This project can be separate into the following parts

the first part is plan representation
three problems are defined in the same action schema, but different initial states and goals, concrete actions will be implemented to represent the plan

then i will define the heuristic to accelerate the search

the last part is to apply different search algorithm and compare them according to output

An optimal sequence of actions is :

```
Load(C1, P1, SFO)

Load(C2, P2, JFK)

Fly(P2, JFK, SFO)

Unload(C2, P2, SFO)

Fly(P1, SFO, JFK)

Unload(C1, P1, JFK)
```

Here is a summary of the first problem

search algorithm	Expansions	goal test	time elapsed	new nodes	optimality	plan length
breadth_first_search	43	56	0.037969431999954395	180	yes	6
breadth_first_tree_search	1458	1459	1.0741759059892502	5960	yes	6
depth_first_graph_search	21	22	0.014547475992003456	84	no	20
depth_limited_search	101	271	0.11047132000385318	414	no	50
uniform_cost_search	55	57	0.04334239299350884	224	yes	6
recursive_best_first_search with h_1	4229	4230	3.6329466230090475	17023	yes	6
greedy_best_first_graph_search with h_1	7	9	0.009883937003905885	28	yes	6
astar_search with h_ignore_preconditions	41	43	0.05023951700422913	170	yes	6
astar_search with h_pg_levelsum	52	54	1.443069318003836	217	yes	6

Since the first problem only require a little bit computation, so I run through all algorithms In the above table.
The optimal length is 6, which is reached by several algorithms.As we can see, breadth_first_search will reach the goal with reasonable expansions and relatively short timeframe and the plan length is optimal.it cost similar time for uniform_cost_search to reach the goal. The time cost for depth_first_search and greedy_search are much less than the previous two algorithm, yet it's plan length are too long, making it a less appealing candidate. Some of the algorithms took much longer timeframe to complete the task, for example, recursive_best_first_seach with h_1 took more than three seconds for this simple question.
hense, the uniform_cost_search/breadth_first_search/astar_search with h_ignore_preconditions are among the best candidates.

An optimal sequence of actions is :

```
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)
```

here is the summary of the second problem

search algorithm	Expansions	goal test	time elapsed	new nodes	optimality	plan length
breadth_first_search	3343	4609	18.600013048999244	30509	yes	9
depth_first_graph_search	436	437	2.739565071009565	3862	no	434
uniform_cost_search	4605	4607	15.422637620009482	41839	yes	9
greedy_best_first_graph_search with h_1	465	467	1.5314277130091796	4179	no	23
astar_search with h_ignore_preconditions	1310	1312	4.839183835996664	11979	yes	9
astar_search with h_pg_levelsum	74	76	153.38616869298858	720	yes	9

It took too long time for some algorithms to complete the second problem, here, I'll just list some candidates that completed the task within minutes.
The optimal plan length here is 9, and as we may expected, uniform_cost_search/breadth_first_search/astar_search with h_ignore_preconditions all achieved the goal with optimal steps. astar_search with h_ignore_preconditions is gaining obvious advantage over the the previous two algorithm in terms of time consumed.
greedy_best_first_graph_search with h_1/depth_first_search are still among the fastest algorithms, and once again, the plan took too many steps to complete.
So, the best algorithm here is astar_search with h_ignore_preconditions, which reached the goal with relatively short time and optimal steps.

An optimal sequence of actions is :

```
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)
```

here is the summary of the third problem

search algorithm	Expansions	goal test	time elapsed	new nodes	optimality	plan length
breadth_first_search	14663	18098	146.7369722459989	129631	yes	12
depth_first_graph_search	3767	3768	84.41157794899482	31703	no	3587
uniform_cost_search	16961	16963	64.49515897000674	149117	yes	12
greedy_best_first_graph_search with h_1	3998	4000	15.647271394002018	35002	no	30
astar_search with h_ignore_preconditions	4444	4446	18.554742489999626	39227	yes	12

for the last problem, i applied the algorithms used in previous problem, and the optimal plan length is 12.
astar_search with h_ignore_preconditions outperformed other algorithms in both time it consumed and plan length it reached. without doubt, astar_search with h_ignore_preconditions is our best choice.

Performance Analysis

None heuristics algorithms
BFS(`breadth_first_search`) is guaranteed to find a goal state if one exists. While it is not true for DFS(`depth_first_graph_search`), which keep track of nodes it explored and expands to the deepest nodes first. This explained why BFS(`breadth_first_search`) always returned the optimal solution but takes much higher nodes expansion and why DFS(`depth_first_graph_search`) takes less node expansion but returned non-optimal solution. So, for planning project, BFS is optimal choice in none heuristics algorithms.

Heuristics algorithms
`astar_search with h_ignore_preconditions`

According to Artificial Intelligence A Modern Approach written by Peter Norvig, "It turns out that this strategy is more than just reasonable: provide that the heuristic function h(n) satisfies certain conditions, A * search is both complete and optimal." `astar_search with h_ignore_preconditions` algorithm created a relaxed version of the problem and it saves lots of time compared with DFS and BFS when search space growth larger.
`astar_search with h_pg_levelsum`
So, In our case , if our strategy is to find the optimal search plan, then the best plan is A-star-search with ignore-preconditions heuristic should be optimal. Nevertheless, the expanded search node for`astar_search with h_pg_levelsum` is much less than all the other search methods. But,it need time to build the planing graph, total time is much longer than the previous method. So if we have enough time and also want the optimal solution, we will surely use `astar_search with h_pg_levelsum as our choice.`