- Air Cargo Action Schema:

```
Action(Load(c, p, a),
   PRECOND: At(c, a) ∧ At(p, a) ∧ Cargo(c) ∧ Plane(p) ∧
Airport(a)
   EFFECT: ¬ At(c, a) ∧ In(c, p))
Action(Unload(c, p, a),
   PRECOND: In(c, p) ∧ At(p, a) ∧ Cargo(c) ∧ Plane(p) ∧
Airport(a)
   EFFECT: At(c, a) ∧ ¬ In(c, p))
Action(Fly(p, from, to),
   PRECOND: At(p, from) ∧ Plane(p) ∧ Airport(from) ∧ Airport(to)
   EFFECT: ¬ At(p, from) ∧ At(p, to))
```

- Problem 1 initial state and goal:

```
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))
```

- Problem 2 initial state and goal:

```
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))
```

- Problem 3 initial state and goal:

```
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))
```

1. **Provide an optimal plan for Problems 1, 2, and 3.**

	P1 – 6 steps	P2 – 9 steps	P3 – 12 steps
Optimal Plan	Load(C2, P2, JFK) Fly(P1, SFO, JFK) Fly(P2, JFK, SFO) Unload(C1, P1, JFK) Unload(C2, P2, SFO)	Load(C1, P1, SFO) Fly(P1, SFO, JFK) Unload(C1, P1, JFK) Load(C2, P2, JFK) Fly(P2, JFK, SFO) Unload(C2, P2, SFO) Load(C3, P3, ATL) Fly(P3, ATL, SFO) Unload(C3, P3, SFO)	Load(C1, P1, SFO) Fly(P1, SFO, ATL) Load(C3, P1, ATL) Fly(P1, ATL, JFK) Unload(C1, P1, JFK) Load(C2, P2, JFK) Fly(P2, JFK, ORD) Load(C4, P2, ORD) Fly(P2, ORD, SFO) Unload(C2, P2, SFO) Unload(C3, P1, JFK) Unload(C4, P2, SFO)

Problem	Search	Plan length	Expansions	Time Sec
1	breath-first	6	43	0.0255
1	breath-first-tree	6	1458	0.8231
	Depth first graph search	12	12	0.00713
1	Depth limited search	50	101	0.079
1	Uniform cost search	6	55	0.0372
1	Recursive_best_first_search_h_1	6	4229	2.3911
1	Greedy_best_first_graph_search h_1	6	7	0.004
1	Astar_search h_1	6	55	0.032
1	astar_search h_ignore_preconditions	6	41	0.379
1	star_search h_pg_levelsum	6	11	1.179
2	breath-first	9	2844	9.182
2	breath-first-tree	-	-	–

2	Depth first graph search	38	44	0.079
2	Depth limited search	-	-	-
2	Uniform cost search	9	4281	29.920
2	Recursive_best_first_search_h_1	-	-	-
2	Greedy_best_first_graph_search_h_1	19	478	1.851
2	Astar_search h_1	9	4281	32.299
2	astar_search h_ignore_preconditions	9	1297	9.047
2	star_search h_pg_levelsum	9	177	153.037
3	breath-first	12	14663	95.842
3	breath-first-tree	-	-	–
3	Depth first graph search	596	627	2.9660
3	Depth limited search	-	-	-
3	Uniform cost search	12	18151	386.722
3	Recursive_best_first_search_h1	-	-	-
3	Greedy_best_first_graph_search_1	26	5398	81.2500
3	Astar_search h_1	12	18151	353.836
3	astar_search h_ignore_preconditions	12	5118	82.344
3	star_search h_pg_levelsum	12	404	839.148

- ***Had to be cancelled as were going beyond 10 min.***

2. Compare and contrast non-heuristic search result metrics for P1, P2 and P3

Optimal Plan

For all the problems, Breath first, and uniform cost produced the optimal plan length. Breath first and uniform cost are good choices among non-

heuristic choices for P1, P2 and P3. BFS is anyways complete and optimal with the only downside of high memory usage.

Execution speed and Memory usage

Looking at the speed, Depth first graph has the least execution time and causes least amount of node expansions.

3. **Compare and contrast heuristic search result metrics using A* with the "ignore preconditions" and "level-sum" heuristics for Problems 1, 2, and 3.**

   All the heuristics yielded an optimal plan. Ignore-preconditions was faster compared to level-sum but the level sum heuristics caused fewer node expansions. Ignore-precondition took least time for problem 2 and 3.

4. **What was the best heuristic used in these problems? Was it better than non-heuristic search planning methods for all problems? Why or why not?**

   While all the heuristic yielded an optimal plan, **ignore-preconditions** was the best. In my tests it turned out to be the fastest though did not result in fewest expansions. **Ignore-precondition** execution time and expansion was not better than non-heuristic search planning for problem 1. Ignore-precondition is faster because of the reduced expansions compared to un-informed searches and hence speedy heuristic calculation.