1530. Number of Good Leaf Nodes Pairs

LCA approach

```
/*
   Get all leafs,
   For each pair of leafs
   ---get their LCA
   ---using that LCA to compute the distance between the pair of leafs
   ---increment the answer if the distance <= given distance

   Time complexity: O(n+3*(nb_leafs^2)*n)=O(n^3)
   Extra space complexity: O(nb_leafs+3n)=O(n)
*/
class Solution {
   public:
      std::vector<TreeNode*> leafs;
   public:
      void get_all_leafs(TreeNode* root){
         if(!root) return;

         get_all_leafs(root->left);
         get_all_leafs(root->right);

         if(!root->left&&!root->right) leafs.push_back(root);
      }

      TreeNode* LCA(TreeNode* root, TreeNode* node1, TreeNode* node2){
         if(!root) return nullptr;

         TreeNode* left=LCA(root->left,node1,node2);
         TreeNode* right=LCA(root->right,node1,node2);

         if(left&&right || root==node1 || root==node2) return root;

         return left?left:right;
      }
```

```
 void compute_distance_from_lca(TreeNode* lca,TreeNode* node,int& tmp, int& dist){
            if(!lca) return;

            if(lca==node) dist=tmp;
            else{
                tmp++;
                compute_distance_from_lca(lca->left,node,tmp,dist);
                tmp--;

                tmp++;
                compute_distance_from_lca(lca->right, node,tmp,dist);
                tmp--;
            }
        }

        int countPairs(TreeNode* root, int distance) {
            get_all_leafs(root);
            int nb_leafs=leafs.size();
            int ans=0;
            for(int i=0;i<nb_leafs-1;++i){
                for(int j=i+1;j<nb_leafs;++j){
                    TreeNode* lca=LCA(root,leafs[i],leafs[j]);
                    int dist1=0;
                    int tmp=0;
                    compute_distance_from_lca(lca,leafs[i],tmp,dist1);
                    int dist2=0;
                    compute_distance_from_lca(lca,leafs[j],tmp,dist2);
                    if(dist1+dist2<=distance) ans++;
                }
            }

            return ans;
        }
};
```