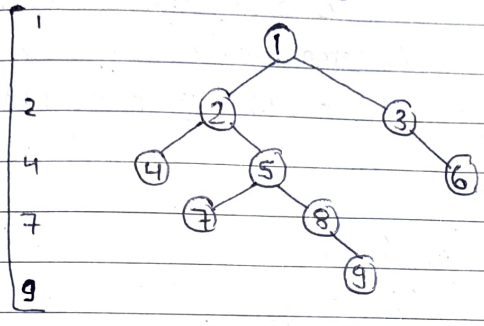
Date _____ Page _____

Day - 155

Trees - 5

* Left view of Binary Tree :



⇉ We have to print the left view that means when you are on the left side of the tree, which node you can see.

⇉ So, every first element of every element is the left view of that level.

⇉ So, we have to level order traversal and push the every first element to the ans array.

Code

```
vector<int> leftview (Node * root){
    queue < Node * > q;
    q.push (root);
    vector<int > ans;
    while( ! q.empty()){
        int n = q.size();
        ans.push_back(q.front() -> data);
```

Date _____ Page _____

```
while(n--){
    Node *temp = q.front();
    q.pop();
    if(temp->left)
        q.push(temp->left);
    if(temp->right)
        q.push(temp->right);
    }
}
return ans;
}
```

⇒ Now, if we have to solve the question by recursion then —

⇒ In recursion, we will traverse left side then right side.

⇒ So, we know that we are visiting the first leftmost node is our ans.

⇒ Then we only have to do above task.

⇒ Now, first we find the no. of levels in the tree then ~~fill~~ make a array of size that of level and fill the values by 0.

⇒ Then first time visiting that level is indicated by value 0.

⇒ If it is 0 then it means its first time and we will change the value by 1.

Date _____ Page _____

⇒ And store that node value in ans array.

⇒ Here, we are visiting tree two times.

⇒ But if we have to visit tree only one time.

⇒ Our task is to know that we are visiting that level first time.

⇒ So, for that, by using the size of our ans array, we can know the above thing.

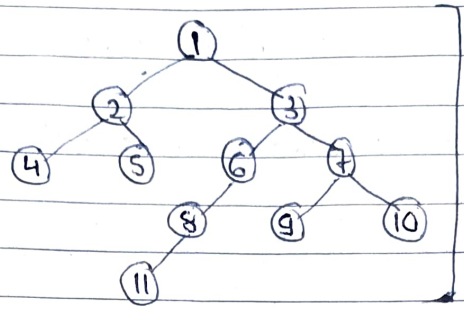
⇒ If the level & the size of ans array is same then we are visiting that level first time.

Code

```
void Lview(Node * root, int level, vector<int> &ans){
    if(!root)
        return;
    if(level == ans.size())
        ans.push_back(root->data);
    Lview(root->left, level+1, ans);
    Lview(root->right, level+1, ans);
}
```

* Right View of Binary Tree:

Date _____ Page _____



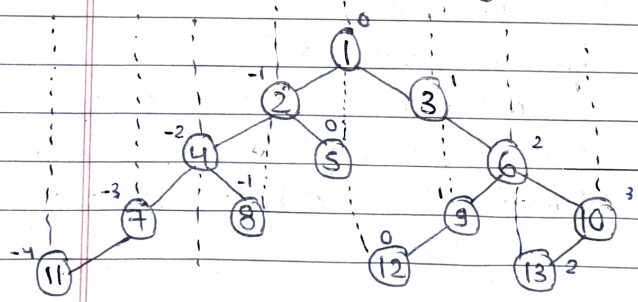
1
3
7
10
11

We have to print the rightmost element of every level.

We can use the previous question approach by using queue.

Simply we will do level order traversal and add the last element to ans array.

* Top View of Binary Tree



1 2 4 7 11 3 6 10

We have to print the elements that can only we see from the top.

Date _____ Page _____

- So, when we go to left side, we will decrease by 1 & when we go right side, we will increase by 1.

- And for the same value nodes, that node ~~have~~ ~~low~~ ~~t~~ is at min level of that nodes, we will select that node.

- First, we will find the position of left most and right most element.
- Then we will create an array of size — $r - l + 1$

- Now we will do level order traversal.
- We know that 1 will go to 4 index in the ans array.
- That means, here we are doing shifting.
- So, we can do that, we will change the numbering i.e. leftmost get 0 & above get increase by 1 no.
- Also, we will create an another array, to know that number node is visited or not.

	0	1	2	3	4	5	6	7
ans	11	7	4	2	1	3	6	10

0	1	2	3	4	5	6	7
1	1	1	1	1	1	1	1

visited

Date _____ Page _____

Code

```
vector<int> topView(Node * root){
    int l=0, r=0;
    find(root, 0, l, r);
    vector<int> ans(r-l+1);
    vector<bool> filled(r-l+1, 0);
    queue<Node *> q;
    queue<int> index;
    q.push(root);
    index.push(-1*l);
    while(!q.empty()){
        Node *temp = q.front();
        q.pop();
        int pos = index.front();
        index.pop();
        if(!filled[pos]){
            filled[pos]=1;
            ans[pos] = temp->data;
        }
        if(temp->left){
            q.push(temp->left);
            index.push(pos-1);
        }
        if(temp->right){
            q.push(temp->right);
            index.push(pos+1);
        }
    }
    return ans;
}
```

Date _____ Page _____

```
void find(Node * root, int pos, int &l, int &r){
    if(!root)
        return;
    l = min(l, pos);
    r = min(r, pos);
    find(root->left, pos-1, l, r);
    find(root->right, pos+1, l, r);
}
```

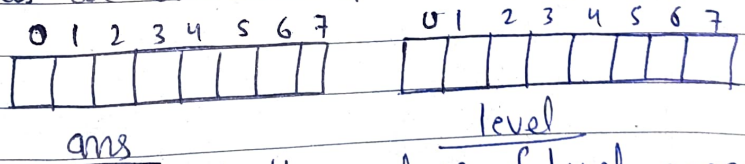
⇉ Now, if we have to solve this question by using recursion.

⇉ we know that recursion works side by side.

⇉ So, here first we have to find left & right most pos.

⇉ Then we will create ans.

⇉ Here, we will fill values according to the level not by ~~any~~ using a filled array as we used in previous approach.

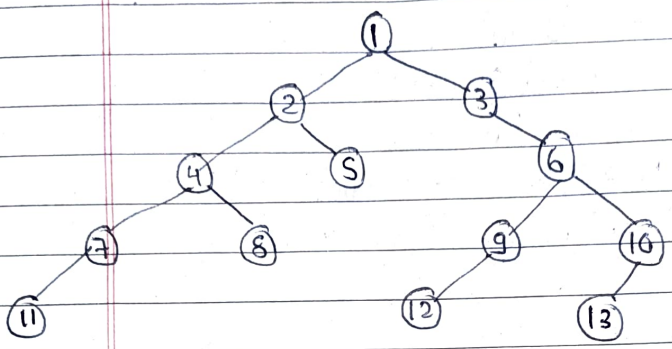
⇉ 

ans level

⇉ we will fill the values of level array in the starting ~~int~~ with INT_MAX.

Date _____ Page _____

* Bottom View of Binary Tree:



⇉ 11 7 4 8 ~~5~~ 12 9 13 10

⇉ we have to print the ~~botto~~ elements that can be see from the bottom of the tree.