- **Use a table or chart to analyze the number of nodes expanded against number of actions in the domain**
  One thing to note is that expanded nodes are (almost) never more than the number of actions in the greedy algorithms, with the exception of problem 4 with the set level heuristic, which is still 107 expansions against 104 actions…still pretty close. That's because greedy won't ever backtrack, it'll just "stubbornly/greedily" go towards the goal without "taking the past in account, only the future". Depth first search behaves similarly (to an extent), as it doesn't really backtrack as well and most of its expanded nodes are actually part of the solution. Its solutions are (understandably) very far from optimal. A* with the level sum heuristic seems to "compete well" with the above in the expanded against actions ratio, which is probably why it's considered an accurate heuristic, even though it's inadmissible which can be seen by its non optimal solution in problem 4. For the rest of the algorithms, they behave somewhat as expected, their expanded against actions ratio increasing "appropriately" with the increase of the number of actions in the domain.

- **Use a table or chart to analyze the search time against the number of actions in the domain**
  The "losers" here are A* algorithms with max level and set level heuristic, especially in problems 3 and 4. Max level probably because it's not very accurate and set level probably because it doesn't take in account mutexes between more than two literals, so it doesn't "steer" the search in the right direction as the problem domains get larger. On the other hand, they are the ones (ones of the few in problem 4 in fact) that do provide the optimal plans. The other algorithms are mostly behaving "as expected" in this aspect, greedy ones taking the lead mostly. Except for the interesting surprise which is how well A* search with the usually "not so great" unmet goals heuristic does in this aspect (and in general). And it even finds the optimal solution in all problems, which I believe is because it's suited for this type of planning problem, as there are not that many actions needed to fulfil a single goal, probably 2-3, load cargo at plane (if it isn't loaded by any chance), fly plane, unload cargo.

- **Use a table or chart to analyze the length of the plans returned by each algorithm on all search problems**
  Depth first search is the "definitive loser" here for all problems and for obvious reasons. It just doesn't work for these types of problems, it chooses actions "at random" and with such large state space it just can't get to a solution even remotely close to the optimal one. Unlike in the pacman case, f.e. in a 10X10 maze, where it would just stick to one side of the wall, but its solutions would still be about up to 2-3 times longer than the optimal one, the difference wouldn't be this much. It's just that there are so many more states in this type of state representation (atomic vs factored). Otherwise, most of the other algorithms do relatively well, as all of them give the

optimal solution in problems 1 and 2. In problem 3, only the greedy ones fail, which is understandable considering the “greedy” part as I explained in the first discussion. They just don’t care about the past actions, they try to get as fast as possible to the goal from whichever state their previous decision lead them to. Their results would probably still be considered “respectable”, considering the amount of expanded nodes i.e. the space they are using for the search (which is much lower than the other algorithms), and the plan lengths are not that far off from the optimal solution, excluding maybe the one using the set level heuristic. Still, nowhere near the “failure” that is depth first search. So, all but greedy and depth first search still deliver the optimal solution in problem 3, while in problem 4, A* with level sum heuristic “joins the club” of the providers of non optimal solution, although just by one step. It can be assumed it’s because of inadmissibility of the level sum heuristic. Still, it uses a lot less space than the ones giving the optimal solution and it’s probably “a good compromise” choice if space (and to some extent time) is important, and we can settle for a “good enough” solution. Since the greedy ones are still delivering worse plan lengths by not saving that much space (and to some extent time).

- **Which algorithm or algorithms would be most appropriate for planning in a very restricted domain (i.e., one that has only a few actions) and needs to operate in real time?**

I’d say about anything except depth first search and greedy, but probably breadth first search would be the most appropriate especially if we want an optimal solution.

- **Which algorithm or algorithms would be most appropriate for planning in very large domains (e.g., planning delivery routes for all UPS drivers in the U.S. on a given day)**

Breadth first search would probably need too much space, and if we could settle for a non optimal solution, we could go with the greedy ones, as they’d be “fast and furious” in delivering something relatively close to the optimal solution. But if we want an optimal solution, since the problem seems to correspond to the air cargo problem in this project and A* with unmet goals behaves really well, that would be the most appropriate choice in my opinion.

- **Which algorithm or algorithms would be most appropriate for planning problems where it is important to find only optimal plans?**

Again, breadth first search would use too much space, but if space is no problem, it could be a valid choice since time wise it behaves better than most A* at least in this particular problem. Otherwise, any of the A* with an admissible heuristic, leading by unmet goals if the problem allows, that is there’s no such thing as some action may achieve multiple goals (which will make it inadmissible), trailed by set level and max level heuristics.