

## Longest Nice Substring

```
public class Solution {
    public String longestNiceSubstring(String s) {
        int n = s.length(), maxLen = 0, startIdx = -1;
        for (int i = 0; i < n; i++) {
            for (int j = i; j < n; j++) {
                if (isNice(s.substring(i, j + 1)) && (j - i + 1) > maxLen) {
                    maxLen = j - i + 1;
                    startIdx = i;
                }
            }
        }
        return startIdx == -1 ? "" : s.substring(startIdx, startIdx + maxLen);
    }

    private boolean isNice(String s) {
        int[] lower = new int[26], upper = new int[26];
        for (char c : s.toCharArray()) {
            if (Character.isLowerCase(c)) lower[c - 'a']++;
            else upper[c - 'A']++;
        }
        for (int i = 0; i < 26; i++) {
            if ((lower[i] > 0 && upper[i] == 0) || (upper[i] > 0 && lower[i] == 0)) return false;
        }
        return true;
    }
}
```

Accepted 73 / 73 testcases passed  
Shiv Chauhan submitted at Feb 24, 2025 21:29

Runtime: 31 ms | Beats 17.30%  
Memory: 45.44 MB | Beats 6.37%

Code | Java

```
public class Solution {
    public String longestNiceSubstring(String s) {
        int n = s.length(), maxLen = 0, startIdx = -1;
        for (int i = 0; i < n; i++) {
            for (int j = i; j < n; j++) {
                if (isNice(s.substring(i, j + 1)) && (j - i + 1) > maxLen) {
                    maxLen = j - i + 1;
                    startIdx = i;
                }
            }
        }
        return startIdx == -1 ? "" : s.substring(startIdx, startIdx + maxLen);
    }

    private boolean isNice(String s) {
        int[] lower = new int[26], upper = new int[26];
        for (char c : s.toCharArray()) {
            if (Character.isLowerCase(c)) lower[c - 'a']++;
            else upper[c - 'A']++;
        }
        for (int i = 0; i < 26; i++) {
            if ((lower[i] > 0 && upper[i] == 0) || (upper[i] > 0 && lower[i] == 0)) return false;
        }
        return true;
    }
}
```

Testcase: Case 1: "YazaAay"

## Reverse Bits

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

The screenshot displays the LeetCode submission interface for the "Reverse Bits" problem. The submission status is "Accepted" with 600/600 testcases passed. The user, Shiv Chauhan, submitted the solution on Feb 24, 2025, at 21:31. The performance metrics show a runtime of 0ms (Beats 100.00%) and memory usage of 41.71 MB (Beats 73.57%). A bar chart indicates the runtime performance relative to other submissions. The code is written in Java and implements a loop to reverse the bits of an integer n.

**Runtime:** 0ms | Beats 100.00% | Memory: 41.71 MB | Beats 73.57%

**Code:**

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

**Testcase:**

Case 1: n = 00000010100101000001111010011100

## Number of 1 Bits

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

The screenshot shows a LeetCode submission interface for the "Number of 1 Bits" problem. The left sidebar displays the problem status as "Accepted" with 598/598 testcases passed, submitted by Shiv Chauhan on Feb 24, 2025. The runtime is 0 ms (Beats 100.00%) and memory is 41.18 MB (Beats 13.49%). A bar chart shows the user's performance compared to others. The main area shows the Java code for the solution, which is a while loop that counts the number of 1 bits in the integer n. The right sidebar shows the test case input n = 11.

Accepted 598 / 598 testcases passed  
Shiv Chauhan submitted at Feb 24, 2025 21:33

Runtime: 0 ms, Beats 100.00%  
Memory: 41.18 MB, Beats 13.49%

Code | Java

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

Testcase

Case 1 Case 2 Case 3 +

n =

11

## Maximum Subarray

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

The screenshot shows the LeetCode submission interface for the "Maximum Subarray" problem. The submission status is "Accepted" with 210/210 testcases passed. The user "Shiv Chauhan" submitted the solution on Feb 24, 2025 at 21:34. The performance metrics are: Runtime 1 ms (Beats 99.52%) and Memory 57.10 MB (Beats 42.51%). A bar chart shows the runtime performance relative to other submissions. The code editor on the right displays the Java solution:

```
1 class Solution {
2     public int maxSubArray(int[] nums) {
3         int sum = 0;
4         int max = Integer.MIN_VALUE;
5         for (int num : nums) {
6             sum += num;
7             max = Math.max(max, sum);
8             if (sum < 0)
9                 sum = 0;
10        }
11        return max;
12    }
13 }
```

The test case input is: `nums = [-2,1,-3,4,-1,2,1,-5,4]`.

## Search a 2D Matrix II

```
class Solution {
    public boolean searchMatrix(int[][] matrix, int target) {
        int n = matrix.length;
        int m = matrix[0].length;
        int row = 0, col = m - 1;

        while (row < n && col >= 0){
            if (matrix[row][col] == target) return true;
            else if (matrix[row][col] < target) row++;
            else col--;
        }
        return false;
    }
}
```

**Accepted** 130 / 130 testcases passed  
Shiv Chauhan submitted at Feb 24, 2025 21:37

**Runtime** 5 ms | Beats 99.70%  
**Memory** 45.66 MB | Beats 93.46%

Code | Java

```
class Solution {
    public boolean searchMatrix(int[][] matrix, int target) {
        int n = matrix.length;
        int m = matrix[0].length;
        int row = 0, col = m - 1;

        while (row < n && col >= 0){
            if (matrix[row][col] == target) return true;
            else if (matrix[row][col] < target) row++;
            else col--;
        }
        return false;
    }
}
```

Testcase

Case 1 Case 2 +

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]]
```

## Super Pow

```
public class Solution {
    private static final int MOD = 1337;

    public int superPow(int a, int[] b) {
        a %= MOD;
        int result = 1;
        for (int digit : b) {
            result = (powMod(result, 10, MOD) * powMod(a, digit, MOD)) % MOD;
        }
        return result;
    }

    private int powMod(int base, int exp, int mod) {
        int result = 1;
        while (exp > 0) {
            if ((exp & 1) == 1) result = (result * base) % mod;
            base = (base * base) % mod;
            exp >>= 1;
        }
        return result;
    }
}
```

The screenshot displays a code editor interface for a problem named 'Super Pow'. The left sidebar shows the 'Problem List' with the current problem selected. The main editor area is divided into two panes. The left pane shows the 'Accepted' status with 57/57 testcases passed, submitted by Shiv Chauhan on Feb 24, 2025 at 21:38. It includes a 'Runtime' section showing 4 ms (Beats 73.72%) and a 'Memory' section showing 44.81 MB (Beats 15.88%). Below this is a bar chart showing the distribution of runtime times. The right pane shows the 'Code' editor with the Java solution. The code defines a class 'Solution' with a static final int MOD = 1337. It has a public method 'superPow' that takes an integer 'a' and an integer array 'b', and a private method 'powMod' that calculates the power of a base to an exponent modulo a given modulus. The 'Testcase' section at the bottom shows a single test case with 'a = 2'.

Accepted 57 / 57 testcases passed  
Shiv Chauhan submitted at Feb 24, 2025 21:38

Runtime: 4 ms, Beats 73.72%  
Memory: 44.81 MB, Beats 15.88%

Code | Java

```
public class Solution {
    private static final int MOD = 1337;

    public int superPow(int a, int[] b) {
        a %= MOD;
        int result = 1;
        for (int digit : b) {
            result = (powMod(result, 10, MOD) * powMod(a, digit, MOD)) % MOD;
        }
    }

    private int powMod(int base, int exp, int mod) {
        int result = 1;
        while (exp > 0) {
            if ((exp & 1) == 1) result = (result * base) % mod;
            base = (base * base) % mod;
            exp >>= 1;
        }
        return result;
    }
}
```

Testcase

Case 1: a = 2