Project Report

After running various search scenarios it became clear that for the same problem, the number of actions is going to be the same.

Problem 1 -> 20 actions

Problem 2-> 72 actions

Problem 3 -> 88 actions

Problem 4 -> 104 actions

Thus we don't really need the number of actions in our analysis to decide which search to run in Problems 3 and 4 since the number of actions are the same. We can base our analysis on the number of nodes, search time and plan length.

Since the number of actions remains constant, we can base our analysis on just the number of nodes.

I decided based on running all the various search on Problems 1 and 2 the following

- For my choice of uninformed search I chose the breadth first search. I decided not to use depth first graph search based on the plan length. It clearly stood out where the plan length was several times larger than all the other search methods.
- For the two greedy best first search with heuristics, I decided to look mostly at the problem to search time values and thus settled on h_unmet_goals and h_pg_levelsum.
- I used similar reasoning for the A* search and settled on h_unmet_goals and h_pg_levelsum. These 2 variants took the least search time for Problem 2.

Based on this I show below the various charts.

Number of nodes expanded against number of actions in the domain

	Problem 1 Nodes	Problem 2 Nodes
breadth_first_search	178	30503
depth_first_graph_search	84	5602
uniform_cost_search	240	46618
greedy_best_first_graph_search h_unmet_goals	29	170
greedy_best_first_graph_search h_pg_levelsum	28	86
greedy_best_first_graph_search h_pg_maxlevel	24	249
greedy_best_first_graph_search h_pg_setlevel	28	84
astar_search h_unmet_goals	206	22522
astar_search h_pg_levelsum	122	3426
astar_search h_pg_maxlevel	180	26594
astar_search h_pg_setlevel	138	9605

	Problem 3 Nodes	Problem 4 Nodes
breadth_first_search	129625	944130
greedy_best_first_graph_search h_unmet_goals	230	280
greedy_best_first_graph_search h_pg_levelsum	126	165
astar_search h_unmet_goals	65711	328509
astar_search h_pg_levelsum	3403	12210

The trend we can see from the above is that in the case of breadth first search the number of nodes increases by a very large number as the number of actions increase. The number of nodes also increase as the number of actions increase from Problems 1 to 4 for all the A* searches, however the increase is not as large/pronounced compared to breadth first search. The number of nodes is smallest for greedy best first search algorithms. Hence if we need the number of nodes to be contained then we need to go for one of the greedy best first search algorithms.

Search time against the number of actions in the domain.

	Problem 1 Search Time(in s)	Problem 2Search Time(in s)
breadth_first_search	0.006012216999977227	1.8561382049999793
depth_first_graph_search	0.004031956999951455	2.821429113000022
uniform_cost_search	0.008869645000004311	3.0883671610001784
greedy_best_first_graph_search h_unmet_goals	0.0016578580000441434	0.03412318100004086
greedy_best_first_graph_search h_pg_levelsum	0.3918519750000087	9.235865254000146
greedy_best_first_graph_search h_pg_maxlevel	0.2970295019999867	18.302485727999965
greedy_best_first_graph_search h_pg_setlevel	0.5463981130000093	13.657493634000048
astar_search h_unmet_goals	0.008764622000057898	2.0866116939998847
astar_search h_pg_levelsum	1.0698425819999784	230.22753026700025
astar_search h_pg_maxlevel	1.076494327999967	1341.520375557
astar_search h_pg_setlevel	1.2624898860000258	1216.431795478

	Problem 3 Search Time(in s)	Problem 4 Search Time(in s)
breadth_first_search	9.655530232999809	84.52975124899967
greedy_best_first_graph_search h_unmet_goals	0.0322112300000299	0.050581279999960316
greedy_best_first_graph_search h_pg_levelsum	19.856513640999992	36.0103674500001
astar_search h_unmet_goals	7.272704508000061	49.337386268000046
astar_search h_pg_levelsum	361.54942000000005	2074.967864246

Clearly there is one conclusion that we can make from the above charts and that is if we want the search time to be minimum, then we should use the greedy_best_first_graph_search h_unmet_goals search. This is clearly the fastest based on the results that can be seen. In the case of all the 4 problems, the greedy_best_first_graph_search h_unmet_goals always has the smallest search time. We can say that the uninformed searches have the smallest time if we consider all the three searches. The maximum search time is for the A* searches. The best search time is exhibited by the greedy best first search algorithms.

Length of the plans

	Problem 1 Plan Length	Problem 2 Plan Length
breadth_first_search	6	9
depth_first_graph_search	20	619
uniform_cost_search	6	9
greedy_best_first_graph_search h_unmet_goals	6	9
greedy_best_first_graph_search h_pg_levelsum	6	9
greedy_best_first_graph_search h_pg_maxlevel	6	9
greedy_best_first_graph_search h_pg_setlevel	6	9
astar_search h_unmet_goals	6	9
astar_search h_pg_levelsum	6	9
astar_search h_pg_maxlevel	6	9
astar_search h_pg_setlevel	6	9

	Problem 3 Plan Length	Problem 4 Plan Length
breadth_first_search	12	14
greedy_best_first_graph_search h_unmet_goals	15	18
greedy_best_first_graph_search h_pg_levelsum	14	17
astar_search h_unmet_goals	12	14
astar_search h_pg_levelsum	12	15

Based on the results that can be seen above, for problems 1 and 2, the plan length is same for all the search methods except depth first graph search. The plan length in the case of Problems 1 and 2 is 20 and 169 respectively. Due to this, I decided not to run this for Problem 3 and 4.

- 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?

  To operate in real time I would chose the greedy_best_first_graph_search h_unmet_goals search. This takes the least amount of time for the search to complete.

- 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)

  For this case I would choose astar_search h_unmet_goals since it has a large number of nodes. I would choose this over breadth first search since the search time for astar_search h_unmet_goals is smaller than breadth_first_search

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

  In this case, I would look at the plan length and choose A* searches over the greedy best first searches. Based on the number for Problem 3 and 4, breadth_first_search has the smallest plan length but I will not choose an uninformed search for a planning problem.