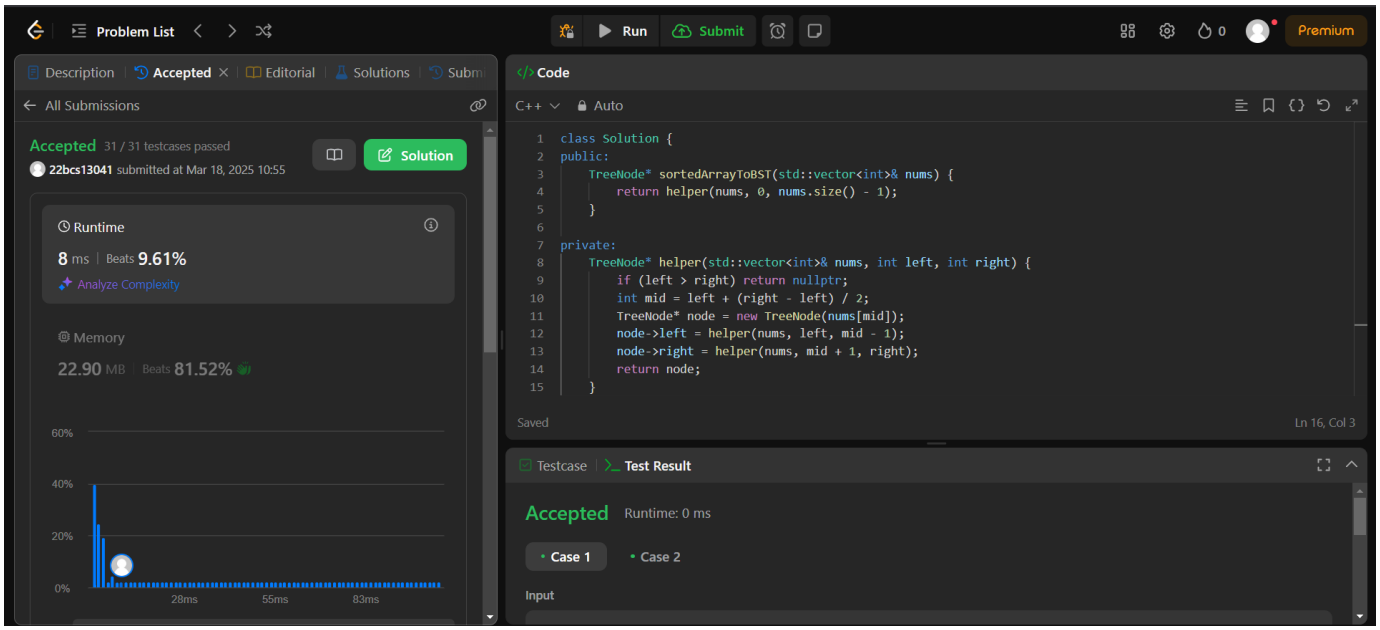
ASSIGNMENT 6

STUDENT NAME: Pranjal Singh
UID: 22BCS13041
BRANCH: CSE
SECTION: 22BCS_FL_IOT_601A
SEMESTER: 6
DATE OF SUBMISSION: 16/3/25
SUBJECT NAME: AP LAB -2
SUBJECT CODE: 22CSP-351

LEET CODE QUESTIONS :

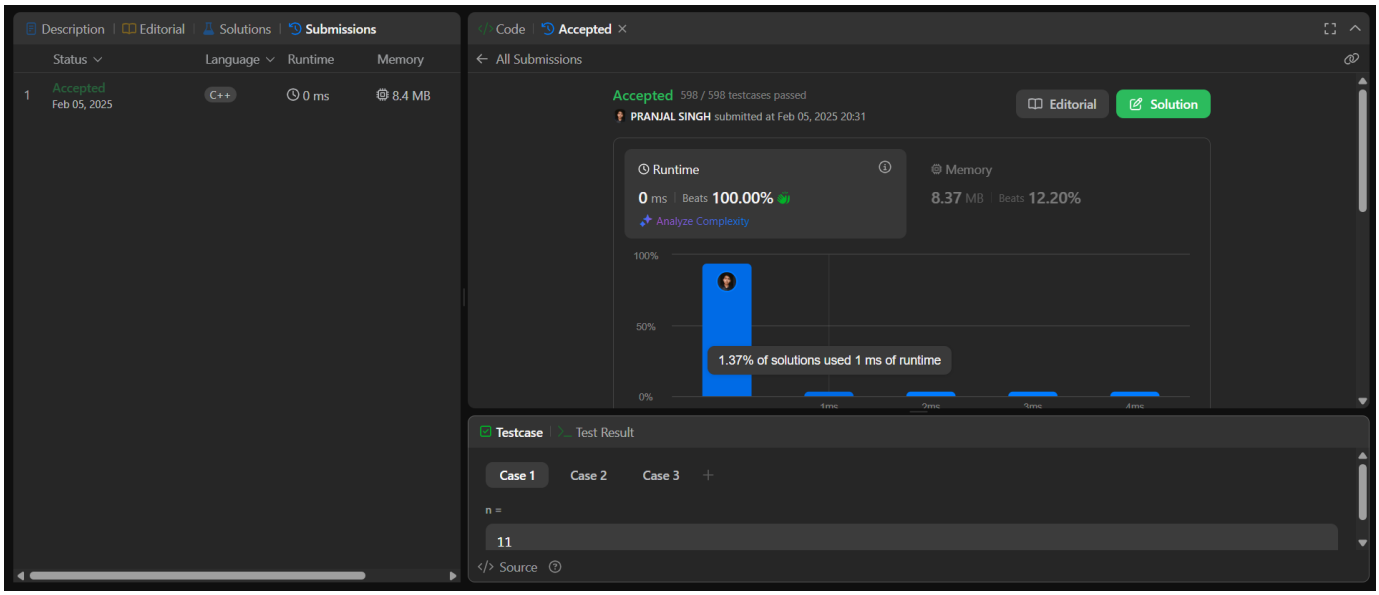
108.CONVERT SORTED ARRAY TO BINARY SEARCH TREE class

```
class Solution {
public:
    TreeNode* sortedArrayToBST(std::vector<int>& nums) {
        return helper(nums, 0, nums.size() - 1);
    }

private:
    TreeNode* helper(std::vector<int>& nums, int left, int right) {
        if (left > right) return nullptr;
        int mid = left + (right - left) / 2;
        TreeNode* node = new TreeNode(nums[mid]);
        node->left = helper(nums, left, mid - 1);
        node->right = helper(nums, mid + 1, right);
        return node;
    }
};
```



191.NUMBER OF 1 BITS

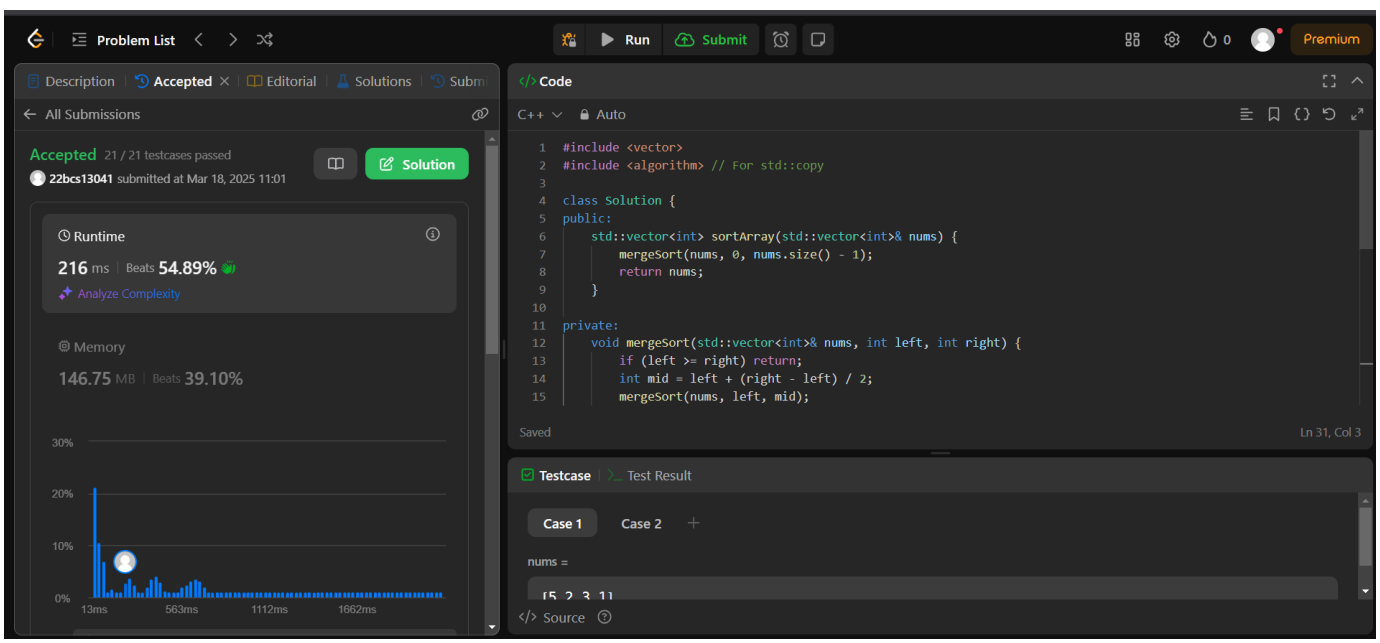
```
class Solution {
public:
    int hammingWeight(int n) {
        int count=0;
        while(n!=0){
            if(n%2!=0) count++;
            n=n>>1;
        }
        return count;
    }
};
```



912.[SORT AN ARRAY]

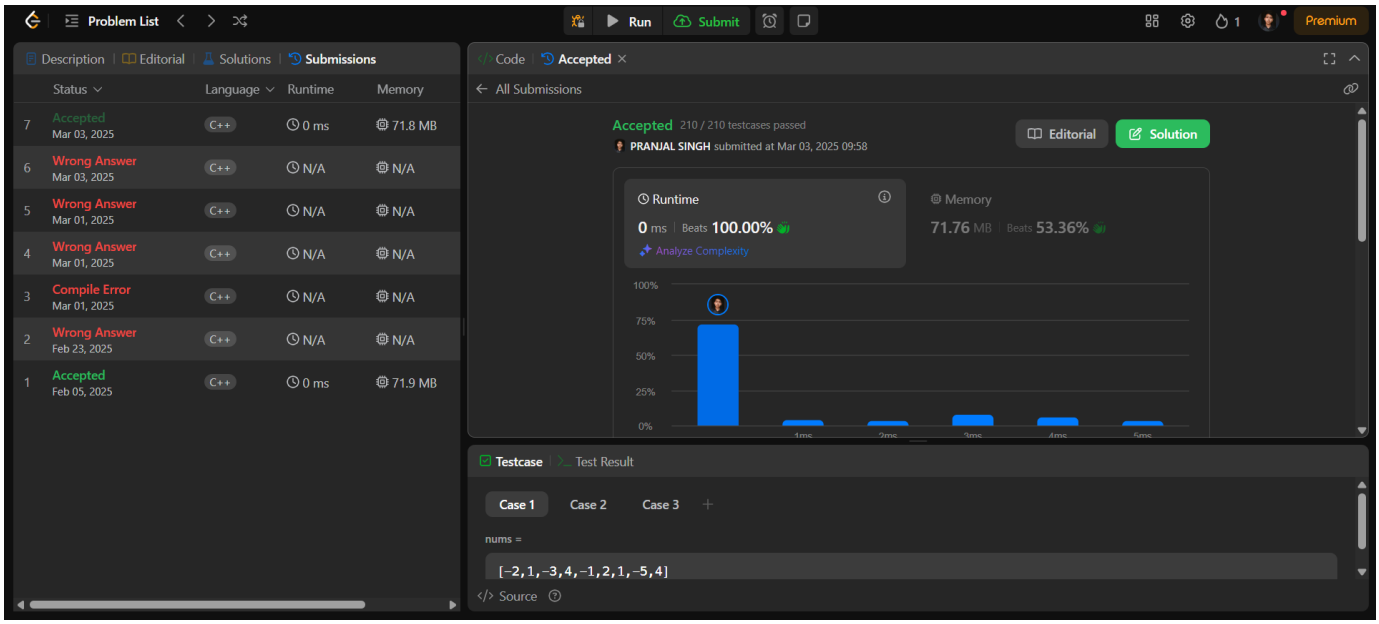
```
#include <vector>
#include <algorithm> // For std::copy

class Solution {
public:
    std::vector<int> sortArray(std::vector<int>& nums) {
        mergeSort(nums, 0, nums.size() - 1);
        return nums;
    }

private:
    void mergeSort(std::vector<int>& nums, int left, int right) {
        if (left >= right) return;
        int mid = left + (right - left) / 2;
        mergeSort(nums, left, mid);
        mergeSort(nums, mid + 1, right);
        merge(nums, left, mid, right);
    }
```

```
    void merge(std::vector<int>& nums, int left, int mid, int right) {
        std::vector<int> temp(right - left + 1);
        int i = left, j = mid + 1, k = 0;
        while (i <= mid && j <= right) {
            if (nums[i] <= nums[j]) temp[k++] = nums[i++];
            else temp[k++] = nums[j++];
        }
        while (i <= mid) temp[k++] = nums[i++];
        while (j <= right) temp[k++] = nums[j++];
        std::copy(temp.begin(), temp.end(), nums.begin() + left);
    }
};
```



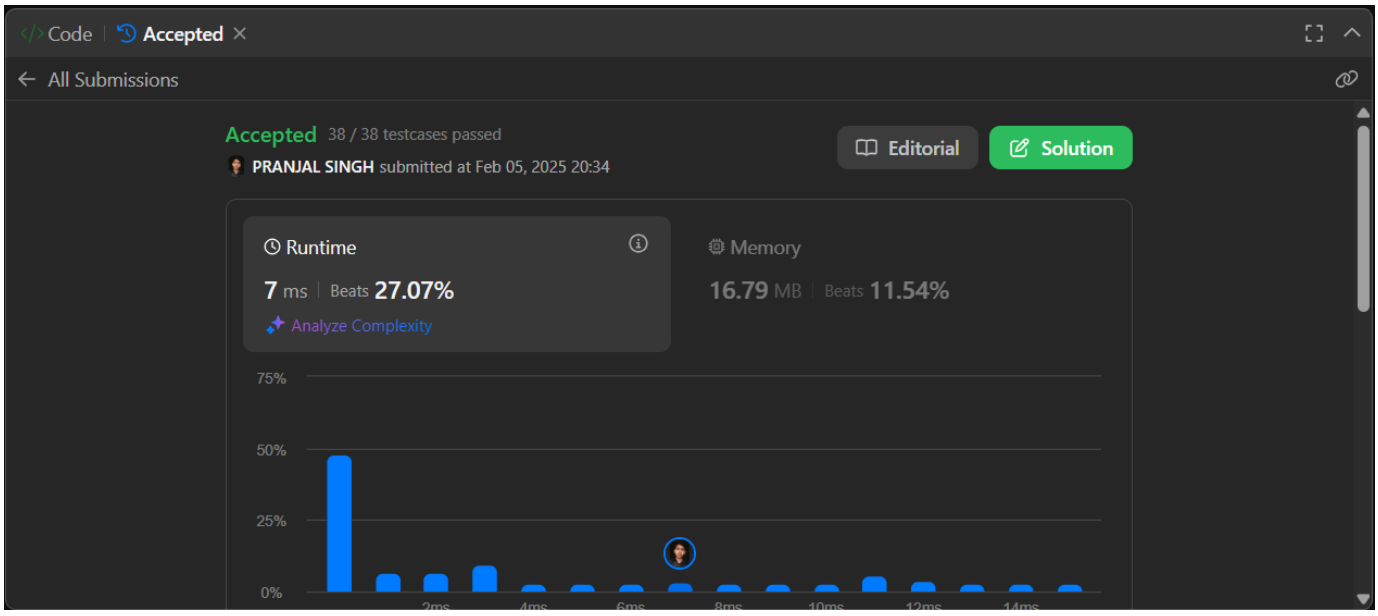
53.MAXIMUM SUBARRAY

```
class Solution {
public:
    int maxSubArray(vector<int>& nums) {
        // Kadane's Algo...
        int maximumsum=INT_MIN;
        int currentsum=0;
        for(int i =0;i<nums.size();i++){
            currentsum=currentsum+nums[i];
            maximumsum=max(currentsum,maximumsum);
            if(currentsum<0){
                currentsum=0;
            }
        }
        return maximumsum;
    }
};
```



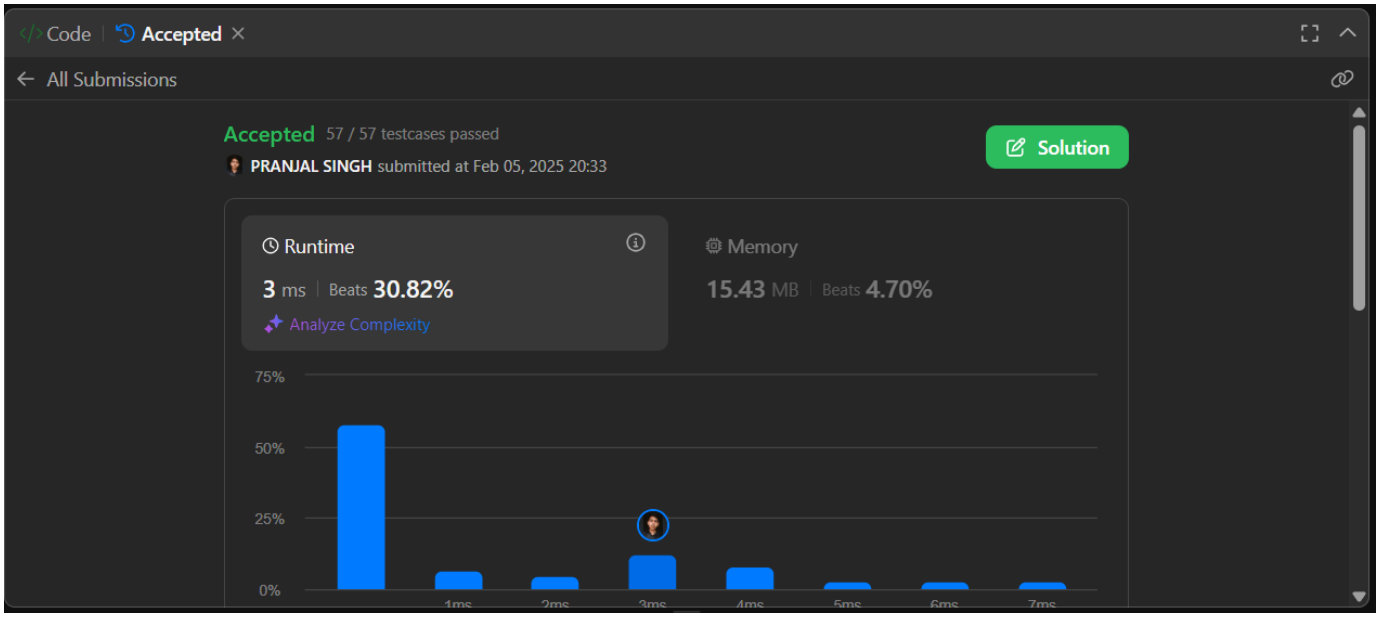
932.BEAUTIFUL ARRAY

```
class Solution {
public:
    vector<int> beautifulArray(int n) {
        if (n == 1) return {1};
        vector<int> left = beautifulArray((n + 1) >> 1);
        vector<int> right = beautifulArray(n >> 1);
        vector<int> ans(n);
        int i = 0;
        for (int& x : left) ans[i++] = x * 2 - 1;
        for (int& x : right) ans[i++] = x * 2;
        return ans;
    }
};
```



372.SUPER POW

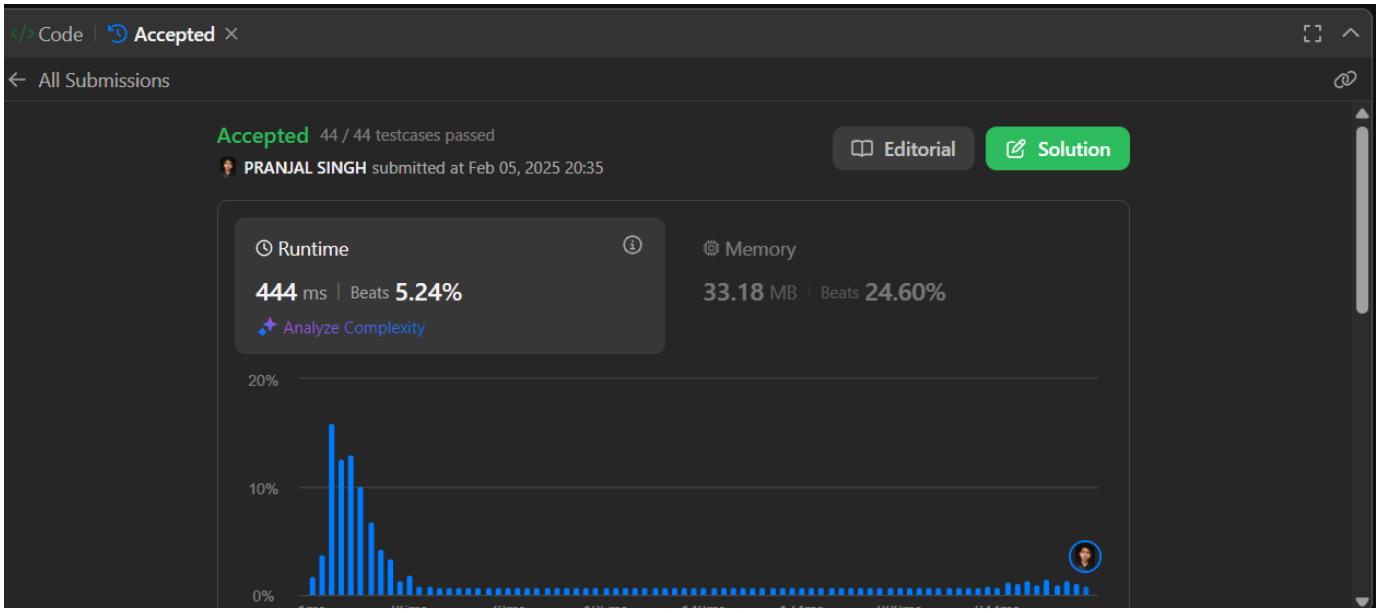
```
class Solution {
    const int base = 1337;
    int powmod(int a, int k){
        a %= base;
        int result = 1;
        for (int i = 0; i < k; ++i)  result = (result * a) % base;
        return result;
    }

public:
    int superPow(int a, vector<int>& b) {
        if (b.empty()) return 1;
        int last_digit = b.back();
        b.pop_back();
        return powmod(superPow(a, b), 10) * powmod(a, last_digit) % base;
    }
};
```



218.THE SKYLINE PROBLEM

```
class Solution {
public:
    vector<vector<int>> getSkyline(vector<vector<int>>& buildings) {
        set<int> poss;
        map<int, int> m;
        for (auto v : buildings) {
            poss.insert(v[0]);
            poss.insert(v[1]);
        }

        int i = 0;
        for (int pos : poss)
            m.insert(pair<int, int>(pos, i++));

        vector<int> highs(m.size(), 0);
        for (auto v : buildings) {
            const int b = m[v[0]], e = m[v[1]];
            for (int i = b; i < e; ++i)
                highs[i] = max(highs[i], v[2]);
        }

        vector<vector<int>> res;
        vector<int> mm(poss.begin(), poss.end());
        for (int i = 0; i < highs.size(); i++) {
            if (i+1 < highs.size() && highs[i] != highs[i + 1])
                res.push_back({mm[i], highs[i]});
            else {
                const int start = i;
                res.push_back({mm[start], highs[i]});
                while (i+1 < highs.size() && highs[i] == highs[i + 1])
                    ++i;
            }
```

```
        };
        return res;
    }
};
```


Code
Accepted
All Submissions
Accepted 44 / 44 testcases passed
PRANJAL SINGH submitted at Feb 05, 2025 20:35
Editorial
Solution
Runtime
444 ms | Beats 5.24%
Analyze Complexity
Memory
33.18 MB | Beats 24.60%
20%
10%
0%