Cousins In Binary Tree II

solved by	Senan
Platform	LeetCode
difficulty	Medium
# Serial	2641
tags	BFS Tree
language	C++
solved on	@23/10/2024
link	https://leetcode.com/problems/cousins-in-binary-tree-ii/description/
Completion	☑

Intuition

The problem is about modifying a binary tree such that each node's value is replaced by the sum of values of nodes at the same level minus the sum of values of its sibling nodes. The root node is a special case, as its value should be set to `0`.

We traverse the tree level by level, first calculating the sum of node values for each level. Then, while traversing again, we update each node's value based on the computed sums.

Approach

1. **Level Sum Calculation**:

   First, we perform a level-order traversal to calculate the sum of node values at each level. This information will later be used to update the node values.

2. **Update Node Values**:

   We perform another level-order traversal. For each node, its value is updated by subtracting the sum of its sibling node values from the precomputed sum of the entire level.

   - For the root, we directly set its value to `0`.
   - For each child node, its new value is computed as:

     $$\text{new value} = \text{level sum} - (\text{left child's value} + \text{right child's value})$$

     This ensures that each node is updated based on the sum of all nodes at the same level except its own siblings.

Complexity

Time Complexity:

- **O(n)** where `n` is the number of nodes in the tree. We perform two level-order traversals (one to calculate sums and another to update values), each of which processes every node once.

Space Complexity:

- **O(n)** because we use a queue for the level-order traversal and an array to store the sum of each level. In the worst case (a full binary tree), the queue and array can

hold `n` elements.

Code

```
class Solution {
public:
    TreeNode* replaceValueInTree(TreeNode* root) {
        if(root == nullptr) return root;

        queue<TreeNode*> q;
        q.push(root);

        vector<int> sum

        while(!q.empty()){
            int lvlSize = q.size();
            int lvlSum = 0;
            for(int i = 0; i < lvlSize; i++){
                TreeNode* nd = q.front();
                q.pop();
                lvlSum += nd->val;
                if(nd->left) q.push(nd->left);
                if(nd->right) q.push(nd->right);
            }
            sums.push_back(lvlSum);
        }


        int currLevel = -1;
        q.push(root);
        while(!q.empty()){
            currLevel++;
            int lvlSize = q.size();
            for(int i = 0; i < lvlSize; i++){
                TreeNode* nd = q.front();
                q.pop();

                int siblingSum = 0;
                if(currLevel == 0) {
                    nd->val = 0;
                } else {
                    if(nd->left) siblingSum += nd->left->val;
                    if(nd->right) siblingSum += nd->right->val;
                }

                if(nd->left){
                    nd->left->val = sums[currLevel + 1] - siblingSum;
                    q.push(nd->left);
                }
                if(nd->right){
                    nd->right->val = sums[currLevel + 1] - siblingSum;
                    q.push(nd->right);
                }
            }
        }

        return root;
```

```
    }
};
```