COT 5407 Introduction to Algorithms

Homework 2

DUE: Thursday, Sept 20, 2012 at 11:00 AM

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

1. Solve the following recurrence, providing a Big-Oh result, assuming that $T(0) = T(1) = 1$:

$$T(N) = N + \frac{3}{N}\sum_{i=0}^{N-1} T(i)$$

2. Consider the following algorithm to sort N elements: Place each element in its own list. Then repeat the following step $N-1$ times: Choose two lists, and merge them. Note that at any point, the merge reduces the number of lists by one, but maintains all lists in sorted order. Thus at the end, what remains is a single sorted list of N elements.

   Assume an efficient implementation of lists, stacks, and queues. Assume that merging two lists will always take time linear in the size of the resulting list.

   (a) Suppose the initial N lists are placed on a stack. At any point, the first two lists are chosen to merge, and the result is placed at the top of the stack. What is the running time of the sorting algorithm?

   (b) Suppose the initial N lists are placed on a queue. At any point, the first two lists are chosen to merge, and the result is placed at the back of the queue. What is the running time of the sorting algorithm?

3. Suppose we want to partition N items into G equal-sized groups of size N/G, such that the smallest N/G items are in group 1, the next smallest N/G items are in group 2, etc. The groups themselves do not have to be sorted. For simplicity, you may assume that N and G are powers of two. Give an $O(N \log G)$ algorithm to solve this problem.

4. Let k be a **constant** that is at least 2.

   Either prove that merging k sorted arrays of N items requires at least $kN - 1$ comparisons, or demonstrate that this assertion is **false** by providing an algorithm that uses fewer than $kN - 1$ comparisons.

   Please note that I am asking for either a lower-bound proof of $kN - 1$ comparisons or a constructive algorithm that uses less than $kN - 1$ comparisons.

5. M is an R-by-C integer matrix in which the keys in each row 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]`).

   True or False: Any algorithm must use at least $R + C - 1$ comparisons. If true, give a lower-bound proof; if false, give a counterexample for a specific R and C that uses less than $R + C - 1$ comparisons.

6. Continuing the previous question, suppose $R = C = N$ (i.e. the matrix is square).

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

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