ASSIGNMENT – 4

Student Name: JEET BHARTI **UID: 22BCS14804**
Branch: BE-CSE **Section: FL_IOT-602 'A'**
Semester: 6th **Date of Performance: 21/02/25**
Subject Name: AP LAB-II **Subject Code: 22CSP-351**

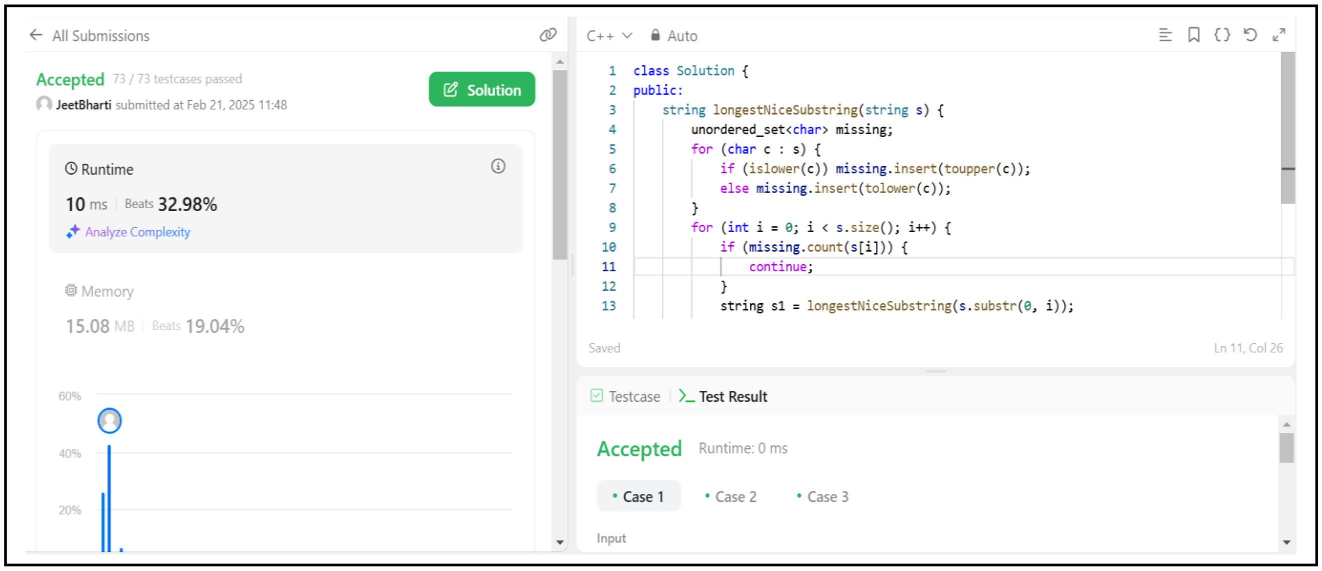
1763. Longest Nice Substring

Aim- 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.

Given a string s, return the longest substring of s that is nice. If there are multiple, return the substring of the earliest occurrence. If there are none, return an empty string.

CODE:-

```
class Solution {
public:
    string longestNiceSubstring(string s) {
        for (int i = 0; i < s.size(); i++) {
            if (s.find(toupper(s[i])) == string::npos || s.find(tolower(s[i])) == string::npos)
                return longestNiceSubstring(s.substr(i + 1));
        }
        return s;
    }
};
```



Accepted 73 / 73 testcases passed
JeetBharti submitted at Feb 21, 2025 11:48

Runtime
10 ms | Beats 32.98%
Analyze Complexity

Memory
15.08 MB | Beats 19.04%

```
class Solution {
public:
    string longestNiceSubstring(string s) {
        unordered_set<char> missing;
        for (char c : s) {
            if (islower(c)) missing.insert(toupper(c));
            else missing.insert(tolower(c));
        }
        for (int i = 0; i < s.size(); i++) {
            if (missing.count(s[i])) {
                continue;
            }
            string s1 = longestNiceSubstring(s.substr(0, i));
```

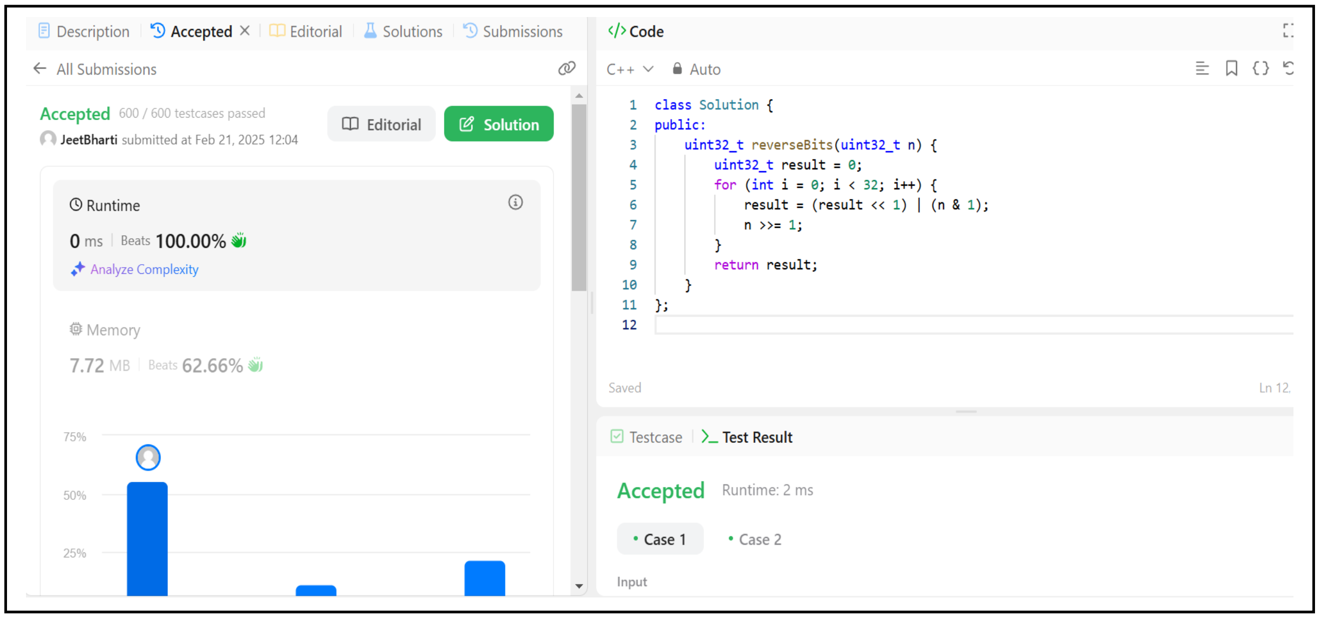
Accepted Runtime: 0 ms

190. Reverse Bits

Aim- Reverse bits of a given 32 bits unsigned integer.

CODE:-

```
class Solution {
public:
    uint32_t reverseBits(uint32_t n) {
        uint32_t result = 0;
        for (int i = 0; i < 32; i++) {
            result = (result << 1) | (n & 1);
            n >>= 1;
        }
        return result;
    }
};
```



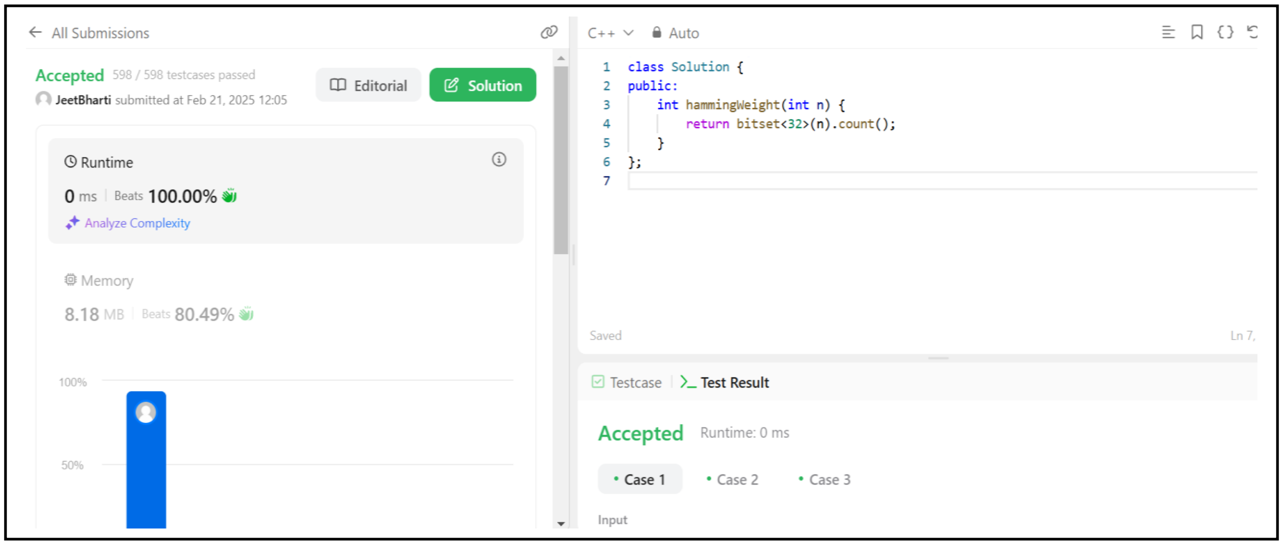
```
class Solution {
public:
    uint32_t reverseBits(uint32_t n) {
        uint32_t result = 0;
        for (int i = 0; i < 32; i++) {
            result = (result << 1) | (n & 1);
            n >>= 1;
        }
        return result;
    }
};
```

191. Number of 1 Bits

Aim- 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).

CODE:-

```
class Solution {
public:
    int hammingWeight(int n) {
        return bitset<32>(n).count();
    }
};
```


All Submissions

Accepted 598 / 598 testcases passed
JeetBharti submitted at Feb 21, 2025 12:05

Runtime
0 ms | Beats 100.00%

Memory
8.18 MB | Beats 80.49%

```
class Solution {
public:
    int hammingWeight(int n) {
        return bitset<32>(n).count();
    }
};
```

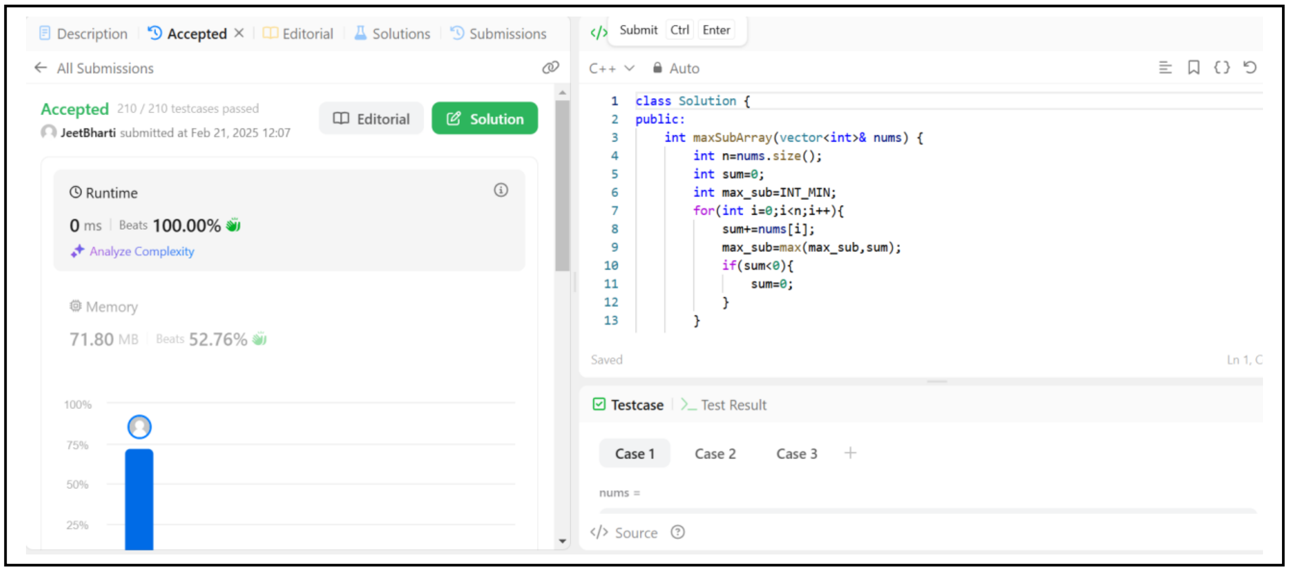
Test Result
Accepted Runtime: 0 ms
Case 1 Case 2 Case 3

53. Maximum Subarray

Aim- Given an integer array nums, find the subarray with the largest sum, and return its sum.

CODE:-

```
class Solution {
public:
    int maxSubArray(vector<int>& nums) {
        int maxSum = nums[0], currentSum = nums[0];
        for (int i = 1; i < nums.size(); i++) {
            currentSum = max(nums[i], currentSum + nums[i]);
            maxSum = max(maxSum, currentSum);
        }
        return maxSum;
    }
};
```



Accepted 210 / 210 testcases passed

JeetBharti submitted at Feb 21, 2025 12:07

Runtime
0 ms | Beats 100.00%
Analyze Complexity

Memory
71.80 MB | Beats 52.76%

```
class Solution {
public:
    int maxSubArray(vector<int>& nums) {
        int n=nums.size();
        int sum=0;
        int max_sub=INT_MIN;
        for(int i=0;i<n;i++){
            sum+=nums[i];
            max_sub=max(max_sub,sum);
            if(sum<0){
                sum=0;
            }
        }
```

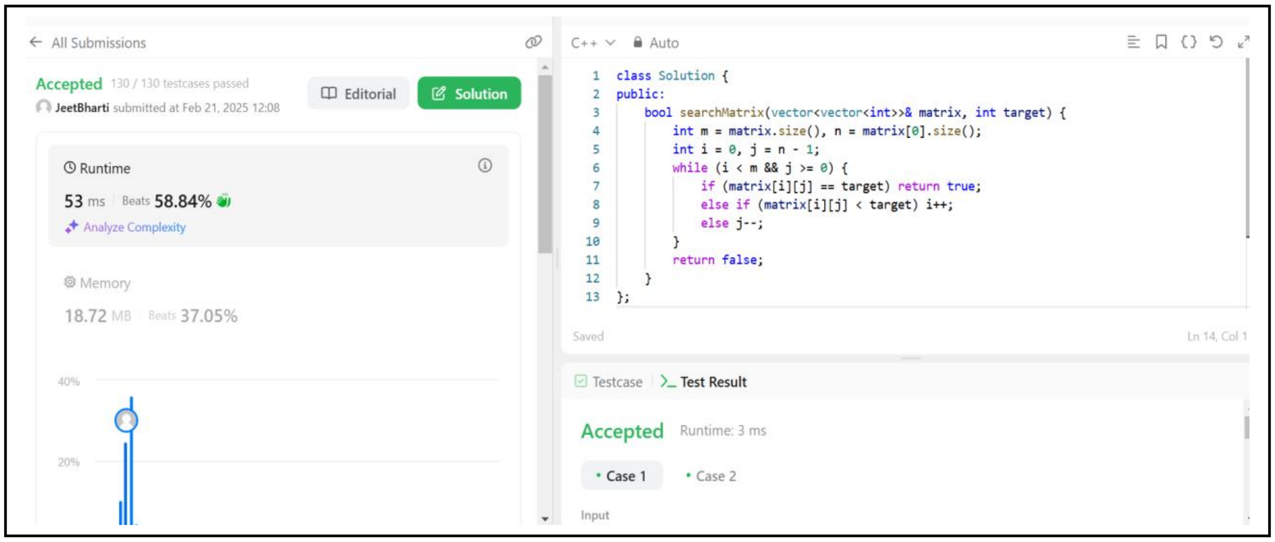
240. Search a 2D Matrix

Aim- Write an efficient algorithm that searches for a value target in an m x n integer matrix matrix. This matrix has the following properties:

- ***Integers in each row are sorted in ascending from left to right.***
- ***Integers in each column are sorted in ascending from top to bottom.***

CODE:-

```
class Solution {
public:
    bool searchMatrix(vector<vector<int>>& matrix, int target) {
        int m = matrix.size(), n = matrix[0].size();
        int i = 0, j = n - 1;
        while (i < m && j >= 0) {
            if (matrix[i][j] == target) return true;
            else if (matrix[i][j] < target) i++;
            else j--;
        }
        return false;
    }
};
```



Accepted 130 / 130 testcases passed
JeetBharti submitted at Feb 21, 2025 12:08

Runtime
53 ms Beats 58.84%
Analyze Complexity

Memory
18.72 MB Beats 37.05%

```
class Solution {
public:
    bool searchMatrix(vector<vector<int>>& matrix, int target) {
        int m = matrix.size(), n = matrix[0].size();
        int i = 0, j = n - 1;
        while (i < m && j >= 0) {
            if (matrix[i][j] == target) return true;
            else if (matrix[i][j] < target) i++;
            else j--;
        }
        return false;
    }
};
```

Saved
Ln 14, Col 1

Accepted Runtime: 3 ms

372. Super Pow

Aim- 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.

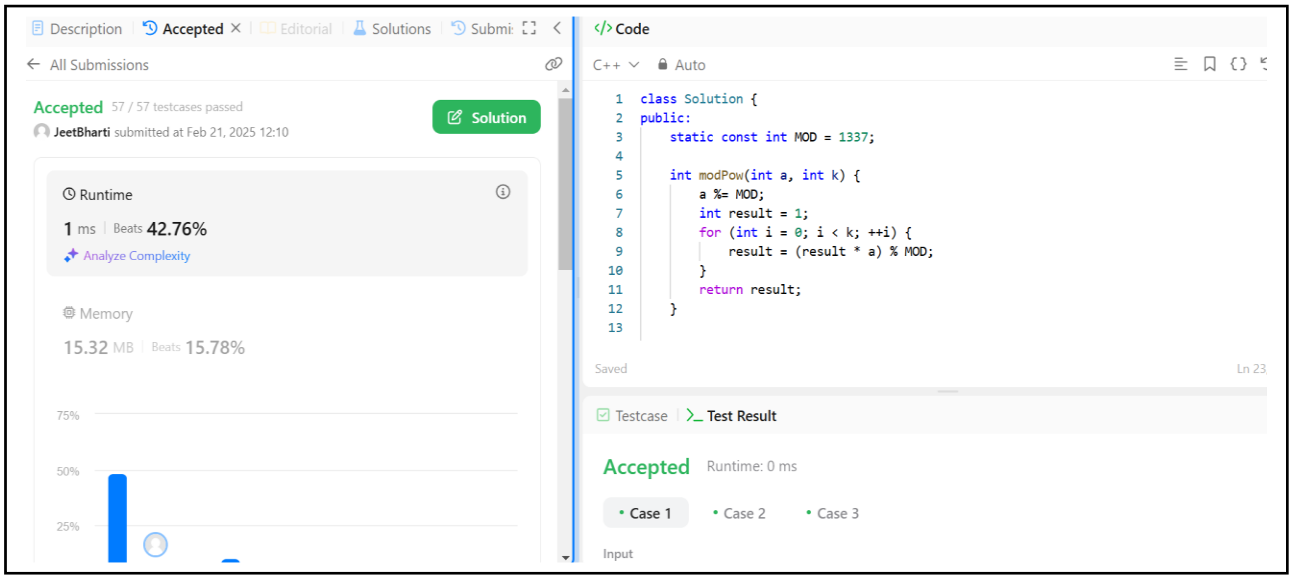
CODE:-

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

    int modPow(int a, int k) {
        a %= MOD;
        int result = 1;
```

```
            for (int i = 0; i < k; ++i) {
                result = (result * a) % MOD;
            }
            return result;
        }

        int superPow(int a, vector<int>& b) {
            if (b.empty()) return 1;
            int lastDigit = b.back();
            b.pop_back();
            int part1 = modPow(a, lastDigit);
            int part2 = modPow(superPow(a, b), 10);
            return (part1 * part2) % MOD;
        }
    };
```



Accepted 57 / 57 testcases passed
JeetBharti submitted at Feb 21, 2025 12:10

Runtime
1 ms | Beats 42.76%

Memory
15.32 MB | Beats 15.78%

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

    int modPow(int a, int k) {
        a %= MOD;
        int result = 1;
        for (int i = 0; i < k; ++i)
            result = (result * a) % MOD;
        }
        return result;
    }
```

Accepted Runtime: 0 ms