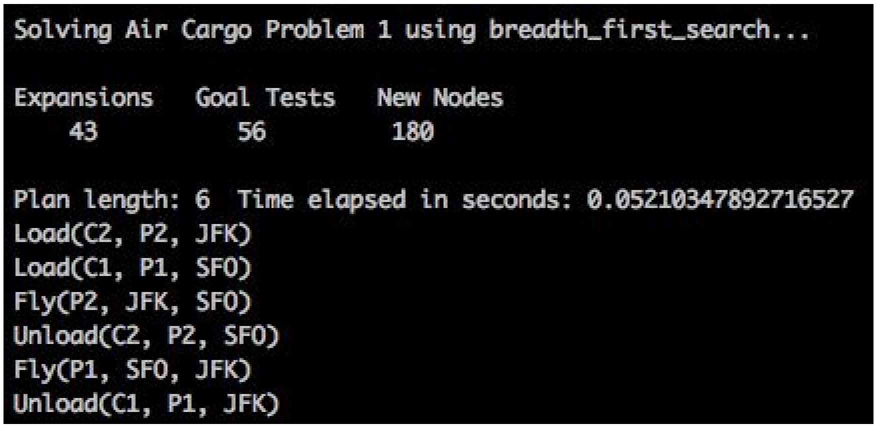
#### TODO: Implement methods and functions in `my_air_cargo_problems.py`
Done.

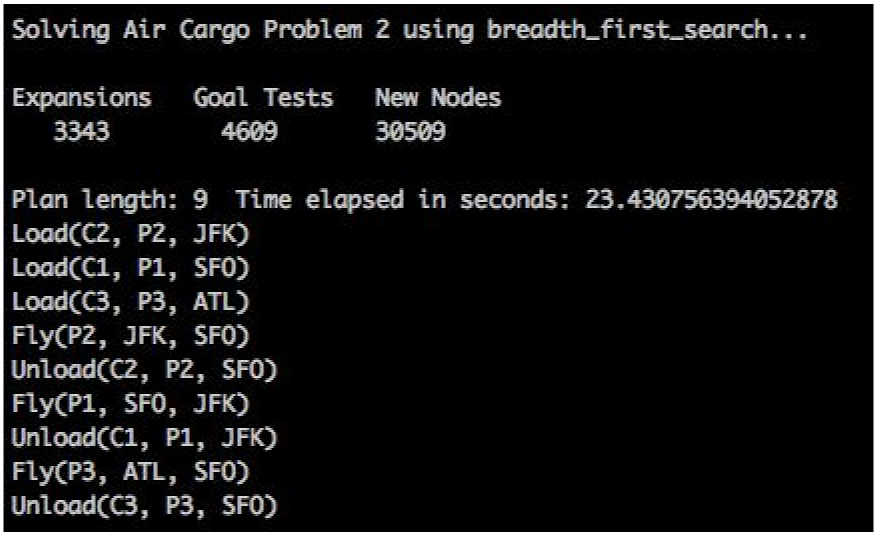
### Part 1 - Planning problems
#### TODO: Experiment and document metrics for non-heuristic planning solution searches
** Run uninformed planning searches for `air_cargo_p1`, `air_cargo_p2`, and `air_cargo_p3`; provide metrics on number of node expansions required, number of goal tests, time elapsed, and optimality of solution for each search algorithm. Include the result of at least three of these searches, including breadth-first and depth-first, in your write-up (`breadth_first_search` and `depth_first_graph_search`).*

breadth_first_search (guarantees optimal solution):
P1:



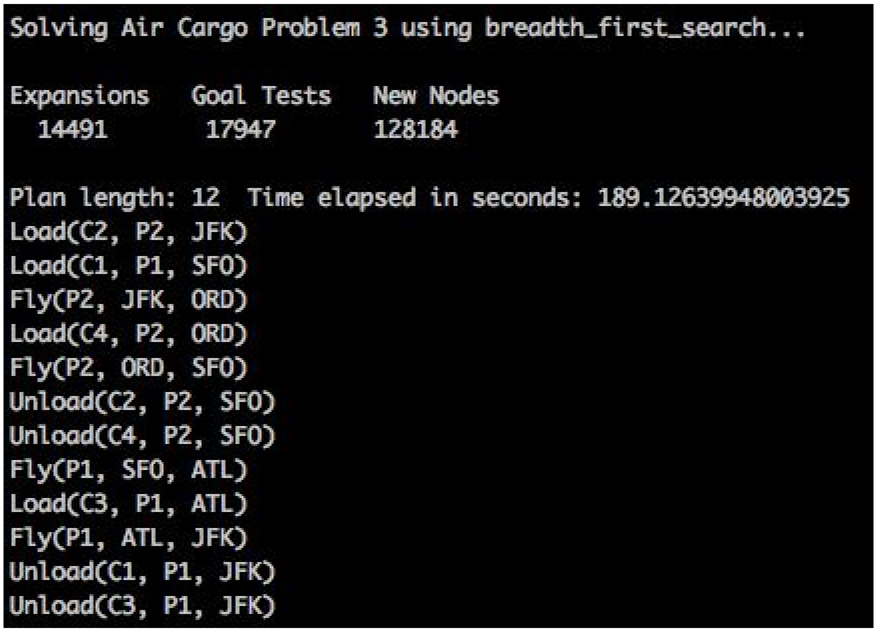
```
Solving Air Cargo Problem 1 using breadth_first_search...

Expansions   Goal Tests   New Nodes
    43          56          180

Plan length: 6  Time elapsed in seconds: 0.05210347892716527
Load(C2, P2, JFK)
Load(C1, P1, SFO)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)
```

P2:



```
Solving Air Cargo Problem 2 using breadth_first_search...

Expansions   Goal Tests   New Nodes
   3343        4609       30509

Plan length: 9  Time elapsed in seconds: 23.430756394052878
Load(C2, P2, JFK)
Load(C1, P1, SFO)
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)
```

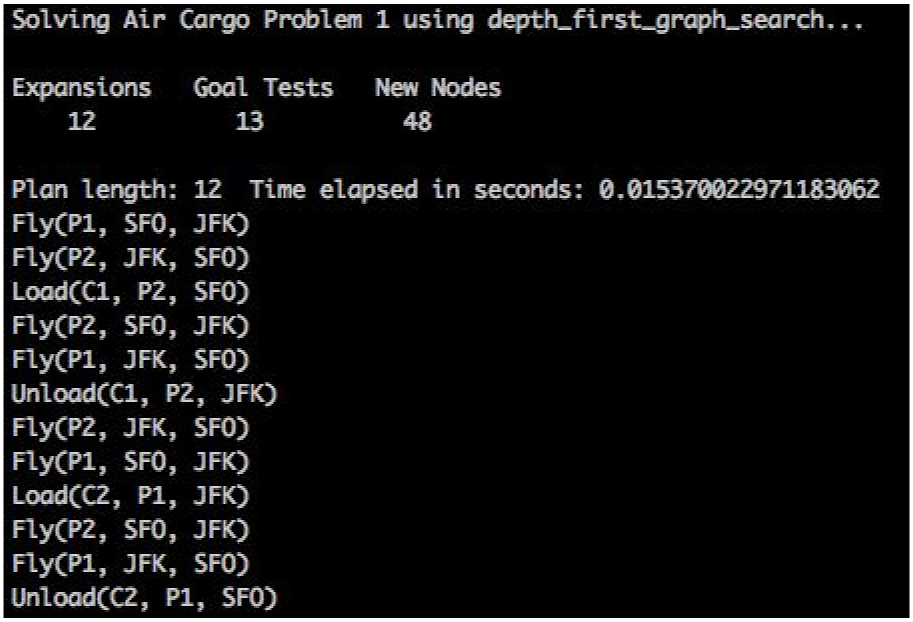
P3:



```
Solving Air Cargo Problem 3 using breadth_first_search...

Expansions   Goal Tests   New Nodes
  14491       17947       128184

Plan length: 12  Time elapsed in seconds: 189.12639948003925
Load(C2, P2, JFK)
Load(C1, P1, SFO)
Fly(P2, JFK, ORD)
Load(C4, P2, ORD)
Fly(P2, ORD, SFO)
Unload(C2, P2, SFO)
Unload(C4, P2, SFO)
Fly(P1, SFO, ATL)
Load(C3, P1, ATL)
Fly(P1, ATL, JFK)
Unload(C1, P1, JFK)
Unload(C3, P1, JFK)
```

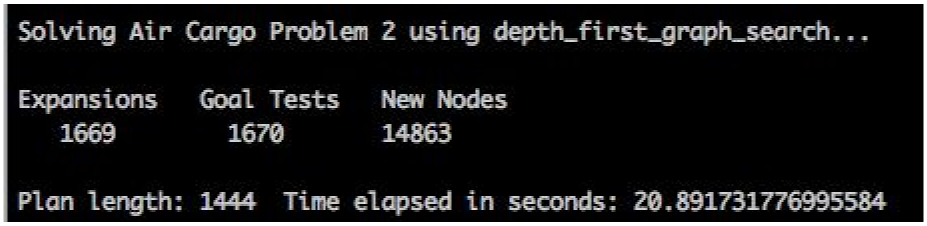
Depth_first_graph_search (Does not guarantee optimal solution)

P1:



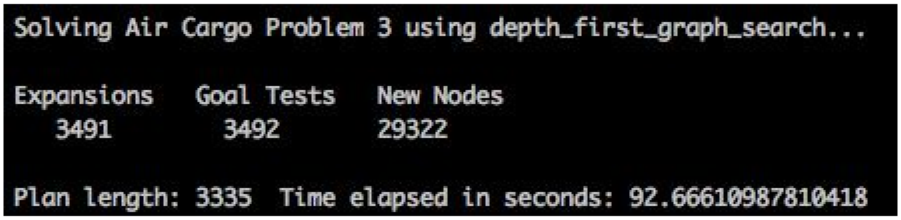
```
Solving Air Cargo Problem 1 using depth_first_graph_search...

Expansions   Goal Tests   New Nodes
    12          13          48

Plan length: 12  Time elapsed in seconds: 0.015370022971183062
Fly(P1, SFO, JFK)
Fly(P2, JFK, SFO)
Load(C1, P2, SFO)
Fly(P2, SFO, JFK)
Fly(P1, JFK, SFO)
Unload(C1, P2, JFK)
Fly(P2, JFK, SFO)
Fly(P1, SFO, JFK)
Load(C2, P1, JFK)
Fly(P2, SFO, JFK)
Fly(P1, JFK, SFO)
Unload(C2, P1, SFO)
```

P2:



```
Solving Air Cargo Problem 2 using depth_first_graph_search...

Expansions   Goal Tests   New Nodes
   1669         1670        14863

Plan length: 1444  Time elapsed in seconds: 20.891731776995584
```

Plan length 1444!
P3:

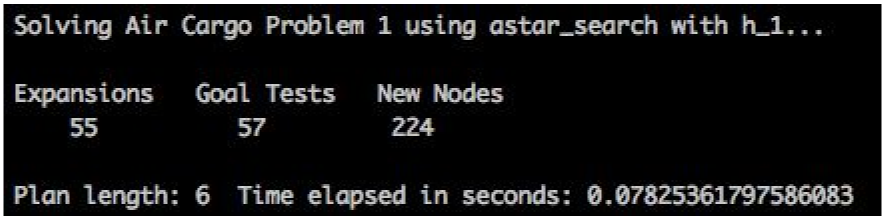


```
Solving Air Cargo Problem 3 using depth_first_graph_search...

Expansions   Goal Tests   New Nodes
   3491         3492        29322

Plan length: 3335  Time elapsed in seconds: 92.66610987810418
```

Interesting, the first time I ran P3 on depth_first_graph_search it took around 3 hours to find a solution. This time it has only taken minute and a half. I suppose it depends on which of the branches decides to dig into, until finding a solution.

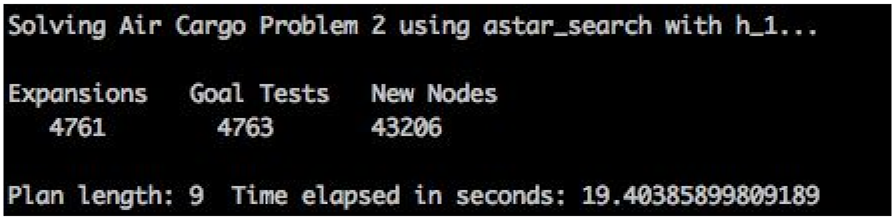
### Part 2 - Domain-independent heuristics
#### TODO: Experiment and document: metrics of A* searches with these heuristics

astar_search h_1
P1:



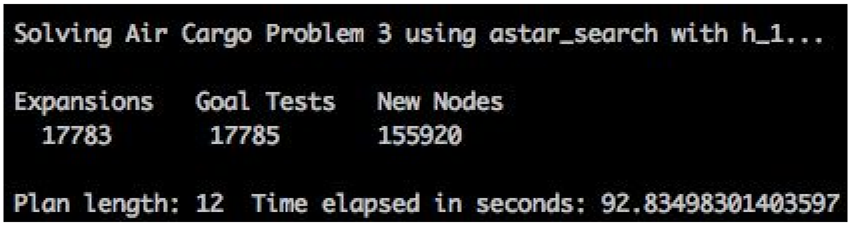
```
Solving Air Cargo Problem 1 using astar_search with h_1...

Expansions   Goal Tests   New Nodes
    55           57          224

Plan length: 6  Time elapsed in seconds: 0.07825361797586083
```

P2:



```
Solving Air Cargo Problem 2 using astar_search with h_1...

Expansions   Goal Tests   New Nodes
   4761         4763        43206

Plan length: 9  Time elapsed in seconds: 19.40385899809189
```

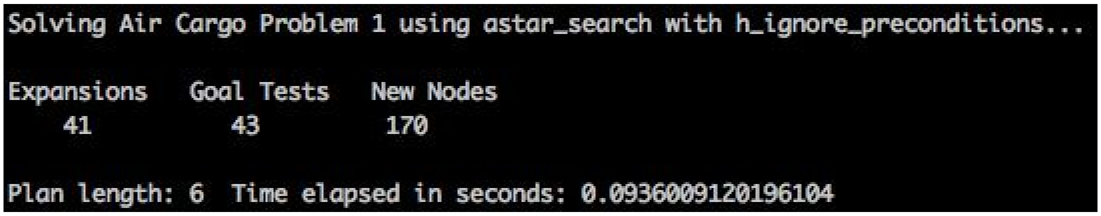
P3:



```
Solving Air Cargo Problem 3 using astar_search with h_1...

Expansions   Goal Tests   New Nodes
  17783        17785       155920

Plan length: 12  Time elapsed in seconds: 92.83498301403597
```

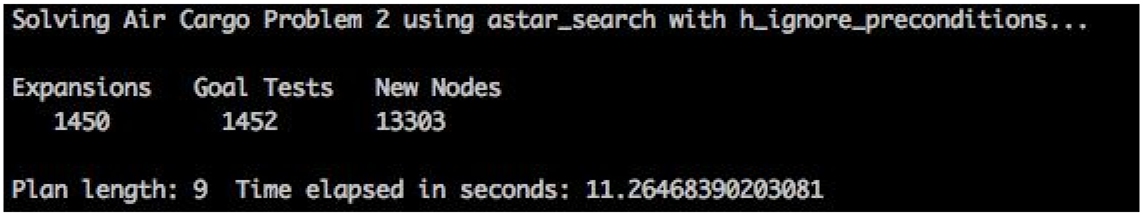
astar_search h_ignore_preconditions

P1:



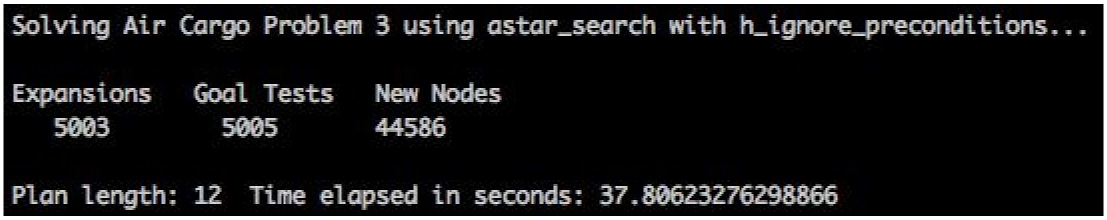
```
Solving Air Cargo Problem 1 using astar_search with h_ignore_preconditions...

Expansions   Goal Tests   New Nodes
    41          43         170

Plan length: 6  Time elapsed in seconds: 0.0936009120196104
```

P2:



```
Solving Air Cargo Problem 2 using astar_search with h_ignore_preconditions...

Expansions   Goal Tests   New Nodes
   1450        1452       13303

Plan length: 9  Time elapsed in seconds: 11.26468390203081
```

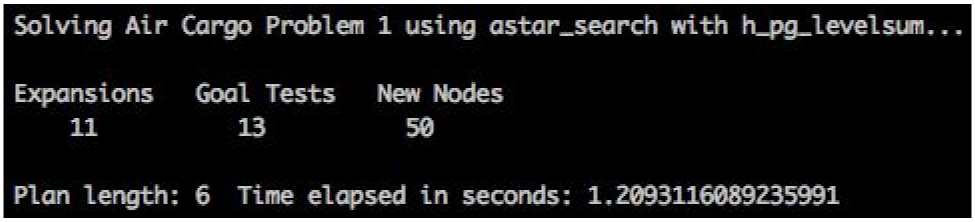
P3:



```
Solving Air Cargo Problem 3 using astar_search with h_ignore_preconditions...

Expansions   Goal Tests   New Nodes
   5003        5005       44586

Plan length: 12  Time elapsed in seconds: 37.80623276298866
```

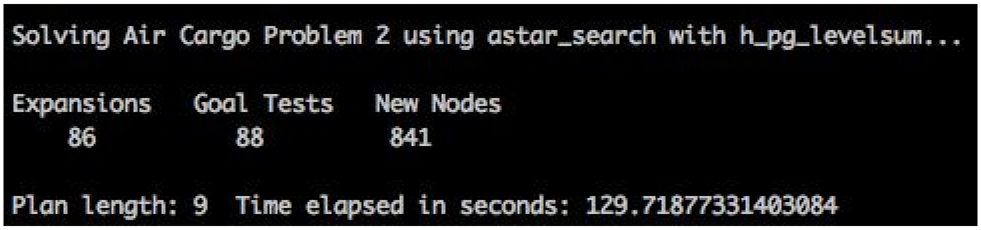
astar_search h_pg_levelsum

P1:



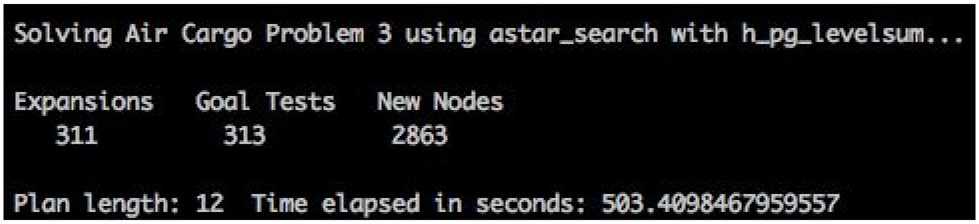
```
Solving Air Cargo Problem 1 using astar_search with h_pg_levelsum...

Expansions   Goal Tests   New Nodes
    11          13          50

Plan length: 6  Time elapsed in seconds: 1.2093116089235991
```

P2:



```
Solving Air Cargo Problem 2 using astar_search with h_pg_levelsum...

Expansions   Goal Tests   New Nodes
    86          88         841

Plan length: 9  Time elapsed in seconds: 129.71877331403084
```

P3:

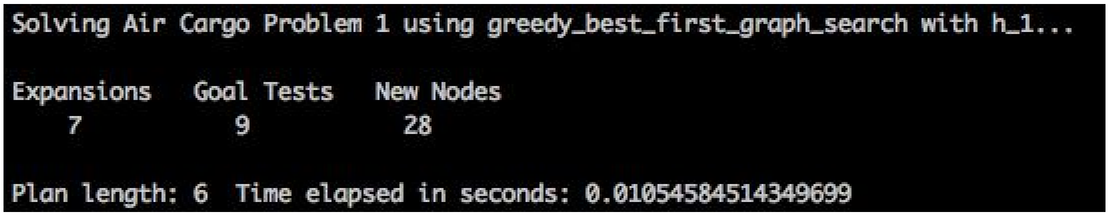


```
Solving Air Cargo Problem 3 using astar_search with h_pg_levelsum...

Expansions   Goal Tests   New Nodes
   311          313          2863

Plan length: 12  Time elapsed in seconds: 503.4098467959557
```

### Part 3: Written Analysis
#### TODO: Include the following in your written analysis.
- Provide an optimal plan for Problems 1, 2, and 3.
breadth_first_search always guarantees an optimal solution.
We see the steps required for each of the problems are 3/6/12.
Solutions provided in Part1

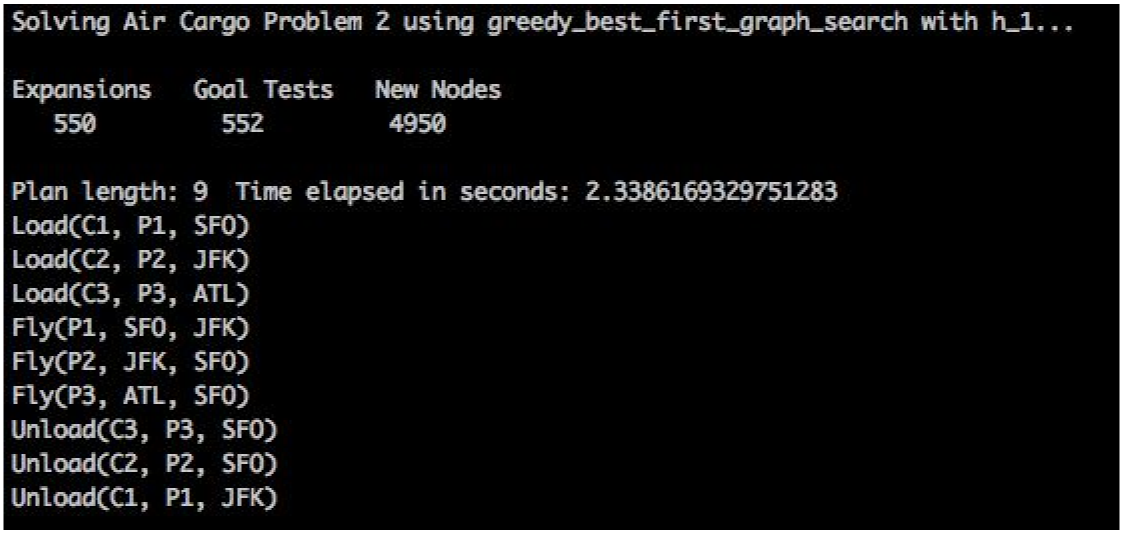
- Compare and contrast non-heuristic search result metrics (optimality, time elapsed, number of node expansions) for Problems 1,2, and 3. Include breadth-first, depth-first, and at least one other uninformed non-heuristic search in your comparison; Your third choice of non-heuristic search may be skipped for Problem 3 if it takes longer than 10 minutes to run, but a note in this case should be included.

Non-heuristic:greedy_best_first_graph_search
P1:



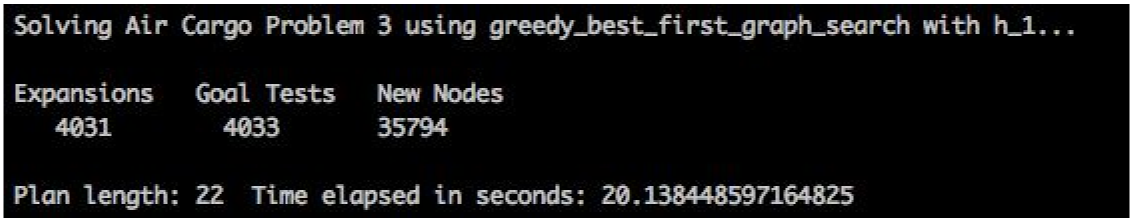
```
Solving Air Cargo Problem 1 using greedy_best_first_graph_search with h_1...

Expansions   Goal Tests   New Nodes
    7            9           28

Plan length: 6  Time elapsed in seconds: 0.010545845143496990
```

P2



```
Solving Air Cargo Problem 2 using greedy_best_first_graph_search with h_1...

Expansions   Goal Tests   New Nodes
   550          552         4950

Plan length: 9  Time elapsed in seconds: 2.338616932975128З
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Load(C3, P3, ATL)
Fly(P1, SFO, JFK)
Fly(P2, JFK, SFO)
Fly(P3, ATL, SFO)
Unload(C3, P3, SFO)
Unload(C2, P2, SFO)
Unload(C1, P1, JFK)
```

P3:



```
Solving Air Cargo Problem 3 using greedy_best_first_graph_search with h_1...

Expansions   Goal Tests   New Nodes
   4031         4033       35794

Plan length: 22  Time elapsed in seconds: 20.138448597164825
```

Breadth_first_search: optimal finding the solution, expands more nodes than the other two
Depth_first_graph_search: non-optimal solutions, fast, expands the least nodes of all, same times as before.
Greedy_best_first_graph_search: optimal finding the solution for P1 and P2 but not for P3 (22 vs 12), expands similar nodes than depth_first, and less than breadth_first, fast, specially for P3.

- Compare and contrast heuristic search result metrics using A* with the "ignore preconditions" and "level-sum" heuristics for Problems 1, 2, and 3.
astar_search h_1: optimal finding the solution, expands more nodes than the other two, and is second in speed.
astar_search h_ignore_preconditions: optimal finding the solution, second in expanding nodes, fastest of the three.
astar_search h_pg_levelsum optimal finding the solution, winner in least expanding nodes, slowest.

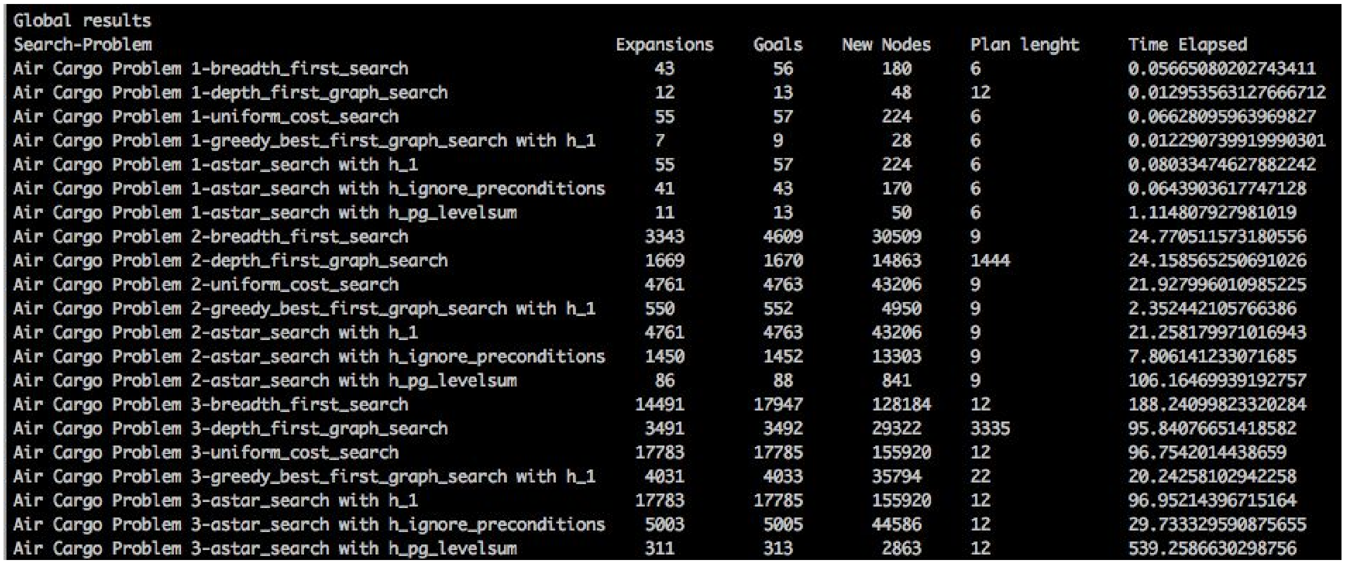
- What was the best heuristic used in these problems? Was it better than non-heuristic search planning methods for all problems? Why or why not?
I think the best heuristic is "**astar_search h_ignore_preconditions"** since it is fast and does not expand too many nodes. If speed was not a concern, I'd use **astar_search h_pg_levelsum**

although it might be the case that the complexity of the planning graph or its implementation plays a big role in it being really slow, in terms of nodes expanded is the best of all A*.

There is one non-heuristic search that is superior to ignore_preconditions, that is **3-greedy_best_first_graph_search with h_1**

After this comparative and having checked all possible search functions for P2, I think **astar_search h_pg_levelsum** demonstrates that using heuristics helps reducing the number of nodes explored for all problems; and although it might have a penalty on computational cost, if the heuristic is admissible and consistent, we’ll have optimal solutions too. In most of the cases, since there are fewer nodes expanded, this means that the search is shorter, so faster to find a solution (except in the already commented case of PG).

Comparative chart:
NOTE:
'breadth_first_tree_search'
'depth_limited_search’
'recursive_best_first_search'
Are not present, since even for P2 problems they were taking longer than 1 hour to complete.



```
Global results
Search-Problem                                                Expansions   Goals   New Nodes    Plan lenght      Time Elapsed
Air Cargo Problem 1-breadth_first_search                           43        56         180     6                0.05665080202743411
Air Cargo Problem 1-depth_first_graph_search                       12        13          48     12               0.012953563127666712
Air Cargo Problem 1-uniform_cost_search                            55        57         224     6                0.06628095963969827
Air Cargo Problem 1-greedy_best_first_graph_search with h_1         7         9          28     6                0.012290739919990301
Air Cargo Problem 1-astar_search with h_1                          55        57         224     6                0.08033474627882242
Air Cargo Problem 1-astar_search with h_ignore_preconditions       41        43         170     6                0.0643903617747128
Air Cargo Problem 1-astar_search with h_pg_levelsum                11        13          50     6                1.114807927981019
Air Cargo Problem 2-breadth_first_search                         3343      4609       30509     9                24.770511573180556
Air Cargo Problem 2-depth_first_graph_search                     1669      1670       14863     1444             24.158565250691026
Air Cargo Problem 2-uniform_cost_search                          4761      4763       43206     9                21.927996010985225
Air Cargo Problem 2-greedy_best_first_graph_search with h_1       550       552        4950     9                2.352442105766386
Air Cargo Problem 2-astar_search with h_1                        4761      4763       43206     9                21.258179971016943
Air Cargo Problem 2-astar_search with h_ignore_preconditions     1450      1452       13303     9                7.806141233071685
Air Cargo Problem 2-astar_search with h_pg_levelsum                86        88         841     9                106.16469939192757
Air Cargo Problem 3-breadth_first_search                        14491     17947      128184     12               188.24099823320284
Air Cargo Problem 3-depth_first_graph_search                     3491      3492       29322     3335             95.84076651418582
Air Cargo Problem 3-uniform_cost_search                         17783     17785      155920     12               96.7542014438659
Air Cargo Problem 3-greedy_best_first_graph_search with h_1      4031      4033       35794     22               20.24258102942258
Air Cargo Problem 3-astar_search with h_1                       17783     17785      155920     12               96.95214396715164
Air Cargo Problem 3-astar_search with h_ignore_preconditions     5003      5005       44586     12               29.733329590875655
Air Cargo Problem 3-astar_search with h_pg_levelsum               311       313        2863     12               539.2586630298756
```