



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Experiment 4

Student Name: Payal Singh

Branch: BE-IT

Semester: 6

Subject Name: AP LAB-II

UID:22BET10347

Section/Group: IOT-702(A)

Date of Performance:20/02/25

Subject Code: 22ITP-351

PROBLEM 1:

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) {
        int n = s.size();
        int k = -1, mx = 0;
        for (int i = 0; i < n; ++i) {
            unordered_set<char> ss;
            for (int j = i; j < n; ++j) {
                ss.insert(s[j]);
                bool ok = true;
                for (auto& a : ss) {
                    char b = a ^ 32;
                    if (!(ss.count(a) && ss.count(b))) {
                        ok = false;
                        break;
                    }
                }
                if (ok && mx < j - i + 1) {
                    mx = j - i + 1;
                    k = i;
                }
            }
        }
    }
}
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
    }  
    return k == -1 ? "" : s.substr(k, mx);  
}  
};
```

Output:

1763. Longest Nice Substring

Easy Topics Companies Hint

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.

Example 1:

Input: *s* = "YazaAay"
Output: "aAa"
Explanation: "aAa" is a nice string because 'A/a' is the only letter of the alphabet in *s*, and both 'A' and 'a' appear. "aAa" is the longest nice substring.

Example 2:

Input: *s* = "Bb"
Output: "Bb"
Explanation: "Bb" is a nice string because both 'B' and 'b' appear. The whole string is a substring.

Example 3:

Input: *s* = "c"
Output: ""

```
1 class Solution {  
2 public:  
3     string longestNiceSubstring(string s) {  
4         int n = s.size();  
5         int k = -1, mx = 0;  
6         for (int i = 0; i < n; ++i) {  
7             unordered_set<char> ss;  
8             for (int j = i; j < n; ++j) {  
9                 ss.insert(s[j]);  
10                bool ok = true;  
11                for (auto& a : ss) {  
12                    char b = a ^ 32;  
13                    if (!(ss.count(a) && ss.count(b))) {  
14                        ok = false;  
15                        break;  
16                    }  
17                }  
18                if (ok && mx < j - i + 1) {  
19                    mx = j - i + 1;  
20                    k = i;  
21                }  
22            }  
23        }  
24        return k == -1 ? "" : s.substr(k, mx);  
25    }  
26 };
```

Accepted Runtime: 0 ms

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

s =
"YazaAay"

Output

"aAa"

Expected

"aAa"

Contribute a testcase

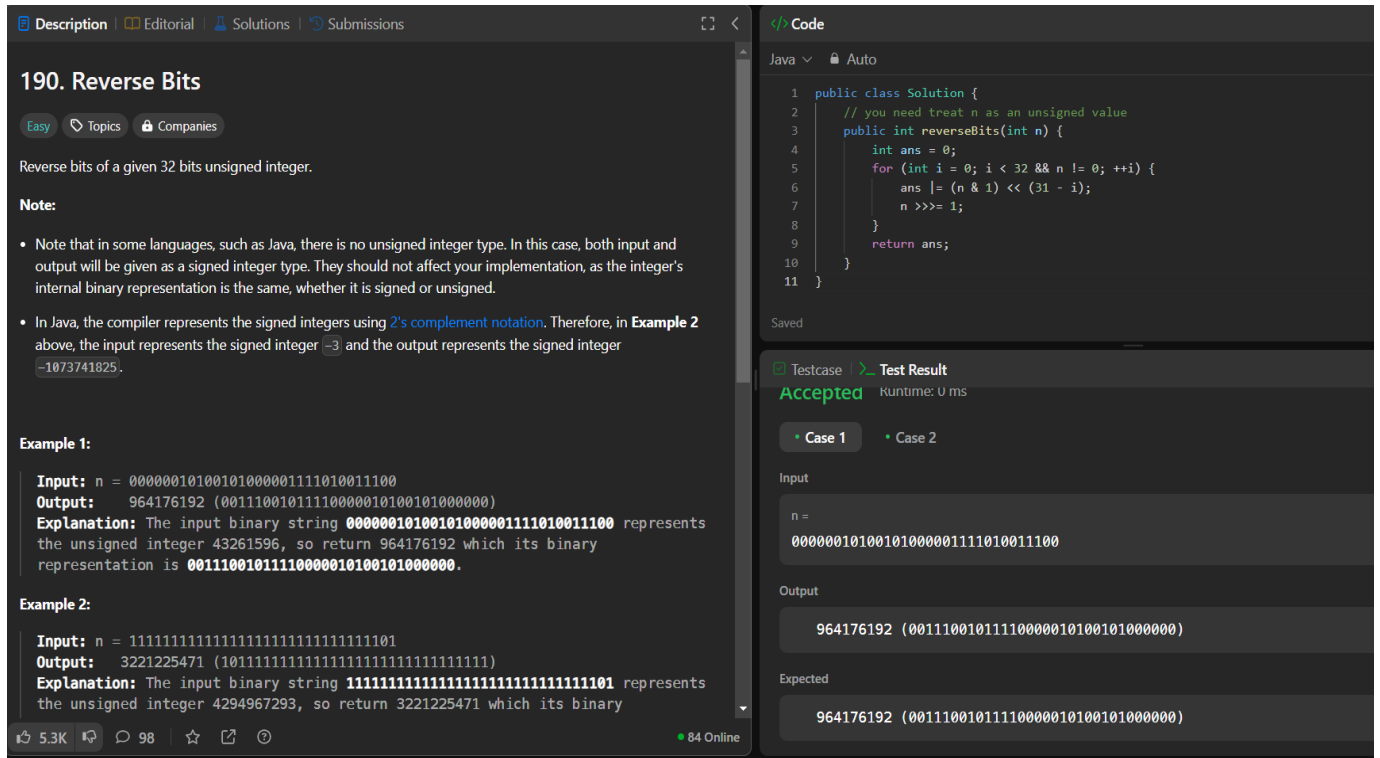
PROBLEM 2:

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

Code:

```
public class Solution {  
    public int reverseBits(int n) {  
        int ans = 0;  
        for (int i = 0; i < 32 && n != 0; ++i) {  
            ans |= (n & 1) << (31 - i);  
            n >>= 1;  
        }  
        return ans;  
    }  
}
```

Output:



The screenshot displays a coding platform interface for a problem titled "190. Reverse Bits". The interface is divided into several sections:

- Description:** Contains the problem statement: "Reverse bits of a given 32 bits unsigned integer." and a note about Java's signed integer type. It also includes two examples with their inputs, outputs, and explanations.
- Code:** Shows the Java code for the solution, which is a class named "Solution" with a method "reverseBits".
- Testcase:** Displays the test results, showing that the solution is "Accepted" with a runtime of 0 ms. It also shows the input and output for the test case.

Example 1:

Input: n = 00000010100101000001111010011100
Output: 964176192 (00111001011110000010100101000000)
Explanation: The input binary string 00000010100101000001111010011100 represents the unsigned integer 43261596, so return 964176192 which its binary representation is 00111001011110000010100101000000.

Example 2:

Input: n = 11111111111111111111111111111111
Output: 3221225471 (10111111111111111111111111111111)
Explanation: The input binary string 11111111111111111111111111111111 represents the unsigned integer 4294967293, so return 3221225471 which its binary representation is 10111111111111111111111111111111.

Testcase: Accepted Runtime: 0 ms

Case 1: Input: n = 00000010100101000001111010011100 Output: 964176192 (00111001011110000010100101000000)

Case 2: Input: n = 11111111111111111111111111111111 Output: 3221225471 (10111111111111111111111111111111)



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

PROBLEM 3:

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:

```
public class Solution {  
    public int hammingWeight(int n) {  
        int ans = 0;  
        while (n != 0) {  
            n &= n - 1;  
            ++ans;  
        }  
        return ans;  
    }  
}
```

Output:

The screenshot displays a coding platform interface for the problem "191. Number of 1 Bits". The left panel contains the problem description, which asks for a function to return the number of set bits in the binary representation of a positive integer n . It includes three examples: Example 1 (Input: $n = 11$, Output: 3), Example 2 (Input: $n = 128$, Output: 1), and Example 3 (Input: $n = 2147483645$). The right panel shows the code editor with the provided Java solution. Below the code editor, the "Test Result" section shows that the solution is "Accepted" with a runtime of 0 ms. The input field shows $n = 11$ and the output field shows 3.

191. Number of 1 Bits

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](#)).

Example 1:

Input: $n = 11$

Output: 3

Explanation:

The input binary string **1011** has a total of three set bits.

Example 2:

Input: $n = 128$

Output: 1

Explanation:

The input binary string **10000000** has a total of one set bit.

Example 3:

Input: $n = 2147483645$

Code:

```
1 public class Solution {  
2     // you need to treat n as an unsigned value  
3     public int hammingWeight(int n) {  
4         int ans = 0;  
5         while (n != 0) {  
6             n &= n - 1;  
7             ++ans;  
8         }  
9         return ans;  
10    }  
11 }
```

Test Result:

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

$n =$

11

Output

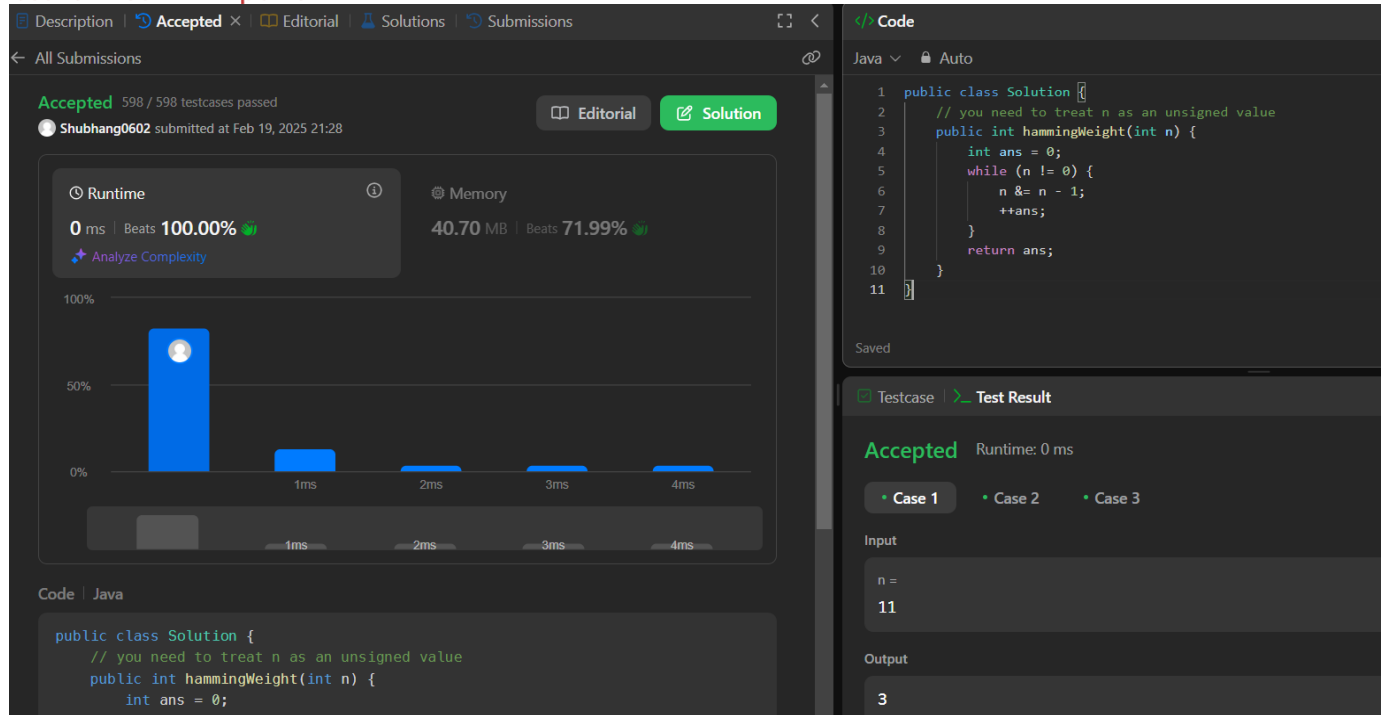
3

Expected



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.



PROBLEM 4:

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

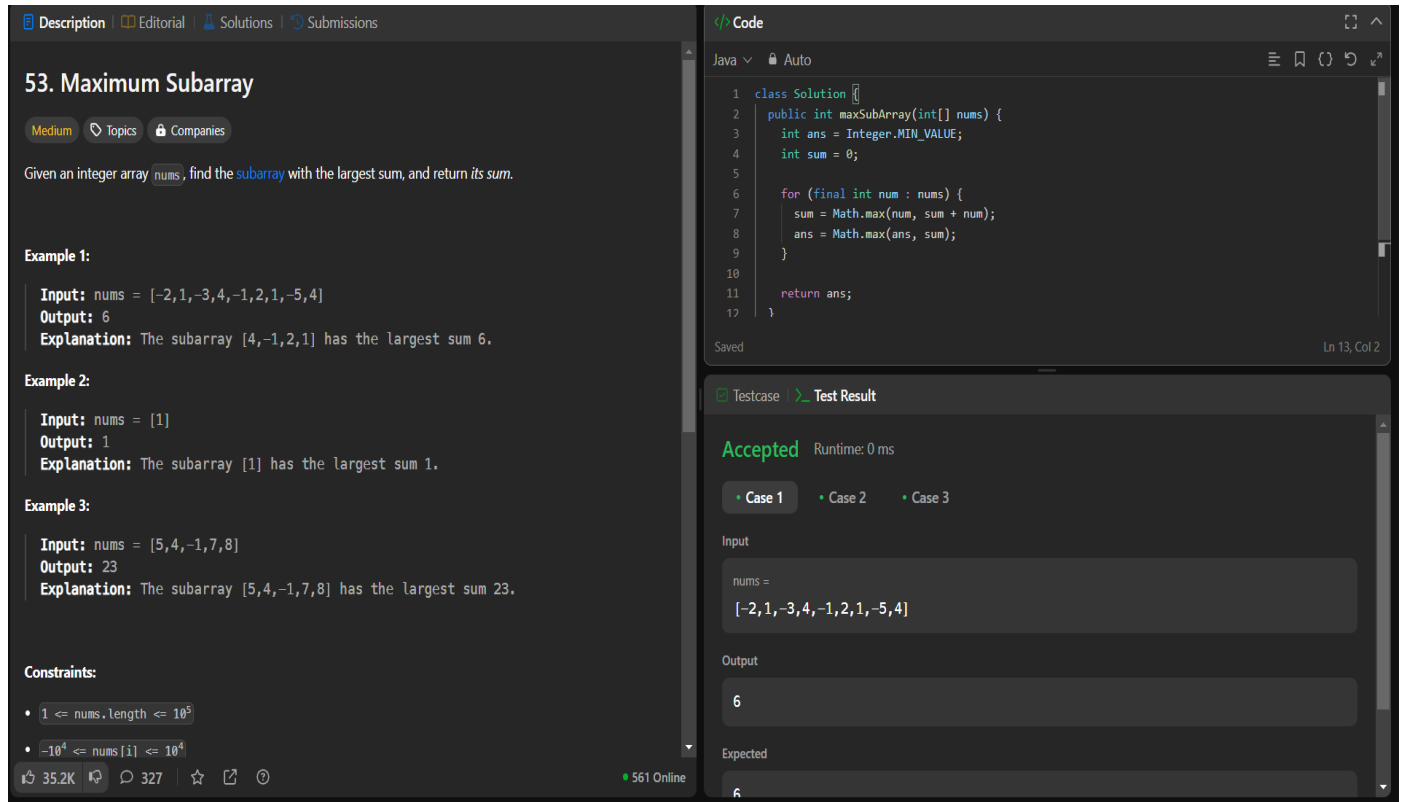
Code:

```
class Solution
{
    public int maxSubArray(int[] nums) {
        int ans = Integer.MIN_VALUE;
        int sum = 0;

        for (final int num : nums) {
            sum = Math.max(num, sum + num);
            ans = Math.max(ans, sum);
        }

        return ans;
    }
}
```

OUTPUT:



53. Maximum Subarray

Medium Topics Companies

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

Example 1:

Input: `nums = [-2,1,-3,4,-1,2,1,-5,4]`
Output: 6
Explanation: The subarray `[4,-1,2,1]` has the largest sum 6.

Example 2:

Input: `nums = [1]`
Output: 1
Explanation: The subarray `[1]` has the largest sum 1.

Example 3:

Input: `nums = [5,4,-1,7,8]`
Output: 23
Explanation: The subarray `[5,4,-1,7,8]` has the largest sum 23.

Constraints:

- $1 \leq \text{nums.length} \leq 10^5$
- $-10^4 \leq \text{nums}[i] \leq 10^4$

35.2K 327 561 Online

```
class Solution {  
    public int maxSubArray(int[] nums) {  
        int ans = Integer.MIN_VALUE;  
        int sum = 0;  
  
        for (final int num : nums) {  
            sum = Math.max(num, sum + num);  
            ans = Math.max(ans, sum);  
        }  
  
        return ans;  
    }  
}
```

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

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

Output

6

Expected

6

PROBLEM 5:

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 boolean searchMatrix(int[][] matrix, int target) {  
        int m = matrix.length, n = matrix[0].length;  
        int left = 0, right = m * n - 1;  
        while (left < right) {  
            int mid = (left + right) >> 1;  
            int x = mid / n, y = mid % n;  
            if (matrix[x][y] >= target) {  
                right = mid;  
            } else {
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
        left = mid + 1;
    }
}
return matrix[left / n][left % n] == target;
}
}
```

Output:

```
</> Code
Java ▾ Auto
1 class Solution {
2     public boolean searchMatrix(int[][] matrix, int target) {
3         int m = matrix.length, n = matrix[0].length;
4         for (int i = m - 1, j = 0; i >= 0 && j < n; ) {
5             if (matrix[i][j] == target) {
6                 return true;
7             }
8             if (matrix[i][j] > target) {
9                 --i;
10            } else {
11                ++j;
12            }
13        }
14        return false;
15    }
16 }
```

Saved Ln 16, Col 2

Testcase | Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

matrix =
[[1,4,7,11,15], [2,5,8,12,19], [3,6,9,16,22], [10,13,14,17,24], [18,21,23,26,30]]

target =

Description | Accepted | Editorial | Solutions | Submissions

All Submissions

Accepted 130 / 130 testcases passed

Shubhang0602 submitted at Feb 19, 2025 21:43

Editorial Solution

Runtime 5 ms | Beats 99.72% | Memory 46.33 MB | Beats 10.93%

Analyze Complexity

Code | Java

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

```
</> Code
Java ▾ Auto
1 class Solution {
2     public boolean searchMatrix(int[][] matrix, int target) {
3         int m = matrix.length, n = matrix[0].length;
4         for (int i = m - 1, j = 0; i >= 0 && j < n; ) {
5             if (matrix[i][j] == target) {
6                 return true;
7             }
8             if (matrix[i][j] > target) {
9                 --i;
10            } else {
11                ++j;
12            }
13        }
14        return false;
15    }
16 }
```

Saved Ln 16, Col 2

Testcase | Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

matrix =
[[1,4,7,11,15], [2,5,8,12,19], [3,6,9,16,22], [10,13,14,17,24], [18,21,23,26,30]]

target =
5

Output
true

Expected
true

Contribute a testcase



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

PROBLEM 6:

Aim: Your task is to calculate $ab \bmod 1337$ 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 int superPow(int a, int[] b) {
        int ans = 1;

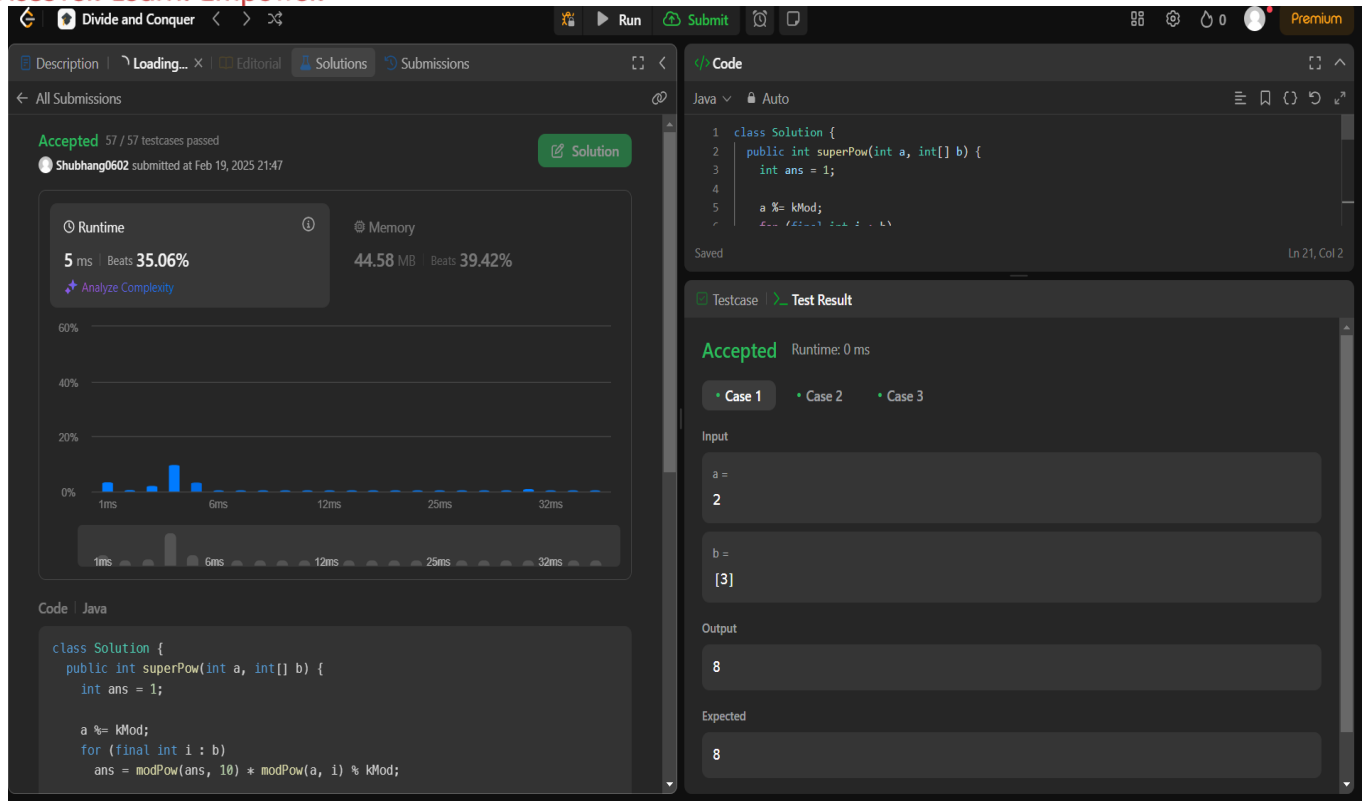
        a %= kMod;
        for (final int i : b)
            ans = modPow(ans, 10) * modPow(a, i) % kMod;

        return ans;
    }

    private static final int kMod = 1337;

    private int modPow(int x, int n) {
        if (n == 0)
            return 1;
        if (n % 2 == 1)
            return x * modPow(x % kMod, (n - 1)) % kMod;
        return modPow(x * x % kMod, (n / 2)) % kMod;
    }
}
```

Output:



PROBLEM 7:

Aim: An array `nums` of length `n` is beautiful if `nums` is a permutation of the integers in the range `[1, n]`.

For every $0 \leq i < j < n$, there is no index `k` with $i < k < j$ where $2 * \text{nums}[k] == \text{nums}[i] + \text{nums}[j]$.

Given the integer `n`, return any beautiful array `nums` of length `n`. There will be at least one valid answer for the given `n`.

Code:

```
class Solution {
    public int[] beautifulArray(int n) {
        int[] arr = new int[n];
        for (int i = 0; i < n; ++i)
            arr[i] = i + 1;
        divide(arr, 0, n - 1, 1);
        return arr;
    }

    private void divide(int[] arr, int l, int r, int mask) {
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
if (l >= r)
    return;
final int m = partition(arr, l, r, mask);
divide(arr, l, m, mask << 1);
divide(arr, m + 1, r, mask << 1);
}

private int partition(int[] arr, int l, int r, int mask) {
    int nextSwapped = l;
    for (int i = l; i <= r; ++i)
        if ((arr[i] & mask) > 0)
            swap(arr, i, nextSwapped++);
    return nextSwapped - 1;
}

private void swap(int[] arr, int i, int j) {
    final int temp = arr[i];
    arr[i] = arr[j];
    arr[j] = temp;
}
}
```

Output:

The screenshot displays a coding platform interface for a problem titled "Divide and Conquer". The solution is in Java and has been accepted. The runtime is 0 ms, and the memory usage is 41.61 MB, which beats 98.51% of other solutions. The code defines a class `Solution` with a method `beautifulArray` that takes an integer `n` and returns an array of integers. The method uses a recursive divide-and-conquer approach to generate the array. The test case shows an input of `n = 4` and an expected output of `[2, 1, 4, 3]`, while the actual output is `[3, 1, 2, 4]`.

```
class Solution {
    public int[] beautifulArray(int n) {
        int[] arr = new int[n];
        for (int i = 0; i < n; ++i)
            arr[i] = i + 1;
        divide(arr, 0, n - 1, 1);
        return arr;
    }
}
```

Runtime: 0 ms | Beats 100.00%
Memory: 41.61 MB | Beats 98.51%

Accepted Runtime: 0 ms

Case 1: Input: n = 4, Output: [3, 1, 2, 4], Expected: [2, 1, 4, 3]



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.