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.

```
def topView(self,root):
        dic = {}

        # variable which is going to store
        # the minimum positional value.
        mi = float('inf')
        ans = []

        if not root:
            return ret

        q = deque([(root, 0)])

        while q:
            cur = q.popleft()
            if cur[1] not in dic:
                dic[cur[1]] = cur[0].data
                mi = min(mi, cur[1])
            if cur[0].left:
                q.append((cur[0].left, cur[1] - 1))
            if cur[0].right:
                q.append((cur[0].right, cur[1] + 1))

        # Starting from the leftmost node and
        # just incrementing it until
        # the rightmost node stored in the dic.
        while mi in dic:
            ans.append(dic[mi])
```

```
            mi += 1
        return ans
```

```
class Solution:
    #Function to return a list of nodes visible from the top view
    #from left to right in Binary Tree.
    def topView(self,root):
        m = {}

        self.fillMap(root, 0, 0, m)
        res = []
        for it in sorted(m.keys()):
            res.append(m[it][0])
        return res

    def fillMap(self,root, d, l, m):
        if(root == None):
            return

        if d not in m:
            m[d] = [root.data, l]
        elif(m[d][1] > l):
            m[d] = [root.data, l]
        self.fillMap(root.left, d - 1, l + 1, m)
        self.fillMap(root.right, d + 1, l + 1, m)
```