CSCE 411: Design and Analysis of Algorithms, Fall 2024

Test 1

Name: _____

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

Write your name at 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

- 12 multiple choice problems (5 points each)
- 3 free answer questions

The total number of points on the exam is 100.

Academic Integrity: On my honor, as an Aggie, 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) In the context of the divide and conquer paradigm, which of the following is the correct recurrence relation for the time complexity of Merge Sort?

   (a) $T(n) = 2T(n/2) + O(n)$
   (b) $T(n) = T(n-1) + O(n)$
   (c) $T(n) = 3T(n/3) + O(n)$
   (d) $T(n) = T(n/2) + O(n)$
   (e) $T(n) = T(n/2) + O(\log n)$

2. (5 points) What is the time complexity $T(n)$ of a procedure that satisfies the base case $T(1) = O(1)$ and the recurrence $T(n) = T(n-1) + O(1)$?

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

3. (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

4. (5 points) In the context of amortized analysis, which of the following best describes the purpose of charging a higher amortized cost to inexpensive operations?

   (a) To account for more expensive operations that may occur later.

   (b) To find the worst-case cost of the current operation.

   (c) To improve the actual running time of future operations.

   (d) To reduce the total time complexity of the algorithm.

   (e) To ensure all operations have the same running time.

5. (5 points) What is the key idea behind the greedy algorithm approach?

   (a) Solving each subproblem optimally and combining them for the overall solution.

   (b) Making a sequence of choices, each of which looks best at the moment.

   (c) Breaking the problem into smaller, independent subproblems.

   (d) Solving the problem using brute force.

   (e) Using dynamic programming to find the global optimum.

6. (5 points) In dynamic programming, the term "memoization" refers to:

   (a) Storing the solutions to subproblems to avoid redundant work.

   (b) Breaking the problem into independent subproblems.

   (c) Recursively solving subproblems.

   (d) To use brute force to solve the problem.

   (e) Ensuring that all subproblems are solved in polynomial time.

7. (5 points) Suppose X is a string of length n and Y is a string of length m. The runtime of solving the Longest Common Subsequence (LCS) problem on X and Y using dynamic programming is:

   (a) $\Theta(m+n)$

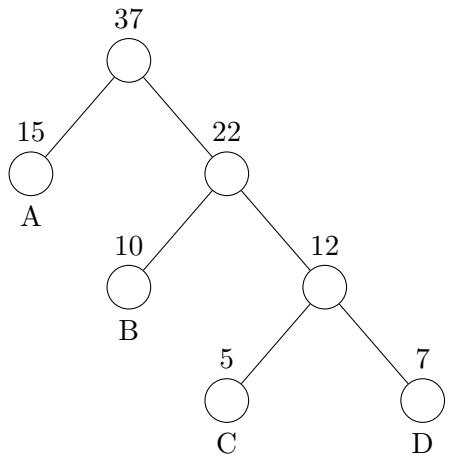
   (b) $\Theta(mn)$

   (c) $\Theta(m^2n^2)$

   (d) $\Theta(2^m+2^n)$

   (e) $\Theta(2^{m+n})$

8. (5 points) Consider the following alphabet $\{A, B, C, D\}$ and the prefix code tree shown below.



For which frequency values below would the above tree be an optimal prefix tree?

(a) `A.freq=0.10`, `B.freq=0.10`, `C.freq=0.40`, `D.freq=0.40`

(b) `A.freq=0.25`, `B.freq=0.25`, `C.freq=0.25`, `D.freq=0.25`

(c) `A.freq=0.30`, `B.freq=0.30`, `C.freq=0.25`, `D.freq=0.15`

(d) `A.freq=0.40`, `B.freq=0.10`, `C.freq=0.10`, `D.freq=0.40`

(e) `A.freq=0.80`, `B.freq=0.10`, `C.freq=0.05`, `D.freq=0.05`

Cheat sheet notes: In the the optimal prefix code problem, the input is an alphabet C and a frequency `c.freq` for each $c \in C$ in a given string s. The goal is to find the prefix code that represents s using the 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.

9. (5 points) In the activity selection problem studied in class, where the goal is to maximize the number of activities, the greedy algorithm selects activities based on:

(a) Shortest duration of the activity

(b) Earliest starting time of the activity

(c) Latest finishing time of the activity

(d) Minimum overlap of activities

(e) Earliest finishing time of the activity

Cheat sheet notes: In the activity selection problem, the input is activities $(a_1, a_2, \ldots, a_n)$ with distinct start and finish times (s_i, f_i) for $i = 1, 2, \ldots, n$, ordered so that $f_1 < f_2 < \ldots < f_n$. The task is to find the largest set of non-overlapping activities.

10. (5 points) How does Strassen's algorithm improve upon the standard matrix multiplication algorithm?

    (a) It reduces the number of matrix additions.

    (b) It reduces the number of matrix multiplications.

    (c) It increases the space complexity to achieve a faster time complexity.

    (d) It uses dynamic programming to store intermediate results.

    (e) It reduces the time complexity of matrix multiplication to $O(n)$.

11. (5 points) In the recurrence relation $T(n) = aT\left(\frac{n}{b}\right) + f(n)$ for the Master Theorem, what does the term $aT\left(\frac{n}{b}\right)$ represent?

    (a) The cost of dividing the problem into subproblems

    (b) The cost of conquering or combining the solutions of the subproblems

    (c) The cost of solving a subproblems of size $\frac{n}{b}$

    (d) The overall time complexity of the algorithm

    (e) The number of recursive calls made by the algorithm

    Cheat sheet notes: 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 $T(n) = aT\left(\frac{n}{b}\right) + f(n)$. The Master Theorem describes the runtime $T(n)$ based on casework comparing $f(n)$ and $n^{\log_b a}$.

12. (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) $\Theta(pqr)$

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.

   Cheat sheet notes: Dynamic programming is for problems with optimal substructure and overlapping subproblems. It solves overlapping suproblems only once each using either a top-down or a bottom-up approach.

   (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) (10 points) Write an asymptotically optimal dynamic programming algorithm to compute $S(n)$, the number of ways to reach the n-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.

   The neighboring wizard proposes the following greedy algorithm for storing the potion of an integer volume V 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.

   Cheat sheet notes: 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".

   HINT: You may assume V is a non-negative integer and that there exists a vial i with $v_i = 1$.

3. (15 points total) We would like to allocate memory for an array of items, for an incoming sequence of insertion operations. However, we do not know the number of items in advance, so we use the following "dynamic resizing" strategy.

   We initialize $i = 0$ and create an array A_i with 2^i slots. We then start filling A_i with items that are inserted.

   Each time A_i is full, we create a new array A_{i+1} with 2^{i+1} memory slots. We manually copy each item in A_i into A_{i+1}, so that A_{i+1} contains all the items that have previously been inserted. Note that increases the runtime by 2^i time. We then continue filling A_{i+1} with items until it also becomes full, at which point we increment i and proceed as above.

   Example: The following illustrates the state of the dynamic array after a possible sequence of $n = 8$ insertion operations:

   1. Initial state. Dynamic array: $\{-\}$
   2. Insert 2. Dynamic array: $\{2, -\}$
   3. Insert 1. Dynamic array: $\{2, 1, -, -\}$
   4. Insert 7. Dynamic array: $\{2, 1, 7, -\}$
   5. Insert 6. Dynamic array: $\{2, 1, 7, 6, -, -, -, -\}$
   6. Insert 3. Dynamic array: $\{2, 1, 7, 6, 3, -, -, -\}$
   7. Insert 1. Dynamic array: $\{2, 1, 7, 6, 3, 1, -, -\}$
   8. Insert 2. Dynamic array: $\{2, 1, 7, 6, 3, 1, 2, -\}$
   9. Insert 1. Dynamic array: $\{2, 1, 7, 6, 3, 1, 2, 1, -, -, -, -, -, -, -, -\}$

   Cheat sheet notes: Accounting and potential method 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 $\widehat{C_i}$ and prove that $\sum_i c_i \leq \sum_i \widehat{c_i}$. For potential method, choose D_i and potential function Φ, and then $\widehat{c_i} = c_i + \Phi(D_i) - \Phi(D_{i-1})$ is given; proving $\Phi(D_i) \geq \Phi(D_0)$ guarantees $\sum_i c_i \leq \sum_i \widehat{c_i}$. For both accounting and potential, must bound $\sum_i \widehat{c_i}$ to prove runtime guarantee.

   (a) (5 points) Suppose writing an item to an empty slot in an array uses $O(1)$ time. Given a sequence of n insertion operations, what is the worst-case runtime for an insertion operation for the above dynamic resizing array?

(b) (10 points) Suppose writing an item to an empty slot in an array uses $O(1)$ time. Given a sequence of n insertion operations, what is the overall TOTAL runtime for the sequence of n insertion operations. Justify your answer.

HINT: You may use the fact that $\sum_{i=1}^{\infty} i \cdot \frac{n}{2^i} = 1 \cdot \frac{n}{2} + 2 \cdot \frac{n}{4} + 3 \cdot \frac{n}{8} + 4 \cdot \frac{n}{16} + \ldots = O(n)$.