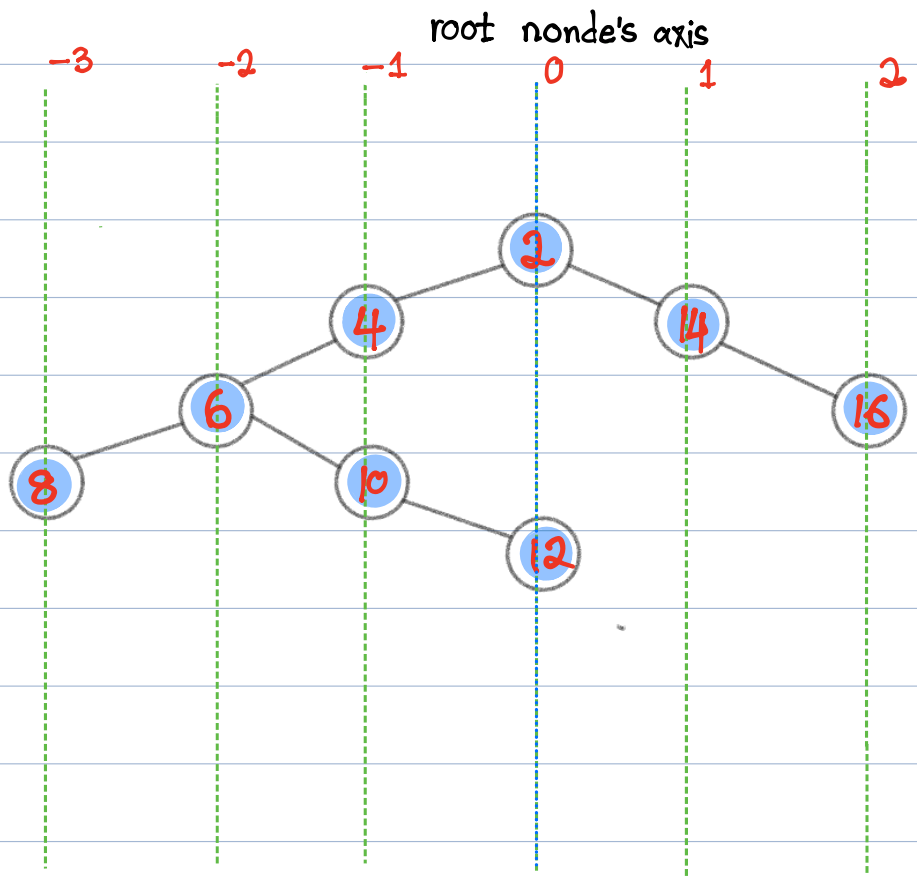
Top view

Given a binary tree, return the nodes visible when the tree is viewed from the top.

use level order traversal to visit each node.

Generate vertical views for each node by keeping a horizontal distance from the root node.

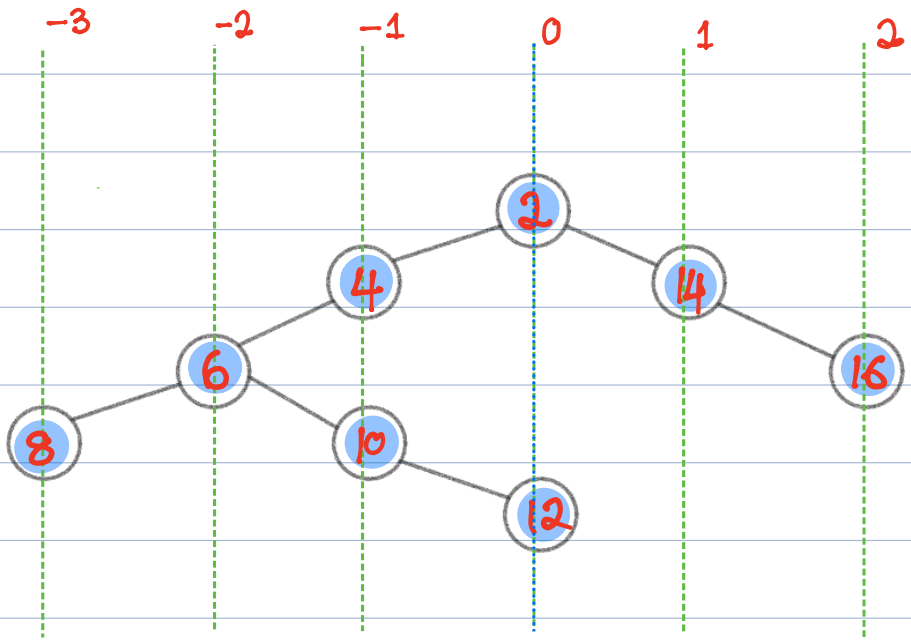
if the axis is on the right side of root node, Add (+1) to it, if on the left side add (-1).



use a dictionary to keep the nodes with their respective distance from the root node as their key and root.data as their values.

The reason to use dictionary is to avoid adding the elements on the same axis more than once.

```
def levelorder(root):
    nodemap = { }
    q = deque()

    if not root:
        return [ ]
    // Add the root node with distance being zero.
    q.append((root, 0))
    while q:
        // get the node with their horizontal distance;
        node, hd = q.popleft()

        //check if horizontal distance is there in the dictionary
        if hd not in nodemap:
            nodemap[hd] = node.data

        if node.left:
            // Add -1, if node is on the left side of poped node.
            q.append((node.left, hd-1))

        if node.right:
            q.append((node.right, hd+1))

    res = [ nodemap[key] for key in sorted(nodemap.keys() )
    return res
```



q = ~~(2,0)~~ ~~(4,-1)~~ ~~(14,+1)~~ ~~(6,-2)~~, ~~(16,+2)~~ ~~(8,-3)~~ ~~(10,-1)~~ ~~(12,0)~~

	hd	node	in nodeMap	node.left	node.right
node,hd = q.popleft(2,0),	0	2	NO	(4,-1)	(14,+1)
node,hd = q.popleft (4,-1)	-1	4	No	(6,-2)	None
node,hd = q.popleft(14,+1)	+1	14	No	None	(16,+2)
node,hd = q.popleft(6,-2)	-2	6	No	(8,-3)	(10,-1)
node,hd = q.popleft(16, +2)	+2	16	No	None	None
node,hd = q.popleft(8,-3)	-3	8	No	None	None
node,hd = q.popleft(10,-1)	-1	10	yes	None	(12,0)
node,hd = q.popleft(12,0)	0	12	yes	None	None

Queue is empty; exit the loop

nodeMap =

key (hd)	node.data
0	2
−1	4
+1	14
−2	6
+2	16
−3	8

nodeMap = {0:2, -1:4, +1:14, −2:6, +2:16, −3:8}