



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Experiment 4

Student Name: Shubhang Deep Singh

Branch: BE-IT

Semester: 6

Subject Name: AP LAB-II

UID:22BET10325

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.length();
        int k = -1;
        int mx = 0;
        for (int i = 0; i < n; ++i) {
            Set<Character> ss = new HashSet<>();
            for (int j = i; j < n; ++j) {
                ss.add(s.charAt(j));
                boolean ok = true;
                for (char a : ss) {
                    char b = (char) (a ^ 32);
                    if (!(ss.contains(a) && ss.contains(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.substring(k, k + mx);  
}  
}
```

Output:

Accepted 73 / 73 testcases passed
Shubhang0602 submitted at Feb 19, 2025 21:55

Runtime: 24 ms Beats 24.95%
Memory: 44.89 MB Beats 28.74%

Code | Java

```
class Solution {  
    public String longestNiceSubstring(String s) {  
        int n = s.length();  
        int k = -1;  
        int mx = 0;  
        for (int i = 0; i < n; ++i) {  
            Set<Character> ss = new HashSet<>();  
            for (int j = i; j < n; ++j) {  
                ss.add(s.charAt(j));  
                boolean ok = true;  
                for (char a : ss) {  
                    char b = (char) (a ^ 32);  
                    if (!ss.contains(a) && ss.contains(b)) {  
                        ok = false;  
                        break;  
                    }  
                }  
                if (ok && mx < j - i + 1) {  
                    mx = j - i + 1;  
                    k = i;  
                }  
            }  
        }  
        return k == -1 ? "" : s.substring(k, k + mx);  
    }  
}
```

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, "aABB" is nice because 'A' and 'a' appear, and 'B' and 'b' appear. However, "abA" is not nice 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: $s = \text{"YazaAay"}$
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: $s = \text{"Bb"}$
Output: "Bb"
Explanation: "Bb" is a nice string because both 'B' and 'b' appear. The whole string is a substring.

Example 3:
Input: $s = \text{"c"}$
Output: ""

Code | Java

```
class Solution {  
    public String longestNiceSubstring(String s) {  
        int n = s.length();  
        int k = -1;  
        int mx = 0;  
        for (int i = 0; i < n; ++i) {  
            Set<Character> ss = new HashSet<>();  
            for (int j = i; j < n; ++j) {  
                ss.add(s.charAt(j));  
            }  
        }  
    }  
}
```

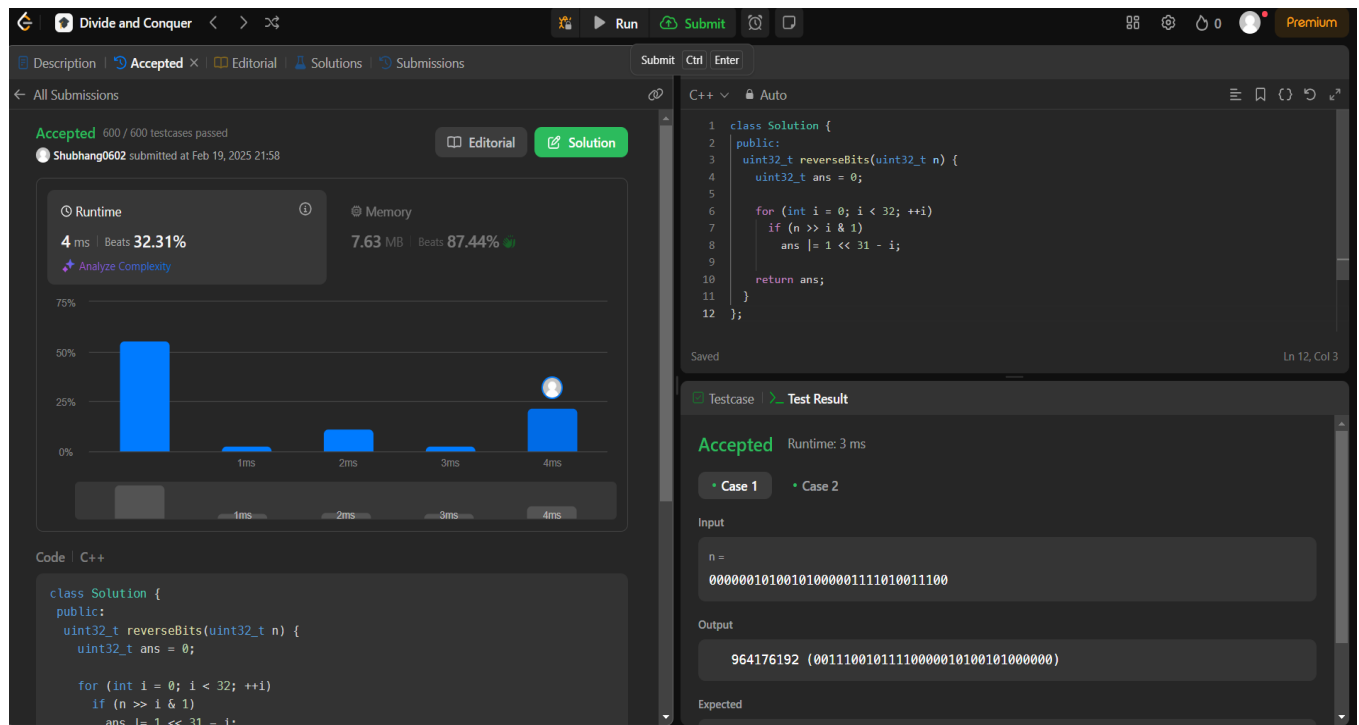
PROBLEM 2:

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

Code:

```
class Solution {  
public:  
    uint32_t reverseBits(uint32_t n) {  
        uint32_t ans = 0;  
  
        for (int i = 0; i < 32; ++i)  
            if (n >> i & 1)  
                ans |= 1 << 31 - i;  
  
        return ans;  
    }  
};
```

Output:



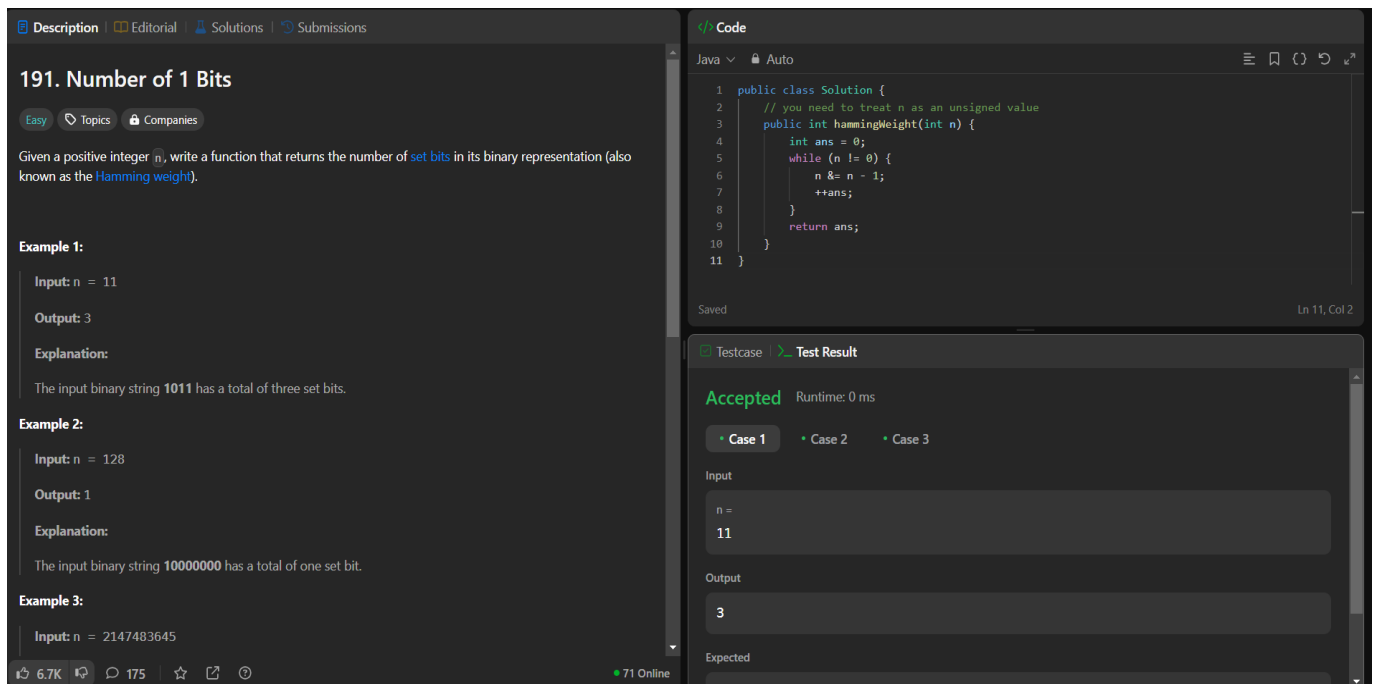
PROBLEM 3:

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. On the left, the problem description for '191. Number of 1 Bits' is shown, including examples and a difficulty level of 'Easy'. The main area on the right shows the Java code for the solution, which uses a while loop to count the number of set bits. Below the code, the 'Test Result' section shows that the solution is 'Accepted' with a runtime of 0 ms. The test case details show an input of n = 11, resulting in an output of 3.

191. Number of 1 Bits
Easy Topics Companies
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
Java Auto
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

PROBLEM 4:

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

Code:

```
class Solution  
{
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
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:

The screenshot displays a coding interface for the 'Maximum Subarray' problem (LeetCode 53). The left pane shows the problem description, which asks to find the subarray with the largest sum in a given integer array. It includes three examples: Example 1 with input [-2,1,-3,4,-1,2,1,-5,4] and output 6; Example 2 with input [1] and output 1; and Example 3 with input [5,4,-1,7,8] and output 23. Constraints specify that the array length is up to 10^5 and elements range from -10^4 to 10^4. The right pane shows the Java code for the solution, which implements Kadane's algorithm. Below the code, the 'Testcase' section shows 'Case 1' as 'Accepted' with a runtime of 0 ms. The input field contains the array [-2,1,-3,4,-1,2,1,-5,4], the output field shows 6, and the expected result is also 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:



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

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

Output:

The screenshot shows a code editor with the following Java code:

```
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 }
```

Below the code editor, the test results are shown:

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



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

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%

Analyze Complexity

Memory

46.33 MB | Beats 10.93%

Runtime (ms)	Percentage
4ms	~2%
5ms	~60%
6ms	~25%
7ms	~5%
8ms	~2%
9ms	~2%
10ms	~5%

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; ) {
```

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:



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

The screenshot displays a LeetCode submission for the 'Divide and Conquer' problem. The solution is implemented in Java, showing a class 'Solution' with a method 'superPow' that calculates $a^b \mod 10$ using a recursive divide-and-conquer approach. The test results indicate that the solution is accepted for all test cases, with a runtime of 5 ms and memory usage of 44.58 MB.

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 the LeetCode submission page for the 'Divide and Conquer' problem. The top navigation bar includes 'Description', 'Accepted' (38 / 38 testcases passed), 'Editorial', 'Solutions', and 'Submissions'. The submission status is 'Accepted' by 'Shubhang0602' on Feb 19, 2025 at 21:50. The runtime performance is shown as a bar chart with a peak at 0ms, beating 100.00% of other submissions. The memory usage is 41.61 MB, beating 98.51%. The code is written in Java, and the test result is 'Accepted' with a runtime of 0ms. The input is 'n = 4' and the output is '[3,1,2,4]'. The expected output is '[2,1,4,3]'. A 'Contribute a testcase' button is visible at the bottom right.

```
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;
    }
}
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.