COSC-211

lecture-11

- Dhyey
Mavani

⇒ Accessing Elements in BST.

⊙ **Helpful tool** : print contents of BST in sorted order

1) One approach:

- find smallest element (left most)
- next smallest element is either the parent or smallest descendent of right child.

2) Alternative approach:

- start at root
  1. print everything in left subtree (I)
  2. print root
  3. print everything in right subtree (II)

Subroutine : starting @ v:

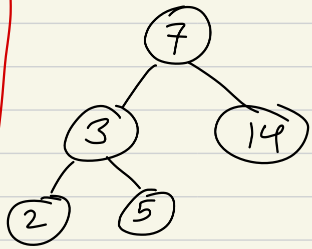
- print left subtree
- print v
- print right subtree

← JAVA CODE

```
class Node<E>{}  ⇐ define it first.
    void printInOrder(){
        if (left != null)            ← print "{"
            left.printInOrder();     ← print "}"
        System.Out.println(this.value);
        if (right != null)           ← print "{"
            right.printInOrder();    ← print "}"
    }
}
```

⇓ Keeping track of depth of tree.

eg.



⇓

Example: CALL STACK: ⇒ ((2)3(5))7(14)

Call's	7.pio	3.pio 7.pio	2.pio 3.pio 7.pio	3.pio 7.pio	5.pio 3.pio 7.pio	3.pio 7.pio	7.pio	14.pio 7.pio	7.pio	∅
step:	1	2	3 ↓ print2	4 ↓ print3	5 ↓ print5	6	7 ↓ print7	8 ↓ print14	9	

⇒ Changing the processing order

→(in-order): process left
process self
process right

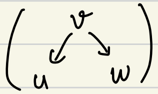
→(pre-order): process self
process left
process right

→(post-order): process left
process right
process self

Q1) What would be printed in pre & post order traversals.

Q2) Write a recursive method to find height
[done in assignment].

⇒ Last Time: AVL trees "height balanced"

→ for every  h(u) and h(w) differ by ≤ 1

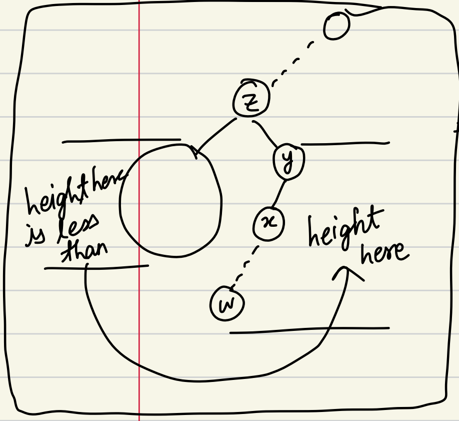
If T is AVL and with n nodes, then $h(T) = O(\log n)$

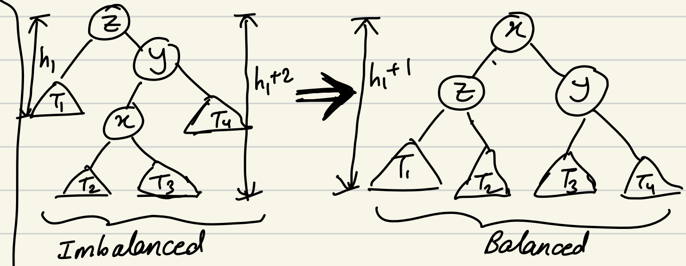
Started: rules for maintaining balance on insertion and removal

Idea for add: create new node "w"

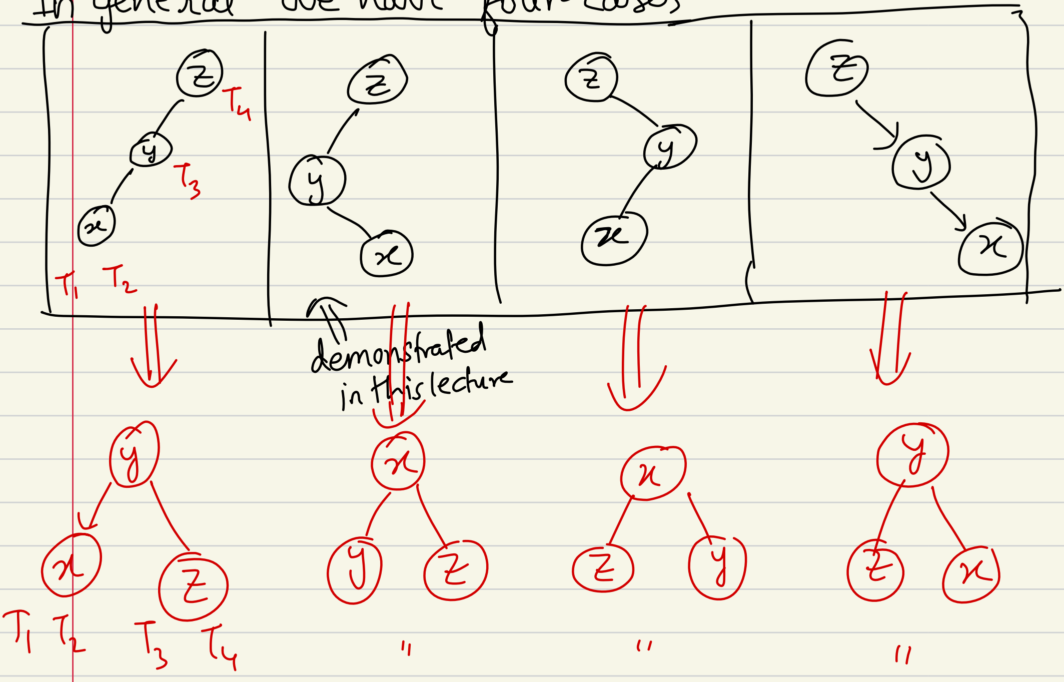
(1) If no imbalance created, then no problem.

(2) Otherwise: z is the first ancestor of w that is imbalanced.





In general we have four-cases



⇒ How to maintain balance with node removal?