Part 1 of Project 1: PageRank algorithm

1. Introduction

PageRank

The web search engine is one of typical distributed system in the Internet. It is designed to search for information on the World Wide Web. The search results are generally presented in a list of results and are often called hits. The PageRank is a well know web graph ranking algorithm that help the Internet user sort the hits by their importance.

PageRank calculates numerical value to each element of a hyperlinked set of web pages, which reflects the probability that the random surfer will access that page. The process of PageRank can be understood as a Markov Chain[1] which needs iterative calculation to converge. One iteration of PageRank calculates the new access probability for each web page based on values calculated in the previous iteration. The iterating will not stop until the number of current iterations is bigger than predefined maximum iterations, or the Euclidian distance between rank values in subsequent two iterations is less than a predefined threshold that controls the accuracy of the output results.

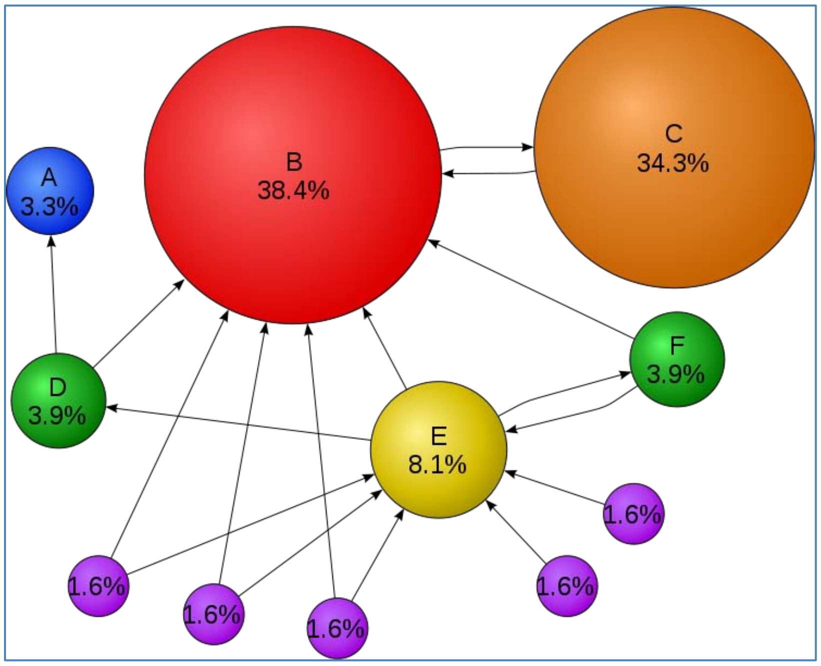

Fig.1 Mathematical PageRank for a simple network in Wikipedia

Fig.1 shows the web graph consist of 11 vertices {A, B, C, D, E, F, G1, G2, G3, G4, G5}. Each vertex refers to unique web page, and the directed edge means there is one link from source web page to target web page. The percentage of each vertex presents the rank value of each web page.

CSCI B534 2011

Notes:

You can implement the sequential PageRank that can run on desktop or laptop. But, when processing larger input data, like web graph contains more than millions web pages, you need run the PageRank application in parallel so that it can aggregate the computing power of multiple compute nodes. Currently, in both industry and academic, the study of large scale web or social graphs becomes increasingly popular. In published paper, the jobs execution engines that claim to support large scale PageRank include: MPI, Hadoop, Dryad, Twister, Pregel.

Project #1 is divided into two parts. In first part, you need implement the sequential version of PageRank. In second part, you will implement a parallel version of PageRank by using the programming interfaces of Hadoop MapReduce job execution engine.

Formula

Equ.1 is the formula to calculate the rank value for each web page. We will learn this formula by applying it to the case in Fig.1. There are 11 web pages in Fig.1 which include: {A, B, C, D, E, F, G1, G2, G3, G4, G5}. Assume the probability distribution for a web surfer accessing all these 11 Page in current iteration is {PR(A), PR(B), PR(C), … PR(G5)}, then the probability for the surfer to access Page B in next iteration is:

PR(B) = PR(D)/2 + PR(E)/3 + PR(F)/2 + PR(C) + PR(G1)/2 + PR(G2)/2 + PR(G3)/2

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

Equ.1: $PR(u) = \sum_{v \in Set} \frac{PR(v)}{L(v)}$

The vertices set in the right of formula contain all the web pages that point to target web page 'u'. The L(v) refers to the out degree of each web page in the vertices set. The initial rank values of each web page, like PR'(u), can be randomized double. After several iteration calculations, the rank values converge to the stationary distribution regardless of what their initial values are. Markov Chain [1]

Damping factor

The PageRank theory holds that even 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**. Various studies have tested differed damping factors, but it is generally assumed that damping factor will be around 0.85. The formula considering damping factor is shown at Equ.2. N refers to the total number of the unique urls.

Equ.2: $PR(u) = \frac{1-d}{N} + d * \sum v \in Set \frac{PR(v)}{L(v)}$

2. Execution guide for sample PageRank code

To help you understand the process of PageRank computation better, we offer an executable PageRank java class file in assignment package. Its usage is:

Java SequentialPageRank [input file name] [output file name] [iteration count] [damping factor]

We also provide a small PageRank input file which is built based the web graph showed in Fig.1. You can evaluate the rank values in Fig. 1 by running SequentialPageRank with following parameters.
Java SequentialPageRank pagerank.input pagerank.output 10 0.85

To check the rightness of output rank values, you can open pagerank.output file and see whether the output results are the same as rank values in Fig. 1. Generally speaking, the larger the number of iterations, the more accurate the computed rank values will be.

3. Programming guide for PageRank

Input Data format

The input data for PageRank application is the web graph in adjacency matrix format [2]. In our sample program, it transfers the web graph into a simplified adjacency matrix. Following is the steps we constructed adjacency matrix for web graph in Fig.1:

1) Construct a set of tuples that describe the web graph structure: WebG = {(A,null), (B, C), (C, B) (D, A, B), (E, B D F), (F, B E), (G1, B E), (G2, B E), (G3, B E), (G4, E), (G5, E)
2) Map letters to numbers. A->0, B->1, C->2, D->3, E->4, F->5, G1->6, G2->7, G3->8, G4->9, G5->10
3) Construct the simplified adjacency matrix based on information in step 1,2.

```
0
1 2
2 1
3 0 1
4 1 3 5
5 1 4
6 1 4
7 1 4
8 1 4
9 4
10 4
```

In the program, you are not supposed to construct the adjacency matrix from web graph yourself. Instead, we will provide each group with different adjacency matrix input file (pagerank.input.1000.urls.groupid) which contains one thousand unique urls. Besides, you can use HashMap<Integer, ArrayList<Integer>> to store the adjacency matrix.

Rank values table

In the program, you need a rank value table that store intermediate rank values within one iteration. The process of PageRank computation is the process of updating the rank values table by applying to Equ.2. The number of elements in the rank values table is the number of unique urls in the web graph you will study, which is 11 in Fig.1.

4. Deliverables

You are required to turn in following items in this assignment.

1) The source code of sequential PageRank you implemented.
2) The executable class file, the README file that describe its usage.
3) Technical report that contains:
   a. The description about the main steps or program flow in your program.
   b. The output file which contains the top 10 ranking url numbers.

5. References

1. http://en.wikipedia.org/wiki/Markov_chain
2. http://en.wikipedia.org/wiki/Adjacency_matrix
3. http://en.wikipedia.org/wiki/PageRank