- **Problem 1**

| Problem 1 | | | | | |
|---|---|---|---|---|---|
| **Planning Search Method** | **Expansions** | **Goal Tests** | **New Nodes** | **Plan length** | **Time elapsed (seconds)** |
| breadth_first_search | 43 | 56 | 180 | 6 | 0.033594774 |
| breadth_first_tree_search | 1,458 | 1,459 | 5,960 | 6 | 1.082238771 |
| depth_first_graph_search | 21 | 22 | 84 | 20 | 0.016030459 |
| depth_limited_search | 101 | 271 | 414 | 50 | 0.103159596 |
| uniform_cost_search | 55 | 57 | 224 | 6 | 0.040610458 |
| recursive_best_first_search h_1 | 4,229 | 4,230 | 17,023 | 6 | 3.075932729 |
| **greedy_best_first_graph_search h_1** | **7** | **9** | **28** | **6** | **0.006662805** |
| astar_search h_1 | 55 | 57 | 224 | 6 | 0.042004495 |
| astar_search h_ignore_preconditions | 41 | 43 | 170 | 6 | 0.033404464 |
| astar_search h_pg_levelsum | 11 | 13 | 50 | 6 | 1.016550344 |

For problem 1, all 10 search algorithms were run to completion in under 5 seconds. Of the 10 searches 8 were able to construct an optimal plan of length 6. Only the depth related searches (depth first graph search and depth limited search) generated plan length much longer than the rest for both uninformed and automatic heuristics.  The 6 actions generated by the other searches were all the same, but not necessarily in the same order. All search methods, with the exceptions of uniform cost search and A* search using a heuristic, generated differing numbers of node expansions, new nodes, and goal tests. Also, the computation times differed between search algorithms.

The optimal uninformed search method was breath-first search, where 43 nodes were expanded, there were 56 goal tests, 180 new nodes, and a computation time of 0.03359 seconds. The depth-first graph search generated fewer nodes and had a shorter computation time, however it generated a plan length much longer than it should have. The worst performing uninformed method was the breadth-first tree search, which generated 1,458 expansions, 1,459 goal states, and 5,960 new nodes. It also had the worst computation time of the uninformed methods, with 1.0822 seconds. The overall optimal method, by many factors, was the greedy best first graph search using a heuristic. It was able to generate the fewest number of expansions with 7, the lowest number of goal tests (9), and the smallest number of new nodes (28). It also had the lowest computation time of 0.00666 seconds. This contrasts with the recursive best first search with a heuristic, that performed worse than any other search method, uninformed or automatic, with 4,229 expansions, 4,230 goal tests, 17,023 new nodes, and a computation time of 3.0759 seconds.

For the A* searches, when ignoring-preconditions the number of expansions, goal states, new nodes, and computation time fell when compared to A* search using a heuristic. When using planning graph level-sum for A*, the number of expansions fell from 41 to 11, the number of goal tests fell from 43 to 13, and the number of new nodes fell from 170 to 50, when compared to the A* search ignoring the preconditions. The computation time for the planning graph level-sum method was higher than for the other two A* searches. As suggested in the book, the results here seem to be a reflection of the advantage that informed searches have over the uninformed searches. The results from the informed greedy best first graph search blew away the competition in terms of expansions, goal tests, new nodes, and computation time because of the inclusion of a heuristic.

The optimal sequence of actions (taken from greedy best first graph search using a heuristic) looks something like this:

Load(C1, P1, SFO)

Load(C2, P2, JFK)

Fly(P1, SFO, JFK)

Fly(P2, JFK, SFO)

Unload(C1, P1, JFK)

Unload(C2, P2, SFO)

- **Problem 2**

| Problem 2 | | | | | |
|---|---|---|---|---|---|
| **Planning Search Method** | **Expansions** | **Goal Tests** | **New Nodes** | **Plan length** | **Time elapsed (seconds)** |
| breadth_first_search | 3,343 | 4,609 | 30,509 | 9 | 8.688528224 |
| breadth_first_tree_search | - | - | - | - | ***> 10 minutes*** |
| depth_first_graph_search | 624 | 625 | 5,602 | 619 | 3.390025295 |
| depth_limited_search | - | - | - | - | ***> 10 minutes*** |
| uniform_cost_search | 4,852 | 4,854 | 44,030 | 9 | 12.35260129 |
| recursive_best_first_search h_1 | - | - | - | - | ***> 10 minutes*** |
| greedy_best_first_graph_search h_1 | 990 | 992 | 8,910 | 21 | 2.505408587 |
| astar_search h_1 | 4,852 | 4,854 | 44,030 | 9 | 12.46071228 |
| **astar_search h_ignore_preconditions** | **1,450** | **1,452** | **13,303** | **9** | **3.792846605** |
| astar_search h_pg_levelsum | 86 | 88 | 841 | 9 | 179.9119138 |

For problem 2, 7 of the 10 search algorithms were able to run to completion in under 10 minutes. The breadth first tree search, depth limited search, and the recursive best first search using h1 were taking much longer than 10 minutes, so the results are not included.

For the uninformed searches, comparing breadth-first search, depth-first graph search, and uniform cost search, both breadth-first and uniform cost search generated a solution plan of 9 actions, but depth-first graph search created a plan with a massive 619 actions. The breadth-first search resulted in 3,343 expansion nodes, 4,609 goal tests, and 30,509 new nodes. This is 1,509 fewer expansion nodes, 245 fewer goal tests, 13,521 fewer new nodes, and 3.66401 seconds faster than the than the 4,852 expansion nodes, 4,854 goal tests, 44,030 new nodes, and 12.3526 seconds elapsed to complete uniform cost search. As in problem 1, depth-first graph search has the fewest number of expansions, goal tests, number new nodes, and time elapsed, then the other two uninformed planning search methods, but the excessive length of the action plan excludes it from being useful.

Most of the automatic heuristic methods were able to run to completion, apart from recursive best first search. Greedy best first graph search was the only automatic search method to create a plan length longer than 9 but did have the fastest run time (2.5054 seconds), and the second fewest expansions (990), goal tests (992), and new nodes (8,910). The A* search ignoring preconditions had better result metrics than the A* search, but not better than the results of A* search planning graph level-sum. Even though A* search planning graph level-sum has fewer expansions, goal tests, and new nodes than the other automatic heuristics, the completion time was by far the worst. The completion time of 179.9119 seconds was about 47.43 times longer than the 3.7928 seconds to complete the A* search ignoring the preconditions. Because of this I would conclude that the A* search ignoring the preconditions provides the optimal solution. This is a nice conclusion since it again backs the notion of the advantage that informed algorithms have over the uninformed or blind search algorithms, such as BFS or DFS. That the informed methods use heuristics to guide the search toward the goal and significantly reduce the amount of states visited.

The optimal sequence of actions (taken from A* search with heuristic ignoring preconditions):

Load(C3, P3, ATL)

Fly(P3, ATL, SFO)

Unload(C3, P3, SFO)

Load(C2, P2, JFK)

Fly(P2, JFK, SFO)

Unload(C2, P2, SFO)

Load(C1, P1, SFO)

Fly(P1, SFO, JFK)

Unload(C1, P1, JFK)

- **Problem 3**

| **Problem 3** | | | | | |
|---|---|---|---|---|---|
| **Planning Search Method** | **Expansions** | **Goal Tests** | **New Nodes** | **Plan length** | **Time elapsed (seconds)** |
| breadth_first_search | 14,663 | 18,098 | 129,631 | 12 | 44.80745945 |
| breadth_first_tree_search | - | - | - | - | ***> 10 minutes*** |
| depth_first_graph_search | 408 | 409 | 3,364 | 392 | 1.784635794 |
| depth_limited_search | - | - | - | - | ***> 10 minutes*** |
| uniform_cost_search | 18,235 | 18,237 | 159,716 | 12 | 54.20389398 |
| recursive_best_first_search h_1 | - | - | - | - | ***> 10 minutes*** |
| greedy_best_first_graph_search h_1 | 5,614 | 5,616 | 49,429 | 22 | 16.89767683 |
| astar_search h_1 | 18,235 | 18,237 | 159,716 | 12 | 53.43924966 |
| **astar_search h_ignore_preconditions** | **5,040** | **5,042** | **44,944** | **12** | **15.33578351** |
| astar_search h_pg_levelsum | 325 | 327 | 3,002 | 12 | 1079.81897 |

As in problem 2, problem 3 was not able to run all search methods on the problem. The same 3 tests as in problem 2 were not able complete in under 10 minutes: breadth first tree search, depth limited search, and recursive best first search.

For the uninformed searches, breadth-first search, depth-first graph search, and uniform cost search are compared. Both breadth-first and uniform cost generated a useful result plan of length 12, and depth-first graph search created a plan with egregious length of 392. Because of this the depth-first graph search is not considered optimal. Breadth-first search created 14,663 expansions, 18,098 goal states, 129,631 new nodes in the 44.807459 seconds it took to generate a solution for the problem. This is better than the 18,235 expansions, 18,237 goal tests, and 159,716 new nodes generated by the uniform cost search in 54.20389 seconds. Here, as in problem 1 and in problem 2 the results of the uniform cost search are the same as those from the A* search. This seems odd since the uniform cost search has $f(n) = g(n)$, but A* should have $f(n) = g(n) + h(n)$, since both answers are the same I'm tempted to think that the estimated cost to get to the goal from n, $h(n)$, is zero and both uniform cost and A* are using $f(n) = g(n)$.

Comparing the automatic heuristic searches, the A* generated the most expansions (18,235), goal tests (18,237), and new nodes (159,716), when compared to greedy-best-first-graph-search, A* search ignoring preconditions, and A* planning-graph-level-sum. The greedy-best-first-graph-search resulted in 5,614 expansions, 5,616 goal tests, 49,429 new nodes, which is the third fewest, but the plan length of 22 is larger than the other 3 search methods who all generated a plan length of 12. The A* search using planning-graph-level-sum has the fewest number of expansions 325, the fewest goal tests 327, and the fewest number of new nodes 3,002 compared to any other search method, both uninformed or automatic. The problem is that the computation time was 1079.81897 seconds. The best solution, like in problem 2, comes from the A* ignoring the preconditions. The number of expansions was 5,040, there were 5,042 goal tests, and 44,944 new nodes, this was only second to the A* search with planning-graph-level-sum, but now the computation time was only 15.33578 seconds.

The optimal sequence of actions (taken from A* search with heuristic ignoring preconditions):

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(C2, P2, SFO)

Unload(C1, P1, JFK)