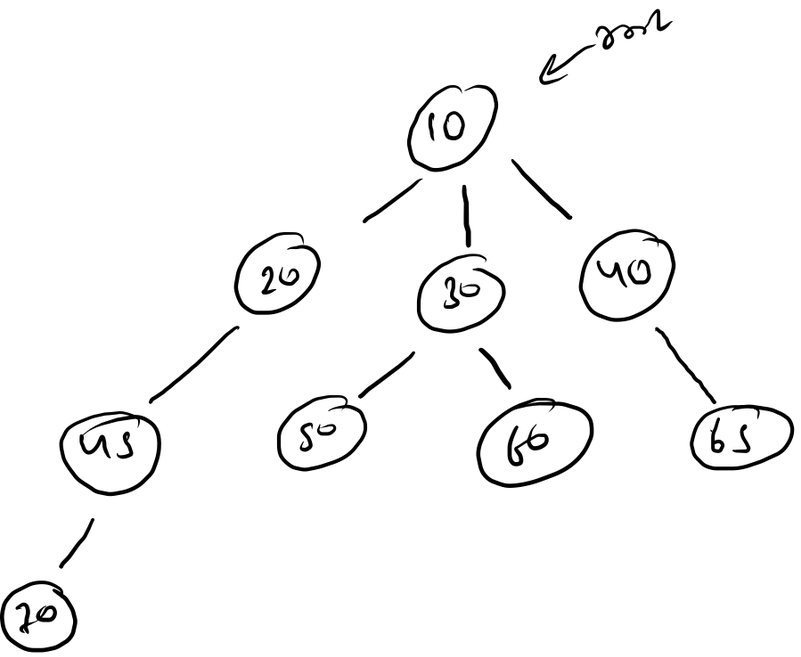
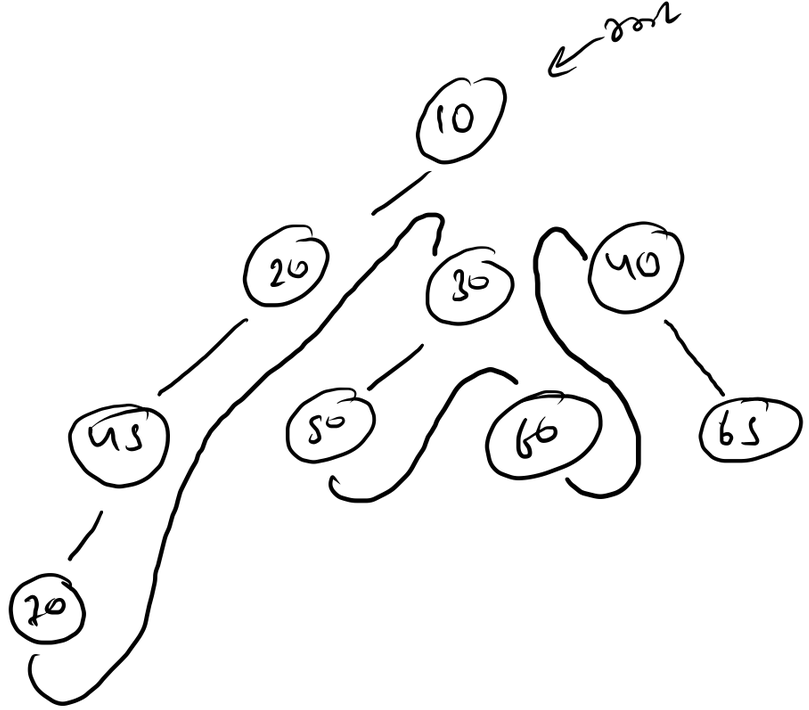
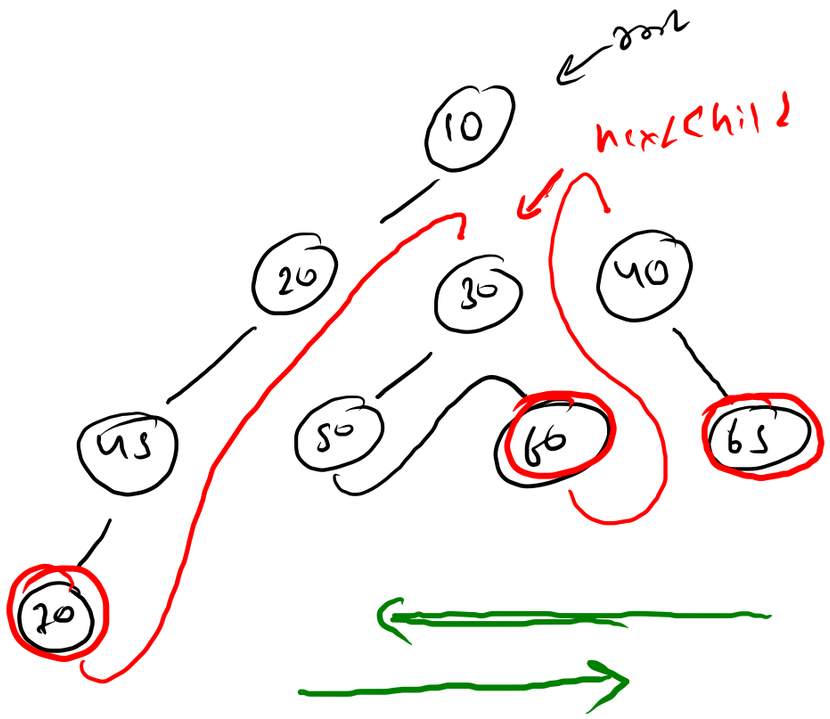




child

2 → Linear

· → leaf

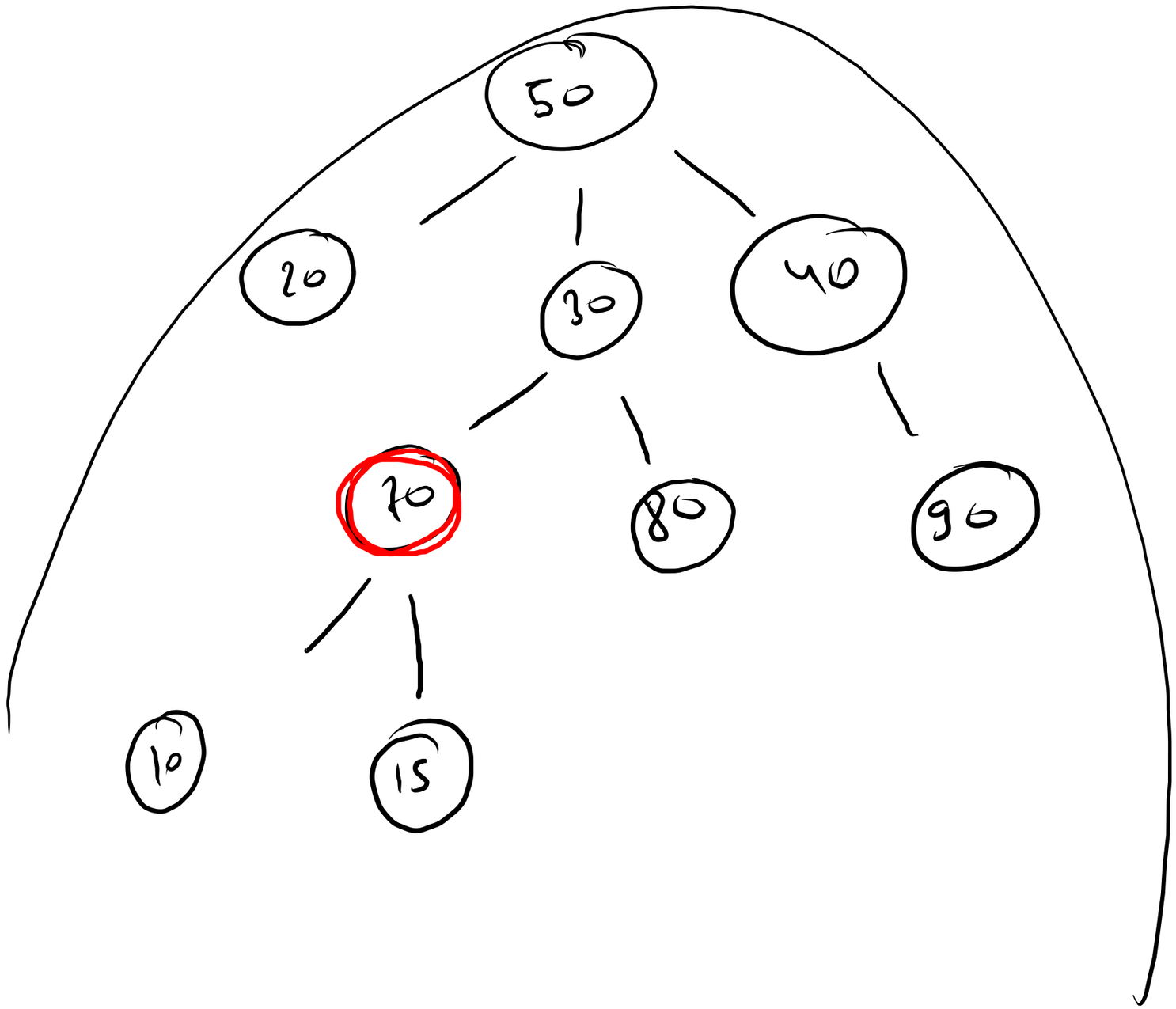


d = 70 ↙ → true

d = 75 → False

child fait

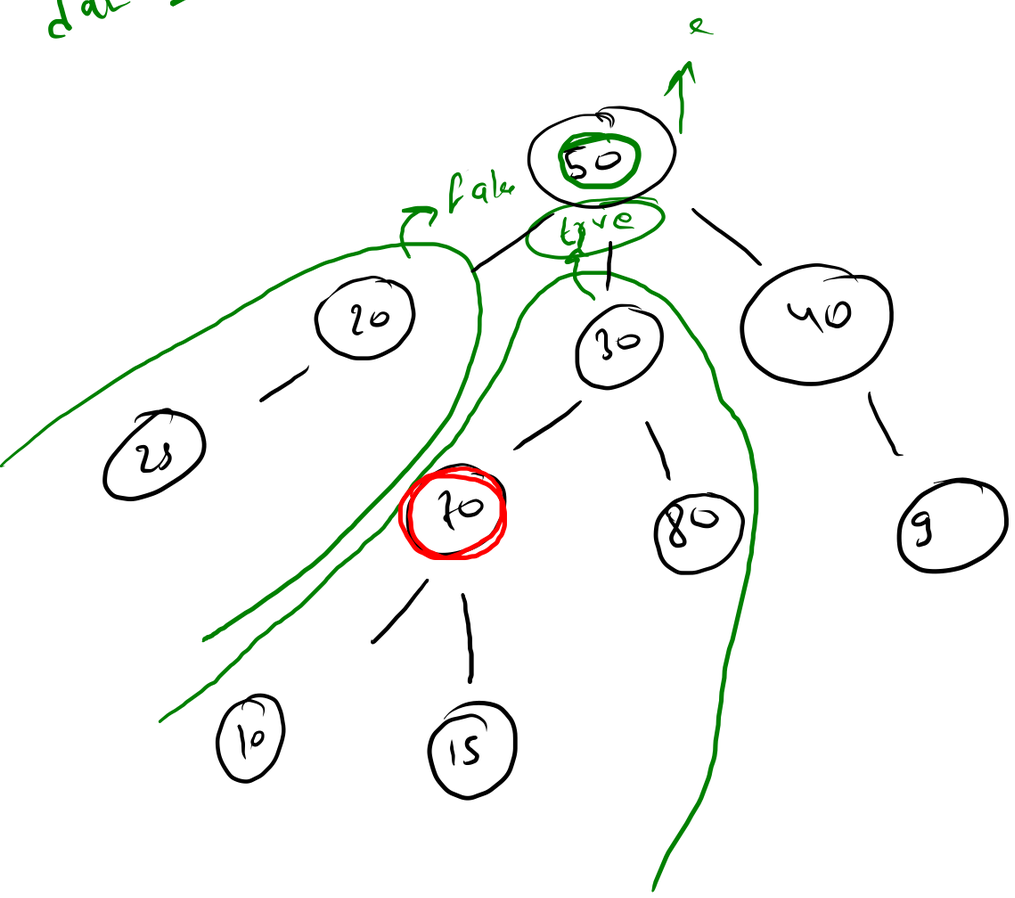
self check

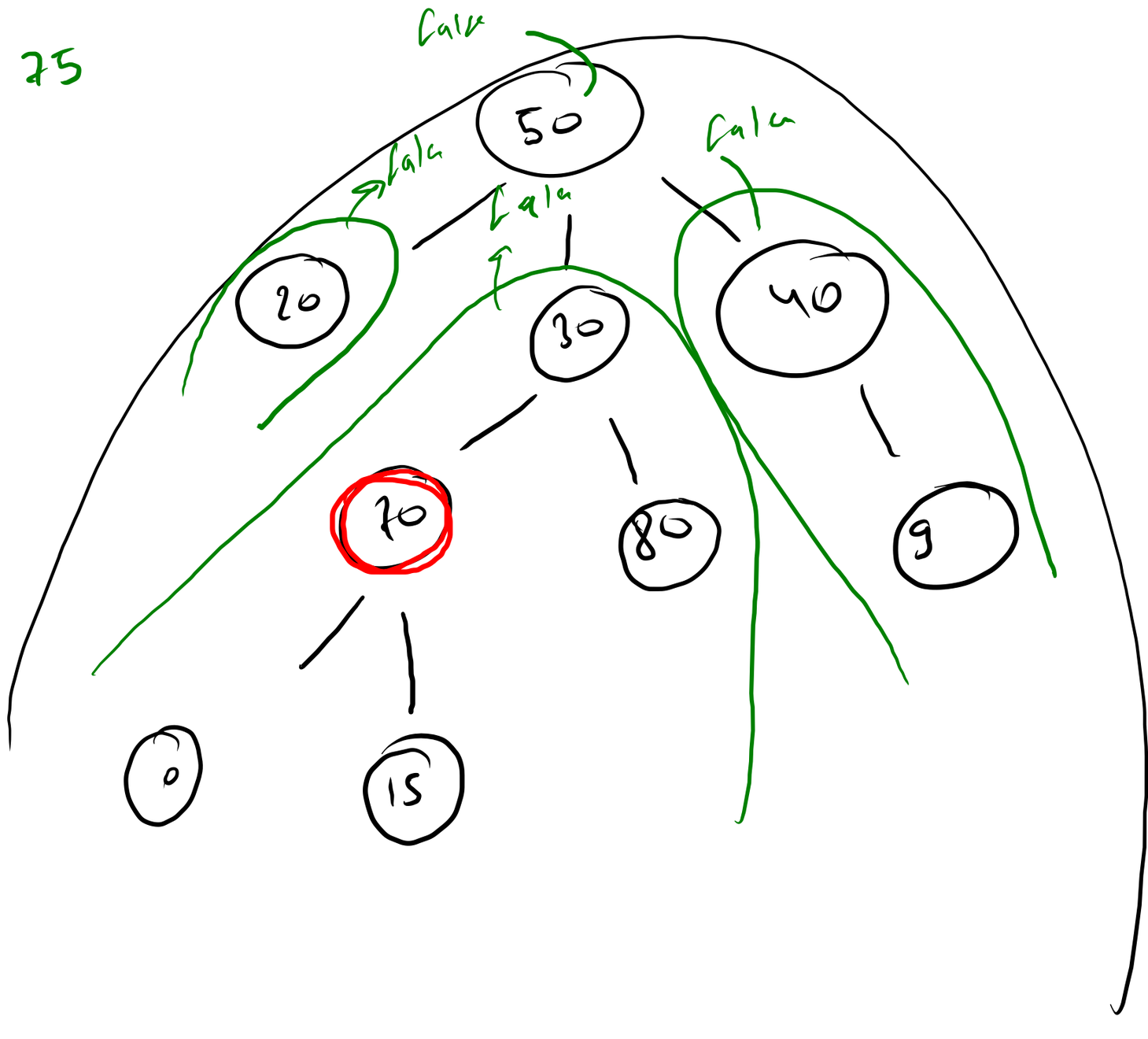


data -> 75

root -> 1 in 1

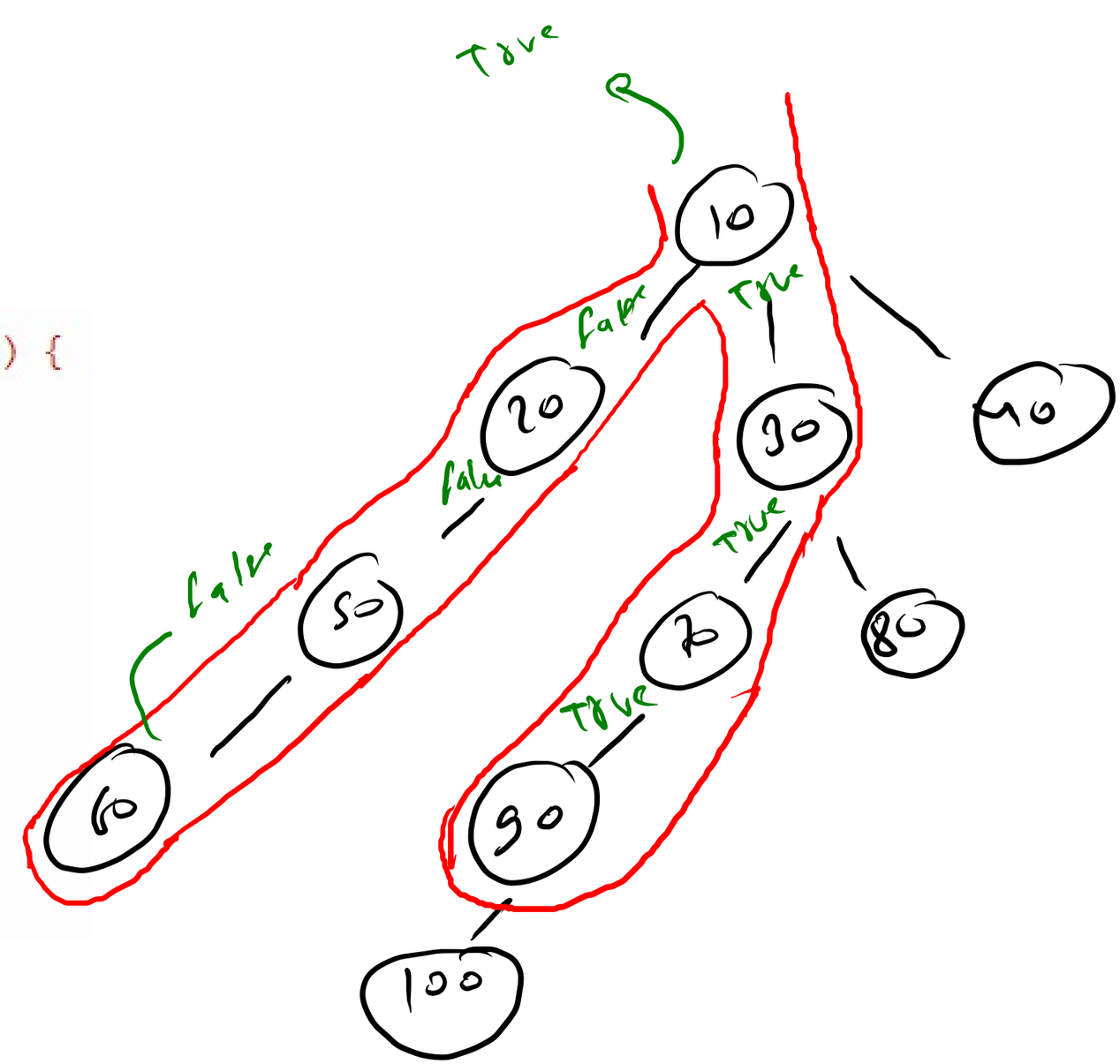
data -> 70





data = 90

```java
public static boolean find(Node node, int data) {
    if(node.data == data){
        return true;
    }
    for(Node child: node.children){
        boolean found = find(child, data);
        if(found){
            return true;
        }
    }
    return false;
}
```



[120, 80, 30, 10]

node

root

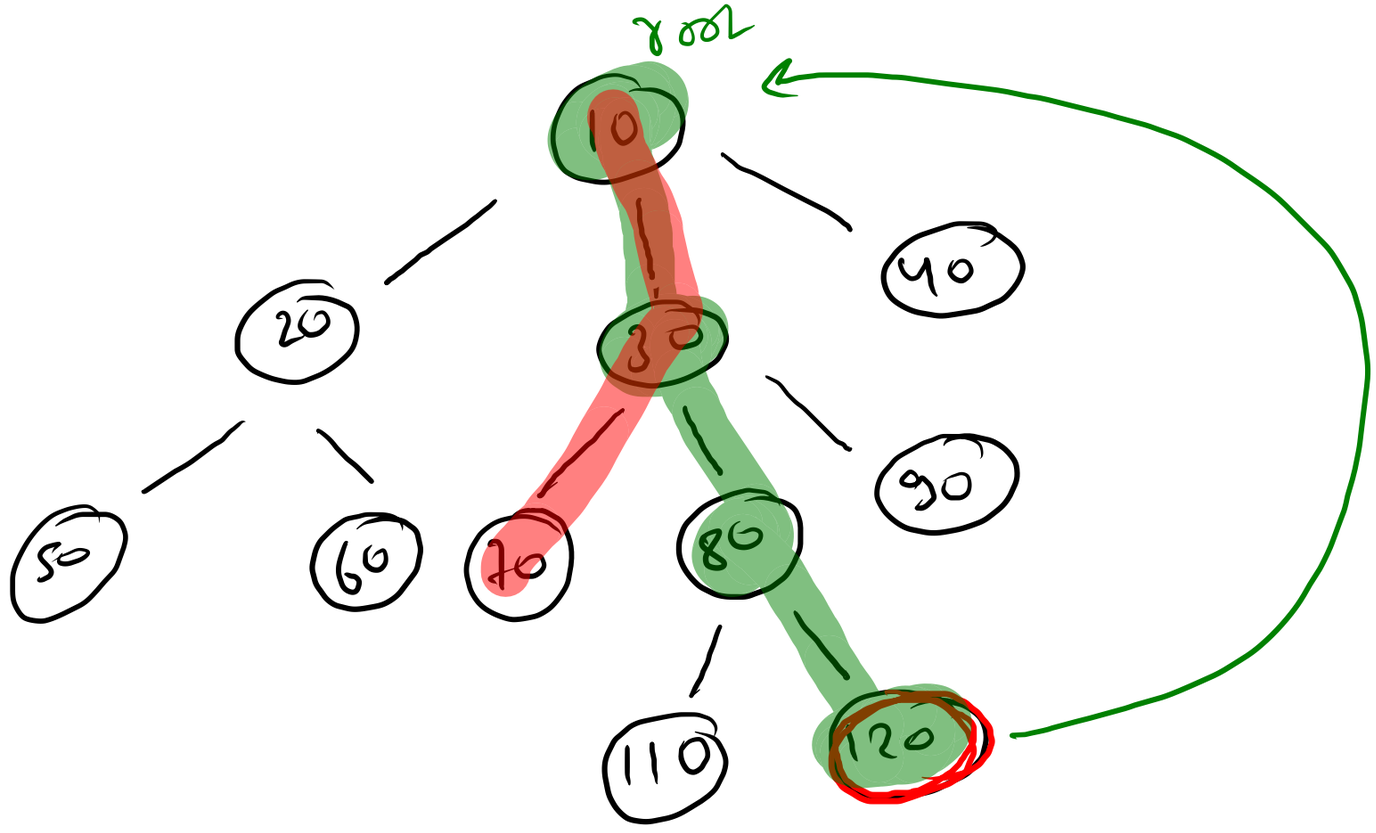
d = 120

d = 70

[70 30 10]

node

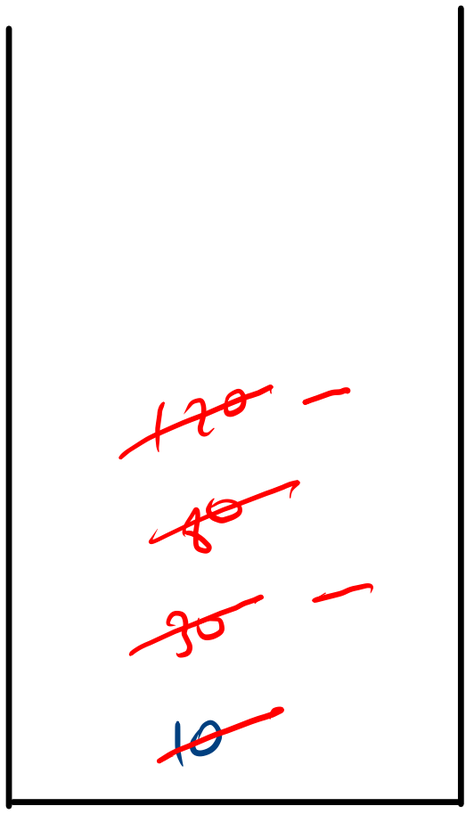
root

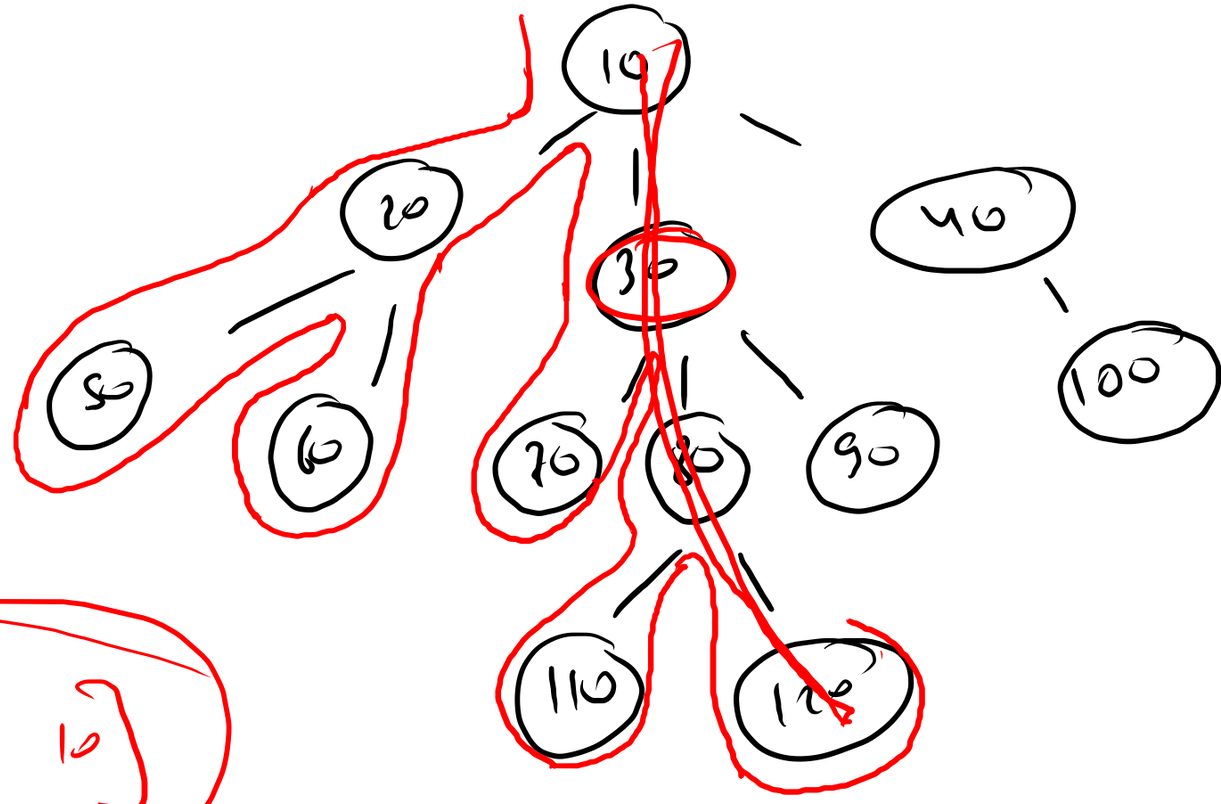


d = 95

[]

[120, 80, 30, 10]





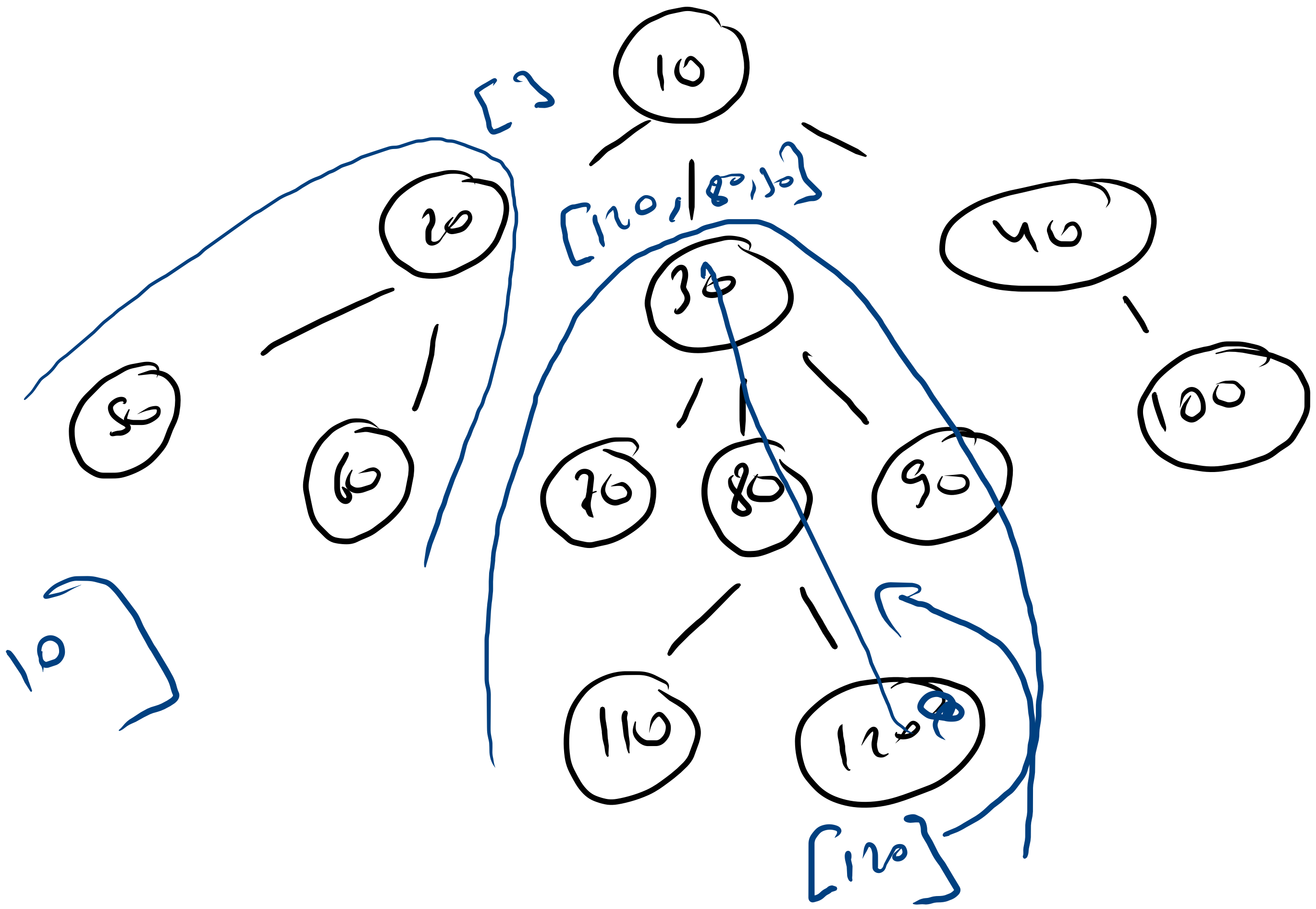
[120 80 30 10]

24
10 20 50 -1 60 -1 -1 30 70 -1 80 110 -1 120 -1 -1 90 -1 -1 40 100 -1 -1 -1
120

find → node to root

x → [ ]

120



[120, 80, 30, 10]

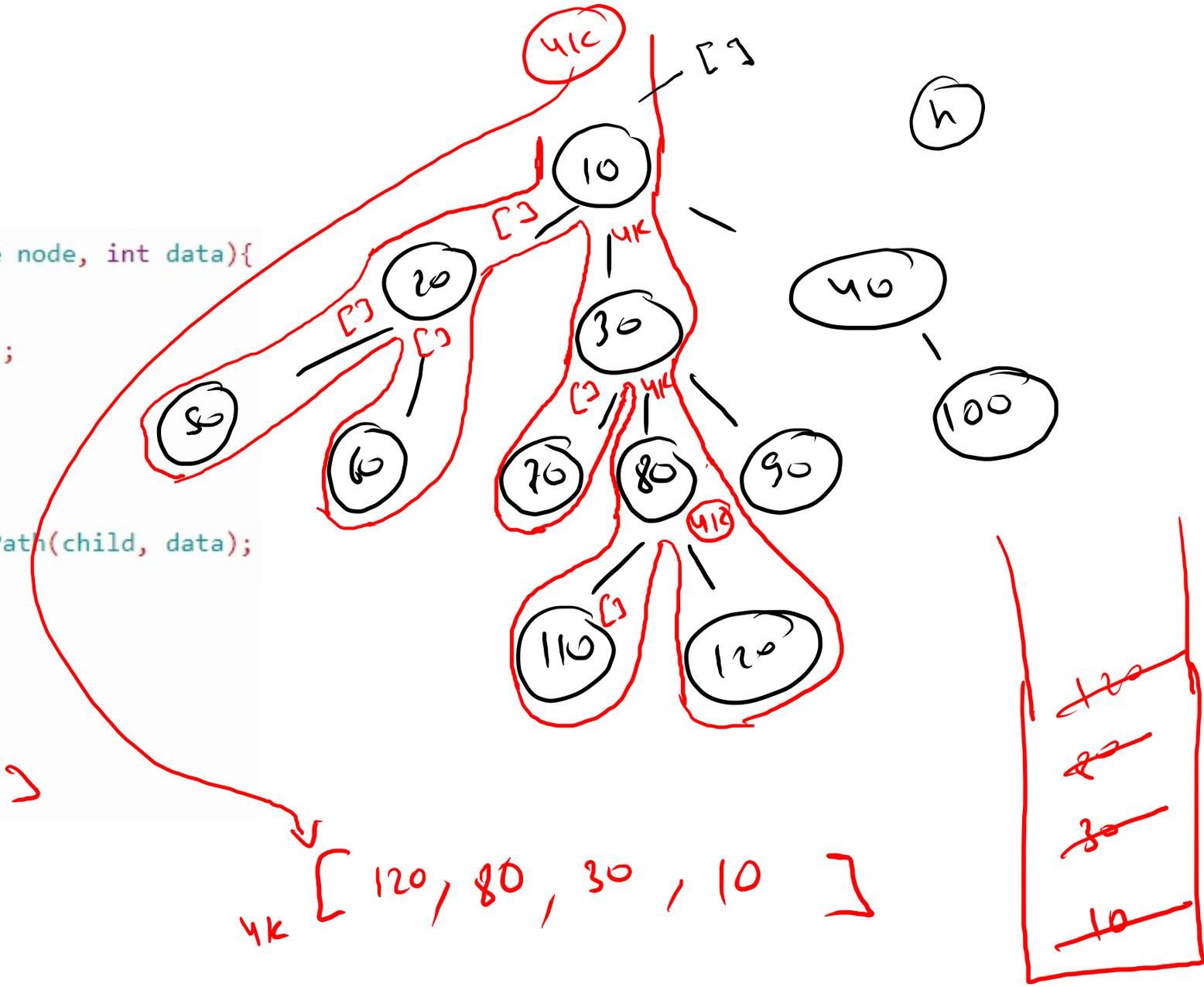
[120, 80, 30, 10]
false

n.data == data {
[self.data]
}

child false
al.size > 0
al.add(n.data)
return al;

ret [ ]

```java
public static ArrayList<Integer> nodeToRootPath(Node node, int data){

    if(node.data == data){
        ArrayList<Integer> al = new ArrayList<>();
        al.add(data);
        return al;
    }

    for(Node child: node.children){
        ArrayList<Integer> childAns = nodeToRootPath(child, data);
        if(childAns.size()>0){
            childAns.add(node.data);
            return childAns;
        }
    }
    ArrayList<Integer> al = new ArrayList<>();
    return al;
}
```

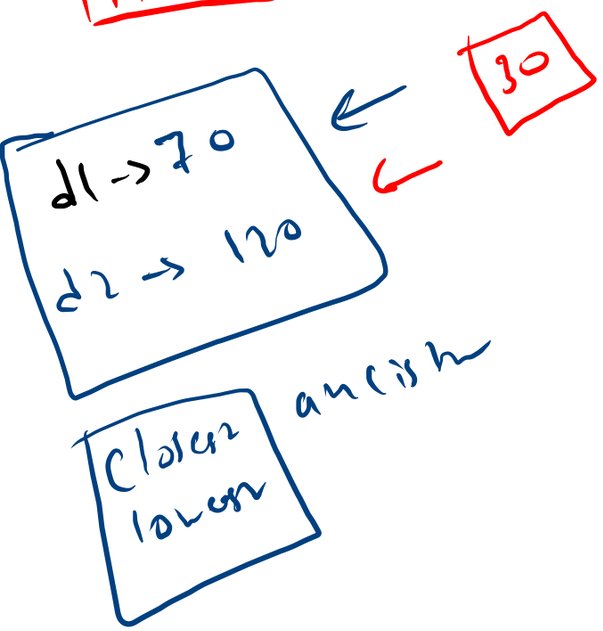


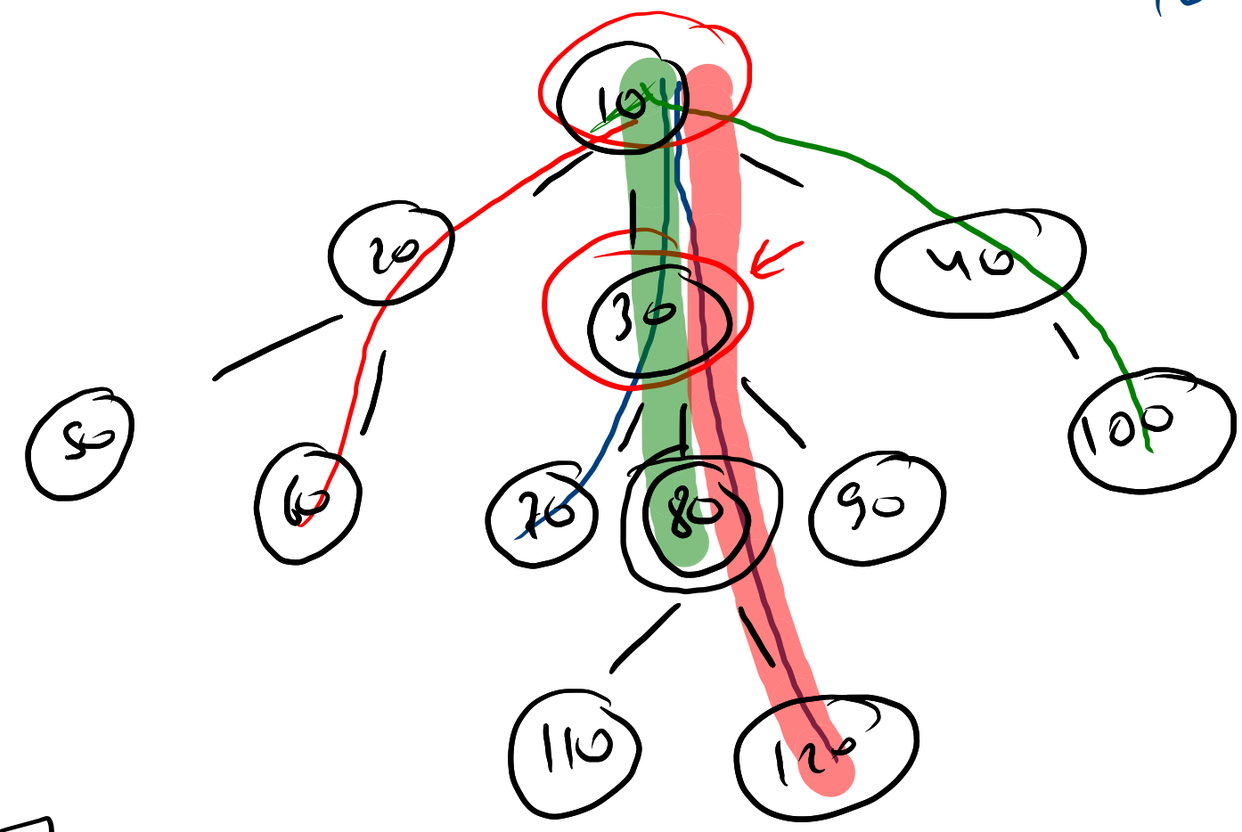
Lowest Common Ancestor (generic Tree)

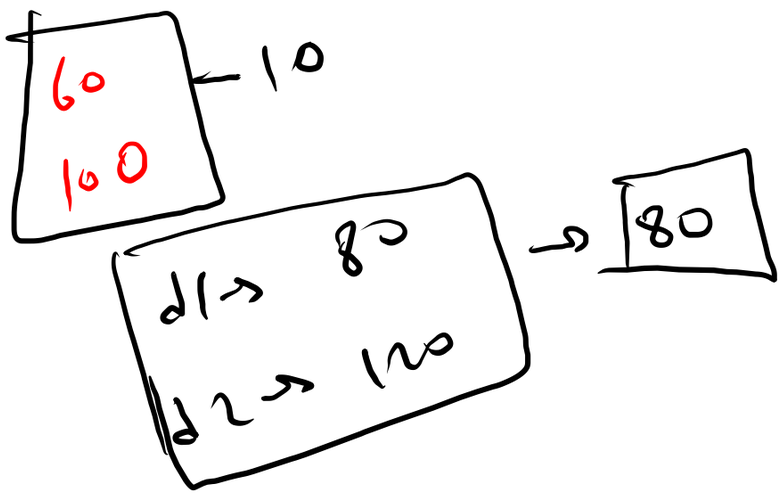
H.W

10 30

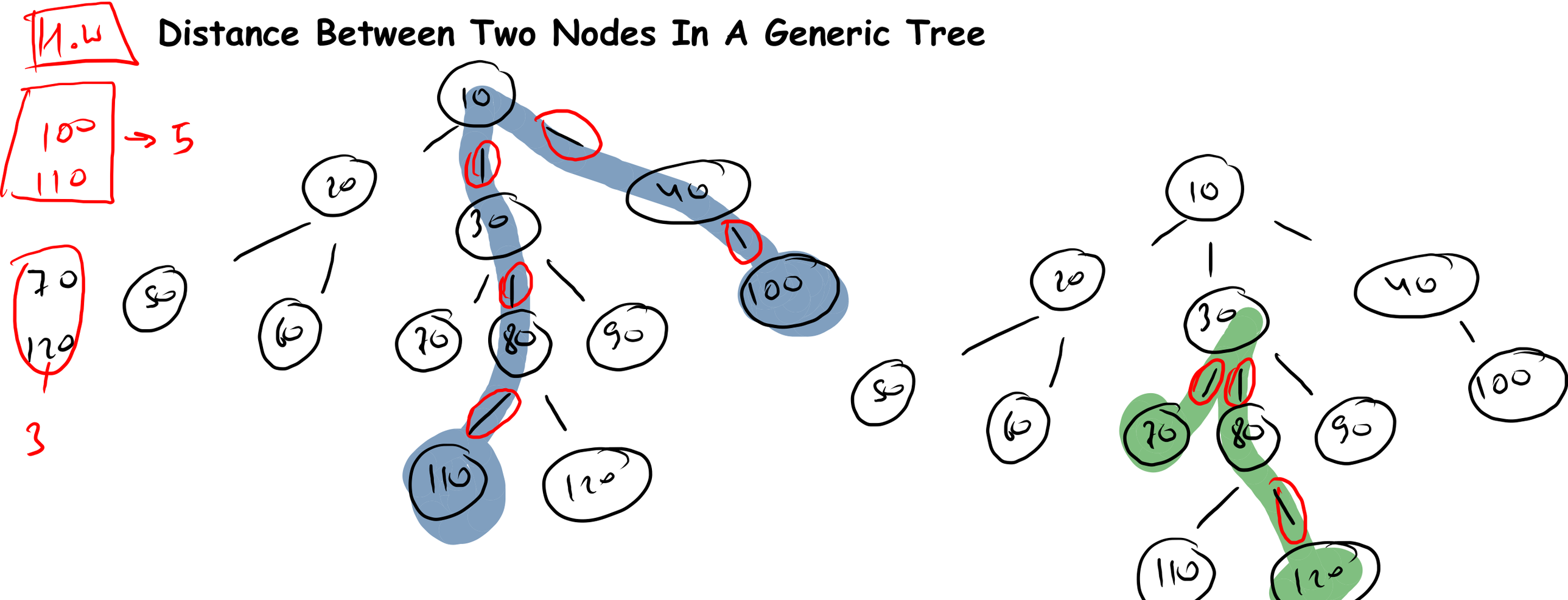
[80, 30, 10]



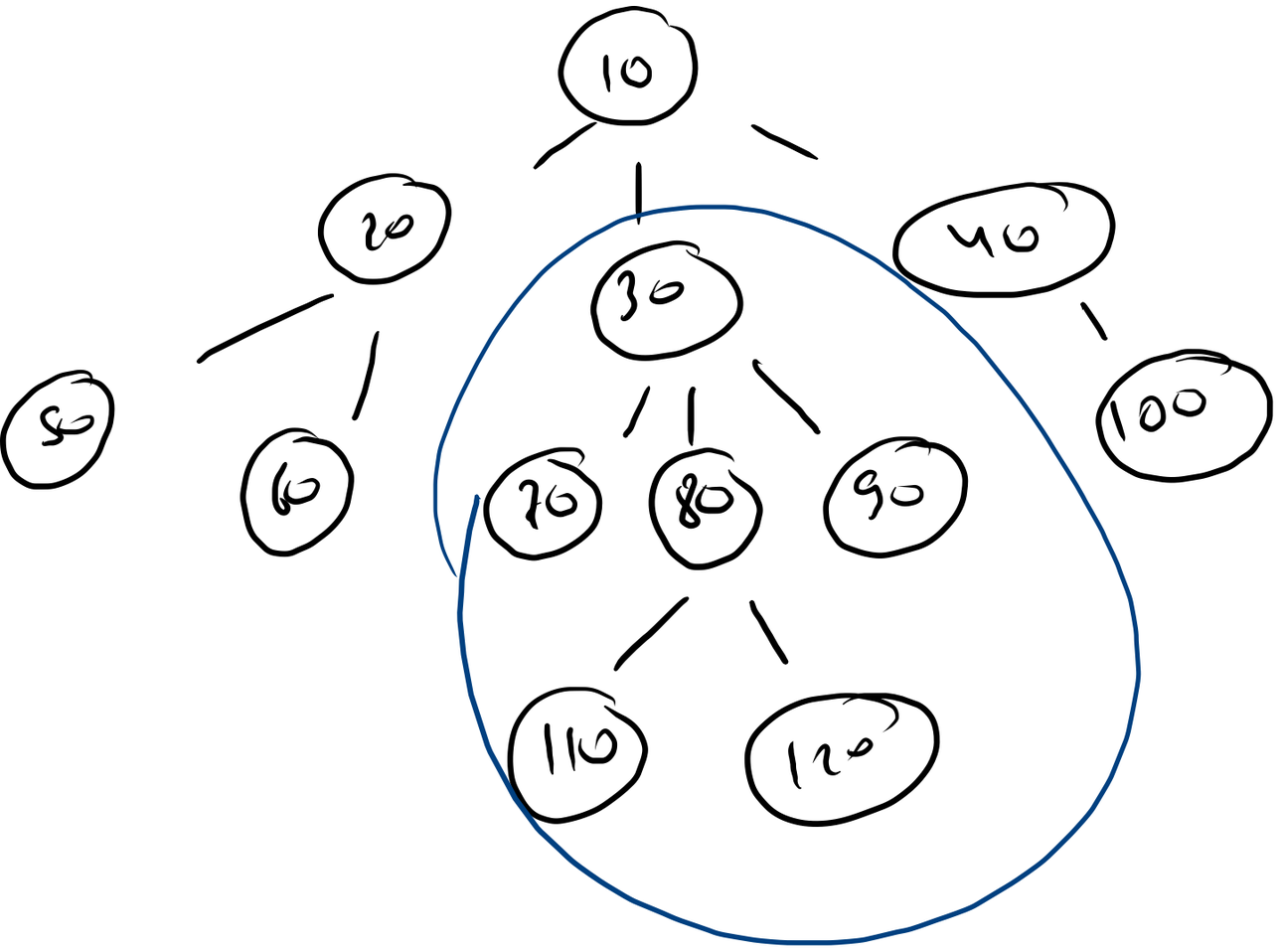


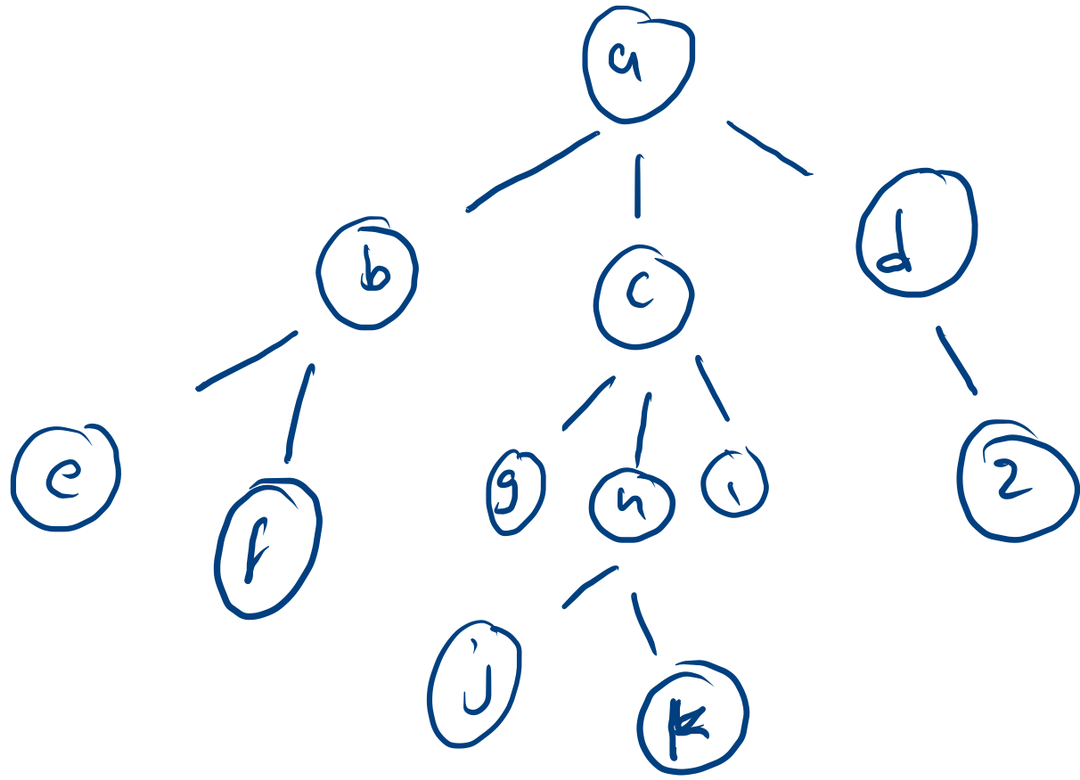


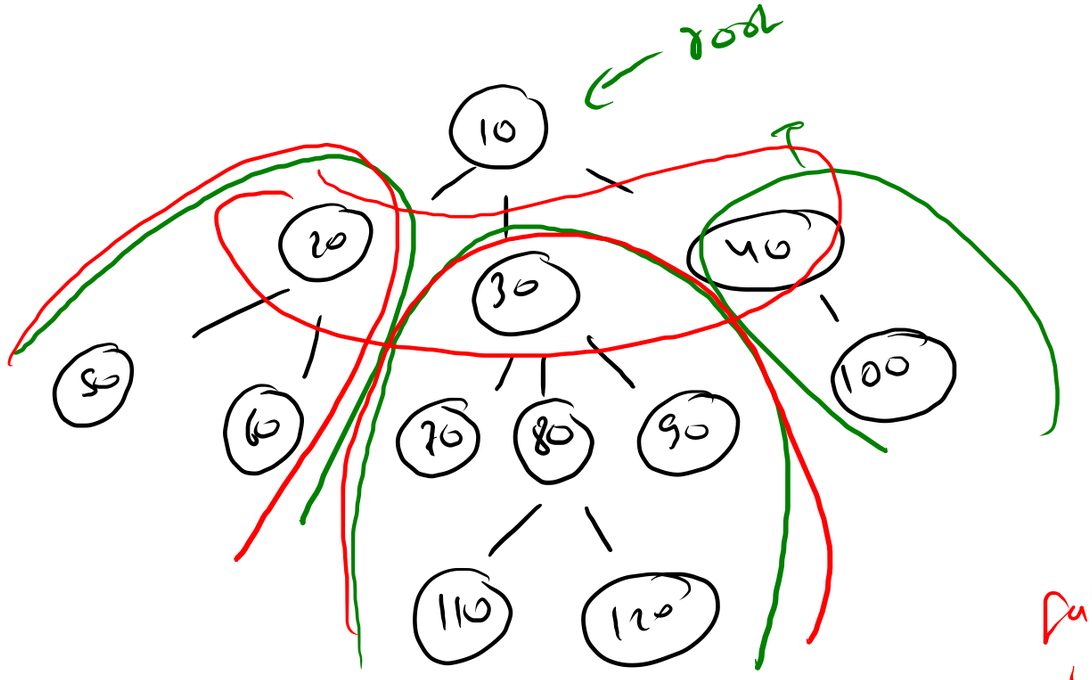
11.4 Distance Between Two Nodes In A Generic Tree

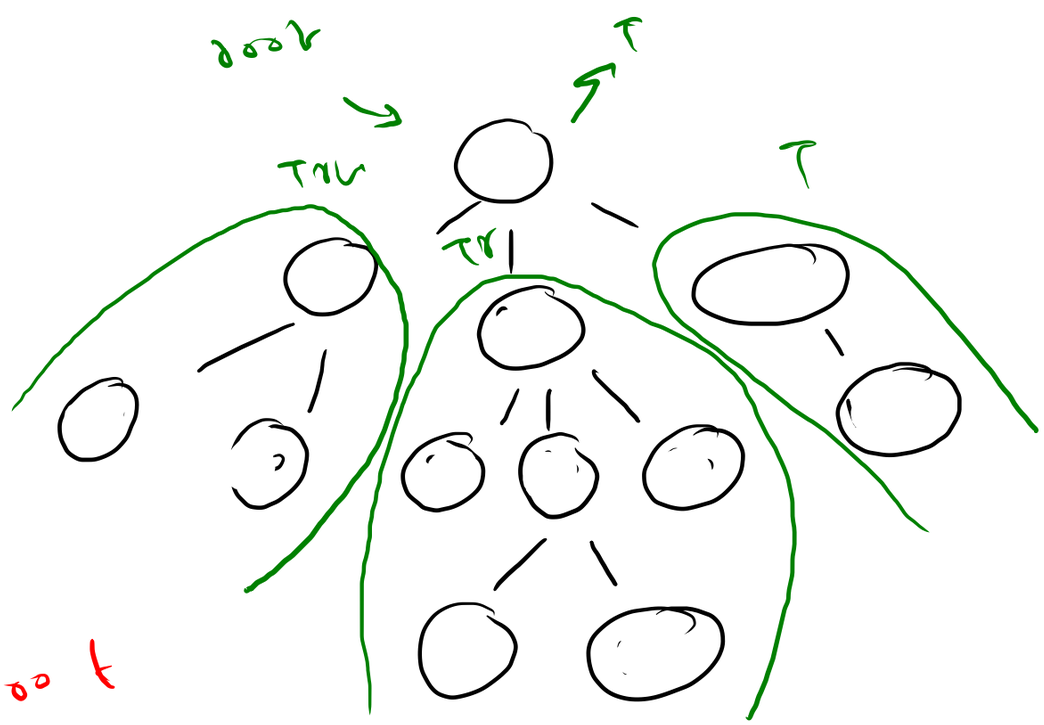



```
24
10 20 50 -1 60 -1 -1 30 70 -1 80 110 -1 120 -1 -1 90 -1 -1 40 100 -1 -1 -1
100
110
```

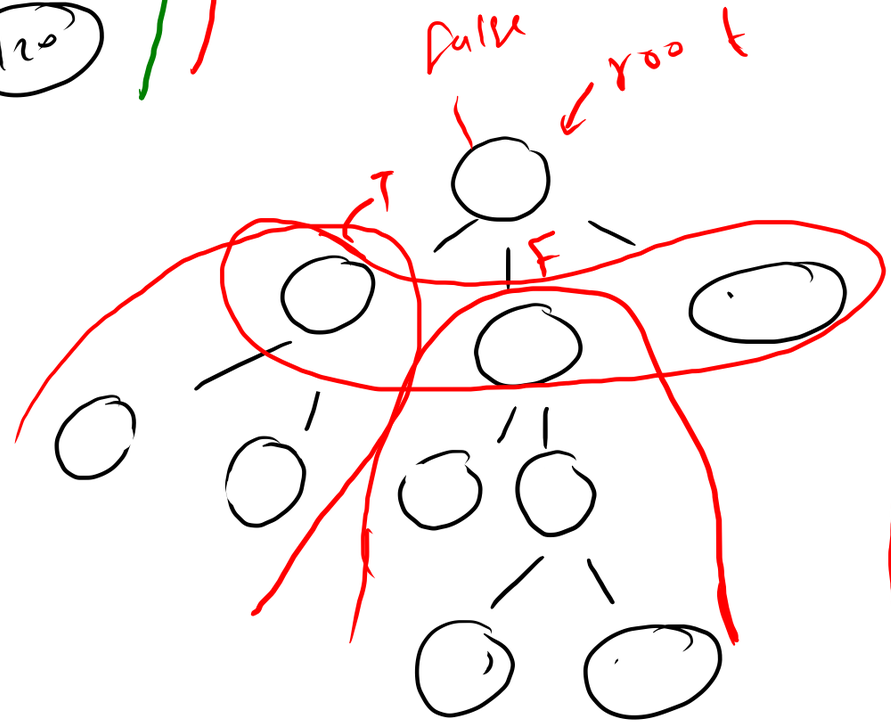

10
20
30
40
5
6
70
80
90
100
110
120

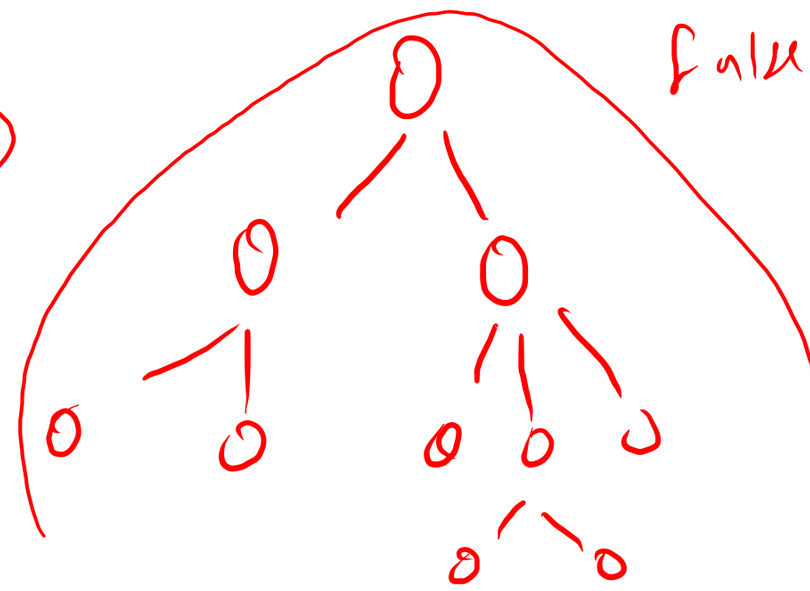

a
b
c
d
e
f
g
h
i
2
j
k

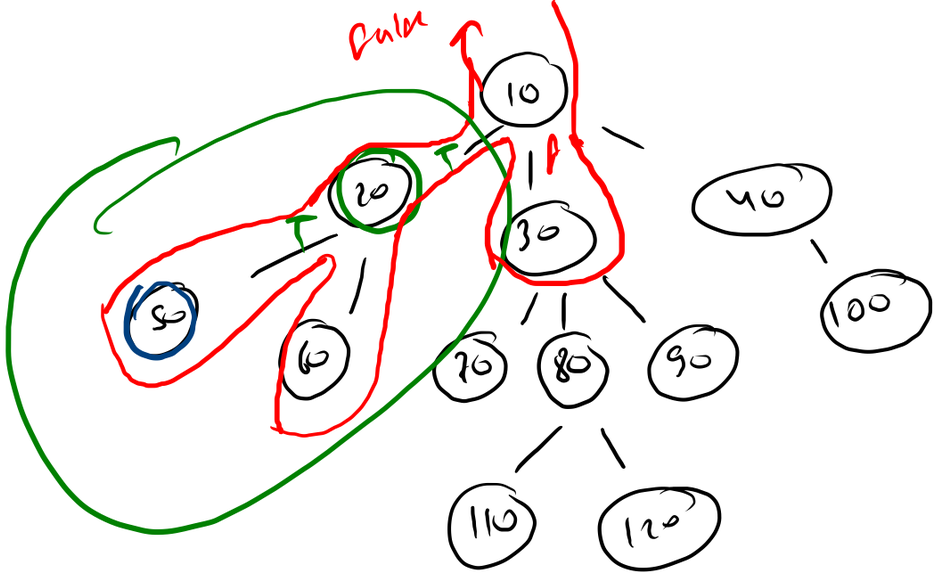


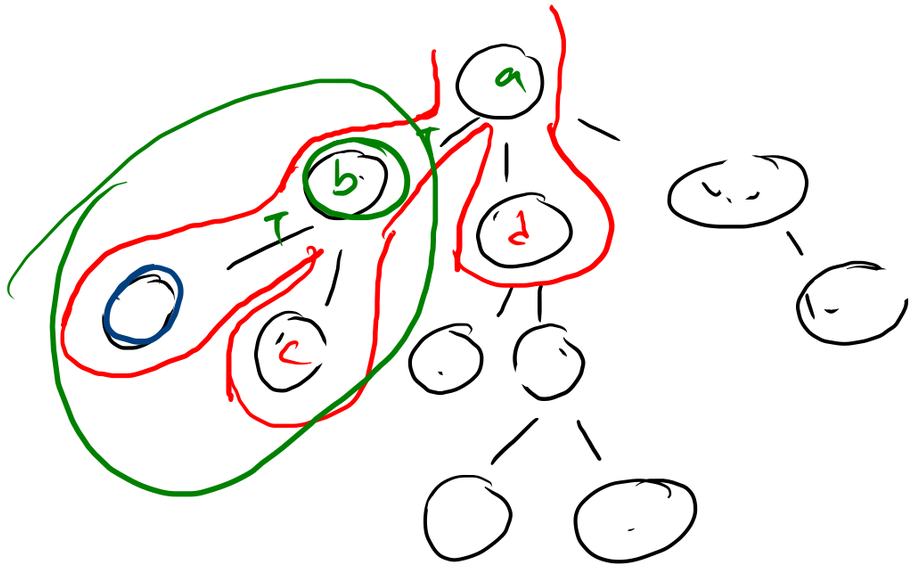


similar(root1, root2)



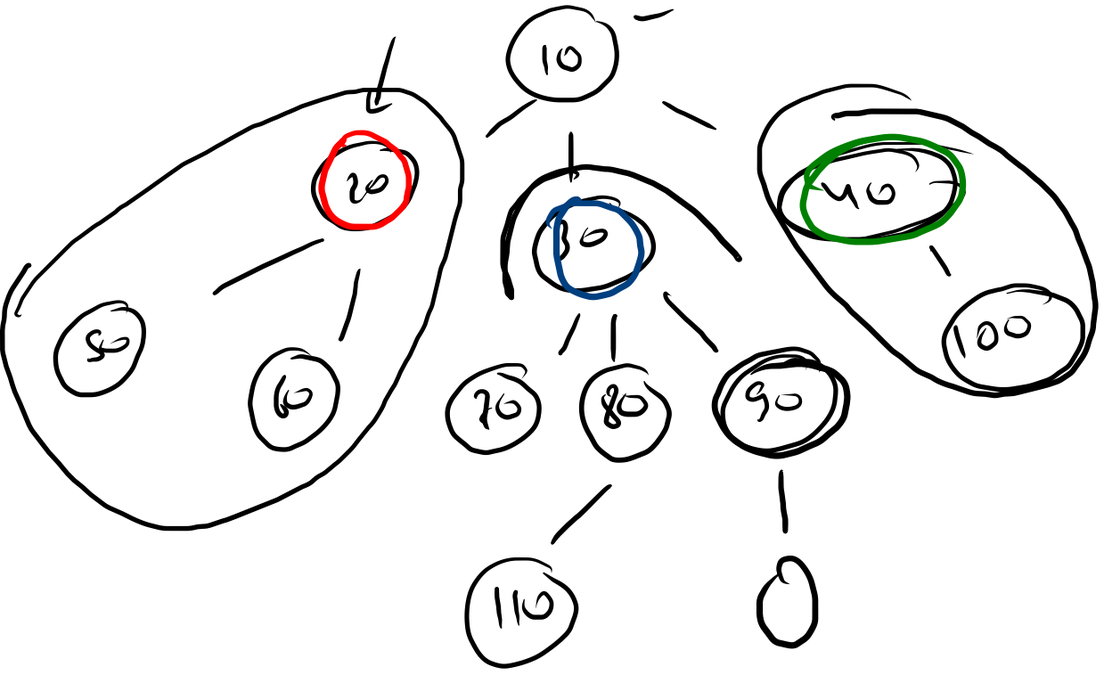


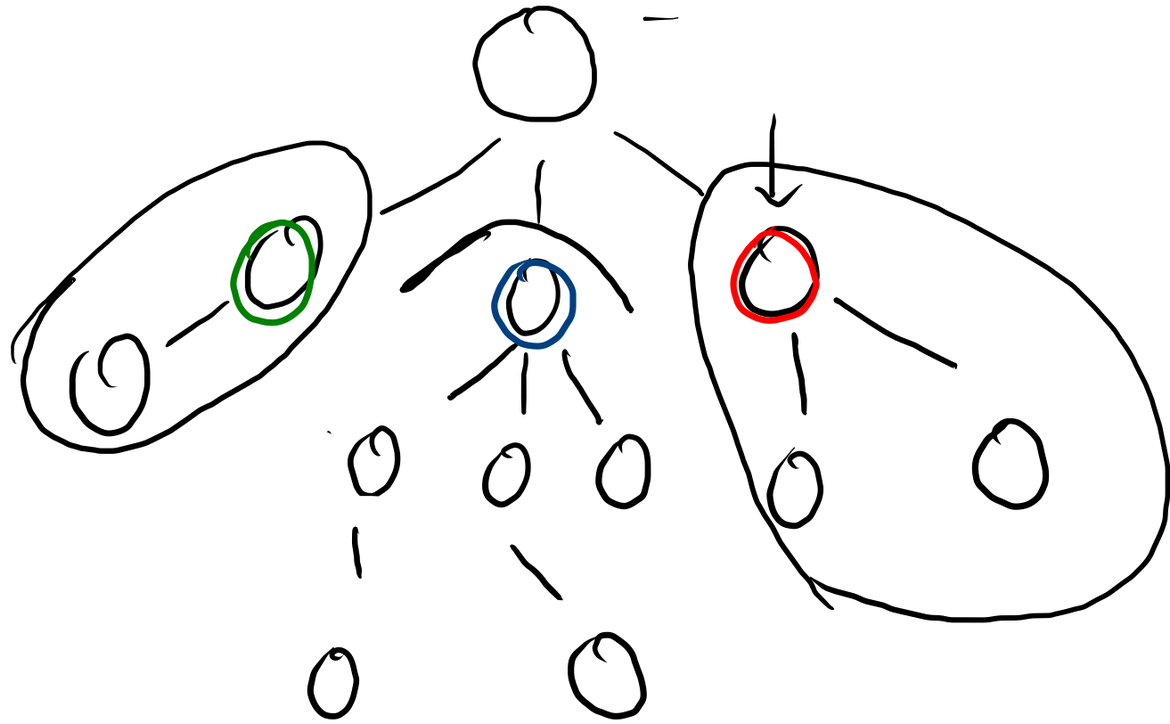


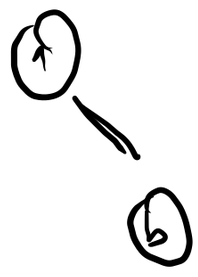


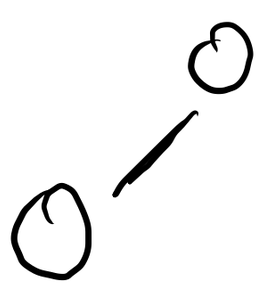

```java
public static boolean areSimilar(Node n1, Node n2) {

    if(n1.children.size() != n2.children.size()){
        return false;
    }

    for(int i=0;i<n1.children.size(); i++){
        Node n1c = n1.children.get(i);
        Node n2c = n2.children.get(i);
        if(areSimilar(n1c, n2c) == false){
            return false;
        }
    }
    return true;
}
```

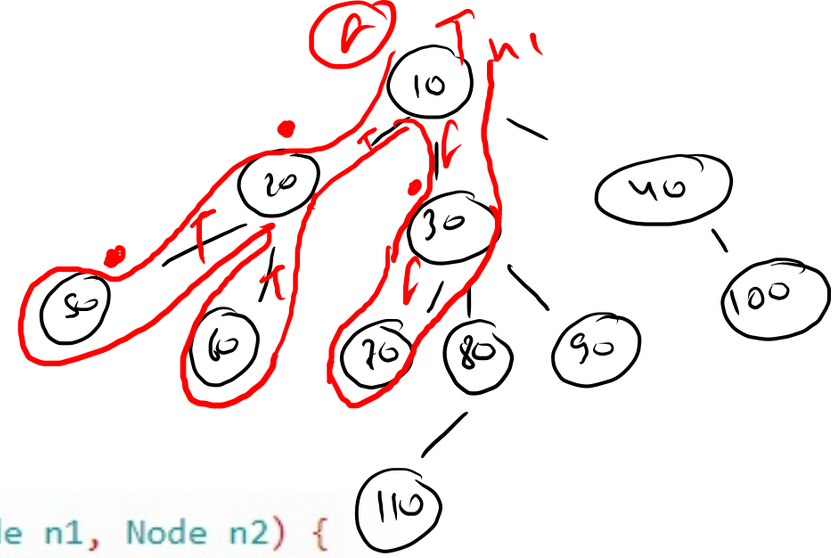
{ 10, d }
false
{10, a}

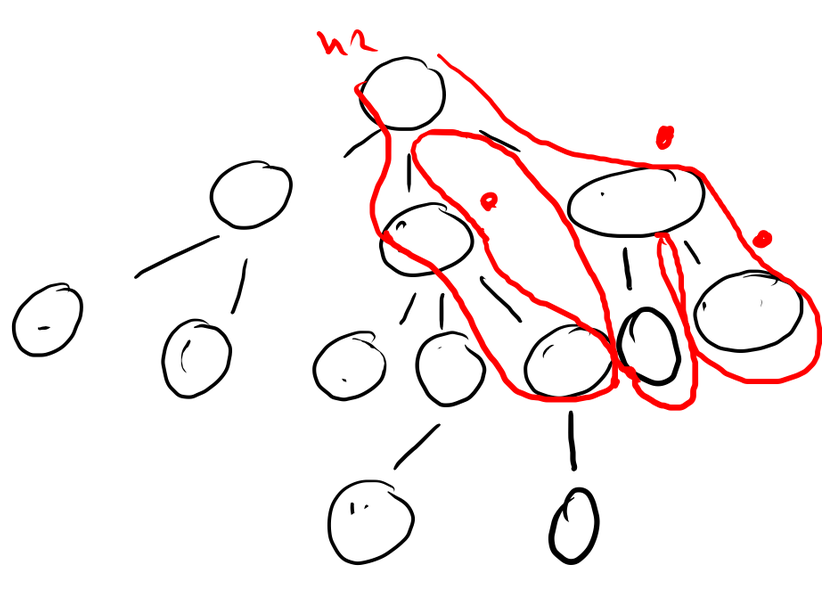

10
20
30
40
50
60
70
80
90
100
110









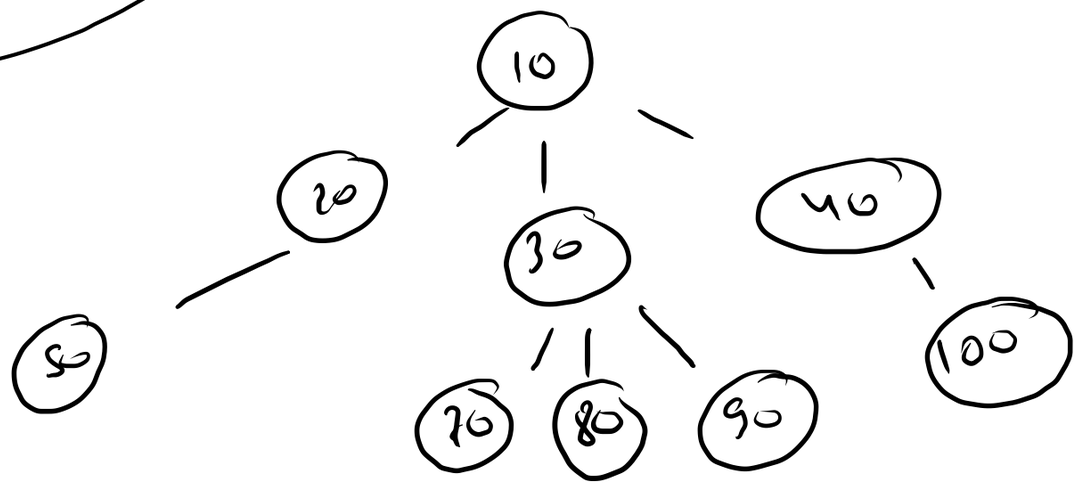


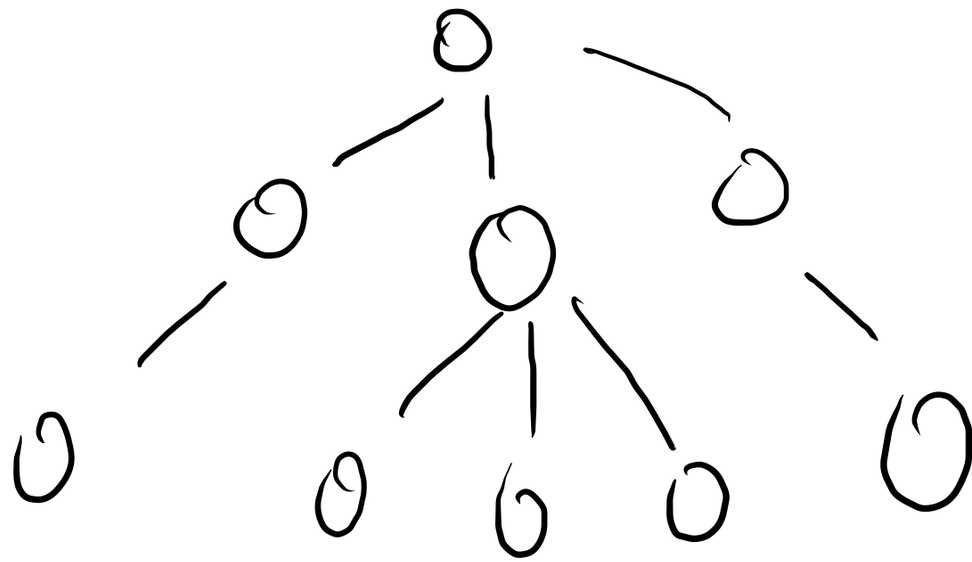

```java
public static boolean areMirror(Node n1, Node n2) {
    if(n1.children.size() != n2.children.size()){
        return false;
    }

    int i=0;
    int j=n1.children.size()-1;
    while(i<n1.children.size()){
        Node n1c = n1.children.get(i);
        Node n2c = n2.children.get(j);

        if(false == areMirror(n1c, n2c)){
            return false;
        }
        i++;
        j--;
    }

    return true;
}
```

11.0

Is Generic Tree Symmetric

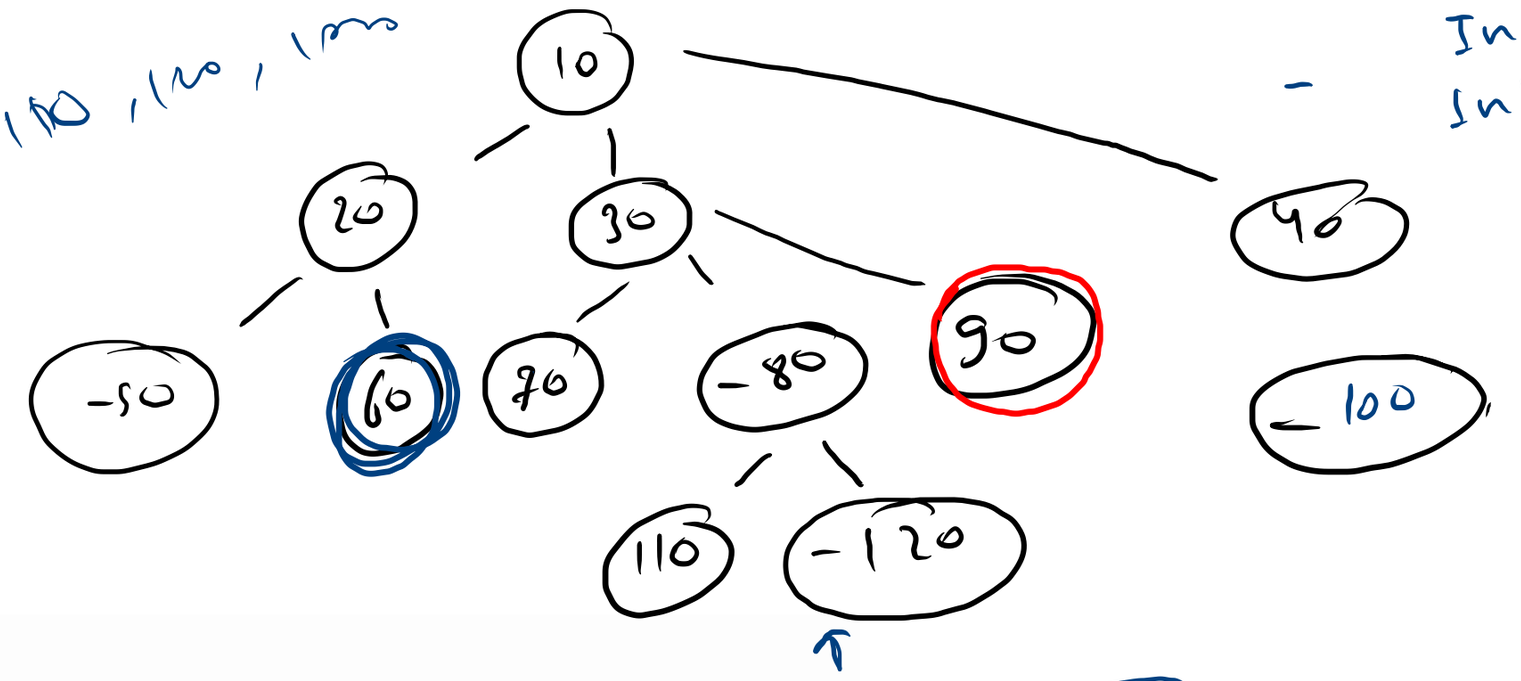
Symetric





d = 70

- ceil → 90, 100, 110, 120, 130
- floor → 60



Integer.MAX_VALUE
Integer.MIN_VALUE

```
24
10 20 -50 -1 60 -1 -1 30 70 -1 -80 110 -1 -120 -1 -1 90 -1 -1 40 -100 -1 -1 -1
70
```

Sample Output

```
CEIL = 90
FLOOR = 60
```

d = -80

floor → -∞

ceil → 10

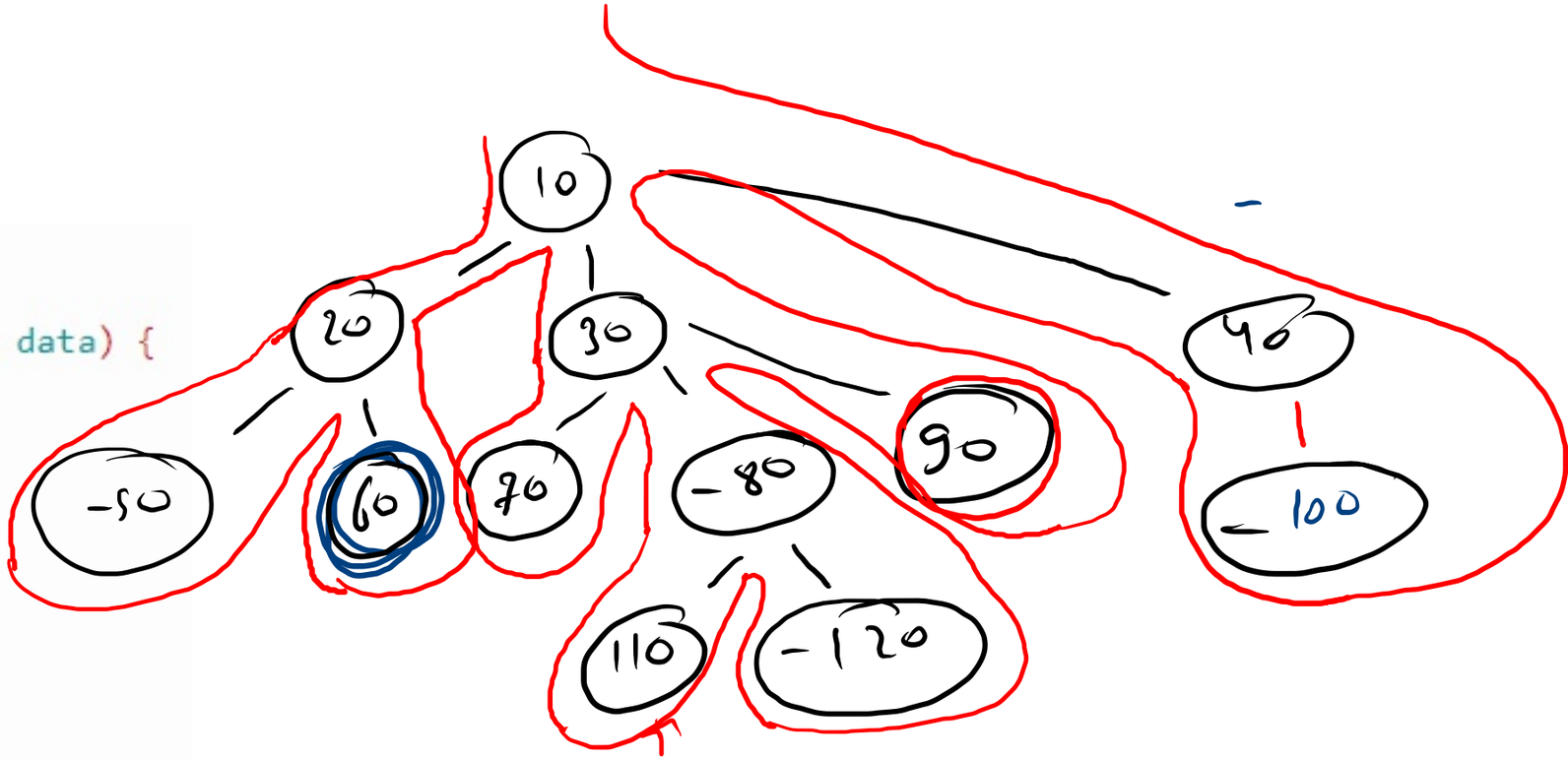
d = 500

floor → 110

ceil → ∞

Data = 20

```java
static int ceil;
static int floor;
public static void ceilAndFloor(Node node, int data) {

    if(node.data < data){
        floor = Math.max(floor, node.data);
    }else if(node.data > data){
        ceil = Math.min(ceil, node.data);
    }

    for(Node child : node.children){
        ceilAndFloor(child, data);
    }
}
```



- ceil ~~∞~~ ~~40~~ 90
- floor ~~-∞~~ ~~10~~ ~~20~~ 60