Modified PageRank Algorithm

Benjamin Wang (ID: 1179478)
Ojas Malwankar (ID: 1610494)

Motivation

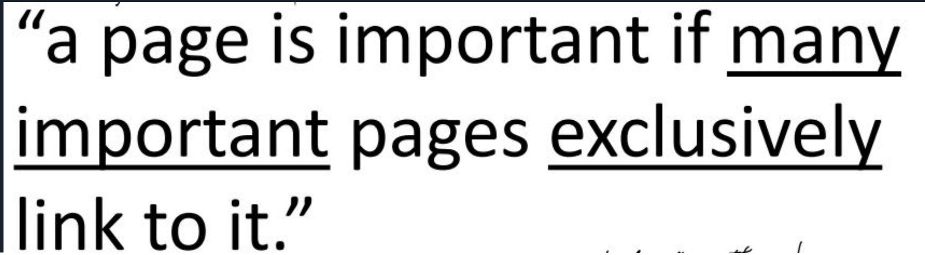
- PageRank is the famous equation on which Google, now known as Alphabet, is founded upon, developed by Google co-founders Larry Page and Sergey Brin.
- It’s known as the $25B eigenvector equation
- PageRank can be applied to any directed graph to rank importance

Objective

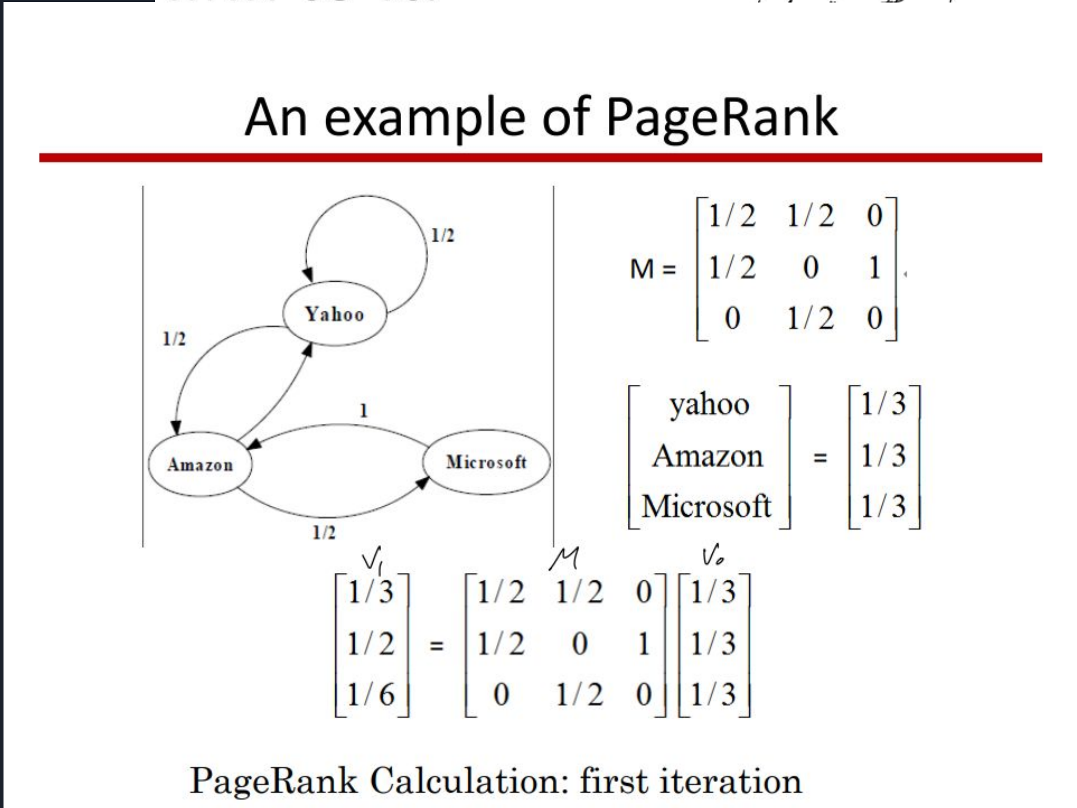
1. Read directed graphs into linear systems of PageRank equations
2. Solve the linear system of PageRank equations using Power Iteration algorithm
3. Converge the Power Iteration algorithm

Algorithm

1. PR(i) = sum(PR(j) / Nj), for j pointing to i
2. sum(PR(i)) == 1.0
3. λV = MV
4. V_(t + 1) = MV_t



“a page is important if many important pages exclusively link to it.”



An example of PageRank

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

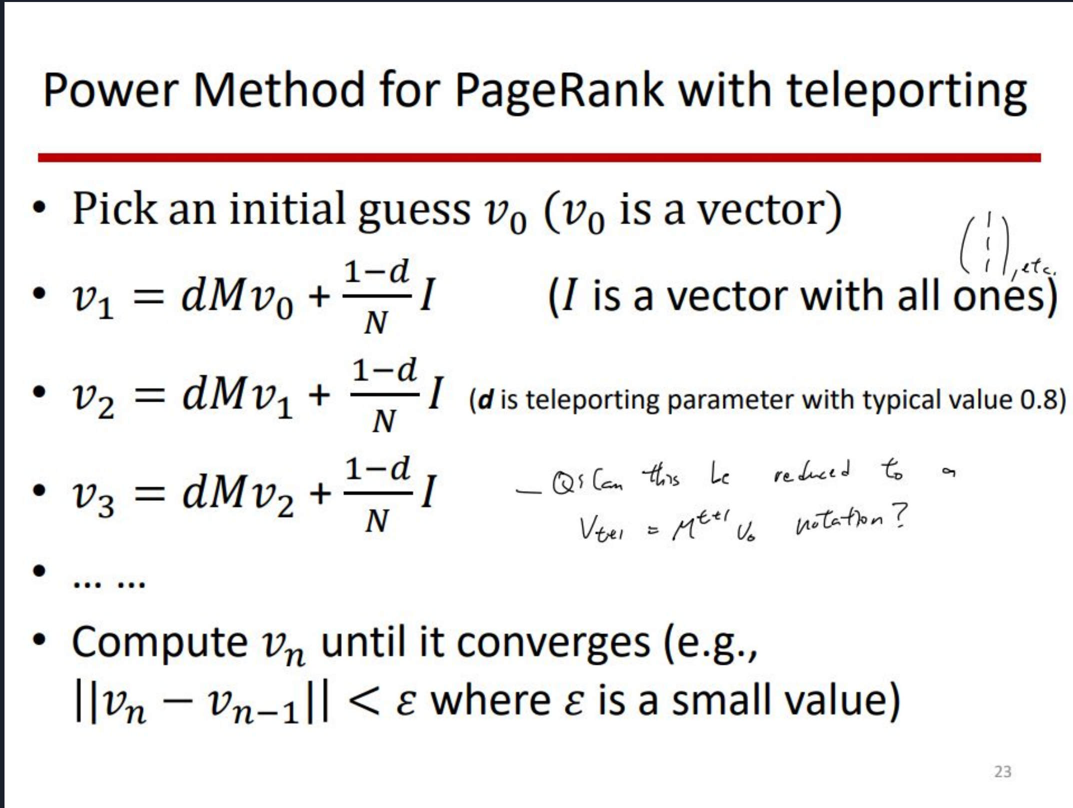
$$\begin{bmatrix} \text{yahoo} \\ \text{Amazon} \\ \text{Microsoft} \end{bmatrix} = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}$$

$$\overset{V_1}{\begin{bmatrix} 1/3 \\ 1/2 \\ 1/6 \end{bmatrix}} = \overset{M}{\begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix}} \overset{V_0}{\begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}}$$

PageRank Calculation: first iteration

Algorithm

1. V_(t + 1) = M^(t + 1) * V_0
2. PR(i) = d*sum(PR(j)/Nj) for j ->i + (1-d)/N
3. Vt = dMV_(t-1) + (1-d)/N*I,
4. || V_t - V_(t-1)|| < ε,



Power Method for PageRank with teleporting

- Pick an initial guess v_0 (v_0 is a vector)
- $v_1 = dMv_0 + \frac{1-d}{N} I$ (I is a vector with all ones) $\begin{pmatrix}1\\1\\1\end{pmatrix}$, etc.
- $v_2 = dMv_1 + \frac{1-d}{N} I$ (***d*** is teleporting parameter with typical value 0.8)
- $v_3 = dMv_2 + \frac{1-d}{N} I$ — Q: Can this be reduced to a $V_{t+1} = M^{t+1} V_0$ notation?
- … …
- Compute v_n until it converges (e.g., $||v_n - v_{n-1}|| < \varepsilon$ where ε is a small value)

Languages/Frameworks

- OpenMP
  - Baseline approach
- C++ Native Threads
  - Most implementations built upon pthreads
  - High level of abstraction

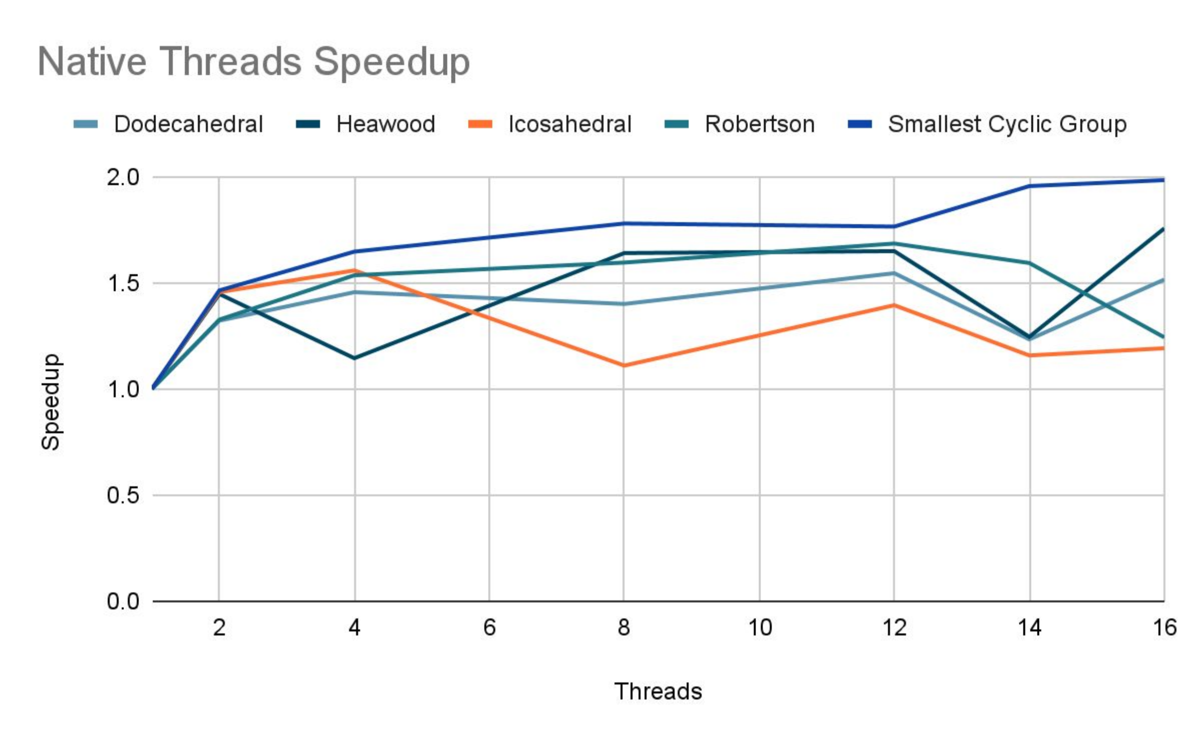
Parallelization

- Tiling of the Matrix-Vector Multiplication in the Power Iteration algorithm *MV* for matrix *M* and vector *V*.
  - Divide up the rows of the Matrix *M* by the number of available threads
  - Partition the Matrix up so each thread can work on its own portion of the matrix
- In OpenMP version, quick parallel for reduction for summing up squared vector elements

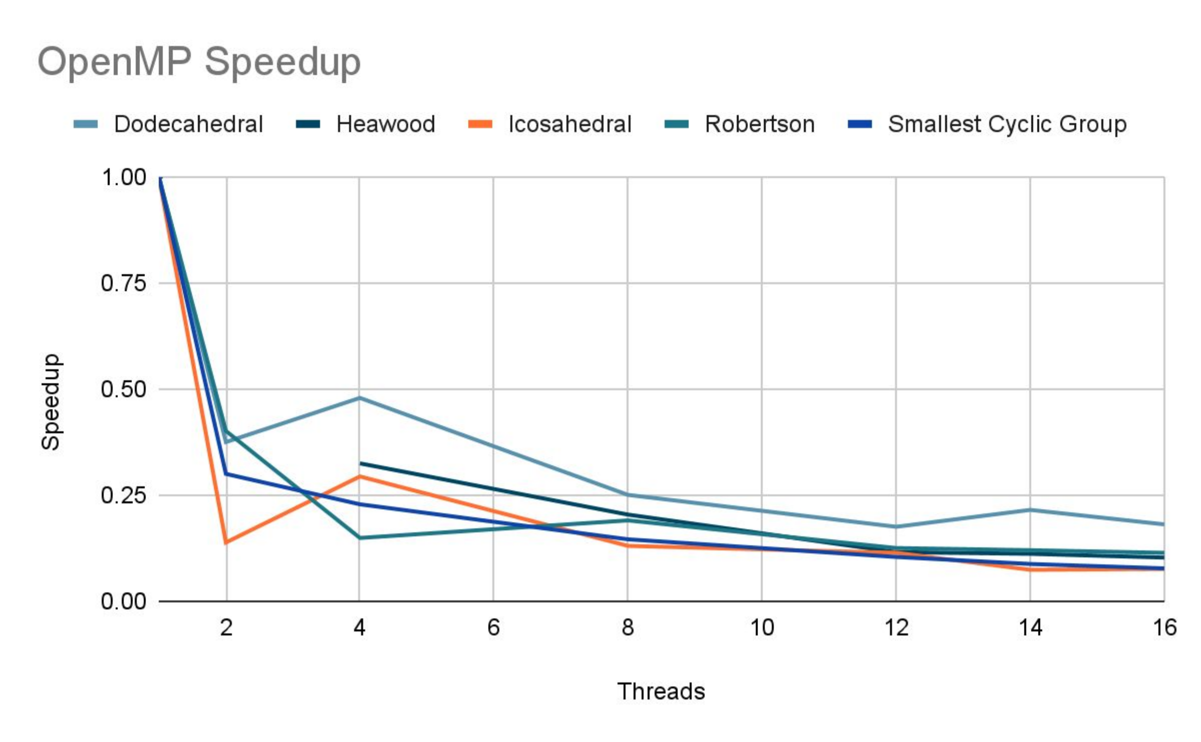
Experimentation

- Dataset
  - Sourced from Open Source PageRank Implementation testing suite
  - Collection of famous graphs from mathematics
- Methodology
  - Run each version on 1, 2, 4, 8, 12, 14, and 16 threads
  - Run on Dodecahedral, Heawood, Icosahedral, Robertson, and smallest cyclic group graphs

Speedup Graphs



Speedup Graphs



Analysis

- OpenMP performed significantly faster than C++ Native Threads
- C++ Native Threads had much better scalability
- Verified that Power Iteration algorithm converges quickly with <100 iterations

Challenges

1. Original proposal was Neural Networks from scratch, we gave up on this
2. Naive PageRank algorithm runs into Dead-Ends
3. Modified PageRank algorithm we used is still simplified, thus we couldn’t compare our PageRank scores with the ground truth scores provided in the testing graph repository

Task Distribution

- Benjamin Wang
  - Researched PageRank, Power Iteration algorithm, and Eigen library
  - Implemented serial and parallel version of PageRank with OpenMP
  - Implemented function for reading input data
- Ojas Malwankar
  - Researched datasets to use for PageRank
  - Implemented parallel version of PageRank with C++ Native Threads

Work Cited

- Fang, Yi “COEN 272 Lecture 8 PageRank” Lecture at Department of Computer Science and Engineering, Santa Clara University, CA, November 15, 2021. Accessed December 6, 2021.

- Panos Louridas, Georgios Gousios, and Yanran Li. “Open-Source PageRank Implementation”, (2014), GitHub Repository, https://github.com/louridas/pagerank