Implementing Sequential Page Rank Algorithm

Purshottam Vishwakarma
School of Informatics and Computing
Indiana University, Bloomington
pvishwak@indiana.edu

Bitan Saha
School of Informatics and Computing
Indiana University, Bloomington
bsaha@indiana.edu

Introduction:

Page Rank Algorithm, developed by Larry Page and Sergey Brin at Stanford University as part of their research, is the brain behind Google search engine. It uses probability distribution across all the pages over the web and assigns a value between 0 and 1 to each web page which determines the popularity of the page. Computation of page rank is an iterative process and the accuracy of the page rank is directly proportional to the number of iterations performed. Considering billions of web pages over the internet it is a highly compute intensive task and requires the algorithm to be run in parallel over a distributed environment. This Project implements the sequential version of the algorithm in a far smaller scale i.e. we would rank only 1000 web pages. *We have tried to make this algorithm highly efficient (3x faster) as compared to our peers.*

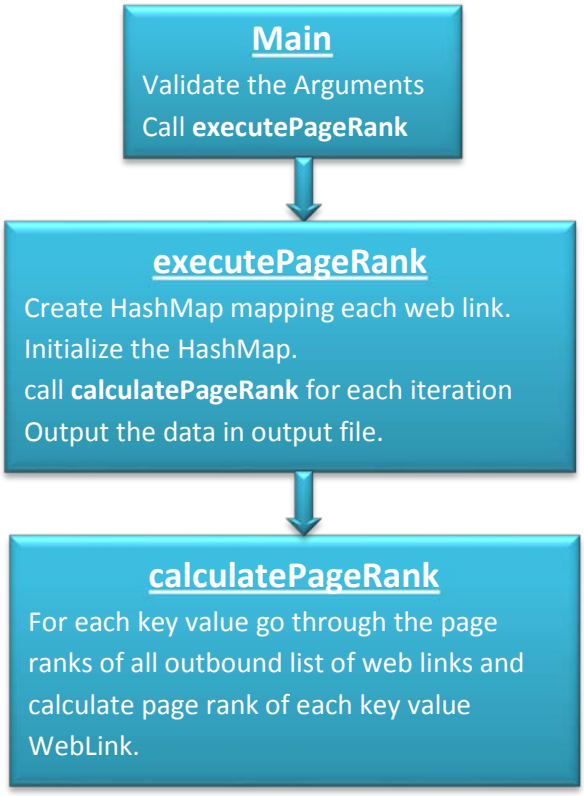
Theory:

In the general case, the PageRank value for any page i can be expressed as:

$$PR(p_i) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)}$$

PR, pagerank (a probability value)
pi , a page under consideration
L(pi), the number of outbound links on page pj
d, damping factor which can be set between 0 and 1 (It is usually set d to 0.85)
N, total number of pages.

Architecture design (e.g. architecture flowchart and text)



Implementation:

- The crux of the algorithm is in maintaining the current and previous states (i.e. iteration) for each URL/WebLink.
- The Page Rank of the master WebLink from the previous State (i.e. iteration) gets added to the Page Rank of the WebLink which appears on the outbound list of the master WebLink. For example, consider following three rows in the adjacency matrix.

  2 1
  3 0 1
  4 1 3 5

  While traversing each row, the page rank of the WebLink 1 gets calculated (cumulative addition) by adding the page rank values of WebLink 2, WebLink 3 and WebLink 4 from the previous state (i.e. iteration. The value is saved in WebLink.*pageRankPreviousIte*)

- The cumulative sum of Page ranks of all the dangling WebLinks is calculated at the end of each iteration and added to the page rank of each individual WebLInk in the next iteration whenever we encounter any WebLink for the **first time**.

- Whenever we encounter any WebLink for the **first time** in an iteration, we calculate its page rank and **change its state** i.e. the current iteration page rank value is saved in previous iteration page rank value and current iteration Page rank value is set to 0. In other words, the change in state takes place only for the first time for any WebLink in a single iteration. Henceforth, the state of the WebLink is not changed again during that particular iteration.

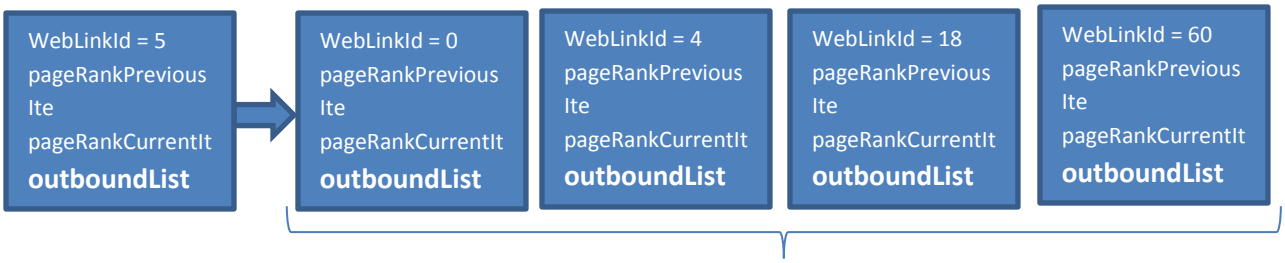
Below are the important data structure used in the program.

1. **Nested Class WebLink**

```
public static class WebLink
{
        private int webLinkId;                  // The Unique Vertices/Web_Page Id
        private double pageRankPreviousIte;     // Page Rank value from the previous iteration.
        private double pageRankCurrentIte;      // Page Rank value in the current iteration.
        private int iterationState;             // To match the iteration state
        private List<WebLink> outBoundList;     // List of Out Bound Web Links
}
```

Each node or URL is represented in the form of this class structure. Each node or WebLink includes the list of all the outbound nodes or WebLinks and holds the state of the page and the associated links. This is initialized in the method *initPageRankMap().* For example, if node 5 contains the outbound list of web pages {0 4 18 60} then it is structurally represented as below.

WebLink is a Nested Class which is static in nature.



Outbound List of webLink 5

2. **HashMap<Integer, WebLink> pageRankValueMap**

This hashMap contains the master key node/WebLink and the master WebLink contains the list of all its outbound WebLinks as shown in the diagram above.

Performance Analysis:

We have benchmarked our program with the provided executable program and found that our program runs 3x faster. The reason for the efficiency is that *we traverse the adjacency matrix i.e. each row only once during each iteration.* The program was designed and implemented considering efficiency as an important factor.

Results:

Below are the page ranks of the top 10 ranking URL numbers.

```
4       0.1378706069078275
34      0.12201478289415381
0       0.1128894671978259
20      0.07753502514138039
146     0.057255738146337457
2       0.048176986656400085
12      0.020251045153269583
14      0.01799035915631113
16      0.013077696931789354
6       0.012928153212791347
```

Acknowledgements:

We acknowledge the efforts of our Instructor **Judy Qiu** and Associate Instructors **Ikhyun Park & and Tak Lon Wu** in making us understand the basic concepts behind page rank.

References

1. Sergey Brin and Lawrence Page, The Anatomy of a Large-Scale Hypertextual Web Search Engine, Stanford University, WWW7 Proceedings of the seventh international conference on World Wide Web 7 , 1998
2. http://en.wikipedia.org/wiki/Markov_chain
3. http://en.wikipedia.org/wiki/Adjacency_matrix
4. http://en.wikipedia.org/wiki/PageRank