	Breadth first search	Breadth first tree search	Depth first graph search	Depth limited search	Uniform cost search	Recursive best first search	Greedy best first graph search	Astar, h1	Astar, h_ignore preconditions	Astar, h_pg_levelsum
Air cargo 1	**Expansions**: 43 **Goal Tests**: 56 **New Nodes**: 180 **Plan length**: 6 **Time (s)**: 0.13	**Expansions**: 1458 **Goal Tests**: 1459 **New Nodes**: 5960 **Plan length**: 6 **Time (s)**: 4.588	**Expansions**: 21 **Goal Tests**: 22 **New Nodes**: 84 **Plan length**: 20 **Time (s)**: 0.066	**Expansions**: 101 **Goal Tests**: 271 **New Nodes**: 414 **Plan length**: 50 **Time (s)**: 0.337	**Expansions**: 55 **Goal Tests**: 57 **New Nodes**: 224 **Plan length**: 6 **Time (s)**: 0.171	**Expansions**: 4229 **Goal Tests**: 4230 **New Nodes**: 17023 **Plan length**: 6 **Time (s)**: 13.467	**Expansions**: 7 **Goal Tests**: 9 **New Nodes**: 28 **Plan length**: 6 **Time (s)**: 0.022	**Expansions**: 55 **Goal Tests**: 57 **New Nodes**: 224 **Plan length**: 6 **Time (s)**: 0.167	**Expansions**: 41 **Goal Tests**: 43 **New Nodes**: 170 **Plan length**: 6 **Time (s)**: 0.205	**Expansions**: 11 **Goal Tests**: 13 **New Nodes**: 50 **Plan length**: 6 **Time (s)**: 1.71
Air cargo 2	**Expansions**: 3343 **Goal Tests**: 4609 **New Nodes**: 30509 **Plan length**: 9 **Time (s)**: 58.540	TOO LONG	**Expansions**: 624 **Goal Tests**: 625 **New Nodes**: 5602 **Plan length**: 619 **Time (s)**: 10.911	TOO LONG	**Expansions**: 4853 **Goal Tests**: 4855 **New Nodes**: 44041 **Plan length**: 9 **Time (s)**: 99.958	TOO LONG	**Expansions**: 970 **Goal Tests**: 972 **New Nodes**: 8726 **Plan length**: 17 **Time (s)**: 20.185	**Expansions**: 4853 **Goal Tests**: 4855 **New Nodes**: 44041 **Plan length**: 9 **Time (s)**: 100.323	**Expansions**: 1506 **Goal Tests**: 1508 **New Nodes**: 13820 **Plan length**: 9 **Time (s)**: 42.329	**Expansions**: 86 **Goal Tests**: 88 **New Nodes**: 841 **Plan length**: 9 **Time (s)**: 171.22
Air cargo 3	**Expansions**: 14663 **Goal Tests**: 18098 **NNodes**: 129631 **Plan length**: 12 **Time (s)**: 334.22	TOO LONG	**Expansions**: 408 **Goal Tests**: 409 **New Nodes**: 3364 **Plan length**: 392 **Time (s)**: 7.733	TOO LONG	TOO LONG	TOO LONG	**Expansions**: 5578 **Goal Tests**: 5580 **New Nodes**: 49150 **Plan length**: 22 **Time (s)**: 189.65	**Expansions**: 18223 **Goal Tests**: 18225 **NNodes**: 159618 **Plan length**: 12 **Time (s)**: 670.712	**Expansions**: 5118 **Goal Tests**: 5120 **NNodes**: 45650 **Plan length**: 12 **Time (s)**: 227.437	**Expansions**: 403 **Goal Tests**: 405 **NNodes**: 3708 **Plan length**: 12 **Time (s)**: 1204.23

Table 1: Result of running each algorithm on each 3 problems

Optimal Plan

	Problem 1	Problem 2	Problem 3
Initial State	At(C1, SFO) At(C2, JFK) At(P1, SFO) At(P2, JFK)	At(C1, SFO) At(C2, JFK) At(C3, ATL) At(P1, SFO) At(P2, JFK) At(P3, ATL)	At(C1, SFO) At(C2, JFK) At(C3, ATL) At(C4, ORD) At(P1, SFO) At(P2, JFK)
List of actions	Load(C1, P1, SFO) Load(C2, P2, JFK) Fly(P1, SFO, JFK) Fly(P2, JFK, SFO) Unload(C1, P1, JFK) Unload(C2, P2, SFO)	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)	Load(C2, P2, JFK) Fly(P2, JFK, ORD) Load(C4, P2, ORD) Fly(P2, ORD, SFO) Unload(C4, P2, SFO) Load(C1, P1, SFO) Fly(P1, SFO, ATL) Load(C3, P1, ATL) Fly(P1, ATL, JFK) Unload(C3, P1, JFK) Unload(C2, P2, SFO) Unload(C1, P1, JFK)
Final State	At(C1, JFK) At(C2, SFO)	At(C1, JFK) At(C2, SFO) At(C3, SFO)	At(C1, JFK) At(C3, JFK) At(C2, SFO) At(C4, SFO)

Table 2: Optimal Plan for each problem

Comparison

	Problem 1	Problem 2	Problem 3
Breadth first search	**Expansions**: 43 **Goal Tests**: 56 **New Nodes**: 180 **Plan length**: 6 **Time (s)**: 0.13	**Expansions**: 3343 **Goal Tests**: 4609 **New Nodes**: 30509 **Plan length**: 9 **Time (s)**: 58.540	**Expansions**: 14663 **Goal Tests**: 18098 **New Nodes**: 129631 **Plan length**: 12 **Time (s)**: 334.22
	Breadth first search finds the optimal path to the goal (minimum number of actions) but that has a cost. Actually it had to expanded more and more node as the problem need to achieve more and more goals, hence the time needed to find the optimal solution is linear in term of number of Goal Test and the number of goal to test is exponential in term of number of depth of the search. Hence even though the breadth first search algorithm finds the best solution it cannot be used when the problem becomes to complex (because it takes to long to way the way to the goal)		
Depth first graph search	**Expansions**: 21 **Goal Tests**: 22 **New Nodes**: 84 **Plan length**: 20 **Time (s)**: 0.066	**Expansions**: 624 **Goal Tests**: 625 **New Nodes**: 5602 **Plan length**: 619 **Time (s)**: 10.911	**Expansions**: 408 **Goal Tests**: 409 **New Nodes**: 3364 **Plan length**: 392 **Time (s)**: 7.733
	As seen in lectures, Depth first search is not guarantee to find the best solution to the problem (we see that for problem 1, 2, 3 here the path is respectively 20, 619, 392 while the best path are 6, 9, 12), but it can resolve the problem faster than using Breadth first search. The thing is : Do we really want to use 619 actions to transport 2 cargo from to places to 2 others places. Well absolutely not ! So, even though Depth first graph search seems promising because it solves the problem quite quickly, the solution we got at the end is not optimal at all and hence we should avoid using such solutions.		
Greedy best first graph search	**Expansions**: 7 **Goal Tests**: 9 **New Nodes**: 28 **Plan length**: 6 **Time (s)**: 0.022	**Expansions**: 970 **Goal Tests**: 972 **New Nodes**: 8726 **Plan length**: 17 **Time (s)**: 20.185	**Expansions**: 5578 **Goal Tests**: 5580 **New Nodes**: 49150 **Plan length**: 22 **Time (s)**: 189.65
	Greedy best first graph search is kind of a «good not optimal » solution. Actually it allows to find a relatively good solution in term of length of the plan while it takes less time then Breadth First search. Hence if we want to have a relatively good trade-off between having a reasonably good solution and don't spend too much computational time greedy best first graph search could be a good compromise.		

Table 3: Comparison and Analysis of non heuristic algorithms

Comparison for A* heuristic

	Problem 1	Problem 2	Problem 3
Astar, h_ignore preconditions	**Expansions**: 41 **Goal Tests**: 43 **New Nodes**: 170 **Plan length**: 6 **Time (s)**: 0.205	**Expansions**: 1506 **Goal Tests**: 1508 **New Nodes**: 13820 **Plan length**: 9 **Time (s)**: 42.329	**Expansions**: 5118 **Goal Tests**: 5120 **New Nodes**: 45650 **Plan length**: 12 **Time (s)**: 227.437
	As we can see A* with the h_ignore preconditions give us optimal solutions (we use A* with admissible heuristic so it's normal). As explained we could have thought, this heuristic is good but need to explore lots of node in comparison to the levelsum heuristic. Why's that ? Because we used a relaxed conditions (we erased the preconditions) and thus in this newly graph the solution is always easier to find than in the real graph as we can execute more action for the same state. Hence this heuristic is admissible and seems to be good. Yet the A* algorithm needs to visit lots of nodes to find the optimal solution as we can see with the figures above.		
Astar, h_pg_levelsum	**Expansions**: 11 **Goal Tests**: 13 **New Nodes**: 50 **Plan length**: 6 **Time (s)**: 1.71	**Expansions**: 86 **Goal Tests**: 88 **New Nodes**: 841 **Plan length**: 9 **Time (s)**: 171.22	**Expansions**: 403 **Goal Tests**: 405 **New Nodes**: 3708 **Plan length**: 12 **Time (s)**: 1204.23
	The A* algorithm with the levelsum give us also the optimal solutions. In this case we are using a planning graph and we are computing the sum of the level where each goal is reach. Once again this heuristic is admissible in the sense that we need to attain all goals in the goal state for the same level. This heuristic seems a lot better in a sense that we don't need to expand lots of nodes in comparison the the h_ignore_preconditions heuristic. Yet I takes longer to compute because at each frontier expansion we need to compute the levelsum heuristic which needs to create a new planning graph. Hence, this heuristic is good in a sense that it doesn't visit too many nodes (use less memory) but is bad in term of time.		

Table 4: Comparison and Analysis of heuristic search

Best heuristic and best non heuristic

I consider to compare the heuristic and non heuristic algorithms with respect to 2 factors :

- Find a optimal solution
- Less time to find the best solution

This is purely arbitrary. We might a choose other factors such as the number of nodes expanded (memory used), or is the solution a good approximate solution (doesn't expand much more nodes than the optimal one). Here I choose The two factors above for two reasons :

- We can find the optimal solution to the problem without fear of using to much memory
- We won't use non optimal solution as it cost a lot here (gasoline, time of travel, human resource and so on...)

	Problem 1	**Problem 2**	**Problem 3**
Best Heuristic (in term of time spend)	Astar, h_ignore preconditions	Astar, h_ignore preconditions	Astar, h_ignore preconditions
	Expansions: 41 **Goal Tests**: 43 **New Nodes**: 170 **Plan length**: 6 **Time (s)**: 0.205	**Expansions**: 1506 **Goal Tests**: 1508 **New Nodes**: 13820 **Plan length**: 9 **Time (s)**: 42.329	**Expansions**: 5118 **Goal Tests**: 5120 **New Nodes**: 45650 **Plan length**: 12 **Time (s)**: 227.437
	So here the **A* with the h_ignore preconditions** algorithm gives use the optimal solution in the least amount of time. In this sens I consider it to be the best heuristic to use for those problems.		
Best Non heuristic (in term of time spend)	Greedy best first graph search	Breadth first search	Breadth first search
	Expansions: 7 **Goal Tests**: 9 **New Nodes**: 28 **Plan length**: 6 **Time (s)**: 0.022	**Expansions**: 3343 **Goal Tests**: 4609 **New Nodes**: 30509 **Plan length**: 9 **Time (s)**: 58.540	**Expansions**: 14663 **Goal Tests**: 18098 **New Nodes**: 129631 **Plan length**: 12 **Time (s)**: 334.22
	Here Breadth first search gives use the best solution in the least amount of time. There is one exception. Greedy best first graph search is faster for the problem 1 ? Why's that ? Because the problem 1 is really really easy and so Greedy best first search choose the best solution very quickly. We can note that Breadth first search finds the best solutions in 0.13s for the first problem so we can consider that at the end of the day the best non heuristic algorithm with respect to the factors I choose to focus on is **Breadth first search**		

Table 5: Comparison between best heuristic and non heuristic search algorithms

Note : We can see that greedy best first graph search give us "relatively" good solution quite quickly. I didn't choose this heuristic because between 22 actions and 12 actions there is still a real gap of money to invest (assume we do the 22 actions greedy best first graph search gives us in real world)

At the end of the day using **A* with heuristic** is better than using non heuristic algorithm. It can be easily understand as heuristics can be seen as educated guesses. The thing is then to come up with good heuristics.