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

Answer: My results where as listed:

Greedy algorithm completed in 0.0 second(s) and completed the task in 6 trips
Brute force alogrithm completed in 0.428 second(s) and completed the task in 5 trips

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

Answer: The greedy algorithm does NOT return an optimal solution, this is due to the limitations of the greedy algorithm not having the context of the other objects when making a decision, and just going for the heaviest object.

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

Answer: The brute force does return an optimal solution, due to the advantage of being able to look at every single case scenario, we can say with certainty that this solution is optimal.

4. 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.

Answer: Due to the higher compute time, as seen above, that comes as a consequence of using the brute force algorithm. Typically implementations of the brute force algorithm are O(2^n) or exponetial.

5. 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

Answer: if i were to implement a greedy algorithm, the objective function would be to start with the heavyiest eggs, and then work downwards as you start to constrain on weight, until you have reached max capacity. This is not as optimal, due to the fact that you dont have the full context of the egg sizes you have, and thus dont make a fully optimized choice.

6. 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

Answer: As stated above, the greedy algorithm is not optimal in this situation. Take an example of n=11 and egg_weights=(1, 5, 8). The greedy algorithm would start with the largest egg, which is 8, and then, due to weight constraints, it will resort to solving the rest of the problem with the 1 egg resulting in 4 eggs being used. However with the context
we are giving using dynamic programming, if we were to use two 5 eggs
and
a 1 egg, we would bring are total number of eggs used down to 3.