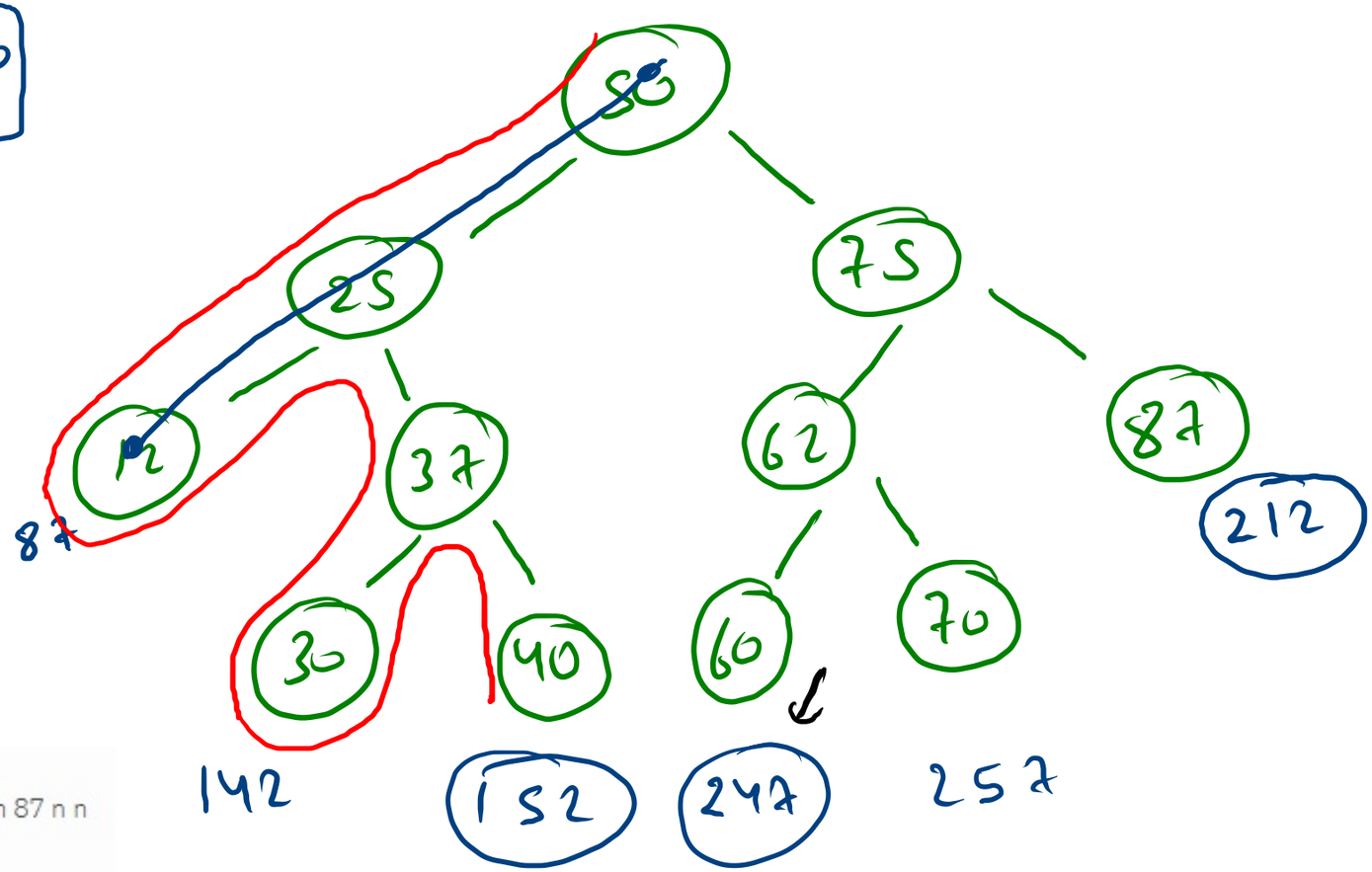
150 ≤ sum ≤ 250



print

50 25 37 40

50 75 62 60

50 75 87

50
75
125
87
212

no to root

23
50 25 12 n n 37 30 n n 40 n n 75 62 60 n n 70 n n 87 n n
150
250

Sample Output

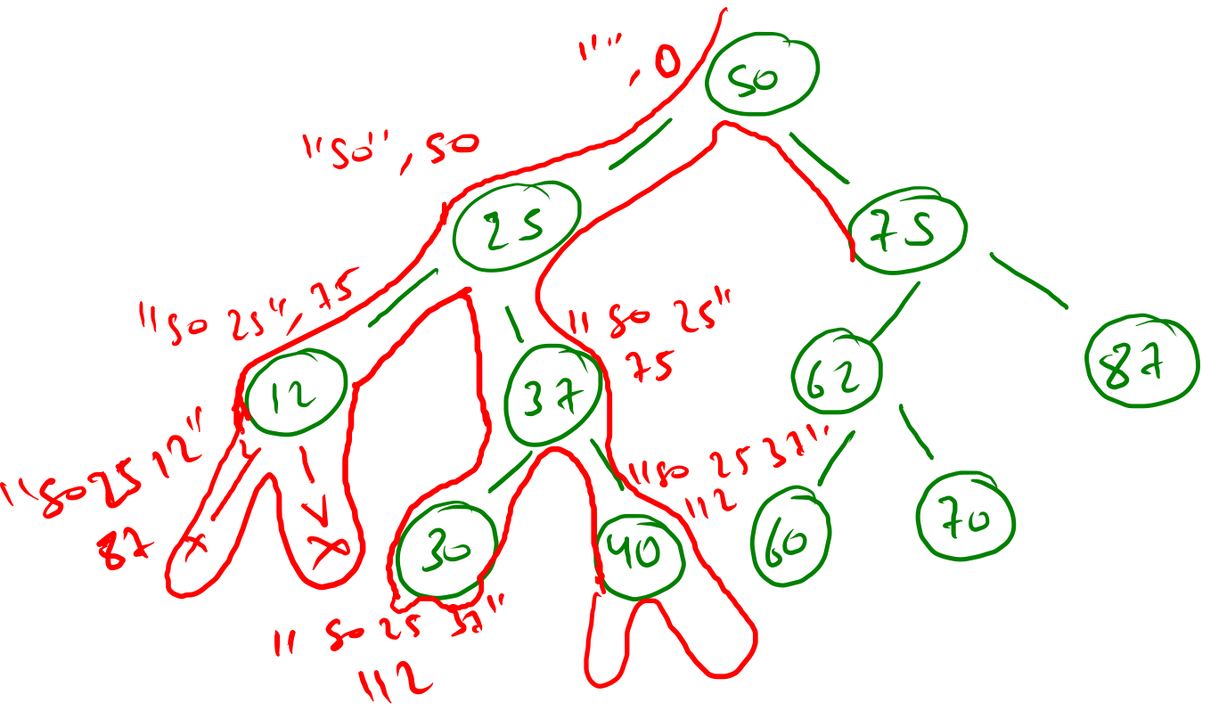
50 25 37 40
50 75 62 60
50 75 87

```java
public static void pathToLeafFromRoot(Node node, String path, int sum, int lo, int hi){
    if(node == null)return;

    if(node.left == null && node.right == null){
        if(lo <= sum+node.data && sum+node.data <= hi){
            System.out.println(path+node.data+" ");
        }
    }

    pathToLeafFromRoot(node.left, path+node.data+" ", sum + node.data, lo, hi);
    pathToLeafFromRoot(node.right, path+node.data+" ", sum + node.data, lo, hi);
}
```

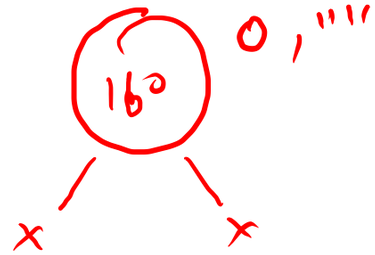
lo = 150

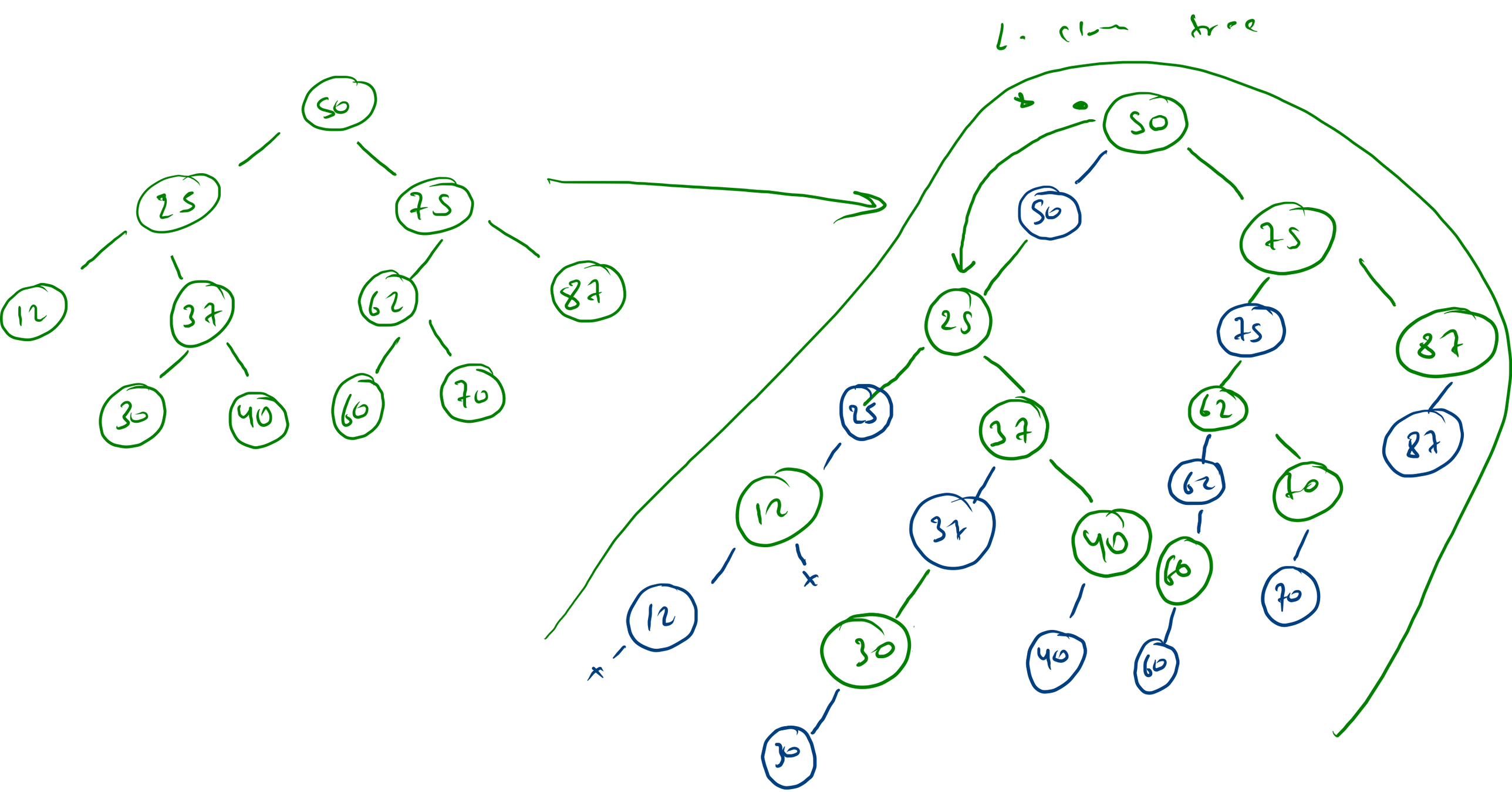
hi = 250

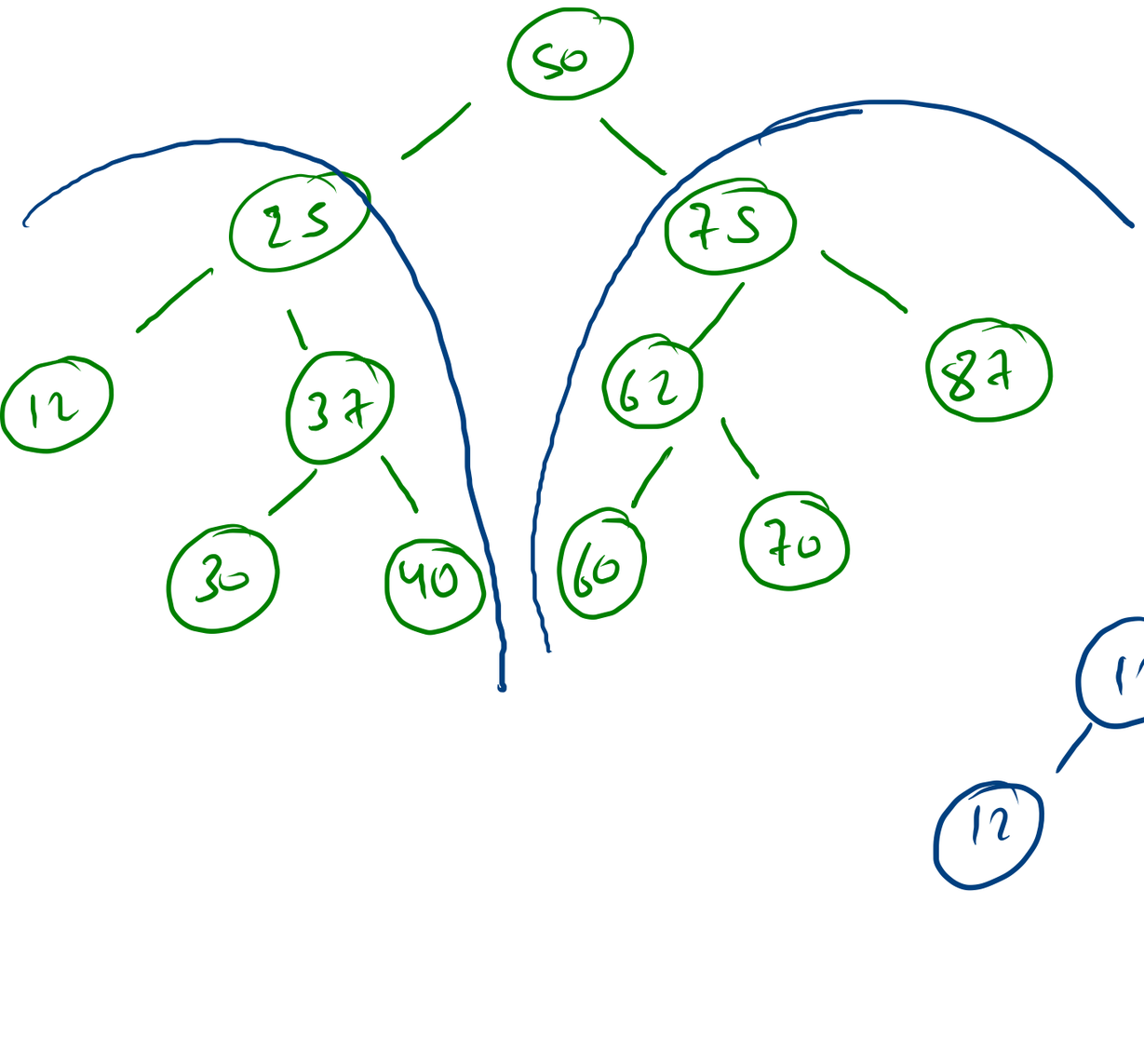
80 25 37 40

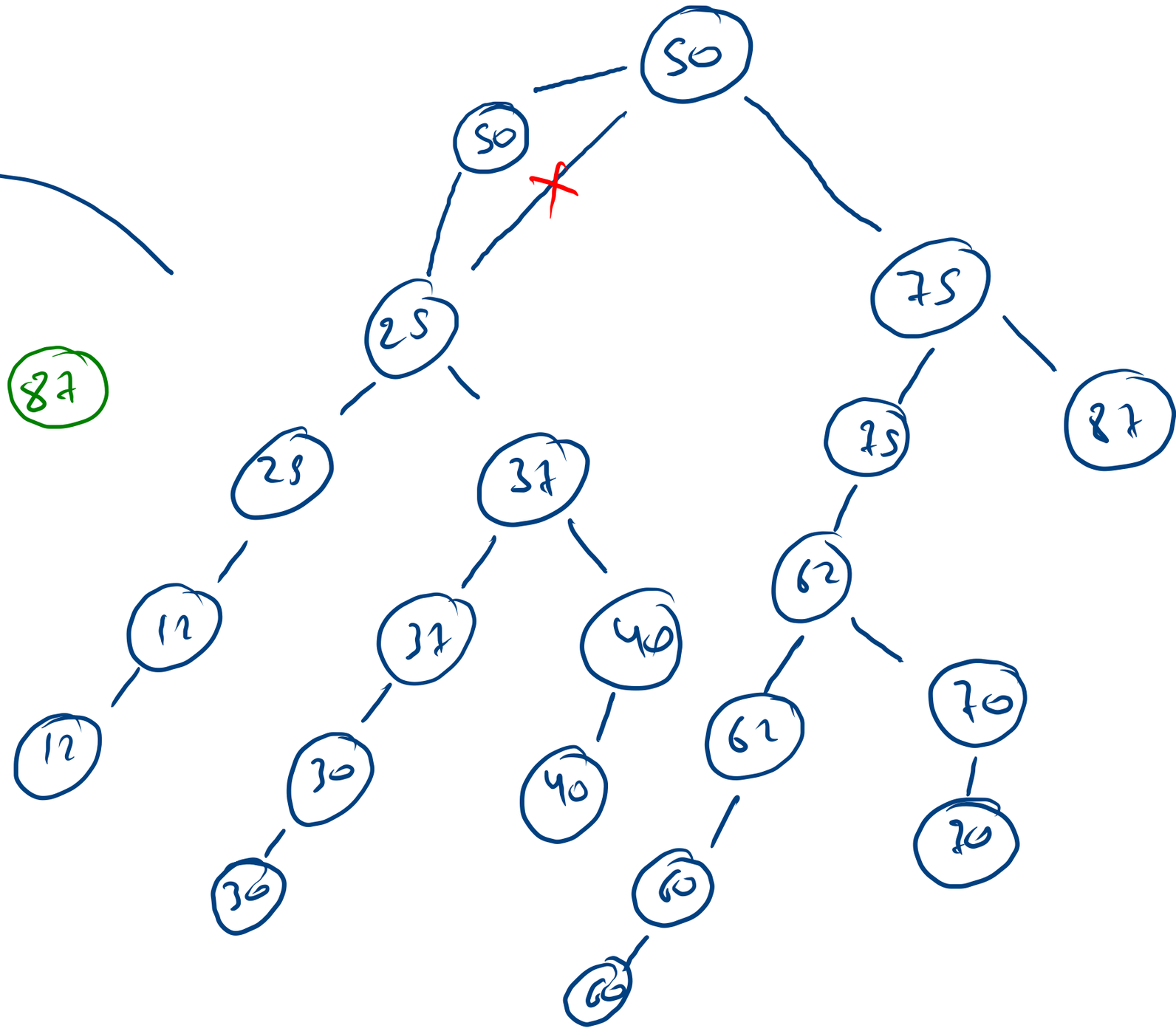


180 ≤ 0 + 160 ≤ 250

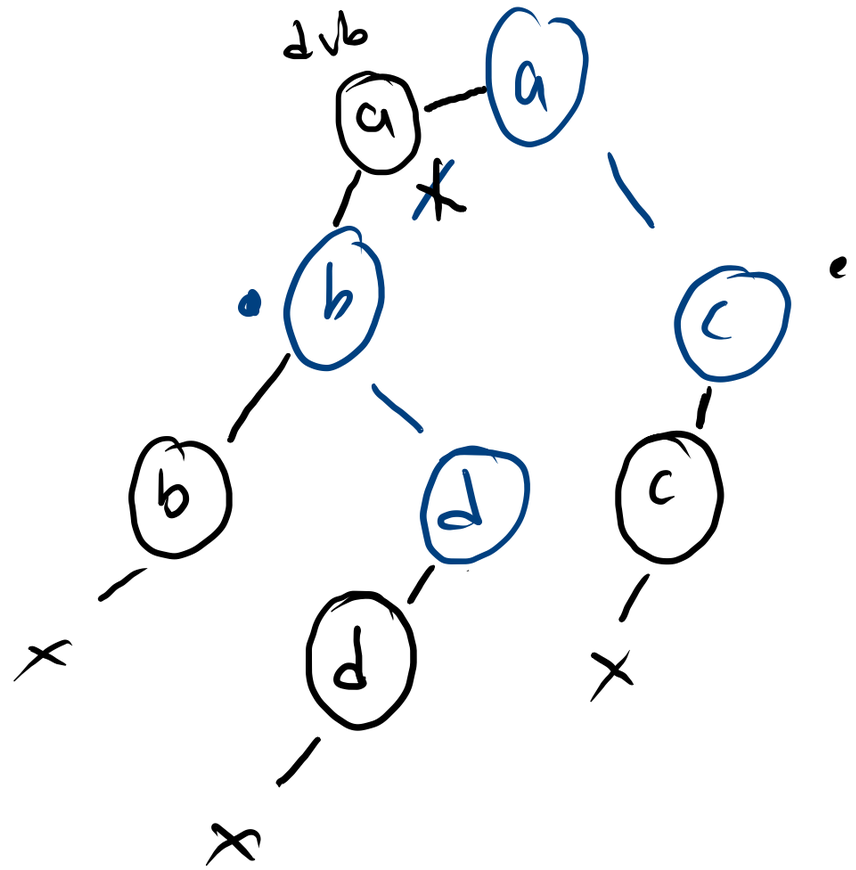


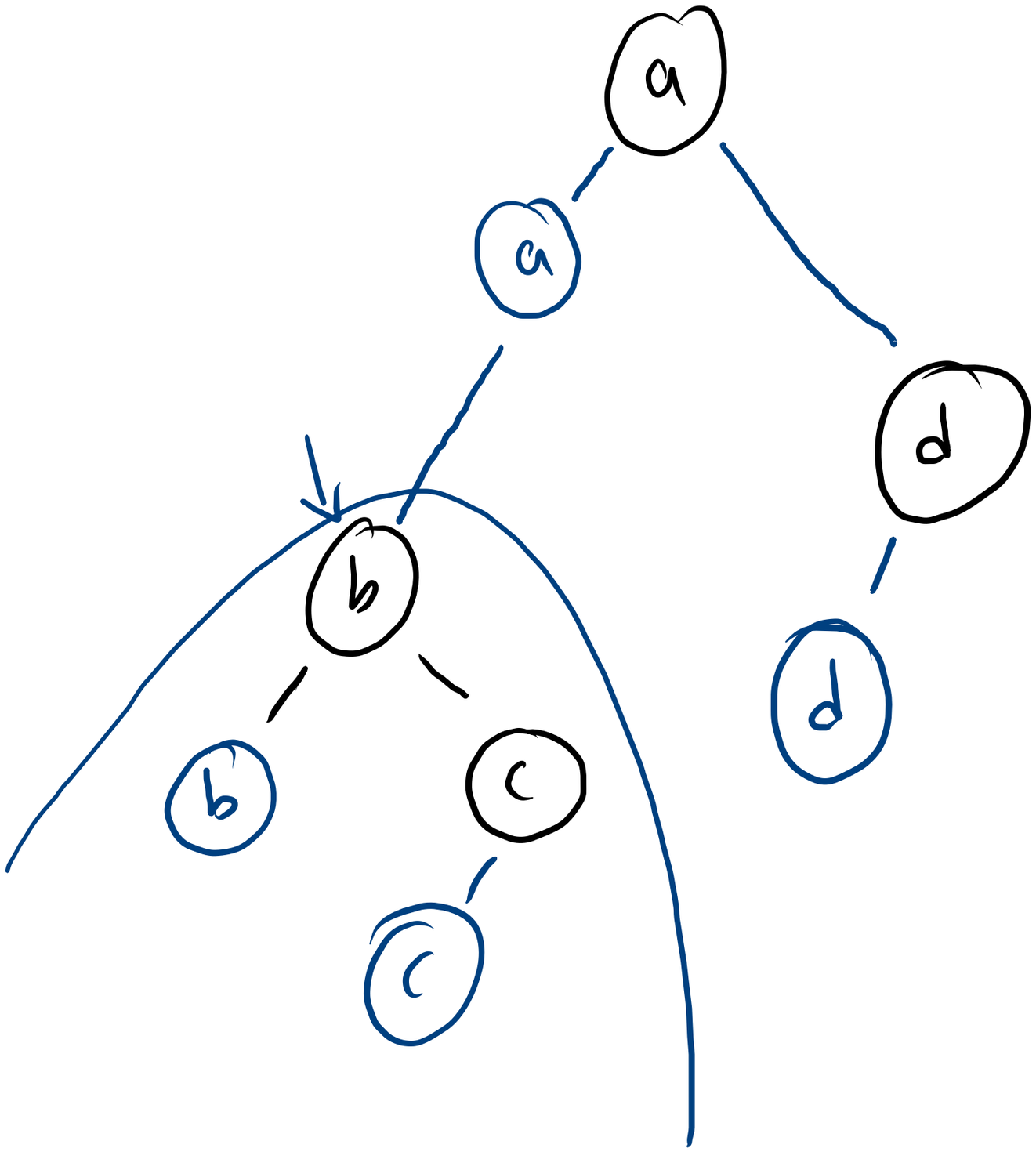

50
25
75
12
37
62
87
30
40
60
70


50
25
75
12
37
62
87
30
40
60
70

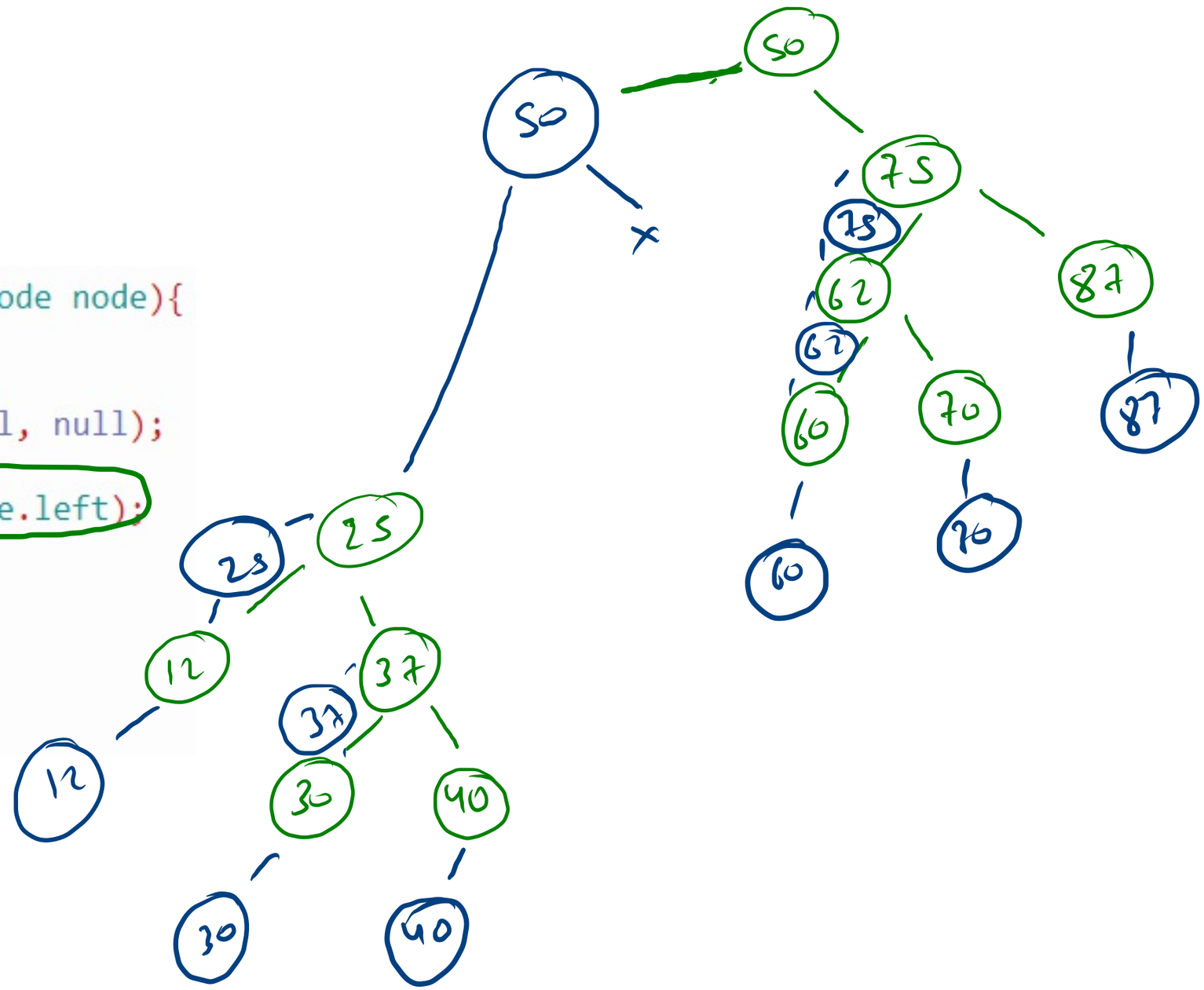

50
50
25
75
23
37
75
87
12
37
40
62
70
12
30
40
62
70
36
60
66

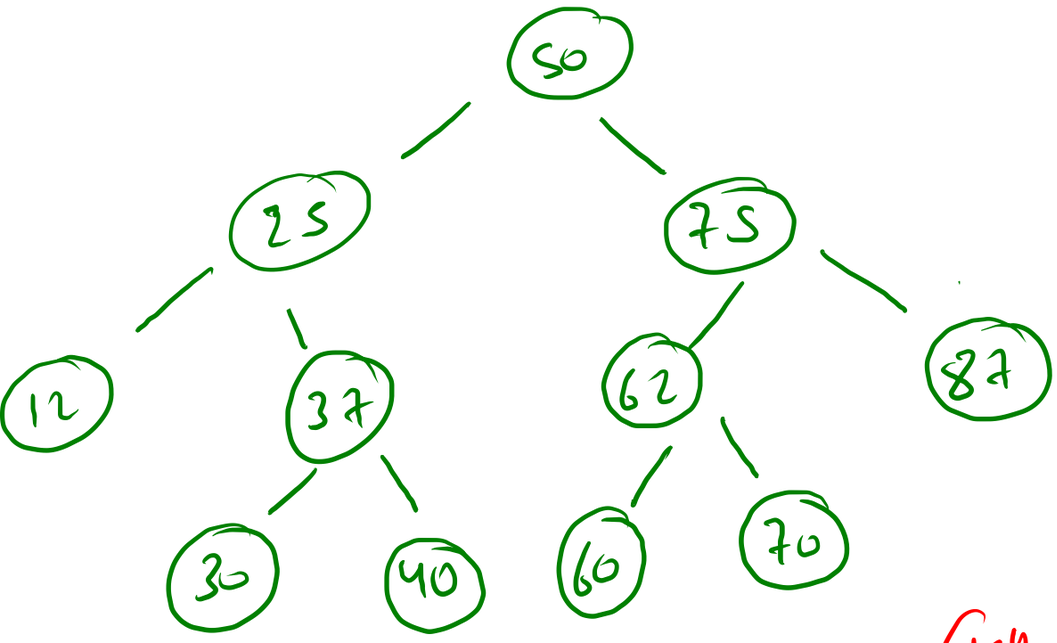
```java
public static Node createLeftCloneTree(Node node){
    if(node == null)return null;

    createLeftCloneTree(node.left);
    createLeftCloneTree(node.right);

    Node dub = new Node(node.data, null, null);
    dub.left = node.left;
    node.left = dub;
    return node;
}
```



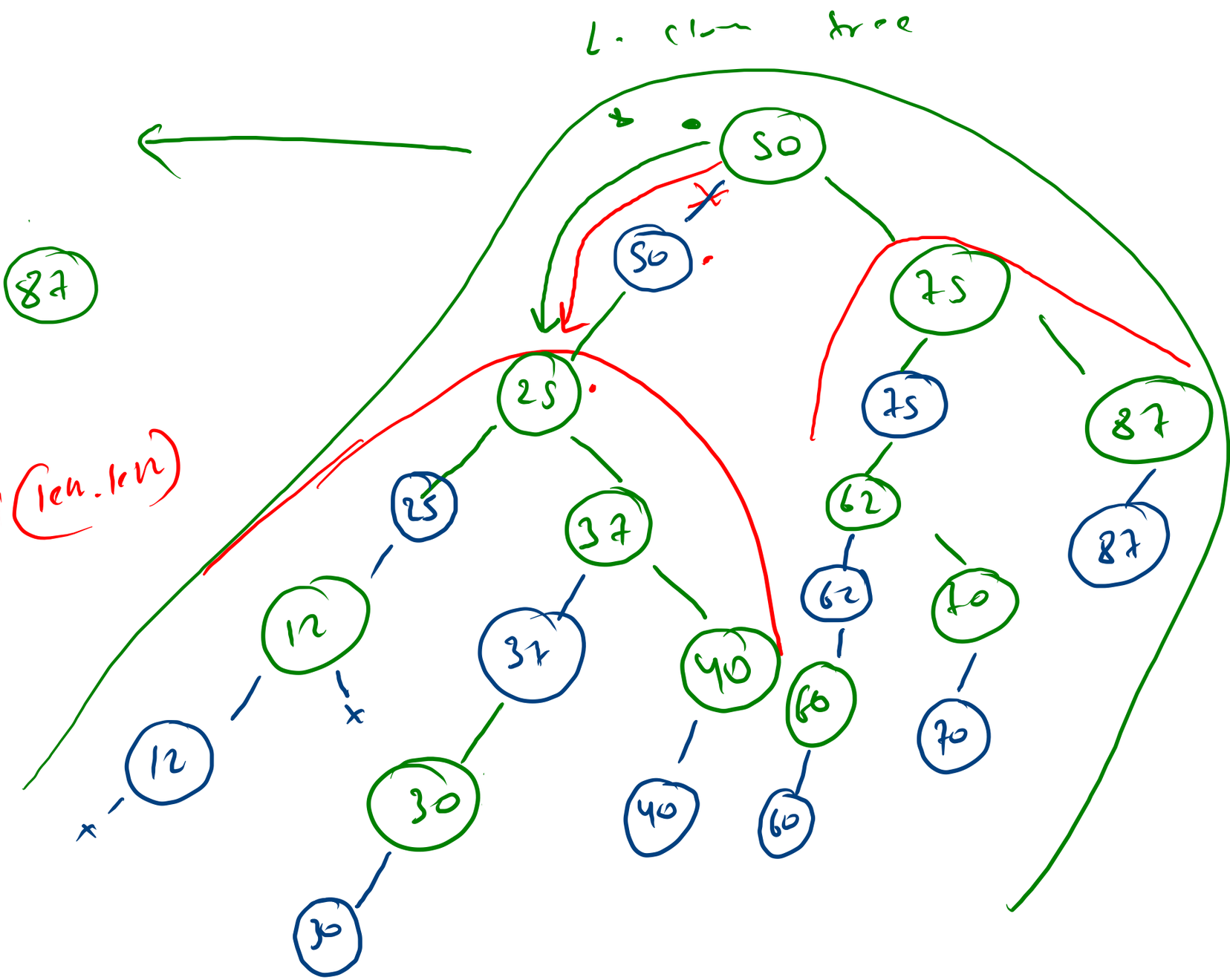

a
a
d
b
b
c
c
d

```java
public static Node createLeftCloneTree(Node node){
    if(node == null)return null;

    Node dub = new Node(node.data, null, null);

    dub.left = createLeftCloneTree(node.left);
    createLeftCloneTree(node.right);

    node.left = dub;
    return node;
}
```

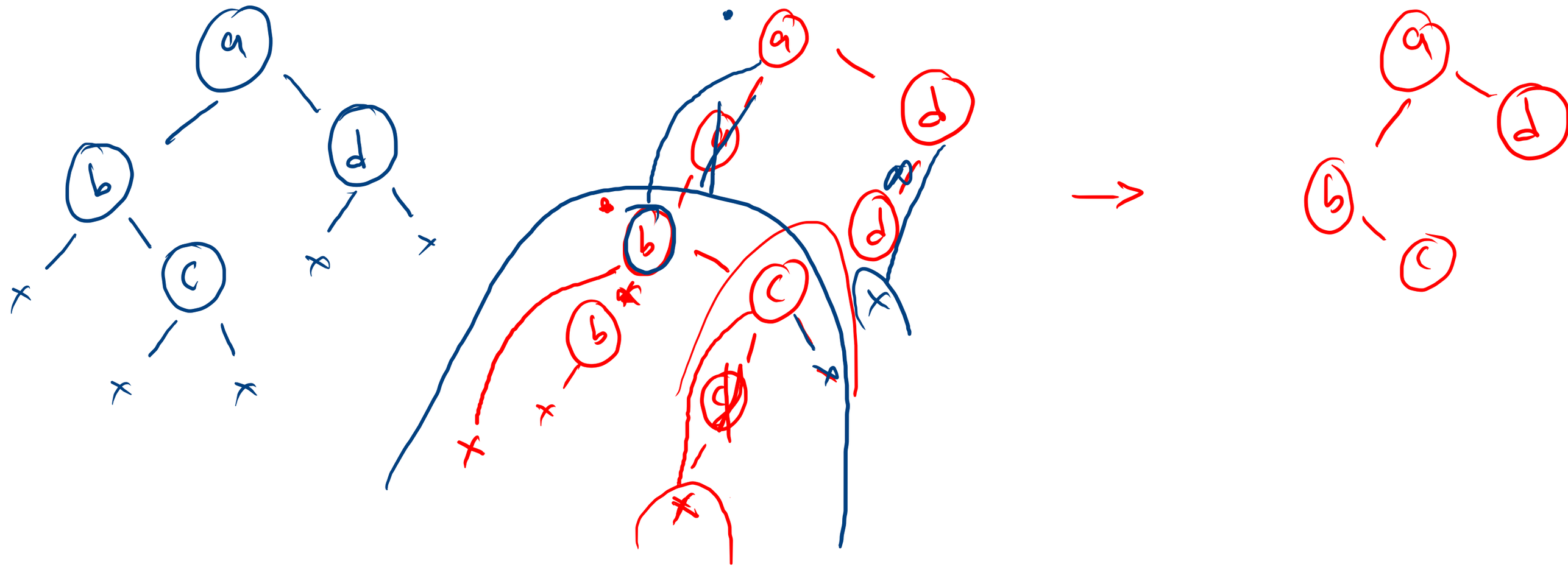


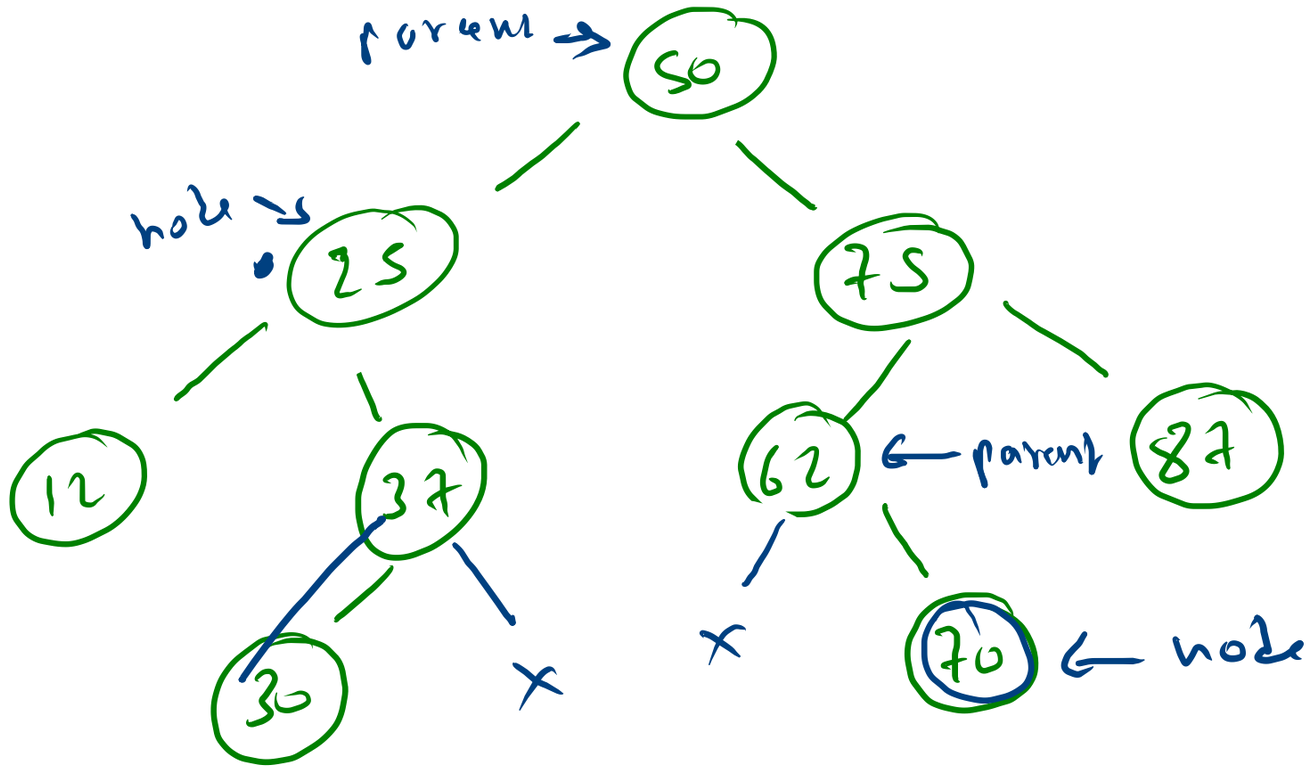


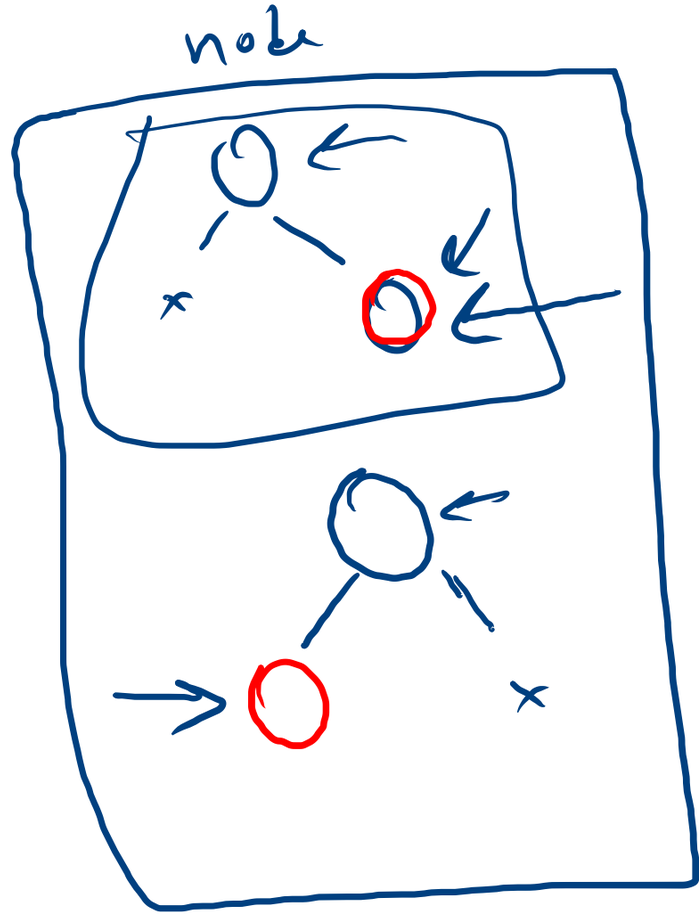
node.left = cal(len.len)



```java
public static Node transBackFromLeftClonedTree(Node node){
    if(node == null)return null;

    node.left = transBackFromLeftClonedTree(node.left.left);
    transBackFromLeftClonedTree(node.right);

    return node;
}
```

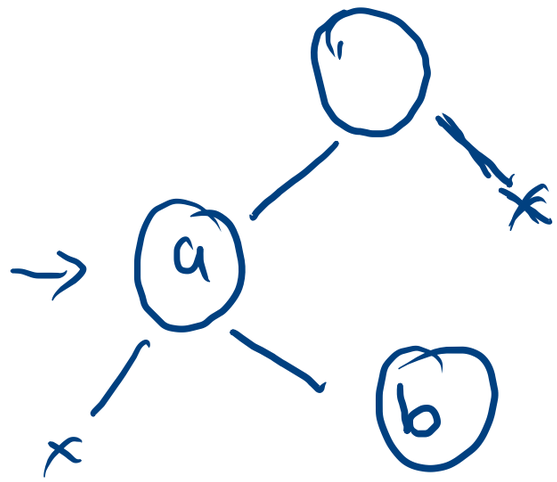


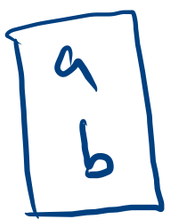

parent
50
node
25
75
12
37
62
parent
87
30
x
x
70
node


node
x
x

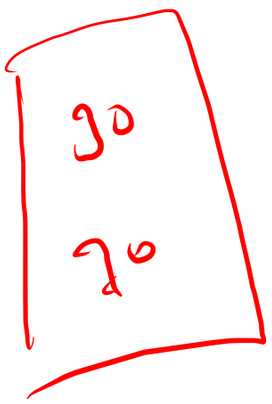
30

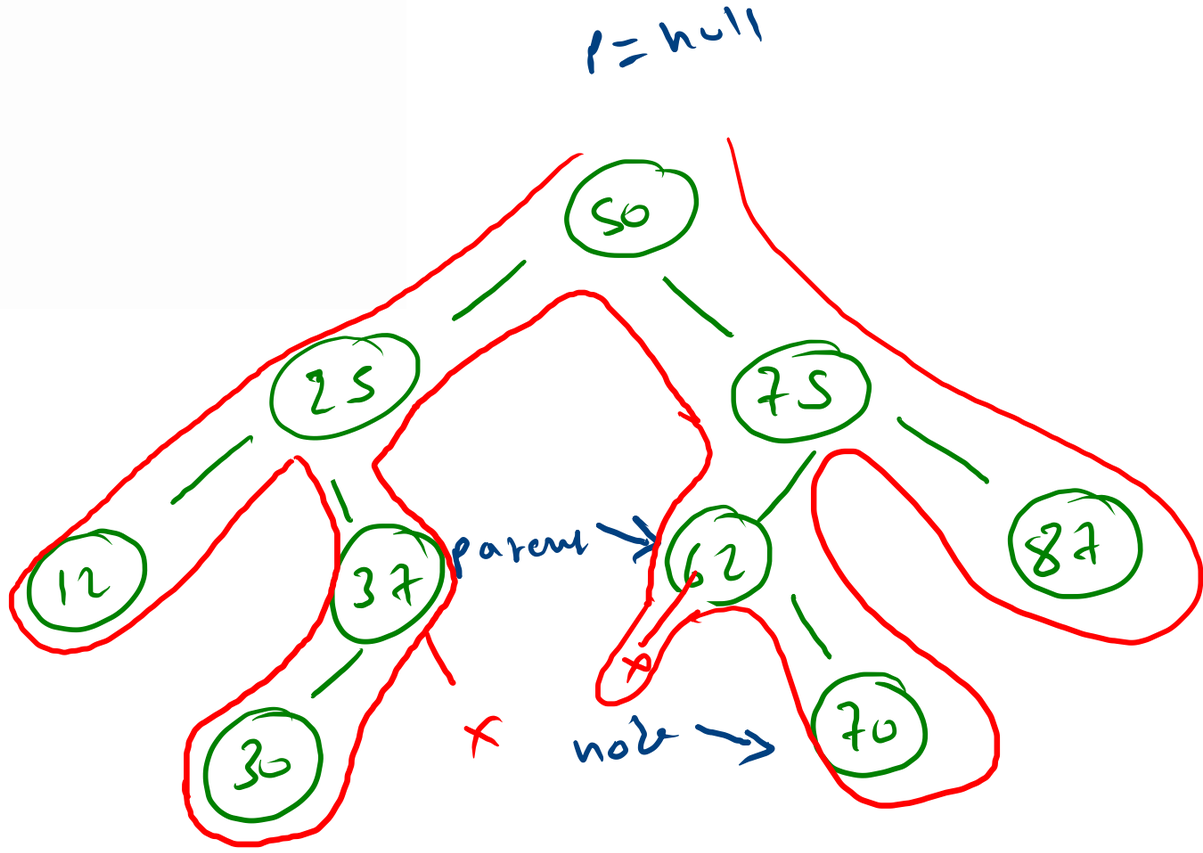
70

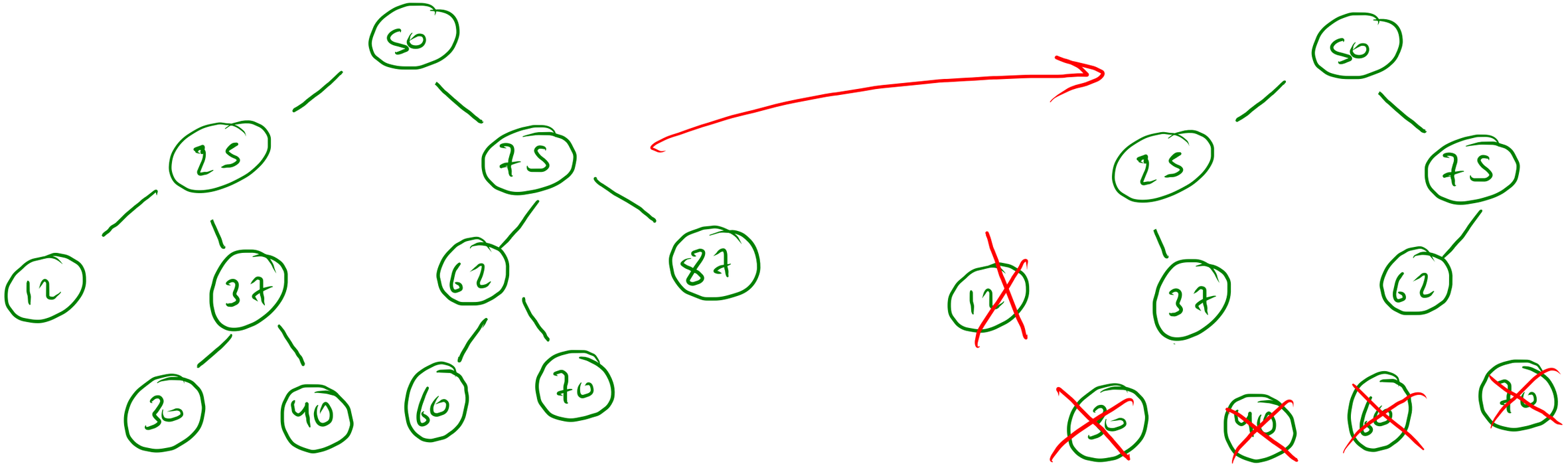

a
x
a
b

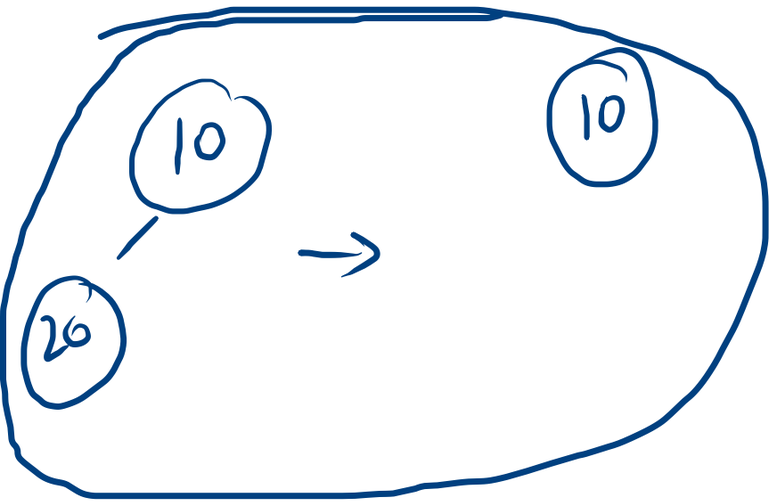

a
b

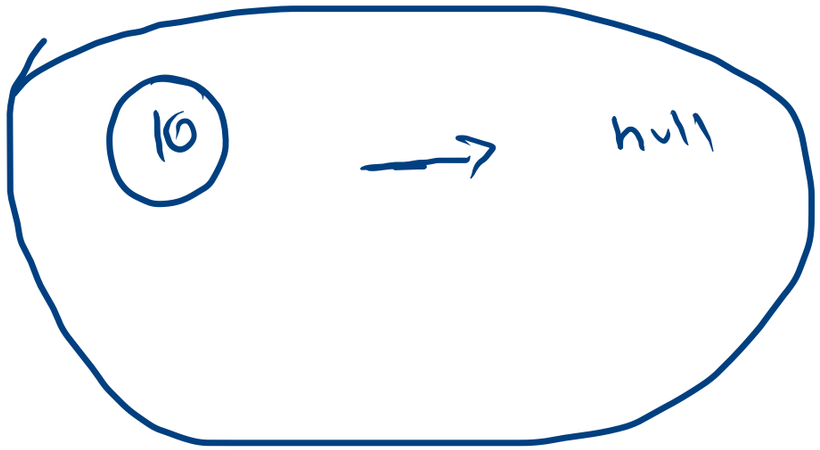
```java
public static void printSingleChildNodes(Node node, Node parent){
    if(node == null)return;

    if(parent != null && (parent.left == null || parent.right == null)){
        System.out.println(node.data);
    }

    printSingleChildNodes(node.left, node);
    printSingleChildNodes(node.right, node);
}
```



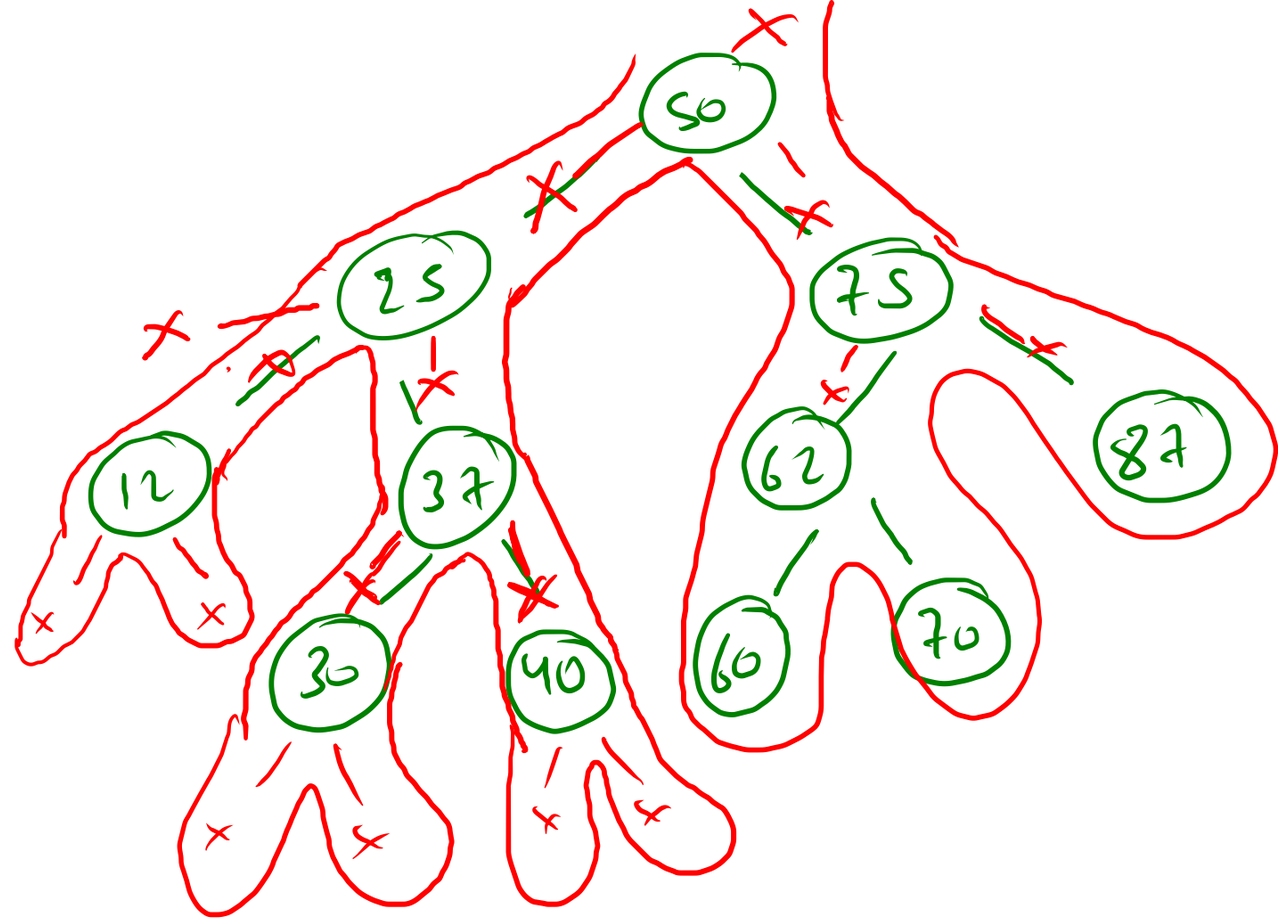



50
25
75
12
37
62
87
30
40
60
70
50
25
75
12
37
62
87
30
40
60
70

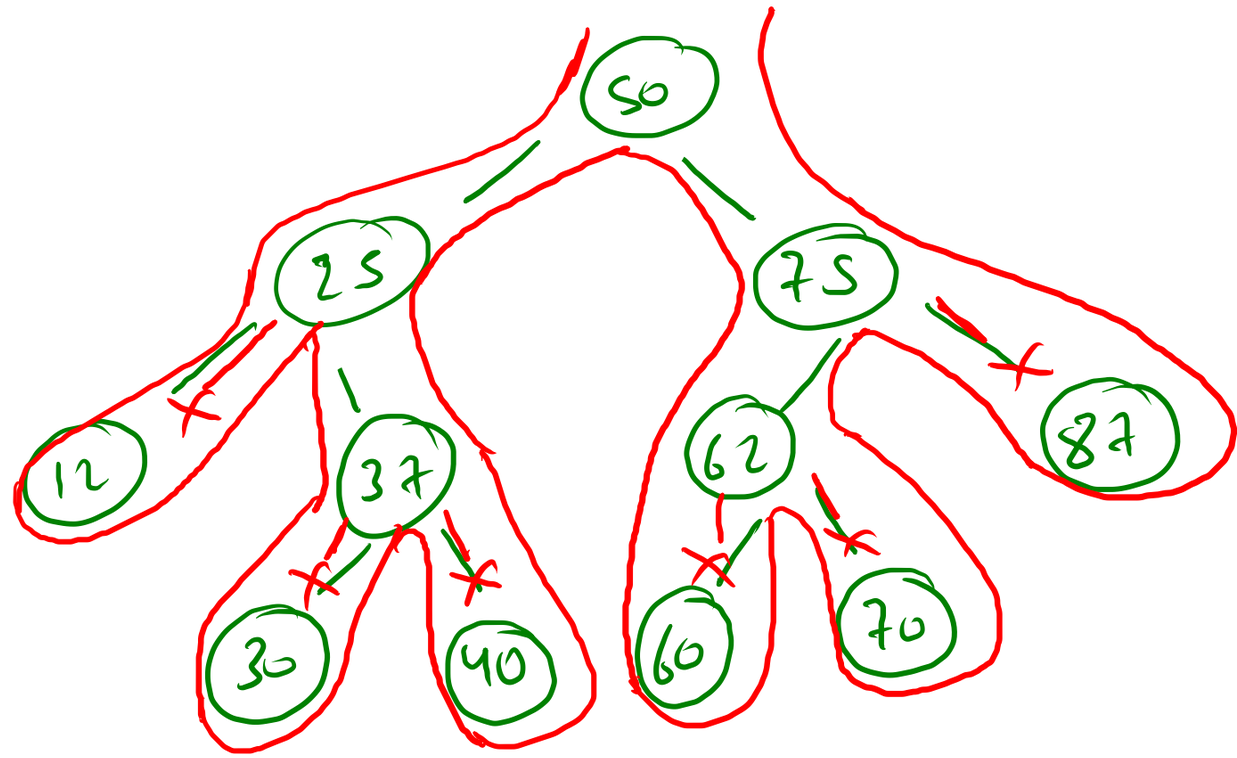

10
10
20

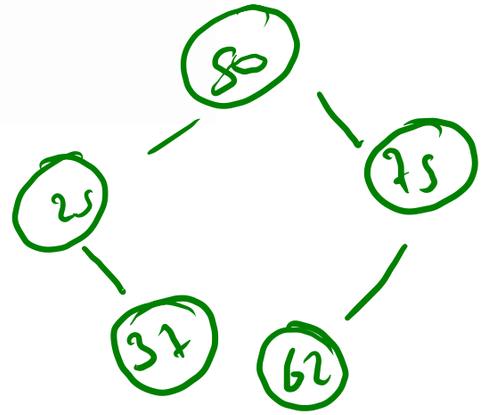

10
null

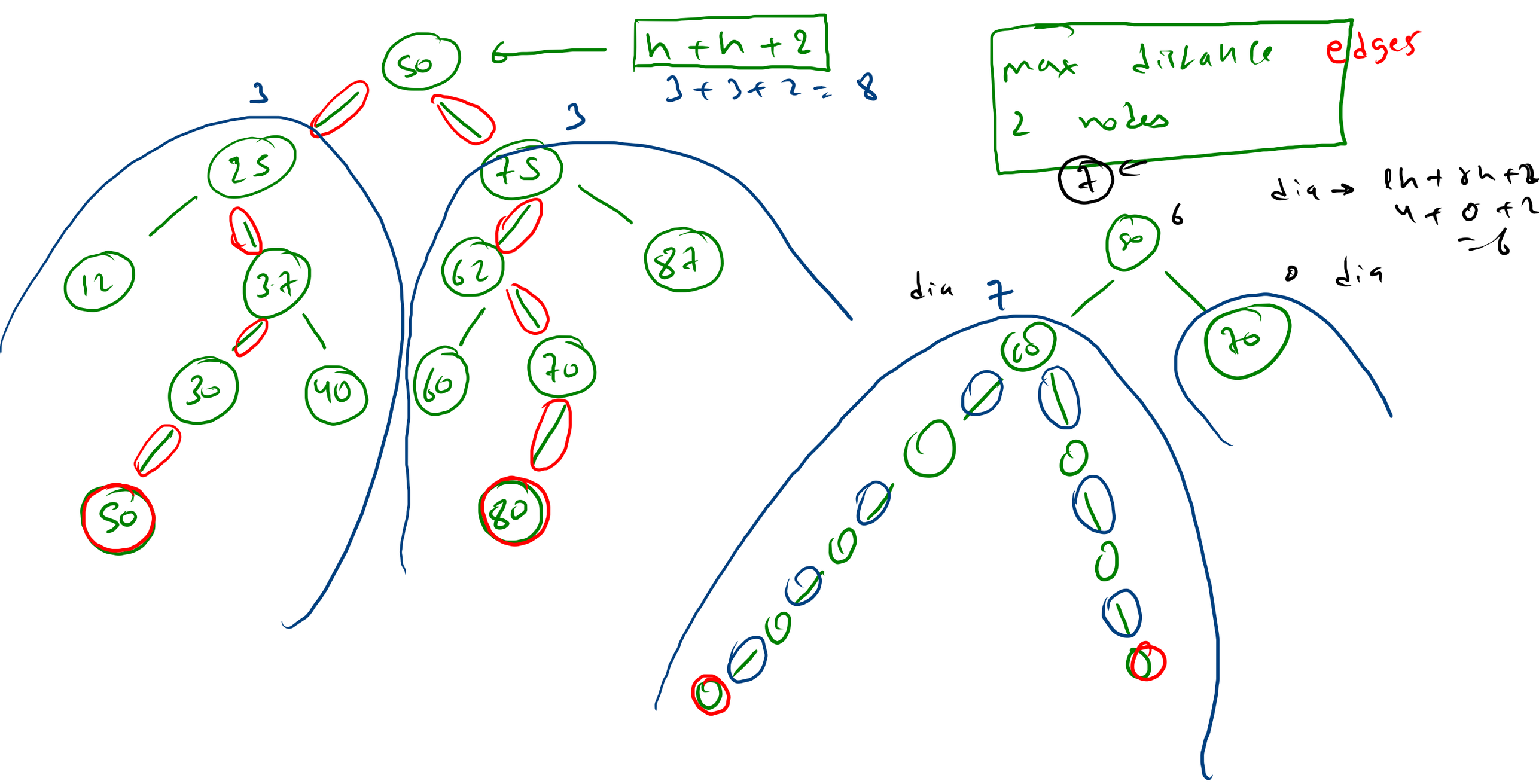
```java
public static Node removeLeaves(Node node){
    if(node == null){
        return null;
    }

    node.left = removeLeaves(node.left);
    node.right = removeLeaves(node.right);

    if(node.left == null && node.right == null){
        return null;
    }

    return node;
}
```



```java
public static Node removeLeaves(Node node){
    if(node == null){
        return null;
    }

    if(node.left == null && node.right == null){
        return null;
    }
    node.left = removeLeaves(node.left);
    node.right = removeLeaves(node.right);

    return node;
}
```






50
h + h + 2
3 + 3 + 2 = 8
3
3
25
75
12
37
62
87
30
40
60
70
50
80
max distance edges
2 nodes
7
dia → 1h + rh + 2
4 + 0 + 2
= 6
6
0 dia
dia 7
90
70
60

```java
public static int diameter1(Node node) {
    if(node == null){
        return 0;
    }

    int ld = diameter1(node.left);
    int rd = diameter1(node.right);

    int lh = height(node.left);
    int rh = height(node.right);

    int dia = lh+rh+2;

    return Math.max(dia, Math.max(ld, rd));
}
```

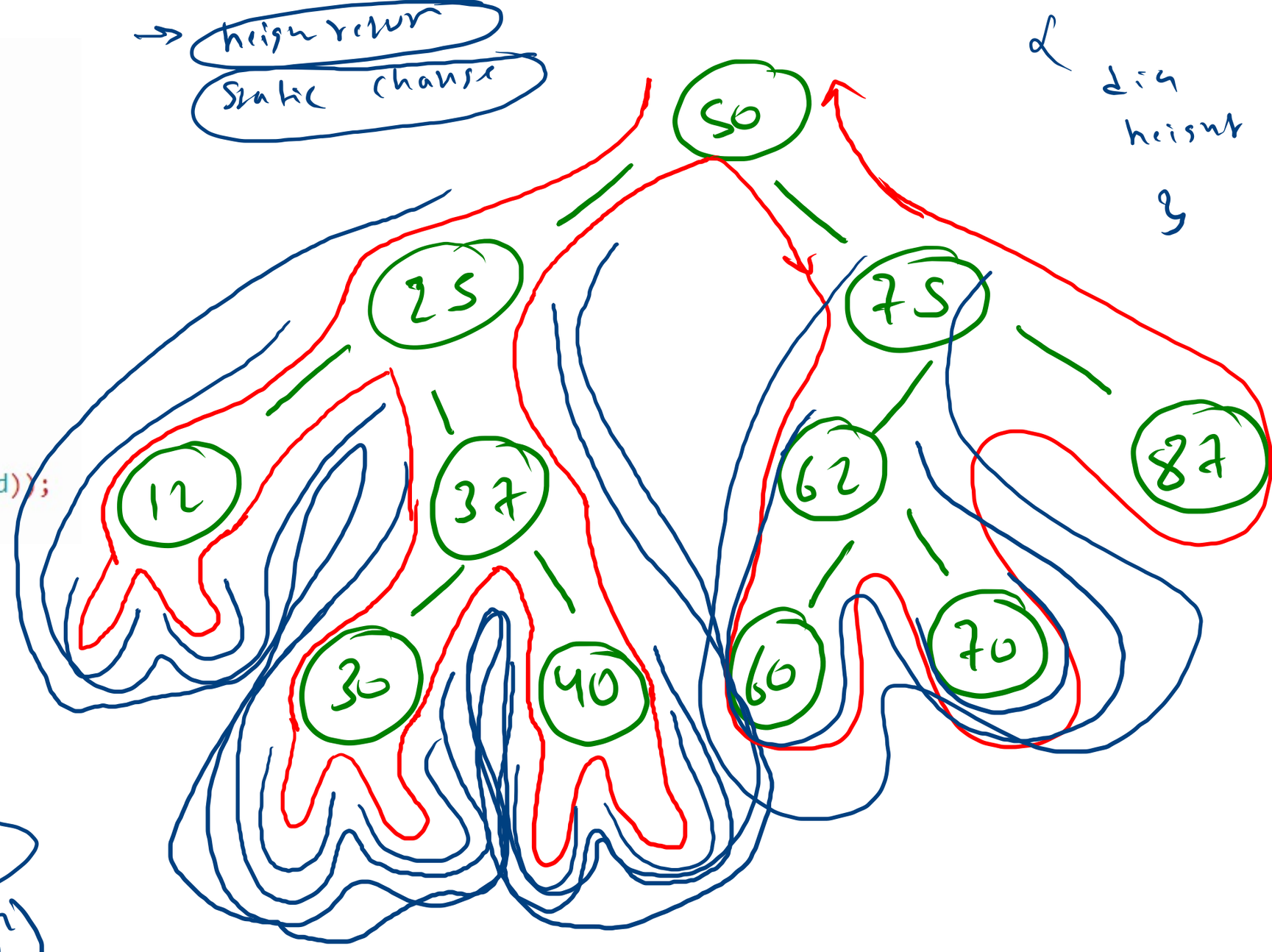
height
diameter1

Merge sort
Quick sort

N
dia → O(N)
ht → O(N) × N
O(N²)

→ height return
Static change

d dia
height
g



```java
public static Pair2 diameter(Node node){
    if(node == null){
        Pair2 p = new Pair2();
        p.dia = 0;
        p.height = -1;
        return p;
    }

    Pair2 lp = diameter(node.left);
    Pair2 rp = diameter(node.right);

    int dia = lp.height + rp.height + 2;
    if(lp.dia > dia){
        dia = lp.dia;
    }
    if(rp.dia > dia){
        dia = rp.dia;
    }
    Pair2 p = new Pair2();
    p.dia = dia;
    p.height = Math.max(lp.height, rp.height) + 1;
    return p;
}
```

O(N)