Assignment-1

-Dhrumil Lotiya, 21d070026

```
Running All Planners
Iteration 1
start is  [1.260946, 2.775157, 2.195977, 5.109171, 3.785821]
goal is  [0.803507, 0.257061, 5.072260, 1.979594, 5.048289]
Running RRT
Running RRTStar
Running PRM
Algorithm | planningTime | numNodes | planLength | planQuality
RRT | 0.048214 | 126 | 18 | 12.667935
RRTStar | 0.225861 | 1139 | 16 | 11.289468
PRM | 0.337380 | 1839 | 65 | 42.617209
-----------------------------------

Iteration 2
start is  [0.998133, 0.600922, 2.325414, 1.977742, 1.664841]
goal is  [0.868853, 2.425340, 4.907467, 1.484522, 0.928177]
Running RRT
Running RRTStar
Running PRM
Algorithm | planningTime | numNodes | planLength | planQuality
RRT | 0.000171 | 7 | 6 | 3.286054
RRTStar | 0.150942 | 1008 | 6 | 3.286054
PRM | 0.613837 | 2446 | 32 | 20.364850
-----------------------------------

Iteration 3
start is  [0.177505, 1.026257, 2.184316, 1.315529, 4.393898]
goal is  [1.432499, 3.157700, 0.343126, 1.648361, 6.206638]
Running RRT
Running RRTStar
Running PRM
Algorithm | planningTime | numNodes | planLength | planQuality
RRT | 0.000183 | 7 | 6 | 3.639292
RRTStar | 0.154851 | 1006 | 6 | 3.592331
PRM | 1.048080 | 3226 | 11 | 7.200053
-----------------------------------

Iteration 4
start is  [0.615684, 3.730609, 1.715115, 1.075135, 3.585744]
goal is  [1.643709, 2.132899, 6.272680, 3.396341, 5.617907]
Running RRT
Running RRTStar
Running PRM
Algorithm | planningTime | numNodes | planLength | planQuality
RRT | 0.020443 | 106 | 12 | 7.940393
RRTStar | 0.170856 | 1043 | 16 | 11.174684
PRM | 3.467249 | 5826 | 17 | 10.961651
-----------------------------------

Iteration 5
start is  [1.564795, 3.193794, 1.005559, 0.257164, 1.470590]
goal is  [1.650443, 5.177416, 2.245896, 1.962292, 2.917021]
Running RRT
Running RRTStar
Running PRM
Algorithm | planningTime | numNodes | planLength | planQuality
RRT | 4.578441 | 2037 | 27 | 19.657030
RRTStar | 22.443722 | 6427 | 19 | 12.168472
```

```
----------------------------------

Iteration 6
start is  [1.278843, 1.767993, 0.357579, 2.719805, 3.605124]
goal is  [0.608719, 0.997844, 2.259501, 4.928406, 6.112846]
Running RRT
Running RRTStar
Running PRM
Algorithm | planningTime | numNodes | planLength | planQuality
RRT | 0.000482 | 7 | 7 | 3.978201
RRTStar | 0.095999 | 1008 | 7 | 3.978201
PRM | 0.288996 | 2173 | 29 | 19.504392
----------------------------------

Iteration 7
start is  [1.464206, 2.271548, 2.753551, 2.369755, 1.458639]
goal is  [0.399280, 1.944180, 0.945028, 3.152831, 4.313935]
Running RRT
Running RRTStar
Running PRM
Algorithm | planningTime | numNodes | planLength | planQuality
RRT | 0.000111 | 6 | 6 | 3.643887
RRTStar | 0.094063 | 1006 | 6 | 3.643887
PRM | 0.417595 | 2610 | 13 | 8.265724
----------------------------------

Iteration 8
start is  [0.581655, 6.103587, 2.581303, 2.580816, 4.747504]
goal is  [1.642706, 2.825502, 0.646351, 4.621982, 0.736192]
Running RRT
Running RRTStar
Running PRM
Algorithm | planningTime | numNodes | planLength | planQuality
RRT | 0.283944 | 225 | 23 | 17.142892
RRTStar | 0.346176 | 1161 | 22 | 15.256480
PRM | 3.681679 | 7247 | 25 | 15.788507
----------------------------------

Iteration 9
start is  [0.966125, 2.471113, 2.035044, 3.095907, 2.556551]
goal is  [1.264254, 1.634296, 1.560129, 4.188065, 1.355818]
Running RRT
Running RRTStar
Running PRM
Algorithm | planningTime | numNodes | planLength | planQuality
RRT | 0.000085 | 4 | 4 | 2.248893
RRTStar | 0.091687 | 1004 | 4 | 1.910303
PRM | 0.227157 | 1927 | 10 | 6.177193
----------------------------------

Iteration 10
start is  [1.671577, 3.197627, 1.050648, 0.719302, 0.639950]
goal is  [1.690375, 5.520205, 1.129270, 5.531364, 3.206646]
Running RRT
Running RRTStar
Running PRM
Algorithm | planningTime | numNodes | planLength | planQuality
RRT | 15.893310 | 4978 | 31 | 22.868449
```

```
-----------------------------------

Iteration 11
start is  [0.061252, 2.106365, 0.622913, 5.490212, 3.780521]
goal is  [1.689418, 5.128557, 1.895210, 0.426150, 5.338904]
Running RRT
Running RRTStar
Running PRM
Algorithm | planningTime | numNodes | planLength | planQuality
RRT | 40.957110 | 7588 | 24 | 17.293367
RRTStar | 16.358384 | 5717 | 27 | 18.437435
PRM | 3.834884 | 7807 | 26 | 16.706582
-----------------------------------

Iteration 12
start is  [0.328452, 0.239218, 3.321170, 0.530177, 1.478272]
goal is  [0.306321, 1.842464, 0.120570, 2.721609, 2.315093]
Running RRT
Running RRTStar
Running PRM
Algorithm | planningTime | numNodes | planLength | planQuality
RRT | 0.000112 | 7 | 7 | 4.279880
RRTStar | 0.095099 | 1007 | 7 | 4.279880
PRM | 0.129138 | 1441 | 15 | 9.182529
-----------------------------------

Iteration 13
start is  [0.067238, 0.596167, 3.235568, 0.164832, 3.247338]
goal is  [1.472428, 0.724970, 3.533663, 0.383475, 5.328332]
Running RRT
Running RRTStar
Running PRM
Algorithm | planningTime | numNodes | planLength | planQuality
RRT | 0.000083 | 6 | 5 | 2.541328
RRTStar | 0.091051 | 1006 | 5 | 2.541328
PRM | 0.420676 | 2616 | 9 | 5.800387
-----------------------------------

Iteration 14
start is  [1.701481, 0.465407, 1.759654, 5.758657, 5.091751]
goal is  [0.148015, 2.218497, 1.816182, 1.531542, 0.044807]
Running RRT
Running RRTStar
Running PRM
Algorithm | planningTime | numNodes | planLength | planQuality
RRT | 0.000172 | 10 | 10 | 6.987841
RRTStar | 0.095410 | 1018 | 12 | 8.114055
PRM | 1.580129 | 5055 | 18 | 11.879896
-----------------------------------

Iteration 15
start is  [0.916553, 5.523853, 2.873842, 0.100197, 2.911768]
goal is  [0.172836, 1.714345, 1.776988, 4.574422, 0.383370]
Running RRT
Running RRTStar
Running PRM
Algorithm | planningTime | numNodes | planLength | planQuality
RRT | 0.043344 | 81 | 19 | 13.384500
```

```
----------------------------------

Iteration 16
start is  [0.029830, 0.179713, 1.242377, 2.267843, 2.904027]
goal is  [0.818274, 1.363888, 4.556009, 2.060651, 3.631731]
Running RRT
Running RRTStar
Running PRM
Algorithm | planningTime | numNodes | planLength | planQuality
RRT | 0.000103 | 7 | 7 | 3.959095
RRTStar | 0.093882 | 1006 | 6 | 3.684638
PRM | 0.239901 | 1975 | 35 | 22.869239
----------------------------------

Iteration 17
start is  [1.314373, 1.306915, 5.410334, 3.714579, 1.295769]
goal is  [1.746366, 0.898892, 3.632315, 5.686211, 1.898009]
Running RRT
Running RRTStar
Running PRM
Algorithm | planningTime | numNodes | planLength | planQuality
RRT | 0.000629 | 10 | 6 | 3.775870
RRTStar | 0.126288 | 1102 | 11 | 7.788188
PRM | 0.280388 | 2131 | 36 | 24.241410
----------------------------------

Iteration 18
start is  [1.185437, 0.855103, 4.075630, 0.055178, 1.975696]
goal is  [1.872000, 0.866725, 2.799824, 3.886202, 6.242917]
Running RRT
Running RRTStar
Running PRM
Algorithm | planningTime | numNodes | planLength | planQuality
RRT | 0.056842 | 246 | 11 | 7.265661
RRTStar | 4.916707 | 3449 | 16 | 11.024212
PRM | 1.190231 | 4300 | 19 | 12.703375
----------------------------------

Iteration 19
start is  [0.426218, 1.742642, 0.215593, 4.096709, 0.707908]
goal is  [1.099413, 0.478845, 1.140681, 3.750850, 0.141873]
Running RRT
Running RRTStar
Running PRM
Algorithm | planningTime | numNodes | planLength | planQuality
RRT | 0.000047 | 4 | 4 | 1.829256
RRTStar | 0.093497 | 1004 | 4 | 1.829256
PRM | 0.288397 | 2130 | 15 | 9.018920
----------------------------------

Iteration 20
start is  [0.982349, 0.219143, 3.429674, 0.405654, 4.714608]
goal is  [1.730509, 0.242792, 0.075942, 1.727338, 1.686718]
Running RRT
Running RRTStar
Running PRM
Algorithm | planningTime | numNodes | planLength | planQuality
RRT | 0.000118 | 8 | 8 | 4.766843
```

```
----------------------------------

Final Results!
Algorithm | avgPlanningTime | avgNumNodes |avgPlanQuality
RRT | 3.094197 | 773 | 8.157833
RRTStar | 2.487050 | 1776 | 8.178167
PRM | 1.893861 | 4434 | 14.699553
----------------------------------
```

=>Interpretations:-

- **Planning Time:** PRM demonstrates the shortest average planning time, followed by RRT*, and then RRT. This suggests that PRM efficiently finds feasible paths within the configuration space and problem instances.
- **Number of Nodes:** PRM generates the highest average number of nodes, indicating its thorough exploration of the configuration space. RRT* generates more nodes than RRT due to its optimization techniques.
- **Plan Quality:** PRM delivers the highest average plan quality, implying it tends to find more optimal paths. RRT* balances planning time and plan quality, while RRT may be preferred for simpler problems or resource-constrained scenarios.

Therefore, in conclusion, RRT* looks like the most suitable planner for the environment as other planning methods have either poor performance in terms of planning time or require very high computational resources for correct implementation.

=>Issues that most suitable planner (RRT*) still has:-

- **Local Optima Trapping:** Despite RRT* being an optimal path improvement over RRT, it may still struggle with escaping local optima in complex or cluttered environments. This can lead to suboptimal or inefficient paths.
- **Sensitivity to Parameters:** The performance of these planners can be highly sensitive to their hyperparameters, such as the exploration

radius in RRT* .Tuning these parameters for optimal performance across different scenarios can be challenging.
- **Dynamic Environments:** None of these planners explicitly handle dynamic environments or moving obstacles, which can pose significant challenges for real-world applications where the environment is not static.

<u>=>Ways to improve the planner:-</u>
- **Optimization Techniques:** Incorporating optimization methods to refine the generated paths could lead to smoother and more efficient trajectories. This may include trajectory optimization, path smoothing algorithms, or using methods like A* search for local path refinement.
- **Parameter Optimization:** Developing automated methods for tuning the planner's hyperparameters based on the specific characteristics of the environment or task requirements could enhance performance. Techniques like grid search, genetic algorithms, or Bayesian optimization could be explored for this purpose.

To generate the above results, we used map1.txt and ran the planners with 20 randomly generated start and goal pairs (randomly generated the pairs once and then fixed those for all the planners).

<u>*Note:</u> I calculated the results using map1.txt and it was during the last moment before submitting that we had to use map2.txt . SInce, there wasn’t enough time left to generate new points now and run everything again, I have stuck to map1.txt. Anyways, I think since we have to compare the results only, it won’t matter much.

To generate this results, I added an extra option, 4, which :
1. Ignore startQ and goalQ
2. Generate a non-colliding start and goal joint configuration

3. Run all the planners, collecting important statistics
4. If any of the planners
5. Repeating step 2 and 3 20 times.
6. Printing all the statistics and averages.

Most algorithms, including RRT, RRT*, and PRM, adhere closely to their respective pseudocode. However, there are some specific adaptations and enhancements:

1. ***RRT*:*** After reaching the goal, RRT* expands an additional 1,000 nodes to refine the path, improving its accuracy.
2. **PRM:** In PRM, I employ a technique where both the start and goal configurations are added to the list of nodes at the beginning. Then, the algorithm stops adding nodes once the subgraphs containing the start and goal nodes connect to each other.
3. **Radius in algorithms:** Algorithms utilizing a radius parameter, such as RRT* and PRM, use a value of 1000 for this hyperparameter. This value is derived from a combination of factors including gamma, hyperball volume, and 1/log(2), ensuring a balanced exploration-exploitation trade-off. Specifically, 1000 is calculated as (gamma/hyperBallVolume * log(2)). This radius value influences the search space exploration and connection of nodes in the algorithm.

<u>Other changes in code which require mentions for proper compilation:-</u>

- ☐ Added a new definition #define ALL 3 in addition to PRM,RRT,RRT*. So, if we want to run all the planners for the same map, we can apply start and goal points as input with (whichPlanner set to =3).
- ☐ I didn’t change anything in grader.py to generate these 20 results because I didn’t have pandas installed and the internet was down, so I didn’t want to waste time waiting for it to come back.

- ☐ Other than these, there aren’t any changes in the code which could affect compilation. It can be run in the way described in PDF(Question PDF).