Heuristic Analysis

Test Result

```
➜  AIND-Planning git:(master) ✗ python -m unittest tests.test_my_air_cargo_problems
..........
----------------------------------------------------------------------
Ran 10 tests in 0.082s

OK
➜  AIND-Planning git:(master) ✗ python -m unittest tests.test_my_planning_graph
.........
----------------------------------------------------------------------
Ran 9 tests in 0.015s

OK
```

- [x] Provide an optimal plan for Problems 1, 2, and 3.
- [x] Compare and contrast non-heuristic search result metrics (optimality, time elapsed, number of node expansions) for Problems 1,2, and 3. Include breadth-first, depth-first, and at least one other uninformed non-heuristic search in your comparison; Your third choice of non-heuristic search may be skipped for Problem 3 if it takes longer than 10 minutes to run, but a note in this case should be included.
- [x] Compare and contrast heuristic search result metrics using A* with the "ignore preconditions" and "level-sum" heuristics for Problems 1, 2, and 3.
- [ ] What was the best heuristic used in these problems? Was it better than non-heuristic search planning methods for all problems? Why or why not?
- [x] Provide tables or other visual aids as needed for clarity in your discussion.

Problem 1

**Optimal Plan: **

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

Metric	Expansions	Goal Tests	New Nodes	Plan Length	Time (sec)
Breadth First Search	43	56	180	6	0.03402195603121072
Breadth First Tree Search	1458	1459	5960	6	0.8423381419852376
Depth First Graph Search	21	22	84	20	0.013952289009466767
Depth Limited Search	101	271	414	50	0.09273918205872178
Uniform Cost Search	55	57	224	6	0.04404362407512963
A* Search with h_1	55	57	224	6	0.035493068979121745
A* with h_ignored_precond	41	43	170	6	0.026572089991532266
A* with h_pg_levelsum	11	13	50	6	1.1508120800135657

Problem 2

**Optimal Plan: **

```
Load(C1, P1, SFO)
Load(C2, P2, JFK)
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)
```

Metric	Expansions	Goal Tests	New Nodes	Plan Length	Time (sec)
Breadth First Search	3343	4609	30509	9	14.65209990600124
Depth First Graph Search	624	625	5602	619	3.64519296400249
Uniform Cost Search	4853	4855	44041	9	12.65471209899988
A* Search with h_1	4853	4855	44041	9	15.691585208987817
A* with h_ignored_precond	1450	1452	13303	9	5.296085149049759
A* with h_pg_levelsum	86	88	841	9	214.66166353900917

Problem 3

**Optimal Plan: **

```
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)
```

Metric	Expansions	Goal Tests	New Nodes	Plan Length	Time
Breadth First Search	14120	17673	124926	12	152.7389162160689
Depth First Graph Search	292	293	2388	288	2.048991840914823
Uniform Cost Search	18234	18236	159707	12	71.5715502500534
A* Search with h_1	18234	18236	159707	12	54.409601973951794
A* with h_ignored_precond	5040	5042	44944	12	21.302646758034825
A* with h_pg_levelsum	325	327	3002	12	1188.6693677880103

Non-heuristic search

Among the 3 problems, the depth-like search algorithm seems to perform worst overall algorithms, they all return a (much) longer plan path than other non-heuristic search algorithm. **BFS** has the best results for all 3 problems, it outperforms other non-heuristic search algorithms by expansion, number of goal tests and number of new nodes. **BFS** also provides optimal solutions for all 3 problems as well.

Heuristic search

The A* with h_1 is basically the uniform cost search to some extent so I'll ignore it (and it performs badly).

Among the 3 problems, **A* with h_pg_levelsum** outperforms all other algorithms (including both the heuristics and the non-heuristics) by expansion, number of goal tests and number of new nodes. Though in problem 3 it has much more new nodes and expansions than DFGS, but it has a way shorter plan length which is somehow the most important factor we should consider. **A* with h_pg_levelsum** also provides optimal solutions for all 3 problems as well.

General Analysis

We can see that, overall the **A* with h_pg_levelsum** can give us optimal results among all 3 problems. Though the DFS has smallest (or second smallest) amount of expansions, number of test goals and number of new nodes, but depth-like algorithms always gives us a very terrible plan length, which is casused by its

searching technique, intuitively. However, **A* with h_pg_levelsum** has a much longer computation time cost than any other algorithms, it seems caused by the heuristic functions. When the state space becomes larger, the computational complexity of **A* with h_pg_levelsum** grows way faster than others, say in problems 3, the state space should be 2 to the power of 32, then the computation time for **A* with h_pg_levelsum** becomes 1188 seconds, which is almost one third hour.

It's hard to say which searching algorithm is best among all category. But if we ignore the time cost of computing the optimal plan, **A* with h_pg_levelsum** will be the best. If we pay more attention to the computing time needed to find the optimal, **BFS** and **A* with h_ignored_precond** algorithm can also provide optimal results and run faster than **A* with h_pg_levelsum**.