Final Examination
2010 December 8, pm 3:15-5:15

• This is an open text and open notes exam.
• Write your answers on your own paper. The instructor will have blank paper in case you have none.
• Each problem is worth 10 points, for a total of 70 points.

1. Illustrate the $O(n)$ BUILD-MAX-HEAP algorithm by converting the following array into a max-heap:

   5, 9, 18, 37, 2, 7, 10, 15, 22, 30, 3, 16, 17, 23, 13, 4, 27, 21

   [8 points]

2. Insert the following values into an initially empty B-tree of order $t = 3$:

   5, 9, 18, 37, 2, 7, 10, 15, 22, 30, 3, 16, 17, 23, 13, 4, 27, 21

   [12 points]

3. Describe an algorithm that, given $n$ integers in the range 0 to $n$, preprocesses its input and then answers any query about how many of the $n$ integers fall into a range $[a..b]$ in $O(1)$ time. Your algorithm should use $O(n)$ preprocessing time. (Hint: look at the comment on line 9 of COUNTING-SORT.)

   [10 points]

4. Suppose that you are given a sequence of $n$ elements to sort. The input sequence consists of $k$ subsequences, each containing $n/k$ elements. The elements in a given subsequence are all smaller than the elements in the succeeding subsequence, and larger than the elements in the preceding subsequence. That is, all that is needed to sort the whole sequence of length $n$ is to sort the $n/k$ elements in each of the $k$ subsequences.

   (a) How long would it take to sort all of the subsequences?
   (b) Give a lower bound on the number of comparisons needed to solve this variant of the sorting problem.
   (c) Does the lower bound of part (b) match the upper bound of part (a)?

   [7 points]

5. Regarding fibonacci heaps, these are to be min-heaps (as in the text).

   (a) Into an initially empty fibonacci heap, insert the values: 18, 15, 8, 3, 17, 9, 25, 28, 6, 11
   (b) Remove the min from that heap.
(c) Into another fibonacci heap, insert the values: 12, 2, 5, 21, 7, 30, 16
(d) Remove the min from that second heap.
(e) Merge the two heaps together.
(f) Remove the min again.
(g) Look at the value that is deepest (furthest from the root) in the biggest tree in your list. Reduce its key to 1.

[13 points]

6. Regarding red-black trees

(a) Into an initially empty RB tree insert the values: 18, 27, 20, 12, 15, 2, 5, 8, 10.
(b) From the RB tree below (dotted lines mean red), delete 17.
(c) From the RB tree drawn on the board, delete the value at the root.

[12 points]

7. Recall the following problem on homework 6, with a solution:

Suppose you have a set $A$ of $n$ nuts and a set $B$ of $n$ bolts, such that each nut in $A$ has a unique matching bolt in $B$. The only kind of comparison you can make is to take a nut-bolt pair $(a, b)$, where $a \in A$ and $b \in B$, and test to see whether the threads of $a$ are larger, smaller, or a perfect match to the threads of $b$. Give an efficient algorithm to match up all the nut and bolts. What can you say about the run-time of your algorithm?

(sol’n)

- pick a random nut $a \in A$
- use $a$ to partition $B$ into $B_0$ (the bolts too small for $a$) and $B_1$ (bolts too big)
  - set aside the bolt(s) that matched $a$
- take the bolt $b$ that matched $a$ and . . .
- use $b$ to partition $A$ into $A_0$ and $A_1$
- apply this method recursively on $(A_0, B_0)$ and $(A_1, B_1)$

This runs in $O(n \log n)$ expected time, and uses a random number generator. Now suppose that no random number generator is available. How can this be solved deterministically and how long does it take?