PROBLEM 1:

The RRT parameters were defined as follow in the code for p1.py

```
n = 30
epsilon = 1.5
init_robot_cell = (0,0)
goal_pos = (9,9)
goal_rad = 0.5
```

RUNTIME DATA over 20 iterations

avg_runtime = 0.01066575050354004
variance_runtime = 4.589042697429441e-05

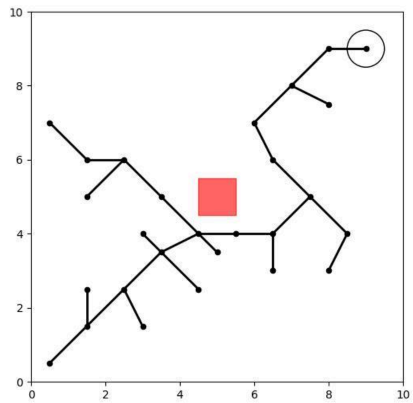
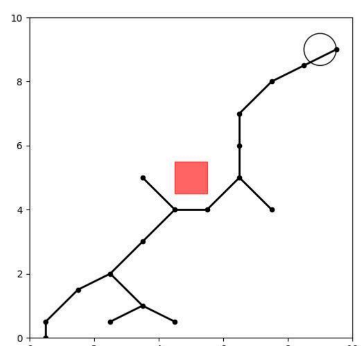


FIGURE of RRT path after 20 iterations

avg_runtime = 0.010732150077819825 s
variance_runtime = 4.798674891138657e-05 s

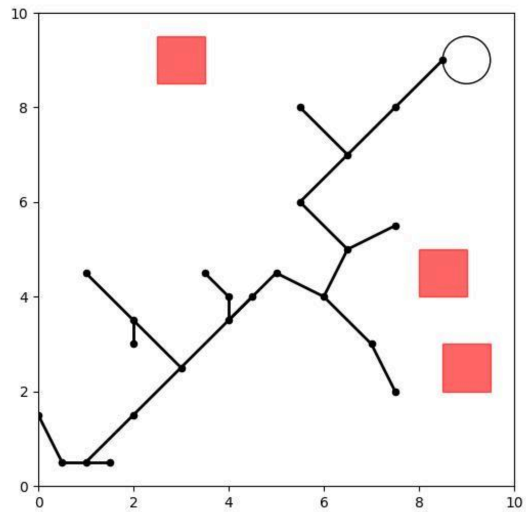


PROBLEM 1B

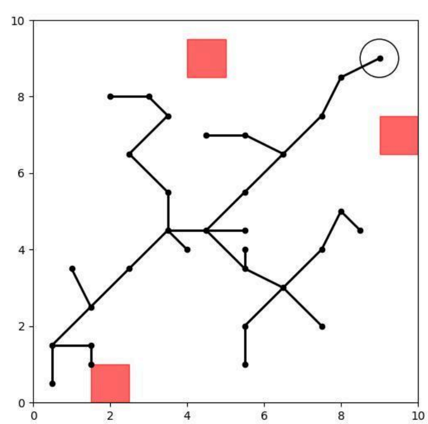
3 obstacles

```
n = 40
epsilon = 1.5
init_robot_cell = (0,0)
goal_pos = (9,9)
goal_rad = 0.5
```

avg_runtime = 0.013198959827423095
variance_runtime = 6.365980571753258e-05



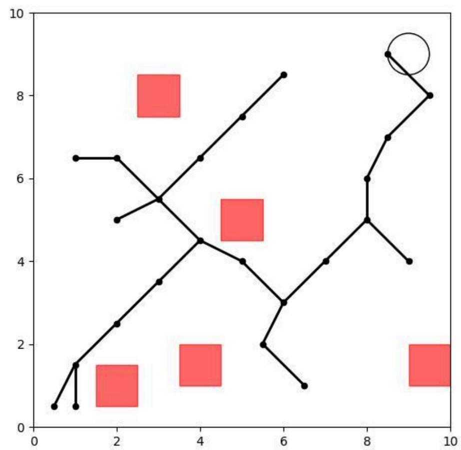
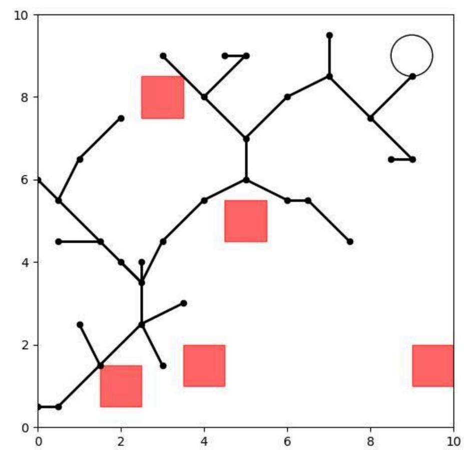
avg_runtime = 0.012833058834075928
variance_runtime = 4.005124290604032e-05



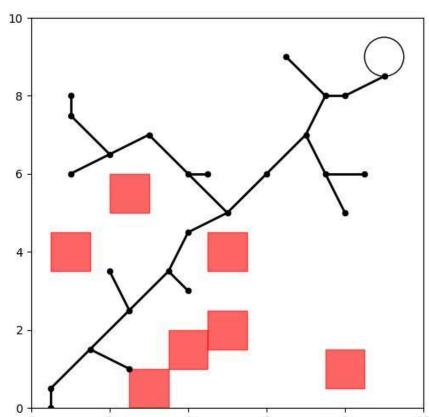
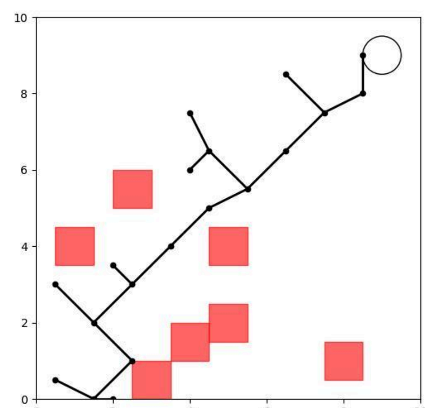
5 obstacles:

```
n = 50
epsilon = 1.5
init_robot_cell = (0,0)
goal_pos = (9,9)
goal_rad = 0.5
```

avg_runtime = 0.01432269811630249
variance_runtime = 2.734092635361513e-05

8 obstacles
Same parameters as previous
avg_runtime = 0.013382542133331298
variance_runtime = 4.9708170957866345e-05

OBSERVATIONS: As the number of obstacles increases, generally the avg_runtime over 20 iterations decreases. This makes sense since the RRT algorithm has fewer potential paths to explore. To accelerate RRT one can bias towards the goal so it doesn't waste time exploring paths that are far away from the goal. In addition, the grid density could be decreased if possible.

PROBLEM 2

Parameters:

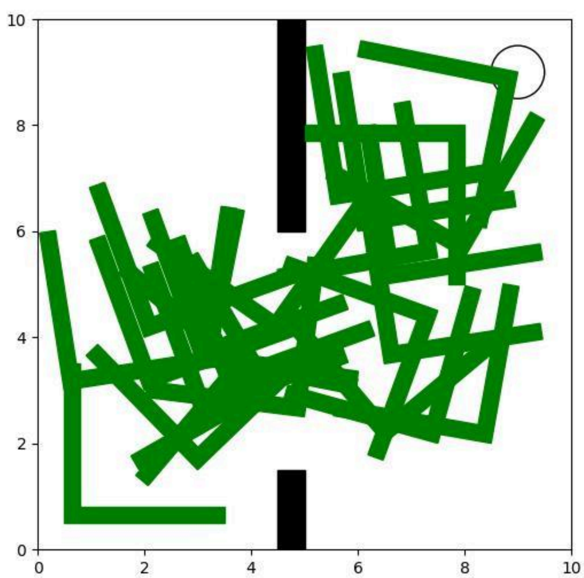
```
n = 100
epsilon = 4.0
theta_step = 20
init_robot_cell = (0,0)
init_robot_pos = (0.5,0.5)
init_theta = 0
init_corners = tuple(rotate_from_init_config(init_robot_pos,init_theta))
init_robot_state = (init_robot_pos, init_theta, init_corners)
goal_pos = (9,9)
goal_rad = 0.5
```

(80 executions of RRT)

Sample configuration paths and runtimes

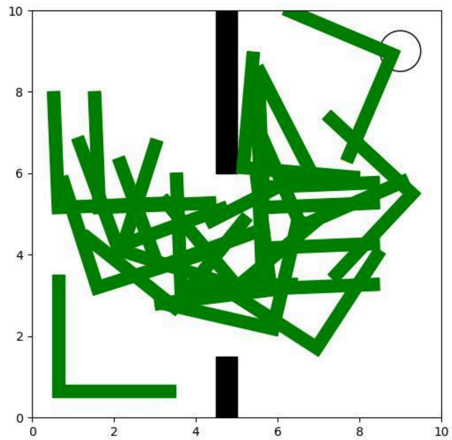
avg_runtime = 0.3261533784866333 s

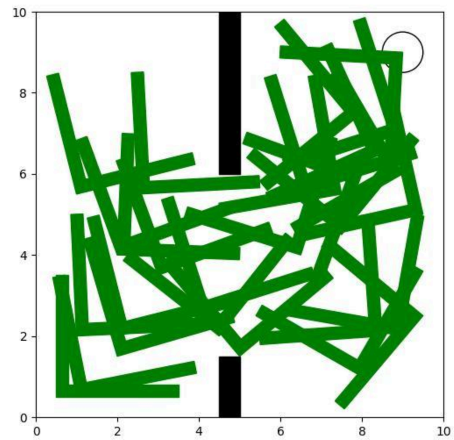
variance_runtime = 3.984019593087456 s



avg_runtime = 0.6083495736122131

variance_runtime = 9.76821592990797



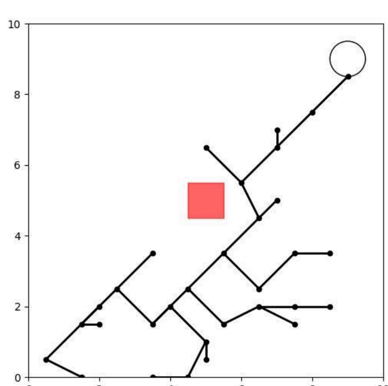


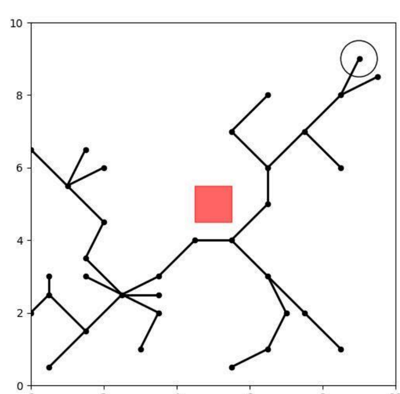
PROBLEM 3

```
n = 40
epsilon = 1.5
r_star = 4.0
init_robot_cell = (0,0)
goal_pos = (9,9)
goal_rad = 0.5
```

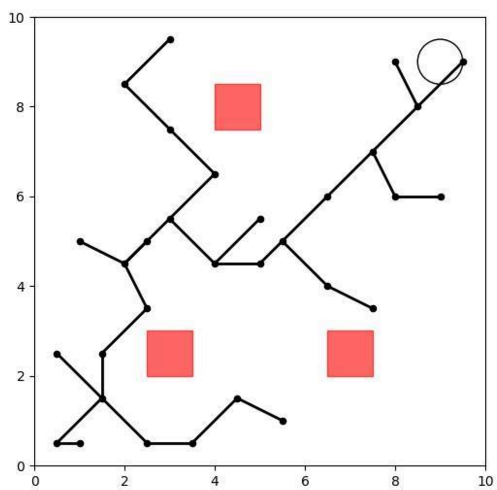
20 iterations

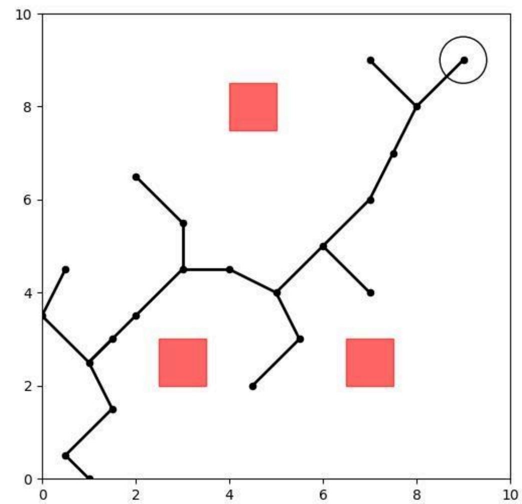
1 obstacle:
avg_runtime = 0.016522789001464845
variance_runtime = 3.297695595377044e-05
Average Path Length: 12.839383340488254



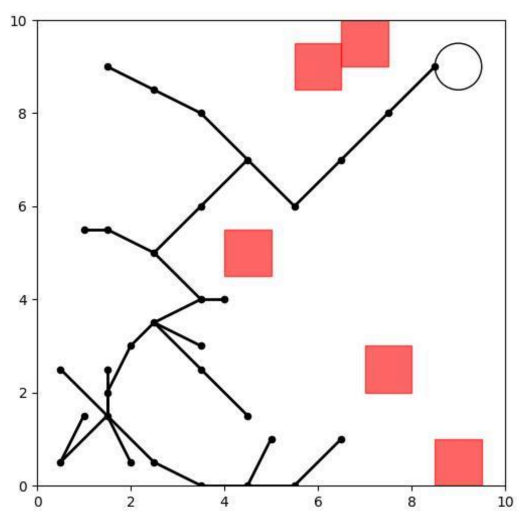


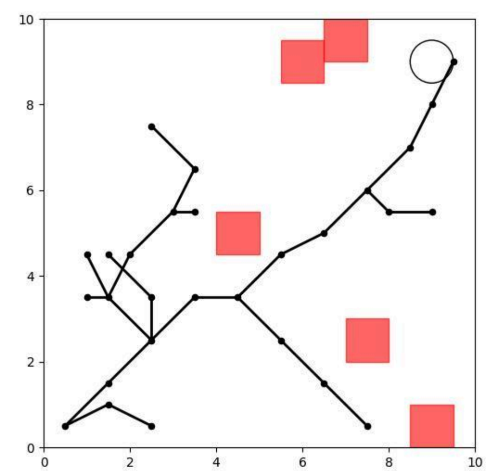
3 obstacles:
Same parameters as above
avg_runtime = 0.014844584465026855
variance_runtime = 1.747485638929902e-05
Average Path Length: 14.526452922967





5 OBSTACLES
Same parameters
Average Path Length: 15.090216591004431
avg_runtime = 0.014636055628458659
variance_runtime = 3.598577291060123e-05

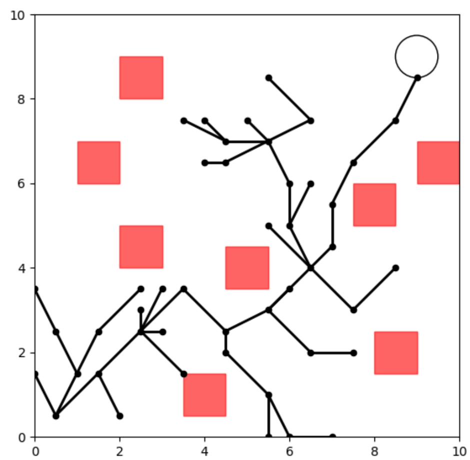


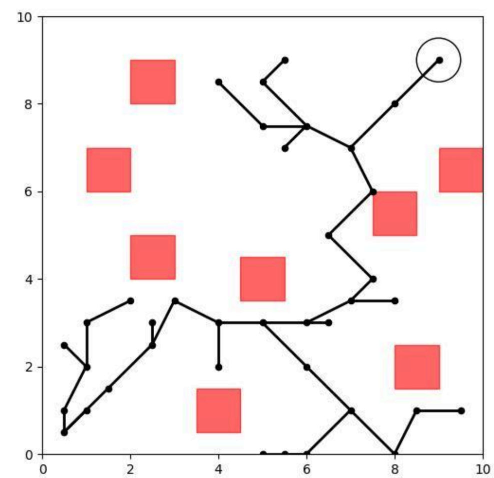


8 OBSTACLES:

```
n = 70
epsilon = 1.5
r_star = 4.0
init_robot_cell = (0,0)
goal_pos = (9,9)
goal_rad = 0.5
```

Average Path Length: 14.552282360937452
avg_runtime = 0.025932757059733073
variance_runtime = 5.143967370671837 6e-05





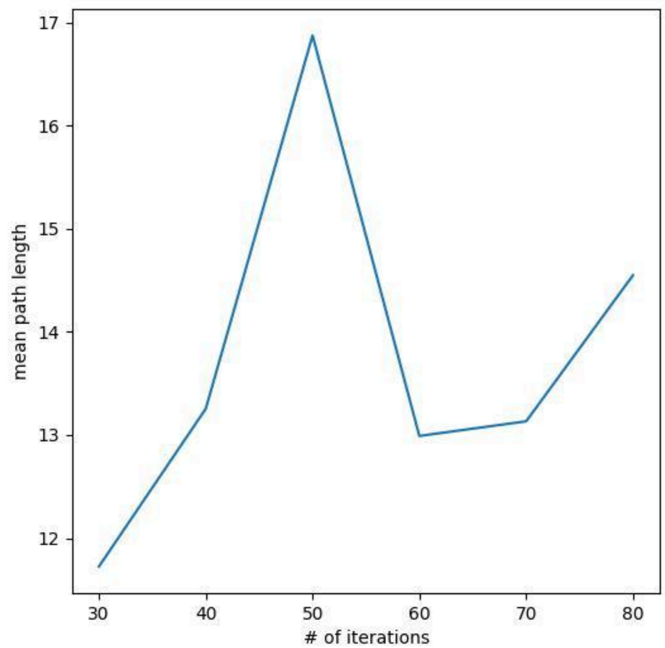


Figure showing relationship between average path length and number of iterations

OBSERVATIONS:

For lower iterations values, the average path length increases. This makes sense since the tree is less developed and creates suboptimal paths without rewiring the tree so much. As the number of iterations increases further, the average path length generally decreases which makes sense. The tree is no rewiring and finding more efficient connections.