Planning search

For this project, we were asked to implemented some parts of a planning search algorithm in order to solve three logistics problems, specifically air cargo problems.

The problem is addressed in a informed and an uninformed way so that we can compare their performance and optimality.

In the first part of this document we explain the problems and the optimal solutions to those problems, and then we discuss the different methods using in the search.

Problems

The solutions presented are not unique, they might be other optimal solutions that are variations of the ones presented (Some of the elements might be in a distinct order as long as the plan is still valid)

Problem 1

The problem is defined as follows

Init(At(C1, SFO) ∧ At(C2, JFK)
∧ At(P1, SFO) ∧ At(P2, JFK)
∧ Cargo(C1) ∧ Cargo(C2)
∧ Plane(P1) ∧ Plane(P2)
∧ Airport(JFK) ∧ Airport(SFO))
Goal(At(C1, JFK) ∧ At(C2, SFO))

And one of the optimal solutions to this problem is a sequence of 6 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)
```

Problem 2

The problem is defined as follows

Init(At(C1, SFO) ∧ At(C2, JFK) ∧ At(C3, ATL)
∧ At(P1, SFO) ∧ At(P2, JFK) ∧ At(P3, ATL)
∧ Cargo(C1) ∧ Cargo(C2) ∧ Cargo(C3)

∧ Plane(P1) ∧ Plane(P2) ∧ Plane(P3)

∧ Airport(JFK) ∧ Airport(SFO) ∧ Airport(ATL))

Goal(At(C1, JFK) ∧ At(C2, SFO) ∧ At(C3, SFO))

And one of the optimal solutions to this problem is a sequence of 9 actions:

```
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)
```

Problem 3

The problem is defined as follows

Init(At(C1, SFO) ∧ At(C2, JFK) ∧ At(C3, ATL) ∧ At(C4, ORD)

∧ At(P1, SFO) ∧ At(P2, JFK)

∧ Cargo(C1) ∧ Cargo(C2) ∧ Cargo(C3) ∧ Cargo(C4)

∧ Plane(P1) ∧ Plane(P2)

∧ Airport(JFK) ∧ Airport(SFO) ∧ Airport(ATL) ∧ Airport(ORD))

Goal(At(C1, JFK) ∧ At(C3, JFK) ∧ At(C2, SFO) ∧ At(C4, SFO))

And one of the optimal solutions to this problem is a sequence of 12 actions:

```
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P1, SFO, ATL)
Load(C3, P1, ATL)
Fly(P2, JFK, ORD)
Load(C4, P2, ORD)
Fly(P1, ATL, JFK)
Fly(P2, ORD, SFO)
Unload(C4, P2, SFO)
Unload(C3, P1, JFK)
Unload(C2, P2, SFO)
Unload(C1, P1, JFK)
```

Search algorithms

The algorithms presented in this section are compared in terms of optimality, speed and number of node expansions.

Uninformed search

The uninformed search algorithms doesn't try to guess which path is more likely to get to the goal (as informed search algorithms do), they just generate new states until one of this states is a goal (or all goals are satisfied in the case of multi-goal problems).

Algorithm	Speed (seconds)	Optimal	Node expansions
Problem 1			
breadth_first_search	0.059	YES	180
breadth_first_tree_search	1.018	YES	1458
depth_first_graph_search	0.053	NO	21
depth_limited_search	0.175	NO	101
uniform_cost_search	0.072	YES	55
recursive_best_first_search_h1	2.703	YES	4229
greedy_best_first_graph_search_h1	0.011	YES	7
Problem 2			
breadth_first_search	26.52	YES	3346
breadth_first_tree_search			
depth_first_graph_search	1.1428	NO	107
depth_limited_search			
uniform_cost_search	22.7452	YES	4761
recursive_best_first_search_h1			
greedy_best_first_graph_search_h1	2.765	YES	550
Problem 3			
breadth_first_search	229.67	YES	14491
breadth_first_tree_search			
depth_first_graph_search	64.44	NO	2099
depth_limited_search			
uniform_cost_search	137.645	YES	17783
recursive_best_first_search_h1			
greedy_best_first_graph_search_h1	38.93	NO	4031
Overall			
breadth_first_search	256.249	YES	18017
breadth_first_tree_search			
depth_first_graph_search	65.6358	FALSE	2227
depth_limited_search			
uniform_cost_search	160.4622	YES	22599
recursive_best_first_search_h1			
greedy_best_first_graph_search_h1	41.706	FALSE	4588

The cells in green represent that that column has the best value on the given problem, the ones in red represent the worst value.

In the overall section, the time and the node expansions of problems one, two and three is added up provided that the algorithm has finished in all cases. An algorithm is considered optimal if it is optimal in all the problems.

Breadth first search and uniform cost search were the only algorithms that yielded an optimal solution given the time limit. In the other hand, despite not getting the optimal solution in all the problems (The solution in the third problem involved 22 action whereas the optimal solution involves 12 actions), greedy best first graph search was the fastest algorithm and it uses less memory than the optimal ones.

The problem of choosing an algorithm depends on the task at hand:

Algorithm	Speed?	Space?	Optimal?
uniform_cost_search	YES	NO	YES
breadth_first_search	NO	YES	YES
greedy_best_first_graph_search_h1	YES	NO	NO
depth_first_graph_search	NO	YES	NO

The columns represent the different options we can consider when choosing an algorithm (whether we consider them important or not).

Informed search

The informed search algorithm try to use heuristic functions to guess which path is more likely to get to the goal. An heuristic function is a way to measure how close we are to a goal.

Algorithm	Speed (seconds)	Optimal	Node expansions
Problem 1			
astar_search_h1	0.069	YES	55
astar_search_h_ignore_preconditions	0.04	YES	41
astar_search_h_pg_levelsum	0.96	YES	11
Problem 2			
astar_search_h1	23.68	YES	4761
astar_search_h_ignore_preconditions	11.3	YES	1450
astar_search_h_pg_levelsum	231.486	YES	86
Problem 3			
astar_search_h1	139.59	YES	17783
astar_search_h_ignore_preconditions	49.91	YES	5003
astar_search_h_pg_levelsum			
Overall			
astar_search_h1	163.339	YES	22599
astar_search_h_ignore_preconditions	61.25	YES	6494
astar_search_h_pg_levelsum			

The cells in green represent that that column has the best value on the given problem, the ones in red represent the worst value.

In the overall section, the time and the node expansions of problems one, two and three is added up provided that the algorithm has finished in all cases. An algorithm is considered optimal if it is optimal in all the problems.

In the case of the informed search algorithm, the problem of choosing an algorithm is simplified because all of them yield optimal solutions and A* with the ignore preconditions heuristic yields the best results in both performance and space usage (considering only algorithms that have been able to complete the problem within the time limits).

Choosing the right is heuristic is a trade-off between computational expense and quality, for example, the level sum heuristic is the best heuristic when it comes to choose the right node to expand, but it is very computational expensive, so it doesn't have to much time to explore.

Informed and uninformed search algorithms comparison

In this section we analyze the performance of the informed versus uninformed algorithms in the three problems.

Algorithm	Speed (seconds)	Optimal	Node expansions
Problem 1			
uniform_cost_search	0.072	YES	55
greedy_best_first_graph_search_h1	0.011	YES	7
astar_search_h_ignore_preconditions	0.04	YES	41
Problem 2			
uniform_cost_search	22.7452	YES	4761
depth_first_graph_search	2.765	YES	550
astar_search_h_ignore_preconditions	11.3	YES	1450
Problem 3			
uniform_cost_search	137.645	YES	17783
greedy_best_first_graph_search_h1	38.93	NO	4031
astar_search_h_ignore_preconditions	49.91	YES	5003

The cells in green represent that that column has the best value on the given problem, the ones in red represent the worst value.

In this table we represent the best (when best is measured by speed) informed and uniformed algorithms, we have included two uninformed algorithms because the best uninformed algorithm was not optimal.

As we can see in this table, greedy best first graph search algorithms yields better results in all the problems, but is not optimal in the last problem, so if optimality is an issue, then A* search ignore preconditions is the best choice.

The gap between informed search algorithms and uninformed search algorithms gets close as the problems grows in size (branching factor) and the goals complexity increases, and eventually the informed search algorithms surpass the uninformed ones when a certain complexity is achieved. The reason for which uninformed search algorithms are often better in small problems is because heuristics are computational expensive, and in those problems is better to explore more nodes than check if that branch is best than another one.