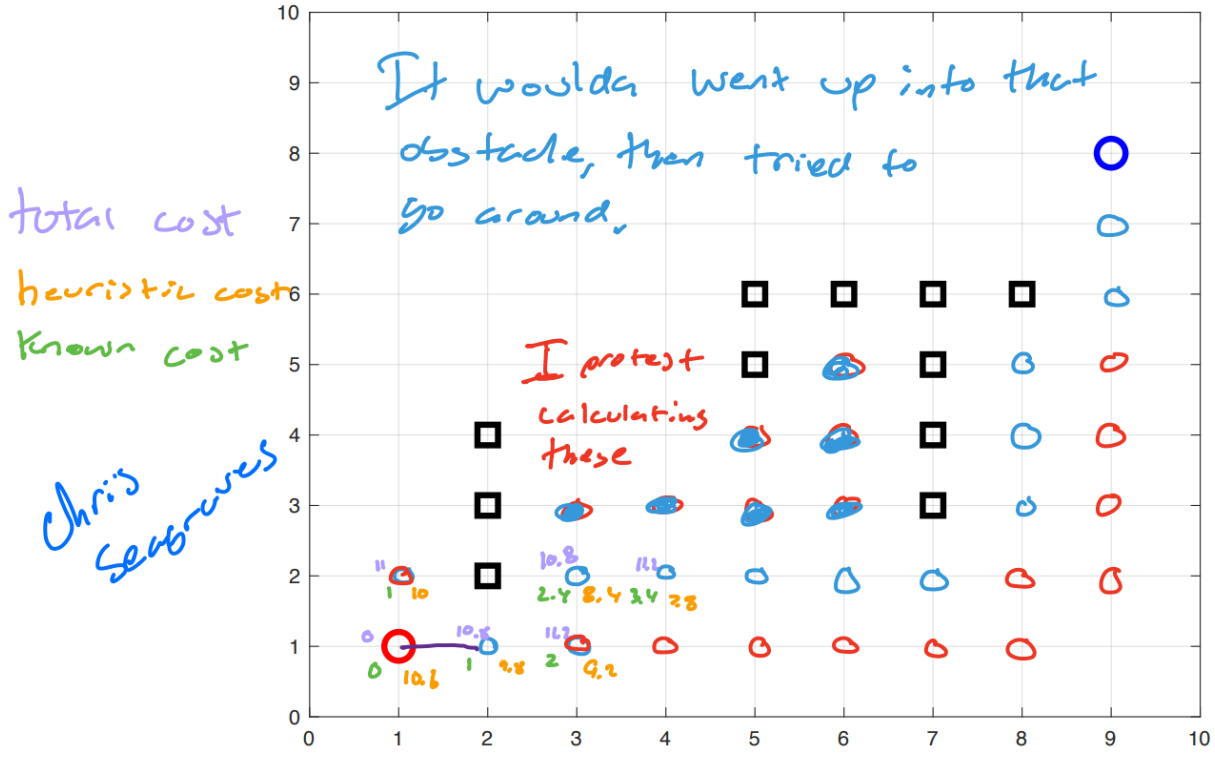
Homework 3
Christopher Seagraves

# Problem 1

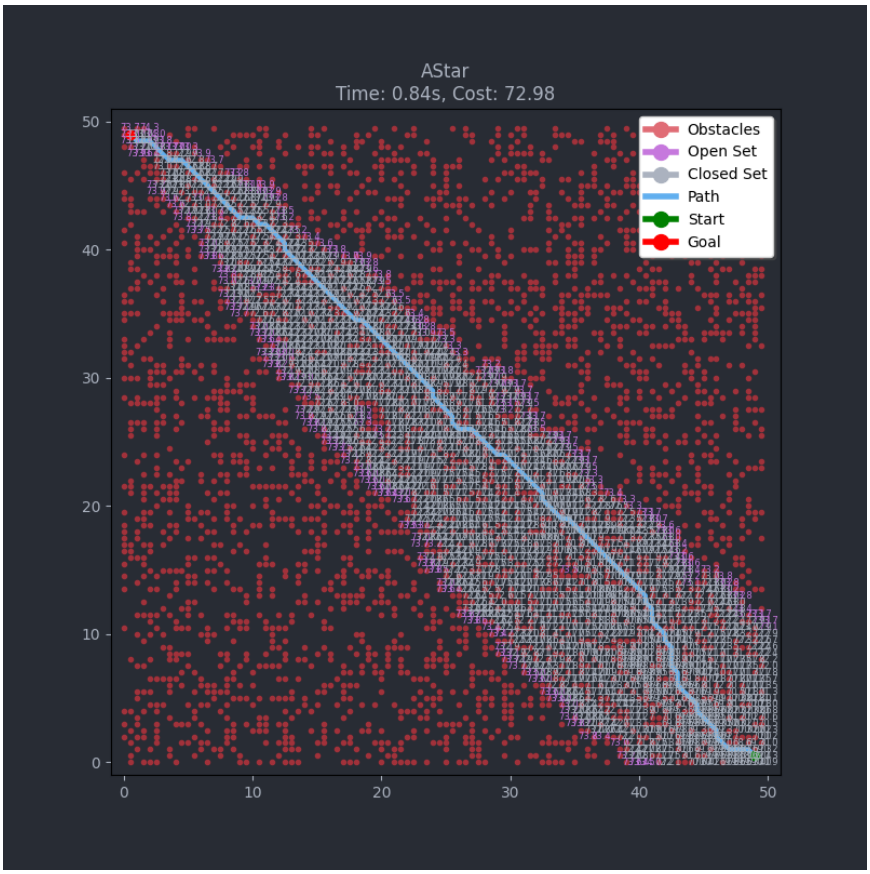
I'm sorry for being lazy, but hopefully doing 6 or so nodes is enough to prove I could have continued...



# Problem 2

It should be noted, if you loop the obstacle list every iteration, this will take a long time - O(nodes-visited * num-obstacles). The solution is to put invalid nodes into a set and check if a given node exists in the set - O(nodes-visited).

./main.py
`https://github.com/nosv1/seagraves_unmanned_systems/blob/main/SearchAlgorithms/main.py`
./AStar.py
`https://github.com/nosv1/seagraves_unmanned_systems/blob/main/SearchAlgorithms/AStar.py`



# Problem 3

It should be noted, there does not exist a solution if you inflate the obstacles greater than the bot radius. For the plot, I'm plotting the obstacle node at size 0.

It should also be noted, the algorithim I used for stepping towards a node:

- generate random node
- step towards node from closest node
- check if snapped-to-grid sub-steps along step are valid
- saving node if step is valid

This lets me use steps that aren't in the grid but check if nodes are valid quickly. Like, for a more relaxed map - with more space between obstacles - I can inflate an obstacle, define the nodes in the obstacle as invalid, snap to a close node, check if it's invalid, and assume my 'non-snapped' node is valid or invalid.

Something else to consider about taking steps towards nodes and using 'non-snapped' nodes is - assuming your obstacle is equa-distant to its center - take sub-steps along your step to see if any of them are invalid; this helps to avoid stepping over part of an obstacle - even if your close node and new node are both valid. However, checking sub-steps bumps the time from O(nodes-visited) to O(nodes-visited * sub-step-ratio).

RRT is terrible for goals close to the edge... It's just luck if the randomizer generated points beyond the edge to step towards the goal... I wound up expanding my randomizer range to 1.5x the map size to help it.
./main.py
`https://github.com/nosv1/seagraves_unmanned_systems/blob/main/SearchAlgorithms/main.py`
./RRT.py
`https://github.com/nosv1/seagraves_unmanned_systems/blob/main/SearchAlgorithms/RRT.py`