Problem A.5 writeup:

1. What were your results from `compare_cow_transport_algorithms`? Which algorithm runs faster? Why?

The brute-force algorithm was significantly slower, and this is a result of its complexity being exponential since it iterates over all possibilities of sets of trips. On the other hand, the complexity of the greedy algorithm was linear since it only iterates over a sorted list of elements, in our case, astronaut cows being the elements?!

2. Does the greedy algorithm return the optimal solution? Why/why not?

The greedy algorithm does not necessarily return the global-optimal solution since it assumes that the optimal set of trips is the one where you choose the cows by order of weight, from heaviest to lightest. In short, it only tries one possibility of the sets of trips, whereas brute-force tries all of them.

3. Does the brute force algorithm return the optimal solution? Why/why not?

The brute-force algorithm does indeed return the global-optimal solution since it iterates over all possibilities of sets of trips and checks which one has the least amount of trips.

Problem B.2 writeup:

1. Explain why it would be difficult to use a brute force algorithm to solve this problem if there were 30 different egg weights. You do not need to implement a brute force algorithm in order to answer this.

The complexity will be at least exponential since we need the powerset. And aside from the powerset, we will also need to figure out the optimal combination of weights within each set, which itself has an exponential complexity too.

2. If you were to implement a greedy algorithm for finding the minimum number of eggs needed, what would the objective function be? What would the constraints be? What strategy would your greedy algorithm follow to pick which coins to take? You do not need to implement a greedy algorithm in order to answer this.

I would iterate through a list of egg weights that is sorted from heaviest to lightest, and pick as many eggs as I can in each iteration, which will ultimately give us the global-optimal solution since this is both a greedy algorithm and dynamic programming (dynamic programming because we know we won't need the light eggs and so we just pick as much of the heavy eggs as we can, essentially eliminating the permutations that have more light eggs than heavy eggs).

3. Will a greedy algorithm always return the optimal solution to this problem? Explain why it is optimal or give an example of when it will not return the optimal solution. Again, you do not need to implement a greedy algorithm in order to answer this.

Yes, it will always return the global-optimal solution as explained in the previous question.