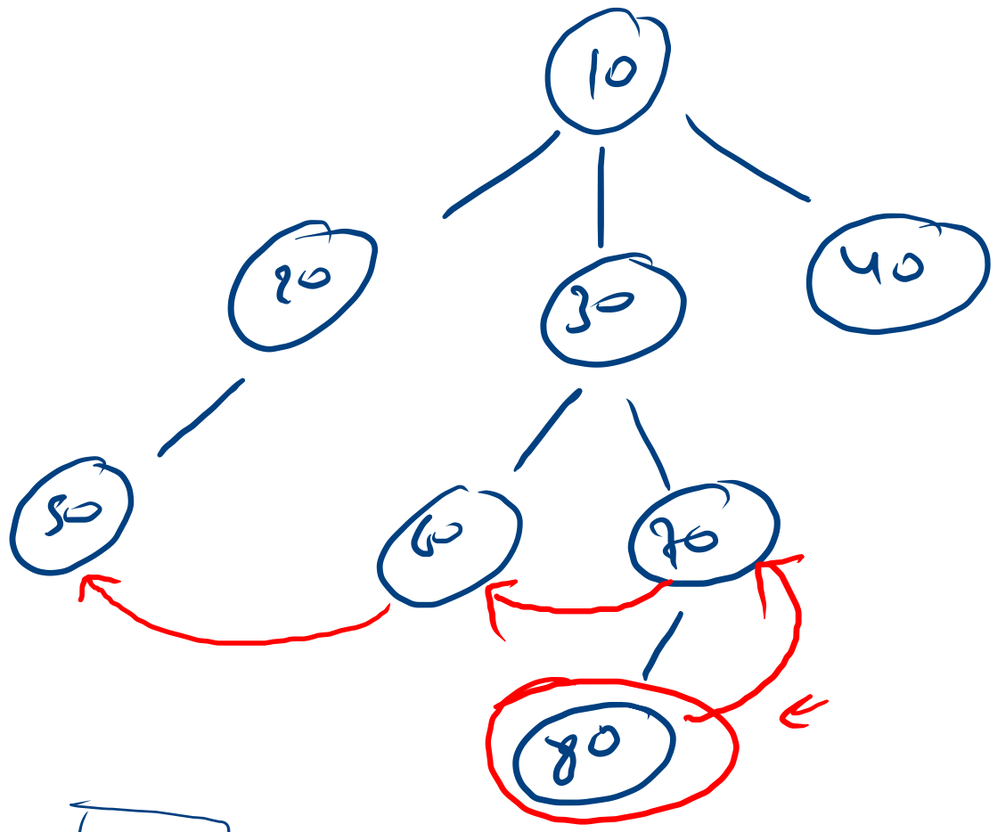
k largest

1 → 80

2 → 70

3 → 60



data 30

ceil → 40

floor → 20

data ∞

floor → 80

data ∞

1 flor → 80 →

2 flr 80 → 70

floor 70 → 60

k
1
2
3

ceil → ∞

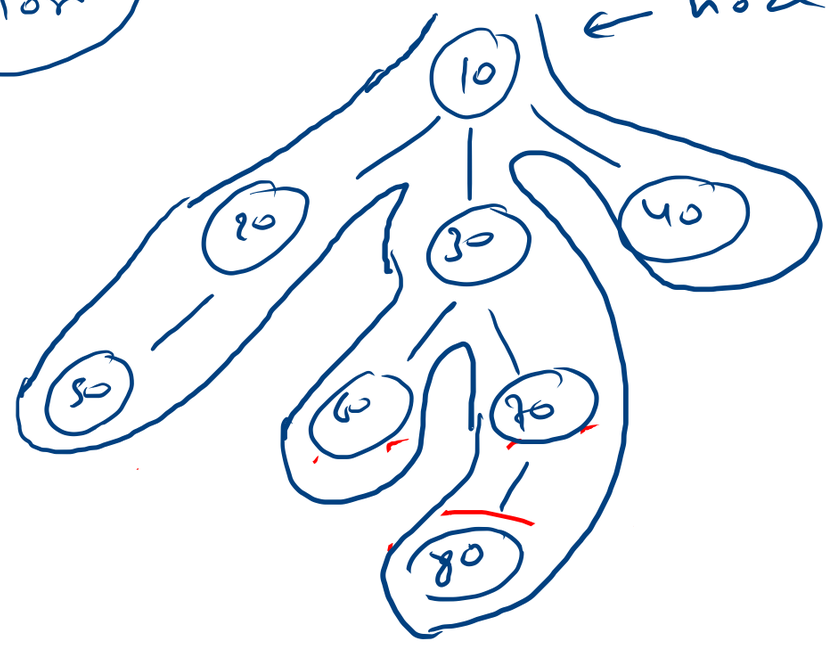
floor → -∞

k = 3

data - ~~∞~~ ~~80~~ ~~20~~ 60

floor

80




```
public static int kthLargest(Node node, int k){
    int data = Integer.MAX_VALUE;∞

    for(int i=0;i<k;i++){
0       ceil = Integer.MAX_VALUE;
1       floor = Integer.MIN_VALUE;
2       ceilAndFloor(node, data);
        data = floor;
    }
    return data;
}
```

floor → ~~-∞~~ 10 20 80 60 70 ~~80~~

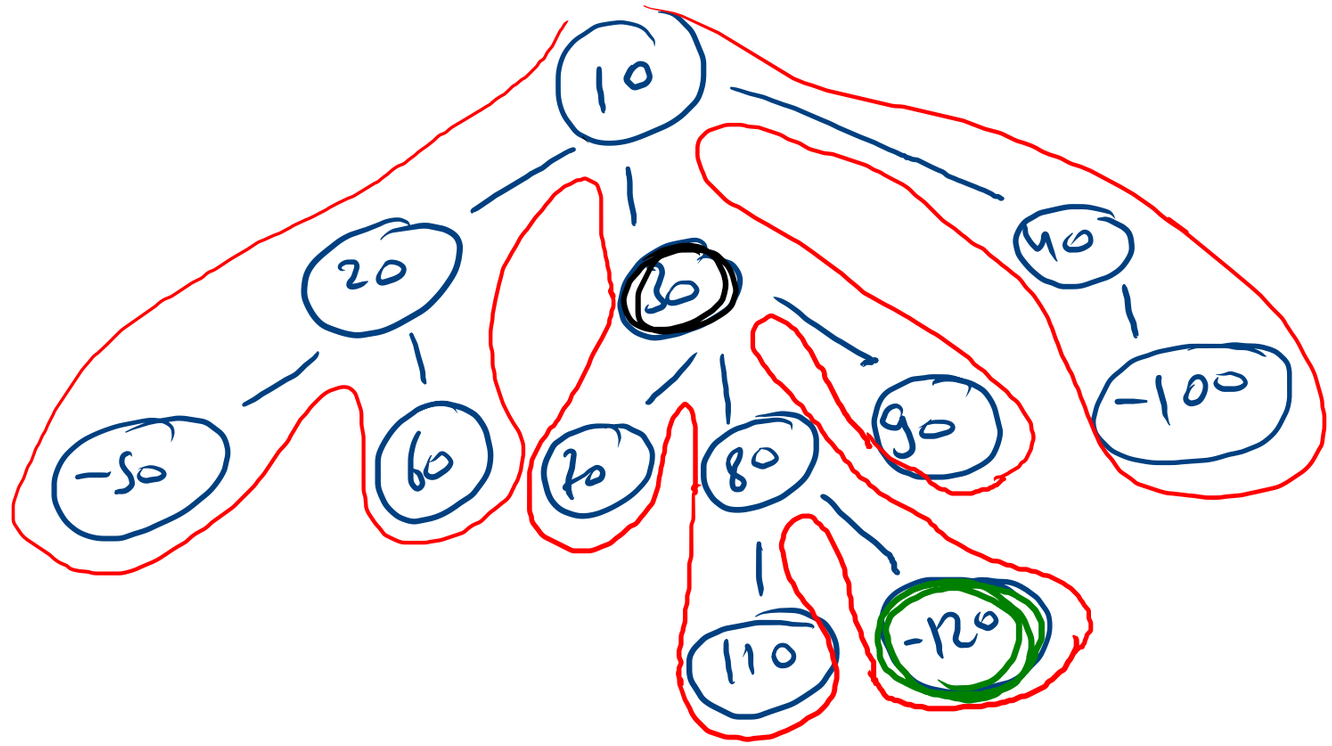
~~-∞~~ ~~70~~ - ∞

Precd traversal

data = -120

Pred → 110

Succ → 90



data = 30

pre = 60

suc = 70

d

travel & change

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

Sample Output

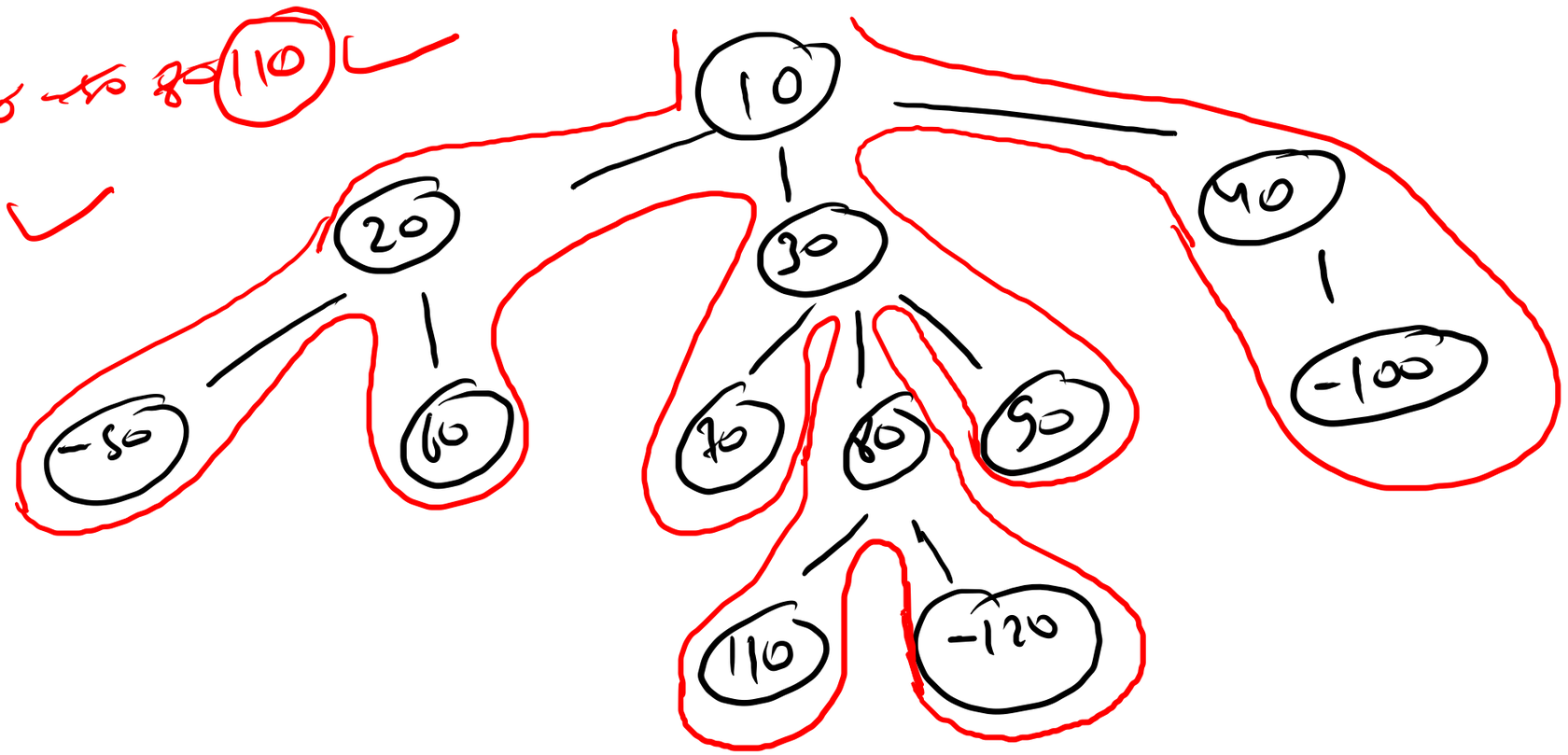
Predecessor = 110
Successor = 90

data → -120

predecessor → ~~0~~ ~~10~~ ~~20~~ ~~30~~ 110

successor → ~~0~~ 90

int stat → 0 ~~1~~ 2



0

pre = node

node.data == data

stat = 1

stat = 1

succ = node

status = 2

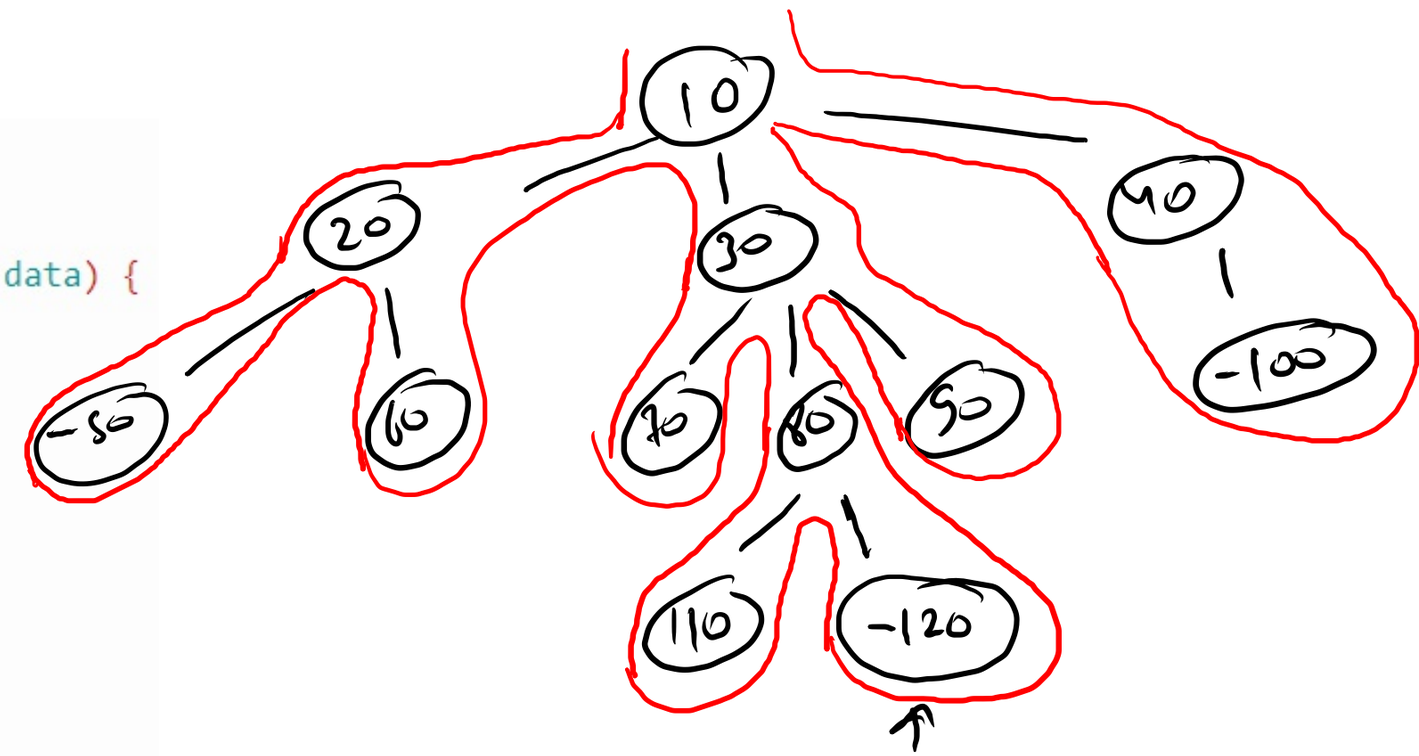
Pre → ~~*~~ ~~10~~ 20 ~~-50~~ ~~60~~ ~~30~~ ~~70~~ ~~80~~ 110

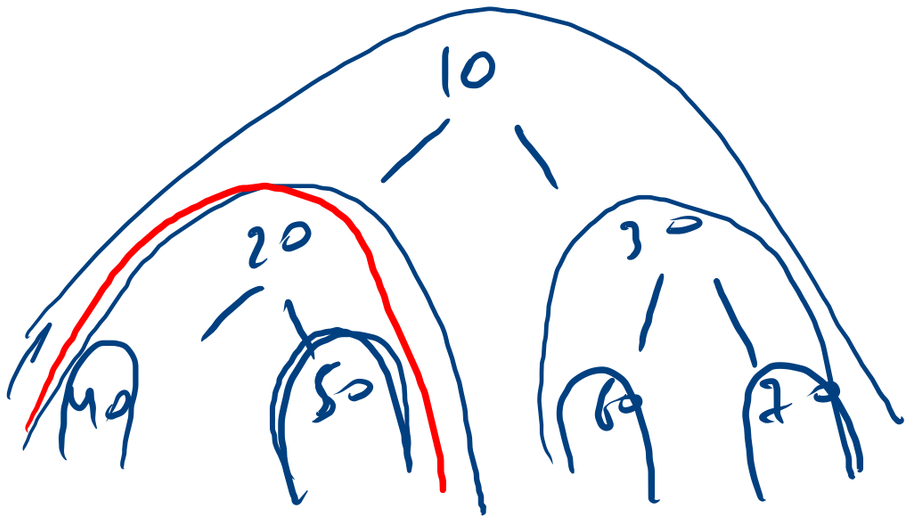
Suc → * 90

status → ~~0~~ + 2

data → -120

```
static Node predecessor;
static Node successor;
static int status = 0;
public static void predecessorAndSuccessor(Node node, int data) {
    if(node.data == data){
        status = 1;
    }else if(status == 0){
        predecessor = node;
    }else if(status == 1){
        successor = node;
        status = 2;
    }

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





Total tree

← 2 → 7

sub arrs

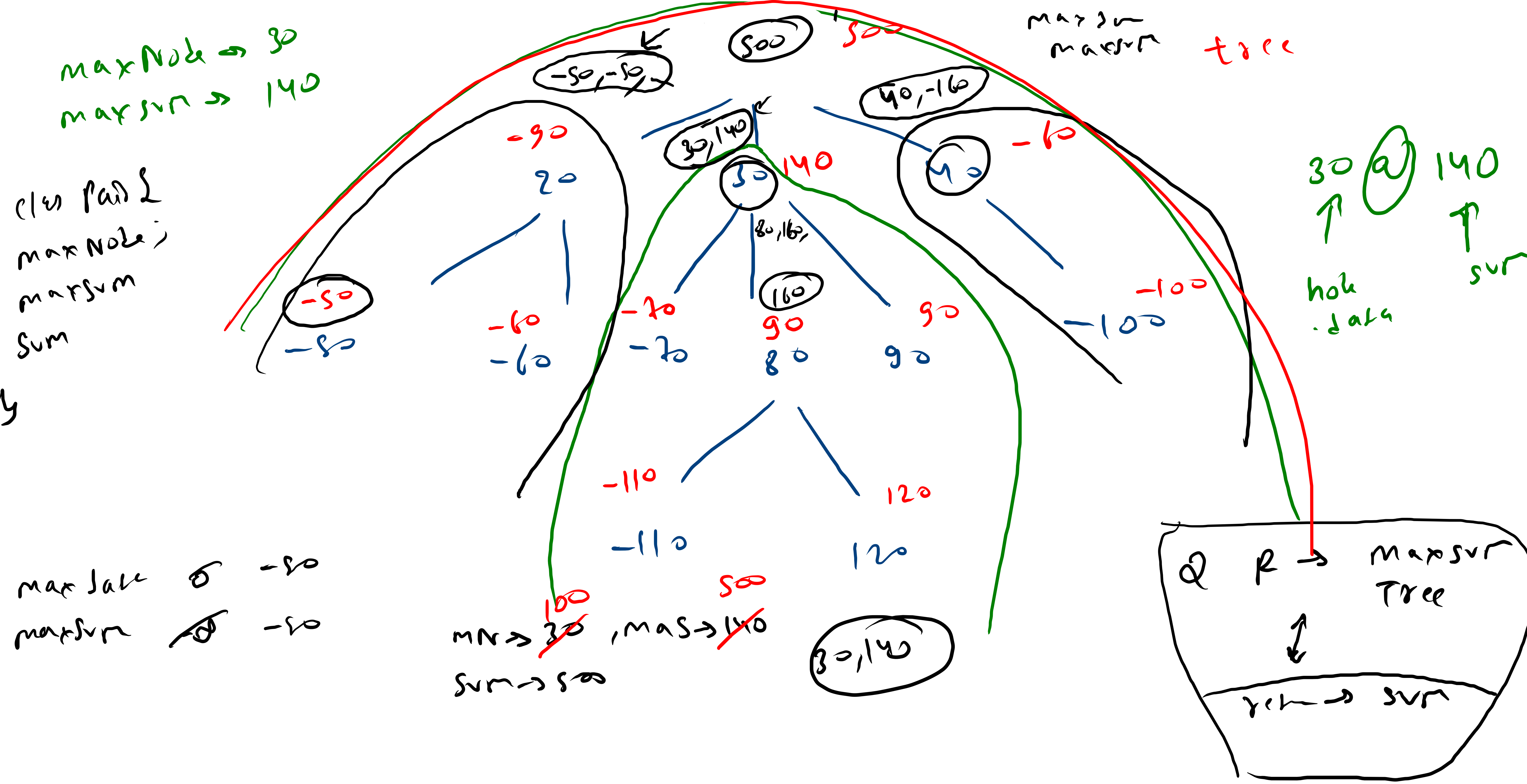
[10 20 30]

10 20

20 30

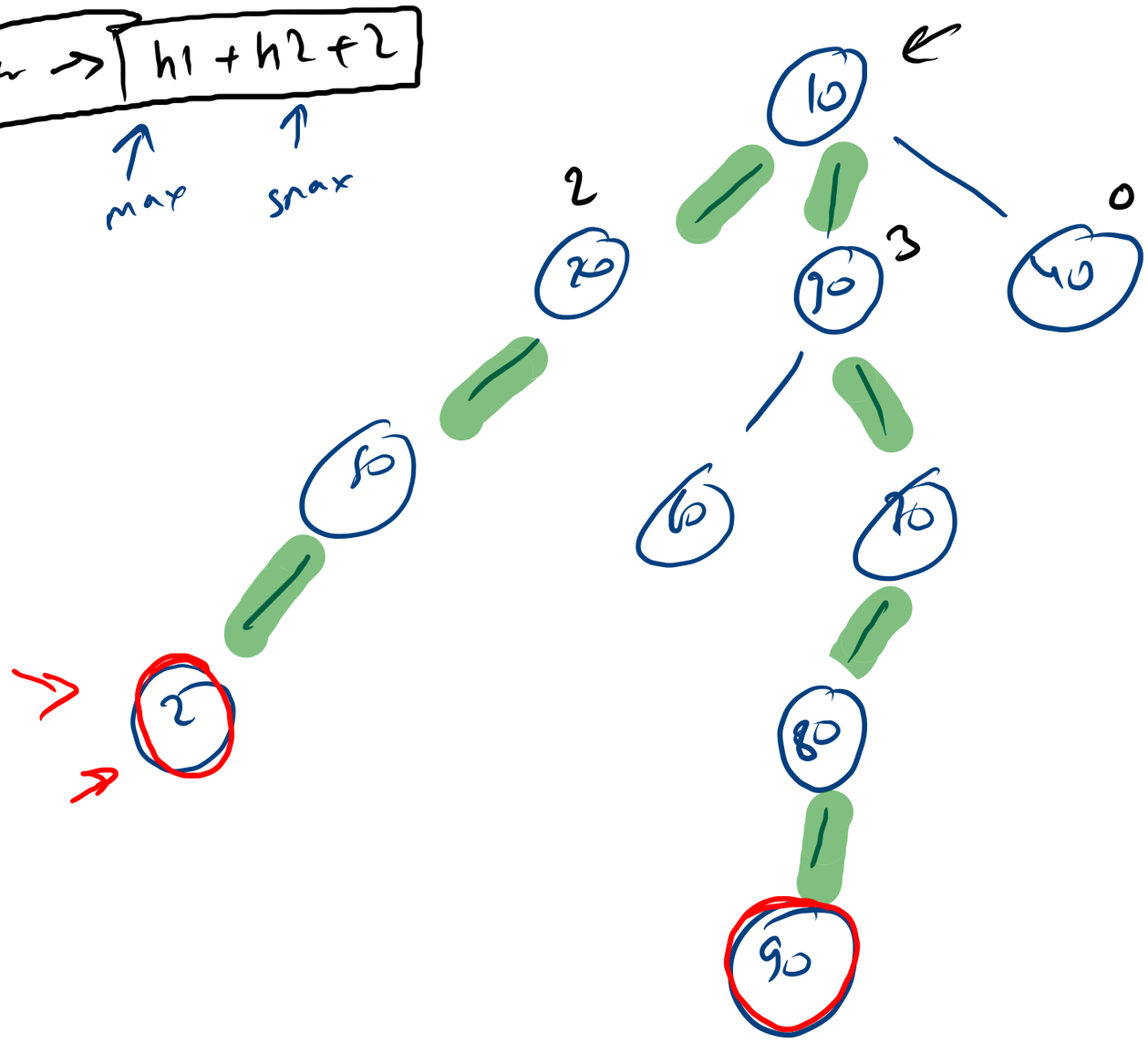
20

30


maxNode → 30
maxsum → 140
class Pair {
maxNode;
maxsum
sum
}
maxsum
maxsum
tree
500
-50, -50
30, 140
40, -160
80, 160
160
30 @ 140
node .data
sum
max node
maxsum
MN → 30 100, MAS → 140 500
sum → 500
30, 140
Q R → maxsum Tree
ret → sum

diameter → h1 + h2 + 2

max smax



diameter → CT

max

N1 — N2

2 — 80

6

return height

maxdia

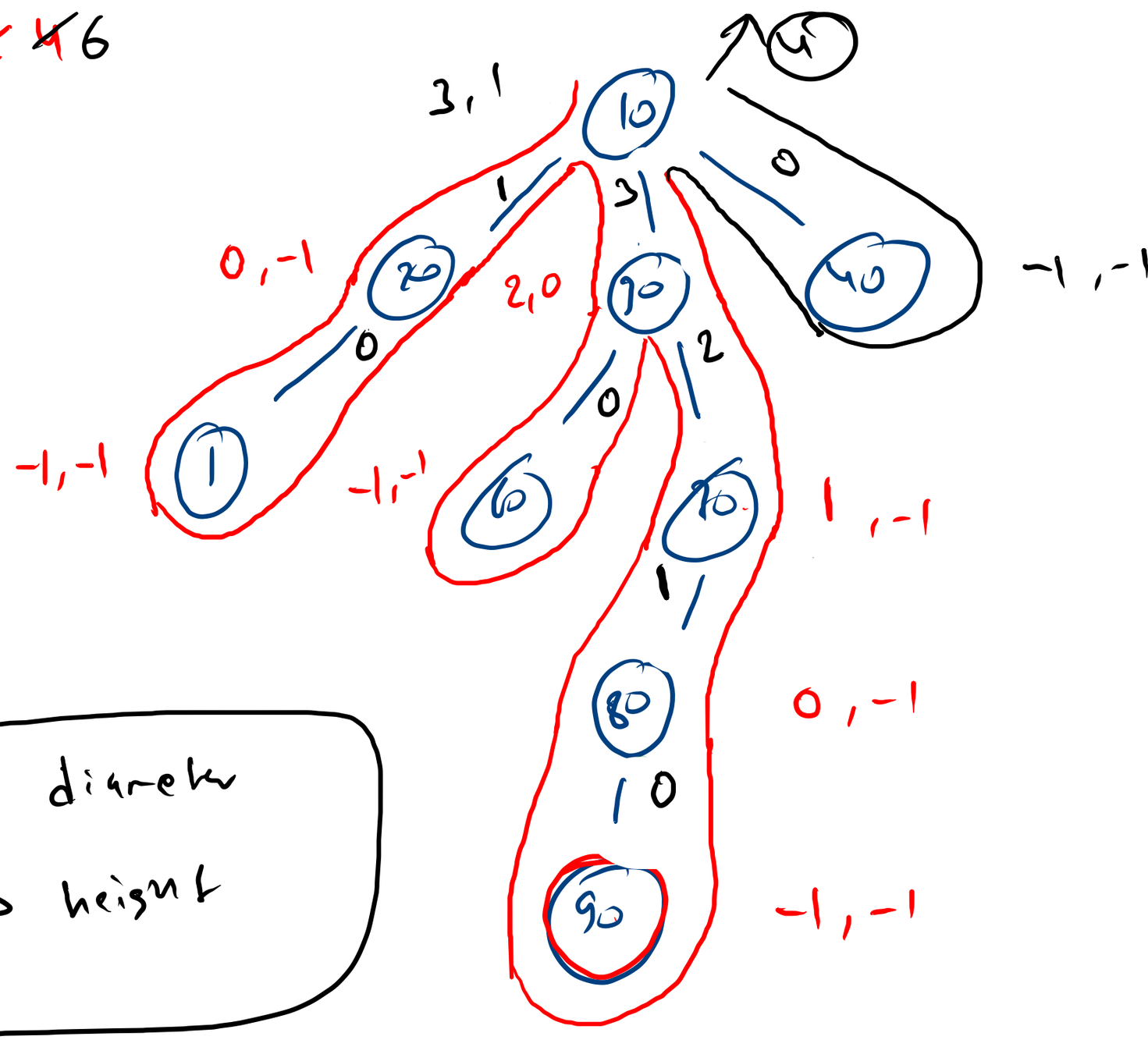
max

set

maxdia → ~~0~~ ~~1~~ ~~2~~ ~~4~~ 6

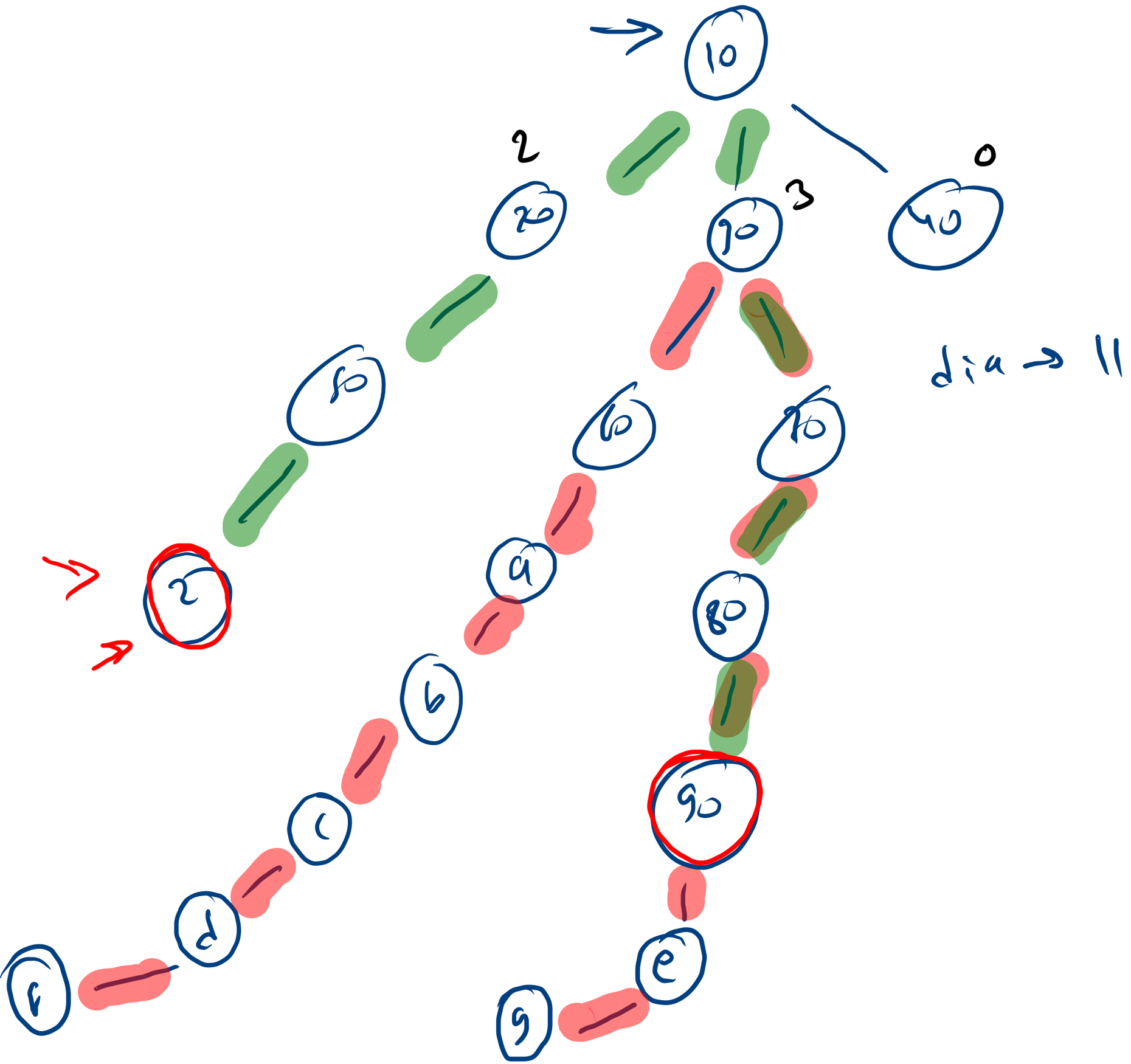
```java
static int maxDia=0;
public static int getDiameter(Node node){
    int h1=-1;// max
    int h2=-1;// smax
    for(Node child: node.children){
        int height = getDiameter(child);
        if(height > h1){
            h2 = h1;
            h1 = height;
        }else if(height > h2){
            h2 = height;
        }
    }
    int dia = h1+h2+2;
    if(dia > maxDia){
        maxDia = dia;
    }
    return h1+1;
}
```

3+1+2 = 6



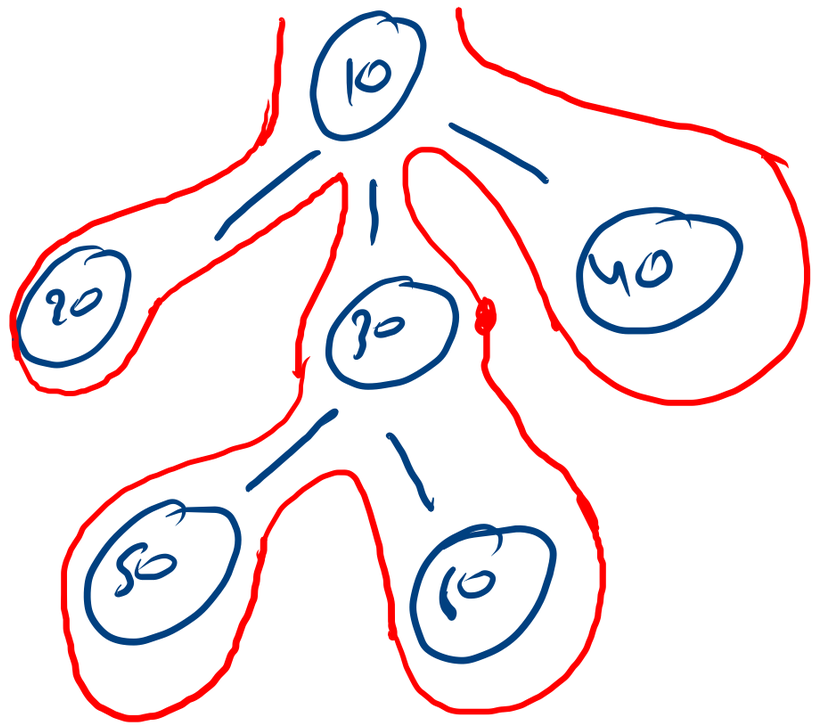
Q → diameter

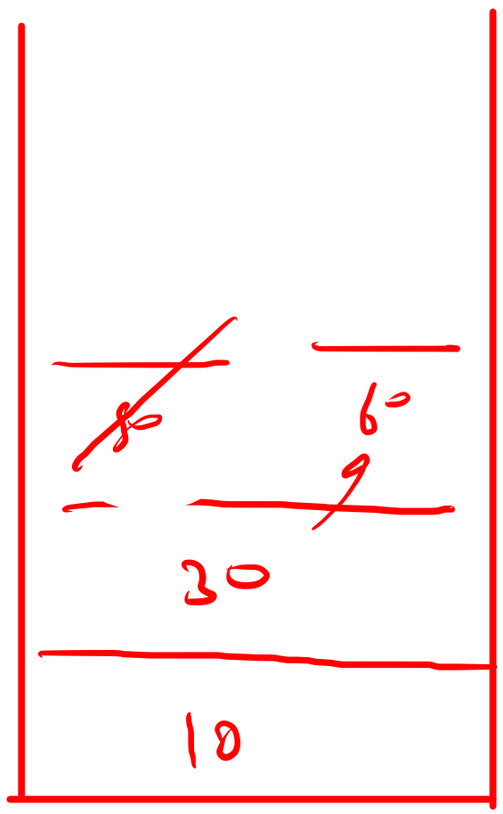
recr → height


dia → 7
dia → 11
2
3
0

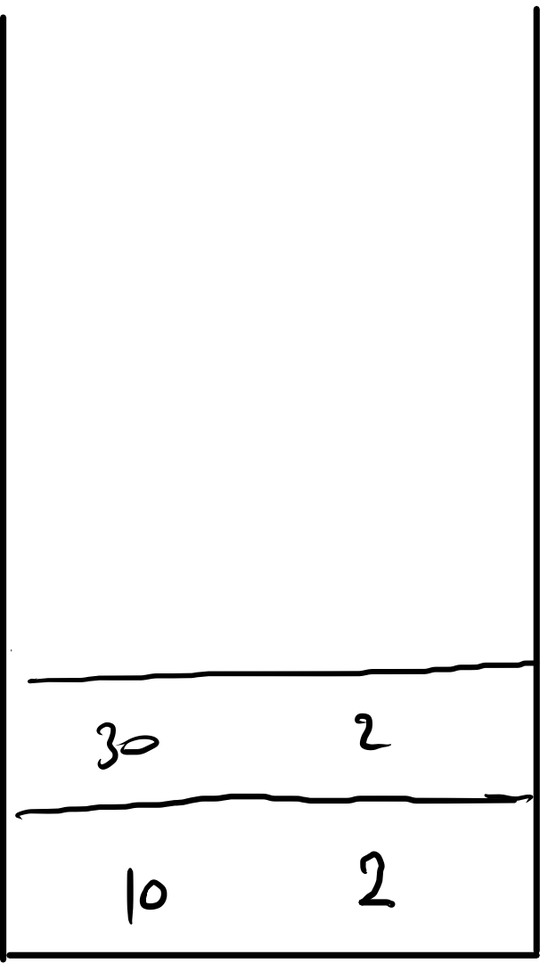
Pre 10 20 30 80 60 40

Post 20 80 60 30 40 10

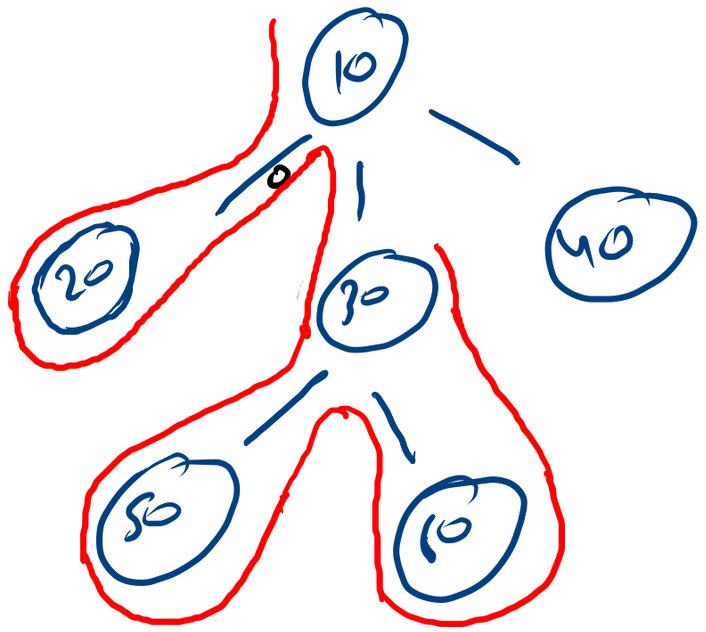




Stack

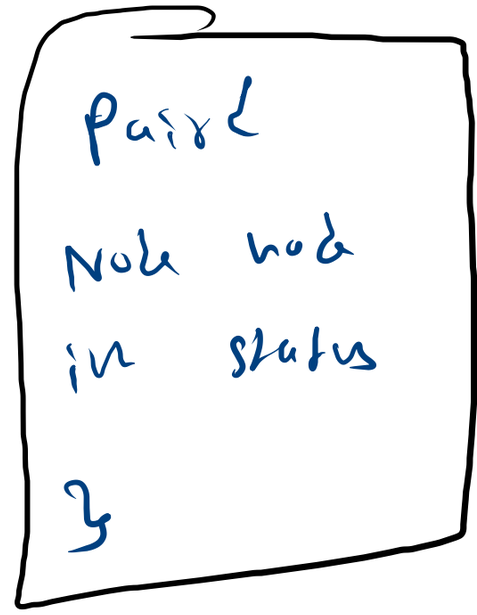


Stack



pre 10 20 30 50 60

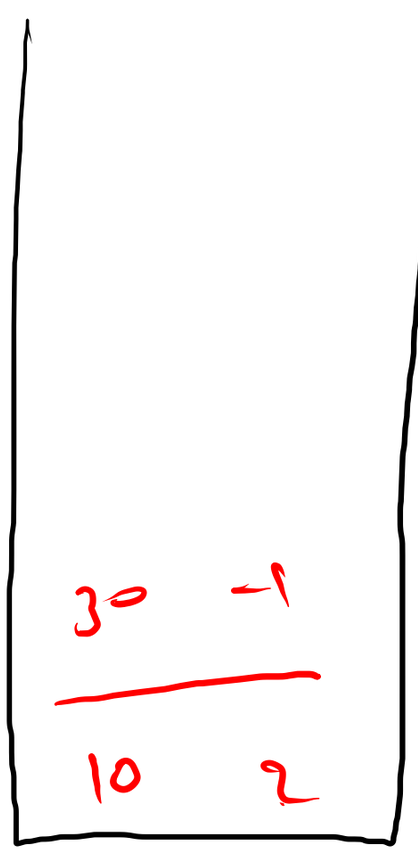
pop 20 50 60 30

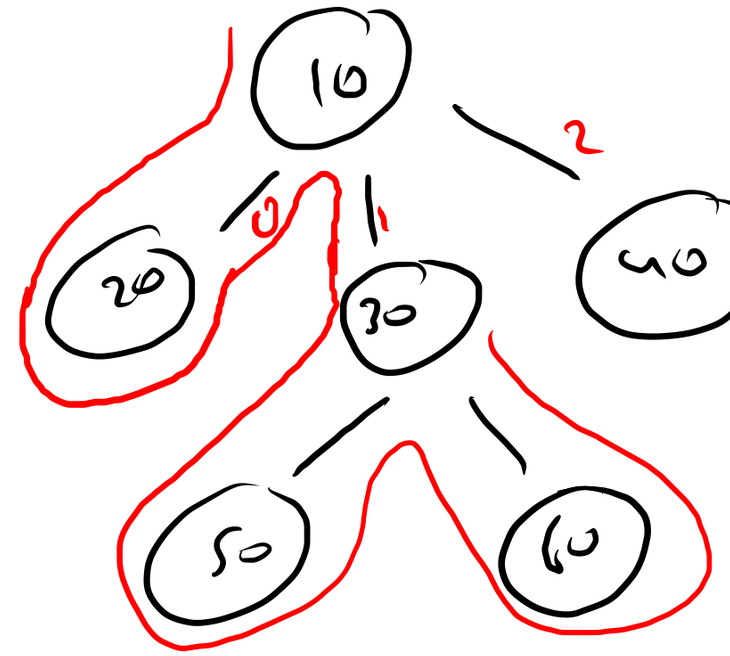


-1 → pre

child.size → done

```
st.push(new Pair(node, -1));
StringBuilder pre = new StringBuilder();
StringBuilder post = new StringBuilder();


while(st.size() > 0){

    Pair p = st.peek();

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

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





ple 10 20

pre 20