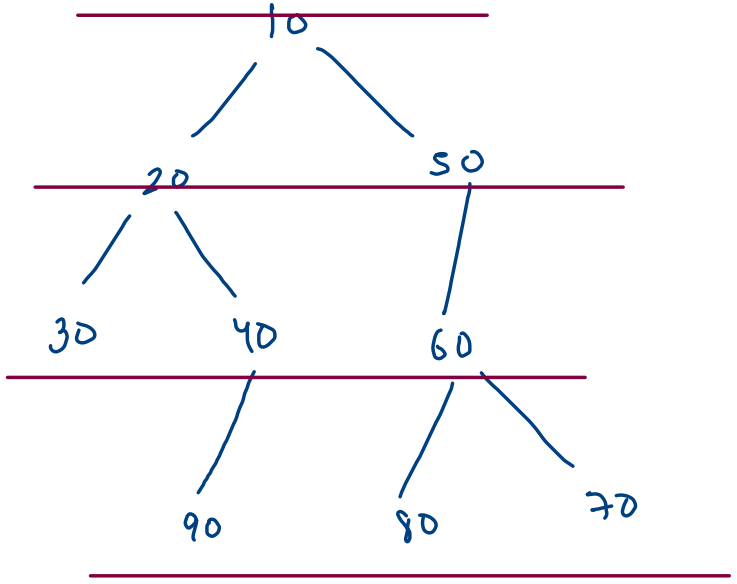
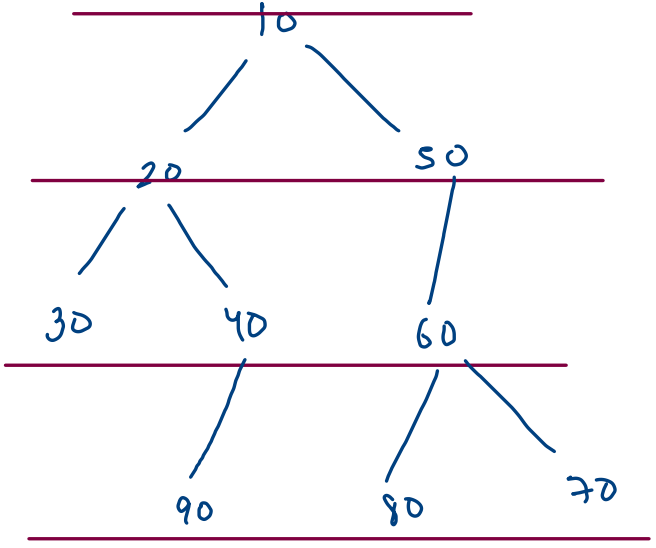
normal level order



10 20 50 30 40 60 90 80 70

```
while (q.size() > 0) {
    rem
    print
    add children
}
```

level order line wise



c = 3

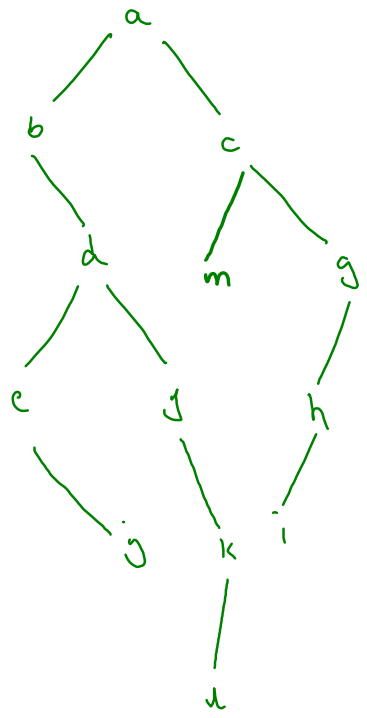
```
while (q.size() > 0) {
    int count = q.size();
    while (count-- > 0) {
        sum
        print
        ac
    }
    → sysO( );
}
```

10

20 50

30 40 60

90 80 70



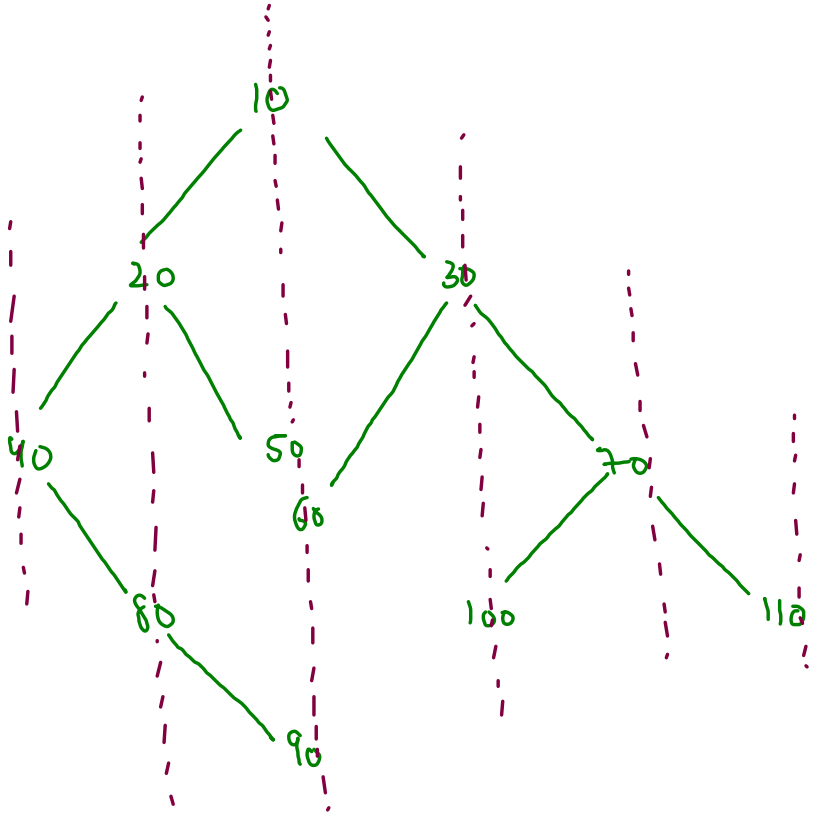
left view -> each horizontal level's first node

lv : a b d e j l

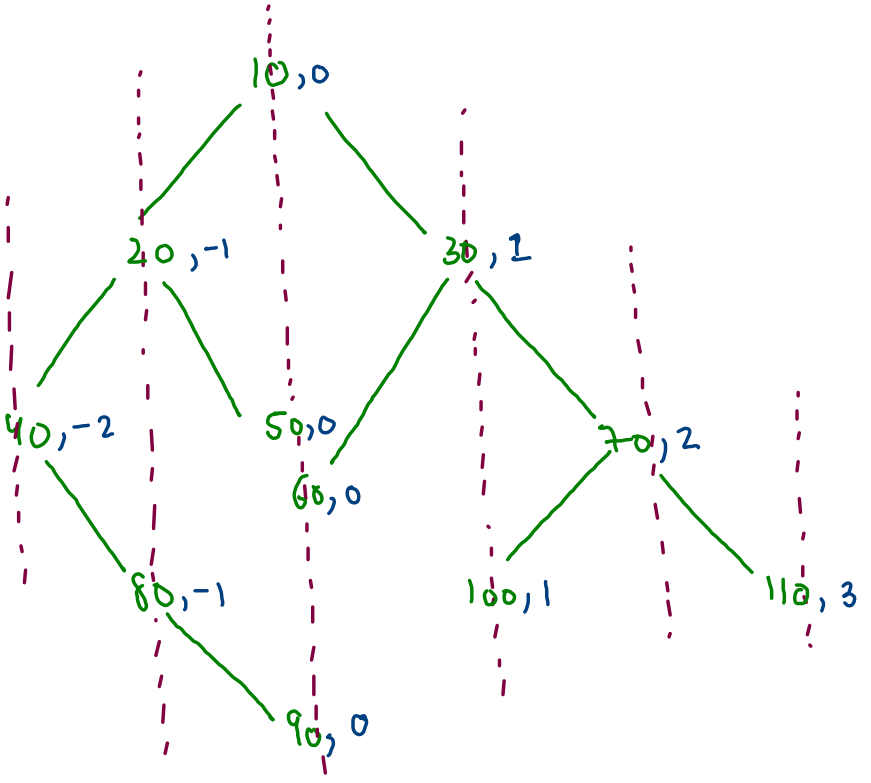
right view -> each horizontal level's last node

rv : a c g h i l

Width Of Shadow Of Binary Tree

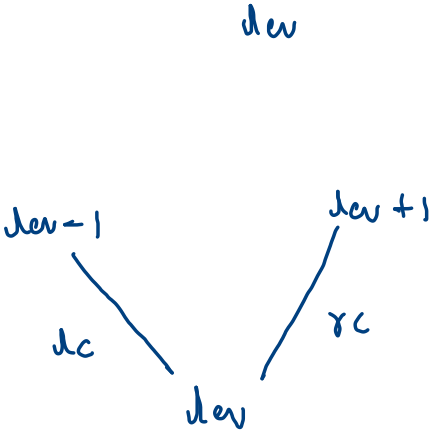


width = no. of vertical lines



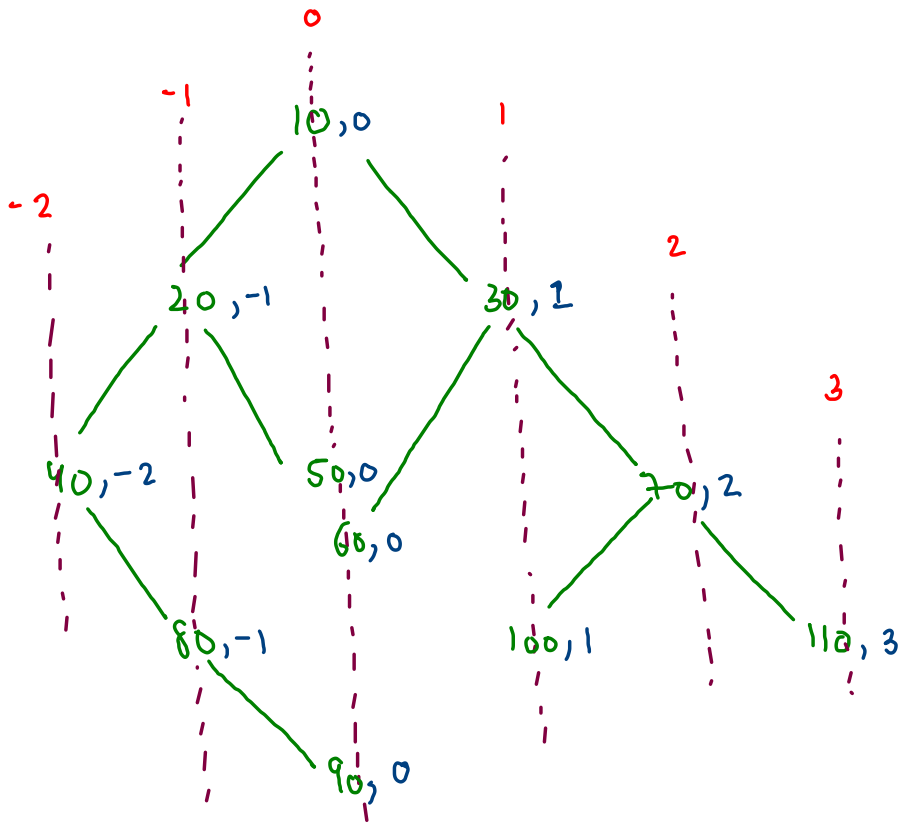
min = -2

max = 3



width = max - min + 1

Vertical Order Traversal Of A Binarytree



AL < AL > ans;

VOT

-2 : 40

-1 : 20 80

0 : 10 50 60 90

1 : 30 100

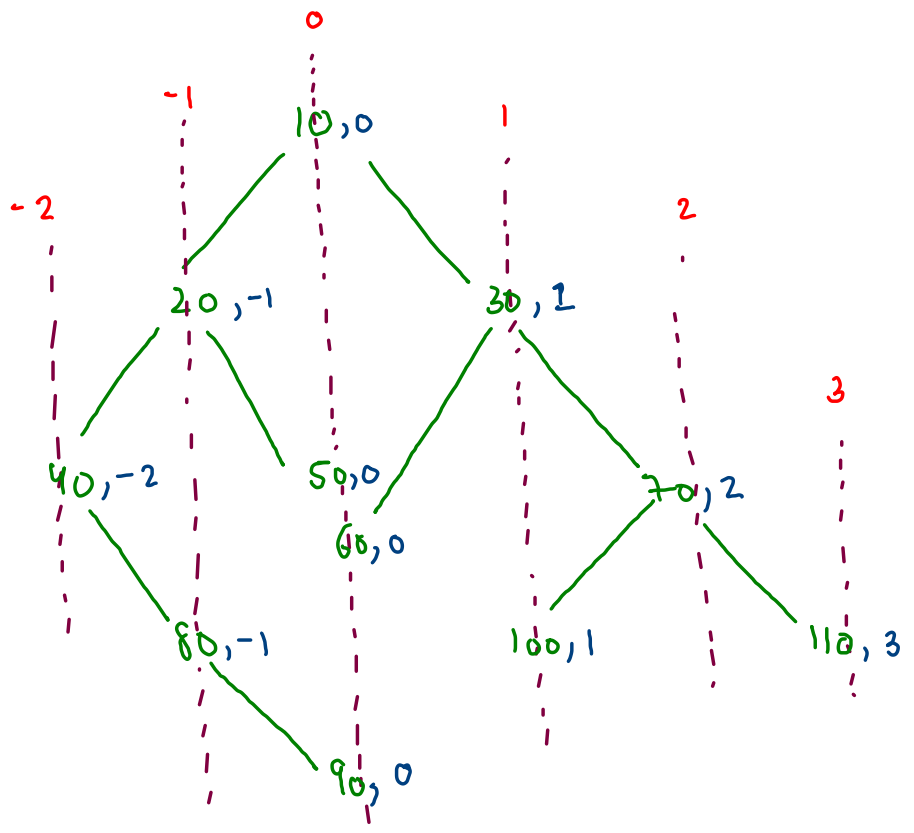
2 : 70

3 : 110

note : reach each vL (Top to bottom)

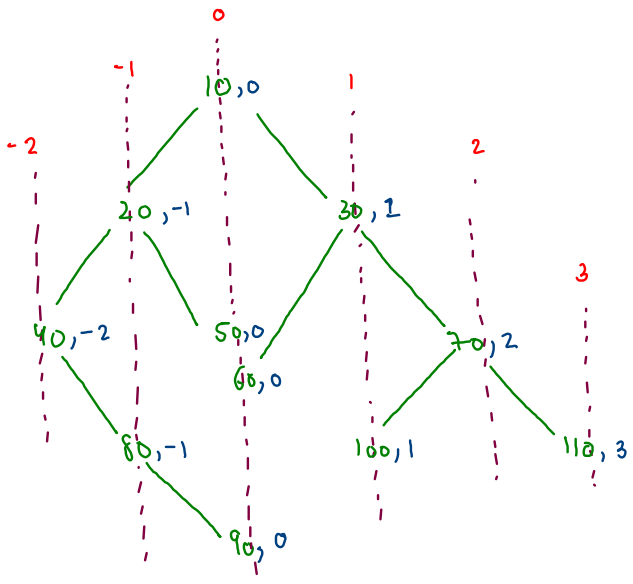
recursion (failed)

0 -> 10, 90, 50, 60



0 -> 10, 50, 60, 90 (No)

top to bottom



do normal

10,0	20,-1	30,1	40,-2	50,0	60,0	70,2	80,-1	100,1	110,3	90,0

min = -2

max = 3

0 -> 10, 50, 60, 90

-1 -> 20, 80

1 -> 30, 100

-2 -> 40

2 -> 70

3 -> 110

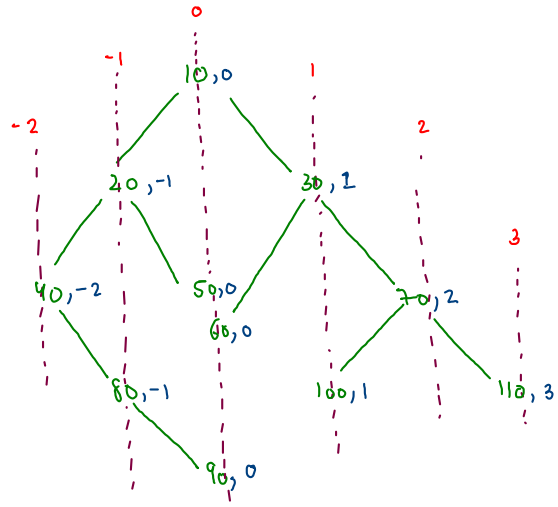
Pair: node

vl (vertical line no.)

HM -> vl vs

AL <Integer>

```java
while(q.size() > 0) {
    //remove
    Pair rem = q.remove();
    min = Math.min(min,rem.vl);
    max = Math.max(max,rem.vl);

    //work
    if(map.containsKey(rem.vl) == false) {
        ArrayList<Integer>list = new ArrayList<>();
        list.add(rem.node.val);
        map.put(rem.vl,list);
    }
    else {
        ArrayList<Integer>list = map.get(rem.vl);
        list.add(rem.node.val);
        map.put(rem.vl,list);
    }

    //add children
    if(rem.node.left != null) {
        q.add(new Pair(rem.node.left,rem.vl-1));
    }
    if(rem.node.right != null) {
        q.add(new Pair(rem.node.right,rem.vl+1));
    }
}
```



0 → 10, 50, 60, 90

-1 → 20, 80

1 → 30, 100

-2 → 40

2 → 70

3 → 110

map

10,0	20,-1	30,1	40,-2	50,0	60,0	70,2	80,-1	100,1	110,3	90,0

min = -2

max = 3

0 → 10, 50, 60, 90

-1 → 20, 80

1 → 30, 100

-2 → 40

2 → 70

3 → 110

map

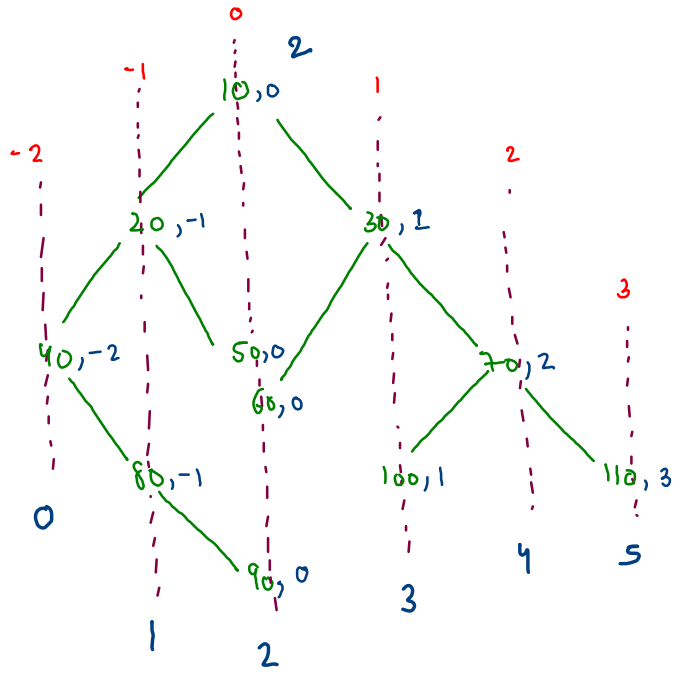
```
//ans creation
ArrayList<ArrayList<Integer>>ans = new ArrayList<>();

for(int i=min; i <= max;i++) {
    ans.add(map.get(i));
}
```

min = -2

max = 3

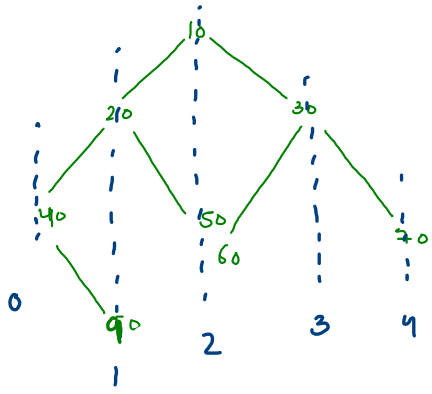
[[40], [20, 80], [10, 50, 60, 90], [30, 100], [70], [110]]

-2 -1 0 1 2 3



[ ]

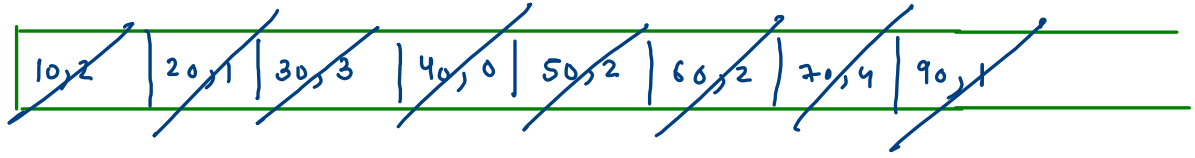
	-2	-1	0	1	2	3
shift	0	1	2	3	4	5




```
width(root,0);

int rvl = -min; //root's vertical line
int w = max - min + 1;


ArrayList<ArrayList<Integer>>ans = new ArrayList<>();

ArrayDeque<Pair>q = new ArrayDeque<>();
q.add(new Pair(root,rvl));

for(int i=0; i < w;i++) {
    ans.add(new ArrayList<>());
}

while(q.size() > 0) {
    //remove
    Pair rem = q.remove();

    //work
    ans.get(rem.vl).add(rem.node.val);

    //add children
    if(rem.node.left != null) {
        q.add(new Pair(rem.node.left,rem.vl-1));
    }
    if(rem.node.right != null) {
        q.add(new Pair(rem.node.right,rem.vl+1));
    }
}
```

VOT without HM

(origin shift)



min = -2

vvl = 2

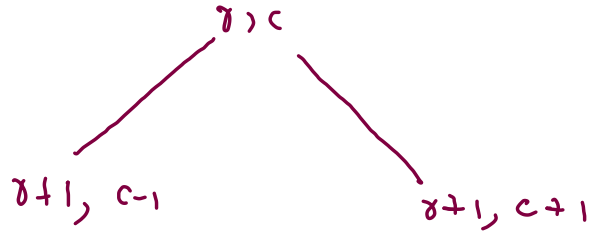
max = 2

w = 5

[[40] [20,90] [10,50,60] [30] [70]]

ans 0 1 2 3 4

leetcode VOT



For each node at position `(row, col)`, its left and right children will be at positions `(row + 1, col - 1)` and `(row + 1, col + 1)` respectively. The root of the tree is at `(0, 0)`.

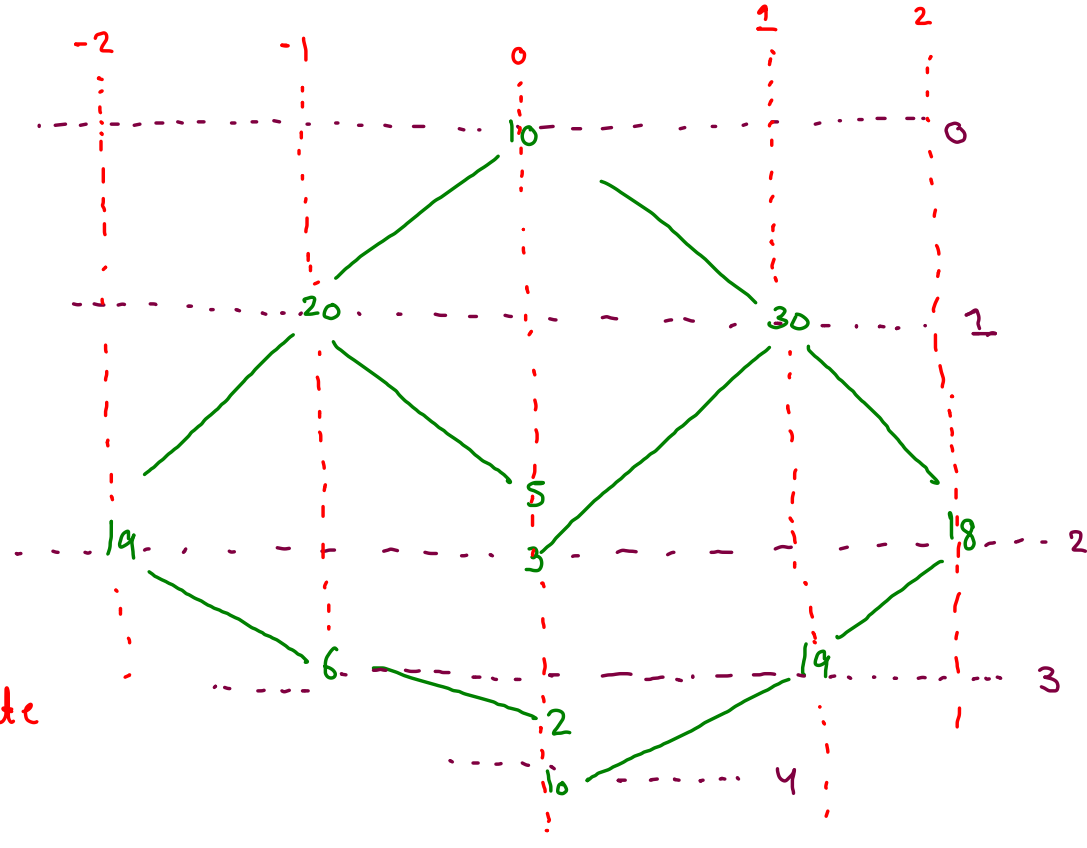
The **vertical order traversal** of a binary tree is a list of top-to-bottom orderings for each column index starting from the leftmost column and ending on the rightmost column. There may be multiple nodes in the same row and same column. In such a case, sort these nodes by their values.

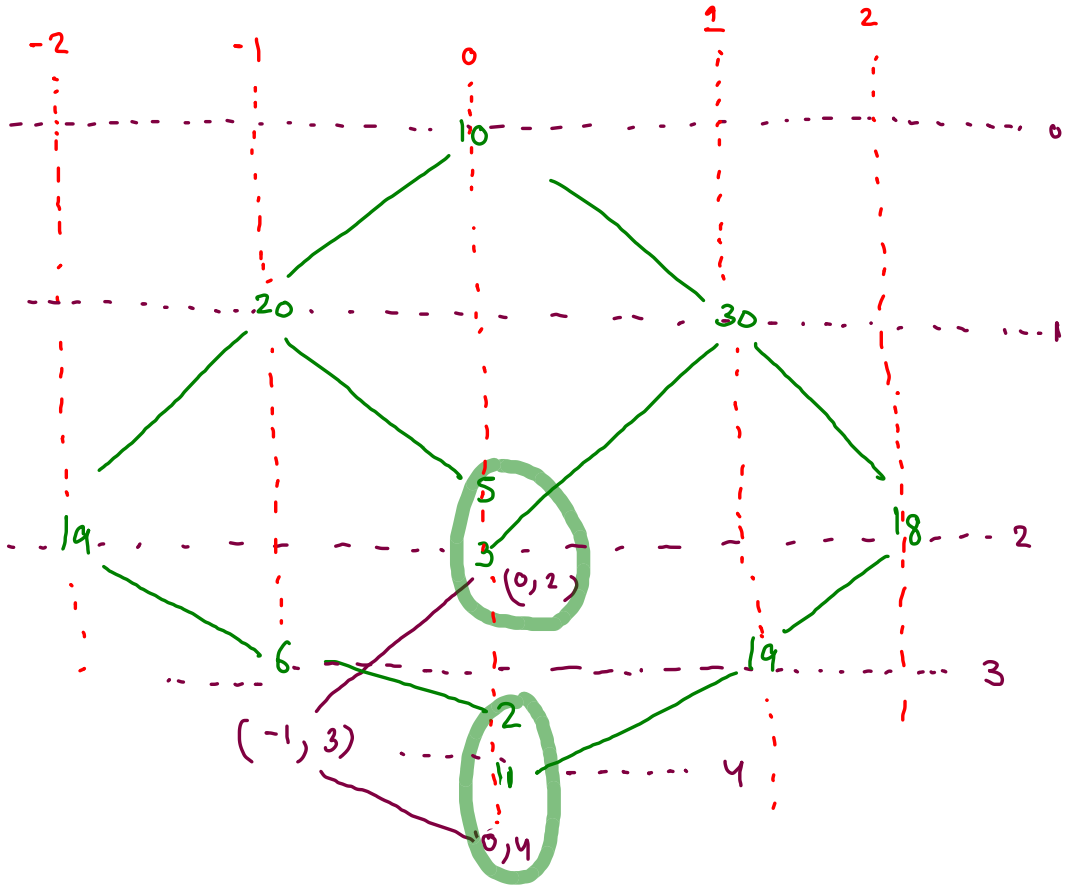
↑ decided by y coordinate

hd (horizontal line)

vd (vertical line)

↳ decided by x coordinate





x, y

→ x (cols)

→ y (rows)

-2 → 19

-1 → 20 6

0 → 10 3 5 2 11

⋮

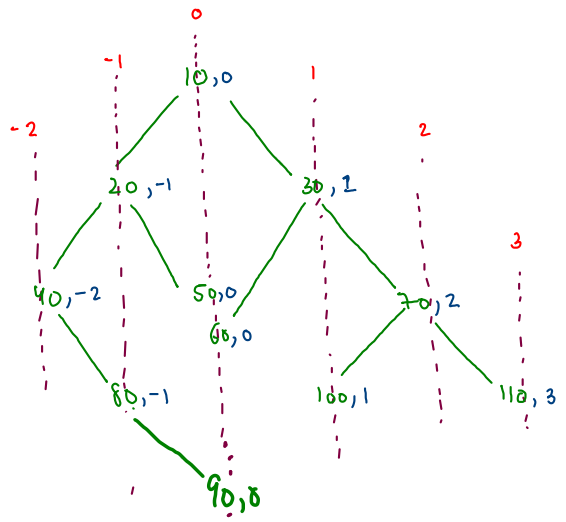
PQ

x, y, value

do

- → highest priority : y (do)
- → if two nods have same y smaller x.
- → if two nodes have x == y values.

Top view & bottom view



top view: first node of each Verticle line

tv: 40 20 10 30 70 110

bottom view: last node of each Verticle line

bv: 40 80 90 100 70 110