Analysis of Planning Search

Problem 1:

Methods	Expansions	Goal Tests	New Nodes	Plan Length	Time
Breadth First Search	43	56	180	6	0.045
Breadth First Tree Search	1458	1459	5960	6	1.453
Depth First Graph Search	21	22	84	20	0.023
Depth Limited Search	101	271	414	50	0.133
Uniform Cost Search	55	57	224	6	0.054
Recursive Best First Search	4229	4230	17023	6	4.078
Greedy Best First Graph Search	7	9	28	6	0.008
A* search with heuristics:					
h_1	55	57	224	6	0.055
Ignore Preconditions	41	43	170	6	0.057
Levelsum	11	13	50	6	1.403

The 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)

Among all the search algorithms, The **Greedy Best First Graph Search** seems the fastest and least search spaces. The Depth First Graph Search has small search space too but it doesn't give the optimal result. The Breadth First Search has a bit more search spaces but result in the optimal plan.

For the A* search with heuristics, all heuristics gives the optimal result. The Levelsum heuristics seems the most accurate, it has small search spaces and optimal result, but the calculating time is too long.

Problem 2:

Methods	Expansions	Goal Tests	New Nodes	Plan Length	Time
Breadth First Search	3343	4609	30509	9	13.14
Breadth First Tree Search	-	-	-	-	-
Depth First Graph Search	624	625	5602	619	5.480
Depth Limited Search	-	-	-	-	-
Uniform Cost Search	4852	4854	44030	9	19.31
Recursive Best First Search	-	-	-	-	-
Greedy Best First Graph Search	990	992	8910	21	3.762

A* search with heuristics:

h_1	4852	4854	44030	9	20.35
Ignore Preconditions	1450	1452	13303	9	6.528
Levelsum	86	88	841	9	355.1

The 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)

For problem2, Breadth First Tree Search, Depth Limited Search, Recursive Best First Search just go beyond the time. These three method doesn't provide good result in previous question either. In this time, **Breadth First Search** gives us the best optimal result. Other methods like Greedy Best First Graph Search, although it is quick and low search spaces, doesn't reach the optimal result.

For the A* search with heuristics, the situation looks the same as before. All heuristics provide optimal results. H_1 and Ignore-Preconditions heuristics seems fast with high search spaces, and Levelsum is slow with small search spaces.

Problem 3:

Methods	Expansions	Goal Tests	New Nodes	Plan Length	Time
Breadth First Search	14120	17673	124926	12	76.50
Breadth First Tree Search	-	-	-	-	-
Depth First Graph Search	1086	1087	9027	1055	22.27
Depth Limited Search	-	-	-	-	-
Uniform Cost Search	18223	18225	159618	12	107.4
Recursive Best First Search	-	-	-	-	-
Greedy Best First Graph Search	5578	5580	49150	14	31.94

A* search with heuristics:

h_1	18223	18225	159618	12	112.9
Ignore Preconditions	5040	5042	44944	12	39.50
Levelsum	325	327	3002	12	1632

The Plan:

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

For problem3, **Breadth First Search** gives result in moderate time. Greedy algorithm and Depth First Graph Search is fast but doesn't give the optimal solution.

For the A* search with heuristics, the result is just same as before, Ignore_preconditons heuristic seems the most fast and reliable one.

Conculsion:

Algorithm like **Breadth First Search** always give the optimal result in a moderate time.

Algorithms use Depth First Graph Search and greedy methods runs and use small spaces but doesn't always reach optimal.

A* algorithm always gives the optimal solutions. Different heuristics cost different search spaces and search time. Ignore_preconditons cost larger search spaces but small running time with the Levelsum heuristics cost much more time but small search spaces.

small justification:

BFS based algorithm search the sample spaces layer by layer, so it can always reach the optimal result by scanning all possible sample result. So, all BFS-based algorithm takes much time and expansion spaces than the others, but it always returns the optimal result.

DFS based algorithm goes straight into the sample depth. The performance of it depends on the questions. In the worst case, assume we traverse the search tree from left to right, the solution may at the very right of the search tree and we almost need to search all the sample spaces. But for average cases, it gives us a result immediately with small expansions although it is not optimal.

Greedy algorithm also use some heuristics, so it can be fast and cost small spaces, but it does not promise the optimal solution.

A* algorithm is a combination of greedy search and uniform cost search. It is a Best-First-Search. The expansion of nodes has order which correspond to the heuristics. The optimistic h finds the lowest-cost path.