St. Francis Xavier University
Department of Computer Science

CSCI 355: Algorithm Design and Analysis
Assignment 3
Due November 6, 2025 at 11:30am

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. Let $f : \mathbb{N} \to \mathbb{Z}$ be a strictly decreasing function; that is, a function where for all $i \in \mathbb{N}$, $f(i) > f(i+1)$. Assume we can evaluate f at any point i in $O(1)$ time.

Consider the *function evaluation* problem: we want to find the first point where f becomes non-positive; that is, we want to find $\min\{i \in \mathbb{N} \mid f(i) \leq 0\}$. Although this point is not given to us, we are told that it is well-defined (i.e., that there exists at least one point i where $f(i) \leq 0$).

Clearly, we can solve the function evaluation problem in $O(n)$ time by evaluating f at every point between 0 and n. Using the divide-and-conquer paradigm, design an algorithm that solves the problem in $O(\log(n))$ time. Justify the runtime of your algorithm.

[4 marks] 2. For each of the following recurrence relations, give an asymptotic tight bound in terms of n using the master theorem if possible, or otherwise indicate that the master theorem does not apply. Briefly justify your answers.

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

(b) $T(n) = 3T(n/3) + \sqrt{n}$.

(c) $T(n) = 3T(n/2) + n^2$.

(d) $T(n) = \frac{1}{2}T(n/2) + \frac{1}{n}$.

[10 marks] 3. Implement Strassen's algorithm in the programming language of your choice, and use your implementation to find the products of the following pairs of matrices.

Your answer must include a *fully commented* copy of your source code, and your implementation must produce a verbose listing of every step of the computation (including all intermediate multiplications). It must not simply output the product matrix.

(a)

$$A = \begin{bmatrix} 1 & 7 \\ 2 & 9 \end{bmatrix} \text{ and } B = \begin{bmatrix} 2 & 2 \\ 1 & 1 \end{bmatrix}.$$

(b)

$$A = \begin{bmatrix} 3 & 14 & 7 & 12 \\ 9 & 5 & 2 & 8 \\ 15 & 6 & 10 & 11 \\ 1 & 13 & 4 & 7 \end{bmatrix} \text{ and } B = \begin{bmatrix} 10 & 2 & 14 & 3 \\ 8 & 11 & 1 & 6 \\ 5 & 13 & 9 & 4 \\ 7 & 12 & 15 & 2 \end{bmatrix}.$$

(c)

$$A = \begin{bmatrix} 17 & 29 & 22 \\ 25 & 18 & 27 \\ 30 & 21 & 16 \end{bmatrix} \text{ and } B = \begin{bmatrix} 28 & 19 & 23 \\ 15 & 26 & 20 \\ 24 & 30 & 18 \end{bmatrix}.$$

[6 marks] 4. Recall the *change-making* problem that started off our study of algorithm design: given an amount of money and a set of denominations, we want to make change equal to that amount of money using the fewest number of coins. Before, we used the greedy paradigm to solve the change-making problem, but now we're more knowledgeable and sophisticated. Now, we're going to use dynamic programming to solve the problem.

To formalize the problem, suppose we are given as input a positive integer A and a set of m positive integer denominations $d[1] < d[2] < \cdots < d[m]$. We want to produce a list of coins $c[1..n]$ such that $\sum_{i=1}^{n} c[i] = A$, where each $c[i]$ is in d and where it is possible that $c[i] = c[j]$ for $i \neq j$ (i.e., repeated coins are allowed).

(a) Define an array $C[i, j]$ to store the information needed to compute the appropriate subproblems of an instance of the change-making problem. Explain what each entry in your array means, and give the entry of the array that will contain the overall solution to the instance of the problem.

(b) Write a Bellman equation describing the structure of the change-making problem, explain each case of your Bellman equation, and informally justify its correctness. (I.e., explain why each case is needed.)

Hint. Three cases suffice.