# Vertical Order of a Binary Tree

Problem

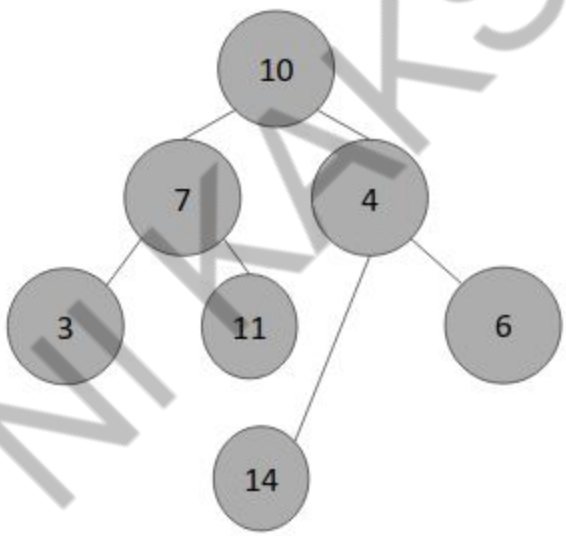
Given a binary tree, we need to print the vertical order of the binary tree.

Example

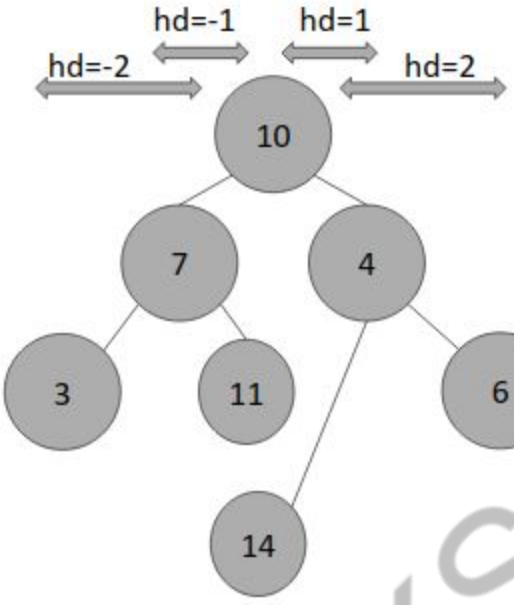
Given a binary tree in array representation

| 10 | 7 | 4 | 3 | 11 | 14 | 6 |
|---|---|---|---|---|---|---|

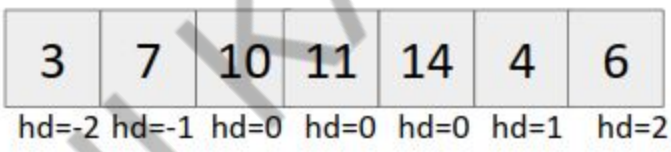
Its binary tree representation will be



Computing horizontal distance approach




Vertical Order




Approach (Using hashing)

1. Create a map (say hd) whose keys store the distance from the node and value stores a vector of nodes at the respective horizontal distance.
2. Start from the root node and recursively call left and right child with (hd-1) and (hd+1) as arguments.

   Base Case:

   *if(current_node == NULL)*

   *return;*
3. Push the value into the vector corresponding to the horizontal distance (hd).
4. Output the map

Code

```cpp
#include<bits/stdc++.h>
using namespace std;

struct Node
{
    int key;
    Node *left, *right;
};

Node* newNode(int key)
{
    Node* node = new Node;
    node->key = key;
    node->left = node->right = NULL;
    return node;
}

void getVerticalOrder(Node* root, int hdis, map<int, vector<int>> &m)
{
    if (root == NULL)
        return;

    m[hdis].push_back(root->key);

    getVerticalOrder(root->left, hdis-1, m);
    getVerticalOrder(root->right, hdis+1, m);
}
```

```cpp
int main()
{
    Node *root = newNode(10);
    root->left = newNode(7);
    root->right = newNode(4);
    root->left->left = newNode(3);
    root->left->right = newNode(11);
    root->right->left = newNode(14);
    root->right->right = newNode(6);

    map < int,vector<int> > m;
    int hdis = 0;
    getVerticalOrder(root, hdis,m);

    for (auto it:m)
    {
        for (auto x:it.second)
            cout << x << " ";
        cout << endl;
    }
    return 0;
}

// time complexity: nlog(n)
```