Written Analysis

Optimal Plans

All taken from BFS

A. **Problem 1:**
    a. Load(C1, P1, SFO)
    b. Load(C2, P2, JFK)
    c. Fly(P2, JFK, SFO)
    d. Unload(C2, P2, SFO)
    e. Fly(P1, SFO, JFK)
    f. Unload(C1, P1, JFK)
B. **Problem 2:**
    a. Load(C1, P1, SFO)
    b. Load(C2, P2, JFK)
    c. Load(C3, P3, ATL)
    d. Fly(P2, JFK, SFO)
    e. Unload(C2, P2, SFO)
    f. Fly(P1, SFO, JFK)
    g. Unload(C1, P1, JFK)
    h. Fly(P3, ATL, SFO)
    i. Unload(C3, P3, SFO)
C. **Problem 3:**
    a. Load(C1, P1, SFO)
    b. Load(C2, P2, JFK)
    c. Fly(P2, JFK, ORD)
    d. Load(C4, P2, ORD)
    e. Fly(P1, SFO, ATL)
    f. Load(C3, P1, ATL)
    g. Fly(P1, ATL, JFK)
    h. Unload(C1, P1, JFK)
    i. Unload(C3, P1, JFK)
    j. Fly(P2, ORD, SFO)
    k. Unload(C2, P2, SFO)
    l. Unload(C4, P2, SFO)

Results

Non-heuristic searches:

- **Problem 1**

	Breadth First Search	Breadth First Tree Search	Depth First Graph Search	Depth Limited Search	Uniform Cost Search
# Node Expansions	43	1458	21	101	55
# Goal Tests	56	1459	22	271	57
Time Elapsed	0.04662697599	1.526307323	0.021734079	0.13778759	0.06098217098
Plan Length	6	6	20	50	6

- **Problem 2**

	Breadth First Search	Breadth First Tree Search	Depth First Graph Search	Depth Limited Search	Uniform Cost Search
# Node Expansions	3343	--	386	222719	4853
# Goal Tests	4609	--	387	2053741	4855
Time Elapsed	21.26071139	--	2.281415977	2871.173857	61.55207567
Plan Length	9	--	380	50	9

- **Problem 3**

	Breadth First Search	Breadth First Tree Search	Depth First Graph Search	Depth Limited Search	Uniform Cost Search
# Node Expansions	14663	--	408	--	18223
# Goal Tests	18098	--	409	--	18225
Time Elapsed	158.0056894	--	2.875226442	--	880.7176038
Plan Length	12	--	392	--	12

Compare and contrast non-heuristic search result metrics

Solutions using Breadth First Tree Search and Depth Limited Search are not going to be used for this comparison, given that some of them were not able to find in less than 20 minutes, for Problems #2 & #3. This means that only Breadth First Search (BFS), Depth First Graph Search (DFS) and Uniform Cost Search (UCS) will be used.

According to Problem 1 results, non-heuristic search ran faster when using DFS. For this algorithm the time elapsed was about half than the one got using BFS, and about a third of what took UCS. The number of node expansions for DFS was also lower than the number got for the other algorithm. Nevertheless, plan length achieved with DFS was much longer than the plan length of the other algorithms. This allows to understand how DFS doesn't necessarily achieve the optimal value, even though it could find a path faster. On the other hand, BFS and UCS obtained a plan length value of 6, being the optimal one.

Results obtained are very similar for the other two plans, where DFS obtains the lowest value for # of node expansions, # of goal tests and time elapsed. BFS always scores the second, and UCS gets the third position. As mentioned previously, DFS isn't an optimal algorithm and its plan length is higher than the length for the other algorithms. It's achieved lengths are disproportionately bigger than the ones achieved with other algorithms (compare DFS's length of 392 vs UCS's 12 in problem 3). Conversely, UCS and BFS always score the same plan length.

Knowing this, DFS could be used as an algorithm for fast execution, in order to check if a route exists between two nodes, though it wouldn't return the shortest one. For this, BFS does a much better job than both UCS and DFS.

Heuristic searches:

- **Problem 1**

	Recursive Best First Search	Greedy Best First Search	A* Search	A* Search Ignore Preconditions	A* Search h_pg_levelsum
# Node Expansions	4229	7	55	55	41
# Goal Tests	4230	9	57	57	43
Time Elapsed	4.35715985	0.00954038300	0.05840180197	0.07556574501	4.851714392
Plan Length	6	6	6	6	6

- **Problem 2**

	Recursive Best First Search	Greedy Best First Search	A* Search	A* Search Ignore Preconditions	A* Search h_pg_levelsum
# Node Expansions	---	998	4853	4853	1245
# Goal Tests	---	1000	4855	4855	1247
Time Elapsed	---	10.69081739	57.89417691	98.22210387	2473.2197602
Plan Length	---	21	9	9	9

- **Problem 3**

	Recursive Best First Search	Greedy Best First Search	A* Search	A* Search Ignore Preconditions	A* Search h_pg_levelsum
# Node Expansions	---	6943	18223	18223	2934
# Goal Tests	---	6945	18225	18225	2936
Time Elapsed	---	180.0797159	860.6675854	1118.181544	8868.1748247

Plan Length	---	22	12	12	12

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

Both A* with constant heuristic (ASH1) and A* with ignore preconditions heuristic (ASIP) obtained the same values of # Node Expansions and # Goal Tests for all three problem executions. Conversely, algorithm ASH1 scored a fewer elapsed time than ASIP, for all three problems, making it an overall better choice than ASIP.

On the other hand, A* with Levelsum Heuristic (ASLH) obtained lower values than the other two algorithms for the same features. ASLH required fewer expansions and goal tests than other algorithms; ASLH obtained in Problem #1 & 2 about ¼ fewer of these, while for Problem # 3 it required about 84% less. Nevertheless, ASLH scored the highest time elapsed, taking up to 148 minutes to find Problem 3 solution.

ASLH takes longer to execute due the computation of its heuristic function (which requires generating graphplan's graph). Although ASLH obtains the optimum results, just like ASH1 and ASIP, it takes much longer than the other algorithms and is not as efficient in computation time. As a result, the best heuristic algorithm to use is ASH1.

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?

Heuristic 1 (constant heuristic associated to ASH1) is the best heuristic used with A*, given that it is capable of finding the optimal results in less time than the rest of the heuristic algorithms.

ASH1 is essentially the same algorithm as UCS, with the difference that it implements a constant heuristic function *g*,

```
def h_1(self, node: Node):
    # note that this is not a true heuristic
    h_const = 1
    return h_const
```

As mentioned earlier, BFS performs better than ASH1/UCS, in terms of elapsed time, # Node Expansions and # Goal Tests, and is optimal. This is why BFS is the best algorithm amongst all.