CIS 102: Fall 2001

Homework Assignment # 4

Divide and Conquer

Part A

1. Search Algorithms
   Consider the modified binary search algorithm so that it splits the input not into two sets of almost-equal sizes, but into three sets of sizes approximately one-third. Write down the recurrence for this ternary search algorithm and find the asymptotic complexity of this algorithm.

   Consider another variation of the binary search algorithm so that it splits the input not into two sets of almost-equal sizes, but into two sets of sizes approximately one-third and two-thirds. Write down the recurrence for this search algorithm and find the asymptotic complexity of this algorithm.

2. Merge and Insertion Sorting
   Consider the following variation on Mergesort for large values of $n$. Instead of recursing until $n$ is sufficiently small, recur at most a constant $r$ times, and then use insertion sort to solve the $2^r$ resulting subproblems. What is the (asymptotic) running time of this variation as a function of $n$?

3. Max-Min Algorithm
   Design and analyze a divide and conquer MAXMIN algorithm that uses $\lceil \frac{3n}{2} \rceil - 2$ comparisons for any $n$.

4. Majority Element Problem
   Assume you have an array $A[1..n]$ of $n$ elements. A majority element of $A$ is any element occurring in more than $n/2$ positions (so if $n = 6$ or $n = 7$, any majority element will occur in at least 4 positions). Assume that elements cannot be ordered or sorted, but can be compared for equality. (You might think of the elements as chips, and there is a testor that can be used to determine whether or not two chips are identical.)

   Design an efficient divide and conquer algorithm to find a majority element in $A$ (or determine that no majority element exists).

   Although there is an $O(n)$ algorithm for this problem, all that I am expecting from you is an $O(n \log n)$ algorithm.
5. Integer Multiplication Problem

(a) Consider the problem of multiplying two large \( n \)-bit integers in a machine where the word size is one bit. Describe the straightforward algorithm that takes \( n^2 \) bit-multiplications.

(b) Find a way to compute the product of the two numbers using three multiplications of \( n/2 \) bit numbers (you will also have to do some shifts, additions, and subtractions, and ignore the possibility of carries increasing the length of intermediate results). Describe your answer as a divide and conquer algorithm. (Hint: You may want to use the fact that \( ad + bc = (a + b)(c + d) - ac - bd \).)

(c) Assume that adding/subtracting numbers takes time proportional to the number of bits involved, and shifting takes constant time. Derive a recurrence relation for the running time of your divide and conquer algorithm. Use the master theorem to get an asymptotic solution to the recurrence.

(d) Now assume that we can find the product of two \( n \)-bit numbers using some number of multiplications of \( n/3 \)-bit numbers (plus some additions, subtractions, and shifts). What is the largest number of \( n/3 \) bit number multiplications that leads to an asymptotically faster algorithms than the \( n/2 \) divide and conquer algorithm above?
Part B

1. **Towers of Hanoi**
   Design a divide-and-conquer algorithm for the Towers of Hanoi problem. Write down the recurrence equation and solve it exactly.

2. **Pseudo-median Problem**
   An $\alpha$-pseudomedian of a list of $n$ distinct values (where $0 < \alpha < 1$) is a value that has at least $n^\alpha$ elements larger than it, and at least $n^\alpha$ elements smaller than it.

   The following is a divide-and-conquer algorithm for computing a pseudomedian for which I am not going to specify what $\alpha$ is, rather it is an exercise to discover what the $\alpha$ is. In this algorithm, assume that $n$ is a power of 3. If $n = 3$, then simply sort the 3 values and return the median of these 3 values and this will be the pseudo-median for some $\alpha$. Otherwise, divide the $n$ items into $\frac{n}{3}$ groups of 3 values. Sort each group of 3, and pick out the $\frac{n}{3}$ medians. Now recursively apply the procedure and return the final value as the pseudomedian of the list.

   (a) Let $T(n)$ be the number of comparisons used by the preceding algorithm for computing the pseudomedian. Write a recurrence relation for $T(n)$ and solve it exactly. Hence show that the algorithm runs in time $O(n)$.

   (b) Let $E(n)$ be the number of values that are smaller than the value found by the preceding algorithm. Write a recurrence relation for $E(n)$, and hence prove that the algorithm does return a pseudomedian. What is the value of $\alpha$?

   Remark: I expect you to understand this material by November 1. There will be a quiz from Part A only. Solutions to Part B will not be posted. You are encouraged to write down the solutions to Part B, acknowledge all the help you get on Part B, write down the date when you did attempt part B, get a signature from a TA or a tutor within a day or two to verify that you did it on the date claimed by you and include it as a part of your portfolio at the end of the quarter. All the graduate students and students aspiring to get a grade of B- or better MUST submit the solutions to Part B.