Recitation 4: Recursion and Dynamic Programming

1 Recursion

1. In class you saw how to implement binary search via recursion. Now try to implement it as an iterative algorithm instead.

2. In class you saw a recursive algorithm to compute the n-th power of real number x. This algorithm ran in $\Omega(n)$ time. Is it possible to design a recursive algorithm that runs in $O(\log(n))$ time? (You may assume that n is an integer.) If so, provide the algorithm. If not, prove that it's impossible.

3. Design a recursive algorithm which generates all the subsets of $\{1, \ldots, n\}$. It should take n as input.

2 Dynamic Programming

1. Suppose you can climb stairs either one or two steps at a time. Design an algorithm to compute the total number of distinct ways to climb a staircase with n stairs.

2. You are given an unlimited supply of coins with denominations 1, 5, 10, 25, and 100 cents. Given a non-negative integer S (in cents), design an algorithm that returns the number of distinct combinations of coins that sum to S. Two combinations that differ only in the order of the same coins are considered the same.

3. Given an array of integers A, design an algorithm to compute the longest increasing subsequence (LIS) in A. The numbers in the subsequence do not have to be adjacent in A. For example: If $A = [3, -2, 5, -4]$ the answer is 2. If $A = [12, 1, 2, -1, 10, 5, 7, -3, 12, 6]$ the LIS is $1, 2, 5, 7, 12$ so you should output 5.