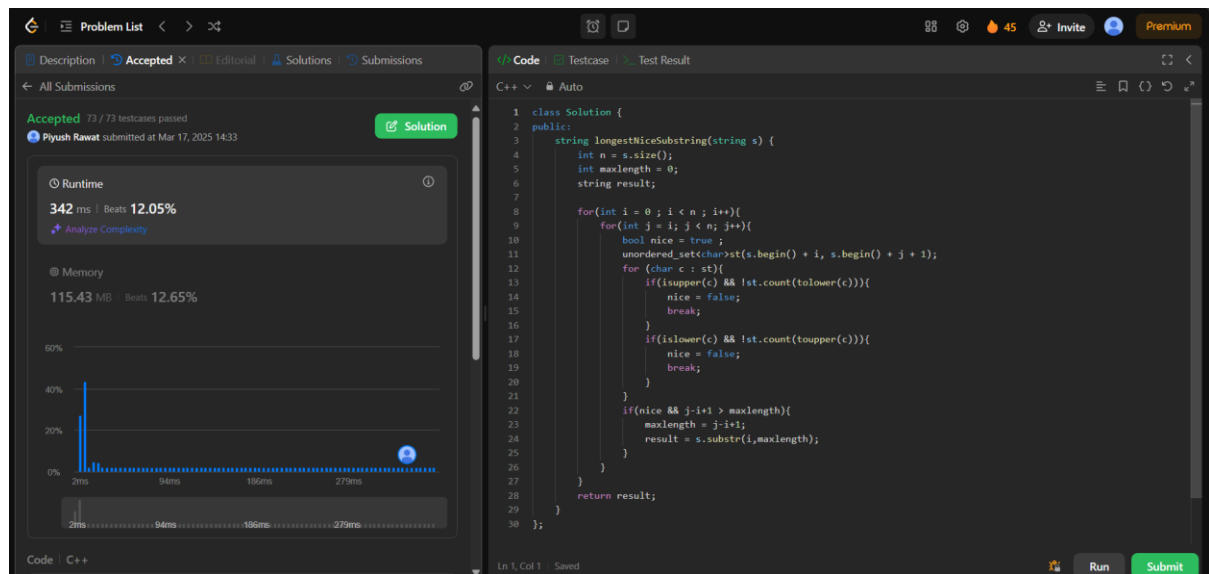
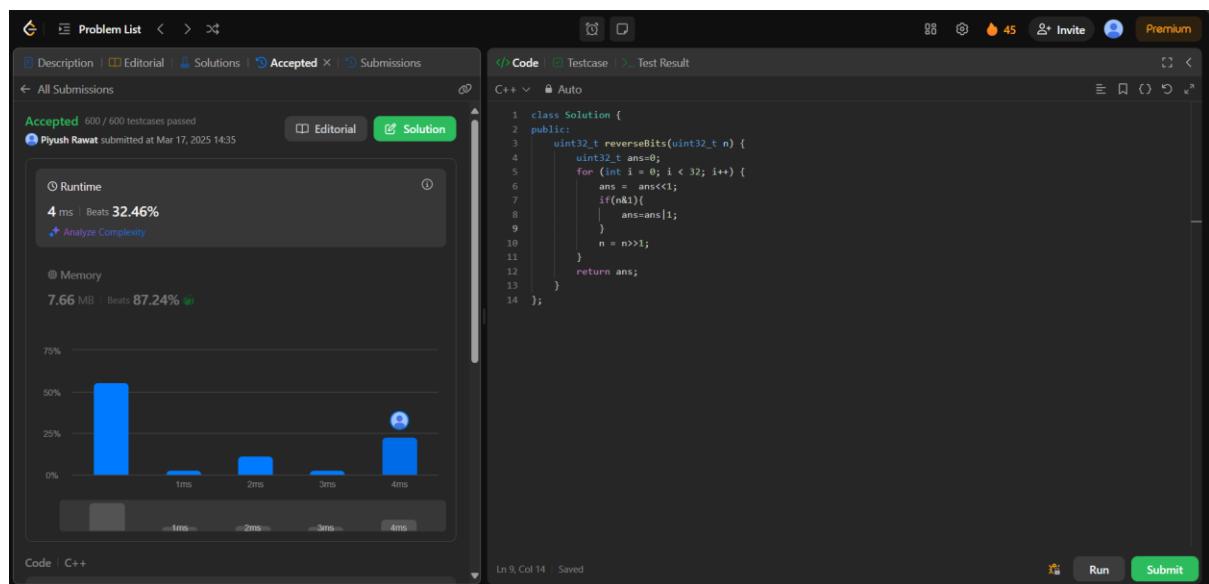


# AP Assignment 4

1. Longest Nice Substring: <https://leetcode.com/problems/longest-nice-substring/description/>



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



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

## AP Assignment 4

The screenshot shows a LeetCode submission for the 'Hamming Weight' problem. The left sidebar displays the submission status: 'Accepted' with 598 / 598 testcases passed, submitted by 'Piyush Rawat' on Mar 17, 2025 at 14:36. The runtime is 0 ms, beating 100.00% of solutions. The memory usage is 8.02 MB, beating 98.43% of solutions. A bar chart shows the runtime distribution across different time limits (1ms, 2ms, 3ms, 4ms). The main code editor shows the C++ solution for the 'Hamming Weight' problem, which uses a while loop to count the number of set bits in an integer.

```
1 class Solution {
2 public:
3     int hammingWeight(int n) {
4         int count = 0;
5         while(n != 0){
6             count += n % 2;
7             n = n / 2;
8         }
9         return count;
10    }
11 };
12
13 //class Solution {
14 // public:
15 //     int hammingWeight(int n) {
16 //         int count = 0;
17 //         for(int i = 31; i >= 0; i--){
18 //             if(((n >> i) & 1) == 1)
19 //                 count++;
20 //         }
21 //         return count;
22 //     }
23 // };
```

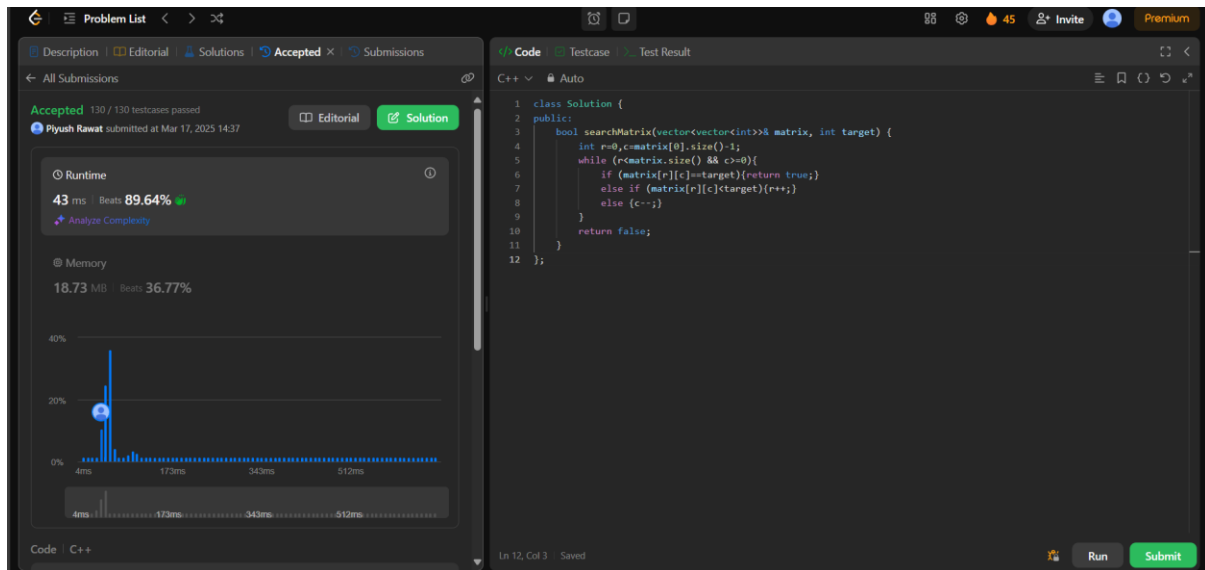
4. Maximum Subarray: <https://leetcode.com/problems/maximum-subarray/description/>

The screenshot shows a LeetCode submission for the 'Maximum Subarray' problem. The left sidebar displays the submission status: 'Accepted' with 210 / 210 testcases passed, submitted by 'Piyush Rawat' on Mar 17, 2025 at 14:37. The runtime is 0 ms, beating 100.00% of solutions. The memory usage is 71.52 MB, beating 97.76% of solutions. A bar chart shows the runtime distribution across different time limits (1ms, 2ms, 3ms, 4ms, 5ms). The main code editor shows the C++ solution for the 'Maximum Subarray' problem, which uses a sliding window approach to find the maximum sum of a contiguous subarray.

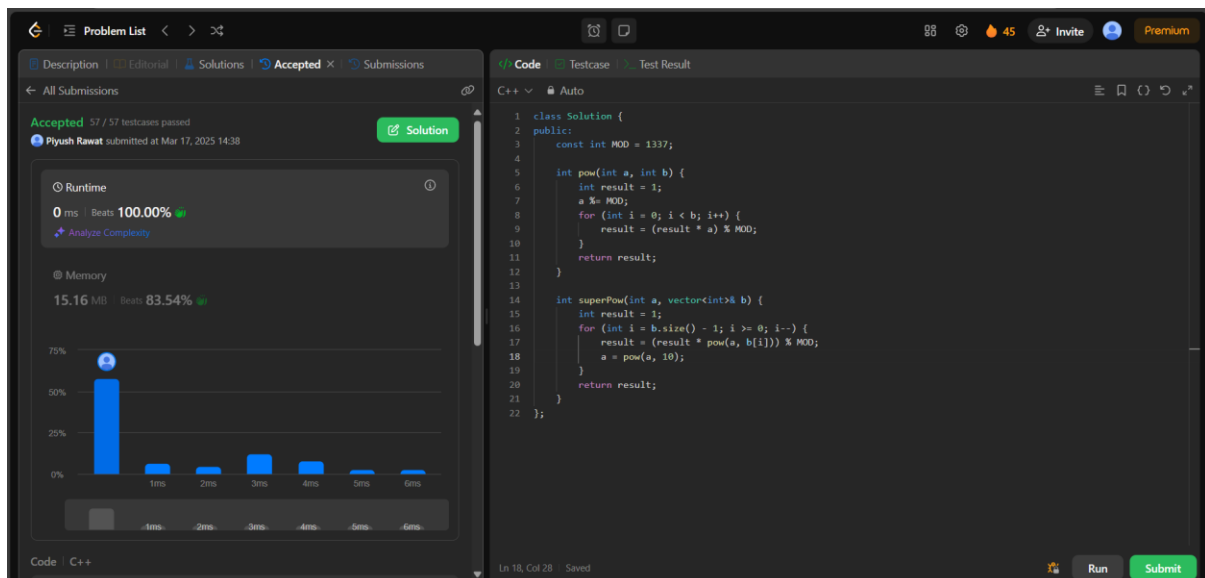
```
1 class Solution {
2 public:
3     int maxSubArray(vector<int>& nums) {
4         int n = nums.size();
5         long long maxi = LONG_MIN, sum = 0;
6         for (int i = 0; i < n; i++) {
7             sum += nums[i];
8             if (sum > maxi) {
9                 maxi = sum;
10            }
11            if (sum < 0) {
12                sum = 0;
13            }
14        }
15        return maxi;
16    }
17 };
18
```

5. Search a 2D Matrix II: <https://leetcode.com/problems/search-a-2d-matrix-ii/description/>

## AP Assignment 4

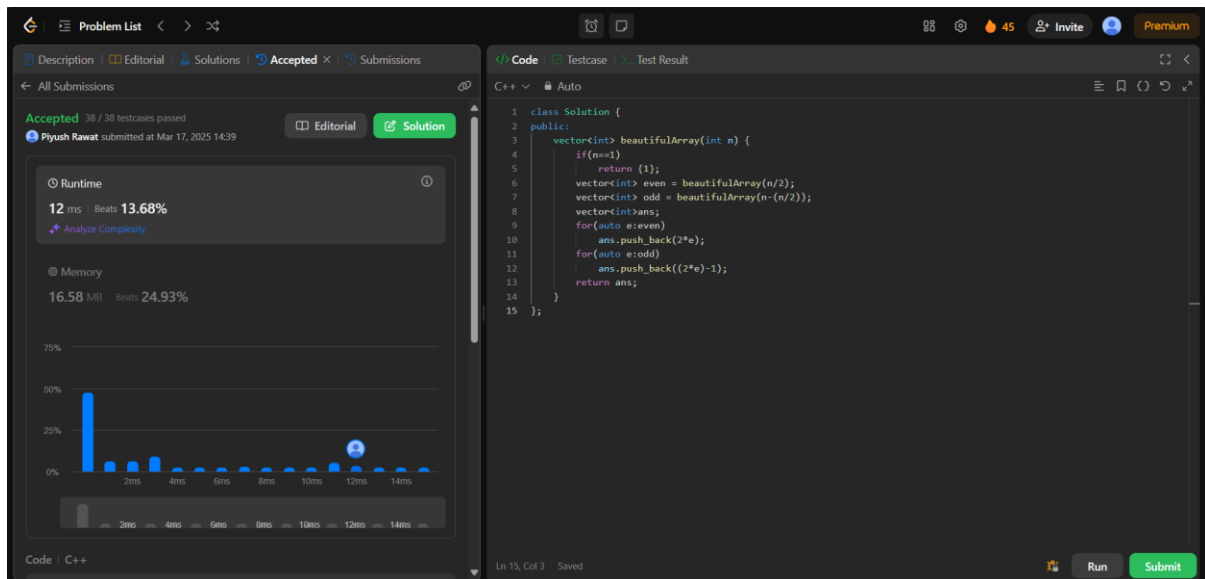


6. Super Pow: <https://leetcode.com/problems/super-pow/description/>

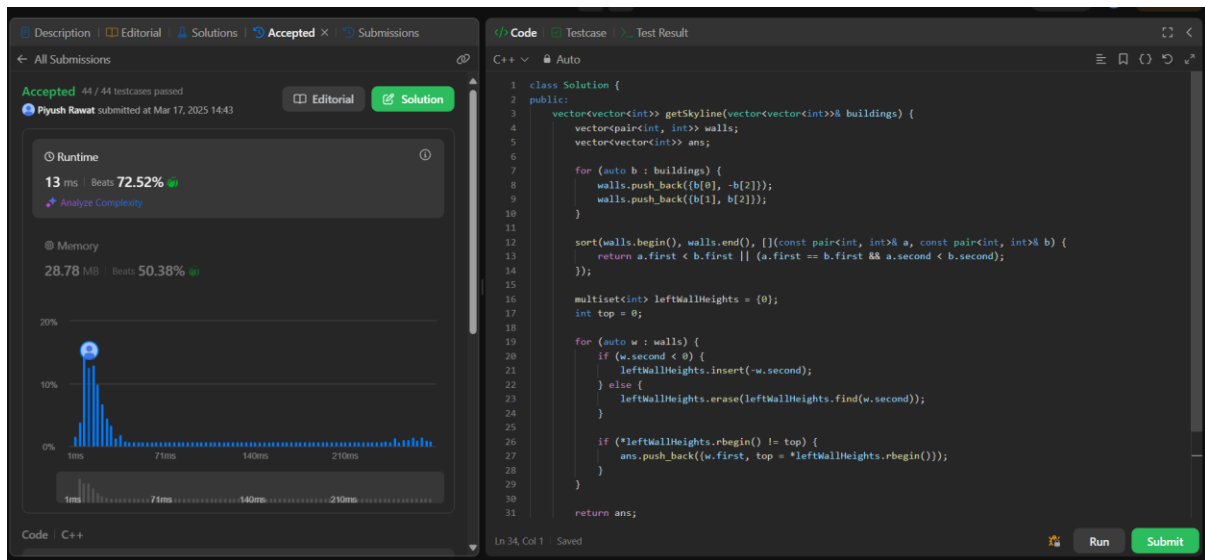


7. Beautiful Array: <https://leetcode.com/problems/beautiful-array/description/>

## AP Assignment 4

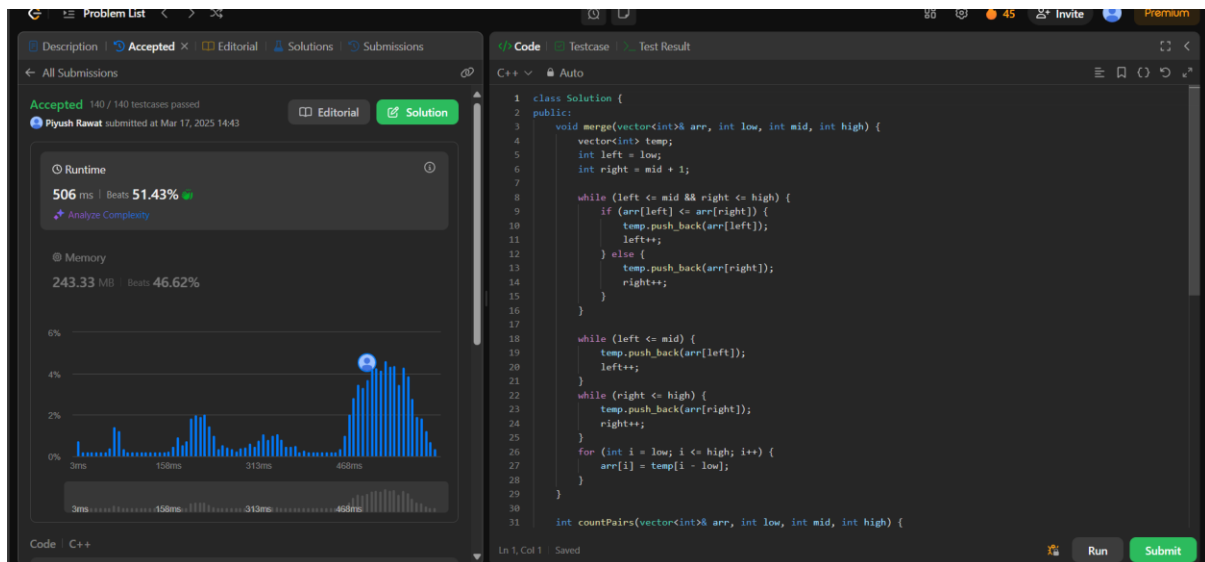


8. The Skyline Problem: <https://leetcode.com/problems/the-skyline-problem/description/>



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

# AP Assignment 4



10. Longest Increasing Subsequence II: <https://leetcode.com/problems/longest-increasing-subsequence-ii/description/>

