CS 161 Winter 2020 Section 7

February 27-28, 2020

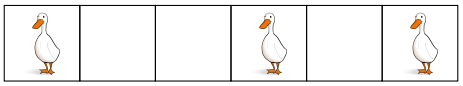
Warm-up: Greedy or Not?

Sometimes it can be tricky to tell when a greedy algorithm applies. For each problem, say whether or not the greedy solution would work for the problem. If it wouldn't work, give a counter example.

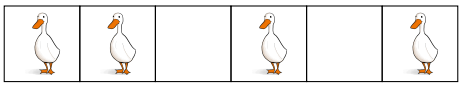
1. You have unlimited objects of different sizes, and you want to completely fill a box with as few objects as possible. (Greedy: Keep putting the largest object possible in for the space you have left)
2. You have unlimited objects, all of which are size 3^k for some integer k, and you want to completely fill a box with as few objects as possible. (Greedy: same approach as the previous problem)
3. You have lines that can fit a fixed number of characters. You want to print out a fixed series of words while using as few lines as possible. (Greedy: Fit as many words as you can on a given line)
4. You want to get from hotel 1 to hotel n, and you can travel at most k distance between hotels before collapsing from exhaustion. Each hotel has a fixed cost per night. Find the minimum cost possible to spend on hotels. (Greedy: Go as far as you can before stopping at a hotel)

Ducks in a Row

You have a dance floor made up of n squares in a line. The troupe of dancing ducks are back, and you'd like to assign ducks to squares. Different spots along the line will result in different quality of dancing: suppose that the location i will result in a dance of quality $T[i]$, where $T[i]$ is a positive integer. Further, you cannot place two ducks directly next to each other: this cramps their style.
Your goal is, given the array T, to create the best dance line-up possible, when summed over all of the n squares on the dance floor.
For example, if the input was $T = [21, 4, 6, 20, 2, 5]$, then you should place ducks in the pattern



and you would obtain dance quality $21 + 20 + 5 = 46$. You would **not** be allowed to place ducks in the pattern



because there are two ducks next to each other.
Design a dynamic programming algorithm which runs in time $O(n)$ which takes as input the array T and returns a single number which is the best total quality of dancing possible. Your algorithm does not need to output the optimal way to place the ducks.

Encoding

Suppose we encode lowercase letters into a numeric string as follows: we encode a as 1, b as 2 . . . and z as 26. Given a numeric string S of length n, develop an $O(n)$ algorithm to find how many letter strings this can correspond to. For example, for the numeric string 123, the algorithm should output 3 because the letter strings that map to this numeric string are abc, lc, and aw.

Knight Moves

Given an 8×8 chessboard and a knight that starts at position $a1$ (the letter denotes the column and the number the row), devise an algorithm that returns how many ways the knight can end up at position xy after k moves. Knights move in an "L" shape: ± 1 square in one direction and ± 2 squares in the other direction.