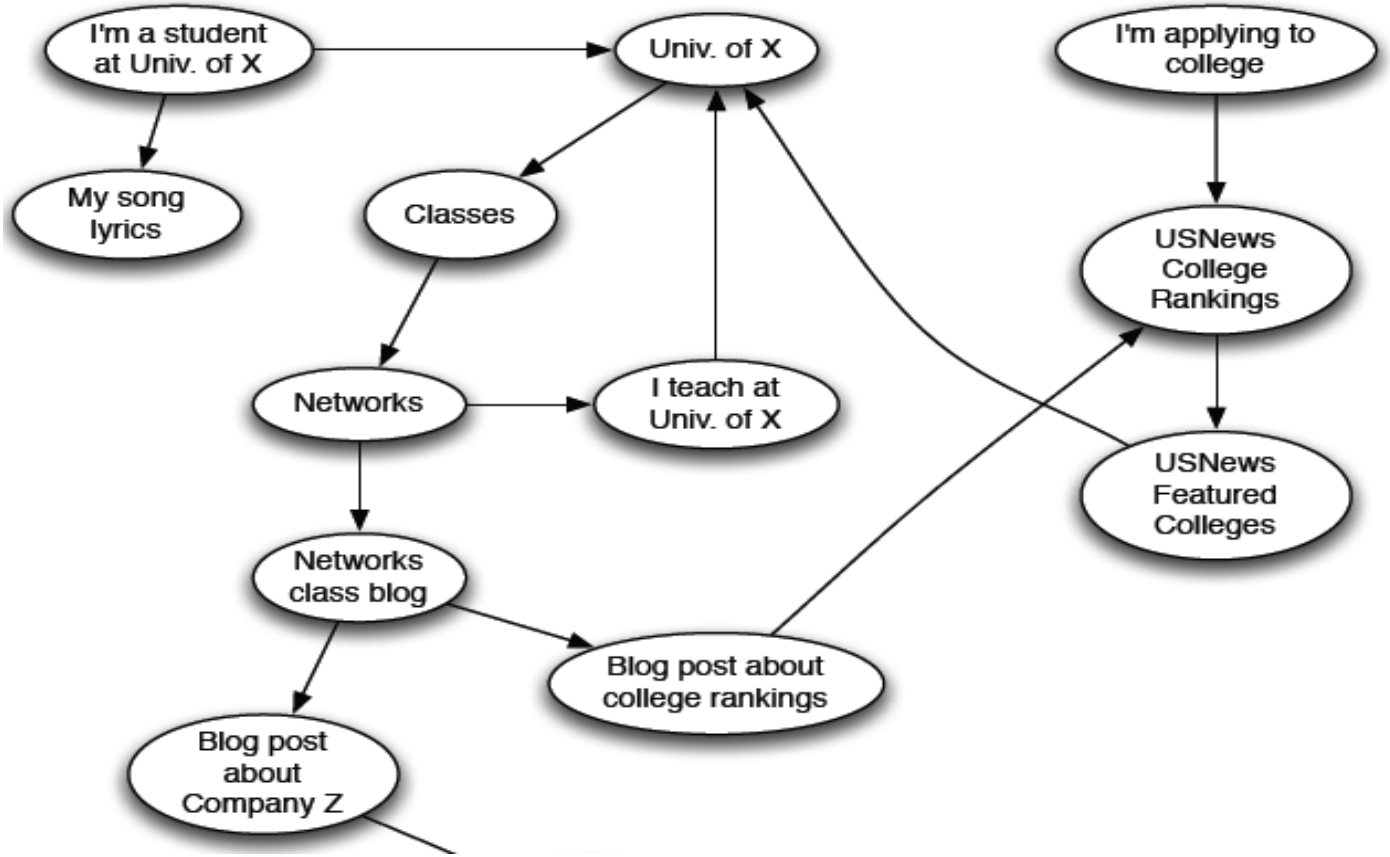
Introduction to Page Ranking

Ramesh Ragala
Assistant Professor (Senior)
VIT Chennai

Page Ranking

- Web as a Directed Graph:



Page Ranking

- Page Ranking → (This term came form Larry Page)google innovation
- Page Ranking was the essential technique for a search engine
- There are many search engines before google search engine.
  - They worked by crawling the WEB and listing the terms found in the page. → inverted Index is used.
  - presence of a term in a header of the page made the page more relevant than would the presence of the term in ordinary text
  - large numbers of occurrences of the term would add to the assumed relevance of the page.

Page Ranking

- Example: (unethical people to fool search engines)
  - If you were selling shirts on web, all you care about was that people would see your webpage, regardless of what they are looking for.
  - Trick: add term “movie” in your page more than 1000 times.
  - When the user issued a search query with a term “movie” , the search engine would list your page first.
  - Problem: prevention of occurring of term “movie”
    - Use background color to characters font color

Page Ranking

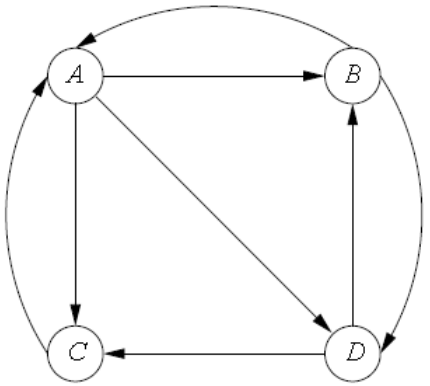
- Term Spam:
  - Techniques for fooling search engines into believing your page is about something it is not.
- The ability of term spammers to operate so easily rendered early search engines almost **useless**.
- Basic Idea: *Search engine believes what other pages say about you instead of what you say about yourself.*
- Google introduced two techniques to overcome that
  - 1. Page Rank
  - 2. The **content of a page** was judged **not** only by the **terms** appearing on that page, but by the **terms used in or near the links to that page**.

Page Ranking

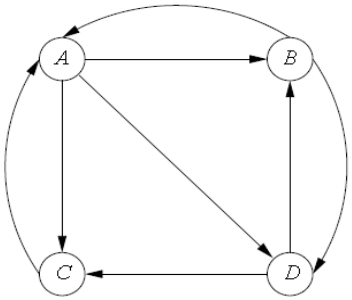
- PageRank was used to simulate where Web surfers, starting at a random page, would tend to congregate if they followed randomly chosen outlinks from the page at which they were currently located, and this process were allowed to iterate many times.
- **Pages that would have a large number of surfers were considered more "important" than pages that would rarely be visited.**
- Google prefers important pages to unimportant pages when deciding which pages to show first in response to a search query.

Page Ranking

- PageRank is a **function that assigns a real number** to **each page** in the Web.
- Higher the PageRank → more important
- There is not a single fixed algorithm for assignment of PageRank.
- Basic PageRank → Idealized PageRank.
- Suppose Random surfer starts at A.
- There are links to B,C and D.



Page Ranking



- Transition Matrix (M):
  - To describe what happens to random surfers after one step.
  - This matrix-M has n-rows and columns, if there are n-pages.
  - The element m_{ij} in row-i and column-j has value **1/k** if page-j has **k-arcs out**, and one of them is to page-i. Otherwise, $m_{ij} = 0$.

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

Page Ranking

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

- Transition Matrix:
  - The first column express that, a surfer has at A has 1/3 probability of next being at each other pages.
  - The second column expresses the fact that a surfer at B has a 1/2 probability of being next at A and the same of being at D.
  - The third column says a surfer at C is certain to be at A next.
  - The last column says a surfer at D has a 1/2 probability of being next at B and the same at C.

Page Ranking

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

- The **probability distribution** for the location of a **random surfer** can be described by a **column vector** whose j^{th} component is the probability that the surfer is at page-j.
  - This probability is the **idealized PageRank** function.

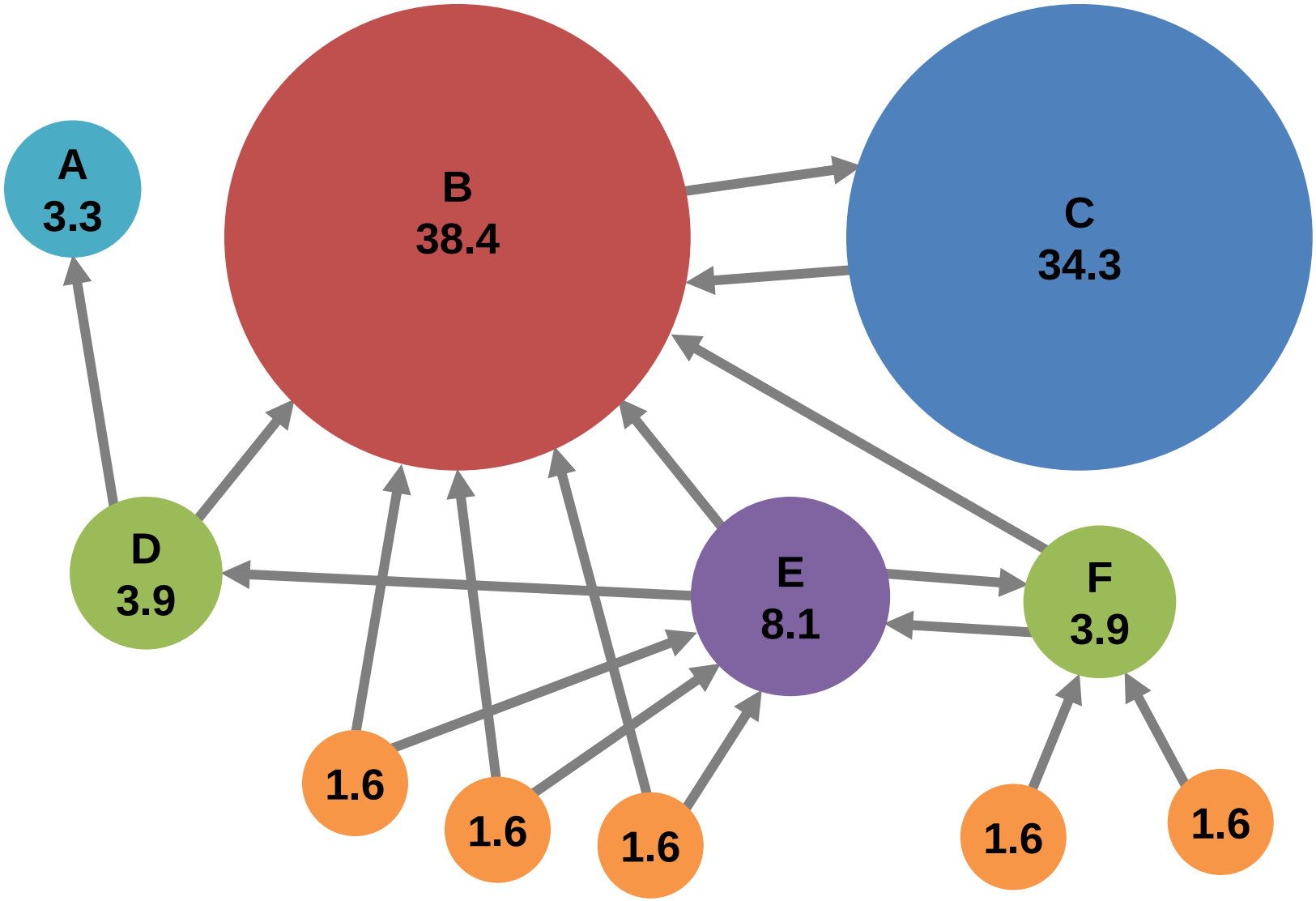
Page Ranking

- Suppose we start a **random surfer** at any of the n-pages of the Web with equal probability.
- Then the initial vector $\mathbf{v_0}$ will have **1/n** for each component.
- If M is the transition matrix of the Web, then after one step, the distribution of the surfer will be $\mathbf{Mv_0}$
- After two steps it will be $\mathbf{M(Mv_0)} = \mathbf{M^2v_0}$, and so on.
- In general, multiplying the initial vector $\mathbf{v_0}$ by **M** a total of i-times will give us the distribution of the surfer after i-steps

Page Ranking

- Reason behind multiplying a distribution vector-v by M gives the distribution **x = M v** at next step.
  - The probability x_i that a random surfer will be at node-i at the next step, is $\sum_j M_{ij} v_j$.
    - M_{ij} is the probability that a surfer at node-j will move to node-i at the next step.
    - V_j is the probability that the surfer was at node- j at the previous step.
  - This sort of behavior is an example of **Markov Processes**.

Page Ranking score



Page Ranking

- The distribution of the surfer approaches a limiting distribution v that satisfies v = Mv, provided two conditions are met:
  - The graph is strongly connected.
  - There are no dead ends.

- The limiting v is a Eigen vector of M.
  - An eigenvector of a matrix M is a vector v that satisfies $v = \lambda Mv$ for some constant eigenvalue λ.

Page Ranking

- Example: (Previous example)
  - There are four nodes, the initial vector v_0 has four components → each 1/4.
  - The sequence of approximations to the limit that we get by multiplying at each step by M is:

$$\begin{bmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{bmatrix}, \begin{bmatrix} 9/24 \\ 5/24 \\ 5/24 \\ 5/24 \end{bmatrix}, \begin{bmatrix} 15/48 \\ 11/48 \\ 11/48 \\ 11/48 \end{bmatrix}, \begin{bmatrix} 11/32 \\ 7/32 \\ 7/32 \\ 7/32 \end{bmatrix}, \cdots, \begin{bmatrix} 3/9 \\ 2/9 \\ 2/9 \\ 2/9 \end{bmatrix}$$

Page Ranking

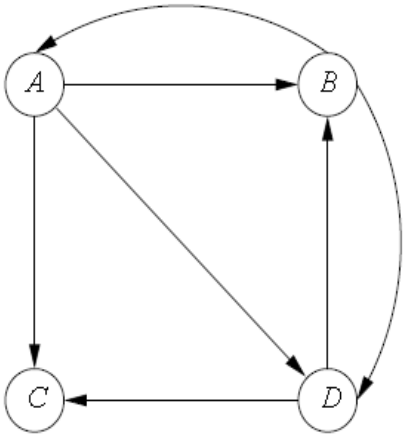
- Problems need to avoid:
  - Dead end → a page that has no links out
  - Spider Traps → groups of pages that all have outlinks but they never link to any other pages.
- These two problems can be solved using “Taxation”.
  - we assume a random surfer has a **finite probability** of leaving the Web at any step, and new surfers are started at each page.

Page Ranking

- Avoiding Dead ends:
  - If we allow dead ends, the transition matrix of the Web is no longer stochastic → sum of columns will be zero not one.
  - A matrix whose column sums are at most 1 is called **substochastic**.

Page Ranking

- Avoiding Dead ends:
- Example:



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

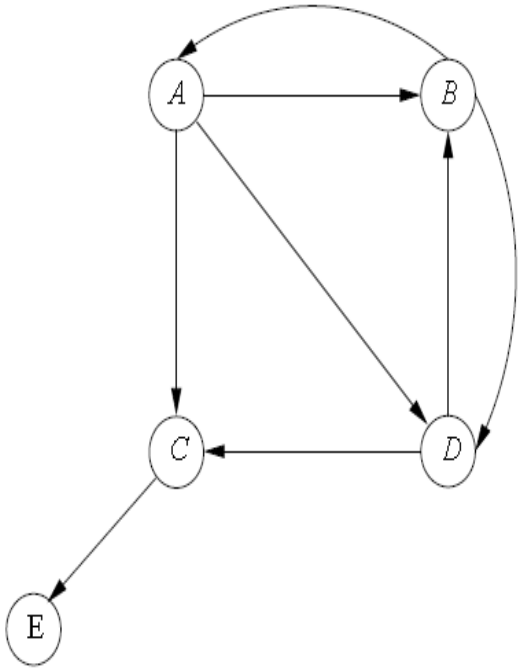
$$\begin{bmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{bmatrix}, \begin{bmatrix} 3/24 \\ 5/24 \\ 5/24 \\ 5/24 \end{bmatrix}, \begin{bmatrix} 5/48 \\ 7/48 \\ 7/48 \\ 7/48 \end{bmatrix}, \begin{bmatrix} 21/288 \\ 31/288 \\ 31/288 \\ 31/288 \end{bmatrix}, \ldots, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Page Ranking

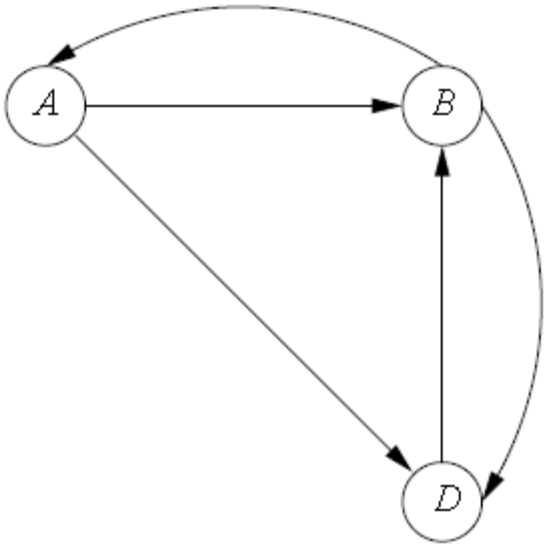
- Avoiding Dead ends:
- Two approaches:
- First approach:
  - Drop the dead ends from the graph, and also drop their incoming arcs.
  - Doing so may create more dead ends, which also have to be dropped, recursively.
  - Finally, we will end up with strongly connected component
- Second approach:
  - Taxation

Page Ranking

- Avoiding Dead ends:
- First Approach: example



Original Graph



After removing dead ends

Page Ranking

- Avoiding Dead ends:
- First Approach: example

$$M = \begin{bmatrix} 0 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 1/2 & 1/2 & 0 \end{bmatrix}$$

Matrix after removing dead ends

$$\begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}, \begin{bmatrix} 1/6 \\ 3/6 \\ 2/6 \end{bmatrix}, \begin{bmatrix} 3/12 \\ 5/12 \\ 4/12 \end{bmatrix}, \begin{bmatrix} 5/24 \\ 11/24 \\ 8/24 \end{bmatrix}, \ldots, \begin{bmatrix} 2/9 \\ 4/9 \\ 3/9 \end{bmatrix}$$

Sequence of Vectors

Page Ranking

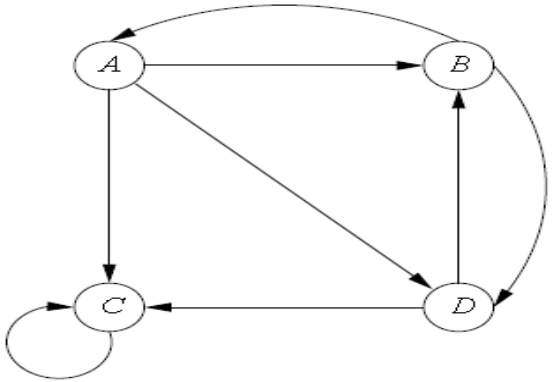
- Avoiding Dead ends:
- First Approach: example
  - In this example:
    - The PageRank of A is 2/9
    - The PageRank of B is 4/9
    - The PageRank of D is 3/9
  - What about PageRank of C and E????
    - Computation of PageRank of deleted nodes will done in the order opposite to that in which they were deleted.
    - C was the node deleted at end. → computation of C has to be done first.

Page Ranking

- Avoiding Dead ends:
- First Approach: example
    - C was the node deleted at end. → computation of C has to be done first.
    - The predecessors are A and D.
    - A has three successors, so it contributes 1/3 of its PageRank to C.
    - Page D has two successors, so it contributes half its PageRank to C.
    - PageRank of C = 1/3 * 2/9 + ½*3/9 = 13/54
    - PageRank of E = PageRank of C (since only one predecessors and that to from C only)

Page Ranking

- Avoiding Dead ends and Spider traps in graph:
- Second Approach: Taxation
  - a spider trap is a set of nodes with no dead ends but no arcs out.
  - These structures can appear intentionally or unintentionally on the Web.



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

A graph with one node spider trap and its transition matrix

Page Ranking

- Avoiding Dead ends and Spider traps in graph:
- Second Approach: Taxation

$$\begin{bmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{bmatrix}, \begin{bmatrix} 3/24 \\ 5/24 \\ 11/24 \\ 5/24 \end{bmatrix}, \begin{bmatrix} 5/48 \\ 7/48 \\ 29/48 \\ 7/48 \end{bmatrix}, \begin{bmatrix} 21/288 \\ 31/288 \\ 205/288 \\ 31/288 \end{bmatrix}, \ldots, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

Page Rank Computation

Page rank at C, since once there a random surfer can never leave.

Page Ranking

- Avoiding Dead ends and Spider traps in graph:
- Second Approach: Taxation
  - Modify the calculation of PageRank by allowing each random surfer a small probability of **teleporting** to a random page, rather than following an out-link from their current page.
  - The iterative step, where we compute a new vector estimate of PageRanks v' from the current PageRank estimate v and the transition matrix M is

$$V' = \beta M v + (1-\beta)e/n$$

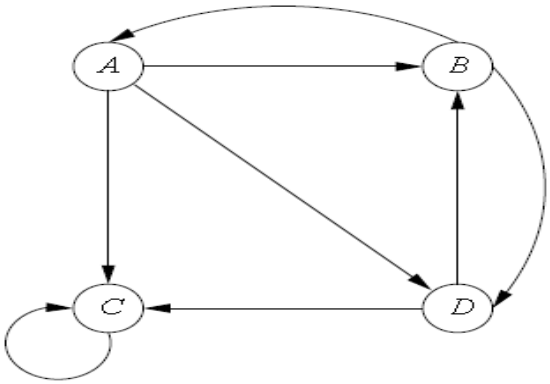
  - β is a chosen constant, range from 0.8 to 0.9
  - e is a vector of all 1's with appropriate number of components
  - n is the number of nodes in the web graph

Page Ranking

- Avoiding Dead ends and Spider traps in graph:
- Second Approach: Taxation
  - If the graph does not have dead ends
    - Surfer as deciding either to follow a link or teleport to a random page.
  - If the graph does have dead ends
    - There is a third possibility, which is that the surfer goes nowhere.

Page Ranking

- Avoiding Dead ends and Spider traps in graph:
- Second Approach: assume β = 0.8 in this example



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

$$\mathbf{v}' = \begin{bmatrix} 0 & 2/5 & 0 & 0 \\ 4/15 & 0 & 0 & 2/5 \\ 4/15 & 0 & 4/5 & 2/5 \\ 4/15 & 2/5 & 0 & 0 \end{bmatrix} \mathbf{v} + \begin{bmatrix} 1/20 \\ 1/20 \\ 1/20 \\ 1/20 \end{bmatrix}$$

Page Ranking

- Avoiding Dead ends and Spider traps in graph:
- Second Approach: assume β = 0.8 in this example

$$\begin{bmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{bmatrix}, \begin{bmatrix} 9/60 \\ 13/60 \\ 25/60 \\ 13/60 \end{bmatrix}, \begin{bmatrix} 41/300 \\ 53/300 \\ 153/300 \\ 53/300 \end{bmatrix}, \begin{bmatrix} 543/4500 \\ 707/4500 \\ 2543/4500 \\ 707/4500 \end{bmatrix}, \cdots, \begin{bmatrix} 15/148 \\ 19/148 \\ 95/148 \\ 19/148 \end{bmatrix}$$

Few Iterations

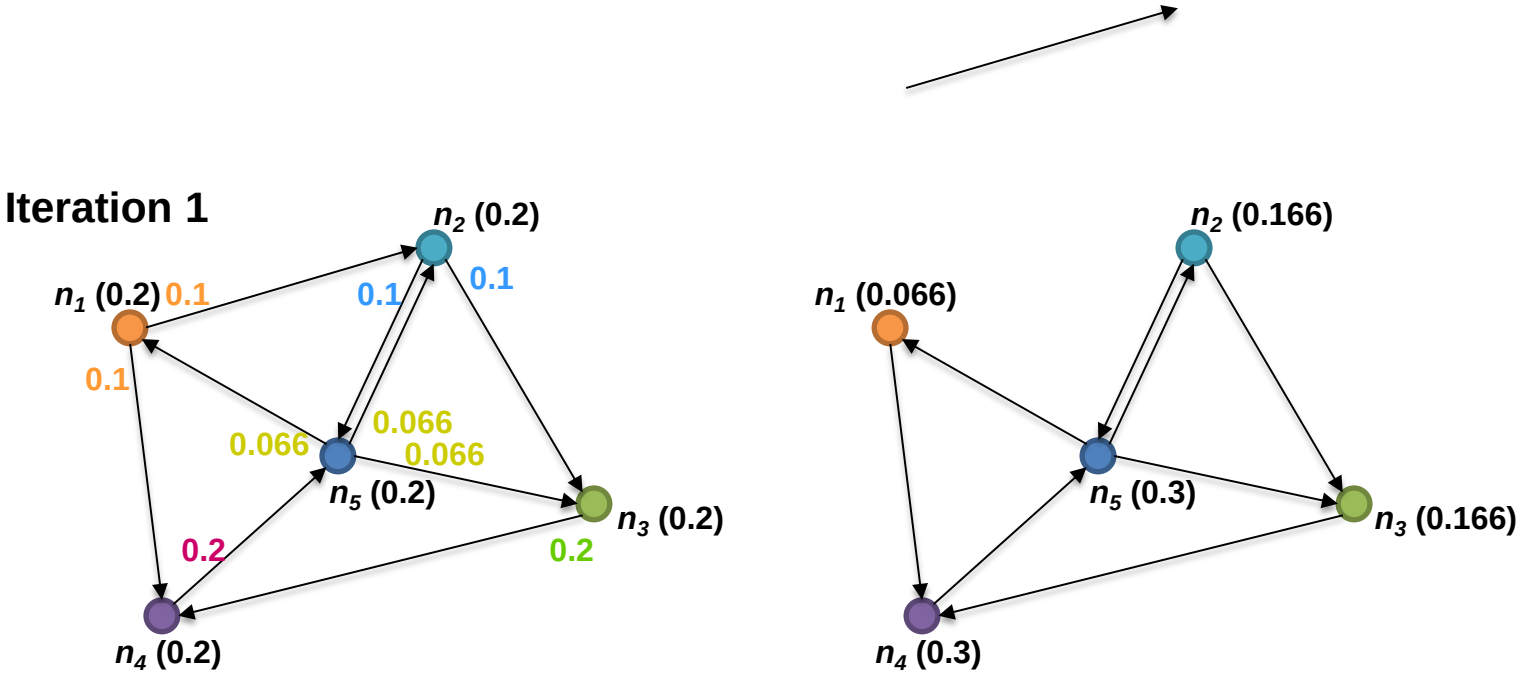
Computing Page Ranking

- Properties of PageRank
  - Can be computed iteratively
  - Effects at each iteration are local
- Sketch of algorithm:
  - Start with seed PRi values
  - Each page distributes PRi “credit” to all pages it links to
  - Each target page adds up “credit” from multiple in-bound links to compute PRi+1
  - Iterate until values converge

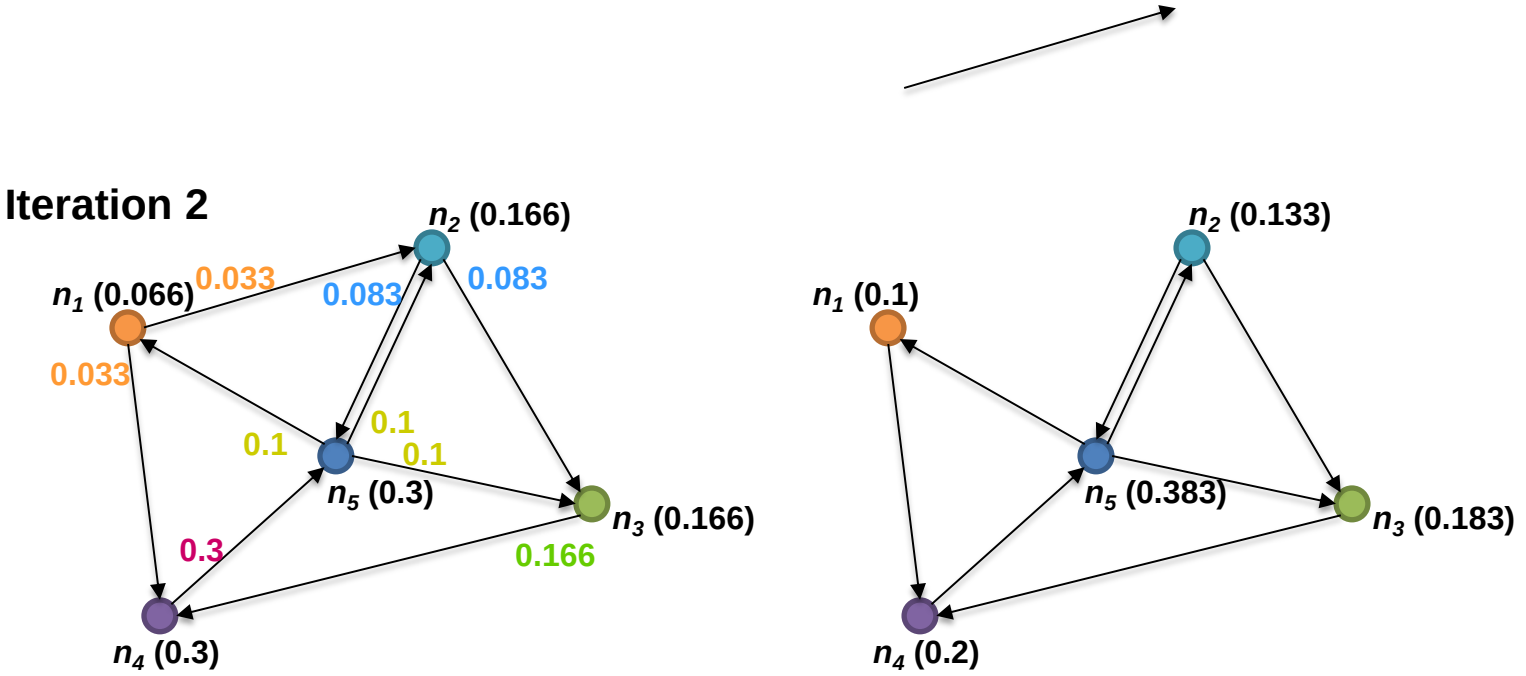
Computing Page Ranking

- Properties of PageRank
  - Can be computed iteratively
  - Effects at each iteration are local
- Sketch of algorithm:
  - Start with seed PRi values
  - Each page distributes PRi “credit” to all pages it links to
  - Each target page adds up “credit” from multiple in-bound links to compute PRi+1
  - Iterate until values converge

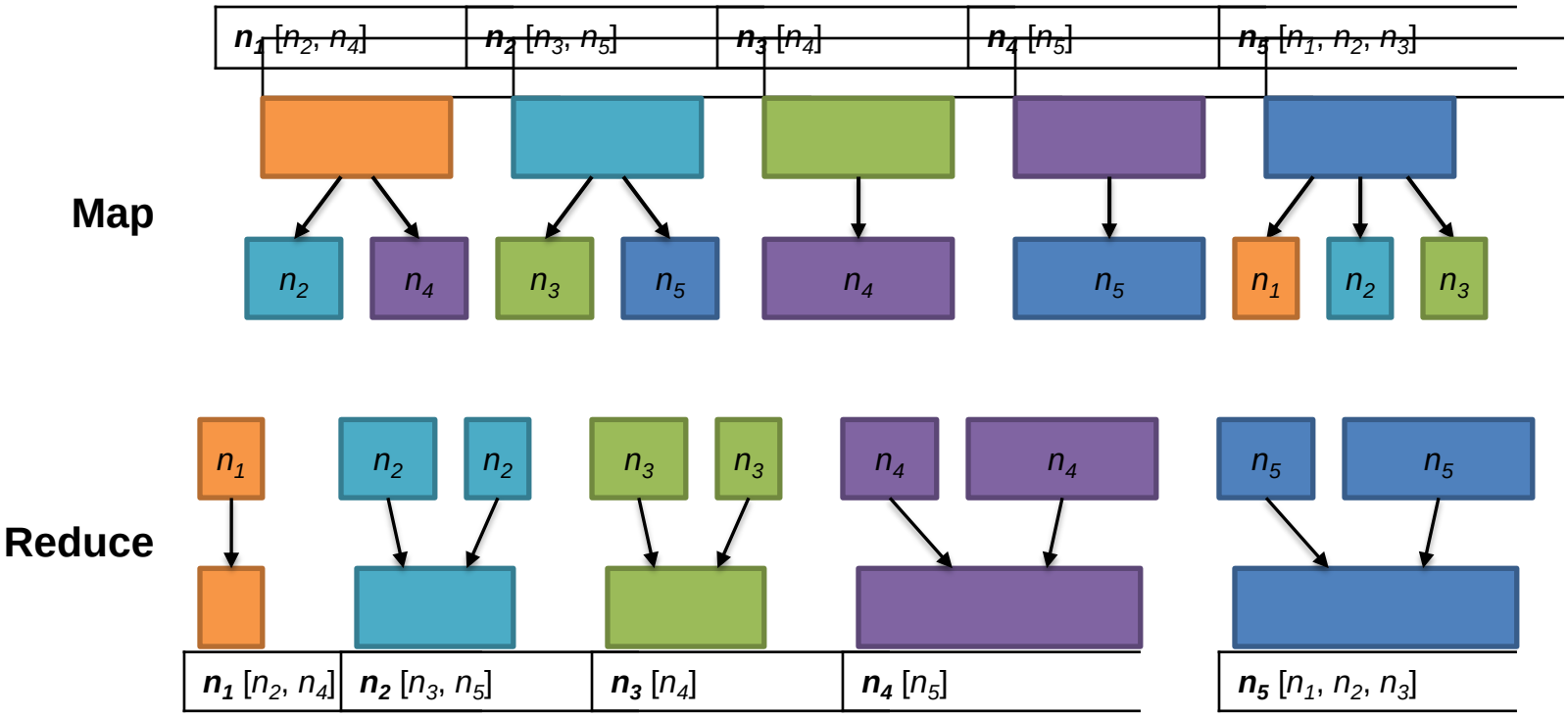
Sample PageRank Iteration (1)



Sample PageRank Iteration (2)



PageRank in MapReduce



Page Ranking Pseudo Code

```
1: class Mapper
2:     method Map(nid n, node N)
3:         p ← N.PageRank/|N.AdjacencyList|
4:         Emit(nid n, N)                          ▷ Pass along graph structure
5:         for all nodeid m ∈ N.AdjacencyList do
6:             Emit(nid m, p)                      ▷ Pass PageRank mass to neighbors

1: class Reducer
2:     method Reduce(nid m, [p1, p2, ...])
3:         M ← ∅
4:         for all p ∈ counts [p1, p2, ...] do
5:             if IsNode(p) then
6:                 M ← p                           ▷ Recover graph structure
7:             else
8:                 s ← s + p                       ▷ Sums incoming PageRank contributions
9:         M.PageRank ← s
10:        Emit(nid m, node M)
```