Delete Nodes and Return Forest

Code

to_delete에 담긴 지울 노드들을 set에 담아 탐색시간을 줄입니다.

```
vector<TreeNode*> delNodes(TreeNode* root, vector<int>& to_delete)
{
    set<int> to_delete_set;
    vector<TreeNode*> res;

    for (auto e : to_delete)
    {
        to_delete_set.insert(e);
    }
    delNode(root, res, to_delete_set, true);
    return res;
}
```

Preorder traversal(root->left->right)를 수행하며 결과를 res에 저장합니다.

```
TreeNode* delNode(TreeNode* node, vector<TreeNode*>& res, set<int>& set, bool is_root)
{
    if (node == nullptr) return nullptr;
    bool del = set.find(node->val) != set.end();
    if (is_root && !del)
    {
        res.push_back(node);
    }
    node->left = delNode(node->left, res, set, del);
    node->right = delNode(node->right, res, set, del);
    if (del) return nullptr;
    return node;
}
```

설명

Node의 종류에는 다음과 같이 네가지가 있습니다.

1. root이고 지워져야하는 노드

-> 자식 노드는 지워지지 않는다면 root가 될 가능성이 있습니다. 현재 노드는 지워져야 하므로 nullptr을 리턴합니다.

2. root가 아니고 지워져야하는 노드

-> 자식 노드는 지워지지 않는다면 root가 될 가능성이 있습니다. 현재 노드는 지워져야 하므로 nullptr을 리턴합니다.

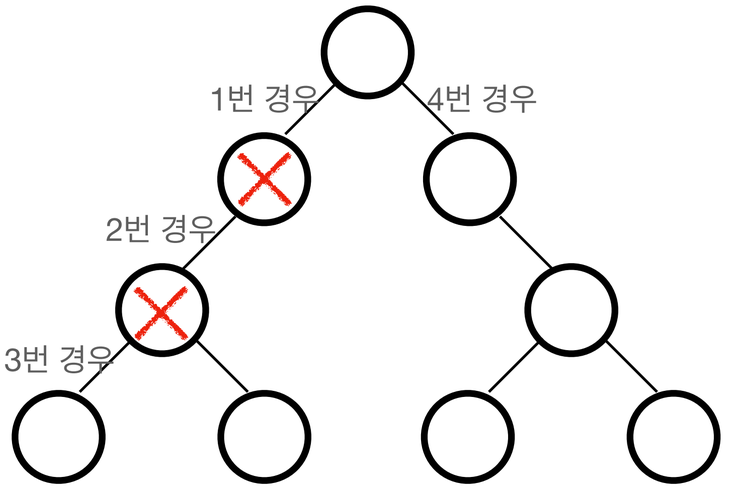
3. root이고 지워지지 않는 노드

-> 결과 Forest에 추가합니다. 리턴 시 현재 노드를 리턴합니다.

4. root가 아니고 지워지지 않는 노드

-> 자식노드는 root가 될 수 없습니다. 리턴 시 현재 노드를 리턴합니다.

* root: 결과Forest에서 부모 노드가 없는 노드



Link

https://github.com/tolerance93/LeetCode/blob/master/Delete%20Nodes%20And%20Return%20Forest/Delete%20Nodes%20And%20Return%20Forest/main.cpp