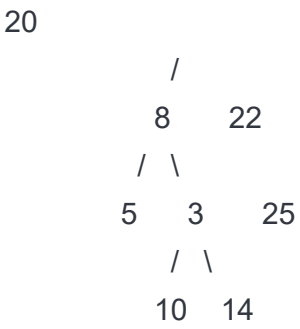
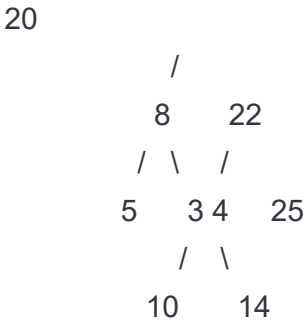
Bottom View of Binary Tree

Given a binary tree, print the bottom view from left to right.
A node is included in bottom view if it can be seen when we look at the tree from bottom.



For the above tree, the bottom view is 5 10 3 14 25.
If there are **multiple** bottom-most nodes for a horizontal distance from root, then print the later one in level traversal. For example, in the below diagram, 3 and 4 are both the bottommost nodes at horizontal distance 0, we need to print 4.



For the above tree the output should be 5 10 4 14 25.

```
def bottomView(root):

    # code here
    seen = {}
    least = [0]
    # minimum = least[0]
    helper(root, seen, 0,0)
    ans = []
    minimum = min(seen.keys())
    while True:
        if minimum in seen:
            ans.append(seen[minimum][0])
            minimum = minimum+1
        else:
```

```
            break
    return ans



def helper(root, seen, level,depth):
    if root is None:
        return

    helper(root.left, seen, level - 1,depth+1)
    helper(root.right, seen, level + 1,depth+1)
    if level in seen:
        if depth>=seen[level][1]:
            seen[level] = [root.data,depth]
    else:
        seen[level] = [root.data,depth]
```

```
from collections import defaultdict
class Solution:
    def bottomView(self, root):
        # code here
        if root is None:
            return []
        ans = {}
        self.helper(root,ans,0,0)
        res = []
        for key in sorted(ans.keys()):
            res.append(ans[key][0])
        return res



    def helper(self,root,ans,level,row):
        if root is None:
            return
        self.helper(root.left,ans,level-1,row+1)
        self.helper(root.right,ans,level+1,row+1)
        if level in ans:
            temp = ans[level]
            if row>=temp[1]:
                del ans[level]
                ans[level] = [root.data,row]
```

```
        else:
            ans[level] = [root.data,row]
```