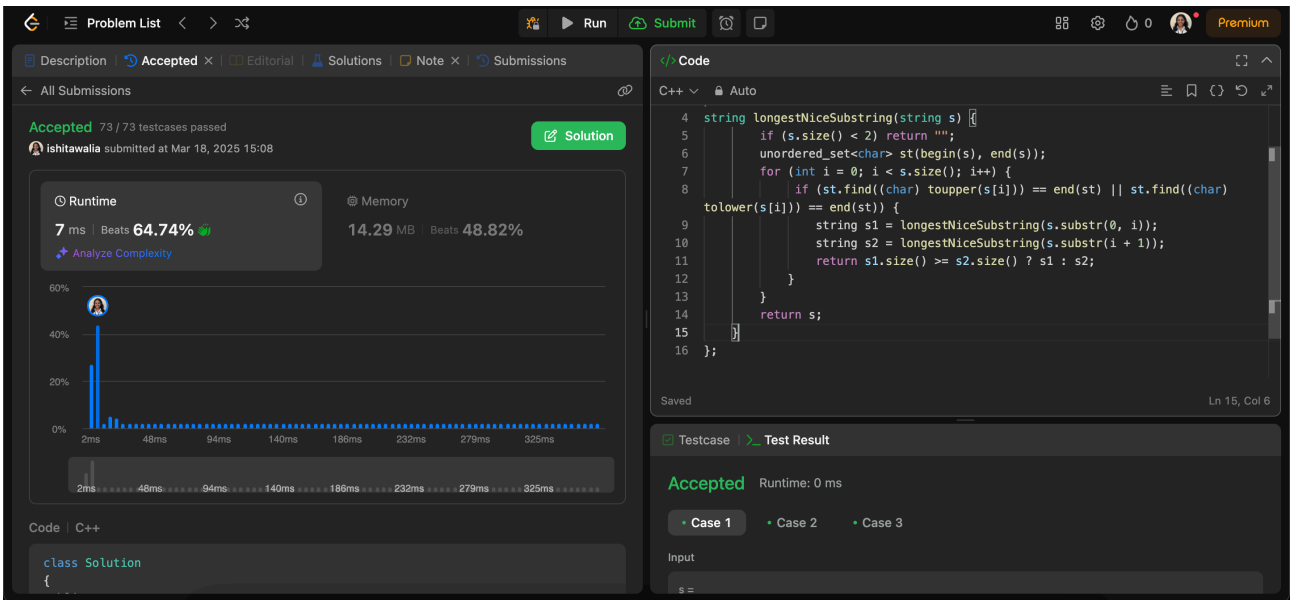
Name : Ishita	**Date of Submission : 18/03/2025**
UID : 22BCS14845	**Subject Code : 22CSP - 351**
Subject : Advance Programming Lab	**Submitted to : Er. Pratima Sonali**

ASSIGNMENT - 3

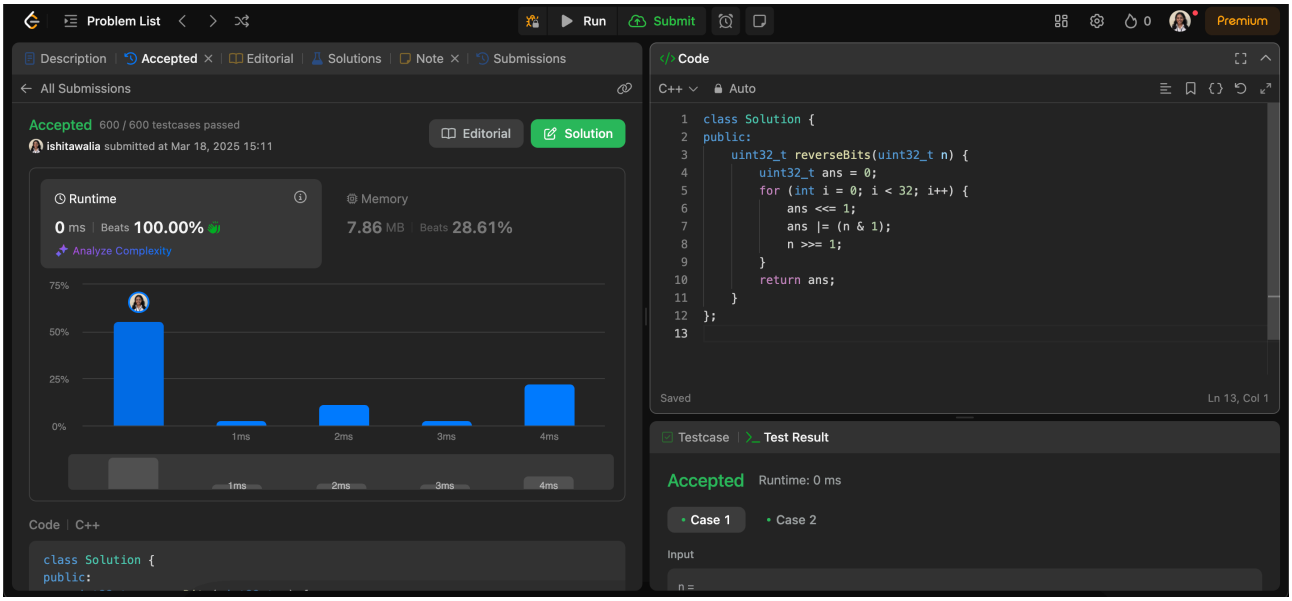
Question 1

```
class Solution
{
public:
string longestNiceSubstring(string s) {
        if (s.size() < 2) return "";
        unordered_set<char> st(begin(s), end(s));
        for (int i = 0; i < s.size(); i++) {
            if (st.find((char) toupper(s[i])) == end(st) || st.find((char) tolower(s[i])) == end(st)) {
                string s1 = longestNiceSubstring(s.substr(0, i));
                string s2 = longestNiceSubstring(s.substr(i + 1));
                return s1.size() >= s2.size() ? s1 : s2;
            }
        }
        return s;
    }
};
```



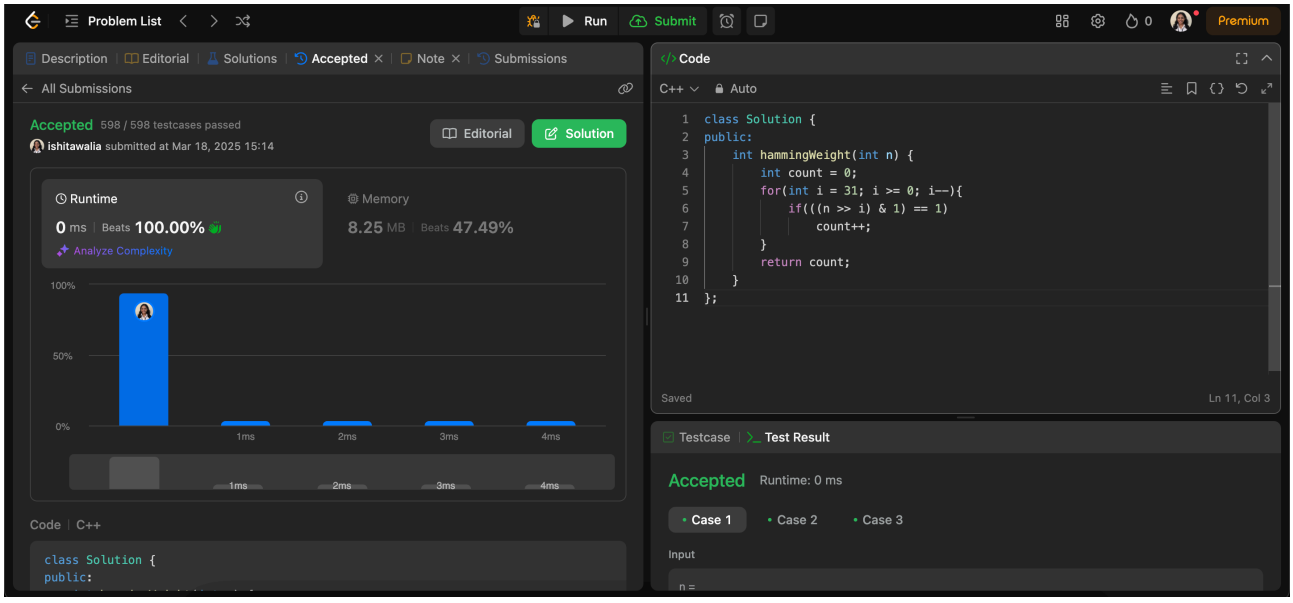
Question 2

```
class Solution {
public:
    uint32_t reverseBits(uint32_t n) {
        uint32_t ans = 0;
        for (int i = 0; i < 32; i++) {
            ans <<= 1;
            ans |= (n & 1);
            n >>= 1;
        }
        return ans;
    }
};
```



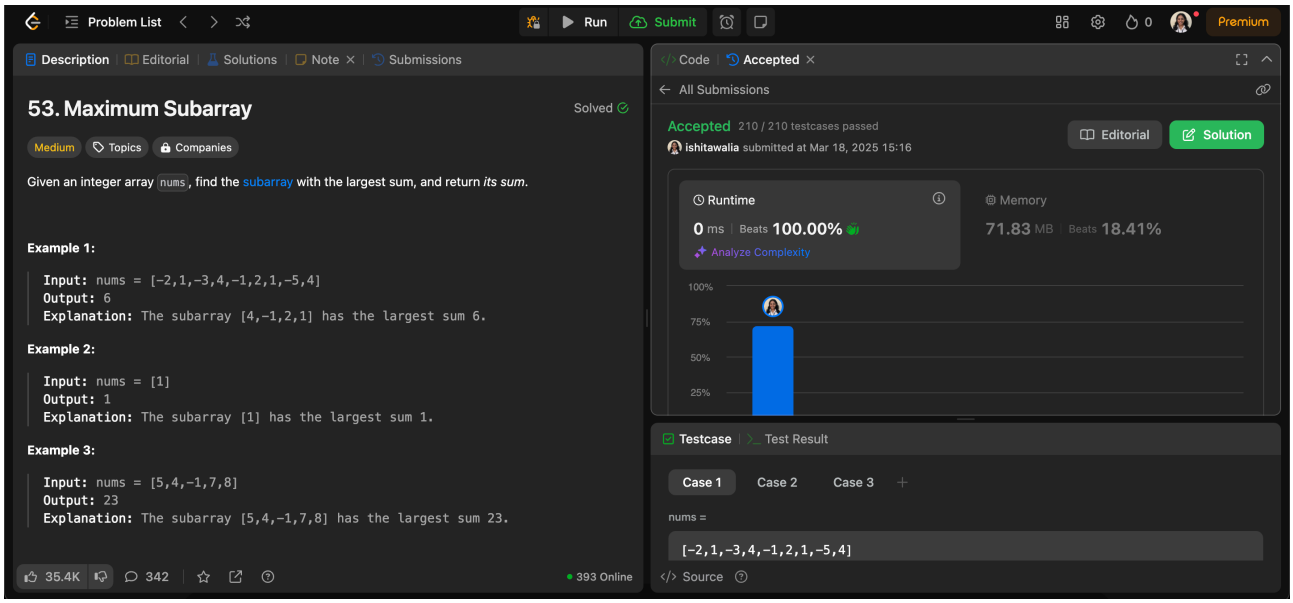
Question 3

```
class Solution {
public:
    int hammingWeight(int n) {
        int count = 0;
        for(int i = 31; i >= 0; i--){
            if(((n >> i) & 1) == 1)
                count++;
        }
        return count;
    }
};
```



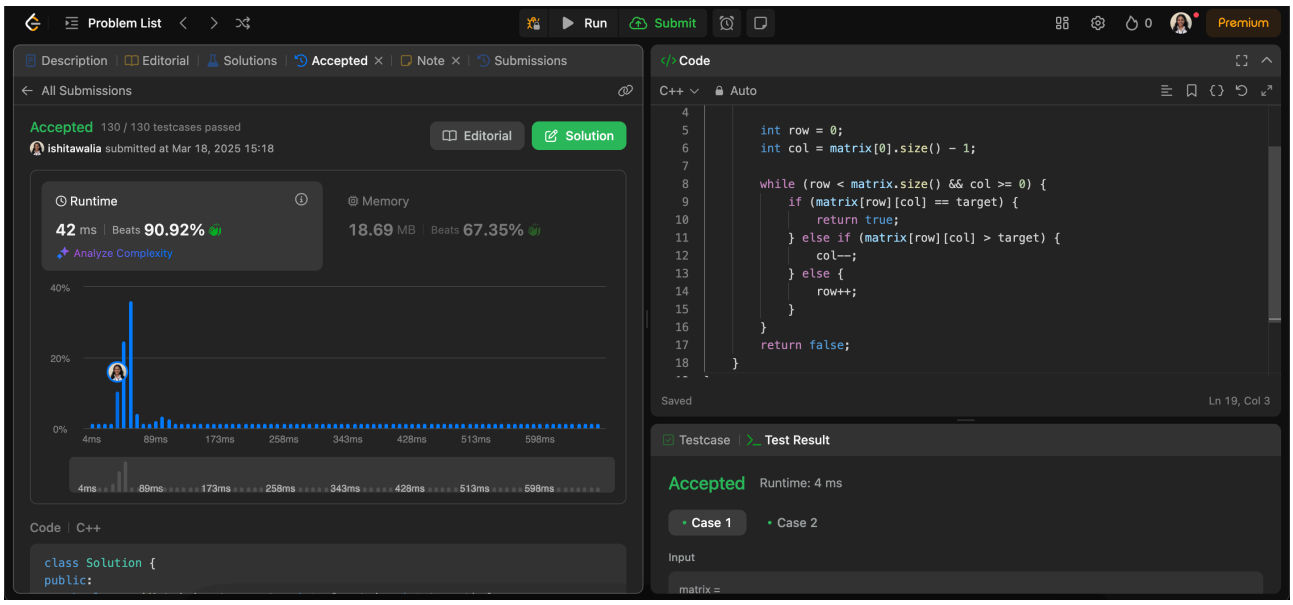
Question 4

```
class Solution {
public:
    int maxSubArray(vector<int>& nums) {
        int curMax = 0, maxTillNow = INT_MIN;
        for(auto c : nums)
            curMax = max(c, curMax + c),
            maxTillNow = max(maxTillNow, curMax);
        return maxTillNow;
    }
};
```



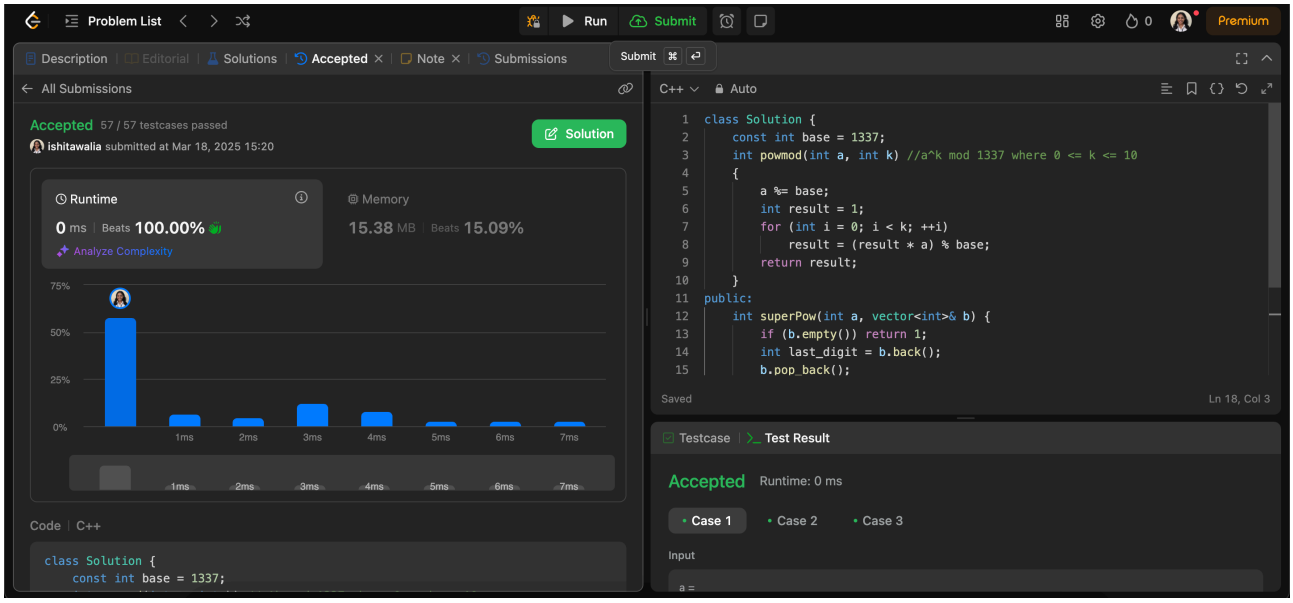
Question 5

```
class Solution {
public:
    bool searchMatrix(vector<vector<int>>& matrix, int target) {

        int row = 0;
        int col = matrix[0].size() - 1;

        while (row < matrix.size() && col >= 0) {
            if (matrix[row][col] == target) {
                return true;
            } else if (matrix[row][col] > target) {
                col--;
            } else {
                row++;
            }
        }
        return false;
    }
};
```



Question 6

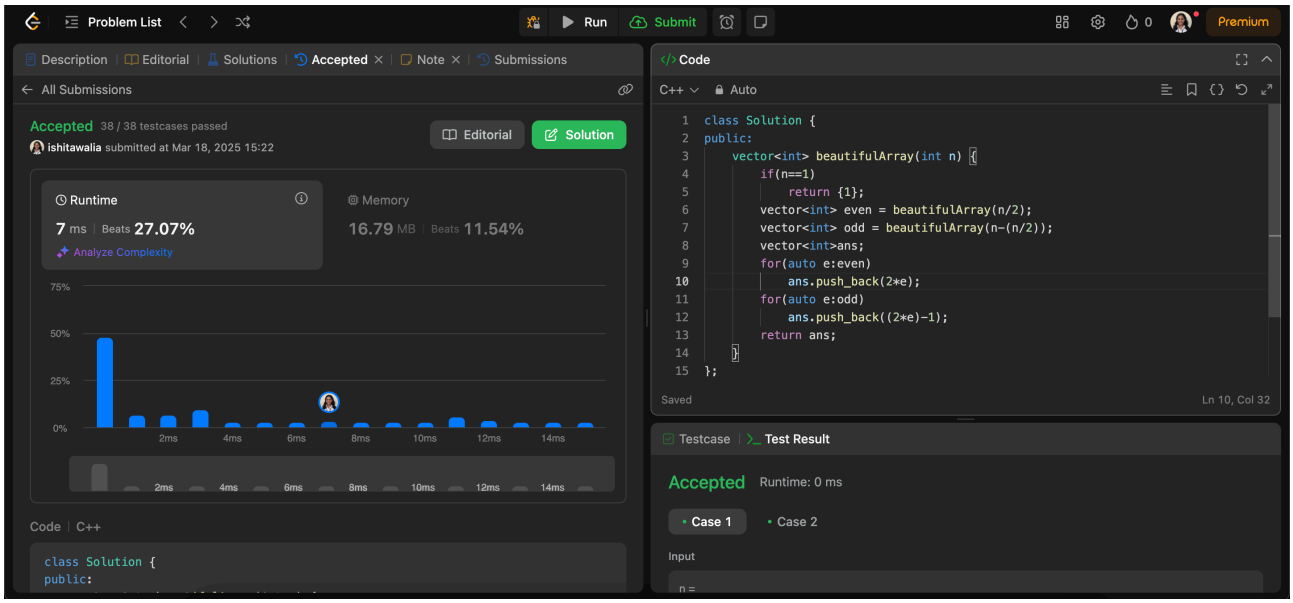
```
class Solution {
    const int base = 1337;
    int powmod(int a, int k) //a^k mod 1337 where 0 <= k <= 10
    {
        a %= base;
        int result = 1;
        for (int i = 0; i < k; ++i)
            result = (result * a) % base;
        return result;
    }
public:
    int superPow(int a, vector<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;
    }
```



};

Question 7

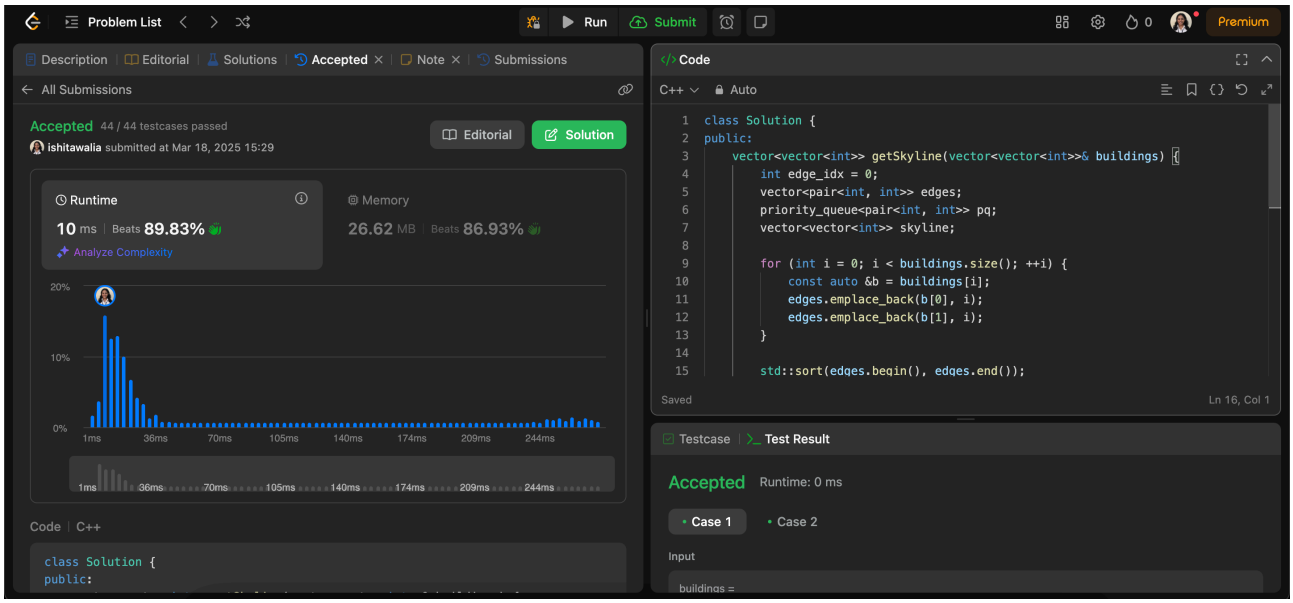
```
class Solution {
public:
    vector<int> beautifulArray(int n) {
        if(n==1)
            return {1};
        vector<int> even = beautifulArray(n/2);
        vector<int> odd = beautifulArray(n-(n/2));
        vector<int>ans;
        for(auto e:even)
            ans.push_back(2*e);
        for(auto e:odd)
            ans.push_back((2*e)-1);
        return ans;
    }
};
```



Question 8

```
class Solution {
public:
    vector<vector<int>> getSkyline(vector<vector<int>>& buildings) {
        int edge_idx = 0;
        vector<pair<int, int>> edges;
        priority_queue<pair<int, int>> pq;
        vector<vector<int>> skyline;

        for (int i = 0; i < buildings.size(); ++i) {
            const auto &b = buildings[i];
            edges.emplace_back(b[0], i);
            edges.emplace_back(b[1], i);
        }

        std::sort(edges.begin(), edges.end());

        while (edge_idx < edges.size()) {
            int curr_height;
            const auto &[curr_x, _] = edges[edge_idx];
```

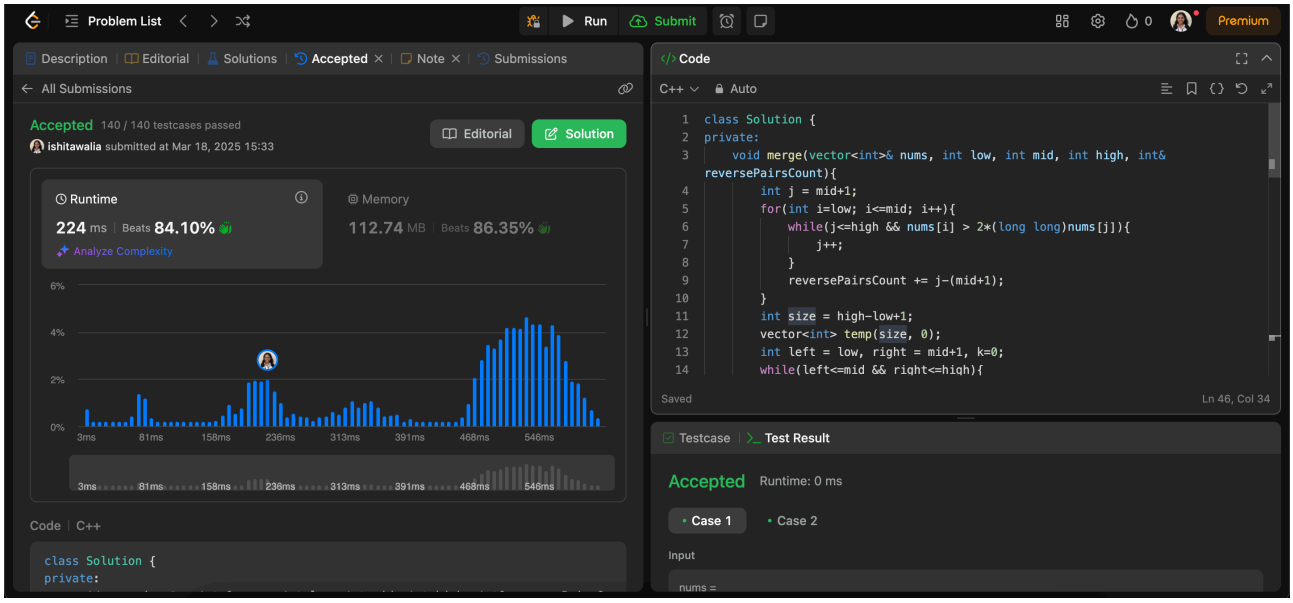
```
            while (edge_idx < edges.size() &&
                   curr_x == edges[edge_idx].first) {
                const auto &[_, building_idx] = edges[edge_idx];
                const auto &b = buildings[building_idx];
                if (b[0] == curr_x)
                    pq.emplace(b[2], b[1]);
                ++edge_idx;
            }
            while (!pq.empty() && pq.top().second <= curr_x)
                pq.pop();
            curr_height = pq.empty() ? 0 : pq.top().first;
            if (skyline.empty() || skyline.back()[1] != curr_height)
                skyline.push_back({curr_x, curr_height});
        }
        return skyline;
    }
};
```



Question 9

```
class Solution {
private:
    void merge(vector<int>& nums, int low, int mid, int high, int& reversePairsCount){
        int j = mid+1;
        for(int i=low; i<=mid; i++){
            while(j<=high && nums[i] > 2*(long long)nums[j]){
                j++;
            }
            reversePairsCount += j-(mid+1);
        }
        int size = high-low+1;
        vector<int> temp(size, 0);
        int left = low, right = mid+1, k=0;
        while(left<=mid && right<=high){
            if(nums[left] < nums[right]){
                temp[k++] = nums[left++];
            }
            else{
                temp[k++] = nums[right++];
            }
        }
```

```
        while(left<=mid){
            temp[k++] = nums[left++];
        }
        while(right<=high){
            temp[k++] = nums[right++];
        }
        int m=0;
        for(int i=low; i<=high; i++){
            nums[i] = temp[m++];
        }
    }

    void mergeSort(vector<int>& nums, int low, int high, int& reversePairsCount){
        if(low >= high){
            return;
        }
        int mid = (low + high) >> 1;
        mergeSort(nums, low, mid, reversePairsCount);
        mergeSort(nums, mid+1, high, reversePairsCount);
        merge(nums, low, mid, high, reversePairsCount);
    }
public:
    int reversePairs(vector<int>& nums) {
        int reversePairsCount = 0;
        mergeSort(nums, 0, nums.size()-1, reversePairsCount);
        return reversePairsCount;
    }
};
```



Question 10

```
class Solution {
public:
    vector<int> seg;
    //Segment tree to return maximum in a range
    void upd(int ind, int val, int x, int lx, int rx) {
        if(lx == rx) {
            seg[x] = val;
            return;
        }
```

```
        int mid = lx + (rx - lx) / 2;
        if(ind <= mid)
            upd(ind, val, 2 * x + 1, lx, mid);
        else
            upd(ind, val, 2 * x + 2, mid + 1, rx);
        seg[x] = max(seg[2 * x + 1], seg[2 * x + 2]);
    }
    int query(int l, int r, int x, int lx, int rx) {
        if(lx > r or rx < l) return 0;
        if(lx >= l and rx <= r) return seg[x];
        int mid = lx + (rx - lx) / 2;
        return max(query(l, r, 2 * x + 1, lx, mid), query(l, r, 2 * x + 2, mid + 1, rx));
    }

    int lengthOfLIS(vector<int>& nums, int k) {
        int x = 1;
        while(x <= 200000) x *= 2;
        seg.resize(2 * x, 0);

        int res = 1;
        for(int i = 0; i < nums.size(); ++i) {
            int left = max(1, nums[i] - k), right = nums[i] - 1;
            int q = query(left, right, 0, 0, x - 1); // check for the element in the range of [nums[i] - k,
nums[i] - 1] with the maximum value
            res = max(res, q + 1);
            upd(nums[i], q + 1, 0, 0, x - 1); //update current value
        }
        return res;
    }
};
```