MIT 6.0002 Problem Set 1 Write Up. - Clarke Homan

Problem A: Transporting Cows Across Space (Comparing Greedy versus Brute Force Algorithms) Writeup

- What were your results from compare_cows_transport_algorithms? Which algorithm runs faster? Why?
    - When both algorithms were executed, the brute force returned with a solution with fewer trips but took longer to compute. The greedy algorithm returned a solution with one extra trip but computed the results in significantly less time. The greedy algorithm converged on the 1st valid case, using an ordered set to select from. The brute force algorithm evaluated all combinations and then selected the 1st trip that didn't exceed weight restrictions but had the lower number of trips.

- Does the greedy algorithm return the optimal solution? Why or why not?
    - The greedy algorithm selects the 1st local maximum (or minimum) without evaluating global data characteristics.

- Does the brute force algorithm return the optimal solution? Why or why not?
    - The brute force algorithm returns the optimal solution because it evaluates all data characteristics but at the cost of computational duration. The lowest or highest solution is always determined as it evaluates all solutions but at the cost of compute time.

Problem B: Golden Eggs (Solving using Dynamic Programming) Writeup

- Explain why it would be difficult to use a brute force algorithm to solve this problem
    - To apply a brute force algorithm approach, you would need to enumerate all of the possible egg combinations, which includes 99 instances of the 1 lb eggs. Given the 4 different egg weight density instances, there would be a lot (exponential) number of cases to evaluate to see if their combined weight meets payload requirements and minimizes the number of eggs carried (minimizes space requirements).

- If you were to implement a greedy algorithm for finding the minimum number of eggs needed, what would the objective function be? What strategy would your greedy algorithm follow to pick which eggs to take?

    - Order (sort) the egg weights in descending order. Take as many heavier eggs that do not exceed the overall payload capacity, push the remainder weight (payload limit minus the current egg weights) to a recursive call to same function. That recursive function takes the next weight class egg that fits in the passed in remainder weight. That function then passes the remainder weight (if any) to a recursive invocation of the same function, and so on. Once the function accounts for the remaining weight (by selecting some egg weight that fits), it returns the number of eggs it needed. The invocating instance takes the egg count that it received from the recursive call and adds it to its selected egg count and then returns the sum. Finally the first invocation of the recursive function has the sum of all sums of selected eggs.

- Will a greedy algorithm always return the optimal solution to this problem?
    - Yes.