BFS

Homework #10

By:

Pratyusha Thundena

CS 303L – L1A

November 6, 2018

### Problem Specification

The objectives included the following: implementing undirected graphs using adjacency lists, performing Breadth First Search (BFS) algorithm on the undirected graphs, and printing the BFS path from a source to all the other nodes in the graph.

### Program Design

This program required the following: File class, FileNotFoundException class, ArrayList class, LinkedList class, Queue class, Scanner class, mediumG.txt and largeG.txt files.

The following steps were required to develop the program:

1. create a method to read input files
2. create method to form a list of nodes using the information from the input files
3. create method to add adjacent/neighbor nodes for each vertex
4. create a method to do BFS and provide way to get path from source 0 to nodes
5. create a driver program to ask the user for file name, read the graph, print the graph, and print path from source 0 to nodes
6. close the scanner

The following constructors and methods were defined within the class:

1. Graph ()

Basic method that creates a list of nodes and adds vertices to the list.

b) getNumNodes ()

Basic method that returns the number of nodes.

c) addNeighbors ()

Basic method that adds a as neighbor of b and adds b as neighbor of a.

d) LinkedList<Integer> getNeighbors ()

Basic method that returns the list of adjacent nodes.

e) print ()

Basic method that prints the graph.

f) readFile ()

Basic method that reads the input files.

g) BFS ()

Basic method that implements BFS.

h) printPath ()

Basic method that determines if path exists and prints the path if it does exist.

i) main ()

Driver program that reads graph, prints graph, and performs BFS.

j) V ()

Constructor that instantiates the number of nodes.

k) nodesList ()

Constructor that helps add neighbors of a node.

l) graph ()

Constructor that instantiates the getter.

m) queue ()

Constructor involved in BFS.

The println method of the System.out object displays the inputs and results for the driver program.

### Testing Plan

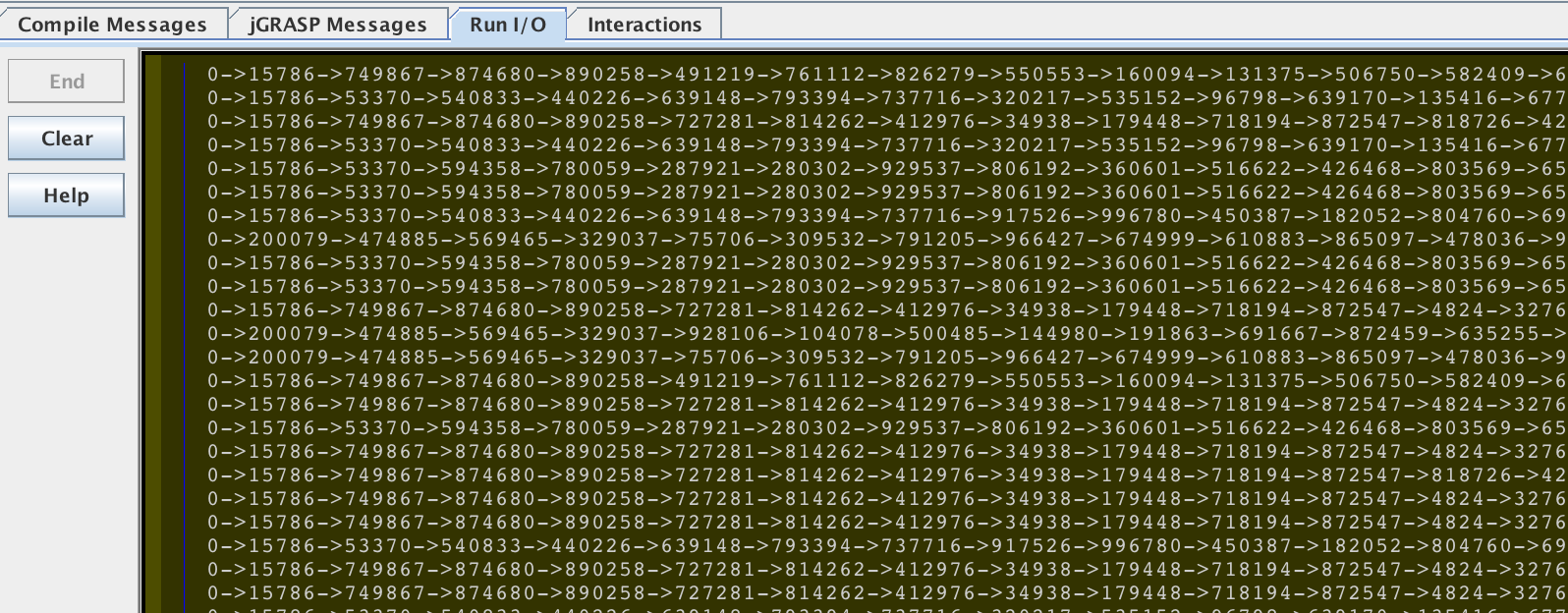
The test plan involved creating adjacency lists and implementing BFS to find a path from source 0 to the nodes by using mediumG.txt and largeG.txt files.

### Results

### Figure 1: Output of mediumG.txt File

### 

### Figure 2: Output of largeG.txt File



### Analysis and Conclusions

The output for mediumG.txt file was obtained within two minutes, while the output for largeG.txt file was obtained within 28 minutes. The latter was primarily due to the text file containing significant amount of data. The input files (mediumG.txt and largeG.txt) contained the number of vertices, the number of edges, and the corresponding list of edges. Undirected graphs were created for mediumG.txt and largeG.txt (Figure 1 and Figure 2). For example, 249 represented the vertex and the listed numbers are the edges (Figure 1). Then the output from source 0 to a list of vertices to 1 represented a path. For example, there was a path from 0 to 160 to 187 to 77 to 78 to 112 to 234 to 130 to 1 (Figure 1). BFS was performed to find a path from the source. For BFS, an iteration through the vertices in loop was performed and was executed V times. Then the edge would be counted for each vertex since each vertex was dequeued at most once, so O (E) for the second loop. Thus, the total running time for BFS traversal was O (V + E).

### References

The parameters and input files (mediumG.txt and largeG.txt) was provided in the homework assignment (by Dr. Bangalore) and Introduction to Algorithms (3rd ed.) was used to do the lab report.