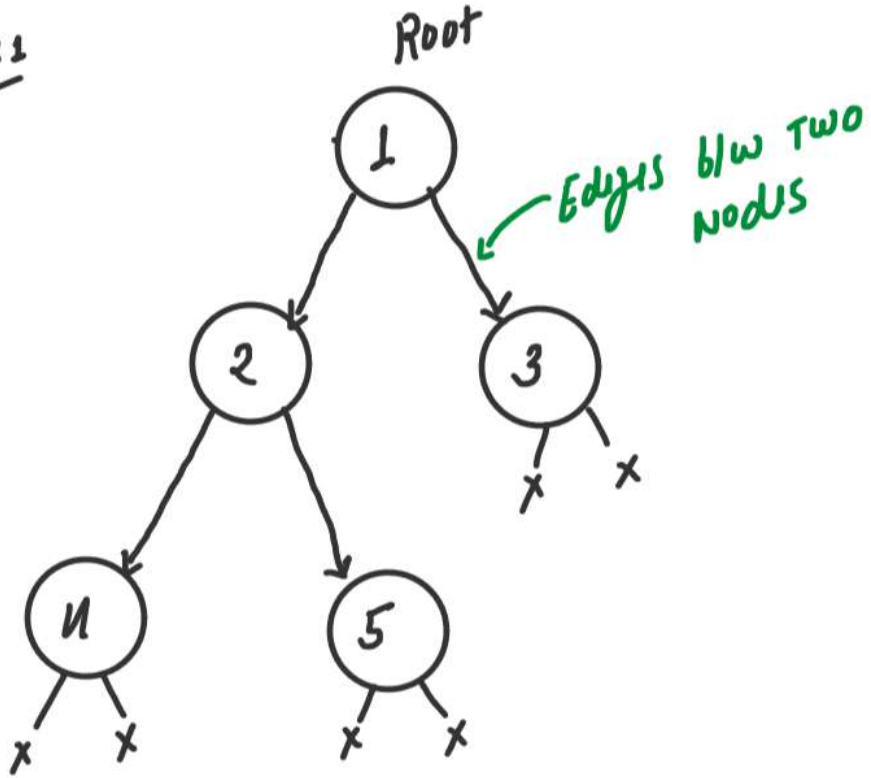
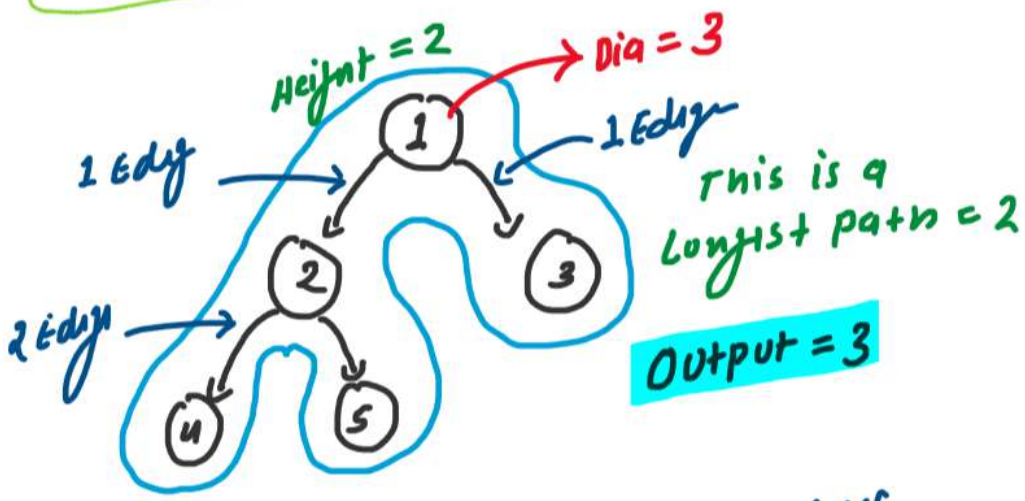
8. Diameter of Binary Tree (Leetcode-543)

Ex1



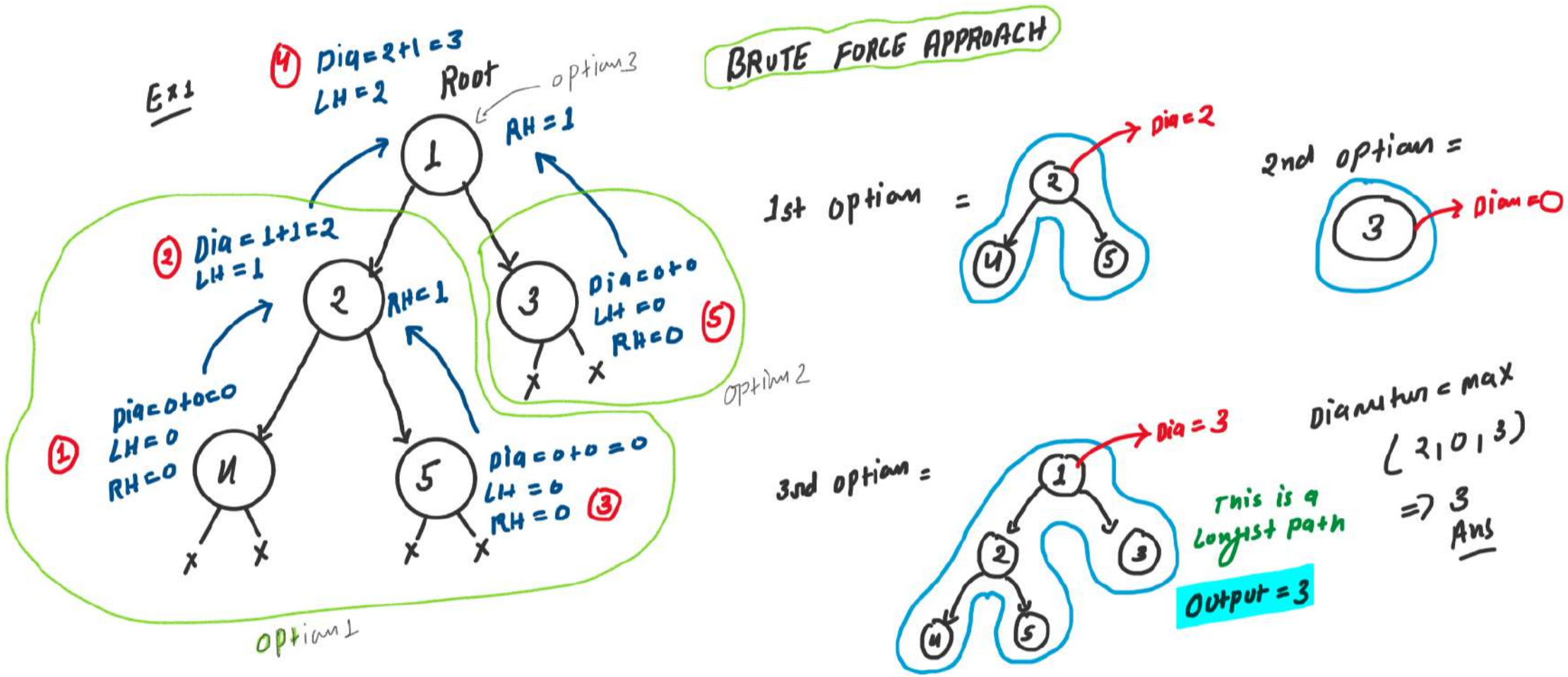
Explanation what is Diameter?



Diameter = LH edges + RH Edges

= 2 + 1

= 3



BRUTE FORCE APPROACH

1st option =

2nd option =

3rd option =

This is a Longest path

Output = 3

Diameter = max (2, 0, 3)

=> 3 Ans

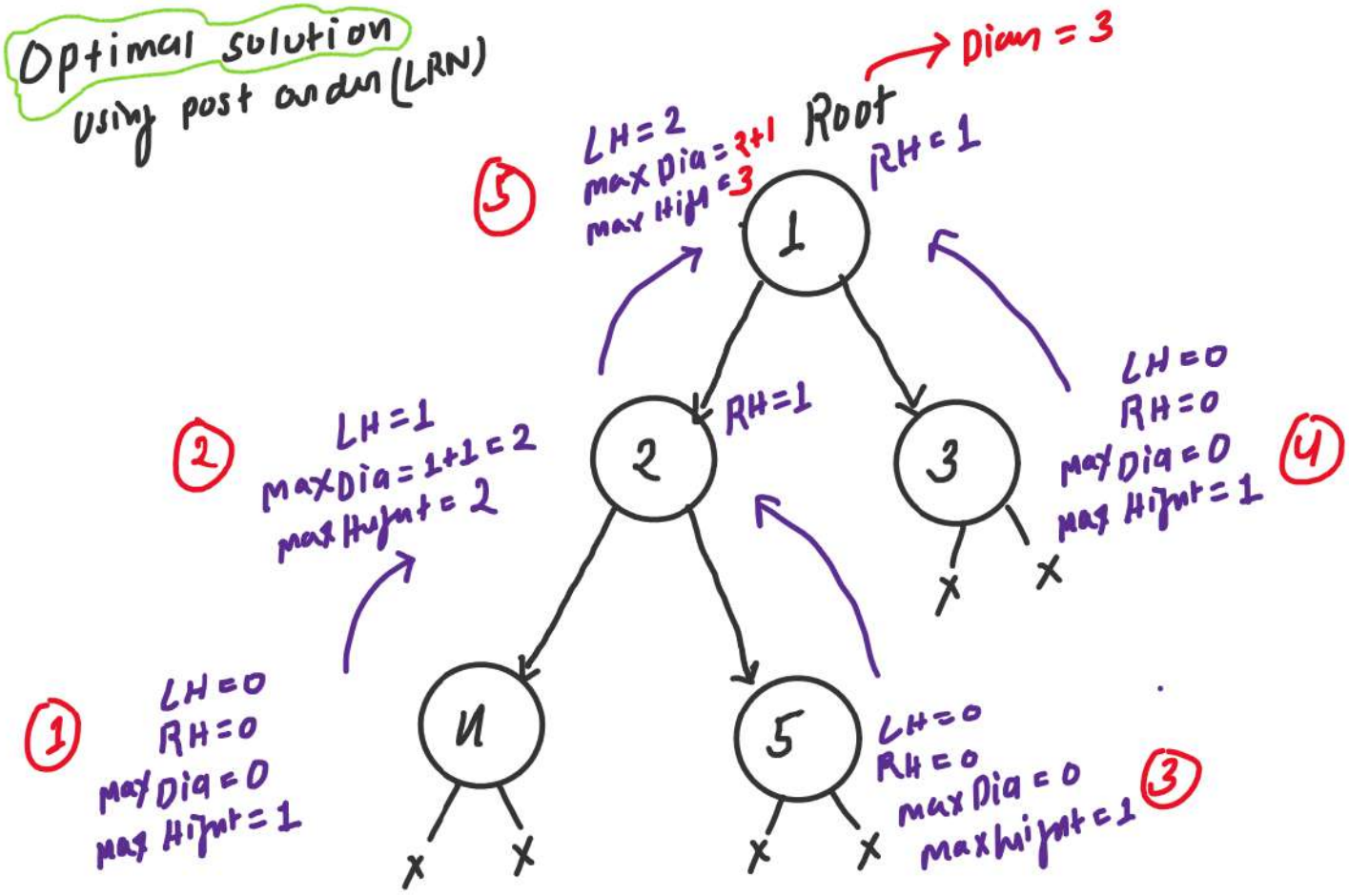
```cpp
// ✅ BRUTE FORCE APPROACH (Overhead of the recursive call)
class Solution {
public:
    int height(TreeNode* root){
        if(root == NULL) return 0;

        int LH = height(root->left);
        int RH = height(root->right);
        int finalHeight = max(LH, RH) + 1;
        return finalHeight;
    }

    int diameterOfBinaryTree(TreeNode* root) {
        //base case
        if(root == NULL) {
            return 0;
        }
        int option1 = diameterOfBinaryTree(root->left);
        int option2 = diameterOfBinaryTree(root->right);
        int option3 = height(root->left) + height(root->right);
        int diameter = max(option1, max(option2, option3));
        return diameter;
    }
};
```

Time complexity: *O(N^2), Where N number of nodes*

Why O(N^2)? *--> Height function is called by right and left subtree separately for each one node.*

Space complexity: *O(N) due to the recursive call stack*

Optimal solution
Using post order (LRN)



Diameter = max(Dia, (LH+RH))
Height = max(LH, RH) + 1

Output

Diameter = 3

Post order
↳ 4 → 2 → 5 → 3 → 1

```cpp
// ✅ OPTIMAL APPROACH (No overhead of the recursive call)
class Solution {
public:
    int height(TreeNode* root, int &diameter){
        if(root == NULL) return 0;

        int LH = height(root->left, diameter);
        int RH = height(root->right, diameter);
        int maxHeight = max(LH, RH) + 1;
        // Update diameter with longest path of tree
        diameter = max(diameter, (LH+RH));
        return maxHeight;
    }

    int diameterOfBinaryTree(TreeNode* root) {
        //base case
        if(root == NULL) {
            return 0;
        }
        int diameter = 0;
        height(root, diameter);
        return diameter;
    }
};
```

Time complexity: *O(N^2), Where N number of nodes*

Why O(N^2)? *--> Height function is not called by right and left subtree separately for each one node.*

Space complexity: *O(H), where H is the height of the binary tree*