Part 1

Question

Experiment and document metrics for non-heuristic planning solution searches

- Run uninformed planning searches for air_cargo_p1, air_cargo_p2, and air_cargo_p3; provide metrics on number of node expansions required, number of goal tests, time elapsed, and optimality of solution for each search algorithm. Include the result of at least three of these searches, including breadth-first and depth-first, in your write-up (breadth_first_search and depth_first_graph_search).
- If depth-first takes longer than 10 minutes for Problem 3 on your system, stop the search and provide this information in your report.

Report ¶

Search algorithms performed

- breadth_first_search
- depth_first_graph_search
- uniform_cost_search

Performance Analysis

The complete output of the search run can be found in the part_1_solution_searches.txt (part_1_solution_searches.txt) file included in the solution root dir. The following table is a summary of those results:

Air Cargo Problem	# of Fluents	Space Size	Search Algorithm	Expansions	Goal Tests	New Nodes	Plan Length	Time Ela
1	12	4,096	breadth_first_search	43	56	180	6	0.025619
			depth_first_graph_search	12	13	48	12	0.009327
			uniform_cost_search	55	57	224	6	0.031954
2	27	134,217,728	breadth_first_search	3,343	4,609	30,509	9	12.36400
			depth_first_graph_search	582	583	5,211	575	2.851154
			uniform_cost_search	4,853	4,855	44,041	9	10.66312
3	32	4,294,967,296	breadth_first_search	14,663	18,098	129,631	12	92.28724
			depth_first_graph_search	627	628	5,176	596	2.939254
			uniform_cost_search	18,223	18,225	159,618	12	47.21466

Performance Metrics Summary

At first glance, it appears that the most efficient search algorithm among the 3 compared is depth_first_graph_search. On each of the scenarios, the number of Expansions and Time Elapsed is significantly lower than the ones showed by the other two algorightms.

Let's take for example Air Cargo Problem 1. In that case, depth_first only needs 12 node Expansions before reaching a solution, whereas breadth_first and uniform_cost need 43 and 55 respectively (>300% expansions). Also, Time Elapsed is significantly lower for depth_first (9msecs as compared to 26 and 32). However, the Plan Length under depth_first turns out to be twice as much as in the other 2 cases (12 steps as opposed to 6).

This pattern is reflected in all the cargo problems. Most noticeably, for Air Cargo Problem 3, the Plan Length for depth_first is 596, whereas for both breadth_first and uniform_cost it's only 12. Nonetheless, the Time Elapsed for them is 1-1/2 mins and 47 secs respectively, whereas it is only 3 secs for depth_first. This shows that depth_first shows potential benefits as far as efficiency, but very poor benefits when it comes to optimality. On the other hand, both breadth_first and uniform_cost are significantly more effective (optimal), but less efficient.

In hindsight, this apparent behavior makes sense though. The depth_first approach deals with exploring a possible path until the end before trying an alternative path. This approach causes the algorithm to find a solution (meet the goal) faster in our case, as it tries to hop from airport to airport while testing the goal. The outcome is a sub-optimal solution found in relatively short time, making the plan length found by depth_first obviously high.

On the other hand, the breadth_first approach deals with testing all possible immediate alternatives before digging deeper into the next available path segments. This approach, although obviously slower, produces more optimal results; i.e., shorter plan lengths. A similar, yet more efficient outcome to breath_first is shown by uniform_cost. This is because this search algorithm underneath the covers is using a best_first approach (with a constant cost function in our case), which in practicallity produces as a more greedy breadth-first tree expansion.

Part 2

Question

Experiment and document: metrics of A* searches with these heuristics

- Run A* planning searches using the heuristics you have implemented on air_cargo_p1, air_cargo_p2 and air_cargo_p3. Provide metrics on number of node expansions required, number of goal tests, time elapsed, and optimality of solution for each search algorithm and include the results in your report.

Report

Performance Analysis

The complete output of the search run can be found in the part_2_solution_searches.txt (part_2_solution_searches.txt) file included in the solution root dir. The following table is a summary of those results:

Air Cargo Problem	# of Fluents	Space Size	Search Algorithm	Expansions	Goal Tests	New Nodes	Plan Length	Time Ela
1	12	4,096	A*_h_1	55	57	224	6	0.03798
			A*_h_ignore_precondition	41	43	170	6	0.03179
			A*_h_pg_levelsum	11	13	50	6	0.76202
2	27	134,217,728	A*_h_1	4,853	4,855	44,041	9	10.3347
			A*_h_ignore_precondition	1,450	1,452	13,303	9	3.71035
			A*_h_pg_levelsum	86	88	841	9	157.177
3	32	4,294,967,296	A*_h_1	18,223	18,225	159,618	12	45.5429
			A*_h_ignore_precondition	5,040	5,042	44,944	12	14.7686
			A*_h_pg_levelsum	318	320	2,934	12	865.748

Performance Metrics Summary

When running A* search using the 3 custom heuristics, we see an interesting result for plan length. They all produce the same plan length (i.e., optimality) for each problem regardless of the heuristics used. This is a little unintuitive, as one would expect a higher optimality for more advanced heuristics. It would be interesting to compare these heuristics in more complex problem spaces, and see if this result-optimality-vs-heuristic-complexity ratio holds true.

When it comes to speed efficiency, h_ignore_precondition is the fastest one by sometimes a factor of 3, whereas h_pg_levelsum turned out to be the slowed by a pretty significant factor. In fact, in cargo problem 3, h_pg_levelsum caused the algorithm to complete in > 10 mins (~14 mins to be exact). A simple assumption of assuming a single action is needed to reach the unfulfilled goals as a heuristic, as in the case of h_ignore_precondition, allows us to reach our goal faster. Assuming the optimality of these heuristics is maintained for more complex problems, this means h_ignore_precondition is the most attractive option, whereas h_pg_levelsum is the least attractive option when it comes to speed.

In terms of space efficiency, in this case h_pg_levelsum shone over the rest by a huge margin. In the cargo 3 problem, this heuristic performed at a ~1.5% of the space of h_1 (which happens to be the least space-efficient one) and at a ~6% of the space of h_ignore_precondition.

All in all, this means simpler heuristics may be the way to go when it comes to speed efficiency, wheras more complex heuristics would be better when it comes to space efficiency. A trade-off decision would have to be involved in the process of selecting proper heuristics for a problem.

Part 3

Question

- Provide an optimal plan for Problems 1, 2, and 3.
- Compare and contrast non-heuristic search result metrics (optimality, time elapsed, number of node expansions) for Problems 1,2, and 3. Include breadth-first, depth-first, and at least one other uninformed non-heuristic search in your comparison; Your third choice of non-heuristic search may be skipped for Problem 3 if it takes longer than 10 minutes to run, but a note in this case should be included.
- Compare and contrast heuristic search result metrics using A* with the "ignore preconditions" and "level-sum" heuristics for Problems 1, 2, and 3.
- What was the best heuristic used in these problems? Was it better than non-heuristic search planning methods for all problems? Why or why not? Provide tables or other visual aids as needed for clarity in your discussion.

Report

Optimal Plan for Problems 1, 2, 3

Problem 1

Load(C1, P1, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)

Problem 2

Load(C3, P3, ATL)
Fly(P3, ATL, SFO)
Unload(C3, P3, SFO)
Load(C1, P1, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)

Problem 3

Load(C2, P2, JFK)
Fly(P2, JFK, ORD)
Load(C4, P2, ORD)

Fly(P2, ORD, SFO)
Unload(C4, P2, SFO)
Load(C1, P1, SFO)
Fly(P1, SFO, ATL)
Load(C3, P1, ATL)
Fly(P1, ATL, JFK)
Unload(C3, P1, JFK)
Unload(C1, P1, JFK)
Unload(C2, P2, SFO)

Comparison/Contrasting of non-heuristic searches

Please see the *Performance Analysis* section in Part 1.

Comparison/Contrasting of heuristic searches

Please see the *Performance Analysis* section in Part 2.

Best Search Method Used

Problem	**Optimal Length**	**Best Non-Heuristic Search (time-wise)**	**Best Non-Heuristic Search (space-wise)**	**Best Heuristic Search (time-wise)**	**Best Heuristic Search (space-wise)**	**Best Overall Time-Wise**	**Best Overall Space-Wise**
1	6	Breadth-First (0.026 secs)	Breadth-First (43 expansions)	A* Ignore Precond (0.032 secs)	A* Level Sum (11 expansions)	Breadth-First	A* Level Sum
2	9	Uniform Cost (10.66 secs)	Breadth-First (3,343 expansions)	A* Ignore Precond (3.71 secs)	A* Level Sum (86 expansions)	A* Ignore Precond	A* Level Sum
3	12	Uniform Cost (47.22 secs)	Breadth-First (14,663 expansions)	A* Ignore Precond (14.77 secs)	A* Level Sum (318 expansions)	A* Ignore Precond	A* Level Sum

Heuristic and Non-Heuristic Best Plans

As described in the previous table, overall the best contender time-wise was A* Ignore Precond.

> Note that the result for Problem 1 seemed to favor breadth-first, but considering it was very close to A* Ignore Precond, and this was run only once (more runs would have to be done for proper benchmarking), I'd consider A* Ignore Precond as the fastest contender overall.

It's important to note that when it comes to space efficiency though, A* Level Sum is a better candidate. However, considering Level Sum performed very poorly when it comes to efficiency, **I'd pick A* Ignore Precond as the best planning method for the Air Cargo Problem space**.