COT 5407 Introduction to Algorithms

Homework 2

DUE: Thursday, September 23, 2010

Please remember that all submissions must be typeset. Handwritten submissions will NOT be accepted.

1. The following divide-and-conquer algorithm is proposed for finding the simultaneous maximum and minimum. If there is one item, it is the maximum and minimum, and if there are two items, then compare them and in one comparison you can find the maximum and minimum. Otherwise, split the input in two halves, divided as evenly as possibly (if N is odd, one of the two halves will have one more element than the other). Recursively find the maximum and minimum of each half, and then in two additional comparisons produce the maximum and minimum for the entire problem.

   (a) Suppose N is a power of 2. What is the exact number of comparisons used by this algorithm?

   (b) Suppose N is of the form $3 * 2^k$. What is the exact number of comparisons used by this algorithm?

   (c) Modify the algorithm as follows: When N is even, but not divisible by four, split the input into sizes of $N/2 - 1$ and $N/2 + 1$. What is the exact number of comparisons used by this algorithm?

2. Suppose that instead of creating groups of five elements, the Blum, Floyd, Pratt, Rivest, Tarjan selection algorithm creates groups of three elements. Does this yield a linear-time algorithms? What would happen with groups of seven elements?

3. You are given two sorted arrays of N numbers. Design an $O(\log N)$ algorithm to find the median of the $2N$ numbers.

4. Prove that merging two sorted arrays of N items requires at least $2N - 1$ comparisons. Please note that I am asking for a lower-bound proof. You must show that if two elements in the merged list are consecutive and from different lists, then they must be compared.

5. M is an N-by-N integer matrix in which the keys in each rows are in increasing order and the keys and in each column are in increasing order (reading top-to-bottom). Consider the problem of determining if an integer x is in M. You may use three-way comparisons (i.e. one comparison of x with `M[i][j]` tells you either that x is less than, equal to, or greater than `M[i][j]`).

   (a) Give an algorithm that uses at most $2N - 1$ comparisons.

   (b) Prove that any algorithm must use at least $2N - 1$ comparisons.