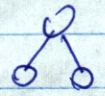
Written

#1 Assume we have n full node in a ~~tree~~ binary tree

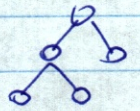
- when n = 1 (base case):



full node = 1

leaves = 2 = full node + 1 ✓

- when n = 2 (base case 2):



full node = 2

leaves = 3 = full node + 1 ✓

- Suppose when n = k, our hypothesis is true
  then when n = k, we have k+1 leaves.
- when n = k+1, it means:
  - we will have two leaves ~~which are grown~~ from a note, which is a leave before
  - then in this case, we will have $k+1 + 2 - 1 = (k+1)+1$ ✓

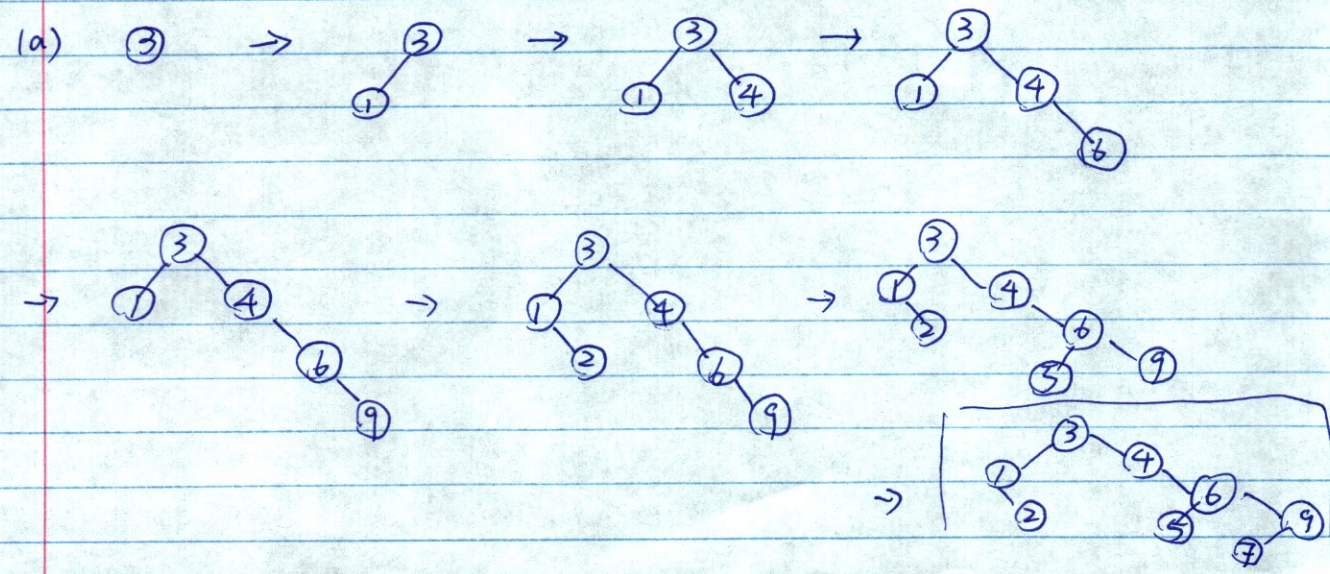
(2 new leaves; old leave, now is a full node)

Then the ~~assum~~ hypothesis is true for $n \in \mathbb{N}^*$

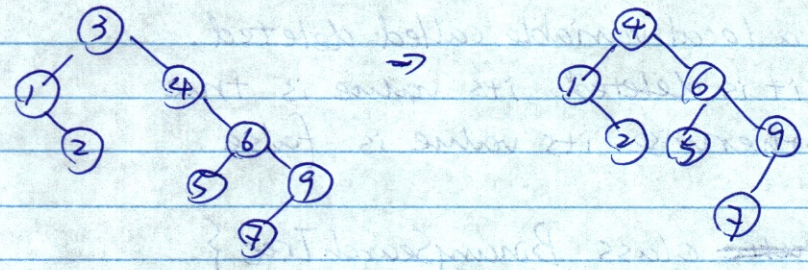
□

#2 3, 1, 4, 6, 9, 2, 5, 7

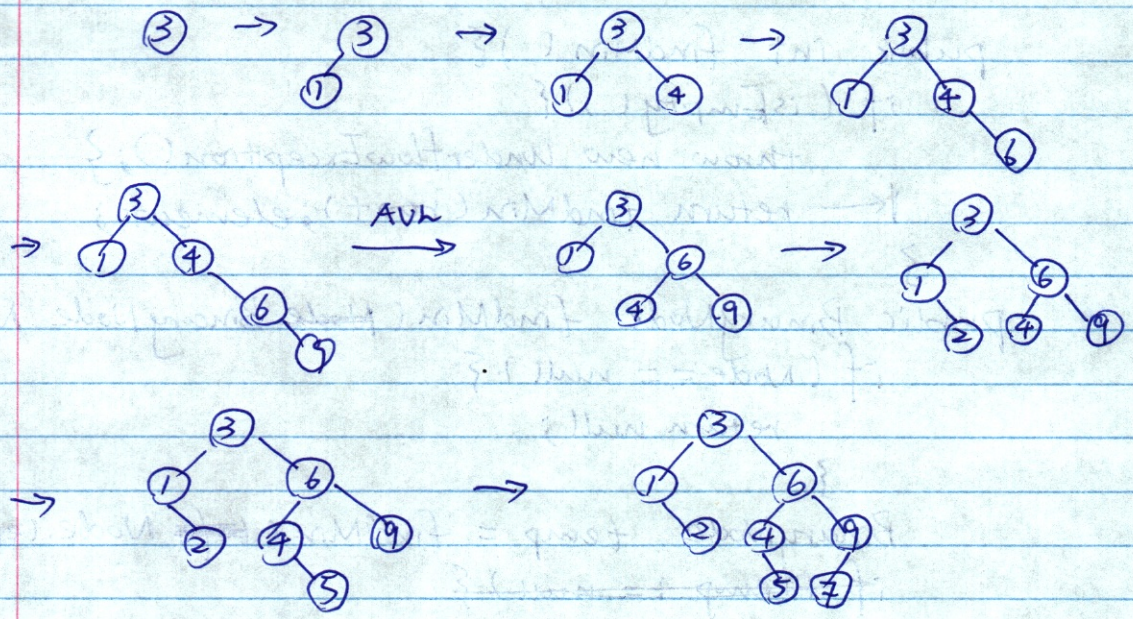
(a)



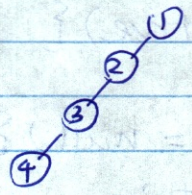
(b)



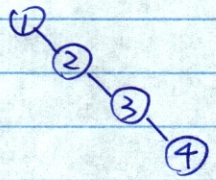
#3 AVL Tree 3, 1, 4, 6, 9, 2, 5, 7



#4



postfix : 4, 3, 2, 1
prefix : 1, 2, 3, 4



postorder : 4, 3, 2, 1
preorder : 1, 2, 3, 4

• postorder and preorder for these two binary trees are identically same. However, they are different binary trees.

[#5] Add a local variable called deleted.
if it is deleted, its value is true
otherwise its value is false

```
public
~~private~~ ~~static~~ class BinarySearchTree {
    private BinaryNode root;
    ~~private deleted boolean deleted~~;

    public int findMin( ){
        if ( isEmpty( )){
            throw new UnderflowException( );}
        ← return findMin( root ); element;
    }
    public BinaryNode findMin( ~~Node~~ BinaryNode Node){
        if ( Node == null ){
            return null;
        }
        BinaryNode temp = findMin( ~~left~~ Node.left );
        ~~if ( temp != null ){~~
        ~~return temp;~~
        if ( temp.deleted != ~~false~~ true ){
            return temp;
        } else if ( temp.right != null ){
            return temp.right;
        } else {
            return null;
        }
    }
    Public static class BinaryNode {
        private ~~deleted~~ boolean deleted; // add
        ⋮ ~~Constructors~~ all other things stay same
    }
}
```