Project 3: Planning Search

Uninformed Searches

Problem 1

The optimal path length of the solution is 6 steps as discovered by breadth first and uniform cost searches. Uniform cost however does seem to take longer and goes through more nodes to find the optimal solution.

Depth search is extremely fast but does not find the most optimal solution. In the world of air cargo, that may be very bad as distances between airports are not small. As an example, the optimal solution discovered by both breadth first and uniform cost say to load Cargo1 on Plane1 and load Cargo2 on Plane2 and fly each plane to their appropriate location once (SFO->JFK for P1 and JFK->SFO for P2); however, depth first search has multiple flies between SFO and JFK for the same plane which is very expensive (time, fuel, etc.).

Optimal sequence:
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P1, SFO, JFK)
Fly(P2, JFK, SFO)
Unload(C1, P1, JFK)
Unload(C2, P2, SFO)

Search	Expansions	Goal Tests	New Nodes	Time (s)	Path Length
Breadth First	43	56	180	.02	6
Depth First	21	22	84	.01	20
Uniform Cost	55	57	224	.03	6

Problem 2

Problem 2 is consistent with results from Problem 1. The optimal length is 9 as discovered by Breadth First and Uniform Cost (more expensive) and depth first is extremely fast but takes 8.5x as many steps.

Optimal Sequence:
Load(C2, P2, JFK)
Fly(P2, JFK, ATL)
Load(C3, P2, ATL)
Fly(P2, ATL, SFO)
Unload(C3, P2, SFO)
Unload(C2, P2, SFO)
Load(C1, P1, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)

Search	**Expansions**	**Goal Tests**	**New Nodes**	**Time (s)**	**Path Length**
Breadth First	3063	4274	25442	4.86	9
Depth First	82	83	511	.11	77
Uniform Cost	4444	4446	36366	6.90	9

Problem 3

Problem 3 is consistent with results from Problem 1 and Problem 2. The optimal length is 12 as discovered by Breadth First and Uniform Cost (more expensive) and depth first is extremely fast but takes 32x as many steps.

Optimal Sequence:
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)

Search	Expansions	Goal Tests	New Nodes	Time (s)	Path Length
Breadth First	14663	18098	129631	29.45	12
Depth First	408	409	3364	1.26	392
Uniform Cost	18223	18225	159618	35.67	12

Heuristic Searches

Problem 1

Each of the heuristic searches finds the optimal solution of 6 steps; however, h_pg_levelsum takes significantly longer (15x). Even though h_pg_levelsum is clearly a good heuristic as the algorithm is taking far fewer steps to get to the goal, the computation cost at each step is too high. h_1 is nearly the same as uninformed Uniform Cost search and h_ignore_preconditions but h_ignore_preconditions goes through fewer nodes to reach the same solution in the same time.

Search	Expansions	Goal Tests	New Nodes	Time (s)	Path Length
A* w/ h_1	55	57	224	.03	6
A* w/ h_ignore_preconditions	41	43	170	.03	6
A* w/ h_pg_levelsum	11	13	50	.46	6

Problem 2

Each of the heuristic searches finds the optimal solution of 9 steps; however, h_pg_levelsum takes significantly longer (11-26x). Even though h_pg_levelsum is clearly a good heuristic as the algorithm is taking far fewer steps to get to the goal, the computation cost at each step is too high. h_1 is nearly the same as uninformed Uniform Cost search. Contrary to Problem 1, h_ignore_preconditions is both faster and goes through fewer nodes to reach the same solution.

Search	Expansions	Goal Tests	New Nodes	Time (s)	Path Length
A* w/ h_1	4444	4446	36366	6.84	9
A* w/ h_ignore_preconditions	1526	1528	12725	2.94	9
A* w/ h_pg_levelsum	293	295	2407	76.76	9

Problem 3

Each of the heuristic searches finds the optimal solution of 12 steps; however, h_pg_levelsum takes significantly longer (6-17x). Even though h_pg_levelsum is clearly a good heuristic as the algorithm is taking far fewer steps to get to the goal, the computation cost at each step is too high. h_1 is nearly the same as uninformed Uniform Cost search. Contrary to Problem 1, h_ignore_preconditions is both faster and goes through fewer nodes to reach the same solution.

Search	Expansions	Goal Tests	New Nodes	Time (s)	Path Length
A* w/ h_1	18223	18225	159618	36.04	12
A* w/ h_ignore_preconditions	5040	5042	44944	11.77	12
A* w/ h_pg_levelsum	331	333	3054	203.98	12

Conclusion

Out of all the searches that return the optimal solution, A* w/ h_ignore_preconditions performed the best (exception was Problem 1 where the it took split seconds longer that is likely just a measuring precision issue and Problem 1 is too small to be a real world problem). Hence, heuristic based searches are the way to go.

I believe h_ignore_preconditions does so well because it is not only a good heuristic on its own but its computation cost is very low. This is especially compared to h_pg_levelsum where the algorithm clearly explores the graph in a better path to the solution but the computation cost of the heuristic is very large. As the course has continuously suggested however, it is better to start with a simple solution and apply intelligence only as needed. Hence, if the only advantage of using h_pg_levelsum is to use less memory but modern hardware infrastructures invalidate that advantage, then the better solution is h_ignore_preconditions as it's easier to implement and low on computation cost.

I did notice however the slowness factor when comparing time between h_pg_levelsum vs. h_ignore_preconditions got lower as the problem got larger. It's hard to say whether that is a true trend because there are only 3 problems, but if it does hold true than h_pg_levelsum would be a much better heuristic as real world problems are likely to be large and an optimal exploration path would be better for memory costs.