CSCE 411: Design and Analysis of Algorithms, Spring 2025

Test 1, Section 501

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: _____

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 suproblems 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

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