MCS – 253P ADVANCED PROGRAMMING AND PROBLEM SOLVING

LAB 6 Program(Print Binary Tree)

Aswin Sampath
saswin@uci.edu
53844684

Date: 08/11/2023

Question:

655. Print Binary Tree

Medium 433 421

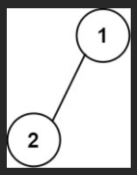
Companies

Given the `root` of a binary tree, construct a **0-indexed** `m x n` string matrix `res` that represents a **formatted layout** of the tree. The formatted layout matrix should be constructed using the following rules:

- The **height** of the tree is `height` and the number of rows `m` should be equal to `height + 1`.
- The number of columns `n` should be equal to $2^{height+1} - 1$.
- Place the **root node** in the **middle** of the **top row** (more formally, at location `res[0][(n-1)/2]`).
- For each node that has been placed in the matrix at position `res[r][c]`, place its **left child** at `res[r+1][c-2^(height-r-1)]` and its **right child** at `res[r+1][c+2^(height-r-1)]`.
- Continue this process until all the nodes in the tree have been placed.
- Any empty cells should contain the empty string `""`.

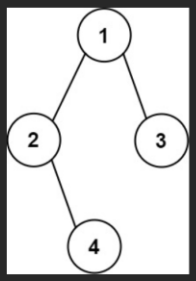
Return *the constructed matrix* `res`.

Example 1:




```
Input: root = [1,2]
Output:
[["","1",""],
 ["2","",""]]
```

Example 2:




```
Input: root = [1,2,3,null,4]
Output:
[["","","","1","","",""],
 ["","2","","","","3",""],
 ["","","4","","","",""]]
```

Code

```cpp
class Solution {
public:
    int findHeight(TreeNode* root){
        if(!root)return 0;
        if(root && root->left && root->right)return 1 + max(findHeight(root->left),findHeight(root->right));
        if(root->left)return 1 + findHeight(root->left);
        return 1+findHeight(root->right);
    }

    void assignMatrix(vector<vector<string>> &ans,TreeNode *root,int r,int c,int height){
        if(!root)return;
        ans[r][c]=to_string(root->val);
        if(root->left)assignMatrix(ans,root->left,r+1,c - pow(2,height-r-1),height);
        if(root->right)assignMatrix(ans,root->right,r+1,c+pow(2,height-r-1),height);
    }

    vector<vector<string>> printTree(TreeNode* root) {
        int m = findHeight(root);
        int height = m-1;
        int n = pow(2,m)-1;
        vector<vector<string>>ans(m,vector<string>(n,""));
        int middle = n/2;
        assignMatrix(ans,root,0,(n-1)/2,height);
        return ans;
    }
};
```

Output: