CS 5633 Analysis of Algorithms – Spring 12

2/9/12

4. Homework

Due **2/21/12** before class

1. **Quicksort decision tree**

   The decision tree on the slides in class was for insertion sort of three distinct elements. Draw the decision tree for quicksort of an array $A[1..3]$ of three elements. Use the deterministic quicksort and the partition routine described on the slides. Annotate the decision tree with the current state of the array.

2. **Lower bound for comparison-based searching**

   Consider the problem of searching for a given key in a sorted array of n numbers. Use a decision tree to show a lower bound of $\Omega(\log n)$ for any comparison-based search algorithm. *(Hint: The decision tree needs to represent the output of the search algorithm in its leaves. What should be stored in the leaves?)*

3. **Median**

   Let $A[1..n]$ be an unsorted array of n numbers, and let k be a positive integer $k \leq n$. Assume both n and k are odd. The task is to design an algorithm that outputs the k numbers that are the closest to the median. (For example, if $A = [2, 1, 7, 4, 6, 5, 8]$ and $k = 3$ then the output should be the numbers $5, 4, 6$, but not necessarily in sorted order.)

   Give pseudocode for such an algorithm that runs in worst-case $O(n)$ time. *(Hint: Use the Select algorithm.)*

4. **Radix sort for strings**
   Given n strings, each of maximum length l_{max} and with total length L. Assume there are k different characters.

   (a) Shortly describe an $O(L + l_{max})$-time algorithm based on radix sort to sort these strings.

   (b) Now consider using radix sort to sort these strings, however using merge sort as the auxiliary sort algorithm (instead of counting sort). What is the runtime of this algorithm? Make your bound as tight as possible.

   (c) Describe how you can use plain counting sort to sort these strings. What is your runtime? *(Hint: You need to convert strings to numbers of a certain base. Which base should you use?)*

5. **Deterministic select**
   The deterministic select algorithm that we covered in class splits the elements into groups of 5.

   (a) Argue that when using groups of 3 the runtime analysis to show linear runtime fails.

   (b) Prove that the runtime of the algorithm is $O(n)$ when splitting the elements into groups of 7.

   (c) What about groups of 9 numbers, or any other odd number ≥ 9?