PART 3: WRITTEN ANALYSIS

1.Provide an optimal plan for Problems 1, 2, and 3.

Problem	Length	Path
1	6	Load(C1, P1, SFO) Load(C2, P2, JFK) Fly(P2, JFK, SFO) Unload(C2, P2, SFO) Fly(P1, SFO, JFK) Unload(C1, P1, JFK)
2	9	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)
3	12	Load(C1, P1, SFO) Load(C2, P2, JFK) 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)

Compare and contrast non-heuristic search result metrics (optimality, time elapsed, number of node expansions) for Problems 1,2, and 3.

Compare and contrast heuristic search result metrics using A with the "ignore preconditions" and "level-sum" heuristics for Problems 1, 2, and 3.*

Problem1			
	Plan length - optimality	Time elapsed in seconds	Number of node expansions
1.breadth_first_search	6 - Yes	0.04380386401317082	43
2.breadth_first_tree_search	6 - Yes	1.201570597011596	1458
3. depth_first_graph_search	20 - No	0.015084580983966589	21
4. depth_limited_search	50 - No	0.12421708201873116	101
5. uniform_cost_search	6 - Yes	0.054987955983961	55
6. recursive_best_first_search h_1	6 - Yes	3.2182372009847313	4229
7. greedy_best_first_graph_search h_1	6 - Yes	0.007885082013672218	7
8. astar_search h_1	6 - Yes	0.049905341991689056	55
9. astar_search h_ignore_preconditions	6 - Yes	0.03924592401017435	41
10. astar_search h_pg_levelsum	6 - Yes	0.7135640219785273	11

Problem2			
	Plan length - optimality	Time elapsed in seconds	Number of node expansions
1.breadth_first_search	9 - Yes	19.786569677991793	3343
2.breadth_first_tree_search	/	/	/
3. depth_first_graph_search	619 - No	5.640385922975838	624
4. depth_limited_search	/	/	/
5. uniform_cost_search	9 – Yes	16.408532400004333	4853
6. recursive_best_first_search h_1	/	/	/
7. greedy_best_first_graph_search h_1	17 – No	3.427020678005647	998
8. astar_search h_1	9 – Yes	17.40108319101273	4853
9. astar_search h_ignore_preconditions	9 – Yes	5.95803467699443	1450
10. astar_search h_pg_levelsum	9 – Yes	67.97764711399213	86

Problem3			
	Plan length - optimality	Time elapsed in seconds	Number of node expansions
1.breadth_first_search	12 - Yes	152.91148800798692	14663
2.breadth_first_tree_search	/	/	/
3. depth_first_graph_search	392 – No	2.9167690370231867	408
4. depth_limited_search	/	/	/
5. uniform_cost_search	12 - Yes	76.94479637299082	18164
6. recursive_best_first_search h_1	/	/	/
7. greedy_best_first_graph_search h_1	26 - No	23.732159229984973	5398
8. astar_search h_1	12 - Yes	80.61351297498913	18164
9. astar_search h_ignore_preconditions	12 - Yes	23.602976485010004	5038
10. astar_search h_pg_levelsum	12 - Yes	391.4863481989887	314

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?

- For Problem 1, non-heuristic search planning methods performs better than all heuristic methods in terms of speed. The fastest heuristic method is slower than the slowest of the three non-heuristic methods tested. Therefore, all search stratagems are optimal for Problem 1.

- For Problem 2, only two non-heuristic methods are optimal, the Breath First Search and Uniform Cost Search. Breath First Search is faster than the best heuristic method, A* Search with “ignore preconditions” was better than other two A* Search stratagems.

- For Problem 3, the best optimal search search planning method is the heuristic A* Search with “ignore preconditions”. The fastest method is the non-heuristic Depth First Graph Search, however, it is not optimal.

- In summary, heuristic search strategies are better than uniformed non-heuristic strategies with the search spaces increasing, if only one method must be chosen, I will select A* Search with “ignore preconditions” as the best one choice. It is the best search planning method for problems 2 and 3 based on the collection data. While it is not the top performing method for problem 1, it is optimal for that problem and is faster than most other methods. Because the good heuristic makes forward and backward search efficient and can be used with A* search to find optimal solutions.