Experiment 2

Student Name: Nitesh Bamber	**UID:** 22BCS13791
Branch: BE-CSE	**Section/Group:** 602/A
Semester: 6th	**Date of Performance:** 4-02-25
Subject Name: Advanced Programming - II	**Subject Code:** 22CSP-351

1. Aim:

To solve the following problems on Leetcode, with the goal of optimizing solutions in terms of time complexity and space efficiency:

(I) **No. of 1 Bits** –
Aim: Given a positive integer `n`, the objective is to return the number of set bits in its binary representation (also known as the Hamming weight). This helps in understanding bit manipulation and the efficient counting of 1-bits.

(II) **Median of Two Sorted Arrays** –
Aim: Given two sorted arrays `nums1` and `nums2` of size `m` and `n` respectively, the objective is to find and return the median of these two sorted arrays with an optimal time complexity of O(log(m+n)), which involves leveraging binary search techniques.

(III) **Sort Colors** –
Aim: Given an array `nums` with `n` objects colored red, white, or blue, the goal is to sort them in-place such that objects of the same color are adjacent, in the order red, white, and blue. The problem focuses on sorting without using library sort functions and utilizing counting or the Dutch National Flag algorithm.

(IV) **Reverse Bits** –
Aim: Given a 32-bit unsigned integer, the task is to reverse its bits. This problem emphasizes bitwise operations and understanding the concept of bit reversal.

(V) **Maximum Subarray** –
Aim: Given an integer array `nums`, the objective is to find the contiguous subarray that has the largest sum, using the Kadane's algorithm to achieve an optimal O(n) time complexity.

(VI) **Search a 2D Matrix II** –
Aim: Given an m x n matrix where each row and column is sorted, the goal is to search for a target value in the matrix. This problem requires efficient traversal techniques to minimize time complexity, using the "top-right" search approach to find the target in O(m + n) time.

(VII) **First Bad Version** –
Aim: Given a versioning system, the goal is to find the first bad version using a binary search approach, ensuring optimal performance with O(log n) time complexity.

(VII) **First Bad Version** –
Aim: Given a versioning system, the goal is to find the first bad version using a binary search approach, ensuring optimal performance with O(log n) time complexity.

(VIII) **Top K Frequent Elements** –
Aim: Given an integer array nums, the objective is to find the k most frequent elements. This problem aims to utilize a priority queue (max-heap) or bucket sort to efficiently retrieve the top k elements with an optimal time complexity of O(n log k).

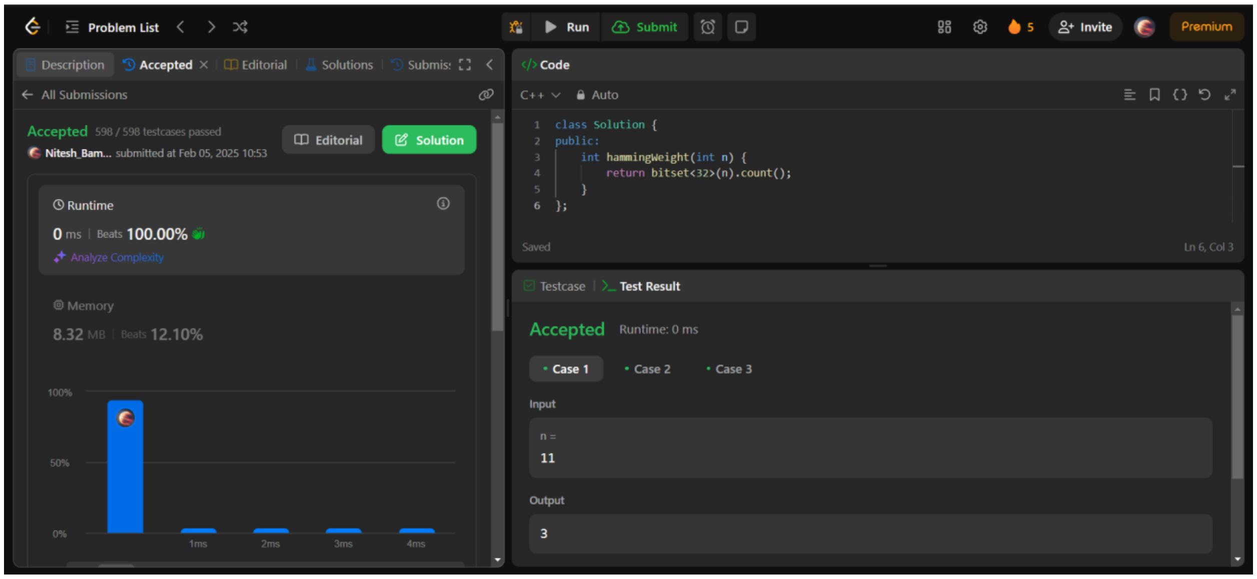
(IX) **Merge Sorted Arrays** –
Aim: Given two sorted integer arrays nums1 and nums2, the goal is to merge them into a single sorted array. This problem focuses on merging with an optimal in-place solution by iterating from the end of the arrays, ensuring a time complexity of O(m + n).

2. Objective:
To efficiently solve a set of algorithmic problems on Leetcode, optimizing for time and space complexity, by implementing advanced techniques such as bit manipulation, binary search, sorting algorithms, and dynamic programming, with the goal of enhancing problem-solving skills and achieving optimal solutions for real-world applications.

3. Implementation/Code:

Problem No. 191 : Number of 1 Bits
Code:

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



Accepted 598 / 598 testcases passed
Nitesh_Bam... submitted at Feb 05, 2025 10:53

Runtime
0 ms | Beats 100.00%

Memory
8.32 MB | Beats 12.10%

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

Accepted Runtime: 0 ms

Case 1 · Case 2 · Case 3

Input

```
n =
11
```

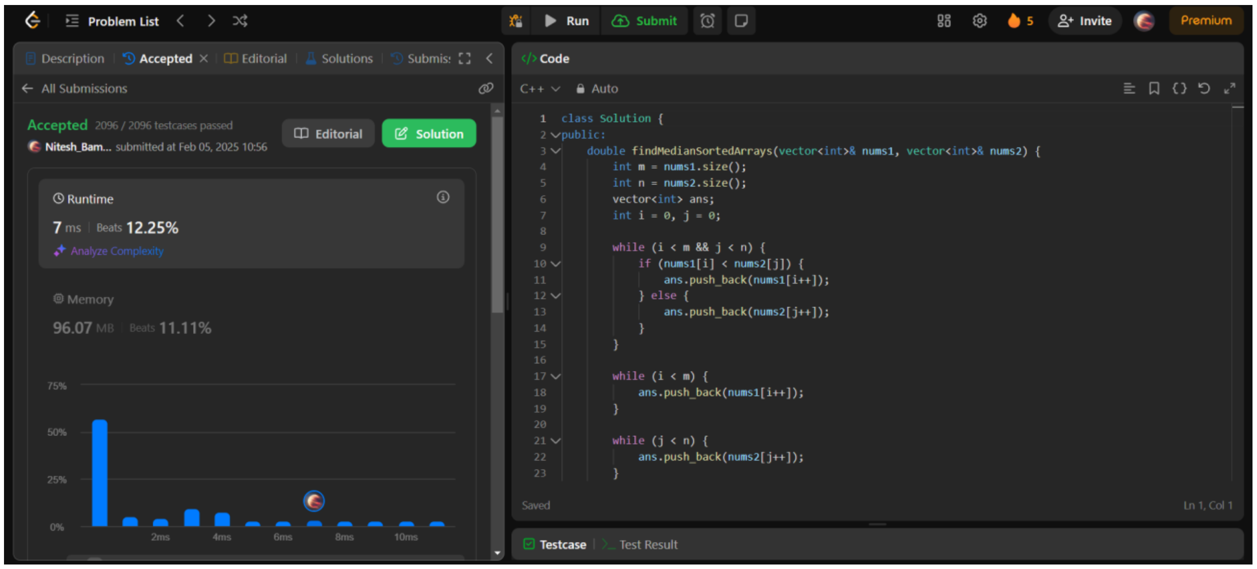
Output

```
3
```

Problem No. 4 : Median of Two Sorted Arrays

Code:

```
class Solution {
public:
    double findMedianSortedArrays(vector<int>& nums1, vector<int>& nums2) {
        int m = nums1.size();
        int n = nums2.size();
        vector<int> ans;
        int i = 0, j = 0;

        while (i < m && j < n) {
            if (nums1[i] < nums2[j]) {
                ans.push_back(nums1[i++]);
            } else {
                ans.push_back(nums2[j++]);
            }
        }

        while (i < m) {
            ans.push_back(nums1[i++]);
        }

        while (j < n) {
            ans.push_back(nums2[j++]);
        }

        int totalSize = m + n;
        if (totalSize % 2 == 1) {
            return ans[totalSize / 2];
        } else {
            return (ans[totalSize / 2 - 1] + ans[totalSize / 2]) / 2.0;
        }
    }
};
```



Accepted 2096 / 2096 testcases passed
Nitesh_Bam... submitted at Feb 05, 2025 10:56

Runtime: 7 ms | Beats 12.25%

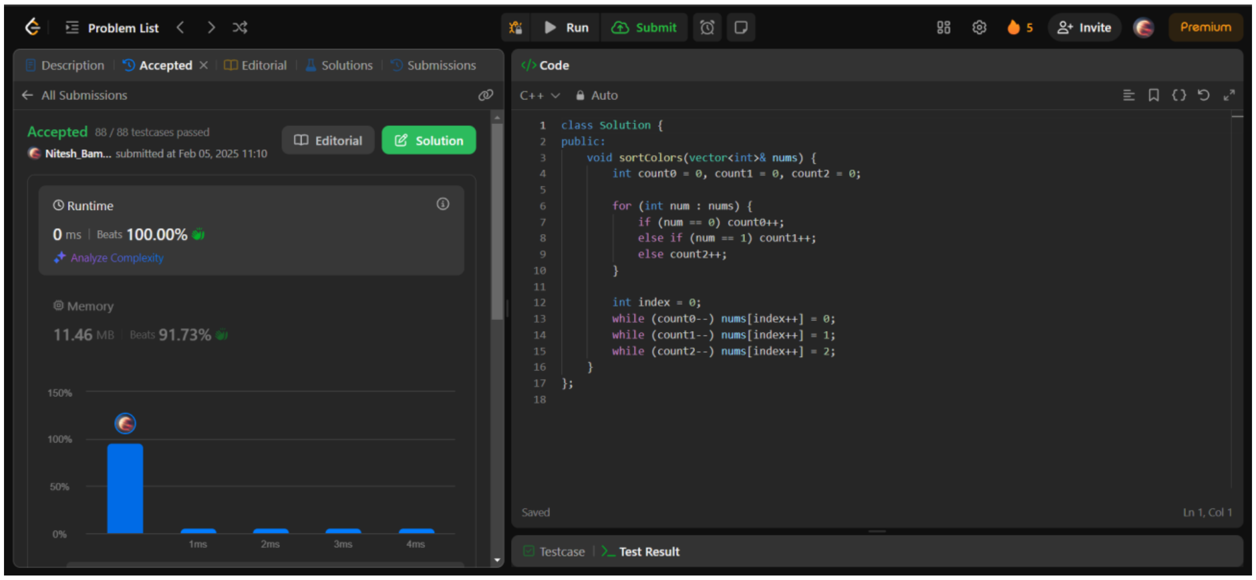
Memory: 96.07 MB | Beats 11.11%

```
class Solution {
public:
    double findMedianSortedArrays(vector<int>& nums1, vector<int>& nums2) {
        int m = nums1.size();
        int n = nums2.size();
        vector<int> ans;
        int i = 0, j = 0;

        while (i < m && j < n) {
            if (nums1[i] < nums2[j]) {
                ans.push_back(nums1[i++]);
            } else {
                ans.push_back(nums2[j++]);
            }
        }

        while (i < m) {
            ans.push_back(nums1[i++]);
        }

        while (j < n) {
            ans.push_back(nums2[j++]);
        }
```

Problem No. 75 : Sort Colors

Code:

```
class Solution {
public:
    void sortColors(vector<int>& nums) {
        int count0 = 0, count1 = 0, count2 = 0;

        for (int num : nums) {
            if (num == 0) count0++;
            else if (num == 1) count1++;
            else count2++;
        }

        int index = 0;
        while (count0--) nums[index++] = 0;
        while (count1--) nums[index++] = 1;
        while (count2--) nums[index++] = 2;
    }
};
```



Accepted 88 / 88 testcases passed
Nitesh_Bam... submitted at Feb 05, 2025 11:10

Runtime
0 ms | Beats 100.00%

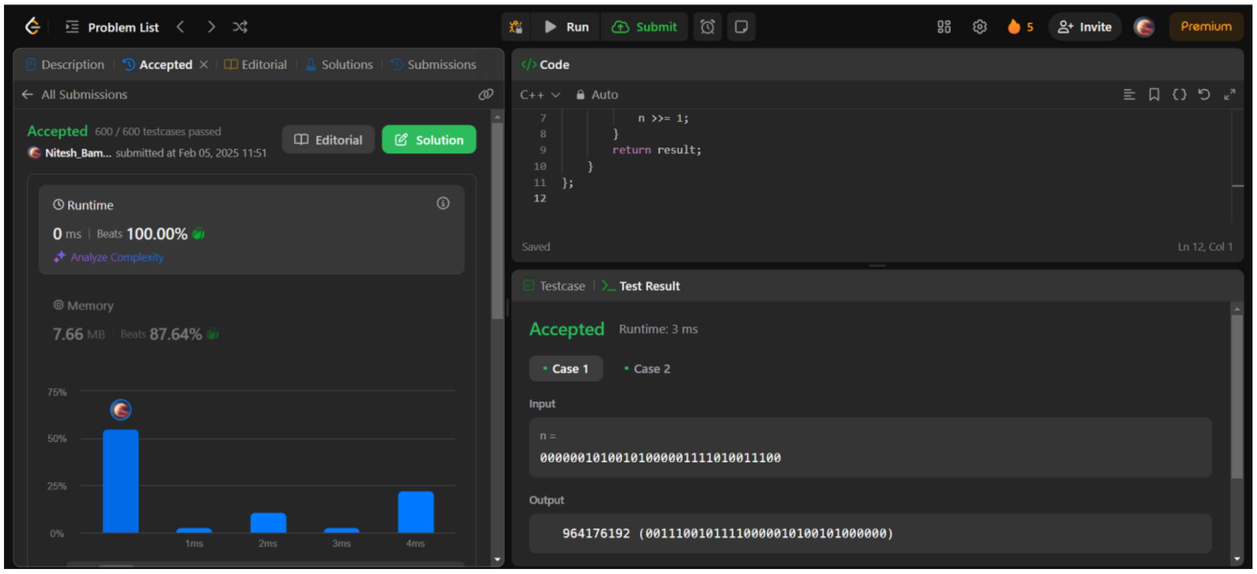
Memory
11.46 MB | Beats 91.73%

```
class Solution {
public:
    void sortColors(vector<int>& nums) {
        int count0 = 0, count1 = 0, count2 = 0;

        for (int num : nums) {
            if (num == 0) count0++;
            else if (num == 1) count1++;
            else count2++;
        }

        int index = 0;
        while (count0--) nums[index++] = 0;
        while (count1--) nums[index++] = 1;
        while (count2--) nums[index++] = 2;
    }
};
```

Problem No. 190 : Reverse Bits

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;
        }
    };
```



Accepted 600 / 600 testcases passed
Nitesh_Bam... submitted at Feb 05, 2025 11:51

Runtime
0 ms | Beats 100.00%
Analyze Complexity

Memory
7.66 MB | Beats 87.64%

```
            n >>= 1;
        }
        return result;
    }
};
```

Saved
Ln 12, Col 1

Accepted Runtime: 3 ms

Input

```
n =
00000010100101000001111010011100
```

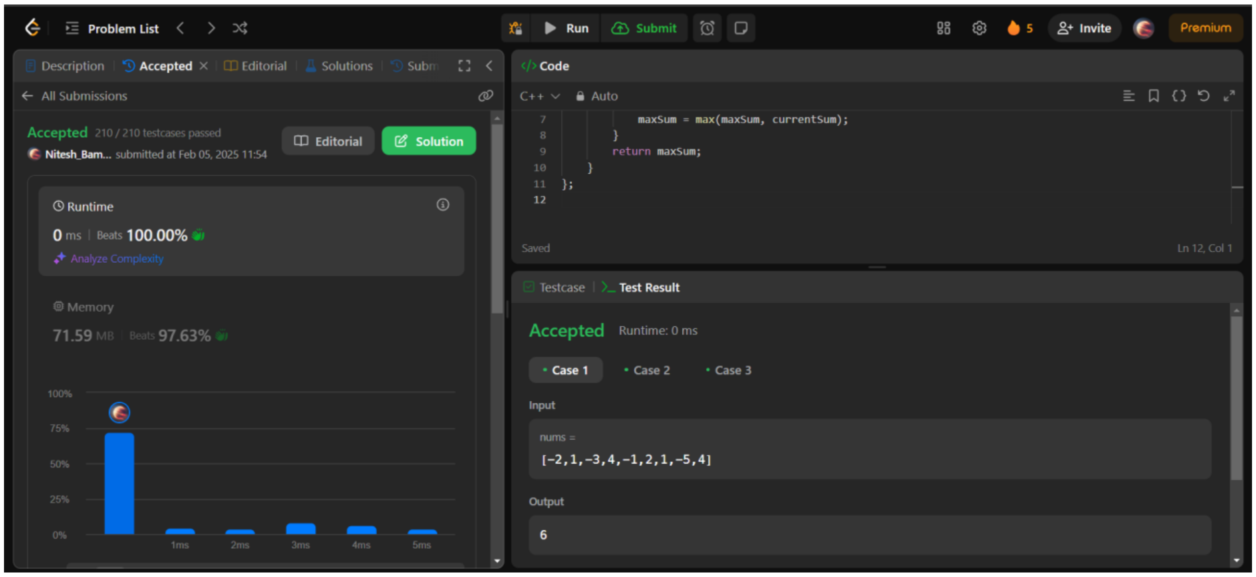
Output

```
964176192 (00111001011110000010100101000000)
```

Problem No. 53 : Maximum Subarray -

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
Nitesh_Bam... submitted at Feb 05, 2025 11:54

Runtime
0 ms | Beats 100.00%

Memory
71.59 MB | Beats 97.63%

```
            maxSum = max(maxSum, currentSum);
        }
        return maxSum;
    }
};
```

Saved
Ln 12, Col 1

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

nums =
[-2,1,-3,4,-1,2,1,-5,4]

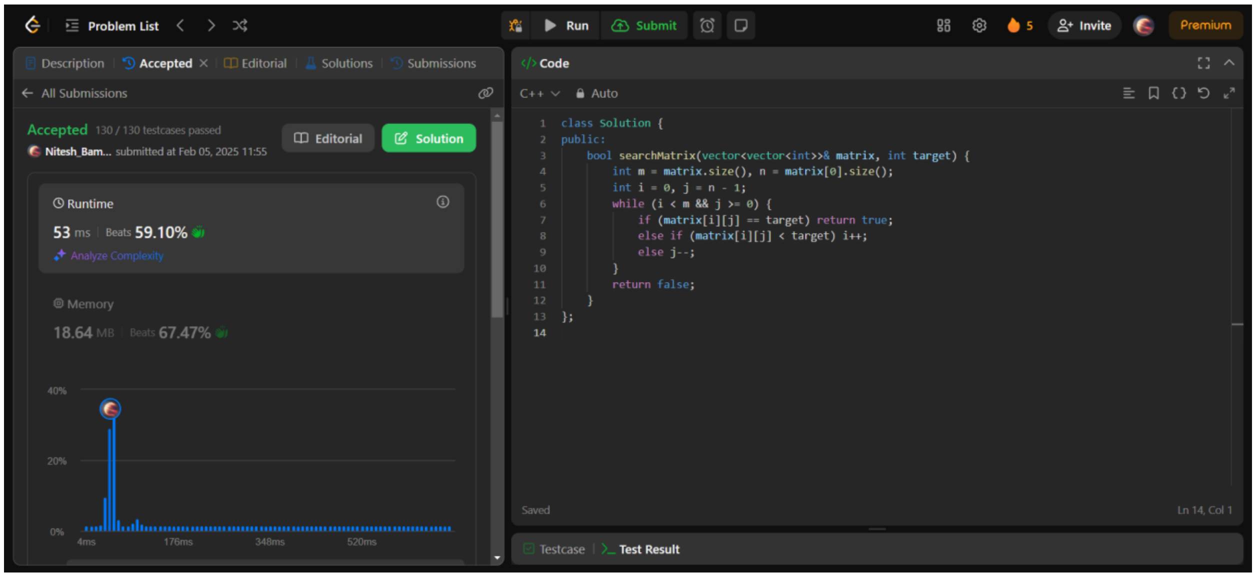
Output

6

Problem No. 240 : Search a 2D Matrix –

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
Nitesh_Bam... submitted at Feb 05, 2025 11:55

Runtime
53 ms | Beats 59.10%

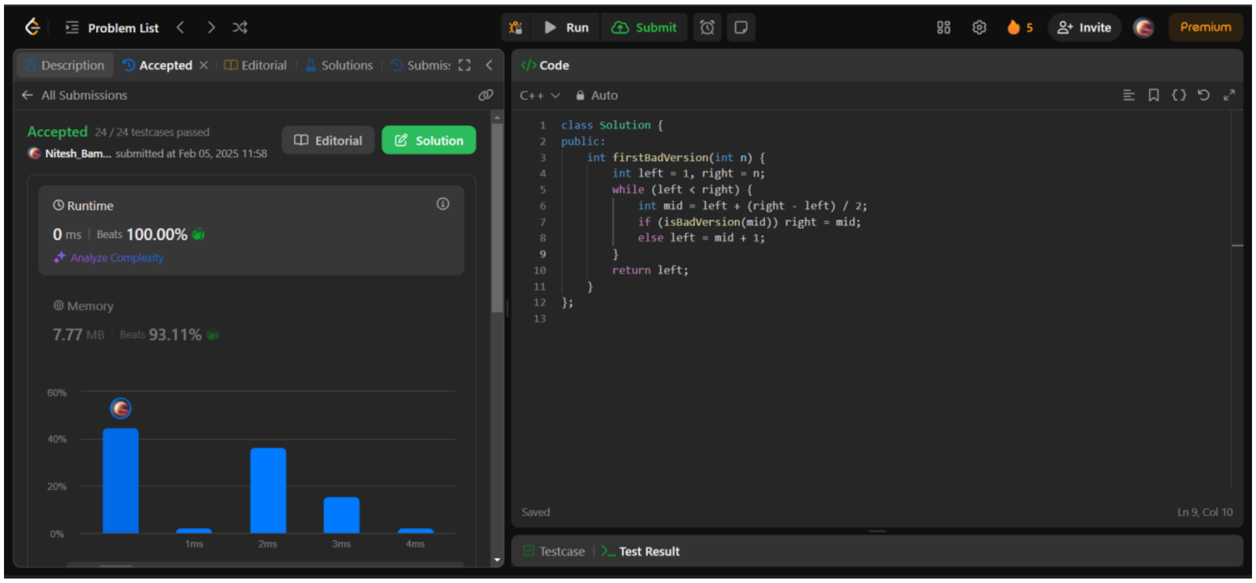
Memory
18.64 MB | Beats 67.47%

```
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;
    }
};
```

Problem No. 278 First Bad Version

Code:

```
class Solution {
public:
    int firstBadVersion(int n) {
        int left = 1, right = n;
        while (left < right) {
            int mid = left + (right - left) / 2;
            if (isBadVersion(mid)) right = mid;
            else left = mid + 1;
        }
        return left;
    }
};
```



Accepted 24 / 24 testcases passed
Nitesh_Bam... submitted at Feb 05, 2025 11:58

Runtime
0 ms | Beats 100.00%

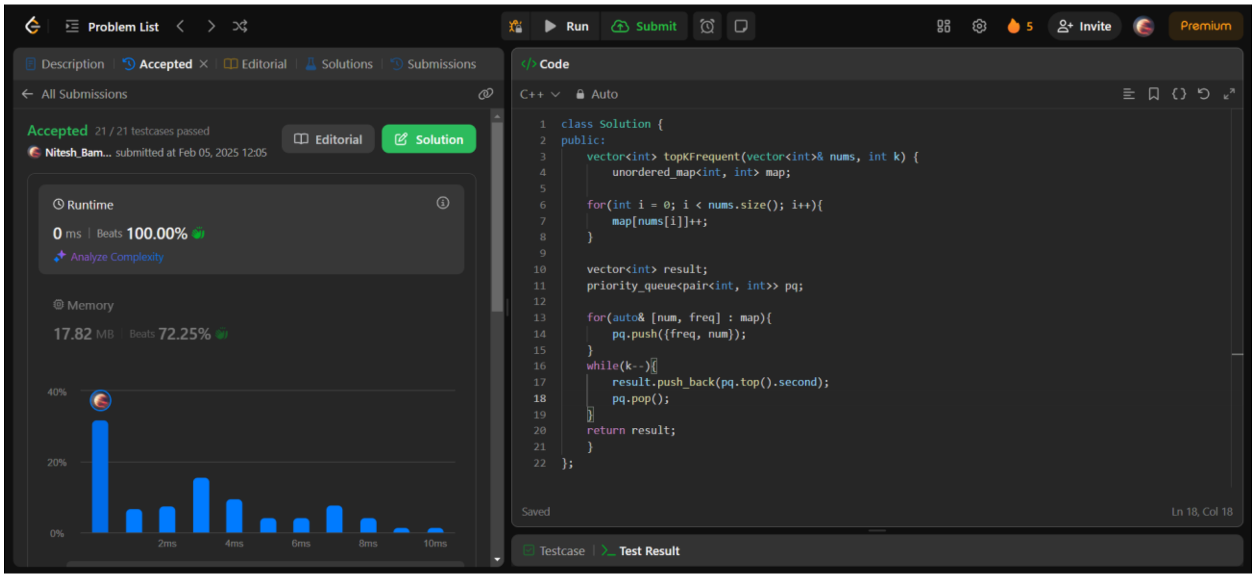
Memory
7.77 MB | Beats 93.11%

```
class Solution {
public:
    int firstBadVersion(int n) {
        int left = 1, right = n;
        while (left < right) {
            int mid = left + (right - left) / 2;
            if (isBadVersion(mid)) right = mid;
            else left = mid + 1;
        }
        return left;
    }
};
```

Prolem No. 347: Top K Frequent Elements –

Code:

```
class Solution {
public:
    vector<int> topKFrequent(vector<int>& nums, int k) {
        unordered_map<int, int> map;

    for(int i = 0; i < nums.size(); i++){
        map[nums[i]]++;
    }
    vector<int> result;
    priority_queue<pair<int, int>> pq;

    for(auto& [num, freq] : map){
        pq.push({freq, num});
    }
    while(k--){
        result.push_back(pq.top().second);
        pq.pop();
    }
    return result;
    }
};
```



Accepted 21 / 21 testcases passed
Nitesh_Bam... submitted at Feb 05, 2025 12:05

Runtime
0 ms | Beats 100.00%
Analyze Complexity

Memory
17.82 MB | Beats 72.25%

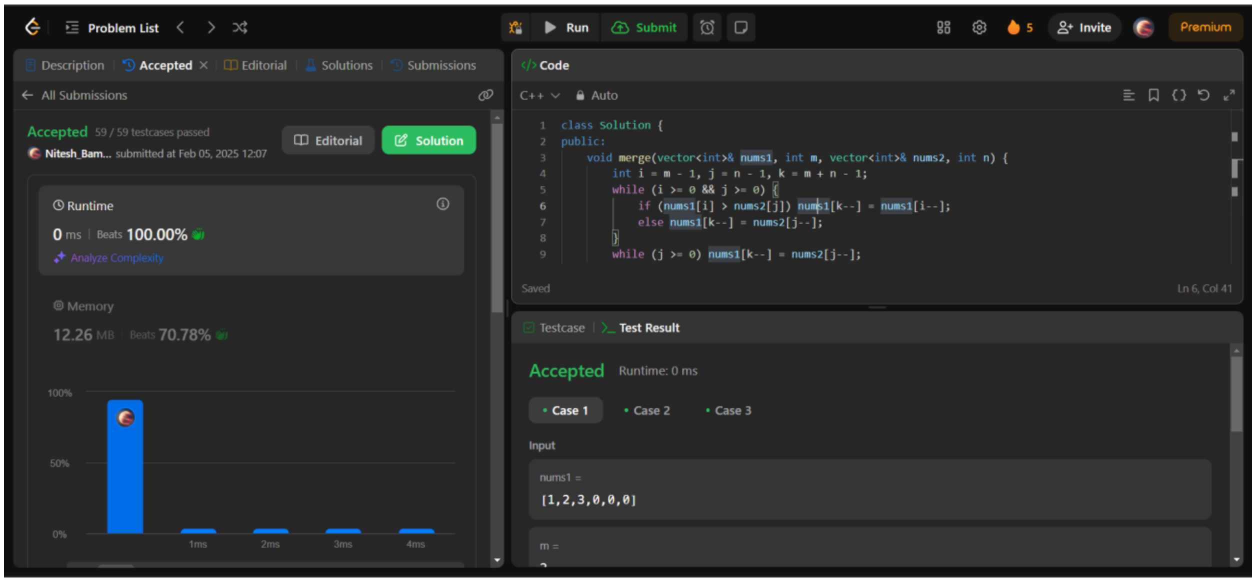
```
class Solution {
public:
    vector<int> topKFrequent(vector<int>& nums, int k) {
        unordered_map<int, int> map;

    for(int i = 0; i < nums.size(); i++){
        map[nums[i]]++;
    }

    vector<int> result;
    priority_queue<pair<int, int>> pq;

    for(auto& [num, freq] : map){
        pq.push({freq, num});
    }
    while(k--){
        result.push_back(pq.top().second);
        pq.pop();
    }
    return result;
    }
};
```

Saved
Ln 18, Col 18

Problem No. 88 : Merge Sorted Arrays

Code:

```
class Solution {
public:
    void merge(vector<int>& nums1, int m, vector<int>& nums2, int n) {
        int i = m - 1, j = n - 1, k = m + n - 1;
        while (i >= 0 && j >= 0) {
            if (nums1[i] > nums2[j]) nums1[k--] = nums1[i--];
            else nums1[k--] = nums2[j--];
        }
        while (j >= 0) nums1[k--] = nums2[j--];
    }
};
```



Accepted 59 / 59 testcases passed
Nitesh_Bam... submitted at Feb 05, 2025 12:07

Runtime
0 ms | Beats 100.00%

Memory
12.26 MB | Beats 70.78%

```
class Solution {
public:
    void merge(vector<int>& nums1, int m, vector<int>& nums2, int n) {
        int i = m - 1, j = n - 1, k = m + n - 1;
        while (i >= 0 && j >= 0) {
            if (nums1[i] > nums2[j]) nums1[k--] = nums1[i--];
            else nums1[k--] = nums2[j--];
        }
        while (j >= 0) nums1[k--] = nums2[j--];
```

Saved
Ln 6, Col 41

Accepted Runtime: 0 ms

Input

nums1 =
[1,2,3,0,0,0]

m =