# Binary Tree Right Side View

Given a binary tree, imagine yourself standing on the*right* side of it, return the values of the nodes you can see ordered from top to bottom.

For example:
Given the following binary tree,

```
   1
```

You should return

```
[1, 3, 4]
```

.

**Credits:**
Special thanks to @amrsaqr for adding this problem and creating all test cases.

## Solution 1

The core idea of this algorithm:

1.Each depth of the tree only select one node.
2. View depth is current size of result list.

Here is the code:

```
public class Solution {
    public List<Integer> rightSideView(TreeNode root) {
        List<Integer> result = new ArrayList<Integer>();
        rightView(root, result, 0);
        return result;
    }

    public void rightView(TreeNode curr, List<Integer> result, int currDepth){
        if(curr == null){
            return;
        }
        if(currDepth == result.size()){
            result.add(curr.val);
        }

        rightView(curr.right, result, currDepth + 1);
        rightView(curr.left, result, currDepth + 1);

    }
}
```

written by zwangbo original link here

## Solution 2

```
class Solution {
public:
    void recursion(TreeNode *root, int level, vector<int> &res)
    {
        if(root==NULL) return ;
        if(res.size()<level) res.push_back(root->val);
        recursion(root->right, level+1, res);
        recursion(root->left, level+1, res);
    }

    vector<int> rightSideView(TreeNode *root) {
        vector<int> res;
        recursion(root, 1, res);
        return res;
    }
};
```

written by gavin5 original link here

## Solution 3

```
class Solution {
public:
    void dfs(TreeNode* root, int lv, vector<int> &res){
        if(!root)   return;
        if(lv>=res.size()) res.push_back(root->val);
        dfs(root->right,lv+1,res);
        dfs(root->left,lv+1,res);
    }

    vector<int> rightSideView(TreeNode* root) {
        vector<int> res;
        dfs(root, 0, res);
        return res;
    }
};
```

written by sjtuldk original link here

From Leetcoder.