PS1

Part A: Transporting Cows Across Space

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

   the greedy algorithm took ~ $10^{-7} sec$
   the brute force algorithm took ~ $10^{-1} sec$

   the greedy algorithm runs faster because it works in a consistent and quite efficient manner. It iterates on each object no more than n times, and in general the complexity is ~ $O(n^2)$

   the brute force algorithm saves up space on the subset generation, but it has to go about randomly through every subset and test if it is valid. in general there are ~ $O(2^n)$ subsets for a given set.

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

   No. this algorithm returns a local optimum because it is programmed to work in a certain way, not testing every option and choose the best out of them, but load the largest cow first, then the next and so on. the way this algorithm goes it can miss a better option to fill the spaceship because it insists on taking the largest cow next instead of trying to reach exactly/as close to the limit as possible with every trip.

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

   the brute force always gives the optimal solution. the reason is that it iterates over each and every possible option and chooses the one with the least amount of trips.

Part B: Golden Eggs

1. **Were 30 different egg weights. You do not need to implement a brute force algorithm in order to answer this.**

   Brute force algorithm tries every combination possible. with 30 different egg weights the complexity of the algorithm will be $2^{30} \approx 10^9$ options to consider, or more accurately $\sum (2^n + 1)$

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.**

   In case we were to implement a greedy algorithm, the objective function to optimize would be the value/volume, i.e. minimizing the number of eggs.

   The constraints would've been the weight limit on the ship.

   in each step the algorithm will check if the heaviest egg can fit in the ship.
   - If it can, keep doing the same thing.
   - if it can't, try fitting the next heaviest.

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.**

   No, it will not always return the optimal solution.

   An example for greedy algorithm supplying a local optimum is:

   target/constraint = 19
   list of egg weights = [14, 6, 1]

   the greedy algorithm will return 6 for the eggs - 14, 1, 1, 1, 1, 1 = 19

   the optimal solution is 4 for the eggs - 6, 6, 6, 1