CSC373 Fall'19
Tutorial 3
Mon. Sept. 30, 2019

Q1 Coin change

Consider again the problem of making change when the denominations are arbitrary.
Input: Positive integer "amount" A, and positive integer "denominations" $d[1] < d[2] < ... < d[m]$.
Output: List of "coins" $c = [c[1], c[2], ..., c[n]]$ where each $c[i]$ is in d, repeated coins are allowed (possible for $c[i] = c[j]$ with $i \neq j$), $c[1]+...+c[n] = A$, and n is minimum. If no solution is possible, output $n = 0$ and an empty list c.
Example: If we only have pennies, dimes and quarters to make change for 30c, then the input is $d = [1, 10, 25]$ and an optimum output is $c = [10, 10, 10]$. If we only have nickels, dimes and quarters to make change for 52c, then an optimum output is $c = []$ – no solution exists.

Follow the dynamic programming paradigm to solve this problem.

(a) Describe the recursive structure of sub-problems.

(b) Define an array that stores optimum values for arbitrary sub-problems.

(c) Give a recurrence relation (Bellman equation) for the array values, based on the recursive structure of sub-problems.

(d) Write a simple algorithm to compute the array values bottom-up.

(e) Use the computed array values to reconstruct an optimum solution; when necessary, define a second array to store partial information about solutions and modify the algorithm from part (d) accordingly. Then, analyze the worst-case runtime of your algorithm carefully. Does it run in polynomial time? Explain.

Q2 Longest Increasing Subsequence

Consider the following Longest Increasing Subsequence (LIS) problem:
Input: $I = \langle a_1, a_2, \ldots, a_n \rangle$ an ordered sequence of n integers.
Output: An ordered subsequence S of I such that each member of S is strictly larger than all the members that have come before it, and S contains as many integers as possible.
Example: For $I = \langle 4, 1, 7, 3, 10, 2, 5, 9 \rangle$, we want $S = \langle 1, 3, 5, 9 \rangle$ or $S = \langle 1, 2, 5, 9 \rangle$. $\langle 6, 7, 8 \rangle$, $\langle 1, 2, 3 \rangle$ are not subsequences (they either include integers not in I or include integers out-of-order); $\langle 4, 1, 7, 10 \rangle$ is not increasing; $\langle 1, 3, 9 \rangle$ is not as long as possible.

In other words, the ordering of S must respect the ordering of I, S's members must be strictly increasing, and S must be as long as possible.

Write an efficient algorithm to solve the longest increasing subsequence problem. Briefly justify its correctness and runtime.