**Report on PageRank algorithm sequential implementation**

**Group 26 - Prashanth Balasubramani and Shree Harsha Sridharamurthy**

The java program calculates the pagerank as per the template provided. Control flow follows the main method's definition:

1. parseArgs:

This method is used to obtain the input from the user and hold the values such as the input and output filepaths, the number of iterations and the damping factor. This method stores these values in the global private variables declared in the beginning of the program.

We have also ensured that the error conditions are handled for improper input.

2. loadInput:

This method uses FileReader and BufferedReader classes to read the input file as specified in the command line. The file is read into an adjacency matrix (adjMatrix) declared globally. Here too, exceptions are handled for missing input file.

3. calculatePageRank:

The main logic for calculating the pagerank is added in this method. The global adjacency matrix **adjMatrix** is used as the input and the global variable **rankValues** is where the page rank values are stored repeatedly for every iteration

The dangling node issue is also addressed by adding all the nodes (except the node under consideration) as making the denominator equal to the total number of nodes.

When we ran the program for pagerank.input file, we obtained results which satisfied the condition that all the page rank values must add up to 1. Similarly, we also executed the program for the input file **pagerank.input.1000.urls.26** corresponding to our group number.

As expected, when we ran the code for very less number of iterations, we got incorrect values and then the output became steady after 10 and more number of iterations.

4. printValues:

In this method, we output the results into a file with the name as specified in the command line argument.

The output files for each iteration are attached separately.

Some output on the screen were as follows:

**$ javac SequentialPageRank.java**

**$ java SequentialPageRank pagerank.input pagerank.output.10 10 0.85**

Number of Iterations : 10

Page:1 PageRank:0.43678478698046236

Page:2 PageRank:0.3894112227116214

Page:4 PageRank:0.08773670222780837

Page:3 PageRank:0.04300968888704591

Page:5 PageRank:0.04300288607610744

Page:0 PageRank:0.0332910400047713

Page:9 PageRank:0.0181441537782284

Page:8 PageRank:0.0181441537782284

Page:7 PageRank:0.0181441537782284

Page:6 PageRank:0.0181441537782284

**$ java SequentialPageRank pagerank.input pagerank.output.400 400 0.85**

Number of Iterations : 400

Page:1 PageRank:0.42556946091232417

Page:2 PageRank:0.37987652340539607

Page:4 PageRank:0.0877185899985423

Page:5 PageRank:0.0429960821295075

Page:3 PageRank:0.0429960821295075

Page:0 PageRank:0.03327333490504069

Page:9 PageRank:0.018142481629920514

Page:8 PageRank:0.018142481629920514

Page:7 PageRank:0.018142481629920514

Page:6 PageRank:0.018142481629920514

**$ java SequentialPageRank pagerank.input.1000.urls.26 pagerank.output.1000.urls.26.10 10 0.85**

Number of Iterations : 10

Page:10 PageRank:0.0977592456313036

Page:0 PageRank:0.07993836930331293

Page:28 PageRank:0.07797319577268325

Page:4 PageRank:0.07782284858586187

Page:34 PageRank:0.03635535217481098

Page:256 PageRank:0.02877380731341521

Page:134 PageRank:0.026577063891009455

Page:38 PageRank:0.02273109826423452

Page:14 PageRank:0.020934632877249747

Page:68 PageRank:0.019757614125512395

**$ java SequentialPageRank pagerank.input.1000.urls.26 pagerank.output.1000.urls.26.400 400 0.85**

Number of Iterations : 400

Page:10 PageRank:0.0949845311461513

Page:0 PageRank:0.07772344757968772

Page:28 PageRank:0.07585651951378598

Page:4 PageRank:0.07546696927440431

Page:34 PageRank:0.03538201207134958

Page:256 PageRank:0.027970670274563802

Page:134 PageRank:0.025568650218598128

Page:38 PageRank:0.022168820517089592

Page:14 PageRank:0.020412385107473886

Page:68 PageRank:0.01924988851448421