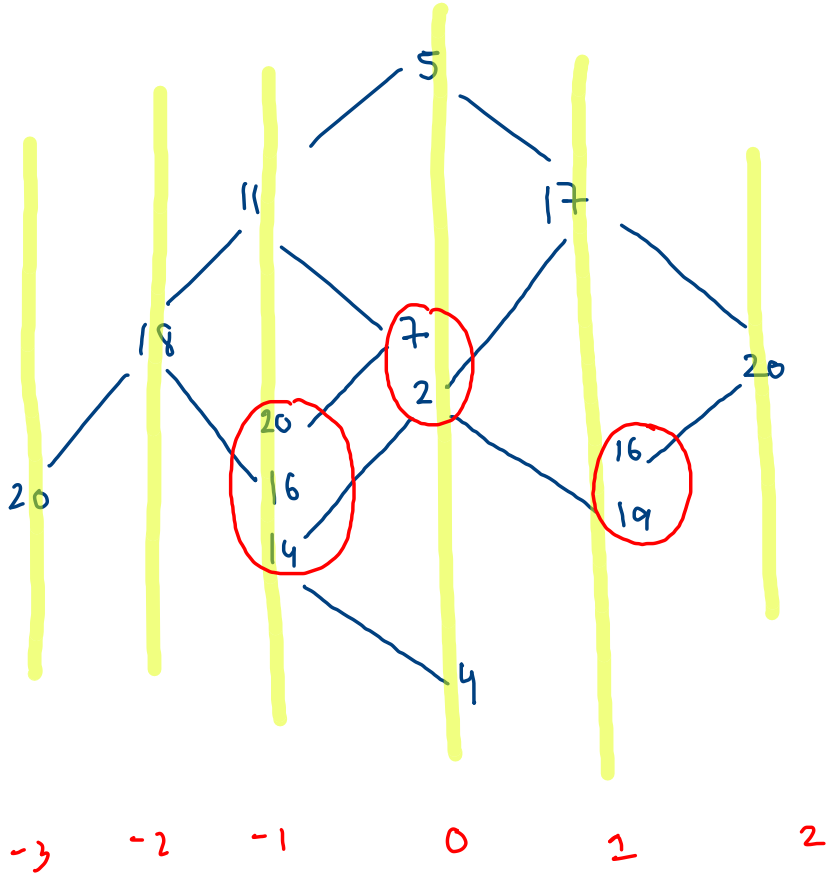


-3 -> 20

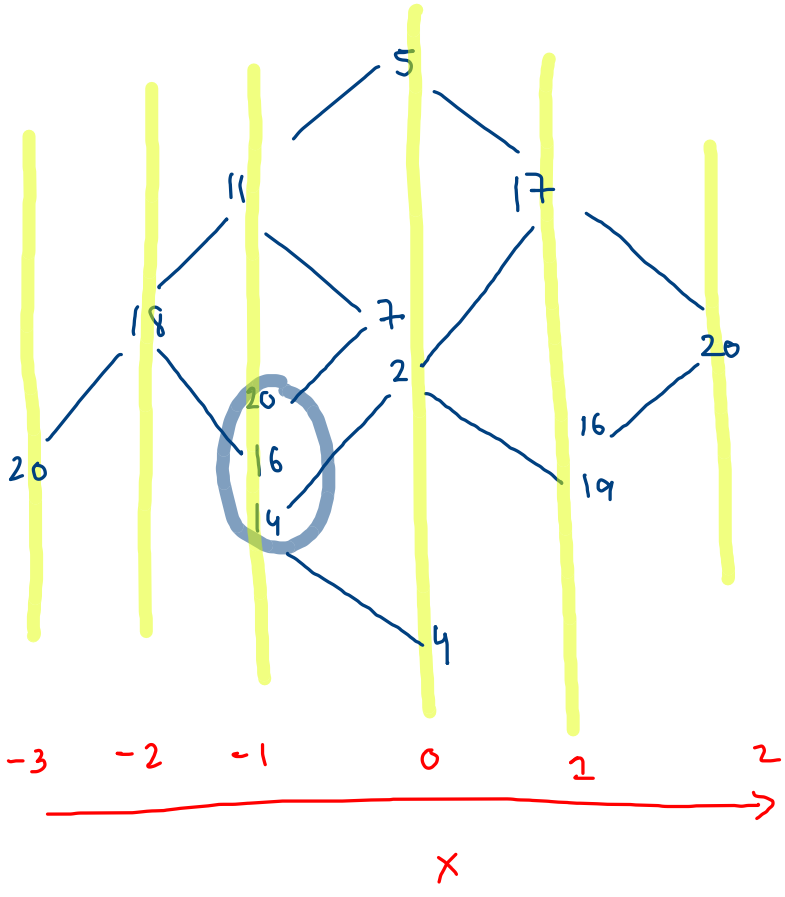
-2 -> 18

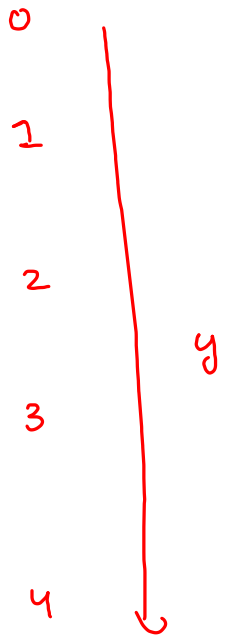
-1 -> 11 14 16 20

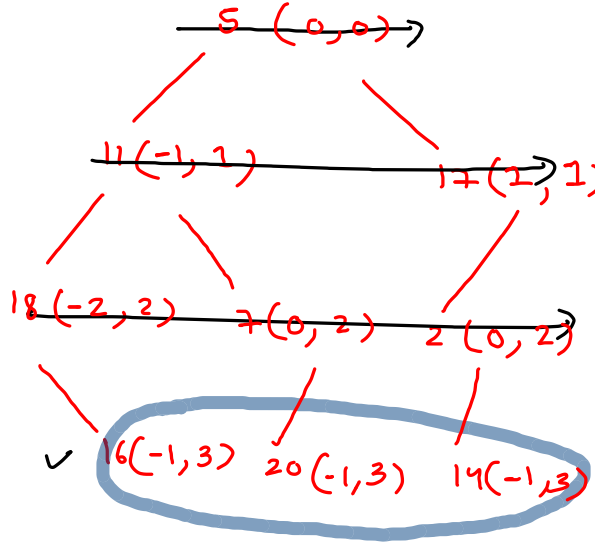
0 -> 5 2 7 4

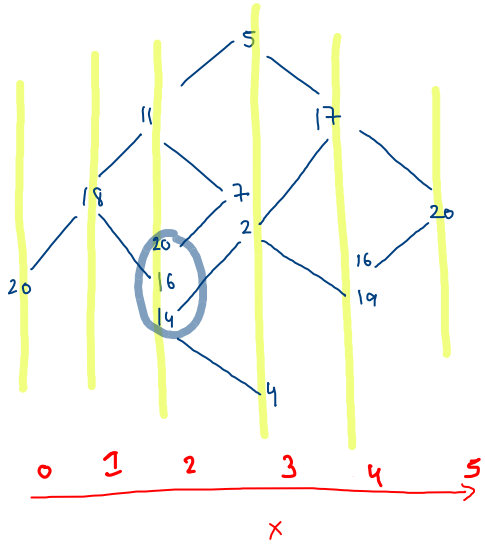
1 -> 17 16 19

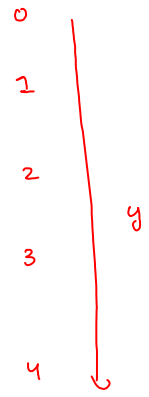
2 -> 20


5
11
17
18
7
2
20
20
16
14
16
19
20
4
-3
-2
-1
0
1
2
X


0
1
2
3
4
y


5 (0,0)
11(-1,1)
17(1,1)
18(-2,2)
7(0,2)
2(0,2)
16(-1,3)
20(-1,3)
19(-1,3)





1st priority -> y

2nd priority -> x

3rd priority -> val (node, x, y)

pan, x, y

lc rc

lcn, x-1, y+1 rcn, x+1, y+1

0 -> 20

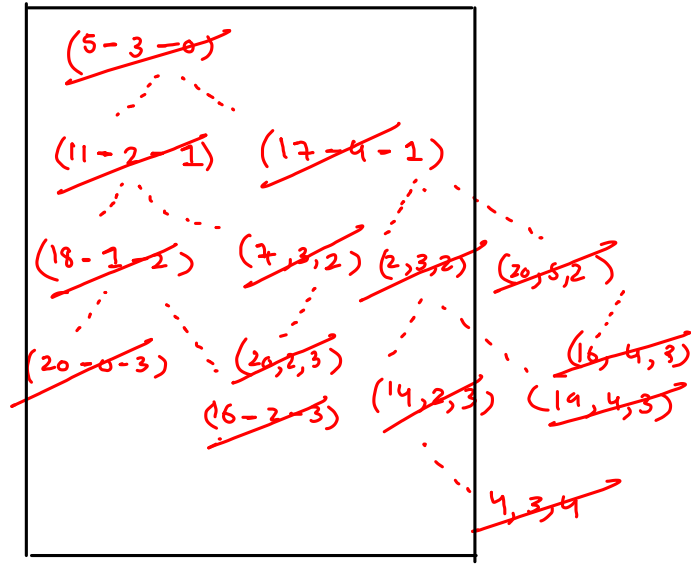
1 -> 18

2 -> 11, 14, 16, 20

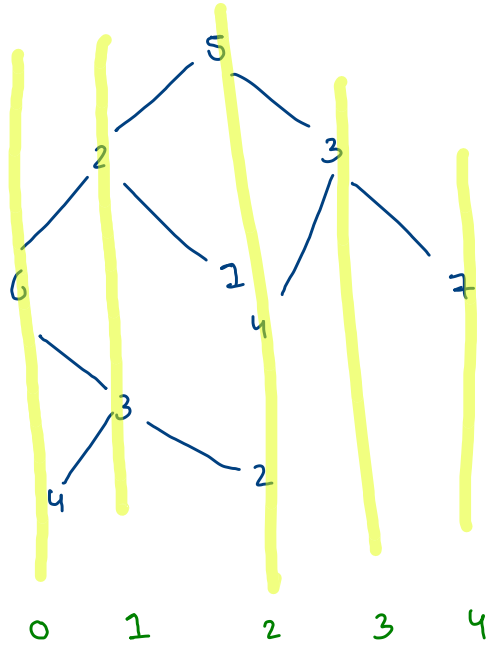
3 -> 5, 2, 7, 4

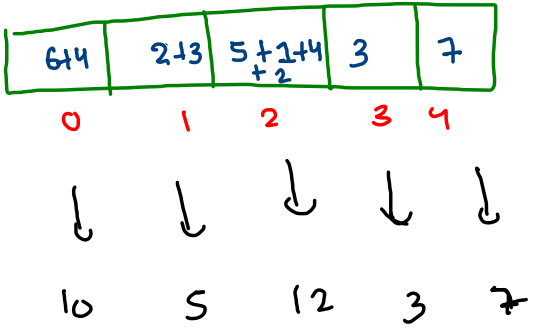
4 -> 17, 16, 19

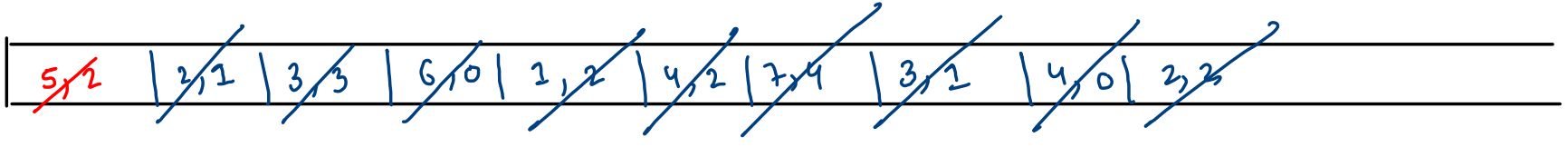
6 -> 20

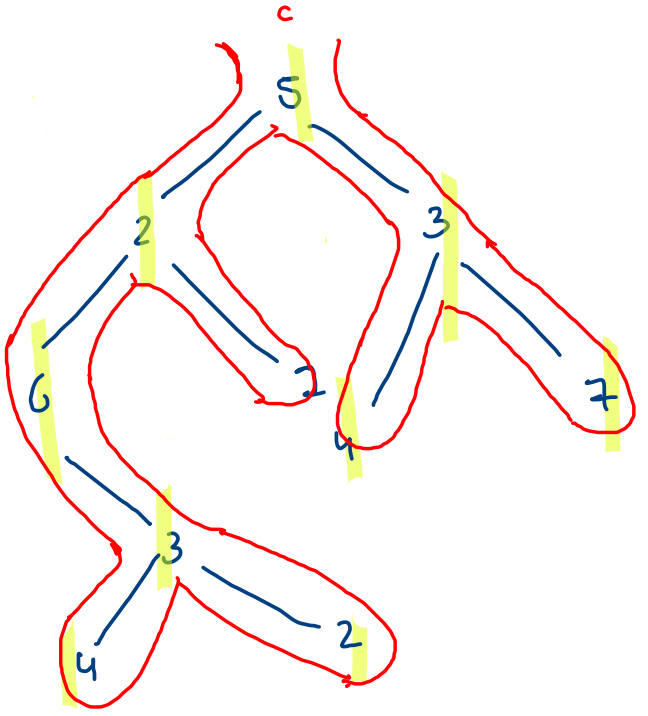


Vertical order sum









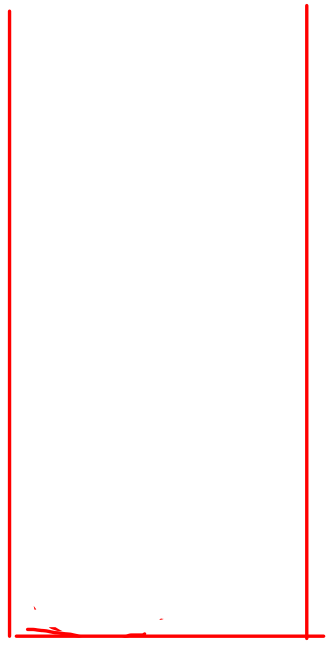
VOS

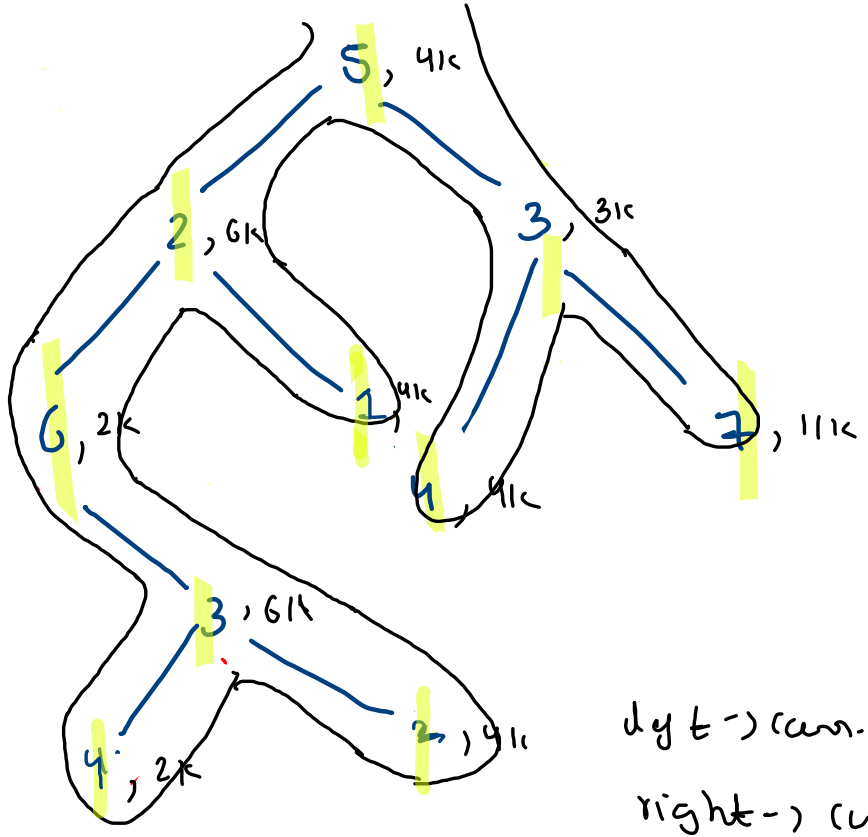
ALX

LL ∪ (sdj)

VOS

without HM & width





(i) Treenode -> node

↓ contribute

cur in LL

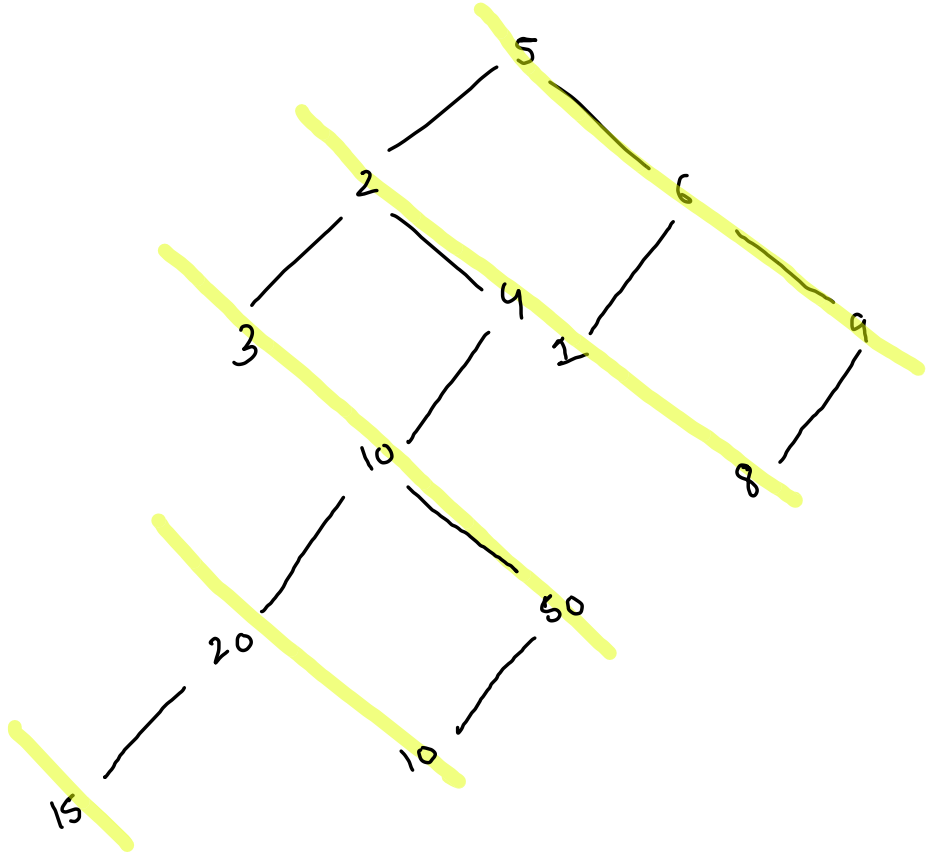
(ii) node.dyt

!= null

dyt -> curr.prev

right -> curr.next

DOS



20	15	63	30	15
0	1	2	3	4

5 6 9 → 20

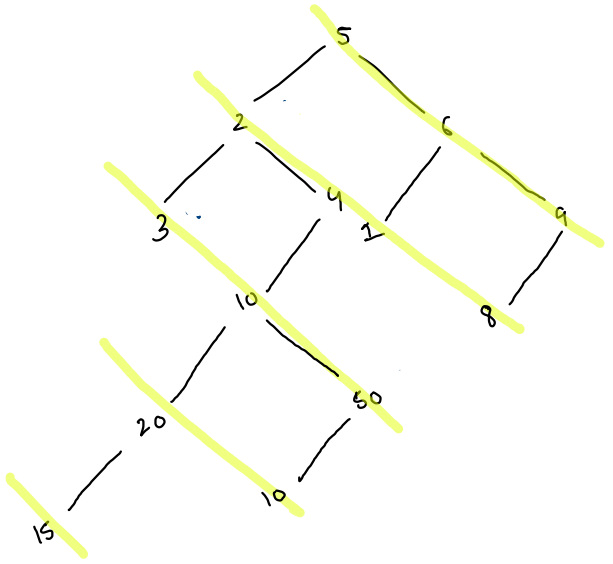
2 4 1 8 → 15

3 10 50 → 63

20 10 → 30

15 → 15

DOS ↙

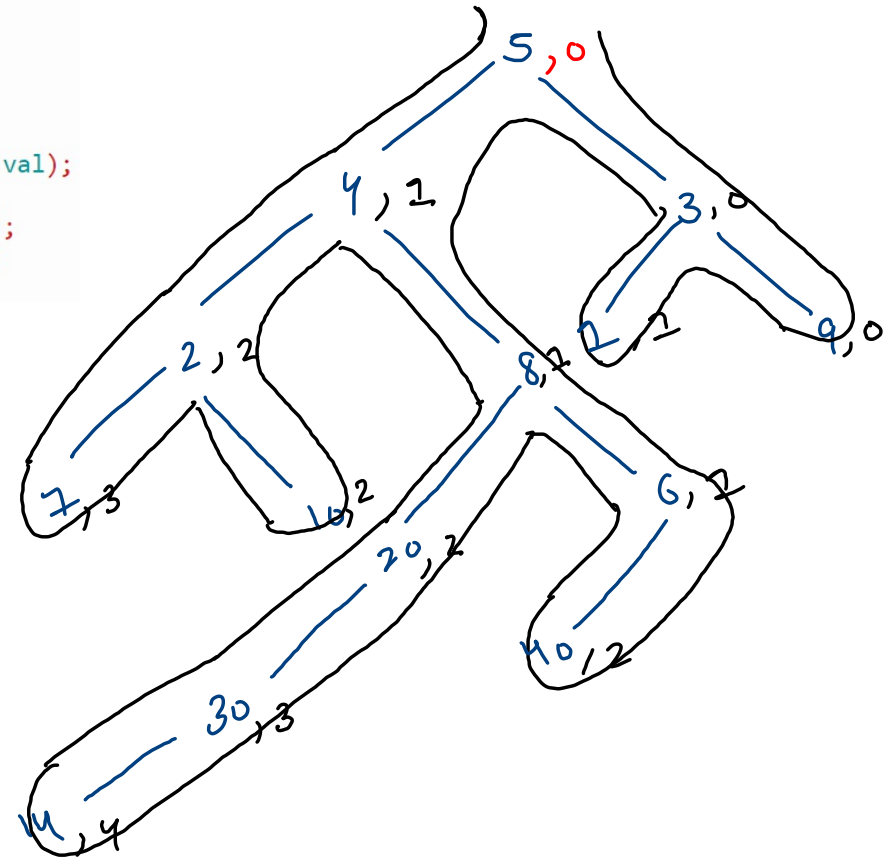


ans. 20 , 15 , 63 , 30 , 15

sum =

5	2	1	8	3	10	20	10	15

```
if(node == null) {
    return;
}

if(dl == ans.size()) {
    ans.add(0);
}
ans.set(dl,ans.get(dl) + node.val);

dos_helper(node.left,dl+1,ans);
dos_helper(node.right,dl,ans);
```



ans : 17 19 72 37 14

0 1 2 3 4

h.w

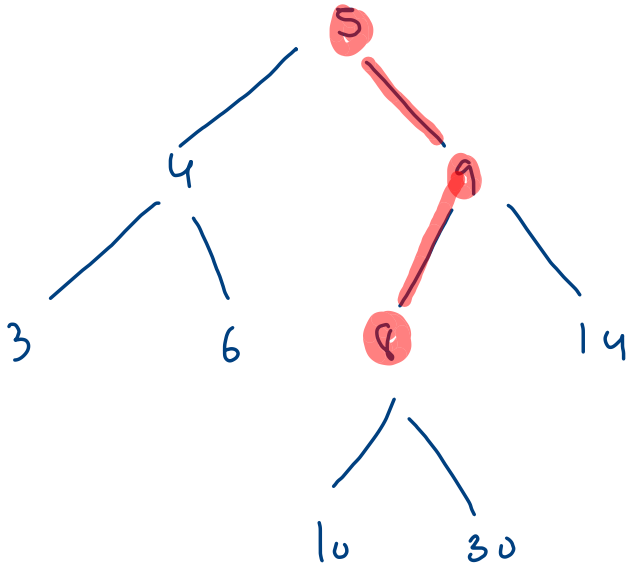
LL

pos , dl

lc rc

dl+1 dl

N2RP



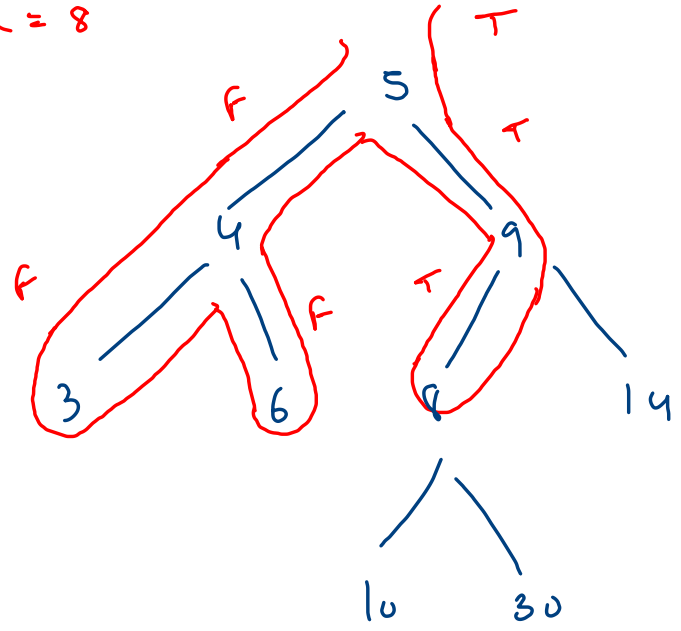
data = 8

[8, 9, 5]

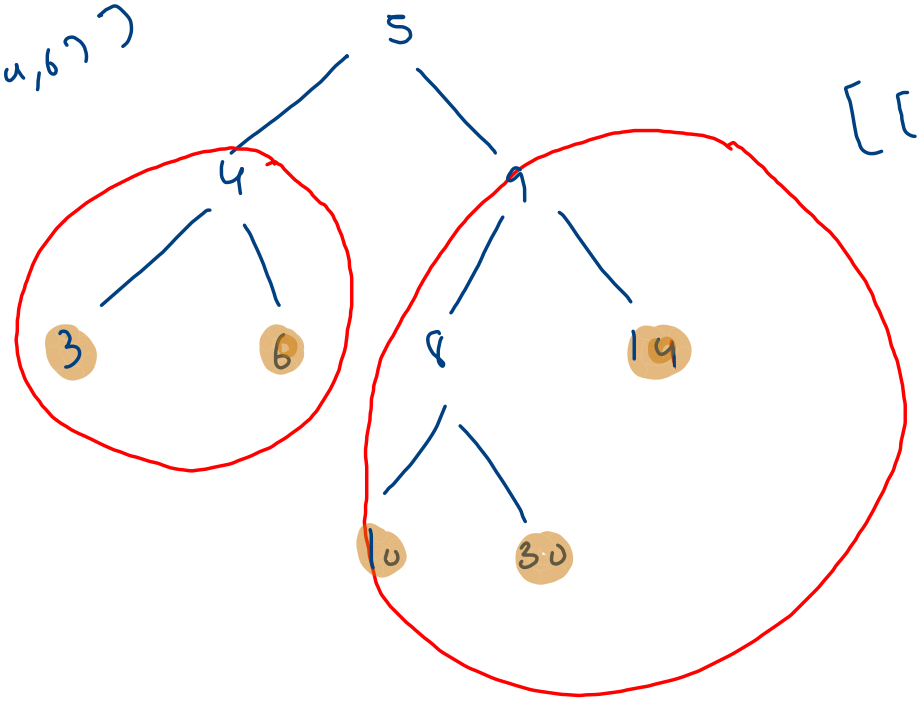
```java
if(node == null) {
    return false;
}

if(node.val == data) {
    ans.add(node);
    return true;
}

boolean lc = n2rp_helper(node.left,data,ans);

if(lc == true) {
    ans.add(node);
    return true;
}

boolean rc = n2rp_helper(node.right,data,ans);

if(rc == true) {
    ans.add(node);
    return true;
}

return false;
```

Stop Share

data = 8



ans (8) (9) (5)



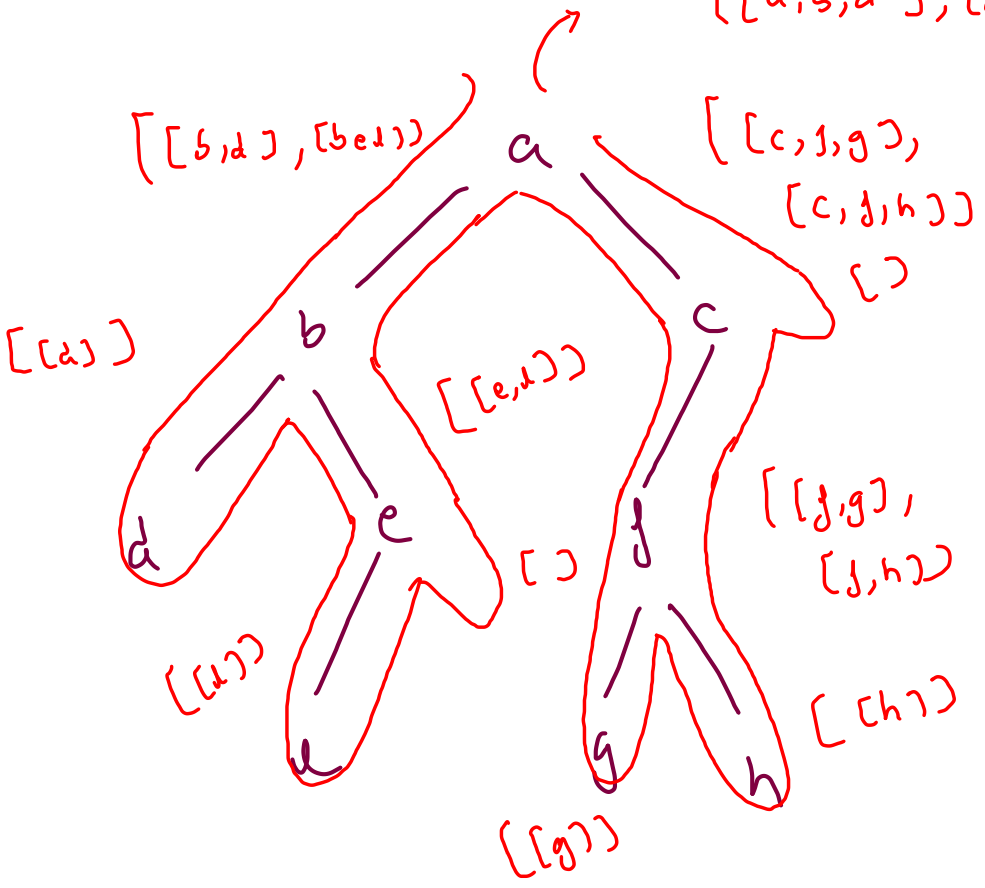
5 4 3

5 4 6

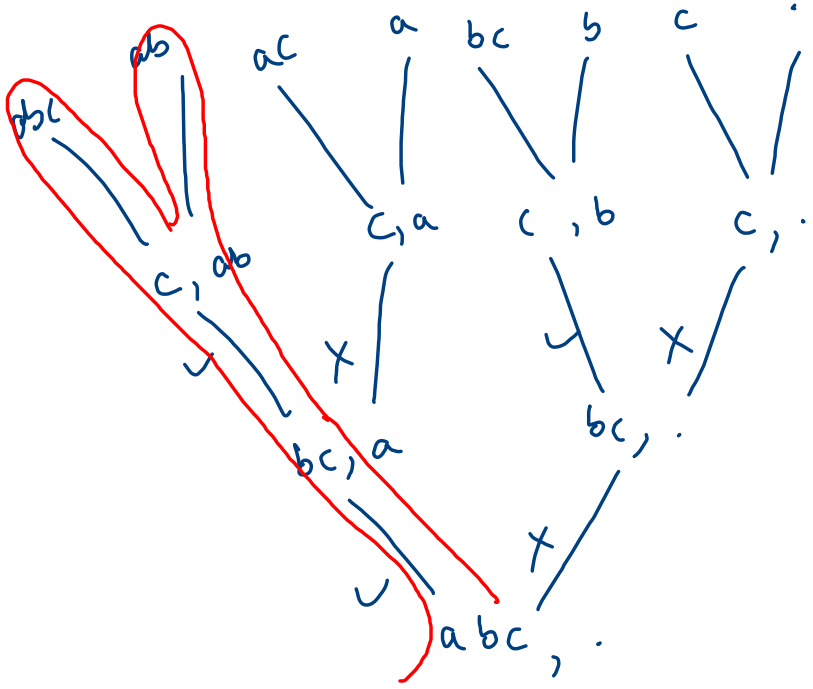
5 9 8 10

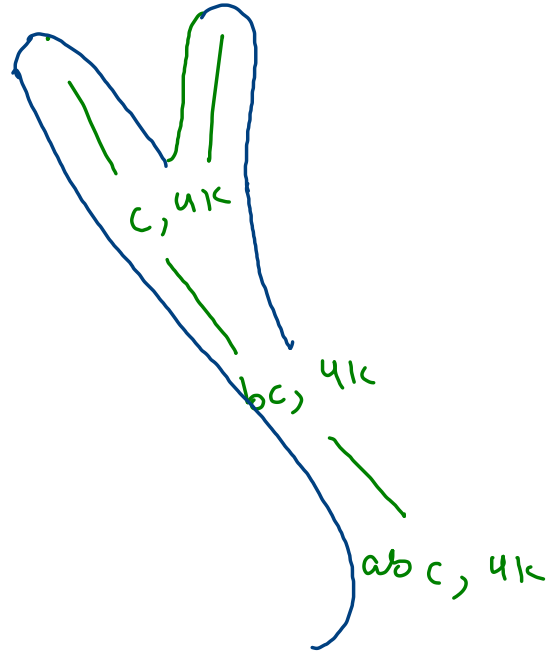
5 9 8 30

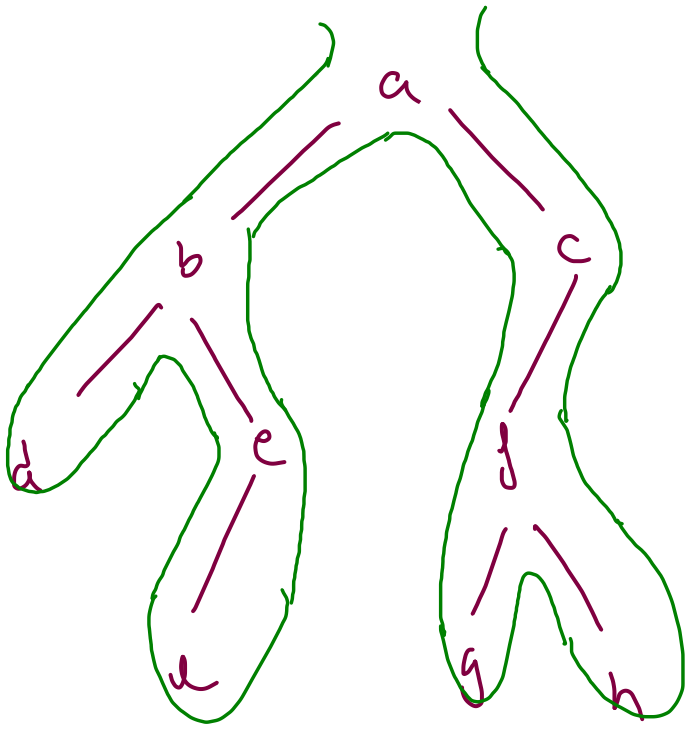
5 9 14




```java
public static ArrayList<ArrayList<Integer>> rootToAllLeafPath(TreeNode root) {
    if(root == null) {
        ArrayList<ArrayList<Integer>>bl = new ArrayList<>();
        return bl;
    }

    if(root.left == null && root.right == null) {
        //Leaf node
        ArrayList<ArrayList<Integer>>bl = new ArrayList<>();
        ArrayList<Integer>list = new ArrayList<>();
        list.add(root.val);
        bl.add(list);
        return bl;
    }

    ArrayList<ArrayList<Integer>>r2lp = new ArrayList<>();
    ArrayList<ArrayList<Integer>>lans = rootToAllLeafPath(root.left);
    ArrayList<ArrayList<Integer>>rans = rootToAllLeafPath(root.right);

    for(ArrayList<Integer>lctoleafpath : lans) {
        lctoleafpath.add(0,root.val);
        r2lp.add(lctoleafpath);
    }

    for(ArrayList<Integer>rctoleafpath : rans) {
        rctoleafpath.add(0,root.val);
        r2lp.add(rctoleafpath);
    }

    return r2lp;
}
```


8k -> [abc ,
ab
ac
a
bc
b
c
abc
c , ab
c , a
c , b
c , .
bc , a
bc , .
abc , .


8k -> [4k
ans
4k
a b
c , 4k
bc , 4k
abc , 4k




```java
public static void r2lp_helper(TreeNode node,ArrayList<Integer>asf,ArrayList<ArrayList<Integer>>ans) {
    if(node == null) {
        return;
    }

    if(node.left == null && node.right == null) {
        //Leaf node
        asf.add(node.val);
        ArrayList<Integer>list = new ArrayList<>(asf);
        ans.add(list);
        asf.remove(asf.size()-1);
        return;
    }

    asf.add(node.val);
    r2lp_helper(node.left,asf,ans);
    r2lp_helper(node.right,asf,ans);
    asf.remove(asf.size()-1);
}
```

asf

4lk

ans

8lk

[

19lk

a,b,d

21lk

a,b,e,f

21k

a,c,f,g

5lk

a,c,f,h