CS 5633 Analysis of Algorithms – Spring 08

1/24/08

2. Homework

Due **1/31/08** before class

Always justify your answers.

1. **Divide and Conquer (5 points)**
   (a) **(3 points)** Design a divide-and-conquer algorithm to compute the average of an array of n numbers. Describe your algorithm in pseudo-code with verbal explanation. (Note: No points will be given for an algorithm that does not have divide-and-conquer properties.)
   (b) **(2 points)** Set up and solve a recurrence relation for the runtime of your algorithm. (You do not need to prove it by induction. A justification for your guess is enough.)

2. **Minimum element (3 points)**
   Where is the minimum element located in a max-heap? How can you compute the minimum? How much time does it take in the worst case (assuming the heap contains n elements)?

3. **Heaps with links (9 points)**

   Suppose that binary max-heaps are represented using explicit links, that means in a standard binary tree representation that uses nodes with pointers/references to left and right children. These heaps, although not represented in arrays, are still required to be almost complete.

   Consider the problem of merging the binary max-heap L with the binary max-heap R. Assume both heaps are complete trees: L has height l and contains $2^l - 1$ nodes, and R has height r and contains $2^r - 1$ nodes. Let $n = \max\{2^l - 1, 2^r - 1\}$.

   a) (3 point)
   Give an $O(\log n)$ algorithm to merge the two heaps if $l = r$.

   b) (3 point)
   Give an $O(\log n)$ algorithm to merge the two heaps if $|l - r| = 1$.

   c) (3 point)
   Give an $O(\log^2 n)$ algorithm to merge the two heaps regardless of l and r. *(Hint: Break the larger heap into smaller subheaps and apply a) and b). Note that a) and b) can even be applied if one of the heaps is almost complete.)*

FLIP OVER TO BACK PAGE ⟹

4. **Guessing and Induction (9 points)**

For each of the following recurrences use the recursion tree method to find a good guess of what it could solve to (make your guess as tight as possible). Then prove that $T(n)$ is in big-Oh of your guess by induction (inductive step and base case). *(Hint: Appendix A in the book has a list of solved summations that might be helpful. For simplicity you may want to use* $\log_3 n$ *instead of* $\log_2 n$ *for the first question.)*

Every recursion below is stated for $n \geq 2$, and the base case is $T(1) = 1$.

(a) **(3 points)**
$T(n) = 3T(\frac{n}{3}) + 2n$

(b) **(3 points)**
$T(n) = 2T(\frac{n}{2}) + n^3$

(c) **(3 points)**
$T(n) = 4T(\frac{n}{2}) + n$