ because there are two shortest paths (c-a-b,c-e-b).
- $\sigma(c,e|a) = 1$ because one of these shortest paths goes through e.



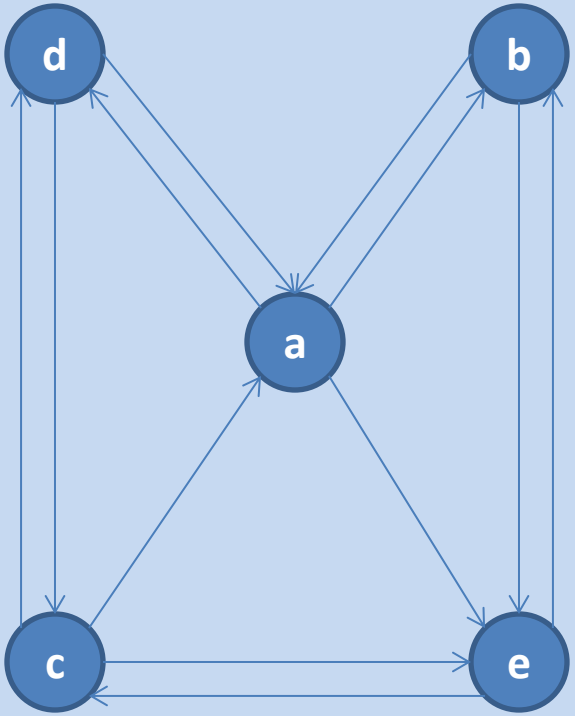
Example

- $\delta(c,a|a) = 1.$
- $\delta(c,b|a) = 0.5.$
- $\delta(c,c|a) = 0.$
- $\delta(c,d|a) = 0.$
- $\delta(c,d|a) = 0.$



Example

- $\delta(c|a)$ intuitively is the sum of dependencies a is involved in that originate at c.
- $\delta(c|a) = \delta(c,d|a) + \delta(c,b|a) + \delta(c,c|a) + \delta(c,d|a) + \delta(c,e|a) = 1.5$.
- $\delta(c|b)$ = DO CALCULATION.



Idea behind Algorithm

- Clearly, $C_B(v) = \sum_{s \in V} \delta(s|v)$ because $\delta(s|v)$ is the ratio in the summation defining between-ness centrality of *v.*
- So one approach is to compute $C_B(v)$ directly via the pseudo-code:

```
bc=0;
foreach s in V do bc = bc + δ(s|v);
Return bc
```

From Brandes 2001

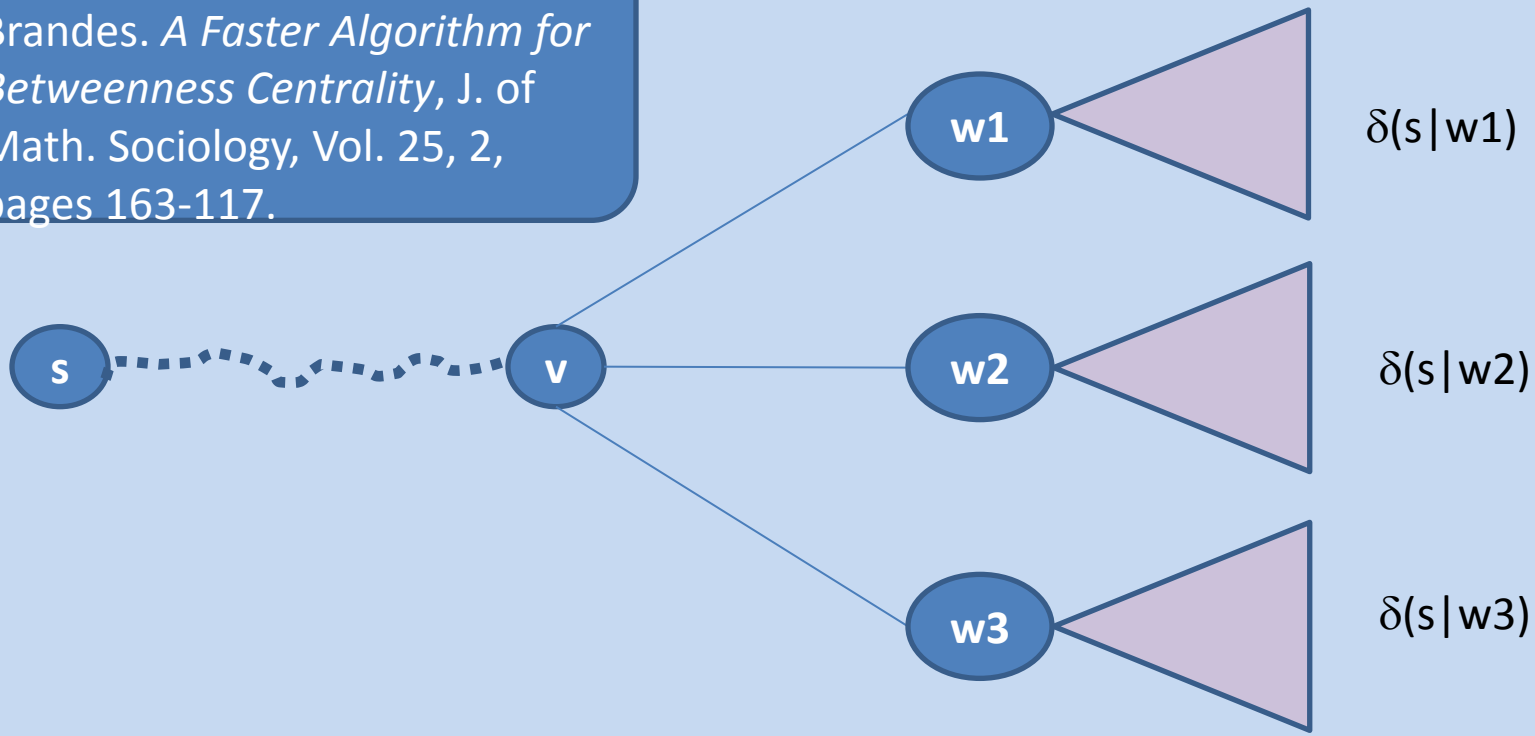
- $P_s(v)$ = { u in V | (u,v) in E & dist(s,v)=dist(s,u) + 1}.
- **Lemma.** $\sigma(s,v) = \sum_{u\ in\ P_s(v)} \sigma(s,u)$.
- Number of SPs between s and v = Sum of number of SPs between s and u where u is in $P_s(v)$.

From Brandes 2001

- **Lemma.** If there is exactly one SP from s to each vertex t in V, then $\delta(s|v) = \sum_{w:v\ in\ P_s(w)} (1 + \delta(s|w))$.
- Why? Because for the condition in the lemma to be true, the graph must be a tree.
- So v lies on either all or none paths between s and some vertex t in V.
- If v is a predecessor, then it lies on the SP for all such paths.

Picture (from Brandes 2001)

Figure is taken from: U. Brandes. *A Faster Algorithm for Betweenness Centrality*, J. of Math. Sociology, Vol. 25, 2, pages 163-117.



Clearly, v lies on the (unique) shortest path from s to all of v's successors. So $\delta(s,t|v)$ is either 0 or 1.

From Brandes 2001

- Brandes proves that:

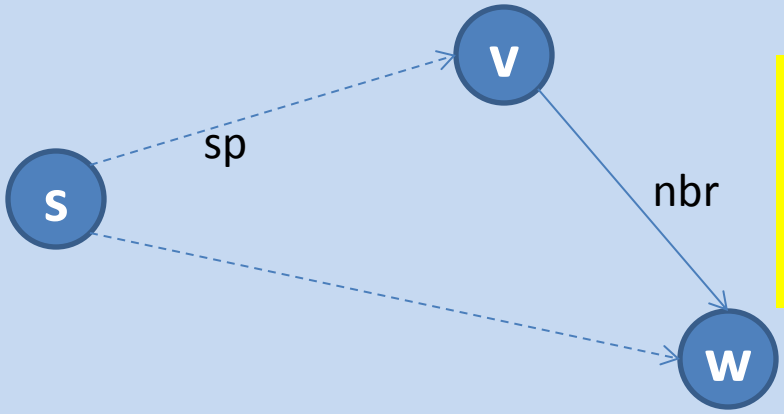
Theorem. $\delta(s|v) = \sum_{w:v\ in\ P_s(w)} \frac{\sigma(s,v)}{\sigma(s,w)} (1 + \delta(s|w))$.

Can do better

- Brandes shows that

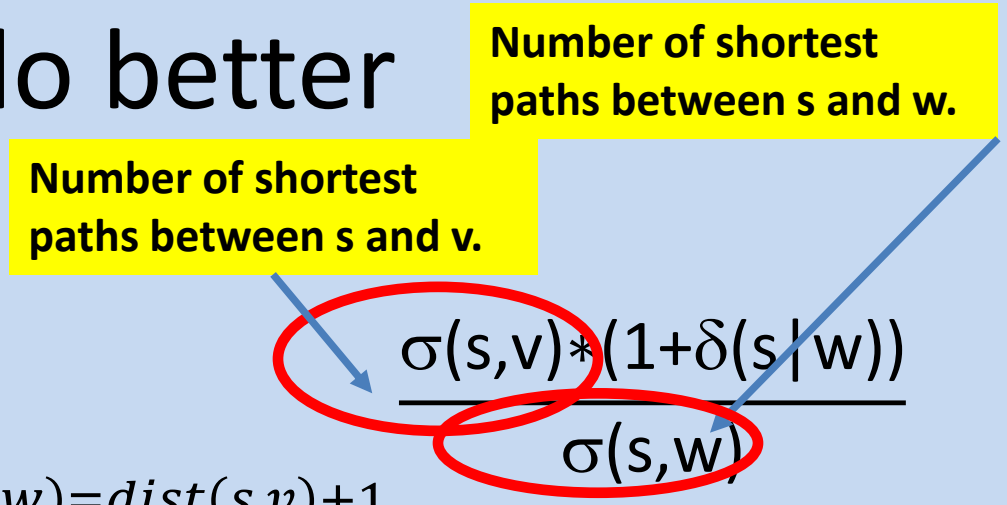
$$\delta(s|v) = \sum_{w:(v,w)\epsilon\, E\ \&\ dist(s,w)=dist(s,v)+1} \frac{\sigma(s,v)*(1+\delta(s|w))}{\sigma(s,w)}$$

- Here, *dist(s,w)* is the shortest path length from s to w.



So we are looking for w's s.t. a shortest path from s to w goes thru v just before reaching w.

Can do better



- Brandes shows that

$$\delta(s|v) = \sum_{w:(v,w)\epsilon\, E\ \&\ dist(s,w)=dist(s,v)+1}$$

- So the two σ terms give the ratio of SPs between s and w that go thru v.
- Dependency of s on w, $\delta(s|w)$, is the percentage of SPs from *s* to vertices *t* that go through w.
- The percentage of such shortest paths that go through v is obtained by multiplying $\delta(s|w)$ by the ratio of SPs between s and w that go thru v. of SPs between s and w that go thru v.

Brandes Algorithm Idea

- Idea 1: Perform a breadth first traversal of the graph. In each traversal, we compute the number of SPs going through a given node.
- Idea 2: $\delta(s,t|v)$ can be aggregated into $\delta(s|v)$'s, reducing some computation.
- **Time Complexity** O(m*n) where m is the number of edges, n is the number of vertices.
- **Space Complexity** O(m+n).

Brandes BC Algorithm

- Outer loop considers each vertex s in V
  - Initialization: Sets Pred(x) = NIL and dist(x) = ∞ for all x. Sets dist(s)=0 and $\sigma(s)=1$. $\sigma(v)$ denotes the number of shortest paths from source s to v.
  - Inner loop
    - Gets element v from queue
    - Finds all w s.t. (w,v) is an edge
    - Updates dist(w), pred(w) and $\sigma(w)$ for such w's.
  - Accumulation
    - Previous loops find SPs from selected s to other vertices

ALGORITHM 1
Betweenness Centrality in Unweighted Graphs

```
C_B[v] ← 0, v ∈ V;
for s ∈ V do
    S ← empty stack;
    P[w] ← empty list, w ∈ V;
    σ[t] ← 0, t ∈ V;   σ[s] ← 1;
    d[t] ← −1, t ∈ V;   d[s] ← 0;
    Q ← empty queue;
    enqueue s → Q;
    while Q not empty do
        dequeue v ← Q;
        push v → S;
        foreach neighbor w of v do
            // w found for the first time?
            if d[w] < 0 then
                enqueue w → Q;
                d[w] ← d[v] + 1;
            end
            // shortest path to w via v?
            if d[w] = d[v] + 1 then
                σ[w] ← σ[w] + σ[v];
                append v → P[w];
            end
        end
    end
    δ[v] ← 0, v ∈ V;
    // S returns vertices in order of non-increasing distance from s
    while S not empty do
        pop w ← S;
        for v ∈ P[w] do δ[v] ← δ[v] + (σ[v]/σ[w]) · (1 + δ[w]);
        if w ≠ s then C_B[w] ← C_B[w] + δ[w];
    end
end
```

Algorithm on left is taken from:
U. Brandes. *A Faster Algorithm for Betweenness Centrality*, J. of Math. Sociology, Vol. 25, 2, pages 163-117.

Brandes BC Algorithm

- Accumulation Step
  - Sets δ(v), the dependency of s on v to 0.
  - Iteratively pops elements from the stack.
  - Updates dependency of s on v
    - $\delta(v) = \delta(v) + (1+ \delta(w))*\sigma[v]/\sigma(w)$
    - For w ≠ s, sets $c_B(w) = c_B(w) + \delta(w)$.