Vaibhav Kumar
Udacity - AIND

Project 2: Implement a Planning Search
Heuristic Analysis

Execution commands

Uninformed planning search
Three uninformed planning searches used were: `breadth_first_search`, `depth_first_graph_search` and `greedy_best_first_graph_search h_1`. The commands used to run this were:

```
>>> python run_search.py -p 1 -s 1 3 7 > output_part1_p1.txt
>>> python run_search.py -p 2 -s 1 3 7 > output_part1_p2.txt
>>> python run_search.py -p 3 -s 1 3 7 > output_part1_p3.txt
```

Heuristic search
Two heuristic searches used were: `astar_search h_ignore_preconditions` and `astar_search h_pg_levelsum`. The commands used to run this were:

```
>>> python run_search.py -p 1 -s 9 10 > output_part2_p1.txt
>>> python run_search.py -p 2 -s 9 10 > output_part2_p2.txt
>>> python run_search.py -p 3 -s 9 10 > output_part2_p3.txt
```

Analysis

A summary of the metrics for each search for each air cargo problem is presented below.

`air_cargo_p1`

Uninformed planning search

Search method	Node expansions	Goal tests	New nodes	Plan Length	Time elapsed (s)	Optimal?
Breadth First Search	43	56	180	6	0.0521	Yes
Depth First Graph Search	12	13	48	12	0.0108	No
Greedy Best First Graph Search h_1	7	9	28	6	0.0063	Yes

Here, we can see that the optimal plan (length 6) is found by Breadth First Search and Greedy Best First Graph Search h_1. Between these two, it can be seen that the latter is more efficient, with only 7 node expansions and a much quicker execution time.

Optimal plan, using Greedy Best First Graph Search h_1 (the more efficient approach):
```
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P1, SFO, JFK)
Fly(P2, JFK, SFO)
Unload(C1, P1, JFK)
Unload(C2, P2, SFO)
```

Heuristic search

Search method	Node expansions	Goal tests	New nodes	Plan Length	Time elapsed (s)	Optimal?
A* with ignore preconditions	41	43	170	6	0.0579	Yes
A* with level-sum	11	13	50	6	1.3378	Yes

Here, we can see both searches provide an optimal plan of length 6. The memory consumption (as inferred from node expansions) with level-sum is lower, though the execution time is faster with ignore-preconditions.

The optimal plan length is the same optimal plan length found with uninformed planning.

Optimal plan, using level-sum (fewer nodes expanded):
```
Load(C1, P1, SFO)
Fly(P1, SFO, JFK)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Unload(C1, P1, JFK)
Unload(C2, P2, SFO)
```

`air_cargo_p2`

Uninformed planning search

Search method	Node expansions	Goal tests	New nodes	Plan Length	Time elapsed (s)	Optimal?
Breadth First Search	3343	4609	30509	9	13.1350	Yes
Depth First Graph Search	582	583	5211	575	4.0104	No
Greedy Best First Graph Search h_1	998	1000	8982	21	3.2779	No

Here, we can see that the optimal plan (length 9) is found by Breadth First Search. It is worth noting, however, that the nodes expanded is significantly higher in this case compared to the other two methods, suggesting a higher memory consumption, and also yielding a longer execution time. Depth First Graph search appears to converge to a solution relatively quickly, however the plan length is far from optimal.

Optimal plan, using Breadth First Search:
```
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)
```

Heuristic search

Search method	**Node expansions**	**Goal tests**	**New nodes**	**Plan Length**	**Time elapsed (s)**	**Optimal?**
A* with ignore preconditions	1450	1452	13303	9	6.3032	Yes
A* with level-sum	86	88	841	9	230.7151	Yes

Here, we can see both searches provide an optimal plan of length 9. The memory consumption (as inferred from node expansions) with level-sum is lower, though the execution time is faster with ignore-preconditions.

The optimal plan length is the same optimal plan length found with uninformed planning.

Optimal plan, using level-sum (fewer nodes expanded):
```
Load(C1, P1, SFO)
Fly(P1, SFO, JFK)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Load(C3, P3, ATL)
Fly(P3, ATL, SFO)
Unload(C3, P3, SFO)
Unload(C2, P2, SFO)
Unload(C1, P1, JFK)
```

`air_cargo_p3`

Uninformed planning search

Search method	Node expansions	Goal tests	New nodes	Plan Length	Time elapsed (s)	Optimal?
Breadth First Search	14663	18098	129631	12	58.6177	Yes
Depth First Graph Search	627	628	5176	596	4.1801	No
Greedy Best First Graph Search h_1	5578	5580	49150	22	23.3501	No

Here, we can see that the optimal plan (length 12) is found by Breadth First Search. Once again, it is noted that the nodes expanded, and therefore memory consumption is much higher, as well as a slower execution time. Depth First Graph search converges to a solution quickest, however the plan length is very poor compared to the other two.

Optimal plan, using Breadth First Search:

```
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)
```

Heuristic search

Search method	Node expansions	Goal tests	New nodes	Plan Length	Time elapsed (s)	Optimal?
A* with ignore preconditions	5040	5042	44944	12	24.1411	Yes
A* with level-sum	325	327	3002	12	1199.1441 (> 10 mins)	Yes

Here, we can see both searches provide an optimal plan of length 12. The memory consumption (as inferred from node expansions) with level-sum is lower, though the execution time is faster with ignore-preconditions.

The optimal plan length is the same optimal plan length found with uninformed planning.

Optimal plan, using level-sum (fewer nodes expanded):

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

Conclusion

In all three air cargo problems, both the uninformed and heuristic set searches converged to a solution. Optimal paths were found in both sets of searches, with the difference lying in the time and efficiency of the method for a given problem.

Within the uninformed search group, the Breadth First Search always successfully found an optimal plan. The number of node expansions, and time taken, is generally found to be greater than the heuristic searches that also find an optimal plan. This makes sense, as the Breadth First Search is searching across the entire breadth of the search tree, thus also ending up finding the shortest path [1].

Both the heuristic methods successfully found optimal plans, although it was found that the A* with level-sum took a significantly longer time than with ignore-preconditions, while expanding fewer nodes (and hence lesser memory consumption).

The difference between the performance metrics of Breadth First Search and the heuristic approaches is quite distinct in problems 2 and 3, most likely due to the greater complexity of these problems where the heuristic approaches have an advantage.

The 'best' method depends on the constraints of computation and time. Breadth First Search will almost certainly find the optimal plan, if some efficiency can be forgone. However, for situations where time is of higher importance, A* search with ignore-preconditions tends to do a generally better job as well as finding the optimal plan. In cases where computational resource usage is constrained, A* search with level-sum would be preferred to get the optimal plan with less memory consumption. A reasonable balance between all these factors, though only slightly sub-optimal, is provided by the Greedy Best First Graph Search h_1.

References

[1] Russel, S. & Norvig, P (2010), *Artificial Intelligence: A Modern Approach, 3e.*