Design and Analysis of Algorithms

Divide and Conquer and Sorting

Homework I: Turn in the solutions to problems 1, 3 and 7 on or before April 13th, 2000 in class. No late submissions.

1. **Stooge Sort:** Professors Howard, Fine, and Howard have proposed the following 'elegant' sorting algorithm:

   STOOGE-SORT (A, i, j)

   if $A[i] > A[j]$
     then exchange $A[i] \leftrightarrow A[j]$
   if $i + 1 \geq j$
     then return
   $k \leftarrow \lfloor (j - i + 1)/3 \rfloor$
   STOOGE-SORT $(A, i, j - k)$
   STOOGE-SORT $(A, i + k, j)$
   STOOGE-SORT $(A, i, j - k)$

   (a) Argue that STOOGE-SORT$(A, 1, length[A])$ correctly sorts the input array $A[1..n]$, where $n = length[A]$

   (b) Give a recurrence for the worst-case running time of STOOGE-SORT and a tight asymptotic (Θ-notation) bound on the worst-case running time.

   (c) Compare the worst-case running time of STOOGE-SORT with that of insertion sort, merge sort, heapsort, and quicksort. Do the professors deserve tenure?

2. k**-way Merging:** Give an $O(n \log_2 k)$ algorithm to merge k sorted lists into one sorted list, where n is the total number of elements in all the input lists. Hint: Use a heap for k-way merging. Analyze the time complexity of your algorithm.

   Prove a tight lower bound on the number of comparisons for k-way merging.

3. **MAX SUM:** Use the divide-and-conquer approach to write an efficient recursive algorithm that finds the maximum sum in any contiguous sublist of a given list of n real values. Analyse your algorithm, and show the results in order notation. Can you do better?

4. **Mode:** The mode of a set of numbers is the number that occurs most frequently in the set. The set $(4, 6, 2, 4, 3, 1)$ has a mode of 4.

   (a) Give an efficient and correct algorithm to compute the mode of a set of n numbers. Analyze the time complexity of your algorithm.

   (b) Suppose that we know that there is an (unknown) element that occurs more than $n/2$ times in the set. Give a worst-case linear-time algorithm to find the mode. For partial credit, your algorithm may run in an expected linear time. Analyze the time complexity of your algorithm.

5. **Searching:** Write an algorithm that searches a sorted list of n items by dividing it into three sublists of almost $n/3$ items. This algorithm finds the sublist that might contain the given item, and divides it into three smaller sublists of almost equal size. The algorithm repeats this process until it finds the item or concludes that the item is not in the list. Analyze your algorithm, and give the results using order notation.

6. **Probabilities for multi-faceted dice:** There are N dice, and the i-th die has its faces numbered from 1 to k_i where each k_i is a positive integer less than K. Assuming that on each die, each number is equally likely, we want to find the probability that when we roll all N dice, the sum of the numbers is exactly T. To do so, we need to count exactly how many ways we can choose integers $d_1, d_2, ..., d_n$ with $d_i \leq k_i$, such that $\sum d_i = T$. (The probability is this count divided by the product of the k_i's.)

   Find an efficient algorithm, and give its time analysis in terms of N and K.

7. **Weighted Median:** For n distinct elements $x_1, x_2, \cdots, x_n$ with positive weights $w_1, w_2, \cdots, w_n$ such that $\sum_{i=1}^{n} w_i = 1$, the weighted median is the element x_k satisfying

$$\sum_{x_i < x_k} w_i \leq 1/2 \text{and} \sum_{x_i > x_k} w_i \leq 1/2$$

   (a) Argue that the median of $x_1, x_2, \cdots, x_n$ is the weighted median of the x_i with weights $w_i = 1/n$ for $i = 1, 2, \cdots, n$.

(b) Show how to compute the weighted median of n elements in $O(n \lg n)$ worst-case time using sorting.

(c) Show how to compute the weighted median in $\Theta(n)$ worst-case time using a linear-time median algorithm.

8. **i-th Largest Element:** Show that, for any constant i, the i-th largest element of an array can be found with $n + O(\lg n)$ comparisons.

9. **Tree Isomorphism:** Give an efficient algorithm to decide, given two rooted binary trees T and T', whether T and T' are isomorphic as rooted trees.

10. **k-th Quantiles:** The k-th quantiles of an n-element set are the $k - 1$ order statistics that divide the sorted seet into k equal-sized (to within 1.) Give an $O(n \lg k)$-time algorithm to list the k-th quantiles of a set.

11. Give the best algorithm you can for the following problem:

    Name Blackjack Hand Card Counting

    Instance An array A of n positive integers (cards with face values) with values from 1 to k, and positive integers $l < n, v < kn$.

    Problem Count the number of sets of l array positions (hands of l cards) whose total value is equal to v.

    Analyze your algorithm in terms of n, k and l. Your algorithm should take time polynomial in all 3 parameters.

12. Professor Olay is consulting for an oil company, which is planning a large pipeline running east to west through an oil field of n wells. From each well, a spur pipeline is to be connected directly to the main pipeline along a shortest path (either north or south). Given x and y coordinates of the wells, how should the professor pick the optimal location of the main pipeline (the one that minimizes the the toal length of the spurs?) Show that the optimal location can be determined in linear time.

13. Take an input a sequence of $2n$ real numbers. Design an $O(n \log n)$ algorithm that partitions the numbers into n pairs, with the property that the partition minimizes the maximum sum of a pair.

14. Given two sets S_1 and S_2 (each of size n), and a number x, describe an $O(n \log n)$ algorithm for finding whether there exists a pair of elements, one from S_1 and one from S_2, that add up to x.

15. Given a set S of n integers and an integer T, give an $O(n^{k-1} \log n)$ algorithm to test whether k of integers in S sum up to T.

16. Consider a rectangular array. Sort the elements in each row into increasing order. Next sort the elements in each column into increasing order. Prove that the elements in each row remain sorted.

17. Let $a_1, \cdots, a_n$ be a squence of elements and let p and q be positive integers. Consider the subsequences formed by selecting every p-th element. Sort these subsequences. Repeat the process for q. Prove that the subsequences of distnsce p remain sorted.