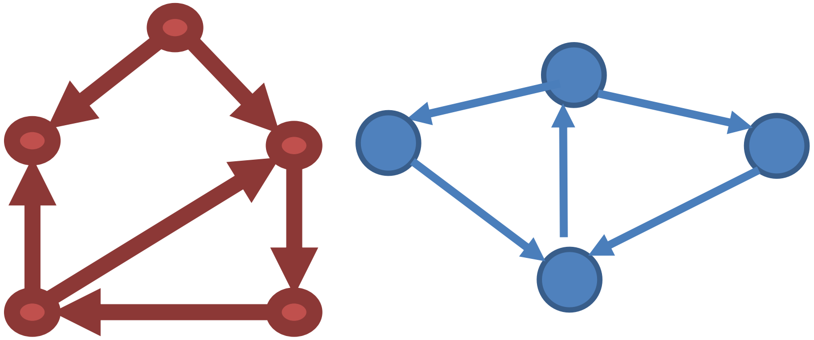
Convergence of Random Walk

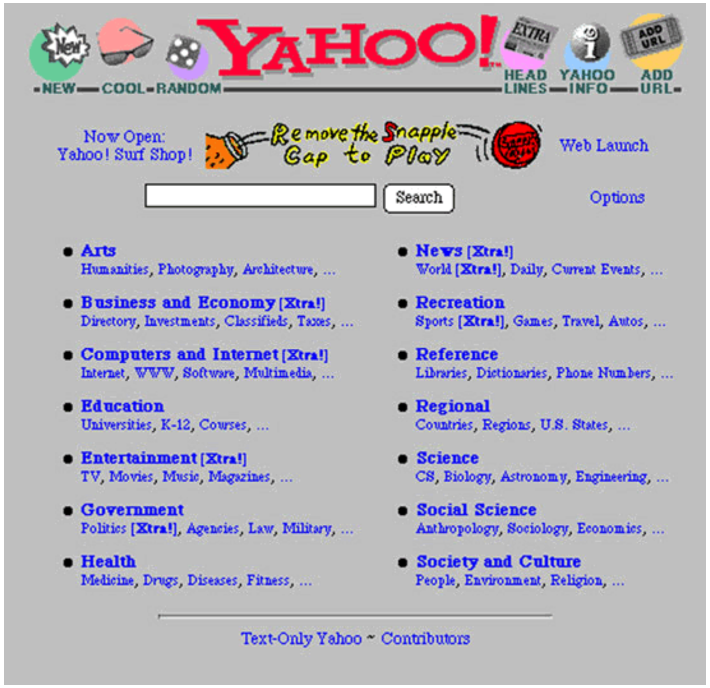
- For every ***connected non-bipartite*** undirected graph G, the distribution Pi converges to a limit and stationary distribution **π**.
- Moreover, if G is regular then this distribution is the uniform distribution on V.
- Intuition:
  - Why connected?
  - Why non-bipartite?



- What about Directed graphs?
  - Has to be ***strongly connected***! (otherwise the "walks will leak").
  - Has to be ***aperiodic,*** i.e., visits to some state (node) S should never be a multiple of k (k>1)
    - if the greatest common divisor of the lengths of its cycles is one

Relation to the web Search?

- **1st generation (Directories):**
  - Manual curation of **web directories** (e.g., Yahoo)
  - Web was growing too quickly to catch up.
- **2nd generation (Information Retrieval)**
  - Altavista
  - Classical Invormation retrieval, processing text in the pages
  - Term Spam
- **3rd generation (Google)**
  - Google page rank
    - Very hard to fake in-links.

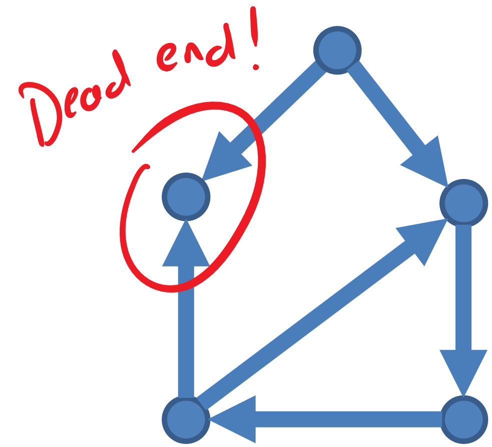


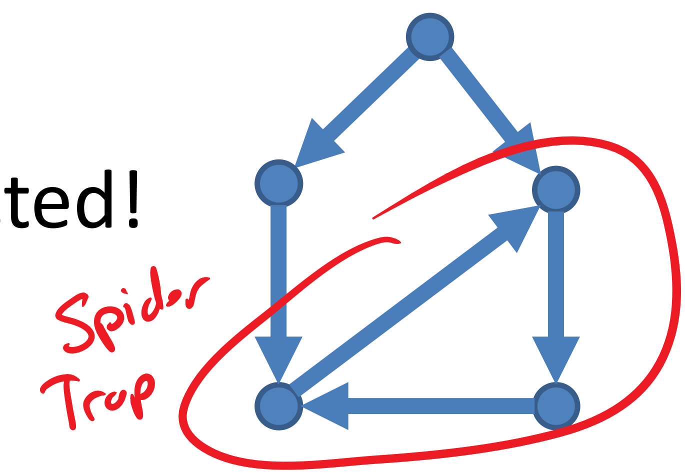
PageRank ”Voting” formulation

- Each page has a budget of ”votes” and distributes them evenly to all the outgoing links
  - E.g., if page j has r_j budget of votes, and n out-links, each link gets r_j/n votes.
  - INSIGHT: **A vote from an important page is worth more.**
- Node’s j own importance is the **sum of the votes on its in-links**.
- Did we see this before??
  - Notice the similarity with the **random walk**

Google Page Rank

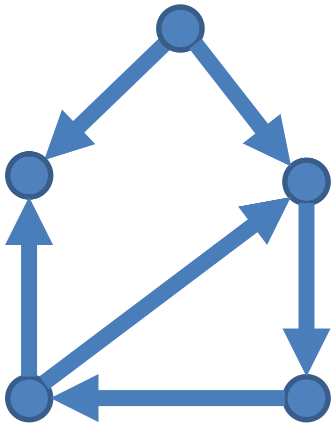
- Google page rank: **principal eigen vector**!
  - How to compute?
    - Power iteration.
  - Any issues?
    - Undirectional vs directional graph?
      - **Dead ends**
        - » Nodes with no out-degree
        - » votes leak out
      - **Spider traps**
- WWW is not strongly connected!

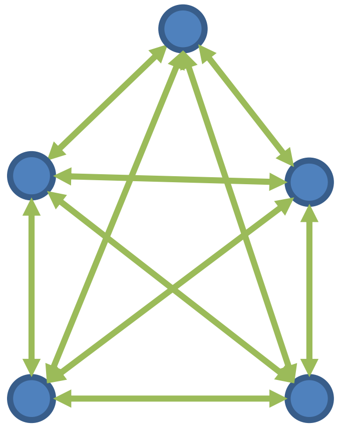




How do we fix PageRank?

- Ideas?
  - Make the graph **strongly connected** and **aperiodic**
  - Google solution
    - Make "**tiny tiny**" links from each node to every other node
    - Keep **the core** of the initial graph

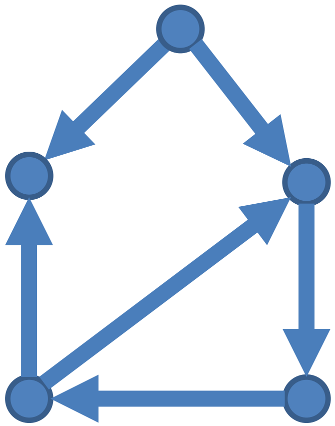


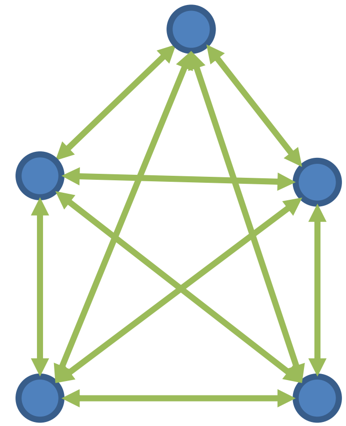


How do we fix PageRank? (cont.)

- Google random walker will
  - With **prob β** follow ”the real” link at random.
  - With **prob 1- β** jump to some random page.
  - Usually β is in the range of 0,8 to 0,9
    - I.e.,Radom walker will ”teleport” from any spider trap after 5-10 steps
  - For **”dead ends” 1-β=1**, i.e., once in the dead end, random walker always teleports to a random node.

Radom Walk Matrix with teleportation





β

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

Transition (random walk) Matrix

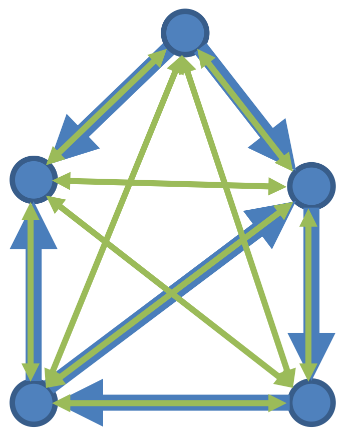
$+$

$(1-\beta)$

$$T=\begin{pmatrix} 0 & 1/4 & 1/4 & 1/4 & 1/4 \\ 1/4 & 0 & 1/4 & 1/4 & 1/4 \\ 1/4 & 1/4 & 0 & 1/4 & 1/4 \\ 1/4 & 1/4 & 1/4 & 0 & 1/4 \\ 1/4 & 1/4 & 1/4 & 1/4 & 0 \end{pmatrix}$$

Teleportation Matrix

Radom Walk Matrix with teleportation



$$M_{\text{pageRank}} = \begin{pmatrix} 0 & 0.45 & 0.45 & 0.05 & 0.05 \\ 0.25 & 0 & 0.25 & 0.25 & 0.25 \\ 0.05 & 0.05 & 0 & 0.05 & 0.85 \\ 0.05 & 0.45 & 0.45 & 0 & 0.05 \\ 0.05 & 0.05 & 0.05 & 0.85 & 0 \end{pmatrix}$$

$$M_{\text{pageRank}} = \beta \begin{pmatrix} 0 & 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} + \begin{pmatrix} 1-\beta & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1-\beta & 0 & 0 \\ 0 & 0 & 0 & 1-\beta & 0 \\ 0 & 0 & 0 & 0 & 1-\beta \end{pmatrix} \begin{pmatrix} 0 & 1/4 & 1/4 & 1/4 & 1/4 \\ 1/4 & 0 & 1/4 & 1/4 & 1/4 \\ 1/4 & 1/4 & 0 & 1/4 & 1/4 \\ 1/4 & 1/4 & 1/4 & 0 & 1/4 \\ 1/4 & 1/4 & 1/4 & 1/4 & 0 \end{pmatrix}$$

Is it fixed now?

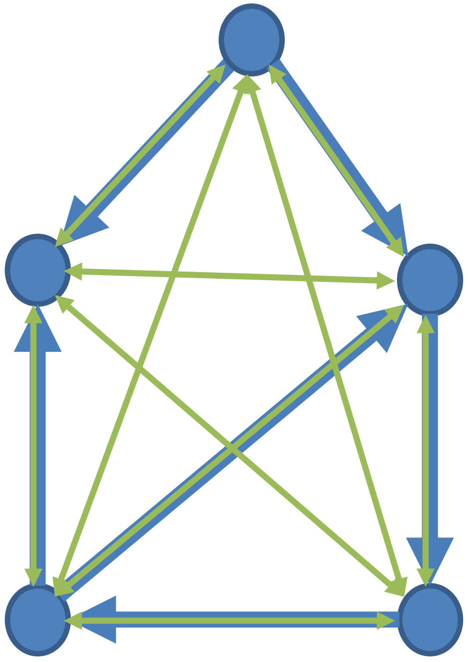
- What if **β=0** for all the nodes?
- What are the problems with teleportation?
  - How does $M_{pageRank}$ look for 10bn pages?
    - Dense random walk matrix!
    - N^2 non-zero elements! (instead of O(N*d))
  - Can you even store it in memory?
  - Insight for the Fix:
    - Interpret teleportation as **fixed tax** (always the same),
    - At every power iteration instead of computing rank vector $\mathbf{r}^{new}$

      $\mathbf{r}^{new}=\mathbf{r}^{old} M_{pageRank}$, we compute $\mathbf{r}^{new} = \beta\,(\mathbf{r}^{old}M)+c$,

      where $c = (1-\beta)/N$ (i.e., a tax)
    - Notice $M_{pageRank}$ is dense and M is sparse matrix!
    - If M contains dead-ends then r^{new} has to be renormalized so that it sums up to 1.

Topic Specific Page Rank

- Instead of teleporting to ”any node” - teleport to ”relevant pages” (teleport set) for topic-specific PageRank
- Avoiding LinkSpam:
  - teleport to trusted pages only (e.g., university homepages)



Graph Spectrum

- **$vA=\lambda v$**
- If A is a real symmetric matrix then it has n eigenvectors and associated n eigenvalues. All n eigenvalues are real $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$
  - If G is a d-regular graph then $\lambda_1=d$
- For random walk matrix M, i.e., normalized adjacency matrix $\lambda_1=1$ **($\pi M = \pi$)**
- We call $\lambda_1, \lambda_2, \ldots, \lambda_n$ the spectrum of graph G
- We call $\lambda_1-\lambda_2$ eigengap (or **spectral gap**)
  - $1-\lambda_2$ for normalized adjacency matrix
- So what if the graph is disconnected?
  - Think of convergence…
  - $\lambda_1=\lambda_2$

Convergence time on Expanders

- There is a connection between expansion of the graph and the spectral gap
  - Large gap (that is small λ_2) implies good expansion and vice versa.
- If G is a connected, d-regular, non-bipartite graph on n vertices, then $\lambda_2 < 1$ and G has mixing time $O(\frac{\log N}{1-\lambda_2})$
  - Practically mixing time is ~ O(logN)

Graph Spectrum (cont.)

- **For curiosity** (you can check Luca Trevisan lectures on Expanders in Youtube):

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

Adjacency Matrix

$$L = \begin{pmatrix} 2 & -1 & -1 & 0 & 0 \\ -1 & 2 & 0 & -1 & 0 \\ -1 & 0 & 3 & -1 & -1 \\ 0 & -1 & -1 & 3 & -1 \\ 0 & 0 & -1 & -1 & 2 \end{pmatrix}$$

Laplacian Matrix

  - Laplacian Matrix $\mathbf{L} := d \cdot \mathbf{I} - \mathbf{A}$
  - Normalized Laplacian matrix **L** = **I** – 1/d * **A** for d-regular graph
  - If $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$ eigenvalues of L then:
    - G has k connected components if $\lambda_k = 0$
    - G has a bipartite connected component if $\lambda_n = 2$

Spectral Graph Partitioning

- We want to assign the nodes of graph G into two groups of size n such that the edge cut is minimized.
  - Calculate the eigenvector **v**2 corresponding to the second smallest eigenvalue $\lambda 2$ of the graph Laplacian.
  - Sort the elements of the eigenvector in order from largest to smallest.
  - Put half of the vertices corresponding to the *n* largest elements in group 1, the rest in group 2

Spectral Graph Partitioning (cont)

- We want to assign the nodes of graph G into two groups of size n1 and n2 such that the edge cut is minimized.
  - Calculate the eigenvector **v**2 corresponding to the second smallest eigenvalue $\lambda 2$ of the graph Laplacian.
  - Sort the elements of the eigenvector in order from largest to smallest.
  - Put the vertices corresponding to the *n*1 largest elements in group 1, the rest in group 2, and calculate the cut size.
  - Then put the vertices corresponding to the *n*1 smallest elements in group 1, the rest in group 2, and recalculate the cut size.
  - Between these two divisions of the network, choose the one that gives the smaller cut size.
- More complicated for more than two groups

More tricks: Community Detection

Modularity matrix B (not sparse):

$$B_{ij} = A_{ij} - \frac{d_i d_j}{2m}$$

- We want to find two natural communities in a graph G.
  - Calculate eigenvector of the largest (most positive) eigenvalue of the modularity matrix B.
  - Assign vertices to communities according to the signs of the vector elements:
    - Positive signs in one

    group and negative into other
    - (bisection only)

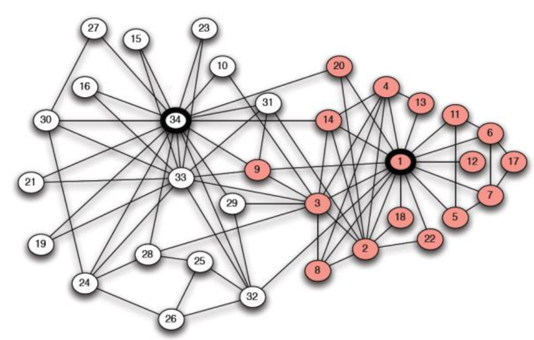


Figure 1.7: From the social network of friendships in the karate club from Figure 1.1, we can find clues to the latent schism that eventually split the group into two separate clubs (indicated by the two different shadings of individuals in the picture).

A bit of fun…

- Avg. FB degree is 200 (suppose).
- Q1
  - Take a random node. What’s its expected degree?
  - What’s an avg degree of it’s neighbors?
    - Or… why your friends are more popular than you are?
- Q2
  - Select an edge *e* uniformly at random. What is the expected degree of *e*’s end-nodes?

Why Your Friends Have More Friends than You Do? [1991]

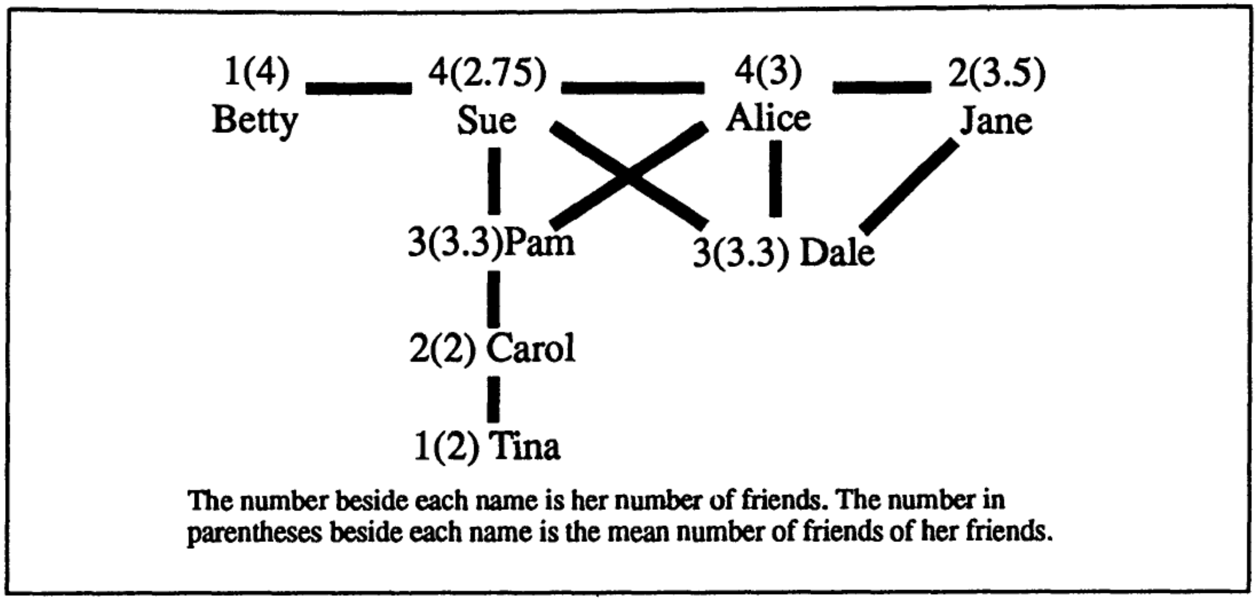


FIG. 1.—Friendships among eight girls at Marketville High School

- Avg degree 2.5

Intuition why

A SUMMARY OF THE NUMBERS OF FRIENDS AND THE MEAN NUMBERS OF FRIENDS OF FRIENDS FOR EACH OF THE GIRLS IN FIGURE 1

	Number of Friends (x_i)	Total Number of Friends of Her Friends (Σx_j)	Mean Number of Friends of Her Friends ($\Sigma x_j/x_i$)
Betty	1	4	4
Sue	4	11	2.75
Alice	4	12	3
Jane	2	7	3.5
Pam	3	10	3.3
Dale	3	10	3.3
Carol	2	4	2
Tina	1	2	2
Total	20	60	23.92
Mean	2.5*	3†	2.99*

* For eight girls.
† For 20 friends.

- **The mean of number of friends $\sum x_i/n$**
- **The mean number of friends among the friends is $(\sum x_i^2)/(\sum x_i)$**
  - **$(\sum x_i^2)/(\sum x_i)$= mean(x) + variance(x)/mean(x).**
  - **the mean among friends is always at least as great as the mean among individuals**
  - **the mean among friends increases with the variance among individuals (think power-law)**