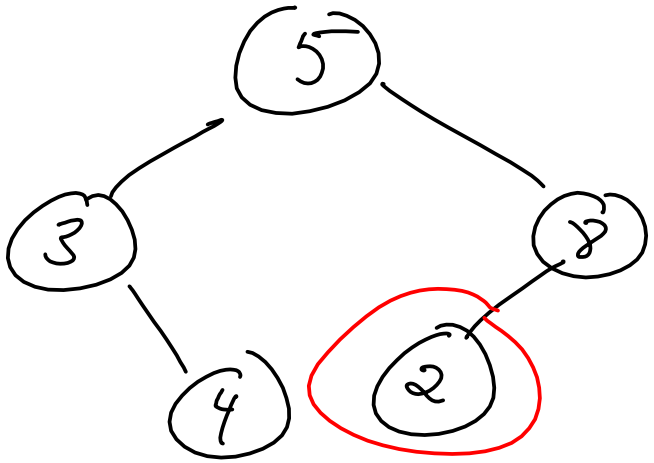
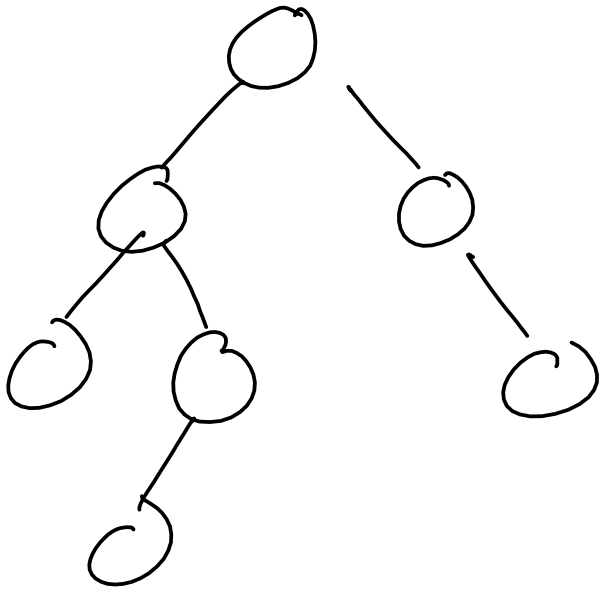
Binary Tree — each node has at most 2 children

BST — binary tree where each node's left subtree has data < that node

likewise for right



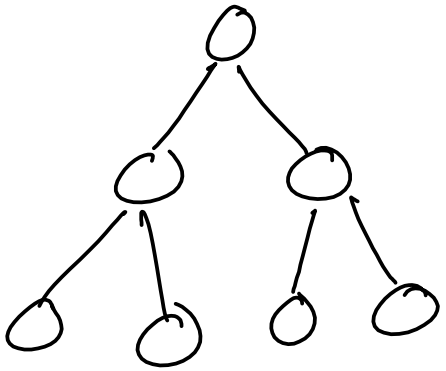
breaks BST invariant



height — # edges between root and furthest leaf

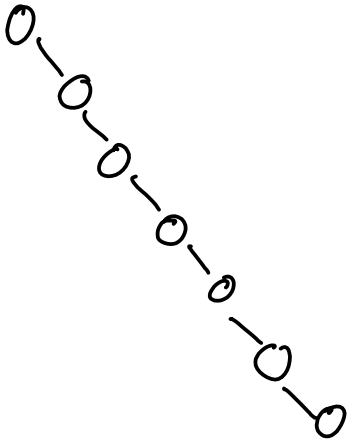
Balancing BSTs

~~O(log n)~~ !!

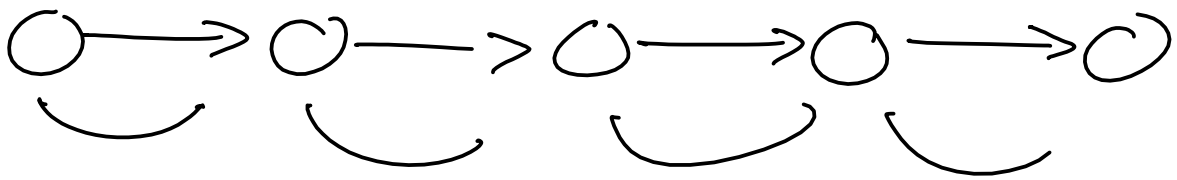


get
insert
update
remove

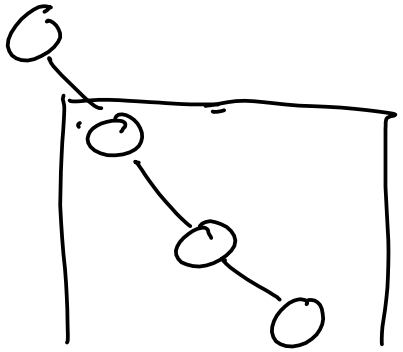
} O(height)



O(n)

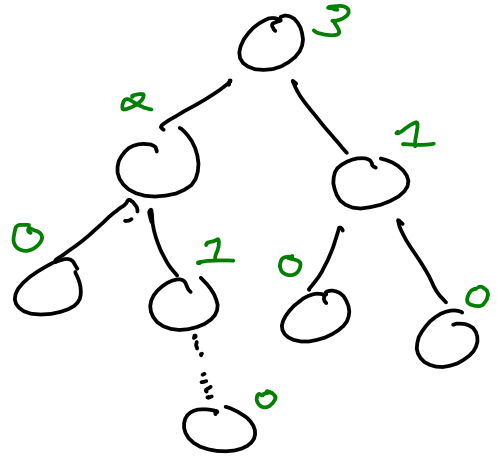


size w/o a field is O(n)



height

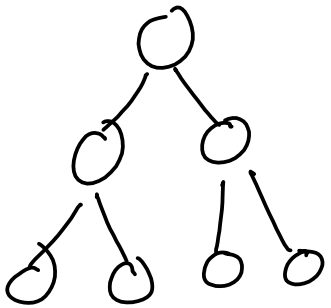
O(n)



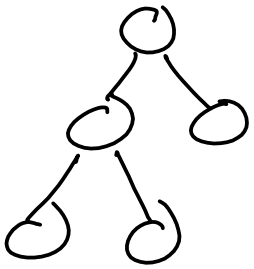
Full

Level 0

Level 1



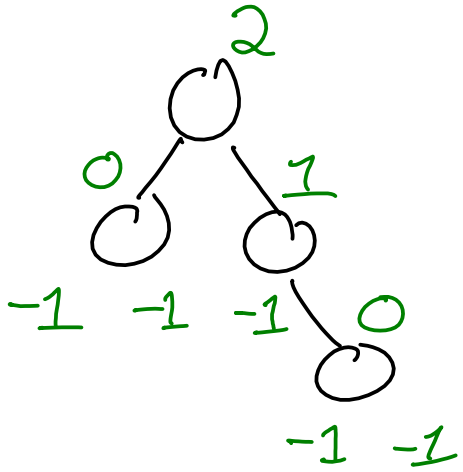
Complete

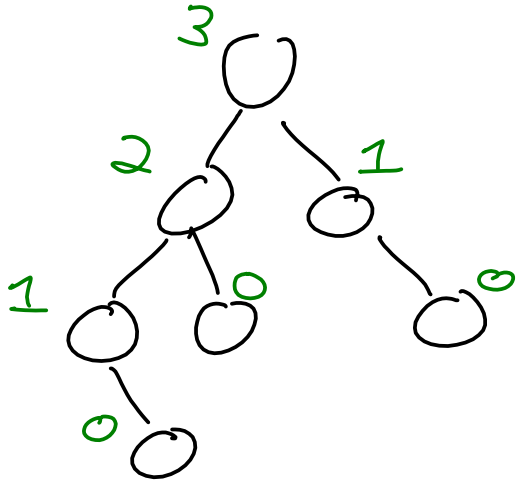


AVL Tree is a BST w/ invariant

Adelson-Velsky
Landis
1962

Every node has left subtree & right subtree which differ in height by at most 1

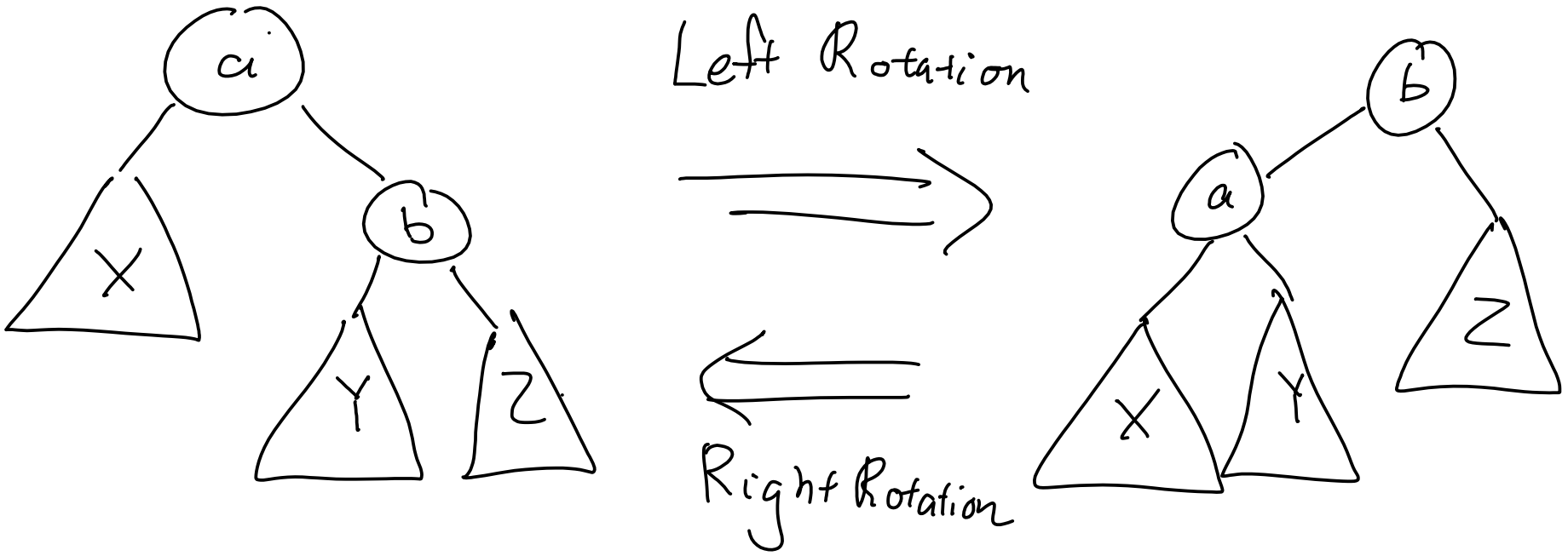




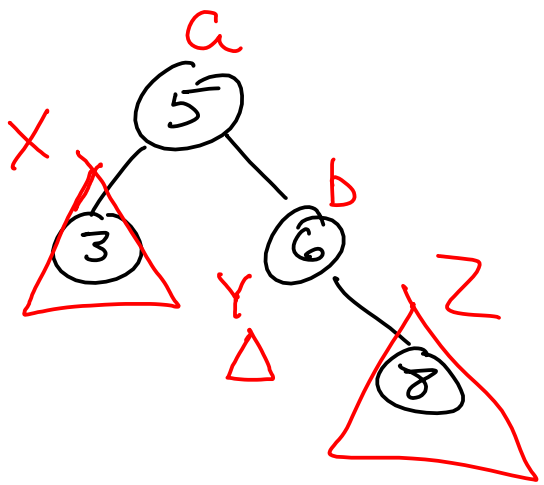
Tree Rotation

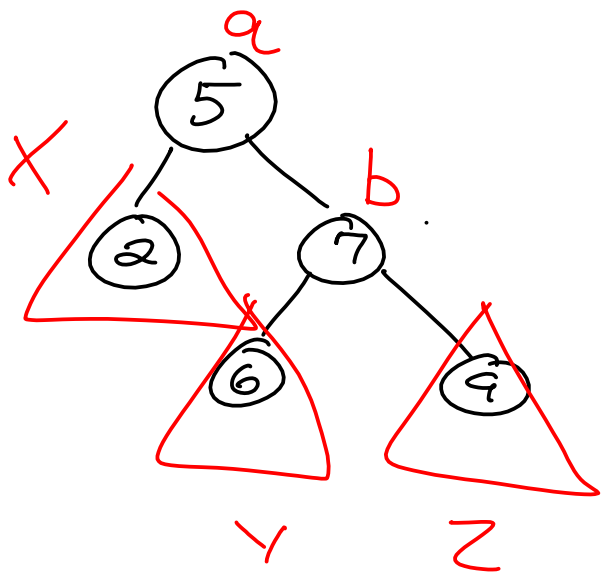
$X < a \quad a < Y < b$

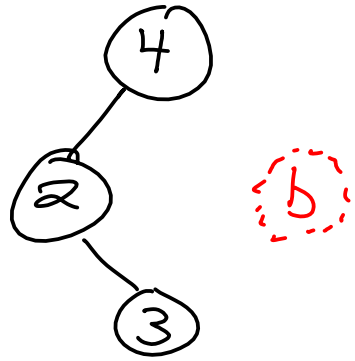
$Z > b$

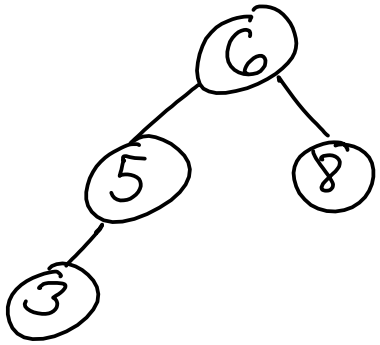


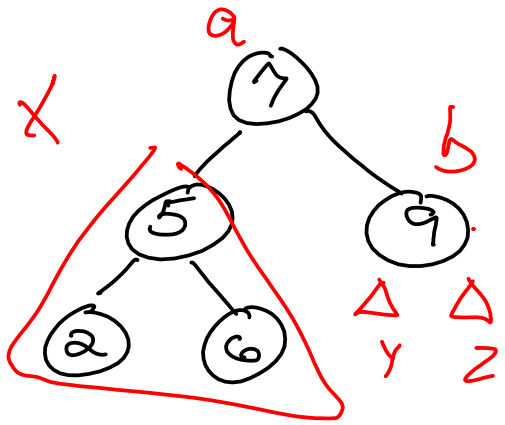
Left -rotate

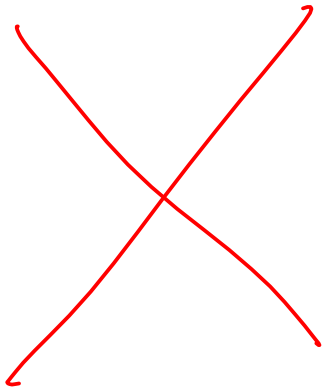


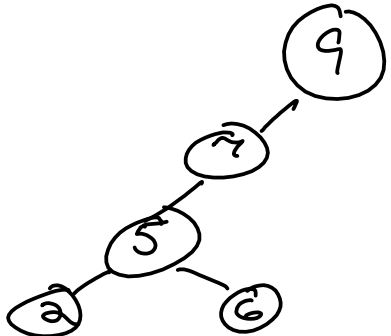












AVL insert

Recompute height

Some Rotations?

```
Function insert(Node node, K key, V value):
    If node is empty tree:
        Return new Node(key, value)
    If key > node->key:
        node->right <- insert(node->right, key, value)
        Return node
    Else If key<node->key:
        node->left <- insert(node->left, key, vale)
        Return node
    Else:
        throw exception!
```