The goal of the experiment is to understand the tradeoffs in speed, optimality, and complexity of progression search as problem size increases.

List of air cargo problems:

1. Air Cargo Problem 1
cargos = ['C1', 'C2']
planes = ['P1', 'P2']
airports = ['JFK', ‘SFO']

2. Air Cargo Problem 2
cargos = ['C1', 'C2', 'C3']
planes = ['P1', 'P2', 'P3']
airports = ['JFK', 'SFO', 'ATL']

3. Air Cargo Problem 3
cargos = ['C1', 'C2', 'C3', 'C4']
planes = ['P1', 'P2']
airports = ['JFK', 'SFO', 'ATL', 'ORD']

4. Air Cargo Problem 4
cargos = ['C1', 'C2', 'C3', 'C4', 'C5']
planes = ['P1', 'P2']
airports = ['JFK', 'SFO', 'ATL', 'ORD']

List of search functions:

1. breadth_first_search (BFS)
2. depth_first_graph_search (DFGS)
3. uniform_cost_search (UCS)
4. greedy_best_first_graph_search (GBFGS) h_unmet_goals
5. greedy_best_first_graph_search (GBFGS) h_pg_levelsum
6. greedy_best_first_graph_search (GBFGS) h_pg_maxlevel
7. greedy_best_first_graph_search (GBFGS) h_pg_setlevel
8. astar_search h_unmet_goals
9. astar_search h_pg_levelsum
10. astar_search h_pg_maxlevel
11. astar_search h_pg_setlevel

Air Cargo Problem 1 analysis:

	Actions	Expansions	Goal Tests	New Nodes	Plan length	Time elapsed in seconds
breadth_first_search	20	43	56	178	6	0.006885124952532351
depth_first_graph_search	20	21	22	84	20	0.003198263002559427
uniform_cost_search	20	60	62	240	6	0.008788752951659262
greedy_best_first_graph_search with h_unmet_goals	20	7	9	29	6	0.001440769061461899
greedy_best_first_graph_search with h_pg_levelsum	20	6	8	28	6	0.298236223985441 03
greedy_best_first_graph_search with h_pg_maxlevel	20	6	8	24	6	0.228283990058116 61
greedy_best_first_graph_search with h_pg_setlevel	20	6	8	28	6	0.866398683050647 4
astar_search with h_unmet_goals	20	50	52	206	6	0.008684478001669 05
astar_search with h_pg_levelsum	20	28	30	122	6	0.762341671041213
astar_search with h_pg_maxlevel	20	43	45	180	6	0.792402382008731 4
astar_search with h_pg_setlevel	20	33	35	138	6	2.339476156048476 7

Air Cargo Problem 2 analysis:

	Actions	Expansions	Goal Tests	New Nodes	Plan length	Time elapsed in seconds
breadth_first_search	72	3343	4609	30503	9	1.861741965985857
depth_first_graph_search	72	624	625	5602	619	2.455505367019213 7
uniform_cost_search	72	5154	5156	46618	9	3.058718324988149
greedy_best_first_graph_search with h_unmet_goals	72	17	19	170	9	0.017277345992624476
greedy_best_first_graph_search with h_pg_levelsum	72	9	11	86	9	6.8627280639484525
greedy_best_first_graph_search with h_pg_maxlevel	72	27	29	249	9	13.88831416296307
greedy_best_first_graph_search with h_pg_setlevel	72	9	11	84	9	18.76257549994625
astar_search with h_unmet_goals	72	2467	2469	22522	9	2.036920440965332
astar_search with h_pg_levelsum	72	357	359	3426	9	180.93695470900275
astar_search with h_pg_maxlevel	72	2887	2889	26594	9	1056.855135875987
astar_search with h_pg_setlevel	72	1037	1039	9605	9	1451.7230668379925

From the above results for Problem 1 and 2:

* We can clearly **exclude DFGS** from Problems 3 and 4, since it has a much higher plan length: 619 with 72 actions in case of problem 2. Whereas, all the rest of the search algorithms have a much lesser plan length.
* Next, based on search time we could **also rule out, Astar search with Max level and Set level heuristics.**
* Both, BFS and UCS obtained optimal plan length.
* GBFGS algorithms have the lowest node expansion of all the algorithms, with minimal search times.

Air Cargo Problem 3 analysis:

	Actions	Expansions	Goal Tests	New Nodes	Plan length	Time elapsed in seconds
breadth_first_search	88	14663	18098	129625	12	10.0132953110151
uniform_cost_search	88	18510	18512	161936	12	13.919488257961348
greedy_best_first_graph_search with h_unmet_goals	88	25	27	230	15	0.033993331948295236
greedy_best_first_graph_search with h_pg_levelsum	88	14	16	126	14	15.607563314959407
greedy_best_first_graph_search with h_pg_maxlevel	88	21	23	195	13	18.433272599009797
greedy_best_first_graph_search with h_pg_setlevel	88	35	37	345	17	89.55691108398605
astar_search with h_unmet_goals	88	7388	7390	65711	12	7.625149716972373
astar_search with h_pg_levelsum	88	369	371	3403	12	276.37868409301154

Air Cargo Problem 4 analysis:

	Actions	Expansions	Goal Tests	New Nodes	Plan length	Time elapsed in seconds
breadth_first_search	104	99736	114953	944130	14	86.73846791905817
uniform_cost_search	104	113339	113341	1066413	14	104.86973032006063
greedy_best_first_graph_search with h_unmet_goals	104	29	31	280	18	0.059426766005344469
greedy_best_first_graph_search with h_pg_levelsum	104	17	19	165	17	27.592078976915218
greedy_best_first_graph_search with h_pg_maxlevel	104	56	58	580	17	66.06880511296913

	Actions	Expansions	Goal Tests	New Nodes	Plan length	Time elapsed in seconds
greedy_best_first_graph_search with h_pg_setlevel	104	107	109	1164	23	409.9392328079557
astar_search with h_unmet_goals	104	34330	34332	328509	14	49.14400636195205
astar_search with h_pg_levelsum	104	1208	1210	12210	15	1588.006164241

From the above results for Problem 3 and 4:

* We can infer that the GBFGS algorithms have the lowest amount of nodes expansion even for large number of actions, when compared with algorithms from the Uninformed search family and from the Astar search family.
* The uninformed search algorithms – BFS and UCS – are expanding with most nodes although it does not seem to have an equivalent impact on its speed.

Use your results to answer the following questions:

- 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?
  Based on the above results, these would be:
  - greedy_best_first_graph_search with h_unmet_goals
  - uniform_cost_search

- 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)
  Based on the above results, this would be:
  - greedy_best_first_graph_search with h_unmet_goals

- Which algorithm or algorithms would be most appropriate for planning problems where it is important to find only optimal plans?
  Based on the above results, these would be:
  - uniform_cost_search
  - breadth_first_search,

  for both restricted and large domains.