**GIT Department of Computer Engineering**

**CSE 222/505 - Spring 2021**

**Homework 8 Report**

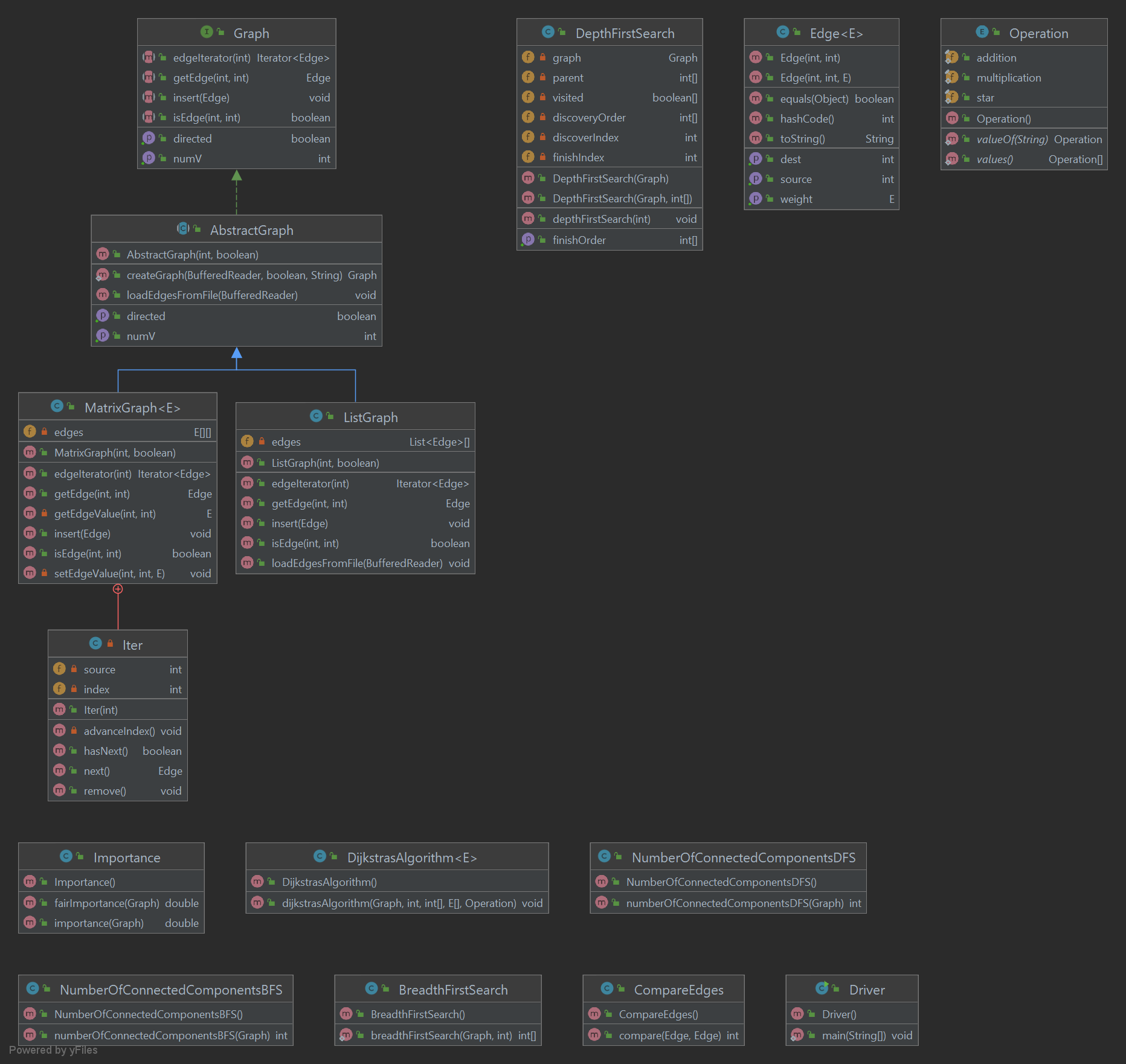
**Sena Erdoğan**

**1901042680**

1. **SYSTEM REQUIREMENTS**

* The system shall provide an algorithm of Dijkstra’s Algorithm that works for both list graphs and matrix graphs.
* Dijkstra’s Algorithm shall work for any type of property the edge carries.
* Dijkstra’s Algorithm shall work for any type of operation that is specified and it shall apply those operations for the appropriate edges and vertices.
* The system shall provide check the running times of the hard coded 40 graphs for both breadth first search and depth first search and it shall let the user know of the running times of each of these 40 graphs for both breadth first search and depth first search.
* The system shall provide an algorithm to compute the sum of the importance values of each vertex of a graph.
* The system shall provide an algorithm to compute the fair importance value of the given graph.

1. **USE CASE AND CLASS DIAGRAMS**



1. **PROBLEM SOLUTION APPROACH**

For the Dijkstra’s Algorithm to work for various types of variables, Edge takes a comparable generic type. Dijkstra’s Algorithm method takes an extra parameter of an enum named Operation that specifies which operation the Dijkstra’s Algorithm will apply on the appropriate edges and vertices. It checks if a graph is a list graph or a matrix graph and applies the corresponding code segment to the graph update the distances.

For the importance value of a graph, if the graph is directed, an UnsupportedOperationException() is thrown. If the graph is undirected, breadth first search is applied to the graph and for each pair of vertices and the values found are added together to find the importance value. The breadth first search returns an integer array and this array’s length gives the number of the shortest paths between the two vertices of the particular vertex pairing. This number is the dividend value for this particular pairing. If the first vertex is the middle element of the returned array, denominator value is incremented by one and so we find the number of paths the first vertex is the middle element in. The dividend value is divided by the denominator value and this value is returns as it is the importance value of the given graph.

For the fair importance value, the importance value is calculated by calling the importance method and the value returned is divided by the square of the number of vertices the graph has and this value is returned as it is the fair importance value of the given graph.

1. **TEST CASES**

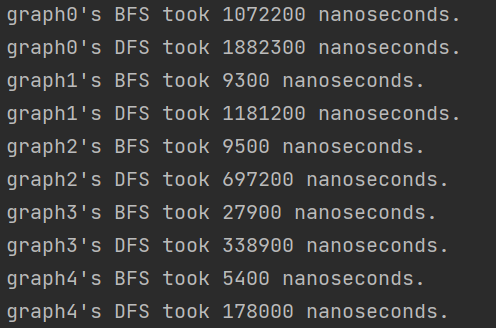
5 list graphs with 1000 vertices, 5 matrix graphs with 1000 vertices, 5 list graphs with 2000 vertices, 5 matrix graphs with 2000 vertices, 5 list graphs with 5000 vertices, 5 matrix graphs with 5000 vertices, 5 list graphs with 10000 vertices and 5 matrix graphs with 10000 vertices are generated. They are all filled with random numbers of edges and these edges start from a random vertex and ends in a random vertex chosen between 0 and 1000. After the initialization of each of these graphs, the breadth first search is performed and the time that passes during this operation is printed in the terminal. After the breadth first search is performed, depth first search is performed similarly and the time that operation takes is printed in the terminal. It can be seen from the results that breadth first search is a lot more efficient than depth first search for list graphs.

Dijkstra’s algorithm is tested on graph39.

Importance value of graph39 is printed in the terminal.

Fair importance value of graph39 is printed in the terminal.

1. **RUNNING AND RESULTS**



|  |  |  |
| --- | --- | --- |
|  | Breadth First Search | Depth First Search |
| graph0 | 1072200 ns | 1882300 ns |
| graph1 | 9300 ns | 1181200 ns |
| graph2 | 9500 ns | 697200 ns |
| graph3 | 27900 ns | 338900 ns |
| graph4 | 5400 ns | 178000 ns |