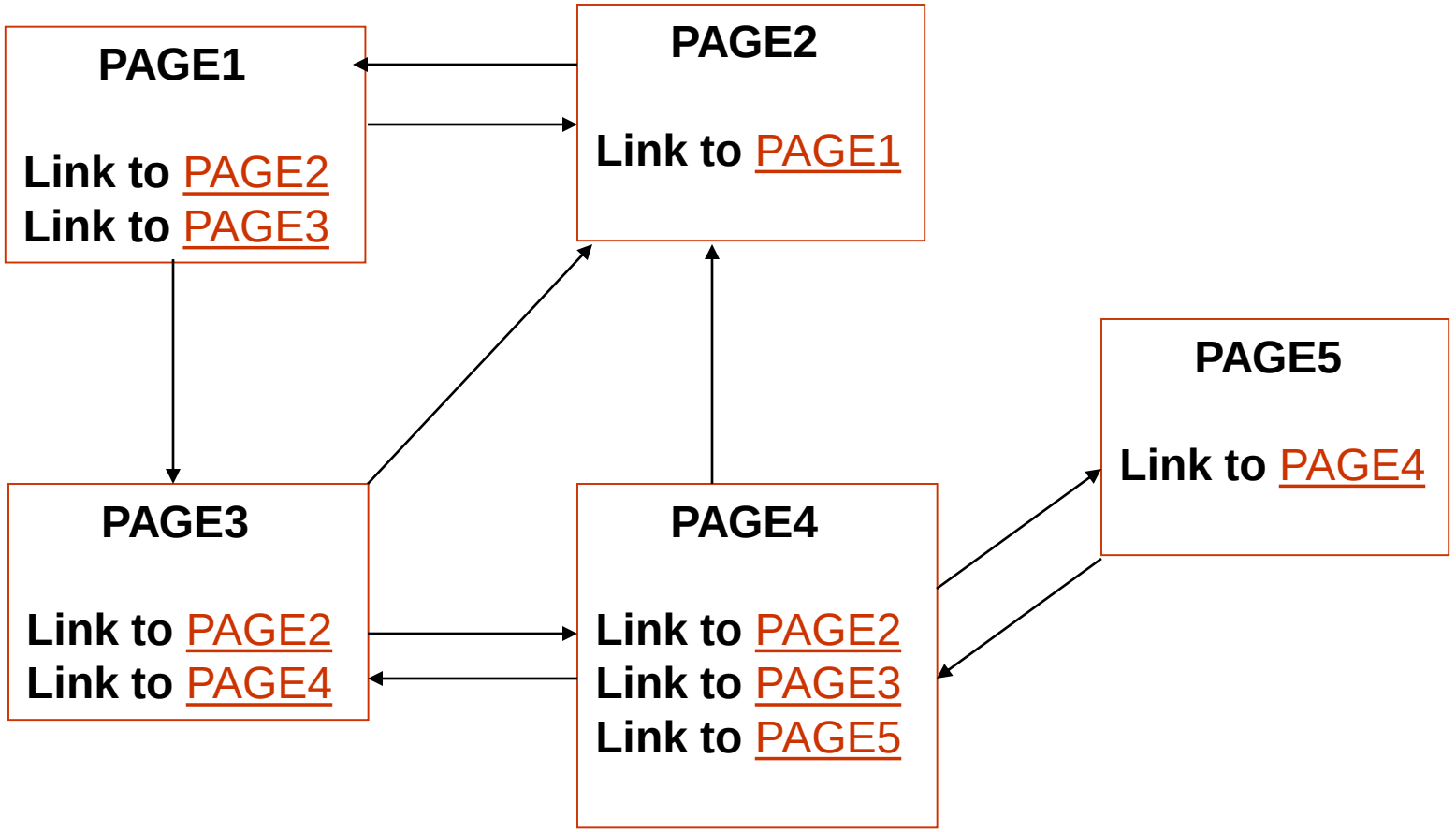
Matrix times Vector : Google PageRank

Ranking the importance of a webpage

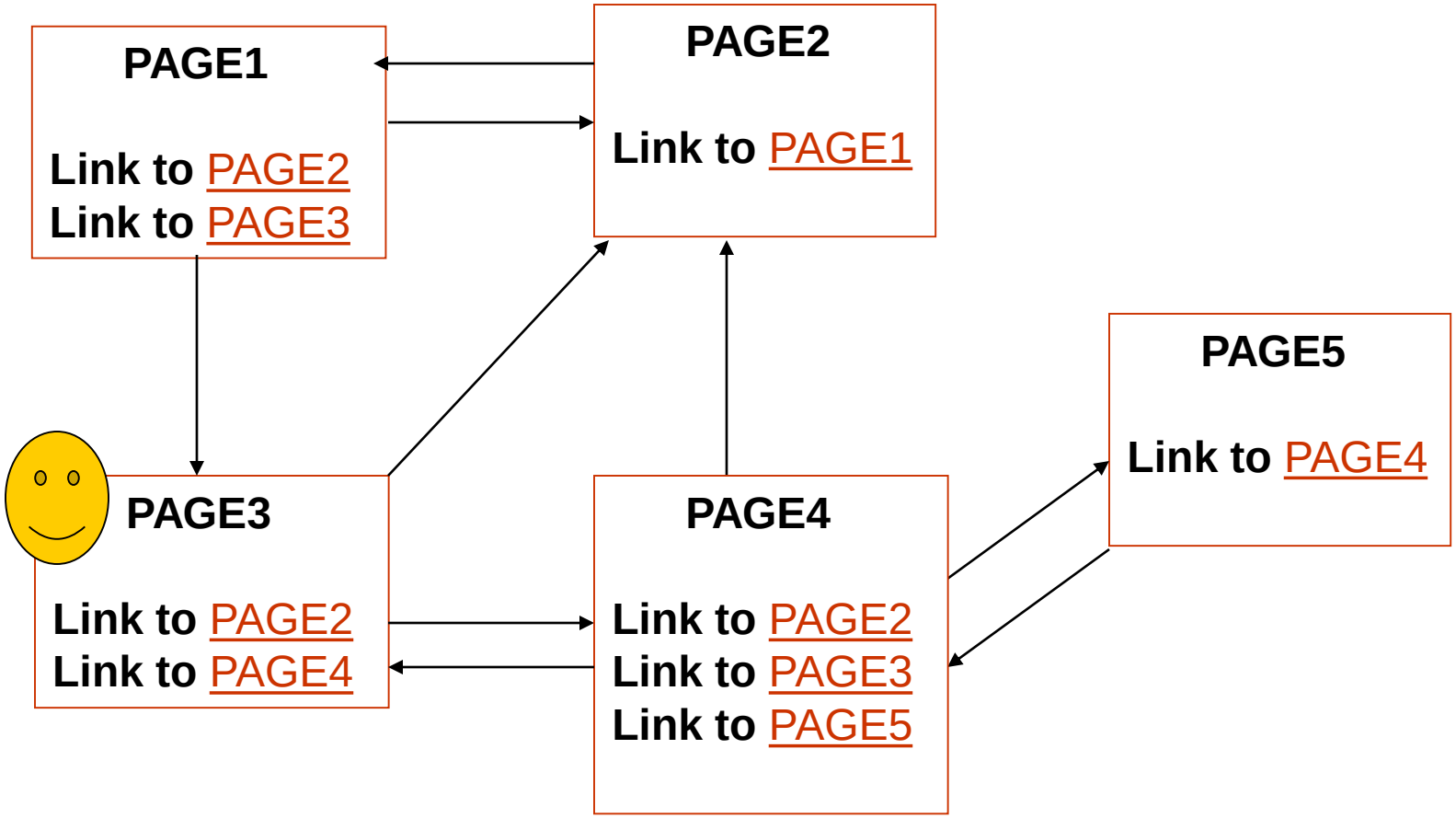
- The priority of a Google search result will depend on many conditions such as relevance.
- Another condition is the pagerank, a measure of how important or authoritative a web page is.
- Invented by Larry Page and Sergey Brin while they were graduate students at Stanford.
- These slides gives some idea what's going on, but see references for actual details of what pagerank really does.

A small internet

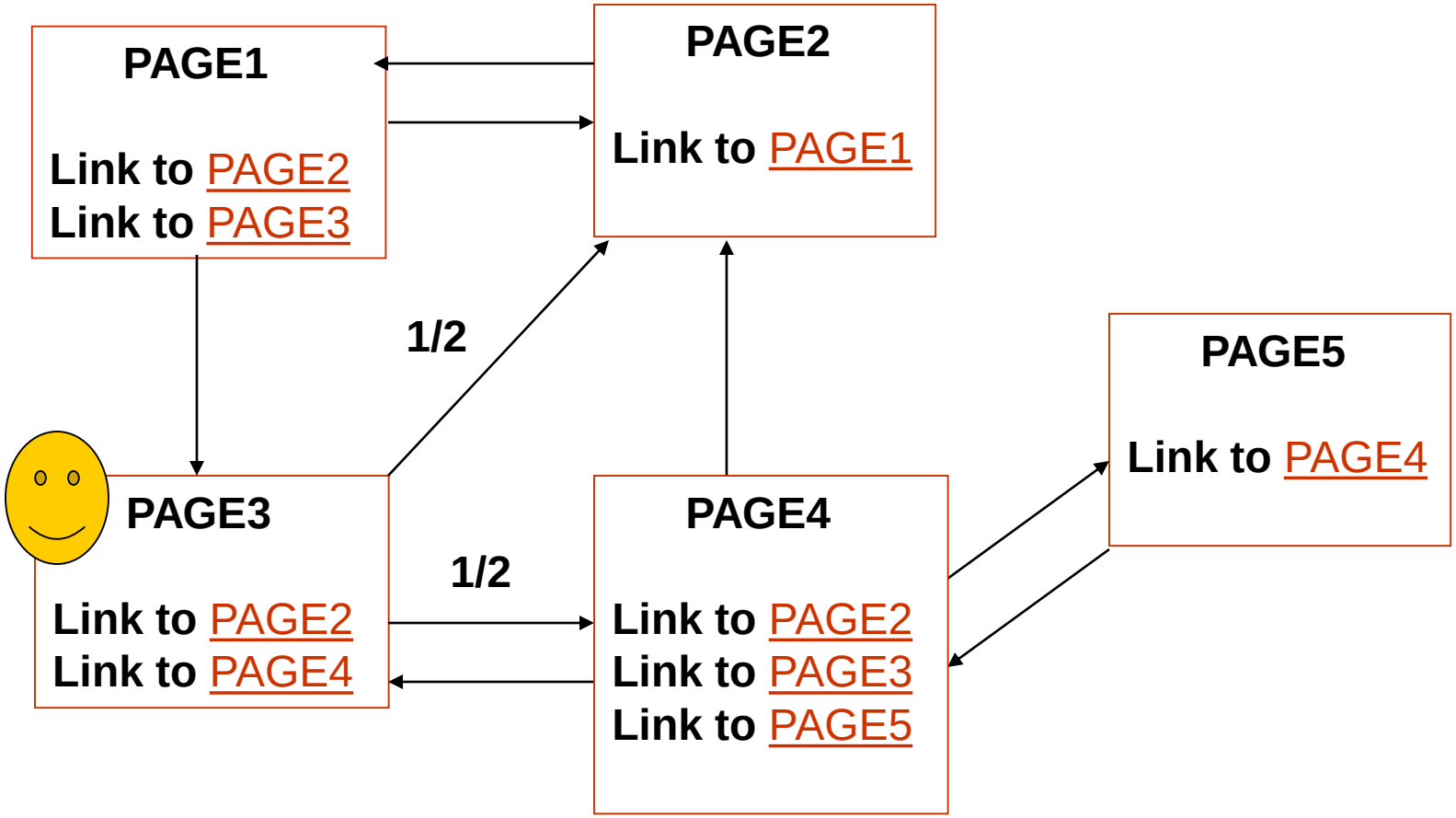
A small internet



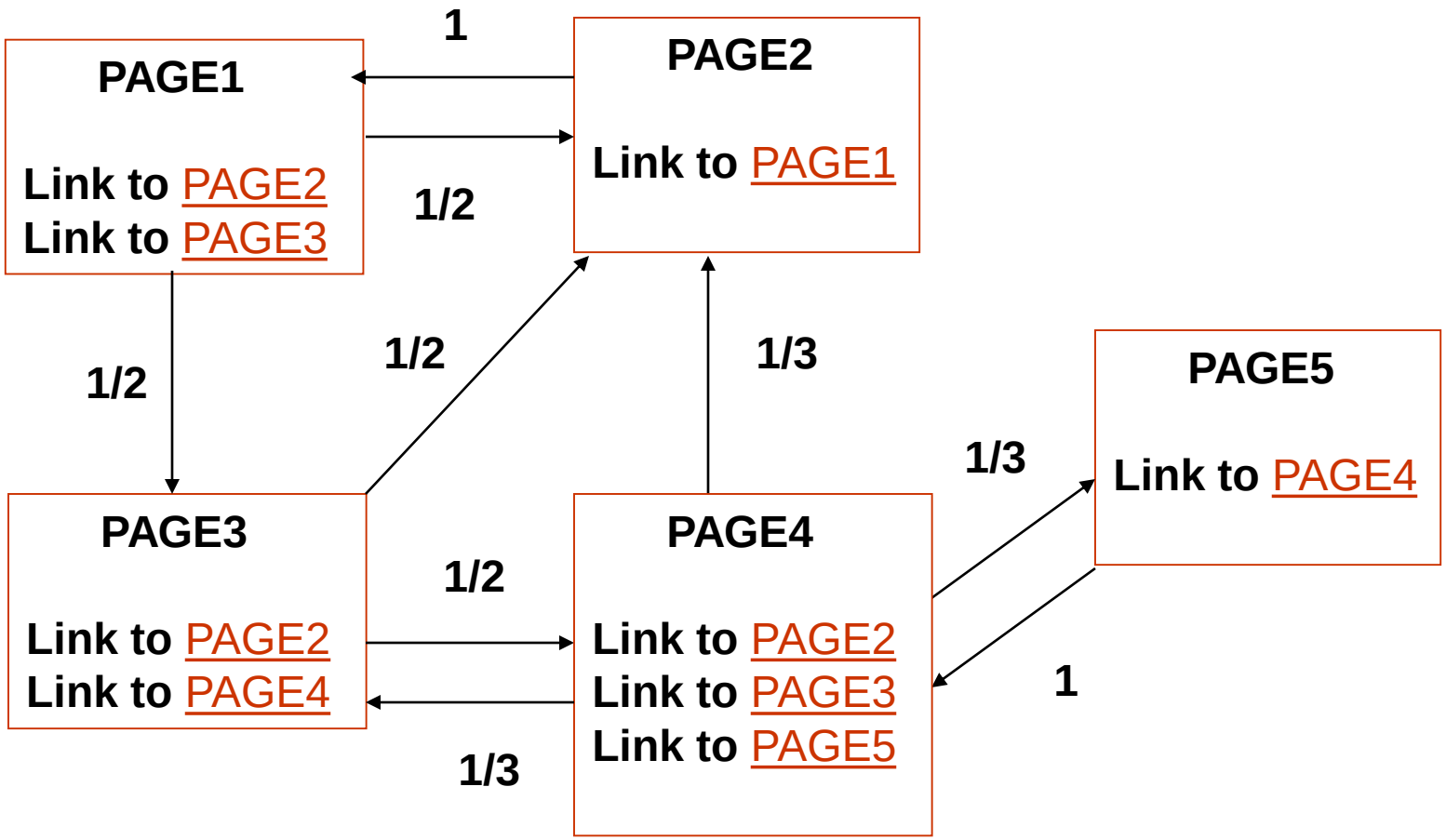
A random web surfer is on page 3



Picks a link at random: 50% page 2, 50% page 4



A directed graph



Graph information in matrix form

$$H = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 1/3 & 0 \\ 1/2 & 0 & 0 & 1/3 & 0 \\ 0 & 0 & 1/2 & 0 & 1 \\ 0 & 0 & 0 & 1/3 & 0 \end{pmatrix}$$

Using the matrix to surf

- Represent the surfer's position by a probability vector. For example start on page1.

$$x = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

- Multiply by H to get new probabilities.

$$Hx = \begin{pmatrix} 0 \\ .5 \\ .5 \\ 0 \\ 0 \end{pmatrix}$$

- Keep going.

Where do we end up?

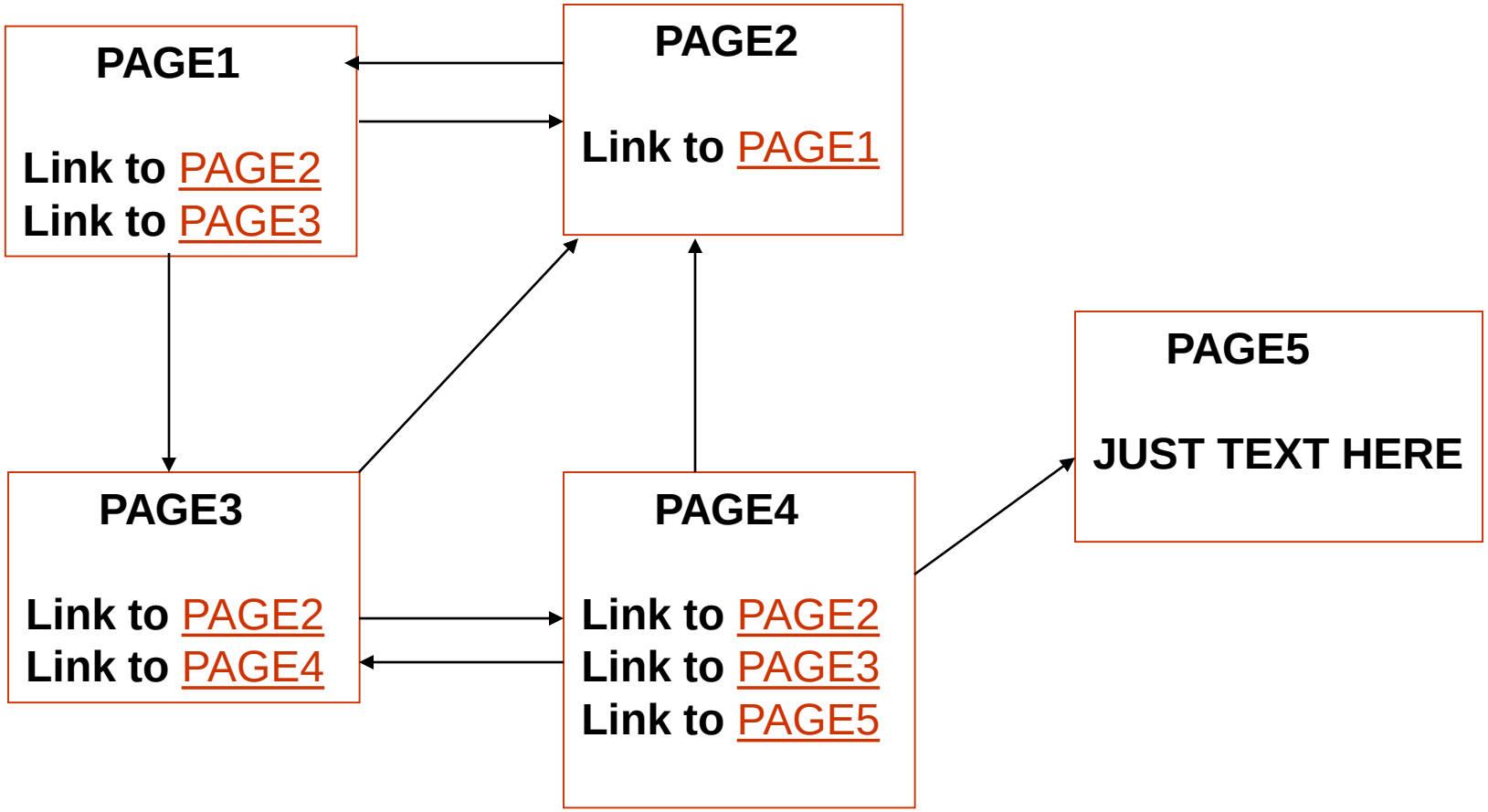
- The final value of x[i] is page i's pagerank.
- Math explanation: it's the eigenvector of H with eigenvalue 1 with entries scaled to sum to one.
- We are doing the power method to compute the largest eigenvalue.

$$x = \begin{pmatrix} .30 \\ .30 \\ .20 \\ .15 \\ .05 \end{pmatrix}$$

Questions?

- Does the result depend on the starting vector?
- Does this process always converge?
- References:
  - Sergey Brin and Lawrence Page, The anatomy of a large-scale hypertextual Websearch engine, Computer Networks and ISDN Systems 33 (1998), 107–117.
  - Amy N. Langville and Carl D. Meyer, Google's PageRank and beyond, Princeton University Press, 2006.
  - ***Google's PageRank***: The ***Math*** Behind the Search Engine. Rebecca S. Wills. Department of ***Mathematics***. North Carolina State University.
  - **How Google Finds Your Needle in the Web's Haystack,** AMS, David Austin, Grand Valley State University

A dangling node:



Dangling node fix

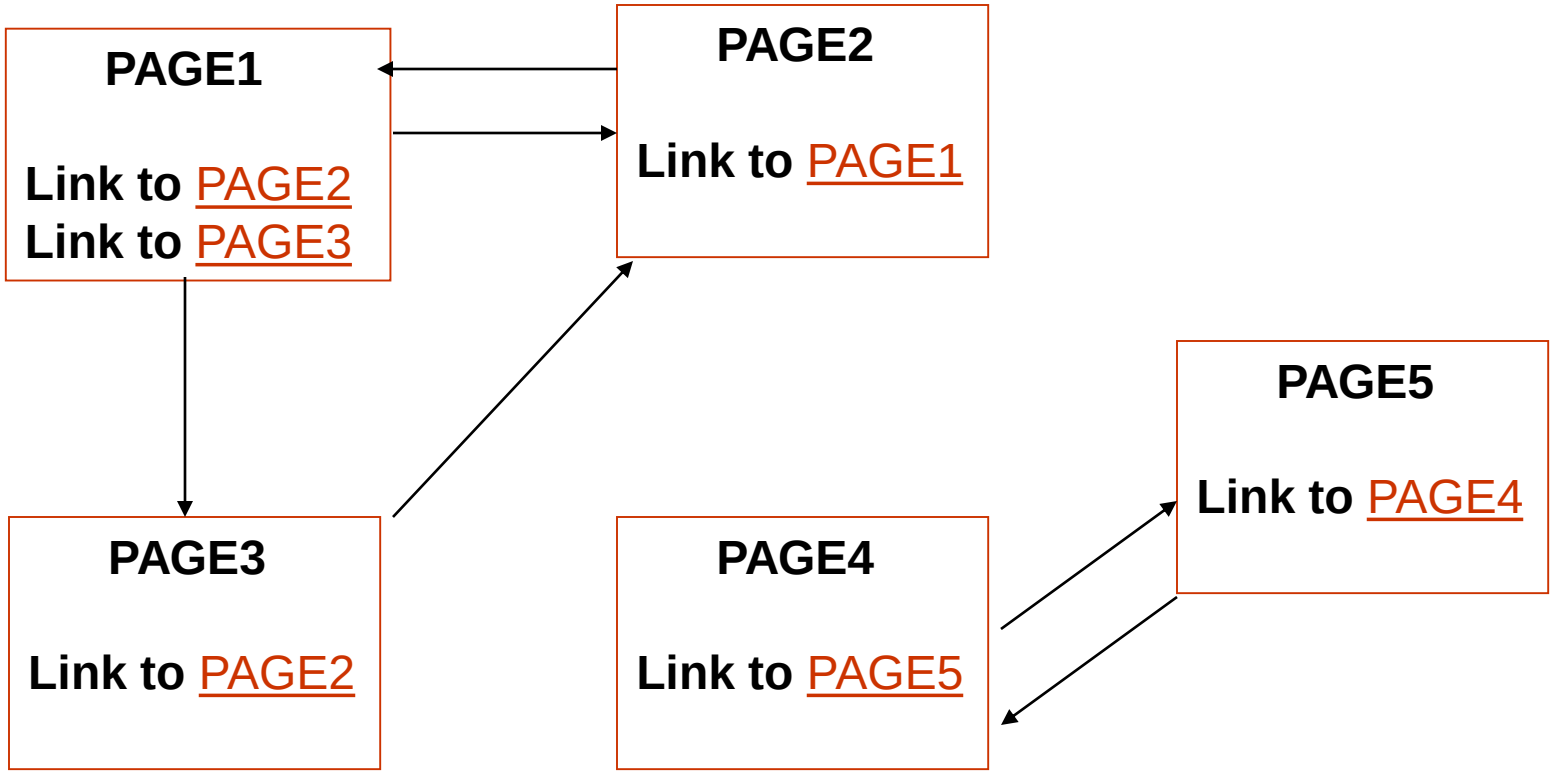
- The random surfer, on arriving at page with no outgoing links, chooses *any* internet page at random.

$$S = H + Dnode = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 1/3 & 0 \\ 1/2 & 0 & 0 & 1/3 & 0 \\ 0 & 0 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 1/3 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 & 1/5 \\ 0 & 0 & 0 & 0 & 1/5 \\ 0 & 0 & 0 & 0 & 1/5 \\ 0 & 0 & 0 & 0 & 1/5 \\ 0 & 0 & 0 & 0 & 1/5 \end{pmatrix}$$

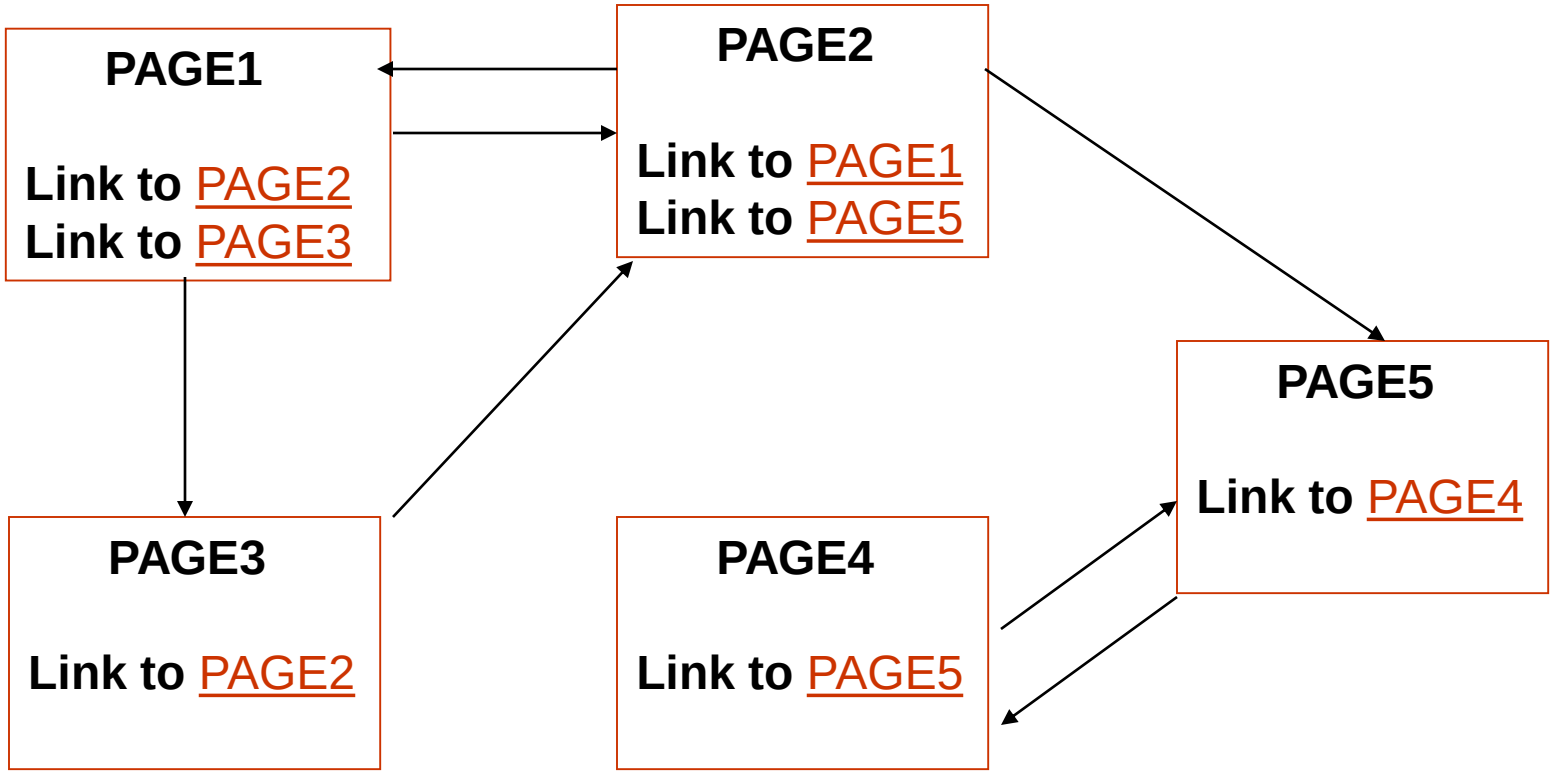
- Note multiplication of vector by second matrix can be easily computed: rank one update.

$$y = (Dnode)x \Leftrightarrow y_i = (1/5)x_5$$

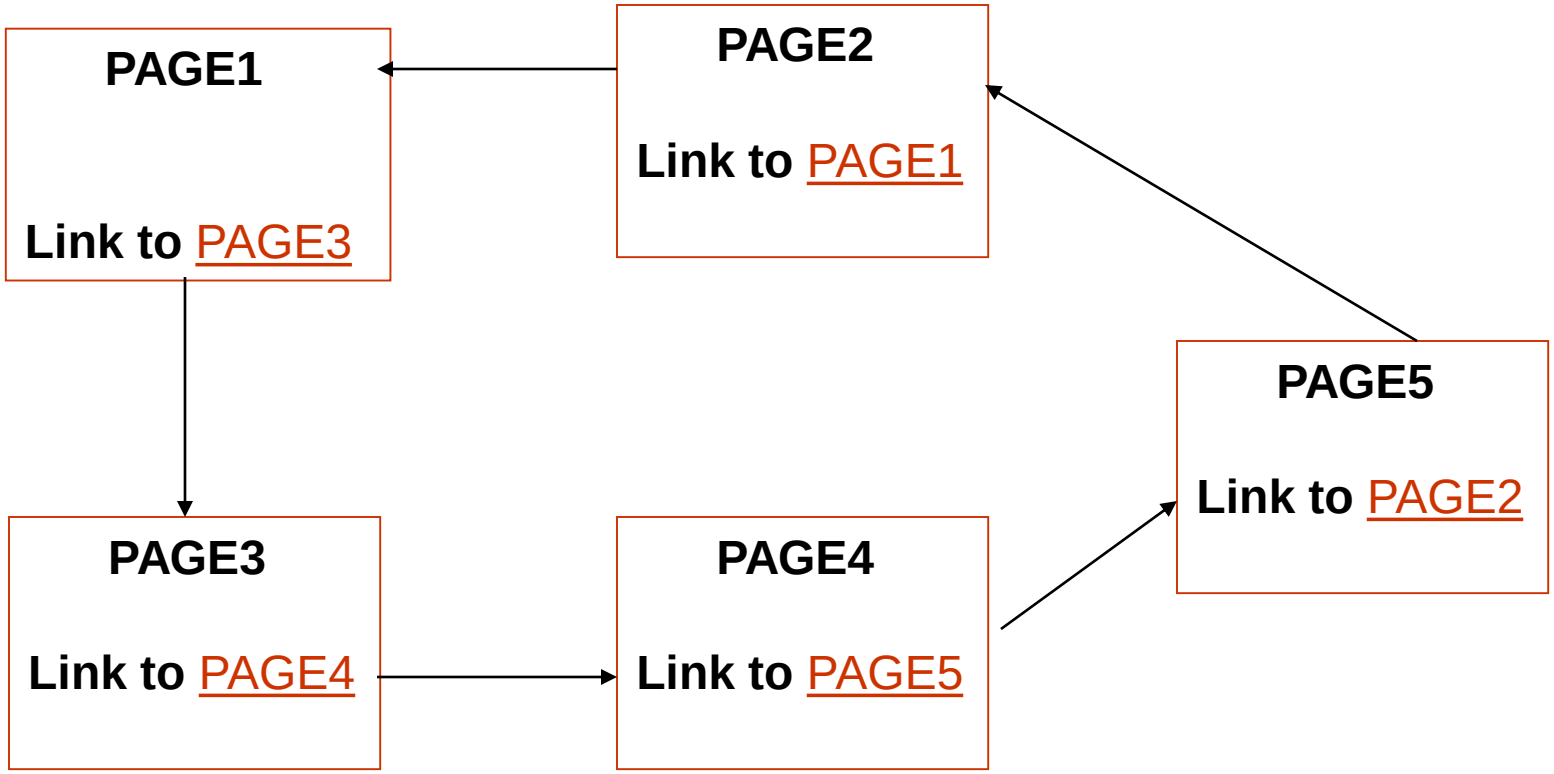
Disconnected internet



Combining the last 2



What about this?



Add a damping factor

- With some probability q, the surfer doesn't follow a link but picks *any* page at random.
- We model this by changing matrix

$$G = (1-q)S + qB$$

$$B = \begin{pmatrix} 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \end{pmatrix}$$

Each entry in B is 1/n

Add a damping factor

$$B = \begin{pmatrix} 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \end{pmatrix}$$

$$G = (1-q)S + qB$$

- Note again multiplication of vector by second matrix can be easily computed: rank one update.

$$y = qBx \Leftrightarrow y_i = q(1/5)\sum_i x_i = q/5$$

Matrix times Vector: Implementations

MatVec Problem: Compute product of matrix (***m x n***) and vector (***n x 1***)

- Compute

$$y = \begin{pmatrix} y_0 \\ y_1 \\ \cdot \\ \cdot \\ y_{m-1} \end{pmatrix} = Ax$$

- Given

$$A = \begin{pmatrix} a_{0,0} & a_{0,1} & \cdot & \cdot & a_{0,n-1} \\ a_{1,0} & a_{1,1} & & & \\ \cdot & & \cdot & & \\ \cdot & & & \cdot & \\ a_{m-1,0} & & & & a_{m-1,n-1} \end{pmatrix} \qquad x = \begin{pmatrix} x_0 \\ x_1 \\ \\ \\ x_{n-1} \end{pmatrix}$$

MatVec: Serial Code (2D array)

```
for (i=0; i < m; i++)
{
        y[i] = 0.0
        for(j=0; j < n; j++)
                y[i] += A[i][j] * x[j];
}
```

MatVec: Serial Code (1D array)

```
for (i=0; i < m; i++)
{
        y[i] = 0.0
        for(j=0; j < n; j++)
                y[i] += A[i*n+j] * x[j];
}
```

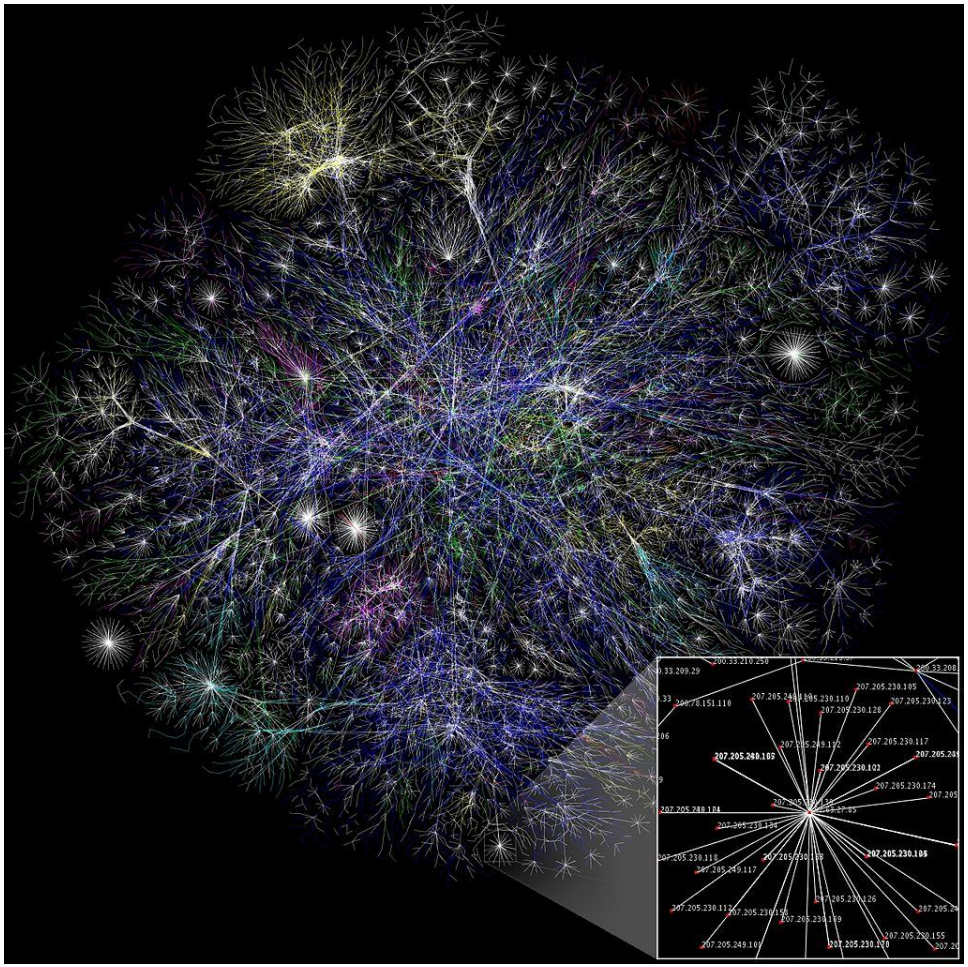
$$A = \left(a_{0,0} \quad a_{0,1} \quad \ldots \quad a_{0,n-1} \quad a_{1,0} \quad a_{1,1} \quad \ldots \quad a_{1,n-1} \quad \ldots \quad \ldots \quad a_{m-2,n-1} \quad a_{m-1,0} \quad a_{m-1,1} \quad \ldots \quad a_{m-1,n-1} \right)$$

Sparse MatVec

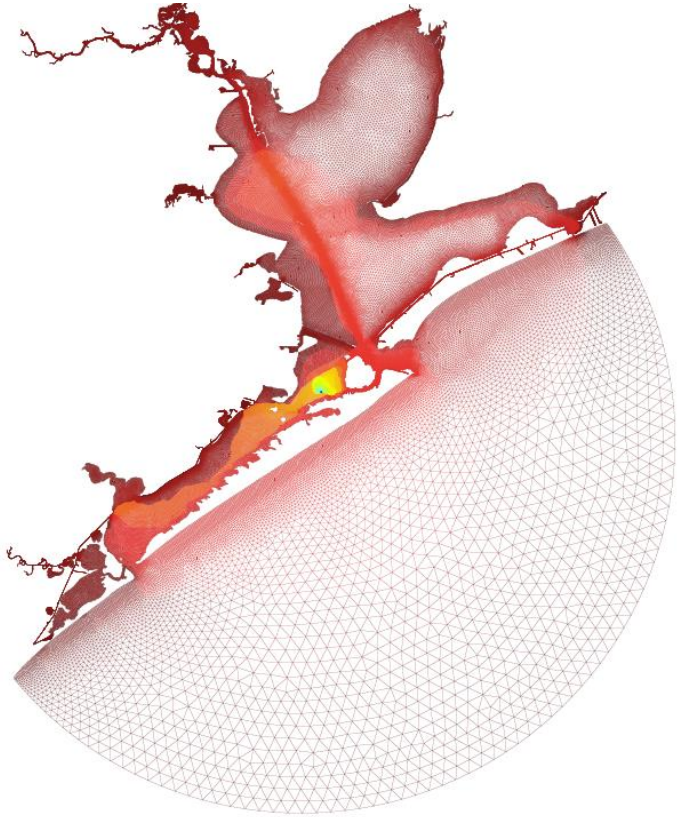
- A sparse matrix has many fewer non-zero elements than m*n
- The MatVec on the previous slide is a appropriate for dense matrices
- For sparse matrix we would like to avoid
  - Unnecessary multiplications by zero elements
  - Unnecessary storage of zero elements

Many applications give rise to sparse linear systems

https://en.wikipedia.org/wiki/Scale-free_network



SUNTANS grid



May have
$n=10^7$

Nonzeros=6n
Dense Storage=n^2

http://www.stanford.edu/~fringer/research.html

Coordinate (COO) List Form Storage

- Store only the non-zeros and their row and column indices.
- MATLAB

Compressed Sparse Row (CSR) Storage

- One array **A** stores the non zero elements in the same order as the previous 1D array option. Its size is the number of nonzeros **nnz**.
- One integer array **iA** stores the starting index for each row in the array **A**. Its size is the number of rows plus one **m+1**. Last entry is **nnz**.
- One integer array **jA** stores the column index for each entry in **A**. Its size is the number of nonzeros **nnz**.

CSR example

$$H = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 1/3 & 0 \\ 1/2 & 0 & 0 & 1/3 & 0 \\ 0 & 0 & 1/2 & 0 & 1 \\ 0 & 0 & 0 & 1/3 & 0 \end{pmatrix}$$

$$A = \begin{bmatrix} 1.0 & 0.5 & 0.5 & .33 & 0.5 & .33 & 0.5 & 1.0 & .33 \end{bmatrix}$$

$$iA = [0 \quad 1 \quad 4 \quad 6 \quad 8 \quad 9]$$

$$jA = \begin{bmatrix} 1 & 0 & 2 & 3 & 0 & 3 & 2 & 4 & 3 \end{bmatrix}$$

CSR example

$$H = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 1/3 & 0 \\ 1/2 & 0 & 0 & 1/3 & 0 \\ 0 & 0 & 1/2 & 0 & 1 \\ 0 & 0 & 0 & 1/3 & 0 \end{pmatrix}$$

Row 1
Stored between at indices
iA[1]=1
to
iA[2]-1=3

$$A = [1.0 \quad 0.5 \quad 0.5 \quad .33 \quad 0.5 \quad .33 \quad 0.5 \quad 1.0 \quad .33]$$

$$iA = [0 \quad 1 \quad 4 \quad 6 \quad 8 \quad 9]$$

$$jA = [1 \quad 0 \quad 2 \quad 3 \quad 0 \quad 3 \quad 2 \quad 4 \quad 3]$$

CSR example

$$H = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 1/3 & 0 \\ 1/2 & 0 & 0 & 1/3 & 0 \\ 0 & 0 & 1/2 & 0 & 1 \\ 0 & 0 & 0 & 1/3 & 0 \end{pmatrix}$$

Row 2 Column 0

Access entry: A[iA[2]]=A[4]=.5

$$A = [1.0 \quad 0.5 \quad 0.5 \quad .33 \quad 0.5 \quad .33 \quad 0.5 \quad 1.0 \quad .33]$$

$$iA = [0 \quad 1 \quad 4 \quad 6 \quad 8 \quad 9]$$

$$jA = [1 \quad 0 \quad 2 \quad 3 \quad 0 \quad 3 \quad 2 \quad 4 \quad 3]$$

Access column number: jA[iA[2]]=jA[4]=0

CSR example: More at https://en.wikipedia.org/wiki/Sparse_matrix

$$H = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 1/3 & 0 \\ 1/2 & 0 & 0 & 1/3 & 0 \\ 0 & 0 & 1/2 & 0 & 1 \\ 0 & 0 & 0 & 1/3 & 0 \end{pmatrix}$$

Row 2 Column 0

Access entry: A[iA[2]]=A[4]=.5

$$A = [1.0 \quad 0.5 \quad 0.5 \quad .33 \quad 0.5 \quad .33 \quad 0.5 \quad 1.0 \quad .33]$$

$$iA = [0 \quad 1 \quad 4 \quad 6 \quad 8 \quad 9]$$

$$jA = [1 \quad 0 \quad 2 \quad 3 \quad 0 \quad 3 \quad 2 \quad 4 \quad 3]$$

Access column number: jA[iA[2]]=jA[4]=0

MatVec: Serial Code (CSR)

```
for (i=0; i < m; i++)
{
        y[i] = 0.0
        for(k=iA(i); k < iA(i+1); k++)
                y[i] += A[k] * x[jA[k]];
}
```

Parallel Thinking Assignment

- Compute pagerank in parallel.
- Main operation y=Hx
- How to parallelize with dense H?
- How to parallelize with sparse H?
- Under what conditions would you expect possible difficulties with load balancing?