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

Yunus Emre Öztürk
1901042619

1. SYSTEM REQUIREMENTS

Non Functional System Requirements:

- 10 MB memory for tests
- Java Runtime Environment

Functional System Requirements:

Generalized Dijkstra:

```
public static int[] dijkstra(GeneralizedGraph g, int start, int property, Oper
atorCreator op, double[] d)
```

To use generalized Dijkstra user must send GeneralizedGraph, vertex number for start, property index to choose weight part, OperatorCreator for create an anonymous class and sent the object of it to use it for wanted operations, double array d for store distances. Operator sent should be associative.

Connected Component Finder:

```
public static int findWithBFS(Graph graph)
```

This static function uses BFS to find connected component count in a graph.

```
public static int findWithDFS(Graph graph)
```

This static function uses DFS to find connected component count in a graph.

Importance Value:

```
public static double[] importanceValueForEach(Graph graph)
```

This method takes graph and looks for importance value and stores it to a double array, indexes of the array represents vertex numbers and answers are importance value of the vertexes.

2. USE CASE AND CLASS DIAGRAMS

3. PROBLEM SOLUTION APPROACH

Generalized Dijkstra:

First of all I thought about how to use many different operators and how to implement more than one weight. Then I created new graph classes to hold more than one weight.

Connected Component Finder:

Main idea to find the different component parts is searching through the unvisited vertices and marks the vertices that visited than incrementing the count of connected component until there is no unvisited vertex.

Importance Value:

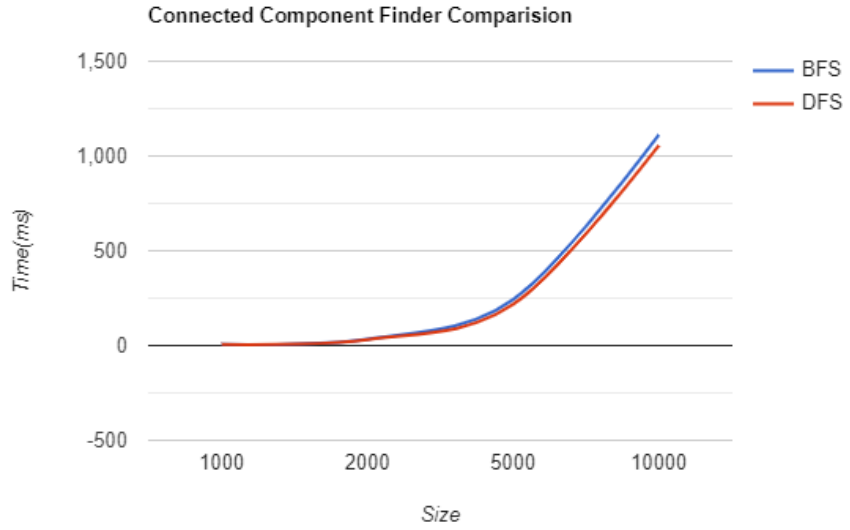
For every vertex an importance value calculation happening. For importance value calculation first of all I used DFS to find Connected component for given vertex and in inner for loops I took start and end vertices for every possible pair of vertex (not symmetric ones) and then make summation for all values and divide the answer by connected component size square.

4. TEST CASES / RUNNING AND RESULTS

```
************Generalized Dijkstra Driver************
MATRIX GRAPH
Test with manuel inputs:
3 weight for each edge and 5 vertex.
Edges: 0<->1 (5.1, 5.0, 5.2), 2<->1(6.2, 6.1, 6.0), 3<->1(7.0, 7.2, 7.1), 4<->1(8.2, 8.0, 8.1)
Printing edges with iterator, takes vertex 1 as starting point:
[(1, 0): 51.1, 50.0, 52.2, ]
[(1, 2): 62.2, 61.1, 60.0, ]
[(1, 3): 70.0, 72.2, 71.1, ]
[(1, 4): 82.2, 80.0, 81.1, ]
Testing this graph with Generalized Dijkstra:
Length of 0th element to remaining elements with add operator (with weight0):
0.0, 51.1, 113.30000000000001, 121.1, 133.3,

Length of 0th element to remaining elements with multiplication operator (with weight0):
0.0, 51.1, 3178.42, 3577.0, 4200.42,

Length of 0th element to remaining elements with star operator (with weight0):
-2509.01, 51.1, -3065.12, -3455.9, -4067.12,

Length of 0th element to remaining elements with add operator (with weight1):
0.0, 50.0, 111.1, 122.2, 130.0,

Length of 0th element to remaining elements with multiplication operator (with weight1):
0.0, 50.0, 3055.0, 3610.0, 4000.0,

Length of 0th element to remaining elements with star operator (with weight1):
-2400.0, 50.0, -2943.9, -3487.8, -3870.0,

Length of 0th element to remaining elements with add operator (with weight2):
0.0, 52.2, 112.2, 123.3, 133.3,

Length of 0th element to remaining elements with multiplication operator (with weight2):
0.0, 52.2, 3132.0, 3711.42, 4233.42,

Length of 0th element to remaining elements with star operator (with weight2):
-2620.44, 52.2, -3019.8, -3588.12, -4100.12,
```

```
LIST GRAPH
Test with manuel inputs:
3 weight for each edge and 5 vertex.
Edges: 0<->1 (5.1, 5.0, 5.2), 2<->1(6.2, 6.1, 6.0), 3<->1(7.0, 7.2, 7.1), 4<->1(8.2, 8.0, 8.1)
Printing edges with iterator, takes vertex 1 as starting point:
[(1, 0): 51.1, 50.0, 52.2, ]
[(1, 2): 62.2, 61.1, 60.0, ]
[(1, 3): 70.0, 72.2, 71.1, ]
[(1, 4): 82.2, 80.0, 81.1, ]
Testing this graph with Generalized Dijkstra:
Length of 0th element to remaining elements with add operator (with weight0):
0.0, 51.1, 113.30000000000001, 121.1, 133.3,

Length of 0th element to remaining elements with multiplication operator (with weight0):
0.0, 51.1, 3178.42, 3577.0, 4200.42,

Length of 0th element to remaining elements with star operator (with weight0):
-2509.01, 51.1, -3065.12, -3455.9, -4067.12,

Length of 0th element to remaining elements with add operator (with weight1):
0.0, 50.0, 111.1, 122.2, 130.0,

Length of 0th element to remaining elements with multiplication operator (with weight1):
0.0, 50.0, 3055.0, 3610.0, 4000.0,

Length of 0th element to remaining elements with star operator (with weight1):
-2400.0, 50.0, -2943.9, -3487.8, -3870.0,

Length of 0th element to remaining elements with add operator (with weight2):
0.0, 52.2, 112.2, 123.3, 133.3,

Length of 0th element to remaining elements with multiplication operator (with weight2):
0.0, 52.2, 3132.0, 3711.42, 4233.42,

Length of 0th element to remaining elements with star operator (with weight2):
-2620.44, 52.2, -3019.8, -3588.12, -4100.12,

Trying to send null to the dijkstra:
Exception has thrown, test passed!
```

```
**********Connected Component Driver**********
----------1000 ELEMENTS GRAPHS----------------
Connected Component Finder with BFS: 6.22704
Connected Component Finder with DFS: 6.2459
----------2000 ELEMENTS GRAPHS----------------
Connected Component Finder with BFS: 34.09937
Connected Component Finder with DFS: 31.26241
----------5000 ELEMENTS GRAPHS----------------
Connected Component Finder with BFS: 242.9111
Connected Component Finder with DFS: 217.3239
----------10000 ELEMENTS GRAPHS----------------
Connected Component Finder with BFS: 1111.16523
Connected Component Finder with DFS: 1055.9157
```



	1000	2000	5000	10000
DFS	6.25 ms	31.26 ms	217.32 ms	1055.92 ms
BFS	6.23 ms	34.10 ms	242.91 ms	1111.17 ms

```
***********Importance Value Driver***********
Test with manuel inputs:
Edges: 0<->1, 1<->2, 0<->3, 3<->2, 0<->4, 4<->2
Should give 0.06 for 0, 0.013 for 1, 0.06 for 2, 0.013 for 3, 0.013 for 4
0.06,
0.013333333333333332,
0.06,
0.013333333333333332,
0.013333333333333332,
Test with 200 element graph with maximum 10 conencted components:
Test passed
Sending null to the method:
Exception has thrown, test passed!
Trying to send directed graph to method:
Exception has thrown, test passed!
```