CSE 4/587
Data Intensive Computing

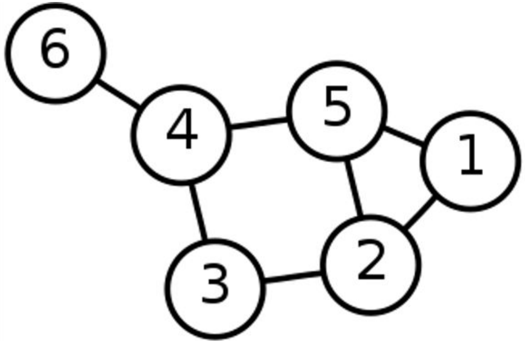
Dr. Eric Mikida
epmikida@buffalo.edu
208 Capen Hall

Day 12
Graph Analytics with MapReduce

Announcements and Feedback

- Project Phase 1 & 2 extended by 1 week
  - Phase 1 due 10/17@11:59PM

What is a Graph?



- A ***graph*** is a structure made up of a set of objects, where some pairs of the objects are "related"
- Mathematically, objects are represented with ***vertices*** (or nodes or points) and the relations between two vertices are represented with ***edges*** (or links or lines)
- Typically, a graph is depicted in diagrammatic form as a set of dots or circles for the vertices, joined by lines or curves for the edges
- Edges can be directed or undirected (a relationship can go both ways)
- Edges can be weighted to show the “strength”, distance, etc

Graph Representations

There are two standard ways to represent a graph G(V,E) ***[V is the set of vertices, E is the set of edges]***

1. adjacency list representation
2. adjacency matrix

An adjacency matrix is 2-Dimensional Array of size *V*x*V*, where *V* is the number of vertices in the graph.

	1	2	3	4
1	0	1	1	0
2	1	0	0	1
3	1	0	0	1
4	0	1	1	0

Undirected Graph

	1	2	3	4
1	0	1	1	0
2	0	0	0	1
3	0	0	0	0
4	0	0	1	0

Directed Graph

An adjacency list is an array of linked lists, where the array size is same as number of vertices in the graph. Every vertex has a linked list. Each node in this linked list represents the reference to another vertex that shares an edge with the current vertex.

Single Source Shortest Path

Sequential solution: Dijkstra's algorithm

```
d[s] = 0
for all other vertices d[v] = ∞
Q = {V} // Q is priority queue based on distances
while Q not empty
    u = min(Q) // node with min d value
    for all vertex v in u.adjacencyList
        if d[v] > d[u] + w[u,v]
            d[v] = d[u] + w[u,v]
    mark u and remove from Q
```

At each iteration of while loop, the algorithm expands the node with the shortest distance and updates distances to all reachable nodes

Single Source Shortest Path

How do we apply this algorithm if we have a graph with large number of nodes and edges between them? MapReduce?

What is the main issue here?

The algorithm is sequential, needs a ***global state***

Global states are not possible with map reduce…

Graph Processing in MapReduce

Let's see how we can handle a graph problem in parallel with MapReduce

Remember, the MapReduce paradigm requires a mapper function and a reducer function

Issues in Processing a Graph in MR

Goal: Start from a given node and label all the nodes in the graph so that we can determine the shortest distance.

Assume all the distance between edges is one.

Issues in Processing a Graph in MR

Goal: Start from a given node and label all the nodes in the graph so that we can determine the shortest distance.

Assume all the distance between edges is one.

Problems to Tackle:

Issues in Processing a Graph in MR

Goal: Start from a given node and label all the nodes in the graph so that we can determine the shortest distance.

Assume all the distance between edges is one.

Problems to Tackle:

1. How do we represent our graph?

Issues in Processing a Graph in MR

Goal: Start from a given node and label all the nodes in the graph so that we can determine the shortest distance.

Assume all the distance between edges is one.

Problems to Tackle:

1. How do we represent our graph?
2. What are our <key,value> pairs?

Issues in Processing a Graph in MR

Goal: Start from a given node and label all the nodes in the graph so that we can determine the shortest distance.

Assume all the distance between edges is one.

Problems to Tackle:

1. How do we represent our graph?
2. What are our <key,value> pairs?
3. How do we iterate through various stages of processing?

Issues in Processing a Graph in MR

Goal: Start from a given node and label all the nodes in the graph so that we can determine the shortest distance.

Assume all the distance between edges is one.

Problems to Tackle:

1. How do we represent our graph?
2. What are our <key,value> pairs?
3. How do we iterate through various stages of processing?
4. When/how do we terminate execution?

Graph Representation and Input Format

Our graphs will be represented as a collection of **Node** objects

```
Node:
    nodeId,
    distanceLabel,
    adjancencyList[nodeId, distance],
    ...
```

Input the graph as text and parse it to build our `<key, value>` pairs

So what are our `<key, value>` *pairs?*

<key, value> pairs

We actually need two types of <key, value> pair:

```
1.  <nodeId n, Node N>      // nodeId to Node object
2.  <nodeId n, distance>    // nodeId to distance so far
```

Iteration

- Each *iteration* in the algorithm is a MapReduce job
- Iterations and termination are coordinate by an external *driver* application (more on this in future lectures)
- The first iteration starts at the source node (with distance 0)
  - It updates and emits all distances for nodes in the adjacency list
- The next iteration takes the output from the previous and updates/emits all distances for nodes connected to this set of nodes
- Continue until termination

Termination

Termination condition also needs to be tracked in the Node class

Terminate when the graph has reached a steady state:

- All the nodes have been labeled with min distance
- Labels no longer change between iterations
- Potentially use other conditions using counters

Mapper Class

```
class Mapper
  method map (nodeId n, Node N)
    d ← N.distance
    emit(n, N)                                // type 1
    for nodeId m in N.adjacencyList
        emit(m, d+1)                          // type 2
```

The method map takes in two parameters, **`nodeId n`** and **`Node N`**

The method produces two key value pairs **`<n, N>`** and the updated distance to all of the adjacent nodes **`<m, d+1>`**

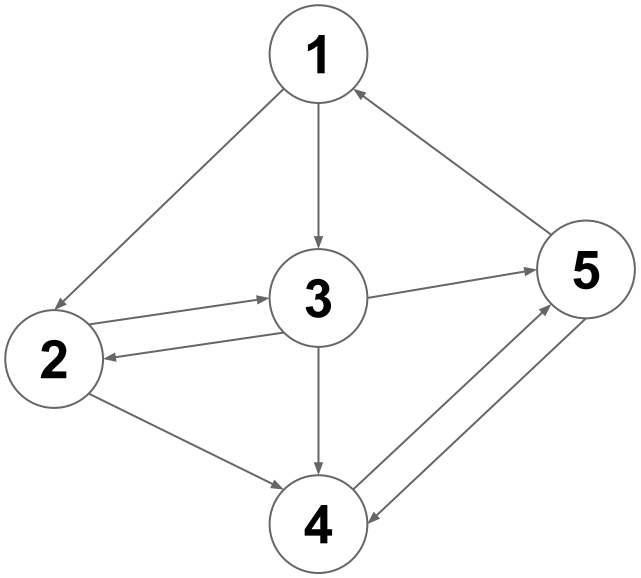
Reducer Class

```
class Reducer
  method Reduce(nodeId n, [d1, d2, d3..])
    dmin ← ∞;      // or a large #
    Node N ← null

    for all d in [d1, d2, ...]:
      if IsNode(d) then N ← d
      else if d < dmin then dmin ← d

      N.distance ← dmin  // update the shortest distance in N
      emit (n, N)
```

Trace with Sample Data

Input Graph

1	0	2:3:
2	10000	3:4:
3	10000	2:4:5
4	10000	5:
5	10000	1:4

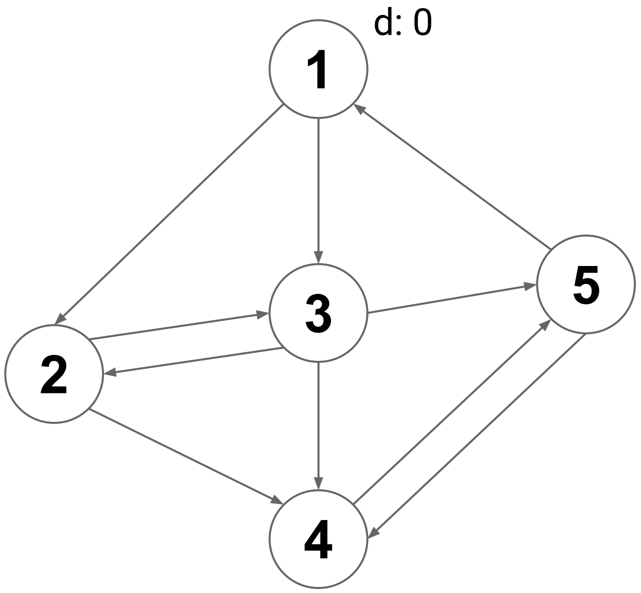


Trace with Sample Data

Input Graph

1	0	2:3:
2	10000	3:4:
3	10000	2:4:5
4	10000	5:
5	10000	1:4

Input to the first iteration

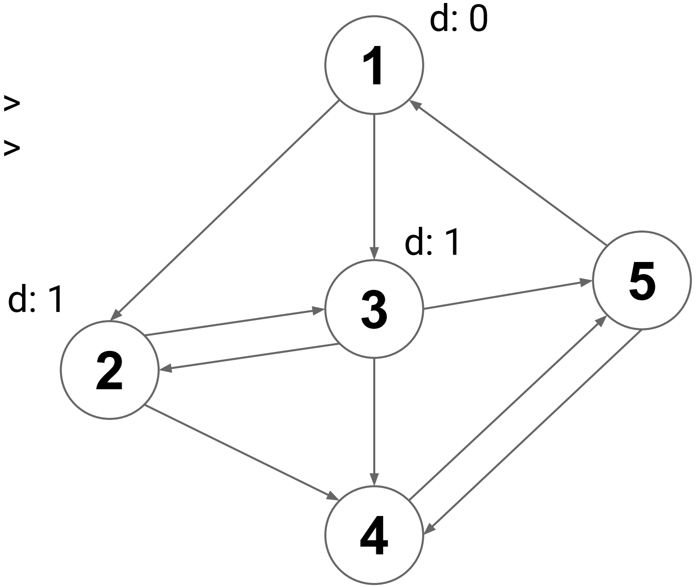


Intermediate Result

Input Graph

1	0	2:3:
2	1	3:4:
3	1	2:4:5
4	10000	5:
5	10000	1:4

<nodeId 2, distance 1>
<nodeId 3, distance 1>



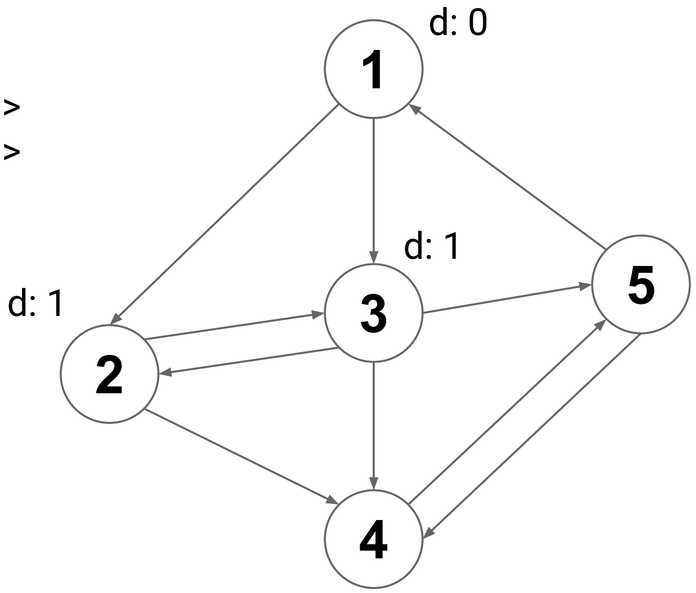
Intermediate Result

Input Graph

1	0	2:3:
2	1	3:4:
3	1	2:4:5
4	10000	5:
5	10000	1:4

Input to the next iteration

<nodeId 2, distance 1>
<nodeId 3, distance 1>

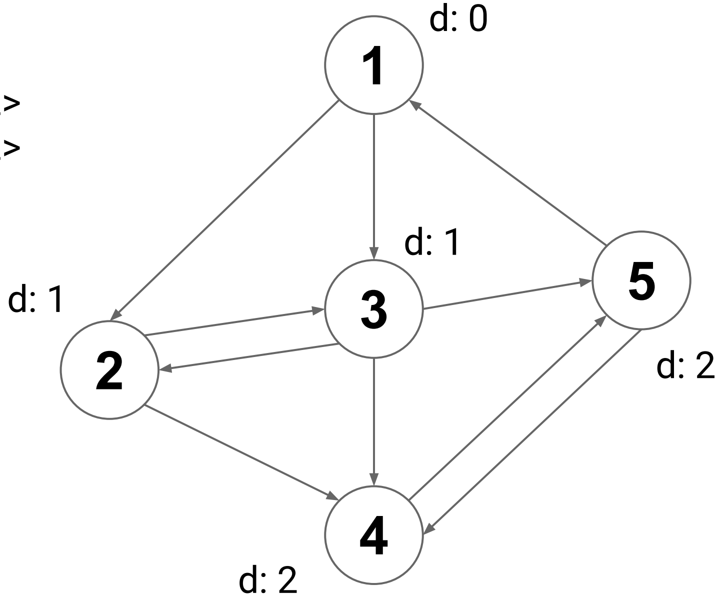


Final Result

Input Graph

1	0	2:3:
2	1	3:4:
3	1	2:4:5
4	2	5:
5	2	1:4

<nodeId 4, distance 2>
<nodeId 5, distance 2>



PageRank

- Last time we looked at PageRank
  - Algorithm for ranking the importance of pages on the internet
  - Internet represented as a graph
    - Pages are vertices
    - Links are edges
- The internet is huge, graph requires parallel processing…

How can we do PageRank in MapReduce?

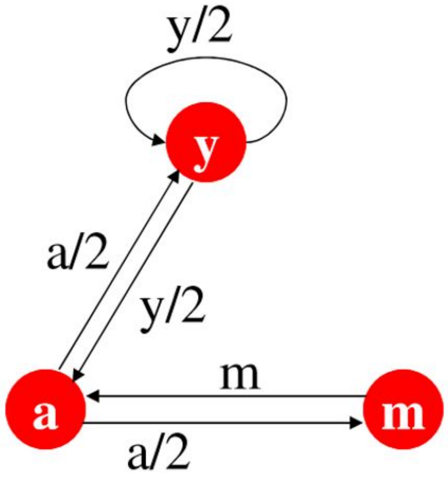
Page Rank: The Flow Model

A link from an *important page* (higher ranking page) is worth more

A page is *important* if it is pointed to by other important pages

Define a "rank" $\boldsymbol{r_j}$ for page $\boldsymbol{j}$ as:

$$r_j = \sum_{i \to j} \frac{r_i}{\mathrm{d_i}}$$



"Flow" equations:

$$\mathbf{r_y} = \mathbf{r_y}/2 + \mathbf{r_a}/2$$
$$\mathbf{r_a} = \mathbf{r_y}/2 + \mathbf{r_m}$$
$$\mathbf{r_m} = \mathbf{r_a}/2$$

Solving the Flow Equation

3 equations, 3 unknowns, no constants

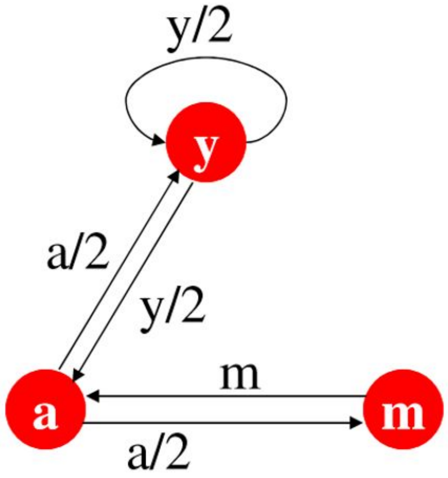
No unique solution: All solutions equivalent modulo the scale factor

Adding an additional constraint forces uniqueness:

$$r_y + r_a + r_m = 1$$

Gaussian Elimination can be used to find the solution.

This method will work for small graphs, but won't scale for larger graphs



"Flow" equations:

$$r_y = r_y/2 + r_a/2$$
$$r_a = r_y/2 + r_m$$
$$r_m = r_a/2$$

Page Rank: Matrix Formulation

Stochastic Adjacency matrix ***M***

$\boldsymbol{M_{ji}} = \mathbf{1/(}\boldsymbol{d_i}\mathbf{)}$ if there is a link from ***i*** to ***j***, else value is 0

Page Rank: Matrix Formulation

Stochastic Adjacency matrix $\boldsymbol{M}$

$\boldsymbol{M_{ji} = 1/(d_i)}$ if there is a link from $\boldsymbol{i}$ to $\boldsymbol{j}$, else value is 0

If $\boldsymbol{r}$ is vector with the initial importance of a page and

$$\sum_i r_i = 1$$

Page Rank: Matrix Formulation

Stochastic Adjacency matrix ***M***

$M_{ji} = 1/(d_i)$ if there is a link from ***i*** to ***j***, else value is 0

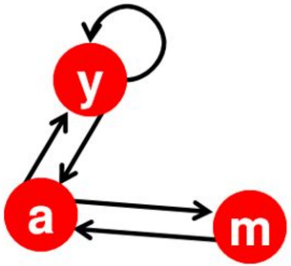
If ***r*** is vector with the initial importance of a page and

$$\sum_i r_i = 1$$

Then the flow equation can be written as

$$\boldsymbol{r} = \boldsymbol{M} \cdot \boldsymbol{r}$$

Solving with Power Iteration



	y	a	m
y	½	½	0
a	½	0	1
m	0	½	0

$$r = M \cdot r$$

$$r_y = r_y/2 + r_a/2$$
$$r_a = r_y/2 + r_m$$
$$r_m = r_a/2$$

$$\begin{bmatrix} y \\ a \\ m \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & 1 \\ 0 & \frac{1}{2} & 0 \end{bmatrix} \begin{bmatrix} y \\ a \\ m \end{bmatrix}$$

Solving with Power Iteration

Given a web graph with ***n*** nodes, where the vertices are pages and edges are hyperlinks

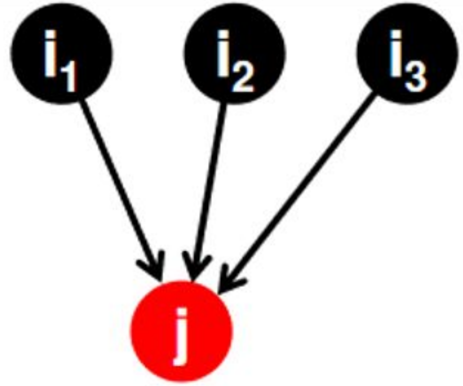
Power iteration: a simple iterative scheme

Suppose there are ***N*** web pages

1. **Initialize:** $r(0) = [1/N,\ldots.,1/N]^T$
2. **Iterate:** $r(t+1) = M \cdot r(t)$
3. **Stop when:** $\|r(t+1) - r(t)\|_1 < \varepsilon$

Random Walk Interpretation

Imagine a random web surfer

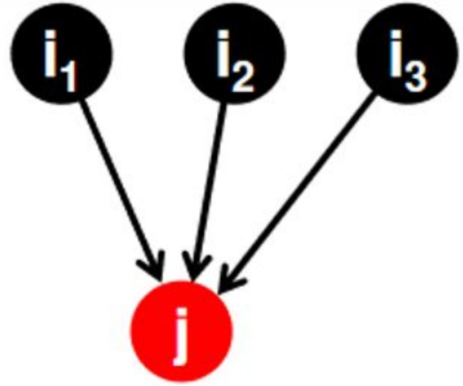


$$r_j = \sum_{i \to j} \frac{r_i}{\mathrm{d}_{\mathrm{out}}(\mathrm{i})}$$

Random Walk Interpretation

Imagine a random web surfer

- At any time ***t***, the surfer is on some page ***i***

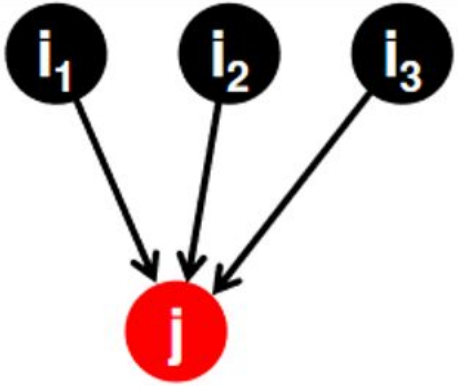


$$r_j = \sum_{i \to j} \frac{r_i}{\mathrm{d_{out}}(\mathrm{i})}$$

Random Walk Interpretation

Imagine a random web surfer

- At any time ***t***, the surfer is on some page ***i***
- At time ***t* + 1**, the surfer follows an out-link from ***i*** uniformly at random
  - Ends up on some page ***j*** linked from ***i***

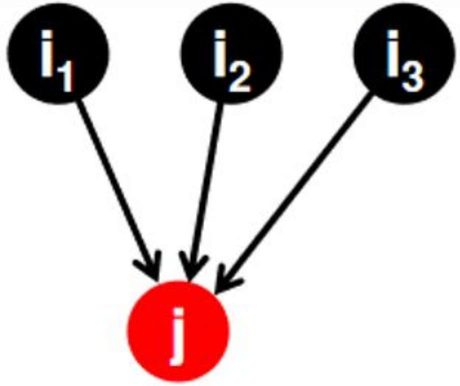


$$r_j = \sum_{i \to j} \frac{r_i}{\mathrm{d}_{\mathrm{out}}(\mathrm{i})}$$

Random Walk Interpretation

Imagine a random web surfer

- At any time ***t***, the surfer is on some page ***i***
- At time ***t* + 1**, the surfer follows an out-link from ***i*** uniformly at random
  - Ends up on some page ***j*** linked from ***i***
- Process repeats infinitely



$$r_j = \sum_{i \to j} \frac{r_i}{\mathrm{d}_{\mathrm{out}}(\mathrm{i})}$$

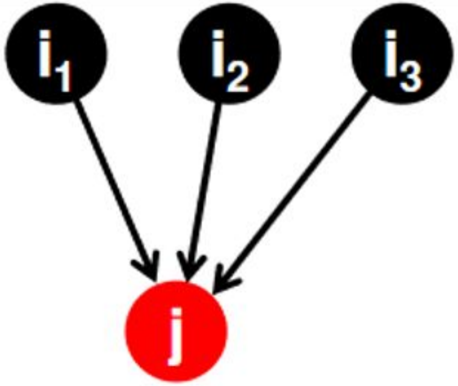
Random Walk Interpretation

Imagine a random web surfer

- At any time ***t***, the surfer is on some page ***i***
- At time ***t* + 1**, the surfer follows an out-link from ***i*** uniformly at random
  - Ends up on some page ***j*** linked from ***i***
- Process repeats infinitely

P*(*t*)** is the vector whose ***ith coordinate is the probability that the surfer is at page ***i*** at time ***t***

So ***P*(*t*)** is a probability distribution over pages



$$r_j = \sum_{i \to j} \frac{r_i}{\mathrm{d}_{\mathrm{out}}(\mathrm{i})}$$

Google Formulation

$$r_j^{(t+1)} = \sum_{i \to j} \frac{r_i^{(t)}}{\mathrm{d}_\mathrm{i}} \quad \text{or equivalently} \quad r = Mr$$

Google Formulation

$$r_j^{(t+1)} = \sum_{i \to j} \frac{r_i^{(t)}}{\mathrm{d}_\mathrm{i}} \quad \text{or equivalently} \quad r = Mr$$

Does this value converge ?

Google Formulation

$$r_j^{(t+1)} = \sum_{i \to j} \frac{r_i^{(t)}}{d_i} \quad \text{or equivalently} \quad r = Mr$$

Does this value converge ?

Does it converge to the results that we want?

Google Formulation

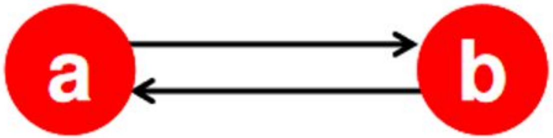
$$r_j^{(t+1)} = \sum_{i \to j} \frac{r_i^{(t)}}{\mathrm{d_i}} \quad \text{or equivalently} \quad r = Mr$$

Does this value converge ?

Does it converge to the results that we want?

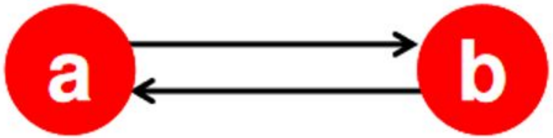
Are the results reasonable?

Does this converge?



$$r_j^{(t+1)} = \sum_{i \to j} \frac{r_i^{(t)}}{\mathrm{d}_i}$$

Does this converge?



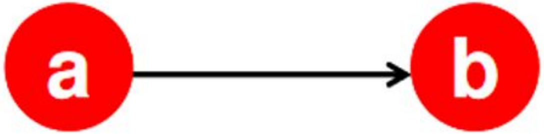
$$r_j^{(t+1)} = \sum_{i \to j} \frac{r_i^{(t)}}{d_i}$$

- **Example:**

$$\begin{array}{ccccc} r_a & & 1 & 0 & 1 & 0 \\ & = & & & & \\ r_b & & 0 & 1 & 0 & 1 \end{array}$$

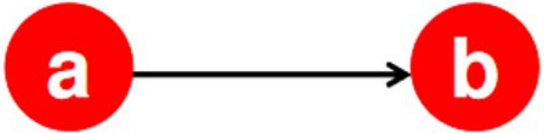
Iteration 0, 1, 2, …

Does this converge to what we want?



$$r_j^{(t+1)} = \sum_{i \to j} \frac{r_i^{(t)}}{d_i}$$

Does this converge to what we want?



$$r_j^{(t+1)} = \sum_{i \to j} \frac{r_i^{(t)}}{\mathrm{d}_\mathrm{i}}$$

- **Example:**

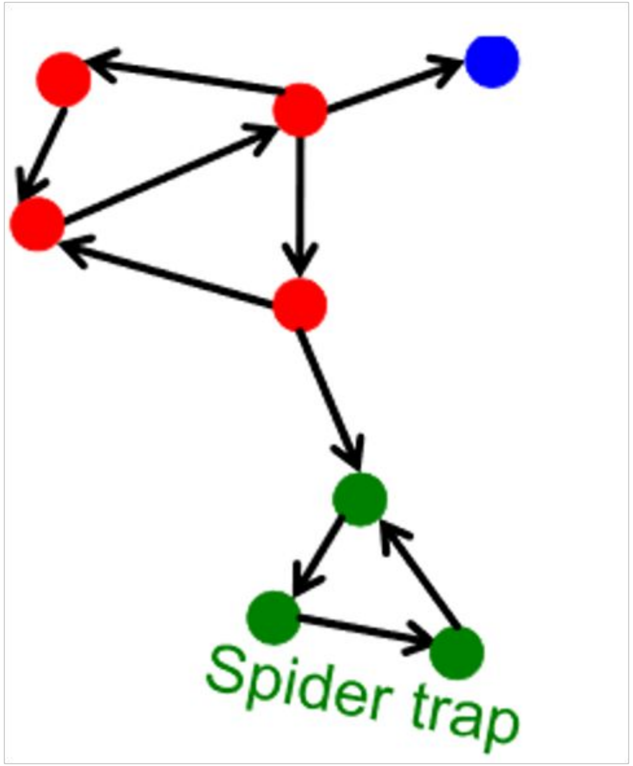
$$\begin{matrix} \mathrm{r_a} \\ \mathrm{r_b} \end{matrix} = \begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{matrix}$$

Iteration 0, 1, 2, …

Page Rank: Problems

Some pages are dead ends:

- Random walk has nowhere to go
- Such pages cause important information to leak



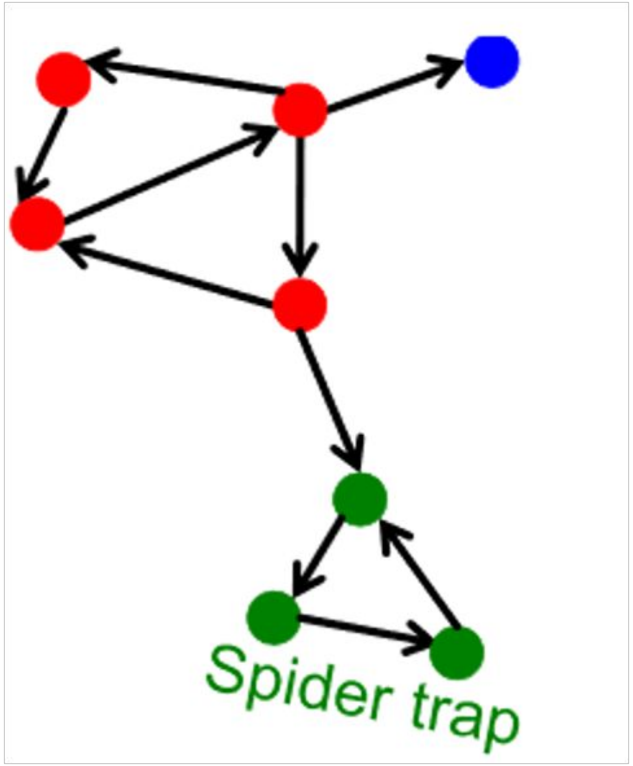
Page Rank: Problems

Some pages are dead ends:

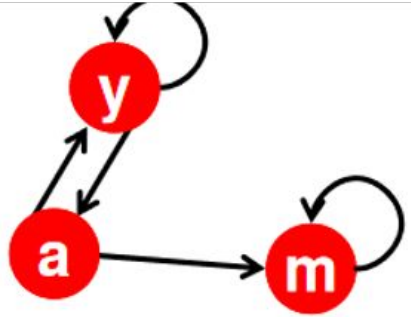
- Random walk has nowhere to go
- Such pages cause important information to leak

Spider traps

- All out-links are within the group
- Random walk gets stuck in a trap
- And eventually spider traps absorbs all importance



Spider Traps

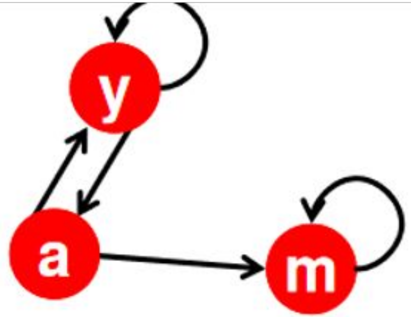


m is a spider trap

	y	a	m
y	½	½	0
a	½	0	0
m	0	½	1

$$r_y = r_y/2 + r_a/2$$
$$r_a = r_y/2$$
$$r_m = r_a/2 + r_m$$

Spider Traps



	y	a	m
y	½	½	0
a	½	0	0
m	0	½	1

m is a spider trap

$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{matrix} 1/3 & 2/6 & 3/12 & 5/24 & & 0 \\ 1/3 & 1/6 & 2/12 & 3/24 & \dots & 0 \\ 1/3 & 3/6 & 7/12 & 16/24 & & 1 \end{matrix}$$

Iteration 0, 1, 2, …

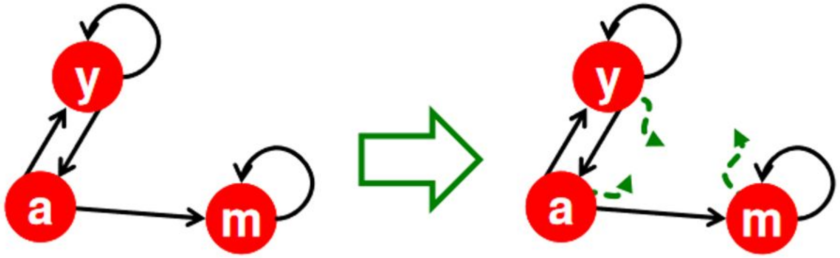
All the PageRank score gets "trapped" in node m.

Solution: Teleports

The Google solution for spider traps: Teleports

At each time step, the random surfer has two options:

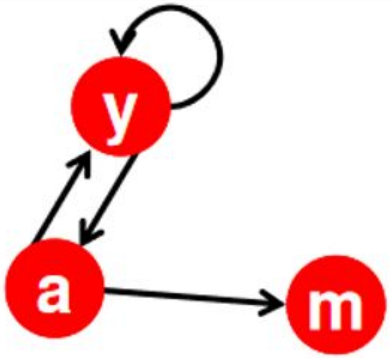
1. With probability β, follow a link at random
2. With prob. 1-β, jump to some random page



Common values for β are in the range 0.8 to 0.9

This will help the surfer to teleport out of spider trap within a few steps

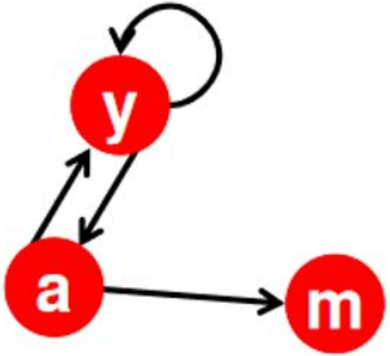
Dead Ends



	y	a	m
y	½	½	0
a	½	0	0
m	0	½	0

$$r_y = r_y/2 + r_a/2$$
$$r_a = r_y/2$$
$$r_m = r_a/2$$

Dead Ends



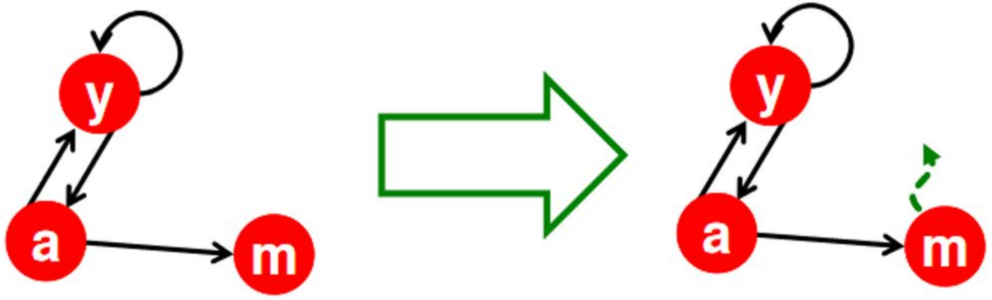
	y	a	m
y	½	½	0
a	½	0	0
m	0	½	0

$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{matrix} 1/3 & 2/6 & 3/12 & 5/24 & & 0 \\ 1/3 & 1/6 & 2/12 & 3/24 & \ldots & 0 \\ 1/3 & 1/6 & 1/12 & 2/24 & & 0 \end{matrix}$$

Iteration 0, 1, 2, …

Solution: Teleports

Teleport with probability 1.0 at dead ends



	y	a	m
y	½	½	0
a	½	0	0
m	0	½	0

	y	a	m
y	½	½	⅓
a	½	0	⅓
m	0	½	⅓

Google's Solution

Googles solution for PageRank:

$$r_j = \sum_{i \to j} \beta \frac{r_i}{d_i} + (1 - \beta) \frac{1}{N}$$

Google's Solution

Googles solution for PageRank:

$$r_j = \sum_{i \to j} \beta \frac{r_i}{d_i} + (1-\beta)\frac{1}{N}$$

In matrix notation:

$$A = \beta\, M + (1-\beta)\left[\frac{1}{N}\right]_{N \times N}$$

PageRank with MapReduce

Let us assume for now that β is 1.0 (no teleporting)

```
class Mapper
  method map (nodeId n, Node N)
    p ← N.pagerank / N.adajacencyList.size
    emit(n, N)  // Emit the graph structure
    for all m in N.adjacencyList
      emit(m, p)// Emit the contributions from N
```

PageRank with MapReduce

```
Class Reducer
  method Reduce(nodeId n, [p1, p2, p3..])
    node N ← null; s ← 0;
    for all p in [p1,p2, ..]:
      if IsNode(p) then: N ← p
      else: s ← s + p
    N.pagerank ← s
    emit(n, N)
```

PageRank with MapReduce

How do we account for dead ends nodes?

- Simply redistribute its PageRank to all other nodes
- One iteration requires PageRank computation + redistribution of “unused” PageRank
    - Track total leaked PageRank during the computation, then redistribute it as a second MapReduce job in the same iteration

PageRank with MapReduce

How do we account for dead ends nodes?

- Simply redistribute its PageRank to all other nodes
- One iteration requires PageRank computation + redistribution of “unused” PageRank
  - Track total leaked PageRank during the computation, then redistribute it as a second MapReduce job in the same iteration

What about breaking out of spider traps?

- This can also be taken care of when the leaked PageRank is redistributed from dead ends

PageRank with MapReduce

How do we account for dead ends nodes?

- Simply redistribute its PageRank to all other nodes
- One iteration requires PageRank computation + redistribution of “unused” PageRank
  - Track total leaked PageRank during the computation, then redistribute it as a second MapReduce job in the same iteration

What about breaking out of spider traps?

- This can also be taken care of when the leaked PageRank is redistributed from dead ends

Second Phase Redistribution Formula:

$$p' = \alpha\left(\frac{1}{|G|}\right) + (1-\alpha)\left(\frac{m}{|G|} + p\right)$$

PageRank with MapReduce

How do we account for dead ends nodes?

- Simply redistribute its PageRank to all other nodes
- One iteration requires PageRank computation + redistribution of “unused” PageRank
  - Track total leaked PageRank during the computation, then redistribute it as a second MapReduce job in the same iteration

What about breaking out of spider traps?

- This can also be taken care of when the leaked PageRank is redistributed from dead ends

Second Phase Redistribution Formula:

Random hop probability

Total leaked PageRank "mass"

$$p' = \alpha \left(\frac{1}{|G|} \right) + (1 - \alpha) \left(\frac{m}{|G|} + p \right)$$

PageRank from first phase

PageRank with MapReduce

How do we know when convergence is reached?

- When the ranks of pages do not change (or change by less than some small epsilon value)

For large graphs, the rank of any particular node is often so small that it underflows standard floating point representations

- A very common solution to this problem is to represent ranks using their logarithms