

**e-Yantra Robotics Competition - 2018**

**NS Task 1 Report 0039**

|  |  |
| --- | --- |
| **Author Name(s)** | **Sarthak Tanwani, Madhur Raghav, Prakhar Gupta, Ayush Aggarwal** |
| **Team ID** | **0039** |
| **Date** | **28th November 2018** |

**Q1. Describe the path planning algorithm you have chosen.**

We have used the well-known Dijkstra’s Algorithm as our path planning algorithm. We have created a complete function named dijkstra() that takes the graph, the start node (src), and the end node (row). This function updates an array (named dist[]) that holds the shortest path. Technically the Dijkstra’s Algorithm requires weights at each edge of the graph; however in our case we have given the value 1 if there is a connection or 0 if there is not.

This is how the Dijkstra’s Algorithm works:

1. It maintains a list of unvisited vertices.
2. It chooses a vertex (the source) and assigns a maximum possible cost (i.e. infinity) to every other vertex.
3. The cost of the source remains zero as it actually takes nothing to reach from the source vertex to itself.
4. In every subsequent step of the algorithm it tries to improve (minimize) the cost for each vertex. Here the cost can be distance, money or time taken to reach that vertex from the source vertex. The minimization of cost is a multi-step process.
   1. For each unvisited neighbor (vertex 2, vertex 3, vertex 4) of the current vertex (vertex 1) calculate the new cost from the vertex (vertex 1).
   2. For e.g. the new cost of vertex 2 is calculated as the minimum of the two ( (existing cost of vertex 2) or (sum of cost of vertex 1 + the cost of edge from vertex 1 to vertex 2) )
5. When all the neighbors of the current node are considered, it marks the current node as visited and is removed from the unvisited list.
6. Select a vertex from the list of unvisited nodes (which has the smallest cost) and repeat step 4.
7. At the end there will be no possibilities to improve it further and then the algorithm ends

**Q2. Describe the algorithm’s specific implementation i.e. how have you implemented it in your task?**

We have used Dijkstra’s path planning algorithm for calculating the shortest path dynamically between 2 nodes. We have saved the complete map for task 1.2 as an undirected graph. The Dijkstra function takes a graph, source node (src), and a final node (row) to calculate the shortest path that the bot should follow. During the complete task we have maintained 3 variables named previous\_node, current\_node and final\_node. These variables are dynamically changed and appropriately provided to the Dijkstra function for every path planning decision.

Once the path is decided we have to move the robot in that path. For that purpose we have used the function robo\_movement. This function takes 3 arguments final\_node, current\_node, and previous\_node by reference and provides suitable instructions to the bot for e.g. forward\_wls(), right\_turn\_wls() and left\_turn\_wls() to go from one node to the other keeping in mind the orientation of the bot at each place. This function has a considerable number of if statements as it takes every possible combination of previous\_node, current\_node and final\_node to calculate the proper alignment and decides how the bot should proceed.