GTU Department of Computer Engineering
CSE 222/505 - Spring 2021
Homework 8 Report

Hikmet Mete Varol
1801042608

1. SYSTEM REQUIREMENTS

1.Software Specification

Operating System : macOS Catalina, Windows 10

Front End : Eclipse, Sublime Text

Rear End : Oracle SQL

Design Tool : UML

2.Hardware Specification

Processor : x86 processor

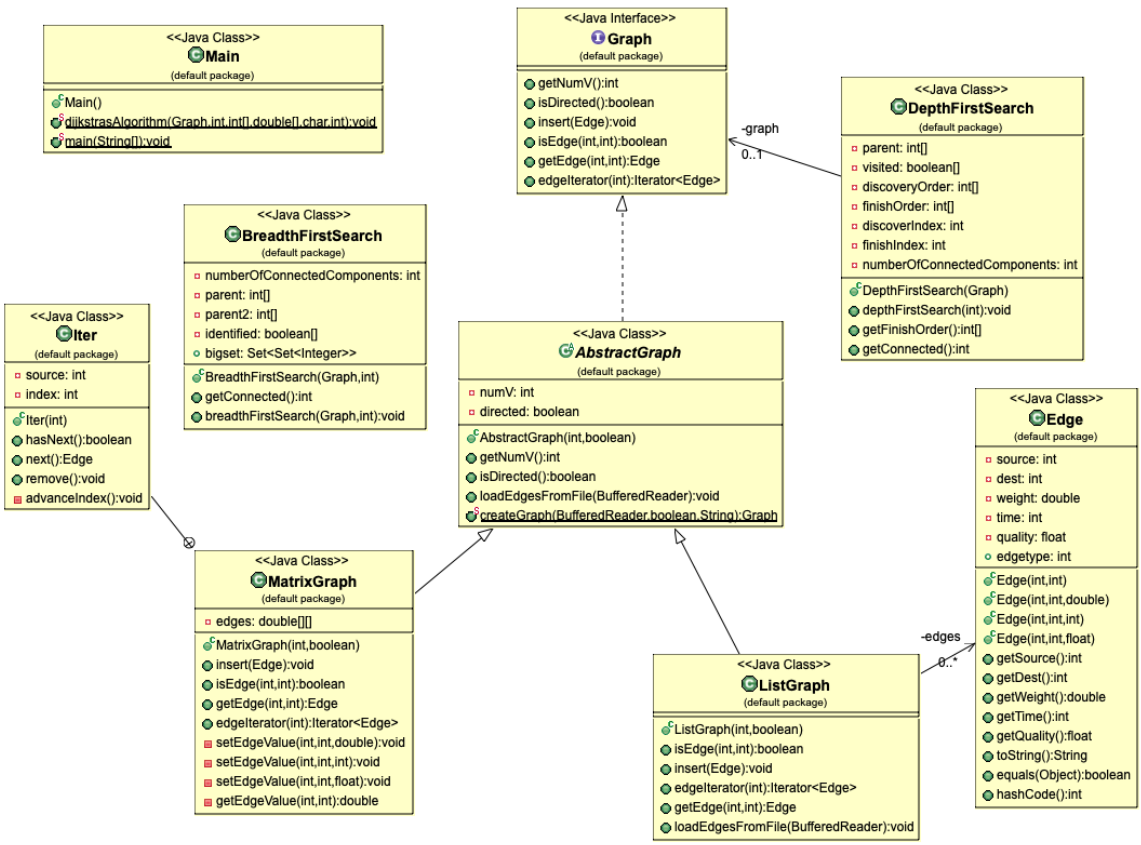
RAM : 512 MB or greater

Hard Disk : 20 GB or greater

3.User Characteristics

Every user; should have basic knowledge of English.

2. CLASS DIAGRAM



3. PROBLEM SOLUTION APPROACH

In the first part, we were asked to improve the Dijkstra's algorithm. I made some additions to the algorithm to make it run efficiently in both graph implementations. Although the first version of the algorithm is more efficient for the matrix graph, it has become efficient for the list graph with additions.

I made additions to the Edge class to keep the information in the graph apart from the weight. The value of quality as a float and time as an integer is stored at the Edge class.

I created an algorithm that works according to various binary operators by using if else statements within the Dijkstra's algorithm.

In the second part, the number of connected components was calculated using bfs and dfs searches. Although the dfs algorithm is simple, I had to use Hashset when counting with bfs. In the result section, the results of the operations were supported by graphics.

4. RUNNING AND RESULTS

```
---- HOMEWORK 8 TEST CASES ----

---- PART 1 ----

-- Q1 --

Directed ListGraph created :   Directed MatrixGraph created :

  S  D   Weight Time Quality     S  D   Weight Time Quality
[(0, 1): 10.0, 1, 1.0]         [(0, 1): 10.0, 1, 1.0]
[(0, 4): 100.0, 1, 1.0]        [(0, 3): 30.0, 1, 1.0]
[(0, 3): 30.0, 1, 1.0]         [(0, 4): 100.0, 1, 1.0]
[(1, 2): 50.0, 1, 1.0]         [(1, 2): 50.0, 1, 1.0]
[(2, 4): 10.0, 1, 1.0]         [(2, 4): 10.0, 1, 1.0]
[(3, 2): 20.0, 1, 1.0]         [(3, 2): 20.0, 1, 1.0]
[(3, 4): 60.0, 1, 1.0]         [(3, 4): 60.0, 1, 1.0]
```

```
Dijkstra's algorithm called for ListGraph :

10.0
50.0
30.0
60.0

Dijkstra's algorithm called for MatrixGraph :

10.0
50.0
30.0
60.0
```

```
Running times for two different type of graph with the same edges :

ListGraph =  397755

MatrixGraph =  28708
```

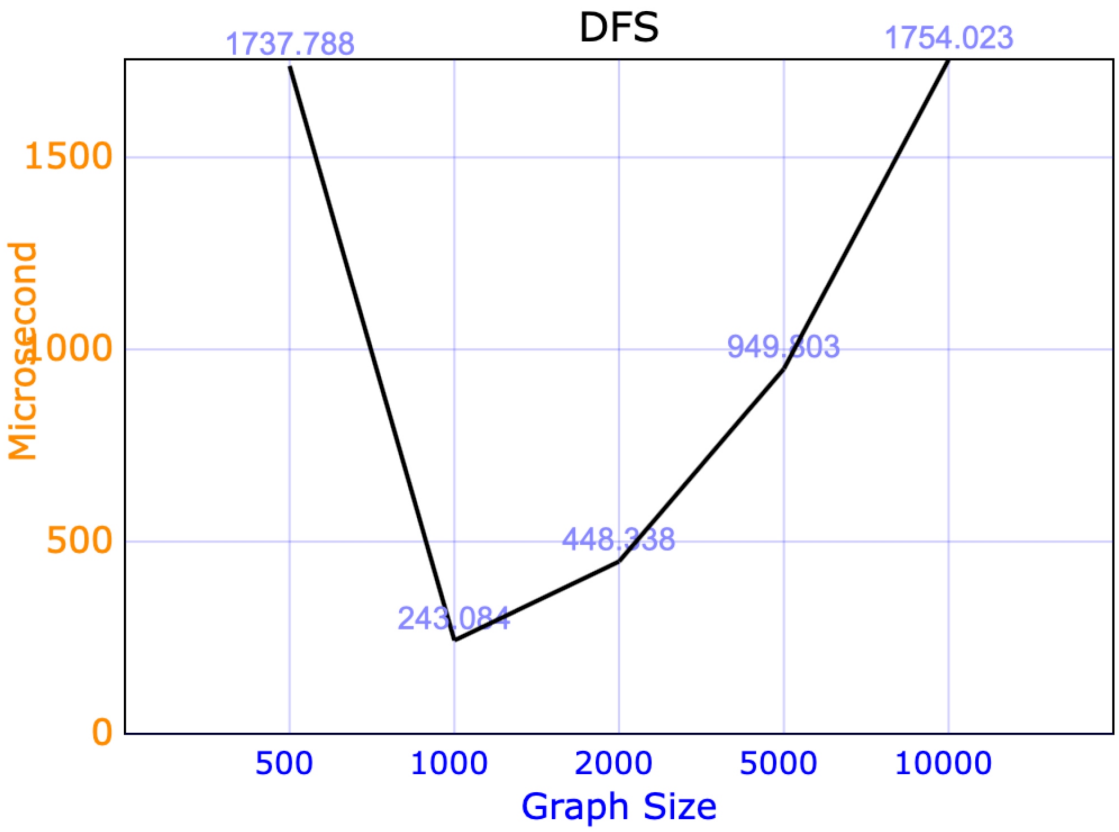
```
-- Q2 --

Directed ListGraph created (Time) :

  S  D   Weight Time Quality
[(0, 1): 1.0, 1, 1.0]
[(1, 2): 1.0, 3, 1.0]
[(1, 3): 1.0, 5, 1.0]
Directed ListGraph created (Quality) :

  S  D   Weight Time Quality
[(0, 1): 1.0, 1, 20.0]
[(1, 2): 1.0, 1, 30.0]
[(1, 3): 1.0, 1, 15.0]
```

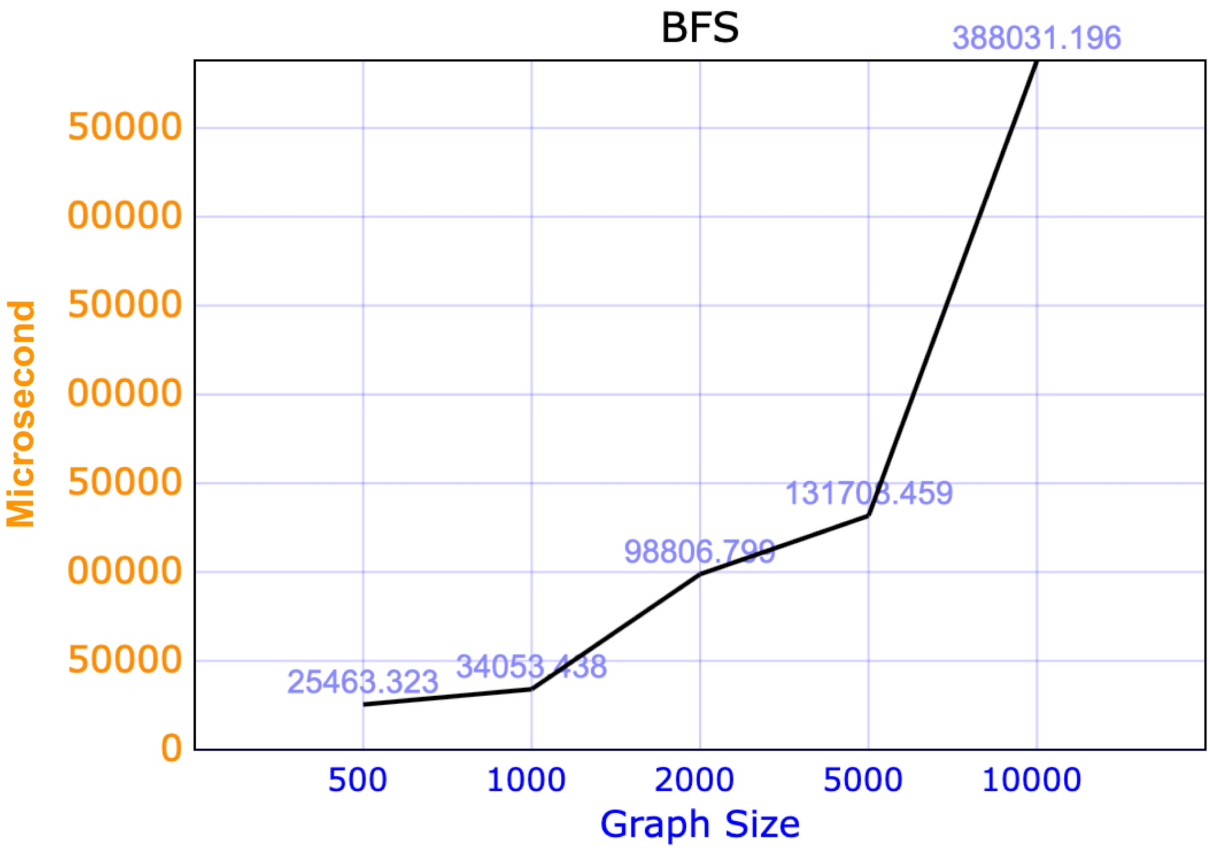
```
Dijkstra's algorithm for Time :

1.0
4.0
6.0

Dijkstra's algorithm for Quality :

20.0
50.0
35.0
```

```
-- Q3 --

Directed ListGraph created :

  S  D   Weight Time Quality
[(0, 1): 10.0, 1, 1.0]
[(0, 4): 100.0, 1, 1.0]
[(0, 3): 30.0, 1, 1.0]
[(1, 2): 50.0, 1, 1.0]
[(2, 4): 10.0, 1, 1.0]
[(3, 2): 20.0, 1, 1.0]
[(3, 4): 60.0, 1, 1.0]

Dijkstra's algorithm for '+' operator :

10.0
50.0
30.0
60.0
```

```
Dijkstra's algorithm for 'x' operator :

10.0
500.0
30.0
100.0

Dijkstra's algorithm for '*' operator :

10.0
-440.0
30.0
-1710.0
```

```
---- PART 2 ----

Check count functions :

DFS = 249
BFS = 249

Results :

500 : BFS = 17849325 DFS = 2042733
1000 : BFS = 29146282 DFS = 542158
2000 : BFS = 102492652 DFS = 452203
5000 : BFS = 137542384 DFS = 1453342
10000 : BFS = 307284479 DFS = 1811625
```


DFS
1737.788
1754.023
1500
1000
500
0
Microsecond
949.803
448.338
243.084
500
1000
2000
5000
10000
Graph Size


BFS
388031.196
50000
00000
50000
00000
50000
00000
50000
0
Microsecond
131703.459
98806.799
34053.438
25463.323
500
1000
2000
5000
10000
Graph Size