



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

WORKSHEET 3

Student Name: Hardik Jhamb

UID: 22BCS16504

Branch: BE-CSE

Section/Group: 22BCS_NTPP-602-A

Semester: 6th

Date of Performance:

Subject Name: AP LAB - II

Subject Code: 22CSP-351

1. Aim: Longest Nice Substring , Reverse Bits , Number of 1 bits

2. Source Code:

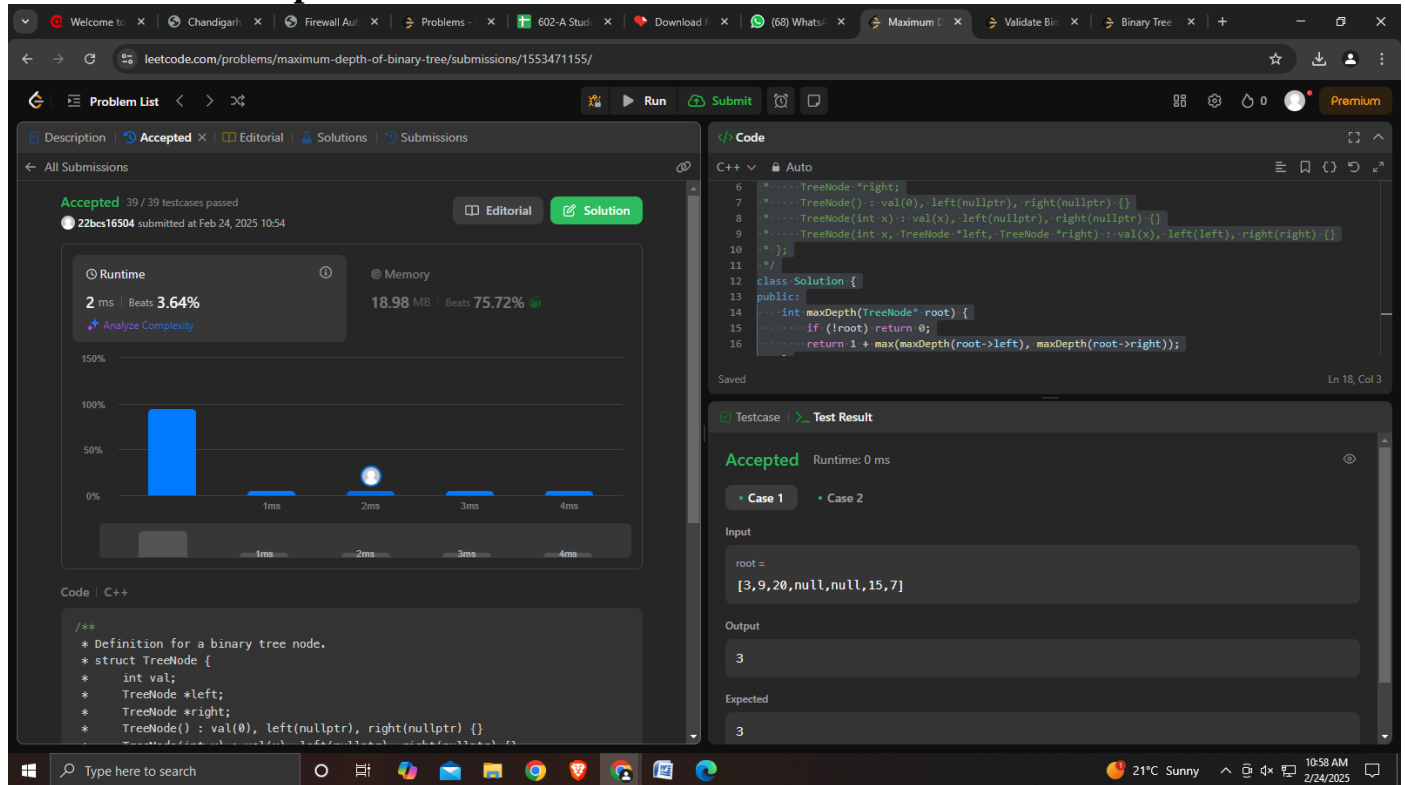
```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    int maxDepth(TreeNode* root) {
        if (!root) return 0;
        return 1 + max(maxDepth(root->left), maxDepth(root->right));
    }
};
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

3. Screenshots of outputs:



2. Aim: Given the root of a binary tree, *determine if it is a valid binary search tree (BST).*

A **valid BST** is defined as follows:

- The left

subtree

of a node contains only nodes with keys **less than** the node's key.

- The right subtree of a node contains only nodes with keys **greater than** the node's key.
- Both the left and right subtrees must also be binary search trees



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Source Code:

```
class Solution {
public:
    TreeNode*prev=nullptr;
    bool flag=true;
    void inorder(TreeNode* root)
    {
        if(root==nullptr)
            return;
        inorder(root->left);
        if(prev!=nullptr && prev->val >= root->val){
            flag=false;
            return;
        }
    }
}
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Screenshots of outputs:

The screenshot displays a web browser window with the URL `leetcode.com/problems/validate-binary-search-tree/submissions/1553471693/`. The page shows the submission details for the 'Validate Binary Search Tree' problem. The submission is marked as 'Accepted' with 86/86 testcases passed. The runtime is 0 ms, which is 100.00% faster than other submissions, and the memory usage is 22.03 MB, which is 19.96% less than other submissions. The code is written in C++ and implements an inorder traversal to validate the BST. The test case input is `[2,1,3]` and the output is `true`.

Runtime: 0 ms | Beats 100.00%
Memory: 22.03 MB | Beats 19.96%

Code:

```
class Solution {
public:
    TreeNode* prev=nullptr;
    bool flag=true;
    void inorder(TreeNode* root)
    {
        if(root==nullptr)
            return;
        inorder(root->left);
        if(prev!=nullptr && prev->val >= root->val){
            flag=false;
        }
        prev=root;
        inorder(root->right);
    }
};
```

Testcase: Case 1
Input: root = [2,1,3]
Output: true
Expected: true



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Aim: Given the root of a binary tree, return *the level order traversal of its nodes' values*. (i.e., from left to right, level by level).

Source Code:

```
class Solution {
public:
    vector<vector<int>> levelOrder(TreeNode* root) {
        if (root == nullptr)
            return {};

        vector<vector<int>> ans;
        queue<TreeNode*> q{{root}};

        while (!q.empty()) {
            vector<int> currLevel;
            for (int sz = q.size(); sz > 0; --sz) {
                TreeNode* node = q.front();
                q.pop();
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

4. Screenshots of outputs:

The screenshot displays a LeetCode submission for the problem "Binary Tree Level Order Traversal". The submission is in C++ and has been accepted, passing all 35 test cases. The runtime is 0 ms, and the memory usage is 17.14 MB. The code implements a breadth-first search (BFS) algorithm to traverse the tree level by level.

Accepted 35 / 35 testcases passed
22bes16504 submitted at Feb 24, 2025 10:55

Runtime 0 ms | Beats 100.00%
Memory 17.14 MB | Beats 44.16%

Code C++

```
class Solution {
public:
    vector<vector<int>> levelOrder(TreeNode* root) {
        if (root == nullptr)
            return {};
        vector<vector<int>> ans;
        queue<TreeNode*> q({root});
        while (!q.empty()) {
            vector<int> currLevel;
```

Testcase **Test Result**

Accepted Runtime: 0 ms

Case 1 **Case 2** **Case 3**

Input

root =
[3,9,20,null,null,15,7]

Output

[[3],[9,20],[15,7]]

Expected

[[3],[9,20],[15,7]]