**Top View of Binary Tree**

Given below is a binary tree. The task is to print the top view of binary tree. Top view of a binary tree is the set of nodes visible when the tree is viewed from the top. For the given below tree

       1  
    /     \  
   2       3  
  /  \    /   \  
4    5  6   7

Top view will be: 4 2 1 3 7  
**Note:**Return nodes from **leftmost**node to **rightmost**node. Also if 2 nodes are outside the shadow of the tree and are at same position then consider the extreme ones only(i.e. leftmost and rightmost).   
For ex - **1 2 3 N 4 5 N 6 N 7 N 8 N 9 N N N N N** will give **8 2 1 3** as answer. Here 8 and 9 are on the same position but 9 will get shadowed.

**Example 1:**

**Input:**

  1

 /    \

2      3

**Output:** 2 1 3

**Example 2:**

**Input:**

  10

   /      \

20        30

/   \    /    \

40   60 90    100

**Output:** 40 20 10 30 100

Solution:-

class Solution

{

static void traverse(TreeMap<Integer,ArrayList<Integer>> tm,Node node,int vlevel,int hlevel){

if(node==null)

return;

else{

if(!tm.containsKey(vlevel)){

tm.put(vlevel,new ArrayList<>());

tm.get(vlevel).add(hlevel);

tm.get(vlevel).add(node.data);

}

else if(hlevel<tm.get(vlevel).get(0)){

tm.get(vlevel).set(0,hlevel);

tm.get(vlevel).set(1,node.data);

}

traverse(tm,node.left,vlevel-1,hlevel+1);

traverse(tm,node.right,vlevel+1,hlevel+1);

}

}

static ArrayList<Integer> topView(Node root)

{

// add your code

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

TreeMap<Integer,ArrayList<Integer>> tm=new TreeMap<>();

traverse(tm,root,0,0);

//System.out.println(tm);

int key=tm.firstKey();

while(tm.containsKey(key)){

ar.add(tm.get(key).get(1));

key++;

}

return ar;

}

}