ME 459/5559 – Robotics and Unmanned Systems
HW #5: DUE October 5th, 2022

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

Problem 1:

Using the obstacle list given below, run (and time [tqdm is a useful package in Python for timing]) your Dijkstra's, A*, and RRT. Make sure you have disabled/commented out any plotting you have in your scripts that might slow down the execution. Show plots of the three trajectories. Create a table that shows the three methods, time to computer, and the total travel cost. Do your results match what you would expect? Explain.

Note: Use a robot size of 1.0 (or radius = 0.5) so you do not go through the obstacle list/walls

Start_x = 1
Start_y = 1
Goal_x = 7
Goal_y = 13
Obstacle_x = [2, 2, 2, 2, 0, 1, 2, 3, 4, 5, 5, 5, 5, 5, 8, 9, 10, 11, 12, 13, 8, 8, 8, 8, 8, 8, 8, 2, 3, 4, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 2, 2, 2, 2, 2, 2, 5, 5, 5, 5, 5, 5, 5, 6, 7, 8, 9, 10, 11, 12, 12, 12, 12, 12]

Obstacle_y = [2, 3, 4, 5, 5, 5, 5, 5, 5, 5, 2, 3, 4, 5, 2, 2, 2, 2, 2, 2, 3, 4, 5, 6, 7, 8, 9, 7, 7, 7, 7, 7, 7, 6, 6, 6, 6, 6, 6, 6, 8, 9, 10, 11, 12, 13, 9, 10, 11, 12, 13, 14, 15, 12, 12, 12, 12, 12, 12, 8, 9, 10, 11, 12]

Max_x = 15
Max_y = 15

Problem 2:

Using the obstacle list given below, run your A* and RRT within the ROS2 (and Gazebo) environment. The turtlebot should follow the waypoints (just use the empty world for right now), ensure that your turtlebot did not drive over any of the obstacle locations.

Show a plot with both the planned path and the actual turtlebot path.

Note: Use a robot size of 1.0 (or radius = 0.5) so you do not go through the obstacle list/walls

Start_x = 1
Start_y = 1
Goal_x = 7
Goal_y = 13
Obstacle_x = [2, 2, 2, 2, 0, 1, 2, 3, 4, 5, 5, 5, 5, 5, 8, 9, 10, 11, 12, 13, 8, 8, 8, 8, 8, 8, 8, 2, 3, 4, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 2, 2, 2, 2, 2, 2, 5, 5, 5, 5, 5, 5, 5, 6, 7, 8, 9, 10, 11, 12, 12, 12, 12, 12]

Obstacle_y = [2, 3, 4, 5, 5, 5, 5, 5, 5, 5, 2, 3, 4, 5, 2, 2, 2, 2, 2, 2, 3, 4, 5, 6, 7, 8, 9, 7, 7, 7, 7, 7, 7, 6, 6, 6, 6, 6, 6, 6, 8, 9, 10, 11, 12, 13, 9, 10, 11, 12, 13, 14, 15, 12, 12, 12, 12, 12, 12, 8, 9, 10, 11, 12]

Max_x = 15
Max_y = 15

Problem 3:

Write a script that solves the Traveling Salesman Problem (TSP) through brute force. Forth this problem you simply need to compute the distance between the different waypoints/goal locations (no path planning required). Show a plot that highlights the optimal path to go from the starting point and visit the 4 waypoints with the minimum possible travelled distance.

Start = (0,0)
Goal 1 = (2,2)
Goal 2 = (5,3)
Goal 3 = (3,4)
Goal 4 = (6,4)

Note: You need to create a lookup table of the cost to go from any node to any other node (2 dimensional table). Use the distance as the cost for this problem. Then use a permutation package to give you all possible permutations of the start connected with the 4 goals (4! Possibilities). You can then go through these 4! possibilities and compute the total cost to travel. Find the minimum, and that is your best path for the TSP.