

GTU Department of Computer Engineering

CSE 222/505 - Spring 2023

Homework 8

Due date: June 4, 2023 – 23:59

You are asked to design and implement a path planning algorithm to find feasible paths for given maps in the homework.

PROBLEM DEFINITION


The path planning problem is the task of finding an optimal or feasible path from a starting point to a goal point in each map while avoiding obstacles. The optimal path is the shortest path between the starting point and the end point on the map. In the homework, you are responsible for finding **feasible path(s)** on the maps. If you can find the **optimal path(s)**, you will get extra points for each map.

INPUT

10 random maps and 4 city maps are given in the homework. Each map is represented in a text file with the following information. The number of lines n represents the length of x and y coordinates of the map.

- The first line: y, x coordinates of the starting point
- The second line: y, x coordinates of the end point
- In each line, 0 and 1 values are given separated by comma. Value 1 represents an obstacle region whereas value 0 represents unoccupied region.
- Each line k represents the k th y coordinate. In each line, starting from the 0th coordinate to the n th coordinate of x , each coordinate (y, x) is represented by a value 0 or 1. If the value in a (y, x) coordinate is 1, it means that the (y, x) coordinate has an obstacle and the feasible path for the map cannot have the (y, x) coordinate.
- Some maps have -1 instead of 1 in their coordinates. Replace them with 1s.

You are responsible to read the text files to create a CSE222Map object which has a start point, an end point, and an $n \times n$ matrix to keep 0/1 value.

. You can convert the text files to their Png files with **any Java library** that you prefer. You will need the Png files of the maps in order to draw their feasible paths that you find with your algorithm.

ALGORITHM

To obtain a feasible path for each map between its starting and end points, we will design an algorithm with following components:

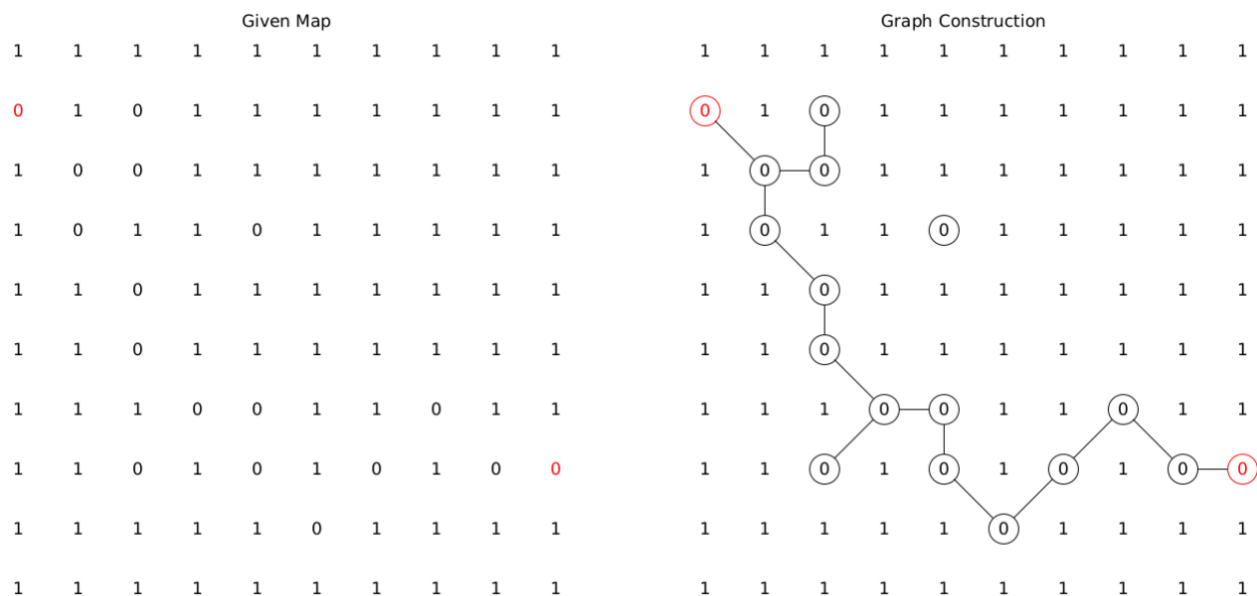
- **Step1:** Construct a graph from the given map:
 - **CSE222Graph graph = new CSE222Graph (CSE222Map map)**
- **Step2.1:** Apply *Dijkstra's Algorithm* to obtain a feasible path from the graph:

Step1: Construct a graph from the given map:

To apply Dijkstra's algorithm to find a feasible path for a map, the map should be represented as a graph. Each node in the graph represents each (y, x) coordinate whose value is 0 in the map. If a coordinate has 1 value, the coordinate cannot exist on the graph. A node whose coordinate (x, y) should be connected with another node if its coordinate is one of the following values:

- $(y-1, x)$
- $(y+1, x)$
- $(y, x-1)$
- $(y, x+1)$
- $(y-1, x-1)$
- $(y-1, x+1)$
- $(y+1, x+1)$
- $(y+1, x-1)$

For example, a 10 x 10 map is given. The starting point and end point of the map are $(0, 1)$ and $(9, 7)$, respectively. Using the map, a graph is constructed. The coordinates whose values are 0 become nodes in the graph. The edges between the nodes are determined by the rule explained above.



Step2.1: Apply Dijkstra's algorithm to The Graph

You can find the implementation of Dijkstra's algorithm in your course textbook. You need to adapt this implementation to your homework. Your Dijkstra's algorithm should consider the graph obtained to find a feasible path for the given map.

Dijkstra's Algorithm

1. Initialize S with the start vertex, s , and $V-S$ with the remaining vertices.
2. **for** all v in $V-S$
3. Set $p[v]$ to s .
4. **if** there is an edge (s, v)
5. Set $d[v]$ to $w(s, v)$.
6. **else**
7. Set $d[v]$ to ∞ .
8. **while** $V-S$ is not empty
9. **for** all u in $V-S$, find the smallest $d[u]$.
10. Remove u from $V-S$ and add u to S .
11. **for** all v adjacent to u in $V-S$
12. **if** $d[u] + w(u, v)$ is less than $d[v]$
13. Set $d[v]$ to $d[u] + w(u, v)$.
14. Set $p[v]$ to u .

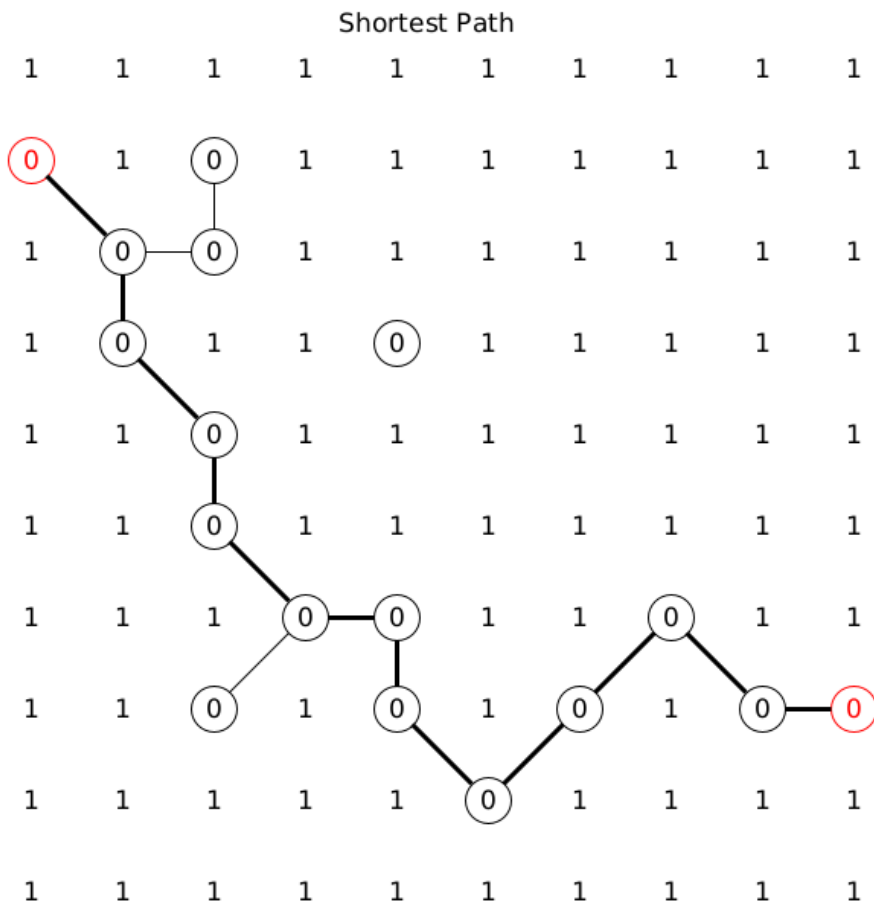
Step2.2: Apply Breadth-First Search algorithm to The Graph

You can find the implementation of BFS algorithm in your course textbook. You need to adapt this implementation to your homework. Your BFS algorithm should consider the graph obtained to find a feasible path for the given map.

Algorithm for Breadth-First Search

1. Take an arbitrary start vertex, mark it identified (color it light gray), and place it in a queue.
2. **while** the queue is not empty
3. Take a vertex, u , out of the queue and visit u .
4. **for** all vertices, v , adjacent to this vertex, u
5. **if** v has not been identified or visited
6. Mark it identified (color it light gray).
7. Insert vertex v into the queue.
8. We are now finished visiting u (color it dark gray).

For example, when one of the path planning algorithms is applied on the example graph, the path will be as follows.



The distance between the starting point and the end point is 12.

Step3: Draw the feasible path on the Png file of the map.

After you obtain the set of the coordinates for a feasible path for the given map, you need to draw a line using these coordinate points on the Png version of the map. **Any drawing Java library is allowed to use.**

MAIN CLASS OF THE HOMEWORK

Your main class basically includes these lines to evaluate by me efficiently. Of course, there will be more lines and extra functions in the main class, but I want to find these keywords in your class. You can create Subdivision method either in a new class or a function under CSE222Graph class.

```
String InputFile = "Map01.txt"
```

```
int X_LENGTH = 500
```

```
int Y_LENGTH = 500
```

```
CSE222Map Map = new CSE222Map (InputFile)
```

```
CSE222Graph Graph = new CSE222Graph (Map)
```

```
CSE222Dijkstra Dijkstra = new CSE222Dijkstra (Graph)
```

```
List DijkstraPath = Dijkstra.findPath()
```

```
CSE222BFS BFS= new CSE222BFS (Graph)
```

```
List BFSPath = BFS.findPath()
```

```
Map.convertPNG()
```

```
Map.drawLine(DijkstraPath)
```

```
Map.drawLine(BFSPath)
```

```
Map.writePath(BFSPath)
```

```
Map.writePath(DijkstraPath)
```

```
System.out.println("Dijkstra Path: "+ Dijkstra.length)
```

```
System.out.println("BFS Path: "+ BFS.length)
```

REPORT

Your report should include complexity analysis and running time performance of the two algorithms.

GENERAL RULES

- If your algorithm cannot find a feasible path, it should print "No feasible path is found."
- Do not use static variables.
- Write comments on your code.
- Create output PNG files in the directory where your source code exists.
- If you cannot find any feasible paths in your homework, you will get 0.

GRADING

- Reading the input files (5 points).
- Constructing a graph of the given map (20 points).
- Applying Dijkstra's algorithm on a graph to find a feasible path (20 points).
- Applying BFS algorithm on a graph to find a feasible path (15 points).
- Drawing the obtained paths on maps (20 points).
- Writing the coordinates of paths into text file with giving format (10 points).
- Conducting an experimental study on report (10 point).

CONTACT

Gizem Süngü

