GTU Department of Computer Engineering
CSE 222/505 - Spring 2022
Homework 8 Report

Ömer Kaan Uslu
1801042642

1. **SYSTEM REQUIREMENTS**

1.Software Specification

Operating System : Windows 10 , macOS Catalina

Front End : Eclipse, Sublime Text

Rear End : Oracle SQL

Design Tool : UML

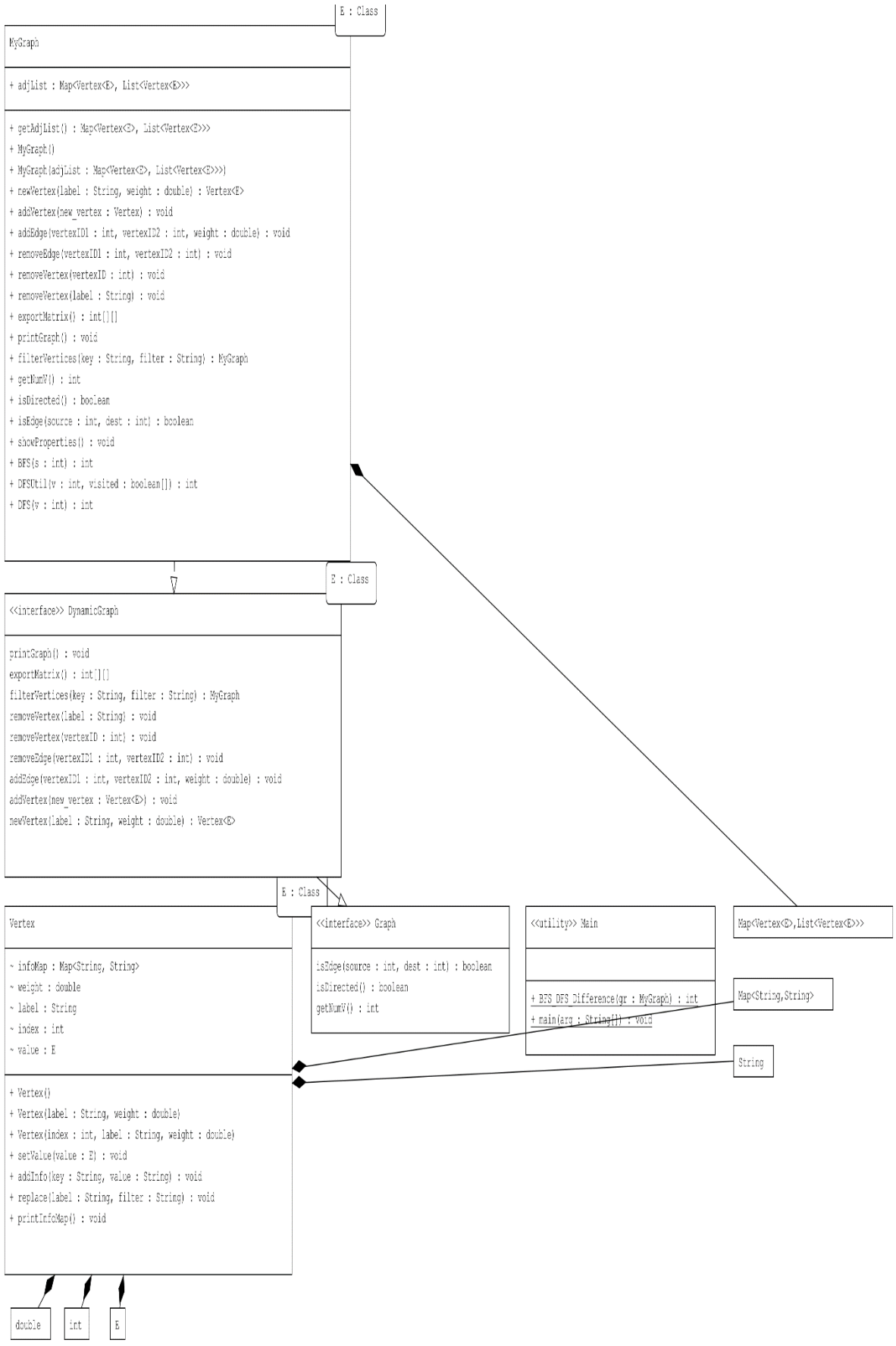
2.Hardware Specification

Processor : x86 processor

RAM : 512 MB or greater

Hard Disk : 20 GB or greater

2. **USE CASE AND CLASS DIAGRAMS**


E : Class
MyGraph
+ adjList : Map<Vertex<E>, List<Vertex<E>>>
+ getAdjList() : Map<Vertex<E>, List<Vertex<E>>>
+ MyGraph()
+ MyGraph(adjList : Map<Vertex<E>, List<Vertex<E>>>)
+ newVertex(label : String, weight : double) : Vertex<E>
+ addVertex(new_vertex : Vertex) : void
+ addEdge(vertexID1 : int, vertexID2 : int, weight : double) : void
+ removeEdge(vertexID1 : int, vertexID2 : int) : void
+ removeVertex(vertexID : int) : void
+ removeVertex(label : String) : void
+ exportMatrix() : int[][]
+ printGraph() : void
+ filterVertices(key : String, filter : String) : MyGraph
+ getNumV() : int
+ isDirected() : boolean
+ isEdge(source : int, dest : int) : boolean
+ showProperties() : void
+ BFS(s : int) : int
+ DFSUtil(v : int, visited : boolean[]) : int
+ DFS(v : int) : int
E : Class
<<interface>> DynamicGraph
printGraph() : void
exportMatrix() : int[][]
filterVertices(key : String, filter : String) : MyGraph
removeVertex(label : String) : void
removeVertex(vertexID : int) : void
removeEdge(vertexID1 : int, vertexID2 : int) : void
addEdge(vertexID1 : int, vertexID2 : int, weight : double) : void
addVertex(new_vertex : Vertex<E>) : void
newVertex(label : String, weight : double) : Vertex<E>
E : Class
Vertex
~ infoMap : Map<String, String>
~ weight : double
~ label : String
~ index : int
~ value : E
+ Vertex()
+ Vertex(label : String, weight : double)
+ Vertex(index : int, label : String, weight : double)
+ setValue(value : E) : void
+ addInfo(key : String, value : String) : void
+ replace(label : String, filter : String) : void
+ printInfoMap() : void
<<interface>> Graph
isEdge(source : int, dest : int) : boolean
isDirected() : boolean
getNumV() : int
<<utility>> Main
+ BFS_DFS_Difference(gr : MyGraph) : int
+ main(arg : String[]) : void
Map<Vertex<E>,List<Vertex<E>>>
Map<String,String>
String
double
int
E

3. **PROBLEM SOLUTION APPROACH AND THEORETICAL RUN TIME ANALYSIS**

In the first question, the goal was to implement a graph. In graph, I used HashMap. HashMap has keys of Vertices and values of List of Vertices. It can be done by setting keys Integers. Since we use adjacency list, in every time adding edge function called, Vertex with VertexID1 is key of corresponding vertex in map, Vertex with VertexID2 is vertex to be added list of corresponding key . In Vertex Class, I add another private HashMap. So User Defined Properties can be done by it easily. HashMap has keys of Strings and values of Strings also.

DFS method is written with recursion, BFS method is written with iteration. Difference of this 2 methods printed with another method in main.

newVertex = O(1)

addVertex = O(1)

addEdge = O(n)

removeEdge = O(n)

removeVertex(Integer) = O(n)

removeVertex (String)= O(n)

exportMatrix = O(n^2)

printGraph = O(n^2)

filterVertices = O(n^2)

4. **TEST CASES**

Test Case No	Test Scenario	Test Steps	Test Data	Expected Result	Actual result	Pass/Fail
1	Add Vertex	Run driver code	Graph,Vertex	Vertex is added to graph	As expected	Pass
2	Remove Vertex	Run driver code	Graph,VertexID	Vertex is removed	As expected	Pass
3	Add Edge	Run driver code	VertexID1,VertexID2	Edge is added.	As expected	Pass

4	Remove Edge	Run driver code	VertexID1,VertexID2	Edge is removed.	As expected	Pass
5	BFS Traversal	Run driver code	Graph	BFS traversal done.	As expected	Pass
6	DFS Traversal	Run driver code	Graph	BFS traversal done.	As expected	Pass

5. **RUNNING AND RESULTS**

```
PRINT GRAPH WITH METHOD


Number of Vertices: 4

 S  D  Weight
[(2,3): 30.0]
[(0,1): 12.0]
[(3,0): 5.0]
[(3,2): 5.0]
[(3,1): 12.0]
[(1,2): 5.0]
```

```
MATRIX VERSION OF GRAPH


0 1 0 0
0 0 1 0
0 0 0 1
1 1 1 0
```

```
PRINT PROPERTIES OF VERTICES


2:   length = 5
0:   length = 5
3:   length = 5
1:   length = 5
```

```
BFS Search      Path
0 1 2 3    ||   47



DFS Search      Path
0 3 1 2    ||   47



 Difference : 0
```

```
AFTER EDGE 3 -> 1 REMOVED


Number of Vertices: 4

 S  D  Weight
[(2,3): 30.0]
[(0,1): 12.0]
[(3,0): 5.0]
[(3,2): 5.0]
[(1,2): 5.0]
```

```
MATRIX VERSION OF THE GRAPH


0 1 0 0
0 0 1 0
0 0 0 1
1 0 1 0
```

```
AFTER VERTEX 3 IS REMOVED


Number of Vertices: 3

 S  D  Weight
[(0,1): 12.0]
[(1,2): 5.0]
```

```
MATRIX VERSION OF THE GRAPH


0 1 0
0 0 1
0 0 0
```

```
LENGTH PROPERTY CHANGED 5 -> 7


 S  D  Weight
[(0,1): 12.0]
[(1,2): 5.0]


PRINT PROPERTIES OF VERTICES


2:   length = 7
0:   length = 7
1:   length = 7
```