COMP 2119A: Solution for Test 2

1

a) Incorrect. Let node 7 be the node u, then its successor is 8, which has a left child.

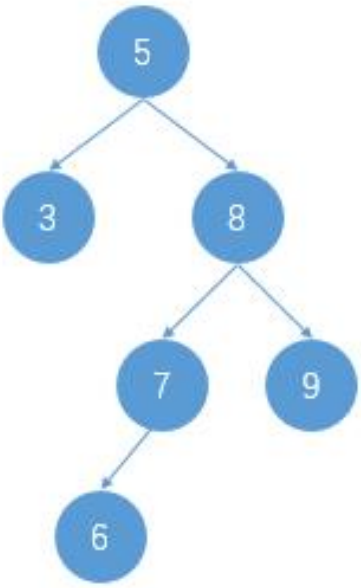

Figure 1: Q1(a)

b) Incorrect. Let node 8 be the node u, then its successor is 9, which has a right child.

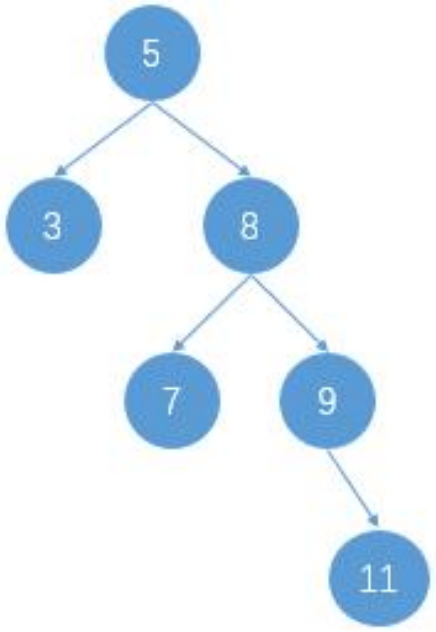

Figure 2: Q1(b)

c) Correct. In the preorder traversal of a binary search tree, for any node $v_i = x$, we can determine a unique position for it according to $v_1, ..., v_{i-1}$, which process is similar with building a binary search tree(insert node).

d) Correct. In any situation, at most 2 rotations are needed to re-balance an AVL tree when we add a new node to it.

e) Incorrect. When we delete a leaf node, the worst case is that all its parents need a rotation, so $O(\log N)$ rotations are need in the worst case when we delete a node from the AVL tree.

2

Proof: We assume the height of M-AVL tree is h, now, consider the total number of nodes in the M-AVL tree. The minimum number of nodes in the M-AVL tree is $n_1 = (1 + 2 + ... + 2^{h-3}) + 2 = 2^{h-2} + 1$. The maximum number of nodes in the M-AVL tree is $n_2 = (1 + 2 + ... + 2^{h-1}) = 2^h - 1$. So the number of nodes in the M-AVL tree is $n_1 \leq n \leq n_2$, i.e.,$2^{h-2} + 1 \leq n \leq 2^h - 1$. We can get that $\log_2(n + 1) \leq h \leq \log_2(n - 1) + 2$. Therefore, $h = O(\log n)$.

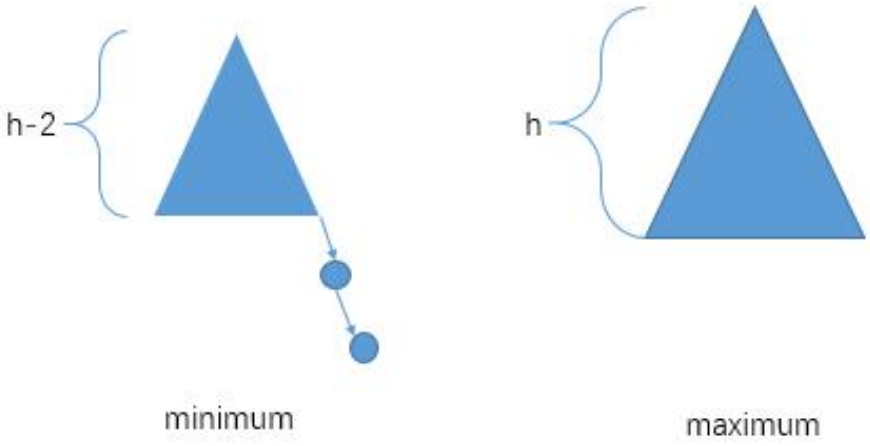

Figure 3: Q2

3

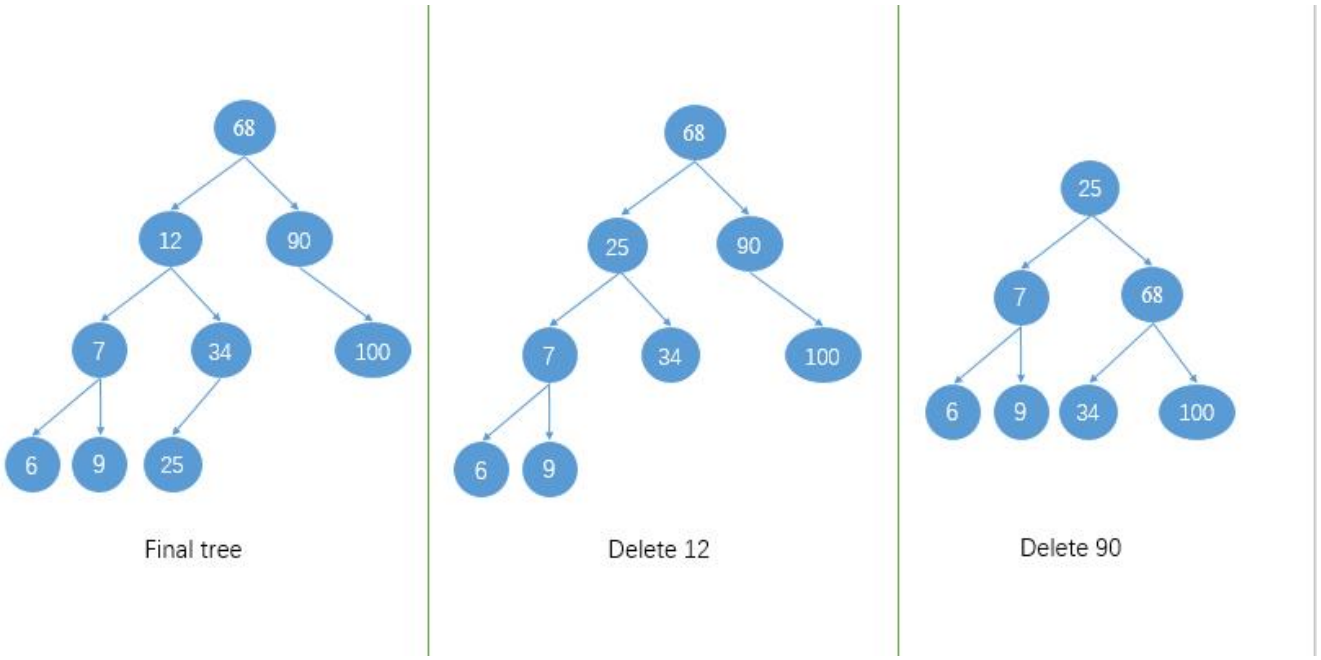

Figure 4: Q3

4

a)False

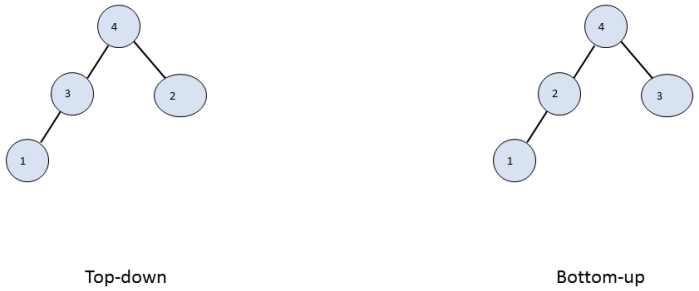

Figure 5: question 4a

counter example see figure5

b)

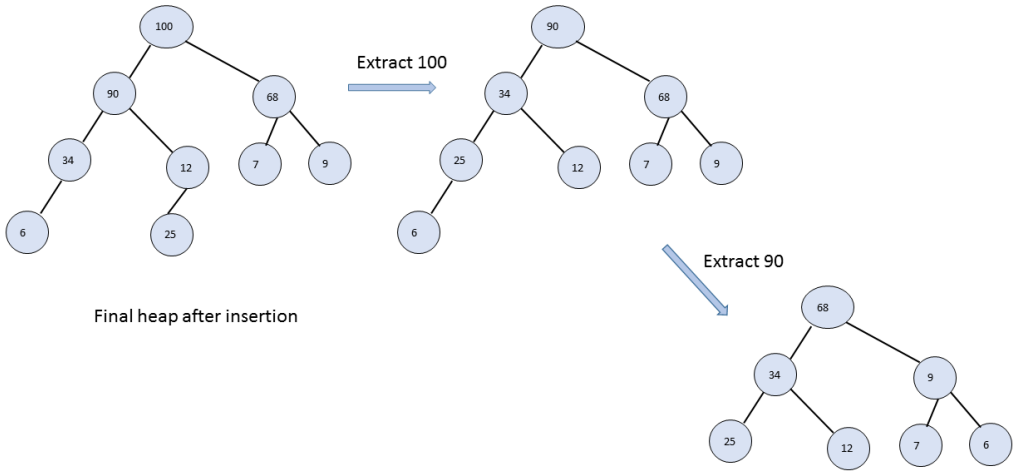

Figure 6: question 4b

see figure6

5

a)
56 87 21 47 91 35 66 0 37 101 25
21 56 87 47 91 35 66 0 37 101 25
21 47 56 87 91 35 66 0 37 101 25
21 47 56 87 91 35 66 0 37 101 25
21 35 47 56 87 91 66 0 37 101 25
21 35 47 56 66 87 91 0 37 101 25
0 21 35 47 56 66 87 91 37 101 25
0 21 35 37 47 56 66 87 91 101 25
0 21 35 37 47 56 66 87 91 101 25
0 21 25 35 37 47 56 66 87 91 101

b)construct a binary search tree.

struct node{
number;
int count;
node $*$ left;
node $*$ right;
node $*$ parent;
}

insert(node $*$ root, element, int count){
if the node has been inserted already, count++;
otherwise, insert into the tree;
}

gethighestF{
int max=0;
for each node in the tree,
if node.count $>$ max
max=node.count;
number=node.number;
return (max, number);
}

time complexity $O(n \log n)$.

c)we can use counting sort.
create $c[1 \cdots k]$
for i form 1 to n:
C[A[i]]++;
max=C[0];
for j from 1 to k:
if C[j]$>$ max: max=C[j], index=j;
return j;

d)(More than one answer)

letter	frequency	code
i	1	0000
l	1	0001
o	3	11
v	1	0010
e	1	0011
y	1	0100
u	2	101
u	2	101
s	1	0101
m	1	0110
c	1	0111
h	1	100