Data Structures and Programming

Spring 2016, Midterm Exam.

April 19, 2016

1. (10 pts) Order the following list of functions by the big-Oh notation in non-decreasing order. Group together (for example, by circling) those functions that are big-Theta of one another. The logarithmic function is of base 2.

   $\sqrt{1000n}$ $\quad n+2^n$ $\quad n^2 \log n$ $\quad 5^n$ $\quad 4n^{3/2}$ $\quad n^{3/2} \log n$ $\quad 2^{2\log n}$ $\quad \log\log n$ $\quad 6\sqrt{n}$ $\quad 2^{100}$ $\quad 2^{10}n+0.01n^2$ $\quad n^{3/2}+n$

   Solution

   $2^{100} < \log\log n < (\sqrt{1000n}, 6\sqrt{n}) < (4n^{3/2}, n^{3/2}+n) < n^{3/2}\log n < (2^{10}n+0.01n^2, 2^{2\log n}) < n^2 \log n < n+2^n < 5^n$

2. (10 pts) Evaluate the following postfix expression, showing the state of the stack at each step. Totally there are 11 steps to show, each corresponding to the encounter of a number or an operation.

   $$6 \;\; 5 \;\; * \;\; 7 \;\; 3 \;\; - \;\; 4 \;\; 8 \;\; + \;\; * \;\; +$$

   Solution:
   6
   6 5
   30
   30 7
   30 7 3
   30 4
   30 4 4
   30 4 4 8
   30 4 12
   30 48
   78

3. (24 pts)

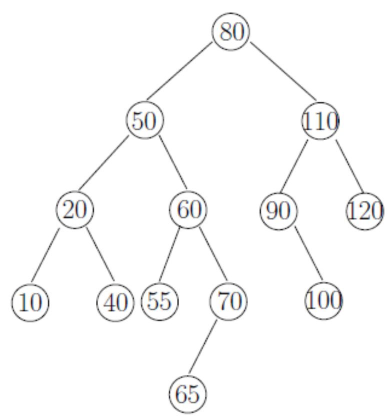
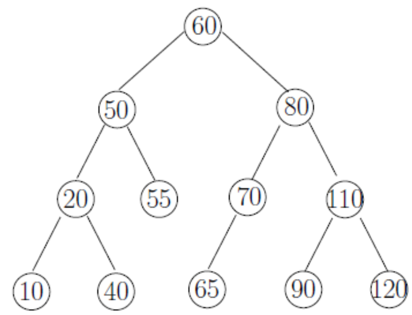


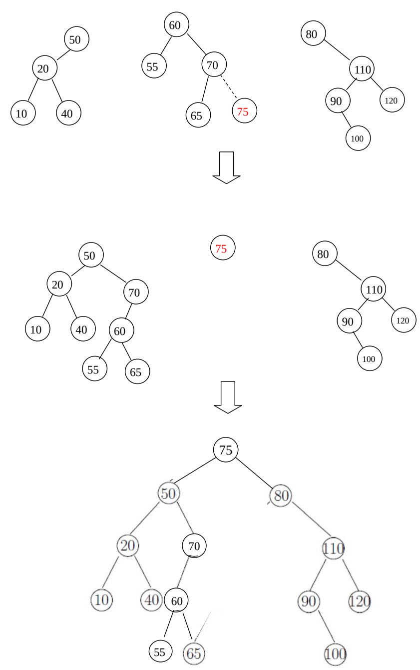
Figure 1: A search tree

   (a) (10 pts) Consider the tree shown in Figure 1. Clearly the tree is an AVL tree. Show the resulting AVL-tree after deleting node 100. Show your derivation in sufficient detail.

   Solution



(b) (10 pts) Again consider Figure 1 but now assume that the tree is a splay tree. Insert 75 into the splay tree in a *top-down* fashion. Show your derivation in sufficient detail.
Solution

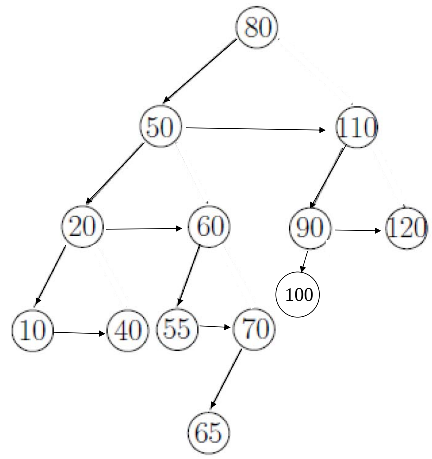


(c) (4 pts) Give the left-child right-sibling representation for the tree shown in Figure 1.
Solution

4. (10 pts) Let S and T be two sets of integers, neither of which is sorted. Given a number $x \in S$, define its successor $succ(x)$ as the smallest number in S that is greater than x (if x is already the maximum number in S, $succ(x) = \infty$). Describe an algorithm to output all numbers $x \in S$ such that the interval $[x, succ(x)]$ does not contain any number in T. Your algorithm must terminate in $O((n+m)\log n)$ time, where $n = |S|$ and $m = |T|$. For example, let $S = \{50, 30, 80, 20, 100\}$ and $T = \{15, 17, 9, 83, 55, 56, 42\}$. We have $succ(30) = 50$, and $succ(100) = \infty$. The numbers in S to be output are 20, 100. Number 30, for instance, should not be output because $[30, succ(30)] = [30, 50]$ contains at least a number of T (i.e., 42). Explain why your algorithm takes $O((n+m)\log n)$ time.

Solution Create an AVL-tree on S in $O(n \log n)$ time. For each number in T, find its predecessor x in S, and mark x if found. Doing so for all numbers in T takes $O(m \log n)$ time. Finally, scan S and output the unmarked numbers in $O(n)$ time.

5. (20 pts) Give the tightest possible worst-case upper bound for each of the following in terms of N. You MUST



choose your answer from the following (not given in any particular order), each of which could be reused (could be the answer for more than one of (1) - (10)). No explanation is needed.

$$O(N^2), O(N^3 \log N), O(N \log N), O(N), O(N^2 \log N), O(N^5), O(2^N), O(N^3), O(\log N), O(1), O(N^4), O(N^N)$$

(1) Delete a value in a Binary Search Tree of size N
Solution $O(N)$

(2) Find the minimum value in a Splay Tree of size N
Solution $O(N)$

(3) The number of rotations needed in inserting a value into an AVL tree of size N
Solution $O(1)$

(4) Pop on a stack containing N elements implemented as a singly-linked list
Solution $O(1)$

(5) Post-order traversal of an AVL Tree containing N elements
Solution $O(N)$

(6) Perform N operations (chosen from insertion, deletion and find) to a initially empty Splay Tree.
Solution $O(N \log N)$

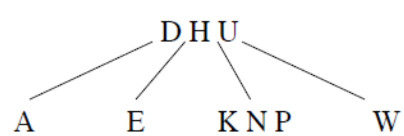
(7) Inserting N values into an initially empty Binary Search Tree
Solution $O(N^2)$

(8) Print out all leaf nodes in an AVL tree in descending order (from largest to smallest).
Solution $O(N)$

(9) Given a binary search tree of N nodes, find which value is the median value, and delete that value
Solution $O(N)$

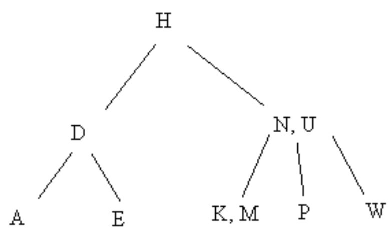
(10) Find the maximum element in a 2-3-4 tree of Size N
Solution $O(\log N)$

6. (16 pts) Consider the following *2-3-4 tree*.



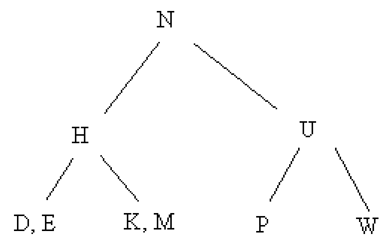
Answer the following questions. Show your derivations in sufficient detail.

(1) (8 pts) Show the tree that results from inserting M into the above tree in a one-pass (i.e., top-down) fashion.
Solution



(2) (8 pts) Show the tree that results from deleting A in what you obtained AFTER inserting M in a two-pass (i.e., bottom-up) fashion.

Solution



7. (10 pts) An array $A[0...k-1]$ of bits (each array element is 0 or 1) stores a binary number $x = \sum_{i=0}^{k-1} A[i] \cdot 2^i$. To add 1 (modulo 2^k) to x, we use the following procedure:

INCREMENT (A, k)
$i \leftarrow 0$
while $i < k$ and $A[i] = 1$
    do $A[i] \leftarrow 0$
        $i \leftarrow i + 1$
if $i < k$
then $A[i] \leftarrow 1$

Given a number x, define the potential $\Phi(x)$ of x to be the number of 1's in the binary representation of x. For example, $\Phi(19) = 3$, because $19 = 10011_2$. Use a potential-function argument to prove that the amortized cost of an increment (i.e., add 1) is $O(1)$, where the initial value in the counter is $x = 0$. Show your derivation in detail.

Solution: $\Phi(x)$ is a valid potential function the number of 1's in the binary representation of x is always nonnegative, i.e., $\Phi(x) \geq 0$ for all x. Since the initial value of the counter is 0, $\Phi_0 = 0$.

Let c_k be the real cost of the operation INCREMENT(A, k), and let d_k be the number of times the while loop body in Lines 3 and 4 execute (i.e., d_k is the number of consecutive 1's counting from the least-significant bit of the binary representation of x). If we assume that executing all of Lines 1, 2, 5, and 6 require unit cost, and executing the body of the while loop requires unit cost, then the real cost is $c_k = 1 + d_k$,

The potential decreases by one every time the while loop is executed. Therefore, the change in potential, $\Delta\Phi = \Phi_k - \Phi_{k-1}$ is at most $1 - d_k$. More specifically, $\Delta\Phi = 1 - d_k$ if we execute Line 6 , and $\Delta\Phi = -d_k$ if we do not.

Thus, using the formula for amortized cost, we get

$$\begin{aligned}
\hat{c_k} &= c_k + \Delta\Phi \\
&= 1 + d_k + \Delta\Phi \\
&\leq 1 + d_k + 1 - d_k \\
&= 2.
\end{aligned}$$

Therefore, the amortized cost of an increment is $O(1)$.