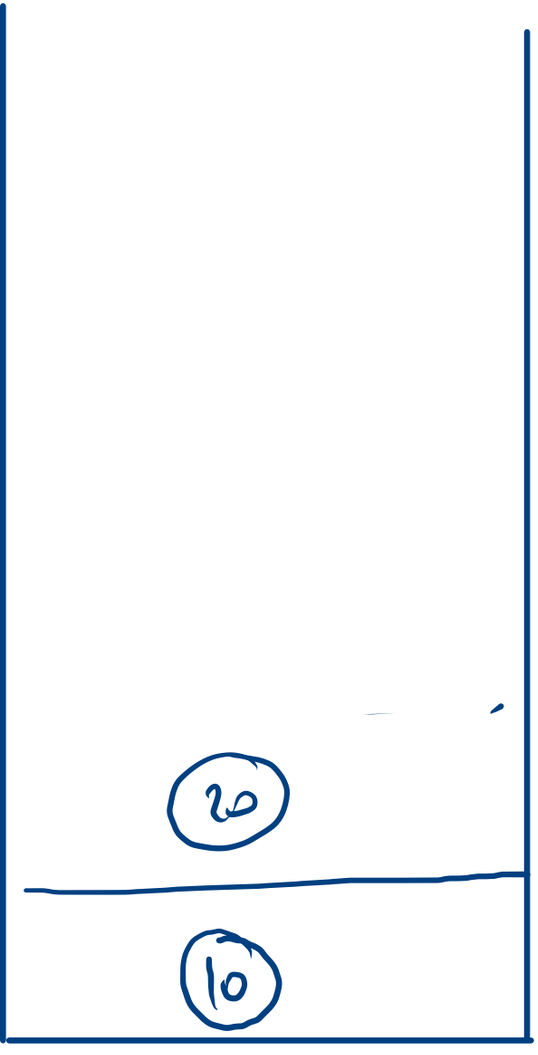
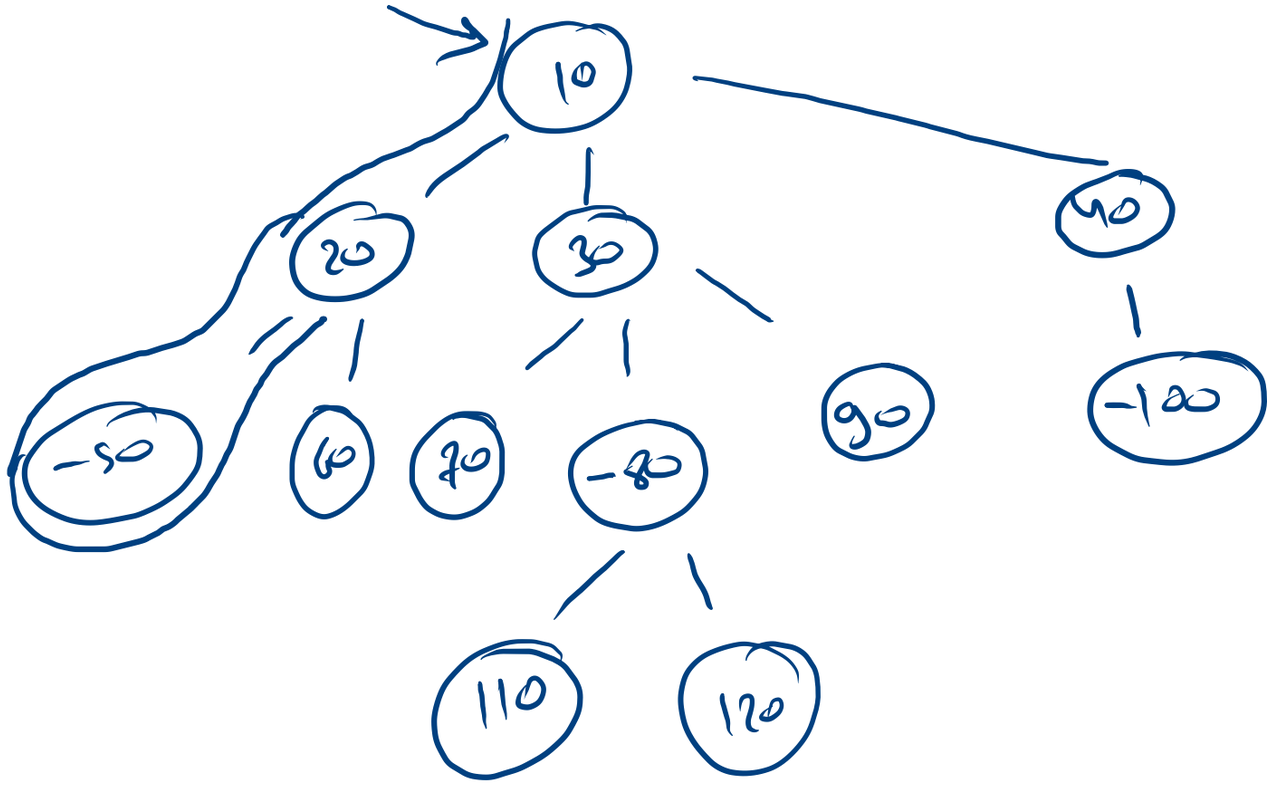
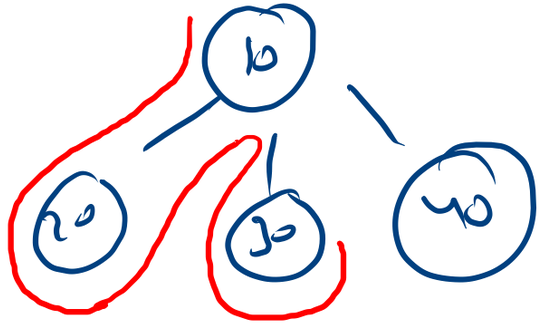
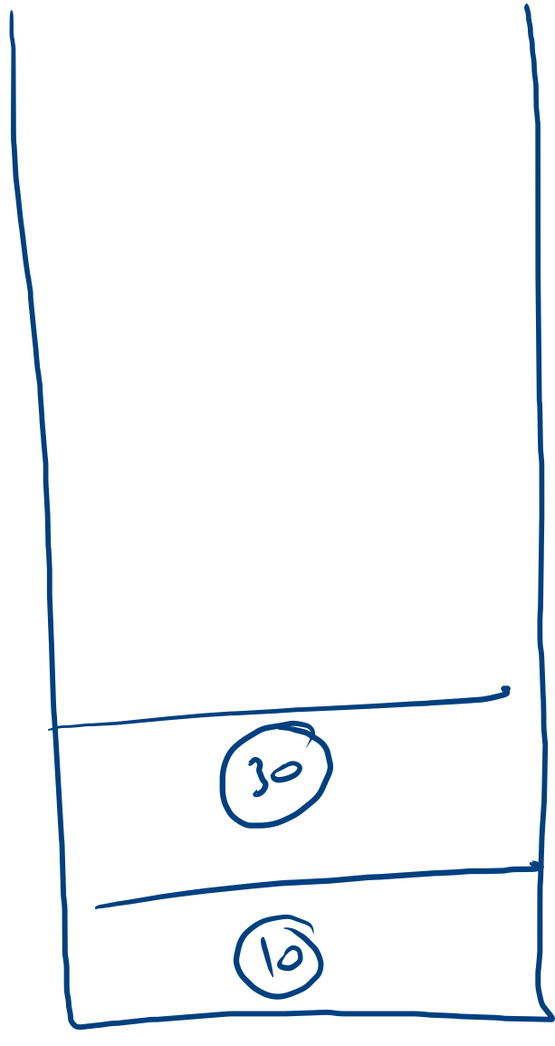

20
10

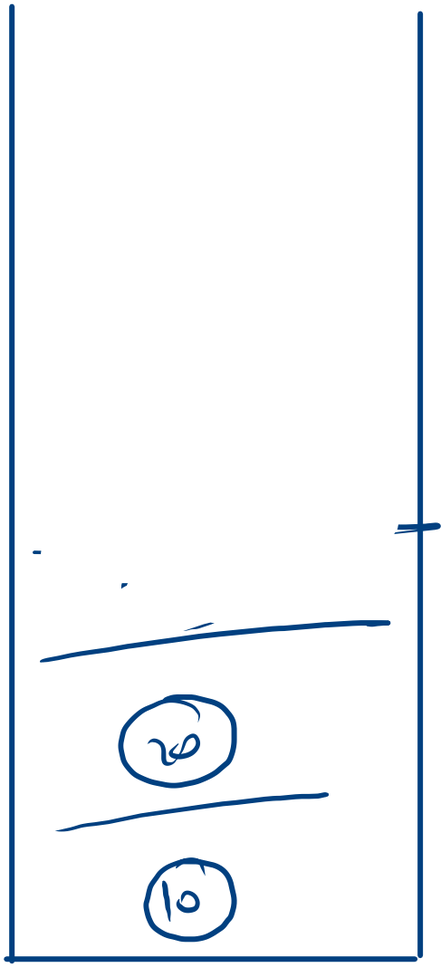

10
20
30
40
-50
60
70
-80
90
-100
110
120

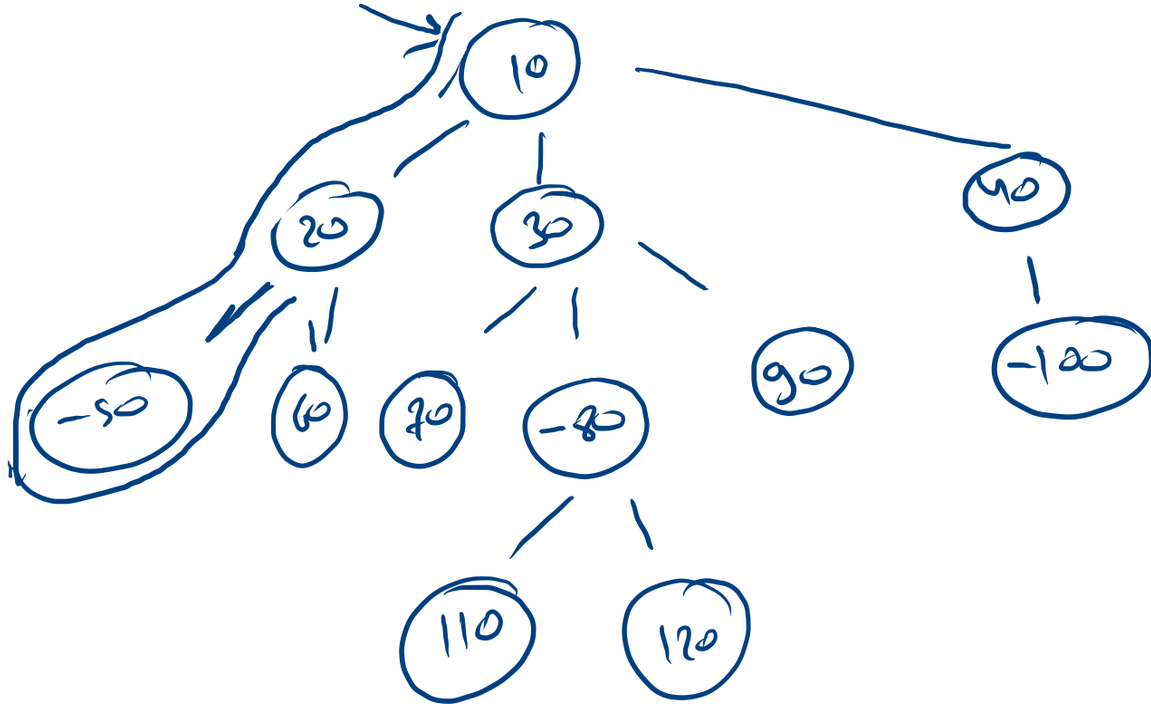





```java
public static void traversals(Node node){
    System.out.println("Node Pre "+node.data);
    for (Node child : node.children){
        System.out.println("Edge Pre "+node.data+"--"+child.data);

        traversals(child);

        System.out.println("Edge Post "+node.data+"--"+child.data);
    }
    System.out.println("Node Post "+node.data);
}
```

Stack<Node>





pair {
  Node
  int state
}

-1 → pre state++

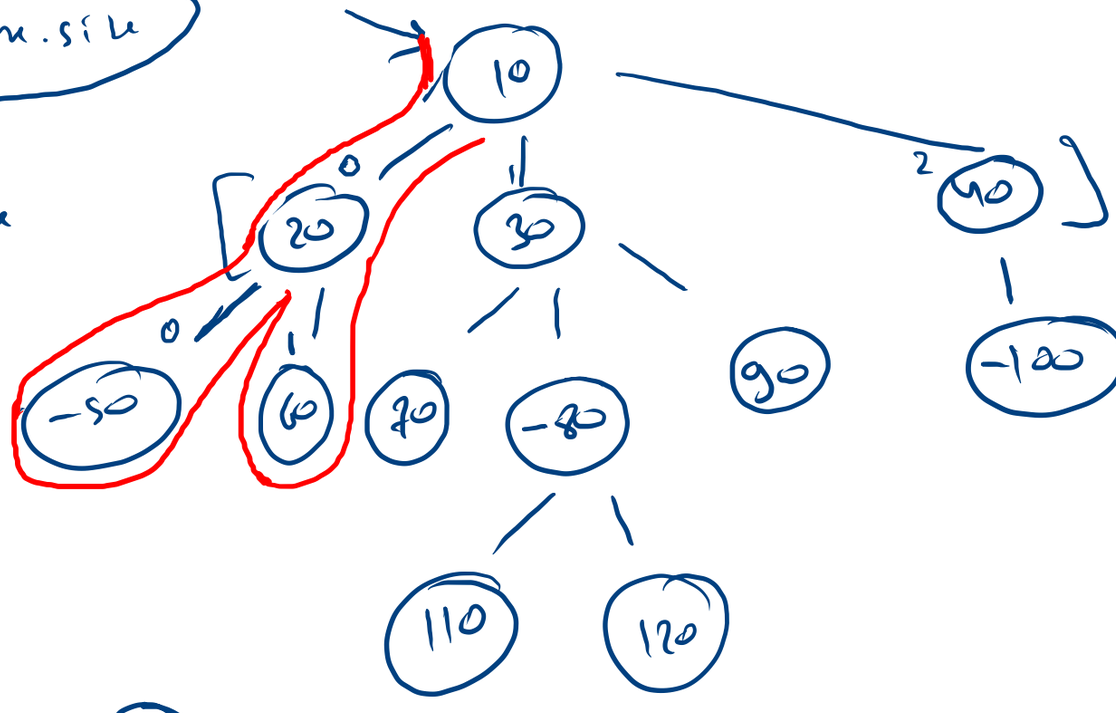
0 ≤ state < children.size

push state++

state == children.size

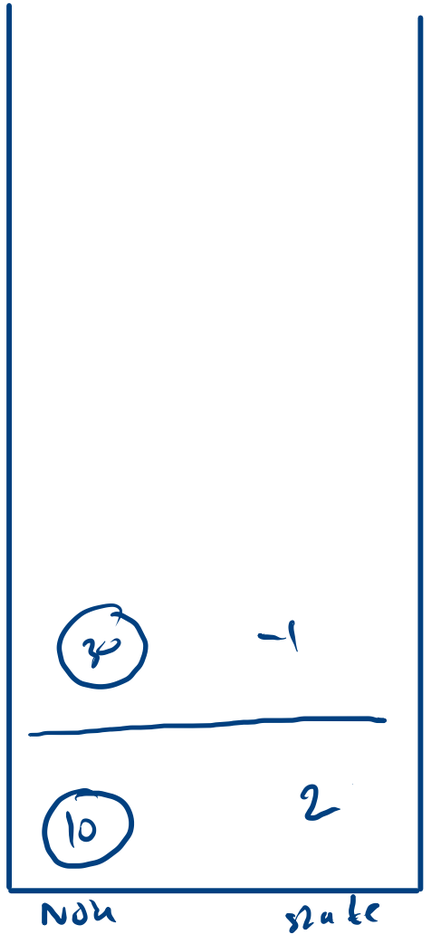
post

pop

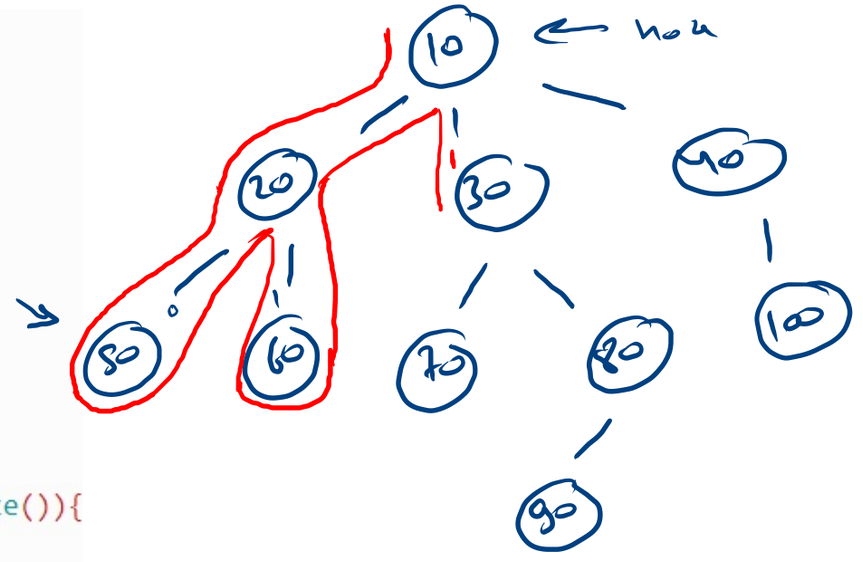


pre 10 20 -50 60

pop 20

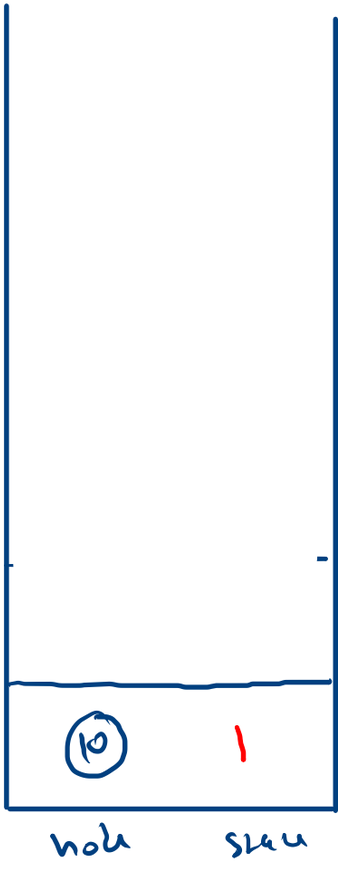


```java
Stack<Pair> st = new Stack<>();
st.push(new Pair(node, -1));

StringBuilder pre = new StringBuilder();
StringBuilder post = new StringBuilder();

while(st.size() > 0){

    Pair p = st.peek();

    if(p.state == -1){
        pre.append(p.node.data+" ");
        p.state++;
    }else if(p.state >= 0 && p.state<p.node.children.size()){
        Node child = p.node.children.get(p.state);
        p.state++;
        st.push(new Pair(child, -1));
    }else{
        post.append(p.node.data+" ");
        st.pop();
    }
}

System.out.println(pre);
System.out.println(post);
```



pre → 10 20 50 60

post → 50 60 20

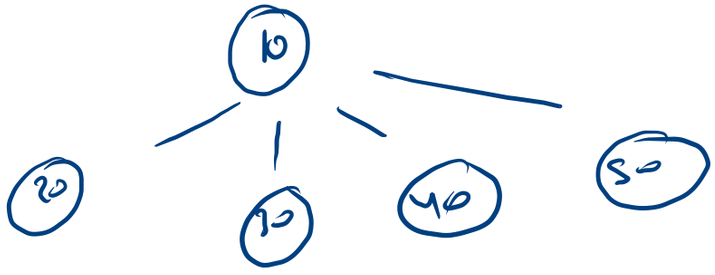


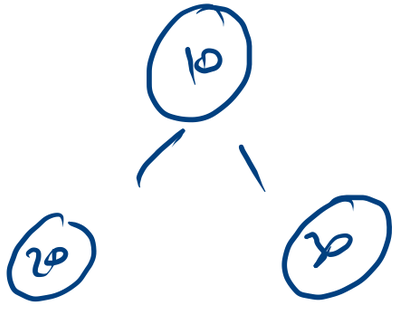
Node {

data

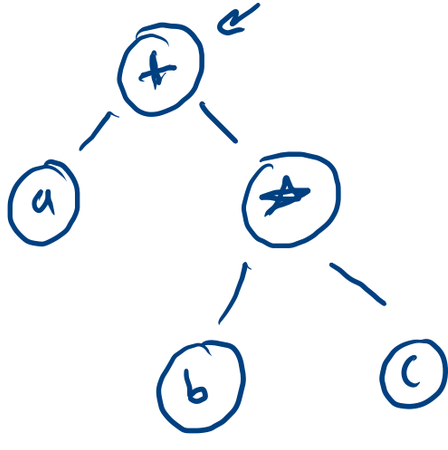
ArrayList<Node> children

}

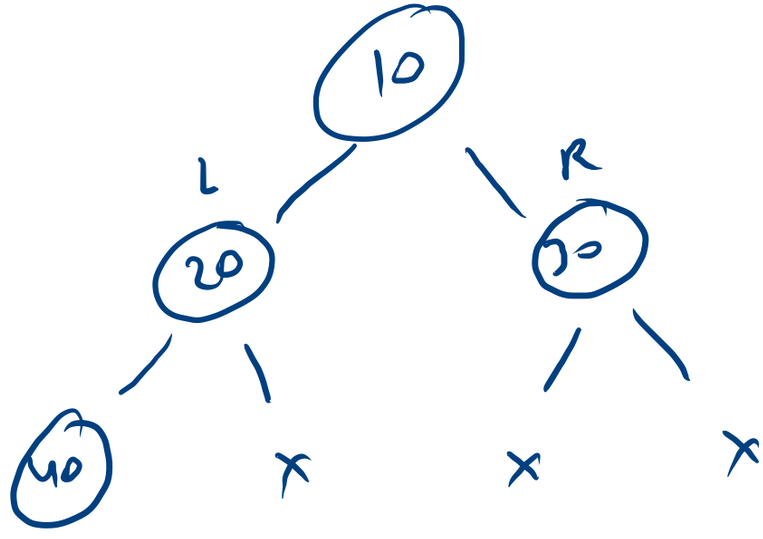




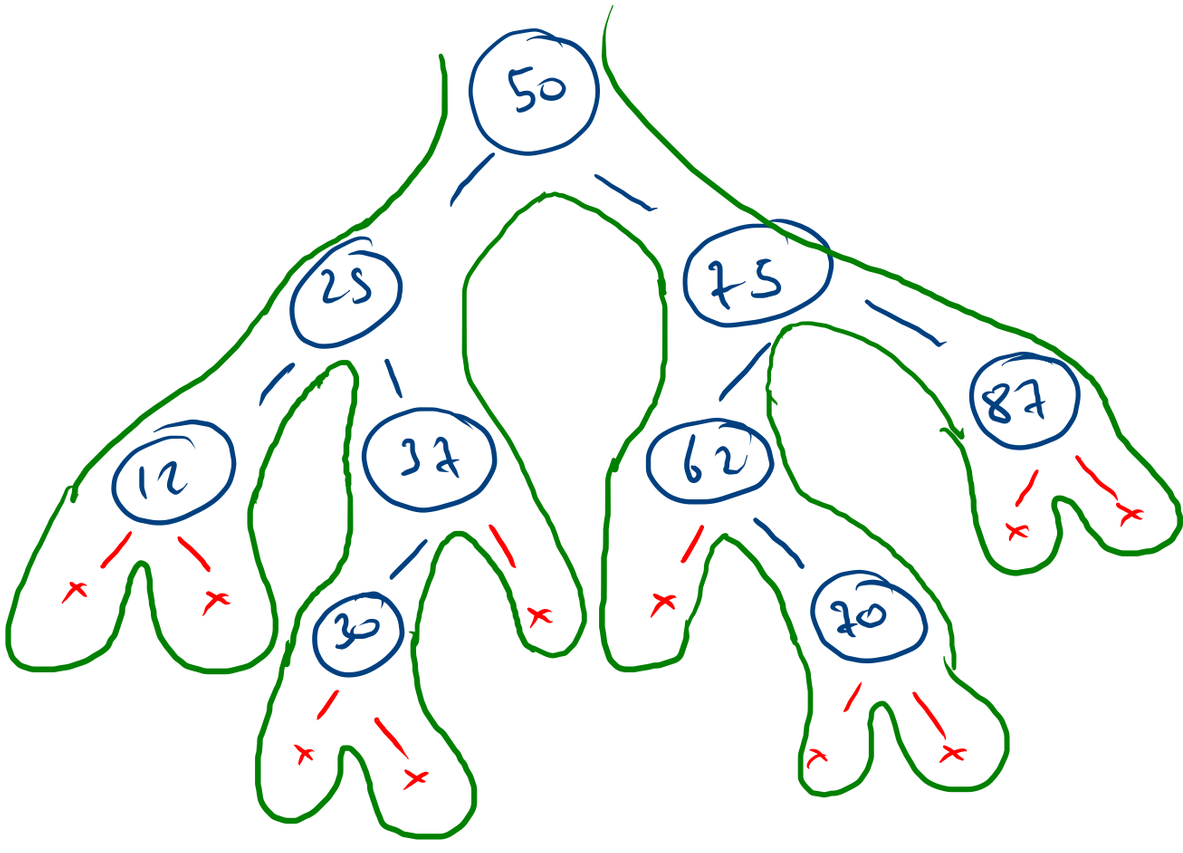
a + b * c



```
Node {
    int data;
    Node left
    Node right
}
```



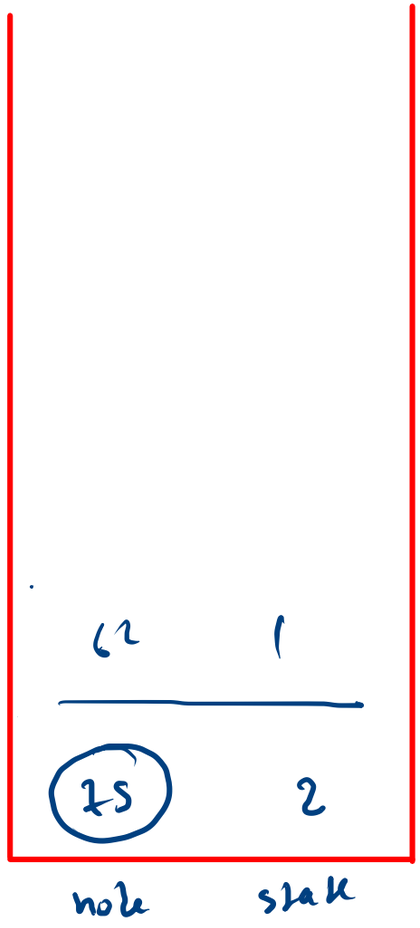
50
25
12
n
n
37
30
n
n
n
75
62
n
70
n
n
87
n
n

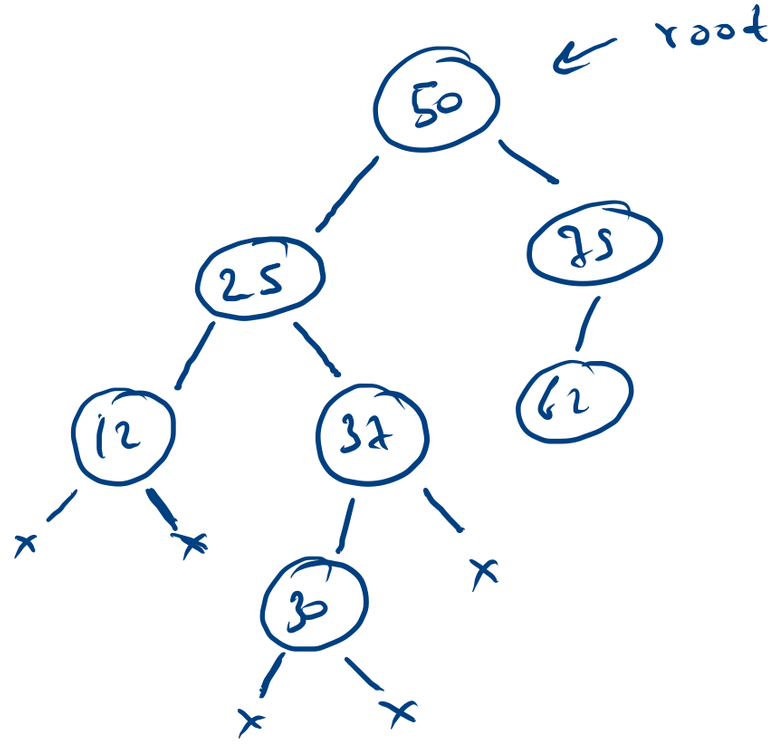


50
25
12
null
n
37
30
n
n
n
75
62
n
70
n
n
87
n
n

int arr[] = new int

Integer arr[]



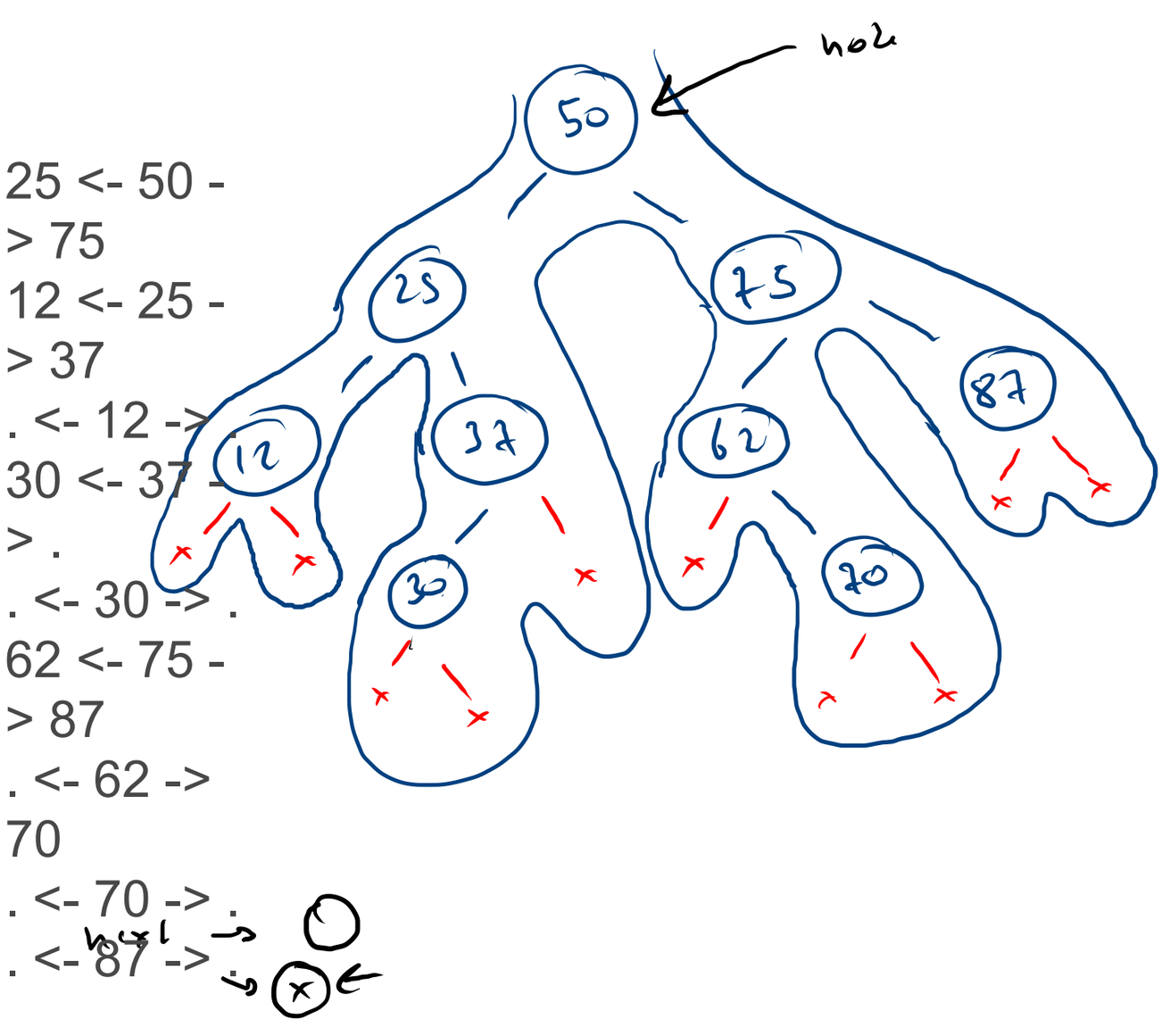


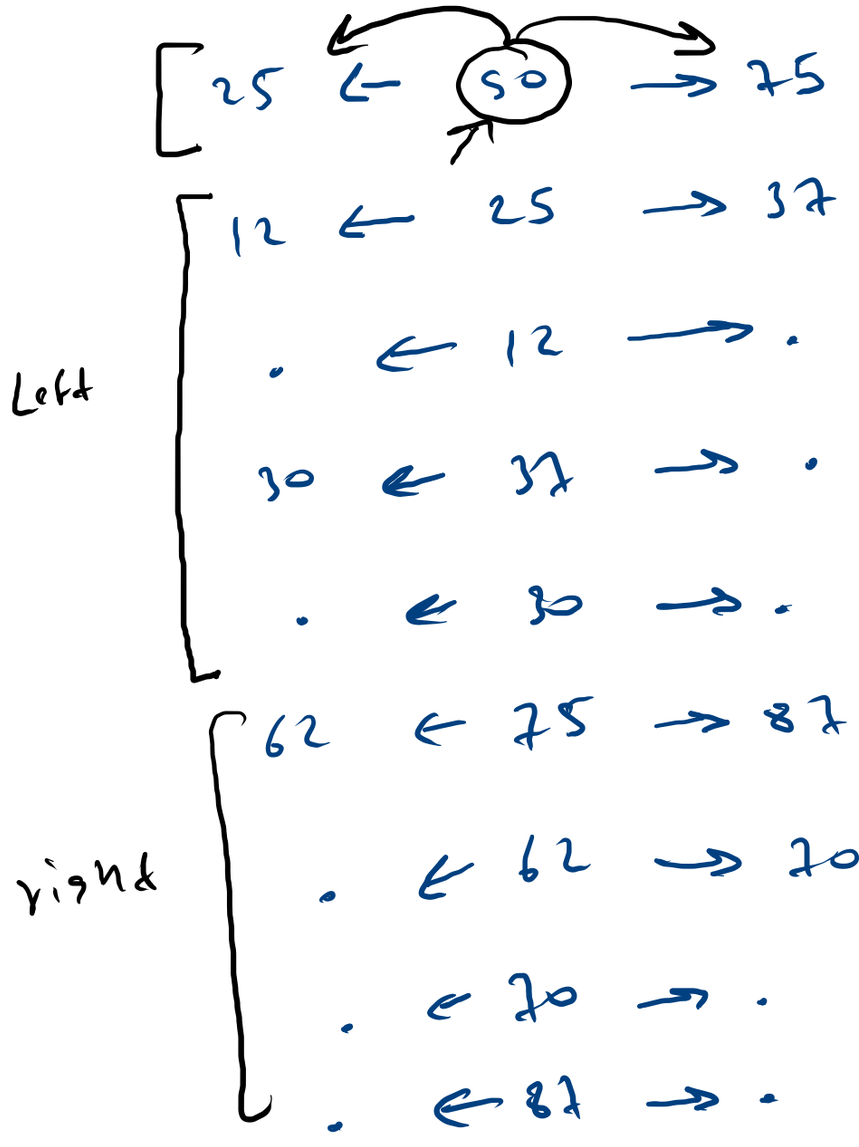
node = new Node (arr[idx])

start = 1 add node as left child

= 2 add right child & pop

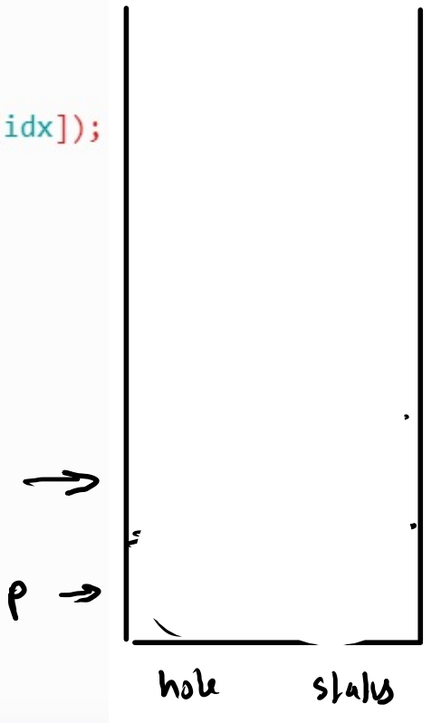
push node

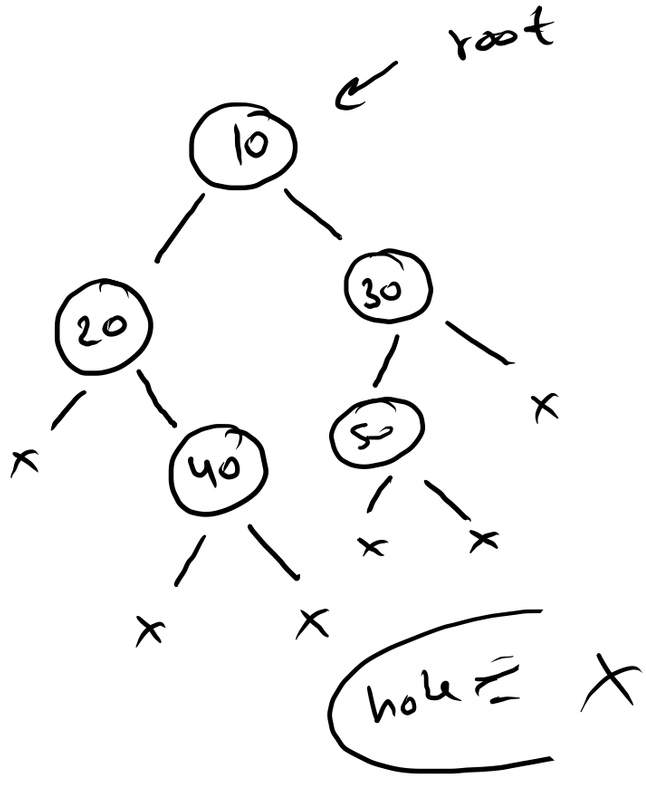

25 <- 50 -
> 75
12 <- 25 -
> 37
. <- 12 ->
30 <- 37 -
> .
. <- 30 -> .
62 <- 75 -
> 87
. <- 62 ->
70
. <- 70 -> .
. <- 87 ->
50
25
75
12
37
62
87
30
70
next

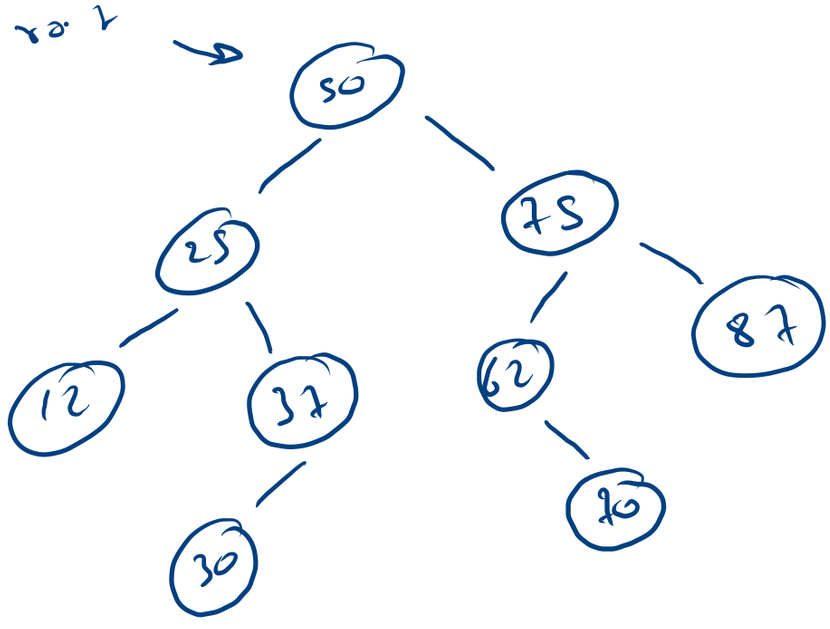

25 ← 50 → 75
Left
12 ← 25 → 37
. ← 12 → .
30 ← 37 → .
. ← 30 → .
right
62 ← 75 → 87
. ← 62 → 70
. ← 70 → .
. ← 87 → .

[10 20 null, 40 n, n, 30, 50, n, n, n]
idx

```
int idx = 1;

Node root = new Node(arr[0]);
Stack<Pair> st = new Stack<>();
st.add(new Pair(root, 1));

while(idx < arr.length){
    Node node = null;
    if(arr[idx] != null)node = new Node(arr[idx]);
    idx++;

    Pair p = st.peek();

    if(p.status == 1){
        p.node.left = node;
        p.status++;
    }else {
        p.node.right = node;
        st.pop();
    }

    if(node != null){
        st.push(new Pair(node, 1));
    }
}
```







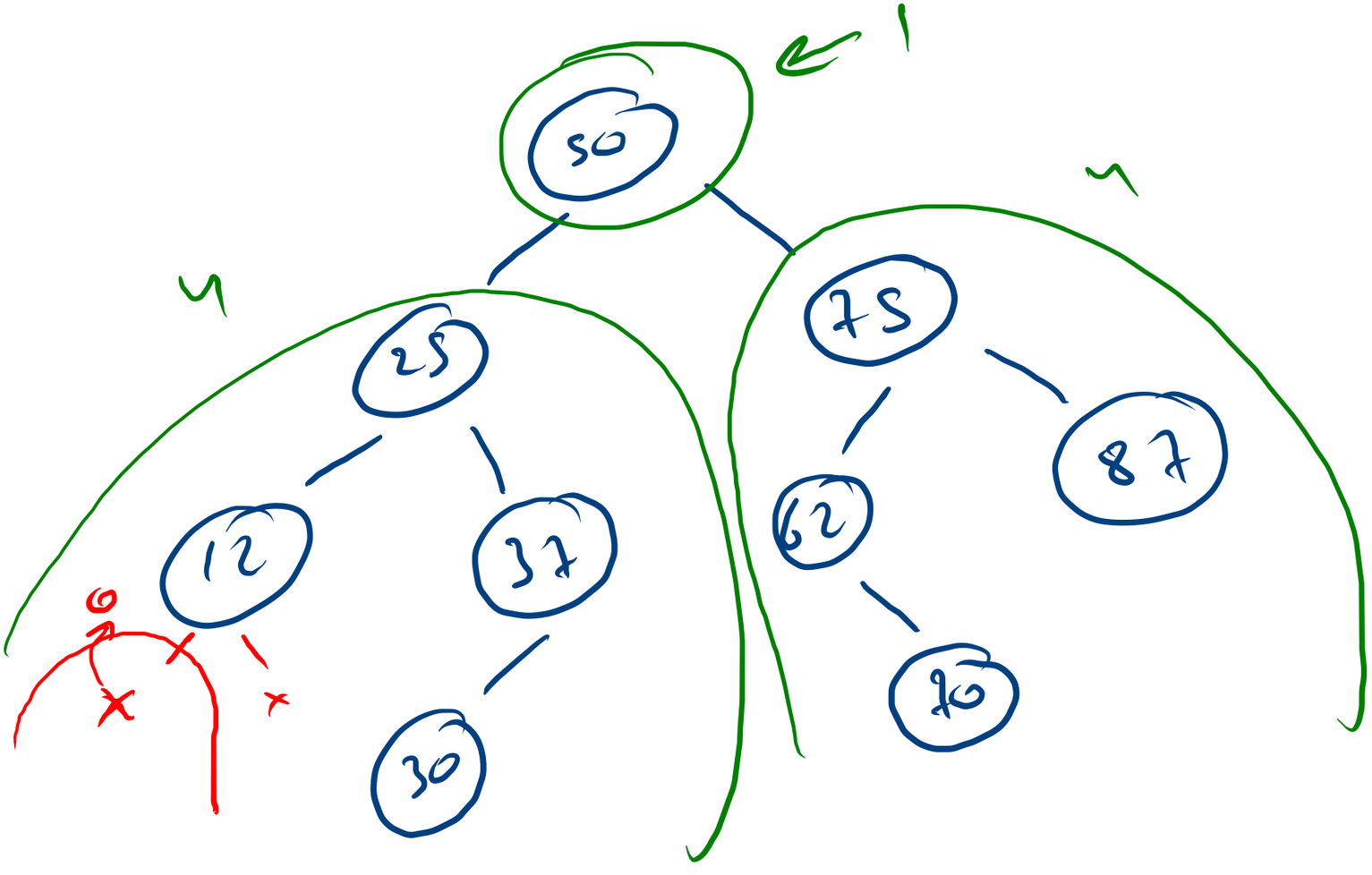
size → 9

sum → 250+12+37+62+87

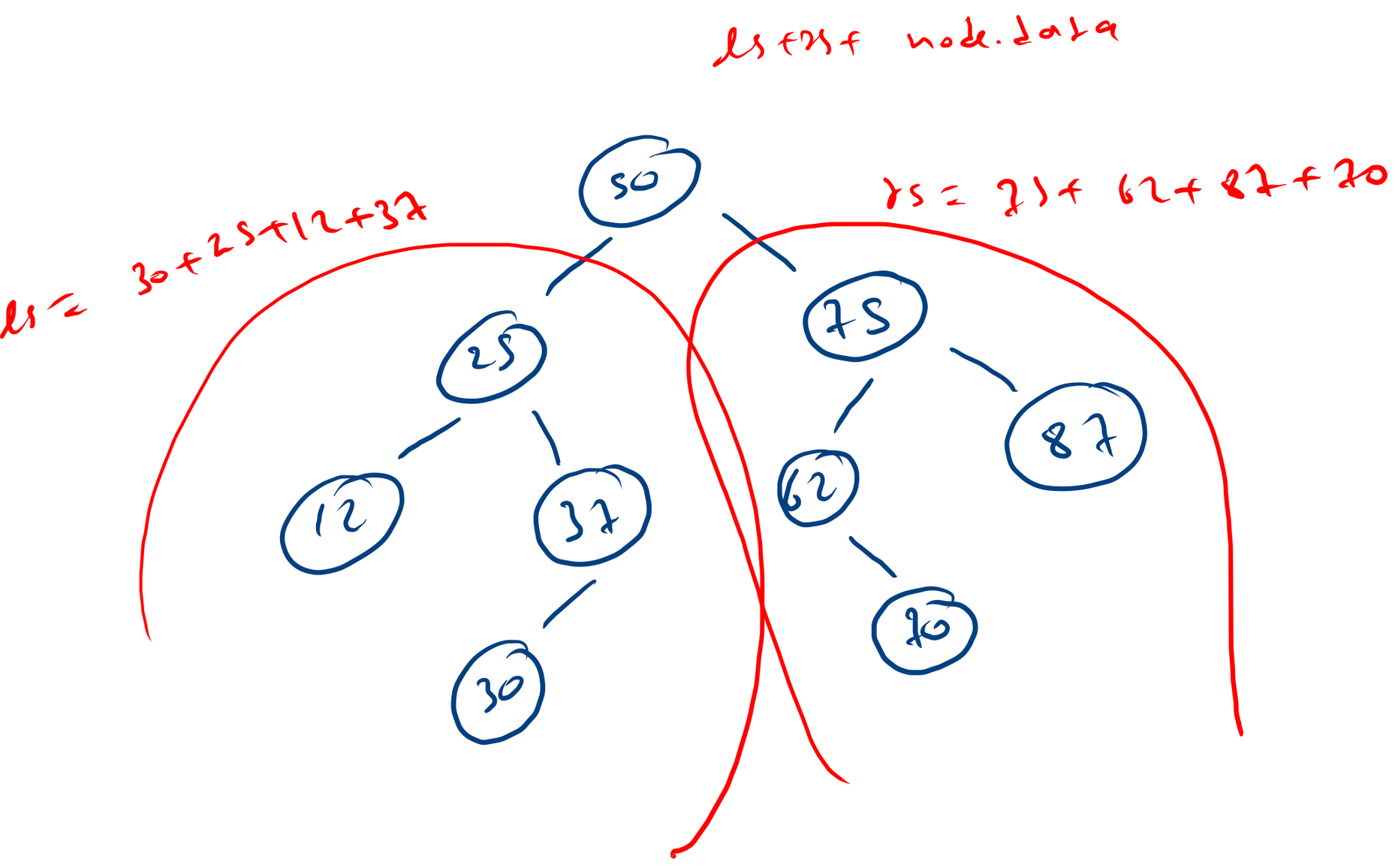
max → 87

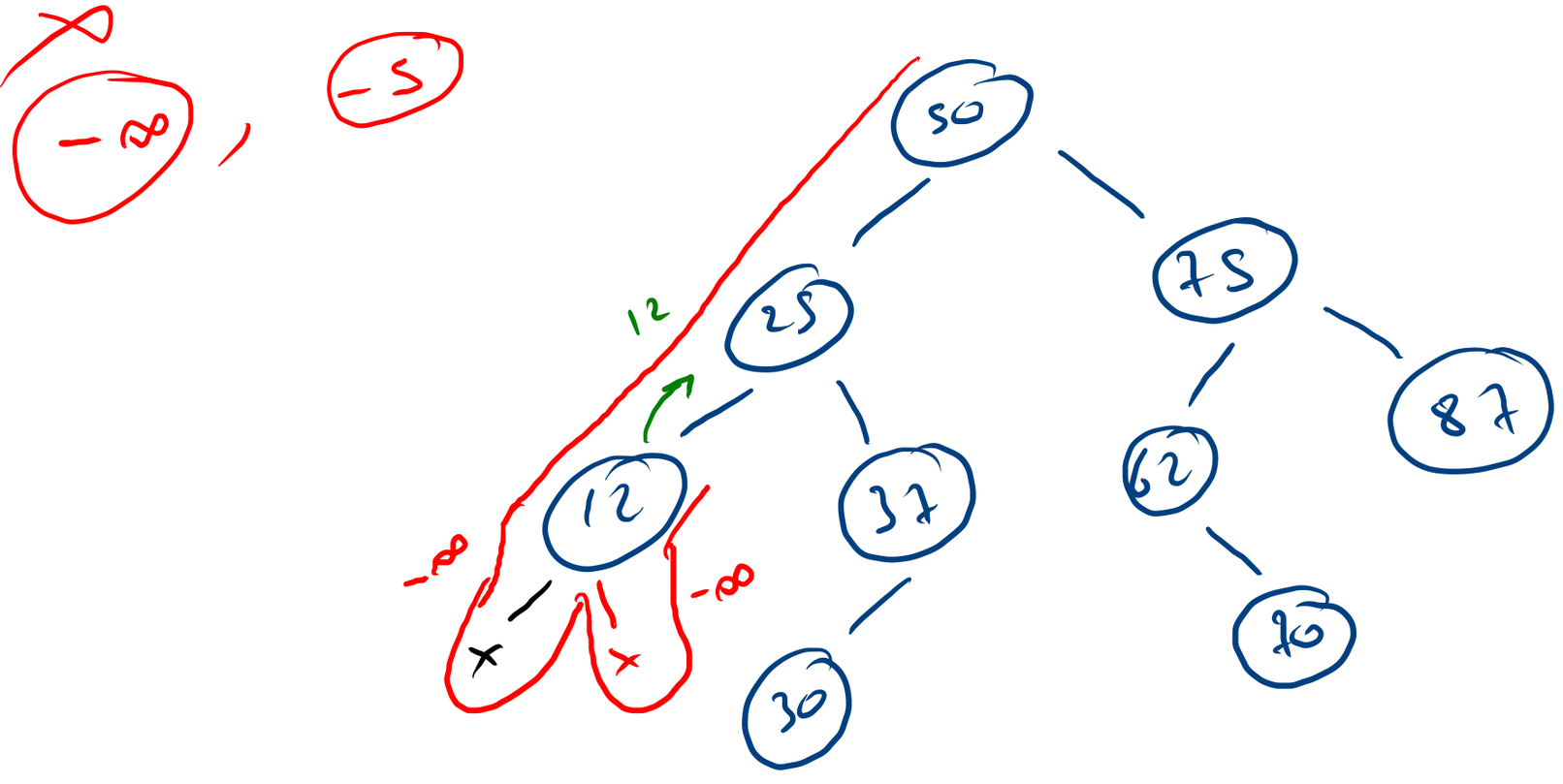
height = 3

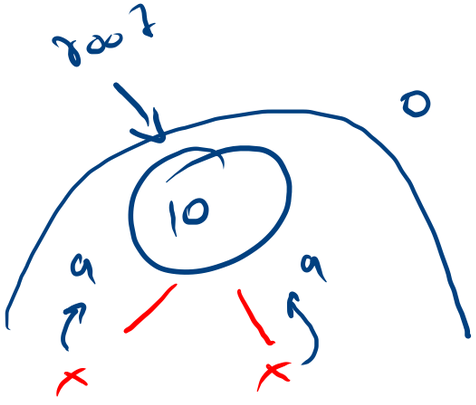
Size, Sum, Maximum And Height Of A Binary Tree



size →


ls += node.data
ls = 30+25+12+37
rs = 75+62+87+20
50
25
75
12
37
62
87
30
70

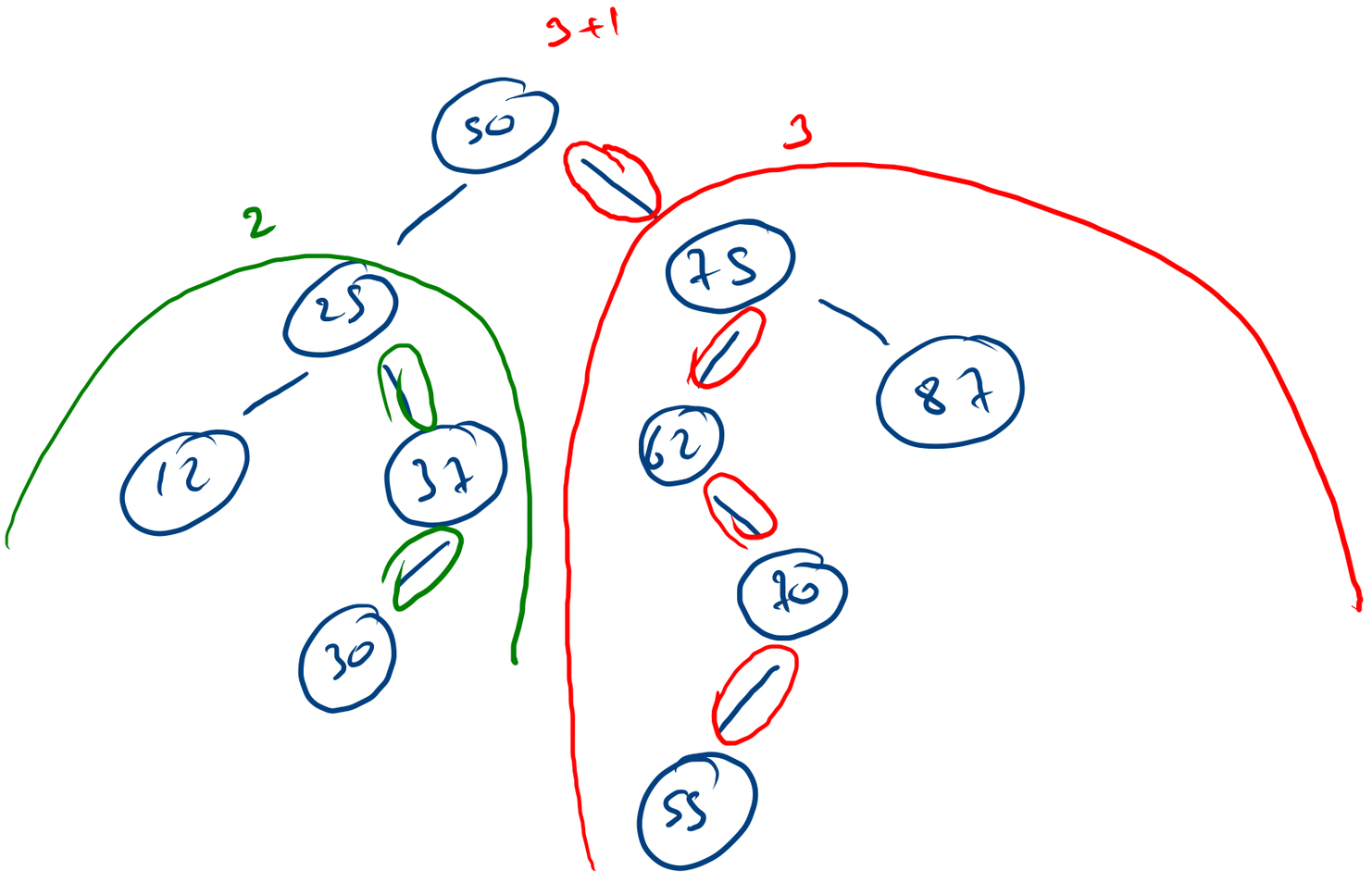

-∞
-5
50
25
75
12
12
37
62
87
-∞
-∞
30
70



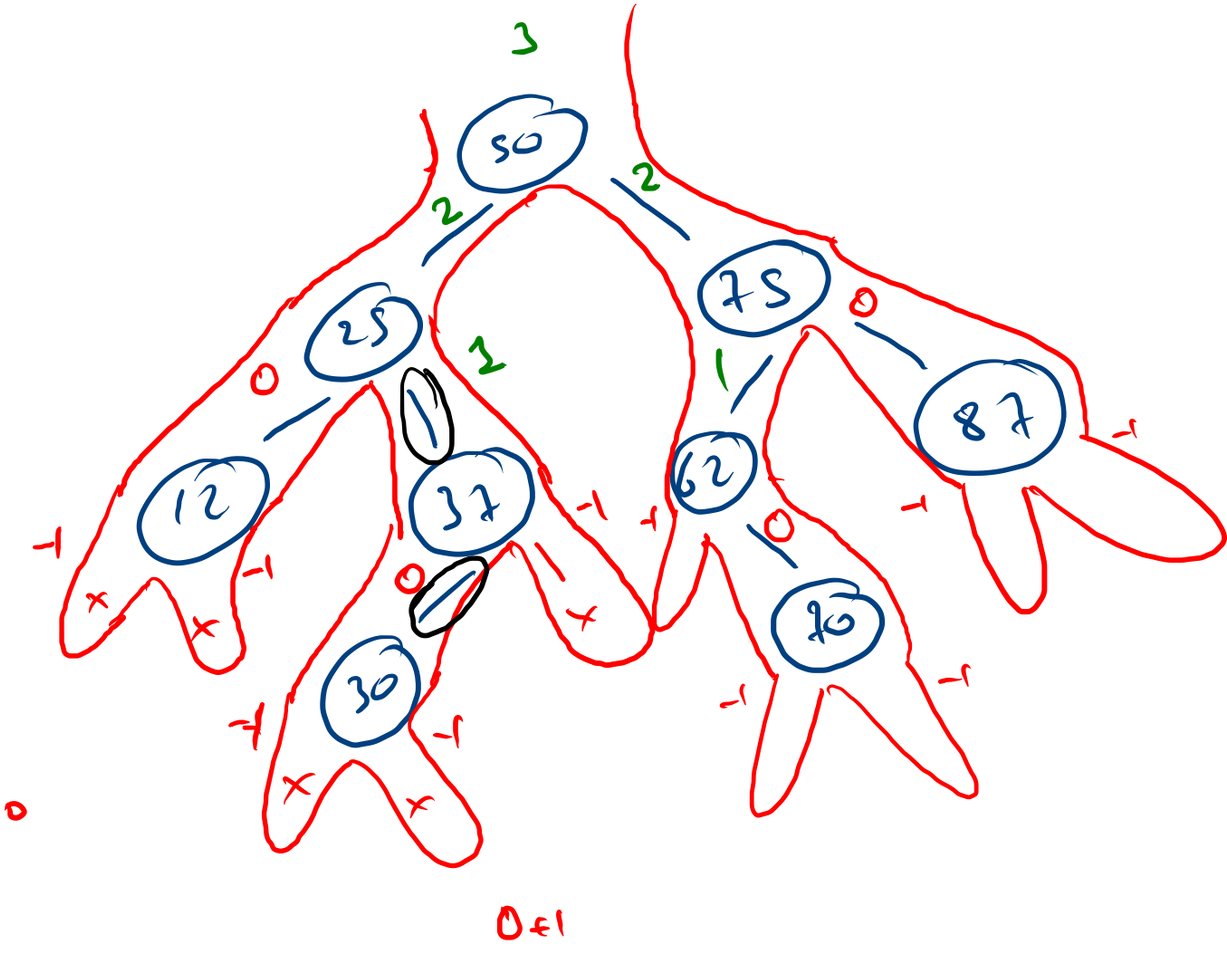
$\theta = \max(a, a) + 1$

$\theta - 1 = \max(a, a)$

$-1 = a$



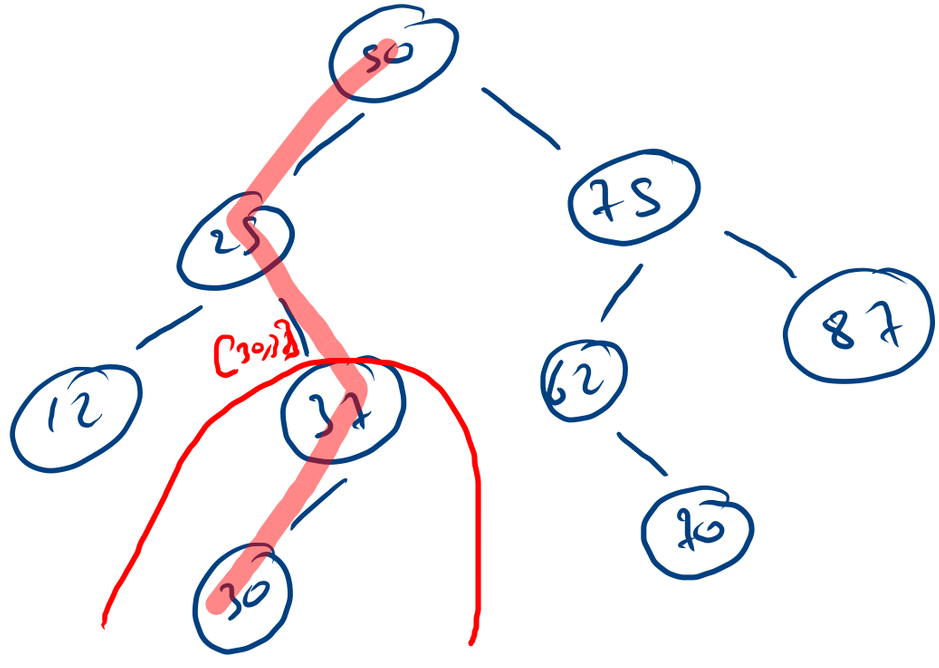
```java
public static int height(Node node) {
    if(node == null){
        return -1;
    }
    int lh = height(node.left);
    int rh = height(node.right);
    return Math.max(lh, rh)+1;
}
```



[30, 37, 25, 50]

data = 30

data = 33 → [ ]



19
50 25 12 n n 37 30 n n n 75 62 n 70 n n 87 n n
30

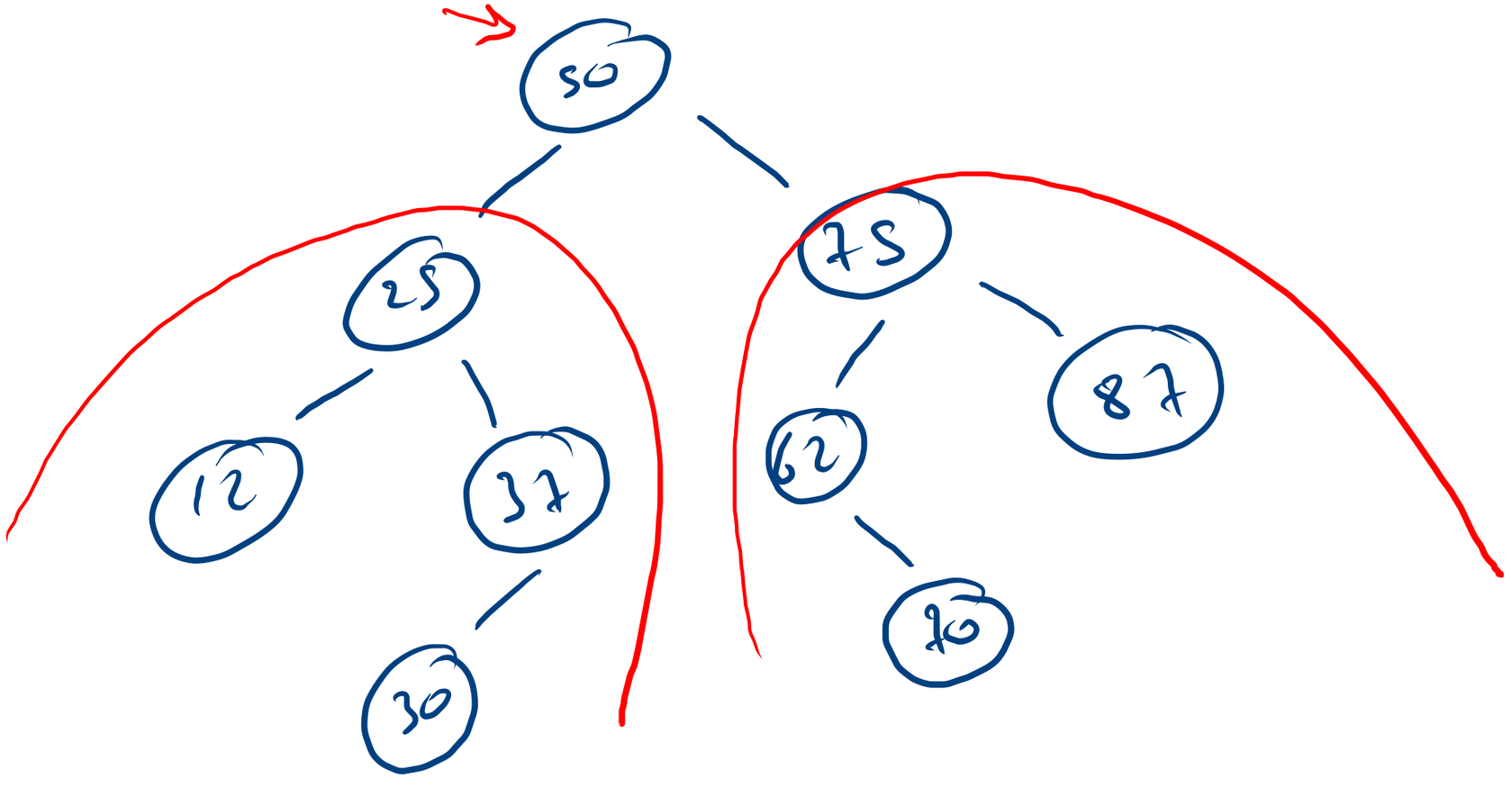
Sample Output

true
[30, 37, 25, 50]

data 30 → true

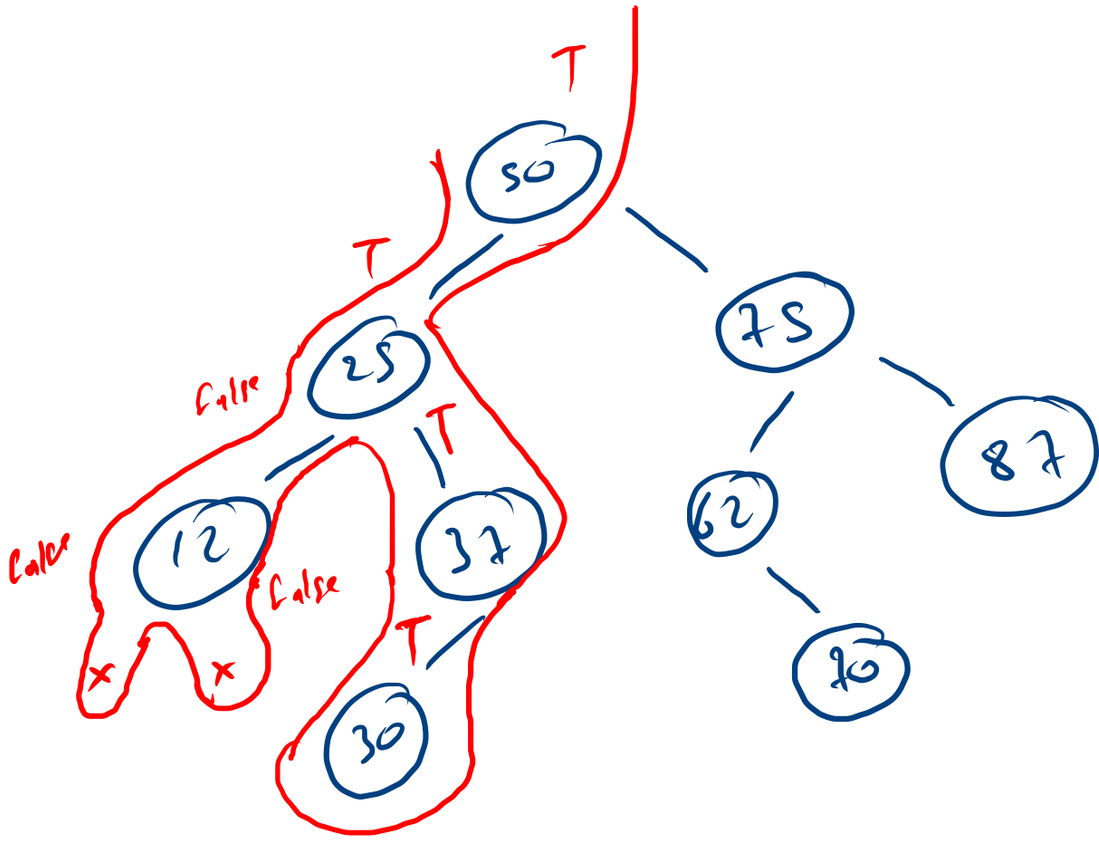
data 33 → false

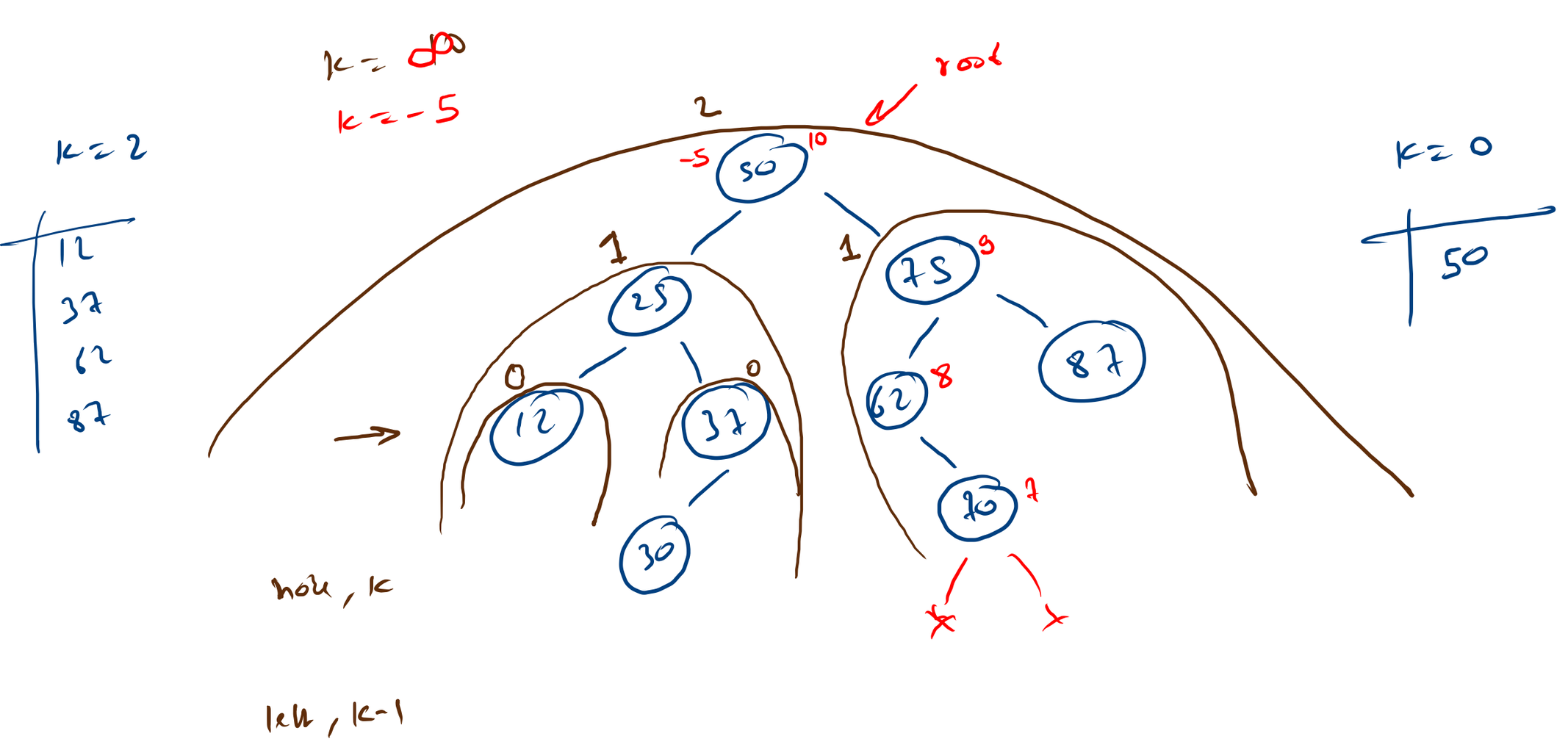
data ?



data = 30

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

    if(node.data == data){
        return true;
    }

    boolean lf = find(node.left, data);
    if(lf){
        return true;
    }

    boolean rf = find(node.right, data);
    if(rf){
        return true;
    }

    return false;
}
```




k = ∞
k = -5
root
2
-5
50
10
1
25
1
75
9
0
0
12
37
62
8
87
30
10
7
k = 2
12
37
62
87
k = 0
50
node, k
left, k-1

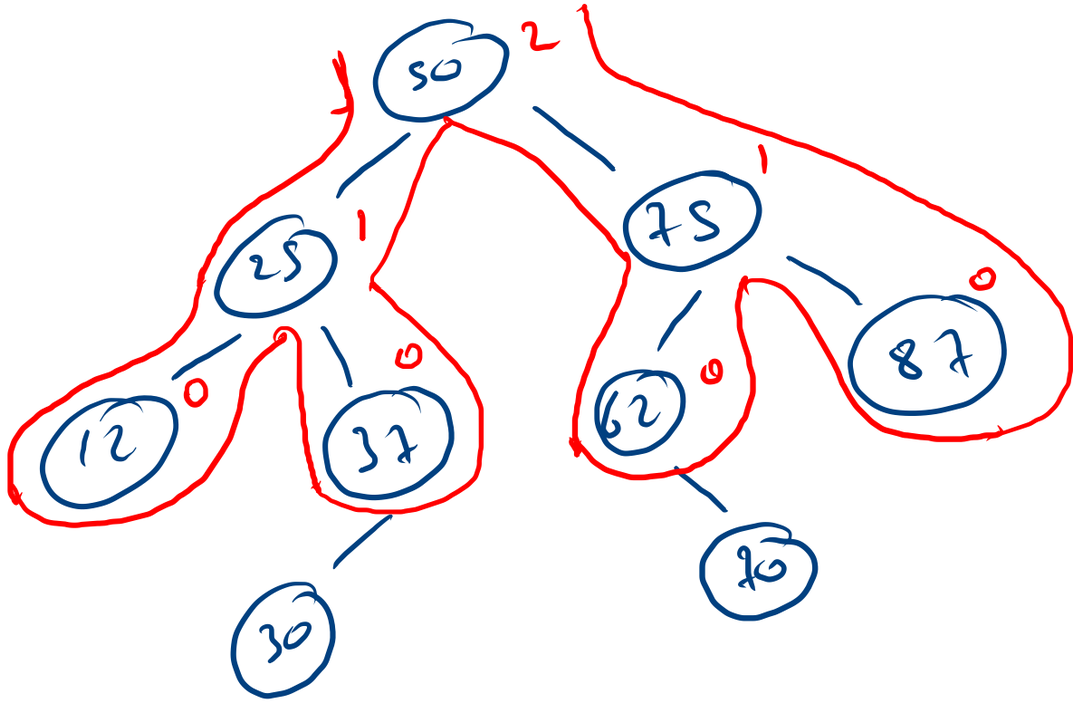
12

37

62

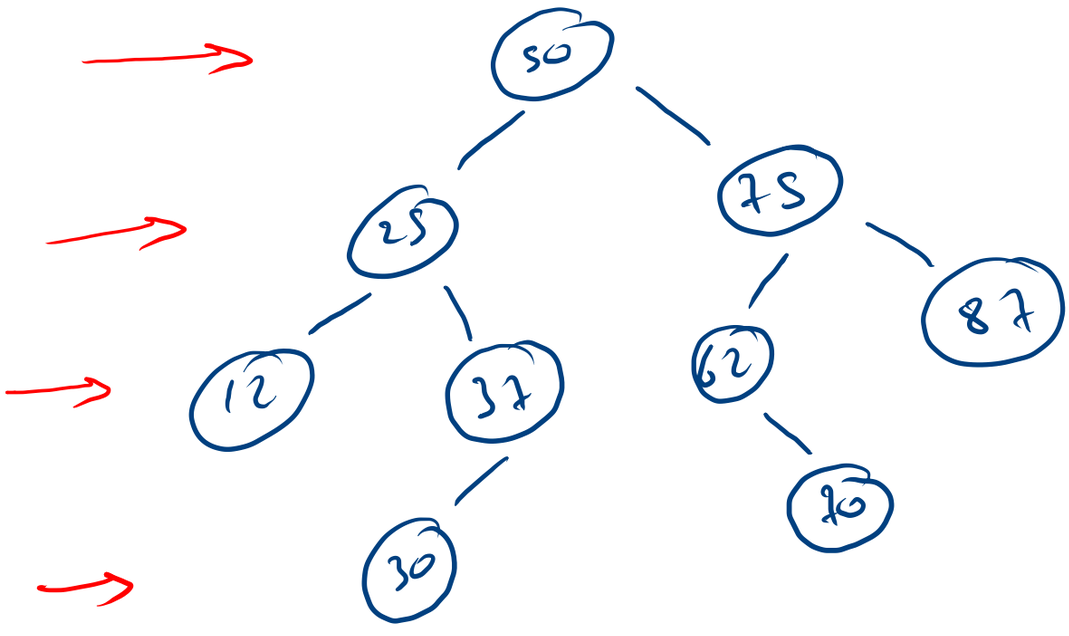
87

```java
public static void printKLevelsDown(Node node, int k){
    if(node == null || k < 0){
        return;
    }

    if(k == 0){
        System.out.println(node.data);
        return;
    }

    printKLevelsDown(node.left, k-1);
    printKLevelsDown(node.right, k-1);

}
```



H.W

Levelorder Traversal Of Binary Tree



50

25 75

12 37 62 87

30 70