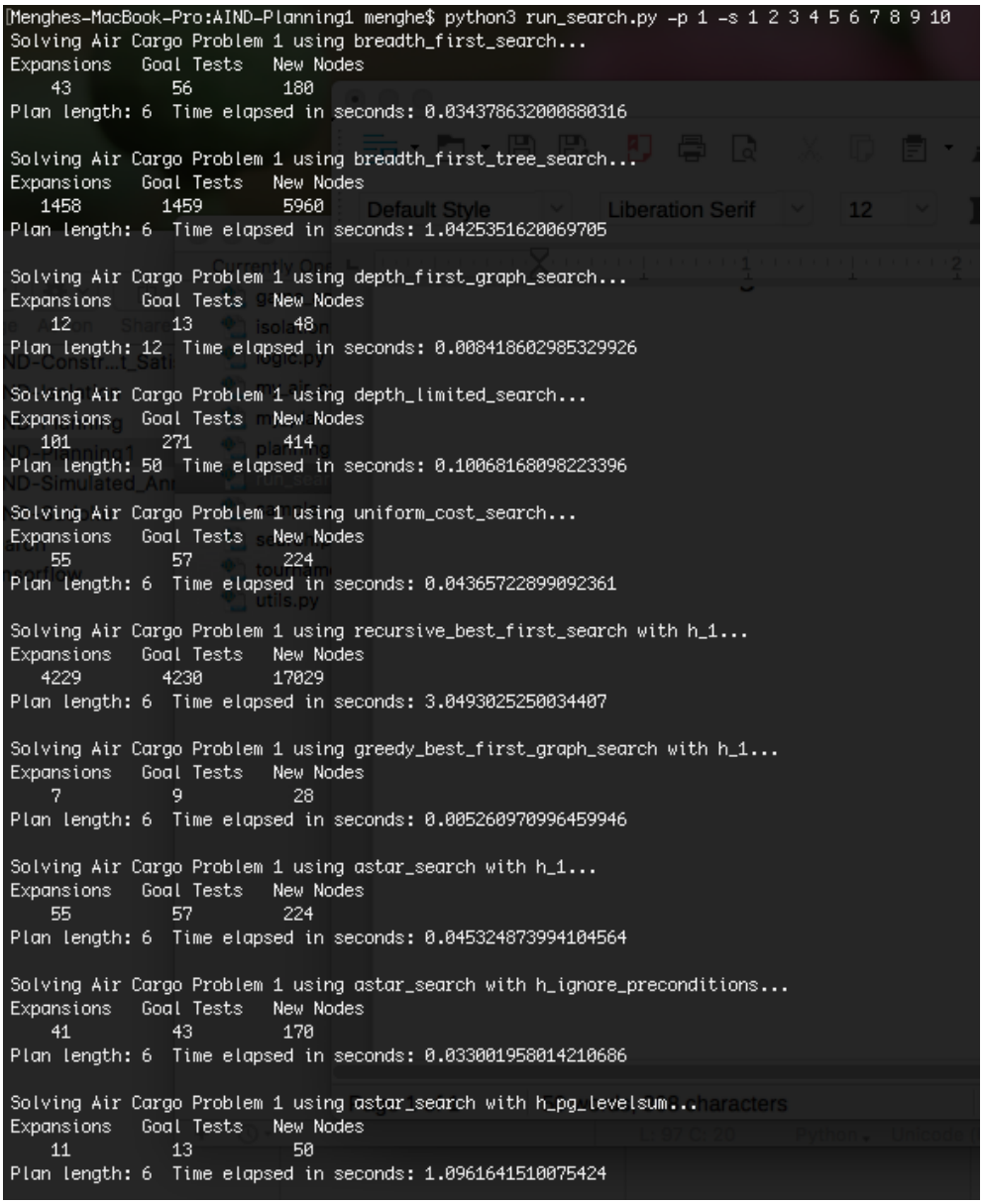
Planning Search Heuristic Analysis

by Menghe Lu

In this short study, we build ten different search algorithms to resolve three air cargo problems. We build planning graph to resolve these three problems. We compare the speed, memory usage for each search algorithm and choose the good balance for each problem.

Problem1: we can run 10 algorithms for the first problem at same time by 'python run_search.py -p 1 -s 1 2 3 4 5 6 7 8 9 10'



I can see breadth_first_tree_search, recursive_best_first_search with h_1 and h_pg_levelsum take

much more time than other algorithms. Readth_first_search and recurisive_best_first_search with h_1 create much more expansions and new nodes than others. Most of algorithms plan length is 6 except plan length of depth limited search which is 50. The shorter plan length, the better solution. Here is an example of breadth_first_search solution.

```
Menghes-MacBook-Pro:AIND-Planning1 menghe$ python3 run_search.py -p 1 -s 1
Solving Air Cargo Problem 1 using breadth_first_search...
Expansions   Goal Tests   New Nodes
    43          56          180
Plan length: 6  Time elapsed in seconds: 0.04203094099648297
Load(C2, P2, JFK)
Load(C1, P1, SFO)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)
```

Problem 2: Because the 10 algorithms execution time exceeded too much time. I run only algorithm 1, 3, 5, 7, 8, 9 which are faster than others.

```
Menghes-MacBook-Pro:AIND-Planning1 menghe$ python3 run_search.py -p 2 -s 1 3 5 7 8 9
Solving Air Cargo Problem 2 using breadth_first_search...
Expansions   Goal Tests   New Nodes
   3346        4612        30534
Plan length: 9  Time elapsed in seconds: 17.35957458501798

Solving Air Cargo Problem 2 using depth_first_graph_search...
Expansions   Goal Tests   New Nodes
   1124        1125        10017
Plan length: 1085  Time elapsed in seconds: 11.177870312996674

Solving Air Cargo Problem 2 using uniform_cost_search...
Expansions   Goal Tests   New Nodes
   4853        4855        44041
Plan length: 9  Time elapsed in seconds: 14.76348240999505

Solving Air Cargo Problem 2 using greedy_best_first_graph_search with h_1...
Expansions   Goal Tests   New Nodes
   998         1000         8982
Plan length: 21  Time elapsed in seconds: 2.817834326007869

Solving Air Cargo Problem 2 using astar_search with h_1...
Expansions   Goal Tests   New Nodes
   4853        4855        44041
Plan length: 9  Time elapsed in seconds: 16.19841562100919

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: 5.265251395991072
```

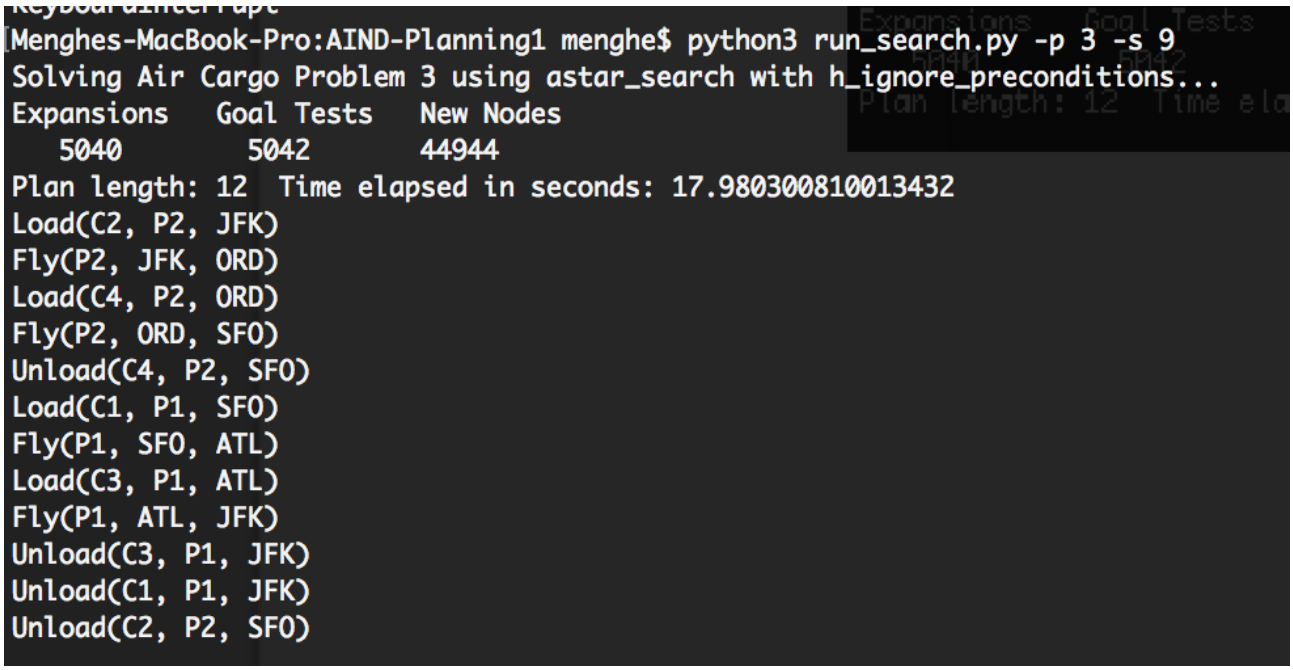
Most of algorithms plan length are 9, but depth_first_grahp_search plan length is 1085. astar_search h_ignore_preconditions takes less time and less plan length than others.
Here is the astar_search h_ignore_preconditions solution for problem 2: python run_search.py -p 2 -s 9

```
Menghes-MacBook-Pro:AIND-Planning1 menghe$ python3 run_search.py -p 2 -s 9
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: 4.110680384001171
Load(C3, P3, ATL)
Fly(P3, ATL, SFO)
Unload(C3, P3, SFO)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
Load(C1, P1, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)
```

Problem 3: We can see problem 3 takes much more time than problem 1 and 2. It creates much more expansions and new nodes because there are four cargos, two planes and four airports.

```
Menghes-MacBook-Pro:AIND-Planning1 menghe$ python3 run_search.py -p 3 -s 1 3 5 7 8 9
Solving Air Cargo Problem 3 using breadth_first_search...
Expansions   Goal Tests   New Nodes
  14663       18098       129631
Plan length: 12  Time elapsed in seconds: 130.6085662410187

Solving Air Cargo Problem 3 using depth_first_graph_search...
Expansions   Goal Tests   New Nodes
  627         628         5176
Plan length: 596  Time elapsed in seconds: 3.792981284990674

Solving Air Cargo Problem 3 using uniform_cost_search...
Expansions   Goal Tests   New Nodes
  18235       18237       159716
Plan length: 12  Time elapsed in seconds: 61.491733203991316

Solving Air Cargo Problem 3 using greedy_best_first_graph_search with h_1...
Expansions   Goal Tests   New Nodes
  5614        5616        49429
Plan length: 22  Time elapsed in seconds: 19.063247326004785

Solving Air Cargo Problem 3 using astar_search with h_1...
Expansions   Goal Tests   New Nodes
  18235       18237       159716
Plan length: 12  Time elapsed in seconds: 63.17349347699201

Solving Air Cargo Problem 3 using astar_search with h_ignore_preconditions...
Expansions   Goal Tests   New Nodes
   5040        5042       44944
Plan length: 12  Time elapsed in seconds: 17.808496475976426
```

I find astar_search with h_ignore_preconditions works better than others when the problem is more complicate because it takes less time to resolve problem correctly. Here is the solution for aster_search with h_ignore_preconditions.



Compare the performance of the algorithms:
depth_first_grahp_search runs faster than breadth_first_search and uniform_cost_search, but it has a longer plan length than the other two. Because depth_first_graph_search starts at the root and explores as far as possible along each branch before backtracking. According to problem 1, 2 and 3 results, we see depth_first_graph_search has much longer plan length than breath_first_search and uniform_cost_search when the problem is more complicated even it takes less time. It means that depth_first_graph_search doesn't give the optimal plan.

breadth_first_search has less plan length than depth_first_graph, but it takes much more time than depth_first_grahp_search and unifrom_cost_search, especially when the problem is more complicated. Because breadth_first_search starts at the tree root and explores the neighbor nodes. first before moving to the next level neighbors.

uniform_cost_search works better than depth_first_graph_search and breadth_first_search considering both running time and plan length. It searches the nodes with the lowest node path cost score first.

Among all ten algorithms, I find aster_search with h_ignore_preconditions runs better than other algorithms according to both running time and plan length results. Because this heuristic estimates the minimum number of actions that must be carried out from the current state in order to satisfy all of the goal conditions by ignoring the preconditions required for an action to be executed.