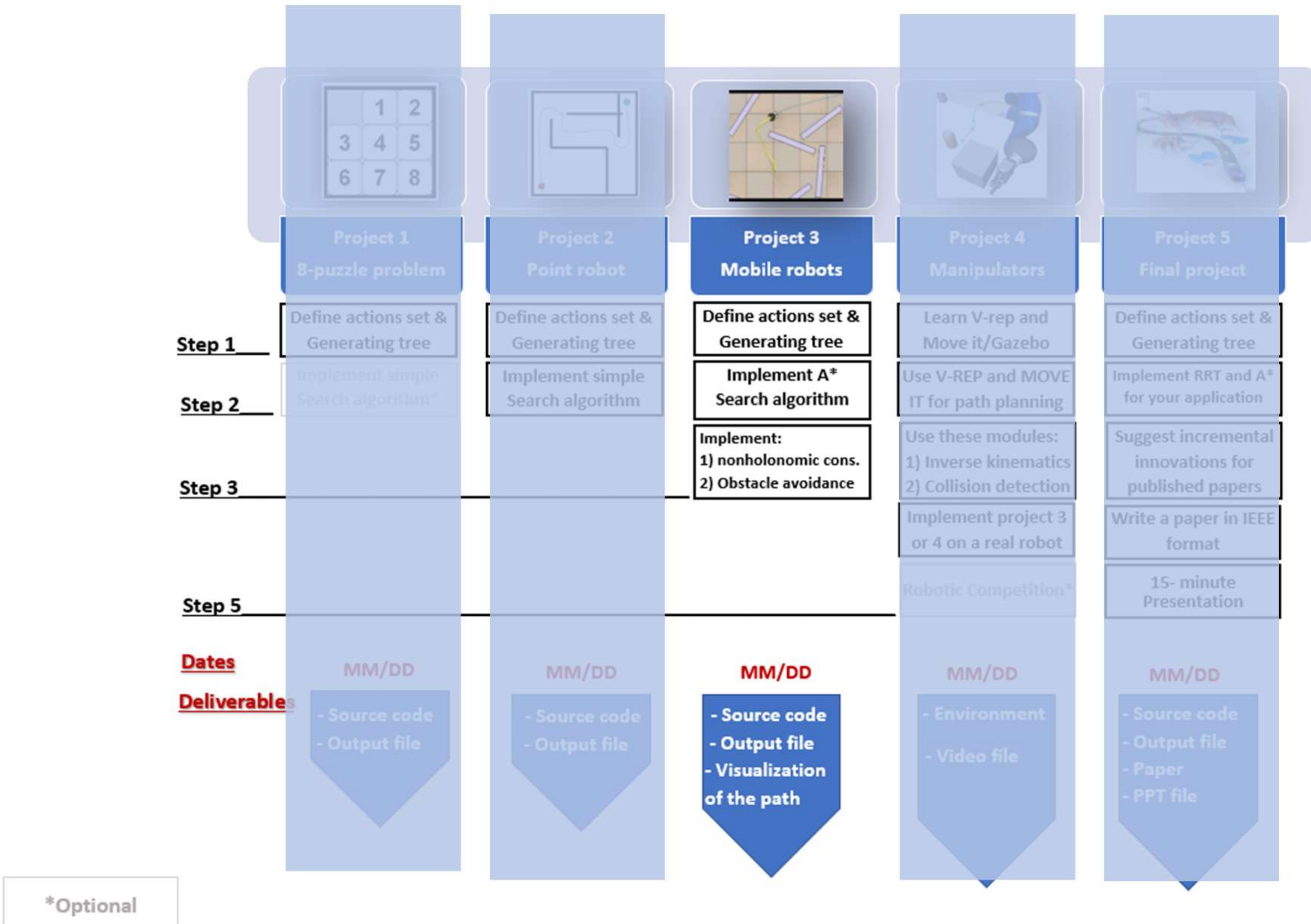
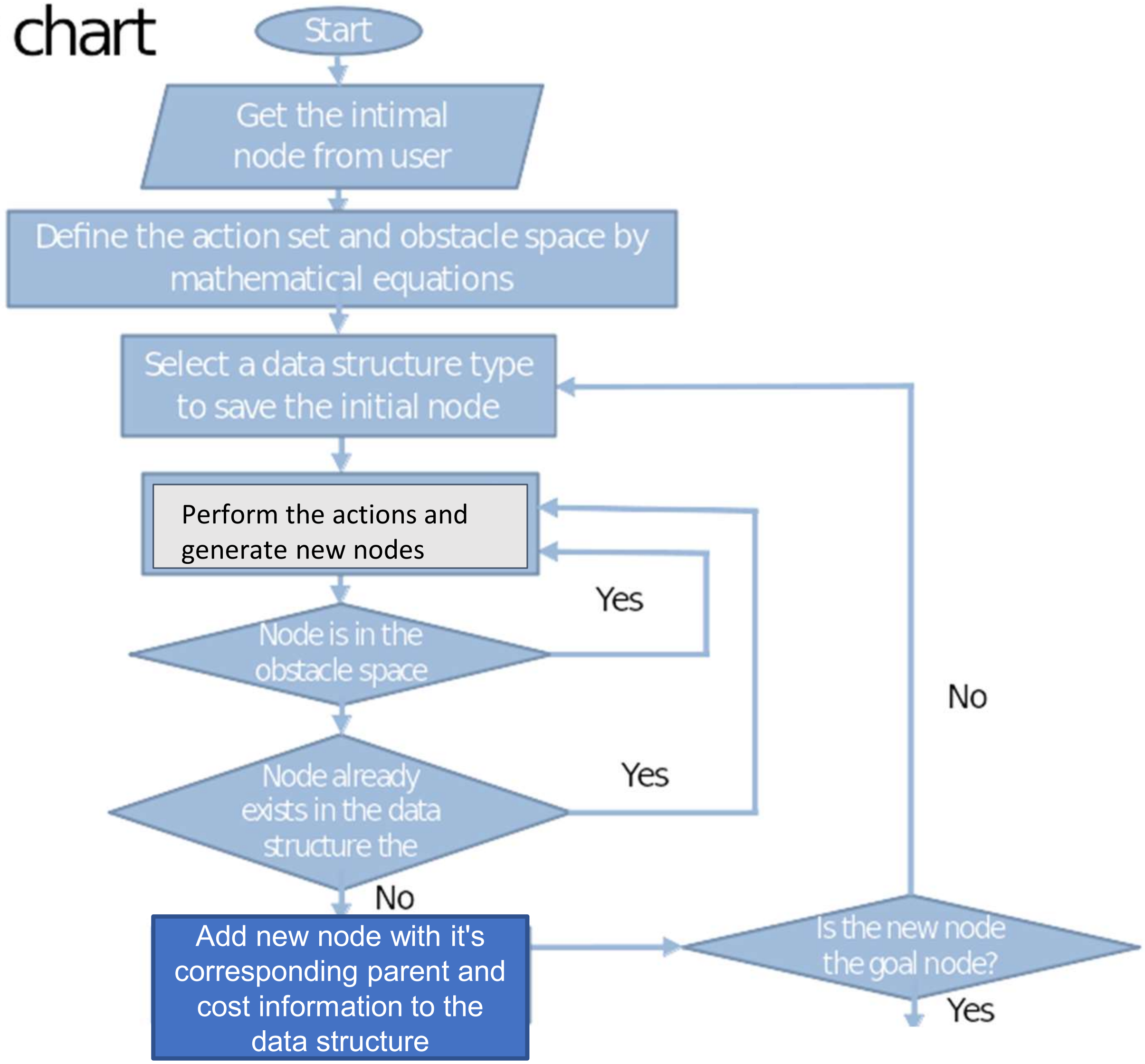
Project 3 Phase 1: Implementation Dijkstra algorithm for a mobile Robot

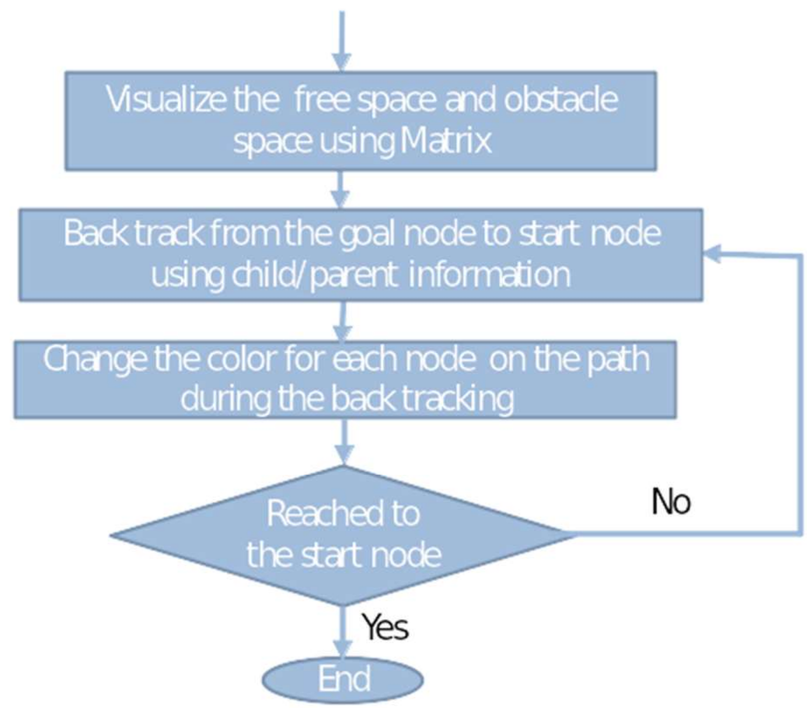
This is a group project.
Due Date (phase1 + phase2) – April 5th, 11.59 PM


1 2
3 4 5
6 7 8
Project 1
8-puzzle problem
Project 2
Point robot
Project 3
Mobile robots
Project 4
Manipulators
Project 5
Final project
Step 1
Step 2
Step 3
Step 5
Define actions set & Generating tree
Implement simple Search algorithm*
Define actions set & Generating tree
Implement simple Search algorithm
Define actions set & Generating tree
Implement A* Search algorithm
Implement:
1) nonholonomic cons.
2) Obstacle avoidance
Learn V-rep and Move it/Gazebo
Use V-REP and MOVE IT for path planning
Use these modules:
1) Inverse kinematics
2) Collision detection
Implement project 3 or 4 on a real robot
Robotic Competition*
Define actions set & Generating tree
Implement RRT and A* for your application
Suggest incremental innovations for published papers
Write a paper in IEEE format
15- minute Presentation
Dates
Deliverables
MM/DD
- Source code
- Output file
MM/DD
- Source code
- Output file
MM/DD
- Source code
- Output file
- Visualization of the path
MM/DD
- Environment
- Video file
MM/DD
- Source code
- Output file
- Paper
- PPT file
*Optional

Flow chart



Start
Get the intimal node from user
Define the action set and obstacle space by mathematical equations
Select a data structure type to save the initial node
Perform the actions and generate new nodes
Node is in the obstacle space
Yes
Node already exists in the data structure the
Yes
No
Add new node with it's corresponding parent and cost information to the data structure
Is the new node the goal node?
No
Yes

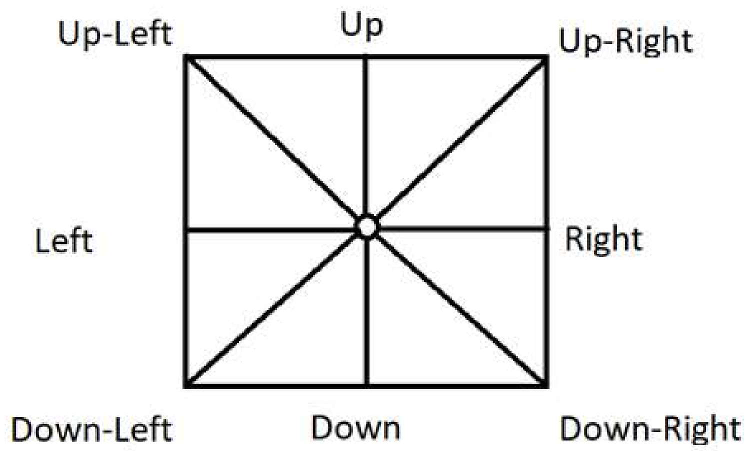

Visualize the free space and obstacle space using Matrix
Back track from the goal node to start node using child/parent information
Change the color for each node on the path during the back tracking
Reached to the start node
No
Yes
End

Pseudo code for Dijkstra Search

```
Q.Insert(x_I) and mark x_I as visited
while Q not empty do
    x ← Q.GetFirst()
    if x ∈ X_G
        return SUCCESS
    forall u ∈ U(x)
        x' ← f(x, u)
        if x' not visited
            Mark x' as visited
            Q.Insert(x')
            Parent( x' ) ⟵ x
            CostToCome(x') ⟵ CostToCome(x) +l(x,u)
            Cost(x') ⟵ CostToCome(x')
        else
            if  Cost(x') > CostToCome(x)+l(x,u)
                    CostToCome(x') ⟵ CostToCome(x) +l(x,u)
                    Cost(x') ⟵ CostToCome(x')
                    Parent(x') ⟵ x
            Resolve duplicate x'
return FAILURE
```

Going through the pseudo code is highly recommended

Project 2 Description
(Retain from previous project)



Project Assumption: Workspace is an 8 connected space, that means now you can move the robot in up, down, left, right & diagonally between up-left, up-right, down-left and down-right directions.

Action sets= {(1,0), (-1,0), (0,1), (0,-1), (1,1), (-1,1),(1,-1),(-1,-1)}

Project 3 Description

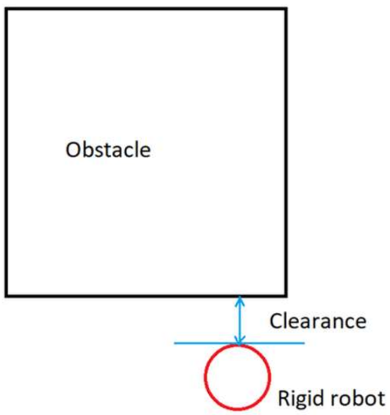
1) Check the feasibility of all inputs/outputs (if user gives start and goal nodes that are in the obstacle space they should be informed by a message and they should try again). (retain from previous project)

2) Implement Dijkstra's Algorithm to find a path between start and end point on a given map for a mobile robot (radius = 10; clearance = 5).

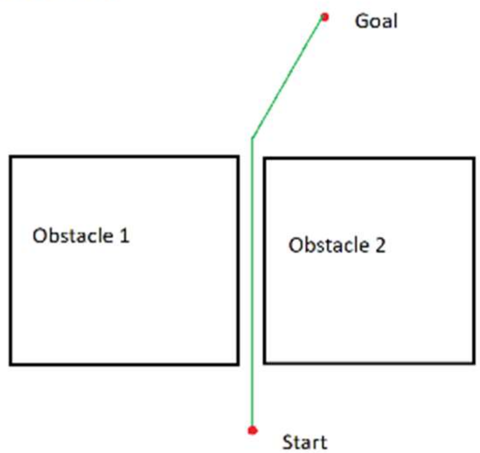
3) Your code must output an animation of optimal path generation between start and goal point on the map. You need to show both the node exploration as well as the optimal path generated. (Some useful tools for simulation are OpenCV/Pygame/Matplotlib). (retain from previous project)

Clearance

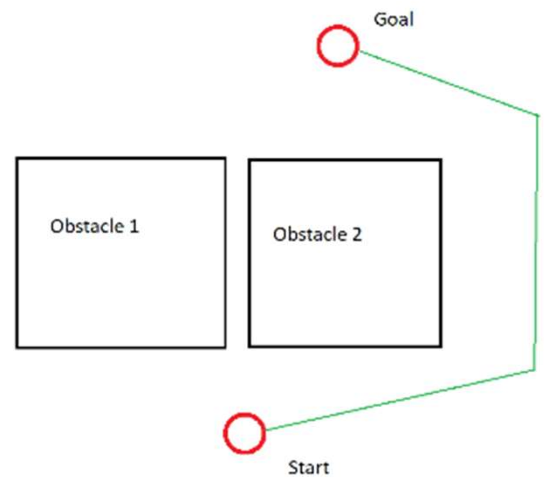
- Clearance is a maximum distance between the obstacle and the extreme point of the rigid robot.



Difference between point and rigid robot



Navigation scenario for point robot



Navigation scenario for rigid robot

Visualization

https://drive.google.com/file/d/1OTvRGCmQ35oXbf5HEe70rL6czHS3PJf5/view

Step 1) Define the actions in a mathematical format (Retain from previous project)

- Use can use the same data structure from project 1 to store the node information.
- Write 8 subfunctions, one for each action. The output of each subfunction is the state of a new node after taking the associated action.

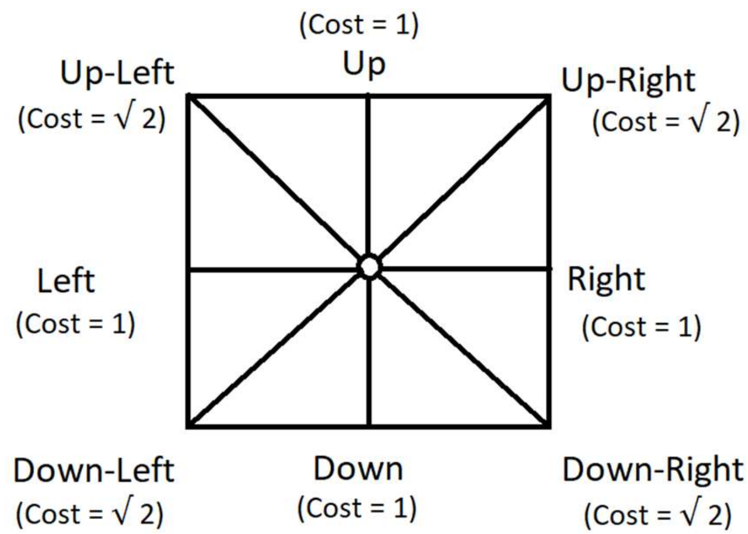
Action sets= {(1,0), (-1,0), (0,1), (0,-1), (1,1), (-1,1),(1,-1),(-1,-1)}

Step 2) Find mathematical representation of free space
(Retain from previous project)

- **Use Half planes and semi-algebraic models** to represent the obstacle space.

Step 3) Generate the graph (Retain from previous project)

- Generate the graph using action set for a 8-connected space and save in a data structure format
- Before saving the nodes, check for the nodes that are within the obstacle space and ignore them
- Cost of moving diagonally is $\sqrt{2}$



Step 4) Find the optimal path (Backtracking)
(Retain from previous project)

- Write a subfunction that compares the current node with the goal node and return TRUE if they are equal.
- While generating each new node this subfunction should be called
- Write a subfunction that once the goal node is reached, using the child and parent relationship, it backtracks from the goal node to initial node and outputs all the intermediate nodes.

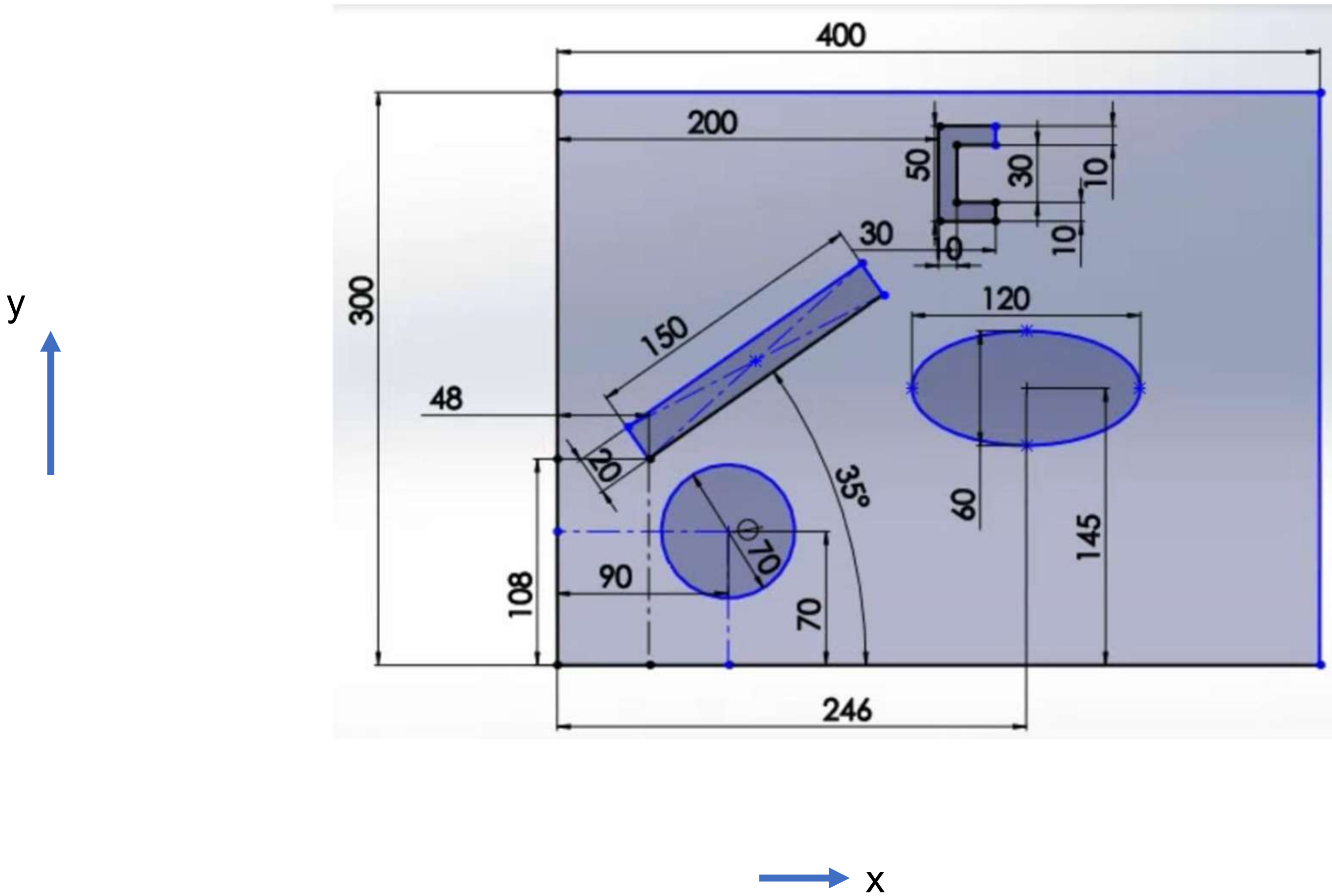
Step 5) Represent the optimal path (Retain from previous project)

- Show optimal path generation animation between start and goal point using a simple graphical interface. You need to show both the node exploration as well as the optimal path generated.

<u>The visualization of (exploration and optimal path) should start only after the exploration is complete and optimal path is found.</u>

Note: A separate document will be provided later this coming week to describe this step

Final Map (towards submission)



Deliverables

Deliverables:

1. ReadMe.txt (Describing how to run the code in a txt format)
2. Source files for
   - Student_name.py
   - GitHub repository link in the URL submission
   - Video recording (start and goal point can be random)

Note: The code should accept start and goal points from the user