Project: Classical Planning

Project Overview: The objective of this project is to implement a classical search agent which will perform a planning task, that is to plan a path. The planning involves planning paths to deliver Cargo using Airplanes. There are several cargo and planes and we need to deliver the cargo from one place to another.

Problem Description:

First, we have the following things:

Cargos	**:**	**e.g.: C1, C2**
Planes	**:**	**e.g.: P1, P2**
Airports	**:**	**e.g. SFO, JFK**

And we have to compare 11 different search algorithms out of which:

- 3 are uninformed search methods (breadth first, depth first, and uniform cost search)
- 8 with heuristics (A star and greedy best first graph search with the heuristics of unmet goals, maxlevel, levelsum and setlevel)

These algorithms are tested against 4 different planning problems which are of increasing complexity.

Problem1:

Precondition: [At(C1, SFO) AND At(C2, JFK)AND At(P1, SFO) AND At(P2, JFK) AND At(P3, ATL)]

Goal: [At(C1, JFK) AND At(C2, SFO)]

Problem2:

Precondition: [At(C1, SFO) AND At(C2, JFK) AND At(C3, ATL) AND At(P1, SFO) AND At(P2, JFK)]

Goal: [At(C1, JFK) AND At(C2, SFO) AND At(C3, SFO)]

Problem3:

Precondition: [At(C1, SFO) AND At(C2, JFK) AND At(C3, ATL) AND At(C4, ORD) AND At(P1, SFO) AND At(P2, JFK)]

Goal: [At(C1, JFK) AND At(C2, SFO) AND At(C3, JFK) AND At(C4, SFO)]

Problem4.

Precondition: [At(C1, SFO) AND At(C2, JFK) AND At(C3, ATL) AND At(C4, ORD) AND At(C5, ORD) AND At(P1, SFO) AND At(P2, JFK)]

Goal: [At(C1, JFK) AND At(C2, SFO) AND At(C3, JFK) AND At(C4, SFO) AND At(C5, JFK)]

Results:

The results of all of the search algorithms on the 4 problems are documented below:

Problem	Algorithm	Actions	Expansions	Goal	New Nodes	Time(S)	Plan Length
Air Cargo Problem 1	breadth_first_search	20	43	56	178	0.0051207029996476194	6
	depth_first_graph_search	20	21	22	84	0.0033476449998488533	20
	uniform_cost_search	20	60	62	240	0.008797023000170157	6
	greedy_best_first_graph_search with h_unmet_goals	20	7	9	29	0.0021083980000184965	6
	greedy_best_first_graph_search with h_pg_levelsum	20	6	8	28	0.1660280410001178	6
	greedy_best_first_graph_search with h_pg_maxlevel	20	6	8	24	0.1660280410001178	6
	greedy_best_first_graph_search with h_pg_setlevel	20	7	9	31	0.45806612399974256	7
	astar_search with h_unmet_goals	20	50	52	206	0.00872173700008716	6
	astar_search with h_pg_levelsum	20	28	30	122	0.39400213099997927	6
	astar_search with h_pg_maxlevel	20	43	45	180	0.4201223709997066	6
	astar_search with h_pg_setlevel	20	51	53	208	1.1304515639999408	6
Air Cargo Problem 2	breadth_first_search	72	3343	4609	30503	1.7777600319996054	9
	depth_first_graph_search	72	624	625	5602	2.6935896929999217	619
	uniform_cost_search	72	5154	5156	46618	3.1492440900001384	9
	greedy_best_first_graph_search with h_unmet_goals	72	17	19	170	0.018240809999952035	9
	greedy_best_first_graph_search with h_pg_levelsum	72	9	11	86	3.4569820609995077	9
	greedy_best_first_graph_search with h_pg_maxlevel	72	27	29	249	5.250136689999636	9
	greedy_best_first_graph_search with h_pg_setlevel	72	26	28	232	14.504780448999554	10
	astar_search with h_unmet_goals	72	2467	2469	22522	2.1025259640000513	9
	astar_search with h_pg_levelsum	72	357	359	3426	94.01699124900006	9
	astar_search with h_pg_maxlevel	72	2887	2889	26594	550.4258675410001	9

	astar_search with h_pg_setlevel	72	2102	2104	19395	1847.872014048	9
Air Cargo Problem 3	breadth_first_search	88	14663	18098	129625	9.341425689999596	12
	depth_first_graph_search	88	408	409	3364	1.0095120759997371	392
	uniform_cost_search	88	18510	18512	161936	13.724576263000017	12
	greedy_best_first_graph_search with h_unmet_goals	88	25	27	230	0.033723187000759935	15
	greedy_best_first_graph_search with h_pg_levelsum	88	14	16	126	8.327938157000062	14
	greedy_best_first_graph_search with h_pg_maxlevel	88	21	23	195	8.229191463999086	13
	greedy_best_first_graph_search with h_pg_setlevel	88	42	44	405	67.17390005800007	18
	astar_search with h_unmet_goals	88	7388	7390	65711	8.020408319000126	12
	astar_search with h_pg_levelsum	88	369	371	3403	180.2969265289994	12
	astar_search with h_pg_maxlevel88	88	9580	9582	86312	3314.091718603	12
	astar_search with h_pg_setlevel	88	5963	5965	54668	9012.379132283	12
Air Cargo Problem 4	breadth_first_search	104	99736	114953	944130	86.47609445699999	14
	depth_first_graph_search	104	25174	25175	228849	3126.12998253243	24132
	uniform_cost_search	104	113339	113341	1066413	112.123152764	14
	greedy_best_first_graph_search with h_unmet_goals	104	29	31	280	0.06141603700001497	18
	greedy_best_first_graph_search with h_pg_levelsum	104	17	19	165	15.819746859000077	17
	greedy_best_first_graph_search with h_pg_maxlevel	104	56	58	580	25.61250781399997	17
	greedy_best_first_graph_search with h_pg_setlevel	104	114	116	1229	314.527089058	24
	astar_search with h_unmet_goals	104	34330	34442	328509	58.37871453499997	14
	astar_search with h_pg_levelsum	104	1208	1210	12210	1049.4098990159991	15
	astar_search with h_pg_maxlevel	104	62077	62077	599376	17238.9861977241	14
	astar_search with h_pg_setlevel	104	37912	37914	373328	51203.0934812847	14

Discussions:

- Use a table or chart to analyze the number of nodes expanded against number of actions in the domain

Measuring Space Complexity: Problem Complexity and Algorithm used

		Action			
		20	**72**	**88**	**104**
Algorithm	breadth_first_search	43	3343	14663	99736
	depth_first_graph_search	21	624	408	25174
	uniform_cost_search	60	5154	18510	113339
	greedy_best_first_graph_search with h_unmet_goals	7	17	25	29
	greedy_best_first_graph_search with h_pg_levelsum	6	9	14	17
	greedy_best_first_graph_search with h_pg_maxlevel	6	27	21	56
	greedy_best_first_graph_search with h_pg_setlevel	7	26	42	114
	astar_search with h_unmet_goals	50	2467	7388	34330
	astar_search with h_pg_levelsum	28	357	369	1208
	astar_search with h_pg_maxlevel	43	2887	9580	62077
	astar_search with h_pg_setlevel	51	2102	5963	37912

The above table compares increase in no of nodes with increase in problem space. Unsurprisingly, as the problem space increases, number of nodes also increase.

Among all the Algorithms, “greedy_best_first_graph_search” has expanded the smallest with a steady increase in no and the search space for it does not increase explosively unlike the other algorithms.
Also, as the problem becomes more complex, the informed search does a better job than all the three un-informed search algorithms.

Except for the depth first search, all the uniformed search methods expanded a much larger number of nodes than the informed searches. This is because the uniformed search methods have no guidance on where the goal state is, and therefore need to explore more space to find the goal.

Among uninformed search methods, depth first search is the most efficient, with the least number of nodes expanded. Depth first search expands the deepest node first as opposed to all the nodes in a layer as per breadth first/uniform cost search, and therefore has a much smaller branching factor.

For the search algorithms with heuristics, we can see that the greedy best first search does much better than a*, and that the level sum heuristic is the best in guiding the agent towards the final goal as both algorithms when using this heuristic expands the least number of nodes. Since the greedy algorithm does not consider the cost of the path it takes to reach a node, it does not need to consider previous nodes so it's memory requirements is smaller.

- Use a table or chart to analyze the search time against the number of actions in the domain

Measuring time complexity: Time required vs problem size and algorithm used

		Action			
		20	**72**	**88**	**104**
Algorithm	breadth_first_search	0.005120702999647619 4	1.7777600319996054	1.7777600319996054	86.47609445699999
	depth_first_graph_search	0.003347644999848853 3	2.6935896929999217	2.6935896929999217	3126.12998253243
	uniform_cost_search	0.008797023000170157	3.1492440900001384	3.1492440900001384	112.123152764
	greedy_best_first_graph_search with h_unmet_goals	0.002108398000018496 5	0.018240809999952035	0.018240809999952035	0.06141603700001497
	greedy_best_first_graph_search with h_pg_levelsum	0.1660280410001178	3.4569820609995077	3.4569820609995077	15.819746859000077
	greedy_best_first_graph_search with h_pg_maxlevel	0.1660280410001178	5.250136689999636	5.250136689999636	25.61250781399997
	greedy_best_first_graph_search with h_pg_setlevel	0.4580661239997425 6	14.504780448999554	14.504780448999554	314.527089058
	astar_search with h_unmet_goals	0.008721737000087 16	2.1025259640000513	2.1025259640000513	58.37871453499997
	astar_search with h_pg_levelsum	0.3940021309999792 7	94.01699124900006	94.01699124900006	1049.4098990159991
	astar_search with h_pg_maxlevel	0.4201223709997066	550.4258675410001	550.4258675410001	17238.9861977241
	astar_search with h_pg_setlevel	1.1304515639999408	1847.872014048	1847.872014048	51203.0934812847

The above table shows how the different algorithms perform in terms of time. For uniformed searches, the breadth first search takes the least time, as opposed to the depth first search, which takes much longer, especially as the problem size increases. This is because in the depth first search, if it goes down the 'wrong' path it will have to backtrack, and a bigger problem means it will spend a lot of time backtracking.

For the informed search algorithms, we can see that the heuristic chosen has a large impact on the time, e.g. using the unmet goals heuristic is about 1000 times faster than the set level heuristic! This is likely because the more complex heuristics take longer to compute, especially if the implementation is not efficient. However, if we compare the heuristics against the node expanded, we can see that the fastest heuristics also ended up with a lot more nodes expanded.

- Use a table or chart to analyze the length of the plans returned by each algorithm on all search problems

Measuring optimality: Algorithms achieving the shortest plan length solution

For the uniformed search algorithms, both breadth first and uniform cost searches are optimal. For the informed searches, both A star and greedy search are optimal for smaller problems, but as the problem size increases, A star gives the optimal solution for all the heuristics except for the level sum heuristic whereas the greedy algorithm slowly diverges away from the optimal solution—likely because as the problem size increases, the cost of the previous path has a bigger and bigger impact on the final cost and as the previous cost is ignored in greedy search, this means it doesn't find the most optimal solution.

		Action			
		20	**72**	**88**	**104**
Algorithm	breadth_first_search	6	9	12	14
	depth_first_graph_search	20	619	392	24132
	uniform_cost_search	6	9	12	14
	greedy_best_first_graph_search with h_unmet_goals	6	9	15	18
	greedy_best_first_graph_search with h_pg_levelsum	6	9	14	17
	greedy_best_first_graph_search with h_pg_maxlevel	6	9	13	17
	greedy_best_first_graph_search with h_pg_setlevel	7	10	18	24
	astar_search with h_unmet_goals	6	9	12	14
	astar_search with h_pg_levelsum	6	9	12	15
	astar_search with h_pg_maxlevel	6	9	12	14
	astar_search with h_pg_setlevel	6	9	12	14

Algorithms on Various scenarios:

1. ***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?***

Ans: For real time, we want an algorithm that runs fast, while giving a short plan length. For our case except for depth first search, all the other algorithms give an optimal solution. Since the greedy search algorithm gives the shortest times among them as well as the least number of nodes expanded and hence uses less memory, it should be a good choice in this scenario.

2. ***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)***

Ans: We would need an algorithm that runs fast even on large problem size (otherwise the day will be over before the planning calculation finishes), and does not expand too many nodes (otherwise the planning computation will run out of memory). A good choice is the greedy search algorithm, even if it doesn't give the most optimal solution, the solution isn't too far off optimal, and it runs quicker and expands less nodes as problem size increases as compared to the others.

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

Ans: From the experiments, we can see that A star, breadth first and uniform cost search all return the optimal plan, even as the problem size increases. If optimality is the only consideration, all of them are appropriate choices. If we need to optimize for memory and/or time as well, then we should use A star (with unmet goal heuristic) since this algorithm along with heuristic combination gives the least increase in both memory and time as problem size increases.

Conclusion

From the above set of experiments, we can see that informed search algorithms with the right heuristics can perform much better than uninformed search algorithms in terms of both speed and memory requirements, and still return optimal plans. from the experiments, the unmet goals heuristic is the most optimal for time, and the level sum heuristic is the most optimal for memory. Depending on the scenario complexity and computation requirements, we can tune the heuristic to suit our need.