MCS – 253P ADVANCED PROGRAMMING AND PROBLEM SOLVING

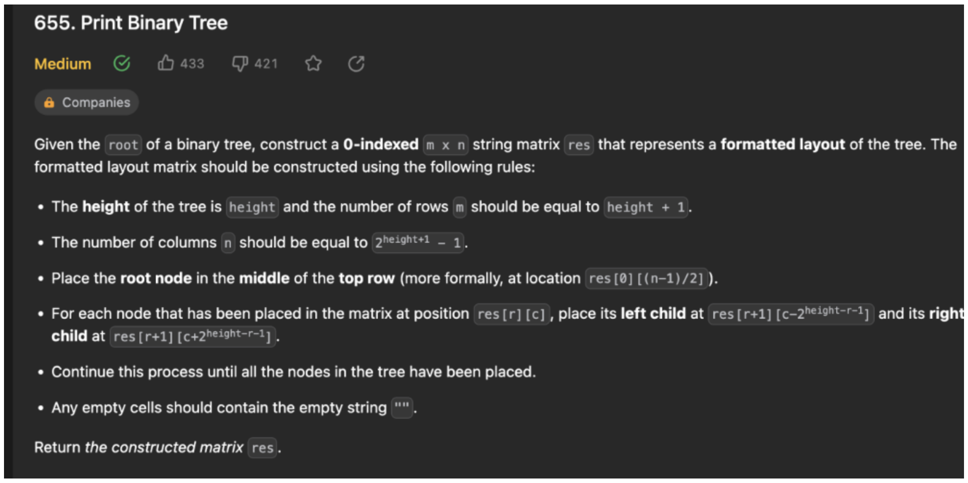
LAB 6 Write Up (Print Binary Tree)

Aswin Sampath
saswin@uci.edu

53844684

Date: 08/11/2023

Question:



655. Print Binary Tree

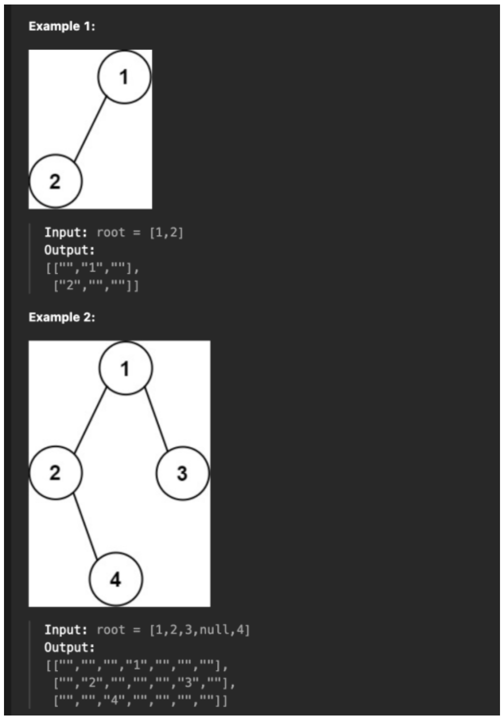
Medium 433 421

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","","","",""]]
```

Understanding the Problem

The task involves constructing a formatted layout matrix of a binary tree, given its root. The matrix must follow specific 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.
- The root node should be placed in the middle of the top row, specifically at res[0][(n-1)/2].
- For each node in the matrix at position res[r][c], its left child should be placed at res[r+1][c-2^(height-r-1)], and its right child should be placed at res[r+1][c+2^(height-r-1)].
- Empty cells in the matrix should contain the empty string "".

The goal is to construct this matrix representation of the binary tree and return it.

Identifying Edge Cases

1. An empty tree: When the root is nullptr, the formatted layout matrix should also be empty.
2. A single-node tree: For a tree with only the root node, the matrix should have one cell containing the value of the root node.

Effective Test Cases

A binary tree with multiple nodes:

Input:

```
    1
   / \
  2   3
 / \
4   5
```

Expected Output:

```
["", "", "", "1", "", "", ""]
["", "2", "", "", "", "3", ""]
["4", "", "5", "", "", "", ""]
```

A larger binary tree:
Input:

```
     1
    / \
   2   3
  / \   \
 4   5   6
/
7
```

Expected Output:

```
["", "", "", "1", "", "", ""]

["", "2", "", "", "", "3", ""]

["4", "", "5", "", "", "6", ""]

["7", "", "", "", "", "", ""]
```

An empty tree (root is nullptr):

Expected Output: An empty matrix.

A single-node tree:

Input:

```
5
```

Expected Output:

```
["5"]
```

Algorithmic Solution

1. Finding the height of the tree using the findHeight function.
2. Initializing the matrix ans of size m x n where m is the height and n is calculated based on the height.
3. Assigning values to the matrix by calling the assignMatrix function recursively.
4. In the assignMatrix function, nodes' values are placed in the matrix at the correct positions based on the rules described.
5. Finally, the constructed matrix is returned.

Time and Space Complexity Analysis

The time complexity for this algorithm is O(n), where n is the number of nodes in the binary tree. The algorithm traverses each node exactly once to construct the matrix.

The space complexity of this algorithm is also O(n) because we are storing the nodes in the matrix, where n is the number of nodes in the tree.