Final Examination
2010 March 17, am10:15-pm12:15

• 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.

1. The max-heap implementation in the text is that of a binary heap. We wish to generalize that to \(d\)-ary heaps. For example, a ternary heap is a \(d\)-ary heap where \(d = 3\). In a ternary heap, every node (i) has one key and \(d = 3\) children and (ii) satisfies the max-heap property - the key stored there is at least as large as any key stored in its children.

(a) If a ternary heap is stored in an array, show how, given an index \(i\) of a node, to find that node’s left, middle, and right children.

(b) What is the height of a ternary tree? What is the height of a \(d\)-ary tree (in terms of \(d\) and \(n\))? 

(c) Describe how to perform an insertion in a ternary heap.

(d) Think about the extractMax method (you do not need to describe it). How long does it take for a ternary heap? For a \(d\)-ary heap?

[12 points]

2. You are given an unsorted array \(A\) of \(n\) comparable items. Describe how to find the \(n/\lg n\) largest elements from \(A\), listed in increasing order. Aim for \(O(n)\) time. Can you give two different algorithms for this?

[10 points]

3. Consider a binary tree (not a binary search tree), each node of which contains the fields \(key\), \(weight\), \(lchild\), and \(rchild\). The first two are of type \(\text{int}\), the other two are pointers. This problem will use only the \(weight\) field, so from now on we ignore the \(key\) field.

The problem here is to find the heaviest weight path from the root to a null node. That is, we want to start at the root, and proceed to a leaf, adding up the weights. What is the largest value you can get from a given tree going from top to bottom?

Write a routine which, given (the root to) a tree \(T\), returns the total weight of the heaviest path. (Hints: recursion, \(O(n)\), short code.)

[10 points]

4. Professor XYZ claims to have invented a new search method which will search for an element in an ordered list in \(O(\lg \lg n)\) time - that is, it significantly beats binary search. Furthermore, it is comparison based, so it has applications to many types of data. Explain why this cannot be the case.

[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: 21, 4, 7, 2
   (b) Remove the min from that heap.
   (c) Into another fibonacci heap, insert the values: 14, 17, 2, 5, 11, 16, 20, 3, 9, 1
   (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. Everyone loves recurrence relations. Solve

   (a) $T(n) = 6T(n/7) + n$
   (b) $T(n) = 6T(n/4) + n \log n$
   (c) $T(n) = 16T(n/4) + n^2$
   (d) $T(n) = 30T(n/3) + n^3$

[8 points]

7. 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 9.

[12 points]

Total: 72 points