Planning Problem Representation

- 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`).

In the following, tables are provided for each problem comparing breadth first search, depth first graph search, and uniform cost search. We can see that breadth first search and uniform cost search are both optimal in that they both identify the minimal course of action required to reach the goal. We can therefore use either of these algorithms to identify the optimal sequence of actions to solve each of the problems. However, neither of these is necessarily efficient. Depth first graph search, on the other hand, generally takes a significantly less amount of time, as well as fewer node expansions and goal tests, than the other two algorithms to reach the goal. It is therefore more efficient than the other two, but not necessarily optimal, as the sequence of actions it generates is generally of a much longer length than that generated by either of the other two algorithms.

Intuitively, this makes sense because both breadth first search and uniform cost search operate somewhat conservatively, exploring the shallowest layer or the layer with least cost (in the case of the latter) at a time, making sure that whenever they reach the goal, they are able to readily identify the minimal path to reach that goal from the initial state. Depth first graph search, on the other hand, fearlessly marches on across levels, hunting for the goal state at every step. Since the goal state is typically not in the first few layers, this gives depth first graph search the advantage of reaching the goal state quicker, but also the disadvantage that it identifies a more circuitous route to reach the same goal.

Problem 1

Optimal Sequence of Actions:
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)

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

	Breadth First Search	Depth First Graph Search	Uniform Cost Search
Node expansions	43	21	55
Goal Tests	56	22	57
Time Elapsed (s)	0.04	0.02	0.05
Plan Length	6	20	6

Problem 2

Optimal Sequence of Actions:
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Load(C3, P3, ATL)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)
Fly(P3, ATL, SFO)
Unload(C3, P3, SFO)

	Breadth First Search	Depth First Graph Search	Uniform Cost Search
Node Expansions	3315	1040	4807
Goal Tests	4572	1041	4809
Time Elapsed (s)	17.64	3.87	14.73
Plan Length	9	85	9

Problem 3

Optimal Sequence of Actions:
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P2, JFK, ORD)
Load(C4, P2, ORD)
Fly(P1, SFO, ATL)
Load(C3, P1, ATL)
Fly(P1, ATL, JFK)
Unload(C1, P1, JFK)
Unload(C3, P1, JFK)
Fly(P2, ORD, SFO)
Unload(C2, P2, SFO)
Unload(C4, P2, SFO)

	Breadth First Search	Depth First Graph Search	Uniform Cost Search
Node Expansions	14663	408	18236
Goal Tests	18098	409	18238
Time Elapsed (s)	138.76	2.52	67.42
Plan Length	12	392	12

- 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. 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?

A comparison of the “ignore preconditions” and “level-sum” heuristics is tabulated below for each of the problems. We note that both heuristics are admissible: “ignore preconditions” because ignoring the preconditions for any action(s) can only reduce the number of actions required to reach the goal state, not increase them, and “level-sum” because by definition it does not over-estimate the level cost needed to reach a goal. They are also both consistent, trivially so, as they would both satisfy the triangle inequality by saturating the equality, i.e. h(n1) = c(n1, a, n2) + h(n2), where n2 is a successor of n1 generated by action a, c(...) is the step cost of getting from n1 to n2, and h(n) denotes the estimated cost of reaching the goal from node n. Therefore, both heuristics are optimal and each generate plans to solve each of the problems with the optimal length. The optimal sequence of actions is already given above.
The better heuristic to use between these two seems to be the “ignore preconditions” heuristic, since it generally takes a lesser time than the “level-sum” heuristic, even though the latter typically requires fewer node expansions and goal tests. Using A* search with the “ignore preconditions” heuristic works about as good as uniform cost search, in that they both provide an optimal sequence of actions in approximately the same minimal time (at least, among the methods investigated), but perhaps uniform cost search is slightly better to use, given the lesser times recorded for each of the problems for that particular search strategy.

Problem 1

	H_ignore_preconditions	H_pg_levelsum
Node Expansions	42	11
Goal Tests	44	13
Time Elapsed (s)	0.07	1.14
Plan Length	6	6

Problem 2

	H_ignore_preconditions	H_pg_levelsum
Node Expansions	3539	148

Goal Tests	3541	150
Time Elapsed (s)	14.92	161.97
Plan Length	9	9

Problem 3

	H_ignore_preconditions	H_pg_levelsum
Node Expansions	13917	315
Goal Tests	13919	317
Time Elapsed (s)	84.78	563.37
Plan Length	12	12