**IR Assignment 2**

**Team Members:**

|  |  |
| --- | --- |
| **Name** | **ID** |
| R. Shriram | 2015A7PS0011H |
| Bhavitej R | 2015A7PS0021H |
| Rakesh Chinthakrindi | 2015A7PS0052H |
| Akash Sebastian | 2015A7PS0967H |

**Code files:**

* main.cpp
* generate\_random\_graph.java
* result.py

**main.cpp:**

No external packages were used.

*Variables used:*

* vector<int> adj[V] (adjacency matrix)
* double\*\* M (Mij/di where di is the no of out links of ith node)
* double\* R (rank matrix)
* double\*\* A (adjacency matrix)
* double\* previous (copy of rank matrix)

*Functions used:*

* void addEdge(vector<int> adj[], int u, int v), adds an edge to the adjacency list.
* void print\_to\_file(), prints the adjacency list to a file named "AdjacencyList.txt".
* void print\_to\_file\_matrix(), prints a matrix to file.
* void matrix\_multiplication(),matrix multiplication between A matrix and R matrix(Rank matrix).
* void copy\_array(),copies the rank matrix into previous for error calculation.
* double error(),calculating error using rmse between R and previous.
* void print\_r(),prints the rank matrix(purely for testing purpose).
* void addleak\_r(),adds back the leak in rank to handle deadend.
* void power\_method(),executes page rank algo using power method.
* void print\_page\_rank(),writes the page ranks to a file.

*Formula used:*

Page Rank algorithm was used.

<https://en.wikipedia.org/wiki/PageRank>

Most of the memory was declared dynamically and have been freed. And beta was taken to be 0.8. All precisions are of double where required.

**generation\_random\_graph.java:**

*Packages:*

* java.io.PrintWriter
* java.util.ArrayList
* java.util.List
* java.util.Random

*Class:*

Class to generate random graph

class random\_graph {

private int nodes;

private List connected;

private List unconnected;

private PrintWriter writer;

random\_graph(int n);

void generate(double percentage);

void addEdge(int u,int v);

}

*Variables used:*

* private int nodes, contains the total no of nodes to be created.
* private List connected, contains the list of connected nodes.
* private List unconnected, contains the list of unconnected nodes.
* private PrintWriter writer, writes the created graph to a text file.
* int extra\_edges, generate a max of (0.1\*nodes\*nodes/100)% edges.
* Random R, used to pick random nodes from unconnected array.

*Functions used:*

* Constructor initializes, takes one parameter the no of nodes and assigns to node. It also initializes all the other variables to default values.
* void generate(double percentage), this function contains the algorithm to generate a graph, the passed parameter is the maximum percentage of edges in node\*node.
* void addEdge(int u,int v), adds an edge directed from u to v.

Note: This program only produces connected graphs.

**result.py:**

The documentation of this is generated using pydoc(result.html). This program aims to read the page rank output produced by main.cpp. And produce a visualization of the graph, which was done using networkx.