CSC 226 SPRING 2022
ALGORITHMS AND DATA STRUCTURES II
ASSIGNMENT 1
UNIVERSITY OF VICTORIA

1. An n-degree polynomial $p(x)$ is an equation of the form

$$p(x) = \sum_{i=0}^{n} a_i x^i$$

where x is a real number and each a_i is a real constant, with $a_n \neq 0$.

(a) Describe a naive $O(n^2)$-time method for computing $p(x)$ for a particular value of x. Justify the runtime.

(b) Consider now the nested form of $p(x)$, written

$$p(x) = a_0 + x(a_1 + x(a_2 + x(a_3 + \ldots + x(a_{n-1} + xa_n)\ldots))).$$

Using the big-Oh notation, characterize the number of multiplications and additions this method of evaluation uses.

2. In the 3SUM problem, we are given as input an array A of n distinct integers and the goal is to check if there are three integers in A that sum to 0. Design a $O(n^2)$ running time algorithm for 3SUM. Describe your algorithm in pseudocode and show that its running time is $O(n^2)$.

3. Recall the LinearSelect algorithm we learnt in the class. Suppose that we modify the algorithm to use groups of size 3 instead of 7. Show that the modified algorithm does not run in $O(n)$ time. You may follow the lecture slide examples from lecture 3 when developing your recurrence equation $T(n)$ for this version of the algorithm. That is, you can use upper bounds to get rid of the ceiling notation and assume an in-place implementation which costs nothing to separate into subsequences. [Note: For a subsequence of 3 elements, it takes at most 3 comparisons to sort them.]

4. **The Master Theorem**

$$\begin{aligned} T(n) &= c \text{ if } n < d \\ &= aT\left(\frac{n}{b}\right) + \Theta(n^c) \text{ if } n \geq d \end{aligned}$$

(a) If $c < \log_b a$, then $T(n)$ is $\Theta(n^{\log_b a})$.

(b) If $c = \log_b a$, then $T(n)$ is $\Theta(n^c \log n)$.

(c) If $c > \log_b a$, then $T(n)$ is $\Theta(n^c)$.

Solve the following recurrence equations using the Master Theorem given above.

(a) $T(n) = 16T(n/4) + n^4$

(b) $T(n) = 125T(n/5) + n^2$

(c) $T(n) = 64T(n/8) + n^2$

5. Suppose that your goal is to come up with a new algorithm for matrix multiplication whose running time is better than the $O(n^{2:807})$ running time of Strassen's matrix multiplication algorithm.

   Your plan to achieve this by coming up with a new method for multiplying two 3×3 matrices using as few multiplications as possible. Show that in order to beat Strassen's algorithm, your method must use 21 multiplications or less. (Hint: Write down the recurrence equation for your method similar to the equation for Strassen's algorithm in the slides and use Master Theorem to solve your recurrence).