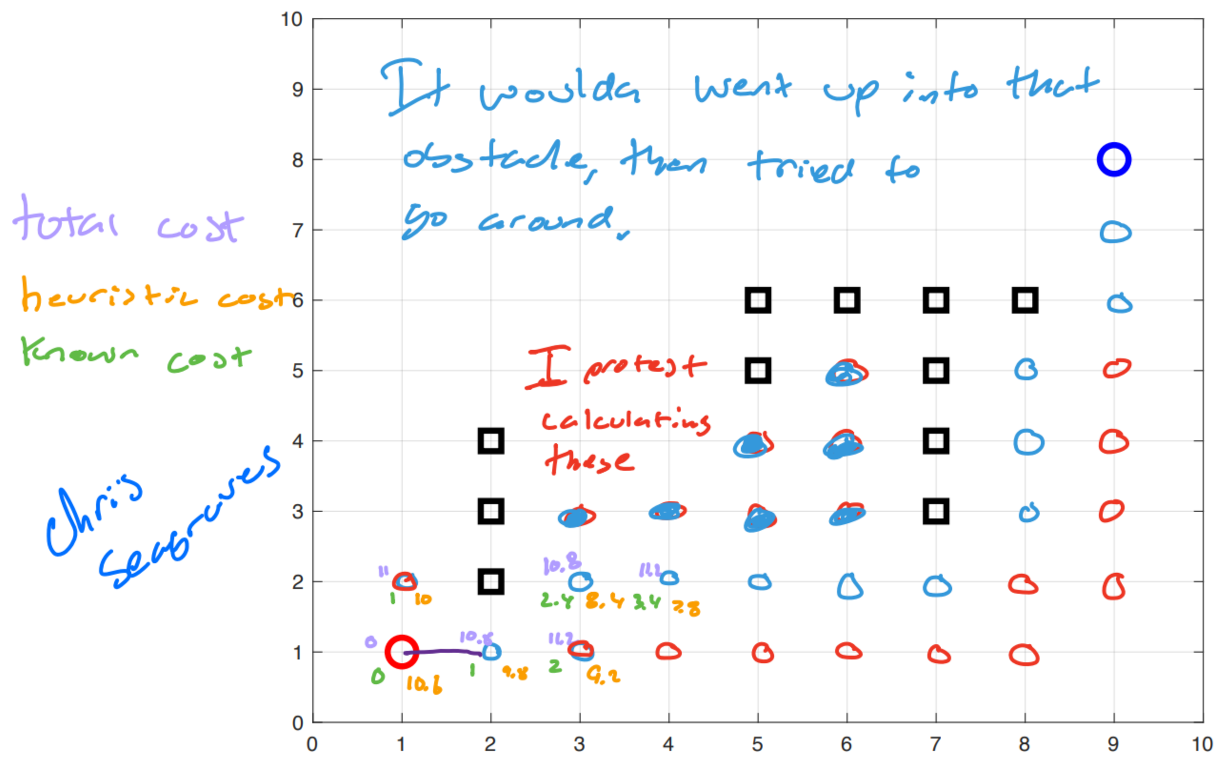
Homework 3
Christopher Seagraves

Problem 1

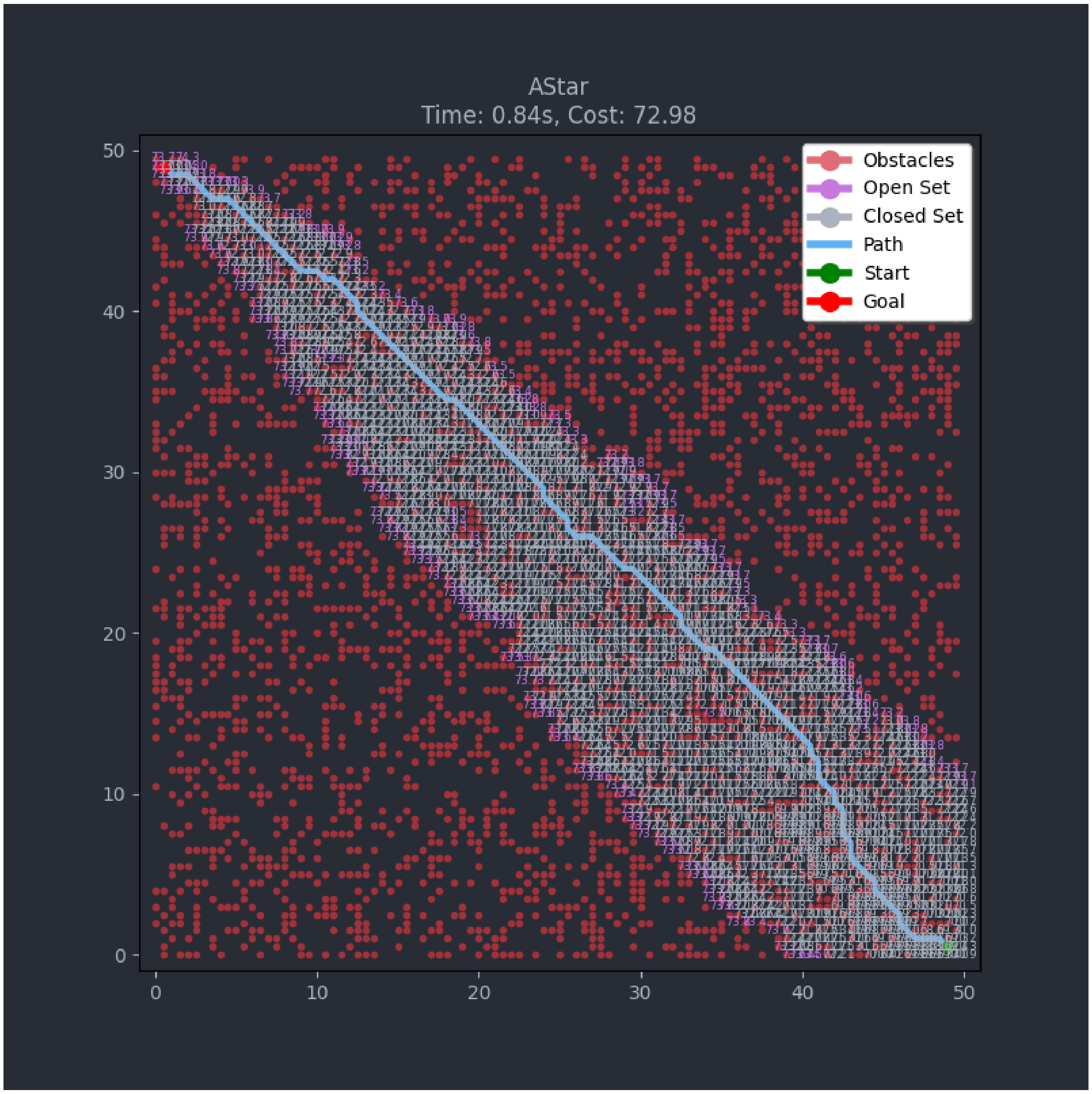
I'm sorry for being lazy, but hopefully doing 6 or so nodes is enough to prove I could have continued...



Problem 2

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

./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 didn't inflate the obstacles at all, so an obstacle is the size of one node.

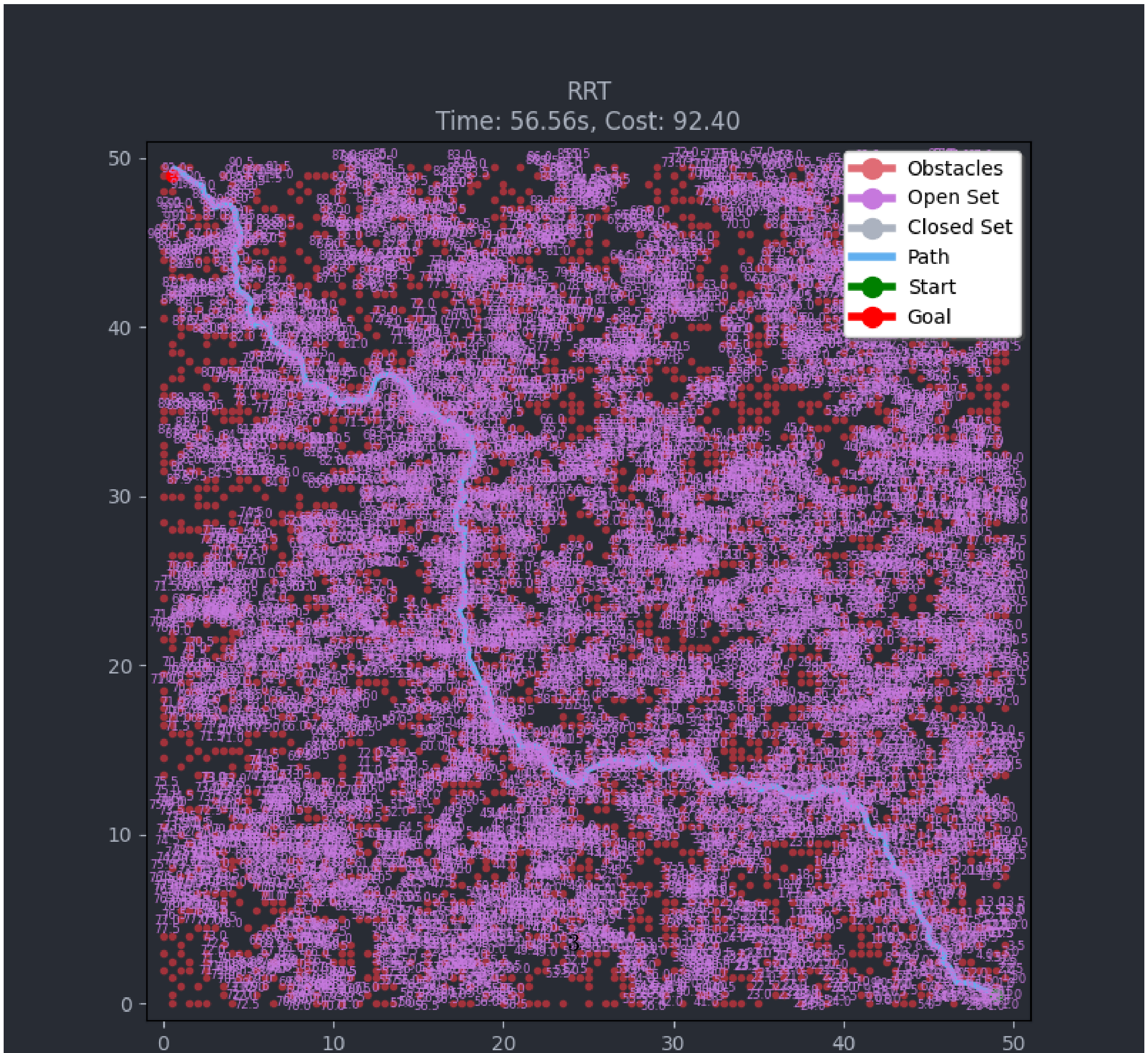
It should also be noted, the algorithim I used for stepping towards a node:

- generate random node
- step towards node from closest node
- snap copy of node to grid
- save stepped node if snapped node is a valid node

This lets me use steps that aren't in the grid but still try and avoid obstacles. 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 you can do about these 'non-snapped' nodes, is when you take a step, you can take smaller steps along the line, and see if any of those steps are invalid - so you don't accidently step over a part of an obstacle.

./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`


RRT
Time: 56.56s, Cost: 92.40
Obstacles
Open Set
Closed Set
Path
Start
Goal