

Comprehensive Exam on Data Structures and Algorithms: Fall 2005

Instructions: Choose any four of the following six questions to answer. Clearly mark which question you want graded on the front of the exam by placing an X in the appropriate slot.

Problem	1	2	3	4	5	6
Graded?						

Note that you will be graded not just on the answers, but also on work. An answer alone will not be worth much if anything at all. No books, calculators, notes, talking, etc. Using any communications device at all (cell phones, walkie talkies, etc.) will be an automatic failing grade.

WARNING: If you do not mark the questions you want graded or mark them ambiguously (i.e. if you decide to mark 5 questions rather than 4, etc.), we will grade your lowest 4 questions.

1. Binary Search Trees and Dynamic Programming. Let $m_1 < m_2 < \dots < m_n$ be distinct integers that are to be stored in a binary search tree. For all $1 \leq i \leq n$, let positive integer weight w_i represent the frequency in which m_i needs to be accessed (i.e., the greater w_i , the more frequently m_i is accessed in the tree). The weighted access cost (WAC) of a binary search tree that stores these integers is defined as

$$WAC = \sum_{i=1}^n w_i d_i,$$

where d_i is the depth of m_i in the tree. Note: the root of the tree is defined to have depth 1, while the depth of a child is one more than the depth of its parent. Using dynamic programming, one can find the binary search tree that minimizes the WAC. Indeed, let $WAC(i, j)$ denote the minimum weighted access cost for a bst that stores integers m_i, m_{i+1}, \dots, m_j . Then the following dynamic programming recurrence relation is valid.

$$WAC(i, j) = \begin{cases} 0 & \text{if } i > j \\ w_i & \text{if } i = j \\ \min_{i \leq k \leq j} \{ \sum_{r=i}^j w_r + WAC(i, k-1) + WAC(k+1, j) \} & \text{if } i < j. \end{cases}$$

- (a) Given the integers and weights

integer	1	4	7	11
weight	20	5	40	10

, compute dynamic programming tables for $WAC(i, j)$ and $k(i, j)$, $1 \leq i, j \leq 4$, where $k(i, j)$ is undefined for $i \geq j$, and $k(i, j)$ equals the value of k that minimizes $WAC(i, j)$ in the above recurrence relation when $i < j$.

- (b) Use the tables $WAC(i, j)$ and $k(i, j)$ to construct a binary search tree of minimum weighted access cost for the given integers and weights. Hint: $k(i, j)$ determines which of the $j - i + 1$ integers will be stored at the root of the subtree that stores integers m_i through m_j .

2. Master Method.

```
int algorithm(int a[], int x, int left, int right){
    if(right-left <= 1) {
        if(x == a[left]) return left;
        if(x == a[right]) return right;
        return UNDEFINED;
    }
    int mid = (left+right)/2;
    if(x == a[mid]) return mid;
    if(x > a[mid]) return algorithm(a,x,mid+1,right);
    return algorithm(a,x,left, mid-1);
}
```

- (a) For the algorithm provided above, provide a recurrence relation of the form

$$S(n) = aS(n/b) + f(n),$$

where $a, b > 0$ are integers, and $S(n)$ is the worst-case number of steps required by the algorithm on inputs $a[], x, left = 0$, and $right = n - 1$, where n is the size of array a .

- (b) Use the Master Method to find an asymptotic solution to the recurrence relation from Part (a). Hint: the Master Theorem states the following.
- i. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $S(n) = \Theta(n^{\log_b a})$
 - ii. If $f(n) = \Theta(n^{\log_b a})$, then $S(n) = \Theta(n^{\log_b a} \cdot \log n)$
 - iii. 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 $c < 1$, then $S(n) = \Theta(f(n))$.

3. NP-completeness. Given a simple graph $G = (V, E)$ and a nonnegative integer K , the Vertex Cover decision problem is to decide if G has K edges e_1, e_2, \dots, e_K for which every vertex $v \in V$ is incident with at least one of the K edges. It is known that Vertex Cover is NP-Complete.

Given a collection of m -bit binary strings s_1, s_2, \dots, s_n , and a nonnegative integer K , the String Sum decision problem is to decide if there are K strings $s_{i_1}, s_{i_2}, \dots, s_{i_K}$ from the collection such that

$$s_{i_1} + s_{i_2} + \dots + s_{i_K} = \mathbf{1},$$

where the addition is bitwise Boolean-algebra addition (e.g. $1001 + 1010 = 1011$), and $\mathbf{1}$ is the m -bit binary string consisting of all ones. Prove that String Sum is an NP-complete decision problem.

4. Sorting.

- (a) Demonstrate the radix-sort algorithm on the array of binary numbers

110101, 011000, 111001, 010010, 001100, 101010, 000001, 111000, 001110, 101010.

- (b) Explain why radix sort is a linear-time algorithm for sorting an array of n binary numbers, each consisting of m bits. Hint: you are to assume that one bit contributes one to the input size.

5. Binary Search Trees.

- (a) An AVL tree is a binary search tree that is balanced, in that, for each tree node n , the left and right subtrees of n have a height difference that does not exceed one. Let $S(h)$, $h = 0, 1, 2, \dots$, denote the size of the smallest AVL tree of height h . For example, $S(0) = 1$ and $S(1) = 2$. Show that for $h \geq 2$, $S(h) = S(h-1) + S(h-2) + 1$. Hint: for an AVL tree of height h and of minimum size, consider the left and right subtrees of its root. What can you say about these subtrees?

- (b) Use Part (a) to show that an AVL tree with n nodes must have $O(\log n)$ depth. Hint: you may assume that the solution f_n to the Fibonacci recurrence equation $f_n = f_{n-1} + f_{n-2}$, $f_0 = 0$, $f_1 = 1$, has the property that $f_n = \Omega(1.1^n)$.

6. Heaps.

- (a) Show the result of inserting 8, 9, 9, 8, 7, 6, 5, 4, 2, 3 into an initially empty binary heap. Show a sequence of heaps leading to the final heap. Hint: you are to assume that higher priority is given to integers of smaller size.

- (b) Prove that inserting n items into an initially empty binary heap can be accomplished in worst-case $O(n \log n)$ steps.