Background
Planning problems are a staple of classical AI and project 3, Implementing a Planning Search, requires that a program for performing planning search is completed (framework provided by Udacity) and that the results of running various types of search using the program are compared with each other. This report documents the results. resulting from my own testing and my analysis of these results.

Uninformed Search Strategies
3 uninformed search strategies were compared for air cargo problems p1, p2 and p3:

- Breadth First Search
- Depth First Search
- Uniform Cost Search

The results are shown in tables 1,2 and 3 below.

Problem	Search	Node Expansions	Goal Tests	Time Elapsed (s)	Plan Length	Planning Path
p1	BFS	43	56	0.19718860317315653	6	Load(C1, P1, SFO) Load(C2, P2, JFK) Fly(P2, JFK, SFO) Unload(C2, P2, SFO) Fly(P1, SFO, JFK) Unload(C1, P1, JFK)
p1	DFS	21	22	0.02392778599523538	20	Fly(P1, SFO, JFK) Fly(P2, JFK, SFO) Load(C2, P1, JFK) Fly(P1, JFK, SFO) Fly(P2, SFO, JFK) Unload(C2, P1, SFO) Fly(P1, SFO, JFK) Fly(P2, JFK, SFO) Load(C2, P2, SFO) Fly(P1, JFK, SFO) Load(C1, P2, SFO) Fly(P2, SFO, JFK) Fly(P1, SFO, JFK) Unload(C2, P2, JFK) Unload(C1, P2, JFK) Fly(P2, JFK, SFO) Load(C2, P1, JFK) Fly(P1, JFK, SFO) Fly(P2, SFO, JFK) Unload(C2, P1, SFO)
p1	UCS	55	57	0.05737865428834286	6	Load(C1, P1, SFO) Load(C2, P2, JFK) Fly(P1, SFO, JFK) Fly(P2, JFK, SFO) Unload(C1, P1, JFK) Unload(C2, P2, SFO)

Table 1 – Results of using uninformed search strategies on air cargo problem p1

Problem	Search	Node Expansions	Goal Tests	Time Elapsed (s)	Plan Length	Plan Path
p2	BFS	3343	4609	26.171465976646548	9	Load(C1, P1, SFO) Load(C2, P2, JFK) 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)
p2	DFS	624	625	4.16513846587114	619	Too large to fit here
p2	UCS	4853	4855	15.863818732601194	9	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(C1, P1, JFK) Unload(C2, P2, SFO) Unload(C3, P3, SFO)

Table 2 – Results of using uninformed search strategies on air cargo problem p2

Problem	Search	Node Expansions	Goal Tests	Time Elapsed (s)	Plan Length	Plan Path
p3	BFS	14663	18098	164.43370933846663	12	Load(C1, P1, SFO) Load(C2, P2, JFK) Fly(P2, JFK, ORD) Load(C4, P2, ORD) Fly(P1, SFO, ATL) Load(C3, P1, ATL) Fly(P1, ATL, JFK) Unload(C1, P1, JFK) Unload(C3, P1, JFK) Fly(P2, ORD, SFO) Unload(C2, P2, SFO) Unload(C4, P2, SFO)
p3	DFS	408	409	2.242594127521531	392	Too large to fit here
p3	UCS	18151	18153	67.93223655932519	12	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(P2, ORD, SFO) Fly(P1, ATL, JFK) Unload(C1, P1, JFK) Unload(C2, P2, SFO) Unload(C3, P1, JFK) Unload(C4, P2, SFO)

Table 3 – Results of using uninformed search strategies on air cargo problem p3

From the above we can see that in all 3 problems BFS and UCS both arrive at the solutions with the lowest number of path steps and in all cases BFS and UCS actually suggest solutions with identical number of path steps. This suggests that the implementation of UCS has the same costs for every action and is therefore the same algorithm as BFS in this instance.

BFS (and in this case UCS) are expected to produce optimal solutions as they explore each level of the solution tree progressively and the first solution encountered will therefore be optimal, if path cost is a non-decreasing function of depth (Russell and Norvig, p82). In this case all actions do have same cost so the condition is met. DFS in contrast produces a very lengthy solution to all 3 problems which is not optimal and again this is to be expected (Russell and Norvig, p86). DFS expands the deepest node first exploring branches of the tree in their entirety before moving on to the next branch. In my tests DFS has found a solution faster and by examining fewer nodes but it is a solution that is much deeper down the tree structure and therefore has many more steps.

In all 3 problems BFS took the longest time to calculate the solution generating a large number of nodes. In contrast DFS was by far the fastest and generated far fewer nodes. From reading the literature (Russell and Norvig, p82 & p87) I did not necessarily expect this as result as BFS is

expected to generate $O(b^d)$ nodes and DFS $O(b^m)$ where b is the branching factor, d is the depth at which the solution is found by BFS and m is maximum depth of any node. This result is probably due more to the nature of the problems than the search algorithms themselves. DFS has simply come across a deep solution in a branch of the search tree relatively early in its search process.

Russell and Norvig, p82, suggests that in the case where all actions have the same cost, UCS will generate slightly more nodes than BFS, $O(b^{d+1})$ nodes versus $O(b^d)$ nodes. This prediction is observed in the results above with UCS generating slightly more nodes than BFS for all 3 problems. However UCS was significantly faster to produce a solution in all 3 problems. I researched the forums and others had reported the same observation. One possibility suggested is that a change to the implementation of the priority queue in the software may have caused this (https://discussions.udacity.com/t/time-elapsed-bfs-vs-ucs/311680/7, accessed 22/7/17 11:15am).

Searching Using Heuristics

Logic was added to the search program to obtain heuristics that could be employed in the A* Search.

- Relaxed Constraint Heuristic (ignore preconditions)
- Level Sum Heuristic using Planning Graph

Tables 4-6 below show the results.

Problem	Search	Node Expansions	Goal Tests	Time Elapsed (s)	Plan Length	Planning Path
p1	A* with ignore preconditions heuristic	41	43	0.04555577861060787	6	Load(C1, P1, SFO) Fly(P1, SFO, JFK) Unload(C1, P1, JFK) Load(C2, P2, JFK) Fly(P2, JFK, SFO) Unload(C2, P2, SFO)
p1	A* with level sum heuristic	11	13	8.32104902106885	6	Load(C1, P1, SFO) Fly(P1, SFO, JFK) Load(C2, P2, JFK) Fly(P2, JFK, SFO) Unload(C1, P1, JFK) Unload(C2, P2, SFO)

Table 4 – Results of using A* search strategy with various heuristics on air cargo problem p1

Problem	Search	Node Expansions	Goal Tests	Time Elapsed (s)	Plan Length	Planning Path
p2	A* with ignore preconditions heuristic	1450	1452	4.7647215308199815	9	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)
p2	A* with level sum heuristic	86	88	319.3406190229286	9	Load(C1, P1, SFO) Fly(P1, SFO, JFK) Load(C2, P2, JFK) Fly(P2, JFK, SFO) Load(C3, P3, ATL) Fly(P3, ATL, SFO) Unload(C1, P1, JFK) Unload(C2, P2, SFO) Unload(C3, P3, SFO)

Table 5 – Results of using A* search strategy with various heuristics on air cargo problem p2

Problem	Search	Node Expansions	Goal Tests	Time Elapsed (s)	Plan Length	Planning Path
p3	A* with ignore preconditions heuristic	5038	5040	22.08229622997277	12	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)
p3	A* with level sum heuristic	314	316	1578.6843441216715	12	Load(C2, P2, JFK) Fly(P2, JFK, ORD) Load(C4, P2, ORD) Fly(P2, ORD, SFO) Load(C1, P1, SFO) Fly(P1, SFO, ATL) Load(C3, P1, ATL) Fly(P1, ATL, JFK) Unload(C1, P1, JFK) Unload(C2, P2, SFO) Unload(C3, P1, JFK) Unload(C4, P2, SFO)

Table 6 – Results of using A* search strategy with various heuristics on air cargo problem p3

A* search using ignore preconditions heuristic produces optimal solutions but does so much faster than BFS/UCS. A* search using Planning Graph Level Sum heuristic also produces an optimal solution but does so much slower than any of the other other strategies tested. The A* searches using heuristics have the advantage of expanding fewer nodes than standard BFS/UCS and DFS which may be significant when faced with large search spaces and/ or limited computer storage. A* with level sum heuristic is particularly frugal when it comes to nodes expanded but is significanty slower.

One further observation that could be made about all of the solutions produced above that they are not solutions that would be followed in real life. They do not take into account actions that can be conducted in parallel and those that need to be conducted in series. To produce more practical solutions for problems of this nature it would be necessary to consider partially ordered plans (Russell and Norvig, p390) which allows for sub-plans of actions with dependencies on each other can be planned to be conducted in parallel with independent sub-plans.

References

Russell, S. J. and Norvig, P., Artificial Intelligence, A Modern Approach, Pearson Education Limited (2016)