Heuristic Analysis Planning and Search

The following is a performance comparison for each of the problems from the planning Algorithms preformed on each problem.

Problem 1:

```
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))
```

For this problem I was able to run all of the algorithms against it both the uninformed and the automatic heuristics with the A* algorithm. Below shows the performance I saw.

Problem1					
Algorithm	Plan Length	Time (s)	Expansions	Goal Tests	New Nodes
breadth_first_search	6	0.035	43	56	180
breadth_first_tree_search	6	0.943	1458	1459	5960
depth_first_graph_search	12	0.007	12	13	48
depth_limited_search	50	0.088	101	271	414
uniform_cost_search	6	0.035	55	57	224
recursive_best_first_search h_1	6	2.77	4229	4230	17029
greedy_best_first_graph_search h_1	6	0.005	7	9	28
astar_search h_1	6	0.04	55	57	224
astar_search h_ignore_preconditions	6	0.041	41	43	170
astar_search h_pg_levelsum	6	0.894	11	13	50

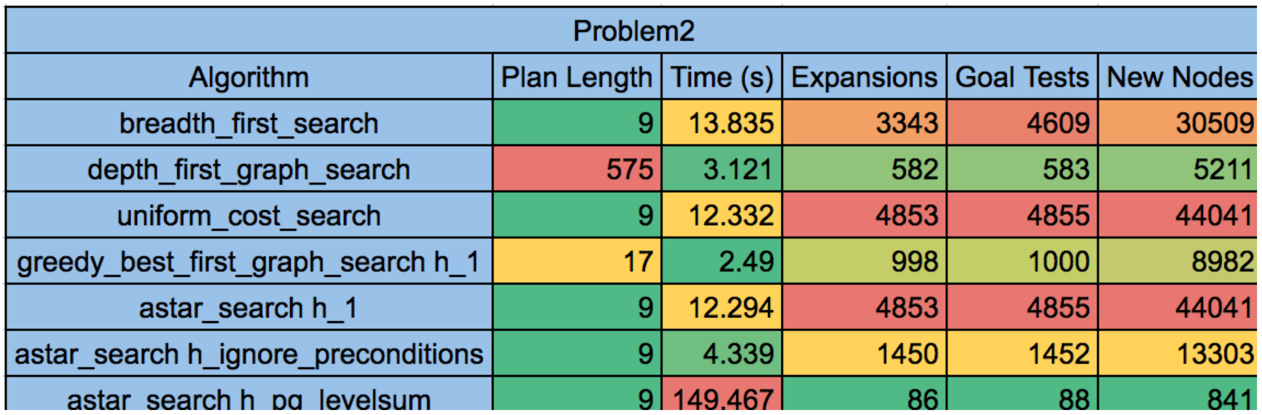
The first thing that jumps out from this data is that both the uninformed and heuristic searches preform pretty similarly for such a simple problem. The uninformed heuristics such as `breadth_first_search` has very similar numbers to `astar_search h_ignore_preconditions` for problem 1.

However, from this data I would take a close look at both the `greedy_best_first_graph_search h_1` and `astar_search h_ignore_preconditions` because they both found the min path very quickly. Also because I know its a relatively easy problem I am not very worried about the greedy algorithm overestimating. Though because of this issue with the Greedy Algorithm I would go with `astar_search h_ignore_preconditions` as the clear winner.

Problem 2

```
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))
```

For this problem I had to drop some of the algorithms such as the `breadth_first_tree_search` because they ran into loops while processing and got hung in an infinite loop and would never finish. The results I saw with the algorithms that did finish are below:



Problem2					
Algorithm	Plan Length	Time (s)	Expansions	Goal Tests	New Nodes
breadth_first_search	9	13.835	3343	4609	30509
depth_first_graph_search	575	3.121	582	583	5211
uniform_cost_search	9	12.332	4853	4855	44041
greedy_best_first_graph_search h_1	17	2.49	998	1000	8982
astar_search h_1	9	12.294	4853	4855	44041
astar_search h_ignore_preconditions	9	4.339	1450	1452	13303
astar_search h_pg_levelsum	9	149.467	86	88	841

From this data I would conclude that the uninformed algorithms preformed quite poorly with `breadth_first_search` taking almost 3 times as long as `astar_search h_ignore_preconditions` where in problem 1 they had been quite similar. Also `depth_first_graph_search` is really fast in finding a way to meet the goals but the path it takes is not realistic as it has a length of 596. The two best performing algorithms are the `astar_search h_ignore_preconditions` and `astar_search h_pg_levelsum` the both find the correct answer at the min plan length. the main difference between the two is since the ignore preconditions heuristic drops all preconditions and every action becomes applicable in every state it has many more expansions, goal tests, and new nodes. Where as `astar_search h_pg_levelsum` has much fewer of these but takes a lot longer to run since first it has to find all the goals then add up all the layers. For problem2 I would pick between the two based on the constraints I had since `astar_search h_ignore_preconditions` has to create so many more new nodes if I was memory constrained this is not the one I would pick. As long as I have plenty of memory I would choose the `astar_search h_ignore_preconditions` since it runs so much faster.

Problem 3

```
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))
```

This problem was similar to Problem 2 in that I had to drop the searches that are sensitive to loops as they would get stuck and not finish. The results I found with the algorithms that did finish are below:

Problem3					
Algorithm	Plan Length	Time (s)	Expansions	Goal Tests	New Nodes
breadth_first_search	12	101.662	14663	18098	129631
depth_first_graph_search	596	3.313	627	628	5176
uniform_cost_search	12	54.211	18236	18238	159726
greedy_best_first_graph_search h_1	26	16.89	5623	5625	49495
astar_search h_1	12	54.459	18236	18238	159726
astar_search h_ignore_preconditions	12	17.279	5038	5040	44926
astar_search h_pg_levelsum	12	775.612	314	316	2894

The results from this problem were similar to Problem 2 only exaggerated further. The uninformed algorithms preformed even worse taking significantly more time like the 101.662 seconds `breadth_first_search` took or if they were fast like

`depth_first_graph_search` then they took way too long of paths (596). The two heuristic algorithms `astar_search h_ignore_preconditions` and `astar_search h_pg_levelsum` strength's and weaknesses were also exaggerated further with `astar_search h_ignore_preconditions` needing even more expansions, goal tests, and new nodes. As well as `astar_search h_pg_levelsum` needing even more time.

Conclusions

For these problems the automatic heuristic algorithms performed much better if I had to choose one to use for these types of problems I would go with `astar_search h_ignore_preconditions` because of the superior speed it showed.