# Optimal Plans

## Problem 1

Load(C2, P2, JFK)

Load(C1, P1, SFO)

Fly(P2, JFK, SFO)

Unload(C2, P2, SFO)

Fly(P1, SFO, JFK)

Unload(C1, P1, JFK)

## Problem 2

Load(C2, P2, JFK)

Load(C1, P1, SFO)

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)

## Problem 3

Load(C2, P2, JFK)

Load(C1, P1, SFO)

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)

# Running Times of Algorithm/Problem Combinations

The following table contains cells for each problem/algorithm combination in terms of its optimality, running time, and the number of expansions.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Problem 1** | **Problem 2** | **Problem 3** |
| BFS | Optimal: yes  Expansions: 43  Runtime: 0.02 sec | Optimal: yes  Expansions: 3343  Runtime: 10.9 sec | Optimal: yes  Expansions: 14663  Runtime: 84.4 sec |
| BFS Tree Search | Optimal: yes  Expansions: 1458  Runtime: 0.76 sec | n/a  Runtime: > 10 min | n/a  Runtime: > 10 min |
| DFS Graph Search | Optimal: no  Expansions: 12  Runtime: 0.007 sec | Optimal: no  Expansions: 582  Runtime: 2.48 sec | Optimal: no  Expansions: 627  Runtime: 2.6 sec |
| Depth Limited Search | Optimal: no  Expansions: 101  Runtime: 0.07 | n/a  Runtime: > 10 min | n/a  Runtime: > 10 min |
| Uniform Cost Search | Optimal: yes  Expansions: 55  Runtime: 0.02 sec | Optimal: yes  Expansions: 4823  Runtime: 9.42 sec | Optimal: yes  Expansions: 18223  Runtime: 41.4 sec |
| Recursive Best First Search with h\_1 | Optimal: yes  Expansions: 4229  Runtime: 2.16 sec | n/a  Runtime: > 10 min | n/a  Runtime: > 10 min |
| Greedy Best First Graph Search with h\_1 | Optimal: yes  Expansions: 7  Runtime: 0.0045 sec | Optimal: no  Expansions: 385  Runtime: 0.75 sec | Optimal: no  Expansions: 5578  Runtime: 13.15 sec |
| A\* Search w/h\_1 | Optimal: yes  Expansions: 55  Runtime: 0.03 sec | Optimal: yes  Expansions: 4823  Runtime: 9.46 sec | Optimal: yes  Expansions: 18223  Runtime: 42.4 sec |
| A\* Search w/h\_ignore\_precon | Optimal: yes  Expansions: 41  Runtime: 0.03 | Optimal: yes  Expansions: 1421  Runtime: 3.3 sec | Optimal: yes  Expansions: 5040  Runtime: 13.75 sec |
| A\* Search w/h\_pg\_levelsum | Optimal: yes  Expansions: 11  Runtime: 0.44 sec | Optimal: yes  Expansions: 86  Runtime: 37.44 sec | Optimal: yes  Expansions: 316  Runtime: 184 sec |

# Analysis

DFS and depth limited search are very efficient in terms of runtime, but they have very non-optimal solutions. They come to conclusions quickly, but not very good solutions in any of the problems. In fact, depth limited search won’t even complete on problem 2 or 3 because it takes > 10 min to run! I found this interesting because it was so efficient in problem 1, but the state space expands so much in problem 2 and 3 that it simply takes too long to run. BFS, on the other hand, always came to the optimal solution. For problem 1 it was also fairly efficient, whereas for problems 2 and 3 its running time starting ramping up pretty dramatically such that by problem 3 it was an unrealistic way of solving the problem because of how long it took (1.5 min).

In terms of A\* heuristic searches, they all performed optimally on all three problems. An interesting dynamic played out for “ignore preconditions” and “levelsum”, however. The levelsum heuristic had optimal results and very low number of expansions, but the heuristic is more complex so it takes quite a long time to run. By contrast, “ignore preconditions” had faster ramp-up in the number of expansions (though still quite reasonable) and its runtime was very fast. For this problem, the “ignore preconditions” heuristic is probably the best option. If, however, memory is an issue or expansions are very expensive for a problem, levelsum is likely to be a very strong candidate.

Ignore preconditions performed the best of all the non-heuristic and heuristic approaches across all problem sets when considering optimality and runtime, but not necessarily from the number of expansions. It did, however, do better than all other optimal non-heuristics in terms of expansions. This definitely seems like the best option in these three problem spaces.