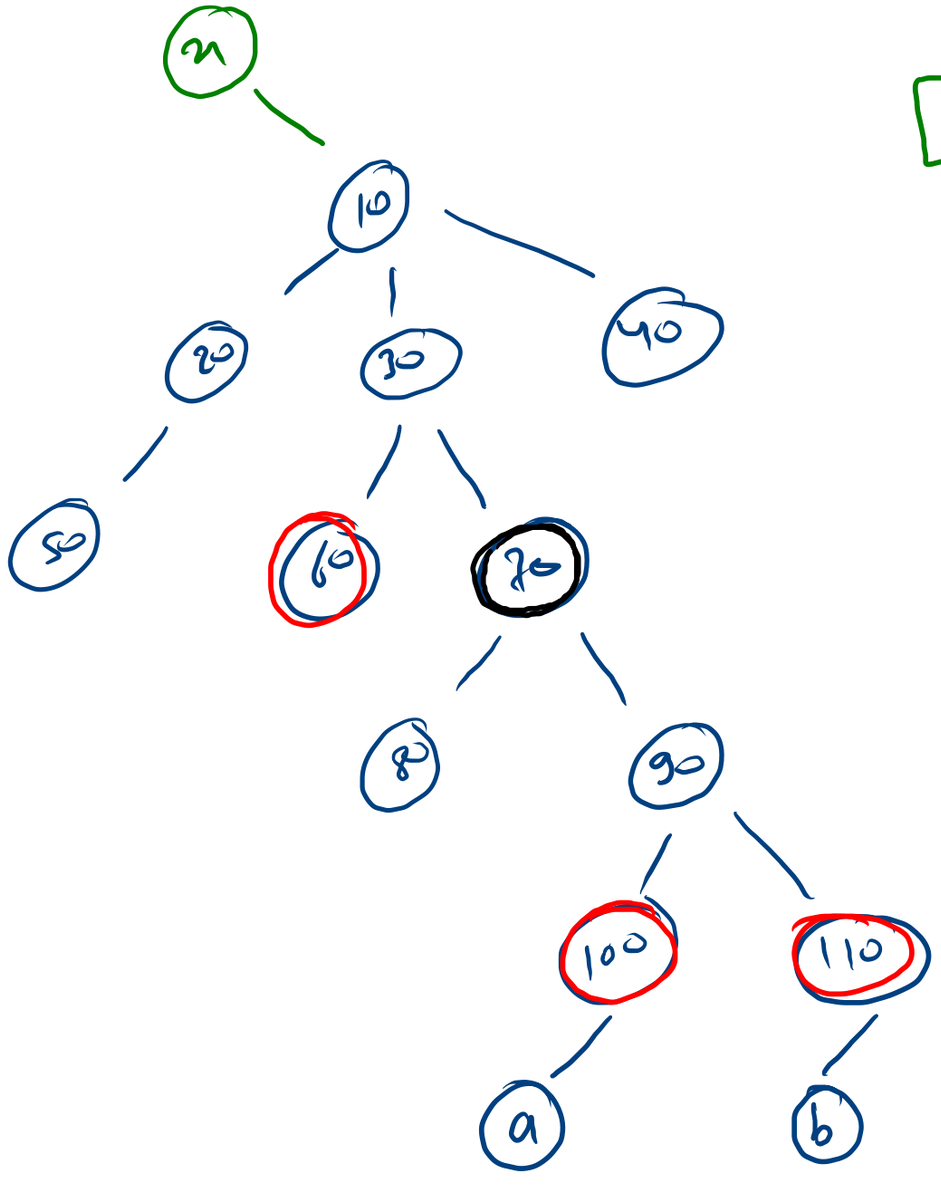
data = 70

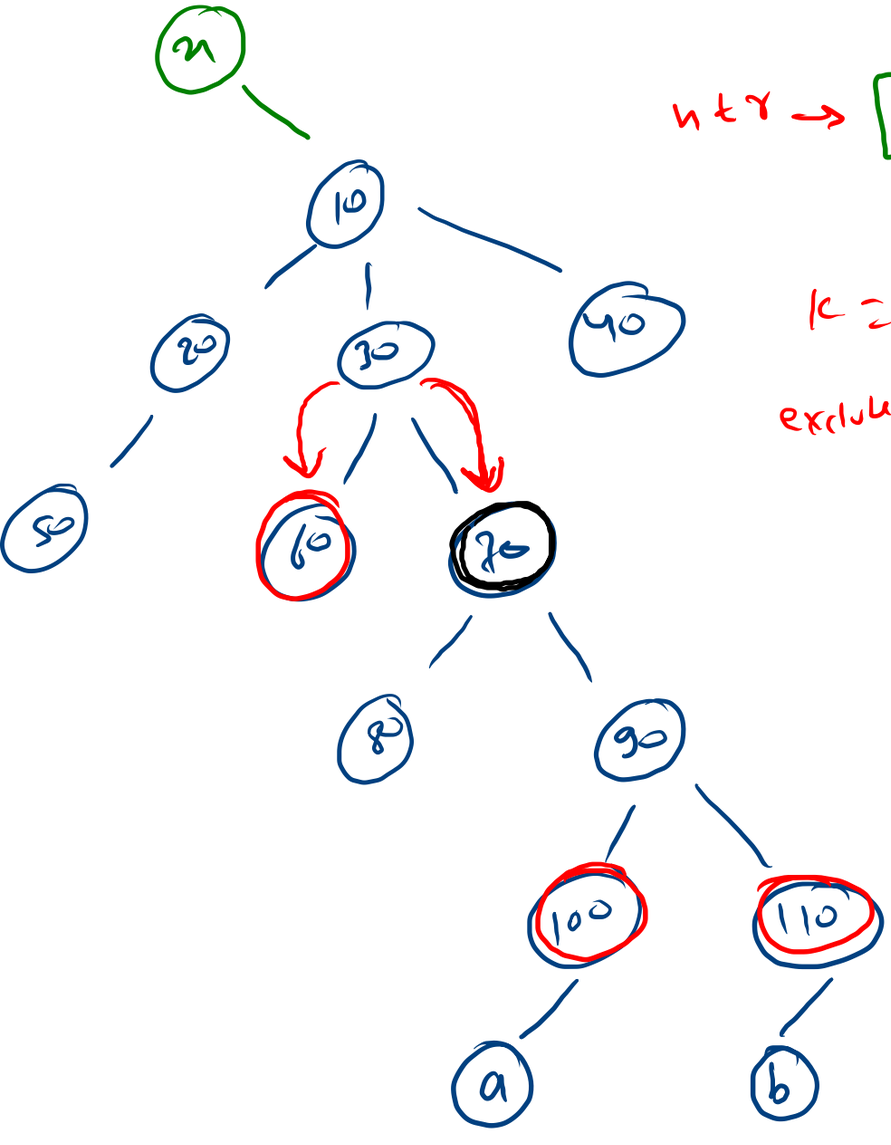
K = 2

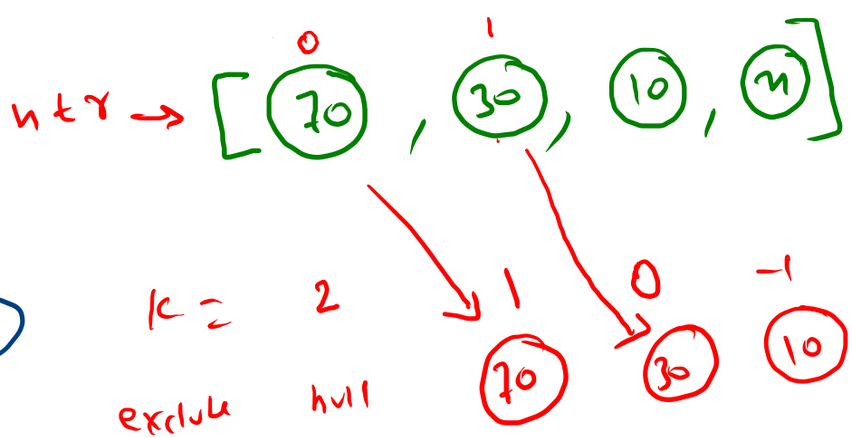
100
110
60
10



[70, 30, 10, n]

2 1 0 -1

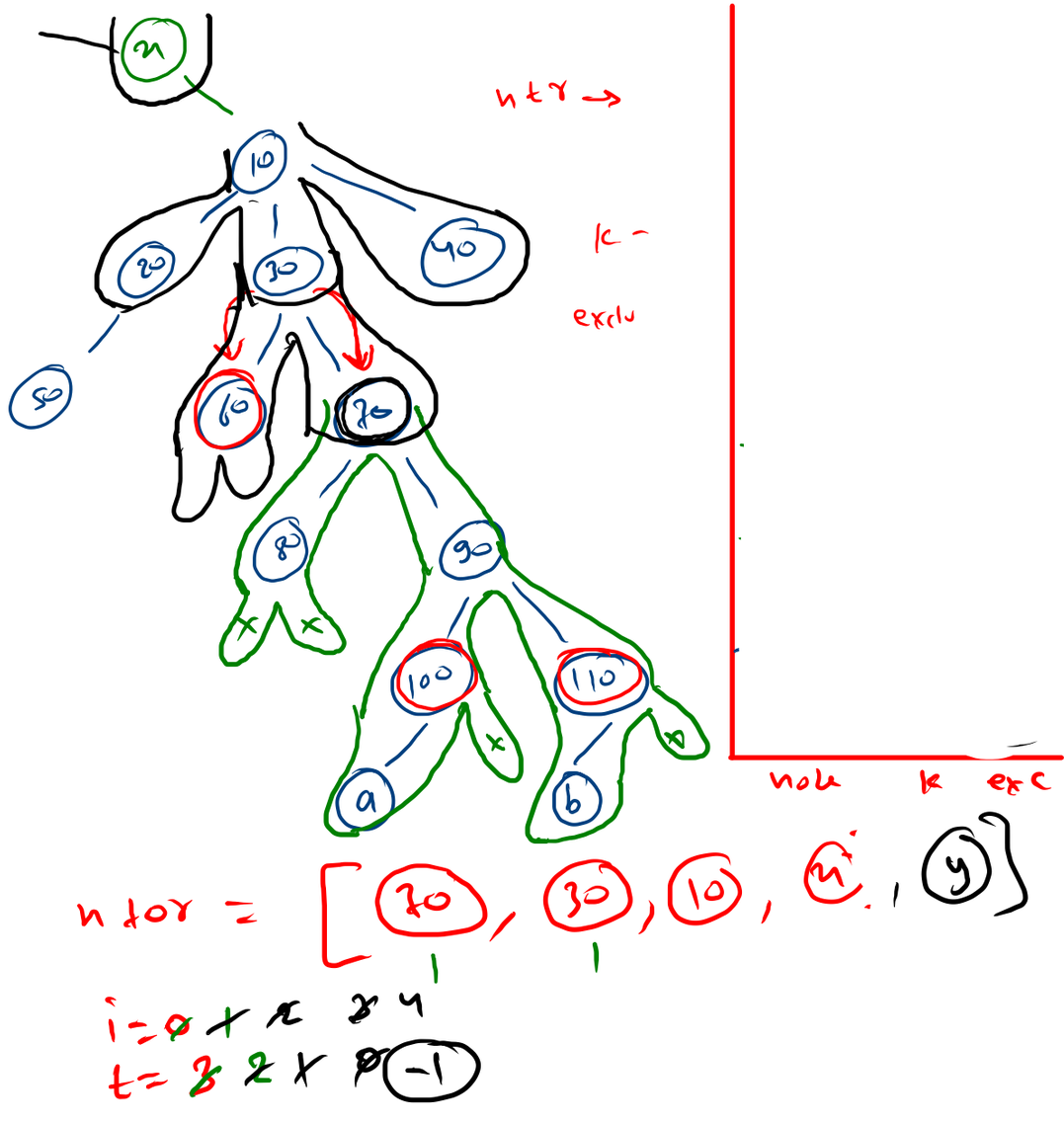


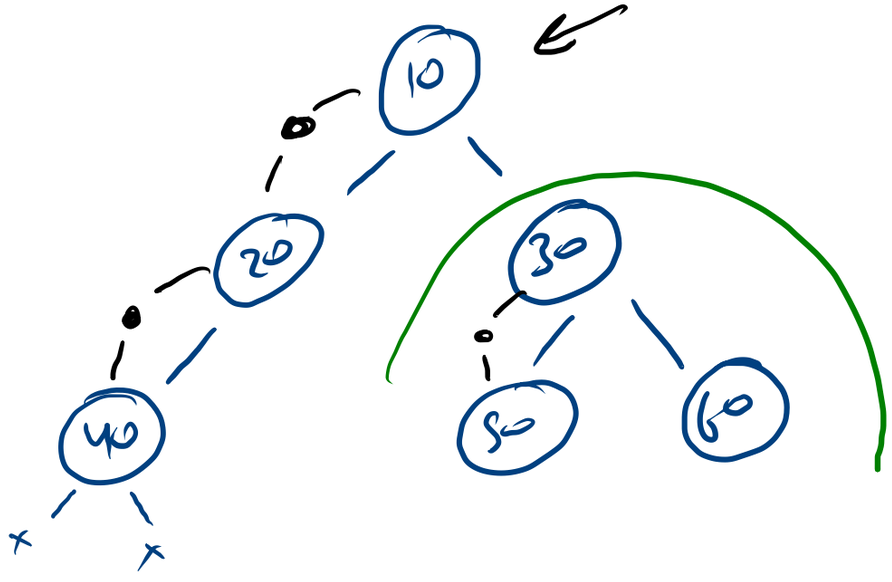


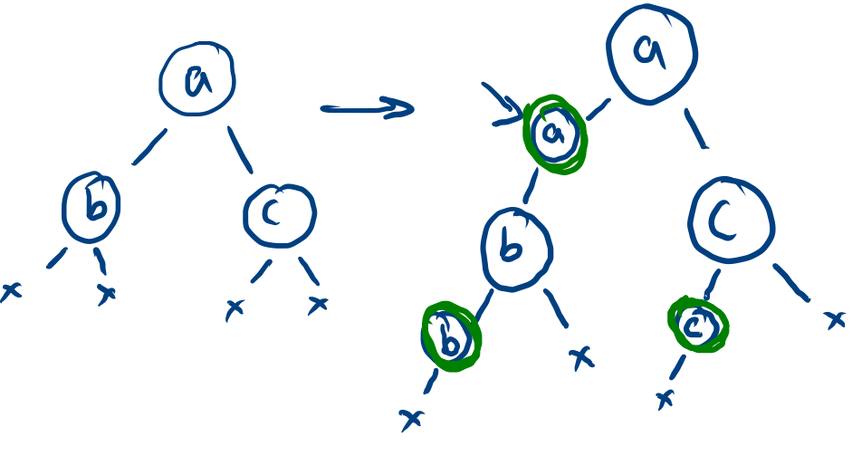

```
k levels down ( Node node, int k, Node exc ){
    if( node == null || node == exc)
        return
}
```

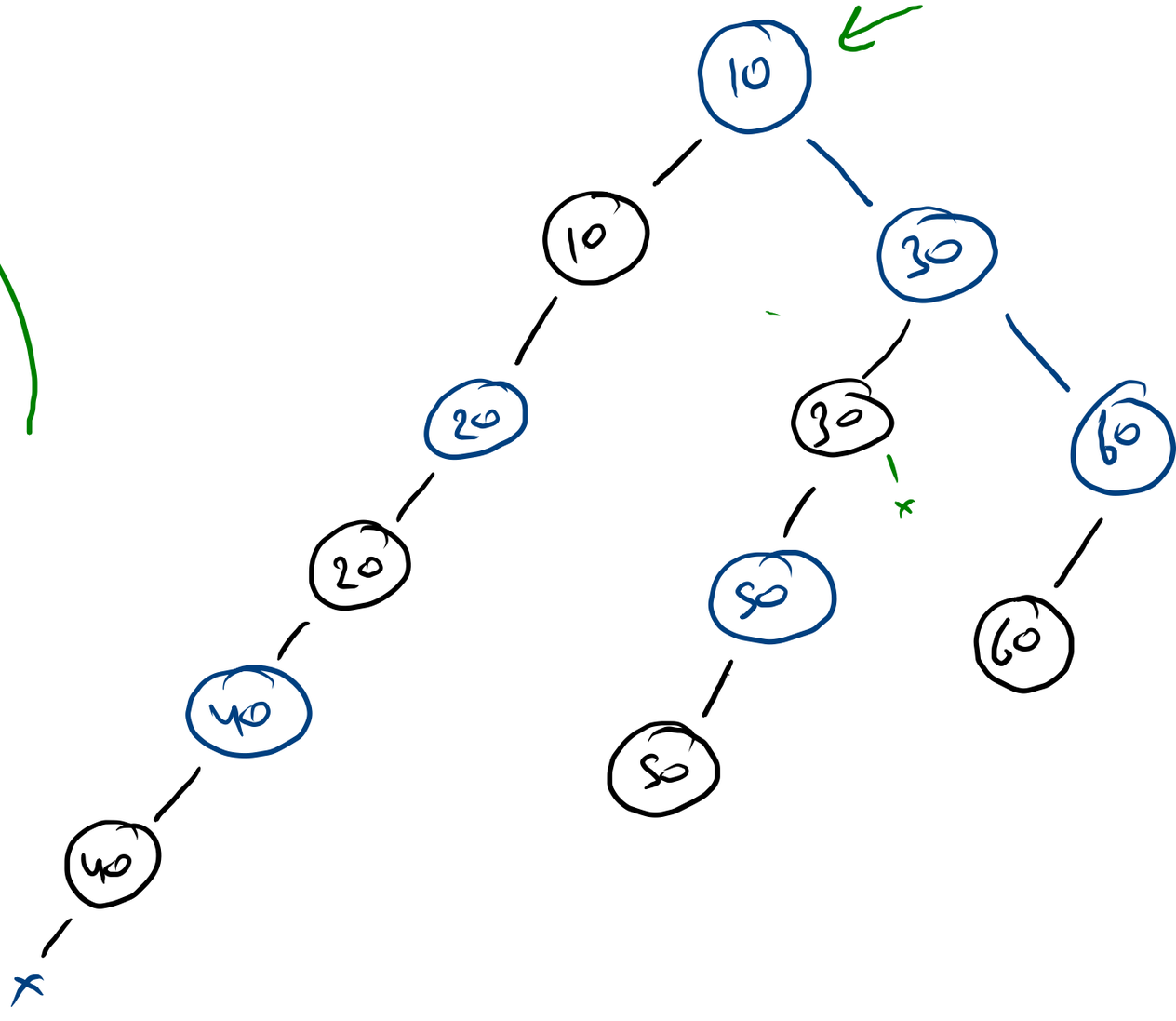
a b 20 40 n

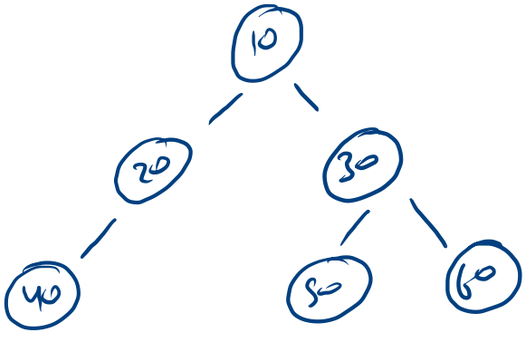
```java
public static void printKNodesFar(Node node, int data, int k) {
    ArrayList<Node> ntor = nodeToRootPath(node, data);

    for(int i=0, t = k;i<ntor.size() && t>=0;i++, t--){
        Node n = ntor.get(i);
        Node exc = i==0? null: ntor.get(i-1);
        printKLevelsDown(n, t, exc);
    }
}

public static void printKLevelsDown(Node node, int k, Node exc){
    if(node == null || node == exc){
        return;
    }

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

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




10
20
30
40
50
60


a
b
c
a
b
c
a
b
c

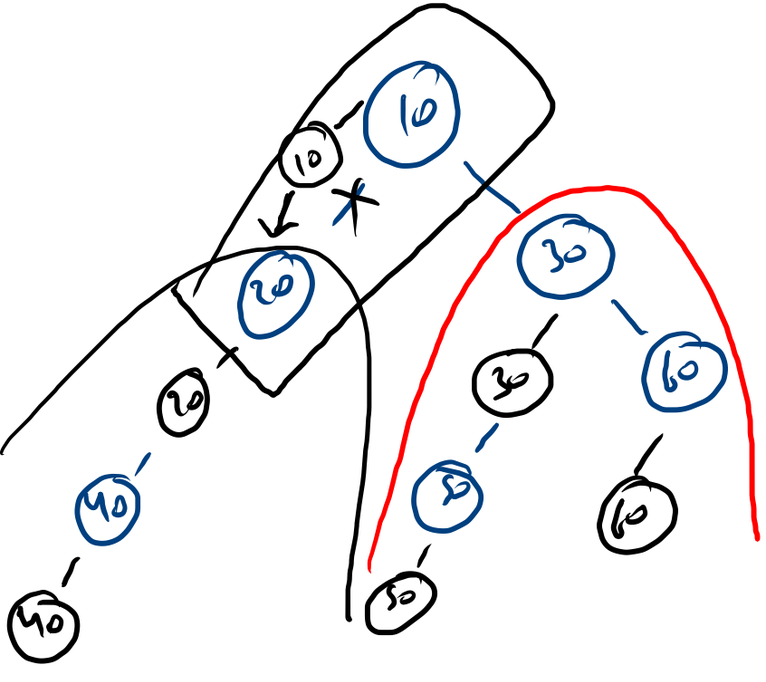

10
10
30
20
30
60
20
50
40
50
60

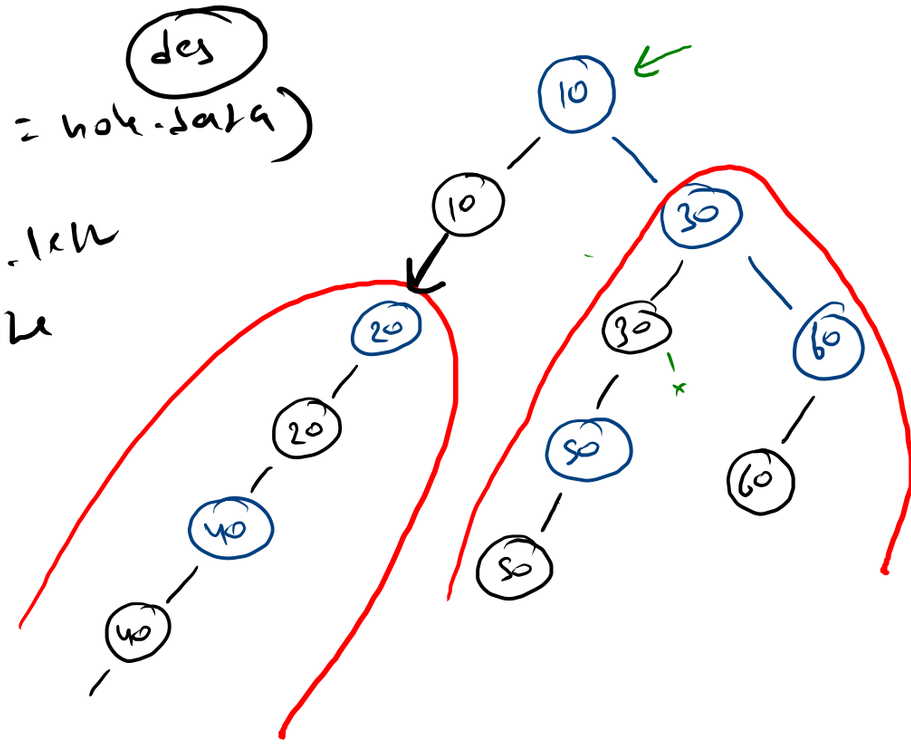


clone (data = node.data)

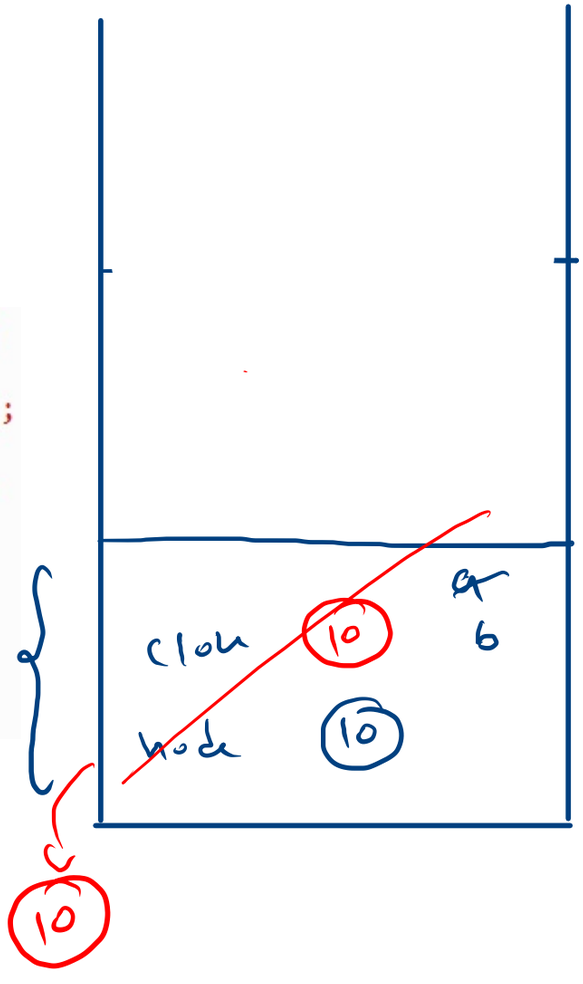
clone.left = node.left

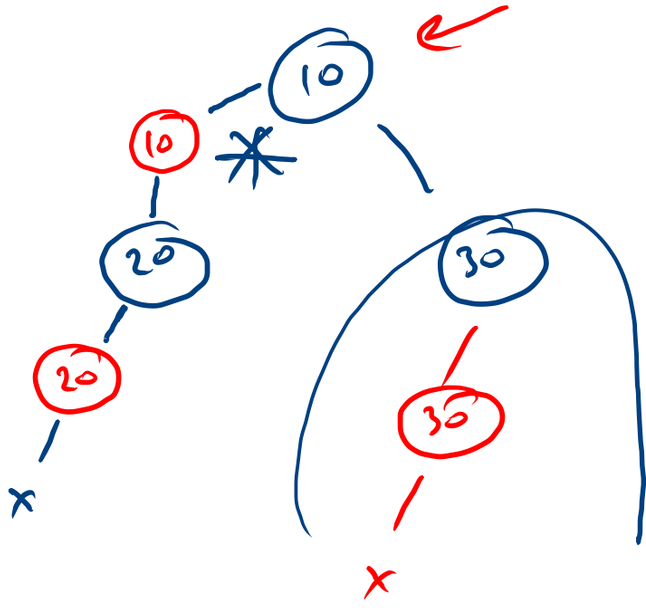
node.left = clone

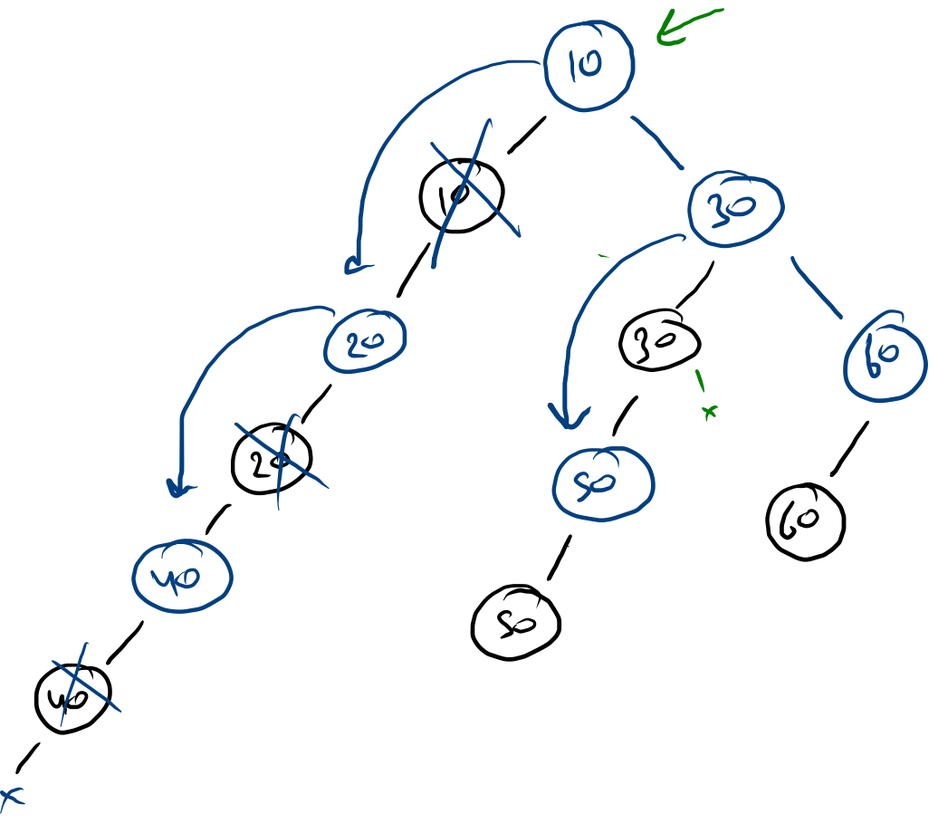


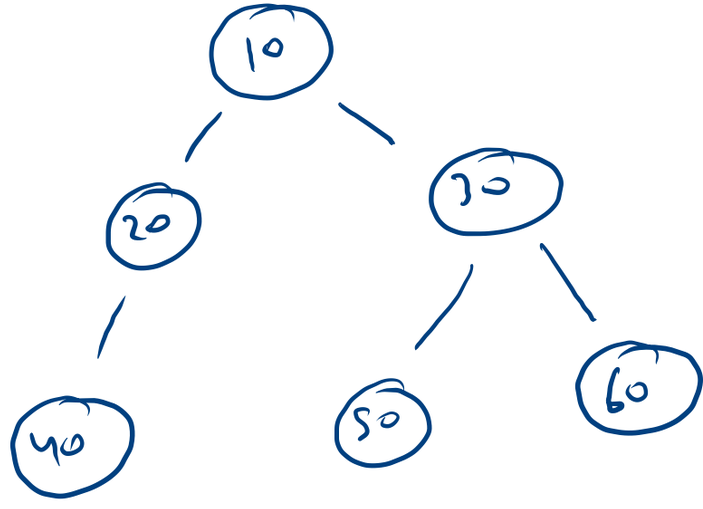


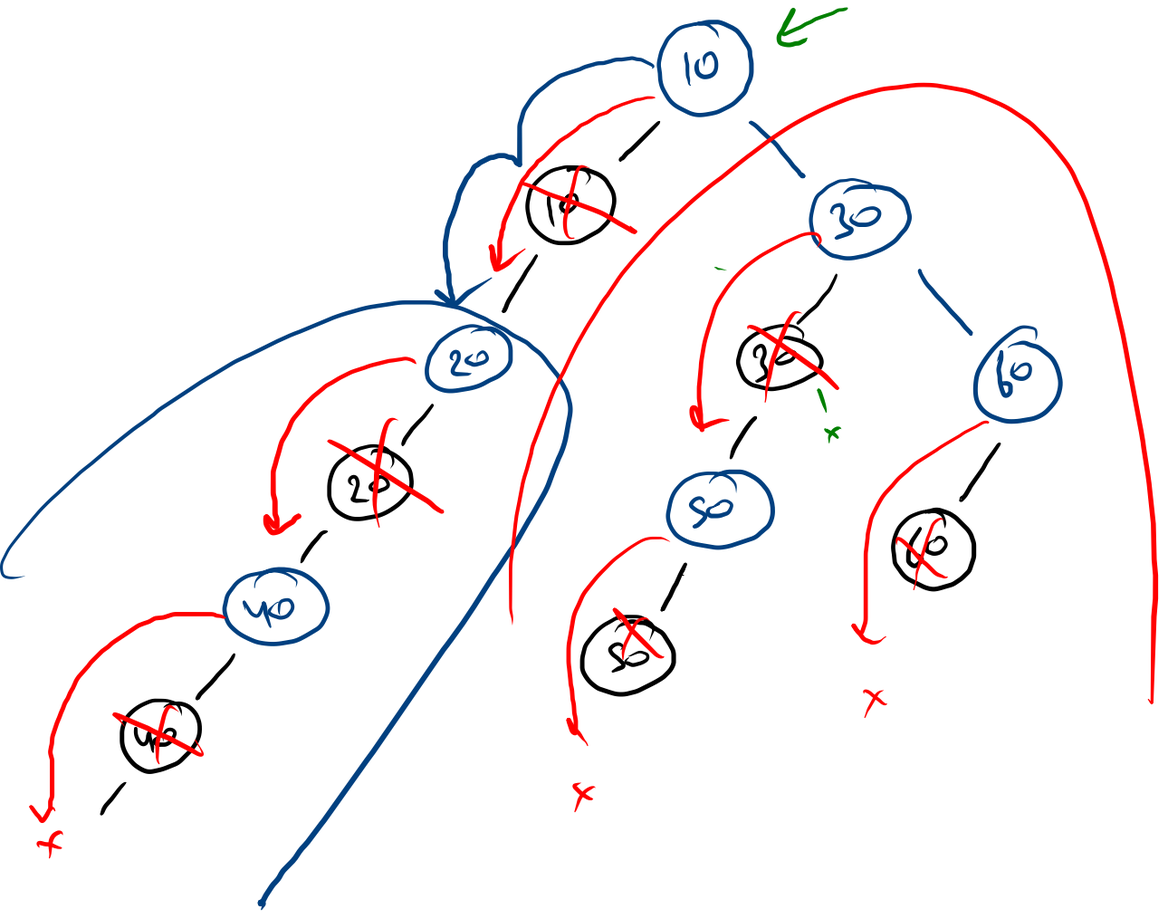
```java
public static Node createLeftCloneTree(Node node){
    if(node == null)return null;
    Node clone = new Node(node.data, null, null);

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

    node.left = clone;

    return node;
}
```





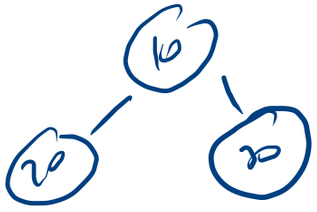

10
10
30
20
30
60
20
50
40
50
60
40


10
20
30
40
50
60

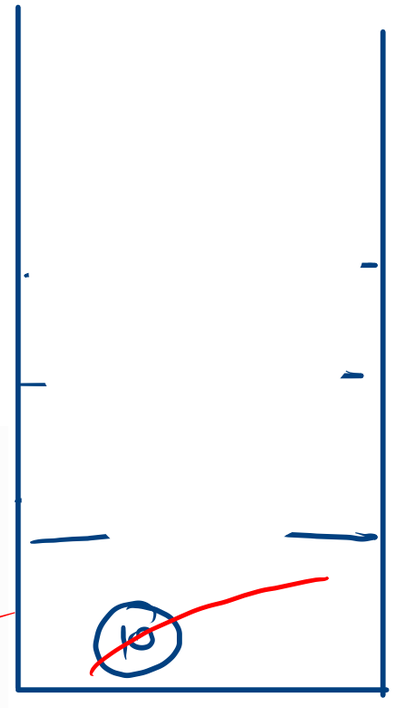


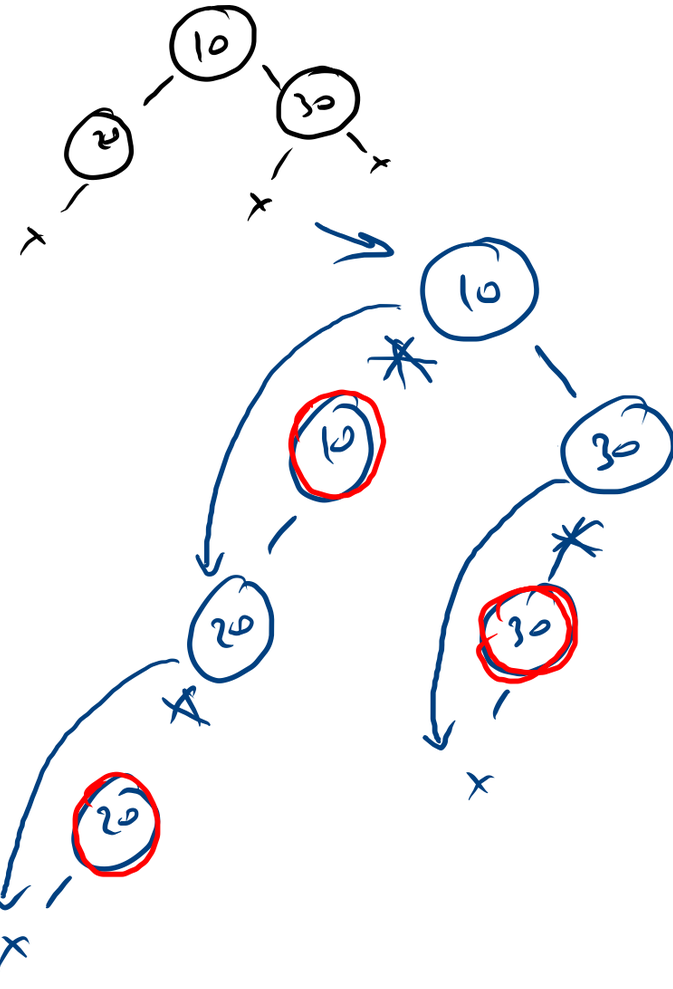
node.left = build(node.left.left)

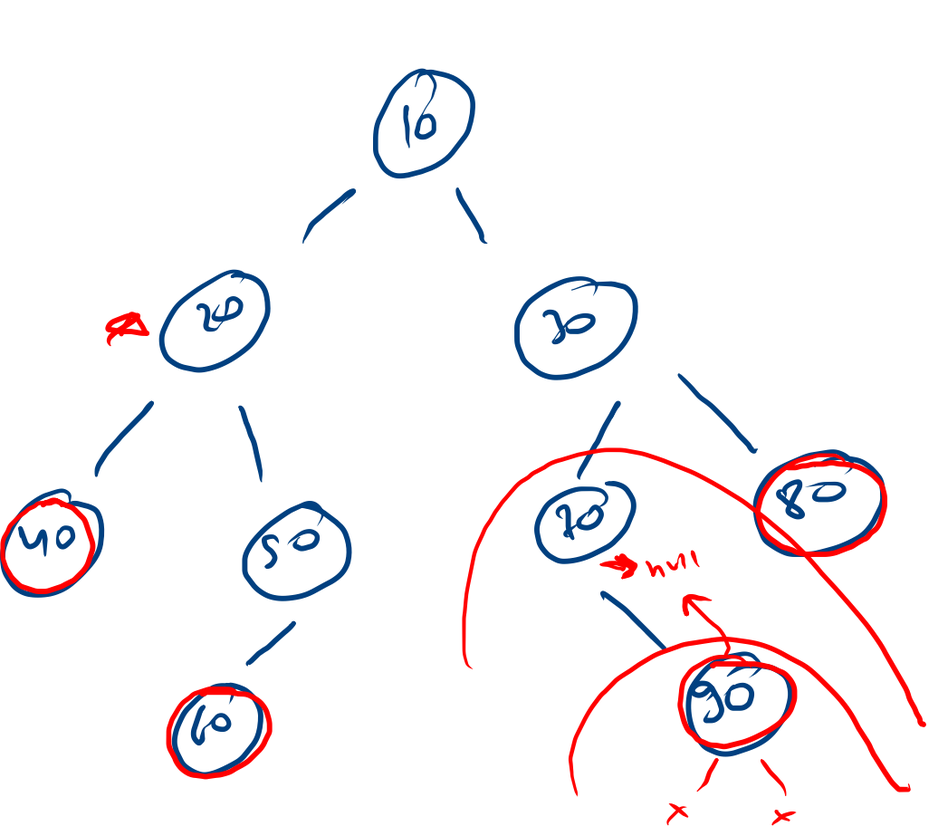
build(node.right)

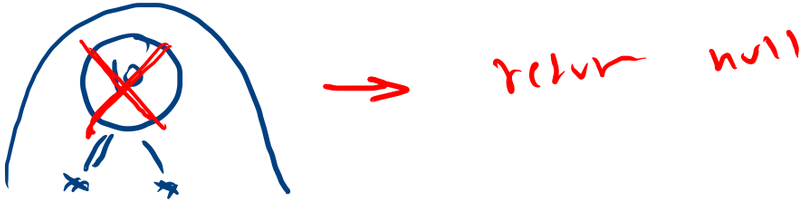


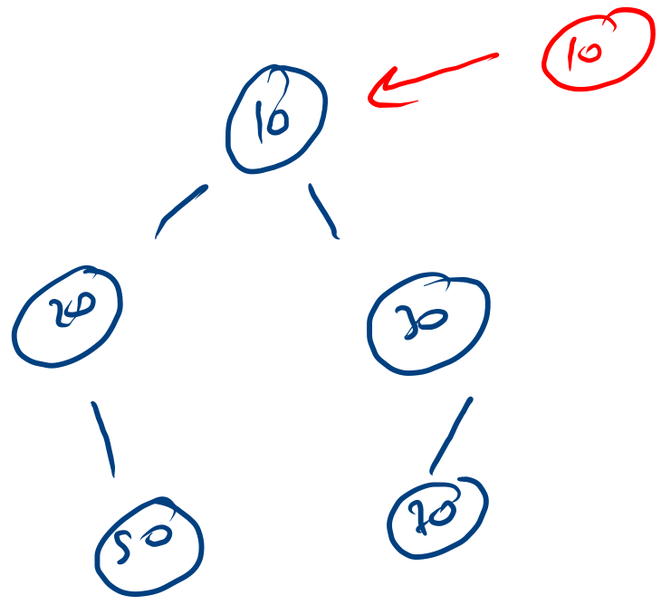

```java
public static Node transBackFromLeftClonedTree(Node node){
    if(node == null)return null;

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

    return node;
}
```

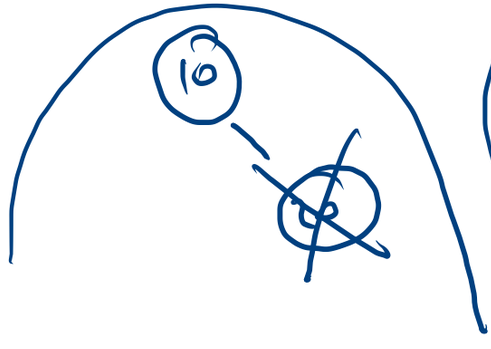


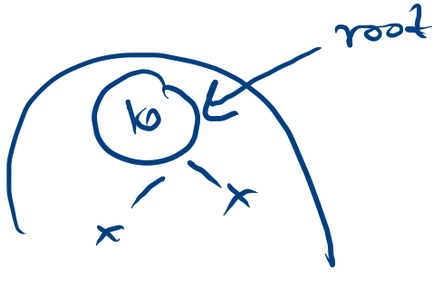


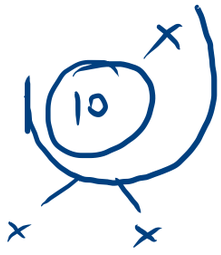

10
20
30
40
50
70
80
60
90
null


10
return null

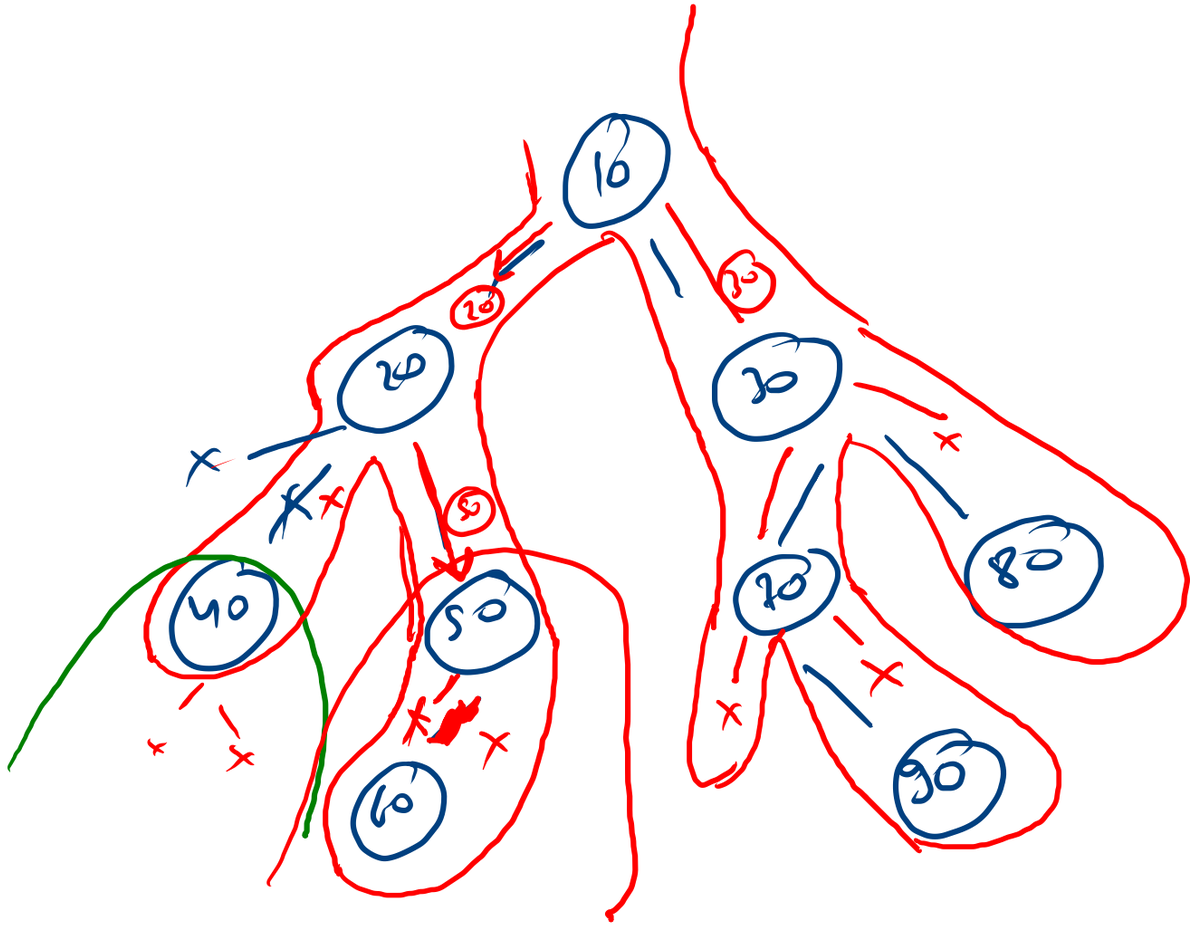

10
10
20
30
50
70


10


root
10




```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;
}
```

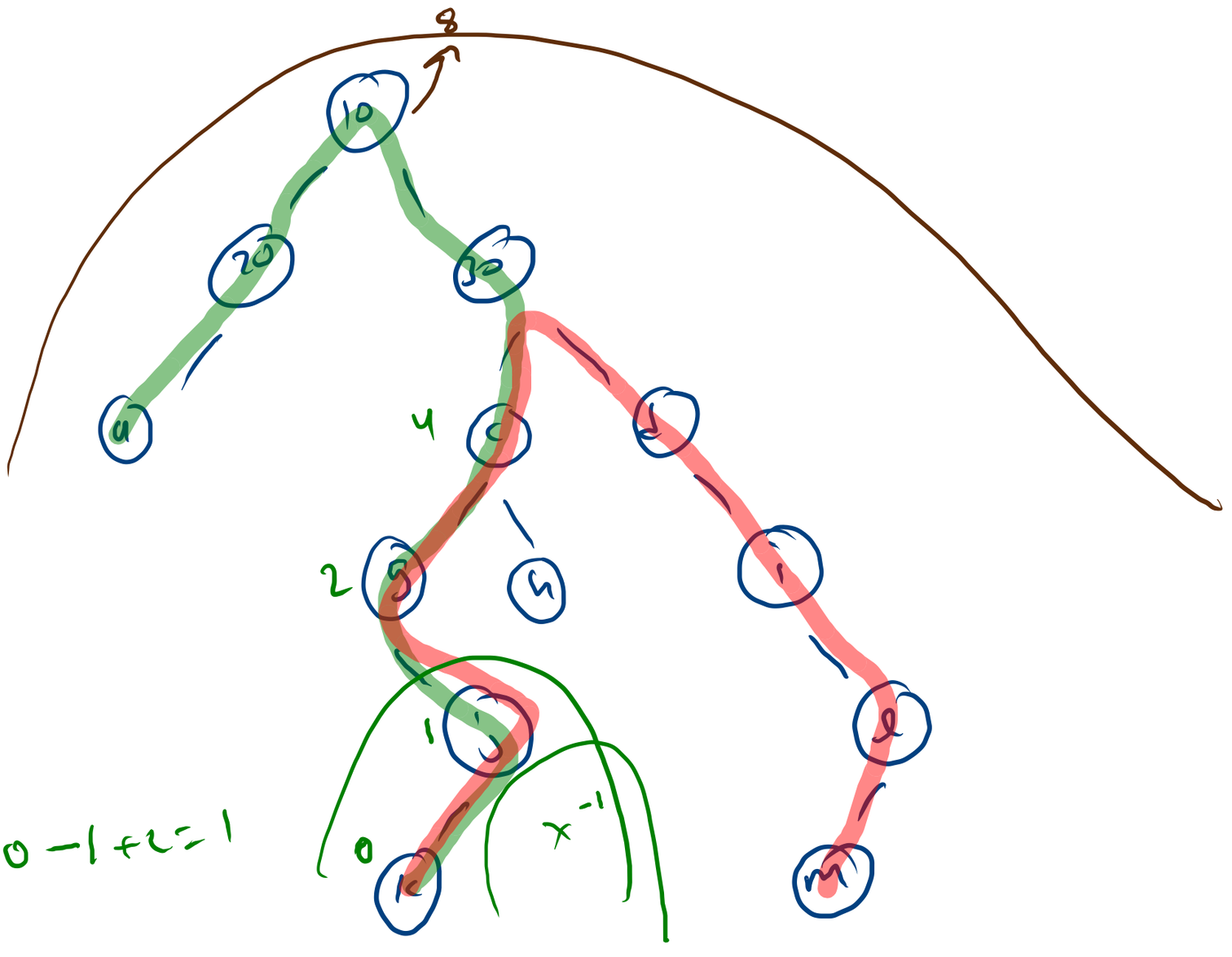


2 nodes

max distance in terms of edges

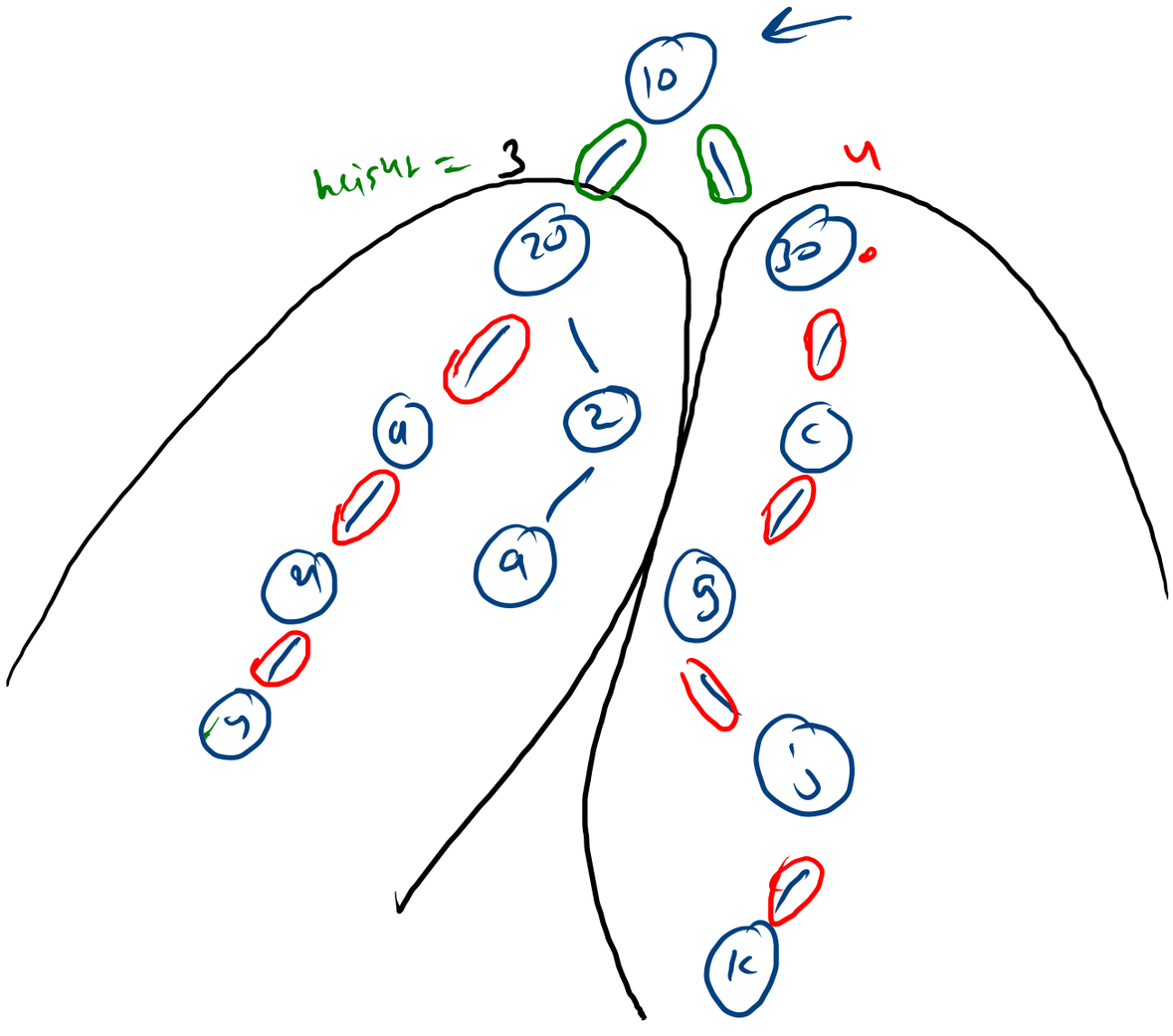
a ... c → 7

k ... m → 8



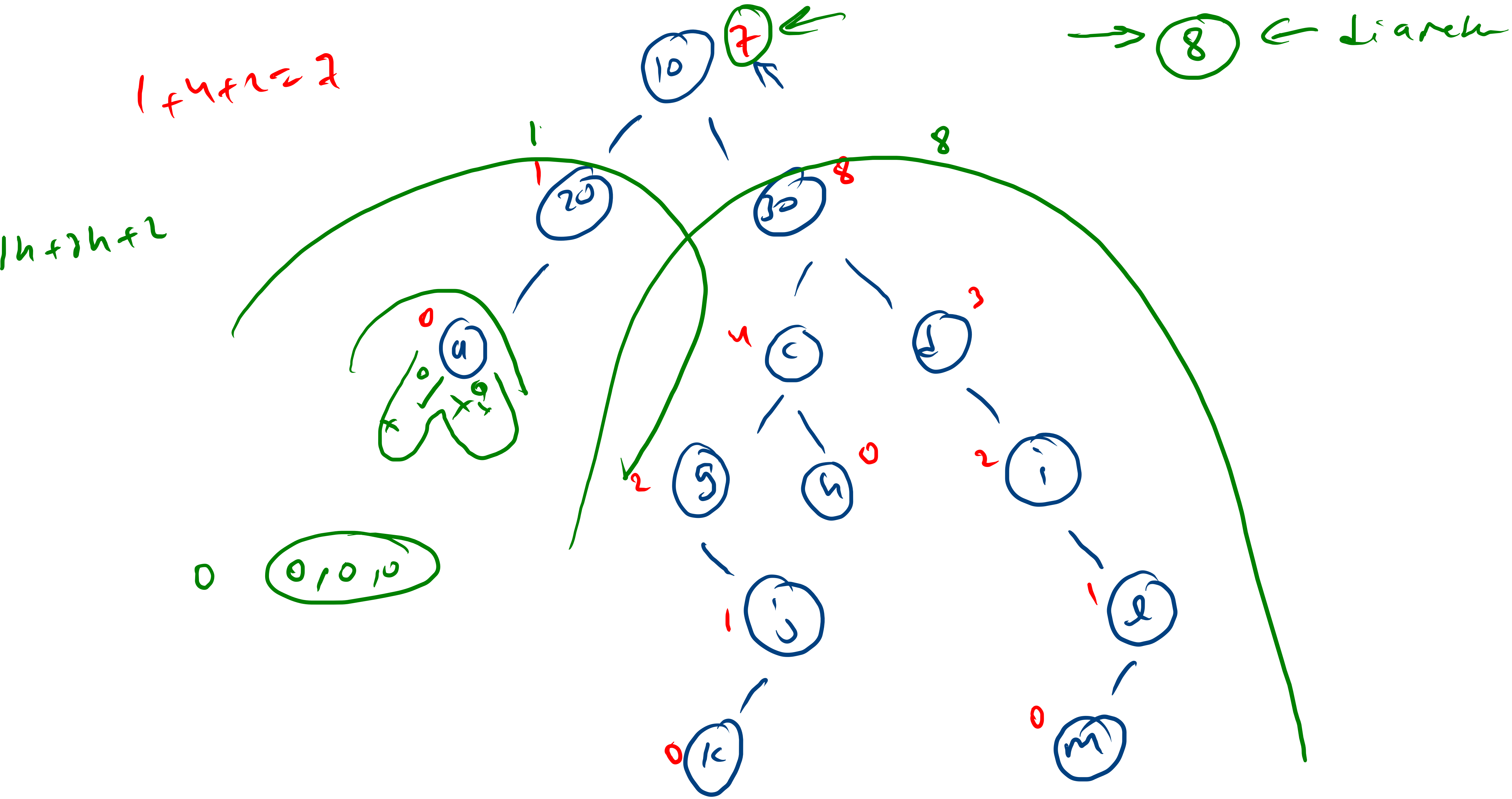
0 - 1 + 2 = 1

2 nodes



3+4+2 = 9

delh + rish + 2


1+4+2=7
1h+2h+2
10
7
8
diameter
20
30
a
c
d
g
h
i
j
k
l
m
0,0,10

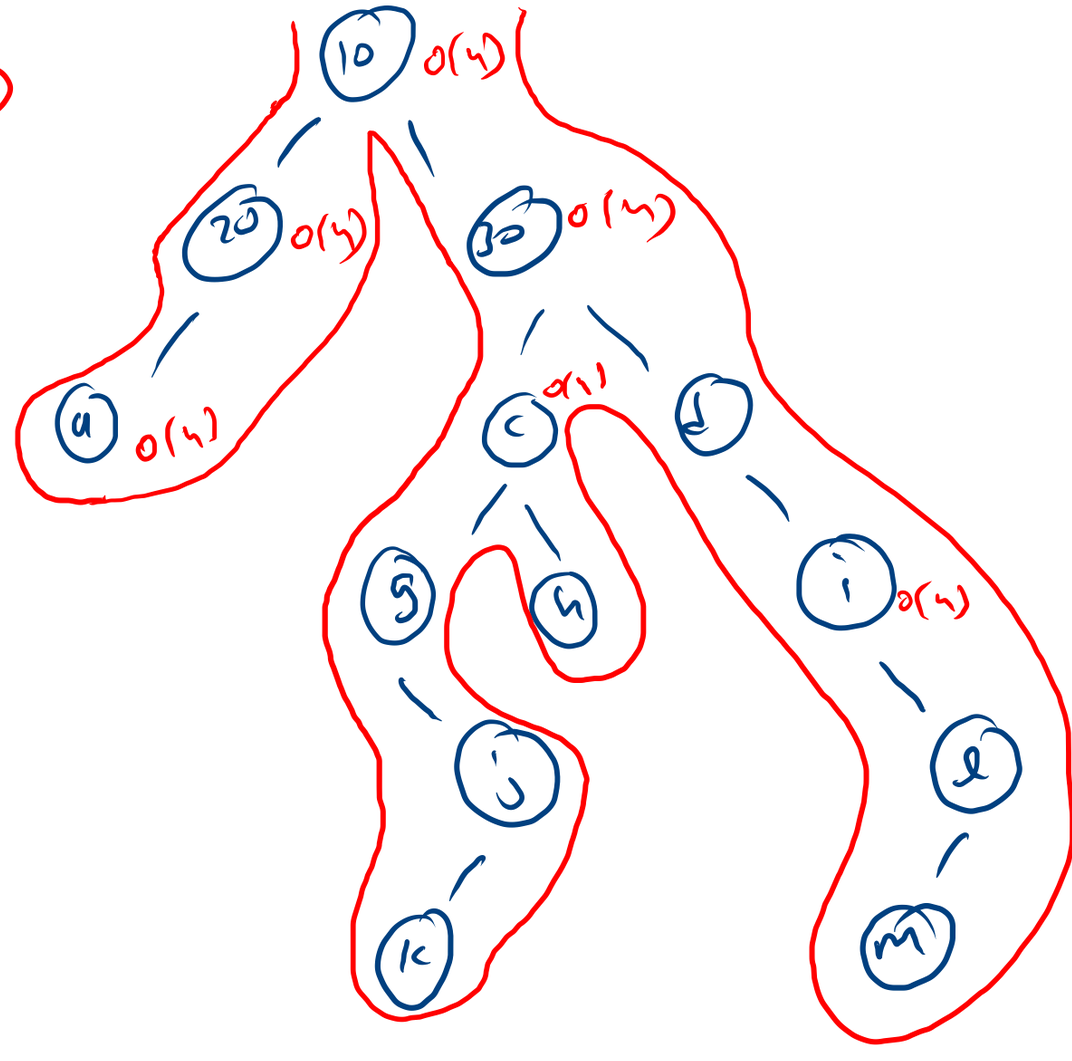
h

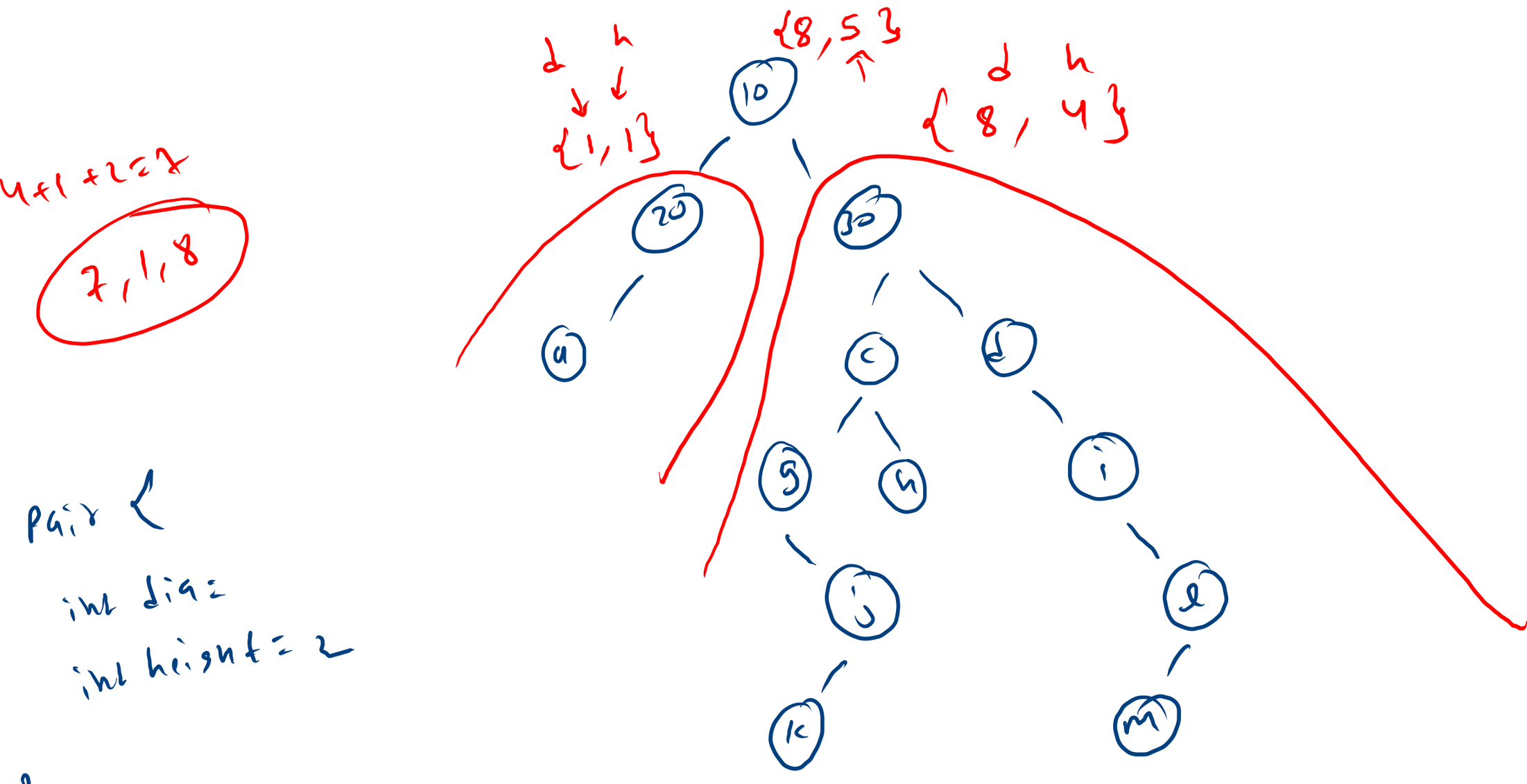
$O(n) + O(n) + O(n) + \cdots$ n times

$\approx O(n^2)$

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

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

    int dis = lh+rh+2;

    int ldia = diameter1(node.left);
    int rdia = diameter1(node.right);

    int max = Math.max(ldia, rdia);
    max = Math.max(max, dis);

    return max;
}
```

O(n)
O(n)





4+1+2=7

7, 1, 8

Pair {

int dia;

int height = 2

}

dis 7

max = 8

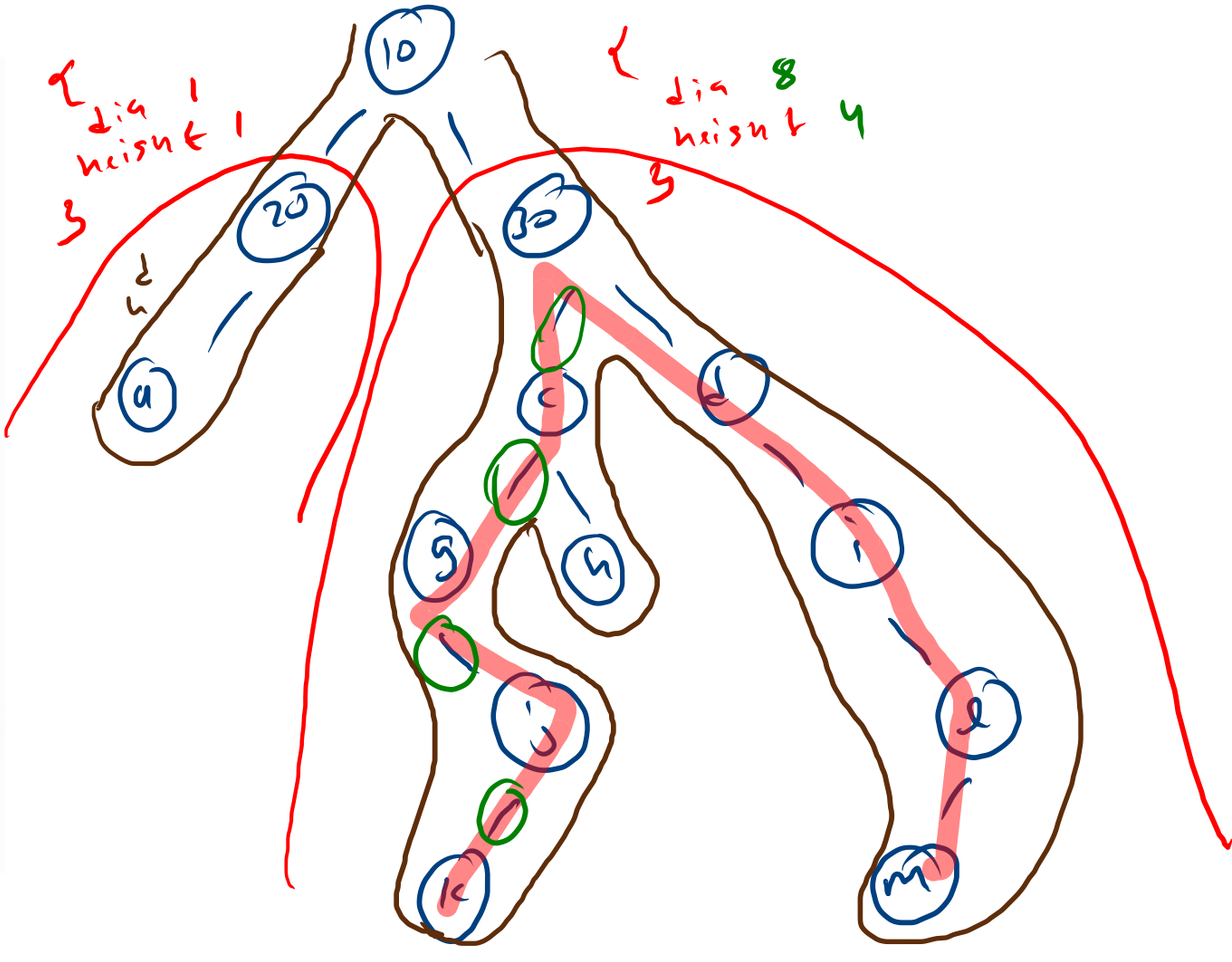
{ dia 8, height 5 }

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

    DiaPair ldia = diameter(node.left);
    DiaPair rdia = diameter(node.right);

    int dis = ldia.height + rdia.height + 2;
    int max = Math.max(ldia.dia, rdia.dia);
    max = Math.max(max, dis);

    DiaPair dp = new DiaPair();
    dp.dia = max;
    dp.height = Math.max(ldia.height, rdia.height)+1;

    return dp;
}
```

O(1)

8, 7

N → O(n)



diameter

height

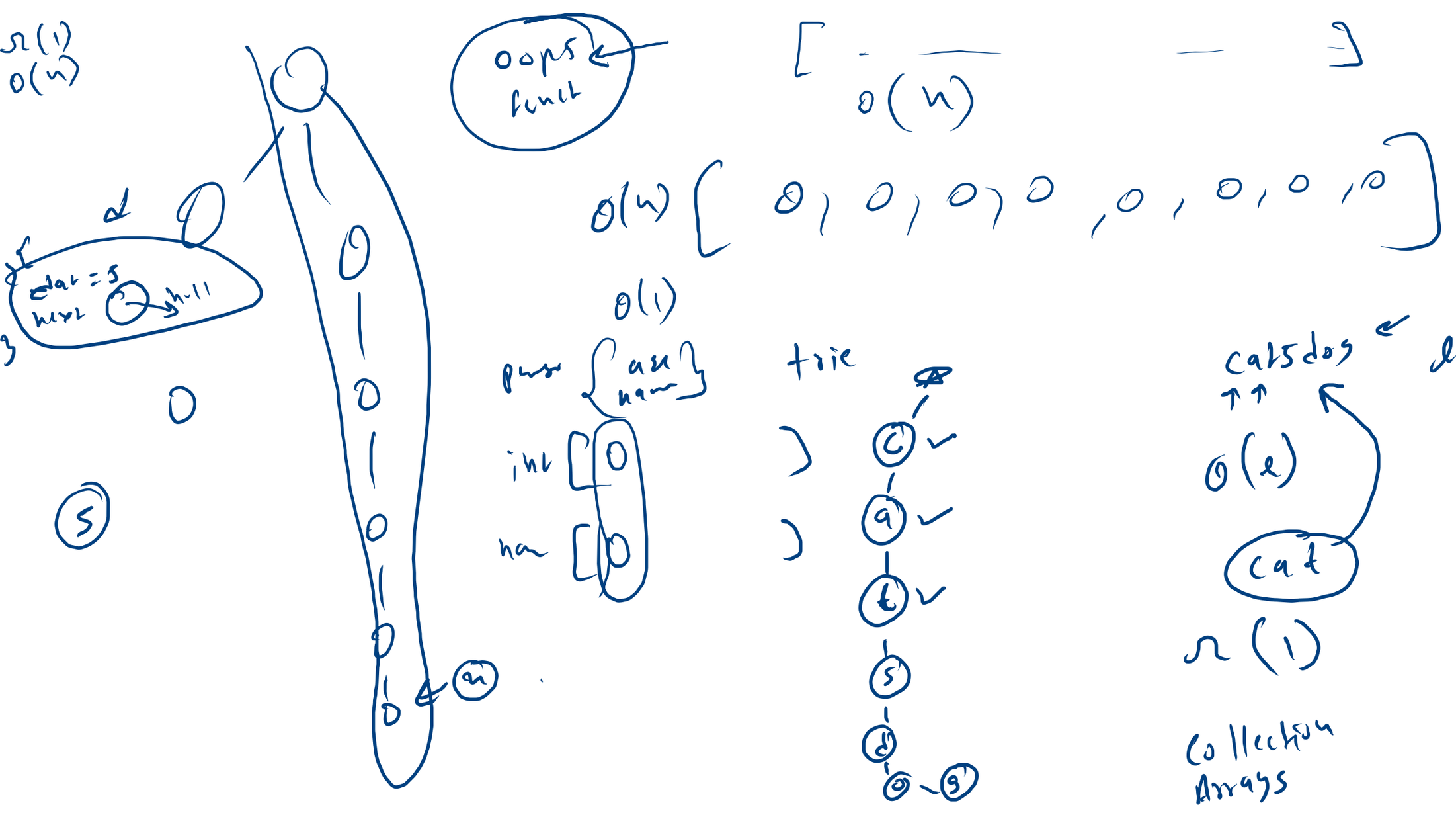
```java
public static int diameter1(Node node) {
    if(node == null){
        return 0;
    }

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

    int dis = lh+rh+2;

    int ldia = diameter1(node.left);
    int rdia = diameter1(node.right);

    int max = Math.max(ldia, rdia);
    max = Math.max(max, dis);

    return max;
}
```

O(n)

h

O(n)

height


O(1)
O(n)
oops
func
O(n)
O(n)
O(1)
push
add
name
int
name
trie
c
a
t
s
d
o
g
catsdog
O(L)
cat
O(1)
Collection
Arrays