1530. Number of Good Leaf Nodes Pairs

Convert binary tree to graph, then perform BFS

```
/*
   Time complexity: O(n+n+n^2)=O(n^2)
   Extra space complexity: O(6n)
*/
class Solution {
   public:
      std::unordered_map<TreeNode*,std::vector<TreeNode*>> graph;
      std::unordered_set<TreeNode*> leafs;
   public:
      void get_leafs(TreeNode* root){
         if(!root) return;
         get_leafs(root->left);
         get_leafs(root->right);
         if(!root->left&&!root->right) leafs.insert(root);
      }

      void binary_tree_2_graph(TreeNode* root){
         if(!root) return;
         if(root->left){
            graph[root].push_back(root->left);
            graph[root->left].push_back(root);
         }

         binary_tree_2_graph(root->left);

         if(root->right){
            graph[root].push_back(root->right);
            graph[root->right].push_back(root);
         }
         binary_tree_2_graph(root->right);

      }
```

```
 int bfs(int distance){
        int ans=0;
        for(auto& leaf: leafs){
            std::queue<std::pair<TreeNode*,int>> q;
            std::unordered_map<TreeNode*,bool> visited;
            q.push({leaf,0});
            visited[leaf]=true;
            while(!q.empty()){
                std::pair<TreeNode*,int> cur=q.front();
                q.pop();

                TreeNode* cur_node=cur.first;
                int cur_level=cur.second;

                if(cur_level<=distance && cur_node!=leaf && leafs.find(cur_node)!=leafs.end()) ans++;

                if(cur_level>distance) break;

                for(auto& node: graph[cur_node]){
                    if(!visited[node]){
                        visited[node]=true;
                        q.push({node,cur_level+1});
                    }
                }
            }
        }
        return ans/2;
    }

    int countPairs(TreeNode* root, int distance){
        get_leafs(root);
        binary_tree_2_graph(root);
        return bfs(distance);
    }
};
```