

# Optimal plans

For each problem example of optimal plan provided and length of the plan. Optimality defined as a total plan length. Each problem has multiple optimal plans.

## Problem 1

Length: 6

```
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

Length: 9

```
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

Length: 12, number of planes < number of cargos

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

# Summary of search methods and heuristics

Bold are approaches using heuristics

## Problem 1

Heuristic	Plan length	Expansions	Goal Tests	New Nodes	Time elapsed sec
1. breadth_first_search	6	43	56	180	0.018
2. breadth_first_tree_search	6	1458	1459	5960	0.494
3. depth_first_graph_search	12	12	13	48	0.0045
4. depth_limited_search	50	101	271	414	0.036
5. uniform_cost_search	6	55	57	224	0.0198
6. recursive_best_first_search with h_1	6	4229	4230	17029	1.5
7. greedy_best_first_graph_search with h_1	6	7	9	28	0.002
8. astar_search with h_1	6	55	57	224	0.02
9. <b>astar_search with h_ignore_preconditions</b>	6	41	43	170	0.016
10. <b>astar_search with h_pg_levelsum</b>	6	11	13	50	1.68

## Problem 2

Heuristic	Plan length	Expansions	Goal Tests	New Nodes	Time elapsed sec
1. breadth_first_search	9	3343	4609	30509	11.91
2. breadth_first_tree_search	timeout	timeout	timeout	timeout	timeout
3. depth_first_graph_search	575	582	583	5211	3.12
4. depth_limited_search	50	222719	2053741	2054119	468.66
5. uniform_cost_search	9	4852	4854	44030	43.58
6. recursive_best_first_search with h_1	timeout	timeout	timeout	timeout	timeout
7. greedy_best_first_graph_search with h_1	15	990	992	8910	6.82
8. astar_search with h_1	9	4852	4854	44030	42.94
9. <b>astar_search with h_ignore_preconditions</b>	9	1506	1508	13820	11.39
10. <b>astar_search with h_pg_levelsum</b>	9	86	88	841	180.26

## Problem 3

Heuristic	Plan length	Expansions	Goal Tests	New Nodes	Time elapsed sec
1. breadth_first_search	12	14120	17673	124926	92.02
2. breadth_first_tree_search	timeout	timeout	timeout	timeout	timeout
3. depth_first_graph_search	660	677	678	5608	3.48
4. depth_limited_search	timeout	timeout	timeout	timeout	timeout
5. uniform_cost_search	12	18223	18225	159618	409.68

Heuristic	Plan length	Expansions	Goal Tests	New Nodes	Time elapsed sec
6. recursive_best_first_search with h_1	timeout	timeout	timeout	timeout	timeout
7. greedy_best_first_graph_search with h_1	22	5578	5580	49150	107.64
8. astar_search with h_1	12	18223	18225	159618	418.14
9. astar_search with h_ignore_preconditions	12	5118	5120	45650	89.286
10. astar_search with h_pg_levelsum	12	414	416	3818	1219

## Results analysis

### Uninformed vs heuristics approaches

Approaches 1-8 here doesn't use heuristic (h1 is fake heuristic), so it have no additional information about which state is better, they can only generate next states and check them for goal/not goal. astar\_search with h\_1 is actually just a uniform\_cost\_search because  $h_1 = \text{const } 1$ .

### Uninformed approaches overview

DF Graph Search and Greedy BF Graph Search are fast, but don't find optimal solution. Nodes expansions count are smaller in this cases, because they never backtrack and stops on first found solution.

BF, BF Tree, Recursive Best First and Uniform Cost are optimal. But with high time and expansions numbers, as they both often go back on planning tree and expand nodes in breadth to find optimal solution. BF stops immediately after solution finding, Uniform continues with last depth level, which leads to higher timing. BF Tree search doesn't care properly about repetitive states, as a result it's out of normal time on the large problems. Same problem presented in Recursive Best First Search approach.

In family of uninformed searches without heuristics BF is the best option because it finds optimal plan, stops right after finding, expand less nodes than other approaches with lesser timings.

### Heuristic based approaches

Effect of adding heuristic could be analysed based on A\* modification. All A\* approaches in this analysis are optimal, all of them have optimistic admissible heuristics, which is a requirement for optimality. For A\* we have version without heuristic - h\_1, version with very simple to compute heuristic with relaxed problem / greedy approach - h\_ignore\_preconditions and sophisticated GraphSearch based heuristic, which is expensive to compute.

From result we can see that heuristic based approaches reduce number of expanded nodes significantly and it case of cheap heuristic could lead to better times. On the same time, using

sophisticated heuristic leads to significant reducing number of nodes expansions, but we get it in cost of total time, which is mostly going to heuristic calculation, not on graph traversing.

## What is the best?

For small problems it's better to use uninformed optimal search like simple BF search. For large problems it's better to use A\* search with easy to compute heuristic, which decreases number of nodes expansion and total time for search. A\* search with sophisticated heuristic could be better if we need to limit the number of expanded nodes for any reason... we have reinforcement learning or memory limitation in some way for example. All this methods leads to optimal plan.