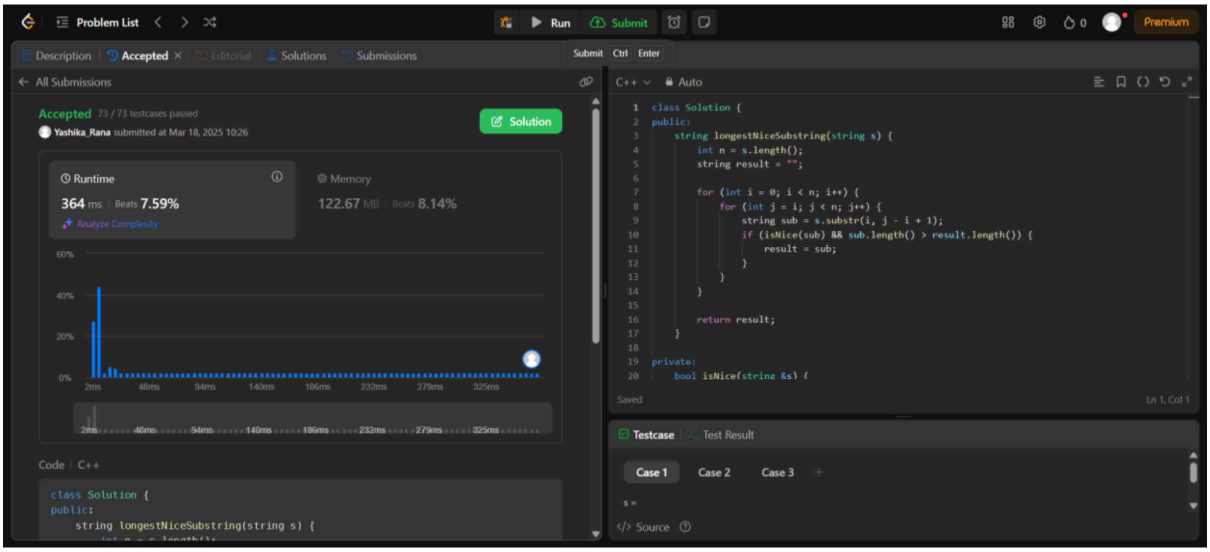
YASHIKA **22BCS10295**

ASSIGNMENT- 3

1. Longest Nice Substring:

```
class Solution {
public:
   string longestNiceSubstring(string s) {
      int n = s.length();
      string result = "";
      for (int i = 0; i < n; i++) {
         for (int j = i; j < n; j++) {
            string sub = s.substr(i, j - i + 1);
            if (isNice(sub) && sub.length() > result.length()) {
               result = sub;
            }
 }  }
      return result;
   }
private:
   bool isNice(string &s) {
      unordered_set<char> st(s.begin(), s.end());
      for (char c : s) {
         if (st.count(tolower(c)) == 0 || st.count(toupper(c)) == 0) {
            return false;
         }  }
      return true;
   }
};
```



Accepted 73 / 73 testcases passed

Yashika_Rana submitted at Mar 18, 2025 10:26

364 ms | Beats 7.59%

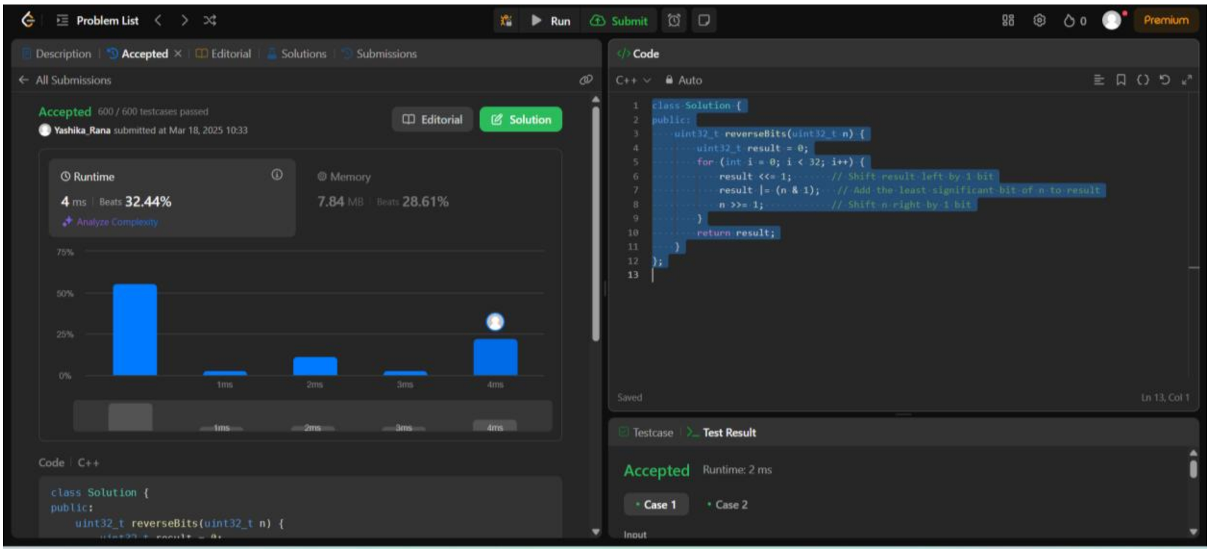
122.67 MB | Beats 8.14%

```
class Solution {
public:
    string longestNiceSubstring(string s) {
        int n = s.length();
        string result = "";

        for (int i = 0; i < n; i++) {
            for (int j = i; j < n; j++) {
                string sub = s.substr(i, j - i + 1);
                if (isNice(sub) && sub.length() > result.length()) {
                    result = sub;
                }
            }
        }

        return result;
    }

private:
    bool isNice(string &s) {
```

s =

2. Reverse Bits:

```
class Solution {
public:
    uint32_t reverseBits(uint32_t n) {
        uint32_t result = 0;
        for (int i = 0; i < 32; i++) {
            result <<= 1;       // Shift result left by 1 bit
            result |= (n & 1);   // Add the least significant bit of n to result
            n >>= 1;           // Shift n right by 1 bit
        }
        return result;
    }
};
```



Accepted 600 / 600 testcases passed

Yashika_Rana submitted at Mar 18, 2025 10:33

Runtime

4 ms | Beats 32.44%

Memory

7.84 MB | Beats 28.61%

Code | C++

```
class Solution {
public:
    uint32_t reverseBits(uint32_t n) {
```

Code

C++ Auto

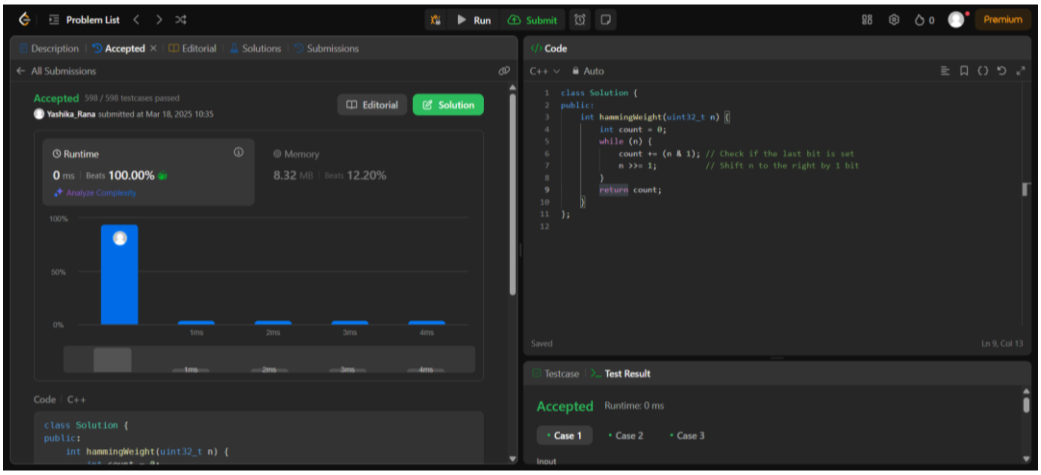
```
class Solution {
public:
    uint32_t reverseBits(uint32_t n) {
        uint32_t result = 0;
        for (int i = 0; i < 32; i++) {
            result <<= 1;          // Shift result left by 1 bit
            result |= (n & 1);    // Add the least significant bit of n to result
            n >>= 1;               // Shift n right by 1 bit
        }
        return result;
    }
};
```

Saved Ln 13, Col 1

Accepted Runtime: 2 ms

3. Number of 1 Bits:

```
class Solution {

public:

    int hammingWeight(uint32_t n) {

        int count = 0;

        while (n) {

            count += (n & 1); // Check if the last bit is set

            n >>= 1;          // Shift n to the right by 1 bit

        }

        return count;

    }

};
```



Accepted 598 / 598 testcases passed

Yashika_Rana submitted at Mar 18, 2025 10:35

Runtime

0 ms | Beats 100.00%

Memory

8.32 MB | Beats 12.20%

Code | C++

```
class Solution {
public:
    int hammingWeight(uint32_t n) {
```

Code

C++ Auto

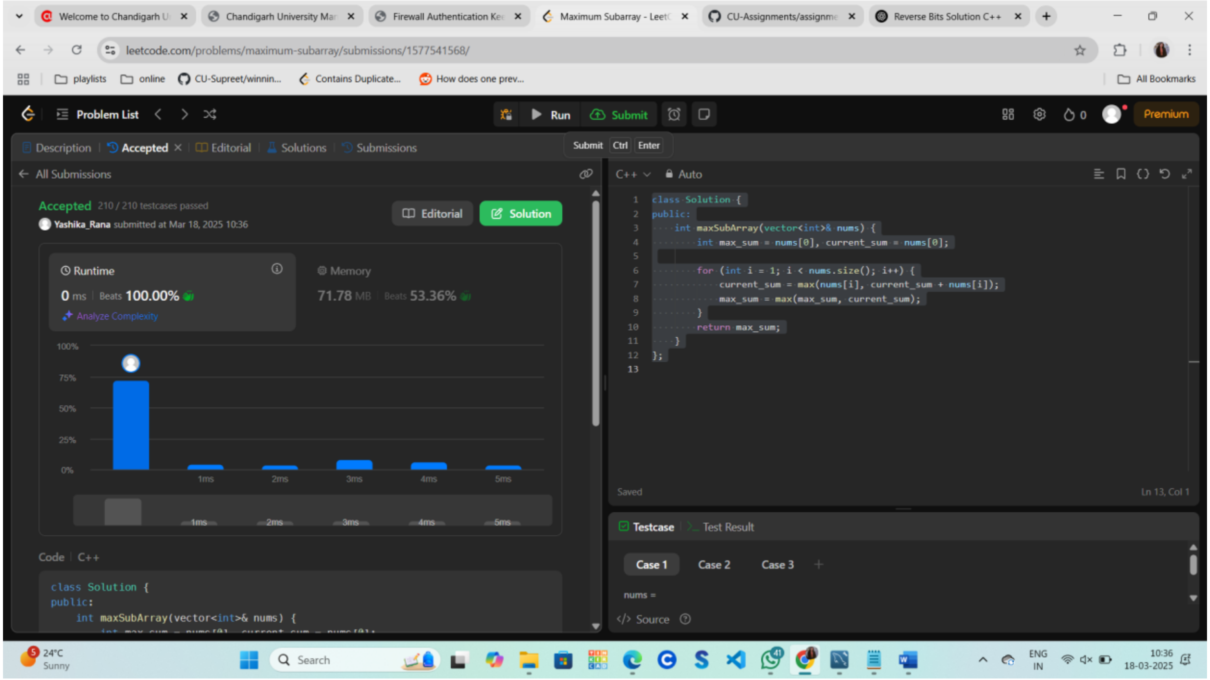
```
class Solution {
public:
    int hammingWeight(uint32_t n) {
        int count = 0;
        while (n) {
            count += (n & 1); // Check if the last bit is set
            n >>= 1;          // Shift n to the right by 1 bit
        }
        return count;
    }
};
```

Saved Ln 9, Col 13

Accepted Runtime: 0 ms

4. Maximum Subarray:

```
class Solution {

public:

    int maxSubArray(vector<int>& nums) {

        int max_sum = nums[0], current_sum = nums[0];



        for (int i = 1; i < nums.size(); i++) {

            current_sum = max(nums[i], current_sum + nums[i]);

            max_sum = max(max_sum, current_sum);

        }

        return max_sum;

    }

};
```



Accepted 210 / 210 testcases passed

Yashika_Rana submitted at Mar 18, 2025 10:36

Runtime

0 ms | Beats 100.00%

Memory

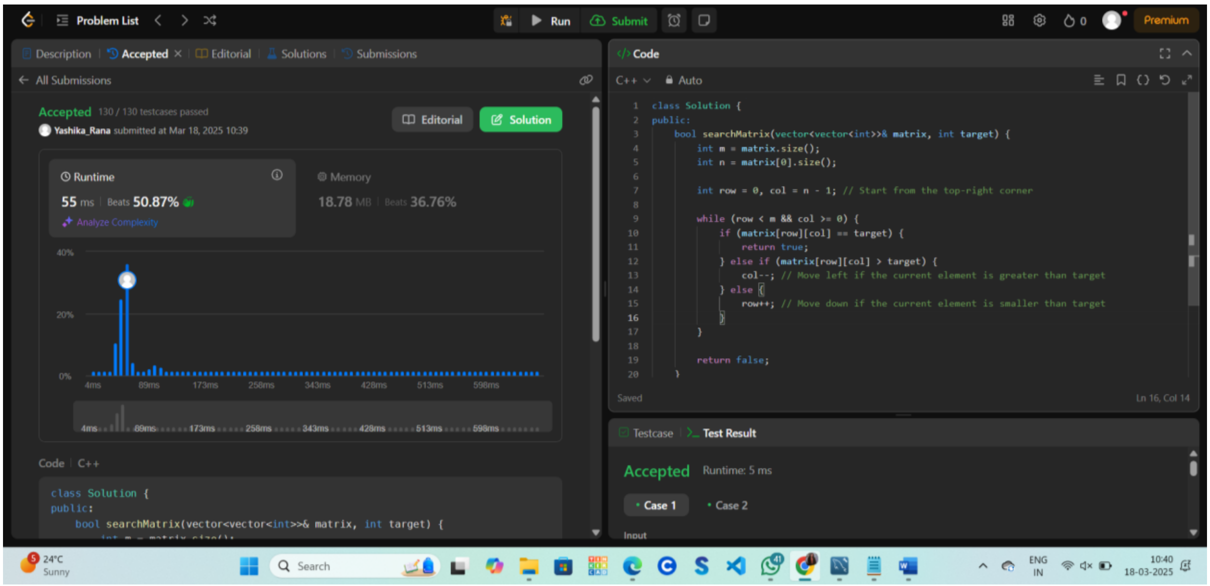
71.78 MB | Beats 53.36%

nums =

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

        for (int i = 1; i < nums.size(); i++) {
            current_sum = max(nums[i], current_sum + nums[i]);
            max_sum = max(max_sum, current_sum);
        }
        return max_sum;
    }
};
```

5. Search a 2D Matrix II:

```
class Solution {

public:

  bool searchMatrix(vector<vector<int>>& matrix, int target) {

    int m = matrix.size();

    int n = matrix[0].size();

    int row = 0, col = n - 1; // Start from the top-right corner

    while (row < m && col >= 0) {

      if (matrix[row][col] == target) {

        return true;

      } else if (matrix[row][col] > target) {

        col--; // Move left if the current element is greater than target

      } else {

        row++; // Move down if the current element is smaller than target

      }

    }


    return false;

  }

};
```



Accepted 130 / 130 testcases passed

Yashika_Rana submitted at Mar 18, 2025 10:39

Runtime

55 ms | Beats 50.87%

Memory

18.78 MB | Beats 36.76%

Code | C++

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

```
class Solution {
public:
    bool searchMatrix(vector<vector<int>>& matrix, int target) {
        int m = matrix.size();
        int n = matrix[0].size();

        int row = 0, col = n - 1; // Start from the top-right corner

        while (row < m && col >= 0) {
            if (matrix[row][col] == target) {
                return true;
            } else if (matrix[row][col] > target) {
                col--; // Move left if the current element is greater than target
            } else {
                row++; // Move down if the current element is smaller than target
            }
        }

        return false;
    }
```

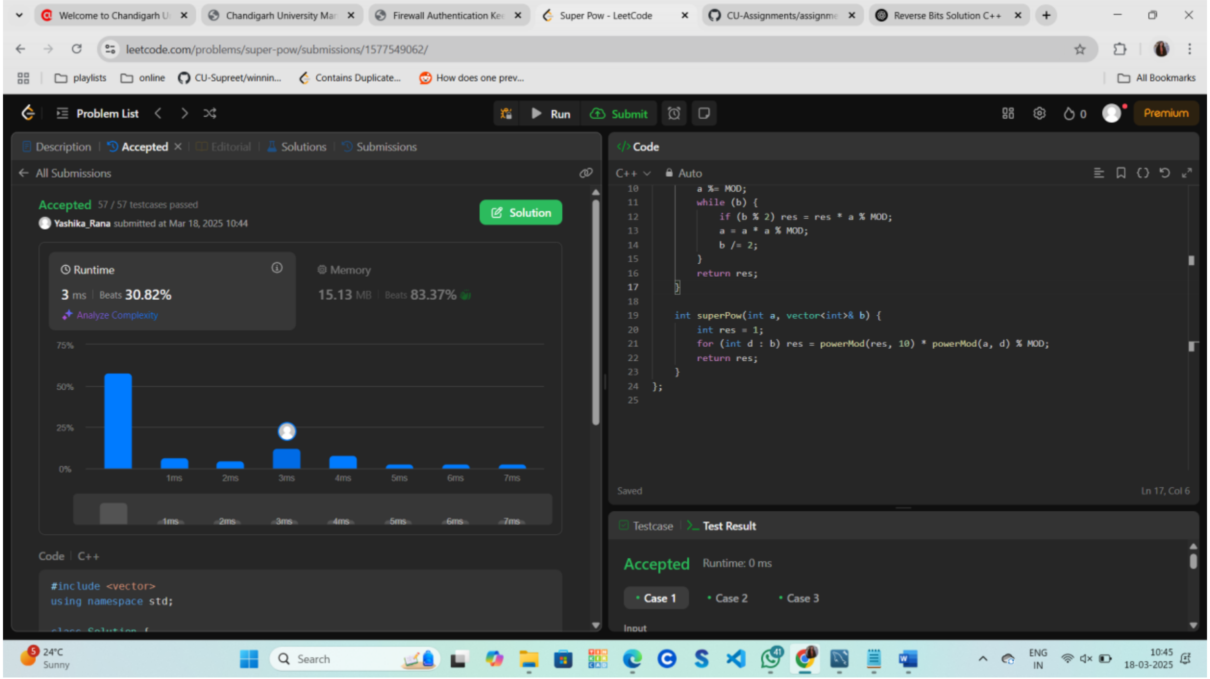
Saved

Ln 16, Col 14

Accepted Runtime: 5 ms

6. Super Pow:

```
class Solution {
public:
    const int MOD = 1337;
    int powerMod(int a, int b) {
        int res = 1;
        a %= MOD;
        while (b) {
            if (b % 2) res = res * a % MOD;
            a = a * a % MOD;
            b /= 2;
        }
        return res;
    }
    int superPow(int a, vector<int>& b) {
        int res = 1;
        for (int d : b) res = powerMod(res, 10) * powerMod(a, d) % MOD;
        return res;
    }
};
```



Accepted 57 / 57 testcases passed

Yashika_Rana submitted at Mar 18, 2025 10:44

Runtime

3 ms | Beats 30.82%

Memory

15.13 MB | Beats 83.37%

Code | C++

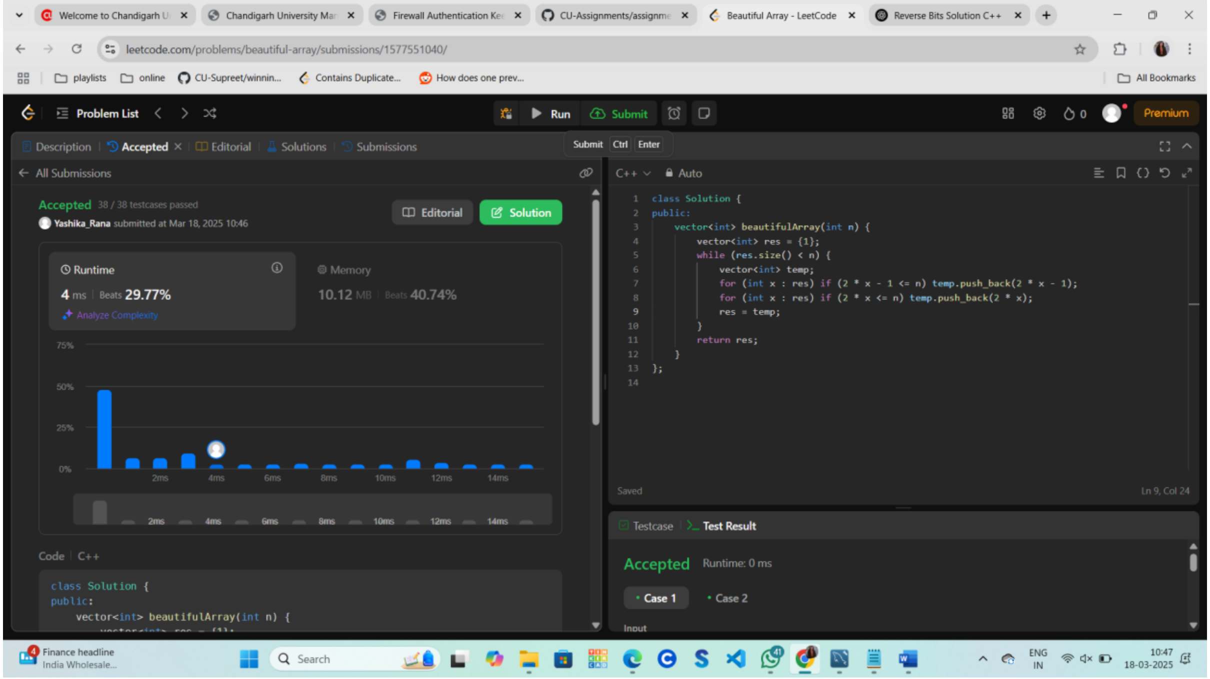
```
#include <vector>
using namespace std;

class Solution {
```

```
        a %= MOD;
        while (b) {
            if (b % 2) res = res * a % MOD;
            a = a * a % MOD;
            b /= 2;
        }
        return res;
    }

    int superPow(int a, vector<int>& b) {
        int res = 1;
        for (int d : b) res = powerMod(res, 10) * powerMod(a, d) % MOD;
        return res;
    }
};
```

Accepted Runtime: 0 ms

7. Beautiful Array:

```
class Solution {
public:
    vector<int> beautifulArray(int n) {
        vector<int> res = {1};
        while (res.size() < n) {
            vector<int> temp;
            for (int x : res) if (2 * x - 1 <= n) temp.push_back(2 * x - 1);
            for (int x : res) if (2 * x <= n) temp.push_back(2 * x);
            res = temp;
        }
        return res;
    }
};
```



Accepted 38 / 38 testcases passed

Yashika_Rana submitted at Mar 18, 2025 10:46

Runtime 4 ms | Beats 29.77%

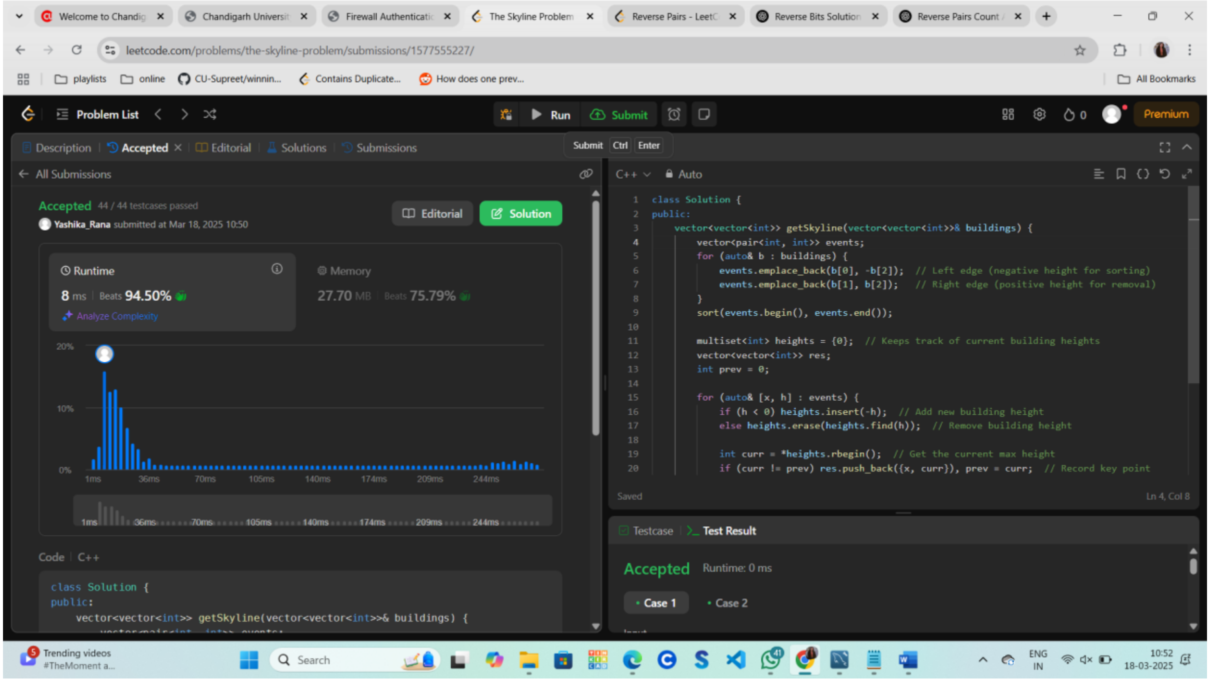
Memory 10.12 MB | Beats 40.74%

```
class Solution {
public:
    vector<int> beautifulArray(int n) {
        vector<int> res = {1};
        while (res.size() < n) {
            vector<int> temp;
            for (int x : res) if (2 * x - 1 <= n) temp.push_back(2 * x - 1);
            for (int x : res) if (2 * x <= n) temp.push_back(2 * x);
            res = temp;
        }
        return res;
    }
};
```

Accepted Runtime: 0 ms

8. The Skyline Problem:

```
class Solution {
public:
    vector<vector<int>> getSkyline(vector<vector<int>>& buildings) {
        vector<pair<int, int>> events;
        for (auto& b : buildings) {
            events.emplace_back(b[0], -b[2]);
            events.emplace_back(b[1], b[2]);
        }
        sort(events.begin(), events.end());
        multiset<int> heights = {0};  // Keeps track of current building heights
        vector<vector<int>> res;
        int prev = 0;
        for (auto& [x, h] : events) {
            if (h < 0) heights.insert(-h);  // Add new building height
            else heights.erase(heights.find(h));  // Remove building height
            int curr = *heights.rbegin();  // Get the current max height
            if (curr != prev) res.push_back({x, curr}), prev = curr;  // Record key point
        }
        return res;
    }
};
```



leetcode.com/problems/the-skyline-problem/submissions/1577555227/

Accepted 44 / 44 testcases passed

Yashika_Rana submitted at Mar 18, 2025 10:50

Runtime 8 ms | Beats 94.50%

Memory 27.70 MB | Beats 75.79%

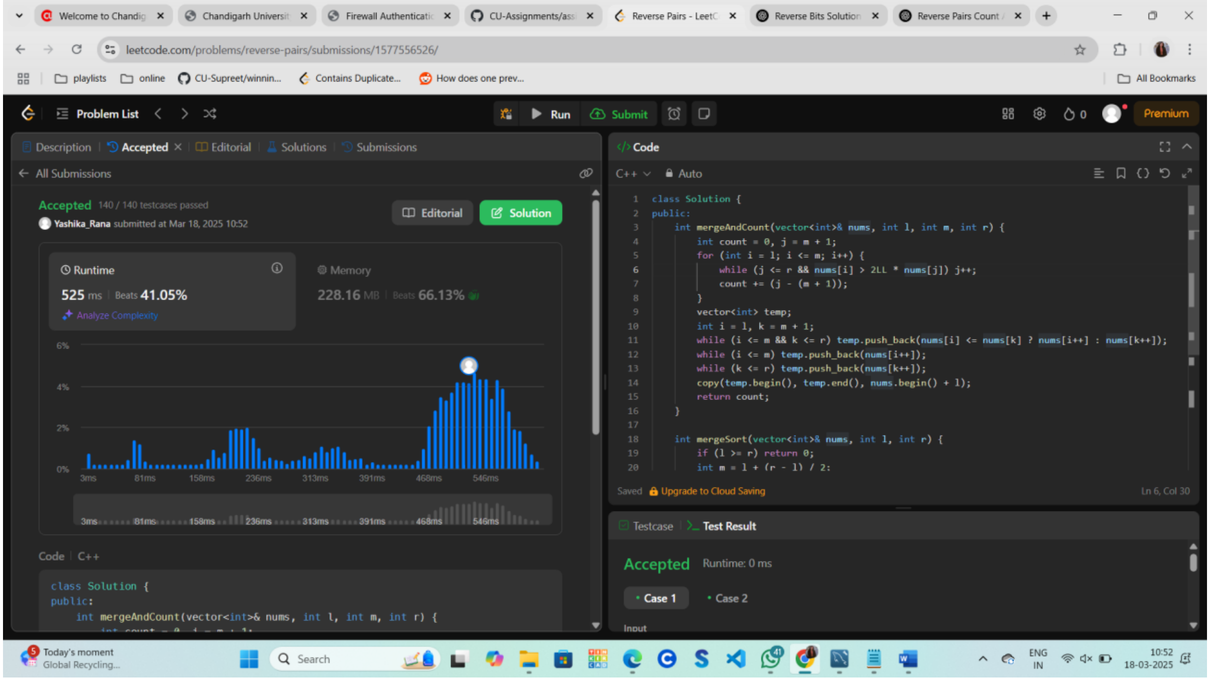
Code | C++

```
class Solution {
public:
    vector<vector<int>> getSkyline(vector<vector<int>>& buildings) {
        vector<pair<int, int>> events;
        for (auto& b : buildings) {
            events.emplace_back(b[0], -b[2]);  // Left edge (negative height for sorting)
            events.emplace_back(b[1], b[2]);   // Right edge (positive height for removal)
        }
        sort(events.begin(), events.end());

        multiset<int> heights = {0};  // Keeps track of current building heights
        vector<vector<int>> res;
        int prev = 0;

        for (auto& [x, h] : events) {
            if (h < 0) heights.insert(-h);  // Add new building height
            else heights.erase(heights.find(h));  // Remove building height

            int curr = *heights.rbegin();  // Get the current max height
            if (curr != prev) res.push_back({x, curr}), prev = curr;  // Record key point
```

Accepted Runtime: 0 ms

9. Reverse Pairs:

```
class Solution {
public:
    int mergeAndCount(vector<int>& nums, int l, int m, int r) {
        int count = 0, j = m + 1;
        for (int i = l; i <= m; i++) {
            while (j <= r && nums[i] > 2LL * nums[j]) j++;
            count += (j - (m + 1));
        }
        vector<int> temp;
        int i = l, k = m + 1;
        while (i <= m && k <= r) temp.push_back(nums[i] <= nums[k] ? nums[i++] : nums[k++]);
        while (i <= m) temp.push_back(nums[i++]);
        while (k <= r) temp.push_back(nums[k++]);
        copy(temp.begin(), temp.end(), nums.begin() + l);
        return count;
    }

    int mergeSort(vector<int>& nums, int l, int r) {
        if (l >= r) return 0;
        int m = l + (r - l) / 2;
        return mergeSort(nums, l, m) + mergeSort(nums, m + 1, r) + mergeAndCount(nums, l, m, r);
    }

    int reversePairs(vector<int>& nums) {
        return mergeSort(nums, 0, nums.size() - 1);
    }
};
```



leetcode.com/problems/reverse-pairs/submissions/1577556526/

Accepted 140 / 140 testcases passed

Yashika_Rana submitted at Mar 18, 2025 10:52

Runtime 525 ms | Beats 41.05%

Memory 228.16 MB | Beats 66.13%

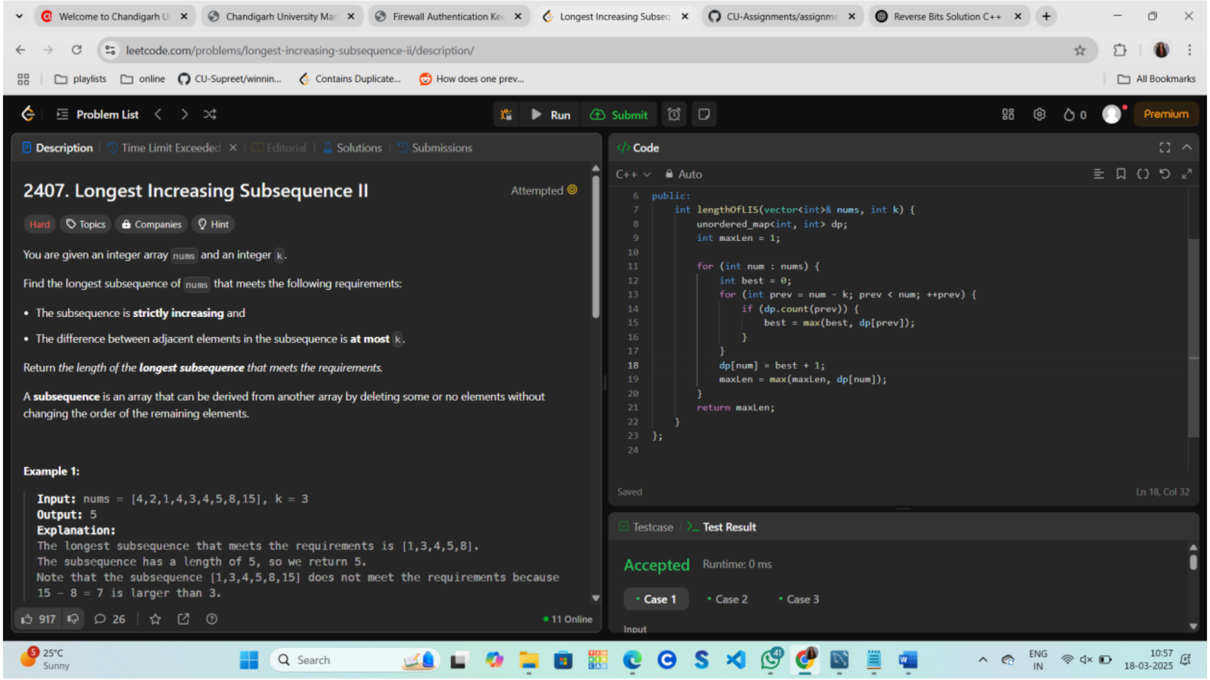
```
class Solution {
public:
    int mergeAndCount(vector<int>& nums, int l, int m, int r) {
        int count = 0, j = m + 1;
        for (int i = l; i <= m; i++) {
            while (j <= r && nums[i] > 2LL * nums[j]) j++;
            count += (j - (m + 1));
        }
        vector<int> temp;
        int i = l, k = m + 1;
        while (i <= m && k <= r) temp.push_back(nums[i] <= nums[k] ? nums[i++] : nums[k++]);
        while (i <= m) temp.push_back(nums[i++]);
        while (k <= r) temp.push_back(nums[k++]);
        copy(temp.begin(), temp.end(), nums.begin() + l);
        return count;
    }

    int mergeSort(vector<int>& nums, int l, int r) {
        if (l >= r) return 0;
        int m = l + (r - l) / 2;
```

Accepted Runtime: 0 ms

10. Longest Increasing Subsequence II:

```
#include <vector>
#include <unordered_map>
using namespace std;
```

```
class Solution {
public:
    int lengthOfLIS(vector<int>& nums, int k) {
        unordered_map<int, int> dp;
        int maxLen = 1;

        for (int num : nums) {
            int best = 0;
            for (int prev = num - k; prev < num; ++prev) {
                if (dp.count(prev)) {
                    best = max(best, dp[prev]);
                }
            }
            dp[num] = best + 1;
            maxLen = max(maxLen, dp[num]);
        }
        return maxLen;
    }
};
```



2407. Longest Increasing Subsequence II

You are given an integer array `nums` and an integer `k`.

Find the longest subsequence of `nums` that meets the following requirements:

- The subsequence is **strictly increasing** and
- The difference between adjacent elements in the subsequence is **at most** `k`.

Return *the length of the **longest subsequence** that meets the requirements.*

A **subsequence** is an array that can be derived from another array by deleting some or no elements without changing the order of the remaining elements.

Example 1:

```
Input: nums = [4,2,1,4,3,4,5,8,15], k = 3
Output: 5
Explanation:
The longest subsequence that meets the requirements is [1,3,4,5,8].
The subsequence has a length of 5, so we return 5.
Note that the subsequence [1,3,4,5,8,15] does not meet the requirements because
15 - 8 = 7 is larger than 3.
```

```cpp
public:
    int lengthOfLIS(vector<int>& nums, int k) {
        unordered_map<int, int> dp;
        int maxLen = 1;

        for (int num : nums) {
            int best = 0;
            for (int prev = num - k; prev < num; ++prev) {
                if (dp.count(prev)) {
                    best = max(best, dp[prev]);
                }
            }
            dp[num] = best + 1;
            maxLen = max(maxLen, dp[num]);
        }
        return maxLen;
    }
};
```

Saved

Ln 18, Col 32

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input