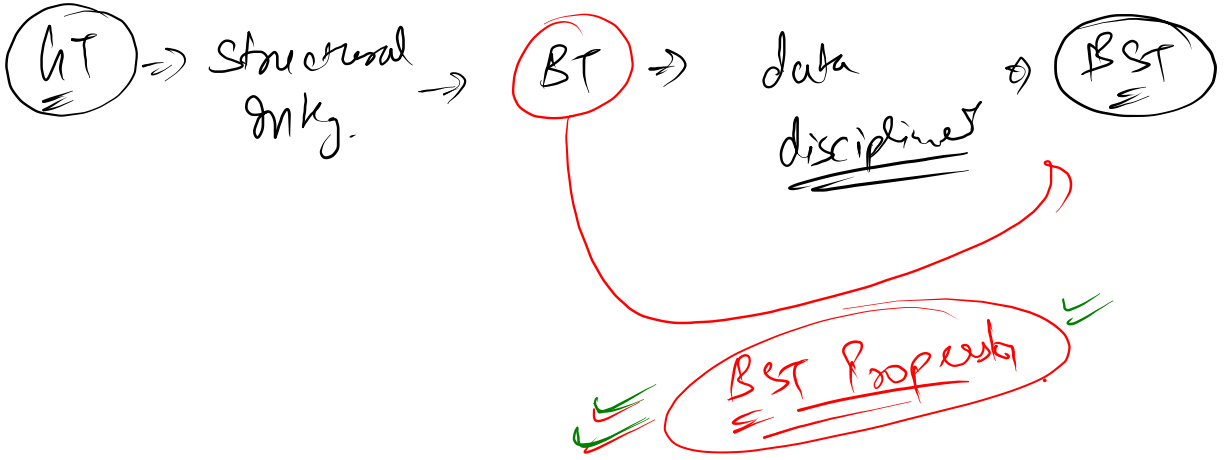
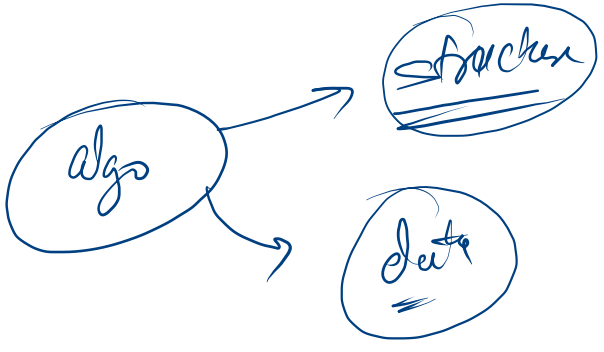
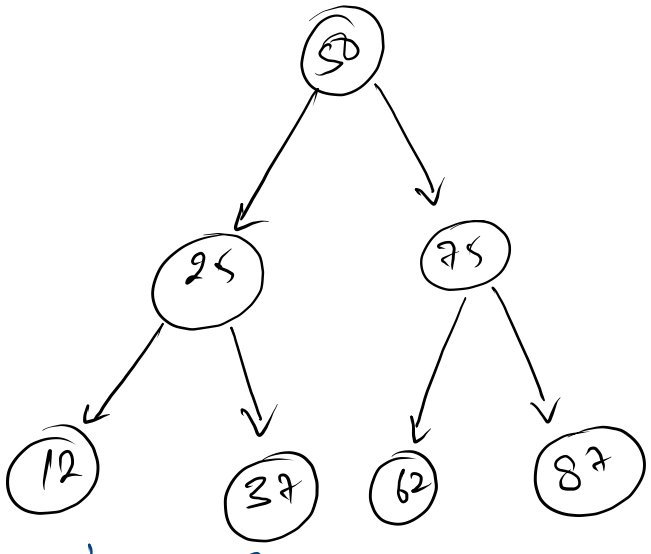
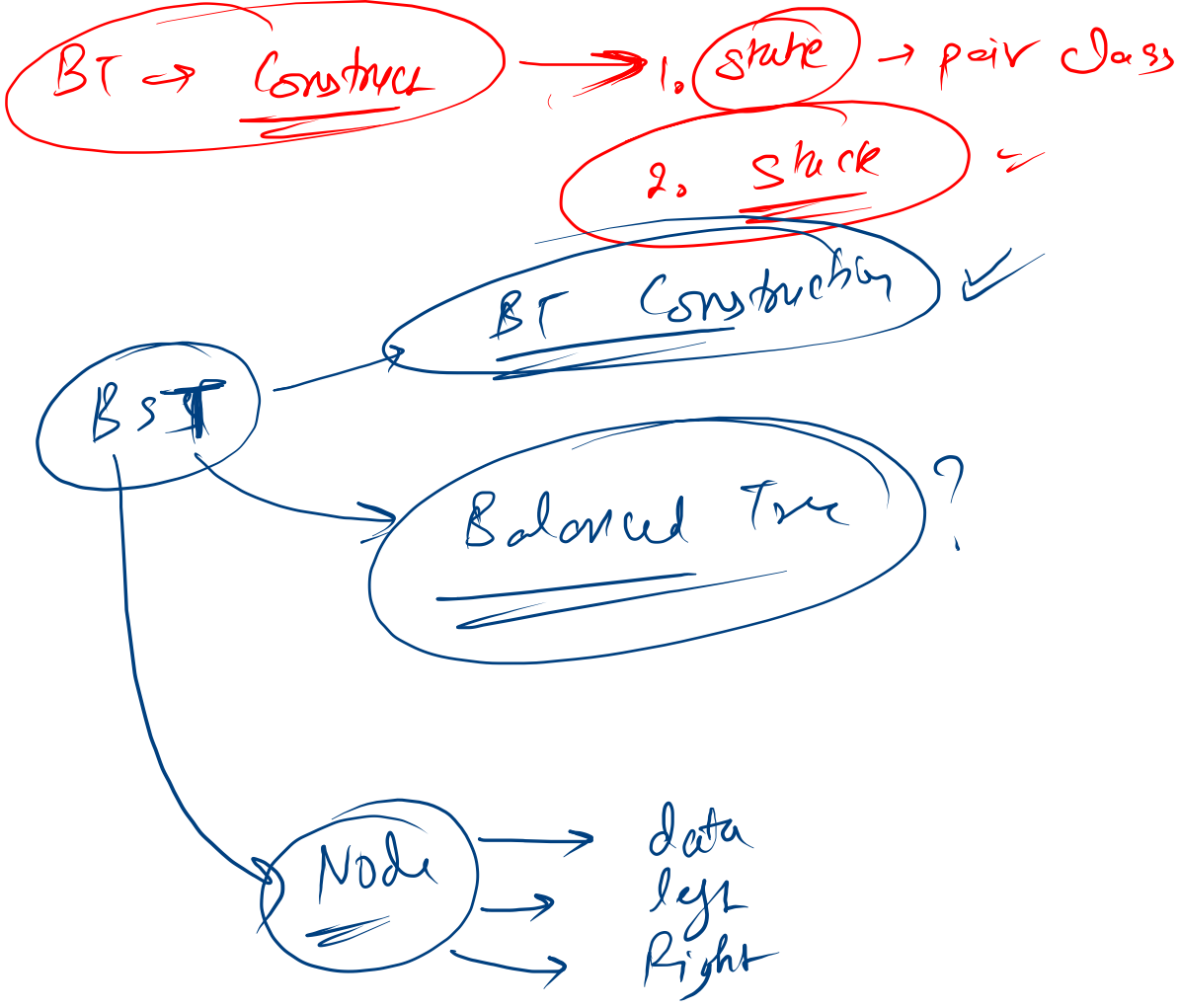

BST
left < root.data < right
BST







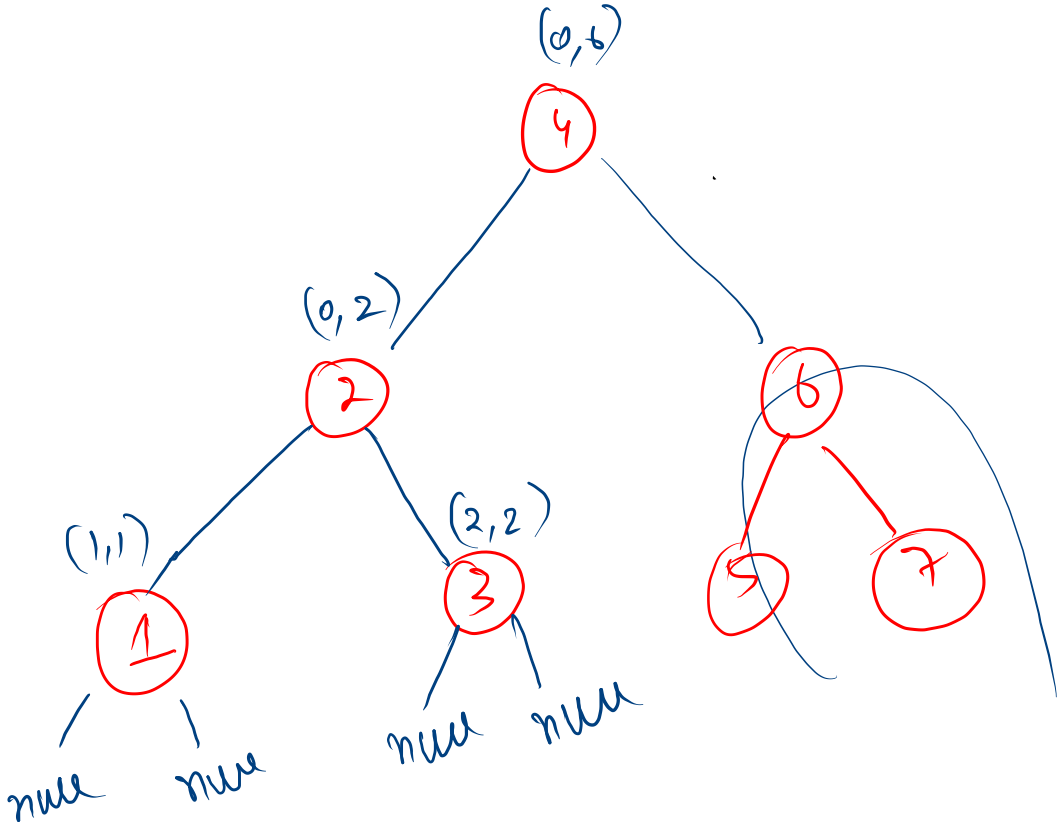
	BT	BST
Size	✓	✓
HT	✓	✓
Display	✓	✓
Max	O(n)	optimization
Min	O(n)	=
find	O(n)	=
NTRP	O(n)	=
LCA	O(n)	=


BT → Construct
1. state → pair Class
2. stack
BT Construction
BST
Balanced Tree ?
Node
data
left
Right

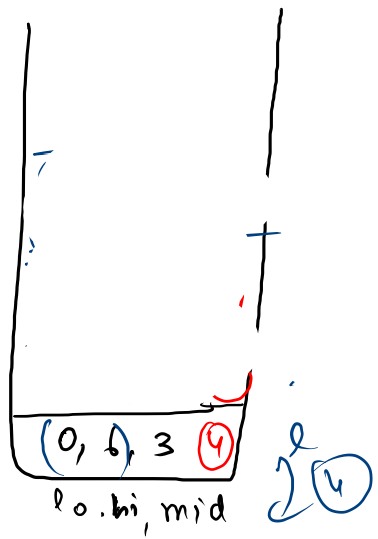
Index →	0	1	2	3	4	5	6
val →	1	2	3	4	5	6	7

Sorted

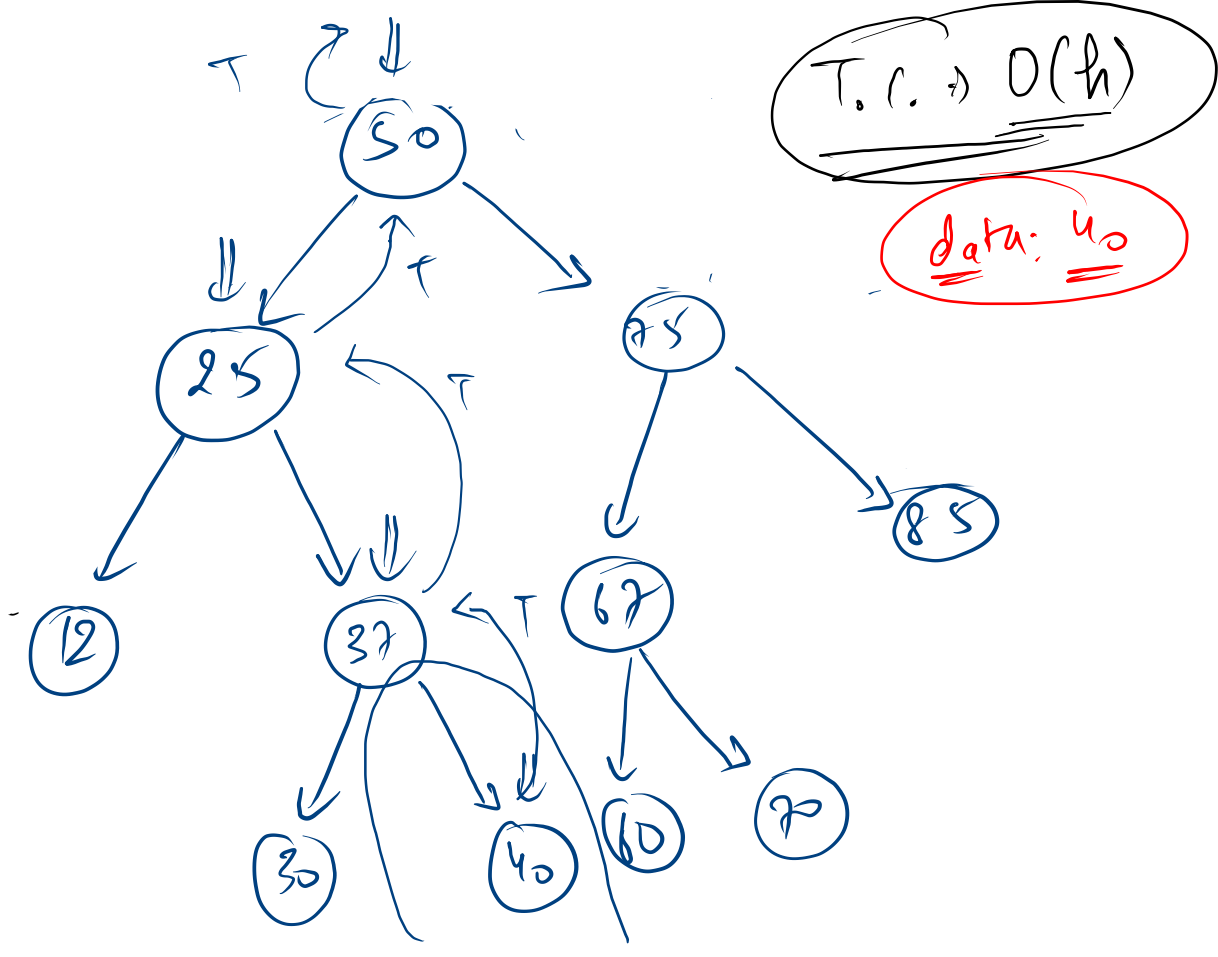
Balanced BST

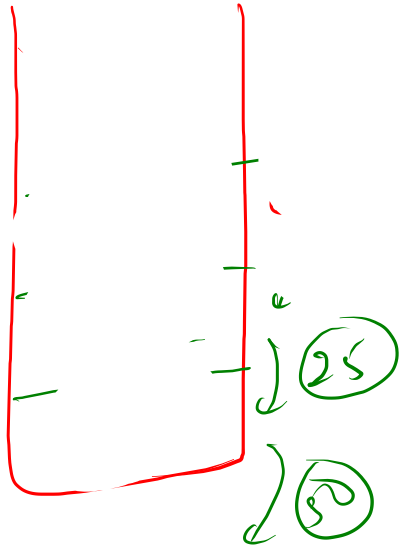


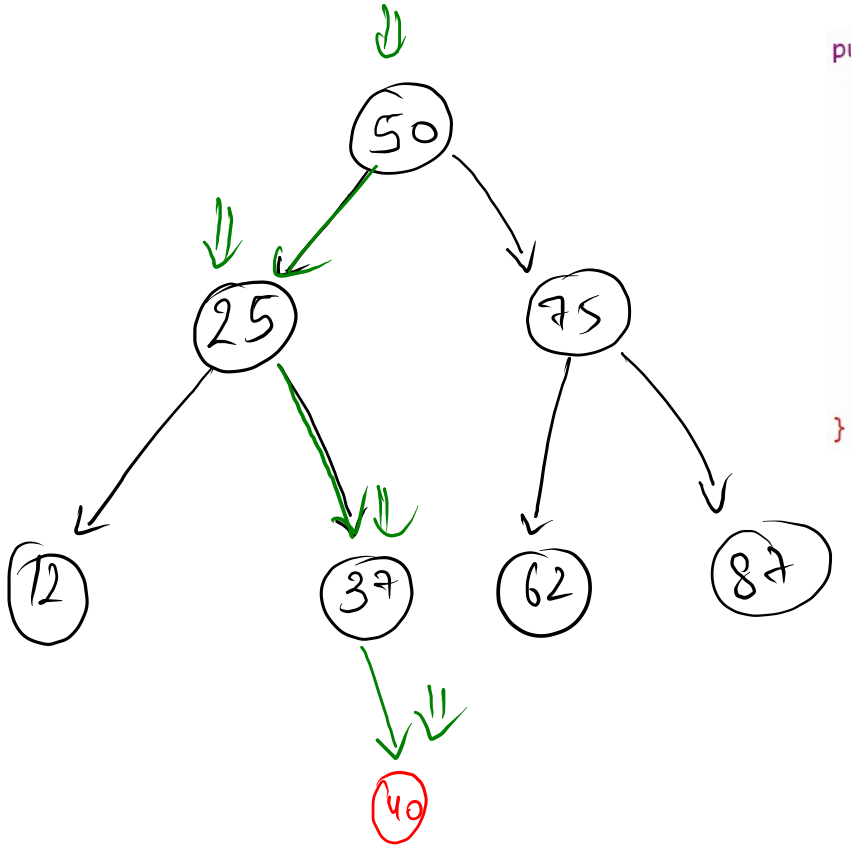

```java
public static Node construct(int []inp,int lo,int hi){
    if(lo > hi){
        return null;
    }
    int mid = (lo+hi)/2;
    Node node = new Node(inp[mid]);

    node.left = construct(inp, lo, mid-1);
    node.right = construct(inp, mid+1, hi);

    return node;
}
```



```java
public static int max(Node node) {
  if(node.right == null){
      return node.data;
  }

  return max(node.right);
}

public static int min(Node node) {
  if(node.left == null){
      return node.data;
  }

  return min(node.left);
}

public static boolean find(Node node, int data){
  if(node == null){
      return false;
  }

  if(node.data == data){
      return true;
  }else if(data > node.data){ // larger - > right
      return find(node.right,data);
  }else{ // smaller - > left
      return find(node.left,data);
  }

}
```



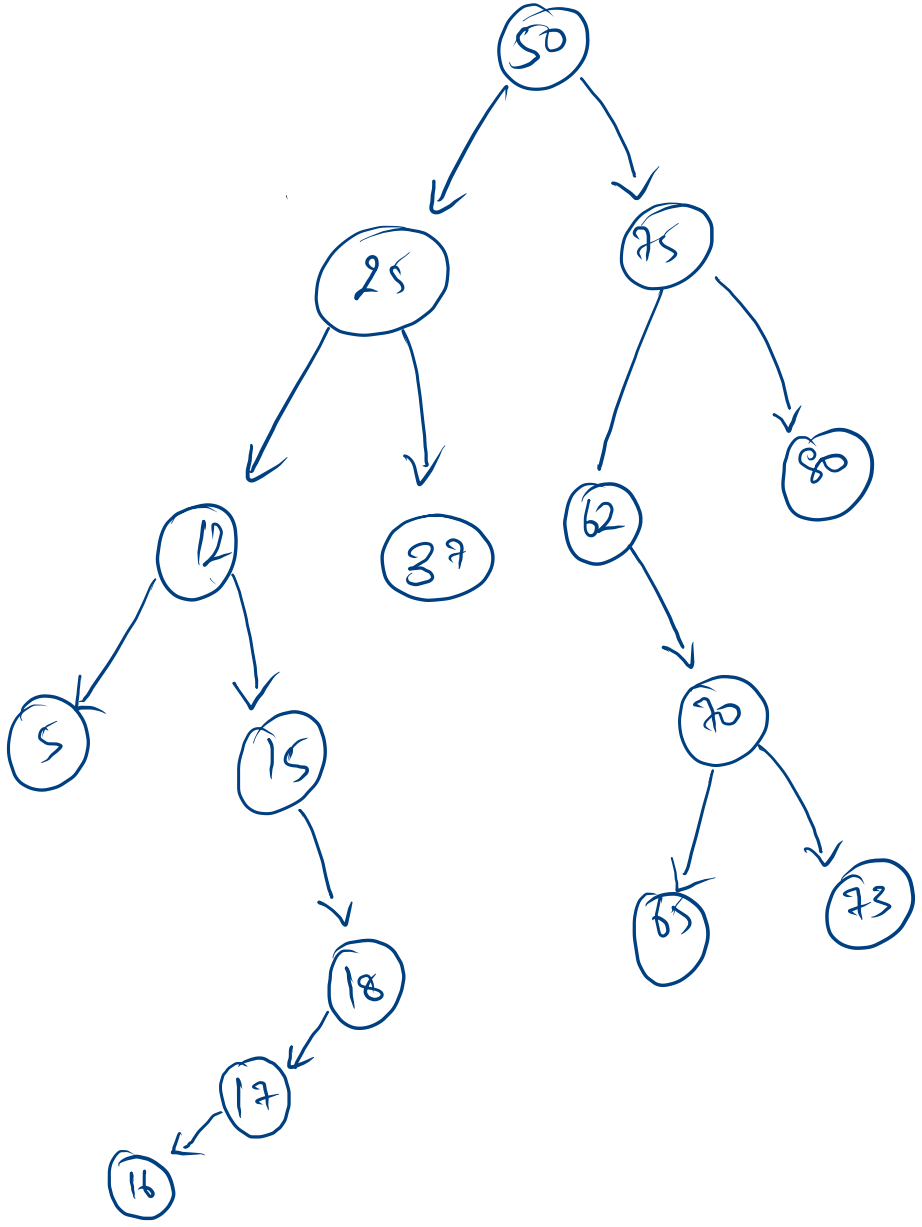





```java
public static Node add(Node node, int data) {
  if(node == null){
      return new Node(data,null,null);
  }

  if(data > node.data){
      node.right = add(node.right,data);
  }else if(data < node.data){
      node.left = add(node.left,data);
  }

  return node;
}
```

→ 1

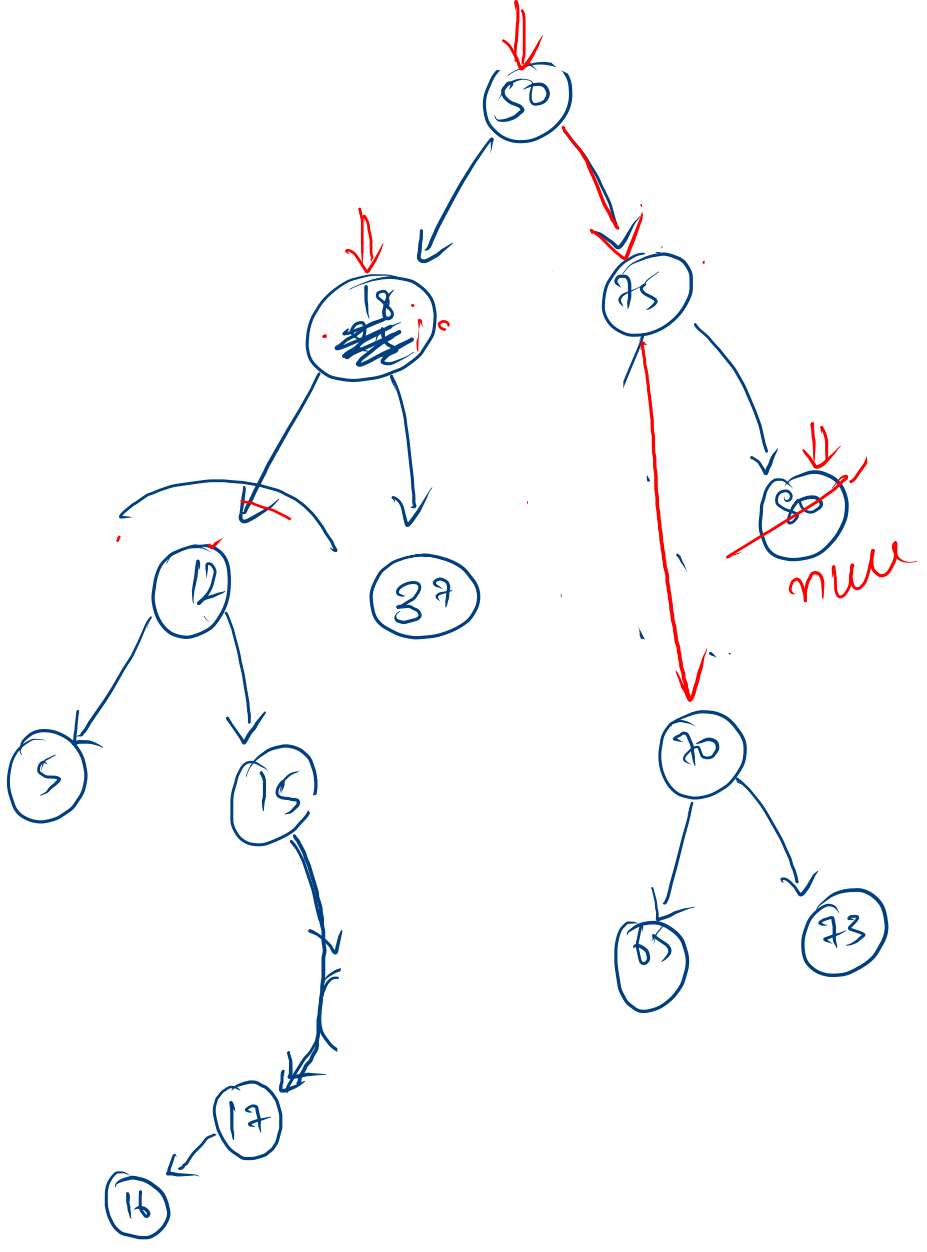
→ 2

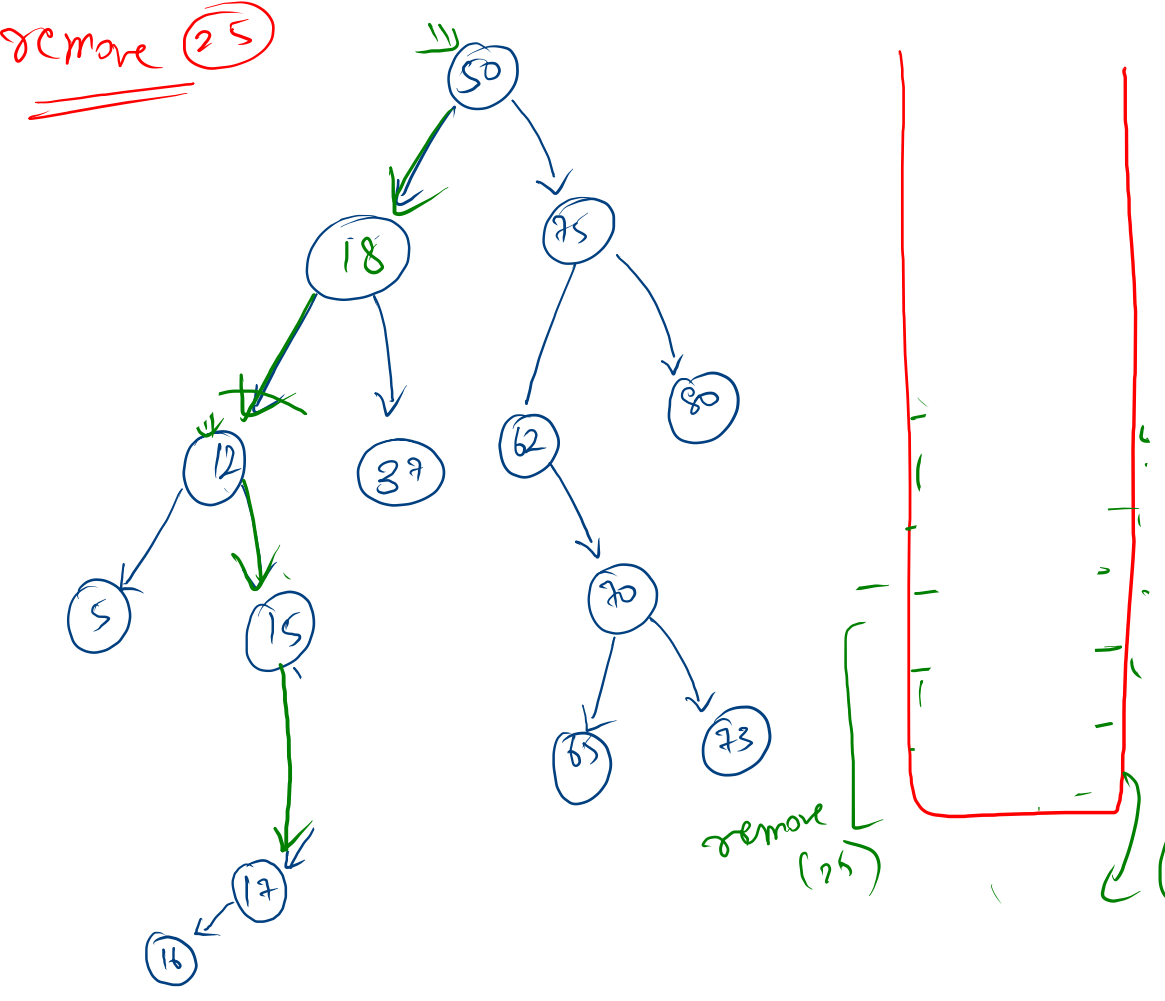

50
25
75
12
37
62
80
5
15
70
18
65
73
17
16

no child ⇒ 80 null

1 child ⇒ 62

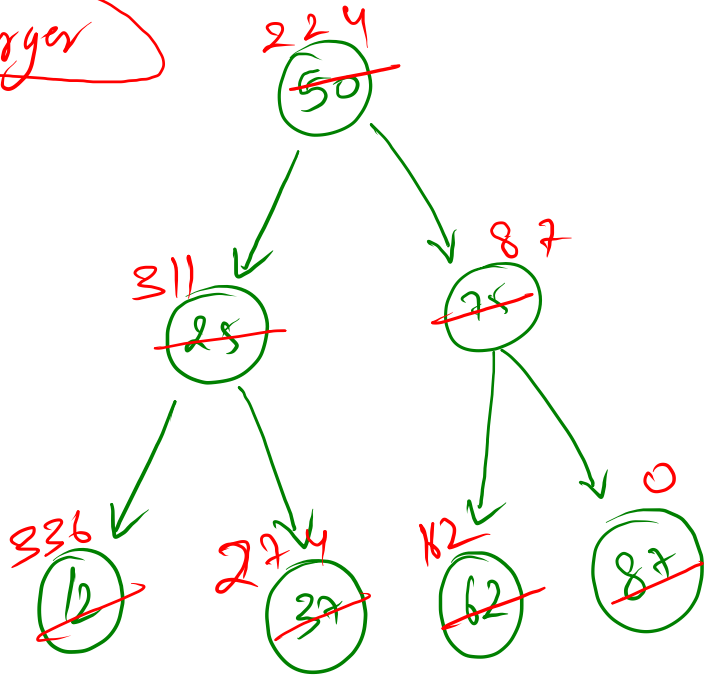
2 child ⇒ 25

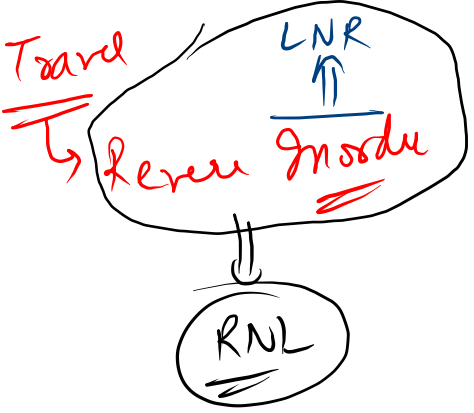


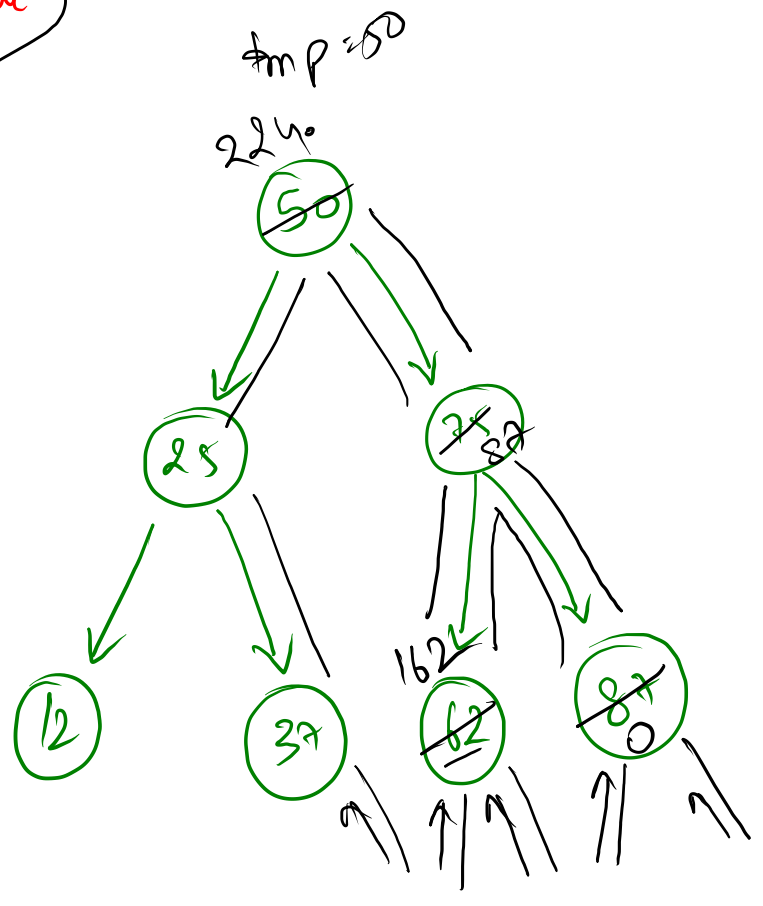


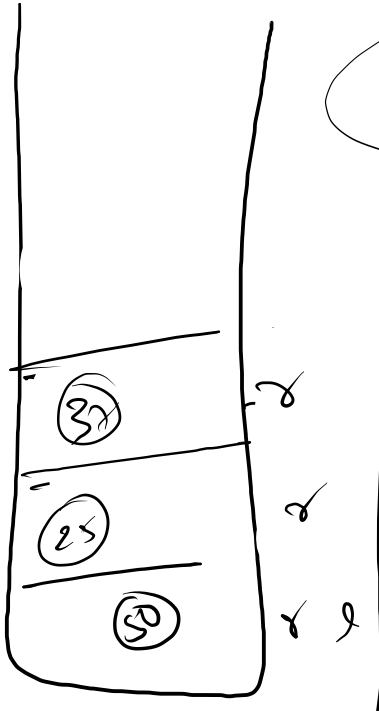

```java
public static Node remove(Node node, int data) {
  if(data > node.data){
      node.right = remove(node.right,data);
  }else if(data < node.data){
      node.left = remove(node.left,data);
  }else{
      Node lchild = node.left;
      Node rchild = node.right;

      if(lchild == null && rchild == null){ // no child
          return null;
      }else if(lchild != null && rchild == null){ // only left child
          return node.left;
      }else if(lchild == null && rchild != null){ // only right child
          return node.right;
      }else{ //both child
          int max = max(lchild);
          node.data = max;
          node.left = remove(lchild,max);
          return node;
      }
  }

  return node;
}
```

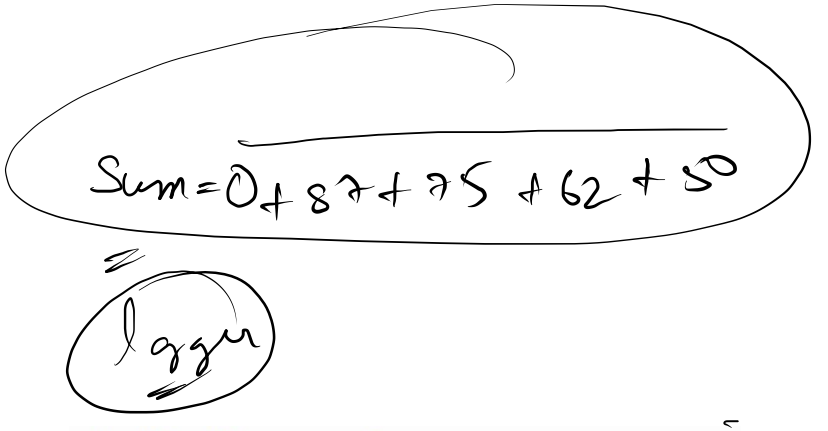
Replace with Sum of Larger



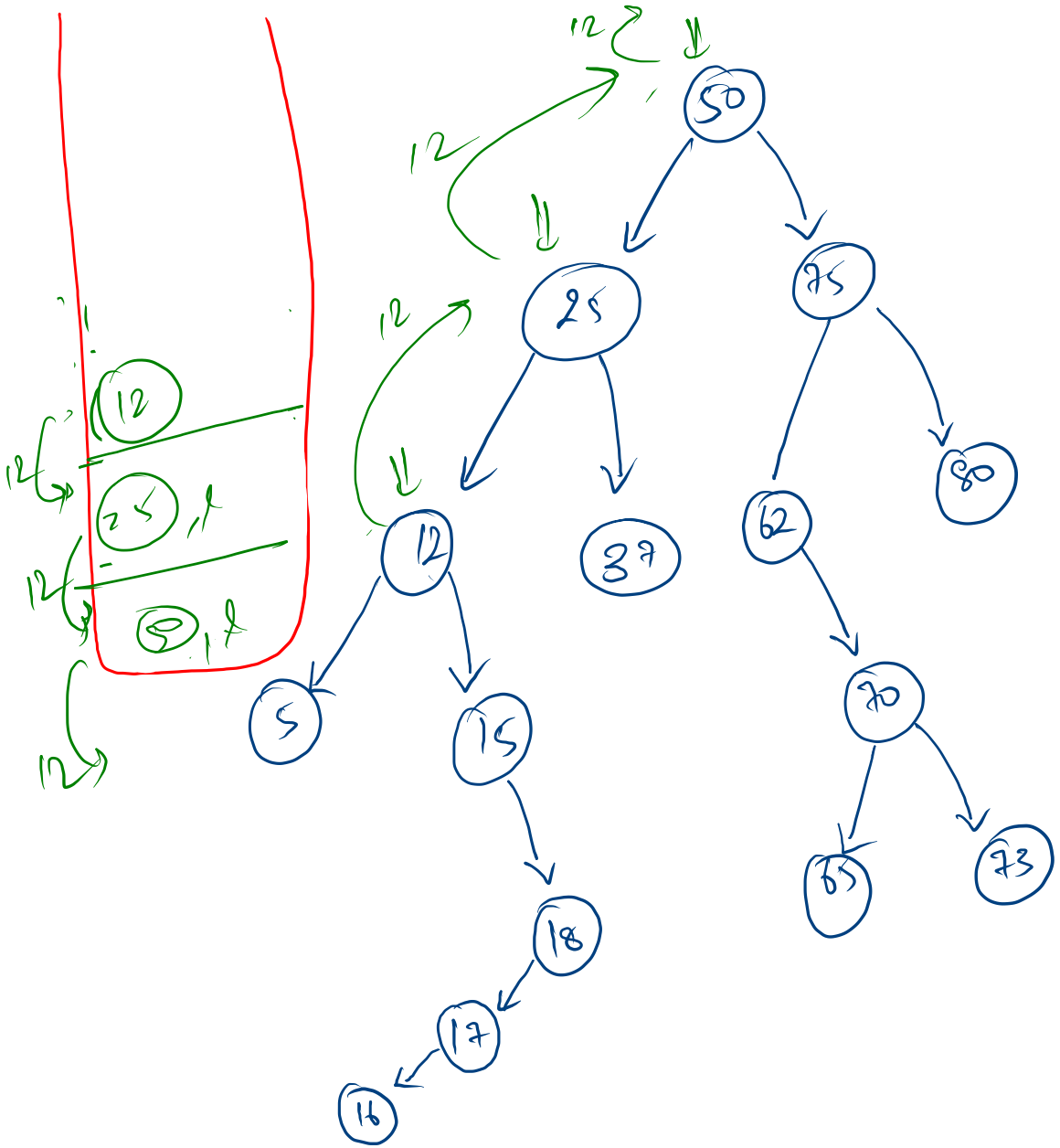









```java
static int sum = 0;
public static void rwsol(Node node){
  if(node == null){
      return;
  }
  rwsol(node.right);

  int tmp = node.data;
  node.data = sum;
  sum += tmp;

  rwsol(node.left);
}
```

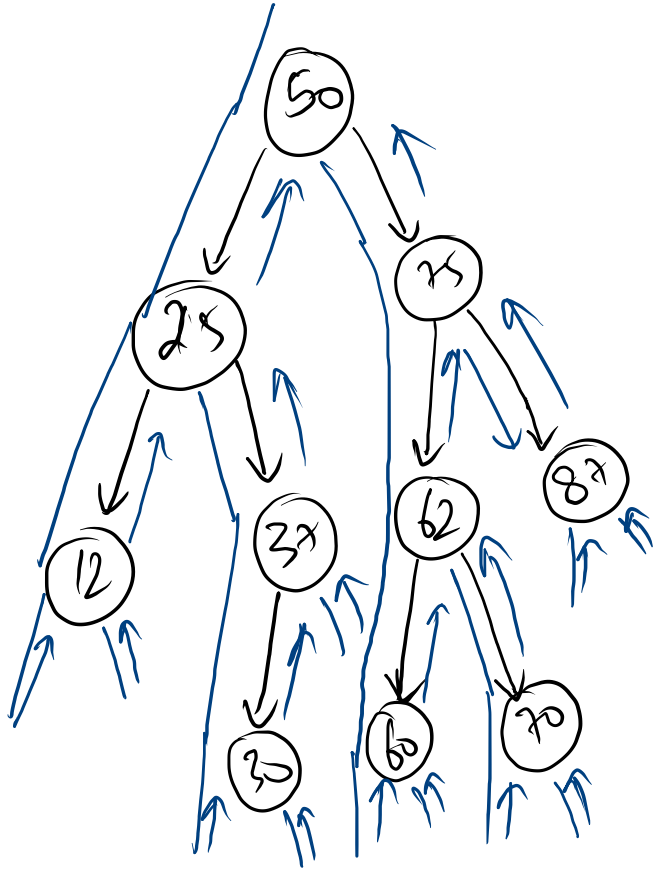


$d_1 = 5$

$d_2 = 15$

(A)

```java
public static int lca(Node node, int d1, int d2) {
  if(d1 < node.data && d2 < node.data){
      return lca(node.left,d1,d2);
  }else if(d1 > node.data && d2 > node.data){
      return lca(node.right,d1,d2);
  }else{
      return node.data;
  }
}
```



	val	cval	<	Find
✗	12	88	T	F
✓	25	75	T	T
✓	30	70	T	T
✗	37	63		F
✗	50	50	F	T
✗	60	40	F	F
✗	62	38	F	F
✗	70	30	F	T
✗	75	25	F	T
✗	87	13	F	F

data ⇒ 100

10

```
public static void tsp(Node node,Node root,int total){
    if(node == null){
        return;
    }

    tsp(node.left,root,total);

    int val = node.data;
    int cval = total - node.data;
    if(val < cval && find(root,cval)){
        System.out.println(val+" "+cval);
    }

    tsp(node.right,root,total);
}
```

✓ 25 75

✓ 30 70