- Use a table or chart to analyze the number of nodes expanded against number of actions in the domain
- Use a table or chart to analyze the search time against the number of actions in the domain
- Use a table or chart to analyze the length of the plans returned by each algorithm on all search problems

Problem 1 (# Actions 20)

Search	**Plan Length**	**Execution Time (Secs)**	Node Expansions	Optimal (Y/N)
breadth_first_search	**6**	0.007108500000413187	43	**Y**
depth_first_graph_search	20	0.0035216239998590027	21	N
uniform_cost_search	**6**	0.01036465200013481	60	**Y**
greedy_best_first_graph_search with h_unmet_goals	**6**	**0.0016989719997582142**	7	**Y**
greedy_best_first_graph_search with h_pg_levelsum	**6**	0.19978979100005745	**6**	**Y**
greedy_best_first_graph_search with h_pg_maxlevel	**6**	0.1345460259999527	**6**	**Y**
greedy_best_first_graph_search with h_pg_setlevel	**6**	0.5793173950000892	**6**	**Y**
astar_search with h_unmet_goals	**6**	0.014761229000214371	50	**Y**
astar_search with h_pg_levelsum	**6**	0.45813816099962423	28	**Y**
astar_search with h_pg_maxlevel	**6**	0.48297298299985414	43	**Y**
astar_search with h_pg_setlevel	**6**	1.35580133700023	33	**Y**

Example :
greedy_best_first_graph_search with h_pg_levelsum...

Plan length: 6 Time elapsed in seconds: 0.19978979100005745
Load(C1, P1, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)

Problem 2 (# Actions 72)

Search	**Plan Length**	**Execution Time (Secs)**	Node Expansions	Optimal (Y/N)
breadth_first_search	**9**	2.0192135680003958	3343	**Y**
depth_first_graph_search	619	3.2947449069997674	624	N
uniform_cost_search	**9**	3.389986189999945	5154	**Y**
greedy_best_first_graph_search with h_unmet_goals	**9**	**0.01904553299982581**	17	**Y**
greedy_best_first_graph_search with h_pg_levelsum	**9**	3.9797020390001308	**9**	**Y**
greedy_best_first_graph_search with h_pg_maxlevel	**9**	6.150366299000325	27	**Y**
greedy_best_first_graph_search with h_pg_setlevel	**9**	14.267016690000219	**9**	**Y**
astar_search with h_unmet_goals	**9**	2.23775918299998	2467	**Y**
astar_search with h_pg_levelsum	**9**	107.26977628399982	357	**Y**
astar_search with h_pg_maxlevel	**9**	626.0371127400003	2887	**Y**
astar_search with h_pg_setlevel	**9**	1296.5292322039995	1037	**Y**

Example :
using greedy_best_first_graph_search with h_unmet_goals...

Plan length: 9 Time elapsed in seconds: 0.01904553299982581
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Load(C3, P3, ATL)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
Fly(P3, ATL, SFO)
Unload(C3, P3, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)

For Problems 3 & 4
run **at least** one uninformed search, two heuristics with greedy best first search, and two heuristics with A*

These constitute one Uninformed Search, two Greedy Best First Searches and two A* Searches.

Problem 3 (# Actions 88)

Search	**Plan Length**	**Execution Time (Secs)**	Node Expansions	Optimal (Y/N)
breadth_first_search	**12**	10.715695653999319	14663	**Y**
greedy_best_first_graph_search with h_unmet_goals	15	**0.036721627999213524**	25	**N**
greedy_best_first_graph_search with h_pg_levelsum	14	9.411978191000344	**14**	**N**
astar_search with h_unmet_goals	**12**	8.606292744999337	7388	**Y**
astar_search with h_pg_levelsum	**12**	204.84883926800012	369	**Y**

Example :
Solving Air Cargo Problem 3 using astar_search with h_unmet_goals...

Plan length: 12 Time elapsed in seconds: 8.606292744999337
Load(C2, P2, JFK)
Fly(P2, JFK, ATL)
Load(C3, P2, ATL)
Fly(P2, ATL, ORD)
Load(C4, P2, ORD)
Fly(P2, ORD, SFO)
Unload(C4, P2, SFO)
Unload(C2, P2, SFO)
Load(C1, P2, SFO)
Fly(P2, SFO, JFK)
Unload(C3, P2, JFK)
Unload(C1, P2, JFK)

Problem 4 (# Actions 104)

Search	**Plan Length**	**Execution Time (Secs)**	Node Expansions	Optimal (Y/N)
breadth_first_search	**14**	99.08785985899999	99736	**Y**

greedy_best_first_graph_search with h_unmet_goals	18	**0.06133506399999078**	29	**N**
greedy_best_first_graph_search with h_pg_levelsum	17	16.97992322799996	**17**	**N**
astar_search with h_unmet_goals	**14**	57.125173928000095	34330	**Y**
astar_search with h_pg_levelsum	15	1165.584004107	1208	**N**

Example :

Solving Air Cargo Problem 4 using astar_search with h_unmet_goals...

Plan length: 14 Time elapsed in seconds: 57.125173928000095
Load(C2, P2, JFK)
Fly(P2, JFK, ATL)
Load(C3, P2, ATL)
Fly(P2, ATL, ORD)
Load(C4, P2, ORD)
Load(C5, P2, ORD)
Fly(P2, ORD, SFO)
Unload(C4, P2, SFO)
Unload(C2, P2, SFO)
Load(C1, P2, SFO)
Fly(P2, SFO, JFK)
Unload(C5, P2, JFK)
Unload(C3, P2, JFK)
Unload(C1, P2, JFK)

Discussion of results:

- From the results above with regards to the search complexity, when nodes expanded for the search are considered, **greedy best first search** consistently performs better than other algorithms even as the actionsize increases.

- **Depth first graph search** does find a quick solution but lacks optimality. It is not optimal because it does not consider if a node is better than another, it just explores the nodes that take it as deep as possible in the graph even if the goal is to the other direction

- From the results above, the **uniform cost search** expands the most nodes followed by breadth first search (Problems 1& 2).

- From the results above, a*star algorithms having better heuristic out-perform the non-heuristic searches as the problem complexity increases. (problem 3) further, a*star algorithms are the slowest except for astar_search_h_unmet_goals.

- For search time i.e. execution speed is considered, for a small problem domain (problem 1) the execution time is very close for different algorithms. The greedy best first searches are once again better even with **increasing actions**. (problem 1,2,3,4)

Questions

1. Restricted Domain : 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?

   Execution time to find the goal critical here. I would settle on the algorithm that runs the fastest. Using this idea Greedy Best First Search (problem 1). Breath First Search would also be a good choice given that the problem domain is very restricted also executes fast and finds the optimal plan (Problem 1&2)

2. Large Domain : 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)

   Our goal here is to find possible optimal plans but also take time and complexity into consideration. A* star algorithm would be the right choice with a better heuristic so that fewer node expansion is likely to lead us to the goal choice. It delivers near optimum paths with very good search performance.

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

   From results above, Uniform Cost Search or Breath First Search (better choice based on node expansion and speed) are the two choices for finding optimal plans as seen from results above.