Implementing a Planning Search

1. Optimal Plan

There are a few different versions of optimal plans for each problem. This is because each action is “instanatenous”. There is no concept of time taken to perform each task and each task is performed sequentially. In theory if you have two planes at the same starting point needing to load cargo and fly to the same place you could load them at the same time (or one at a time) and fly them at the same time to the same destination. Since our simple algorithm doesn’t work this way, plans like Load, Fly, Unload, Load, Fly, Unload, could be equivalent to Load, Load, Fly, Fly, Unload, Unload, or Load, Fly, Load, Fly, Unload, Unload, etc. While some searches had different results, they might have had “optimal results” meaning the smaller number of steps to accomplish the goal.

I will show the results of the top 4 searches. These all produced the smallest plan lengths in acceptable amount of time. Even though they might be different, they can all be considered optimal in terms of plan length.

```
python run_search.py -p 1 -s 1
Solving Air Cargo Problem 1 using breadth_first_search...
Expansions   Goal Tests   New Nodes
   43         56         180
Plan length: 6  Time elapsed in seconds: 0.027739397992263548
Load(C2, P2, JFK)
Load(C1, P1, SFO)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)
```

```
python run_search.py -p 1 -s 5
Plan length: 6
Solving Air Cargo Problem 1 using uniform_cost_search...
Expansions   Goal Tests   New Nodes
   55         57         224
Plan length: 6  Time elapsed in seconds: 0.03437601499899756
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P1, SFO, JFK)
```

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

python run_search.py -p 1 -s 8
Solving Air Cargo Problem 1 using astar_search with h_1...
Expansions   Goal Tests   New Nodes
   55         57         224
Plan length: 6  Time elapsed in seconds: 0.03279911399295088
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P1, SFO, JFK)
Fly(P2, JFK, SFO)
Unload(C1, P1, JFK)
Unload(C2, P2, SFO)

python run_search.py -p 1 -s 9
Solving Air Cargo Problem 1 using astar_search with h_ignore_preconditions...
Expansions   Goal Tests   New Nodes
   41         43         170
Plan length: 6  Time elapsed in seconds: 0.03736400400521234
Load(C1, P1, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)

python run_search.py -p 2 -s 1
Solving Air Cargo Problem 2 using breadth_first_search...
Expansions   Goal Tests   New Nodes
  3343        4609       30509
Plan length: 9  Time elapsed in seconds: 13.636947063001571
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)
```

```
python run_search.py -p 2 -s 5
Solving Air Cargo Problem 2 using uniform_cost_search...
Expansions   Goal Tests   New Nodes
   4853        4855        44041
Plan length: 9  Time elapsed in seconds: 11.210942698002327
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Load(C3, P3, ATL)
Fly(P1, SFO, JFK)
Fly(P2, JFK, SFO)
Fly(P3, ATL, SFO)
Unload(C3, P3, SFO)
Unload(C2, P2, SFO)
Unload(C1, P1, JFK)

python run_search.py -p 2 -s 8
Solving Air Cargo Problem 2 using astar_search with h_1...
Expansions   Goal Tests   New Nodes
   4853        4855        44041
Plan length: 9  Time elapsed in seconds: 11.454999515990494
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Load(C3, P3, ATL)
Fly(P1, SFO, JFK)
Fly(P2, JFK, SFO)
Fly(P3, ATL, SFO)
Unload(C3, P3, SFO)
Unload(C2, P2, SFO)
Unload(C1, P1, JFK)

python run_search.py -p 2 -s 9
Solving Air Cargo Problem 2 using astar_search with h_ignore_preconditions...
Expansions   Goal Tests   New Nodes
   1428        1430        13085
Plan length: 9  Time elapsed in seconds: 3.876375399995595
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)

python run_search.py -p 3 -s 1
Solving Air Cargo Problem 3 using breadth_first_search...
Expansions   Goal Tests   New Nodes
  14663       18098       129631
Plan length: 12  Time elapsed in seconds: 94.99353891800274
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)

python run_search.py -p 3 -s 5
Solving Air Cargo Problem 3 using uniform_cost_search...
Expansions   Goal Tests   New Nodes
  18233       18235       159697
Plan length: 12  Time elapsed in seconds: 48.442402720000246
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(P2, ORD, SFO)
Unload(C4, P2, SFO)
Fly(P1, ATL, JFK)
Unload(C3, P1, JFK)
Unload(C2, P2, SFO)
Unload(C1, P1, JFK)

python run_search.py -p 3 -s 8
Solving Air Cargo Problem 3 using astar_search with h_1...
Expansions   Goal Tests   New Nodes
  18233       18235       159697
Plan length: 12  Time elapsed in seconds: 49.29397620300006
```

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

python run_search.py -p 3 -s 9
Solving Air Cargo Problem 3 using astar_search with h_ignore_preconditions...
Expansions   Goal Tests   New Nodes
   4859   4861   43129
Plan length: 12 Time elapsed in seconds: 15.309105210006237
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)

2. Compare and contrast non-heuristic search result metrics
Uniform Cost Search - This was the best performing non-heuristic search algorithm in terms of plan length and time. It was slightly worse than Breadth First Search in terms of expansions and goal tests.

Breadth First Search - This was the best performing non-heuristic search algorithm in terms of plan length and expansions, goal tests, and new nodes. It is worse than Uniform Cost Search in terms of time taken. The time is pretty comparable with the simpler problems, but as you add literals, the time increases a lot. Problem 3 takes almost twice as long as Uniform Cost Search. If memory is an issue, this algorithm might be a good alternative to Uniform Cost Search, but with he current problems and my computer setup, this is definitely second.

Depth First Graph Search - While this doesn't produce an optimal path, it is extremely fast and has the smallest number of expansions, goal tests, and new nodes. If speed or memory are an issue, this is a good algorithm, however the plan isn't great. In the first example, you start out with a plan at JFK and SFO. The first thing that this algorithm does is fly the planes to the opposite airports for no reason. While this is extremely wasteful with regard to airplanes, the speed, and small memory footprint could be useful in a situation where all actions are instantaneous, and an optimal solution isn't necessary. It's speed could even be used to test if a solution is possible before passing it to a slower but more optimal algorithm.

3. Compare and contrast heuristic search result metrics using A* with the "ignore preconditions" and "level-sum" heuristics

A Star Search Ignore Preconditions - This was the best performing heuristic algorithm in terms of plan length, time taken, and was great in terms of expansions, goal tests, and new nodes. It was actually the best overall of all searches. Only Depth First Graph search was significantly better in terms of memory and time, but that produced a widely un-optimised plan.

A Star Search - This was the best performing search in terms of plan length, but wasn't as good as Ignore Preconditions when it comes to memory or execution time. Ignore Preconditions is simply better.

A Star Search Level Sum - This performed decently well for problems 1 and 2. It was able to find optimal plans, but the time increased drastically when parameters were added. Problem 2 took 519 seconds where Ignore Preconditions took 3.88 and Problem 3 never finished.

4. The best heuristic

The best heuristic overall was the Ignore Preconditions heuristic. This simply computes the number of states that still need to change and assumes that each takes one action to change them. It worked surprisingly well in this example. It was the hands down winner of all the search algorithms that produce optimal solutions. It also works very quickly and doesn't take up much memory. I would think that it wouldn't work as well for problems were single actions effect more parameters or if the preconditions get more complicated. This also worked much better than the other non heuristic search functions. This was even more apparent as the parameters increased. An increase in either added greatly to both the nodes and time. This isn't surprising. The Artificial Intelligence Third Edition by Russell and Norvig sings the praises of the algorithm. On page 98 they say "A* is optimally efficient for any given consistent heuristic. That is, no other optimal algorithm is guaranteed to expand fewer nodes than A*..." They go on to talk

about the importance of a good heuristic function. In this case, after trying a few different heuristics, we can see that ignore preconditions, combined with A* works great for this particular problem of this particular size.

Results Table

1 breadth_first_search

	Expansions	Goal Tests	New Nodes	Time	Plan Length	
Air Cargo Problem 1	43	56	180	0.027	**6**	Tied Best Score andThird Best Time
Air Cargo Problem 2	3343	4609	30509	13.637	**9**	
Air Cargo Problem 3	14663	18098	129631	94.99	**12**	

2 breadth_first_tree_search

	Expansions	Goal Tests	New Nodes	Time	Plan Length
Air Cargo Problem 1	1458	1459	5960	0.89	**6**
Air Cargo Problem 2	Timeout 30+ min				
Air Cargo Problem 3	Timeout 30+ min				

3 depth_first_graph_search

	Expansions	Goal Tests	New Nodes	Time	Plan Length
Air Cargo Problem 1	12	13	48	0.008	12
Air Cargo Problem 2	1669	1670	14863	12.32	1444
Air Cargo Problem 3	592	593	4927	2.70	571

4 depth_limited_search

	Expansions	Goal Tests	New Nodes	Time	Plan Length
Air Cargo Problem 1	101	271	414	0.09	50
Air Cargo Problem 2				Timeout 10+ min	
Air Cargo Problem 3				Timeout 30+ min	

5 uniform_cost_search

	Expansions	Goal Tests	New Nodes	Time	Plan Length
Air Cargo Problem 1	55	57	224	**0.036**	Tied Best Score and Tied Second Best Time
Air Cargo Problem 2	4853	4855	44041	**11.21**	**9**
Air Cargo Problem 3	18233	18235	159697	**48.44**	**12**

6 recursive_best_first_search h_1

	Expansions	Goal Tests	New Nodes	Time	Plan Length
Air Cargo Problem 1	4229	4230	17029	2.80	6
Air Cargo Problem 2				Timeout 10+ min	
Air Cargo Problem 3				Timeout 10+ min	

7 greedy_best_first_graph_search h_1

	Expansions	Goal Tests	New Nodes	Time	Plan Length
Air Cargo Problem 1	7	9	28	0.005	6
Air Cargo Problem 2	339	401	3617	0.963	25
Air Cargo Problem 3	4188	4190	37008	11.59	27

8 astar_search h_1

	Expansions	Goal Tests	New Nodes	Time	Plan Length	
Air Cargo Problem 1	55	57	224	**0.033**	**6**	Tied Best Score and Tied Second Best Time
Air Cargo Problem 2	4853	4855	44041	**11.45**	**9**	
Air Cargo Problem 3	18233	18235	159697	**49.29**	**12**	

9 astar_search h_ignore_preconditions

	Expansions	Goal Tests	New Nodes	Time	Plan Length	
Air Cargo Problem 1	41	43	170	**0.037**	**6**	Tied Best Score and Best Time Overall Winner
Air Cargo Problem 2	1428	1430	13085	**3.889**		
Air Cargo Problem 3	4859	4861	43129	**15.31**	**12**	

10 astar_search h_pg_levelsum

	Expansions	Goal Tests	New Nodes	Time	Plan Length
Air Cargo Problem 1	55	57	224	0.097	6
Air Cargo Problem 2	4853	4855	44041	519.85	9
Air Cargo Problem 3	Timeout 30+ min				