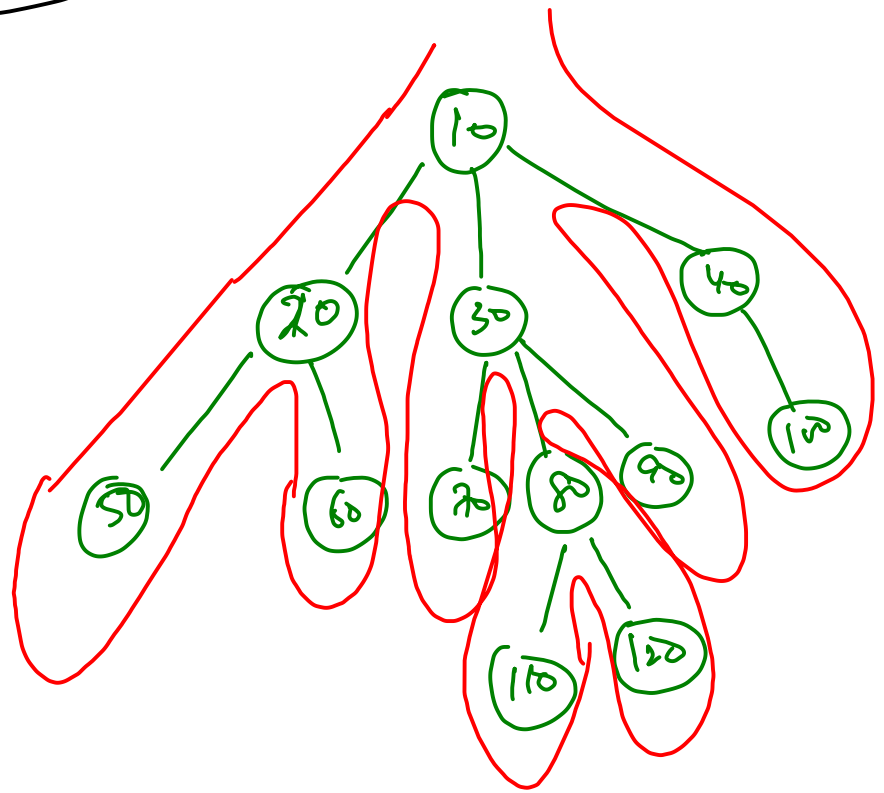
MultiSolver

in one iteration

- Size
- Sum
- Max
- Min



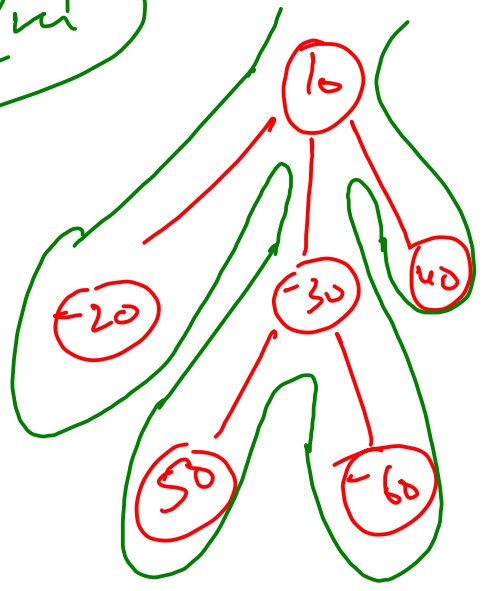
Size = ~~0~~ 1 ~~2~~ ~~3~~ ~~4~~

Sum = 0 + 10 + 20 + 30 + 40

Max = ~~-∞~~ ~~10~~ ~~20~~ ~~30~~ 60

Min = ~~+∞~~ 10

Multisolver



2-3

O(n)

O(1)

O(n)

```
static int size , sum , max , min;
public static void multiSolver(Node node){
    size++;
    sum += node.data;
    max = Math.max(max,node.data);
    min = Math.min(min,node.data);

    for(Node child : node.children){
        multiSolver(child);
    }
}
```

```
size = 0;
sum = 0;
max = Integer.MIN_VALUE;
min = Integer.MAX_VALUE;
multiSolver(root);
System.out.println("Size : "+size);
System.out.println("sum : "+sum);
System.out.println("max : "+max);
System.out.println("min : "+min);
```

Size = 6

Sum = 10 - 20 - 30 + 50 - 60 + 40

Max = 10 50

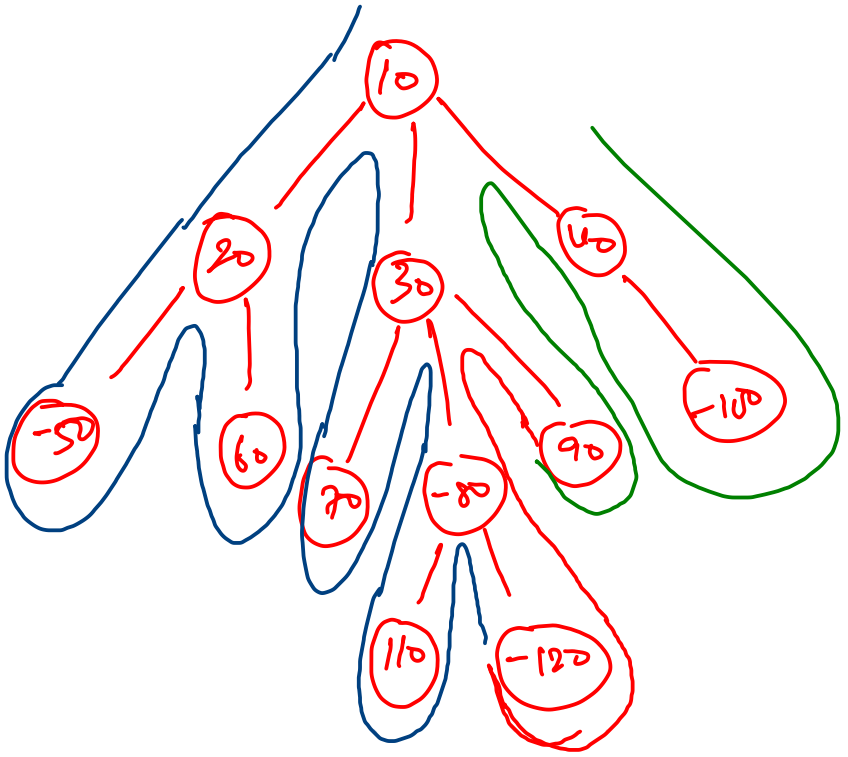
Min = 10 -10 -30 -60

D: -120

0 : not found

1 : found

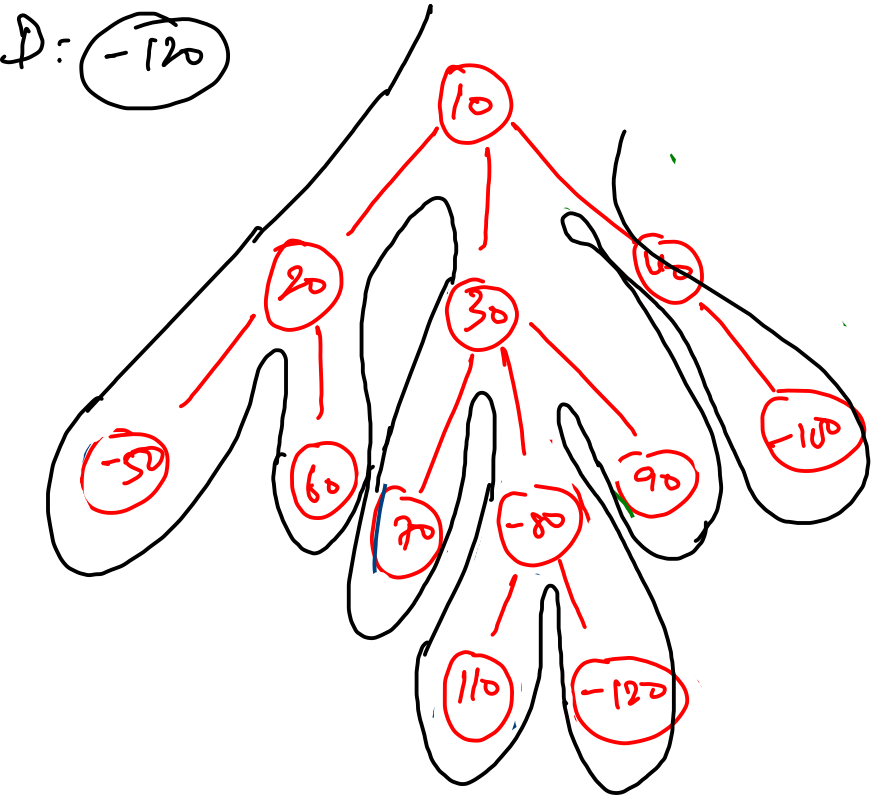
2 : Successor set



state = ~~1~~ 2

predecessor = ~~null~~ ~~10~~ ~~20~~ ~~-50~~ ~~10~~ ~~30~~ ~~70~~ ~~-80~~ 110

successor = ~~null~~ 90



pre = null 10 20 -30 60 30 70 -80 110

suc = null 90

stack = φ X 2

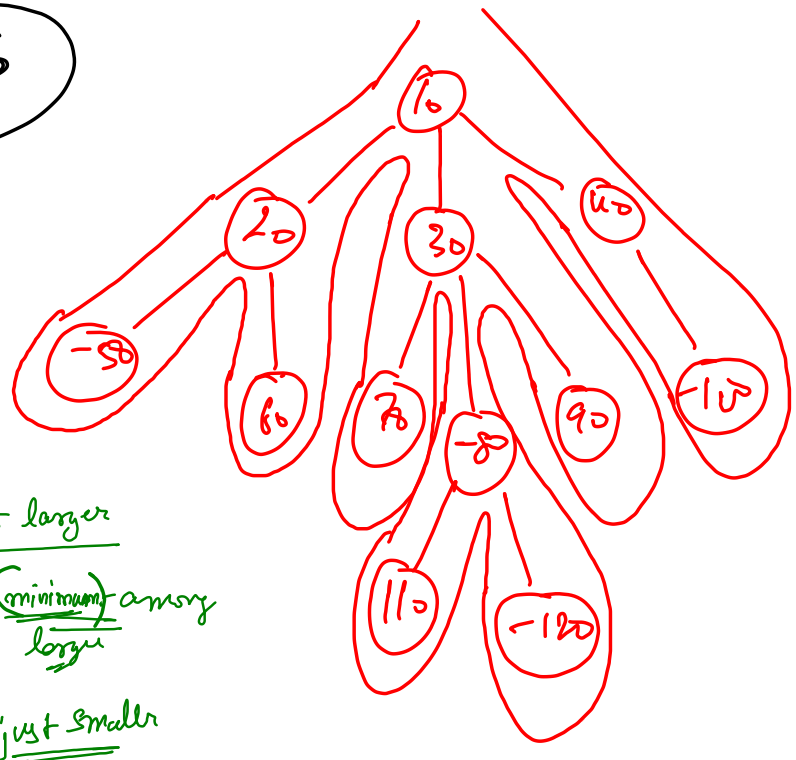
```
public static void predecessorAndSuccessor(Node node, int data) {
    if(state == 0){
        if(node.data == data){
            state++;
        }else{
            predecessor = node;
        }
    }else if(state == 1){
        successor = node;
        state++;
    }

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

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

Data = 70



Ceil → just larger (minimum among larger)

floor → just smaller → maximum among smaller

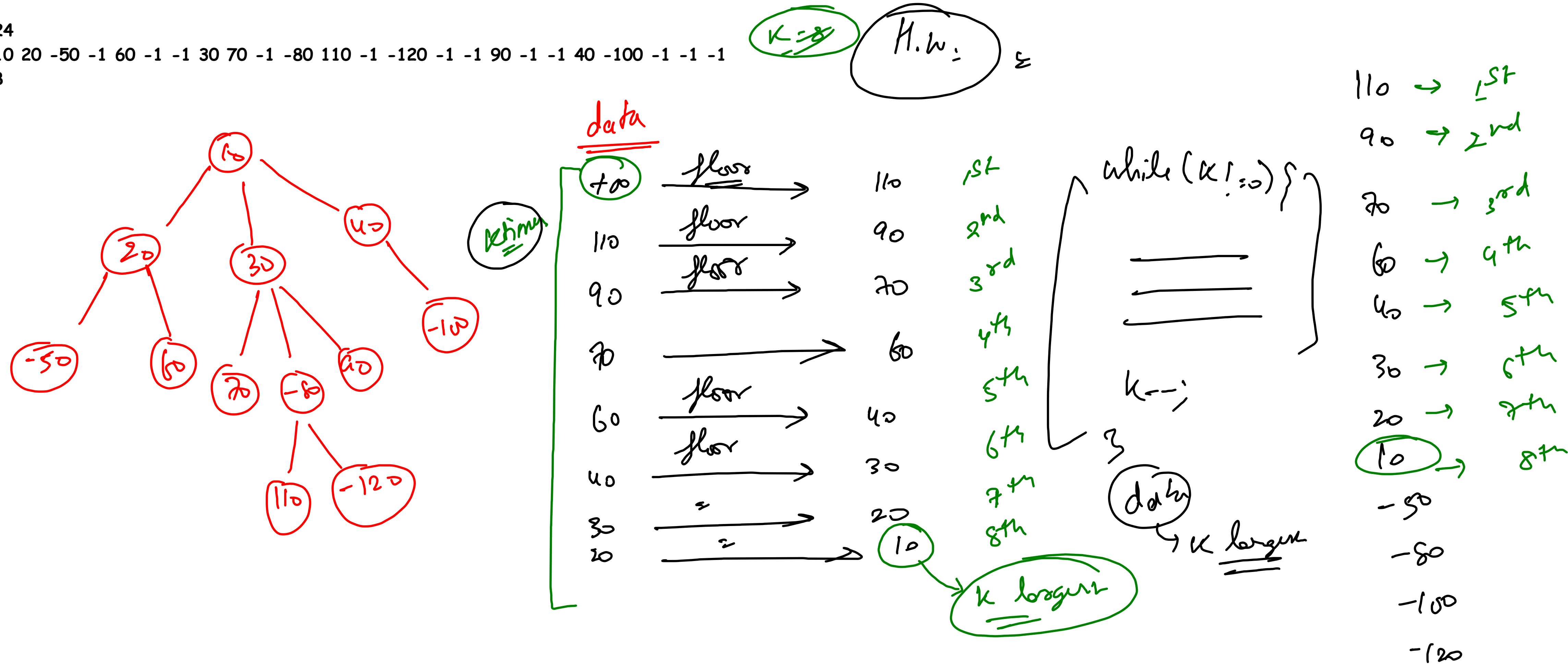
```
if (node.data < data){
    if (node.data > floor) {
        floor = node.data;
    }
}

if (node.data > data){
    if (node.data < ceil){
        ceil = node.data;
    }
}
```

✓ ceil = ~~+∞~~ ~~110~~ 90

✓ floor = ~~-∞~~ ~~10~~ ~~20~~ 60

-120
-100
-80
-50
10
20
30
40
60 ← floor
80
90 ← Ceil
110

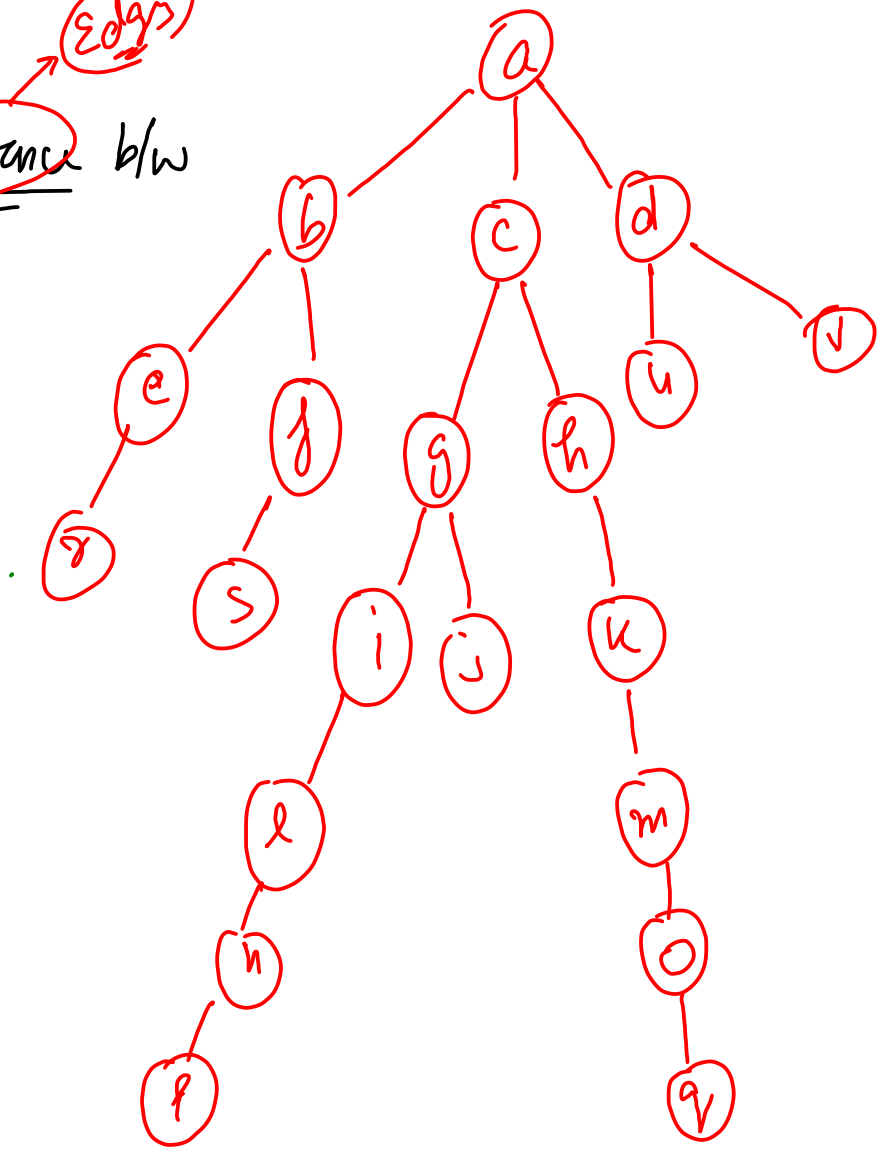
24
10 20 -50 -1 60 -1 -1 30 70 -1 -80 110 -1 -120 -1 -1 90 -1 -1 40 -100 -1 -1 -1
8



Diameter

↳ Max Distance b/w two nodes

Edges



r - j ⇒ 6

r - p ⇒ 9

r - q ⇒ 7

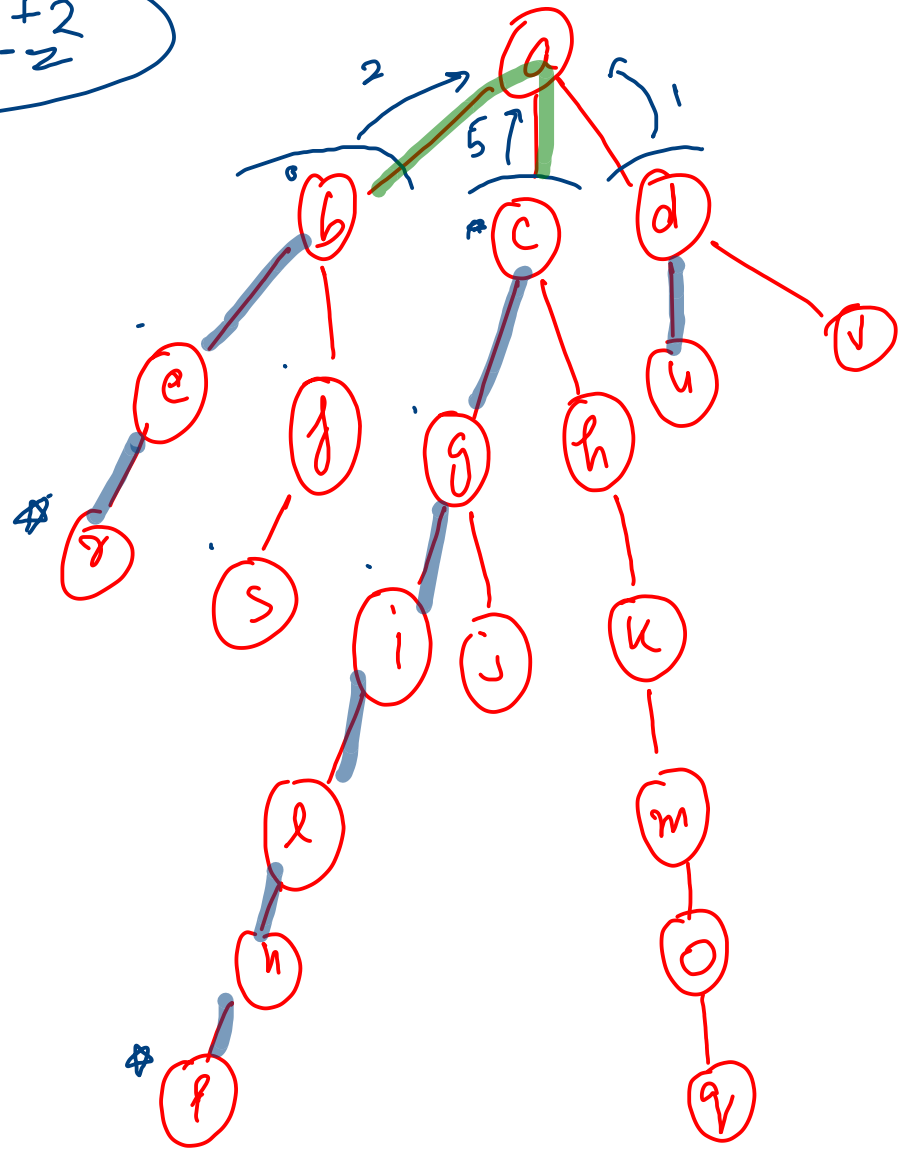
p - q ⇒ 10

p - u ⇒ 8

v - s ⇒ 5

Diameter

diaMode = $\frac{lh + slh}{2} + 2$



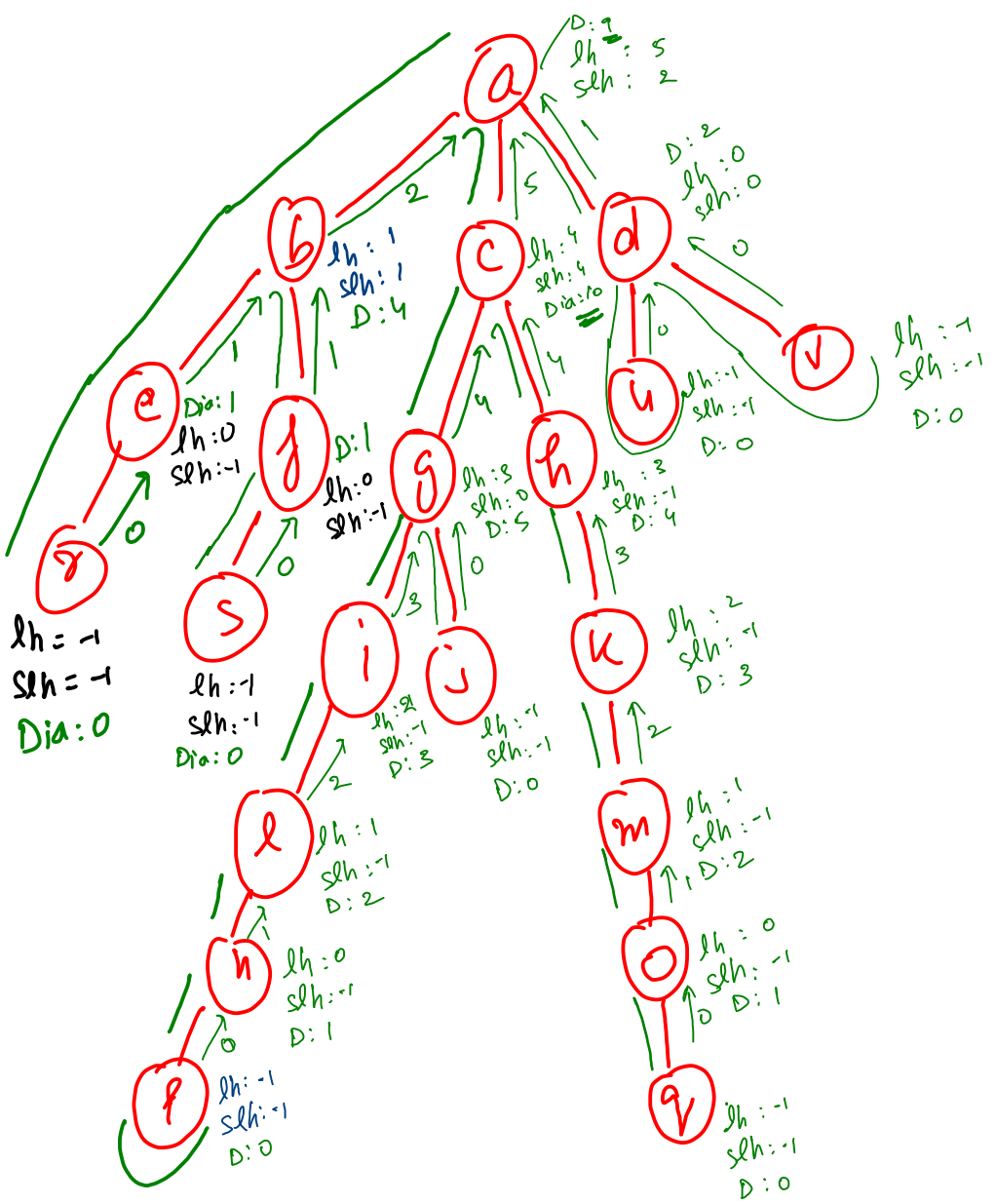
Travel & change

→ return → height

Dia of Tree → 0 / 1 / 0

Dia of Node

→ lh + slh + 2



lh = 10 slh = 5 cht = 4

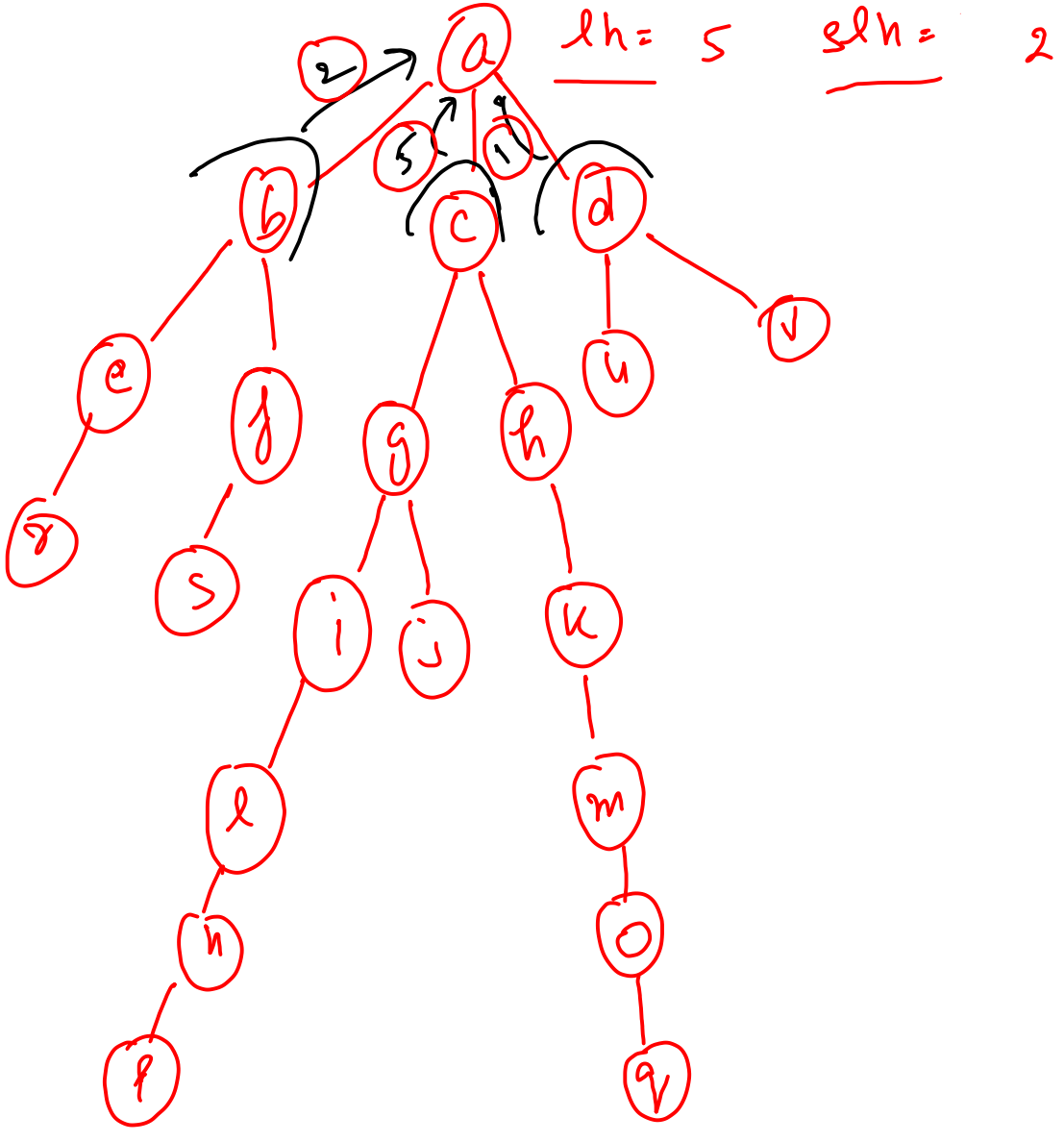
lh = 10 slh = ~~5~~ 6 cht = 6

slh = cht

lh = ~~10~~ 11 slh = ~~5~~ 10 cht = 11

slh = lh;

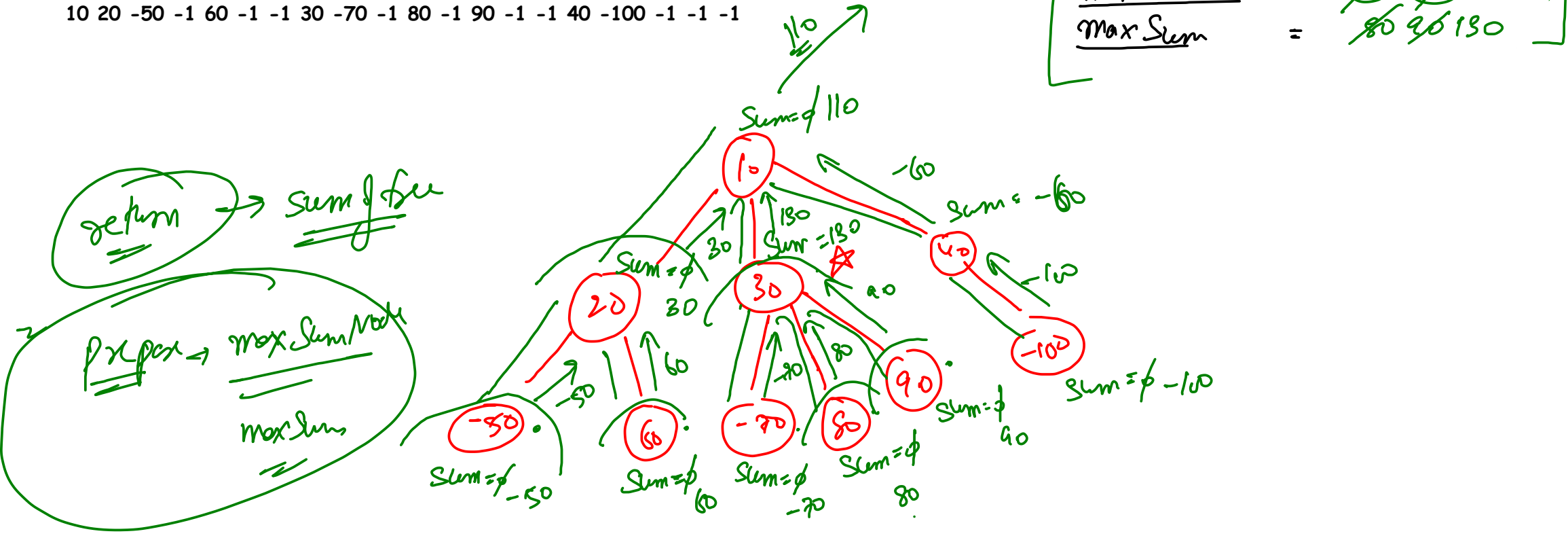
lh = cht;

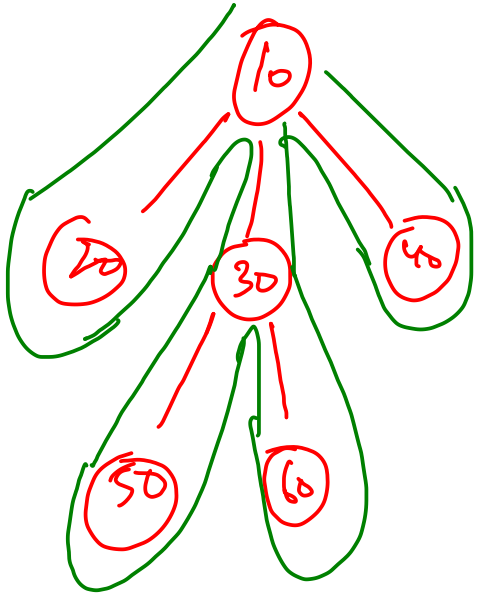



```java
static int diaOfTree;
public static int diameter(Node node){
    // height

    int lh = -1 , slh = -1;

    for(Node child : node.children){
        int cht = diameter(child);

        if(cht > lh){
            slh = lh;
            lh = cht;
        }else if(cht > slh){
            slh = cht;
        }
    }

    int diaOfNode = lh+slh+2;
    if(diaOfNode > diaOfTree){
        diaOfTree = diaOfNode;
    }

    return lh+1;
}
```

20

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



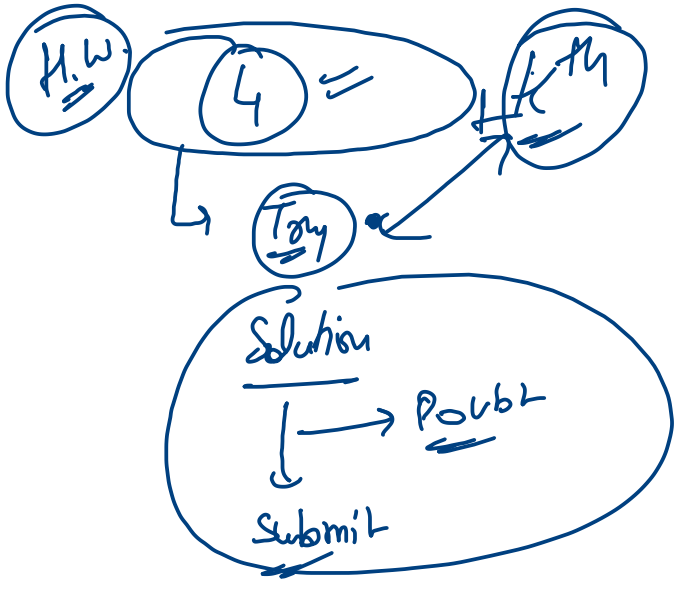


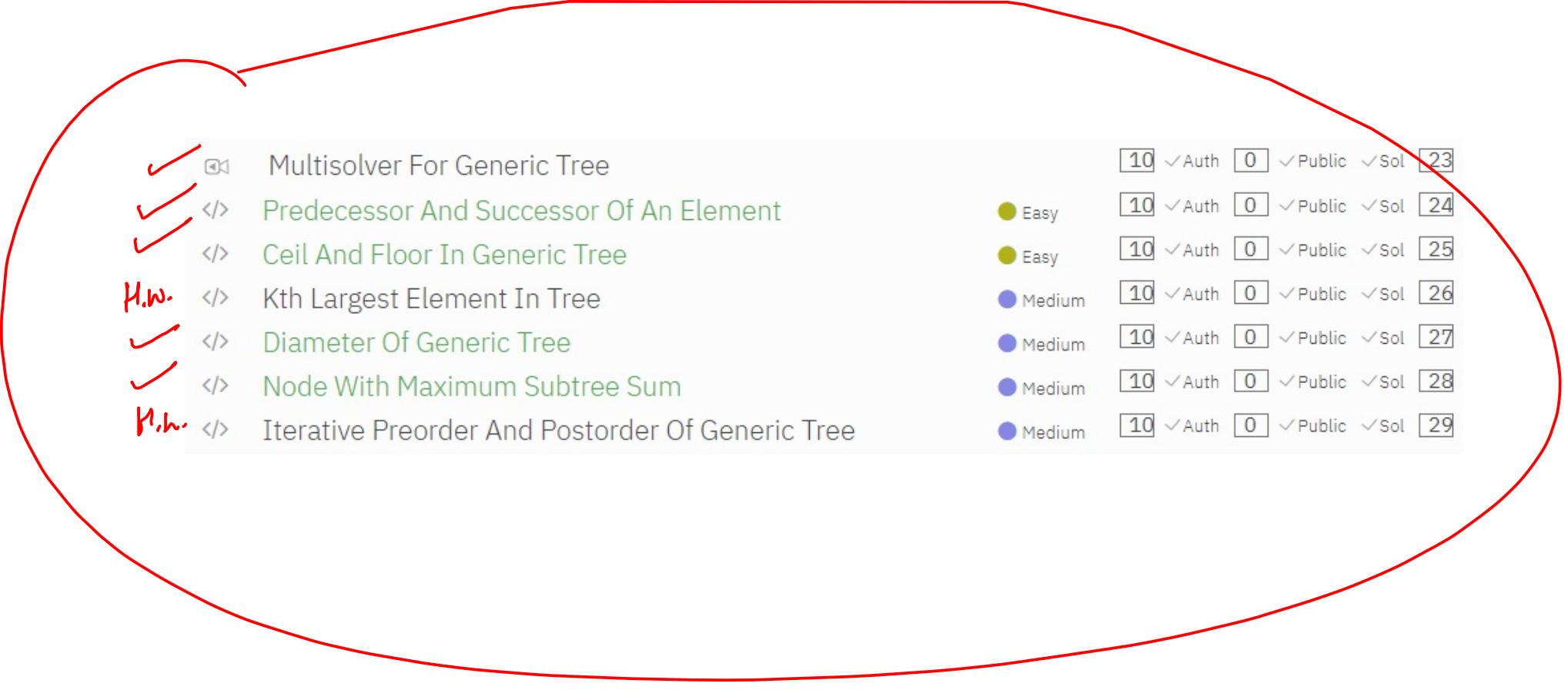
Pre	Post
✓ 10	✓ 20
✓ 20	✓ 50
✓ 30	✓ 60
✓ 50	✓ 30
✓ 60	✓ 40
✓ 40	✓ 10

State

-1 => pre, state++

size → post, pop




Multisolver For Generic Tree
10 Auth 0 Public Sol 23
Predecessor And Successor Of An Element
Easy
10 Auth 0 Public Sol 24
Ceil And Floor In Generic Tree
Easy
10 Auth 0 Public Sol 25
H.W.
Kth Largest Element In Tree
Medium
10 Auth 0 Public Sol 26
Diameter Of Generic Tree
Medium
10 Auth 0 Public Sol 27
Node With Maximum Subtree Sum
Medium
10 Auth 0 Public Sol 28
H.W.
Iterative Preorder And Postorder Of Generic Tree
Medium
10 Auth 0 Public Sol 29