Assignment-4

Advanced Programming Lab

AAYUSH TEWARI (22BCS11444)

Question 1. A string s is **nice** if, for every letter of the alphabet that s contains, it appears **both** in uppercase and lowercase. For example, "abABB" is nice because 'A' and 'a' appear, and 'B' and 'b' appear. However, "abA" is not because 'b' appears, but 'B' does not. (LONGEST NICE SUBSTRING)

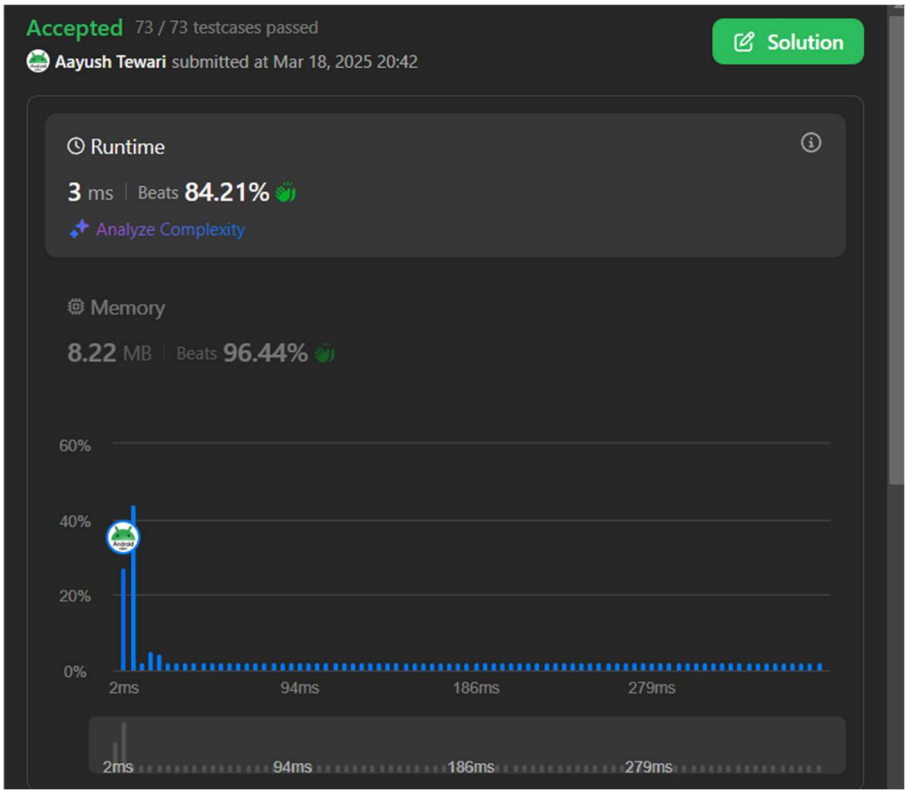
Solution 1. Code Snippet:

Code

C++ Auto

```cpp
string longestNiceSubstring(string s) {
        int n = s.size();
        int startIndex = -1, maxLength = 0;

        for (int i = 0; i < n; ++i) {
            int lower = 0, upper = 0;

            for (int j = i; j < n; ++j) {
                char c = s[j];

                if (islower(c)) lower |= 1 << (c - 'a');
                else upper |= 1 << (c - 'A');

                if (lower == upper && (j - i + 1) > maxLength) {
                    maxLength = j - i + 1;
                    startIndex = i;
                }
            }
        }

        if (startIndex == -1) {
            return "";
        } else {
            return s.substr(startIndex, maxLength);
        }
    }
```

OUTPUT:



Question 2. Reverse bits of a given 32 bits unsigned integer.(REVERSE BITS)

Solution 2.

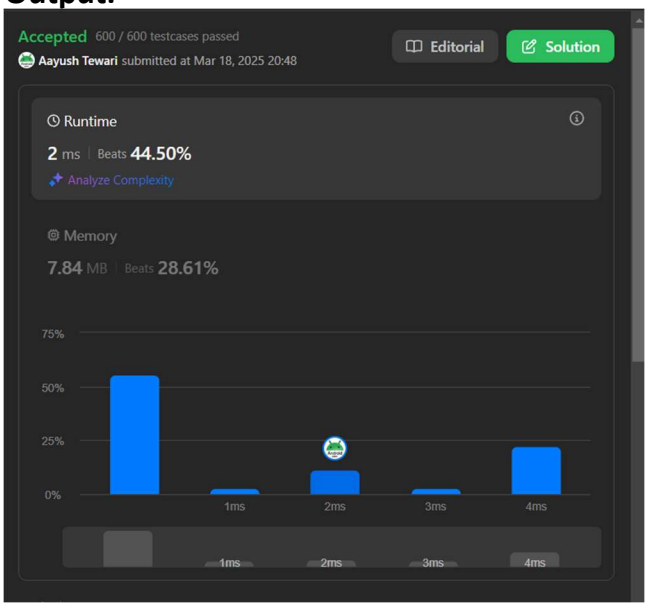
Code Snippet:

Code

C++ Auto

```
class Solution {
public:
    uint32_t reverseBits(uint32_t n) {
        uint32_t res = 0;

        for(int i=0; i < 32; i++){
            if(n >> i & 1){
                res |= 1 << 31-i;
            }
        }
        return res;
    }
};
```

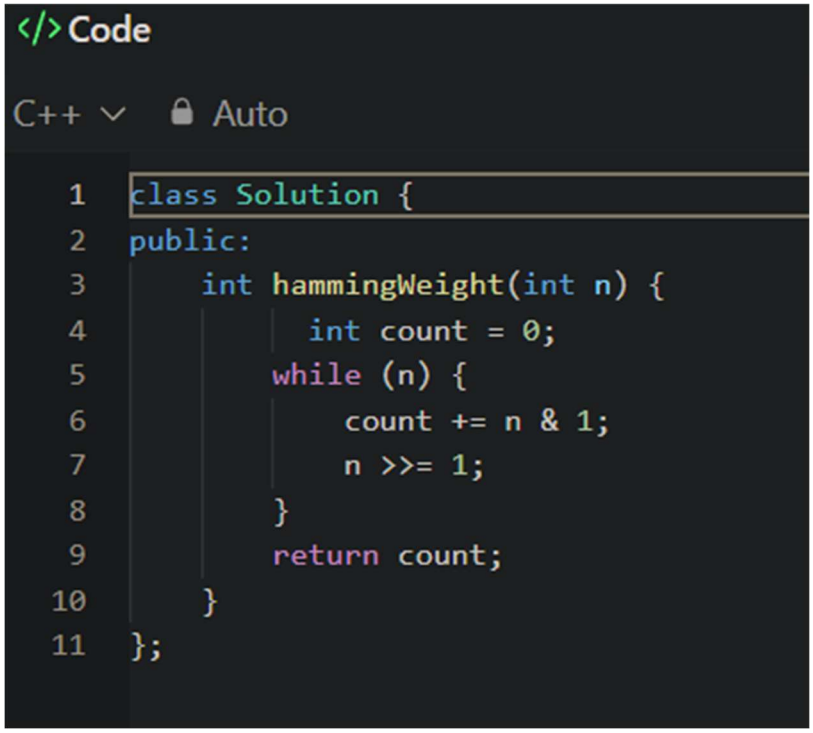
Output:



Question 3. Given a positive integer n, write a function that returns the number of set bits in its binary representation (also known as the Hamming weight).(NUMBER OF 1BITs)

Solution 3.

Code Snippet



Output:

Question 4. Given an integer array nums, find the subarray with the largest sum, and return *its sum*.(MAXIMUM SUBARRAY)

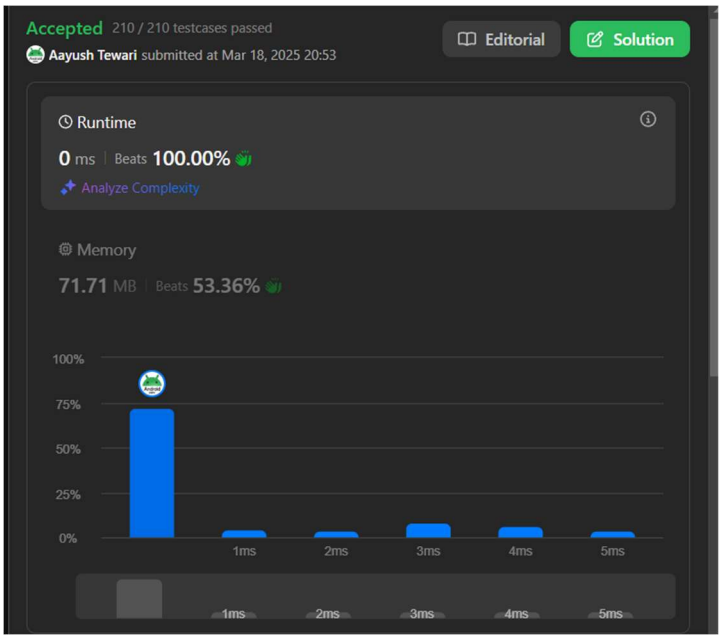
Solution 4.

Code Snippet:

C++ Auto

```cpp
class Solution {
public:
    int maxSubArray(vector<int>& nums) {
        int maxSum=INT_MIN;
        int currentSum=0;
        for(int i=0;i<nums.size();i++){
            currentSum+=nums[i];
            maxSum=max(maxSum,currentSum);

            if(currentSum<0){
                currentSum=0;
            }
        }
        return maxSum;
    }
};
```

Output:



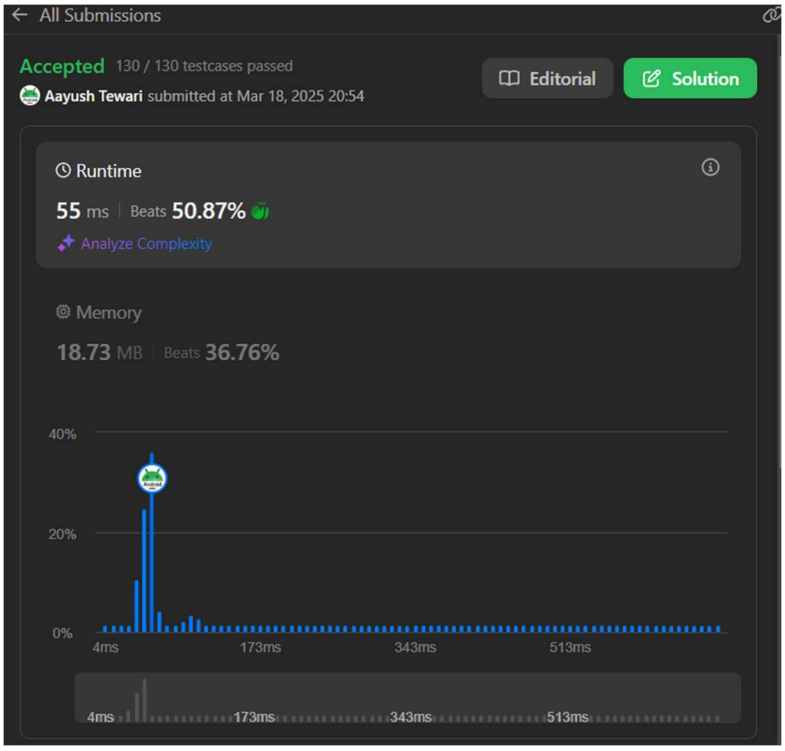
Question 5. Write an efficient algorithm that searches for a value target in an m x n integer matrix matrix. .(SEARCH A 2D MATRIX II)

Solution 5:

.

```
Auto
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;

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

        return false;
    }
};
```

Output:



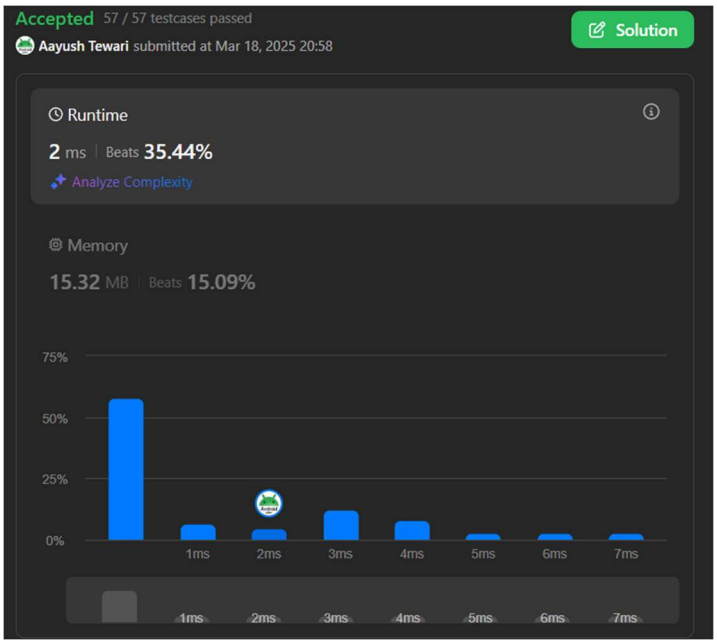
Question 6. Your task is to calculate a^b mod 1337 where a is a positive integer and b is an extremely large positive integer given in the form of an array.(SUPER POW)

Solution 6.

Code Snippet:

```cpp
class Solution {
public:
 const int MOD = 1337;

  int quickPow(int x, int y) {
        int res = 1;
        x %= MOD;
        while (y > 0) {
            if (y % 2 == 1) {
                res = (res * x) % MOD;
            }
            x = (x * x) % MOD;
            y /= 2;
        }
        return res;
    }

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

Output:



Question 7. Beautiful Array.

Solution 7.

Code Snippet

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

        while (result.size() < n) {
            vector<int> next;

            for (int num : result) {
                if (num * 2 - 1 <= n) {
                    next.push_back(num * 2 - 1);
                }
            }

            for (int num : result) {
                if (num * 2 <= n) {
                    next.push_back(num * 2);
                }
            }

            result = next;
        }

        return result;
    }
};
```

Output:

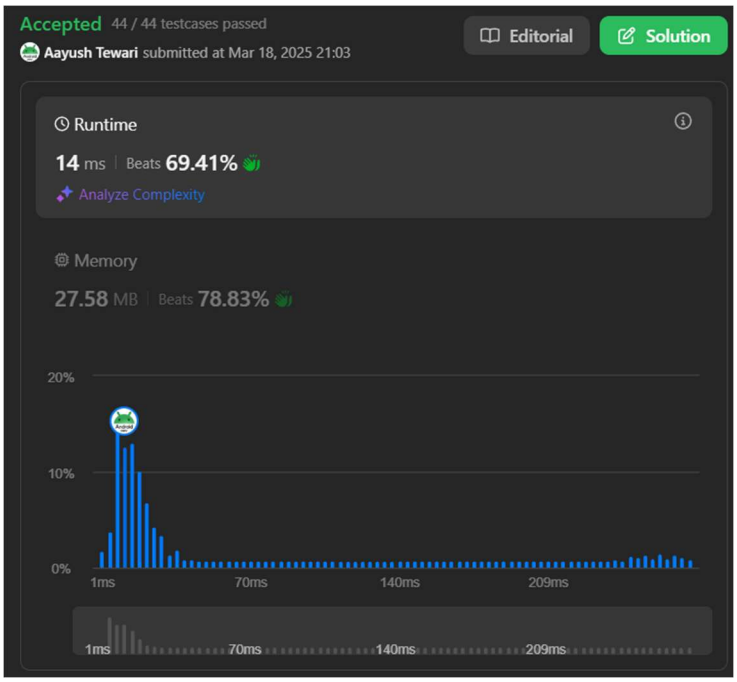
Question 8. A city's **skyline** is the outer contour of the silhouette formed by all the buildings in that city when viewed from a distance. Given the locations and heights of all the buildings, return *the **skyline** formed by these buildings collectively*..(**The Skyline problem**)

Solution 8.

Code Snippet:

```
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};
        vector<vector<int>> result;
        int prevMax = 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 currMax = *heights.rbegin();
```

Output:



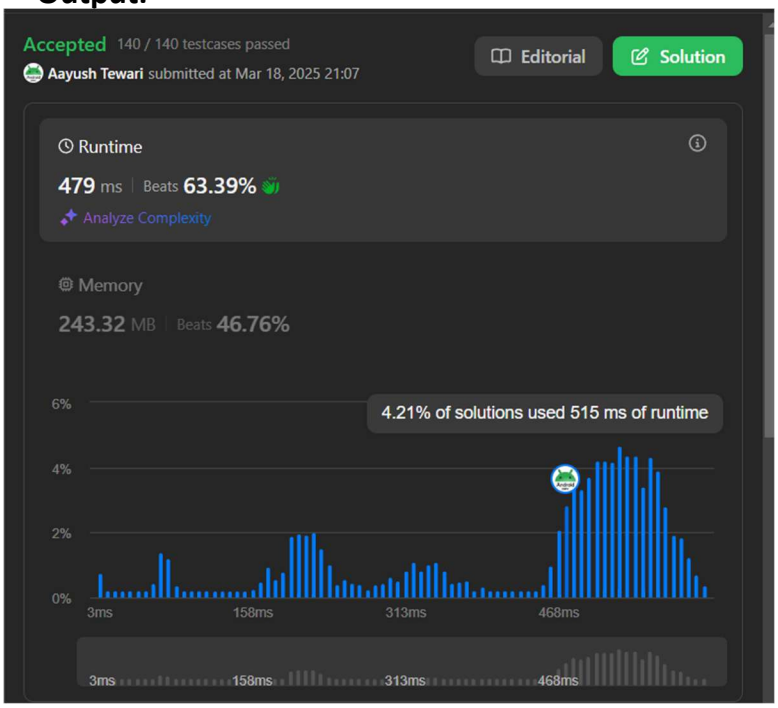
Question 9. Given an integer array nums, return *the number of* ***reverse pairs*** *in the array*.(**Reverse Pairs***)*

Solution 9.

Code Snippet:

```
class Solution {
public:
 int mergeAndCount(vector<int>& nums, int left, int mid, int right) {
        int count = 0, j = mid + 1;

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

        vector<int> temp;
        int i = left, k = mid + 1;

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

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

Output:



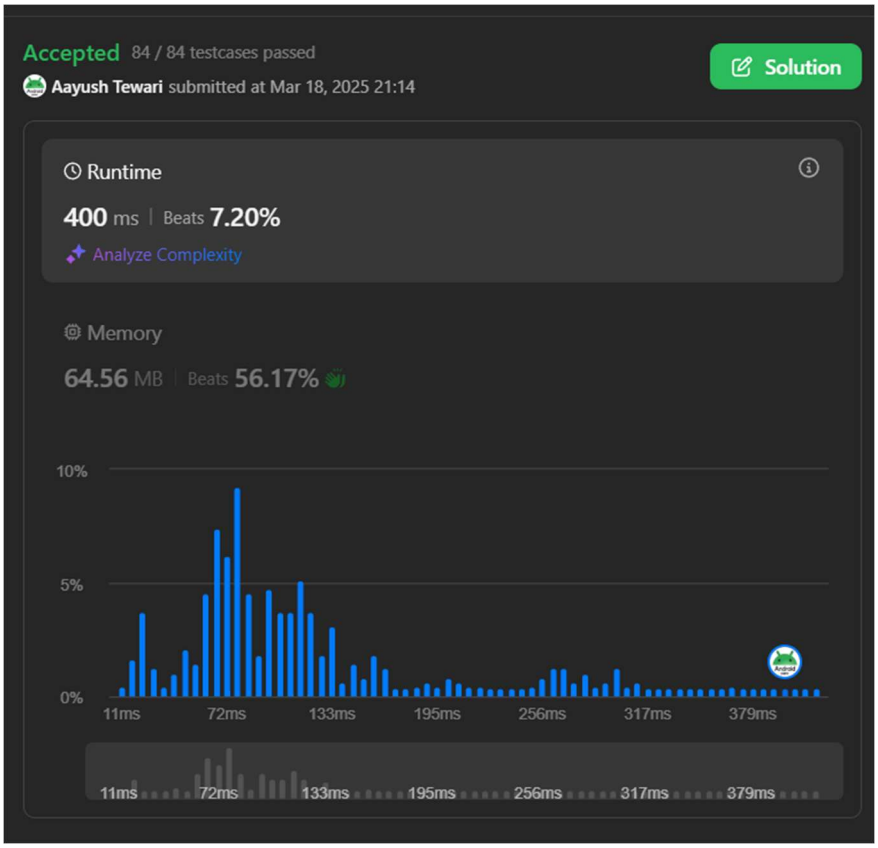
Question 10. Longest Increasing Subsequence II

Solution 10.

Code Snippet:

```
class Solution {
public:
    int lengthOfLIS(vector<int>& nums, int k) {
        int maxNum = *max_element(nums.begin(), nums.end());
        vector<int> segTree(4 * maxNum, 0);

        function<int(int, int, int, int, int)> query = [&](int node, int start, int end, int l, int r) -> int {
            if (r < start || end < l) return 0;
            if (l <= start && end <= r) return segTree[node];

            int mid = (start + end) / 2;
            return max(query(2 * node + 1, start, mid, l, r),
                       query(2 * node + 2, mid + 1, end, l, r));
        };


        function<void(int, int, int, int, int)> update = [&](int node, int start, int end, int idx, int value) {
            if (start == end) {
                segTree[node] = value;
                return;
            }
            int mid = (start + end) / 2;
            if (idx <= mid)
                update(2 * node + 1, start, mid, idx, value);
            else
                update(2 * node + 2, mid + 1, end, idx, value);
```

Output:


Accepted 84 / 84 testcases passed
Aayush Tewari submitted at Mar 18, 2025 21:14
Solution
Runtime
400 ms | Beats 7.20%
Analyze Complexity
Memory
64.56 MB | Beats 56.17%
10%
5%
0%
11ms
72ms
133ms
195ms
256ms
317ms
379ms