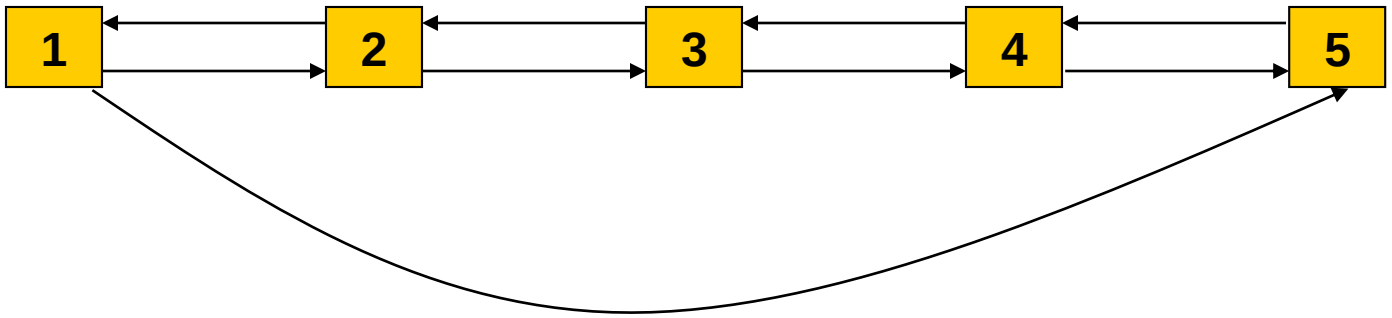
Assignment 3: PageRank
Matrix times Vector

Assignment #3 Page Rank

- Based on number of pages, NumPg
- Page(1) has a link to Page(2) and Page(NumPg)
- Page(NumPg) has a link to Page(NumPg-1)
- All others, Page(i) has a link to Page(i-1) and Page(i+1)

Internet with NumPg=5



Matrix before adding damping matrix NumPg=10

$$\begin{bmatrix}
0.0 & 0.5 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.5 & 0.0 & 0.5 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.5 & 0.0 & 0.5 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.5 & 0.0 & 0.5 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.5 & 0.0 & 0.5 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.5 & 0.0 & 0.5 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.5 & 0.0 & 0.5 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.5 & 0.0 & 0.5 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.5 & 0.0 & 1.0 \\
0.5 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.5 & 0.0
\end{bmatrix}$$

Assignment #3 Setup and Parameters

- Initial guess for pageranks x is 1/(NumPg), that is each page has equal rank.
- Damping factor q=0.15
- Perform Matvecs over and over (K=number of times the matvec is done=1000).
- For the first small problem, print final x vector. For large problem, do not print the final x, but do find the minimum and maximum final page rank values.

Assignment #3 A: OpenMP dense

- Write code to compute pagerank in shared memory.
- Parallelize the MatVec code in the slides "Week 5 Google and MatVec" using openMP. Slide [21] uses 2d arrays, slide [22] uses a 1d array. I would recommend the 1d option, but you may choose.
- You should build and store the entire G matrix (including the damping) as a dense matrix.
- Run code on blueshark. Use the *.sh files in the OpenMP codes folder on CANVAS as examples of your submission script.

Assignment #3 B: OpenMP sparse

- Write code to compute pagerank in shared memory.
- Parallelize the sparse MatVec code in the slides "Week 5 Google and MatVec" using openMP. See slide [32].
- You should build and store the sparse S matrix (without damping) as a CSR matrix.
- Implement the damping without extra storage as in slide [18].
- Run code on blueshark.

Assignment #3 C: MPI dense

- Write code to compute pagerank in parallel distributed memory.
- Use the MPI ideas to implement the matrix vector product as in the slides on "Week 5 MatVec".
  - Distribute the dense matrix by rows.
  - Do not store the entire matrix on each process. Use only the storage space required for the part of the matrix the process owns.
  - Each process builds the part of the matrix it owns
  - You may assume the number of pages is a multiple of the number of MPI processes.
- Run code on Blueshark

Problem 1: Small problem

- NumPg=16
- Run on one processor
- Print the final page rank vector and include in your write up
- For debugging, you can run on more threads/processes and verify that you compute the same pagerank.

Problem 2: Large Problem Speedup

- NumPg=1600
- Run on 1,2,4,8 threads (OMP A & B)
- Run on 1,2,4,8,16 processors (MPI C).
- In your write up report the run times and speedups.
- Include the final maximum and minimum values in the page rank vector in your write up.
- Discussion question 1: Do you see perfect speedup? If not, why do you think this is so, and how might you change the experiments to see greater speedup?
- Discussion question 2: How do the approaches (A,B,C) compare in terms of performance and programming effort?

Assignment #3

- Write up due Thursday Feb 28. Include code listings and the information requested on the previous 3 slides.
- Extra credit D: MPI sparse
  - Discuss and implement a sparse matrix vector multiply in MPI using the minimum communication required. Do not assume any particular sparsity structure for the S matrix.
  - Distribute the sparse matrix by rows.
  - Do not store the entire matrix on each process. Use only the storage space required for the part of the matrix the process owns.
  - You may assume the number of pages is a multiple of the number of MPI processes.