Day 7 19/11/2024

1.[**31. Next Permutation**](https://leetcode.com/problems/next-permutation/)

Solved

Medium

Topics

Companies

A **permutation** of an array of integers is an arrangement of its members into a sequence or linear order.

* For example, for arr = [1,2,3], the following are all the permutations of arr: [1,2,3], [1,3,2], [2, 1, 3], [2, 3, 1], [3,1,2], [3,2,1].

The **next permutation** of an array of integers is the next lexicographically greater permutation of its integer. More formally, if all the permutations of the array are sorted in one container according to their lexicographical order, then the **next permutation** of that array is the permutation that follows it in the sorted container. If such arrangement is not possible, the array must be rearranged as the lowest possible order (i.e., sorted in ascending order).

* For example, the next permutation of arr = [1,2,3] is [1,3,2].
* Similarly, the next permutation of arr = [2,3,1] is [3,1,2].
* While the next permutation of arr = [3,2,1] is [1,2,3] because [3,2,1] does not have a lexicographical larger rearrangement.

Given an array of integers nums, *find the next permutation of* nums.

The replacement must be [**in place**](http://en.wikipedia.org/wiki/In-place_algorithm) and use only constant extra memory.

**Example 1:**

**Input:** nums = [1,2,3]

**Output:** [1,3,2]

**Example 2:**

**Input:** nums = [3,2,1]

**Output:** [1,2,3]

**Example 3:**

**Input:** nums = [1,1,5]

**Output:** [1,5,1]

**Constraints:**

* 1 <= nums.length <= 100
* 0 <= nums[i] <= 100

Code:

class Solution {

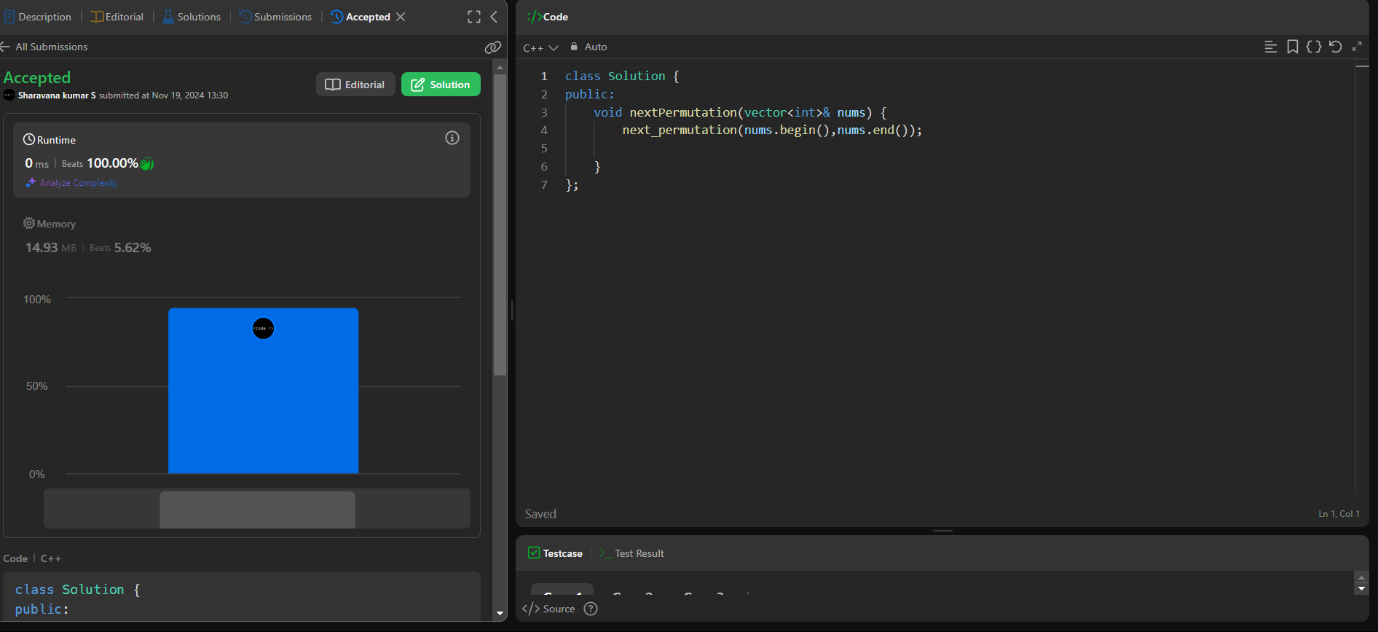
public:

    void nextPermutation(vector<int>& nums) {

        next\_permutation(nums.begin(),nums.end());

    }

};



2.[**54. Spiral Matrix**](https://leetcode.com/problems/spiral-matrix/)

Solved

Medium

Topics

Companies

Hint

Given an m x n matrix, return *all elements of the* matrix *in spiral order*.

**Example 1:**



**Input:** matrix = [[1,2,3],[4,5,6],[7,8,9]]

**Output:** [1,2,3,6,9,8,7,4,5]

**Example 2:**



**Input:** matrix = [[1,2,3,4],[5,6,7,8],[9,10,11,12]]

**Output:** [1,2,3,4,8,12,11,10,9,5,6,7]

**Constraints:**

* m == matrix.length
* n == matrix[i].length
* 1 <= m, n <= 10
* -100 <= matrix[i][j] <= 100

Code:

class Solution {

public:

    vector<int> spiralOrder(vector<vector<int>>& matrix) {

        int r=matrix.size();

        int c=matrix[0].size();

        vector<int> ans;

        int top=0;

        int left=0;

        int right=c-1;

        int bottom=r-1;

        while(top<=bottom && left<=right){

            for(int i=left;i<=right;i++){

                ans.push\_back(matrix[top][i]);

            }

            top++;

            for(int i=top;i<=bottom;i++){

                ans.push\_back(matrix[i][right]);

            }

            right--;

            if(top<=bottom){

                for(int i=right;i>=left;i--){

                    ans.push\_back(matrix[bottom][i]);

                }

                bottom--;

            }

            if(left<=right){

                for(int i=bottom;i>=top;i--){

                    ans.push\_back(matrix[i][left]);

                }

                left++;

            }

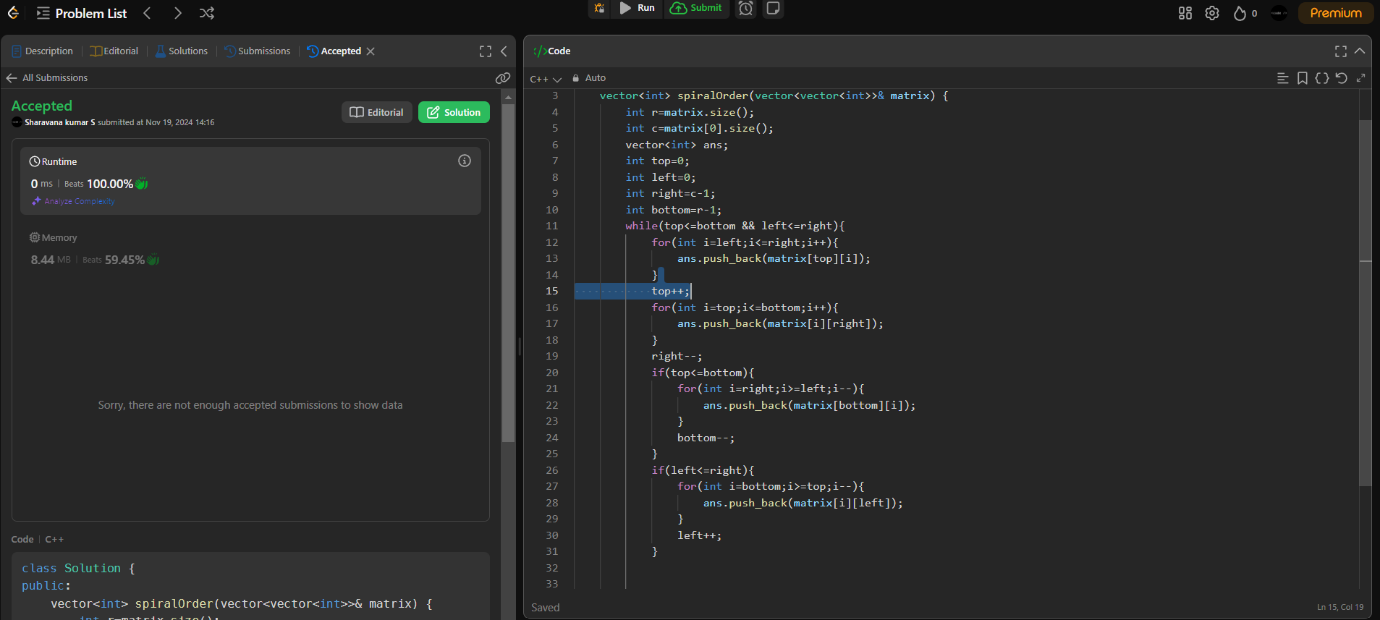
        }

        return ans;

    }

};

Output:



3.[**. Longest Substring Without Repeating Characters**](https://leetcode.com/problems/longest-substring-without-repeating-characters/)

Solved

Medium

Topics

Companies

Hint

Given a string s, find the length of the **longest**

**substring**

 without repeating characters.

**Example 1:**

**Input:** s = "abcabcbb"

**Output:** 3

**Explanation:** The answer is "abc", with the length of 3.

**Example 2:**

**Input:** s = "bbbbb"

**Output:** 1

**Explanation:** The answer is "b", with the length of 1.

**Example 3:**

**Input:** s = "pwwkew"

**Output:** 3

**Explanation:** The answer is "wke", with the length of 3.

Notice that the answer must be a substring, "pwke" is a subsequence and not a substring.

**Constraints:**

* 0 <= s.length <= 5 \* 104
* s consists of English letters, digits, symbols and spaces.

Code:

class Solution {

public:

    int lengthOfLongestSubstring(string s) {

        int n = s.size();

        int i = 0, j = 0;

        int l = 0;

        unordered\_map<char, int> m;

        while (j < n) {

            if(m.find(s[j])!=m.end()){

                if(m[s[j]]>=i){

                  i=m[s[j]]+1;

                }

            }

            l=max(l,j-i+1);

            m[s[j]]=j;

            j+=1;

        }

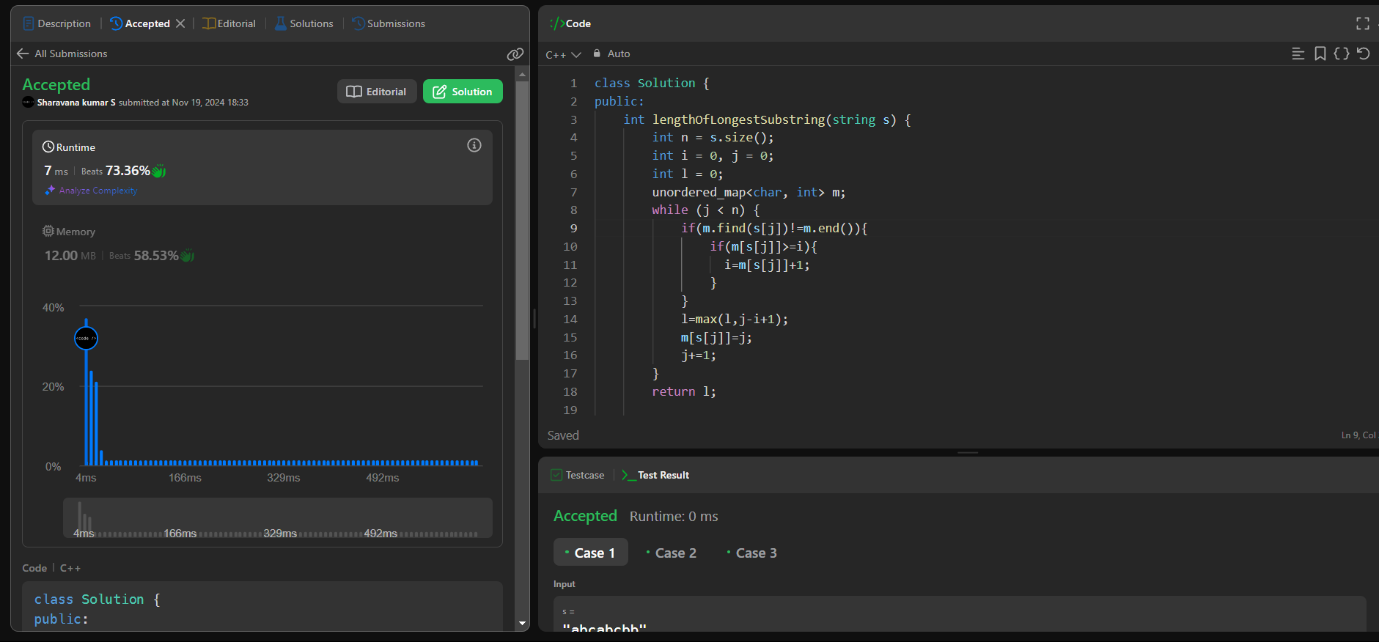
        return l;

        return l;

    }

};

Output:



4.[**203. Remove Linked List Elements**](https://leetcode.com/problems/remove-linked-list-elements/)

Solved

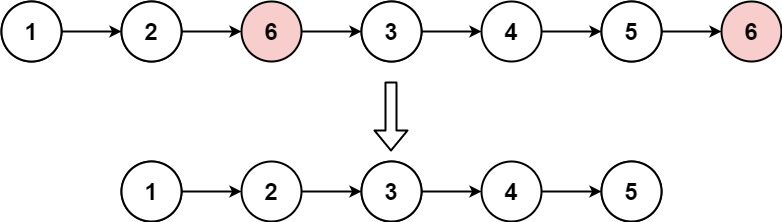
Easy

Topics

Companies

Given the head of a linked list and an integer val, remove all the nodes of the linked list that has Node.val == val, and return *the new head*.

**Example 1:**



**Input:** head = [1,2,6,3,4,5,6], val = 6

**Output:** [1,2,3,4,5]

**Example 2:**

**Input:** head = [], val = 1

**Output:** []

**Example 3:**

**Input:** head = [7,7,7,7], val = 7

**Output:** []

**Constraints:**

* The number of nodes in the list is in the range [0, 104].
* 1 <= Node.val <= 50
* 0 <= val <= 50

Code:

/\*\*

 \* Definition for singly-linked list.

 \* struct ListNode {

 \*     int val;

 \*     ListNode \*next;

 \*     ListNode() : val(0), next(nullptr) {}

 \*     ListNode(int x) : val(x), next(nullptr) {}

 \*     ListNode(int x, ListNode \*next) : val(x), next(next) {}

 \* };

 \*/

class Solution {

public:

    ListNode\* removeElements(ListNode\* head, int val) {

        vector<int> v;

        ListNode\* temp=head;

        ListNode\* one = new ListNode(0);

        ListNode\* temp2=one;

        while(temp){

            if(temp->val!=val){

                one->next=new ListNode(temp->val);

                one=one->next;

            }

            temp=temp->next;

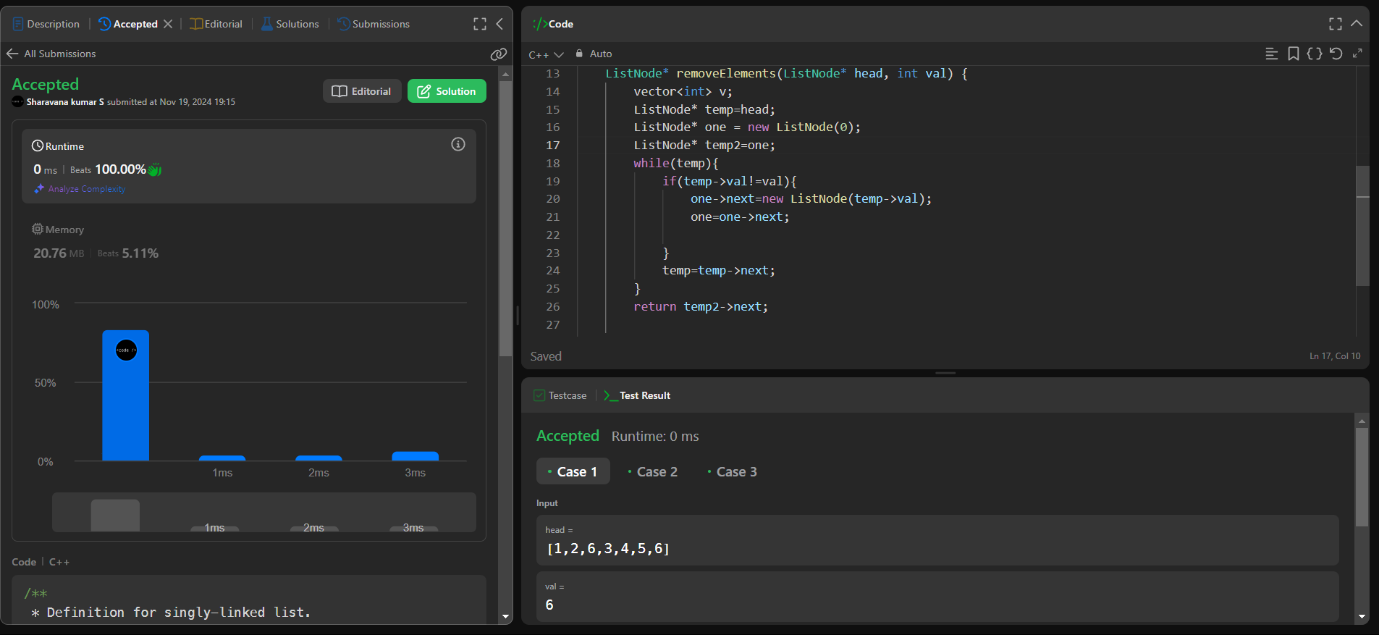
        }

        return temp2->next;

    }

};

Output:



5.[**234. Palindrome Linked List**](https://leetcode.com/problems/palindrome-linked-list/)

Solved

Easy

Topics

Companies

Given the head of a singly linked list, return true*if it is a*

*palindrome*

*or*false*otherwise*.

**Example 1:**



**Input:** head = [1,2,2,1]

**Output:** true

**Example 2:**



**Input:** head = [1,2]

**Output:** false

**Constraints:**

* The number of nodes in the list is in the range [1, 105].
* 0 <= Node.val <= 9

**Follow up:** Could you do it in O(n) time and O(1) space?

Code:

/\*\*

 \* Definition for singly-linked list.

 \* struct ListNode {

 \*     int val;

 \*     ListNode \*next;

 \*     ListNode() : val(0), next(nullptr) {}

 \*     ListNode(int x) : val(x), next(nullptr) {}

 \*     ListNode(int x, ListNode \*next) : val(x), next(next) {}

 \* };

 \*/

class Solution {

public:

    bool isPalindrome(ListNode\* head) {

        stack<int> v;

        ListNode\* temp=head;

        while(temp){

            v.push(temp->val);

            temp=temp->next;

        }

        temp=head;

        while(temp){

            if(temp->val!=v.top()){

                return false;

            }

            temp=temp->next;

            v.pop();

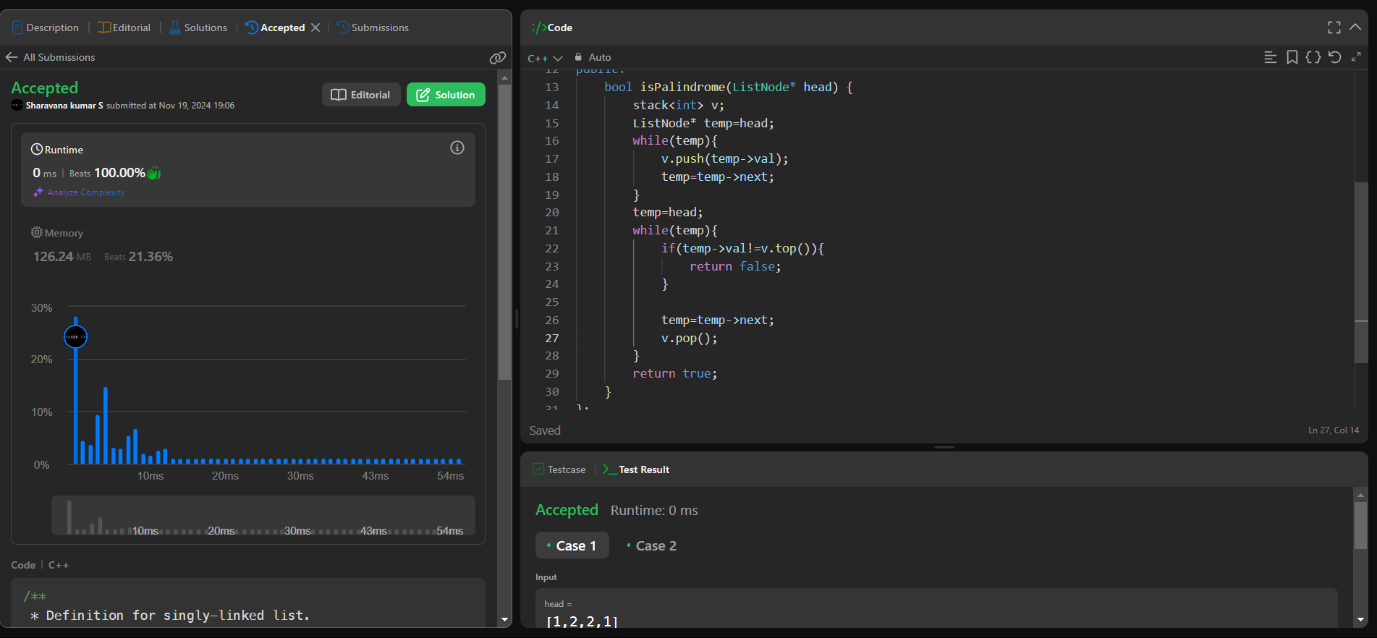
        }

        return true;

    }

};

Output:



6.[**64. Minimum Path Sum**](https://leetcode.com/problems/minimum-path-sum/)

Solved

Medium

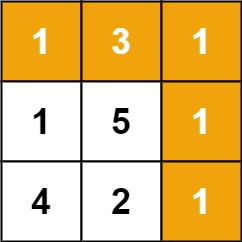
Topics

Companies

Given a m x n grid filled with non-negative numbers, find a path from top left to bottom right, which minimizes the sum of all numbers along its path.

**Note:** You can only move either down or right at any point in time.

**Example 1:**



**Input:** grid = [[1,3,1],[1,5,1],[4,2,1]]

**Output:** 7

**Explanation:** Because the path 1 → 3 → 1 → 1 → 1 minimizes the sum.

**Example 2:**

**Input:** grid = [[1,2,3],[4,5,6]]

**Output:** 12

**Constraints:**

* m == grid.length
* n == grid[i].length
* 1 <= m, n <= 200
* 0 <= grid[i][j] <= 200

Code;

class Solution {

public:

    int minPathSum(vector<vector<int>>& grid) {

         int m = grid.size();

        int n = grid[0].size();

        vector<vector<int>> dp(m, vector<int>(n, 0));

        for (int i = 0; i < m; i++) {

            for (int j = 0; j < n; j++) {

                if (i == 0 && j == 0) {

                    dp[i][j] = grid[i][j];

                } else {

                    int up = (i > 0) ? dp[i - 1][j] : INT\_MAX;

                    int left = (j > 0) ? dp[i][j - 1] : INT\_MAX;

                    dp[i][j] = grid[i][j] + min(up, left);

                }

            }

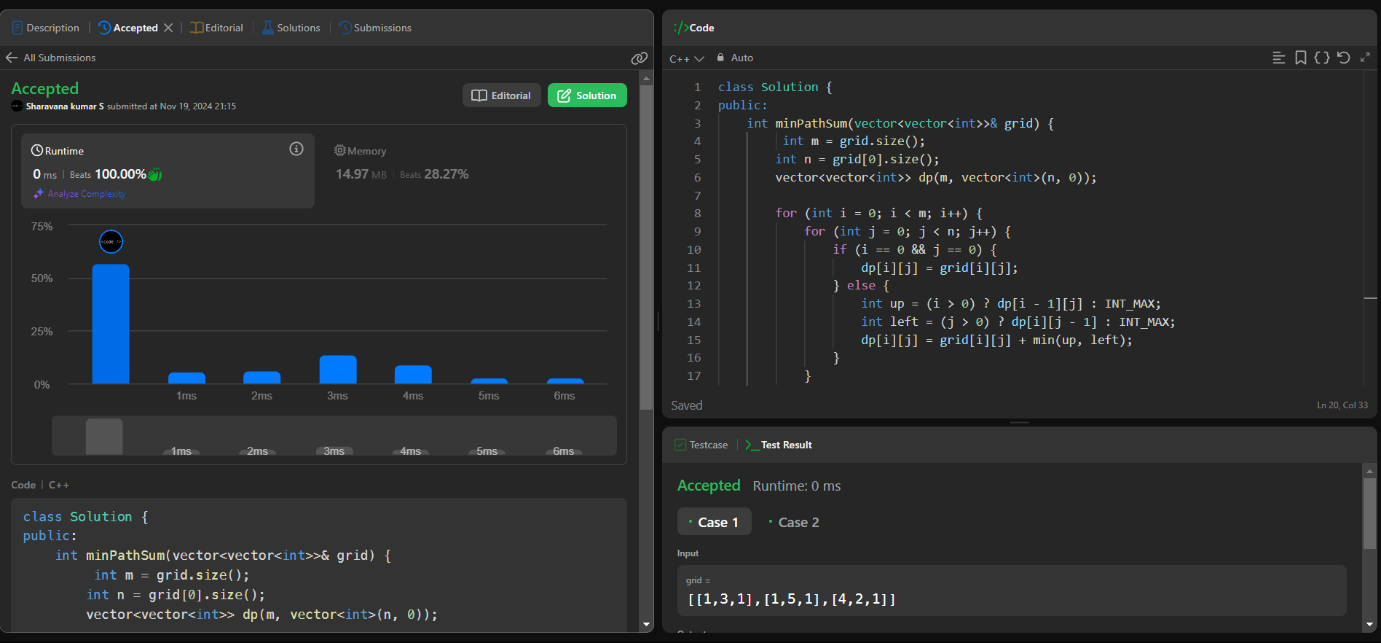
        }

        return dp[m - 1][n - 1];

    }

};

Output:



7.[**98. Validate Binary Search Tree**](https://leetcode.com/problems/validate-binary-search-tree/)

Solved

Medium

Topics

Companies

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.

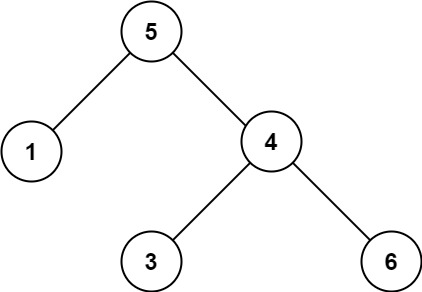
**Example 1:**



**Input:** root = [2,1,3]

**Output:** true

**Example 2:**



**Input:** root = [5,1,4,null,null,3,6]

**Output:** false

**Explanation:** The root node's value is 5 but its right child's value is 4.

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 {

    bool dfs(TreeNode\* root,long low,long high){

        if(root==NULL){

            return true;

        }

        if(root->val<=low || root->val>=high){

            return false;

        }

        bool left=dfs(root->left,low,root->val);

        bool right=dfs(root->right,root->val,high);

        return left && right;

    }

public:

    bool isValidBST(TreeNode\* root) {

        return dfs(root,LONG\_MIN,LONG\_MAX);

    }

};

Output:

