# COSC 3320 Practice Problems Set 1

September 27, 2023

## Problem 1

Solve the following recurrences by finding a Big-Theta form for $T(n)$. Justify your work. You may assume the existence of base cases $T(0) = T(1) = 1$. You may use the master theorem if applicable, or mathematical induction.

$$T(n) = T\left(\frac{n}{2}\right) + 5$$
$$T(n) = 7T\left(\frac{n}{2}\right) + 10n^2$$
$$T(n) = 2T(n-1) + 1$$
$$T(n) = 2T\left(\frac{n}{8}\right) + \sqrt{n}$$

## Problem 2

You are given $n$ stones (assume that $n$ is a power of 2) each having a distinct weight. You are also given a two-pan balance scale (no weights are given). For example, given two stones, you can use the scale to compare which one is lighter by placing the two stones on the two different pans. The goal is the find the heaviest and the lightest stone by using as few weighings as possible. Give a divide and conquer strategy that uses $\frac{3n}{2} - 2$ weighings. Justify algorithm correctness and the number of steps taken.

## Problem 3

You are given an $n \times n$ matrix represented as a 2-dimensional array $A[1 \ldots n][1 \ldots n]$ (there are totally $n^2$ elements). Each row of $A$ is sorted in increasing order and each column is sorted in increasing order. Your goal is to find whether some given element $x$ is in $A$ or not. Note that you can do only comparisons between elements (no hash table, or any other set data structures). Give an algorithm that takes $\mathcal{O}(n)$ comparisons.

[Hint: try to start on one some corner, and eliminate one row/column per move].

## Problem 4

You are given an array $A$ of $n$ integers, both positive and negative. The maximum possible sum of contiguous elements of the array is the maximum value of $Sum(i, j) = A[i] + A[i+1] + \ldots A[j-1] + A[j]$ over any $i, j$ satisfying $0 \leq i \leq j < n$. For example, the array $[1, -5, 7, 10, -3, 15]$ has maximum sum 29, which is represented by the subarray $[7, 10, -3, 15]$.

Define a dynamic programming algorithm which can find the maximum value $Sum(i, j)$ in $\mathcal{O}(n)$ time (note we don't need to know the $i, j$ values that make it maximum). Provide the base cases, recursive subproblems and the recurrence that defines the solutions of larger subproblems in terms of smaller ones.

[Hint: The first and most difficult step is finding out the recursive relationship of larger problems in terms of smaller ones. If we have the solution on the range $[0, i]$, it matters whether the subarray that yields the maximum sum includes $i$ or does not, when determining whether it is relevant to a solution on $[0, i+1]$. These distinctions can be solved by adding additional subproblems.]