CS 3343 Analysis of Algorithms – Spring 09

3/17/09

6. Homework

Due: Tuesday 3/24/09 before class

Justify all your answers.

1. **Radix sort and counting sort (6 points)**
   Given n numbers between 0 and $f(n)-1$. Please give the runtimes for counting sort, and for radix sort (with optimized r) for each of the functions $f(n)$ below, and state which of the two sorting algorithms is faster in each case.

   a) $f(n) = n^4$ b) $f(n) = 2^n$ c) $f(n) = n^n$

2. **Red-black trees (6 points)**
   Find a sequence of numbers which, when incrementally inserted into a red-black tree, causes the following sequence of rotations:

   right, right, right, left.

   You may start with an initially non-empty tree, and you may insert numbers that do not cause any rotations. But there should not be any additional rotations performed.

   Draw the sequence of trees that you obtain after each insertion. For each such tree indicate the node that violates the red-black tree condition, indicate the nodes that participate in the rotation, the type of the rotation, and the subtrees that correspond to each other before and after the rotation.
   Hint: Use a red-black tree demo from the web.

3. **Quicksort-ConvexHull (6 points)**
   Consider the convex hull algorithm we discussed in class. This algorithm was similar to mergesort (in that it first divided the problem into two halves, and in the end merged the results). Now we will consider an algorithm that is similar to Quicksort: The Quicksort-ConvexHull algorithm does NOT sort the n input points by their x-coordinate as a preprocessing step. So, the input for the algorithm are n unsorted points. Quicksort-ConvexHull first partitions the points according to their x-coordinate, using the Partition routine of Quicksort. It then recursively solves the two subproblems, and in the end merges the subsolutions using the linear-time Merge routine for convex hulls discussed in class.

   (a) If the pivot is the first element, what are the best-case and worst-case runtimes of Quicksort-ConvexHull?

   (b) If the pivot is chosen at random, what is the expected (i.e., average) runtime?

Flip over to back page $\Longrightarrow$

4. **Multi-min (6 points)**
   Consider the following task: Given an unsorted array of n numbers, find the k smallest numbers and output them in sorted order.
   Describe three inherently different algorithms that solve this problem. Analyze their runtimes in terms of n and k (so you should have n and k in the big-Oh notation). Try to find the fastest possible algorithm. Which of your algorithms is the fastest?