



## ASSIGNMENT-4

**Student Name:** Deepa Kumari

**UID:** 22BCS10272

**Branch:** CSE

**Section/Group:** 22BCS\_IOT-609/B

**Semester:** 6<sup>th</sup>

**Subject Code:** 22CSP-351

**Subject Name:** Advanced Programming Lab-II

### 1. Problem Statement :

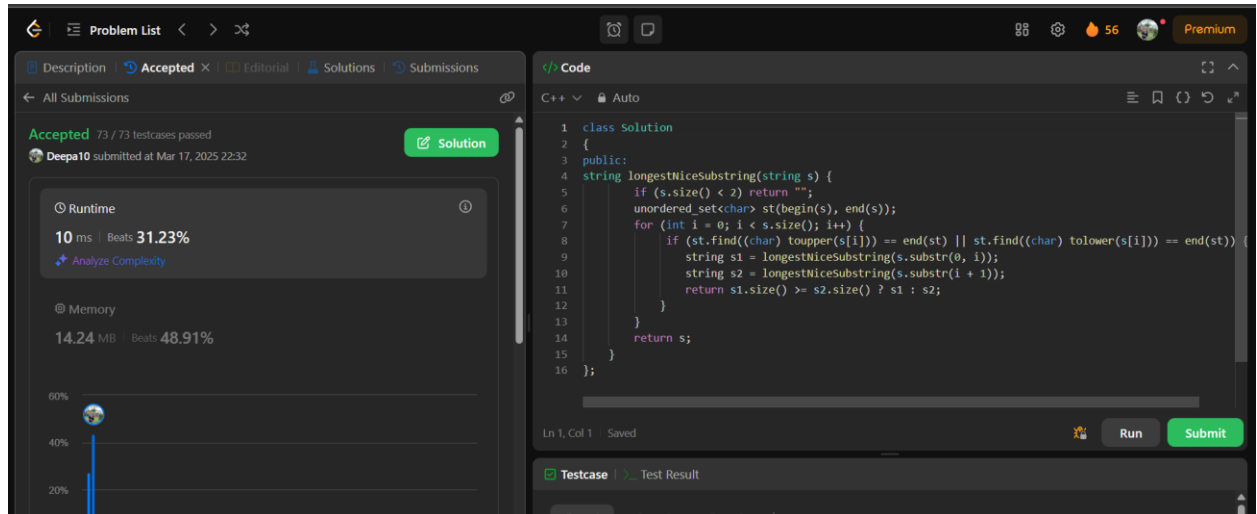
#### Longest Nice Substring

<https://leetcode.com/problems/longest-nice-substring/description/>

#### Code:

```
class Solution {
    public String longestNiceSubstring(String s) {
        if (s.length() < 2) return "";
        char[] arr = s.toCharArray();
        Set<Character> set = new HashSet<>();
        for (char c: arr) set.add(c);
        for (int i = 0; i < arr.length; i++) {
            char c = arr[i];
            if (set.contains(Character.toUpperCase(c)) &&
                set.contains(Character.toLowerCase(c))) continue;
            String sub1 = longestNiceSubstring(s.substring(0, i));
            String sub2 = longestNiceSubstring(s.substring(i+1));
            return sub1.length() >= sub2.length() ? sub1 : sub2;
        }
        return s;
    }
}
```

## OUTPUT:



```

1 class Solution
2 {
3 public:
4     string longestNiceSubstring(string s) {
5         if (s.size() < 2) return "";
6         unordered_set<char> st(begin(s), end(s));
7         for (int i = 0; i < s.size(); i++) {
8             if (st.find((char) toupper(s[i])) == end(st) || st.find((char) tolower(s[i])) == end(st)) {
9                 string s1 = longestNiceSubstring(s.substr(0, i));
10                string s2 = longestNiceSubstring(s.substr(i + 1));
11                return s1.size() >= s2.size() ? s1 : s2;
12            }
13        }
14        return s;
15    }
16 };

```

## 2. Problem Statement:

### Reverse Bits

<https://leetcode.com/problems/reverse-bits/description/>

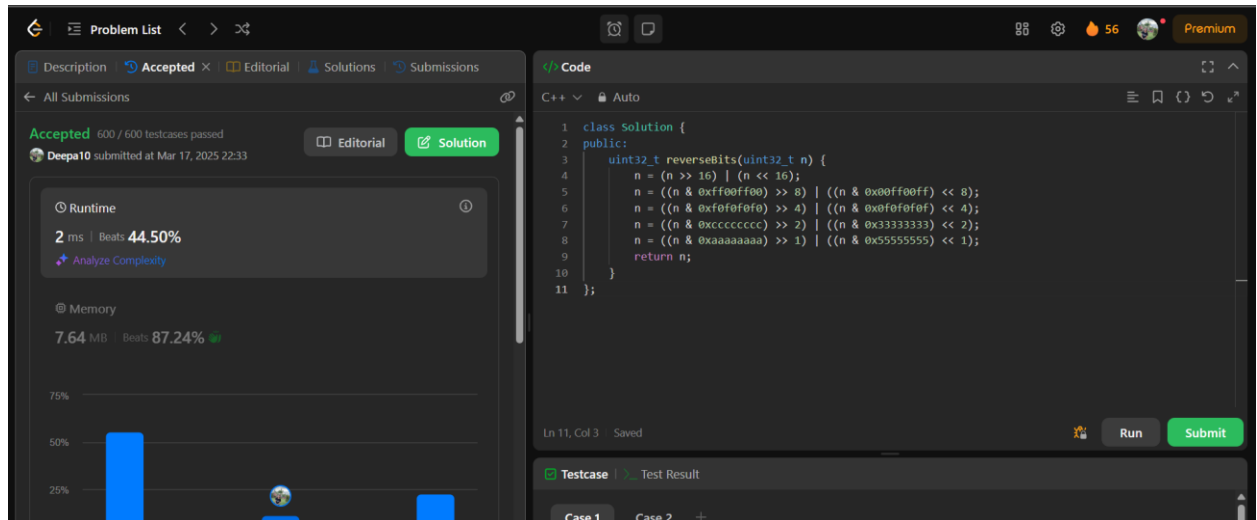
### Code:

```

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

```

## OUTPUT:



```
1 class Solution {
2 public:
3     uint32_t reverseBits(uint32_t n) {
4         n = (n >> 16) | (n << 16);
5         n = ((n & 0xffff00ff) >> 8) | ((n & 0x00ff00ff) << 8);
6         n = ((n & 0xf0f0f0f0) >> 4) | ((n & 0x0f0f0f0f) << 4);
7         n = ((n & 0xcccccccc) >> 2) | ((n & 0x33333333) << 2);
8         n = ((n & 0xaaaaaaaa) >> 1) | ((n & 0x55555555) << 1);
9         return n;
10    }
11};
```

### 3. Problem Statement:

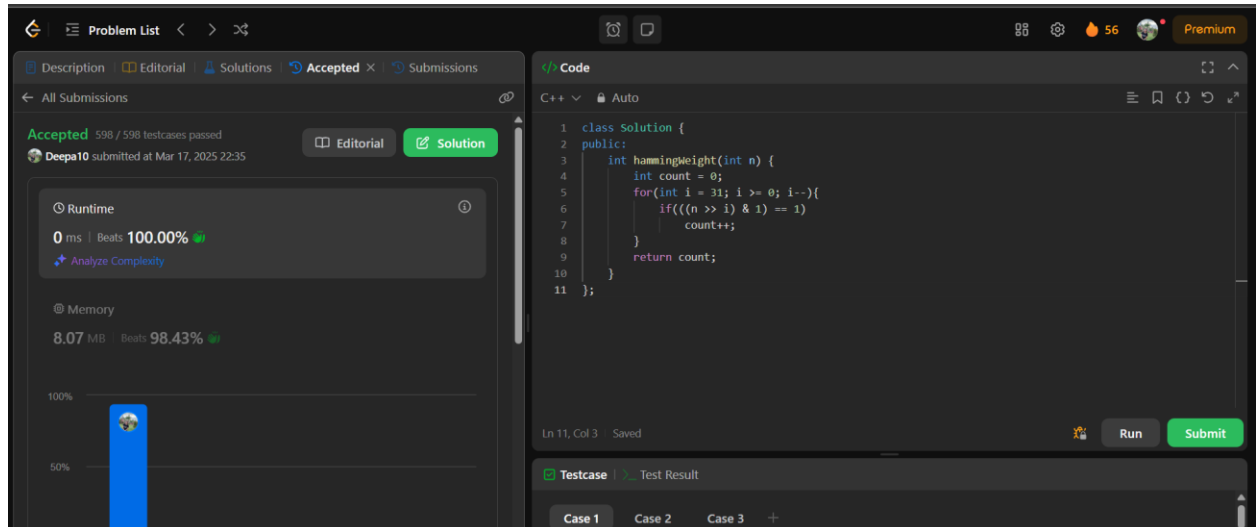
#### Number of 1 Bits

<https://leetcode.com/problems/number-of-1-bits/description/>

#### CODE:

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

## OUTPUT:



```
1 class Solution {
2 public:
3     int hammingweight(int n) {
4         int count = 0;
5         for(int i = 31; i >= 0; i--){
6             if(((n >> i) & 1) == 1)
7                 count++;
8         }
9         return count;
10    }
11};
```

## 4. Problem Statement:

### Maximum Subarray

<https://leetcode.com/problems/maximum-subarray/description/>

## CODE:

```
class Solution {
public:
    int maxSubArray(vector<int>& nums) {
        int sum=0;
        int maxi=nums[0];

        for (int i=0;i<nums.size();i++){
            sum=sum+nums[i];
            maxi=max(maxi,sum);

            if (sum<0)
            {
                sum=0;
            }
        }
    }
};
```

```

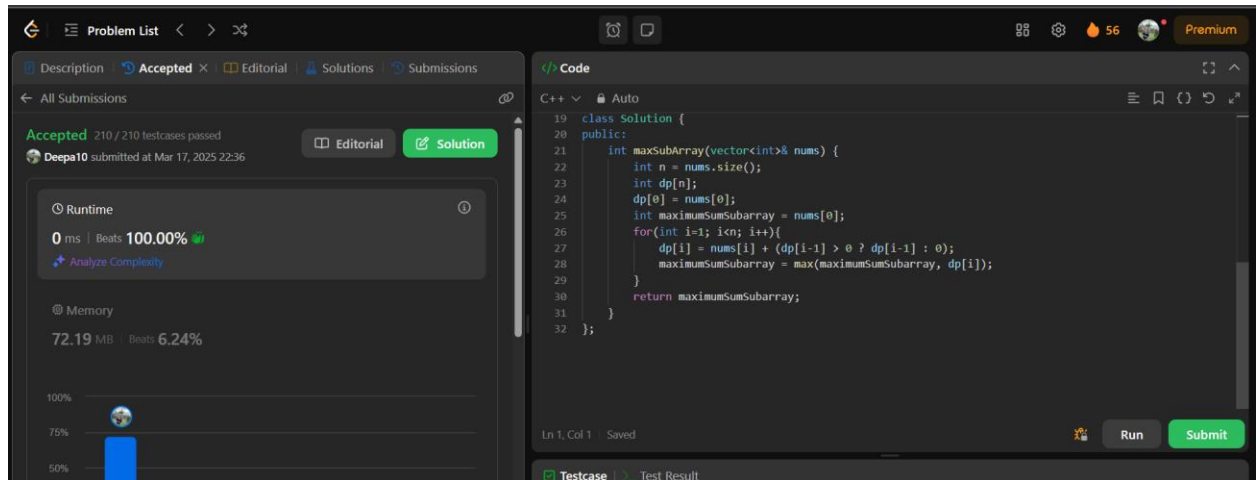
    }
    return maxi;

}

};

```

## OUTPUT:



```

19 class Solution {
20 public:
21     int maxSubArray(vector<int>& nums) {
22         int n = nums.size();
23         int dp[n];
24         dp[0] = nums[0];
25         int maximumSumSubarray = nums[0];
26         for(int i=1; i<n; i++){
27             dp[i] = nums[i] + (dp[i-1] > 0 ? dp[i-1] : 0);
28             maximumSumSubarray = max(maximumSumSubarray, dp[i]);
29         }
30         return maximumSumSubarray;
31     }
32 };

```

## 5. Problem Statement:

### Search a 2D Matrix II

<https://leetcode.com/problems/search-a-2d-matrix-ii/description/>

### CODE:

```

class Solution {
public:
    bool searchMatrix(vector<vector<int>>& matrix, int target) {
        if (matrix.empty() || matrix[0].empty()) return false;

        int rows = matrix.size();
        int cols = matrix[0].size();
        int row = 0, col = cols - 1;
        while (row < rows && col >= 0) {
            if (matrix[row][col] == target) return true;
            else if (matrix[row][col] > target) col--;
        }
    }
};

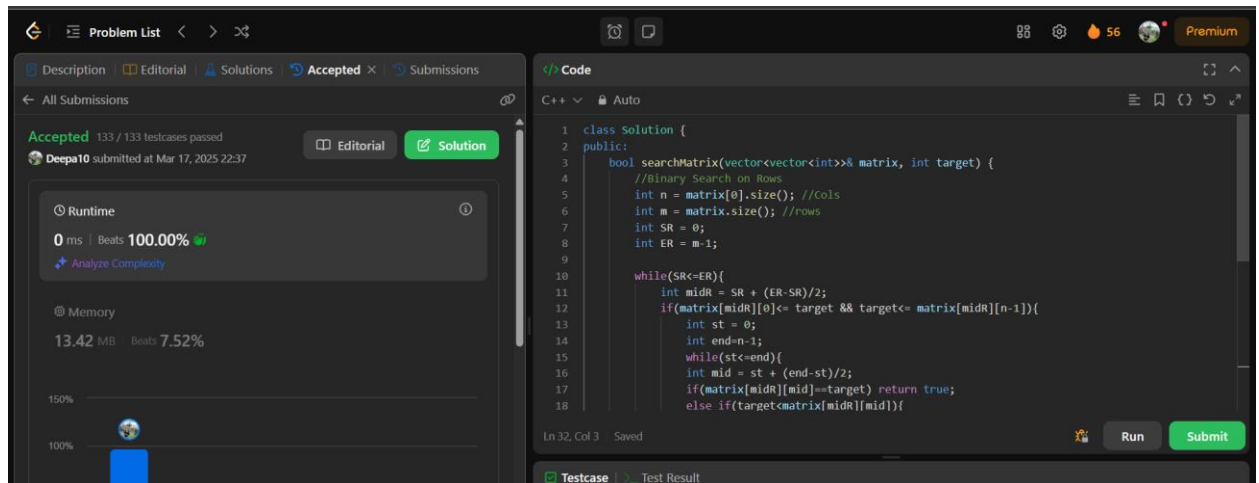
```

```

        else row++;
    }
    return false;
}
};

```

## OUTPUT:



## 6. Problem Statement:

### Super Pow

<https://leetcode.com/problems/super-pow/description/>

## CODE:

```

class Solution {
public:
    const int MOD = 1337;

    int powMod(int a, int b) {
        int result = 1;
        a %= MOD;
        for (int i = 0; i < b; i++) {
            result = (result * a) % MOD;

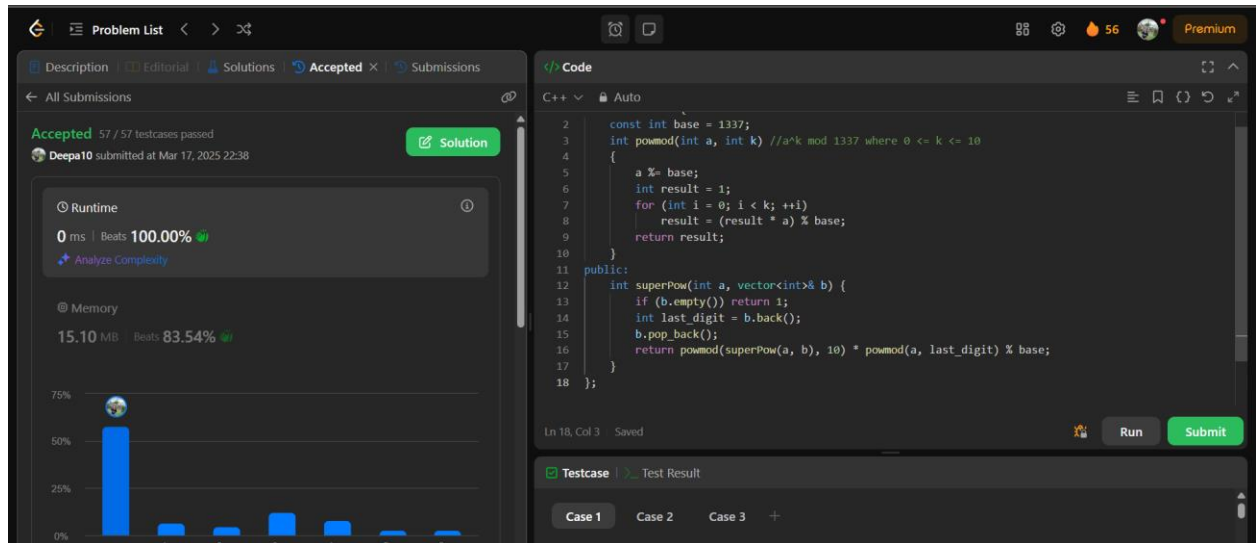
```

```

    }
    return result;
}
int superPow(int a, vector<int>& b) {
    int result = 1;
    for (int digit : b) {
        result = powMod(result, 10) * powMod(a, digit) % MOD;
    }
    return result;
}
};

```

## OUTPUT:



## 7. Problem Statement:

### Beautiful Array

<https://leetcode.com/problems/beautiful-array/description/>

### CODE:

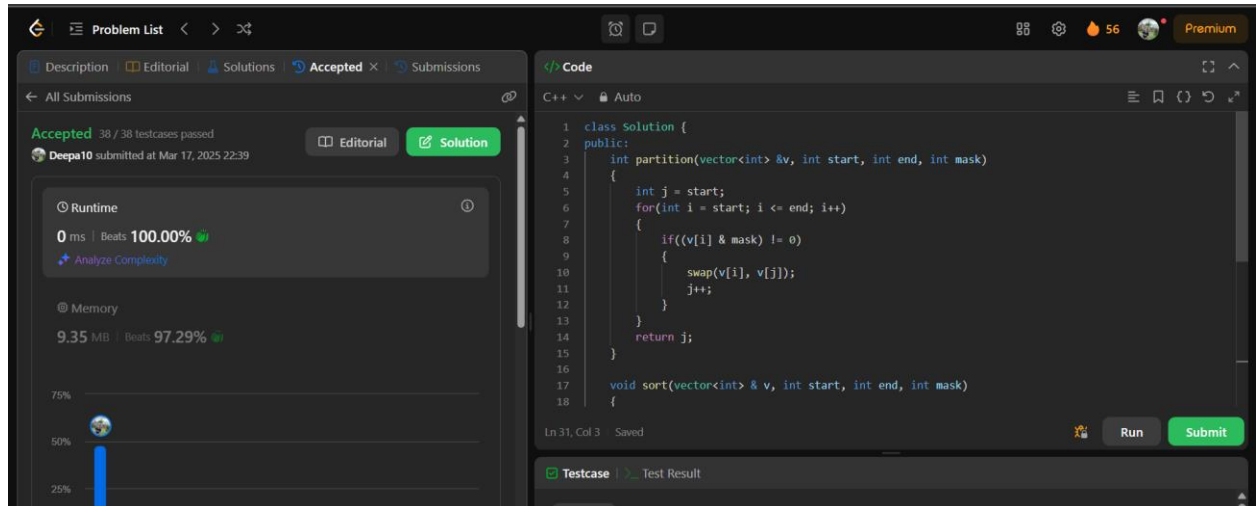
```

class Solution {
public:
    vector<int> beautifulArray(int n) {
        if (n == 1) return {1};
    }
};

```

```
vector<int> oddPart = beautifulArray((n + 1) / 2);
vector<int> evenPart = beautifulArray(n / 2);
vector<int> result;
for (int num : oddPart) result.push_back(2 * num - 1);
for (int num : evenPart) result.push_back(2 * num);
return result;
}
};
```

## OUTPUT:



## 8. Problem Statement:

### The Skyline Problem

<https://leetcode.com/problems/the-skyline-problem/description/>

## CODE:

```
class Solution {
public:
    vector<vector<int>> getSkyline(vector<vector<int>>& buildings) {
        vector<pair<int, int>> events;
        vector<vector<int>> result;
```



```
    for (auto& b : buildings) {
        events.emplace_back(b[0], -b[2]);
        events.emplace_back(b[1], b[2]);
    }

    sort(events.begin(), events.end(), [](pair<int, int>& a, pair<int, int>&
b) {
        if (a.first != b.first) return a.first < b.first;
        return a.second < b.second;
    });

    multiset<int> heights = {0};
    int prevHeight = 0;

    for (auto& e : events) {
        int x = e.first, h = e.second;

        if (h < 0) {
            heights.insert(-h);
        } else {
            heights.erase(heights.find(h));
        }

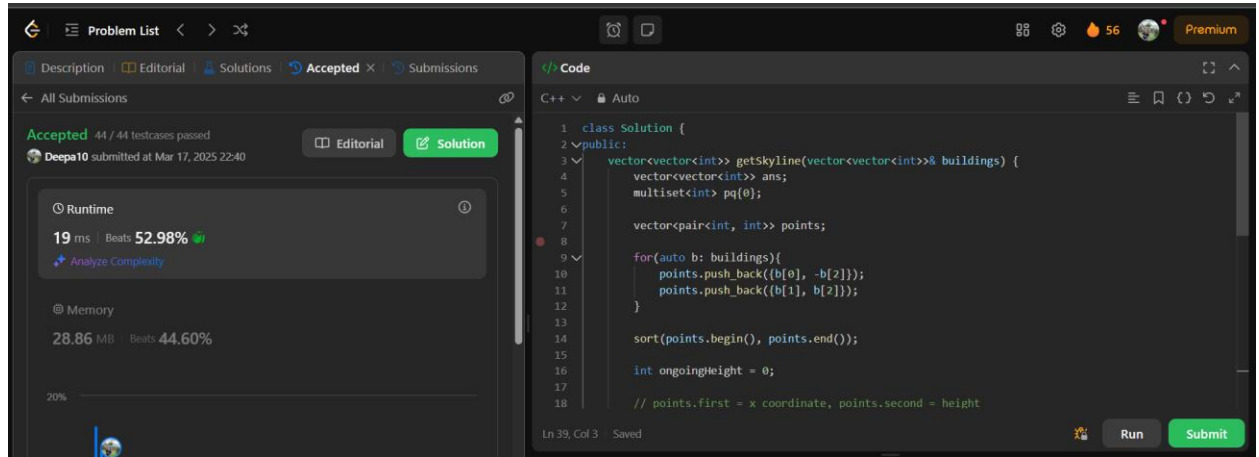
        int currHeight = *heights.rbegin();

        if (currHeight != prevHeight) {
            result.push_back({x, currHeight});
            prevHeight = currHeight;
        }
    }

    return result;
```

```
}  
};
```

## OUTPUT:



```
1 class Solution {  
2 public:  
3     vector<vector<int>> getSkyline(vector<vector<int>>& buildings) {  
4         vector<vector<int>> ans;  
5         multiset<int> pq(0);  
6  
7         vector<pair<int, int>> points;  
8  
9         for(auto b: buildings){  
10             points.push_back({b[0], -b[2]});  
11             points.push_back({b[1], b[2]});  
12         }  
13  
14         sort(points.begin(), points.end());  
15  
16         int ongoingHeight = 0;  
17  
18         // points.first = x coordinate, points.second = height
```

## 9. Problem Statement:

### Reverse Pairs

<https://leetcode.com/problems/reverse-pairs/description/>

### CODE:

```
class Solution {  
public:  
    int reversePairs(vector<int>& nums) {  
        return mergeSort(nums, 0, nums.size() - 1);  
    }  
  
private:  
    int mergeSort(vector<int>& nums, int left, int right) {  
        if (left >= right) return 0;  
  
        int mid = left + (right - left) / 2;  
        int count = mergeSort(nums, left, mid) + mergeSort(nums, mid + 1,  
right);
```

```
int j = mid + 1;
for (int i = left; i <= mid; i++) {
    while (j <= right && nums[i] > 2LL * nums[j]) j++;
    count += (j - (mid + 1));
}
```

```
merge(nums, left, mid, right);
return count;
}
```

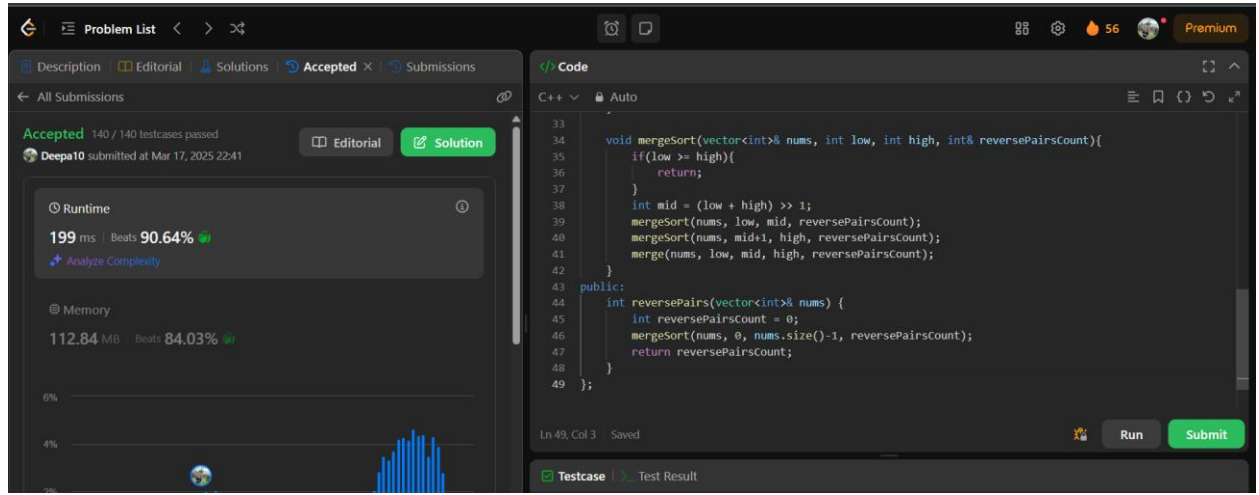
```
void merge(vector<int>& nums, int left, int mid, int right) {
    vector<int> temp;
    int i = left, j = mid + 1;

    while (i <= mid && j <= right) {
        if (nums[i] <= nums[j]) temp.push_back(nums[i++]);
        else temp.push_back(nums[j++]);
    }

    while (i <= mid) temp.push_back(nums[i++]);
    while (j <= right) temp.push_back(nums[j++]);

    for (int k = 0; k < temp.size(); k++) {
        nums[left + k] = temp[k];
    }
}
```

**OUTPUT:**



The screenshot shows a LeetCode submission for the problem 'Longest Increasing Subsequence II'. The solution is written in C++ and uses a merge sort algorithm to count the number of reverse pairs in an array. The submission is accepted, with a runtime of 199 ms (beating 90.64% of submissions) and a memory usage of 112.84 MB (beating 84.03% of submissions). The code is as follows:

```

33 void mergeSort(vector<int>& nums, int low, int high, int& reversePairsCount){
34     if(low >= high){
35         return;
36     }
37     int mid = (low + high) >> 1;
38     mergeSort(nums, low, mid, reversePairsCount);
39     mergeSort(nums, mid+1, high, reversePairsCount);
40     merge(nums, low, mid, high, reversePairsCount);
41 }
42
43 public:
44     int reversePairs(vector<int>& nums) {
45         int reversePairsCount = 0;
46         mergeSort(nums, 0, nums.size()-1, reversePairsCount);
47         return reversePairsCount;
48     }
49 };

```

## 10. Problem Statement

### Longest Increasing Subsequence II:

<https://leetcode.com/problems/longest-increasing-subsequence-ii/description/>

### CODE:

```

class SegmentTree {
    vector<int> tree;
    int size;

```

```

public:

```

```

    SegmentTree(int n) : size(n) {
        tree.resize(4 * n, 0);
    }

```

```

    void update(int index, int value, int node = 1, int start = 0, int end = -1)
    {
        if (end == -1) end = size - 1;
        if (start == end) {
            tree[node] = value;

```

```
        return;
    }
    int mid = (start + end) / 2;
    if (index <= mid) update(index, value, 2 * node, start, mid);
    else update(index, value, 2 * node + 1, mid + 1, end);
    tree[node] = max(tree[2 * node], tree[2 * node + 1]);
}

int query(int left, int right, int node = 1, int start = 0, int end = -1) {
    if (end == -1) end = size - 1;
    if (left > end || right < start) return 0;
    if (left <= start && end <= right) return tree[node];
    int mid = (start + end) / 2;
    return max(query(left, right, 2 * node, start, mid), query(left, right, 2
* node + 1, mid + 1, end));
}

};

class Solution {
public:
    int lengthOfLIS(vector<int>& nums, int k) {
        int maxVal = *max_element(nums.begin(), nums.end());
        SegmentTree segTree(maxVal + 1);
        int maxLength = 0;

        for (int num : nums) {
            int bestPrev = segTree.query(max(0, num - k), num - 1);
            int currLength = bestPrev + 1;
            segTree.update(num, currLength);
            maxLength = max(maxLength, currLength);
        }
    }
};
```



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
        return maxLength;  
    }  
};
```

## OUTPUT:

The screenshot displays a C++ code submission on a platform. The left sidebar shows the submission status: **Accepted** (84 / 84 testcases passed), submitted by **Deepa10** at Mar 17, 2025 22:43. The runtime is **110 ms** (Beats **43.69%**) and memory is **143.65 MB** (Beats **26.50%**). A bar chart at the bottom of the sidebar shows the distribution of runtime and memory usage. The main code editor shows the following C++ code:

```
1 class MaxSegmentTree {  
2 public:  
3     int n;  
4     vector<int> tree;  
5     MaxSegmentTree(int n) : n(n) {  
6         int size = (int)(ceil(log2(n)));  
7         size = (2 * pow(2, size)) - 1;  
8         tree = vector<int>(size);  
9     }  
10  
11     int max_value() { return tree[0]; }  
12  
13     int query(int l, int r) { return query_util(0, l, r, 0, n - 1); }  
14  
15     int query_util(int i, int ql, int qr, int l, int r) {  
16         if (l >= ql && r <= qr) return tree[i];  
17         if (l > qr || r < ql) return INT_MIN;  
18     }  
19 }  
20  
21 int main() {  
22     int n;  
23     cin >> n;  
24     MaxSegmentTree tree(n);  
25     for (int i = 0; i < n; i++) {  
26         int val;  
27         cin >> val;  
28         tree.update(i, val);  
29     }  
30     int q, r;  
31     while (q < r) {  
32         int l, r;  
33         cin >> l >> r;  
34         cout << tree.query(l, r) << " ";  
35         if (q % 10 == 9) cout << "\n";  
36         q++;  
37     }  
38     return 0;  
39 }
```