CSE222-HW8 REPORT

Elifnur Kabalcı
1801042617

ABOUT SEE FILES:

In the location I have sent in the file, vs code also sees txt files on my computer. That's why I sent it that way.

GRAPH CLASS

This class is my main class. Main is here. I get filename from user and generate map object. I'm checking if the start or end points are not 0. Thus, if there is such an error, the program closes without running at all. Then the graph object is generated and the map object is embedded in the graph. I have defined the edge arraylist. My initial thought was to keep the 0 coordinates at the edge and run the algorithms accordingly, but I could not integrate the algorithms that should be used. So I assigned map I in the map object to edge. Then I ran the algorithms one by one. While BFs was running, I commented out Dikstra's lines. Thus, I tried to reduce the time that occurs in Dijkstra. First of all, I printed the shortestPath coordinates to the screen for testing. Then I transferred this data to txt files containing outputs. With the help of a filewriter, I transferred the name to the file being read and the file specific to the method used.

MAP CLASS

Here, I read the file requested by the user to be examined and transferred it to a data storage tool. In this part, I used 2d int array. I also recorded the start and end positions in this section.

BREATHFIRST CLASS

I sent the start coordinate, end coordinate and map I used to this class. Here is the method written for the path finding algorithm.

First of all, I defined a 2d array to hold the previous nodes and filled it with -1. Thus, if there is no data entry, we can understand that nothing happened before. We assigned the start to the queue we defined and gave -2 to the previous one. What makes this special is to make the program stop when it sees the node whose previous u was -2. We checked if it was valid by visiting the queue one by one and checking its neighbors. Thus, the design of the queue was completed. Then I tried to find the shortest path from this queue. For this, the while in control statement became the predetermined number of -2.

DIJKSTRA CLASS

This is the part I had the most difficulty with. Because no matter how hard I tried, instead of printing the whole path, it went to the direct end node and did not create a path. The main thing in this algorithm is to find the values with the minimum distance between the given coordinates. I didn't say the ready node patterns first, but it didn't connect the currentnode to the previousnode. I tried removing the node build completely but got the same result in it. Then I defined the node structure myself and implemented the data myself. This time the previousnode worked fine.

TEXTTOIMAGE CLASS

In this part, I used more ready-made molds. While creating the object, I sent the positions of the txt and png as parameters. Then I sent the shortest path list to convert it to png with the turned method and draw a path.

OUTPUT GENERATION

I have prepared output files. These files show the coordinates of the path followed. I got the filename from the user. When we write “map01”, “map01 + (name of search algorithm) + output.txt” is given. I did not give the file name or output name in main. All can be changed. I give the coordinate values in the file. So I also suppressed the number of steps as output to the terminal. And for each txt file I printed it to the screen. Since the files from 1 to 10 are 500 in size, I continued by defining direct 500, however. When I switched to specific maps, I used 1000 instead of 500 in the definition in the map class because of their size.

BFS OUTPUTS:

```
Enter a string: map01.txt
Shortest Path:
BFS # of steps: 1332
```

```
Enter a string: map02.txt
Shortest Path:
BFS # of steps: 820
```

```
Enter a string: map03.txt
Shortest Path:
BFS # of steps: 1017
```

```
Enter a string: map04.txt
Shortest Path:
BFS # of steps: 865
```

```
Enter a string: map05.txt
Shortest Path:
BFS # of steps: 930
```

```
Enter a string: map06.txt
Shortest Path:
BFS # of steps: 798
```

```
Enter a string: map07.txt
Shortest Path:
BFS # of steps: 982
```

```
Enter a string: map08.txt
Shortest Path:
BFS # of steps: 970
```

```
Enter a string: map09.txt
Shortest Path:
BFS # of steps: 1173
```

```
Enter a string: map10.txt
Shortest Path:
BFS # of steps: 642
```

```
Enter a string: pisa.txt
Shortest Path:
BFS # of steps: 2034
```

```
Enter a string: tokyo.txt
Shortest Path:
BFS # of steps: 1213
```

```
Enter a string: triumph.txt
Shortest Path:
BFS # of steps: 1543
```

```
Enter a string: vatican.txt
Shortest Path:
BFS # of steps: 2000
```

DIJKSTRA OUTPUTS:

```
Enter a string: map01.txt
Shortest Path:
Dijikstra # of steps: 1332
```

```
Enter a string: map02.txt
Shortest Path:
Dijikstra # of steps: 820
```

```
Enter a string: map03.txt
Shortest Path:
Dijikstra # of steps: 1017
```

```
Enter a string: map04.txt
Shortest Path:
Dijikstra # of steps: 865
```

```
Enter a string: map05.txt
Shortest Path:
Dijikstra # of steps: 930
```

```
Enter a string: map06.txt
Shortest Path:
Dijikstra # of steps: 798
```

```
Enter a string: map07.txt
Shortest Path:
Dijikstra # of steps: 982
```

```
Enter a string: map08.txt
Shortest Path:
Dijikstra # of steps: 970
```

```
Enter a string: map09.txt
Shortest Path:
Dijikstra # of steps: 1173
```

```
Enter a string: map10.txt
Shortest Path:
Dijikstra # of steps: 642
```

```
Enter a string: pisa.txt
Shortest Path:
Dijikstra # of steps: 1130
```

```
Enter a string: tokyo.txt
Shortest Path:
Dijikstra # of steps: 1157
```

```
Enter a string: tokyo.txt
Shortest Path:
Dijikstra # of steps: 1157
```

```
Enter a string: vatican.txt
Shortest Path:
Dijikstra # of steps: 1826
```

Running Times Outputs:

```
Enter a filename: map01.txt
BFS # of steps: 1332
Running time: 91 milliseconds
Dijikstra # of steps: 1332
Running time: 60 milliseconds
```

```
Enter a filename: map02.txt
BFS # of steps: 820
Running time: 47 milliseconds
Dijikstra # of steps: 820
Running time: 54 milliseconds
```

```
Enter a filename: map03.txt
BFS # of steps: 1017
Running time: 43 milliseconds
Dijikstra # of steps: 1017
Running time: 79 milliseconds
```

```
Enter a filename: map04.txt
BFS # of steps: 865
Running time: 53 milliseconds
Dijikstra # of steps: 865
Running time: 52 milliseconds
```

```
Enter a filename: map05.txt
BFS # of steps: 930
Running time: 45 milliseconds
Dijikstra # of steps: 930
Running time: 51 milliseconds
```

```
Enter a filename: map06.txt
BFS # of steps: 798
Running time: 42 milliseconds
Dijikstra # of steps: 798
Running time: 51 milliseconds
```

```
Enter a filename: map07.txt
BFS # of steps: 982
Running time: 91 milliseconds
Dijikstra # of steps: 982
Running time: 53 milliseconds
```

```
Enter a filename: map08.txt
BFS # of steps: 970
Running time: 42 milliseconds
Dijikstra # of steps: 970
Running time: 53 milliseconds
```

```
Enter a filename: map09.txt
BFS # of steps: 1173
Running time: 52 milliseconds
Dijikstra # of steps: 1173
Running time: 65 milliseconds
```

```
Enter a filename: map10.txt
BFS # of steps: 642
Running time: 42 milliseconds
Dijikstra # of steps: 642
Running time: 38 milliseconds
```

```
Enter a filename: pisa.txt
BFS # of steps: 1130
Running time: 262 milliseconds
Dijikstra # of steps: 1130
Running time: 240 milliseconds
```

```
Enter a filename: tokyo.txt
BFS # of steps: 1157
Running time: 193 milliseconds
Dijikstra # of steps: 1157
Running time: 187 milliseconds
```

```
Enter a filename: triumph.txt
BFS # of steps: 1543
Running time: 223 milliseconds
Dijikstra # of steps: 1543
Running time: 300 milliseconds
```

```
Enter a filename: vatican.txt
BFS # of steps: 1826
Running time: 205 milliseconds
Dijikstra # of steps: 1826
Running time: 213 milliseconds
```

Map01-> dijkstra, map02->bfs, map03-> bfs, map04->dijkstra, map05->bfs, map06->bfs, map07->dijktra, map08->bfs, map09->bfs, map10->dijkstra, pisa->bfs, Tokyo->dijkstra, triumph->bfs, Vatican->bfs

There are best solutions for my code. In general, according to the given maps and my code, there are 9 bfs and 5 dijkstra. Based on this, we can say that bfs does a better job on these maps.

File Views(png files , BFS output coordinates, Dijkstra output coordinates, original txt files)

.vscode	5/29/2023 10:36 AM	File folder	
bin	5/29/2023 2:58 PM	File folder	
lib	5/29/2023 10:36 AM	File folder	
src	6/3/2023 2:06 AM	File folder	
map02.png	6/3/2023 2:28 AM	PNG File	1 KB
map03.png	6/3/2023 2:28 AM	PNG File	1 KB
map04.png	6/3/2023 2:28 AM	PNG File	1 KB
map05.png	6/3/2023 2:29 AM	PNG File	1 KB
map06.png	6/3/2023 2:29 AM	PNG File	1 KB
map07.png	6/3/2023 2:29 AM	PNG File	1 KB
map08.png	6/3/2023 2:29 AM	PNG File	1 KB
map09.png	6/3/2023 2:29 AM	PNG File	1 KB
map10.png	6/3/2023 2:29 AM	PNG File	1 KB
README.md	4/27/2023 11:06 PM	MD File	1 KB

graph	6/3/2023 2:29 AM	File folder	
map01.txt	5/29/2023 10:55 AM	Text Document	489 KB
map01BFSoutput.txt	6/2/2023 5:44 PM	Text Document	13 KB
map01DIJKSTRAoutput.txt	6/3/2023 2:21 AM	Text Document	13 KB
map02.txt	5/29/2023 10:56 AM	Text Document	489 KB
map02BFSoutput.txt	6/2/2023 5:44 PM	Text Document	8 KB
map02DIJKSTRAoutput.txt	6/3/2023 2:28 AM	Text Document	8 KB
map03.txt	5/29/2023 10:57 AM	Text Document	489 KB
map03BFSoutput.txt	6/2/2023 5:44 PM	Text Document	10 KB
map03DIJKSTRAoutput.txt	6/3/2023 2:28 AM	Text Document	10 KB
map04.txt	5/29/2023 10:57 AM	Text Document	489 KB
map04BFSoutput.txt	6/2/2023 5:44 PM	Text Document	8 KB
map04DIJKSTRAoutput.txt	6/3/2023 2:28 AM	Text Document	8 KB
map05.txt	5/29/2023 10:58 AM	Text Document	489 KB
map05BFSoutput.txt	6/2/2023 5:44 PM	Text Document	9 KB
map05DIJKSTRAoutput.txt	6/3/2023 2:29 AM	Text Document	9 KB
map06.txt	5/29/2023 10:59 AM	Text Document	489 KB
map06BFSoutput.txt	6/2/2023 5:44 PM	Text Document	8 KB
map06DIJKSTRAoutput.txt	6/3/2023 2:29 AM	Text Document	8 KB
map07.txt	5/29/2023 10:59 AM	Text Document	489 KB
map07BFSoutput.txt	6/2/2023 5:44 PM	Text Document	10 KB
map07DIJKSTRAoutput.txt	6/3/2023 2:29 AM	Text Document	10 KB

map08.txt	5/29/2023 11:00 AM	Text Document	489 KB
map08BFSoutput.txt	6/2/2023 5:45 PM	Text Document	10 KB
map08DIJKSTRAoutput.txt	6/3/2023 2:29 AM	Text Document	10 KB
map09.txt	5/29/2023 11:01 AM	Text Document	489 KB
map09BFSoutput.txt	6/2/2023 5:45 PM	Text Document	11 KB
map09DIJKSTRAoutput.txt	6/3/2023 2:29 AM	Text Document	11 KB
map10.txt	5/29/2023 11:01 AM	Text Document	489 KB
map10BFSoutput.txt	6/2/2023 5:45 PM	Text Document	7 KB
map10DIJKSTRAoutput.txt	6/3/2023 2:29 AM	Text Document	7 KB
pisa.txt	5/29/2023 11:02 AM	Text Document	1,954 KB
pisaBFSoutput.txt	6/2/2023 5:51 PM	Text Document	20 KB
pisaDIJKSTRAoutput.txt	6/3/2023 2:31 AM	Text Document	11 KB
tokyo.txt	5/29/2023 11:02 AM	Text Document	1,954 KB
tokyoBFSoutput.txt	6/2/2023 5:51 PM	Text Document	12 KB
tokyoDIJKSTRAoutput.txt	6/3/2023 2:03 AM	Text Document	12 KB
triumph.txt	5/29/2023 11:02 AM	Text Document	1,954 KB
triumphBFSoutput.txt	6/2/2023 5:47 PM	Text Document	15 KB
triumphDIJKSTRAoutput.txt	6/3/2023 2:03 AM	Text Document	15 KB
vatican.txt	5/29/2023 11:03 AM	Text Document	1,954 KB
vaticanBFSoutput.txt	6/2/2023 5:52 PM	Text Document	19 KB
vaticanDIJKSTRAoutput.txt	6/3/2023 2:03 AM	Text Document	18 KB