6.0002 Fall 2016 MIT OCW
Problem Set 1: Writeup

Octavio V.

Problem A.5

1. When I ran `compare_cow_transport_algorithms()`, I found the following results:
    a. `greedy_cow_transport` executed in 0.003987789154052734 s in 6 trips
    b. `brute_force_cow_transport` executed in 0.7275633811950684 s in 5 trips

   Clearly, the greedy algorithm runs much faster. This is because it is implemented so as to only add the heaviest cows first, regardless of how many trips it will take, as opposed to exploring the entire space of all possible trip configurations. The brute force algorithm instead explores the space of all trip configurations (=partitions of the cows) before deciding which are valid and which is best, which involves exploring on the order of 2^(# of cows), i.e. the size of the power set of the set of cows.
2. The greedy algorithm does NOT return the optimal solution. This is because it is only concerned with transporting the heaviest cows before transporting lighter cows from the sorted list of cows, as opposed to minimizing the number of trips this will take. If the weight limit were increased, then the number of trips required would begin to decrease under the greedy implementation.
3. The brute force algorithm DOES return the optimal solution. This is because it explores all possible trip configurations, and first decides which meet the weight limit constraint. Then it selects among all possible choices which minimizes the number of trips required.

Problem B.2

1. It would be difficult to use a brute force algorithm to solve this problem with 30 different egg weights because the algorithm would need to check, for a given number of eggs of each weight, whether that egg or set of eggs will achieve the desired weight. For any given target weight, there is a huge number of possible ways to partition the target into the sum of any combination of the 30 individual egg weights, all of which must be checked.
2. If we were to implement a greedy algorithm, the objective function would be to maximize the increase in weight per addition of egg (to minimize the number of eggs). The only constraint would be to stay under the target weight while adding egg weights. So, the greedy algorithm would start with the maximal elements in the tuple of egg weights, add as many of them as are possible before going over limit, and then move to the next largest egg weight and repeat.
3. A greedy algorithm seemingly will always return the optimal solution to this problem. It is optimal because this problem naturally deals with maximizing each sum, which lends itself to being greedy because the greedy algorithm will always take the largest element for as long as it is allowed to. This problem is different from the cow transport problem because in this case, there are no individual weight limits per set of items added, and the goal is not to transport all the eggs but simply to reach a target weight with as few eggs as possible.