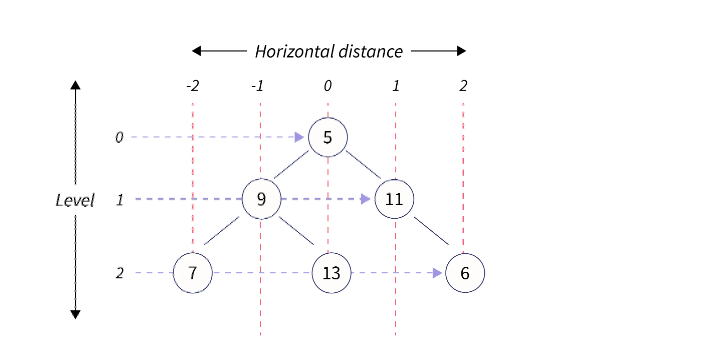
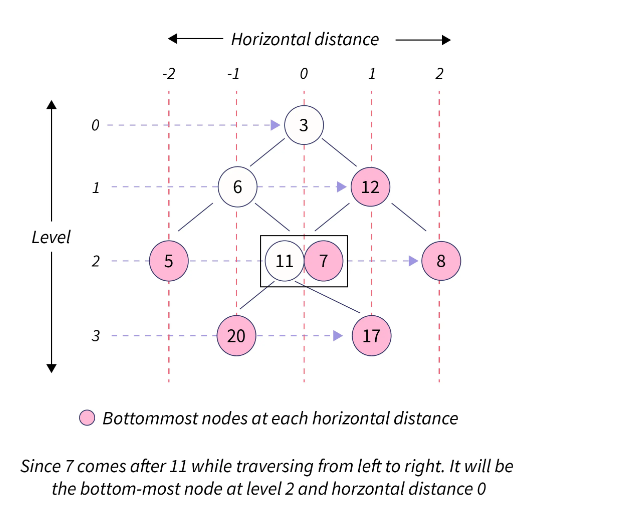


The bottom view of this binary tree will be 7 9 13 11 6.



In this case, we'll get the output as 7 9 13 11 6.

But how about a different scenario. What if there are two child nodes at the same level and horizontal distance?



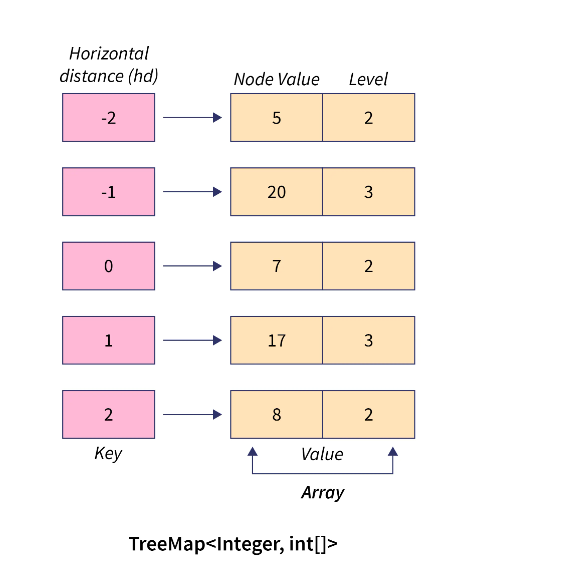
**Explanation**

Just like the x-axis, the root node is present at the origin (with horizontal distance, say hd, being 0) and at **height 0**. The immediate child nodes of the root node will be at **height 1** with the left child node having a horizontal distance of **hd-1** and the right child node having a horizontal distance of **hd+1**.

And if there are two nodes present at the same height as well as horizontal distance, the bottom-most node will be the one that comes later while traversing from left to right.

* The bottom view of a binary tree can be found in two ways:
  + HashMap + Recursion, in which the preorder traversal takes place
  + Queue, with the level order traversal (preorder traversal)

### HashMap and Recurssion Approach:



### Time Complexity

Any comparison to insert, remove, and get an element in a TreeMap takes O(logN) time. So, for N nodes in a tree, it will be:

O(N \* log N), where N is the number of nodes present in the binary tree

### Space Complexity

Since we have used a map in our code, the space complexity will be: O(N), where N is the number of nodes