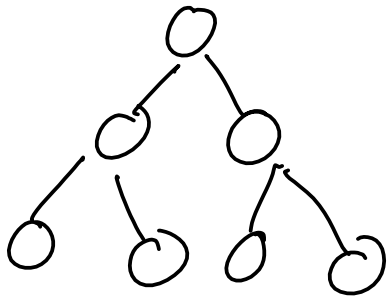
BST each node carrying K, V

left K < this K < right K

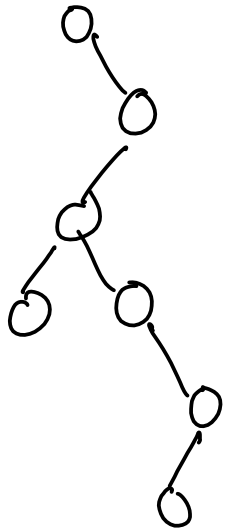
Impl of ADT "dictionary"

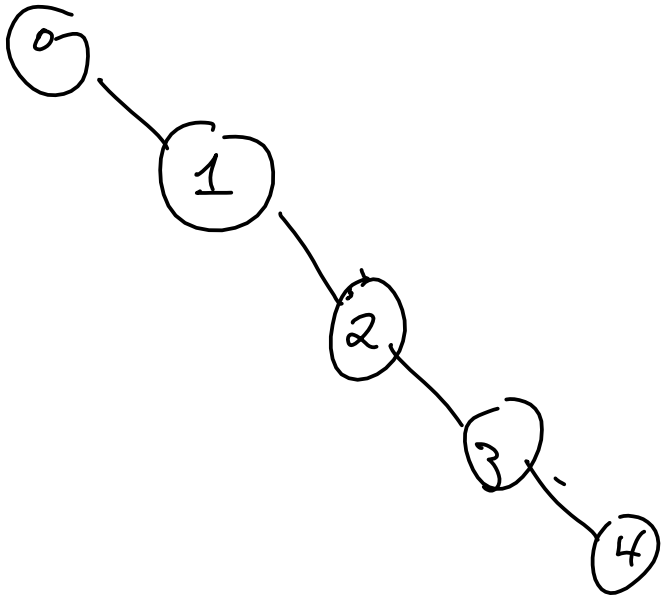


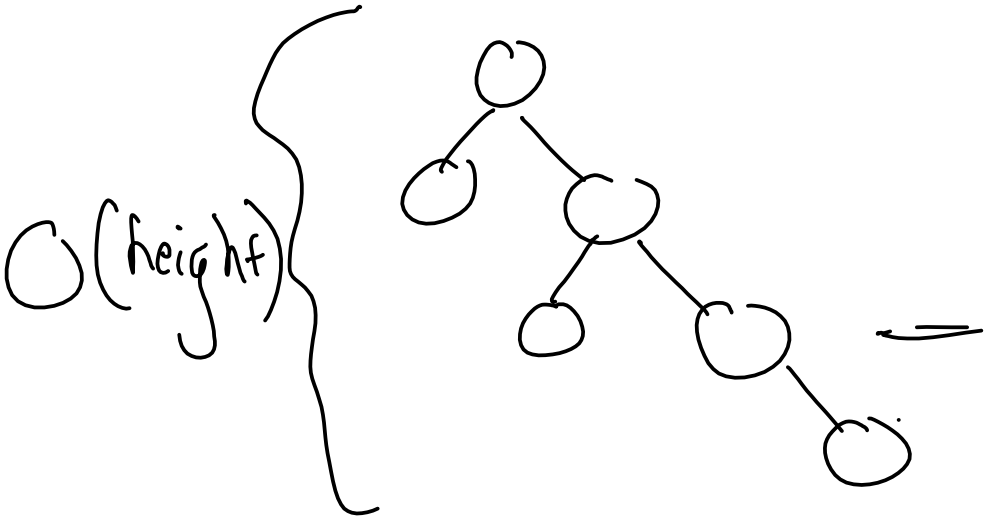
n = 7 ~ 3 steps

$2^3 = 8$

"Full"





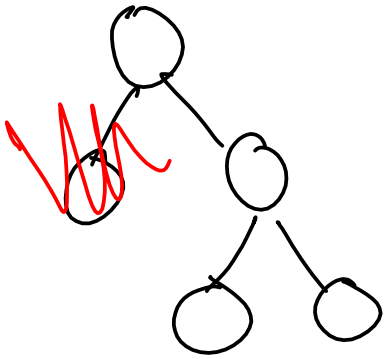


Lookups $O(\log n)$

Updates $O(\log n)$

includes "fixing" the tree

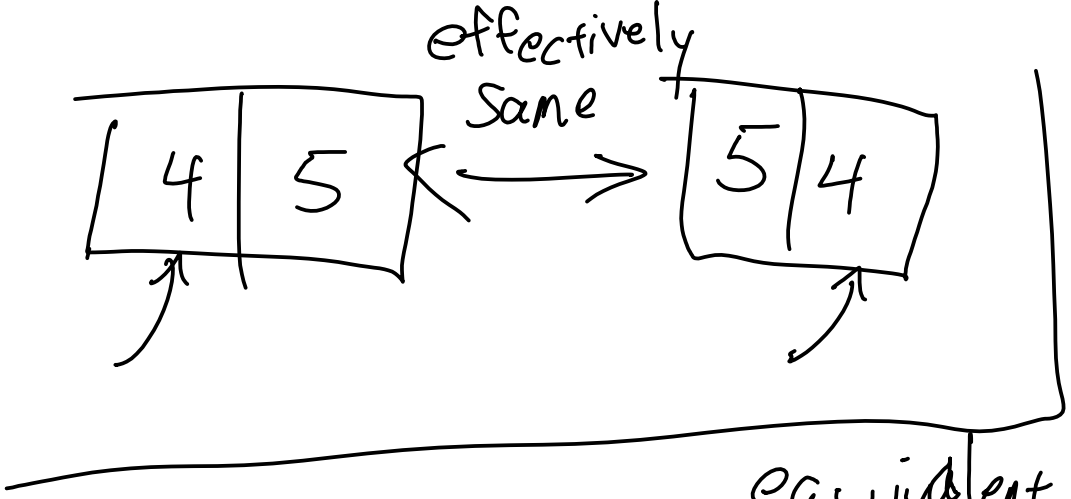
We'd like for height to be $O(\log n)$

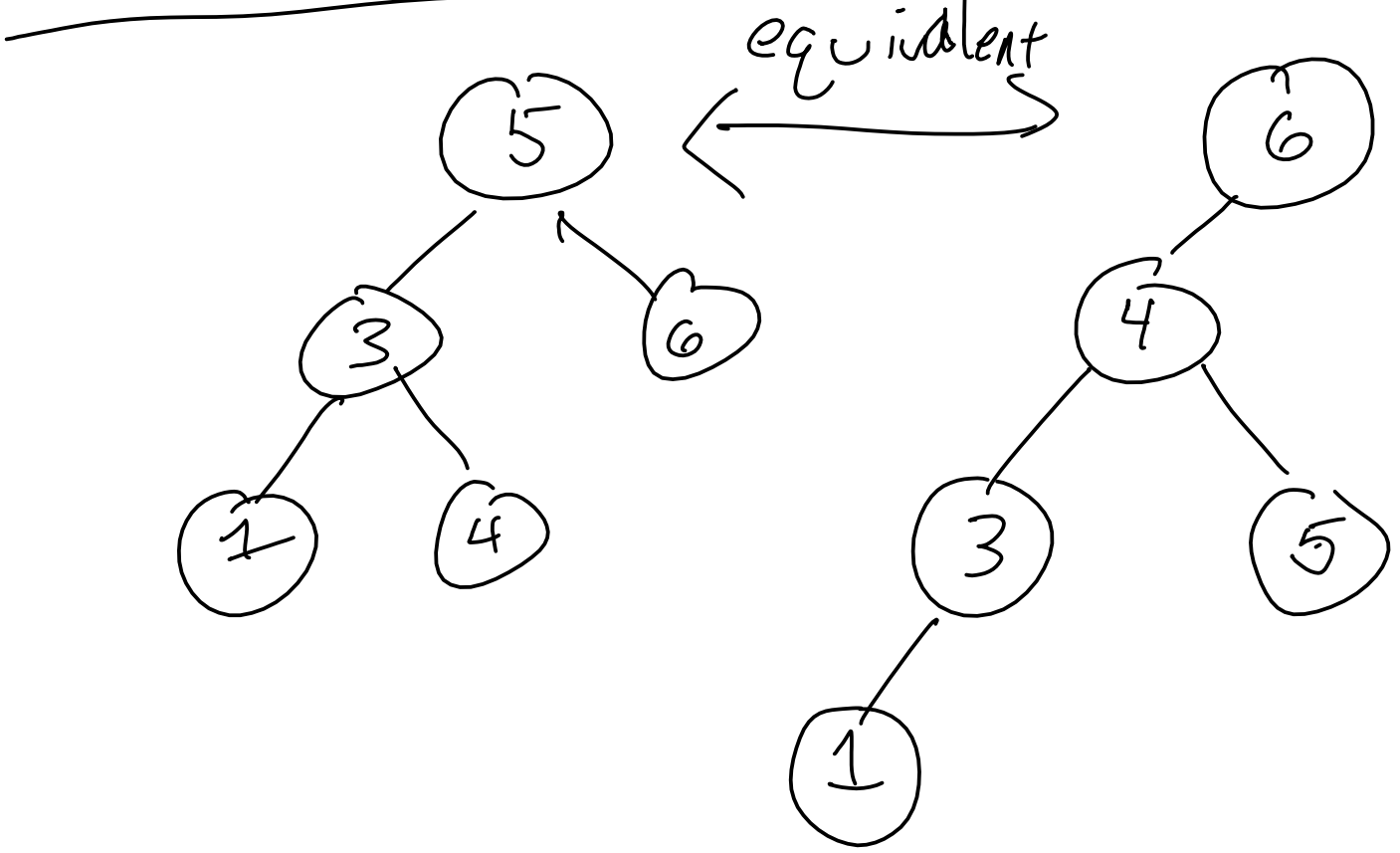


AVL Tree is a BST that stays "balanced"

Balanced: for any node: height(left child) and height(right child) differ by at most 1

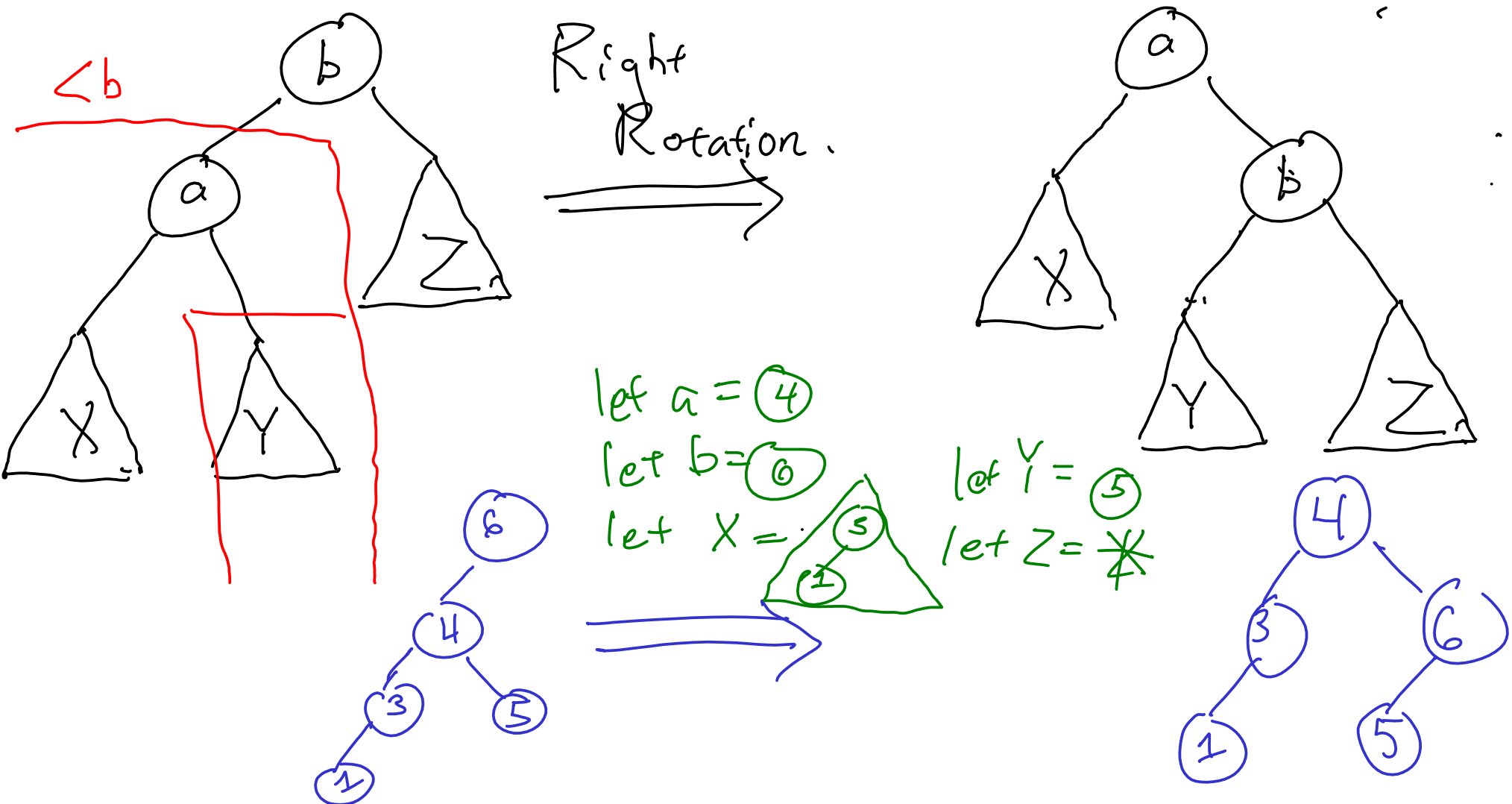
Rotation


effectively
same
4
5
5
4

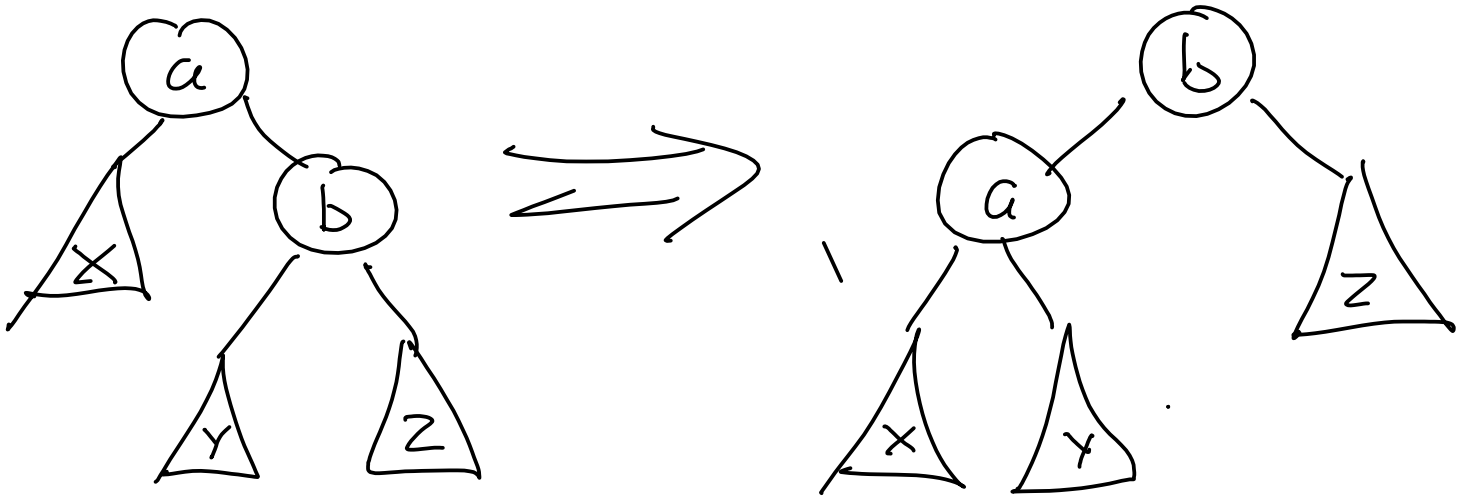

equivalent
5
3
6
1
4
6
4
3
5
1

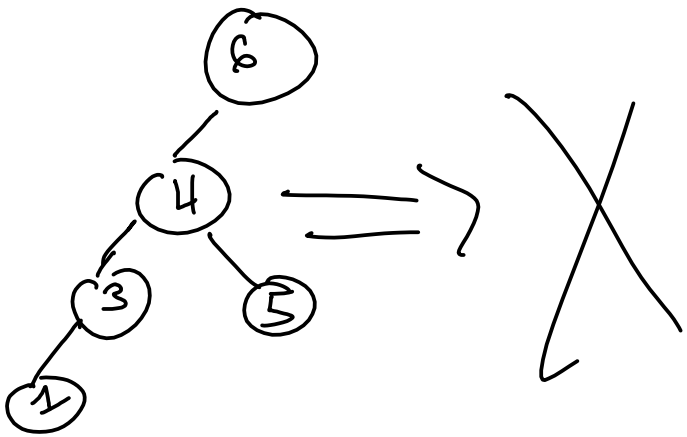
A tree rotation converts one (bin) tree to another

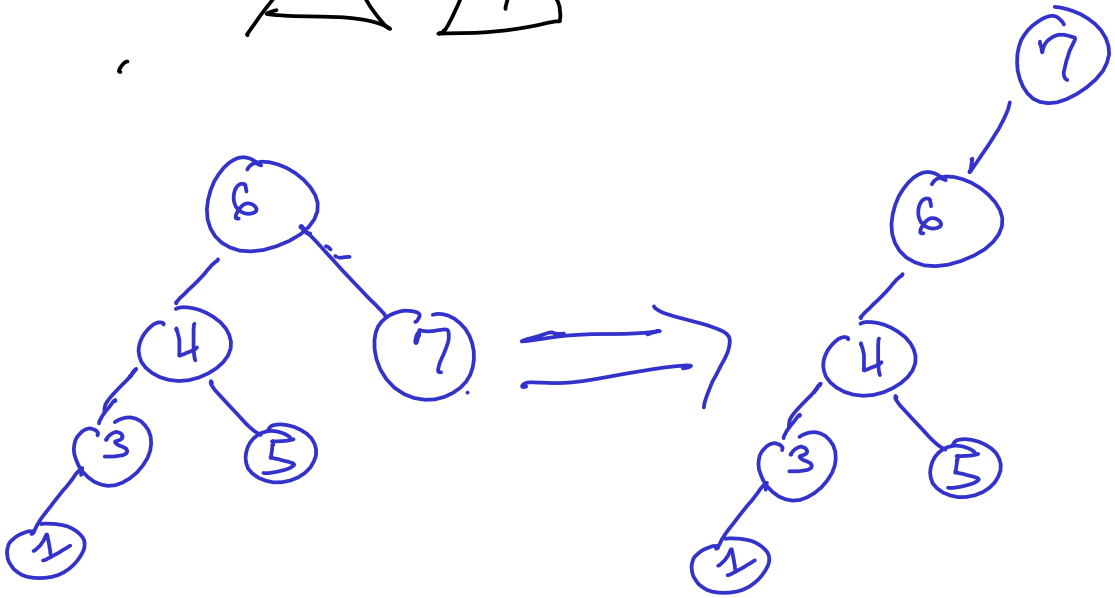
2 kinds: left right



Left Rotation







AVL invariant : $|\text{left height} - \text{right height}| \leq 1$

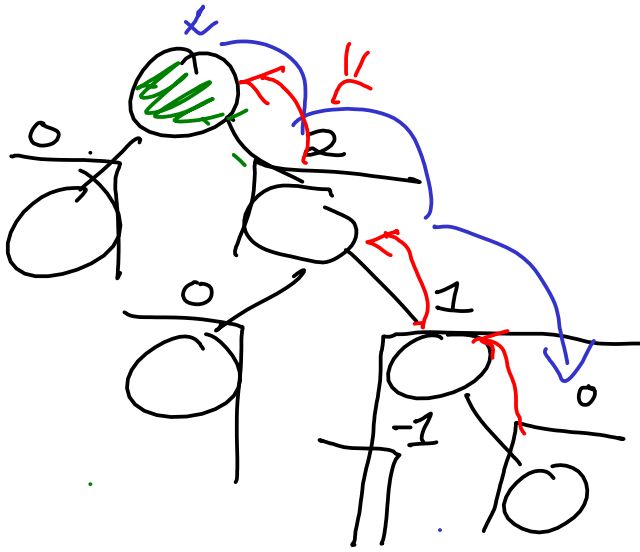
```
template <typename K, typename V>
class AVLTreeNode
    K key;
    V value;
    ... left;
    ... right;
    int height;
```

change when tree changes

add remove

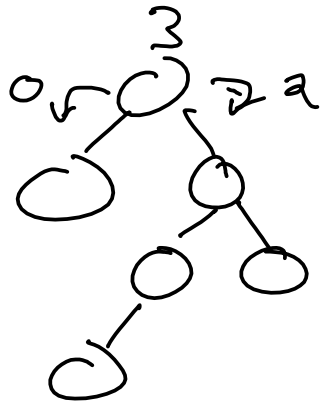
rotate

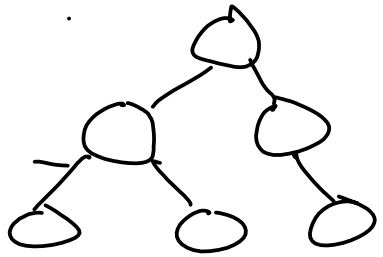
Rebalancing



Method Insert (key, value):

```
If this->key < key:
    If this->right ≠ NULL:
        this->right->Insert(key, value)
        -> rebalance()
    Else:
        this->right = new Node(key, value)
        recompute height
Else:
    ...
```





Left	Difference Old	Difference New
empty	0	1
single	1	0