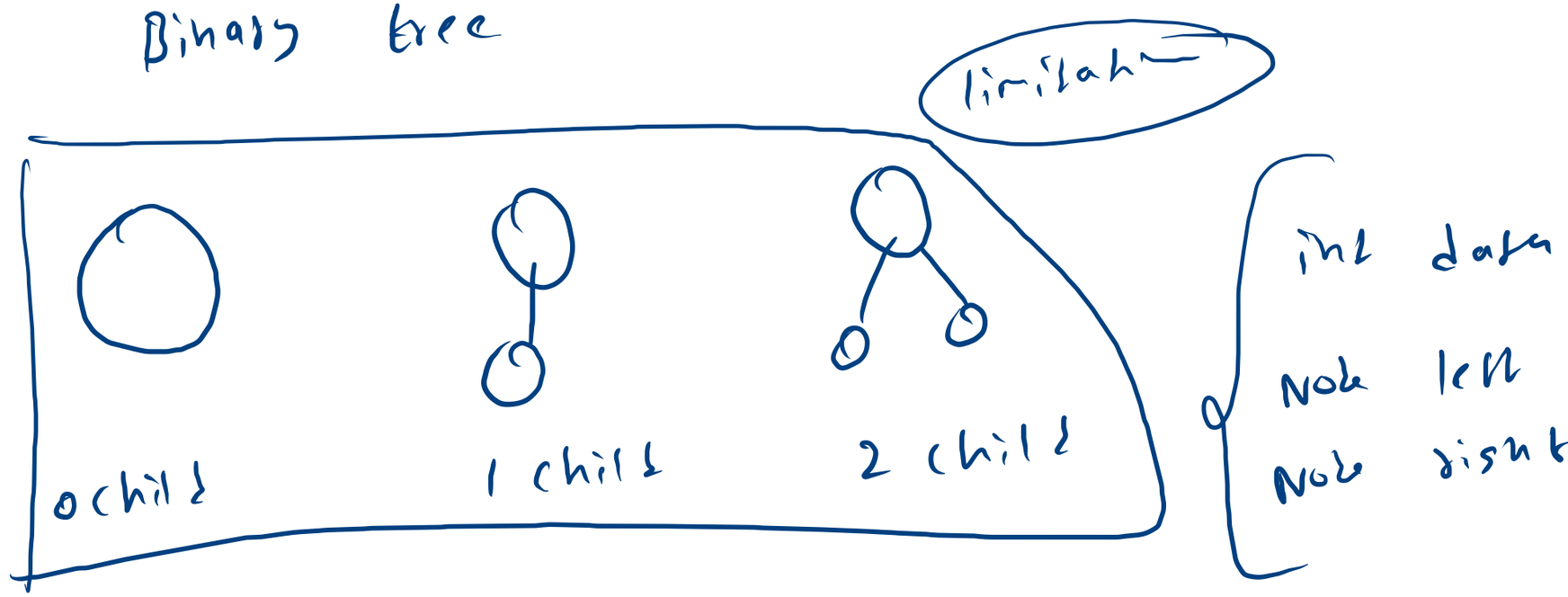
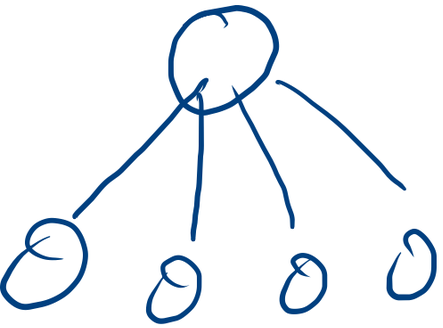
Binary tree

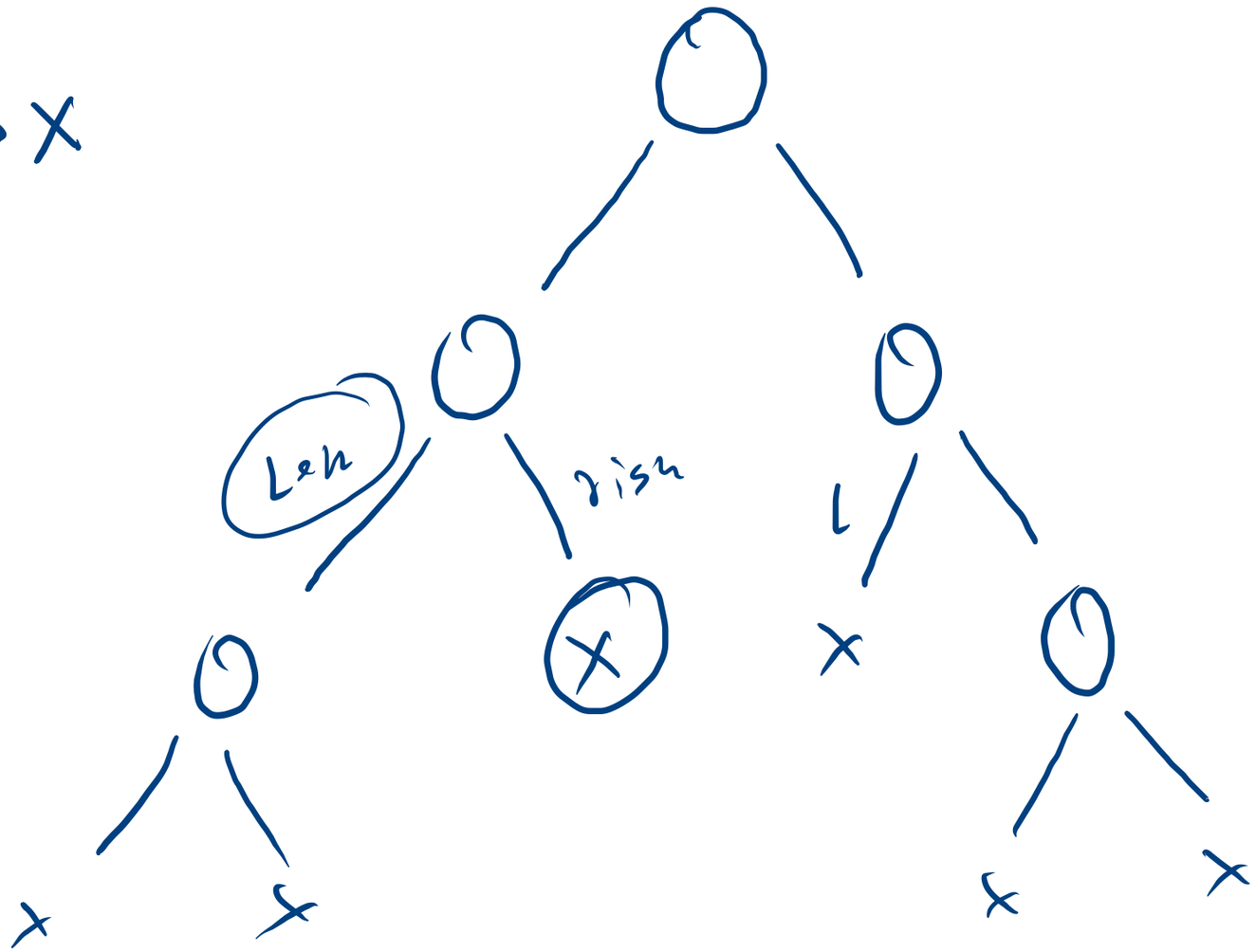


data

AL child



Node next

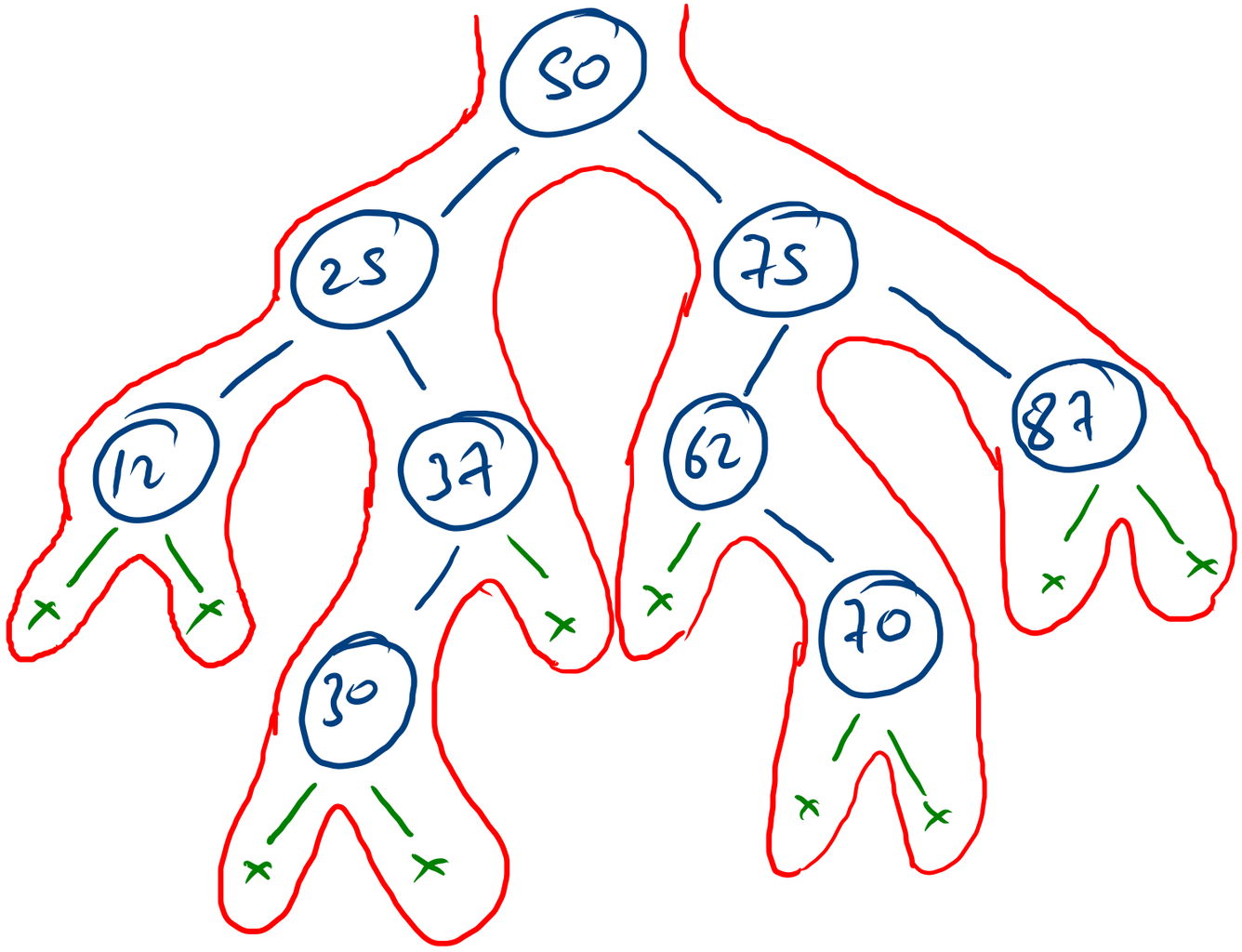

Left
right
L

0
1
2

[ ] → 0

[•] → 1

[• , •] → 2



inp	forrat
50	70
25	h
12	h
h	87
h	h
37	h
30	
h	
h	
h	
75	
62	
h	

0 → left

1 → right

50 ←
25 ←
12 ←
h ←
h ←
37 ←
30 ←
h ←
h ←
h ←
75 ←
62 ←
h

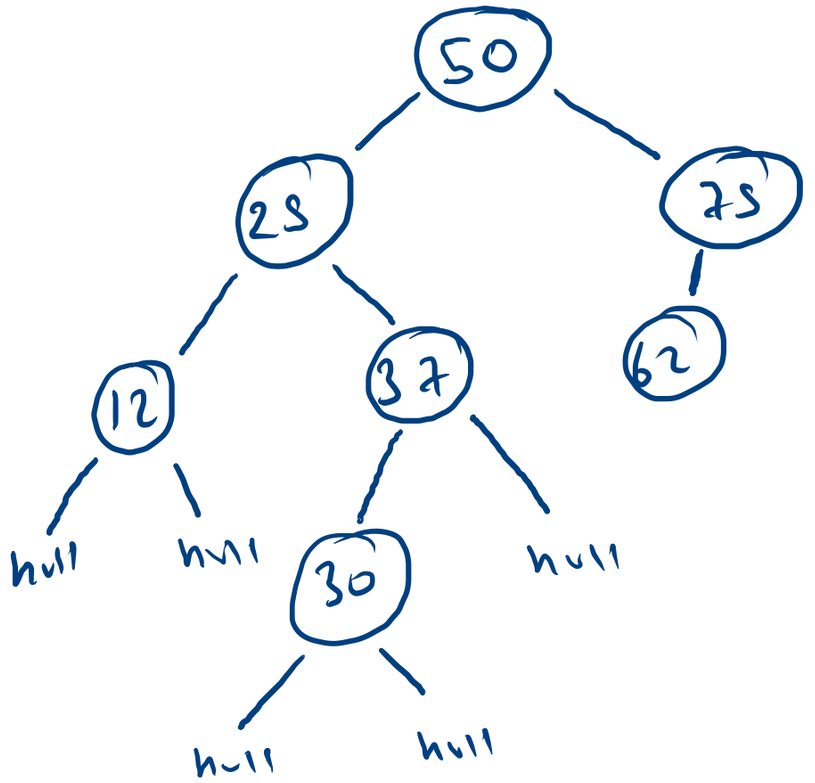
70
h
h
87
h
h
h

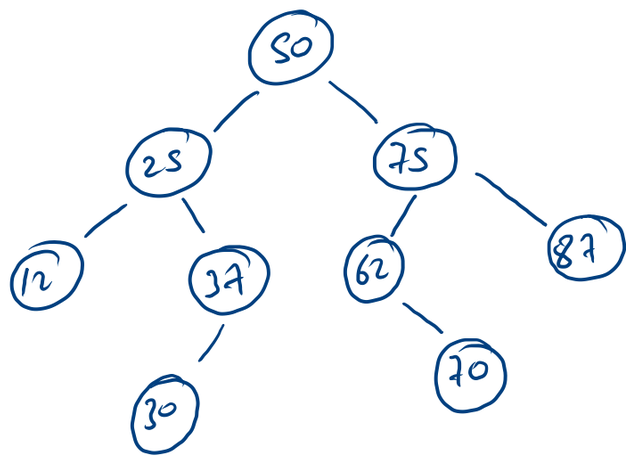
node = new Node ←

make node child of peek ←

if node != null ←
push

62	0
75	0
37	1
25	1
50	1



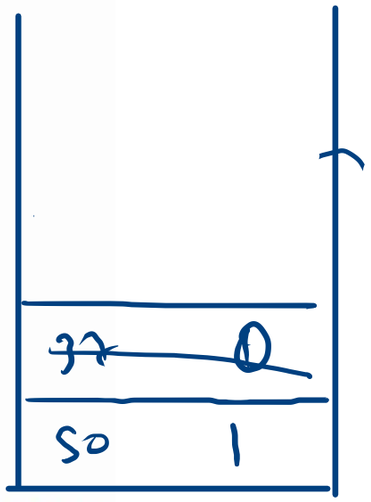


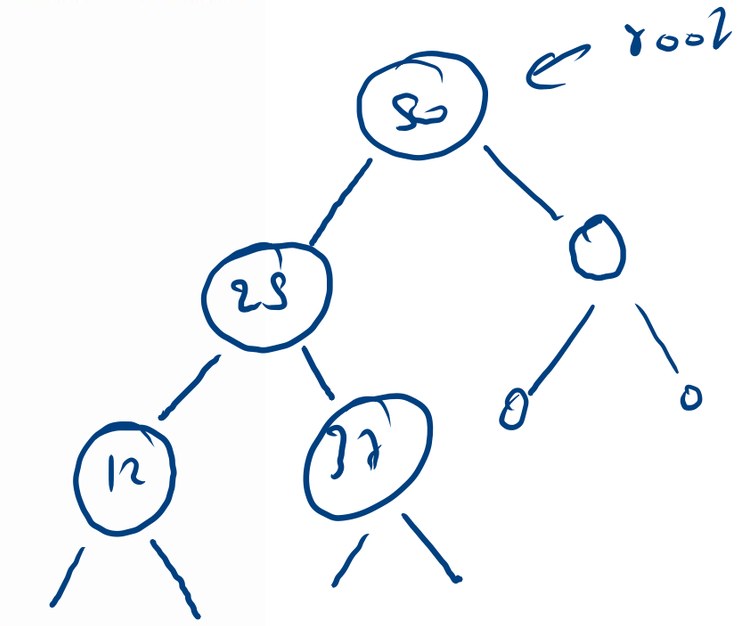
```java
Stack<Pair> st = new Stack<>();
root = new Node(arr[0], null, null);
int idx=1;
Pair p = new Pair(root, 0);
st.push(p);

while(st.size() > 0){
    Node node;
    if(arr[idx] == null){
        node = null;
    }else{
        node = new Node(arr[idx], null, null);
    }
    idx++;

    Pair top = st.peek();
    if(top.status == 0){
        top.node.left = node;
        top.status = 1;
    }else{
        top.node.right = node;
        st.pop();
    }

    if(node != null){
        st.push(new Pair(node, 0));
    }
}
```

new

child

push





→ 50
→ 25
→ 12
→ h
→ h
→ 37
30
h
h
h
75
12
h

70
h
h
87
h
h

```java
public static void display(Node node){
    String left="";
    if(node.left == null){
        left = ".";
    }else{
        left = ""+node.left.data;
    }

    String right="";
    if(node.right == null){
        right = ".";
    }else{
        right = ""+node.right.data;
    }

    System.out.println(left+" <- "+node.data+" -> "+ right);

    if(node.left != null)display(node.left);
    if(node.right != null)display(node.right);
}
```

25 ← 50 → 75

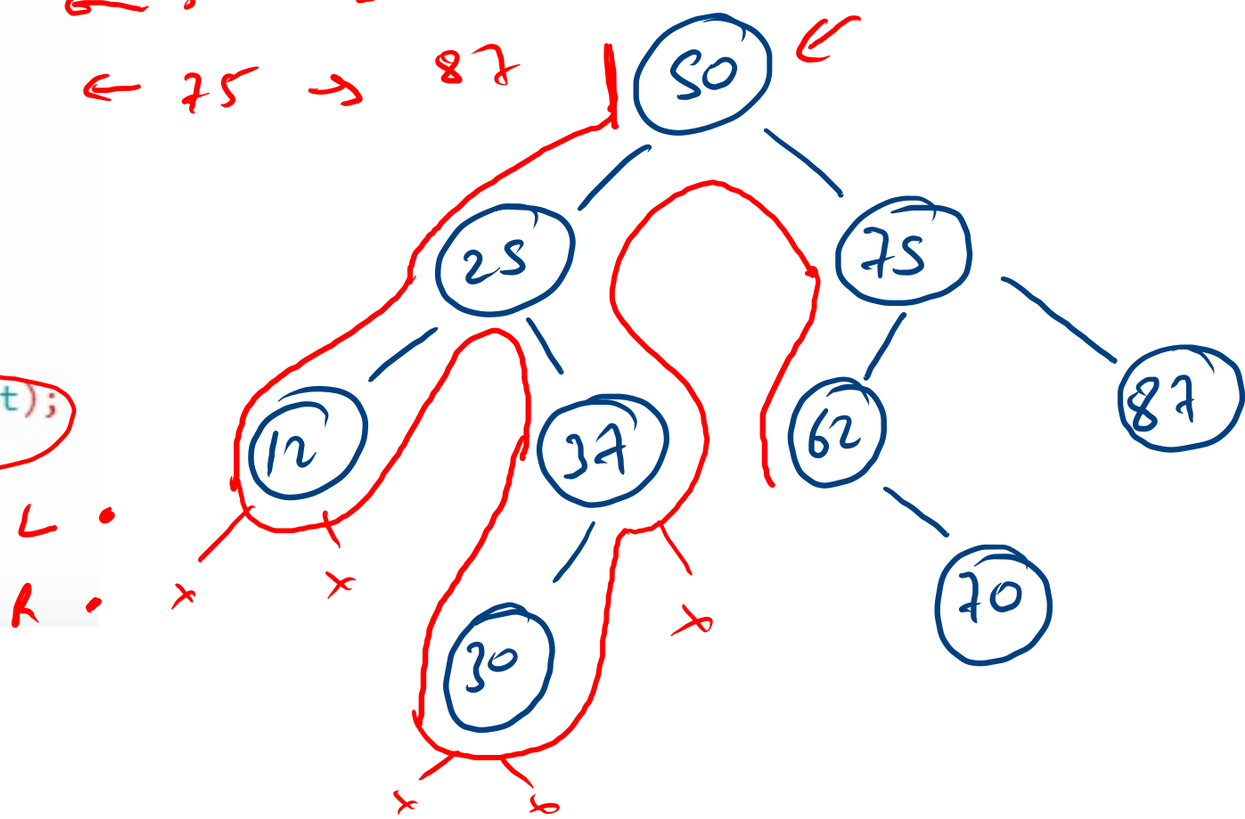
12 ← 25 → 37

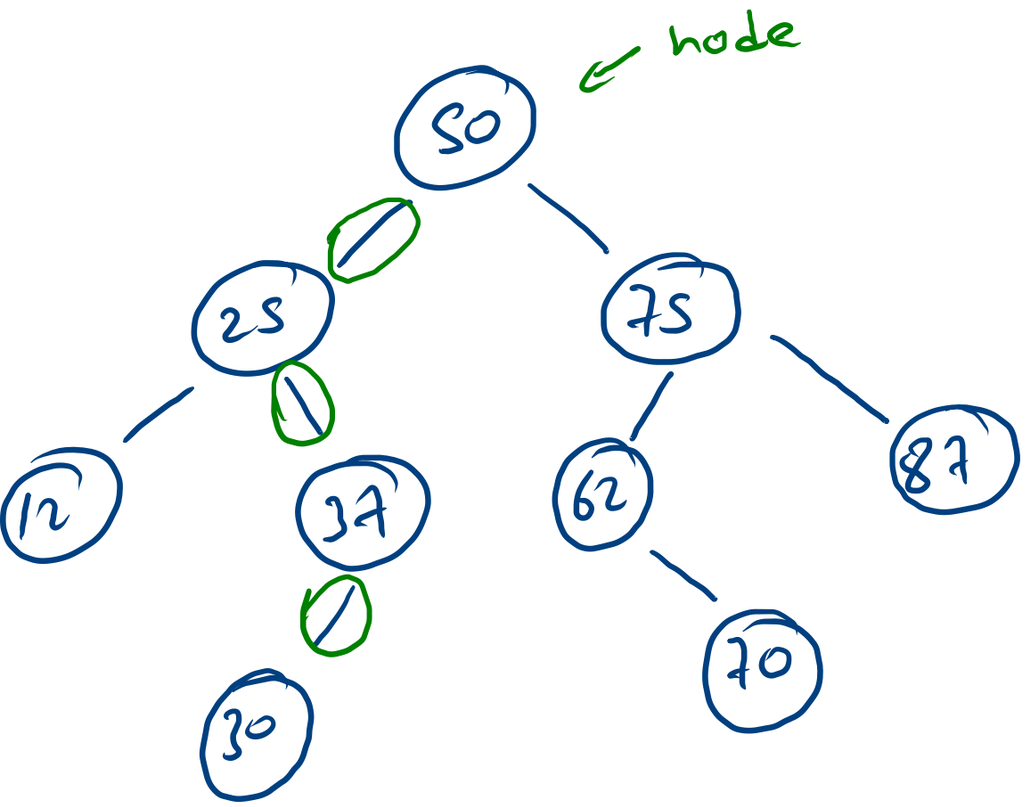
. ← 12 → .

30 ← 37 → .

. ← 30 → .

62 ← 75 → 87





Size → 5

sum → 12+25+30+37+62+70+87

max → 87

height → 3

1/2

call(node.next)

Left
right

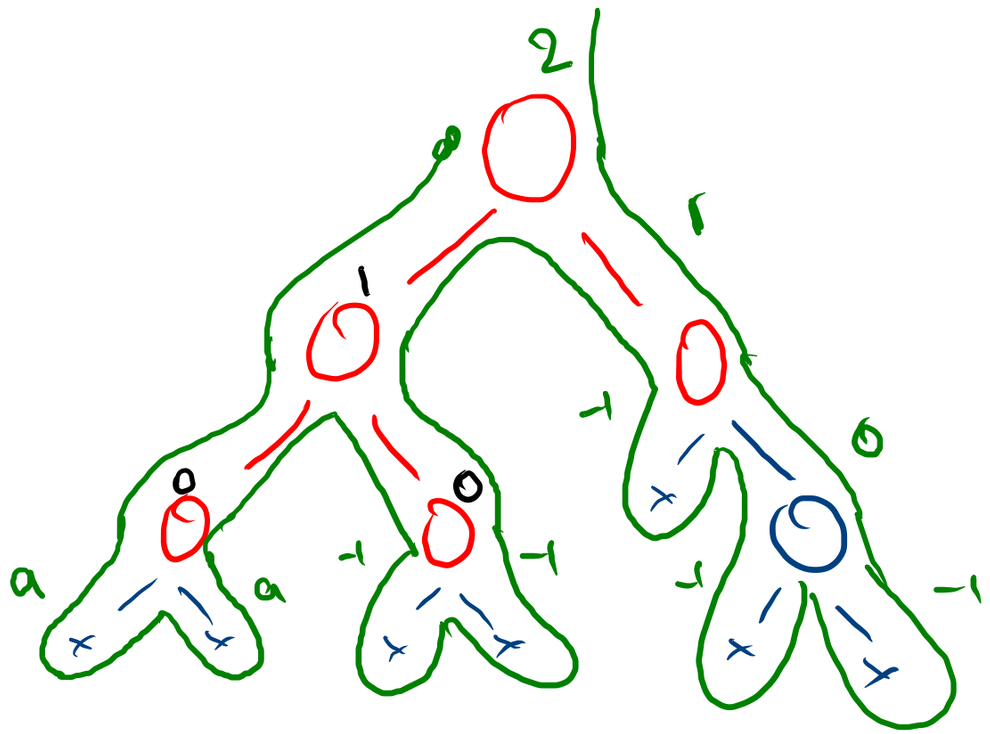
call(left)

call(right)

0+1

```java
public static int height(Node node) {
    if(node == null){
        return ????; 0 , -1
    }
    int lh = height(node.left);
    int rh = height(node.right);

                          0 , 0
    return Math.max(lh, rh)+1;
}
```

-1 +1



1+1=2

0+1=1

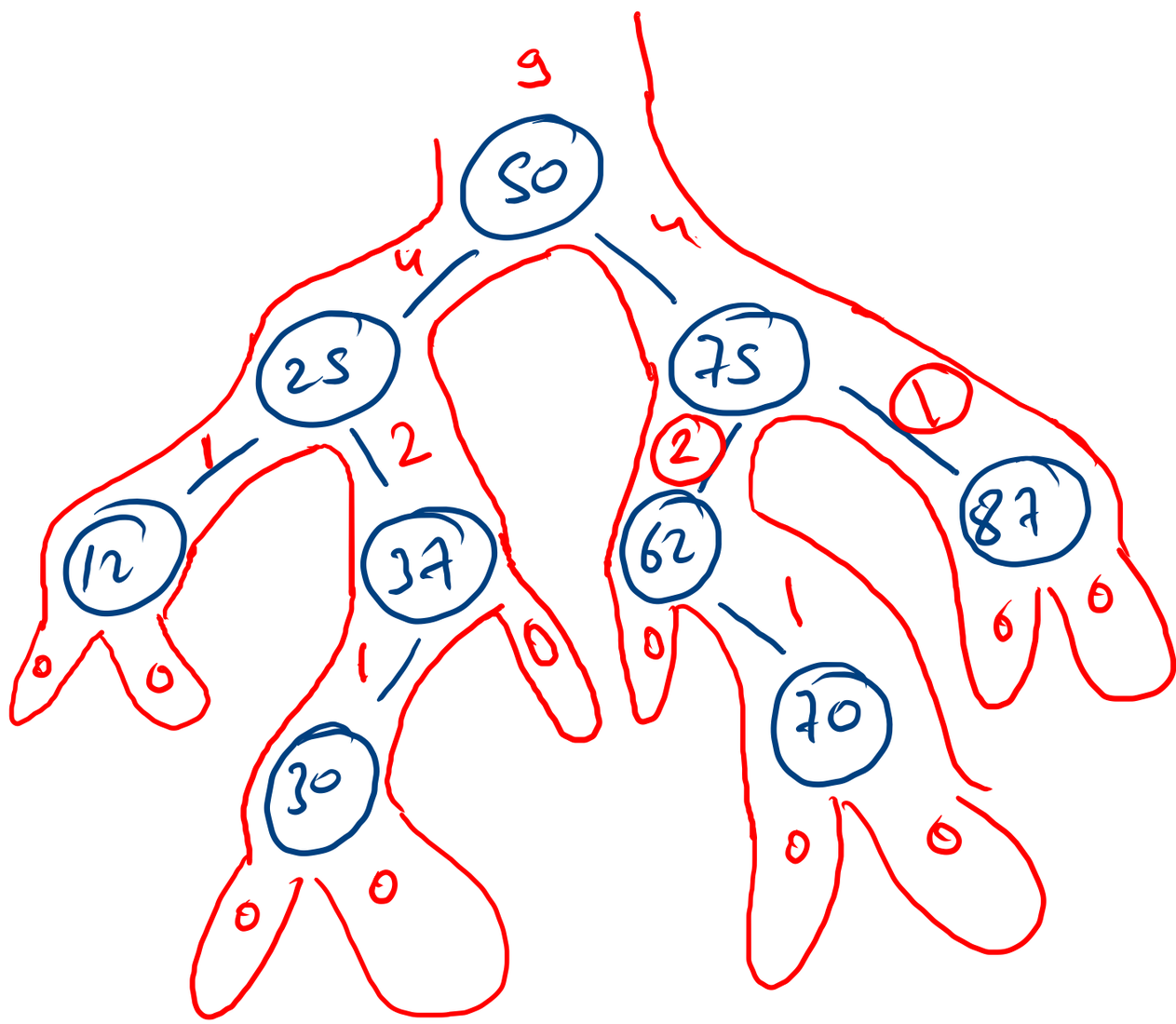
-1+1=0

0 = max(0, 0) +1

0-1 = max(0, 0)

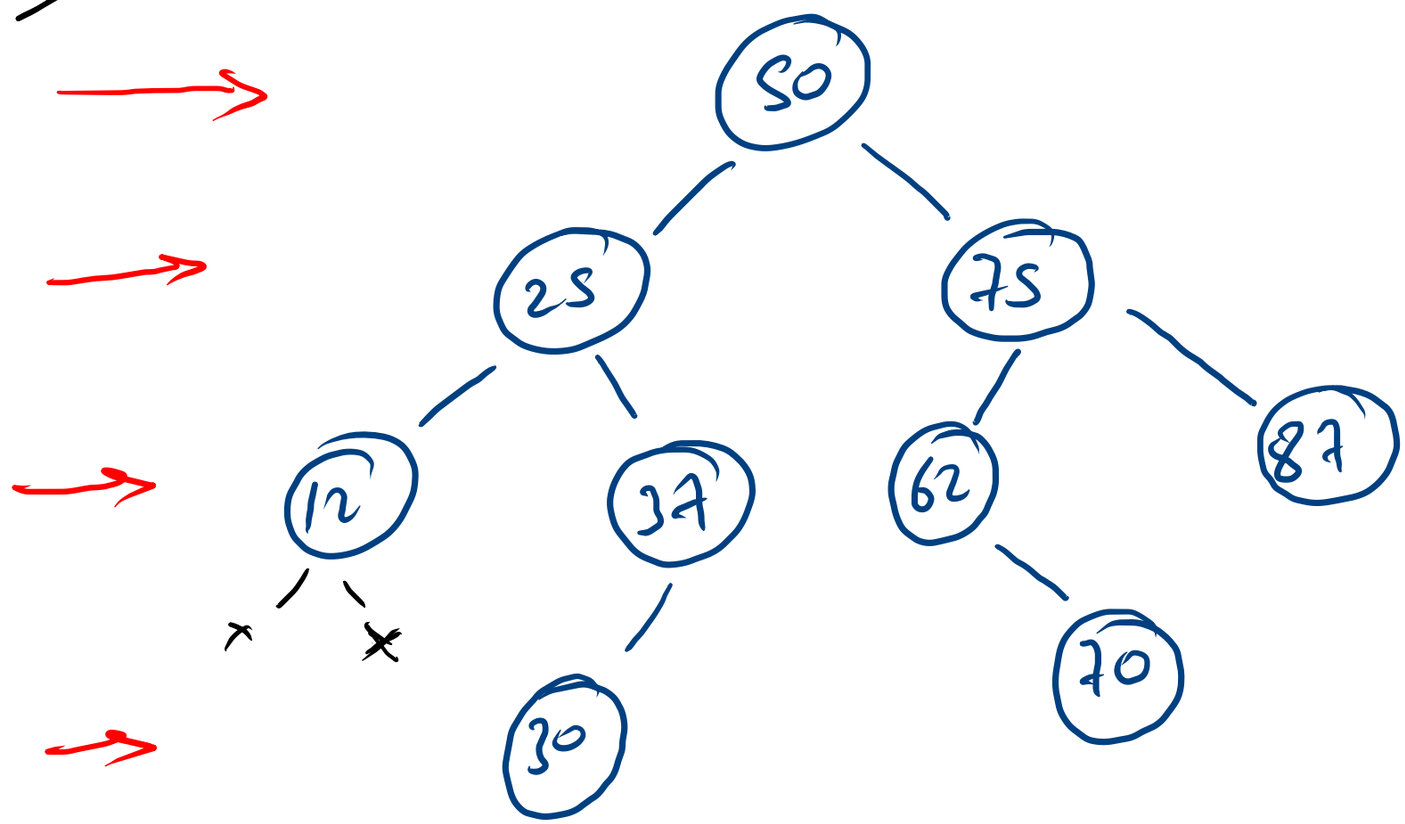
-1 = 0 ←

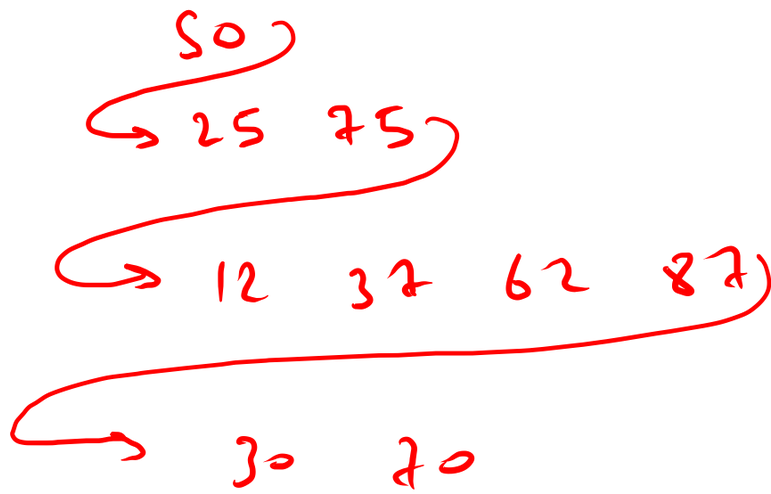
```java
public static int size(Node node) {
    if(node == null)return 0;

    int left = size(node.left);
    int right = size(node.right);

    return left+right+1;
}
```

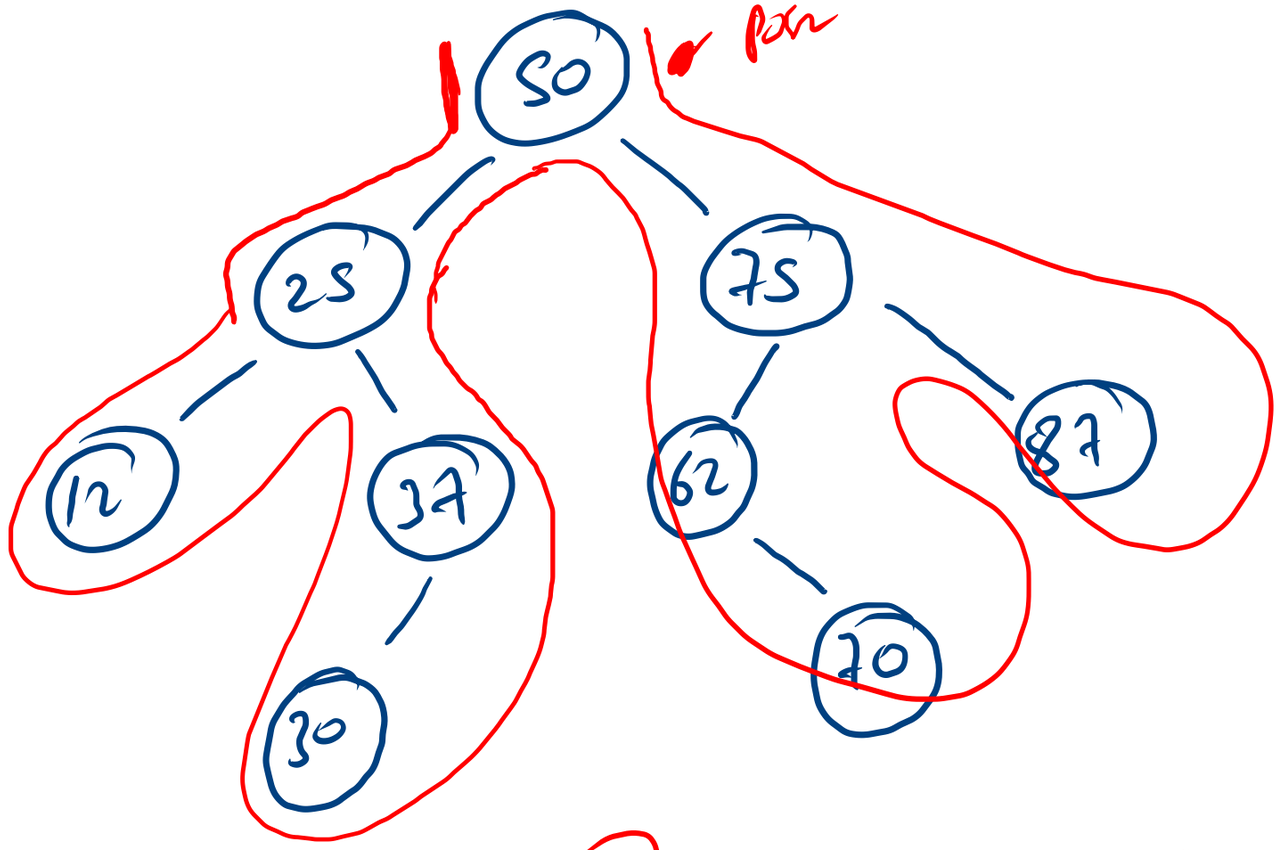


H.W.

Levelorder Traversal Of Binary Tree







Pre

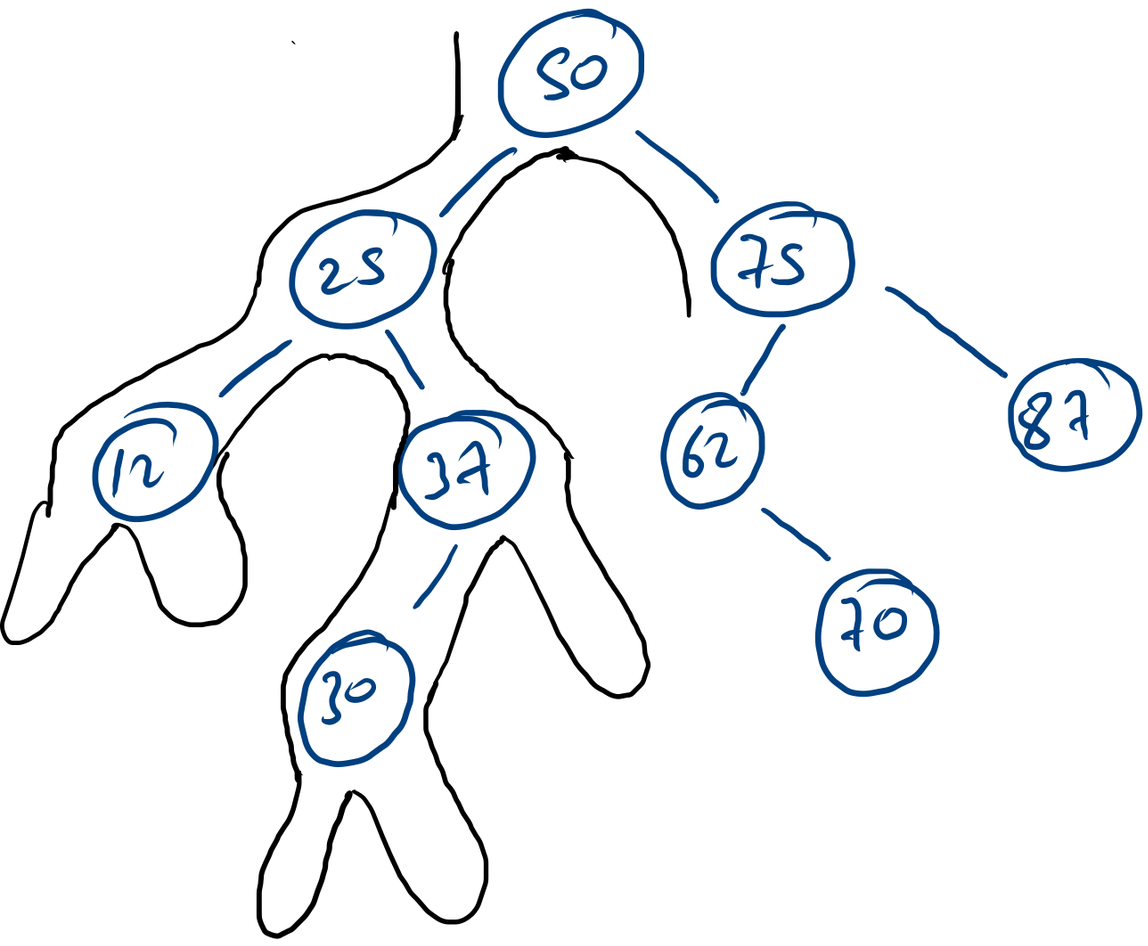
50

Post

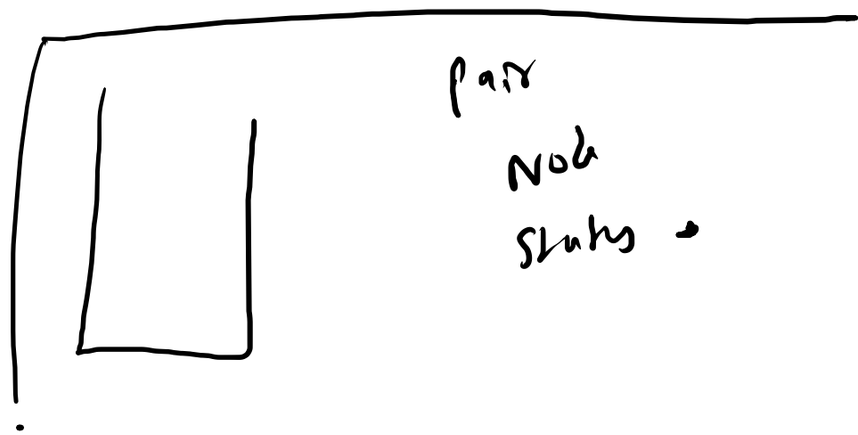
50

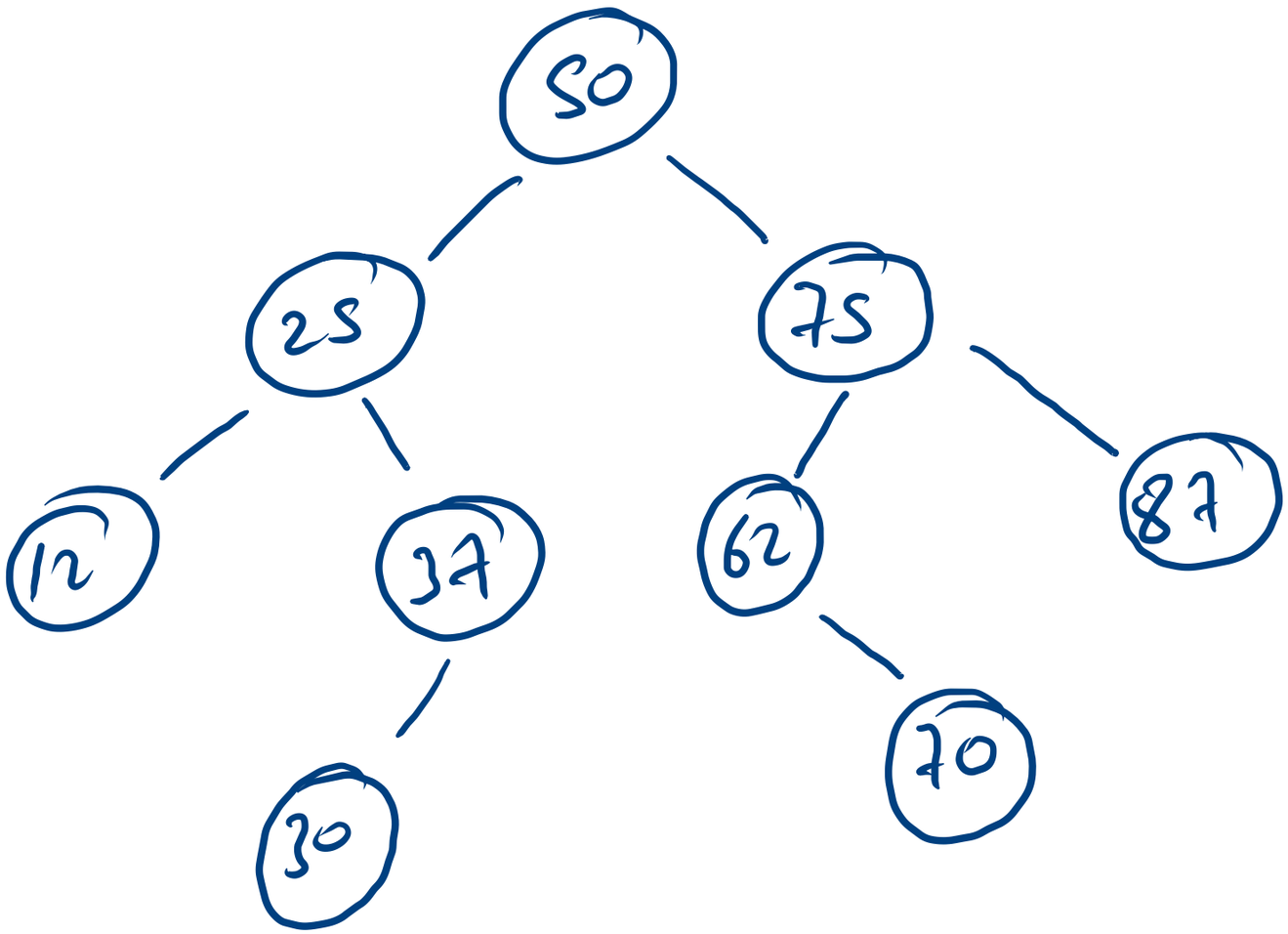
Inord

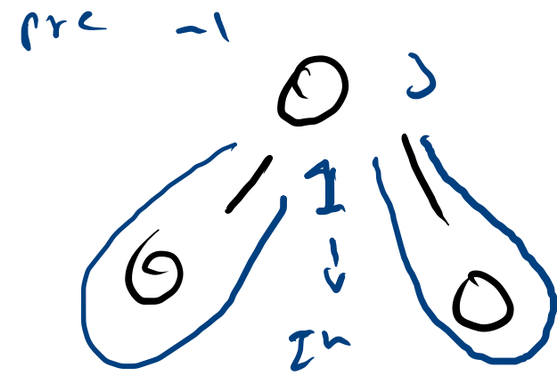
50



Pre	Post	In
50	12	12
25	30	25
12	37	30
37	25	37
30		50
75		

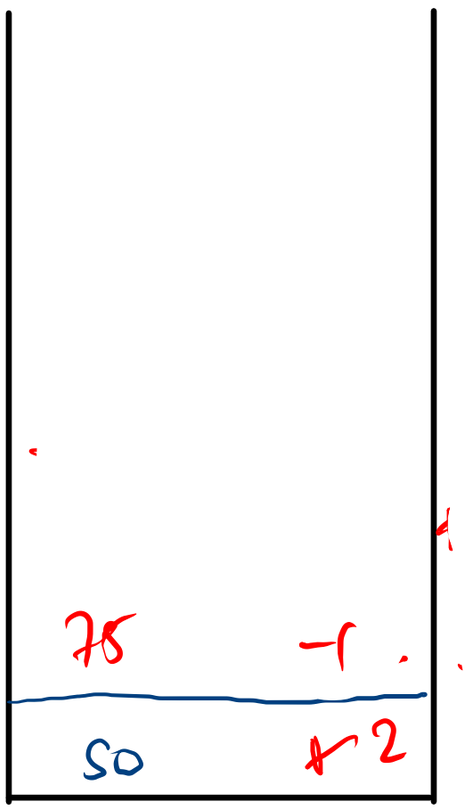


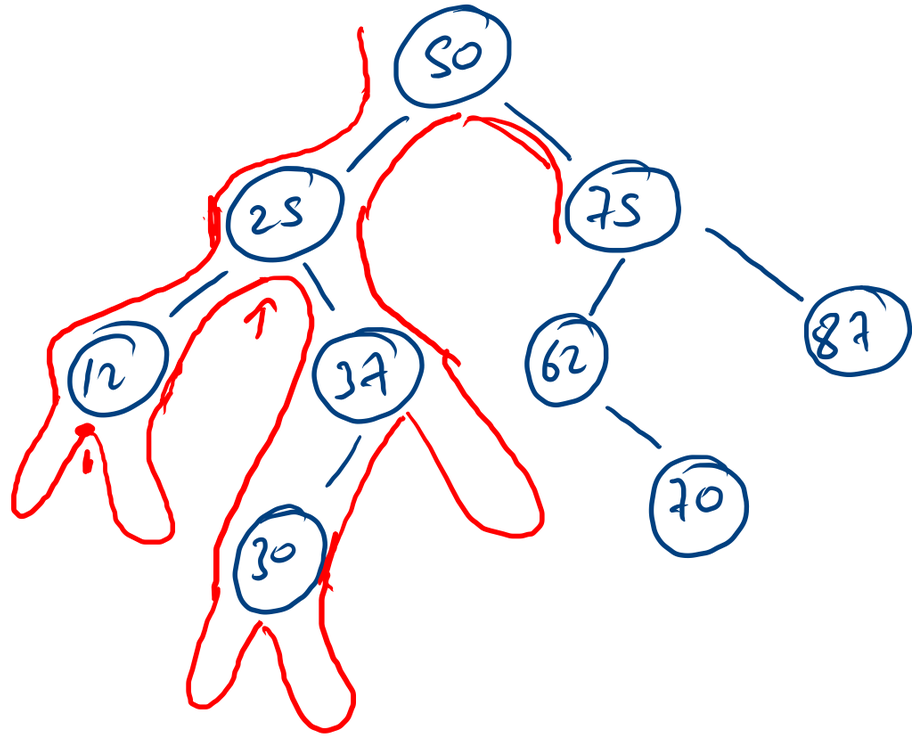




-1	pre
0	left
1	In
2	right
3	pos

```java
while(st.size() > 0){
    Pair2 p = st.peek();

    if(p.status == -1){
        //pre
        pre.append(p.node.data+" ");
        p.status++;
    }else if(p.status == 0){
        // left
        if(p.node.left != null){
            st.push(new Pair2(p.node.left, -1));
        }
        p.status++;
    }else if(p.status == 1){
        // inorder
        in.append(p.node.data+" ");
        p.status++;
    }else if(p.status == 2){
        // right
        if(p.node.right != null){
            st.push(new Pair2(p.node.right, -1));
        }
        p.status++;
    }else{
        // post
        post.append(p.node.data+" ");
        st.pop();
    }
}
```





Pre	50	25	12	37	30
in	12	25	30	37	50
post	12	30	37	25	

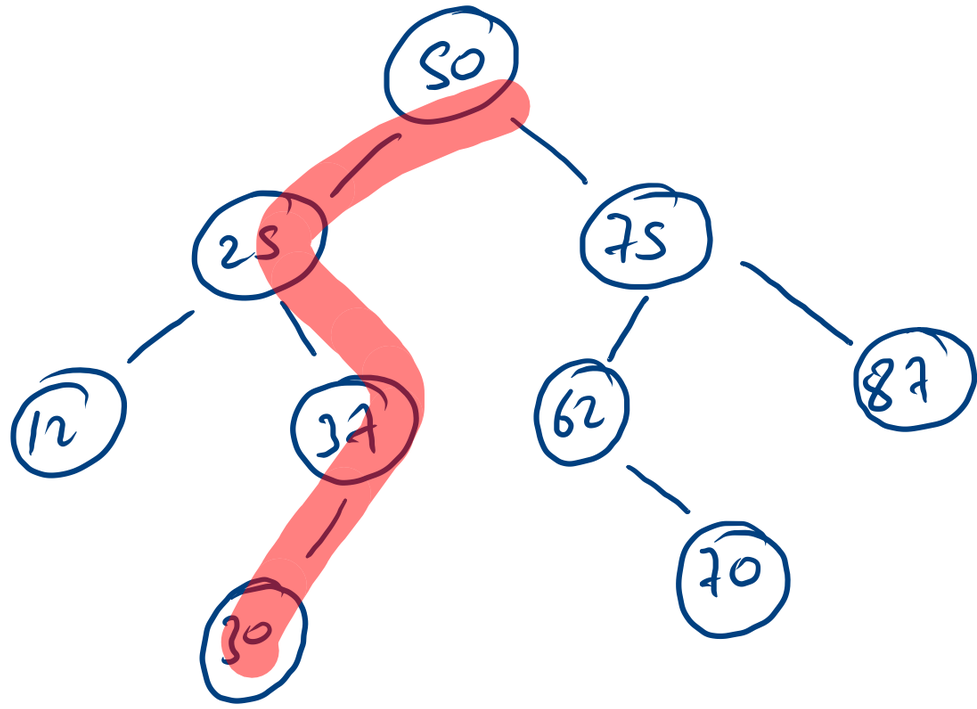
N.W.

Find And Nodetorootpath In Binary Tree

data → 30

find → true

nlist → [30, 37, 25, 50]



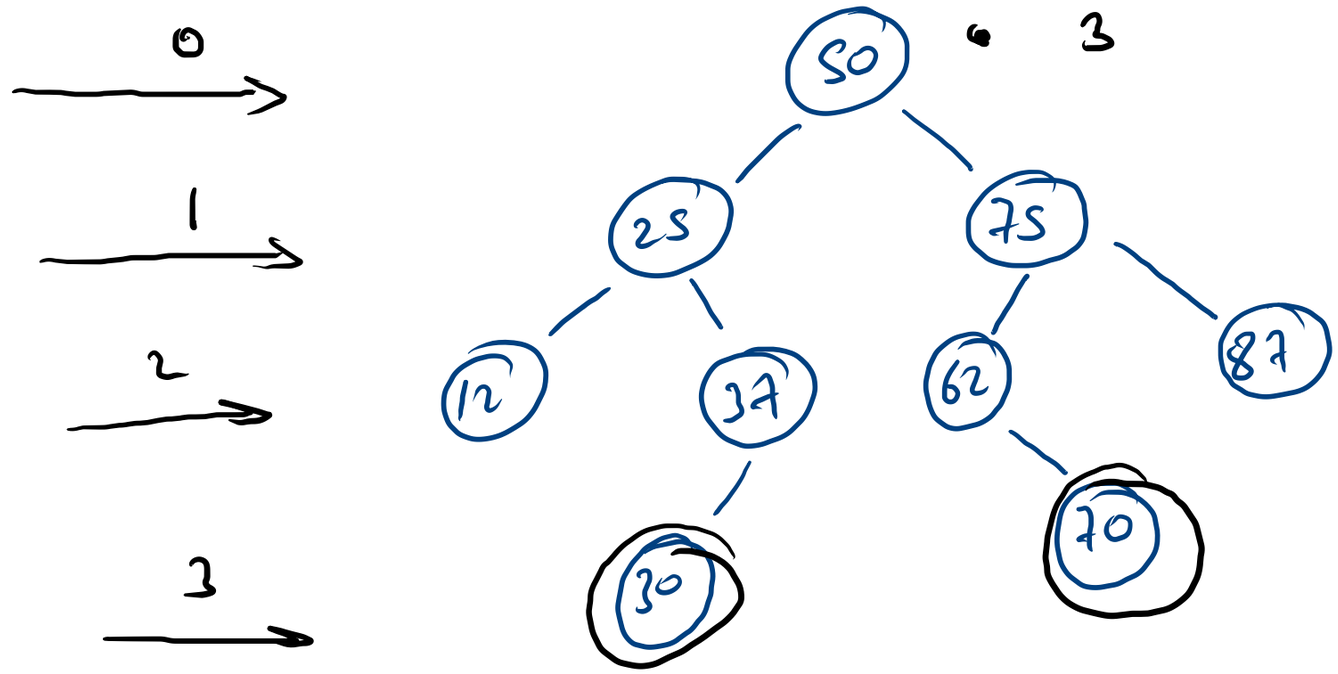
data 91

find → false

nlist → [ ]

level → 3

30
70



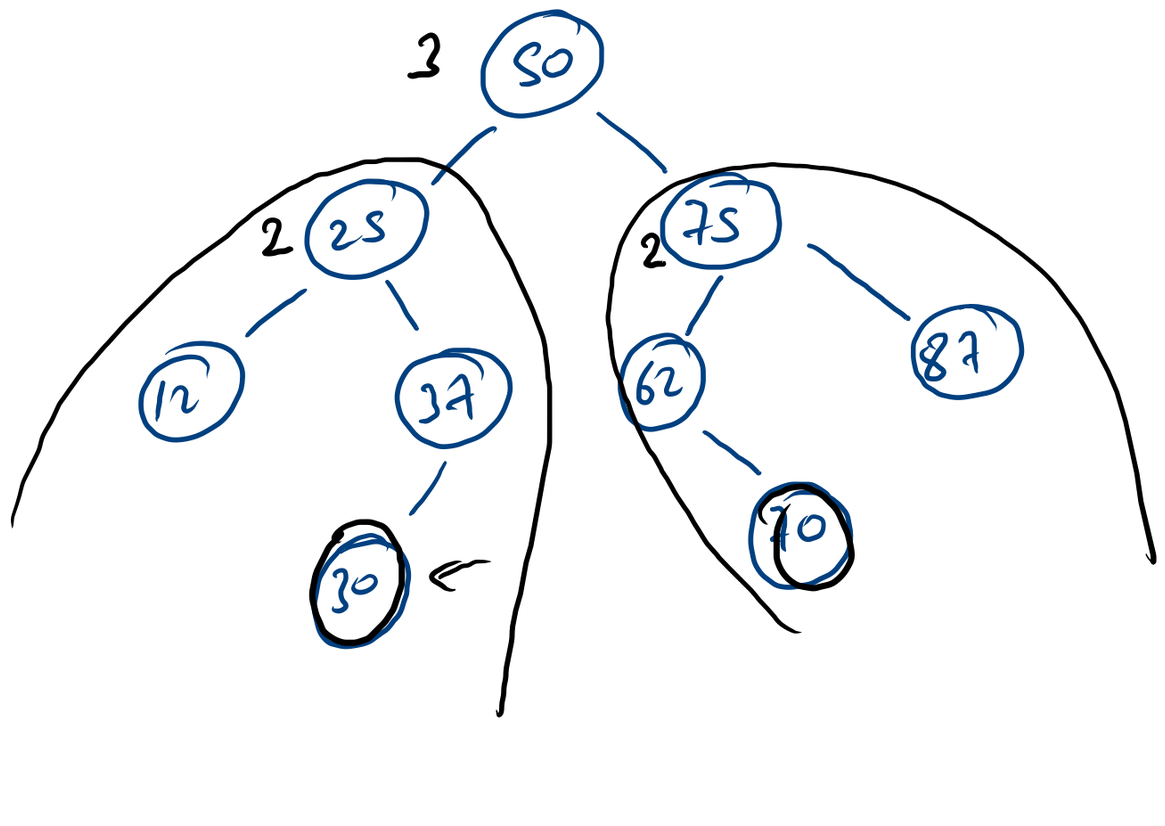
level = 2

12
37
62
87

$$-\infty \leq level \leq +\infty$$

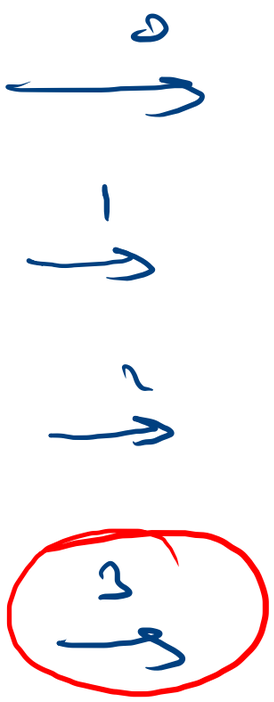
k

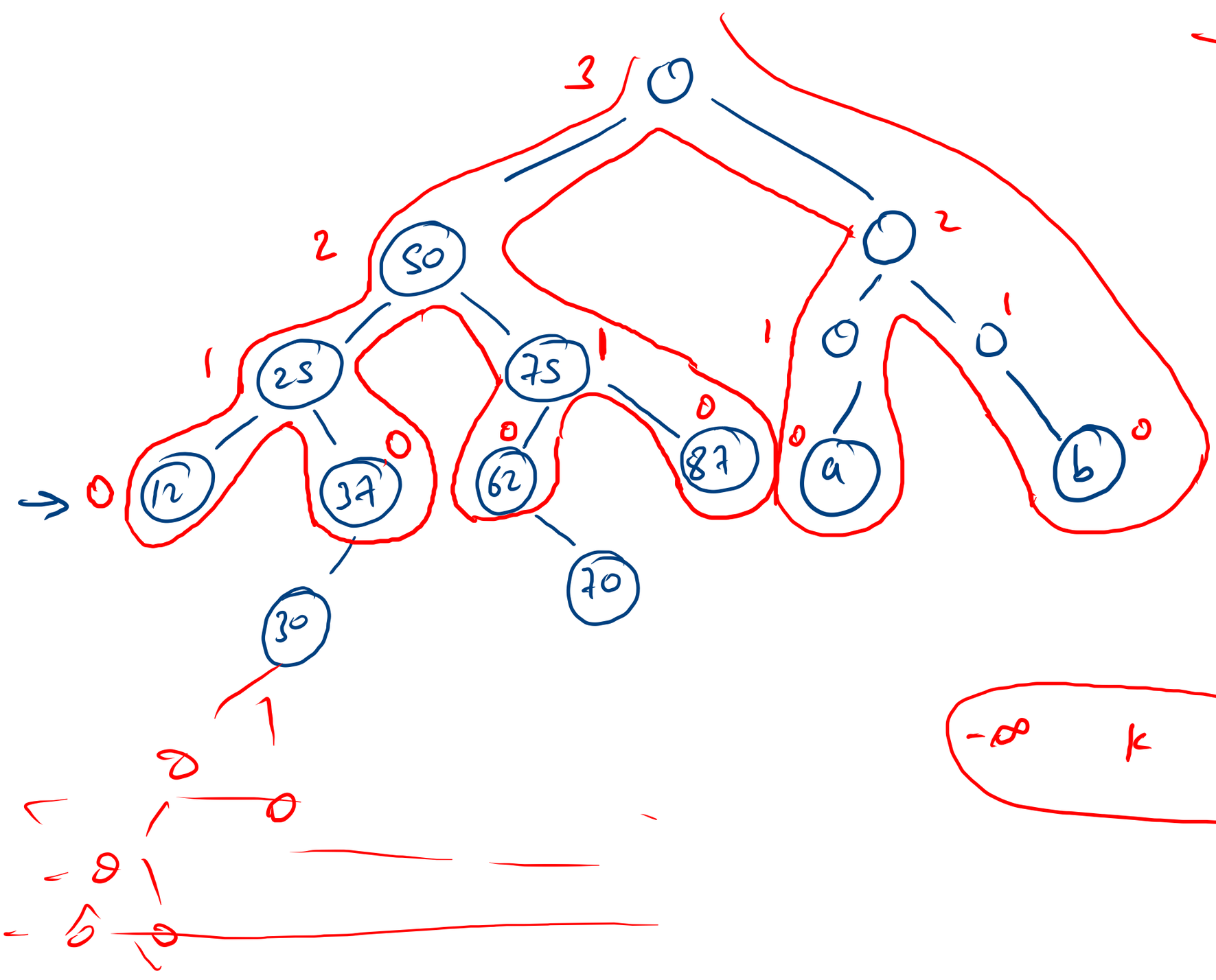
level → 3



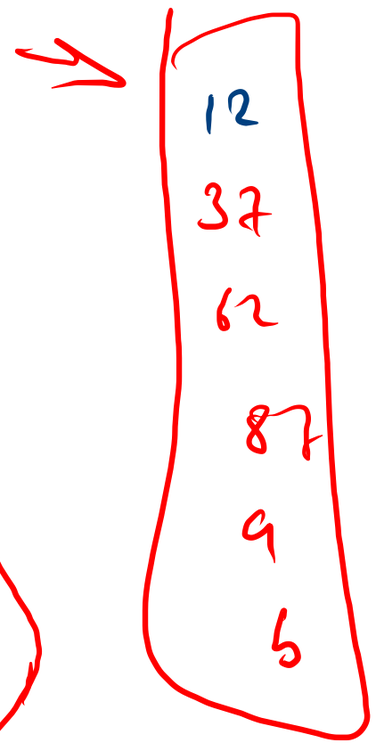
print (left, k-1)

print (right, k-1)




3
2
1
0
50
25
75
12
37
62
87
30
70
a
b

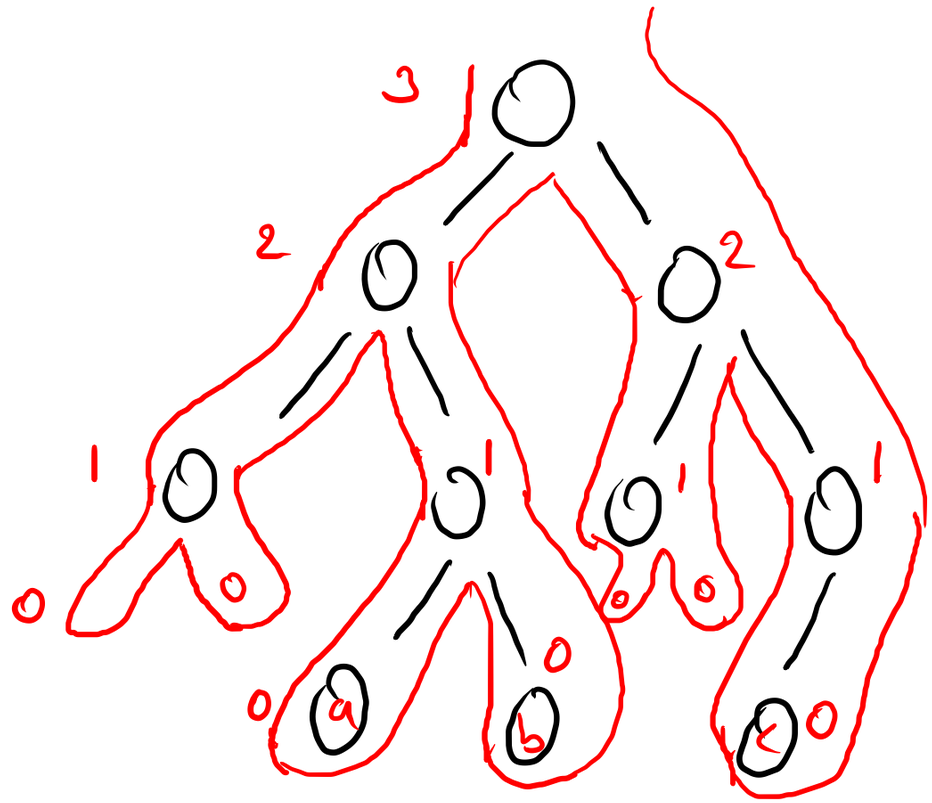
lc = 10
10

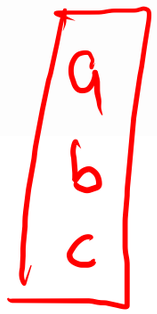

12
37
62
87
a
b

-∞
k
+∞

k=3

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

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

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



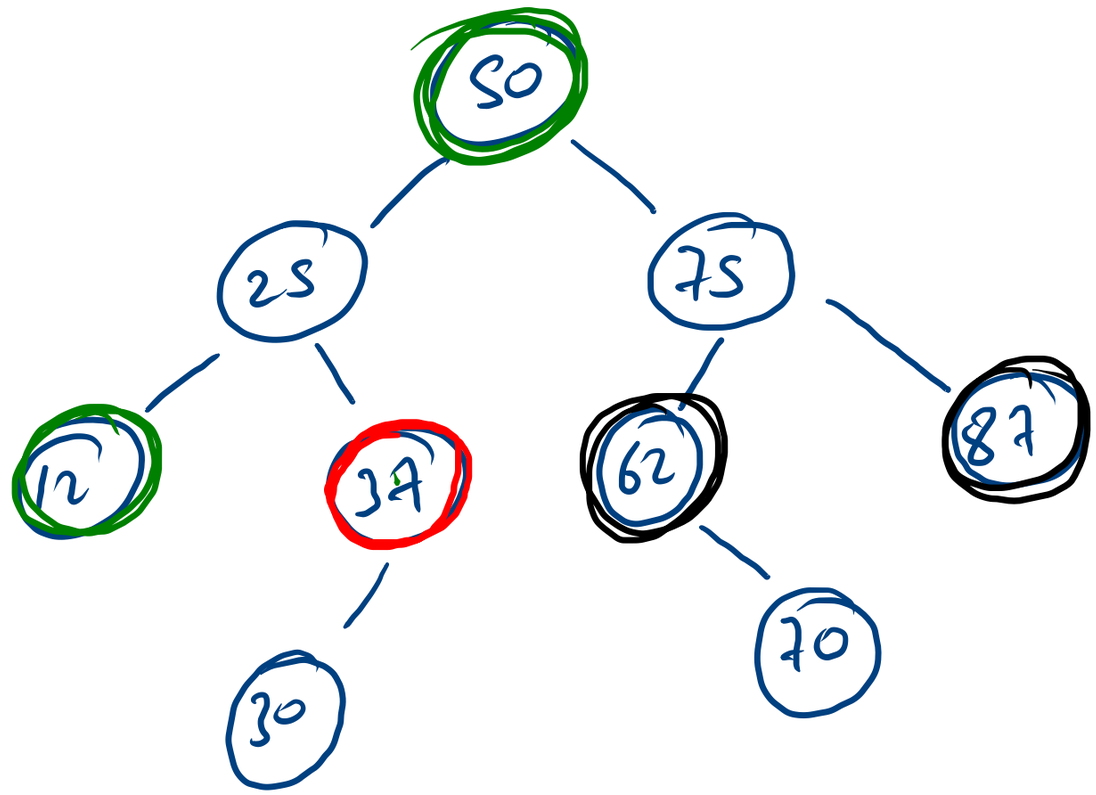


data → 37

k → 2

k distance

12
50

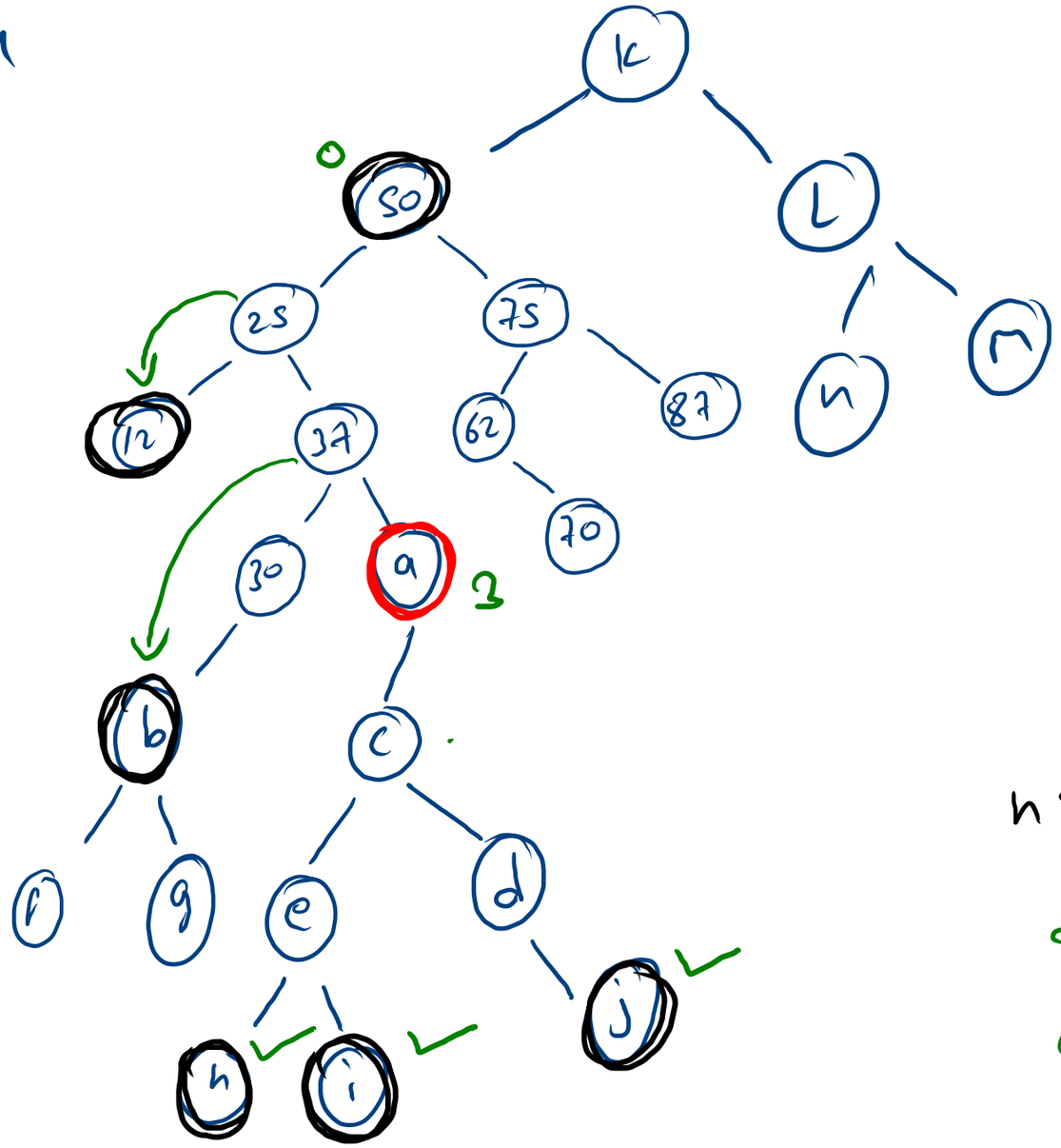


k = 4

62
87

data → a

k = 3

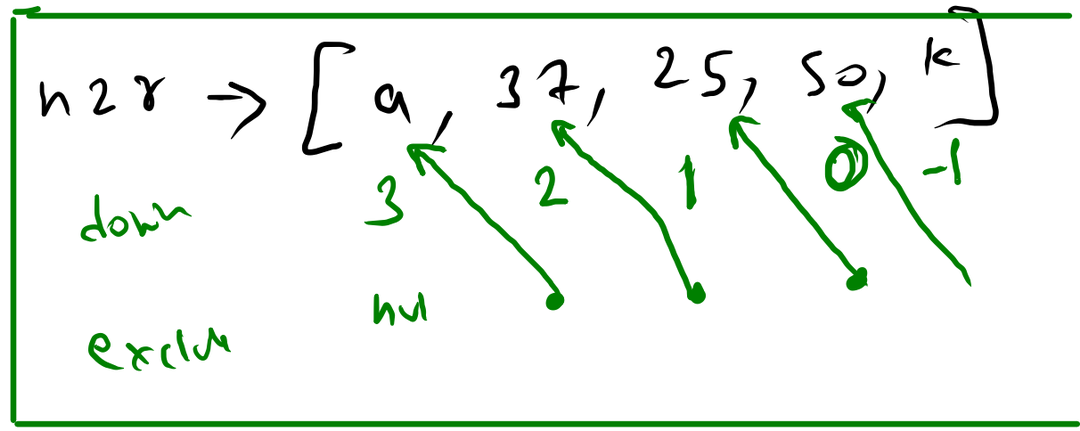


-1 k

n2r → [a, 37, 25, 50, k]

down

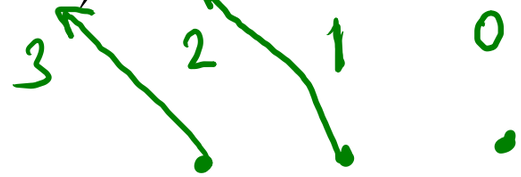
exclu



n2r → [a, 37, 25, 50, k]

down

exclu



```java
public static void printKNodesFar(Node node, int data, int k) {
    ArrayList<Integer> al = nodeToRootPath(node, data);

    int down = k;
    Node exc = null;

    for(int i=0;i<al.size() ;i++){

        Node curr = al.get(i);
        printKLevelsDown(curr, down, exc);

        down--;
        exc = curr;
    }
}
```

k = 3

	0	1	2	3	4
al	9	37	25	50	k
down	3	2	1	0	-1
exc	null	9	37	25	50