Project: Build a Forward Planning Agent **Date:** 2021/12/15

Student: Serhat Gungor

I've coded the required parts in the **my_planning_graph.py** file as requested. The descriptions under the pseudocode folder helped me to complete the missing parts in the functions. I've used the workspace provided by Udacity to complete the project. I also test my code with the unit tests provided with the below command.

```
$ python -m unittest -v
```

All tests are executed and passed successfully.

All strategies are executed with the below command and outputs are saved to result1.txt.

```
$ python run_search.py -p 1 2 -s 1 2 3 4 5 6 7 8 9 10 11 > results1.txt
```

I've analysed the output and put the results into the below table.

Problem	Name of the Algorithm	Number of Actions	Expansions	Goal Tests	New Nodes	Plan Length	Time to complete (in seconds)
Problem 1	breadth_first_search	20	43	56	178	6	0.006048254999768687
Problem 1	**depth_first_graph_search**	**20**	**21**	**22**	**84**	**20**	**0.0033795150002333685**
Problem 1	uniform_cost_search	20	60	62	240	6	0.009576515999924595
Problem 1	greedy_best_first_graph_search with h_unmet_goals	20	7	9	29	6	0.001559328999974241
Problem 1	greedy_best_first_graph_search with h_pg_levelsum	20	6	8	28	6	0.46008576499980336
Problem 1	greedy_best_first_graph_search with h_pg_maxlevel	20	6	8	24	6	0.32669202299985045
Problem 1	greedy_best_first_graph_search with h_pg_setlevel	20	6	8	28	6	0.5776511809999647
Problem 1	astar_search with h_unmet_goals	20	50	52	206	6	0.010423342999729357
Problem 1	astar_search with h_pg_levelsum	20	28	30	122	6	1.1619366879999689
Problem 1	astar_search with h_pg_maxlevel	20	43	45	180	6	1.1719834590003302
Problem 1	astar_search with h_pg_setlevel	20	33	35	138	6	1.3749870599999667
Problem 2	breadth_first_search	72	2243	4609	30503	9	1.911493285999768
Problem 2	depth_first_graph_search	72	624	625	5602	**619**	3.0315413139996963
Problem 2	uniform_cost_search	72	**5154**	5156	46618	9	3.3105547799996202
Problem 2	greedy_best_first_graph_search with h_unmet_goals	72	17	19	170	9	0.01959469999974317
Problem 2	greedy_best_first_graph_search with h_pg_levelsum	72	9	11	86	9	10.063476838000042
Problem 2	greedy_best_first_graph_search with h_pg_maxlevel	72	27	29	249	9	20.364351934000297
Problem 2	greedy_best_first_graph_search with h_pg_setlevel	72	9	11	84	9	14.227307254999687
Problem 2	astar_search with h_unmet_goals	72	2467	2469	22522	9	2.222913088000041
Problem 2	astar_search with h_pg_levelsum	72	357	359	3426	9	258.5342292629998
Problem 2	astar_search with h_pg_maxlevel	72	2887	2889	26594	9	**1486.566629256004**
Problem 2	astar_search with h_pg_setlevel	72	1037	1039	9605	9	**1273.4373111599998**

I’ve observed below when I check the numbers after performing searches;

- For the both problems **depth_first_graph_search** has the largest Plan Length.
- When the number of actions increase as in the Problem 2 case, astar algorithm execution durations increase very high compared to other algorithms. Especially ***astar_search with h_pg_maxlevel*** and ***astar_search with h_pg_setlevel*** execution duration increases too high.
- ***breadth_first_search*** and ***greedy_best_first_graph_search with h_unmet_goals*** algorithms performed best in both problems in terms of execution time.
- ***uniform_cost_search*** consumed more memory since the number of expansions has increased too much with the number of actions increase as in the Problem 2.

Because of the above findings I’ve decided to exclude ***depth_first_graph_search***, ***uniform_cost_search***, ***astar_search with h_pg_maxlevel*** and ***astar_search with h_pg_setlevel*** algorithms for the Problem 3 and Problem 4.

I’ve run the searches with the below command for the Problem 3 and Problem 4.

```
$ python run_search.py -p 3 4 -s 1 4 5 6 7 8 9 > results2.txt
```

Outputs are as below.

Problem	Name of the Algorithm	Number of Actions	Expansions	Goal Tests	New Nodes	Plan Length	Time to complete (in seconds)
Problem 3	breadth_first_search	88	14663	18098	129625	12	10.6202921900001
Problem 3	**greedy_best_first_graph_search with h_unmet_goals**	**88**	**25**	**27**	**230**	**15**	**0.0365858269999535**
Problem 3	greedy_best_first_graph_search with h_pg_levelsum	88	25	16	126	14	22.268172348000007
Problem 3	greedy_best_first_graph_search with h_pg_maxlevel	88	21	23	195	13	27.20469433800008
Problem 3	greedy_best_first_graph_search with h_pg_setlevel	88	35	37	345	17	79.42815533399994
Problem 3	astar_search with h_unmet_goals	88	7388	7390	65711	12	8.42503173599971
Problem 3	astar_search with h_pg_levelsum	88	369	371	3403	12	408.9909707979996
Problem 4	breadth_first_search	104	99736	114953	944130	14	94.5770081390001
Problem 4	**greedy_best_first_graph_search with h_unmet_goals**	**104**	**29**	**31**	**280**	**18**	**0.060416020000047865**
Problem 4	greedy_best_first_graph_search with h_pg_levelsum	104	17	19	165	17	40.96184991400014
Problem 4	greedy_best_first_graph_search with h_pg_maxlevel	104	56	58	580	17	98.13891851400012
Problem 4	greedy_best_first_graph_search with h_pg_setlevel	104	10	109	1164	23	354.0742276320002
Problem 4	astar_search with h_unmet_goals	104	34330	34332	328509	14	55.115972457000225
Problem 4	astar_search with h_pg_levelsum	104	1208	1210	12210	15	2290.7155714360006

Q1. Which algorithm or algorithms would be most appropriate for planning in a very restricted domain (i.e., one that has only a few actions) and needs to operate in real time?

A1. Since it needs to operate in real time, execution duration is important in choosing an algorithm. From the above 4 problems Problem 1 has lesser actions and for this problem ***greedy_best_first_graph_search with h_unmet_goals*** perfroms the best in terms of speed. As a second algorithm ***depth_first_graph_search*** can be used since it perfroms second best in terms of speed.

Q2. Which algorithm or algorithms would be most appropriate for planning in very large domains (e.g., planning delivery routes for all UPS drivers in the U.S. on a given day)

A2. This problem is smilar to Problem 4 executed above. When the results are checked ***greedy_best_first_graph_search*** algortihms would performe better than the other ones since they have lesser node expansions. Having less expansions will lead shorter paths.

Q3. Which algorithm or algorithms would be most appropriate for planning problems where it is important to find only optimal plans?

A3. When all the results are checked***, breadth_first_search*** algorithm perfroms best in planning the optimal path. ***astar_search with h_unmet_goals*** has found the optimal path as well but execution duration is longer than ***breadth_first_search***.