AHMET ELBURUZ GÜRBÜZ 150116024
CSE2046 – ANALYSIS OF ALGORITHMS
HOMEWORK 1 – PART 1 – CODING

a) In the beginning of the algorithm, I got the name of the file from the user in command line. I read the file to skip comment lines. I assigned the first number after skipping comment lines to the int numbOfLaptops variable.

I wrote a static class called Node to keep the coordinates and radius together on a regular basis. I created a nodes array of node type with a length of numbOfLaptops. In a for loop, I assign x, y, r to the Nodes where I create the object.In addition I assigned the ID I will use later. So I kept the information of all the coordinates and radius in a Node array. Using the coordinates and radius, I wrote the checkNeigbours method to see if two wireless ranges intersect. If this method returns true, that is, if two coverage areas intersect, I started creating my graph using the addEdge method.

While creating a graph in this project, I used the adjacency list representation. I used HashMap for this. At the beginning of the main class, I created the object because I will put the graph into the hashmap. The use of the Map Interface class. These classes store data according to key-value logic. When inserting an object into the Map, it uses a key that points to this object. When the object wants to get it, this key value is queried and the object is quickly brought among other objects. Therefore , I will keep the edges (neighbors) of each vertex in a linked list .I have a static class called Edge. The purpose of this class is to show at which point the edges of the node end. So this class is required to provide information about the neighbor.

Now let's examine the addEdge method. The first thing in this method is to temporarily create a val value of type LinkedList <Edge> .In the first stage, since this value will be empty, it falls to the else condition and a linkedlist is created here. It adds the dest edge, which is our target in the linkedlist.In the first stage, since this value will be empty, it falls to the else condition and a new linkedlist is created here. Destinition edge is added in our list. Then I match the given key value with object g. Of course, since this is an undirected graph, I did the opposite of the process. After finishing adding the edges by checking the neighbors in the for loop in the main class, our g graph is literally formed.

Now it is time to find the shortest distance from our first corner to all other corners. For this, I use the printShortestDistance method. This method is based on the BFS algorithm.2 arrays are created. With these arrays we will check the distance and whether they are visited. I filled all of these arrays with the value -1 and false. I assigned to the visit of the source vertex to be true and the distance from it to zero. I add my main vertex to queue. I update the distance between them according to a source vertex and indicate that they are visited. When my target and the endVertex of the element in my linkedlist are equal, I say that the route was found and I'm throwing the arraylist that I will print . At the last stage, I write the elements of the arraylist to the output.txt in the correct format.

In summary, I kept all the coordinates in the nodes. I checked if these nodes are neighbours to each other. If they are neighbors, I have each other added neighbors on my graph. I found the shortest path with the BFS algorithm in the graph created by the adjacency list. I took the shortest paths to the arraylist and finally I printed this arraylist in the appropriate format.

b)Time Complexity: Time complexity of an algorithm quantifies the amount of time taken by an algorithm to run as a function of the length of the input. Since I have 1 for loops in the main class and printShortestPath method O(n^2) so the time complexity of my algorithm is Θ(n^3).

Space Complexity: Space complexity of an algorithm quantifies the amount of space or memory taken by an algorithm to run as a function of the length of the input.

An in-place algorithm transforms the input without using any extra memory. As the algorithm executes, the input is usually overwritten by the output and no additional space is needed for this operation. Similarly, an algorithm which is not in-place is called not-in-place or out-of-place algorithm. Unlike an in-place algorithm, the extra space used by an out-of-place algorithm depends on the input size.

Since i am using linked lists , hashmap and array lists and working on them , the space complexity of my code is not in-place.

test1.txt

```
## Lines beginning with # denote comments
## The correct output for this input is
## 0
## 1
## 2
3
0	0	0.666
1	0	0.666
2	0	0.666
```

output.txt

```
0
1
2
```

test2.txt

```
## Lines beginning with # denote comments
## Three agents at the corners of a right triangle
## The correct output for this input is
## 0
## 2
## 1
3
0	5	2
4	2	2
0	2	3
```

output.txt

```
0
2
1
```

test3.txt

```
## Lines beginning with # denote comments
## The correct output for this input is
## 0
## 1
## 3
## 2
## 2
5
1       4       1
1       2       2
5.5     2       1
1       0       1
4       2       1
```

output.txt

```
0
1
3
2
2
```

test4.txt

```
## Lines beginning with # denote comments
## The correct output for this input is
## 0
## 0
## 0
## 0
4
0       3       0.5
0       0       0.5
1       0       0.666
2       0       0.666
```

output.txt

```
0
0
0
0
```

test5.txt

```
9
1       1       1
2       2       1
4       1.5     1.5
4       3       1
6       3       2
8       4       2
2       7       2
4       9       1
6.5     5.5     1.5
```

output.txt

```
0
1
2
3
3
4
0
0
4
```

test6.txt

```
15
1       1       1
2       2       1
3       3       1
4       4       1
5.2     5.2     1
3.2     1       1
4.5     1.2     1
4.5     3       1
7       4.5     1
6       1       1
7       2       1
8       3       1
9.5     3       1
9       1       1
11      3       1
```

output.txt

```
0
1
2
3
4
2
3
3
5
4
5
6
7
0
8
```