St. Francis Xavier University
Department of Computer Science

CSCI 355: Algorithm Design and Analysis
Assignment 3
Due March 16, 2023 at 1:15pm

Assignment Regulations.

- This assignment must be completed individually.
- Please include your full name and email address on your submission.
- You may either handwrite or typeset your submission. If your submission is handwritten, please ensure that the handwriting is neat and legible.

[5 marks] 1. For each of the following recurrence relations $T(n)$, give asymptotic tight bounds in terms of n. You may assume that $T(n) = \Theta(1)$ for $n < 2$. Briefly justify your answers.

(a) $T(n) = 25T(n/5) + n^2$.

(b) $T(n) = T(9n/10) + n$.

(c) $T(n) = 7T(n/2) + n^2$.

(d) $T(n) = T(n/2) + T(n/4) + T(n/8) + n$. (*Hint.* Does the master theorem apply here?)

[6 marks] 2. We say that a sorting algorithm sorts *in place* if only a constant number of elements of the input are ever stored outside of the array. (For example, storing one element in a temp variable.)

We learned in lecture that all comparison-based sorting algorithms require $\Omega(n \log(n))$ comparisons to sort an array of n elements. However, we can do better than this lower bound if we allow for non-comparison-based sorting algorithms: algorithms that don't directly compare array elements to sort.

Suppose you are given an array $A[0, \ldots, n-1]$ that contains a permutation of the first n natural numbers. Using a non-comparison-based technique, give an in-place algorithm to sort A in one pass in $O(n)$ time, and give a brief justification of why your algorithm runs in linear time.

Note. The condition that your algorithm must sort A in one pass precludes you from simply overwriting the array with all of the elements in order!

[8 marks] 3. (a) Consider the following matrices:

$$A = \begin{bmatrix} 1 & 3 \\ 7 & 5 \end{bmatrix} \text{ and } B = \begin{bmatrix} 6 & 8 \\ 4 & 2 \end{bmatrix}.$$

Using Strassen's algorithm, calculate the matrix product $C = AB$. Show all your work.

Note. If you like, you can implement Strassen's algorithm in the programming language of your choice to complete this question. If you do this, please submit your code and its output showing each step of the computation.

(b) Recall that Strassen's algorithm recursively partitions its matrices A and B into half-sized blocks. This partitioning is easy to perform when the matrices are each of size $n \times n$, where n is a power of two.

How can you modify Strassen's algorithm to find the product of two $n \times n$ matrices when n is not a power of two? Explain your modification, and show how the resulting modified algorithm would still have a time complexity of $O(n^{\log_2(7)}) \approx O(n^{2.81})$.

[6 marks] 4. Suppose you want to exchange one country's currency for another. Instead of making a direct exchange, you notice that you can gain a small advantage by making a sequence of intermediate trades through other currencies until you get the currency you want.

You can trade n different currencies, numbered C_1 through C_n. You have currency C_1 and you want to obtain currency C_n. Each currency may appear at most once in any sequence of trades. For each pair of currencies C_i and C_j, there is an exchange rate r_{ij}: if you start with d units of currency C_i, you can obtain $d \cdot r_{ij}$ units of currency C_j.

A sequence of trades sometimes requires you to pay a commission fee, which is only charged at the end of the sequence. Suppose that f_k is the commission you are charged for making a total of k trades.

(a) Prove that if $f_k = 0$ for all $k \in \{1, 2, \ldots, n\}$ (i.e., no commission is charged), then the problem of finding the best sequence of exchanges from currency C_1 to currency C_n exhibits optimal substructure and is a good candidate for dynamic programming.

Note. You do not need to give a dynamic programming algorithm. You only need to prove that the problem exhibits optimal substructure.

(b) Give a counterexample to show that if f_k is allowed to be nonzero (i.e., commission may be charged), then the same problem does not necessarily exhibit optimal substructure and is not a good candidate for dynamic programming.