e-Yantra Robotics Competition Plus
(eYRC+ 2015)
eYRCPlus_CS#2862

Team leader name	Amitesh Prasad
College	National Institute of Science And Technology, Berhampur
e-mail	amitesh.prasad0007@gmail.com
Date	**20/12/2015**

NOTE:

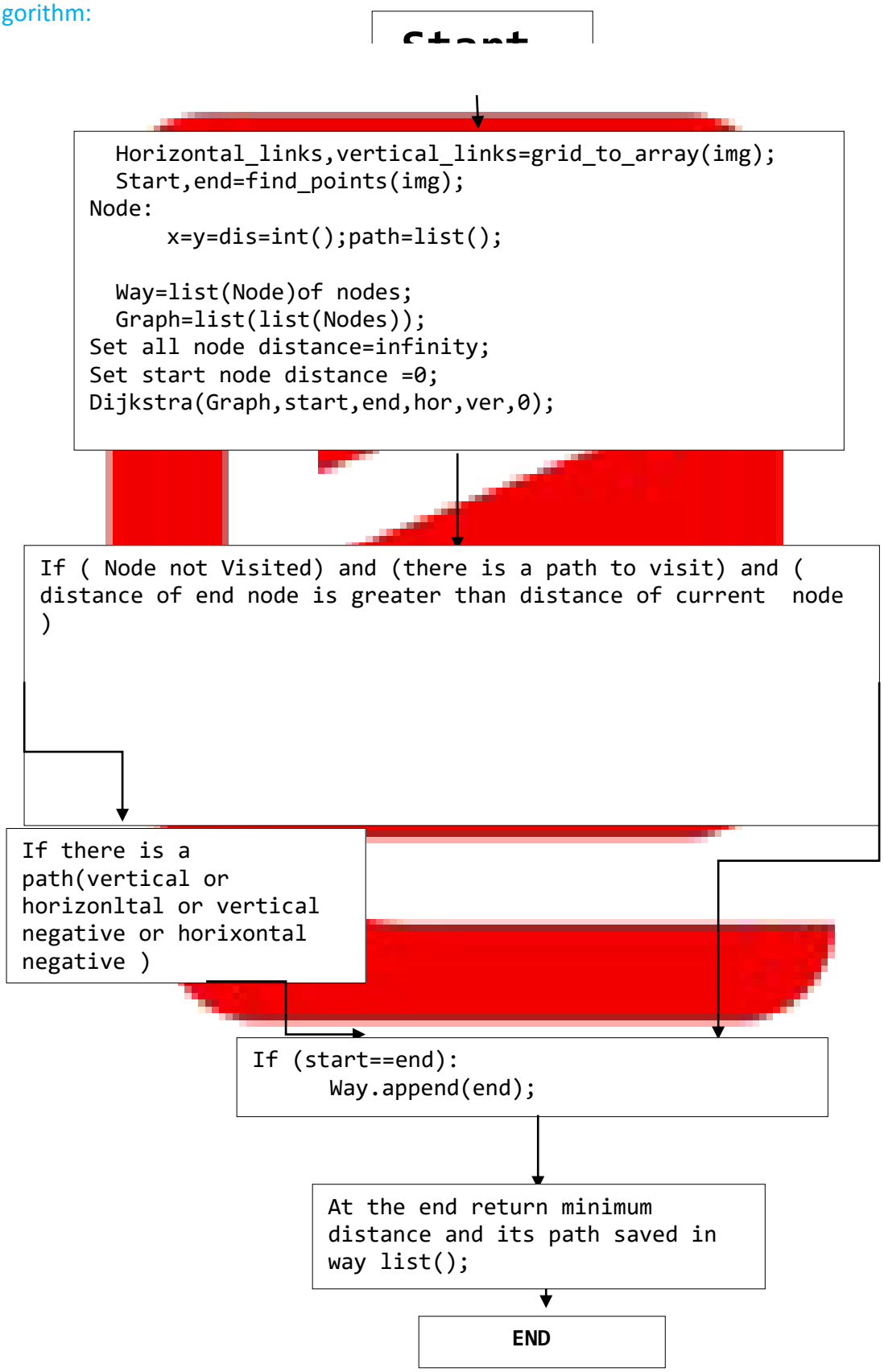
- We recommend you to **first complete the Experiments** and then answer the questions given below.
- Teams have to modify this document itself. **Do not create a new document.**

Algorithm (20)

1. Describe the algorithm used in Experiment 2: Path Planning (Please refer the *CS_Task2_Experiments* folder for information related to Experiment 2).

Ans 1 : In the Path Planning problem we have used the DFS with the Dijkastra's technique.
We have used the recursive way to find and explore the path from starting node.
Step 1: Convert the Image into two Matrix horizontal links and
vertical links.
Which we use to find if there is a horizontal edge or vertical edge from the current node or not.
Step 2: Find the start node and end node of the image . We have searched all the nodes and the node which is green is the start node and the node which is red is the end node.
Step 3: implementing my algorithm .I have used a recursive algorithm to find and explore all the possible paths from start node with calculating the path and distance together. From current node go to left ,right,up,down depending on if there is a path or not.
If the current node is end node then update the distance and path of that node. We have used pruning here which will reduce the search space i.e if the current value of end node dis is less than the current value of distance of current node do not traverse forward.

Algorithm:


Start
Horizontal_links,vertical_links=grid_to_array(img);
Start,end=find_points(img);
Node:
x=y=dis=int();path=list();
Way=list(Node)of nodes;
Graph=list(list(Nodes));
Set all node distance=infinity;
Set start node distance =0;
Dijkstra(Graph,start,end,hor,ver,0);
If (Node not Visited) and (there is a path to visit) and (distance of end node is greater than distance of current node)
If there is a path(vertical or horizonltal or vertical negative or horixontal negative)
If (start==end):
Way.append(end);
At the end return minimum distance and its path saved in way list();
END

2. Explain any two other algorithms that you could have used in Experiment 2: Path Planning.

Ans2:We can use other algorithms like A*,Bellman–Ford algorithm.

A* algorithm:

A* uses a closest first search and finds a shortest path from a given initial node to final node . As A* traverses the graph, it builds up a tree of partial paths. The leaves of this tree are stored in a priority queue that orders the leaf nodes by a cost function (in our case it is just number of edges or cost is 1), which combines a heuristic estimate of the cost to reach a goal and the distance traveled from the initial node

it first searches the routes that appear to be most likely to lead towards the goal. it also takes the distance already traveled into account;

Bellman–Ford algorithm:

Bellman–Ford is based on the principle of relaxation, in which an old value of distance is replaced by minimum value . the approximate distance to each vertex is replaced by the minimum of its old value with the length of a newly found path. the Bellman–Ford algorithm simply relaxes all the edges, and does this |V|-1 times, where |V| is the number of vertices in the graph.