Algorithms: CSE 202 — Homework 0

For each problem, provide a high-level description of your algorithm. Please make sure to include the necessary details that are crucial for its correctness and efficiency. Prove its correctness and analyze its time complexity.

Problem 1: Maximum sum among nonadjacent subsequences

Find an efficient algorithm for the following problem:

We are given an array of real numbers $V[1..n]$. We wish to find a subset of array positions, $S \subseteq [1...n]$ that maximizes $\sum_{i \in S} V[i]$ subject to no two consecutive array positions being in S. For example, say $V = [10, 14, 12, 6, 13, 4]$, the best solution is to take elements $1, 3, 5$ to get a total of $10 + 12 + 13 = 35$. If instead, we try to take the 14 in position 2, we must exclude the 10 and 12 in positions 1 and 3, leaving us with the second best choice $2, 5$ giving a total of $14 + 13 = 27$.

Problem 2: Maximum difference in an array

Given an array A of integers of length n, find the maximum value of $A(i) - A(j)$ over all choices of indexes such that $j > i$.

Problem 3: Maximum difference in a matrix

Given an $n \times n$ matrix $M[i, j]$ of integers, find the maximum value of $M[c, d] - M[a, b]$ over all choices of indexes such that both $c > a$ and $d > b$.

Problem 4: Pond sizes

You have an integer matrix representing a plot of land, where the value at a location represents the height above sea level. A value of zero indicates water. A pond is a region of water connected vertically, horizontally, or diagonally. The size of a pond is the total number of connected water cells. Write a method to compute the sizes of all ponds in the matrix.

Problem 5: Frequent elements

Design an algorithm that, given a list of n elements in an array, finds all the elements that appear more than $n/3$ times in the list. The algorithm should run in linear time. n is a nonnegative integer.

You are expected to use comparisons and achieve linear time. You may not use hashing. Neither can you use excessive space. Linear space is sufficient. Moreover, you are expected to design a deterministic algorithm. You may also not use the standard linear-time deterministic selection algorithm. Your algorithm has to be more efficient (in terms of constant factors) than the standard linear-time deterministic selection algorithm.