RMIT University - School of Science

Assignment 1
Six Degrees of Separation [Report]

Student ID	s3712755
Student Name	Moafaq Ashshareef
Course Code	COSC 2123/1285
Course Title	Algorithms and Analysis
Submission Date	15 Apr 2018

Introduction:

The focus on this report is to find out how efficient are the usage of both the Adjacent Matrix and Incidence Matrix in representing graphs of friendship. The input used to build up the graph is from (facebook_combinedSmall.txt) which has smaller sample of data than the one provided. Some vertices have more density than others i.e. more edges. The timing taken to add the input file into forming a graph including addition of vertices and edges will be counted in addition to the time taken to remove them. As well as for finding neighbours of a vertex and shortest path between two vertices. `System.currentTimeMillis()` is used inside GraphTester class to do than (see figure 1)

```
287
288             // process the operations
289             start = System.currentTimeMillis();
290             processOperations(inReader, graph, verticesOutWriter, edgesOutWriter, neighbourOutWriter, distanceOutWriter);
291         } catch (IOException e) {
292             System.err.println(e.getMessage());
293         }
294         long timeTaken = System.currentTimeMillis() - start;
295         System.out.println("Time Taken in Milliseconds= " + timeTaken );
296
297     } // end of main()
```

Figure 1: System.currentTimeMillis() in GraphTester.java

For both adjacency and incidence matrices, an empty 2D array is firstly defined of type int which then are modified when adding edges. For vertices, ArrayList is used to make it easier to expand it every time a vertex is added.

The graph is loaded using the following command in which the time is printed after running it when passing commands which is stored in testAddCommand.in for example. Different .in files were created to test for the other scenarios. Tests were done with both adjmat and sample implementations.

```
java -cp jopt-simple-5.0.2.jar:sample.jar: GraphTester adjmat -f
facebook_combinedSmall.txt < testAddCommand.in
```

```
1   import java.io.*;
2   import java.util.*;
3
4   public class Generator {
5       public static void main(String[] args) {
6           try {
7               PrintWriter writer = new PrintWriter("testAddCommand.in", "UTF-8");
8               for (int i = 0; i < 1000; i++) {
9                   writer.println("AV " + i);
10              }
11              System.out.println("File Created");
12              writer.close();
13          } catch (IOException e) {
14              System.out.println("Something went wrong while creating the file");
15          }
16      }
17  }
```

Figure 2: testAddCommand.in generator

To test the `AdjMatrix` code, python test command was run as follow:

```
Moafaqs-Air:CW1 moafaq$ python assign1TestScript.py -v javaSrc adjmat testing/tests/test1.in
javac -cp .:jopt-simple-5.0.2.jar:sample.jar *.java


java -cp .:jopt-simple-5.0.2.jar:sample.jar GraphTester adjmat test1-adjmat.vert.out test1-adjmat.edge.out test1-adjmat.neigh.out test1-adjmat.dist.out
Testing: java -cp .:jopt-simple-5.0.2.jar:sample.jar GraphTester adjmat test1-adjmat.vert.out test1-adjmat.edge.out test1-adjmat.neigh.out test1-adjmat.d
ist.out
Time Taken in Milliseconds= 193

SUMMARY: GraphTester has passed 1 out of 1 tests.
PASSED: test1
```

Figure 3: Python test of AdjMatrix Class

Scenario 1 Growing friendship graph (Additions):

After loading the graph of facebook_combinedSmall.txt, additional random vertices where added using the `AdjMatrix addVertex` function. Average time taken to add 1000 vertices is: 9.8ms

While when running the sample class, average was faster at about: 3.4ms

For edge adding, the scrip adds edges between existing vertices of the facebook graph. Time taken was: 3.4ms while for sample implementation, it was: 3.6ms

Scenario 2 Neighbourhoods and shortest paths:

Finding neighbours of a vertex took 124ms in average after running the command 5 times. For sample implementation it was 127.8ms in average

Finding shortest path took 153.6ms in average after running the command 5 times. For sample implementation it was 318ms in average!

Scenario 3 Shrinking friendship graph (Removals):

Removing edges of existing edges in facebook graph took 3ms in average after running the command 5 times. For sample implementation it was 3.4ms in average

Evaluation:

In theory, higher edge density i.e. more edges for each vertex, operation would take longer time.