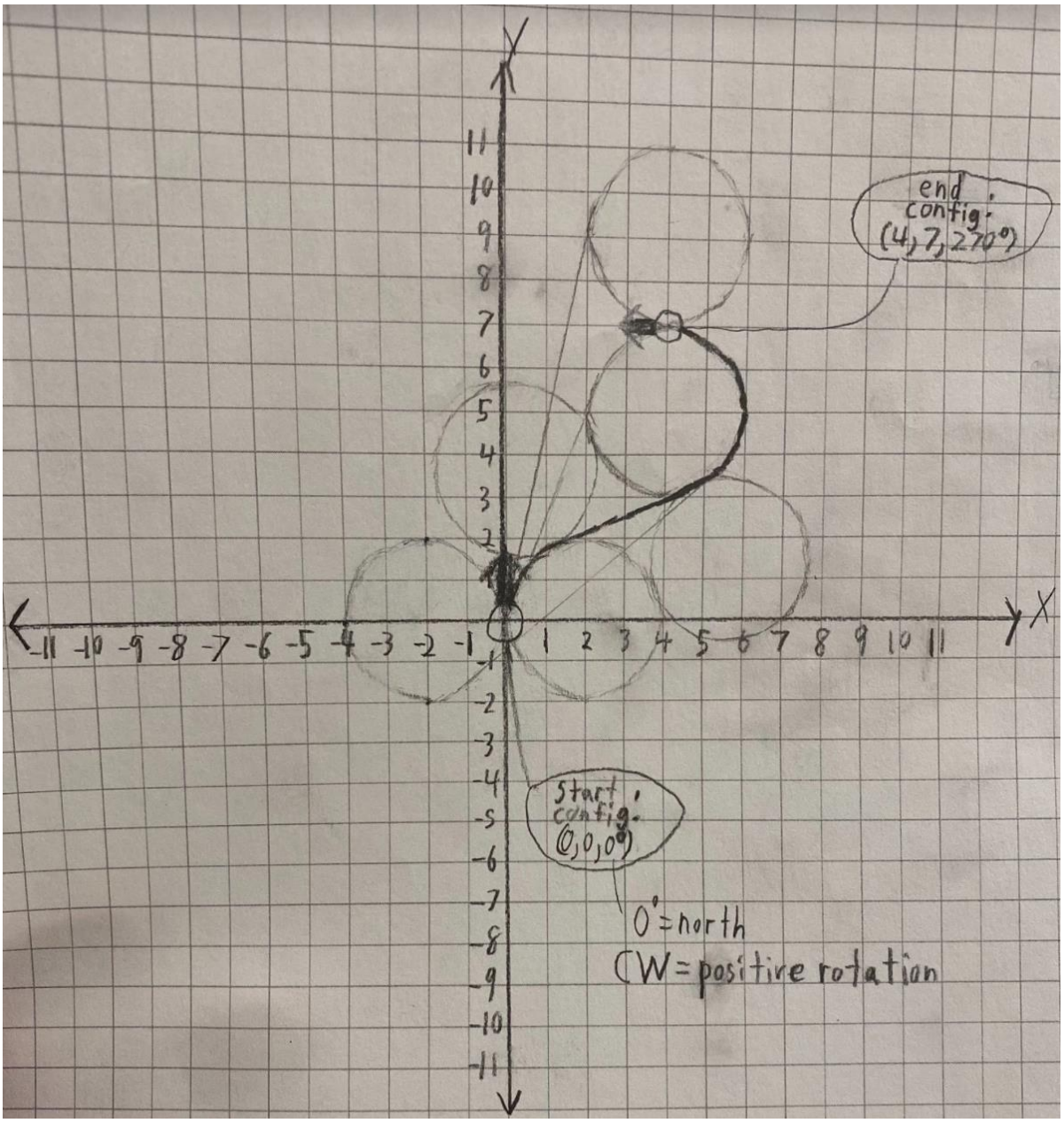
ME 459/5559 – Robotics and Unmanned Systems
HW #4: DUE March 4th, 2022

LATE HOMEWORK WILL BE DEDUCTED 10% PER DAY AFTER THE DUE DATE

Problem 1:

Given a starting configuration of (0,0,0°) for x, y, and heading (0° = north) and a goal configuration of (4,7,270°) (270° = west). With a minimum turning radius of 2, show the graphical method of computing the Dubin's path.



Problem 2:

Given the set of configurations/waypoints below and the Python script on Canvas, show the trajectory using Dubins curves/paths with a minimum turning radius of 2.0.

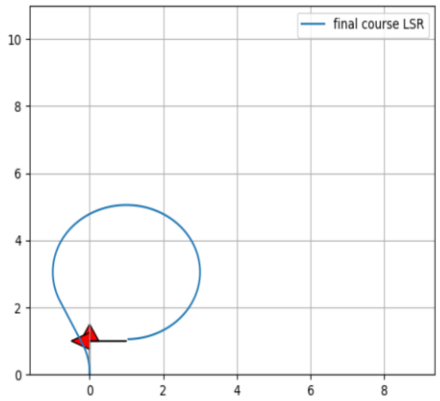
Waypoints/Configurations:

Configuration #1: [0,0, 90°]

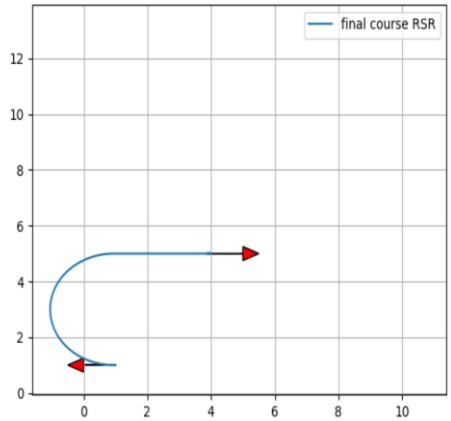
Configuration #2: [1,1,180°] Configuration #3: [4,5,0°]

Configuration #4: [7,7,270°]

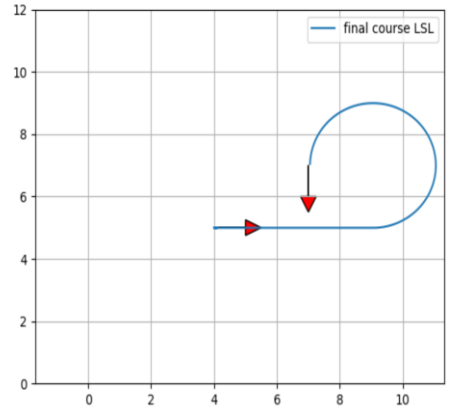
From configuration #1 to configuration #2:



From configuration #2 to configuration #3:



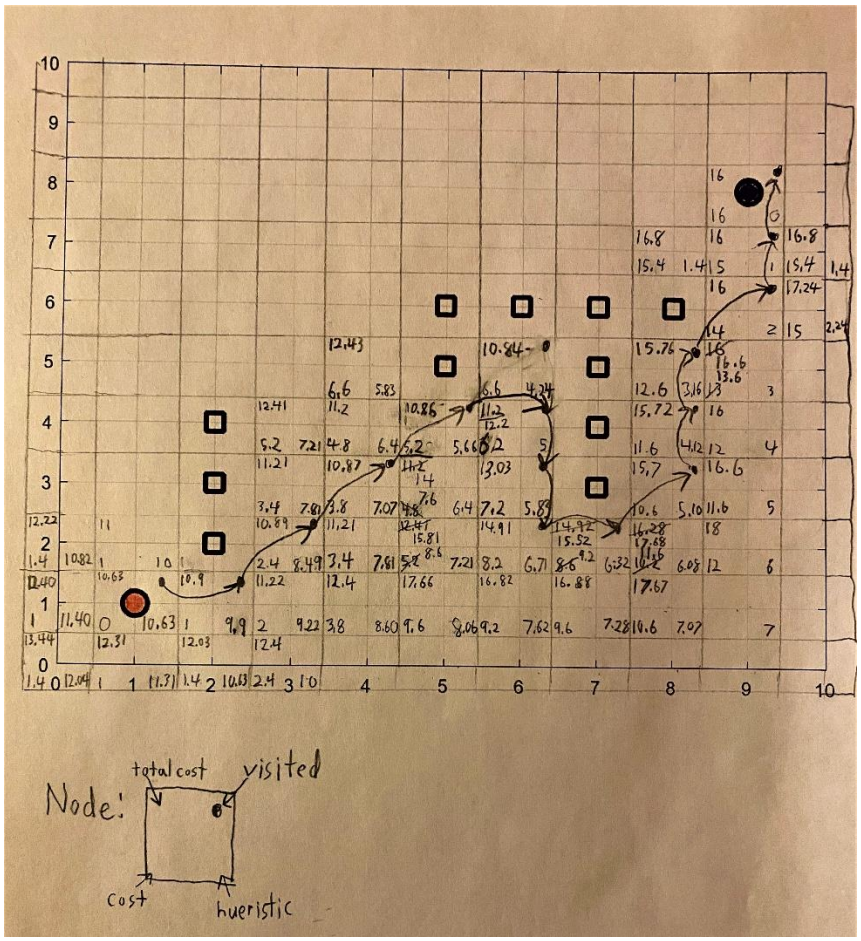
From configuration #3 to configuration #4:



Problem 3:

Given the map below, and a grid spacing of 1, use A* to compute the path **(by hand)** from start to finish (start = red, goal = blue, obstacles = black).

Show your work (i.e. show the travel cost, heuristic cost, and total cost for each node visited).

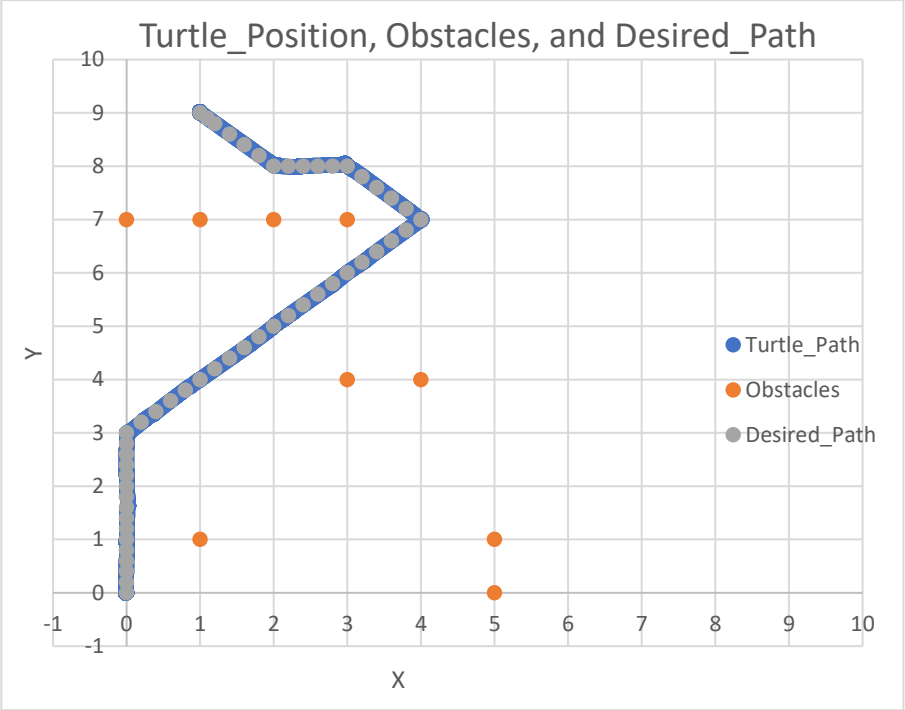


Problem 4:

Take your complete Dijkstra's script and integrate it with your working Turtlebot3 control architecture (minimum turning radius = 0m). Starting point of (0,0) and a goal of (1,9). Use the obstacle list: ((1,1), (4,4), (3,4), (5,0), (5,1), (0,7), (1,7), (2,7), and (3,7)). Show the x vs y position of the robot, desired path, and obstacles. This plot should include all of the information! (Two lines and a series of dots/markers for the obstacles).

Using Dijkstras:

Gridsize-10X10 Step Size = 1

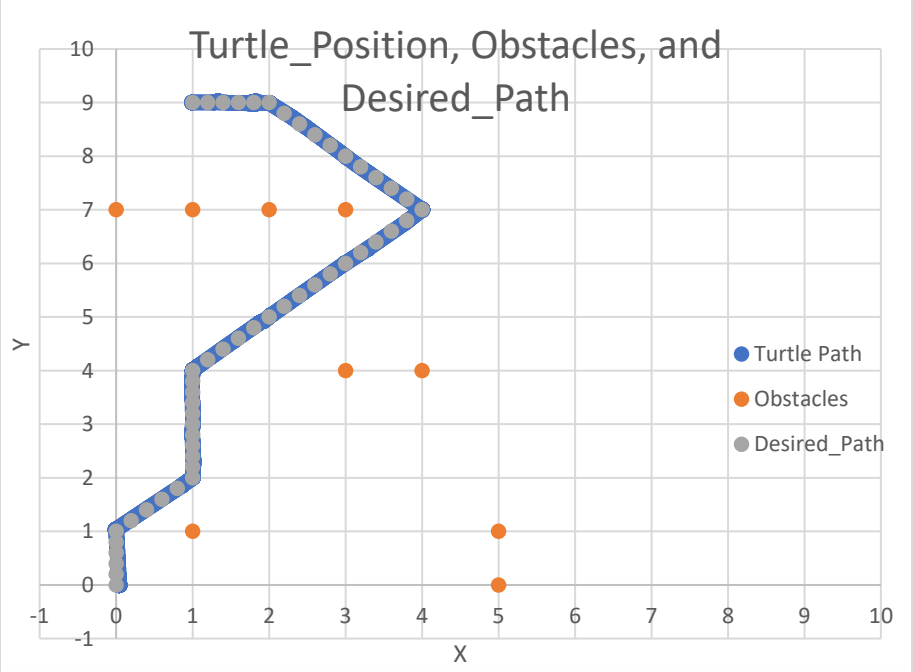


Problem 5:

Modify your Dijkstra's software to run the A* algorithm. This should require only small modifications to the cost estimate for each cell. Rerun the same simulation environment as Problem 4 and show the x vs y graph.

Using Astar:

Gridsize-10X10 Step Size = 1



Submit your Python code.

Problem 6:

Modify your Dijkstra/A* code (standalone Python script, not ROS) to use the RRT method to get from the start to the goal. Use the same obstacle list and bounding box. Use a distance to jump (from nearest node in the tree) of 0.5.

Create a plot showing the tree (valid nodes) and the corresponding path to get from the start to the goal for the same map as Problem 4.

Submit your Python code.

I'm Still Working This Problem! I Will Be Done Soon And Will Resubmit!