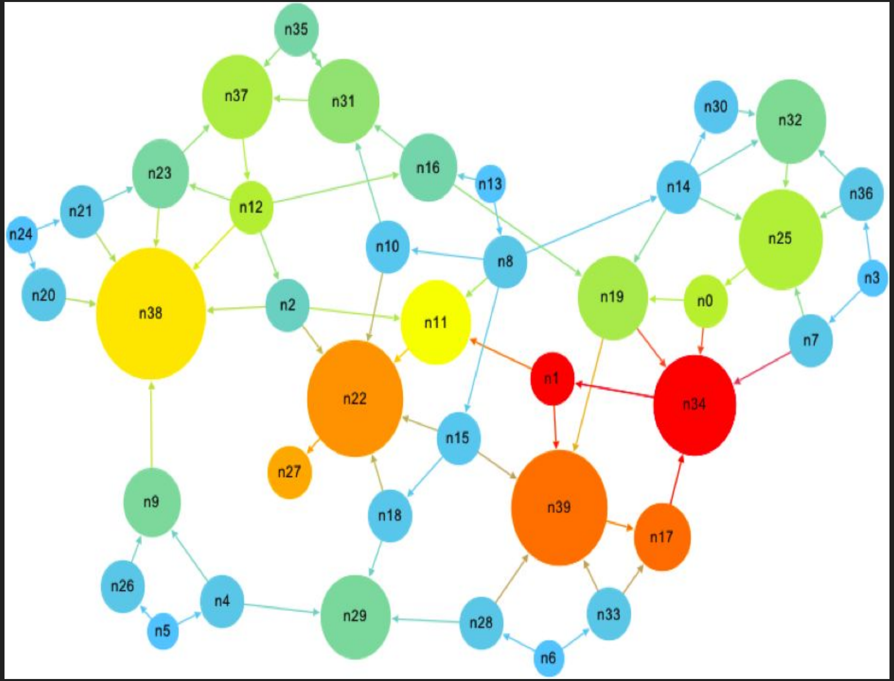
Google PageRank (OpenMP)

Niharika Khare (2018201002)
Sopnesh Gandhi (2018201064)
Shreya Upadhyay (2018201091)

INTRODUCTION

PageRank is link analysis algorithm used by Google Search to rank websites. PageRank was named after Larry Page, one of the founders of Google. PageRank is a way of measuring the importance of website pages.



BASIC IDEA

PageRank works by counting the number and quality of links to a page to determine a rough estimate of how important the website is. The underlying assumption is that more important websites are likely to receive more links from other websites.

CONCEPT OF DAMPING FACTOR

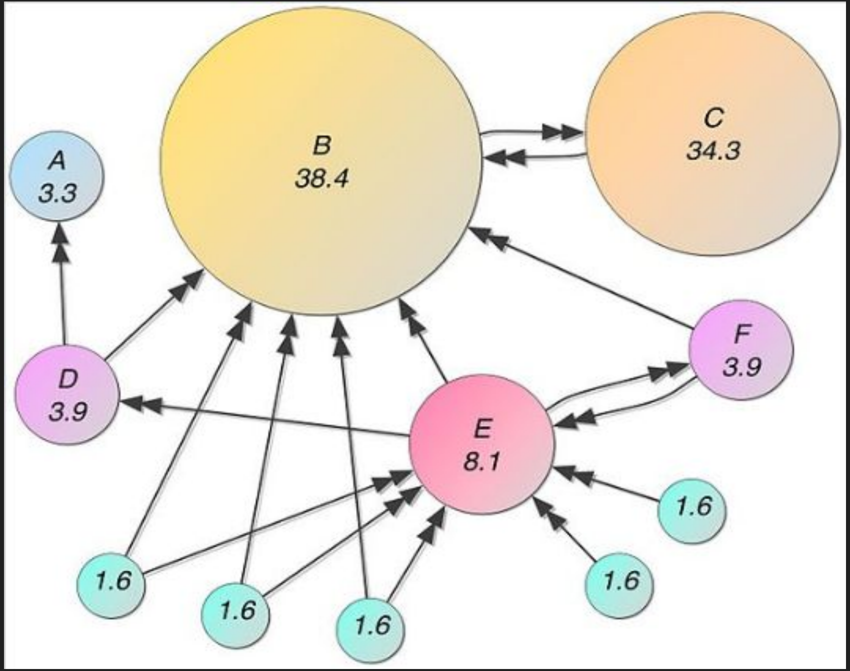
The PageRank theory holds that an imaginary surfer who is randomly clicking on links will eventually stop clicking. The probability, at any step, that the person will continue is a damping factor *d*.

$$PR(A) = \frac{PR(B)}{L(B)} + \frac{PR(C)}{L(C)} + \frac{PR(D)}{L(D)}.$$

$$PR(A) = \frac{1-d}{N} + d\left(\frac{PR(B)}{L(B)} + \frac{PR(C)}{L(C)} + \frac{PR(D)}{L(D)} + \cdots\right)$$

More About PageRank

The PageRank value of a page reflects the chance that the random surfer will land on that page by clicking on a link. It is a Markov chain in which the states are pages, and the transitions, which are all equally probable, are the links between pages.



ITERATIVE METHOD FOR CALCULATING PAGE RANK

1. Initially, at time step t=0, assume equal page ranks for all of the N pages. Ranks are stored in rank matrix, R.

$$PR(p_i; 0) = \frac{1}{N}$$

2. Rank matrix is updated for each iteration, using the damping factor d

$$\mathbf{R}(t+1) = d\mathcal{M}\mathbf{R}(t) + \frac{1-d}{N}\mathbf{1}, \text{(*)}$$

Matrix M is defined as,

$$\mathcal{M}_{ij} = \begin{cases} 1/L(p_j), & \text{if } j \text{ links to } i \\ 0, & \text{otherwise} \end{cases}$$

3. Repeat step 2 for each iteration (time step). The computation converges when the rank matrix at time step t differs from rank matrix at time t+1 only by a small threshold.

$$|\mathbf{R}(t+1) - \mathbf{R}(t)| < \epsilon,$$

OpenMP Directives

Parallel :- Defines a parallel region, which is code that will be executed by multiple threads in parallel.

for :- Causes the work done in a for loop inside a parallel region to be divided among threads.

atomic :- Specifies that a memory location that will be updated atomically.

OpenMP Clauses

Reduction(+ : var) :- Specifies that one or more variables that are private to each thread are the subject of a reduction operation at the end of the parallel region.

shared(var) :- Specifies that one or more variables should be shared among all threads.

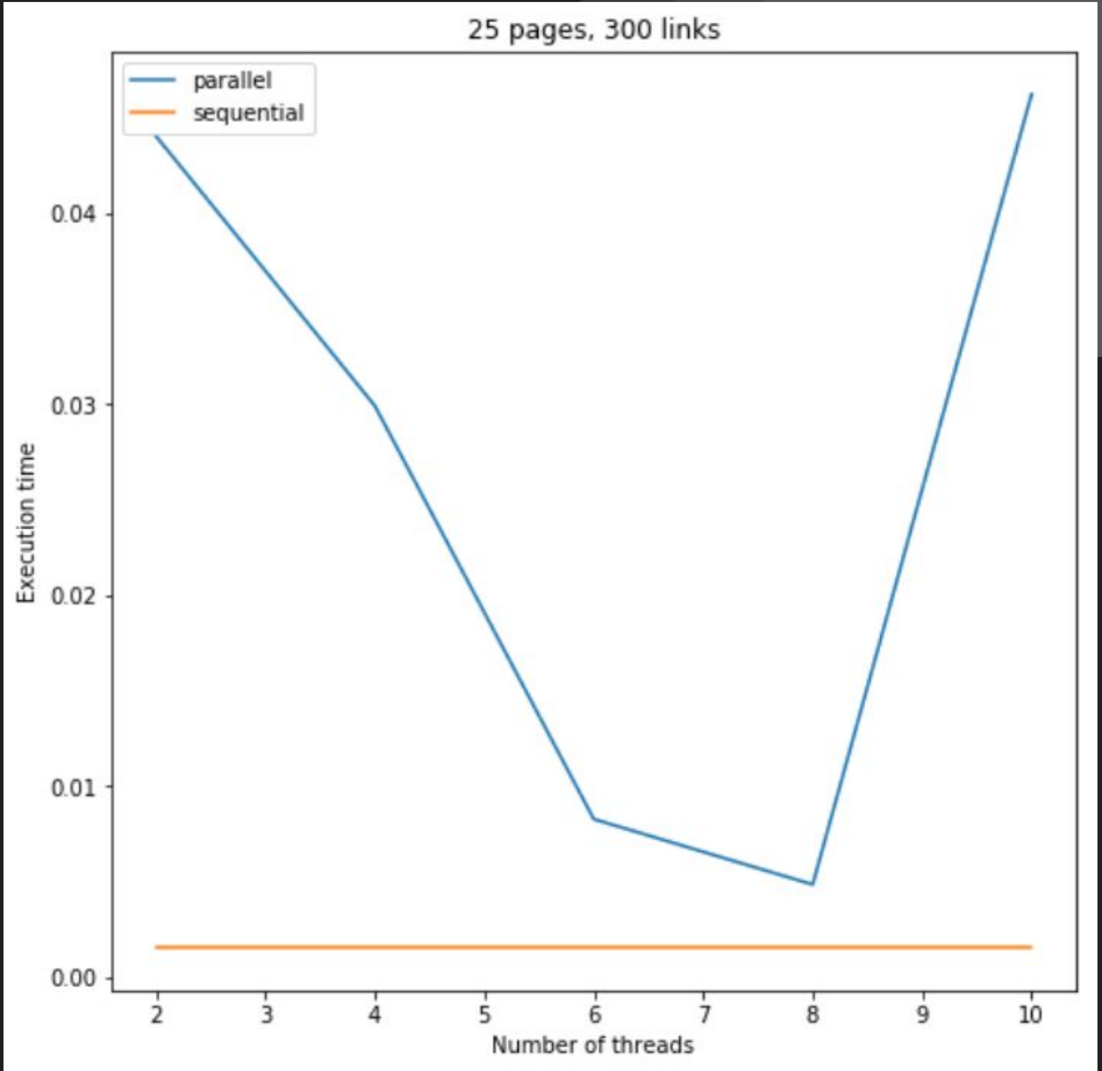
num_threads(num) :- Sets the number of threads in a thread team.

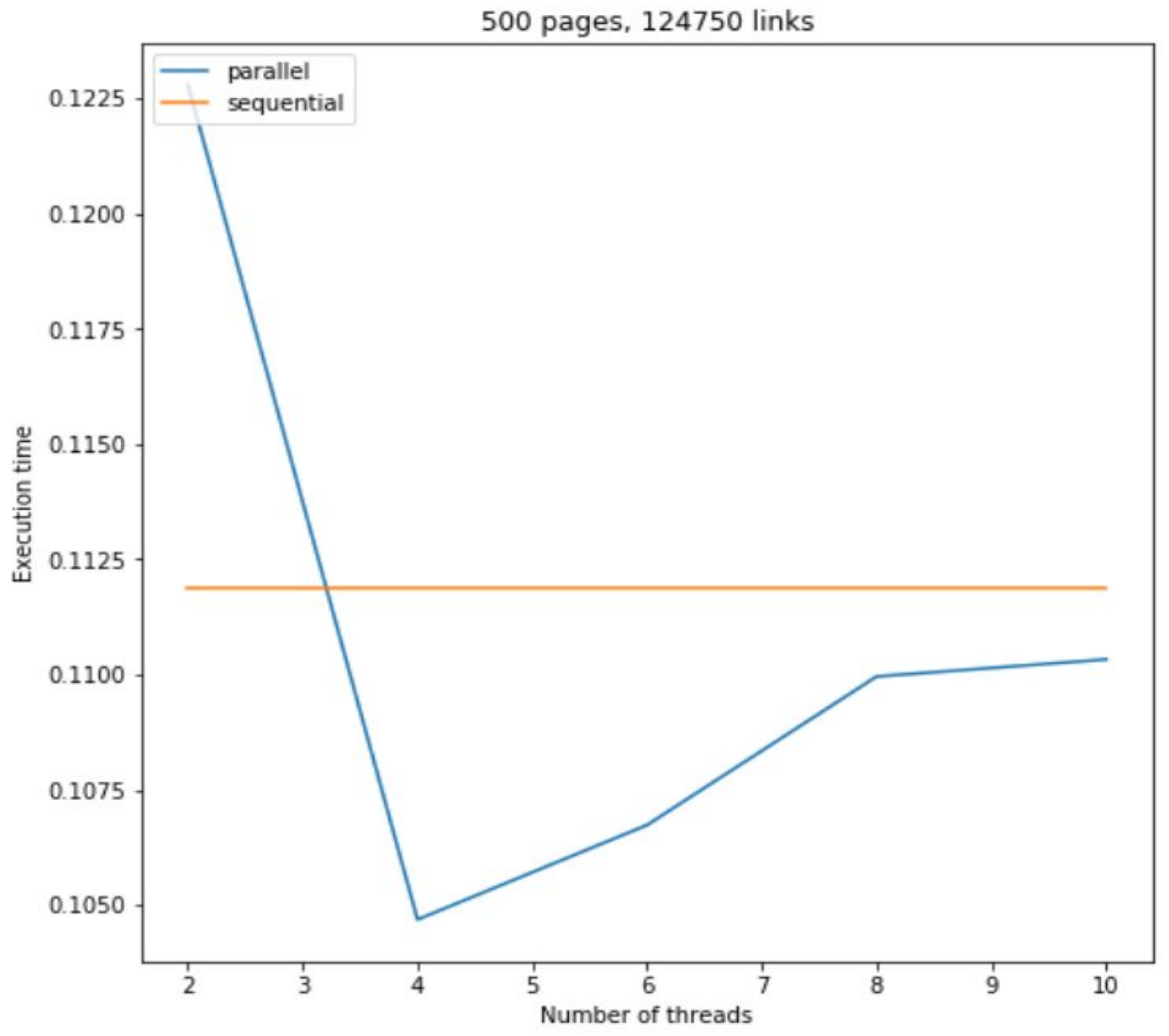
RESULTS

Datasets used:

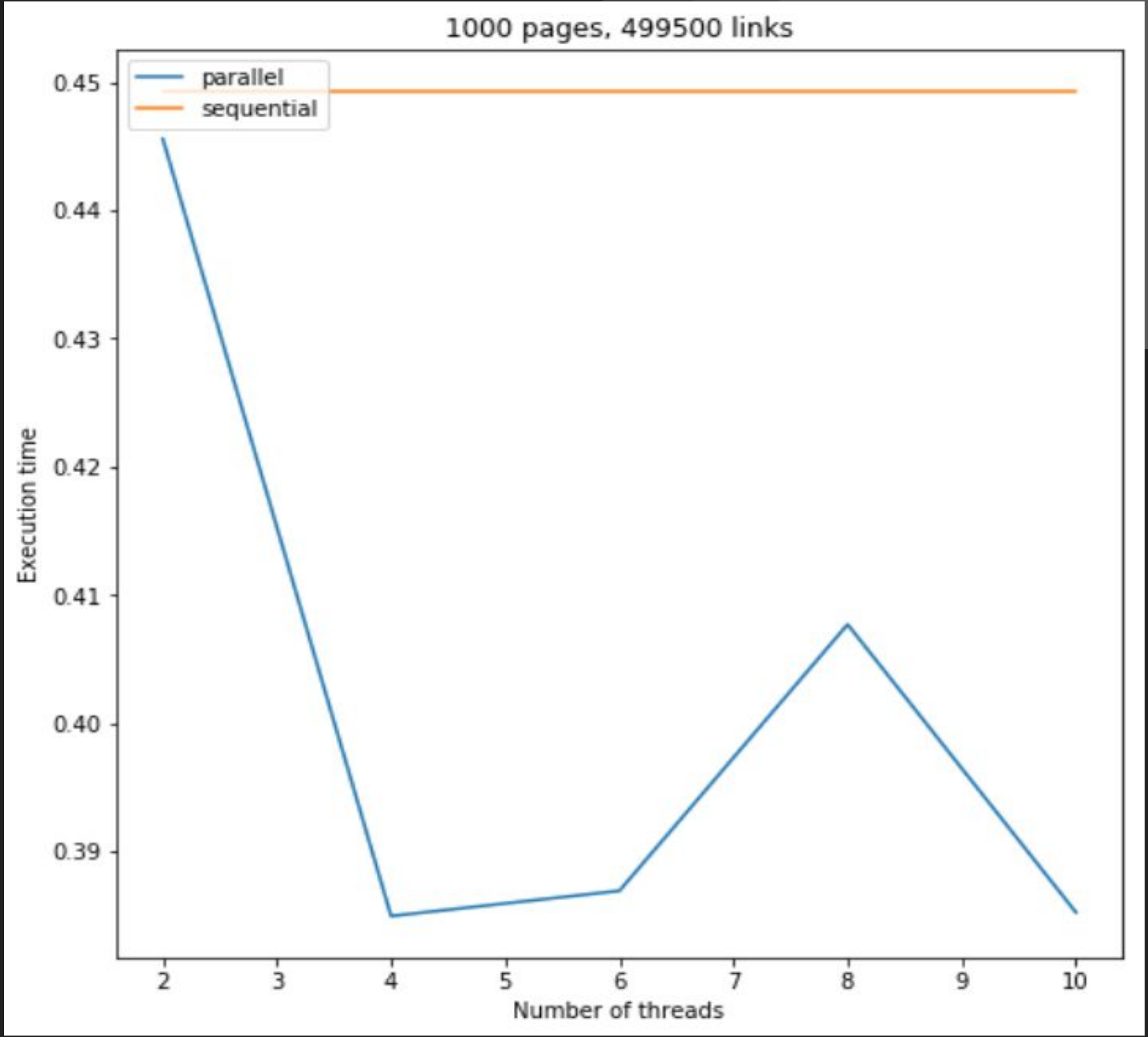
1. 25 pages, 300 links
2. 500 pages, 124750 links
3. 1000 pages, 499500 links

Parallel version of algorithm with different number of threads was executed on above datasets and compared with sequential algorithm’s output.


25 pages, 300 links
parallel
sequential
Execution time
Number of threads
0.00
0.01
0.02
0.03
0.04
2
3
4
5
6
7
8
9
10



500 pages, 124750 links
parallel
sequential
Execution time
0.1225
0.1200
0.1175
0.1150
0.1125
0.1100
0.1075
0.1050
2
3
4
5
6
7
8
9
10
Number of threads


1000 pages, 499500 links
parallel
sequential
Execution time
0.45
0.44
0.43
0.42
0.41
0.40
0.39
Number of threads
2
3
4
5
6
7
8
9
10

CONCLUSION

We observed that parallelism can increase computational overhead if:

1. Number of pages is less (ex: 25 pages)
2. Number of threads is too less or too more.

In general, parallelism would provide a better time efficiency than sequential execution.

THANK YOU