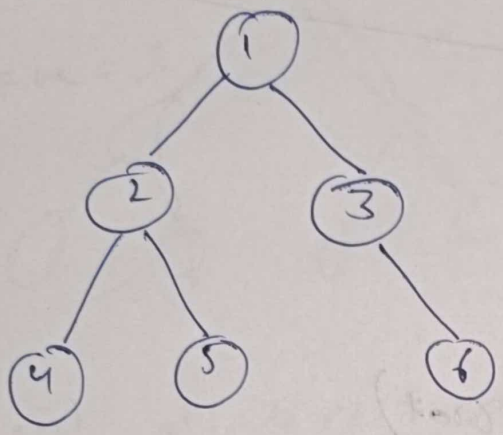
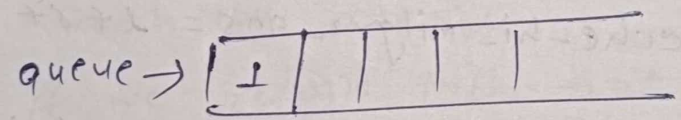
Zig-Zag Traversal



1
3 2
4 5 6

Approch :- Same approach as level order traversal.

(i) initialy we take a queue & insert root in it



(ii) Further we take a flag which is either is 0 or 1 of the base basis of 0 or 1 we insert values into vector

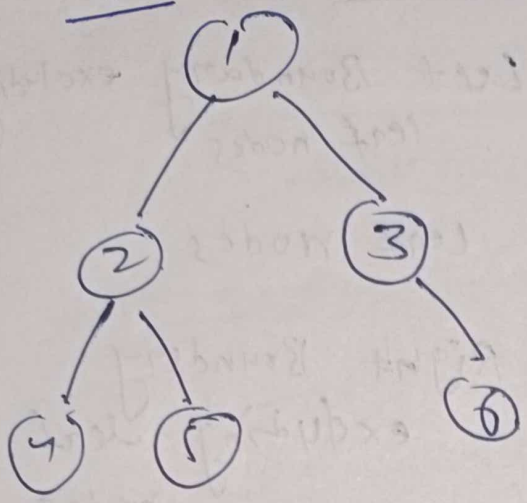
if flag == 0 that means push into vector form left to right

flag == 0 ——push——→

if flag == 1 push into vector from right to left

flag == 1 ←——push——

ex



flag = 0 / 1

1	2	3	4	5	6

~~temp = 1~~

result =

1		
3		2
4	5	6

1st iteration

queue.size() = 1

for (i = 0 to 1)

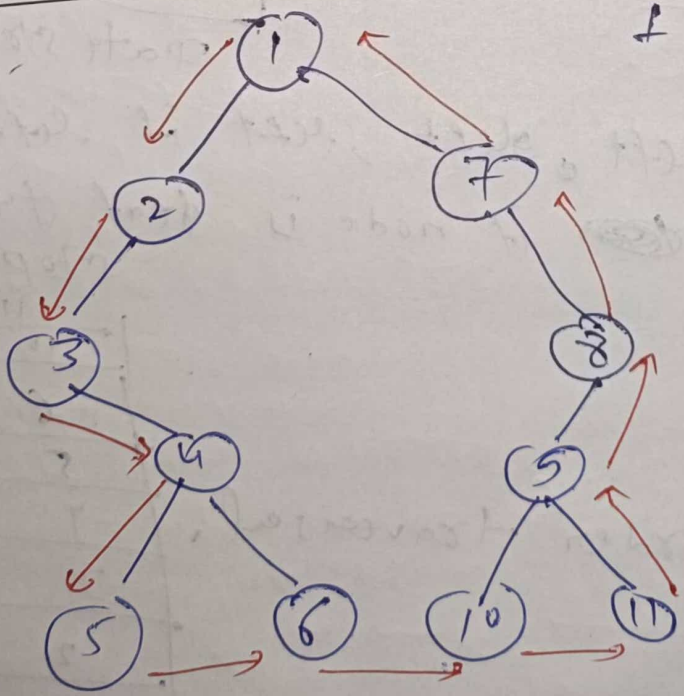
{

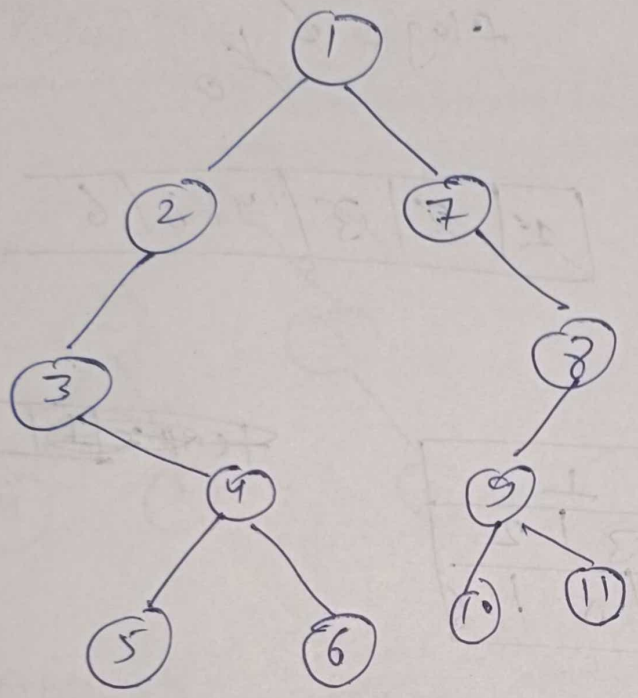
}

Boundary Traversal

1 2 3 4 5 6 10 11

9 8 7





Approch:-

(i) Left Boundary excluding leaf nodes

(ii) Leaf nodes

(iii) Right Boundary exluding leaf nodes

(i) Lef Boundary excluding leaf

first insert root into DS

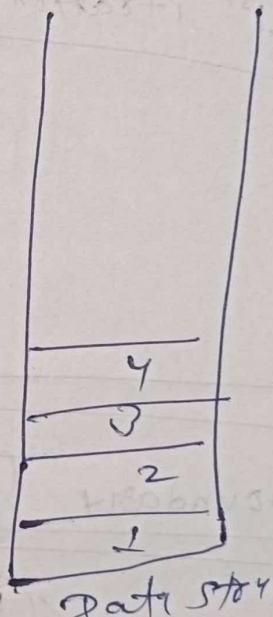
Now go left of the root = 2 so insert it into DS.

Now same go left of 2 = 3

3's left == NULL go for right = 4

4's left == 5 but it is leaf then stop

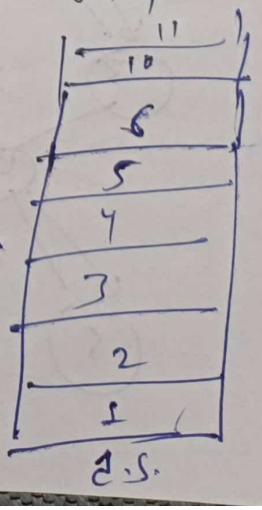
That means go left, left, left if left not exist go right if node is leaf then stop



Data structure

(ii) leaf nodes

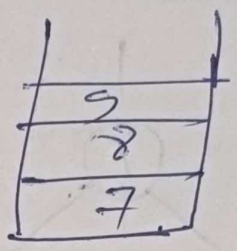
follow pre order traversal.



D.S.

(iii) right boundary traversal excluding leaf & root.

temp ← vector.



temp

go root → right = 7

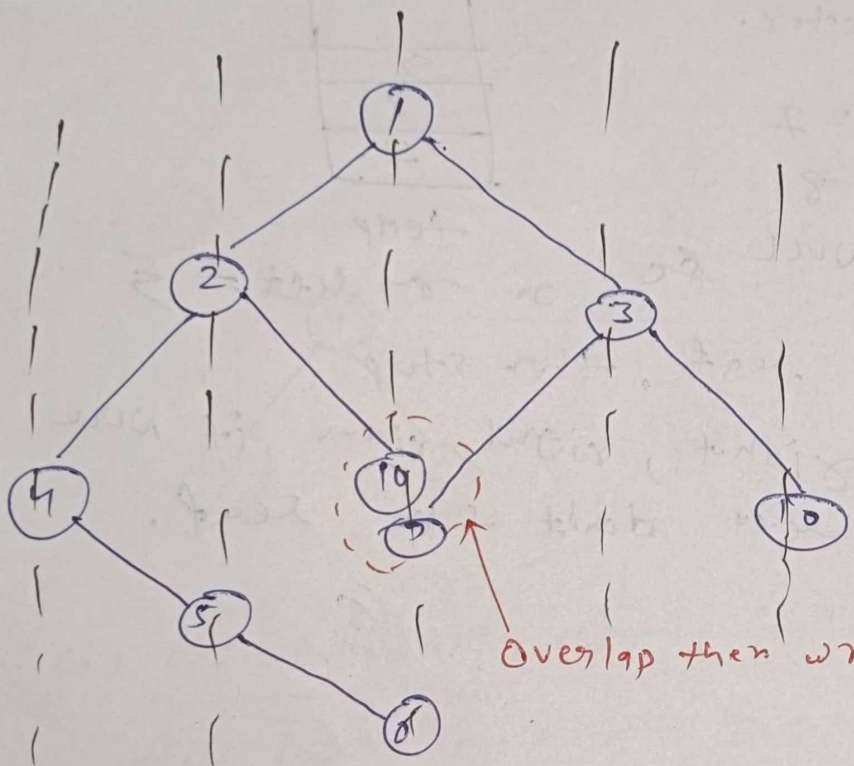
7 → right = 8

8 → right = NULL So or. r → left = 5

5 → right = leaf then stop

that means right, right, rim if NULL left donot take leaf.

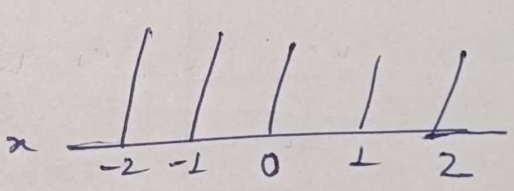
Vertical Order Traversal of Binary Tree

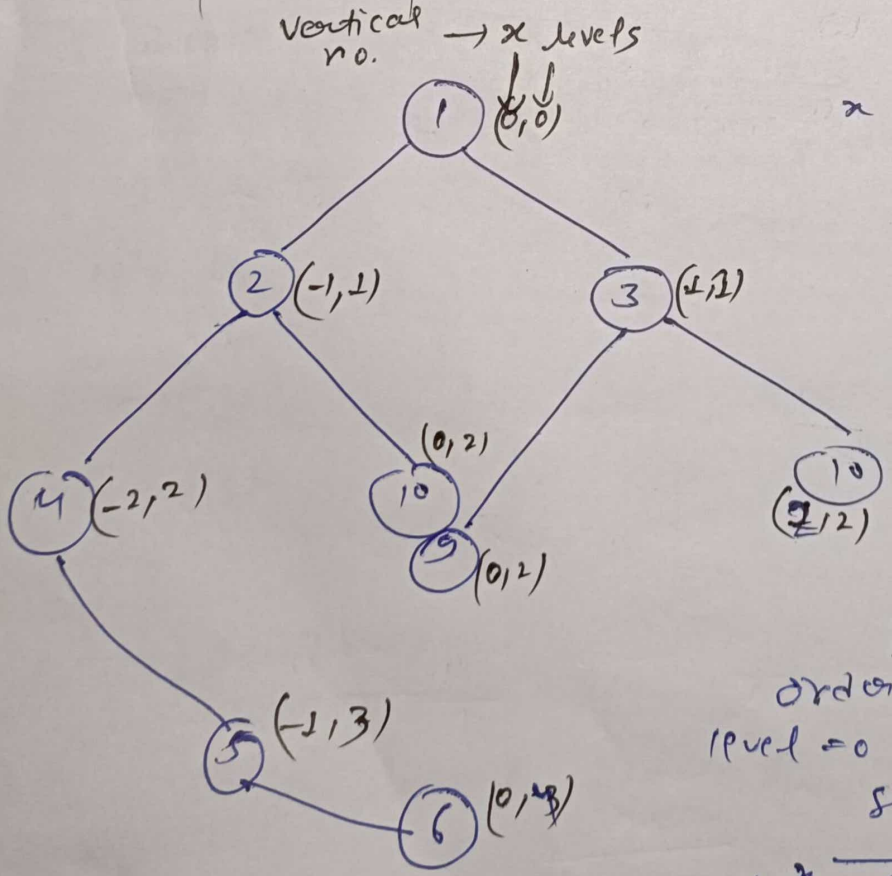


Overlap then write smaller value first.

ans

4
2 5
1 9 10 6
8
10





for x we can itrate the asscending order of x that mean first -2, then -1, 0 & so on

also same for level itrate from asending order that means first level = 0 then level = 1 2 3 so on

x → 4
2 5
1 9 10 6
8
10

level

So we need to traverse every node and assign them (x, level)

where

x = vertical no.

level = level no. start from 0, 1, 2, ---

Intuition :- we can gave every vertical a number so that our traversing become easy And we assign every node to a coordinate

type of coordinate is like

(vertical no, level no)

By this coordinating our traversing become ceasy

How :- we traverse them in increasing order of vertical and in vertical we traverse in increasing order of levels.

like first go for minimum vertical = -2

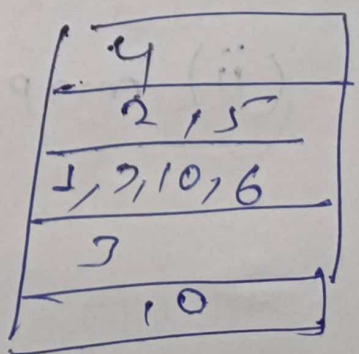
vertical = -2 level = 2

vertical = -1 levels = 1, 3

vertical = 0 levels = 0, 2, 4

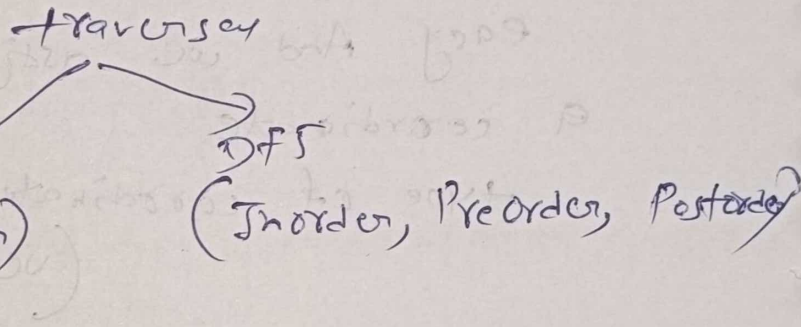
vertical = 1 levels = 1

vertical = 2 levels = 2



Now think about the Data Structure that means which kind of DS you use for solution.

Before moving on DS think about the traversal of Binary tree because mainly you have two kind of traversal



for this question you can use any of the traversal but we use level order because in our intuition we discussing about levels. so:

→ Now DS.

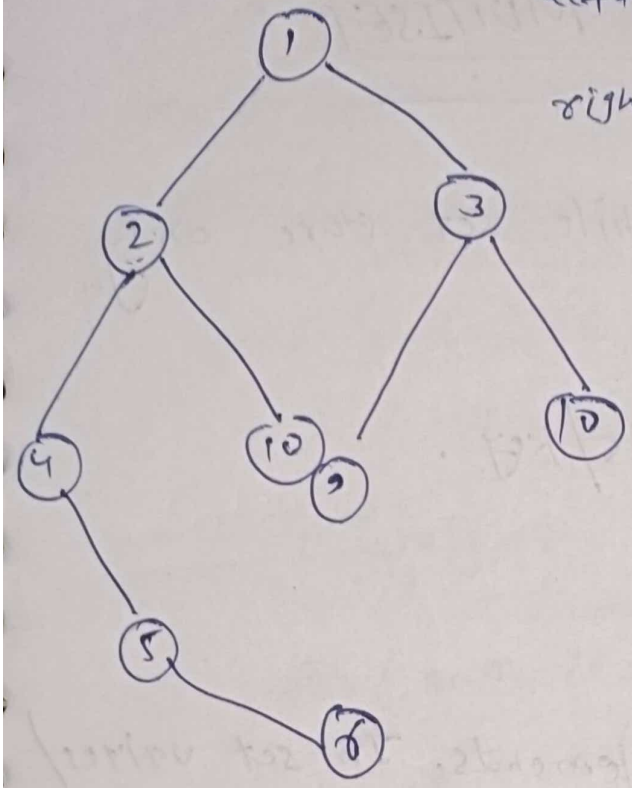
we use two data structure

(i) queue<(node, vertical, level)>

(ii) map<int, map<int, multiset<int>>>;

vertical — levels — (levels, values)

Store the values of levels in sorted order and value can be same so multiset.



left → vertical -- level ++

right → vertical ++ level ++

~~(1, 0, 0)~~

node = (1, 0, 0)

node val, vertical, level

1 → left = (2, -1, 1)

1 → right = (3, 1, 1)

~~(2, -1, 1)~~ (3, 1, 1)

node = (2, -1, 1)

2 → left = (4, -2, 2)

2 → right = (10, 0, 2)

~~(3, 1, 1)~~ (4, -2, 2) (10, 0, 2)

node = (3, 1, 1)

3 → left = (9, 0, 2)

3 → right = (10, 2, 2)

Data structure

vertical	levels	values
0	0	{1}
-1	1	{2}
1	1	{3}
-2	2	{4}

~~(4, -2, 2)~~ (10, 0, 2) (9, 0, 2) (10, 2, 2)

node = (4, -2, 2)

4 → left = NULL

4 → right = (5, -1, 3)

So . On.

T.C. = O(n) * O(log n)

S.C. = O(n)