Project 3

This paper presents the results of using the different search strategies to solve the three different problem sets for the the project, and discusses their performance characteristics. The results for each problem are presented in separate tables below.

Not all strategies completed in a humanly reasonable time. For example, I let the Breadth First Tree search run all day while I was at work – which allowed it sufficient time to suck up all the memory in my computer, forcing me to do a hard reboot. (I am not very happy with Breadth First Tree search right now…). Those methods are marked in the tables with the dreaded DNF attribute.

Problem 1

Problem 1

	Breadth First	Breadth First Tree	Depth First Graph	Depth Limited	Uniform Cost	Recursive Best First	Greedy Best First Graph	A* H1	A* Ignore Preconditions	A* PG Level Sum
Expansions	43	1458	21	101	55	4229	7	55	41	11
Goal Tests	56	1459	22	271	57	4230	9	57	43	13
New Nodes	180	5960	84	414	224	17023	28	224	170	50
Elapsed Time	0.031	0.942	0.014	0.097	0.040	2.765	0.005	0.041	0.041	1.612
Plan Length	6	6	20	50	6	6	6	6	6	6
Plan	Load(C1, P1, SFO) Load(C2, P2, JFK) Fly(P2, JFK, SFO) Unload(C2, P2, SFO) Fly(P1, SFO, JFK) Unload(C1, P1, JFK)	Load(C1, P1, SFO) Load(C2, P2, JFK) Fly(P2, JFK, SFO) Unload(C2, P2, SFO) Fly(P1, SFO, JFK) Unload(C1, P1, JFK)	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)	Load(C1, P1, SFO) Load(C2, P2, JFK) Unload(C1, P1, SFO) Load(C1, P1, SFO) Unload(C1, P1, SFO) Load(C1, P1, SFO) Unload(C1, P1, SFO) Load(C1, P1, SFO) Unload(C1, P1, SFO) Load(C1, P1, SFO) Unload(C1, P1, SFO) Load(C1, P1, SFO) Unload(C1, P1, SFO) Load(C1, P1, SFO) Unload(C1, P1, SFO) Load(C1, P1, SFO) Unload(C1, P1, SFO) Load(C1, P1, SFO) Unload(C1, P1, SFO) Load(C1, P1, SFO)	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(C2, P2, JFK) Load(C1, P1, SFO) Fly(P2, JFK, SFO) Unload(C2, P2, SFO) Fly(P1, SFO, JFK) Unload(C1, P1, JFK)	Load(C1, P1, SFO) Load(C2, P2, JFK) Fly(P1, SFO, JFK) Fly(P2, JFK, SFO) Unload(C2, P2, SFO) Unload(C1, P1, JFK)	Load(C1, P1, SFO) Load(C2, P2, JFK) Fly(P1, SFO, JFK) Fly(P2, JFK, SFO) Unload(C2, P2, SFO) Unload(C1, P1, JFK)	Load(C1, P1, SFO) Load(C2, P2, JFK) Fly(P1, SFO, JFK) Unload(C1, P1, JFK) Load(C2, P2, JFK) Fly(P2, JFK, SFO) Unload(C2, P2, SFO) Unload(C1, P1, SFO)	Load(C1, P1, SFO) Fly(P1, SFO, JFK) Load(C2, P2, JFK) Fly(P2, JFK, SFO) Unload(C1, P1, SFO) Unload(C2, P2, SFO)

All the strategies succeeded for Problem 1, though warning signs are already present for some strategies even with such a limited problem.

As expected, the breadth first, Uniform Cost, and the A* strategies all found an optimal plan. Given sufficient time, all six of these strategies are guaranteed to find the shortest plan, and since each step carries the same cost in this project, the shortest path is an optimal plan; note that not all six found the same plan, but they are all of the same length.

Note: yes, A*+H1 is equivalent to Uniform Cost. I won't discuss it further, and include it in the tables solely for completeness and to make the font a little too small to read comfortably.

Both of the two depth first strategies find plans, but neither plan is optimal. As depth first strategies are not guaranteed to find a solution at all, it is risky to use unbounded versions of them for a general problem where you aren't certain of a reasonable response time.

The two "best first" searches also found optimal plans.

Although all the strategies succeeded, three of the methods show disturbing characteristics.

1. Breadth First Tree search shows more than a one magnitude difference in expansions, goal tests, new nodes, and elapsed time when compared with the Breadth First search results.
2. Depth Limited search discovered a troublingly long path, and the expansions, goal tests, *et al* reflect that.
3. Recursive Best First, although finding an optimal plan, required almost 100x the time and resources of Breadth First to do it.

These characteristics do not bode well for more complex problems.

Problem 2

Problem 2

	Breadth First	Breadth First Tree	Depth First Graph	Depth Limited	Uniform Cost	Recursive Best First	Greedy Best First Graph	A* H1	A* Ignore Preconditions	A* PG Level Sum
Expansions	3343	*DNF*	624	*DNF*	4853	*DNF*	998	4853	1506	86
Goal Tests	4609		625		4855		1000	4855	1508	88
New Nodes	30509		5602		44041		8982	44041	13820	841
Elapsed Time	13.808		3.396		45.568		7.622	46.015	14.558	174.584
Plan Length	9		619		9		21	9	9	9
Plan	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)		Fly(P3, ATL, SFO) Fly(P1, SFO, ATL) Fly(P3, SFO, JFK) Fly(P1, ATL, JFK) Fly(P2, JFK, ATL) Fly(P3, JFK, ATL) Fly(P2, ATL, SFO) Fly(P3, ATL, SFO) Load(C2, P1, JFK) Fly(P2, SFO, ATL) Fly(P1, JFK, ATL) Fly(P2, ATL, JFK) Fly(P1, ATL, SFO) Fly(P3, SFO, ATL) Fly(P1, SFO, JFK) Load(C3, P3, ATL) Fly(P3, ATL, SFO) Fly(P2, JFK, ATL) Fly(P3, SFO, JFK) Fly(P2, ATL, SFO) Fly(P1, JFK, ATL) Fly(P2, SFO, JFK)		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)		Load(C1, P1, SFO) Load(C2, P2, JFK) Load(C3, P3, ATL) Fly(P1, SFO, ATL) Fly(P2, JFK, ATL) Fly(P3, ATL, JFK) Fly(P2, ATL, SFO) Unload(C2, P2, SFO) Fly(P2, SFO, ATL) Fly(P3, JFK, SFO) Load(C2, P3, SFO) Fly(P3, SFO, JFK) Fly(P1, ATL, JFK) Unload(C1, P1, JFK) Load(C1, P3, JFK) Fly(P1, JFK, ATL) Fly(P3, JFK, SFO) Unload(C3, P3, SFO) Unload(C2, P3, SFO) Fly(P3, SFO, JFK) Unload(C1, P3, JFK)	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)	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)	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(C3, P3, SFO) Unload(C2, P2, SFO) Unload(C1, P1, JFK)

And this is one of those more complex problems.

As mentioned above, Breadth First Tree search brought my computer to its knees in a multi-hour run. Depth Limited and Recursive Best First were not given as much leeway, but they ran a long, long time. All three of these get the dreaded DNF for problems 2 & 3.

Ignoring A*+H1 as redundant leaves six strategies to look at for the final two problems.

Again, as expected, Breadth First, Uniform Cost and the A* methods all find optimal plans, though, also again, not the same plan. But which of the four is “the best”? Well, as long as the heuristic is admissible, the A* strategies will never be worst than Uniform Cost, so we can prune Uniform Cost from the rest of the analysis.

But of the remaining three, what is “best” really depends on the problem, in particular how much is “costs” to expand a node, test a goal, etc. In theory A* will outperform an uninformed breadth first search in terms of the number of nodes processed, but an uninformed breadth first search has much less overhead, so it can process many more nodes in a given amount of time. For less complex problems this can mean that the uninformed search is best. Indeed we can see in the table for Problem 2 that A*+Ignore Preconditions (A*+IP) processes ~1/3 the nodes that Breadth First does, the cost of deciding which nodes to process means that it actually takes slightly longer to come up with an equivalent plan. In turn, A*+IP processes almost 15x more nodes than A*+ PG Level Sum (A*+LS), yet it finds an equivalent plan more than 10x faster.

Greedy Best First search beats all of those for speed, coming in almost 2x faster than Breadth First, but its plan is far from optimal, with 21 steps rather than the optimal 9. If calculation speed is more important than optimality (*ie*: “just do something!!!”) then this may be a viable alternative, but it is not guaranteed to find a solution…

And neither is our speed champ for Problem 2: Depth First Graph search comes in more than 2x as fast as Greedy Best First. But its plan...619 steps is 610 more than the optimal 9. The success of Depth First Graph is heavily dependent on the initial conditions of the problem, as we shall see in Problem 3.

Problem 3

Problem 3

	Breadth First	Breadth First Tree	Depth First Graph	Depth Limited	Uniform Cost	Recursive Best First	Greedy Best First Graph	A* H1	A* Ignore Preconditions	A* PG Level Sum
Expansions	14663	*DNF*	408	*DNF*	18223	*DNF*	5578	18223	5118	404
Goal Tests	18098		409		18225		5580	18225	5120	406
New Nodes	129631		3364		159618		49150	159618	45650	3718
Elapsed Time	169.767		1.858		416.173		105.086	398.206	88.404	1192.641
Plan Length	12		392		12		22	12	12	12
Plan	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)		Fly(P1, SFO, ORD) Fly(P2, JFK, ORD) Fly(P1, ORD, ATL) Fly(P2, ORD, ATL) Fly(P1, ATL, JFK) Fly(P2, ATL, SFO) Load(C2, P1, JFK) Fly(P2, SFO, ORD) Fly(P1, JFK, ORD) Fly(P2, ORD, ATL) Fly(P1, ORD, ATL) Fly(P2, ATL, JFK) Fly(P1, ATL, SFO) Unload(C2, P1, SFO) Fly(P1, SFO, ORD) Fly(P2, JFK, ORD) Fly(P1, ORD, ATL) Fly(P2, ORD, ATL) Fly(P1, ATL, JFK) Fly(P2, ATL, SFO) Load(C2, P2, SFO) Fly(P2, SFO, ORD) Fly(P1, JFK, ORD)		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(C4, P2, SFO) Unload(C3, P1, JFK) Unload(C2, P2, SFO) Unload(C1, P1, JFK)		Load(C1, P1, SFO) Load(C2, P2, JFK) Fly(P1, SFO, ORD) Load(C4, P1, ORD) Fly(P2, JFK, ATL) Load(C3, P2, ATL) Fly(P2, ATL, ORD) Fly(P1, ORD, ATL) Unload(C4, P1, ATL) Fly(P1, ATL, ORD) Fly(P2, ORD, ATL) Load(C4, P2, ATL) Fly(P2, ATL, ORD) Unload(C3, P2, ORD) Load(C3, P1, ORD) Fly(P1, ORD, JFK) Unload(C3, P1, JFK) Unload(C1, P1, JFK) Fly(P1, JFK, ORD) Fly(P2, ORD, SFO) Unload(C4, P2, SFO) Unload(C2, P2, SFO)	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(C4, P2, SFO) Unload(C3, P1, JFK) Unload(C2, P2, SFO) Unload(C1, P1, JFK)	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(C4, P2, SFO) Unload(C3, P1, JFK) Unload(C2, P2, SFO) Unload(C1, P1, JFK)	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(C4, P2, SFO) Unload(C3, P1, JFK) Unload(C2, P2, SFO) Unload(C1, P1, JFK)

Depth First Graph wins the speed championship here hands down, more than 40x faster than any other strategy.

It also loses the optimality championship with a plan length of 392, a full 380 steps longer than the optimal plan. Note that even though Problem 3 is more complex than Problem 2, it finds a plan that is a couple of hundred steps shorter than in Problem 2. This speaks to the variation due to initial conditions. It’s not impossible for Depth First Graph to come up with an optimal plan, it’s just really, really, ***really*** unlikely.

Greedy Best First continues to perform quickly without creating a cripplingly long plan. However, its performance is moving back towards the uninformed Breadth First, now processing almost a third of the nodes as Breadth First, and worse, it solves Problem 3 in roughly the same amount of time, all without producing an optimal plan.

Breadth First, which is really what we want to compare the other strategies against because it is guaranteed to find a shortest-plan solution, processes about 4x as many nodes in about 8x the time as compared to Problem 2. This demonstrates how the processing time for such an uninformed strategy increases non-linearly with the number of nodes.

As the complexity increases, A*+IP really starts to shine. For Problem 3 it is not only processing ~1/3 the nodes of Breadth First, it’s also significantly faster than Breadth First.

Compare A*+LS is really great...except when it isn’t. It processes 10x fewer nodes than A*+IP, roughly the same number as processed by Depth First Graph (while coming up with a plan that is 380 steps shorter!). It is a really exceptionally efficient strategy.

In terms of nodes.

But (there’s always a but, isn’t there?), the computation cost is extremely high. The crux is that even though it processes 10x fewer nodes than A*+IP, it takes 10x longer to do it – that is, each node requires 100x more processing time than for A*+IP. While it is likely that in a sufficiently complex problem the reduction in nodes will eventually overtake that overhead, for a problem as simple as Problem 3 it is just not an efficient strategy.

Ultimately, though, it’s pretty clear that as problems become more complex, an A* with a good heuristic will leave other methods in the dust, though case can be made for using depth first searches when the depth is relatively bounded and an optimal solution is not required.