



## Assignment-4

**Name – Vedant Aggarwal**

**Semester - 6**

**Subject - Advanced Programming Lab-2**

**UID – 22BCS13945**

**Date - 17-03-2025**

**Subject Code - 22CSP-351**

- **Divide and Conquer:**

### 1. Longest Nice Substring-

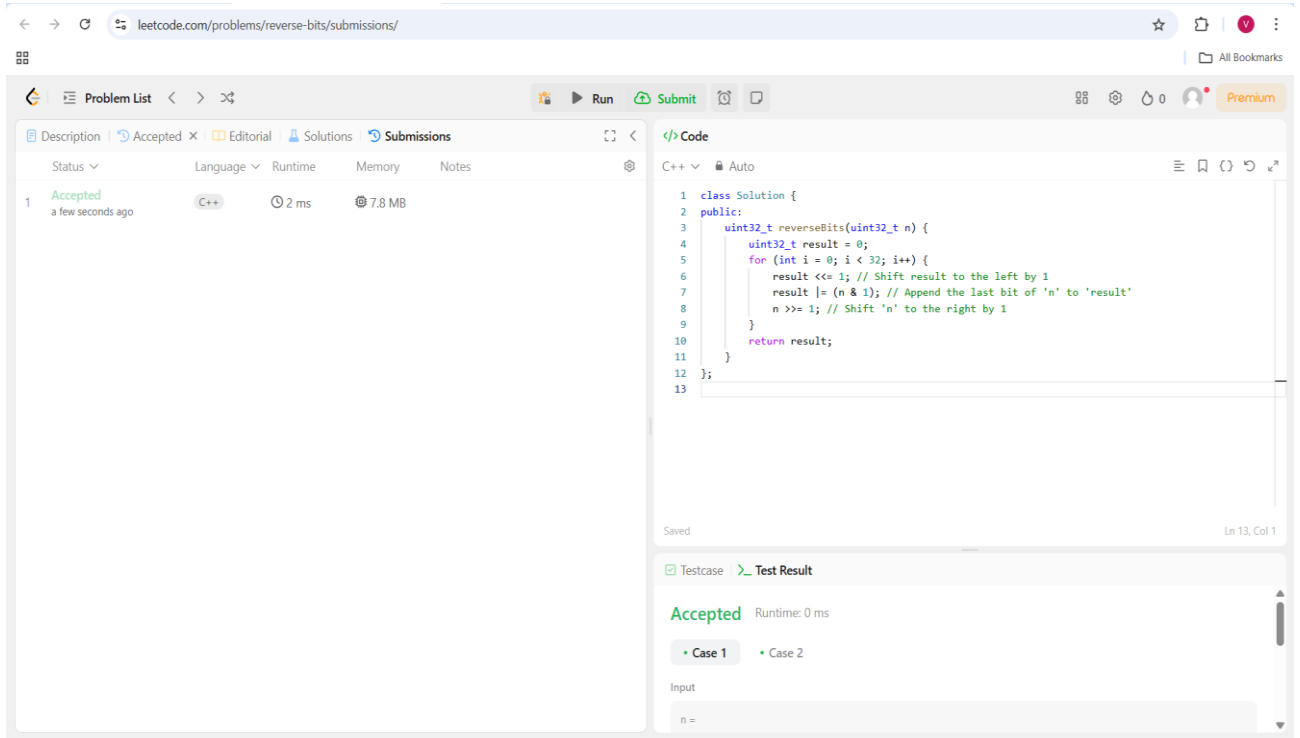
Screenshot of a LeetCode submission for the "Longest Nice Substring" problem. The submission is accepted. The code is in C++ and uses a recursive divide-and-conquer approach. The test case "YazaAay" is shown with the result "YazaAay".

```
1 class Solution {
2 public:
3     string longestNiceSubstring(string s) {
4         if (s.size() < 2) return "";
5
6         unordered_set<char> seen(s.begin(), s.end());
7
8         for (int i = 0; i < s.size(); ++i) {
9             if (seen.count(tolower(s[i])) && seen.count(toupper(s[i]))) continue;
10
11             string left = longestNiceSubstring(s.substr(0, i));
12             string right = longestNiceSubstring(s.substr(i + 1));
13
14             return left.size() >= right.size() ? left : right;
15         }
16
17         return s;
18     };
19 };
20
```

Testcase: Case 1 Case 2 Case 3 +

s = "YazaAay"

## 2. Reverse Bits-

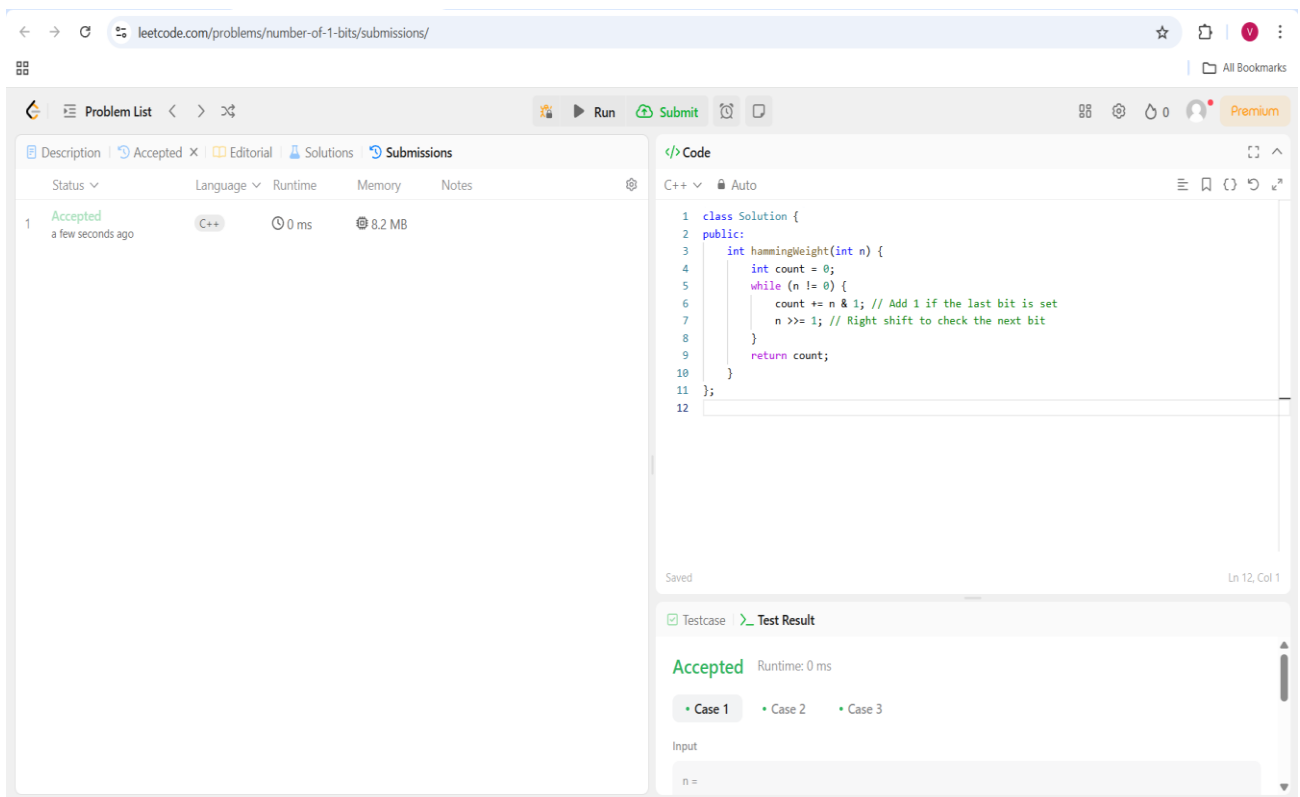


The screenshot shows the LeetCode interface for the 'Reverse Bits' problem. The submission is accepted in C++ with a runtime of 2 ms and memory of 7.8 MB. The code is as follows:

```
1 class Solution {
2 public:
3     uint32_t reverseBits(uint32_t n) {
4         uint32_t result = 0;
5         for (int i = 0; i < 32; i++) {
6             result <<= 1; // Shift result to the left by 1
7             result |= (n & 1); // Append the last bit of 'n' to 'result'
8             n >>= 1; // Shift 'n' to the right by 1
9         }
10        return result;
11    }
12 };
13
```

The test result shows 'Accepted' with a runtime of 0 ms. The input is 'n = '.

## 3. Number of 1 Bits-

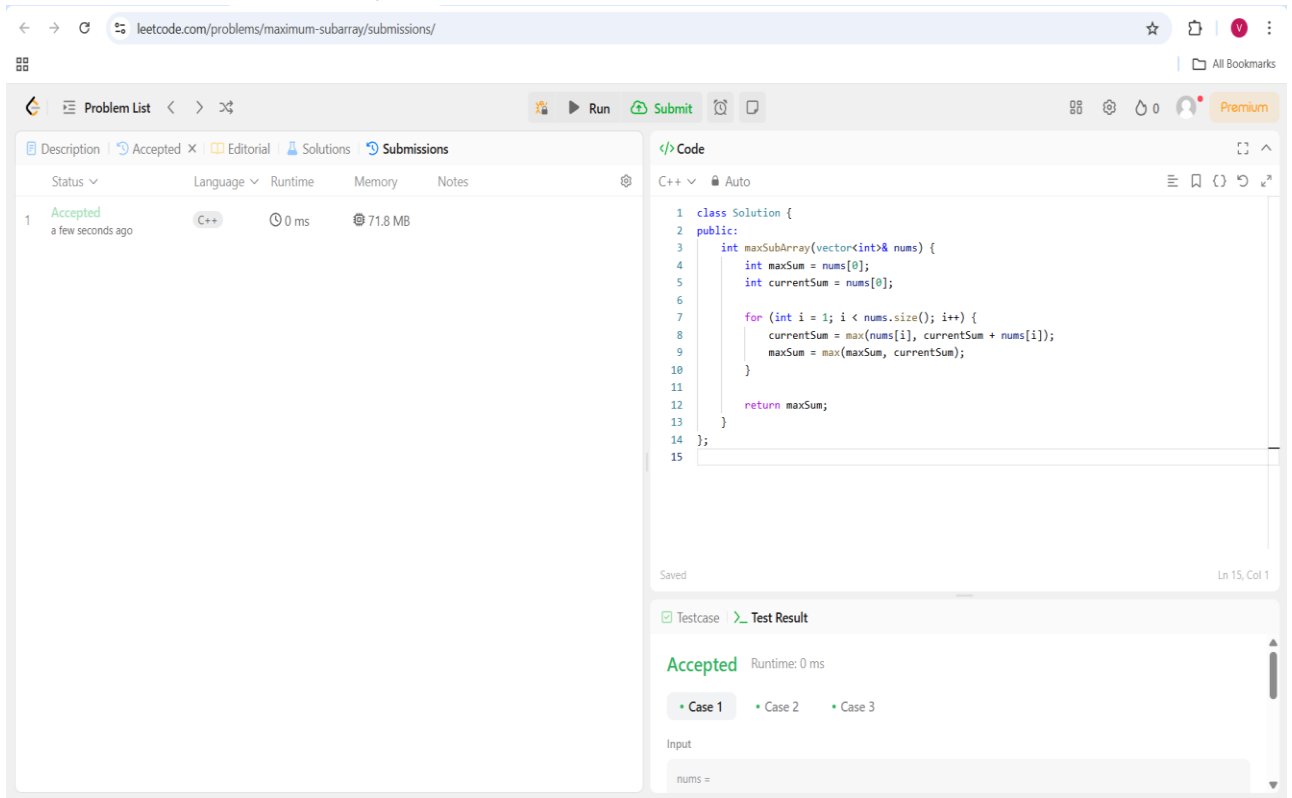


The screenshot shows the LeetCode interface for the 'Number of 1 Bits' problem. The submission is accepted in C++ with a runtime of 0 ms and memory of 8.2 MB. The code is as follows:

```
1 class Solution {
2 public:
3     int hammingWeight(int n) {
4         int count = 0;
5         while (n != 0) {
6             count += n & 1; // Add 1 if the last bit is set
7             n >>= 1; // Right shift to check the next bit
8         }
9         return count;
10    }
11 };
12
```

The test result shows 'Accepted' with a runtime of 0 ms. The input is 'n = '.

## 4. Maximum Subarray-



leetcode.com/problems/maximum-subarray/submissions/

Problem List < > <img alt="LeetCode logo" data-bbox="155 145 165 155"/> Run Submit <img alt="LeetCode logo" data-bbox="585 145 595 155"/> Premium

Description Accepted Editorial Solutions Submissions

Status Language Runtime Memory Notes

1 Accepted a few seconds ago C++ 0 ms 71.8 MB

```
1 class Solution {
2 public:
3     int maxSubArray(vector<int>& nums) {
4         int maxSum = nums[0];
5         int currentSum = nums[0];
6
7         for (int i = 1; i < nums.size(); i++) {
8             currentSum = max(nums[i], currentSum + nums[i]);
9             maxSum = max(maxSum, currentSum);
10        }
11
12        return maxSum;
13    }
14 };
15
```

Saved Ln 15, Col 1

Testcase Test Result

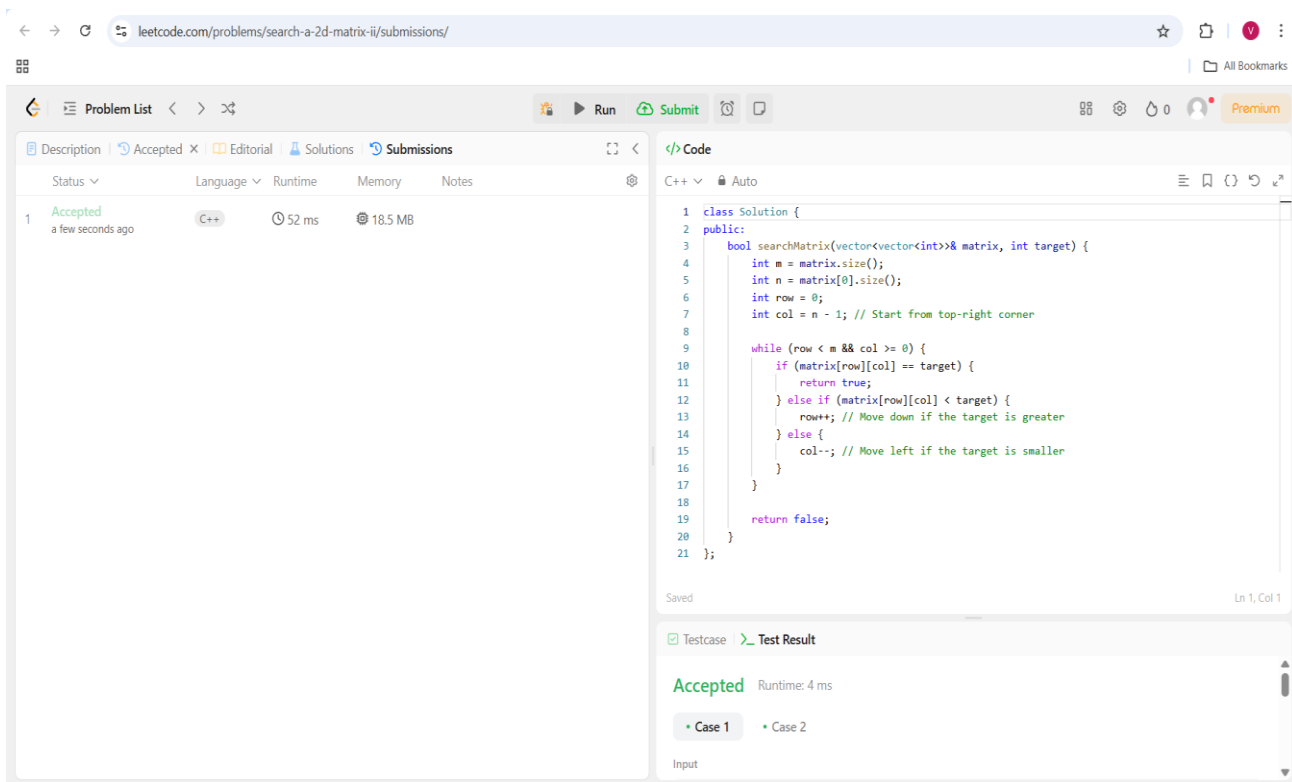
Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

nums =

## 5. Search a 2D Matrix II-



leetcode.com/problems/search-a-2d-matrix-ii/submissions/

Problem List < > <img alt="LeetCode logo" data-bbox="155 565 165 575"/> Run Submit <img alt="LeetCode logo" data-bbox="585 565 595 575"/> Premium

Description Accepted Editorial Solutions Submissions

Status Language Runtime Memory Notes

1 Accepted a few seconds ago C++ 52 ms 18.5 MB

```
1 class Solution {
2 public:
3     bool searchMatrix(vector<vector<int>>& matrix, int target) {
4         int m = matrix.size();
5         int n = matrix[0].size();
6         int row = 0;
7         int col = n - 1; // Start from top-right corner
8
9         while (row < m && col >= 0) {
10             if (matrix[row][col] == target) {
11                 return true;
12             } else if (matrix[row][col] < target) {
13                 row++; // Move down if the target is greater
14             } else {
15                 col--; // Move left if the target is smaller
16             }
17         }
18
19         return false;
20     }
21 };

```

Saved Ln 1, Col 1

Testcase Test Result

Accepted Runtime: 4 ms

Case 1 Case 2

Input

## 6. Super Pow-

← → ↺

leetcode.com/problems/super-pow/submissions/

☆

All Bookmarks

Problem List < > 🔍

Run Submit ⌂ 📄

Description Accepted Editorial Solutions Submissions

Status ▾ Language ▾ Runtime Memory Notes

1 Accepted  
a few seconds ago C++ 4 ms 15.3 MB

</> Code

C++ v Auto

```
1 class Solution {  
2 public:  
3     const int MOD = 1337;  
4  
5     int power(int a, int k) {  
6         a %= MOD;  
7         int result = 1;  
8         for (int i = 0; i < k; i++) {  
9             result = (result * a) % MOD;  
10        }  
11        return result;  
12    }  
13  
14    int superPow(int a, vector<int>& b) {  
15        if (b.empty()) return 1;  
16        int lastDigit = b.back();  
17        b.pop_back();  
18  
19        int part1 = power(a, lastDigit);  
20        int part2 = power(superPow(a, b), 10);  
21  
22        return (part1 * part2) % MOD;  
23    }  
24 };
```

Saved Ln 24, Col 3

Testcase Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

## 7. Beautiful Array-

Screenshot of the LeetCode submission interface for the "Beautiful Array" problem. The interface shows the problem description, a list of submissions, and the code editor.

**Submissions Table:**

Status	Language	Runtime	Memory	Notes
1 Accepted a few seconds ago	C++	1 ms	9.8 MB	

**Code Editor:**

```
1 class Solution {
2 public:
3     vector<int> beautifulArray(int n) {
4         vector<int> result = {1};
5         while (result.size() < n) {
6             vector<int> temp;
7             for (int num : result) if (num * 2 - 1 <= n) temp.push_back(num * 2 - 1);
8             for (int num : result) if (num * 2 <= n) temp.push_back(num * 2);
9             result = temp;
10        }
11        return result;
12    }
13 };
14
```

**Testcase:**

Accepted Runtime: 0 ms

• Case 1 • Case 2

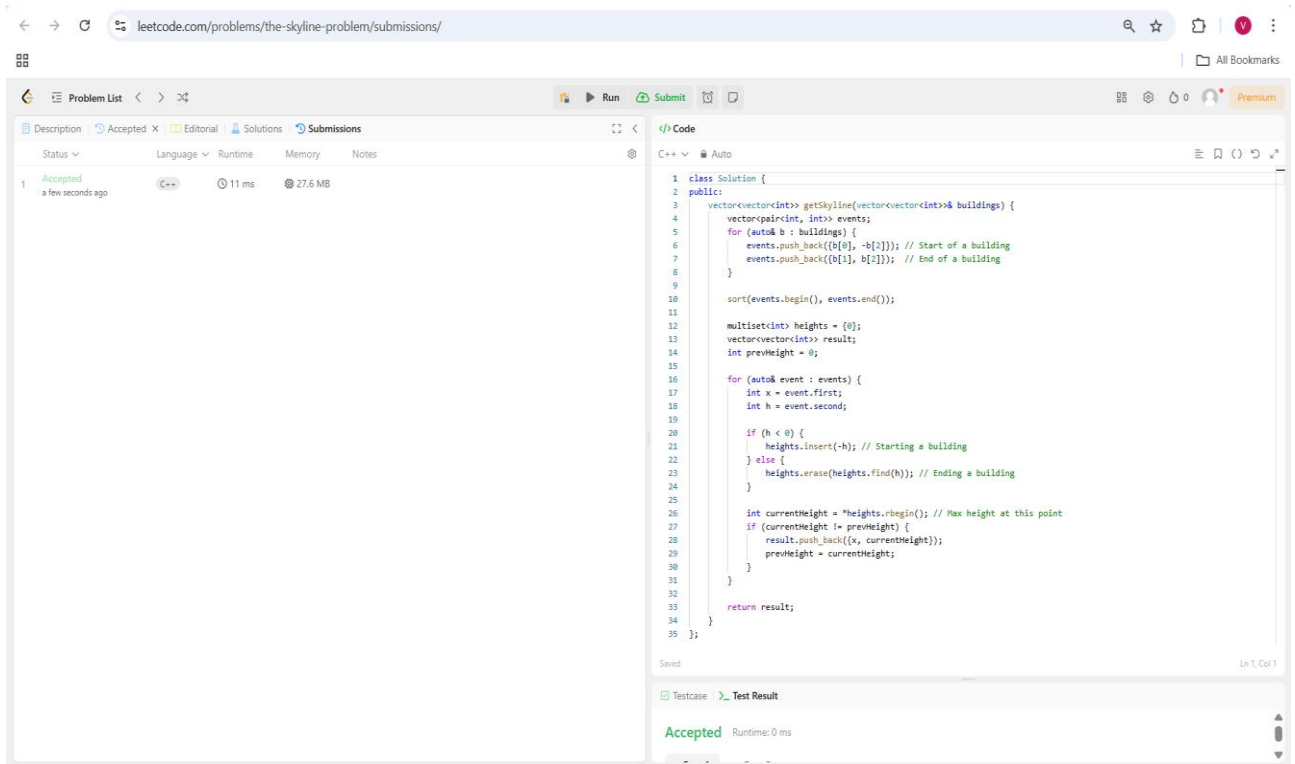
Input

n =

4

Output

## 8. The Skyline Problem-

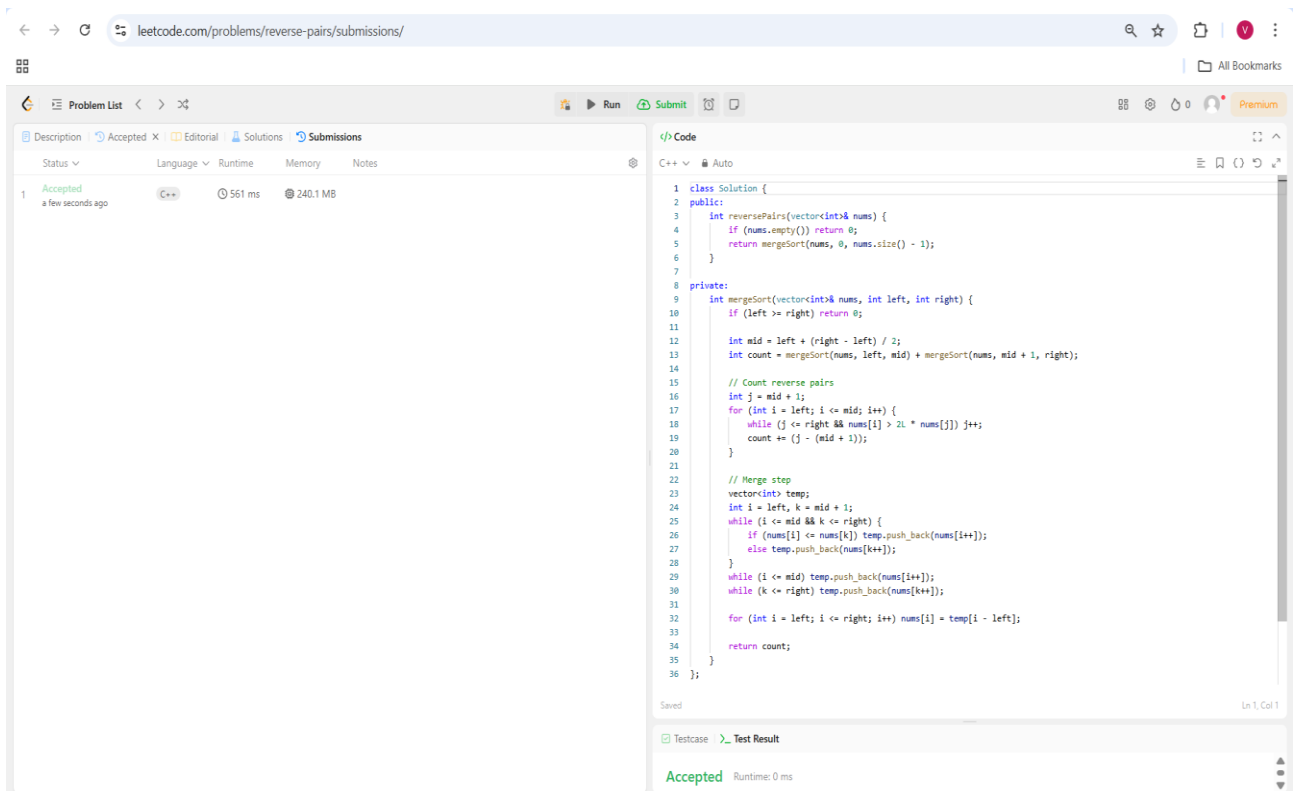


The screenshot shows a LeetCode submission for the problem "The Skyline Problem". The submission is in C++ and has a status of "Accepted". The code is as follows:

```
1 class Solution {
2 public:
3     vector<vector<int>> getSkyline(vector<vector<int>>& buildings) {
4         vector<pair<int, int>> events;
5         for (auto& b : buildings) {
6             events.push_back({b[0], -b[1]}); // Start of a building
7             events.push_back({b[1], b[1]}); // End of a building
8         }
9
10        sort(events.begin(), events.end());
11
12        multiset<int> heights = {0};
13        vector<vector<int>> result;
14        int prevHeight = 0;
15
16        for (auto& event : events) {
17            int x = event.first;
18            int h = event.second;
19
20            if (h < 0) {
21                heights.insert(-h); // Starting a building
22            } else {
23                heights.erase(heights.find(h)); // Ending a building
24            }
25
26            int currentHeight = *heights.rbegin(); // Max height at this point
27            if (currentHeight != prevHeight) {
28                result.push_back({x, currentHeight});
29                prevHeight = currentHeight;
30            }
31        }
32
33        return result;
34    };
35};
```

The submission is accepted with a runtime of 0 ms.

## 9. Reverse Pairs-



The screenshot shows a LeetCode submission for the problem "Reverse Pairs". The submission is in C++ and has a status of "Accepted". The code is as follows:

```
1 class Solution {
2 public:
3     int reversePairs(vector<int>& nums) {
4         if (nums.empty()) return 0;
5         return mergeSort(nums, 0, nums.size() - 1);
6     }
7
8 private:
9     int mergeSort(vector<int>& nums, int left, int right) {
10        if (left >= right) return 0;
11
12        int mid = left + (right - left) / 2;
13        int count = mergeSort(nums, left, mid) + mergeSort(nums, mid + 1, right);
14
15        // Count reverse pairs
16        int j = mid + 1;
17        for (int i = left; i <= mid; i++) {
18            while (j <= right && nums[i] > 2 * nums[j]) j++;
19            count += (j - (mid + 1));
20        }
21
22        // Merge step
23        vector<int> temp;
24        int i = left, k = mid + 1;
25        while (i <= mid && k <= right) {
26            if (nums[i] <= nums[k]) temp.push_back(nums[i++]);
27            else temp.push_back(nums[k++]);
28        }
29        while (i <= mid) temp.push_back(nums[i++]);
30        while (k <= right) temp.push_back(nums[k++]);
31        for (int i = left; i <= right; i++) nums[i] = temp[i - left];
32
33        return count;
34    }
35};
```

The submission is accepted with a runtime of 0 ms.

## 10. Longest Increasing Subsequence II-

Screenshot of the LeetCode interface showing the "Longest Increasing Subsequence II" problem. The interface includes a browser address bar, a navigation bar with "Problem List", "Run", "Submit", and "Premium" buttons, and a main content area with a "Submissions" tab and a "Code" editor.

The "Submissions" tab displays a table of submission results:

Status	Language	Runtime	Memory	Notes
Accepted a few seconds ago	C++	2928 ms	83.2 MB	
Time Limit Exceeded 6 minutes ago	C++	N/A	N/A	
Time Limit Exceeded 7 minutes ago	C++	N/A	N/A	
Time Limit Exceeded 7 minutes ago	C++	N/A	N/A	

The "Code" editor shows the following C++ code:

```
1 class Solution {
2 public:
3     int lengthOfLIS(vector<int>& nums, int k) {
4         vector<int> dp(100001, 0);
5         int longest = 0;
6
7         for (int num : nums) {
8             int maxlen = 0;
9             for (int i = max(0, num - k); i < num; ++i) {
10                 maxlen = max(maxlen, dp[i]);
11             }
12             dp[num] = maxlen + 1;
13             longest = max(longest, dp[num]);
14         }
15
16         return longest;
17     }
18 };
```

The "Test Result" section shows the submission was "Accepted" with a runtime of 0 ms. The input for Case 1 is:

```
nums =
[4, 2, 1, 4, 3, 4, 5, 8, 15]
```