**1.     For each of the three puzzle types, what do you think is the best search strategy, and why? You should take all four factors into considerations (completeness, optimality, time complexity, space complexity).**

For the water jug problem, the best search strategy is bfs. From a completeness standpoint, bfs is a complete search algorithm since it will traverse through all possible BFS and find first solution. This also means that bfs is also optimal because it will return the best solution. However, it is not efficient in space and time complexity since it has to keep the breadth of the expansion nodes. However, given that the water jugs do not have many breadth states, it is ok for bfs to be used. This means that dfs and its search algorithms are not needed as much as the states generated are not too many. Uniform-cost is simply a bfs since it the states do not have costs associated. Because there are no costs, uniform-cost will be more inefficient, doing more overhead work to keep track of costs when costs are not accounted for. Informed searches will be difficult and dependent on the heuristics. Given the heuristics I have created, they are not admissible and can lead to less optimal solutions.

For the path planning city problem, the best search strategy is uniform-cost. Uniform-cost is a complete and optimal solution since it will find a solution given that such a solution exists and finds the optimal solution with cost accounted using a priority queue. The time and space complexity of uniform-cost is exponential. However, this is fine since the optimal solution will still be reached. As well, the branching factor of the given problem is not very high and thus the solution will not take very long or that much space. A\* is another alternative. However, the heuristic in this given problem is using location. By using location, we cannot guarantee that the cost of the heuristic will be less than or equal to the actual cost. One scenario is that two cities can be close to each other but the cost is high, maybe due to construction or road block that makes the path cost higher than taking another city route to reach the destination. However, if the heuristic is less than or equal to actual cost, then A\* can be a good solution.

For the pancake problem, the best search strategy is A\*. If the heuristics are admissible, A\* is complete and optimal since it will perform no worse than uniform-cost. Time and space complexity is exponential. However, compared to uniform-cost and other uninformed search algorithms, the heuristics help to narrow the searches to provide a more efficient search, where the breadth of the expansion nodes are large.

**2.     For the water jugs and pancakes problem, what heuristic function(s) did you make? Give your rationale. Also, discuss whether you think they are admissible and consistent.**

For the water jug problem, I used two heuristics that are variants of the manhattan distance to the goal state.

The ‘goal\_diff\_strict’ heuristic returns the absolute value of the difference in all the jugs of the goal state and the current state. For example a current state of [1, 3] and goal state of [0, 2] will return a cost of 2. The ‘goal\_diff\_relaxed’ heuristic returns the absolute value of the difference in only goal state jugs that are not empty. For example, a current state of [1,3] and goal state of [0, 2] will return a cost of 1.

The rationale for these two heuristics is that the smaller in difference to the goal state, the closer the current state is to the goal state. The relaxed heuristic ignores the ‘0’ in goal states so that more states can be accounted for.

I think neither heuristics are admissible and consistent because both can have a state that has a heuristic cost greater than actual cost. An example is a goal state of [0, 1] and [1, 4]. The actual cost would be two because once can simply empty the jug with 4 and transfer 2 to 0 to reach [0, 1]. However, the two heuristics will give a cost greater than 2. The strict heuristic will give a cost of 4 while the relaxed heuristic will give a cost of 3.

For the path planning problem, I used euclidean and manhattan distance based on the location of the city. The rationale behind this heuristic is that the closer the two cities are location wise, the lower the cost of the actual path. Both euclidean and manhattan will result in similar heuristic estimation since the location will be used however.

I think neither heuristics are admissible and consistent because both the euclidean and manhattan distance are assuming that closer locations will result in shorter costs. This will not always be the case as there are cases where two cities can be very close but the actual cost is actually very large due to external constraints such as roadblocks or traffic.

For the pancake problem, I used the heuristic function of manhattan distance based on neighbors and manhattan distance based on goal state. In the manhattan distance based on neighbors, the heuristic function ignores flipped pancakes and finds the distance of the current size to neighbors of the same size in the goal state.

For example, a goal state of [1, 2, 3] and current state of [2, -1, 3] will return a cost breakdown of:

       pancake\_size - cost

       [1] - 0

       [2] - 1

       [3] - 1

       Total cost: 2

In the manhattan distance to goal state, the heuristic ignores the flipped state and looks at the distance of the current size to the goal state’s size’s index. For the same example, the breakdown will be:

[1] - 1

[2] - 1

[3] - 0

Total cost: 2

The rationale for the distance to goal state is that the closer in order to the goal, the closer the state is to the goal. The rationale for the neighbor is that good states will have close neighbors, even if the order is not correct. This is more relaxed since the states might be flipped.

I do not think both heuristics are admissible and consistent because there are still cases where the heuristic cost is greater than actual cost. For example, goal state of [1, 2, 3] and current state of [-2, -1, 3] will have a cost greater than 1 for both heuristics.

**3. What heuristic function did you use for each puzzle class? Did all the outcomes make sense (e.g., do the time/space complexities of different search strategies match your expectation based on our class discussions? What about optimality and completeness?)**

For each puzzle, I tried all the algorithms on it. The outcomes makes sense given the algorithms. The bfs has more space used than iddfs but less time complexity because it keeps track of the explored list. Similarly, ida\* also has high time complexity for the same reason. The informed searches generally have less time complexity than the uninformed searches because the heuristics help to narrow some of the state space searches. On problems like path planning however, both uniform-cost and astar have similar time and space complexity because the heuristic has a similar cost to actual cost. Greedy’s time complexity can fluctuate depending on heuristic and also luckiness (order in which the states were picked). In all test cases unfortunately, the time complexity was significantly higher.

Optimality and completeness were discussed in the questions above. However, in general, bfs, a\*, unicost, ida\* and iddfs are optimal and complete while dfs and greedy are not necessarily optimal and complete.

**Discussion**

The dfs and ida\* loops infinitely for jugs problem and pancakes because the explored list is not kept and thus, nodes that have previously be visited can be continuously visited.

The pancake problem breaks a lot of the algorithms simply because it has a lot of possible states to iterate through. In test\_pancakes1.config, there are 11 different sizes. This means that the max branching factor is 11. In the test\_pancakes2.config, the branching factor is even larger. Because of this searching for a solution takes up a very long time. Only by using heuristics, can we shorten the search time by guessing which state is better at reaching the goal state.