Air Cargo Problem 1 & 2

Search Type	Problem	Actions	Expansions	Goal Tests	New Nodes	Plan Length	Elapsed Time
breadth_first_search	Air Cargo Problem 1	20	43	56	178	6	0.002414419
depth_first_graph_search	Air Cargo Problem 1	20	21	22	84	20	0.001248689
uniform_cost_search	Air Cargo Problem 1	20	60	62	240	6	0.003701539
greedy_best_first_graph_search with h_unmet_goals	Air Cargo Problem 1	20	7	9	29	6	0.000629323
greedy_best_first_graph_search with h_pg_levelsum	Air Cargo Problem 1	20	6	8	28	6	0.084912862
greedy_best_first_graph_search with h_pg_maxlevel	Air Cargo Problem 1	20	6	8	24	6	0.061936103
greedy_best_first_graph_search with h_pg_setlevel	Air Cargo Problem 1	20	6	8	28	6	0.254290131
astar_search with h_unmet_goals	Air Cargo Problem 1	20	50	52	206	6	0.004021022
astar_search with h_pg_levelsum	Air Cargo Problem 1	20	28	30	122	6	0.211817105
astar_search with h_pg_maxlevel	Air Cargo Problem 1	20	43	45	180	6	0.19963853
astar_search with h_pg_setlevel	Air Cargo Problem 1	20	33	35	138	6	0.345972114
breadth_first_search	Air Cargo Problem 2	72	3343	4609	30503	9	0.50554769
depth_first_graph_search	Air Cargo Problem 2	72	624	625	5602	619	0.617331535
uniform_cost_search	Air Cargo Problem 2	72	5154	5156	46618	9	0.813684989
greedy_best_first_graph_search with h_unmet_goals	Air Cargo Problem 2	72	17	19	170	9	0.005457783
greedy_best_first_graph_search with h_pg_levelsum	Air Cargo Problem 2	72	9	11	86	9	1.202490742
greedy_best_first_graph_search with h_pg_maxlevel	Air Cargo Problem 2	72	27	29	249	9	1.8329536
greedy_best_first_graph_search with h_pg_setlevel	Air Cargo Problem 2	72	9	11	84	9	3.966579294
astar_search with h_unmet_goals	Air Cargo Problem 2	72	2467	2469	22522	9	0.537189862
astar_search with h_pg_levelsum	Air Cargo Problem 2	72	357	359	3426	9	31.72259381
astar_search with h_pg_maxlevel	Air Cargo Problem 2	72	2887	2889	26594	9	186.7280147
astar_search with h_pg_setlevel	Air Cargo Problem 2	72	1037	1039	9605	9	362.087484

Air Cargo Problem 3 & 4

Search Type	Problem	Actions	Expansions	Goal Tests	New Nodes	Plan Length	Elapsed Time
uniform_cost_search	Air Cargo Problem 3	88	18510	18512	161936	12	3.43086771
greedy_best_first_graph_search with h_unmet_goals	Air Cargo Problem 3	88	25	27	230	15	0.008802804
greedy_best_first_graph_search with h_pg_levelsum	Air Cargo Problem 3	88	14	16	126	14	2.96006642
greedy_best_first_graph_search with h_pg_maxlevel	Air Cargo Problem 3	88	21	23	195	13	2.950953477
greedy_best_first_graph_search with h_pg_setlevel	Air Cargo Problem 3	88	35	37	345	17	22.6747849
astar_search with h_unmet_goals	Air Cargo Problem 3	88	7388	7390	65711	12	2.002800088
astar_search with h_pg_levelsum	Air Cargo Problem 3	88	369	371	3403	12	62.12944247
uniform_cost_search	Air Cargo Problem 4	104	113339	113341	1066413	14	29.00661468
greedy_best_first_graph_search with h_unmet_goals	Air Cargo Problem 4	104	29	31	280	18	0.015597347
greedy_best_first_graph_search with h_pg_levelsum	Air Cargo Problem 4	104	17	19	165	17	5.172758431

greedy_best_first_graph_search with h_pg_maxlevel	Air Cargo Problem 4	104	56	58	580	17	8.571929625
greedy_best_first_graph_search with h_pg_setlevel	Air Cargo Problem 4	104	107	109	1164	23	101.9006727
astar_search with h_unmet_goals	Air Cargo Problem 4	104	34330	34332	328509	14	12.83332614
astar_search with h_pg_levelsum	Air Cargo Problem 4	104	1208	1210	12210	15	334.3509003

Report results:

Search Time Analysis:

As the **domain size** (number of cargos, planes, and airports) increases from **Air Cargo Problem 1 to 4**, the **search time** generally increases across all search algorithms. Uniform-cost search shows a steep rise in search time, from 0.5 seconds in Problem 1 to 29 seconds in Problem 4, due to the exhaustive nature of the algorithm. In contrast, **greedy best-first search** consistently performs faster because it does not aim for optimality and only uses the heuristic for guidance. The **A*** search, particularly with complex heuristics like h_pg_levelsum, sees the most dramatic increase in search time, especially in larger problems like Problem 4, where it takes 334 seconds, largely due to the higher computational overhead of evaluating the heuristic at each node.

Optimality:

In terms of **optimality**, **A*** is guaranteed to find the **optimal solution** (in terms of the shortest plan or number of actions) because it balances both the path cost g(n) and the heuristic h(n). This is evident across all four problems, where A* always finds the shortest plan (e.g., 6 steps in Problem 1 and 14 in Problem 4). On the other hand, **greedy best-first search**, which focuses solely on the heuristic, often produces **suboptimal solutions**. For instance, in Problem 4, it finds a plan with 18 steps compared to the optimal 14 steps found by A*, trading off optimality for faster computation. **Uniform-cost search**, while also finding the optimal solution, tends to take longer than A* due to its lack of heuristic guidance.

Complexity:

Heuristic Influence:

The **complexity** of the search algorithms grows significantly with the increase in domain size from Problems 1 to 4. This is reflected in the number of **expansions**, **goal tests**, and **new nodes** generated. For example, in **uniform-cost search**, expansions increase from 43 in Problem 1 to 113,339 in Problem 4, demonstrating the exhaustive nature of UCS as it expands nodes without the benefit of a heuristic. **A*** search strikes a balance between complexity and optimality, expanding more nodes than greedy search but fewer than uniform-cost search, especially when using efficient heuristics like h_pg_levelsum. **Greedy best-first search**, by focusing purely on the heuristic, minimizes node expansions, making it much faster but often at the expense of finding the optimal solution.

The choice of **heuristic** has a significant impact on both search time and complexity. Simpler heuristics, such as **h_unmet_goals**, result in faster computations but less informed guidance, leading to suboptimal solutions in both A* and greedy search. More sophisticated heuristics, like **h_pg_levelsum**, offer better guidance by incorporating more detailed information from planning graphs but come with a high computational cost. For example, in **Problem 4**, A* with h_pg_levelsum expands only 1,208 nodes, far fewer than uniform-cost search, but it takes 334 seconds due to the overhead of calculating the heuristic at each node. In contrast, **greedy best-first search** with h_pg_levelsum is faster, taking only 5.17 seconds, but it sacrifices optimality by expanding only 17 nodes and generating a suboptimal plan.

1. Which algorithm or algorithms would be most appropriate for planning in a very restricted domain (i.e., one that has only a few actions) and needs to operate in real time?

For a restricted domain with few actions that needs to operate in real time, **greedy best-first search** would be the most appropriate. Since it focuses solely on the heuristic and expands fewer nodes, it provides **quick, suboptimal solutions**, which is ideal for real-time planning where speed is more important than optimality. The lack of complex calculations per node ensures that the search is fast and responsive.

2. Which algorithm or algorithms would be most appropriate for planning in very large domains (e.g., planning delivery routes for all UPS drivers in the U.S. on a given day)?

In very large domains, **greedy best-first search** with an informed heuristic would be suitable if **speed is critical** and **suboptimal solutions are acceptable**. However, if some degree of solution quality is important, **A*** would be more appropriate. A* with an efficient, domain-specific heuristic can handle the trade-off between expanding fewer nodes and ensuring near-optimal solutions, although it may require significant computational resources.

3. Which algorithm or algorithms would be most appropriate for planning problems where it is important to find only optimal plans?

For problems that require finding **only optimal plans**, **A*** is the best choice. It balances the path cost g(n) and heuristic h(n), guaranteeing the optimal solution as long as the heuristic is admissible. Alternatively, **uniform-cost search** could be used if no heuristic is available, but A* is generally faster because it uses heuristic guidance to reduce unnecessary node expansions.