CSCE 411: Design and Analysis of Algorithms, Fall 2025

Test 1B, Section 500

Name: _____

Instructions: DO NOT OPEN THIS EXAM UNTIL YOU ARE TOLD TO.

Write your name at the top of this page. Read the explanation of the test format, and the academic integrity statement, and sign at the bottom of the page. WHEN TAKING THE EXAM, WRITE YOUR INITIALS AT THE TOP OF EACH PAGE.

Test Format: The test consists of

- 11 multiple choice problems (5 points each)
- 3 free response problems

The total number of points on the exam is 100.

Academic Integrity: On my honor, I will complete the exam without giving or receiving help from anyone else, and without consulting or using any resources other than *my own* single note sheet that I am allowed as a part of the exam.

Signed: _____

Multiple Choice Questions

Instructions: Circle a single answer for each multiple choice problem.

1. (5 points) Let A be a $p \times q$ matrix and B be a $q \times r$ matrix. Using the simple definition of matrix multiplication, what is the runtime complexity of computing AB?

   (a) $\Theta(p + q + r)$

   (b) $\Theta(pq + qr)$

   (c) $\Theta(pq + qr + rp)$

   (d) $\Theta(pqr)$

2. (5 points) Which of the following sets of codeword lengths for symbols $\{a, b, c, d\}$ could correspond to an **optimal Huffman code** for some distribution of symbol frequencies?

   (a) $\{a : 1,\ b : 1,\ c : 2,\ d : 3\}$

   (b) $\{a : 1,\ b : 2,\ c : 2,\ d : 3\}$

   (c) $\{a : 1,\ b : 1,\ c : 1,\ d : 3\}$

   (d) $\{a : 2,\ b : 2,\ c : 2,\ d : 2\}$

   (e) $\{a : 1,\ b : 3,\ c : 3,\ d : 3\}$

3. (5 points) In the **potential method** of amortized analysis, which of the following best describes how the amortized cost of an operation is computed?

   (a) It is the actual cost of the operation plus the change in a potential function that reflects the data structure's stored work.

   (b) It is the average of the actual costs of all operations performed so far.

   (c) It is the maximum possible cost of any single operation in the sequence.

   (d) It is the difference between the actual cost and the total cost of all previous operations.

   (e) It is the total cost of all operations divided by the total number of future operations.

4. (5 points) The greedy approach always gives the optimal solution for which of the following problems?

   (a) 0/1 Knapsack Problem

   (b) Activity Selection Problem

   (c) Change-Making Problem

   (d) Matrix Chain Multiplication Problem

   (e) Rod Cutting Problem

5. (5 points) What is the time complexity $T(n)$ of a procedure that satisfies the base cases $T(2) = T(1) = \Theta(1)$ and the recurrence $T(n) = T(\sqrt{n}) + \Theta(n)$? You may assume $\log \log n$ is an integer.

   (a) $\Theta(n^2)$

   (b) $\Theta(n\sqrt{n})$

   (c) $\Theta(n)$

   (d) $\Theta(\sqrt{n})$

   (e) $\Theta(\log n)$

6. (5 points) In the **accounting method** for amortized analysis, which of the following correctly describes how amortized costs are assigned to operations?

   (a) Each operation is charged an amortized cost that may be higher than its actual cost, and the extra "credit" is stored to pay for future expensive operations.

   (b) Each operation is charged exactly its actual cost, ensuring the total amortized cost equals the total actual cost.

   (c) Each operation is charged less than its actual cost, and the deficit is covered by borrowing from future operations.

   (d) Each operation is assigned a uniform amortized cost equal to the average cost across all possible inputs.

   (e) Each operation is assigned a cost that depends only on the number of previous operations, not their cost.

7. (5 points) Suppose you develop an optimized method to multiply 3×3 matrices using exactly 18 multiplications and 24 additions. Using this method, you design a divide-and-conquer algorithm for multiplying two $n \times n$ matrices where $n = 3^p$ for $p \in \mathbb{N}$, by dividing the matrices into square 3×3 blocks. What would the runtime $T(n)$ would the corresponding recurrence relationship give?

   (a) $T(n) = \Theta(n^2)$
   (b) $T(n) = \Theta(n^{\log_3 18})$
   (c) $T(n) = \Theta(n^{\log_3 24})$
   (d) $T(n) = \Theta(n^{\log_9 18})$
   (e) $T(n) = \Theta(n^{\log_9 24})$

8. (5 points) In the schedule attendance problem, you must assign each of n football games to either home or away. For each $i \in [n]$, the attendance for the i-th home game is H_i and the attendance for the i-th away game is A_i. You must design a schedule that maximizes the attendance, but you cannot schedule two consecutive home games in a row. Which of the following is a recurrence for $\text{OPT}[j]$, the maximum possible attendance for the first j games, assuming $j > 1$?

   (a) $\text{OPT}[j] = \max\big(A_j + \text{OPT}[j-1],\, H_j + \text{OPT}[j-2]\big)$
   (b) $\text{OPT}[j] = \max(A_j, H_j) + \text{OPT}[j-1]$
   (c) $\text{OPT}[j] = \max_{i<j} A_i + H_i + \text{OPT}[i]$
   (d) $\text{OPT}[j] = A_j + H_j$
   (e) $\text{OPT}[j] = \text{OPT}[j-1] + A_j + H_j$

9. (5 points) Which of the following is NOT a fundamental step in the divide and conquer paradigm?

   (a) Dividing the problem into subproblems
   (b) Recursively solving the subproblems
   (c) Merging the solutions of the subproblems
   (d) Making a locally optimal choice at each step
   (e) All of the above are fundamental steps

10. (5 points) You are given a number of matrices $A_1, \ldots, A_n$ and your goal is to minimize the number of scalar multiplications used to compute the matrix product $A_1 A_2 \cdot \ldots \cdot A_n$. Which algorithmic approach is best suited to solve this problem?

    (a) Dynamic programming
    (b) Greedy algorithm
    (c) Divide and conquer
    (d) Binary search
    (e) Prefix sum

11. (5 points) Consider a stack with a global counter k, initially set to 1, and the following operations:

    - MULTIPUSH — pushes exactly k items $x_1, \ldots, x_k$ onto the stack and resets k to 1 (takes $O(k)$ time),
    - INCREMENT() — increments the global counter k by 1 (takes $O(1)$ time),
    - POP() — pops one item from the stack (takes $O(1)$ time).

    Starting from an empty stack, consider a sequence of n operations of these three types. What is the best upper bound on the total time required for the whole sequence?

    (a) $O(n^2)$
    (b) $O(n \log n)$
    (c) $O(n)$
    (d) $O(\sqrt{n})$
    (e) $O(\log n)$

Long Answer Questions

1. (15 points total) You are climbing the Zachry staircase, which has n steps. You can climb either 1, 2, or 3 steps at a time (if you really stretch your legs). Your task is to determine the number of distinct ways to reach the top of the staircase.

   Example: If $n = 4$, the distinct ways to reach the top are:

   1. $1 + 1 + 1 + 1$
   2. $1 + 1 + 2$
   3. $1 + 2 + 1$
   4. $2 + 1 + 1$
   5. $2 + 2$
   6. $1 + 3$
   7. $3 + 1$

   Thus, the total number of ways is 7.

   (a) (5 points) Let $S(n)$ be the number of ways to reach the n-th step. Write down the recurrence relation for $S(n)$.

   (b) (5 points) What is the runtime of an asymptotically optimal dynamic programming algorithm to compute $S(n)$ for a given $n \geq 0$?

   (c) (5 points) How many different ways are there to reach the 10-th step?

2. (10 points) An alchemist has n different types of vials for storing potions. For $i \in [n]$, the i-th type of vial stores $v_i \geq 1$ liters of potions and the alchemist has an infinite number of vials of each type. After brewing a new potion, the alchemist would like to store the potion in the fewest vials possible, and without losing any of the potion. However, the Alchemists' Guild requires that a vial must be completely full before another vial can be used and moreover, there exists a vial type that stores 1 liter of potions.

   The neighboring wizard proposes the following greedy algorithm for storing the potion of an integer volume $V \geq 1$ in the fewest possible number of vials:

   > While $V > 0$:
   >
   > 1. Pick the vial j with the largest volume v_j such that $v_j \leq V$.
   > 2. If no such j exists and $V > 0$, output IMPOSSIBLE.
   > 3. Pour v_j liters of the potion into the vial and update the remaining volume to be $V \leftarrow V - v_j$

   Prove or disprove the correctness of the suggested greedy algorithm for storing the potion of volume V in the fewest possible number of vials.

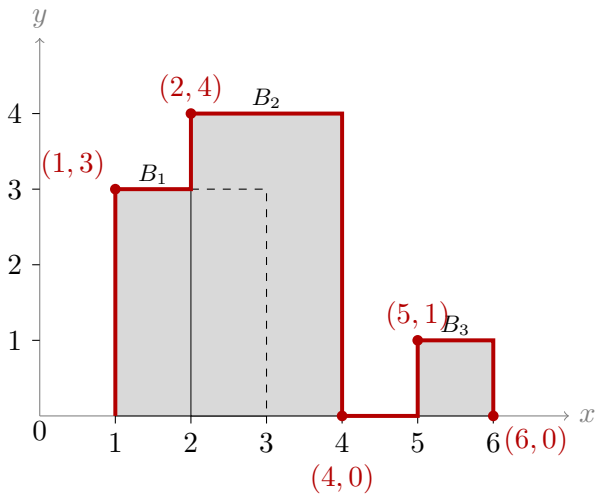
3. (20 points) **The skyline problem.** You are given a set of n buildings on a city skyline. Each building is represented as a triplet (L, R, H) where:

   - L is the x-coordinate of the left edge of the building,
   - R is the x-coordinate of the right edge of the building, and
   - H is the height of the building.

   The skyline is defined as the outer contour formed by the union of all buildings. The skyline should be represented as a list of "key points" (x, y) that capture the corners of the outer contour, sorted by x-coordinate.

   Example: Given the buildings $\{(1, 3, 3), (2, 4, 4), (5, 6, 1)\}$, the skyline key points are:

   $$(1, 3),\ (2, 4),\ (4, 0),\ (5, 1),\ (6, 0)$$



   (a) (5 points) Given the buildings $\{(1, 5, 11), (2, 7, 6), (3, 9, 13), (12, 16, 7), (14, 25, 3)\}$, compute the skyline key points.

(b) (10 points) Describe a **divide-and-conquer** algorithm to compute the skyline for n buildings. Include a description of how the buildings are split, how the subproblems are solved, and how the results are merged.

(c) (5 points) Let $T(n)$ be the runtime of your divide-and-conquer algorithm. Write a recurrence for $T(n)$ and solve it asymptotically.

Divide-and-conquer. 3 steps: (1) Divide, (2) Conquer, and (3) Combine. Runtimes satisfy a recurrence relationship, which can be turned into an actual runtime using a few different techniques.

Dynamic programming. For problems with optimal substructure and overlapping subproblems. Solves overlapping subproblems only once each using either a top-down or a bottom-up approach.

Greedy algorithms. Make a decision at each step that is "locally" the best choice. Proving a greedy algorithm is optimal for a certain optimization problem typically involves proving that making a greedy choice at the first step is "safe."

Accounting and potential method. Similarity: both store "credit" and "pay" ahead of time. The accounting method stores credit in individual steps. The potential method stores credit as "potential" in a data structure D_i. For accounting method, define $\hat{c}_i$ and prove that $\sum_i c_i \leq \sum_i \hat{c}_i$. For potential method, choose D_i and potential function Φ, and then $\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1})$ is given; proving $\Phi(D_i) \geq \Phi(D_0)$ guarantees $\sum_i c_i \leq \sum_i \hat{c}_i$. For both accounting and potential, must bound $\sum_i \hat{c}_i$ to prove runtime guarantee.

Master theorem. Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on the nonnegative integers by the relation $T(n) = aT(n/b) + f(n)$.

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.
2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \log n)$.
3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$.

Strassen's algorithm. Relies on knowing how to multiply 2×2 matrices using a small number of additions and 7 multiplications. Uses this to multiply two matrices of size $n \times n$ (where $n =$ power of 2) by breaking them into blocks and recursively calling a matrix-matrix multiplication function 7 times. Has recurrence relationships $T(n) = 7T(n/2) + \Theta(n^2)$.

Matrix multiplication problem. For $i = 1, 2, \ldots n$, let A_i be a matrix of size $p_{i-1} \times p_i$. Find the way to parenthesize the matrix chain $A_1 A_2 \cdots A_n$ so that the total computational cost is minimized. It takes $\Theta(pqr)$ operations to multiply AB if A has size $p \times q$ and B has size $q \times r$.

The activity selection problem. Let $(a_1, a_2, a_3, \ldots a_n)$ be activities with distinct start and finish times (s_i, f_i) for $i = 1, 2, \ldots, n$, ordered so that $f_1 < f_2 < \ldots < f_n$. Find the largest set of non-overlapping activities.

Multipop stack. Has operations PUSH(S, x) (pushes x onto S), POP(S) (pops top element off), and MULTIPOP(S, k) (pops $\min\{|S|, k\}$ elements). Running n total operations always has $O(n)$ runtime; key idea is that you can't pop an element until you've pushed it.

The optimal prefix code problem. Given an alphabet C and a frequency $c.freq$ for each $c \in C$ in a given string s, find the prefix code that represents s using a minimum number of bits. A prefix code associated each character with a binary codeword, such that the codeword for one character is never the start of a codeword for another character. A prefix code can be associated with a binary tree in which each character is associated with a leaf of the tree.

Binary counter. Stores an integer in binary using a $\{0,1\}$ array A. Incrementing A updates the binary number stored in A to represent the next integer. Cost is given in terms of number of bits flipped. $A = [0101]$ represents $0 \times 2^0 + 1 \times 2^1 + 0 \times 2^2 + 1 \times 2^3 = 2 + 8 = 10$.

BottomUpMatrixChainMultiplication($\mathbf{p} = [p_0, p_1, \ldots, p_n]$)
  $n = \text{length}(\mathbf{p}) - 1$
  Let $m[1 \ldots n][1 \ldots n]$ be an empty $n \times n$ array
  for $i = 1$ to n **do**
    $m[i, i] = 0$
  end for
  for $\ell = 2$ to $n - 1$ **do**
    for $i = 1$ to $n - \ell - 1$ **do**
      $j = i + \ell - 1$
      $m[i, j] = \infty$
      for $k = i$ to $j - 1$ **do**
        $q = m[i, k] + m[k + 1, j] + p_{i-1} p_k p_j$
        if $q < m[i, j]$ **then**
          $m[i, j] = q$
        end if
      end for
    end for
  end for
  Return $m[1, n]$

GreedyCoinChange(C, $v = [v_1, v_2, \ldots, v_n = 1]$)
  $n = \text{length}(v)$
  Let $m = [0..0]$ be an empty array of n zeros
  for $i = 1$ to n **do**
    while $C \geq v_i$ **do**
      $m[i] = m[i] + 1$
      $C = C - v_i$
    end while
  end for
  Return m

Assorted Reminders

- $\sum_{i=0}^{n} \frac{1}{2^i} \leq 2$
- $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$