

Report

Q1– describe what I code does and doesn't do.

Currently, I have achieved all requirements of completion core part and (15/20) challenge part which one of them is about taking into account restrictions information, other one is that allow user to switch fastest path or shortest path for navigation. But, I have not done the traffic light part.

Q2– give a detailed pseudocode algorithm for A search.*

Step1: Create a set (visitedNodeSet) to store all visited node

Step2: Create a priority(fringe) queue to orderly small cost of A search node

Step3: fringer offer the navigating start node

Step4:

BEGIN Loop(Finger is not empty)

 Fringe poll a smallest cost node
 visitedNodeSet add the node

 IF the polled node is the target node

 BEGIN Loop(the parent node of polled node exist)

 List of shortest path to store the segment
 between this node and its parent node

 Assign this node to its parent node to
 continue loop

 END Loop

 Reverse the elements of the list

 Return the list

 END IF

BEGIN LOOP(iteratively go through all linked segment from the start node)

Check the constrains of the segment

(Check if the segment is for car

And check if the segment is for one way

And if the startNode of the segment is restriction)

IF (the end node of the segment is NOT visited yet)

Transfer the end node to A search node type

SET isOfFringe To false;

BEIGIN loop (iteratively go through all A search nodes of priority queue)

IF (the end node is in the priority queue ahead)

IF(start cost of end node < the cost of same node of priority)

The same node of the fringe of parent node is assigned
to the end node of parent node

The same node of the fringe update edge

The same node update the G cost from start

END IF

SET isOfFringe To true;

Break;

END IF

END BEIGIN

IF (end node of segment is not in fringe)

Fringe offer the end node

END IF

END IF

END LOOP

END LOOP

Step 5: Return new empty ArrayList , if does not find the shortest path

Q3–describe your path cost and heuristic estimate

Path cost = Gcost from start node + Heuristic cost to Target Node

Gcost From start =

the length of segment of between current node and its parent node +

Gcost from start of its parent node

Heuristic cost to target =

$\text{Math.hypot}(\text{current.x} - \text{targetNode.x}, \text{current.y} - \text{targetNode.y})$

Q4—give a detailed pseudocode algorithm for the articulation points

Step1:

```
BEGIN LOOP(literately go through all nodes of the graph)
    Connect every node with their neighbours
END Loop
```

Step2:

Create a set of node to collect all unvisited nodes (unvisitedNodes), initiate it by passing a set of all nodes of the graph

Step3:

Create a set to collection all nodes which are articulationPoints (articulations)

Step4:

```
BEGIN Loop(go through all unvisited nodes)
    Initiate number of subtree equals 0
    Initial start node is assigned to next node of iterator of unvisitedNode Set
    Initial the count of start node equals 0
    BEIGIN loop(go through all neighbour node of this strat node)
        IF(count of neighbour node is equals to the MAX_count)
            Call method called
            "iteraArtPts(neighbour node, start node, set of articulation, set of unvisitedNodes)"
            Increase the number of subtree by 1
        END IF
    END LOOP
    IF (number of subtree > 1)
        Articulations set add the start node
    END IF
```

Unvisited set remove the start node
END loop
Return the set of articulations

Step5:

Create a method called

```
"iteraArtPts(neighbour node, start node, set of articulation, set of unvisitedNodes)"
```

Step6:

Create a stack to store all Articulation stack objects

Step7:

Create the parent stack element of the neighbour node

Step8:

Push the first stack element based on the neighbour node to the stack

Step9:

Q5—outline how you tested that your program wonvrked.

G