Ethan Fulbright

6.0002 – Problem Set 1

A.5 – Writeup

1.

My greedy algorithm ran in 0.0 seconds (below the timing threshold) and required 6 trips, while my brute force algorithm ran in approximately 1.1 seconds and only required 5 trips. The greedy algorithm is faster because it doesn't have to take the time to check every possible solution in the problem space.

2.

The greedy algorithm does NOT return the optimal solution, because it doesn't check all possible solutions and never backtracks. This means it may return a locally optimal solution instead of a globally optimal one.

3.

The brute force algorithm does return the optimal solution, albeit slowly (especially as the input grows), because it considers ALL possible solutions and chooses the best one (i.e., the one that requires the minimum number of trips).

B.2 Writeup

1.

This problem would be very difficult using a brute force algorithm and 30 different egg weights because the problem space it would have to completely explore is enormous. The number of permutations of 30 egg weights is a prohibitively large and computationally expensive set to calculate, and would make the algorithm unusably slow.

2.

A greedy algorithm for the eggs would use the same objective function – minimizing the number of eggs needed to fill a given spaceship weight limit. The contraints would be the spaceship weight limit and the set of egg weights available. The strategy a greedy algorithm would use involves taking the largest egg which will fit within the given spaceship weight limit, and then recursing, making the same choice at each juncture (will the heaviest egg fit? If so, take it. If not, eliminate it from consideration and take the next heaviest egg).

3.

A greedy algorithm will NOT always return the optimal solution to this problem. For instance, if the egg weights available are (1, 2, 6, 12, 20) and the weight limit is 37, a greedy algorithm taking the heaviest egg it can fit will produce a minimum number of eggs equal to 5 (1 * 20 + 1 * 12 + 0 * 6 + 2 * 2 + 1 * 1 = 37). However, the optimal solution here involves NOT choosing the heaviest egg at the beginning, and results in 4 eggs (0 * 20 + 3 * 12 + 0 * 6 + 0 * 2 + 1 * 1 = 37). The greedy algorithm is incapable of finding this solution.