Graphs II

Centrality, and algorithms you should know

Duen Horng (Polo) Chau
Georgia Tech

Partly based on materials by
Professors Guy Lebanon, Jeffrey Heer, John Stasko, Christos Faloutsos, Le Song

Centrality

= “Importance”

Why Node Centrality?

What can we do if we can rank all the nodes in a graph (e.g., Facebook, LinkedIn, Twitter)?

- Find **celebrities** or influential people in a social network (Twitter)
- Find "**gatekeepers**" who connect communities (headhunters love to find them on LinkedIn)
- What else?

More generally

Helps **graph analysis, visualization, understanding**, e.g.,

- let us rank nodes, group or study them by centrality
- only show subgraph formed by the **top 100 nodes**, out of the millions in the full graph
    - similar to google search results (ranked, and they only show you 10 per page)

Can also compute edge centrality.
Here we focus on node centrality.

Degree Centrality (easiest)

Degree = number of neighbors

For directed graphs

- in degree = # incoming edges
- out degree = # outgoing edges

Algorithms?

- Sequential scan through edge list
- What about for a graph stored in SQLite?

Computing degrees using SQL

Recall simplest way to store a graph in SQLite:

```
edges(source_id, target_id)
```

1. Create index for each column

2. Use **group by** statement to find node degrees

```
select count(*) from edges group by source_id;
```

Betweenness Centrality

High betweenness

- = important "gatekeeper" or liaison

A node's betweenness

- = $\sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}$

Number of shortest paths between s and t that involves v

Number of shortest paths between s and t

Needs to compute all-pairs shortest path (O(N³)). Slow.

Clustering Coefficient

Technically not a centrality measure, but useful

A node's clustering coefficient is a measure of how close the node's neighbors are from forming a clique.

- Value of 1 = neighbors form a clique
- Value of 0 = No edges among neighbors

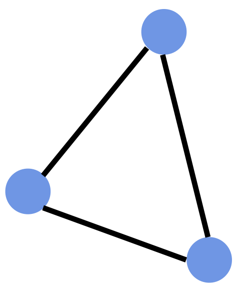
(Assuming undirected graph)

Computing Clustering Coefficient...

Requires triangle counting

Real social networks have a lot of triangles

- Friends of friends are friends



But: triangles are expensive to compute

(3-way join; several approx. algos)

Can we do that quickly?

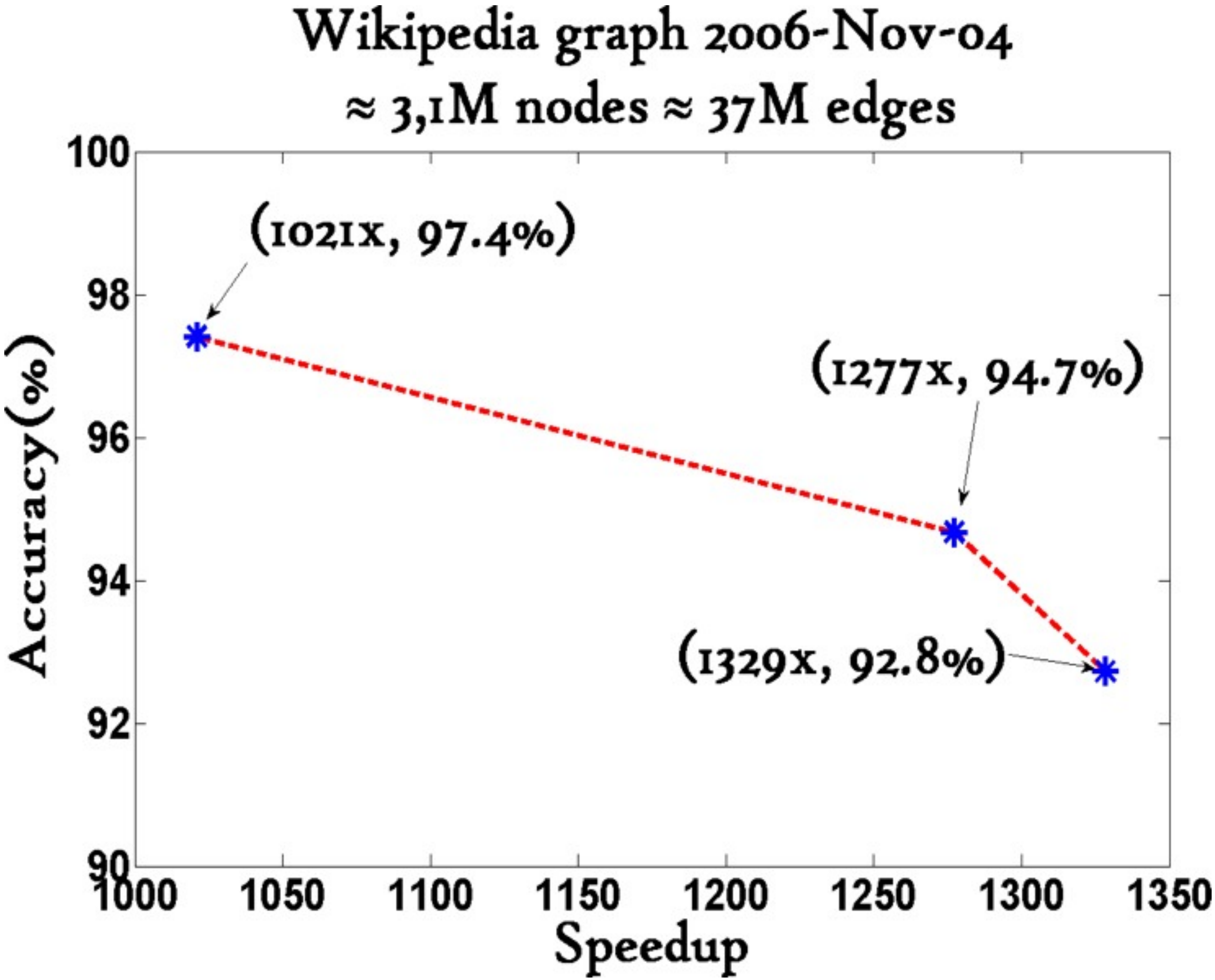
details

Super Fast Triangle Counting
[Tsourakakis ICDM 2008]

But: triangles are expensive to compute
(3-way join; several approx. algos)
Q: Can we do that quickly?
A: Yes!

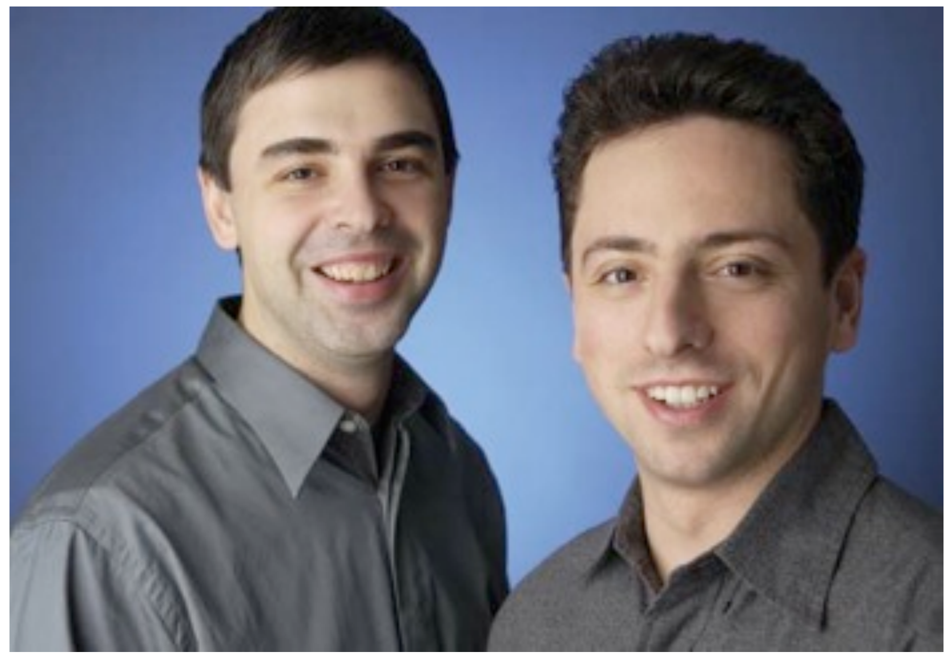
#triangles = 1/6 Sum ($(\lambda i)^3$)

(and, because of skewness,
we only need the top few eigenvalues!



1000x+ speed-up, >90% accuracy

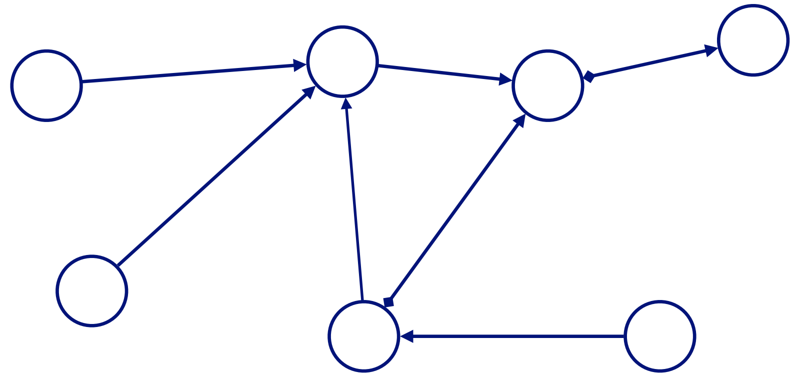
PageRank (google)

Larry Page  Sergey Brin

Brin, Sergey and Lawrence Page (1998). *Anatomy of a Large-Scale Hypertextual Web Search Engine*. 7th Intl World Wide Web Conf.

Problem: PageRank

Given a directed graph, find its most interesting/central node

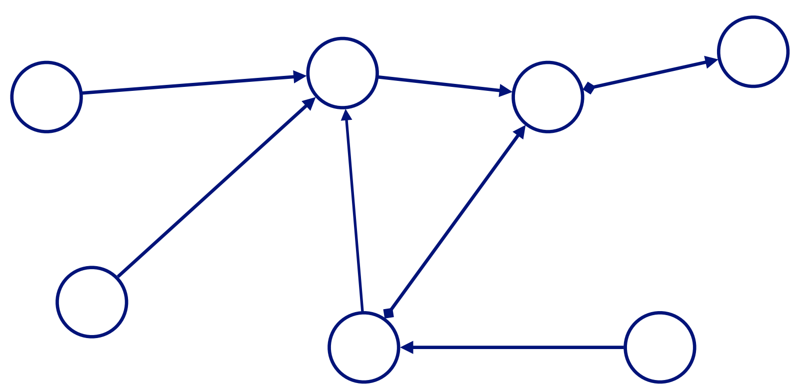


A node is important, if it is connected with important nodes (recursive, but OK!)

Problem: PageRank - solution

Given a directed graph, find its most interesting/central node

Proposed solution: Random walk; spot most ‘popular’ node (-> steady state prob. (ssp))

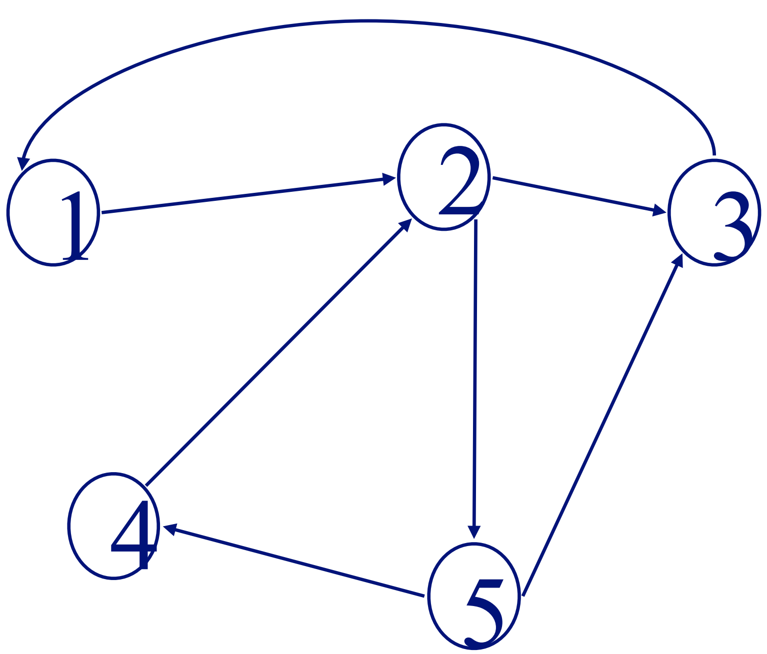


“state” = webpage

A node has high ssp, if it is connected with high ssp nodes (recursive, but OK!)

(Simplified) PageRank

- Let $\mathbf{A}$ be the adjacency matrix;
- let $\mathbf{B}$ be the transition matrix: transpose, column-normalized - then



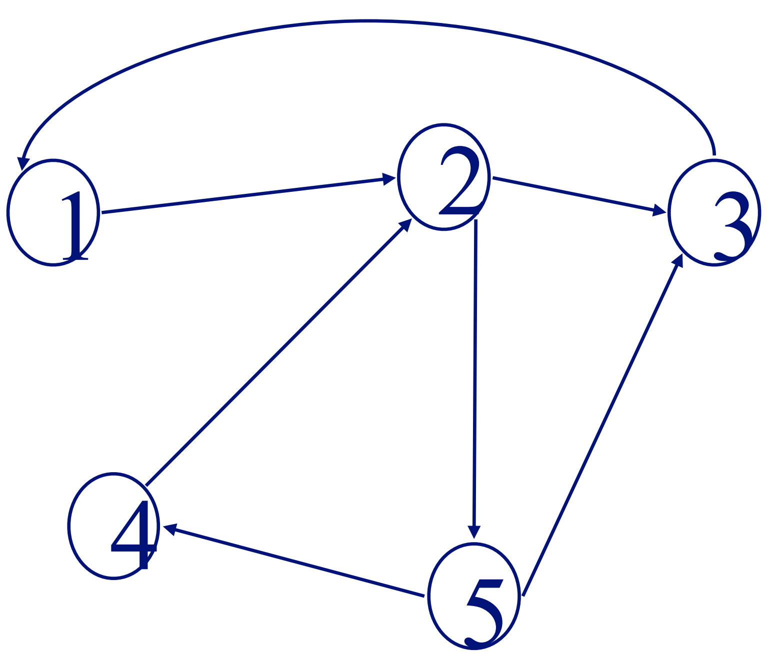
To \ From

$$\mathbf{B}$$

$$\begin{bmatrix} & & 1 & & \\ 1 & & & 1 & \\ & 1/2 & & & 1/2 \\ & & & & 1/2 \\ & 1/2 & & & \end{bmatrix} \begin{bmatrix} p1 \\ p2 \\ p3 \\ p4 \\ p5 \end{bmatrix} = \begin{bmatrix} p1 \\ p2 \\ p3 \\ p4 \\ p5 \end{bmatrix}$$

(Simplified) PageRank

- **B p = p**



$$\overset{\mathbf{B}}{\begin{bmatrix} & & 1 & & \\ 1 & & & 1 & \\ & 1/2 & & & 1/2 \\ & & & & 1/2 \\ & 1/2 & & & \end{bmatrix}} \overset{\mathbf{p}}{\begin{bmatrix} p1 \\ p2 \\ p3 \\ p4 \\ p5 \end{bmatrix}} \overset{=}{=} \overset{\mathbf{p}}{\begin{bmatrix} p1 \\ p2 \\ p3 \\ p4 \\ p5 \end{bmatrix}}$$

(Simplified) PageRank

- $\mathbf{B}\,\mathbf{p} = 1 * \mathbf{p}$
- thus, $\mathbf{p}$ is the **eigenvector** that corresponds to the highest eigenvalue (=1, since the matrix is column-normalized)
- Why does such a $\mathbf{p}$ exist?
  - $\mathbf{p}$ exists if $\mathbf{B}$ is nxn, nonnegative, irreducible [Perron–Frobenius theorem]

(Simplified) PageRank

- $\mathbf{B}\ \mathbf{p} = 1 * \mathbf{p}$
- thus, $\mathbf{p}$ is the **eigenvector** that corresponds to the highest eigenvalue (=1, since the matrix is column-normalized)
- Why does such a $\mathbf{p}$ exist?
  - $\mathbf{p}$ exists if $\mathbf{B}$ is nxn, nonnegative, irreducible [Perron–Frobenius theorem]

(Simplified) PageRank

- In short: imagine a particle randomly moving along the edges
- compute its steady-state probabilities (ssp)

Full version of algo:
  with occasional random jumps

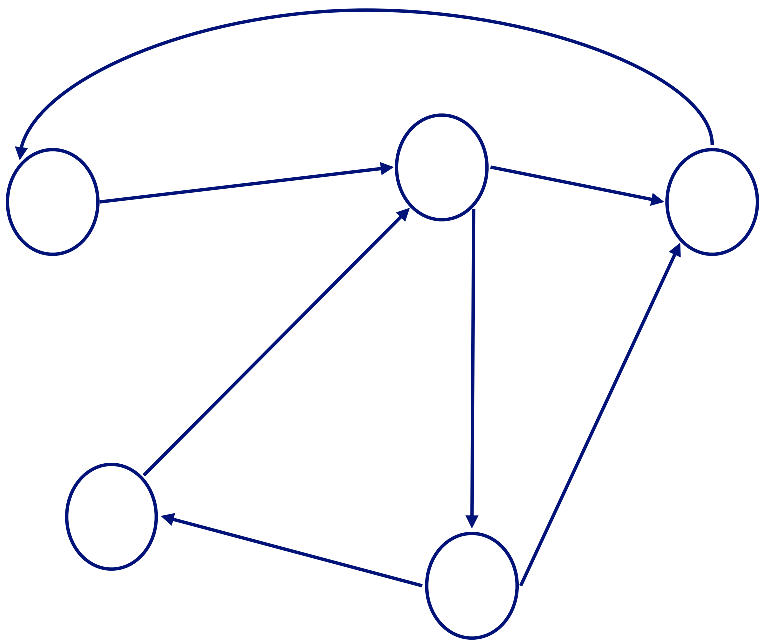
Why? To make the matrix irreducible

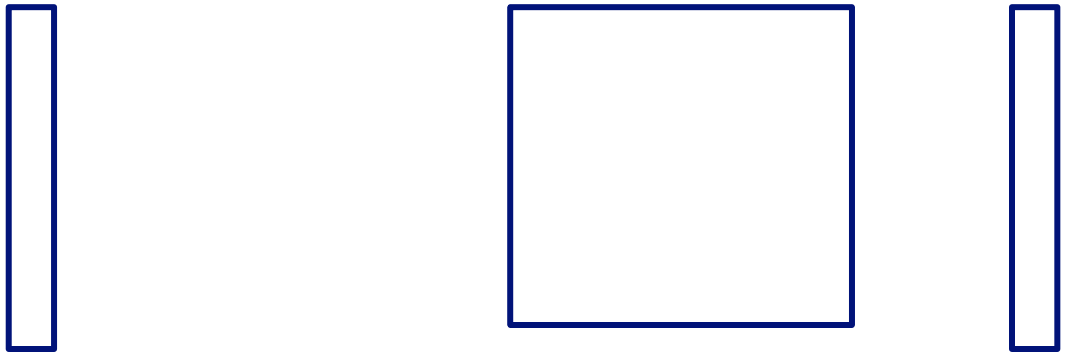
Full Algorithm

- With probability *1-c*, fly-out to a random node
- Then, we have

  $\mathbf{p} = c\,\mathbf{B}\,\mathbf{p} + (1-c)/n\;\mathbf{1} \Rightarrow$

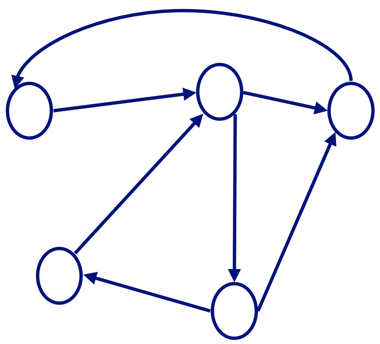
  $\mathbf{p} = (1-c)/n\;[\mathbf{I} - c\,\mathbf{B}]^{-1}\;\mathbf{1}$





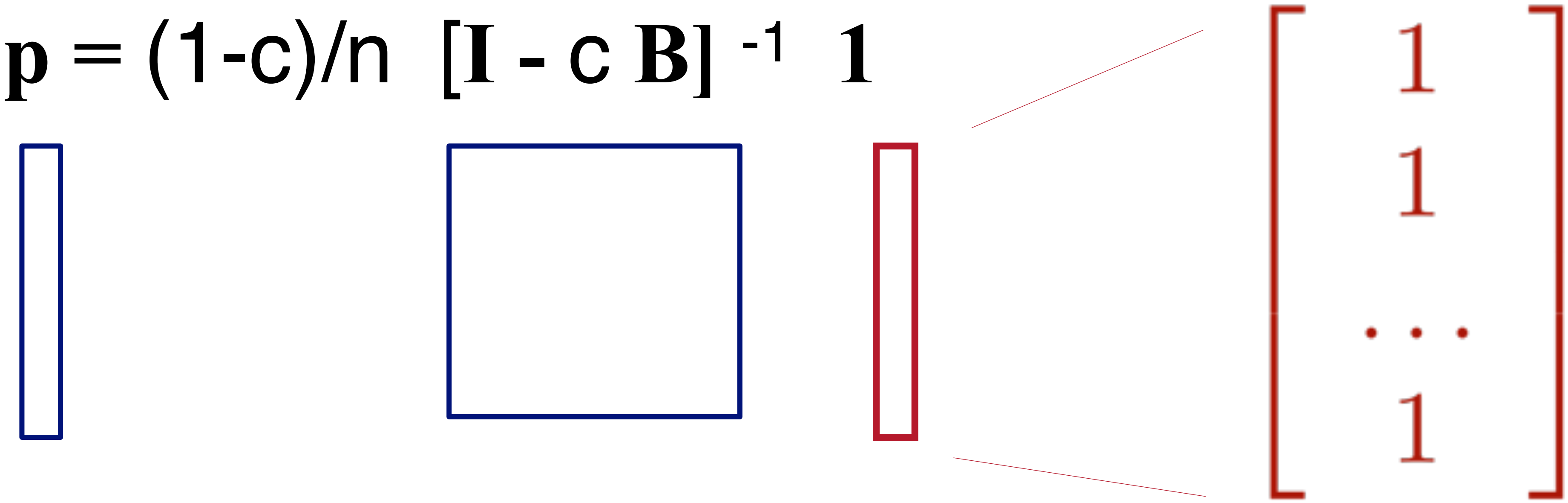
Full Algorithm

- With probability *1-c*, fly-out to a random node
- Then, we have



$\mathbf{p} = c\ \mathbf{B}\ \mathbf{p} + (1\text{-}c)/n\ \mathbf{1} \Rightarrow$

$\mathbf{p} = (1\text{-}c)/n\ \ [\mathbf{I} - c\ \mathbf{B}]^{-1}\ \ \mathbf{1}$



Alternative notation – eigenvector viewpoint

$\mathbf{M}$ Modified transition matrix

$$\mathbf{M} = c\,\mathbf{B} + (1-c)/n \;\; \mathbf{1} \;\; \mathbf{1}^T$$

Then

$$\mathbf{p} = \mathbf{M}\,\mathbf{p}$$

That is: the steady state probabilities =

PageRank scores form the *first eigenvector* of the 'modified transition matrix'

PageRank for graphs (generally)

You can compute PageRank for **any graphs**

Should be in your algorithm “toolbox”

- Better than simple centrality measure (e.g., degree)
- Fast to compute for large graphs (O(E))

But can be “misled” (Google Bomb)

- How?

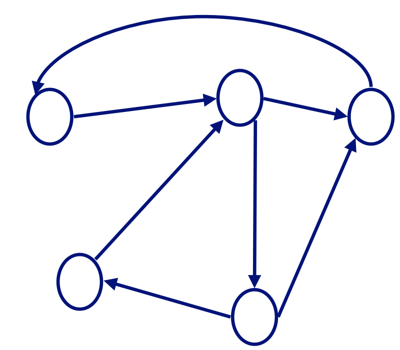
Personalized PageRank

Make one small variation of PageRank

- Intuition: not all pages are equal, some more relevant to a person’s specific needs
- How?

“Personalizing” PageRank

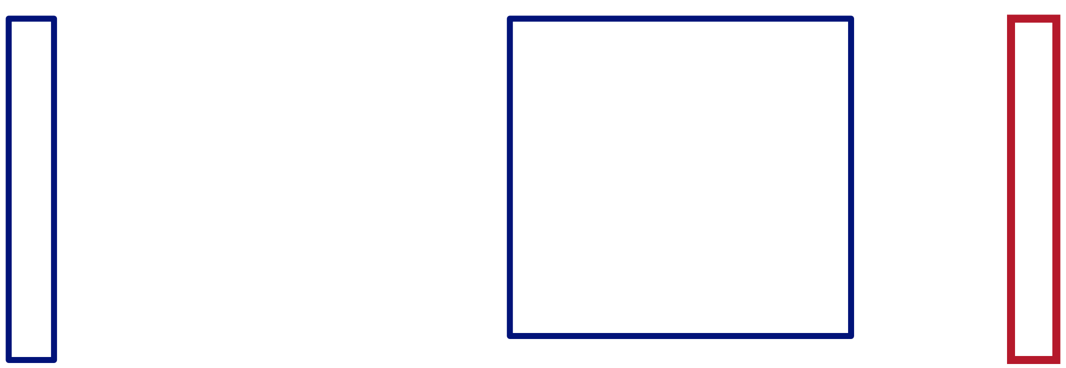
- With probability $1\text{-}c$, fly-out to ~~a random node~~ **some preferred nodes**



- Then, we have

$$\mathbf{p} = c\,\mathbf{B}\,\mathbf{p} + (1\text{-}c)/n\ \mathbf{1} \Rightarrow$$

$$\mathbf{p} = (1\text{-}c)/n\ [\mathbf{I} - c\,\mathbf{B}]^{-1}\ \mathbf{1}$$



$$\begin{bmatrix} 1 \\ 1 \\ \cdots \\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 \\ 1 \\ 0 \\ \cdots \\ 0 \\ 1 \end{bmatrix}$$

Why learn about Personalized PageRank?

Can be used for recommendation, e.g.,

- If I like this webpage, what would I also be interested?
- If I like this product, what other products I also like? (in a user-product bipartite graph)

Again, very flexible. Can be run on any graph

Will see some interactive tools in next lecture that uses Personalized PageRank