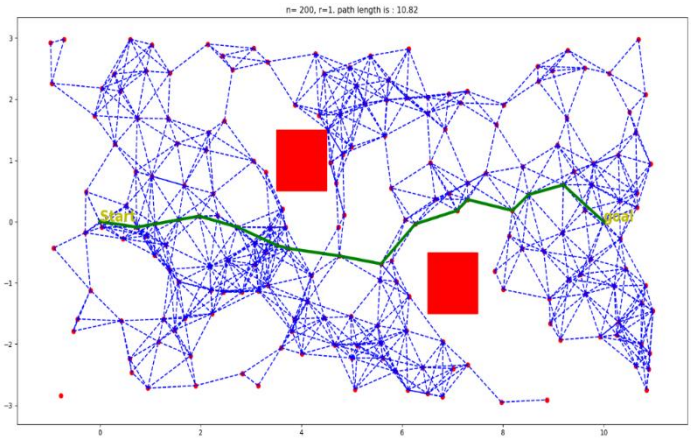
Homwork4:

Exercise 1:

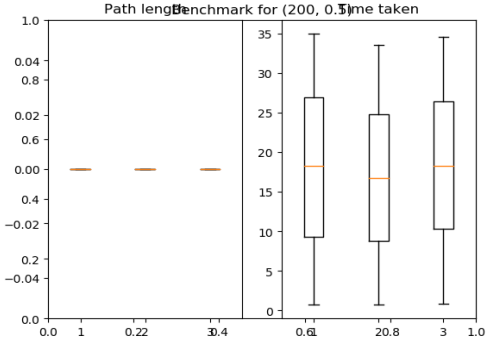
a- A start path is: [[4, 'end'], [4, 'k'], [4, 'c'], [0, 'start']] with 10 total iterations
b- TO get the Dijkstra algorithm, I remove the function that prioritize the node with the smallest cost + heuristic. I also did such as that the algorithm will have to visit all node before finding the shortest path or visit most nodes
c- Dijkstra path is: [[4, 'end'], [4, 'k'], [4, 'c'], [0, 'start']] with 234 total iterations
d- A* perform better than Dijkstra. I will choose Dijkstra because I can visit every node with it but will A* the [path can be found before very node is visited
e- I will add a function that look for the shortest cost in all direction possible at very node. That is at a node instead of going down the tree, the search will go in all direction of the graph only ignoring node that has already been visited

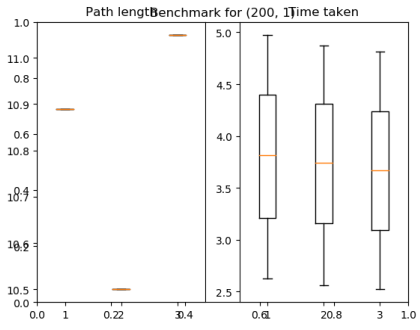
Exercise 2:

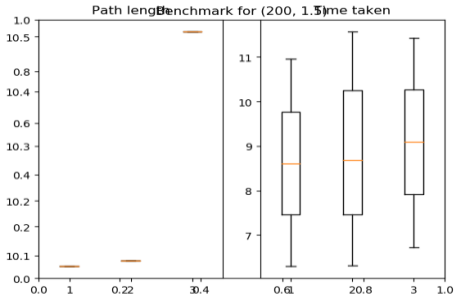
a- i- Path length in figure.

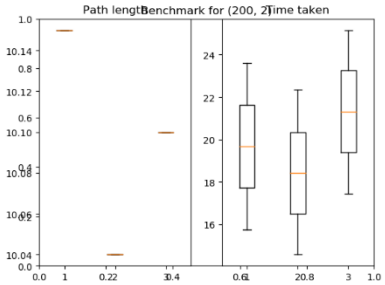


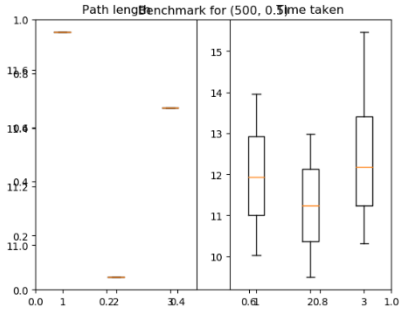
ii- Benchmark

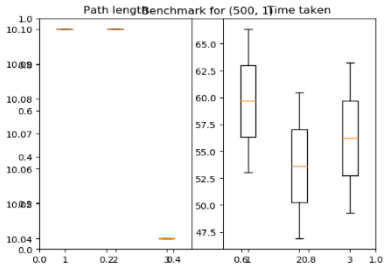


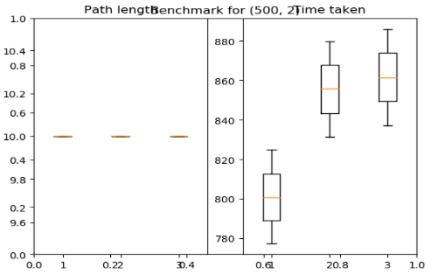


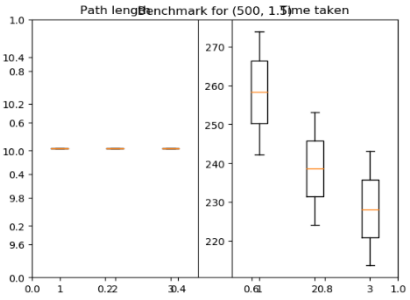












iii- From the performance graph above, We should take into account multiple criteria to choose the optimal n,r

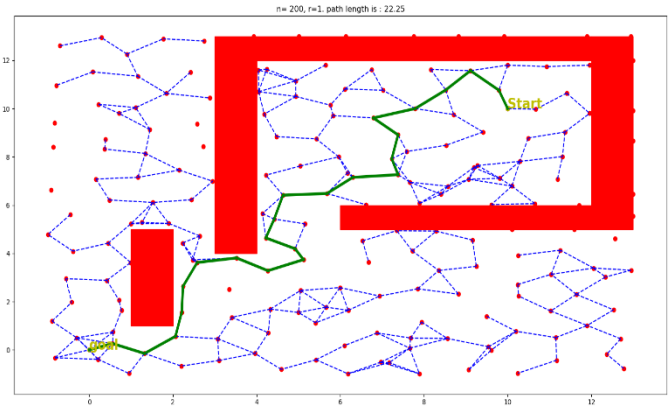
- Time: Hour long it takes to find a solution
- Path length: The smaller path length
- Uniformity of path length: less deviation between all three runs

Using the above principle n=200 and r=2, could be a good too choice. Less deviation between the three-path length, closed to the shortest path from all 8 and one of the smaller times to run. Also not path was found for the option (200,0.5)

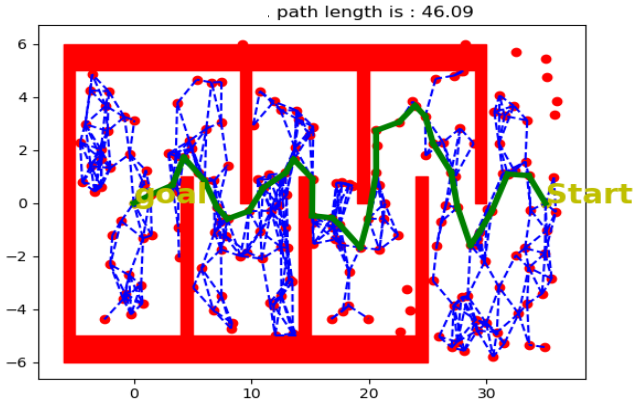
iv-

b-

i- W1 n=200, r=1 (it had to find a solution as the samples usually are sparse)



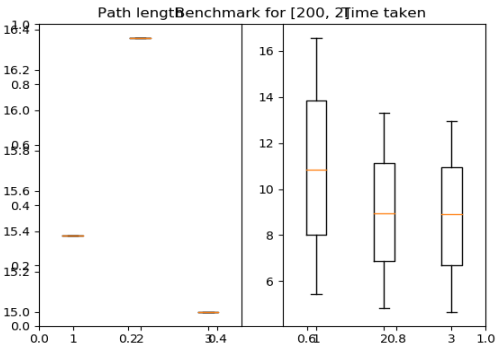
W2: n=200, r=2

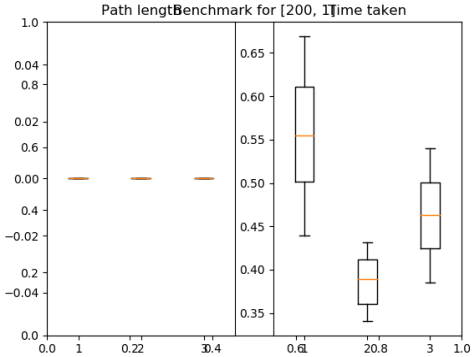


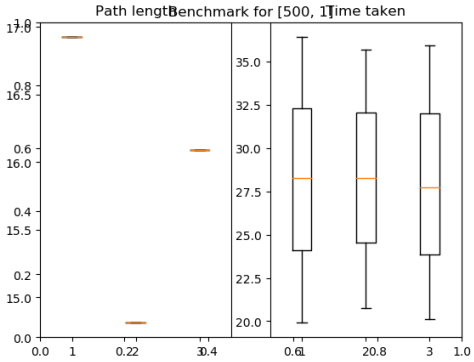
i- Benchmark

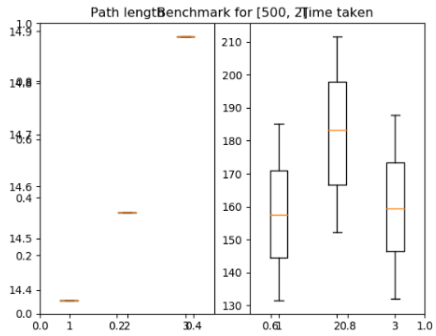
W1:

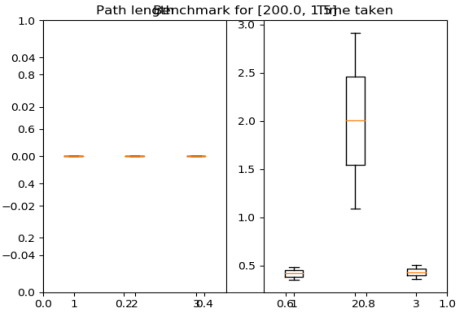
c-

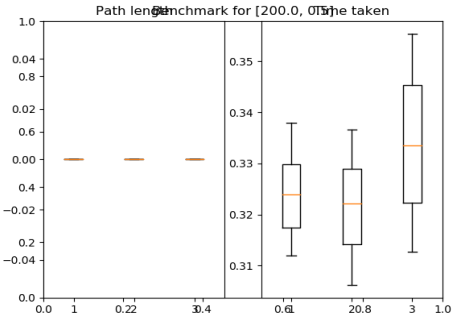


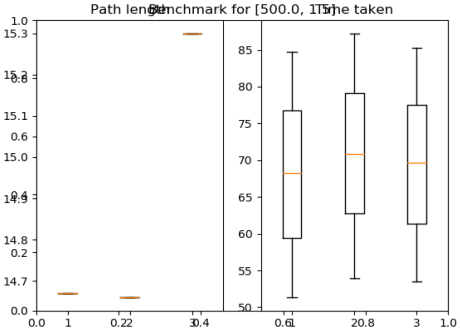


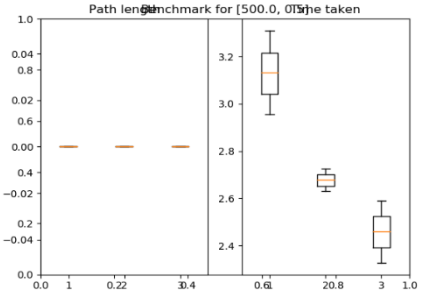






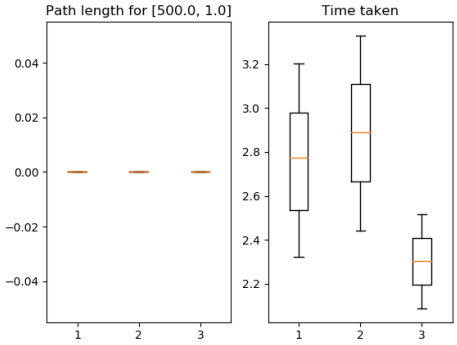


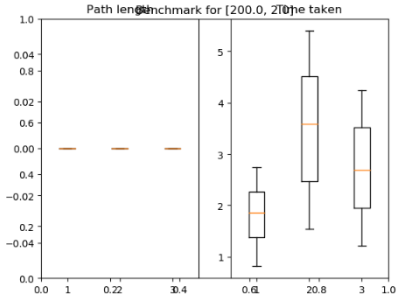


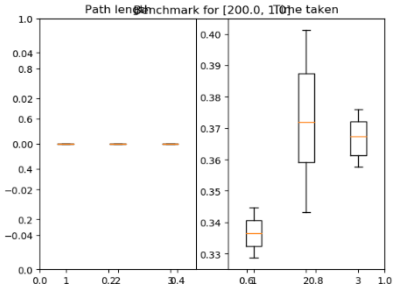


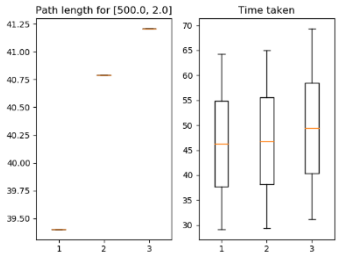
ii- The optimal n and r: n=500, n=1 here should be the optimal n and r. It is difficult to find a path for some options while the time taken for the other is higher with path lengths closed to the value chosen.

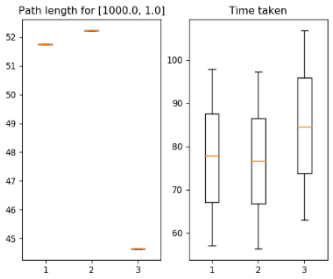
W2:

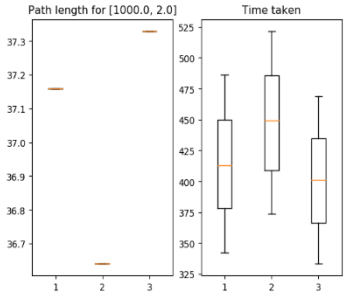












iii-The optimal n and r: n=500, n=2 here should be the optimal n and r. It is difficult to find a path for the first three options while the time taken for the las two is higher with path lengths closed to the value chosen.

iii- Path smoothing.

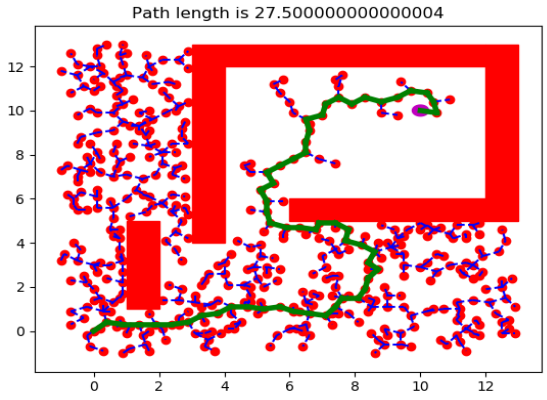
After path smoothing the optimal solution remains the same. The computation time to sample the workspace for each option did not vary too much as the sampling number is the same. Also, the initial path before smoothing did not vary much for each option for technique used. The smoothing has been done after a path is found by resampling between every two point of the path and reconnecting.

c- No, the PRM will not need modification. We can use the theat1, theta2 as x, y sample the c_space and construct the obstacle free path to the goal position thetas goal by avoiding obstacles. Therefore, we all be working in the c-space.

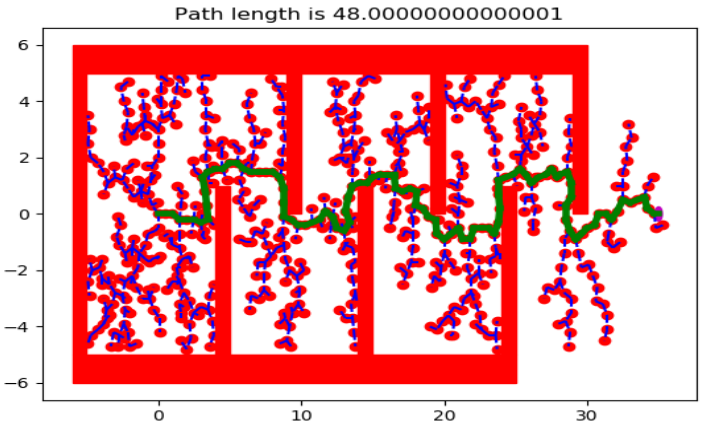
Exercise 3:

a- The plot of the path

W1:

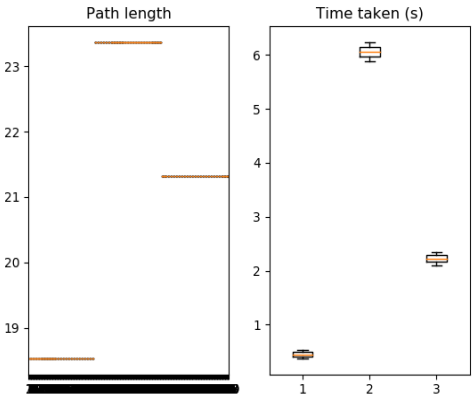


W2:

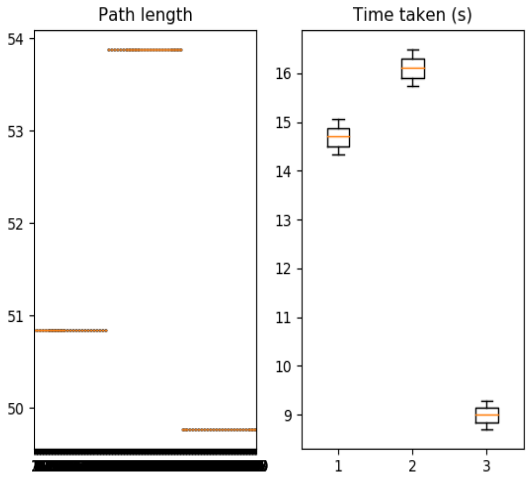


b- Benchmark

W1:



W2:



c- No, the RRT will not need modification. We can use the theat1, theta2 as x, y coordinate to grow the tree toward the goal position theta goals by avoiding obstacles. Therefore, we all be working in the c-space

Exercise 4:

Each planning algorithm has it advantages and disadvantages.

- Gradient descent Algorithm with potential function: This algorithm can give use a path to goal by moving the to the global minimum in the workspace that should be located at the goal. The algorithm is not that fast because we must visit to many points before finding the global minimum. Also, it is not complete due to local minimum.
- PRM: The probabilistic roadmap algorithm is a sampling approach to planning. We sample all our workspace randomly and eliminate samples that end in obstacle. Each point is them connect to their neighbor in a certain radius if the resulting edge does not end going through obstacle. After every not connected. We can then try to find a path to goal. This algorithm is not complete but probabilistic complete as it can find a path in infinite time and can not notify of there is not path.
- RRT: the one implemented in this assignment can find path to goal if it exits but cannot let us know if there is not path. We can sample the workspace randomly and connect it a node in our three that is closer to this sample as a distance r. The Bias option allow choosing the goal as our sample with x probability.
- The front wave algorithm: Similar to gradient descent with each point in the workspace sample and number in decreasing order from the start to goal. The nodes are number from the goal with obstacle nodes having same index number. Moving toward start while numbering, every number of an already number node should be x+1 index. Because the front ware algorithm is a breadth-first search we can know when there is not path to goal, thus complete with only on global minimum at the goal. However, sampling/discrete algorithm can be computationally intensive thus not suitable for high dimension and big workspaces.