CS3230: Assignment for Week 3

Due: Monday, 29th Aug 2022, 7 pm SGT.

Please upload **1 PDF file per question (total 3 files)** containing your solution (handwritten & scanned, or typed) by 29th Aug, 7 pm on Canvas. You may discuss the problems with your classmates, though you should write up your solutions on your own. Please note the names of your collaborators in your submission. You may want to refer to the plagiarism policy from Lecture 0.

For this module, when the phrase **'design and analyze'** is used, you are expected to provide pseudocode along with a proof of correctness and runtime analysis. Note that this problem set will be graded for correctness, not just for effort.

1. Solve the following recurrences:

   (a) $T(n) = 4T(n/2) + \Theta(n^2)$

   (b) $T(n) = T(0.6n) + T(0.3n) + O(n^2)$

   (c) $T(n) = T(\sqrt{n}) + T(n^{0.25}) + O(n)$

   Among the above recurrences, for each recurrence solvable by master theorem, please indicate which case it belongs to (either case 1, 2, or 3) with proper reasoning and state the (time) complexity. For the recurrences that are not solvable by master theorem, solve them using any of the methods taught in the class. Note that no marks will be given for answers without proof.

2. Let A be an array of n positive integers (not necessarily distinct). Given a value V, we want to find $N(A, V)$, the number of unique pairs (i, j) such that

$$1 \le i \le j \le n \text{ and } \prod_{k=i}^{j} A[k] \le V.$$

In other words, $N(A,V)$ is the number of nonempty, contiguous subarrays of A whose product is at most V. For example, if $A = [2,2,3]$ and $V = 4$, the acceptable pairs are $(i,j) = (1,1),(1,2),(2,2),(3,3)$ which gives $N(A,V) = 4$.
Design and analyze an iterative algorithm to find $N(A,V)$ in $O(n)$ time. You may assume that $n \geq 1$ and $V \geq max(A)$. Partial marks will be given for algorithms slower than $O(n)$.

3. Given an array A with n distinct elements, a pair (i,j) with $1 \leq i < j \leq n$ is called *bad* if $A[i] > A[j]$. Design and analyze an algorithm to find the number of bad pairs in an array A and runs in $O(n \lg n)$ time. For example, if $A = [3,1,2]$, your algorithm should return 2, for the bad pairs $(i,j) = (1,2),(1,3)$. Partial marks will be given for algorithms slower than $O(n \lg n)$.