CS201c: Midsem solutions

Note. Marking scheme is only tentative. It should give you an idea of the relative importance of various steps in the correct solution. The actual marks will depend on an overall evaluation of the answer written by you.

Solution 1.

(5 points)

All paths from a leaf node (where key k will be inserted) to the root of tree T will have maximum number of nodes i.e., 2t-1 nodes. This is because each of these nodes will get split and a key will be promoted to their respective parents.

The remaining nodes can have minimum number of keys i.e., t-1 keys each.

(5 points)

N(h) = Number of keys in a B-tree of order t and height x, when every node has exactly t-1 keys.

Level	# of nodes	# of keys
0	1	t-1
1	t	t(t-1)
2	t^2	t^2(t-1)
3	t^3	t^3(t-1)
.		
.		
.		
x	t^x	t^x(t-1)

Total number of keys = (1+t+...+t^x)(t-1) = t^{x+1}-1.

Let P be the root-to-leaf path, where each node has exactly 2t-1 keys.

(2t-1) trees, each with size N(h-1), hang from the first node of P,
(2t-1) trees, each with size N(h-2), hang from the second node of P,
.
.
.
(2t-1) trees, each with size N(0), hang from the second from last node of P.

Thus, our answer to the question is:

$(2t-1)$ [$N(h-1)+N(h-2)+...+N(0)$] + $(h+1)(2t-1)$

$= (2t-1) [(t^h + t^{h-1} + \dots + t) - (h) + (h+1)]$

$= (2t-1) [(t^{h+1} - t)/(t-1) + 1]$
$= (2t-1) [(t^{h+1}-1) / (t-1)]$

Note. We can alternatively proceed like this:

Level	# of nodes	# of keys
0	1	$(2t-1)$
1	$2t$	$(2t-1)(t-1) + (2t-1) = t(2t-1)$
2	$2t^2 + t$	$(2t^2+t)(t-1)+t = t^2(2t-1)$
3	$2t^3+t^2+t$	$(2t^3+t^2+t)(t-1)+t=t^3(2t-1)$
4	$2t^4+t^3+t^2+t$	$t^4(2t-1)$
.		
.		
.		
h	$2t^h + t^{h-1} + t^{h-2} + \dots + t$	$(2t^h + t^{h-1} + t^{h-2} + \dots + t)(t-1)+t$ $= t^h(2t-1)$

Sum these up and simplify to get:

$(2t-1) (1+ t + t^2 + \dots + t^h)$
$= (2t-1) [(t^{h+1}-1) / (t-1)]$

Solution 2.

(5 points)

Note that the k children of B_k are $B_{0}, B_{1}, \ldots, B_{k-1}$.

The root node of B_k has exactly k children.

Let n_k denote the number of nodes with exactly 5 children in B_k.

Then,

$$n_k = I(k) + n_{k-1} + n_{k-2} + \ldots + n_{0}$$

(Here $I(k)=1$ if $k==5$. Else $I(k)=0$.)

(5 points)

Here is the computation:

k	n_k
0	0
1	0
2	0
3	0
4	0
5	1
6	1
7	2
8	4
9	8

10	16
11	32
12	64

Note. If answer is 128, 5 points.

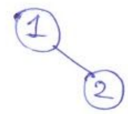
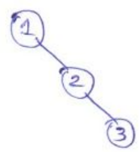
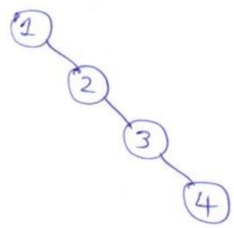
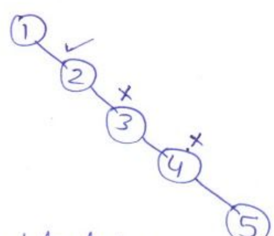
<u>Solution 3.</u>

Correct final tree. 5 points.
Justification / Steps. 5 points.

Any attempt at illustrating the steps, with incorrect scapegoats and rebalancing operations. ~3 points.

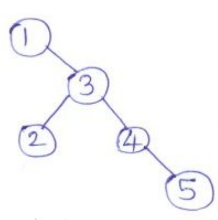
SOLUTION 3

4

		n	$\log_{3/2} n$
(i)		1	0
(ii)		2	1.7
(iii)		3	2.7
(iv)		4	3.41
(v)		5	3.96

need to find scapegoat.

2 is the scapegoat



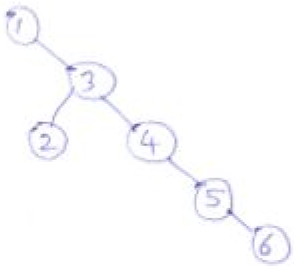
a = 4

SOLUTION 4

Ans 4 (contd)

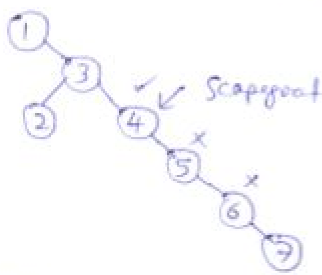
n $\log_{\frac{3}{2}} n$

(vi)

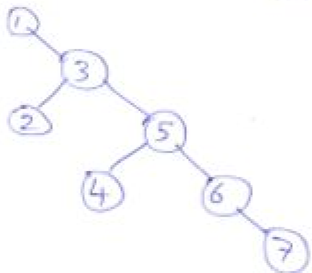


6 4.41

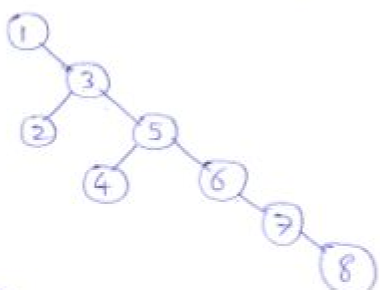
(vii)



7 4.79

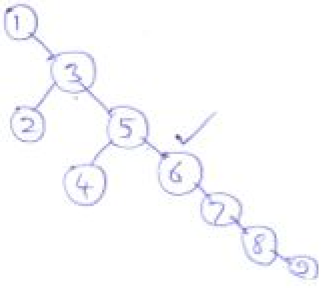


(viii)



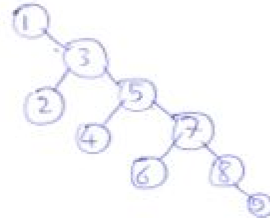
8 5.12

(ix)



9 5.41

FINAL TREE



Solution 4.

Correct answer for both even as well as odd paths. (9 points)

n=2k+1, k>0. Final height is k+1.
n=2k, k>1, Final height is k+1.

For n=1, final height is 0, and for n=2, final height is 1.

Note. A few correct examples, but incorrect general answer, will get around 4 points. Approximately correct general answer, supported by evidence, will get around 6 points.

Proof by induction (6 points)

Odd paths:

Consider the following statement:

P(2k+1), k>0:

Consider a splay tree, which is a path on 2k+1 keys. Then, searching for key 1 converts it into a splay tree T_{2k+1} such that: (i) 1 is the root of the tree. (ii) the left child of 1 is null and the right child is a tree of height k.

Base case: P(3) is true.

Inductive hypothesis: If P(2k+1) is true, then P(2k+3) true. (Reason: Each step is a zig-zig step)

Even paths (2k, k>1):

Now consider a path of length 2k, where k>=1. Splaying key 1 in this path, leads to a sequence of k-1 zig-zig steps, followed by a zig step. After the first (k-1) zig-zig steps, by P(2(k-1)+1):

(i) the right child of the root node is null, and
(ii) the left child of the left child of the root node is null, and the right child of the left child of root node has height k-1.

Thus, after the zig step, the height of the final tree equals k+1.

Note. Around 12 points for giving the final, detailed structure of tree for even as well as odd values of n. However, not giving the final expression for height will lead to deduction of 2 marks from this score.

Solution 5.

First way to count.

(2 points)
When we go down by 1 step in an AVL tree, the height of the subtree decreases by either 1 or 2.

(4 points)
We have to go down by 8 steps, to reach level 8. Each such path is labeled by a unique sequence of -1’s and -2’s

(6 points)
Count such paths using the following table:

Valid	# of -2’s	# of -1’s	# of paths
Yes	0	8	C(8,0)=1
Yes	1	7	C(8,1)=8
Yes	2	6	C(8,2)=28
Yes	3	5	C(8,3)=56
Yes	4	4	C(8,4)=70
No	5	3	...
...			

(3 points)

Total number of paths
= 1+8+28+56+70 = 163.

Second way to count.

(3 points)
Recall Quiz 1 question. Since AVL tree has height exactly 12, the minimum length of a root-to-leaf path is 6. Thus, T is a complete tree till level 6, and at this level it has exactly 2^6=64 nodes.

(2 points)
When we go down by 1 step in an AVL tree, the height of the subtree decreases by either 1 or 2.
Each such path is labeled by a unique sequence of -1's and -2's

(Table 1: 4 points)
(Table 2: 3 points)

We have to go down 6 steps:

Residual height	# of -2's	# of -1's	*Maximum* number of such nodes
6	0	6	C(6,0)=1
5	1	5	C(6,1)=6
4	2	4	C(6,2)=15
3	3	3	C(6,3)=20
2	4	2	C(6,4)=15
1	5	1	C(6,5)=6
0	6	0	C(6,6)=1

Table 1

Residual height	Level needed	Minimum # of nodes
0	2	0
1	2	0

2	2	1
3	2	3
4	2	4
5	2	4
6	2	4

Table 2

(3 points)

Thus, minimum number of nodes at level 8:

4*(1+6+15)+3*20+1*15=163.

Solution 6.

(2 points)
As each new key inserted is the smallest key in the heap till now, it will bubble up till the root location.

(2 points)
Finding location of key i in original heap H, as well as the depth of the subtree with this key as the root.

(3 points)
Define:

pos(t1, w):

Input situation: Key k is at the root of a subtree of heap H of height w.
At the lowermost level of this subtree, t1 leaves have been inserted.

Output: Final location of key k, after all keys in its subtree have been inserted.

(5 points)
Pseudo-code:

```
pos(t1,w):
        Three cases:

        (i) 0 <= t1 < 2^w
                Go to left child of key k.
                Set t1=t1+1.
                w=w-1
                Repeat.

        (ii) 2^w <= t1 < 2^{w+1}
                Go to right child of key k.
                Set t1= t1 - 2^w +1
                    w=w-1
                 Repeat.

        (iii) t1 = 2^{w+1}
                Stop. This is the final position of key k.
```

For A(i,t), just call pos(0, height of subtree rooted at i)

(3 points)

Proving your answer.

- Running time and space. 2 point.
- Correctness. 1 point.

Solution 7.

First solution (uses counting).

(4 points)

A treap is constructed by entering the keys into an empty BST in order of increasing priorities. Every permutation of the n=2^{h+1}-1 keys is equally probable.

(11 points)

Let N(h) denote the number of permutations which generate the given tree of height h.

(correct answer - 7 points)

N(h) = (N(h-1))^2 . C(2^{h+1}-2, 2^h-1) --- (1)

(proof - 4 points)

Proof. Use product rule.

The first entry of the permutation should be the key 2^h.

There are N(h-1) permutations which generate the left subtree, and similarly N(h-1) permutations which generate the right subtree.

Every pair (t1, t2) of permutations from these two sets can be interleaved with each other in C(2^{h+1}-2, 2^h-1)) different ways.

(5 points)

Let P(h) be the probability referred to in the question.

Note that:

P(h) = N(h) / [(2^{h+1}-1)!] ---- (2)

Use (1) and (2) to prove the required formula by induction.

Second solution (uses conditional probability)

Events

E1 = root of tree is key 2^h
E2 = left subtree is a complete binary tree
E3 = right subtree is a complete binary tree

(4 points)
$\Pr(E1) = 1 / (2^{h+1}-1)$

Proof. This is just the probability that the smallest priority key is 2^{h}.

(6 points)
$\Pr(E2 \mid E1) = P(h-1)$

Since all permutations are equally probable, once the first key in the permutation is 2^h, the remaining keys can still appear in any order.

The left subtree depends only on the relative ordering of keys less than root key, and all orderings are equally probable. Hence,

$\Pr(E2 \mid \text{priority of key } 2^h \text{ is } p) = P(h-1)$

(6 points)
$\Pr(E3 \mid E1 \text{ AND } E2) = \Pr(E3 \mid E1) = P(h-1)$

The right subtree depends only on the relative ordering of keys belonging to it, and given that events E1 and E2 have occurred, all such orderings are equally probable.

(4 points)
$\Pr(E1 \text{ AND } E2 \text{ AND } E3) = \Pr(E1) \cdot \Pr(E2 \mid E1) \cdot \Pr(E3 \mid E1 \text{ AND } E2)$

$$= 1 / (2^{h+1}-1) \cdot [P(h-1)]^2$$

Establish the formula by induction.

Note. Correct reasoning using conditional probabilities is worth at least 8 points.