CS 5633 Analysis of Algorithms – Spring 04

2/18/04

6. Homework

Due **2/25/04** before class

1. **Deterministic select (4 points)**
   Argue why or why not the deterministic SELECT algorithm works if you divide into ...
   a) ... groups of 3 elements.
   b) ... groups of 7 elements.

2. **Sorting (6 points)**
   Suppose you are given a sorted list of n elements followed by $f(n)$ randomly ordered elements. How would you sort the entire list if ...
   a) ... $f(n) = O(1)$?
   b) ... $f(n) = O(\log n)$?
   c) ... $f(n) = \sqrt{n}$?

3. **Smallest k numbers (6 points)**
   Given an unsorted array of n numbers, find the k smallest numbers and output them in sorted order.
   Describe the algorithms that implement the following methods with the best asymptotic worst-case runtimes, and analyze the runtimes in terms of n and k (so you should have n and k in the big-Oh notation).
   a) Sort the numbers first. (Since we don't know anything about the numbers, the sorting algorithm has to be comparison-based.)
   b) Use a priority queue.
   c) Use an order statistic algorithm to find the k-th smallest number, partition around it, and sort the k smallest numbers.

4. **Size of the universe (3 points)**
   Let U be a universe of keys which we hash into a table with m slots. Show that if $|U| > (n-1)m$ then the worst-case running time for hashing with separate chaining is $O(n)$. *Hint: Show that there is a subset of n keys of U that all hash to the same slot.*

5. **Filling a hash table (4 points)**
   How long could it take in the worst case to insert n keys into an initially empty hash table, using ...
   a) ... separate chaining with unordered lists?
   b) ... separate chaining with ordered lists?
   c) ... linear probing?
   c) ... quadratic probing?

FLIP OVER TO BACK PAGE $\Longrightarrow$

6. **Open addressing (1 point)**
   Why is open addressing called like that?

7. **Deleting in hash tables (4 points)**
   Describe how to **efficiently** implement the DELETE function in hash tables with

   a) ... separate chaining.

   b) ... open addressing.