HEURISTIC ANALYSIS

Problem 1 initial state and goal: [Cargo - 2, Airports - 2, Planes - 2]

```
Init(At(C1, SFO) ∧ At(C2, JFK)
        ∧ At(P1, SFO) ∧ At(P2, JFK)
        ∧ Cargo(C1) ∧ Cargo(C2)
        ∧ Plane(P1) ∧ Plane(P2)
        ∧ Airport(JFK) ∧ Airport(SFO))
Goal(At(C1, JFK) ∧ At(C2, SFO))
```

Search Results:

Algorithm	Nodes Expanded	Goal Test	New Nodes	Plan length	Time in seconds
breadth_first_search	43	56	180	6	0.749
breadth_first_tree_search	1458	1459	5960	6	2.289
depth_first_graph_search	12	13	48	12	0.221
depth_limited_search	101	271	414	50	0.2209
uniform_cost_search	55	57	224	6	0.1055
recursive_best_first_search h_1	4229	4230	17029	6	6.437
greedy_best_first_graph_search h_1	7	9	28	6	0.119
astar_search h_1	55	57	224	6	0.1051
astar_search h_ignore_preconditions	41	43	170	6	0.1014
astar_search h_pg_levelsum	11	13	50	6	2.001

Problem 2 initial state and goal: [Cargo - 3, Airports - 3, Planes - 3]

```
Init(At(C1, SFO) ∧ At(C2, JFK) ∧ At(C3, ATL)
        ∧ At(P1, SFO) ∧ At(P2, JFK) ∧ At(P3, ATL)
        ∧ Cargo(C1) ∧ Cargo(C2) ∧ Cargo(C3)
        ∧ Plane(P1) ∧ Plane(P2) ∧ Plane(P3)
        ∧ Airport(JFK) ∧ Airport(SFO) ∧ Airport(ATL))
Goal(At(C1, JFK) ∧ At(C2, SFO) ∧ At(C3, SFO))
```

Search Results:

Algorithm	Nodes Expanded	Goal Test	New Nodes	Plan length	Time in seconds
breadth_first_search	3343	4609	30509	9	21.854
breadth_first_tree_search	-	-	-	-	timeout
depth_first_graph_search	582	583	5211	575	7.456
depth_limited_search	-	-	-	-	timeout
uniform_cost_search	4853	4855	44041	9	15.3959
recursive_best_first_search h_1	-	-	-	-	timeout
greedy_best_first_graph_search h_1	998	1000	8982	13	6.0061
astar_search h_1	4853	4855	44041	9	29.1073

astar_search h_ignore_preconditions	1450	1452	13303	9	18.6803
astar_search h_pg_levelsum	86	88	841	9	181.9675

Problem 3 initial state and goal: [Cargo - 4, Airports - 4, Planes - 2]

```
Init(At(C1, SFO) ∧ At(C2, JFK) ∧ At(C3, ATL) ∧ At(C4, ORD)
        ∧ At(P1, SFO) ∧ At(P2, JFK)
        ∧ Cargo(C1) ∧ Cargo(C2) ∧ Cargo(C3) ∧ Cargo(C4)
        ∧ Plane(P1) ∧ Plane(P2)
        ∧ Airport(JFK) ∧ Airport(SFO) ∧ Airport(ATL) ∧ Airport(ORD))
Goal(At(C1, JFK) ∧ At(C3, JFK) ∧ At(C2, SFO) ∧ At(C4, SFO))
```

Search Results:

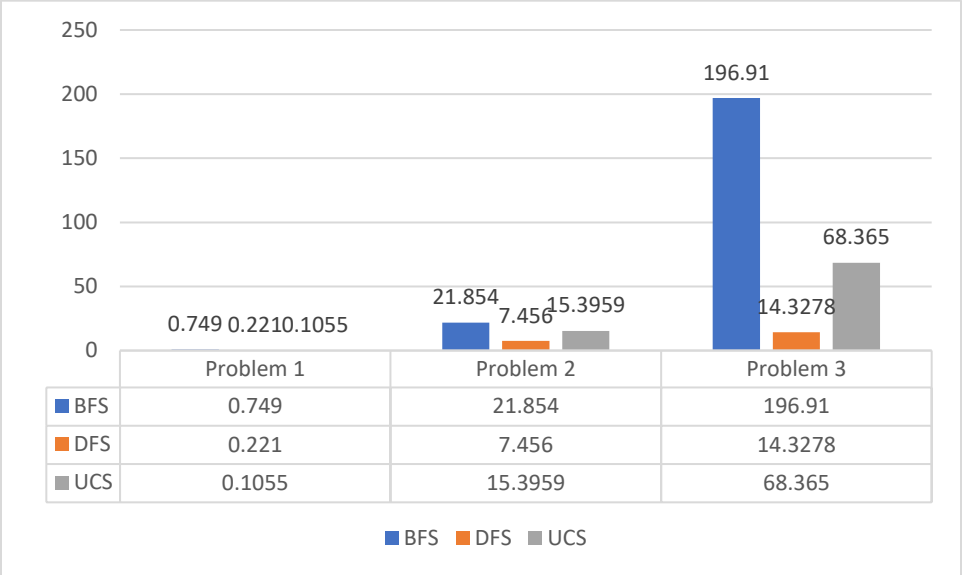
Algorithm	Nodes Expanded	Goal Test	New Nodes	Plan length	Time in seconds
breadth_first_search	14663	18098	129631	12	196.91
breadth_first_tree_search	-	-	-	-	timeout
depth_first_graph_search	627	628	5176	596	14.3278
depth_limited_search	-	-	-	-	timeout
uniform_cost_search	18223	18225	159618	12	68.365
recursive_best_first_search h_1	-	-	-	-	timeout
greedy_best_first_graph_search h_1	5579	5581	49159	22	63.2218
astar_search h_1	18223	18225	159618	12	159.08988
astar_search h_ignore_preconditions	5040	5042	44944	12	230.2545
astar_search h_pg_levelsum	324	326	2993	12	1208.524

Comparison:

Algorithm	Problem	Nodes Expanded	Goal Test	New Nodes	Plan length	Time in seconds
	1	43	56	180	6	0.749
	2	3343	4609	30509	9	21.854
breadth_first_search	3	14663	18098	129631	12	196.91
	1	12	13	48	12	0.221
	2	582	583	5211	575	7.456
depth_first_graph_search	3	627	628	5176	596	14.3278
	1	55	57	224	6	0.1055
	2	4853	4855	44041	9	15.3959
uniform_cost_search	3	18223	18225	159618	12	68.365
	1	41	43	170	6	0.1014
astar_search	2	1450	1452	13303	9	18.6803
[h_ignore_preconditions]	3	5040	5042	44944	12	230.2545
	1	11	13	50	6	2.001
	2	86	88	841	9	181.9675
astar_search [level-sum]	3	324	326	2993	12	1208.524

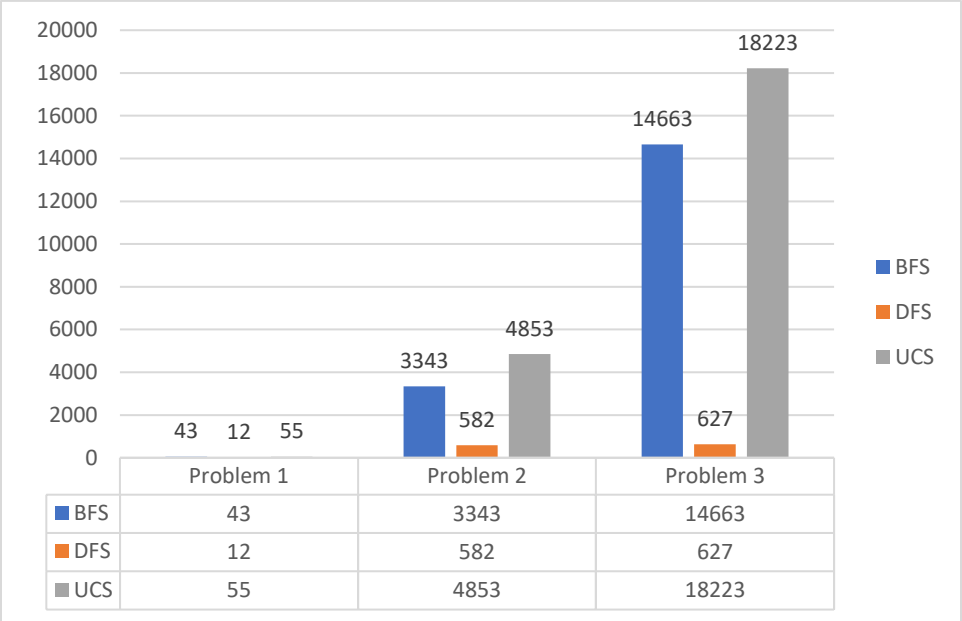
Uniformed Non-Heuristic Search:

The results of BFS, DFS and UCS is summarized in the above table where we can see that the goal test is done on every node.



Execution Time:

From the above chart we can see that the time taken to reach the goal by DFS is lower compared to the UCS and BFS. So DFS is the faster planning algorithm when we just consider execution time.



Nodes Expanded:

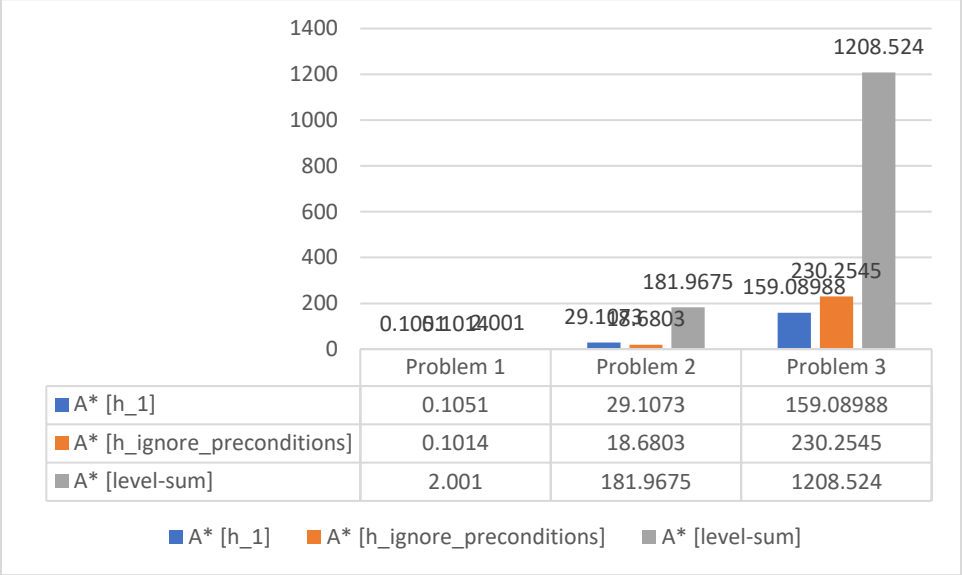
The memory required by an algorithm is analyzed by the number of the expanded nodes. From the above chart we can see that the DFS is less number of nodes compared to the BFS and UCS. Which also explain the reason why DFS took low time.

Optimal Solution:

The path length of the DFS is high compared to the BFS and UCS. BFS and UCS provides the path length of 6, 9 and 12 for all the three problems.

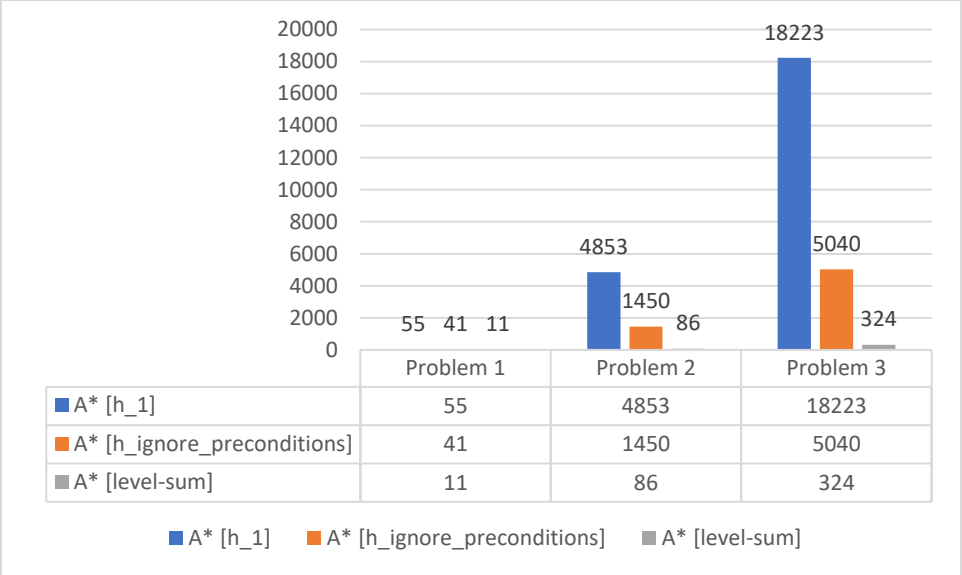
Heuristic Search:

The heuristic planning for analyzed with h_1, h_ingnore_preconditions and level_sum. We can see that the h_1 is same as UCs as the heuristic always returns 1.



Execution Time:

From the above chart we can see that the time taken to reach the goal by h_ignore_precondition is lower compared to the level-sum. A* level-sum suffers with high computation hence it takes more time.



Nodes Expanded:

From the above chart we can see that the number of nodes expanded by the least by level-sum compared to the h_ignore_predctions.

Optimal Solution:

For the given problem all the considered heuristic provide the optimal solution with the path length of 6,9 and 12 respectively.

Best Heuristics:

- Uniform Cost Search (UCS): Comparing with BFS and DFS for the given problem this search will be optimal.
- Breath First Search (BFS): Shortest way to reach the goal, but it takes more compared to the other searches.
- Depth First Search (DFS): Faster compared to the breath first search, but it takes more length to reach the goal, not an optimal solution.
- A* Search: Ignore precondition needs more expansion compared to the level-sum.

I think that for better heuristics, negative effects of the problem make more complicated so removing will easier to calculate.

Optimal Solution:

Problem1:

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

Problem2:

Load(C1, P1, SFO)
Load(C2, P2, JFK)
Load(C3, P3, ATL)
Fly(P1, SFO, JFK)
Fly(P2, JFK, SFO)
Fly(P3, ATL, SFO)
Unload(C3, P3, SFO)
Unload(C2, P2, SFO)
Unload(C1, P1, JFK)

Problem3:

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