

Comprehensive Exam in Advanced Algorithms : Fall 2003

1. Determine which of the following possibilities is true for the following functions: (i) $f = o(g)$ (ii) $g = o(f)$ (iii) $f = \Theta(g)$. Justify your answer mathematically. If the base of a logarithm is not explicitly written, you may assume it is 2.

- (a) $f(n) = (\log n)^{100}$ and $g(n) = n^{.001}$
(b) $f(n) = \sum_{i=1}^n \frac{1}{i}$ and $g(n) = n \log n$
(c) $f(n) = \log_5 \frac{n}{7}$ and $g(n) = 10 \log 5n$
(d) $f(n) = 2^{((\log n)^2)}$ and $g(n) = n^{\log \sqrt{n}}$

2. **The 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 recurrence $T(n) = aT(n/b) + f(n)$ where we interpret n/b to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then $T(n)$ can be bounded asymptotically as follows. 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 $T(n) = \Theta(n^{\log_b a} \log n)$. 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 $T(n) = \Theta(f(n))$.

The Maximum Subsequence Sum Problem (MSS) is defined as the problem of finding the maximum subsequence sum of an array $a[]$ of integers. In other words, determining the value of

$$\max_{0 \leq i \leq j < n} \left\{ \sum_{k=i}^j a[k] \right\}$$

For example, the MSS of $\{-7, 3, -2, 1, 2, -5, 4\}$ is $3 + (-2) + 1 + 2 = 4$. (Another way to get 4 is by just taking the final element of the array by itself.)

Consider the following algorithm for solving MSS. If $a[]$ has one element, then that element is the MSS. Otherwise, partition $a[]$ into two (roughly) equal sized subsequences $a_L[]$ and $a_R[]$ corresponding to the left and right halves of the array. Recursively find the MSS of a_L and a_R . Call them *leftMax* and *rightMax*. Compare these values with the largest sum, call it *midMax*, involving terms from both a_L and a_R that include the middle element and return the largest of the three values.

For example, if $a[] = \{-7, 3, -2, 1, 2, -5, 4\}$, then $a_L[] = \{-7, 3, -2, 1\}$ and $a_R[] = \{2, -5, 4\}$. Moreover, *leftMax* = 2, *rightMax* = 2, and *midMax* = 4. Use an appropriate method to determine the asymptotic complexity of this algorithm.

3. Professor Stewart is consulting for the president of a corporation that is planning a company party. The company has a hierarchical structure; that is, the supervisor relation forms a tree rooted at the president. The personnel office has ranked each employee with a conviviality rating, which is a real number. In order to make the party fun for all attendees, the president does not want both an employee and his or her immediate supervisor to attend. The goal is to maximize the sum of the conviviality ratings of the employees attending the party.
- (a) Consider the following structure. Each employee (including the president) has three subordinates numbered 1 through 3. There are 4 levels in the tree (where level 1 is the president). The conviviality rating of an employee is equal to the maximum of (i) twice his superior's conviviality rating minus his own number and (ii) 1. The president has conviviality rating 2. Determine the optimal guest list for this structure.
 - (b) Write down an equation relating the total conviviality rating of a tree to the total conviviality ratings of its subtrees.
 - (c) Describe a dynamic programming algorithm which solves this problem.

4. In the subset-sum problem (SSP), we are given a finite set S of positive integers and a target t . We ask whether there is a subset $S' \subseteq S$ whose elements sum to t . For example, if

$$S = \{1, 2, 7, 14, 49, 98, 343, 686, 2409, 16808, 17206, 117705, 117993\}$$

and $t = 138457$, then the subset $S' = \{1, 2, 7, 98, 343, 686, 2409, 17206, 117705\}$ is a solution. You may use the fact that the subset-sum problem is NP-complete.

The set-partition problem (SPP) takes as input a set S of positive integers. The question is whether the numbers can be partitioned into two sets A and $S - A$ such that $\sum_{x \in A} x = \sum_{x \in S-A} x$. Show that the set-partition problem is NP-complete.