COL 106 Lecture 18

Topic : AVL Trees

Recap:

Binary Search Tree Operations

☑ List in sorted order	(Accessor)	O(n)
☑ Search	(Accessor)	O(h)
☑ Add element (Insertion)	(Modifier)	O(h)
☑ Remove element (Deletion)	(Modifier)	O(h)
☑ Min, Max,	(Accessor)	O(h)
☑ Predecessor, Successor	(Accessor)	O(h).

Problem: $h \approx n$ in the worst case.

Adelson – Velsky – Landis (AVL) Trees

Definition: A binary tree is height-balanced if for every node q, the heights of the subtrees of q differ by at most 1.

Definition: Let $(U, <)$ be an ordered set. A binary tree with keys from U is called an AVL tree if it is a binary search tree and it is height-balanced.

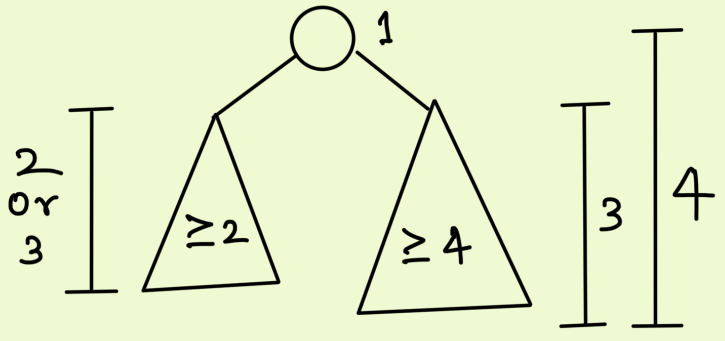
What is the min and max height of a height-balanced tree with n nodes?

Min height: $\lceil \log_2(n+1) \rceil$ – Attained by almost-complete binary trees

(Check: Almost complete $\Rightarrow$ Height-balanced.)

What is the min # nodes in a height-balanced tree of height h?

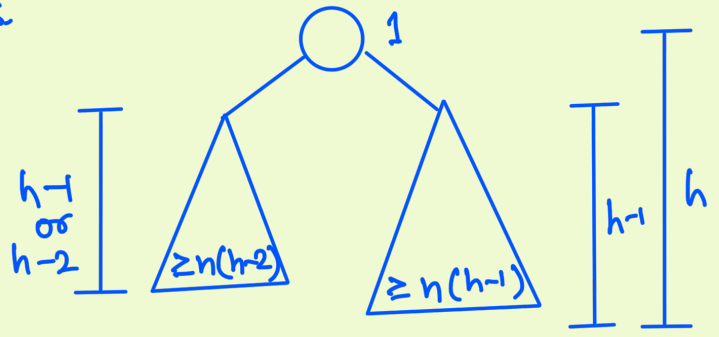
h	min # nodes
0	0
1	1
2	2
3	4
4	4+2+1=7



Claim: Let $n(h)$ denote the min # nodes in a height-balanced tree of height h. Then

$$n(h) = n(h-1) + n(h-2) + 1.$$

Proof: One subtree of root has height $h-1$.
$\therefore \geq n(h-1)$ nodes
The other subtree of root has height $h-1$ or $h-2$.
$\therefore \geq n(h-2)$ nodes



$$n(h) = n(h-2) + n(h-1) + 1$$

Claim: If n is the number of nodes and h is the height of a height-balanced tree, then

$$2^{h/2} \leq n \leq 2^h - 1$$

(proved already)

$$\lceil \log_2(n+1) \rceil \leq h \leq 2 \log_2 n.$$

(proved already)

Proof: $n(h) = n(h-1) + n(h-2) + 1 \geq 2n(h-2)$

If h even $\quad n(h) \geq 2^{h/2-1} n(2) = 2^{h/2-1} \cdot 2 = 2^{h/2}$

If h odd $\quad n(h) \geq 2^{\frac{h-1}{2}-1} n(3) = 2^{\frac{h-1}{2}-1} \cdot 4 \geq 2^{h/2}$

Accessor operations on AVL trees.

- ☑ List in sorted order $O(n)$
- ☑ Search
- ☑ Min, Max,
- ☑ Predecessor, Successor

} $O(h) = O(\log n)$

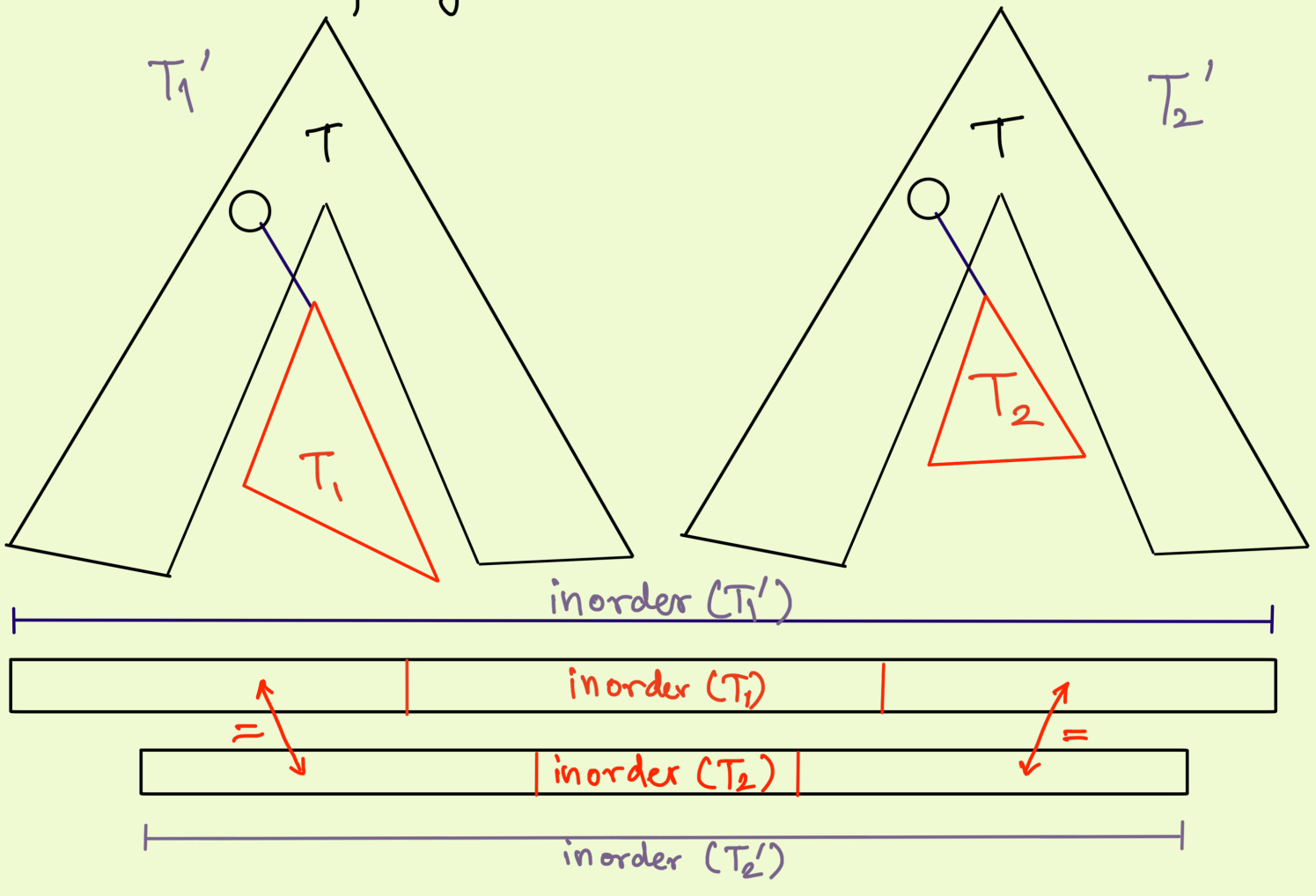
} Exactly like on Binary Search Trees!

Modifier operations on AVL trees

- ☐ Add element (Insertion)
- ☐ Remove element (Deletion)

} 1. Insert / Delete like in a Binary Search Tree
2. Fix imbalances
↳ How?

"Cut-Paste Property" of Inorder Traversal



"Cut-Paste Property" of Inorder Traversal of BSTs.

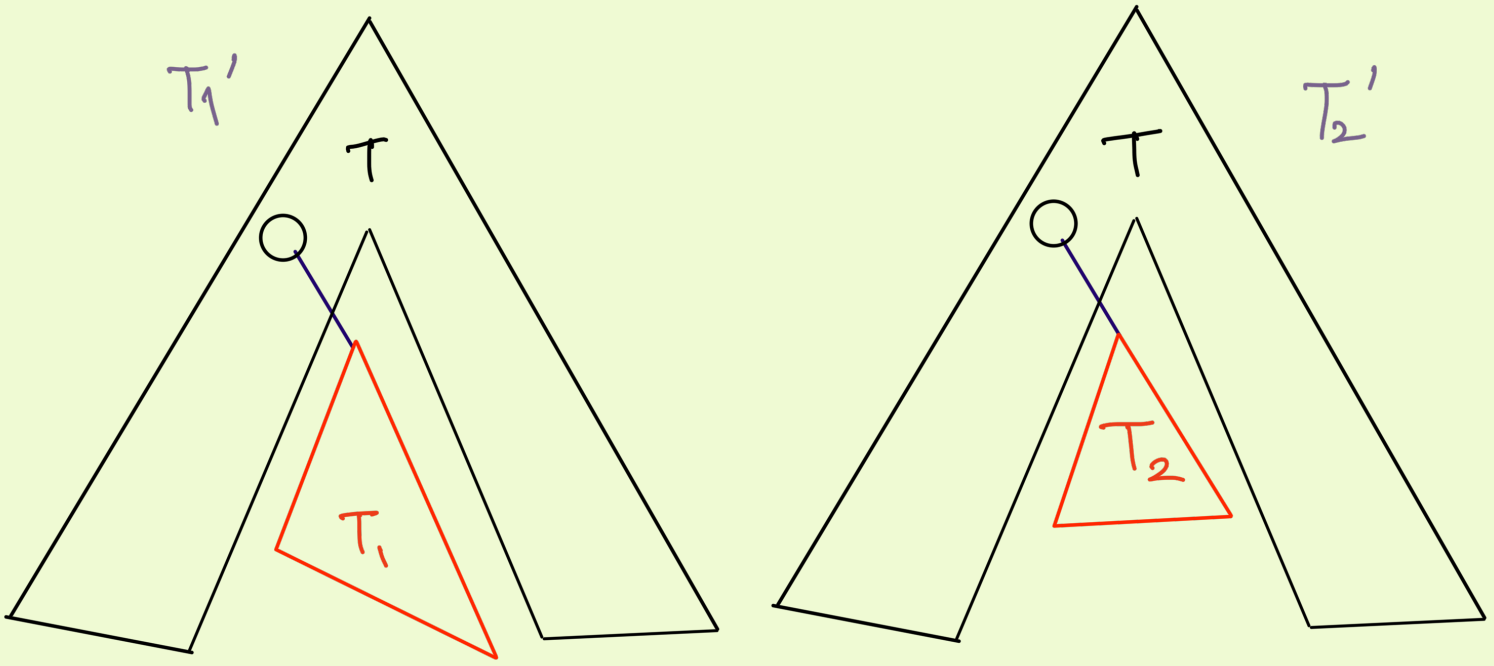


Figure 1

Claim: In the setup of Figure 1, if
(1) T_1' is a binary search tree,
(2) T_2 is a binary search tree on the same keys as T_1,
Then T_2' is a binary search tree on the same keys as T_1'.

Proof: (1) $\Rightarrow$ T_1 is a binary search tree. — (3)

(2) and (3) $\Rightarrow$ inorder (T_1) = inorder (T_2). — (4)

(4) and Cut-Paste property of inorder traversal of arbitrary trees $\Rightarrow$ inorder (T_1') = inorder (T_2'). — (5)

(1) and (5) $\Rightarrow$ T_2' is a binary search tree on the same keys as T_1'.

Note that by defining the inorder traversal of a binary tree, we don't define the structure of the tree