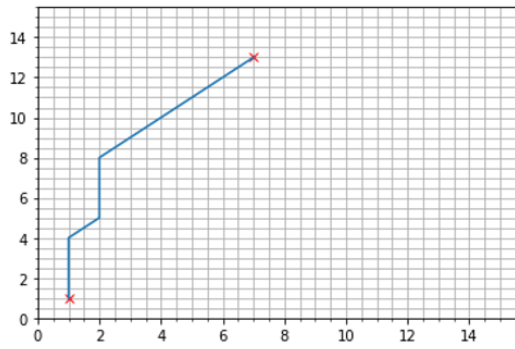
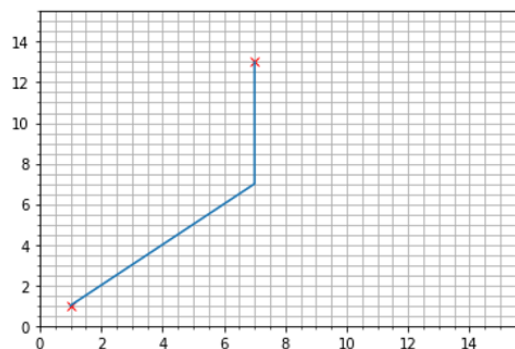
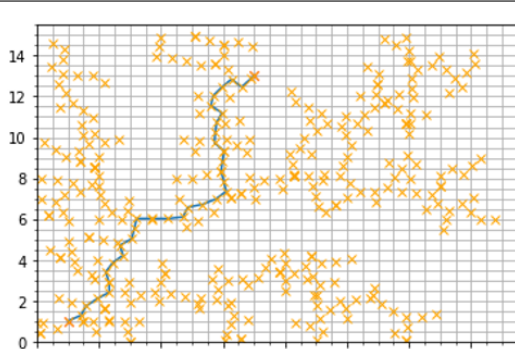


ME 459/5559 – Robotics and Unmanned Systems
HW #5: DUE March 24th, 2022

Problem 1:

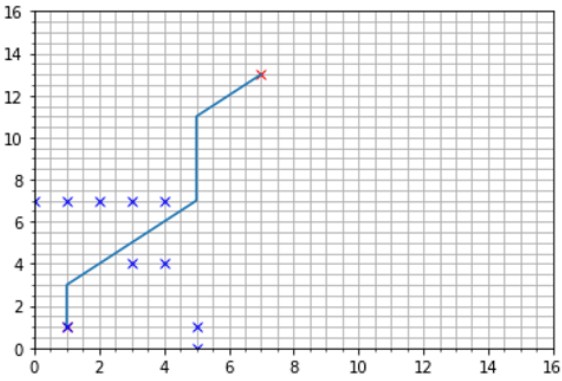
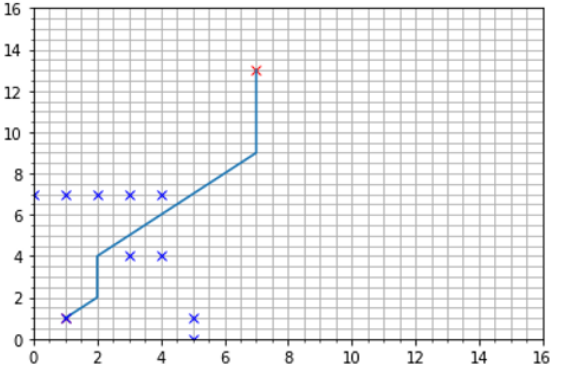
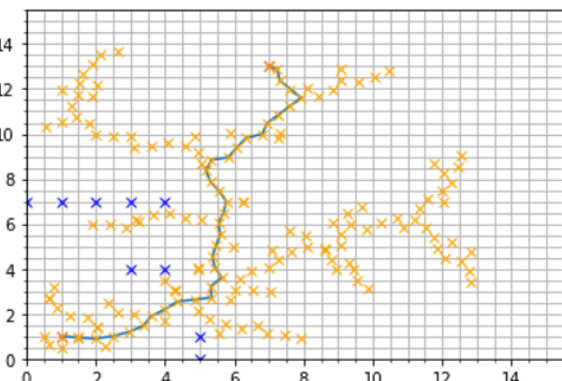
Using the obstacle list given below, run (and time [tqdm is a useful package in Python for timing]) your Dijkstra's, A*, and RRT. Make sure you have disabled/commented out any plotting you have in your scripts that might slow down the execution. Show plots of the three trajectories. Create a table that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

algorithm	Plot	Cost	Run Time
Dijkstra		14.48528	0.0099998
A_Star		14.48528	0.0069962
RRT		16.66681	0.11025

Yes, the Dijkstra and A_star algorithm have the same cost, but the Dijkstra has a much larger compute time because it searches all of the C-space. RRT gives a random suboptimal path and although the compute time is not going to be consistent the time to compute is likely to be relatively fast with no obstacles in the way.

Problem 2:

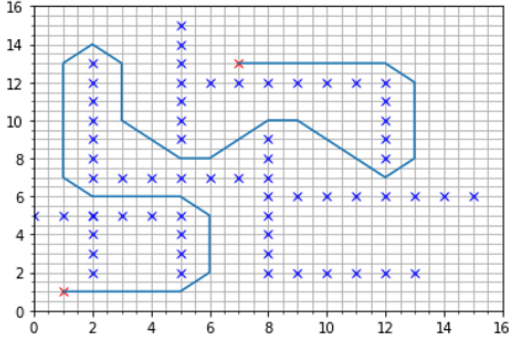
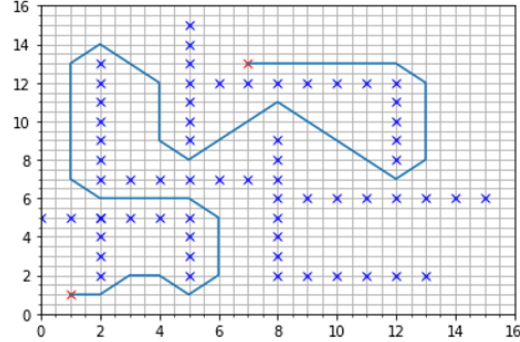
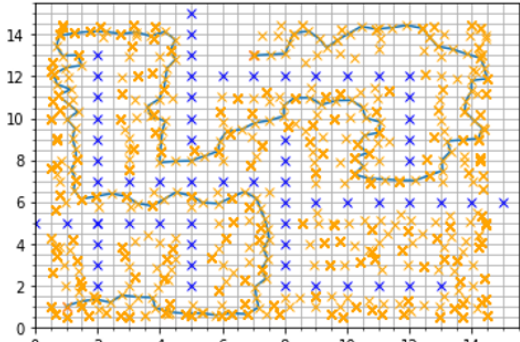
Using the obstacle list given below, run (and time [tqdm is a useful package in Python for timing]) your Dijkstra's, A*, and RRT. Make sure you have disabled/commented out any plotting you have in your scripts that might slow down the execution. Show plots of the three trajectories. Create a table that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

algorithm	Plot	Cost	Run Time
Dijkstra	 A 2D plot with x and y axes ranging from 0 to 16. The plot shows a blue line representing the optimal path from a start point at (1, 1) to a goal point at (7, 13). The path starts at (1, 1), goes up to (1, 3), then right to (5, 3), then up to (5, 11), and finally right to (7, 13). There are several blue 'x' marks representing obstacles at various coordinates, including (1, 7), (2, 7), (3, 7), (4, 7), (5, 7), (3, 4), (4, 4), (5, 1), and (6, 1).	14.48528	0.014999
A_Star	 A 2D plot with x and y axes ranging from 0 to 16. The plot shows a blue line representing the optimal path from a start point at (1, 1) to a goal point at (7, 13). The path starts at (1, 1), goes up to (1, 4), then right to (5, 4), then up to (5, 9), and finally right to (7, 13). There are several blue 'x' marks representing obstacles at various coordinates, including (1, 7), (2, 7), (3, 7), (4, 7), (5, 7), (3, 4), (4, 4), (5, 1), and (6, 1).	14.48528	0.007974
RRT	 A 2D plot with x and y axes ranging from 0 to 16. The plot shows a green line representing the sub-optimal path from a start point at (1, 1) to a goal point at (7, 13). The path is more complex and winding than the others, starting at (1, 1), going up to (1, 3), then right to (5, 3), then up to (5, 11), and finally right to (7, 13). There are many orange 'x' marks representing obstacles, including (1, 7), (2, 7), (3, 7), (4, 7), (5, 7), (3, 4), (4, 4), (5, 1), and (6, 1).	17.27259	0.059197

Yes, both the Dijkstra and the A_star algorithm return optimal paths and with slightly high compute times. Again, the RRT algorithm returned a sub optimal path with a relatively fast compute time. Although there were some obstacles, they weren't constricting enough to effect the time to compute time for the RRT algorithm.

Problem 3:

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.

algorithm	Plot	Cost	Run Time
Dijkstra		49.79899	0.014001
A_Star		51.45584	0.013995
RRT		72.02978	171.1114

Yes, The Dijkstra algorithm returns a optimal path. The time to compute the Dijkstra's algorithm has remained mostly unchanged. Since Dijkstra search's all of the C-space then adding more obstacles to the C-space won't increase the compute time much. A_star returned a nearly optimal path and has a slight increase from problem two. Unlike Dijkstra, A_star and RRT are forced to search more of the C-space when more barriers are added between the start and the goal. Therefore, adding more obstacle will increase their time to compute. With RRT the ratio of obstacle to C-space has a large effect on the time to compute. The more constricting the map is the more likely points will not be valid, thus getting thrown out, and a much larger amount of nodes in the node tree will likely be produced.

Problem 4:

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.

The shortest path is = (1, 3, 2, 4)

The cost is = 9.071067811865476

