GIT Department of Computer Engineering

CSE 222/505 - Spring 2021

Homework 8 Report

CANBERK ARICI

171044062

SYSTEM REQUIREMENTS

There is Dijkstra's Algorithm in part1 which includes ListGraph and MatrixGraph representations. These representations are provided because they Ire requested. Also this method supports several properties like distance, time, quality. Besides these functionalities, method supports combining path Iight and edge Iight with different operators like +, * or -.

Generating random graphs with different sizes which are 1000, 2000, 5000, 10000 vertices and various sparsity and making them contain several connected components are provided in part2. Finding number of connected components in graphs are also provided by BFS and DFS both. Detailed results are written on console after run program.

Calculating importance value is supported by dividing the summation result by square of the number of vertices in the connected component including v. Summation is summation of divisions. Numerator of division is the number of shortest paths between any two vertices u and w in the connected component including v and denominator of division is the number of shortest paths betIen any two vertices u and w (in the connected component including v) in which the vertex v is an intermediate vertex on the shortest path from u to w.

Functional Requirements:

dijkstrasAlgorithm:

- Runs efficiently for both ListGraph and MatrixGraph representations of the graph.

- Runs for any specified property of the edges.

- Runs for any specified associative operator.

testPart1:

- Allows user to test dijkstra's generalized algorithm with ListGraph or MatrixGraph representation according to user's choice and also user can choose property of edge and operator.

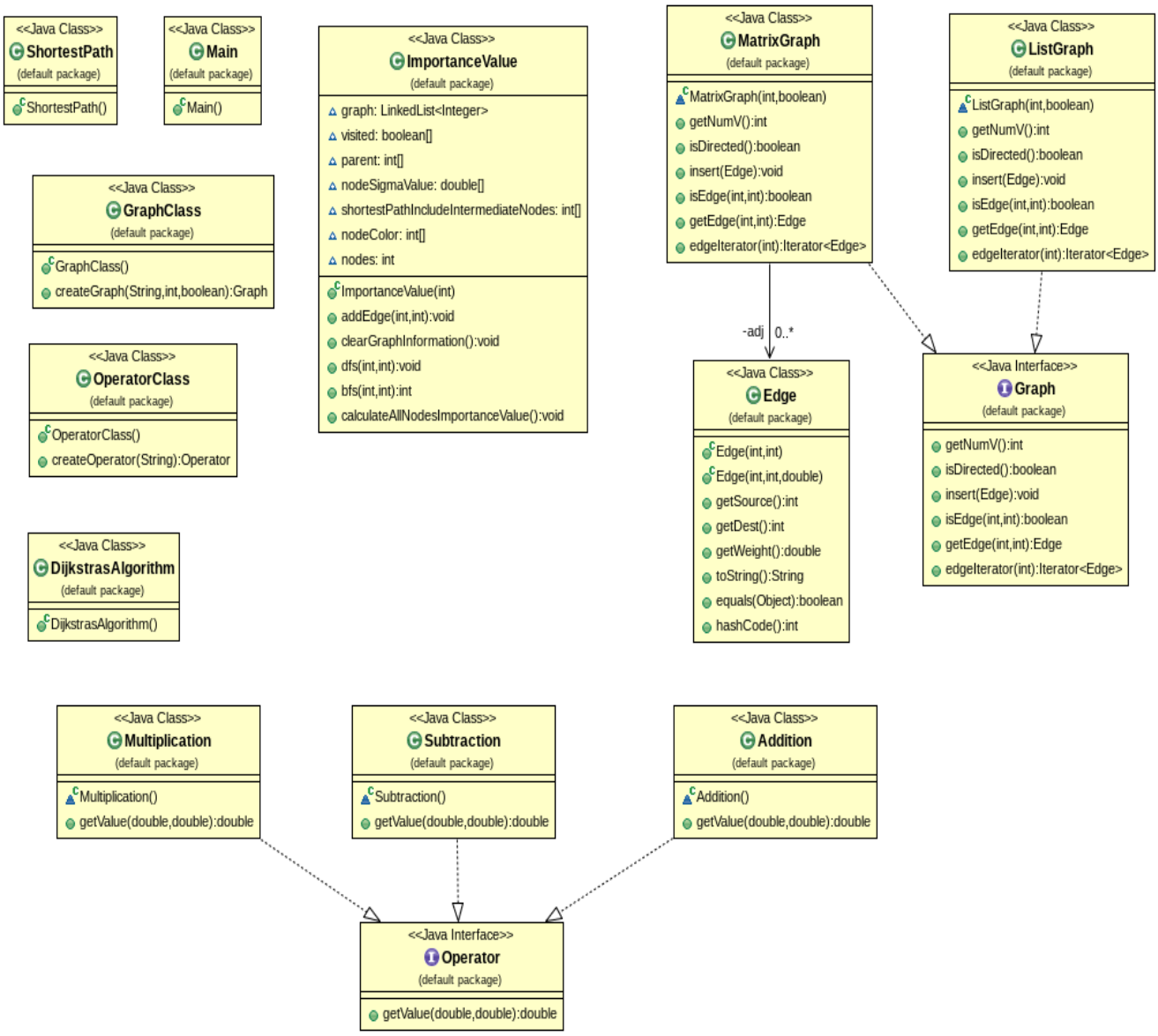
testPart2:

- Generates 10 random graphs with different sizes which are 1000, 2000, 5000, 10000 vertices and various sparsity and making them contain several connected components are provided in part2. Finds number of connected components in graphs are also provided by BFS and DFS both. Prints detailed results on console.

testPart3:

- Allows user to enter graph information and calculate importance value of each vertex of graph that is generated according to given values by use.

CLASS DIAGRAM



PROBLEM SOLUTIONS APPROACH

Part 1 is the most challenging part is to figure out how to make generic solutions that work for both ListGraph and MatrixGraph. At first I figured out there should be one graph interface and both ListGraph and MatrixGraph will implement that. I have implemented design for this and users can decide on the fly which type of graph should be passed in our dijkstra algorithm. I have chosen the same approach of design to make a generic approach while comparing path weight with edge weight where operation can be any specified associative operator.

In part 2, to generate the graph with random number of edges and random node u, v edge. I used Random of java. Second part is very straight forward BFS and DFS algorithm implementation. I generate 10 random graphs for each graph size of 1000, 2000, 5000 and 10000 vertices and various sparsity and they contain several connected components. I collected the running times and put figure of them in report.

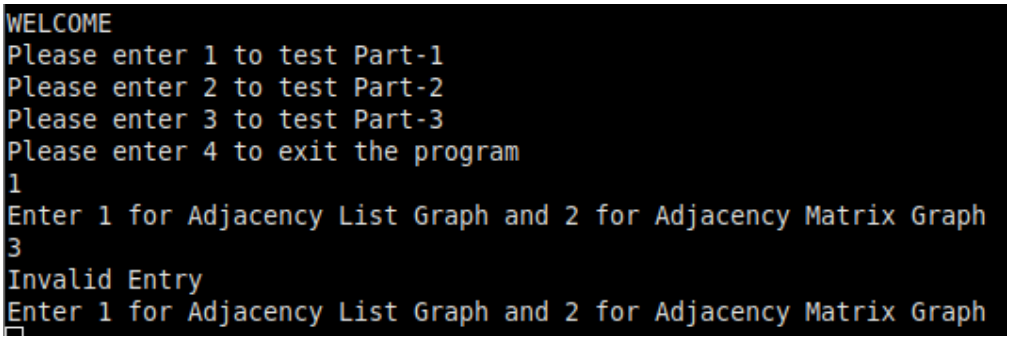
In part 3, there are three parts of the solution which are as follow

- Find out the number of connected node groups in the graph.
- From each of the connected node groups, we need to find out the importance value of a node v. If u, w two arbitrary pairs of nodes then for each vertex V, we need to count the summation of shortest path where v is a part of that and this value should be divided by the number of all possible shortest paths of u, w. Let's say we call this sigma(V) for each arbitrary vertex V of that particular group.
- Finally importance value of vertex V is sigma(V) / N^ 2

To solve the part I used dfs and colored each of the connected components nodes with the same color. I can do this part with BFS too. Recursive dfs implementation is much lesser than BFS implementation, that is one of reasons I pick Dfs as it is easy to implement. In order to find out how many shortest paths exist between arbitrary nodes (u, w) I used BFS for that. If there is ‘r’ possible shortest path between any pair node (u, w) for any arbitrary vertex v, I keep a global variable `ShortestPathIncludeIntermediateNodes` to keep the count of how many path v is present and keep this count for every possible pair(u, w) in a global variable `nodeSigmaValue`. In there is N number of nodes in a connected component groups the importance value of vertex of V is sigma(V) / N^ 2.

TEST CASES

Test Case #	Test Case Description
1	Entering invalid entry(Part1)



Test Case #	Test Case Description
2	Entering a vertex number which is greater than or equal to number of vertices(Part 1)

```
WELCOME
Please enter 1 to test Part-1
Please enter 2 to test Part-2
Please enter 3 to test Part-3
Please enter 4 to exit the program
1
Enter 1 for Adjacency List Graph and 2 for Adjacency Matrix Graph
1
Enter 1 if edges are directed and 2 edges are undirected
2
Enter 1 to combine an edge weight with a path weight with addition operation
Enter 2 to combine an edge weight with a path weight with subtraction operation
Enter 3 to combine an edge weight with a path weight with multiplication operation
1
Enter number of nodes in the graph
4
Enter the start node in the graph
2
Enter number of edges in the graph
2
Please enter property of edge:
1-Distance
2-Time
3-Quality
1
Please enter edge information in a line as such u v and parameter where u and v are nodes and parameter is distance
1 7 2
Invalid Entry
```

Test Case #	Test Case Description
3	Entering a vertex number which is greater than or equal to number of vertices(Part 3)

```
WELCOME
Please enter 1 to test Part-1
Please enter 2 to test Part-2
Please enter 3 to test Part-3
Please enter 4 to exit the program
3
Enter number of nodes in the graph
4
Enter number of edges in the graph
3
Enter edges information in a line as such u v where u and v are nodes
1 5 2
Invalid Entry
```

RUNNING COMMAND AND RESULTS

1) ListGraph representation with multiplication operator according to distance property(Part1)

```
WELCOME
Please enter 1 to test Part-1
Please enter 2 to test Part-2
Please enter 3 to test Part-3
Please enter 4 to exit the program
1
Enter 1 for Adjacency List Graph and 2 for Adjacency Matrix Graph
1
Enter 1 if edges are directed and 2 edges are undirected
2
Enter 1 to combine an edge weight with a path weight with addition operation
Enter 2 to combine an edge weight with a path weight with subtraction operation
Enter 3 to combine an edge weight with a path weight with multiplication operation
3
Enter number of nodes in the graph
5
Enter the start node in the graph
2
Enter number of edges in the graph
3
Please enter property of edge:
1-Distance
2-Time
3-Quality
1
Please enter edge information in a line as such u v and parameter where u and v are nodes and parameter is distance
2 1 3
3 2 4
1 3 2


Distance from 2 to 0 is Infinity
Distance from 2 to 1 is 3.000000
Distance from 2 to 2 is 0.000000
Distance from 2 to 3 is 4.000000
Distance from 2 to 4 is Infinity
```

2) ListGraph representation with subtraction operator according to time property (Part1)

```
WELCOME
Please enter 1 to test Part-1
Please enter 2 to test Part-2
Please enter 3 to test Part-3
Please enter 4 to exit the program
1
Enter 1 for Adjacency List Graph and 2 for Adjacency Matrix Graph
1
Enter 1 if edges are directed and 2 edges are undirected
2
Enter 1 to combine an edge weight with a path weight with addition operation
Enter 2 to combine an edge weight with a path weight with subtraction operation
Enter 3 to combine an edge weight with a path weight with multiplication operation
2
Enter number of nodes in the graph
6
Enter the start node in the graph
0
Enter number of edges in the graph
3
Please enter property of edge:
1-Distance
2-Time
3-Quality
2
Please enter edge information in a line as such u v and parameter where u and v are nodes and parameter is time
1 2 3
2 0 5
4 1 2


Time from 0 to 0 is 0.000000
Time from 0 to 1 is 2.000000
Time from 0 to 2 is 5.000000
Time from 0 to 3 is Infinity
Time from 0 to 4 is 0.000000
Time from 0 to 5 is Infinity
```

3) ListGraph representation with addition operator according to quality (Part1)

```
WELCOME
Please enter 1 to test Part-1
Please enter 2 to test Part-2
Please enter 3 to test Part-3
Please enter 4 to exit the program
1
Enter 1 for Adjacency List Graph and 2 for Adjacency Matrix Graph
1
Enter 1 if edges are directed and 2 edges are undirected
2
Enter 1 to combine an edge weight with a path weight with addition operation
Enter 2 to combine an edge weight with a path weight with subtraction operation
Enter 3 to combine an edge weight with a path weight with multiplication operation
1
Enter number of nodes in the graph
3
Enter the start node in the graph
2
Enter number of edges in the graph
2
Please enter property of edge:
1-Distance
2-Time
3-Quality
3
Please enter edge information in a line as such u v and parameter where u and v are nodes and parameter is quality
1 2 2
0 2 1


Quality from 2 to 0 is 1.000000
Quality from 2 to 1 is 2.000000
Quality from 2 to 2 is 0.000000
```

4) MatrixGraph representation with multiplication operator according to time property(Part1)

```
WELCOME
Please enter 1 to test Part-1
Please enter 2 to test Part-2
Please enter 3 to test Part-3
Please enter 4 to exit the program
1
Enter 1 for Adjacency List Graph and 2 for Adjacency Matrix Graph
2
Enter 1 if edges are directed and 2 edges are undirected
2
Enter 1 to combine an edge weight with a path weight with addition operation
Enter 2 to combine an edge weight with a path weight with subtraction operation
Enter 3 to combine an edge weight with a path weight with multiplication operation
3
Enter number of nodes in the graph
5
Enter the start node in the graph
2
Enter number of edges in the graph
3
Please enter property of edge:
1-Distance
2-Time
3-Quality
2
Please enter edge information in a line as such u v and parameter where u and v are nodes and parameter is time
2 3 1
4 1 3
1 3 4


Time from 2 to 0 is Infinity
Time from 2 to 1 is 4.000000
Time from 2 to 2 is 0.000000
Time from 2 to 3 is 1.000000
Time from 2 to 4 is 12.000000
```

5) MatrixGraph representation with addition operator according to distance property (Part1)

```
WELCOME
Please enter 1 to test Part-1
Please enter 2 to test Part-2
Please enter 3 to test Part-3
Please enter 4 to exit the program
1
Enter 1 for Adjacency List Graph and 2 for Adjacency Matrix Graph
2
Enter 1 if edges are directed and 2 edges are undirected
2
Enter 1 to combine an edge weight with a path weight with addition operation
Enter 2 to combine an edge weight with a path weight with subtraction operation
Enter 3 to combine an edge weight with a path weight with multiplication operation
1
Enter number of nodes in the graph
5
Enter the start node in the graph
3
Enter number of edges in the graph
4
Please enter property of edge:
1-Distance
2-Time
3-Quality
1
Please enter edge information in a line as such u v and parameter where u and v are nodes and parameter is distance
1 2 1
3 4 4
2 3 2
4 2 1


Distance from 3 to 0 is Infinity
Distance from 3 to 1 is 3.000000
Distance from 3 to 2 is 2.000000
Distance from 3 to 3 is 0.000000
Distance from 3 to 4 is 3.000000
```

6) MatrixGraph representation with subtraction operator according to quality property (Part1)

```
WELCOME
Please enter 1 to test Part-1
Please enter 2 to test Part-2
Please enter 3 to test Part-3
Please enter 4 to exit the program
1
Enter 1 for Adjacency List Graph and 2 for Adjacency Matrix Graph
2
Enter 1 if edges are directed and 2 edges are undirected
2
Enter 1 to combine an edge weight with a path weight with addition operation
Enter 2 to combine an edge weight with a path weight with subtraction operation
Enter 3 to combine an edge weight with a path weight with multiplication operation
2
Enter number of nodes in the graph
5
Enter the start node in the graph
3
Enter number of edges in the graph
3
Please enter property of edge:
1-Distance
2-Time
3-Quality
3
Please enter edge information in a line as such u v and parameter where u and v are nodes and parameter is quality
2 1 3
3 1 2
1 3 4


Quality from 3 to 0 is Infinity
Quality from 3 to 1 is 4.000000
Quality from 3 to 2 is 1.000000
Quality from 3 to 3 is 0.000000
Quality from 3 to 4 is Infinity
```

7) Generating 10 random graphs with different sizes which are 1000, 2000, 5000, 10000 vertices and various sparsity and making them contain several connected components are provided in part2. Finding number of connected components in graphs are also provided by BFS and DFS both. Printing detailed results on console.(Part 2)

```
WELCOME
Please enter 1 to test Part-1
Please enter 2 to test Part-2
Please enter 3 to test Part-3
Please enter 4 to exit the program
2
Finding number of components with bfs. Execution time in nanoseconds: 247208
Here for 500 vertices number of connected components is 8
Finding number of components with dfs. Execution time in nanoseconds: 202708
Here for 500 vertices number of connected components is 8
This graph number 1 is with 500 vertices and 1013 edges


Finding number of components with bfs. Execution time in nanoseconds: 528450
Here for 1000 vertices number of connected components is 63
Finding number of components with dfs. Execution time in nanoseconds: 530118
Here for 1000 vertices number of connected components is 63
This graph number 2 is with 1000 vertices and 1468 edges


Finding number of components with bfs. Execution time in nanoseconds: 729373
Here for 2000 vertices number of connected components is 648
Finding number of components with dfs. Execution time in nanoseconds: 1118325
Here for 2000 vertices number of connected components is 648
This graph number 3 is with 2000 vertices and 1393 edges


Finding number of components with bfs. Execution time in nanoseconds: 3819144
Here for 5000 vertices number of connected components is 116
Finding number of components with dfs. Execution time in nanoseconds: 3882410
Here for 5000 vertices number of connected components is 116
This graph number 4 is with 5000 vertices and 9506 edges


Finding number of components with bfs. Execution time in nanoseconds: 2015247
Here for 10000 vertices number of connected components is 8558
Finding number of components with dfs. Execution time in nanoseconds: 2609807
Here for 10000 vertices number of connected components is 8558
This graph number 5 is with 10000 vertices and 1442 edges
```

```
Finding number of components with bfs. Execution time in nanoseconds: 89846
Here for 500 vertices number of connected components is 428
Finding number of components with dfs. Execution time in nanoseconds: 89920
Here for 500 vertices number of connected components is 428
This graph number 6 is with 500 vertices and 72 edges


Finding number of components with bfs. Execution time in nanoseconds: 361590
Here for 1000 vertices number of connected components is 317
Finding number of components with dfs. Execution time in nanoseconds: 369054
Here for 1000 vertices number of connected components is 317
This graph number 7 is with 1000 vertices and 703 edges


Finding number of components with bfs. Execution time in nanoseconds: 846961
Here for 2000 vertices number of connected components is 12
Finding number of components with dfs. Execution time in nanoseconds: 1019078
Here for 2000 vertices number of connected components is 12
This graph number 8 is with 2000 vertices and 5183 edges


Finding number of components with bfs. Execution time in nanoseconds: 641016
Here for 5000 vertices number of connected components is 4281
Finding number of components with dfs. Execution time in nanoseconds: 567144
Here for 5000 vertices number of connected components is 4281
This graph number 9 is with 5000 vertices and 719 edges


Finding number of components with bfs. Execution time in nanoseconds: 3397155
Here for 10000 vertices number of connected components is 1240
Finding number of components with dfs. Execution time in nanoseconds: 2850523
Here for 10000 vertices number of connected components is 1240
This graph number 10 is with 10000 vertices and 11073 edges
```

8) Finding importance values of each vertices of graph that is generated according to given graph(Part-3)

```
WELCOME
Please enter 1 to test Part-1
Please enter 2 to test Part-2
Please enter 3 to test Part-3
Please enter 4 to exit the program
3
Enter number of nodes in the graph
7
Enter number of edges in the graph
5
Enter edges information in a line as such u v where u and v are nodes
3 4
1 6
3 2
2 5
4 6
Importance Value of vertex 0 is 0.000000
Importance Value of vertex 1 is 0.138889
Importance Value of vertex 2 is 0.111111
Importance Value of vertex 3 is 0.083333
Importance Value of vertex 4 is 0.055556
Importance Value of vertex 5 is 0.000000
Importance Value of vertex 6 is 0.000000
```