

Homework 3

February 18, 2003

Due Date: February 25

This homework will be a bit shorter than the first two homeworks so that you also have time to review homeworks 1 and 2 in preparation for the exam on February 27. Thus this homework is worth 75 versus 100 points.

1. (25 points) In this problem you are to use the **decision tree lower bound technique** for proving lower bounds for the following two *problems*. Be sure to very clearly explain anything in your proof that is specific to the problem.

- (a) Prove the best lower bound you can (using the decision tree technique) on the time complexity of the problem of computing which elements in an *UNSORTED* array A are less than a given element x under the model of computation in which you can only access A by asking if $A[i] < A[j]$ or $A[i] < x$, for i and j integers from 0 to $n - 1$.

- (b) Prove the best lower bound you can (using the decision tree technique) on the time complexity of a comparison based algorithm for the following problem: You are given a sorted array A (of n elements) and two elements x and y where $x \leq y$. The algorithm is required to compute how many elements in A are less than both x and y , how many elements of A are between x and y , and how many elements of A are bigger than both x and y .

2. (10 pts) Show the execution of radix sort on the following example. Each letter should be used as a radix sort digit. (The final output should be sorted in increasing alphabetical order.) You need just show the array after each pass of counting sort.

ACB

DFA

FKL

DCL

FCA

DKA

AKB

3. (15 pts) Suppose you are given the task to sort one thousand numbers between 0 and $10^{12} - 1$ (i.e. the keys are 12 digit numbers). You have decided to use radix sort but need to decide how many digits to group for each radix sort digit. Which is best among having 1 digit per radix sort digit, 3 digits per radix sort digit, or 6 digits per radix sort digit? You are provided with a counting sort procedure with exact time complexity of $5n + 4k$. Show your work.
4. (25 pts) Let L_1, \dots, L_r be r unsorted lists, whose elements hold integers in the range $[0, k - 1]$. Let n be the total number of elements among all of the lists. That is, if n_i is the number of elements in L_i , then $n = n_1 + n_2 + \dots + n_r$. Describe an algorithm with **total worst-case** asymptotic time complexity of $O(r + k + n)$ for getting all of the r lists into sorted order. Be sure to analyze the time complexity of your algorithm.

If you cannot think of an algorithm with the specified time complexity, then for partial credit describe the most efficient algorithm you can and correctly analyze its time complexity.

Challenge Problem:

Only work on this after you have completed the required problems. Note that you can obtain 100% without doing this.

1. (5 points) Prove that any comparison-based algorithm for constructing a binary search tree from an arbitrary list of n elements takes $\Omega(n \log n)$ time in the worst case.

Hint: Think about reducing the problem of sorting to performing a set of operations on a binary search tree. That is, show that if a faster algorithm existed for constructing a binary search tree then you would violate the $\Omega(n \log n)$ comparison-based sorting lower bound. You may want to review an in-order traversal of a binary search tree.