Breadth_first_search:

Problem	Expansions	Goal Tests	Time Elapsed	Optimality
1	43	56	0.0328	Yes
2	3343	4609	11.30	Yes
3	14663	18098	87.84	Yes

Depth_first_graph_search:

Problem	Expansions	Goal Tests	Time Elapsed	Optimality
1	21	22	0.0097	No
2	624	625	2.938	No
3	408	409	1.400	No

Uniform_cost_search:

Problem	Expansions	Goal Tests	Time Elapsed	Optimality
1	55	57	0.0239	Yes
2	4853	4855	8.497	Yes
3	18151	18153	38.23	Yes

A*_ignore_precond:

Problem	Expansions	Goal Tests	Time Elapsed	Optimality
1	41	43	0.02751	Yes
2	1450	1452	3.209	Yes
3	5038	5040	13.47	Yes

A*_planning_graph:

Problem	Expansions	Goal Tests	Time Elapsed	Optimality
1	11	13	0.6073	Yes
2	86	88	54.85	Yes
3	313	315	259.2	Yes

Optimal Plans

```
   Problem 1:

       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:

       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)

   Problem 3:

        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)
```

Analysis:
The best heuristic is A* search ignoring preconditions based on optimality and time elapsed. Compared to planning graph, the heuristic ignoring preconditions is easy to compute. In the related relaxed problem, every action is applicable in every state, and any single goal fluent can be achieved in one step. So we simply add up the number of unsatisfied goals to calculate the number of steps required to solve the relaxed problem.

However, for simple problems like problem 1, other methods like uniform cost search may perform better, which is a simpler algorithm but still effective for simple problems. It uses the path cost in the best_first_graph_search which uses a priority queue. Besides, it also makes two improvements upon bread-first search. The first is that the goal test is applied to a node when it is selected for expansion rather than when it is first generated. The second difference is that a test is added in case a better path is found to a node currently on the frontier. These improvements can be seen from the differences in node expansion and goal tests counts.