For this AI project, we are creating an agent to plan cargo movements via plane, between airports. The environment is completely deterministic and the actions available to the agent are load, unload, and fly. This is illustrated via the action scheme below. The objects available are planes, airports, and cargo. We will execute classical planning searches without heuristics as well as heuristic driven A* searches and searches using planning graphs.

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 Schema:

Problem 1:

The task for problem 1 is to transport 2 cargo units [C1, C2] using 2 planes [P1, P2], between 2 airports [SFO, JFK].

Problem 1 Initial State and Goals:

```
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:

The task for problem 2 is to transport 3 cargo units [C1, C2, C3] using 3 planes [P1, P2, P3], between 3 airports [SFO, JFK, ATL].

Problem 2 Initial State and Goals:

```
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:

The task for problem 3 is to transport 4 cargo units [C1, C2, C3, C4] using 2 planes [P1, P2], between 4 airports [SFO, JFK, ATL, ORD].

Problem 3 Initial State and Goals:

```
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))
```

Optimal Solutions:

Problem 1 Optimal Solution:

Plan length: 6

Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)

Problem 2 Optimal Solution:

Plan length: 9

Load(C1, P1, SFO)
Load(C2, P2, JFK)
Load(C3, P3, ATL)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)
Fly(P3, ATL, SFO)
Unload(C3, P3, SFO)

Problem 3 Optimal Solution:

Plan length: 12

Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P2, JFK, ORD)
Load(C4, P2, ORD)
Fly(P1, SFO, ATL)
Load(C3, P1, ATL)
Fly(P1, ATL, JFK)
Unload(C1, P1, JFK)
Unload(C3, P1, JFK)
Fly(P2, ORD, SFO)
Unload(C2, P2, SFO)
Unload(C4, P2, SFO)

Non-Heuristic Solutions:

Non-Heuristic Metrics for Problem 1

Search Type	Plan Length	Expansions	Goal Tests	New Nodes	Elapsed Time(s)
Breadth First	6	43	56	180	0.18
Depth First Graph	20	21	22	84	0.11
Breadth First Tree	6	1458	1459	5960	5.35
Depth Limited	50	101	271	414	0.56

4 search types were executed for plan 1 and completed within a reasonable time frame. Breadth first search and breadth first tree search returned the optimal solution, whereas the others did not.

Most Optimal Solution: Both breadth first search. The process for breadth first was more optimal than breadth first tree in this case since it was faster and less expansive.

Least Optimal Solution: Depth limited search was least optimal solution, with 44 additional steps beyond the optimal solution.

Most Expansive or Longest Solution: Breadth first graph search was much longer and more expansive than all other searches.

Least Expansive or Fastest Solution: Depth first graph search was the fastest and least expansive solution, but was sub-optimal, using 14 additional steps beyond that of the 6-step optimal solution.

Non-Heuristic Metrics for Problem 2

Search Type	Plan Length	Expansions	Goal Tests	New Nodes	Elapsed Time(s)
Breadth First	9	3343	4609	30509	68.94
Depth First Graph	619	624	625	5602	18.64
Breadth First Tree	Run time excessive	Run time excessive	Run time excessive	Run time excessive	Run time excessive
Depth Limited	50	222719	2053741	2054119	19518.44

4 searches were attempted for problem 2, with only 3 completing in a reasonable time frame. Breadth first search found the optimal solution, whereas the others did not.

Most Optimal Solution: Breadth first search found the optimal solution.

Least Optimal Solution: Depth first graph search found a solution with an additional 610 steps beyond the optimal solution.

Most Expansive or Longest Solution: For searches that completed depth limited search was the longest and most expansive, though breadth first tree search is the likely winner.

Least Expansive or Fastest Solution: The least expansive solution was also the least optimal, breadth first search.

Non-Heuristic Metrics for Problem 3

Search Type	Plan Length	Expansions	Goal Tests	New Nodes	Elapsed Time(s)
Breadth First	12	14463	18098	129631	572.67
Depth First Graph	392	408	409	3364	9.68
Breadth First Tree	Run time excessive	Run time excessive	Run time excessive	Run time excessive	Run time excessive
Depth Limited	Run time excessive	Run time excessive	Run time excessive	Run time excessive	Run time excessive

4 searches were attempted for problem 3, with only 2 completing in a reasonable time frame. Breadth first search found the optimal solution, whereas the others did not.

Most Optimal Solution: Breadth first search found the optimal plan, though it was the longest and most expansive of completed searches.

Least Optimal Solution: Depth first graph search was the least optimal plan with 380 steps beyond the optimal solution.

Most Expansive or Longest Solution: For completed searches the most expansive and longest solution was breadth first search.

Least Expansive or Fastest Solution: Depth limited graph search was the fastest, least expansive, and least optimal completed search.

Heuristic/Planning Graph Driven Searches

Heuristic Models for Problem 1

Search Type	Plan Length	Expansions	Goal Tests	New Nodes	Elapsed Time(s)
A* h_1	6	55	57	224	0.24
A* h_1_ignore_preconditions	6	41	43	170	0.25
A* h_pg_levelsum	6	11	13	50	5.68

Heuristic Models for Problem 2

Search Type	Plan Length	Expansions	Goal Tests	New Nodes	Elapsed Time(s)
A* h_1	9	4853	4855	4401	241.19
A* h_1_ignore_preconditions	9	1506	1508	13820	85.62
A* h_pg_levelsum	9	86	88	841	587.98

Heuristic Models for Problem 3

Search Type	Plan Length	Expansions	Goal Tests	New Nodes	Elapsed Time(s)
A* h_1	12	18236	18238	159276	2907.49
A* h_1_ignore_preconditions	12	5118	5120	4650	460.90
A* h_pg_levelsum	12	404	406	3718	21017.08

Trends held for problems 1, 2, and 3, though scale rose with problem complexity (3 > 2 >> 1). Each A* search found the optimal solution. Both heuristics outperformed A*h_1 in terms of space (expansions, goal tests, new nodes). While the levelsum heuristic took the longest it found the optimal solution in the least amount of space.

It's not clear which heuristic is best. It really comes down to the system running the search and whether or not we value memory economy or time economy. Ignore preconditions is the most time efficient while levelsum is the most space/memory efficient.