Right View of Binary Tree (GFG)

Given a Binary Tree, find **Right view** of it. Right view of a Binary Tree is set of nodes visible when tree is viewed from **right** side.

Right view of following tree is 1 3 7 8.

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

  8
```

```python
class Solution:
    #Function to return list containing elements of right view of binary tree.
    def rightView(self,root):

        # code here
        res=[]
        seen = set()
        helper(root,res,0,seen)
        return res



def helper(root,res,level,seen):
    if root is None:
        return
    if level not in seen:
        seen.add(level)
        res.append(root.data)
    helper(root.right,res,level+1,seen)
    helper(root.left,res,level+1,seen)
```

Second Approach:

```python
def rightSideView(self, root):
        deque = collections.deque()
        if root:
            deque.append(root)
        res = []
        while deque:
            size, val = len(deque), 0
            for _ in range(size):
                node = deque.popleft()
                val = node.val # store last value in each level
                if node.left:
                    deque.append(node.left)
                if node.right:
                    deque.append(node.right)
            res.append(val)
        return res
```

```
import queue
class Solution:
    #Function to return list containing elements of right view of binary tree.
    def rightView(self,root):

        # code here
        ans = []
        queue  = deque()

        if root is None:
            return []
        queue.appendleft(root)

        while True:
            n = len(queue)
            if n==0:
                break
            count = 0
            while count<n:
                temp = queue.popleft()
                if count==n-1:
                    ans.append(temp.data)
                if temp.left:
                    queue.append(temp.left)
                if temp.right:
                    queue.append(temp.right)
                count = count+1
        return ans
```