



ASSIGNMENT – 4

Student Name: JEET BHARTI

Branch: BE-CSE

Semester: 6th

Subject Name: AP LAB-II

UID: 22BCS14804

Section: FL_IOT-602 'A'

Date of Performance: 21/02/25

Subject Code: 22CSP-351

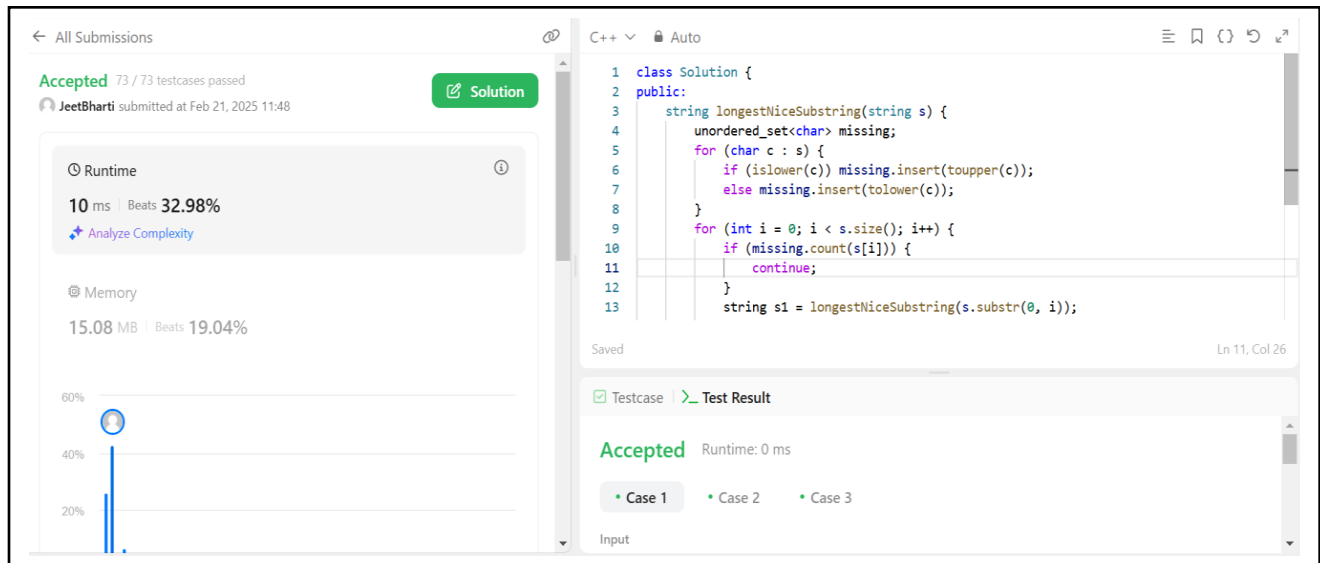
1763. Longest Nice Substring

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;
    }
};
```



The screenshot shows a coding platform interface. On the left, a sidebar displays submission details: 'Accepted 73 / 73 testcases passed', 'JeetBharti submitted at Feb 21, 2025 11:48', and a 'Solution' button. Below this, performance metrics are shown: 'Runtime 10 ms Beats 32.98%' and 'Memory 15.08 MB Beats 19.04%'. A bar chart at the bottom left shows the user's performance relative to others. The main area displays the C++ code for a 'longestNiceSubstring' function. The code uses an unordered_set to track characters and iterates through the string to find the longest substring with unique characters. The test result section shows 'Accepted' with a runtime of 0 ms and three test cases passed.

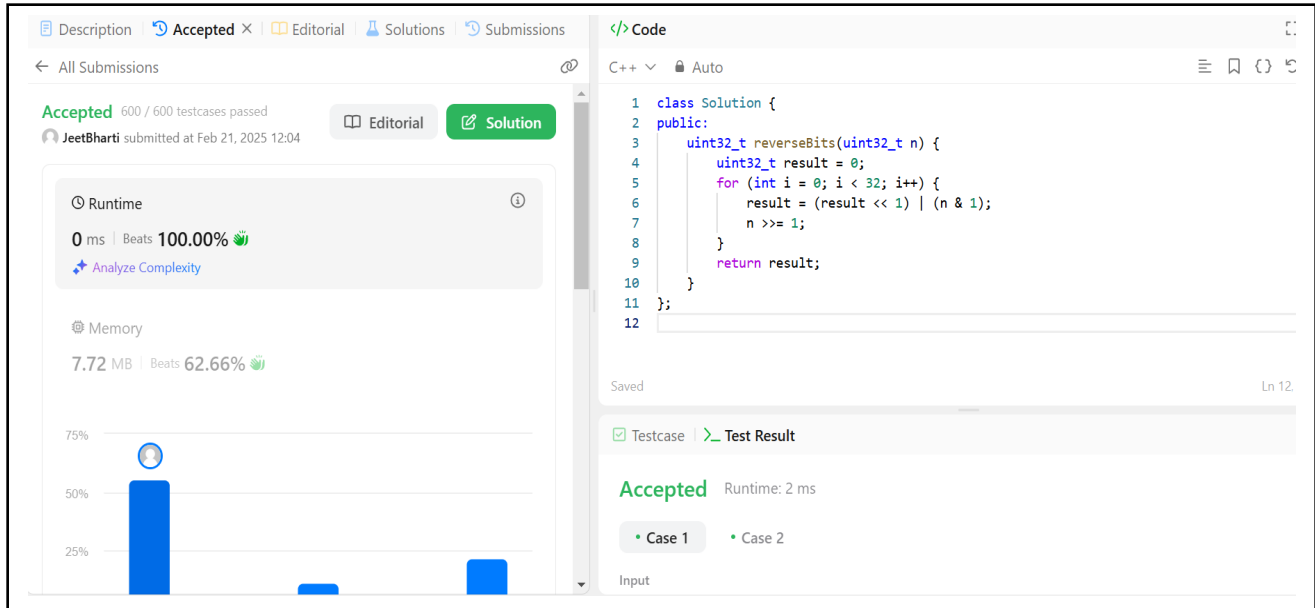
```
1 class Solution {
2 public:
3     string longestNiceSubstring(string s) {
4         unordered_set<char> missing;
5         for (char c : s) {
6             if (islower(c)) missing.insert(toupper(c));
7             else missing.insert(tolower(c));
8         }
9         for (int i = 0; i < s.size(); i++) {
10             if (missing.count(s[i])) {
11                 continue;
12             }
13             string s1 = longestNiceSubstring(s.substr(0, i));
```

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;
    }
};
```



The screenshot displays a coding competition interface. On the left, the 'All Submissions' tab is active, showing a submission by 'JeetBharti' that is 'Accepted' with 600/600 testcases passed. The submission details include a runtime of 0 ms (Beats 100.00%) and memory usage of 7.72 MB (Beats 62.66%). A bar chart shows the submission's performance relative to others. On the right, the 'Code' tab is active, displaying a C++ solution for reversing the bits of a 32-bit unsigned integer. The code uses a loop to shift bits and build the result. Below the code, the 'Testcase' tab shows the solution is 'Accepted' with a runtime of 2 ms. The input field is visible at the bottom.

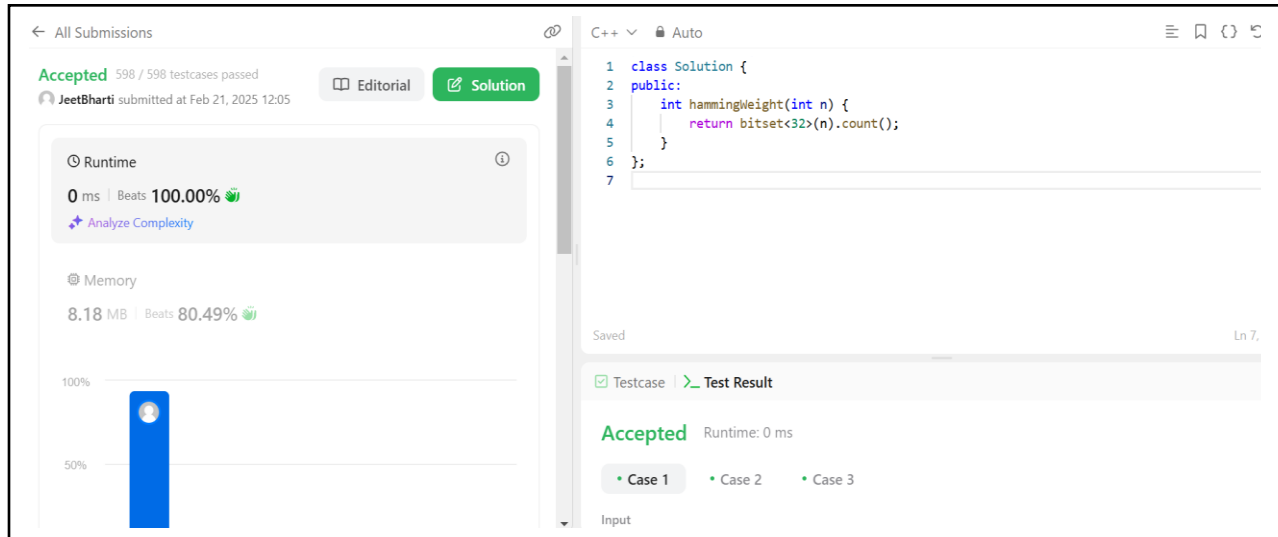
```
1 class Solution {
2 public:
3     uint32_t reverseBits(uint32_t n) {
4         uint32_t result = 0;
5         for (int i = 0; i < 32; i++) {
6             result = (result << 1) | (n & 1);
7             n >>= 1;
8         }
9         return result;
10    }
11 };
12
```

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();
    }
};
```



The screenshot shows a submission for the 'Maximum Subarray' problem. The status is 'Accepted' with 598 / 598 testcases passed. The user 'JeetBharti' submitted it on Feb 21, 2025 at 12:05. The runtime is 0 ms (Beats 100.00%) and memory is 8.18 MB (Beats 80.49%). The code is in C++ and implements a solution for finding the maximum subarray sum. The test result shows 'Accepted' with a runtime of 0 ms for Case 1.

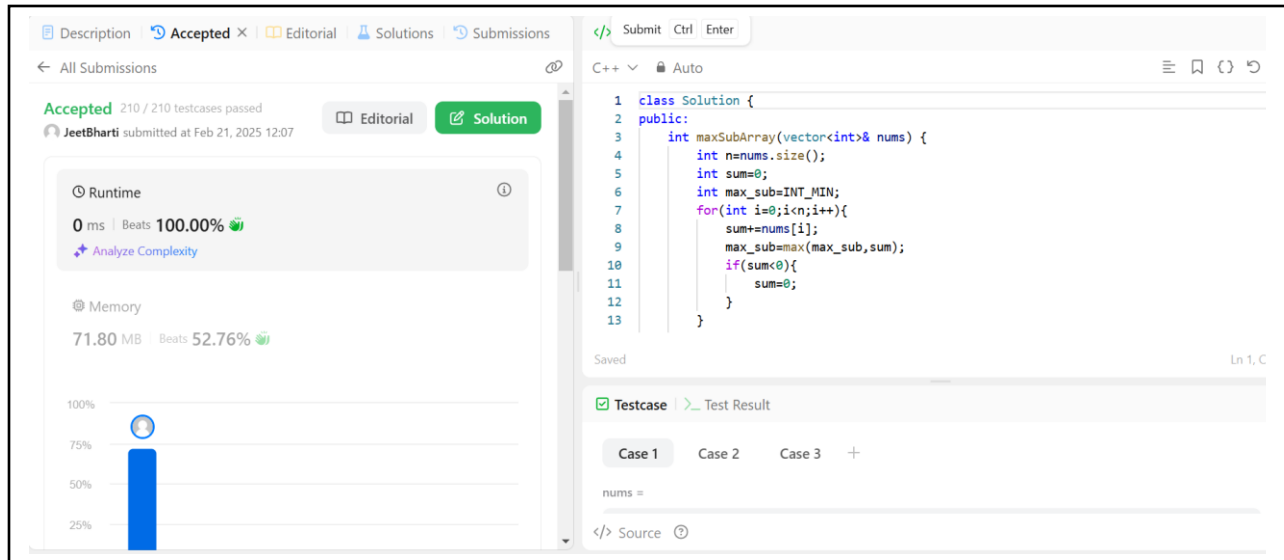
```
1 class Solution {
2 public:
3     int hammingWeight(int n) {
4         return bitset<32>(n).count();
5     }
6 };
7
```

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;
    }
};
```



240. Search a 2D Matrix

Aim- Write an efficient algorithm that searches for a value target in an $m \times n$ 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.*

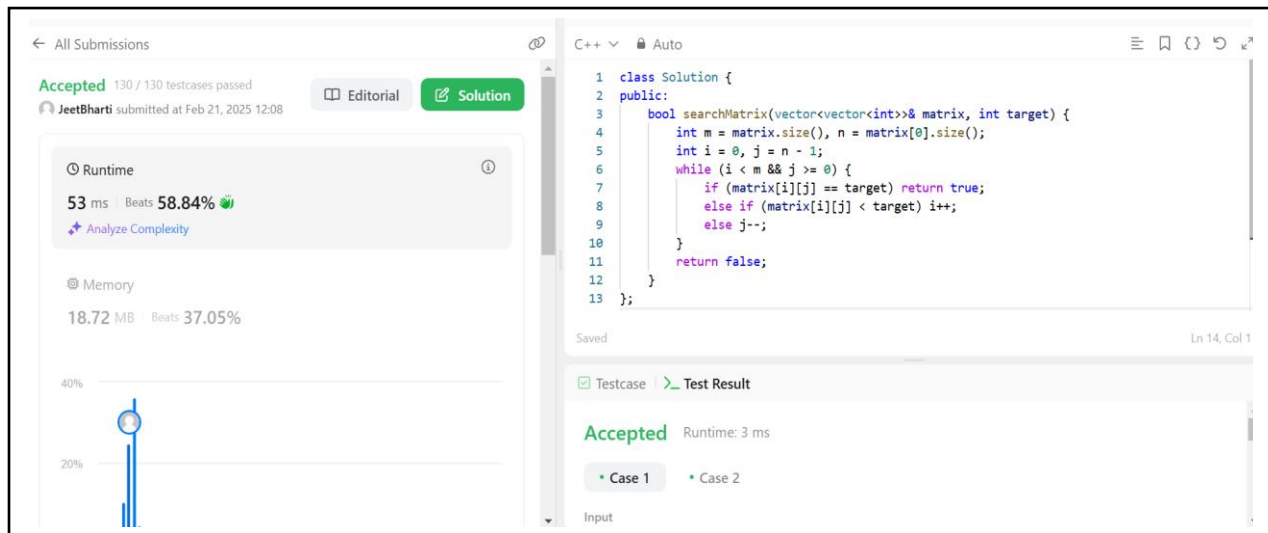
CODE:-

```
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%

```

1 class Solution {
2 public:
3     bool searchMatrix(vector<vector<int>>& matrix, int target) {
4         int m = matrix.size(), n = matrix[0].size();
5         int i = 0, j = n - 1;
6         while (i < m && j >= 0) {
7             if (matrix[i][j] == target) return true;
8             else if (matrix[i][j] < target) i++;
9             else j--;
10        }
11        return false;
12    }
13 };

```

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

372. Super Pow

Aim- Your task is to calculate $a^b \bmod 1337$ where a is a positive integer and b is an extremely large positive integer given in the form of an array.

CODE:-

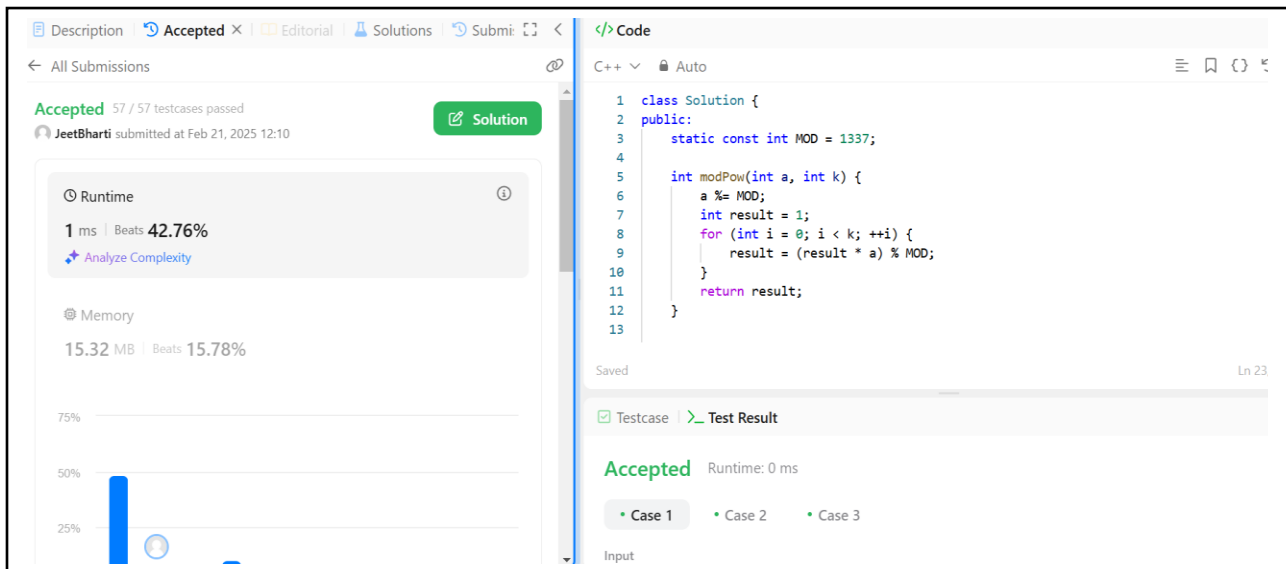
```

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;  
    }  
};
```



The screenshot displays a submission page for a C++ problem. The submission is marked as "Accepted" with 57/57 testcases passed. The user "JeetBharti" submitted it on Feb 21, 2025, at 12:10. The runtime is 1 ms, beating 42.76% of other submissions, and the memory usage is 15.32 MB, beating 15.78%. The code is as follows:

```
1 class Solution {  
2 public:  
3     static const int MOD = 1337;  
4  
5     int modPow(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 }
```

The test result shows "Accepted" with a runtime of 0 ms. The input is shown as "Case 1", "Case 2", and "Case 3".