**Advanced Programming 2**

**Assignment 4**

1. **1763.Longest Nice Substring**

**Code:**

class Solution

{

public:

string longestNiceSubstring(string s) {

        if (s.size() < 2) return "";

        unordered\_set<char> st(begin(s), end(s));

        for (int i = 0; i < s.size(); i++) {

             if (st.find((char) toupper(s[i])) == end(st) || st.find((char) tolower(s[i])) == end(st)) {

                string s1 = longestNiceSubstring(s.substr(0, i));

                string s2 = longestNiceSubstring(s.substr(i + 1));

                return s1.size() >= s2.size() ? s1 : s2;

            }

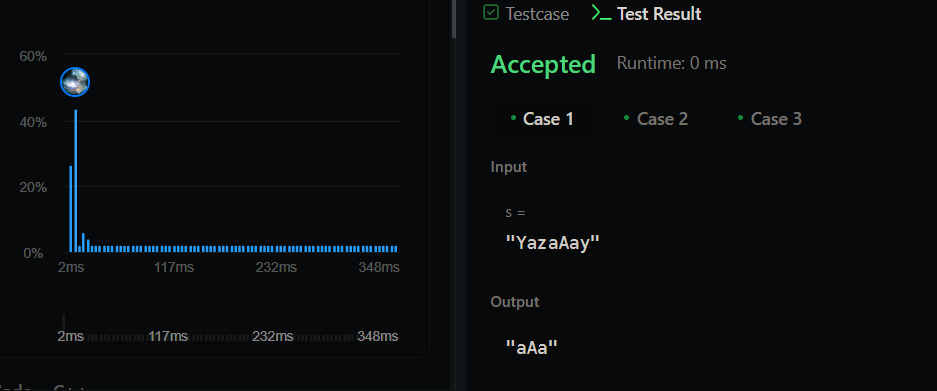
        }

        return s;

    }

};

**Output:**

****

1. **190.Reverse Bits**

**Code:**

class Solution {

public:

    uint32\_t reverseBits(uint32\_t n) {

        n = ((n & 0xffff0000) >> 16) | ((n & 0x0000ffff) << 16);

        n = ((n & 0xff00ff00) >> 8) | ((n & 0x00ff00ff) << 8);

        n = ((n & 0xf0f0f0f0) >> 4) | ((n & 0x0f0f0f0f) << 4);

        n = ((n & 0xcccccccc) >> 2) | ((n & 0x33333333) << 2);

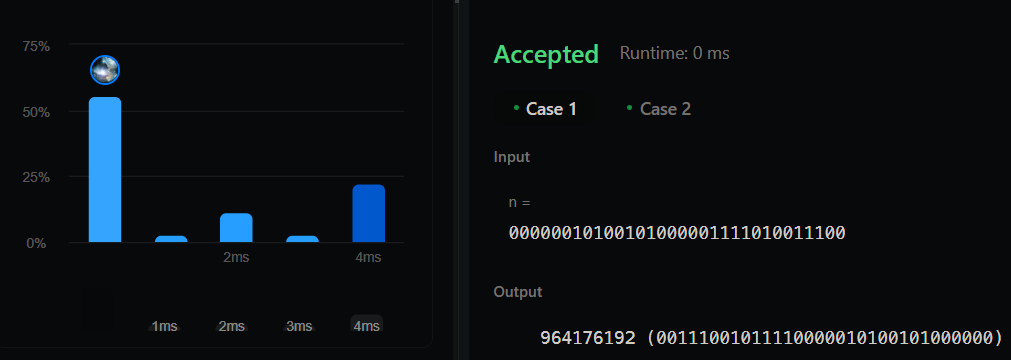
        n = ((n & 0xaaaaaaaa) >> 1) | ((n & 0x55555555) << 1);

        return n;

    }

};

**Output:**

****

1. **191.Number of 1 Bits**

**Code:**

class Solution {

public:

    int hammingWeight(int n) {

      int count = 0;

        for(int i = 31; i >= 0; i--){

            if(((n >> i) & 1) == 1)

                count++;

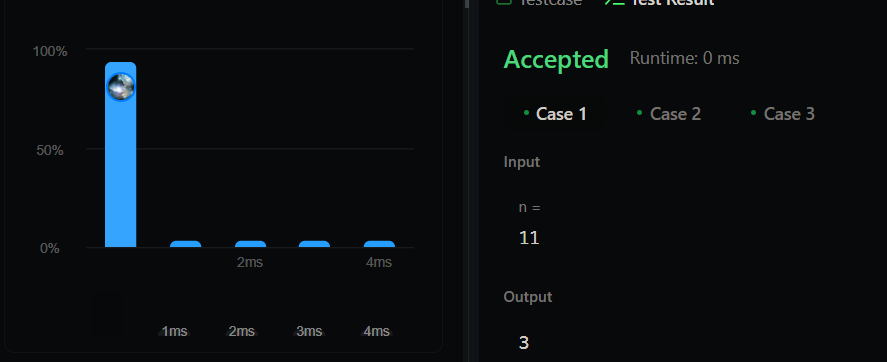
        }

        return count;

    }

};

**Output:**

****

1. **53.Maximum Subarray**

**Code:**

class Solution {

public:

    int maxSubArray(vector<int>& nums) {

        int res = nums[0];

        int total = 0;

        for (int n : nums) {

            if (total < 0) {

                total = 0;

            }

            total += n;

            res = max(res, total);

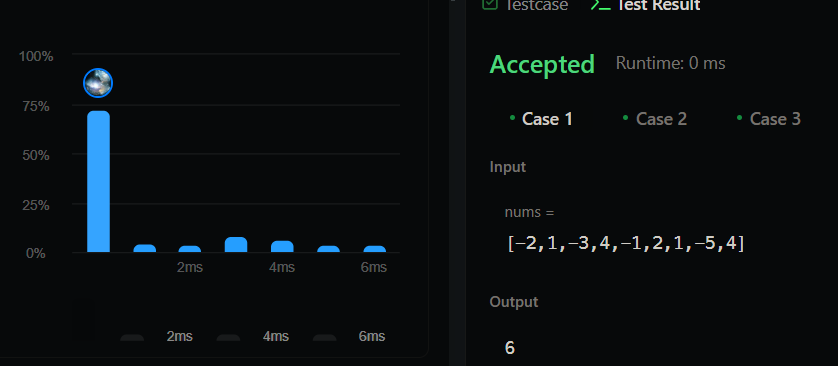
        }

        return res;

    }

};

**Output:**

****

1. **240.Search a 2D Matrix II**

**Code:**

class Solution {

public:

    bool searchMatrix(vector<vector<int>>& matrix, int target) {

        int cols = matrix[0].size() - 1;

        int n = matrix.size() - 1;

        int rows = 0;

        while(rows <= n && cols >= 0){

            int toCompare = matrix[rows][cols];

            if(toCompare > target){

                cols--;

            }else if(toCompare < target){

                rows++;

            }else{

                return true;

            }

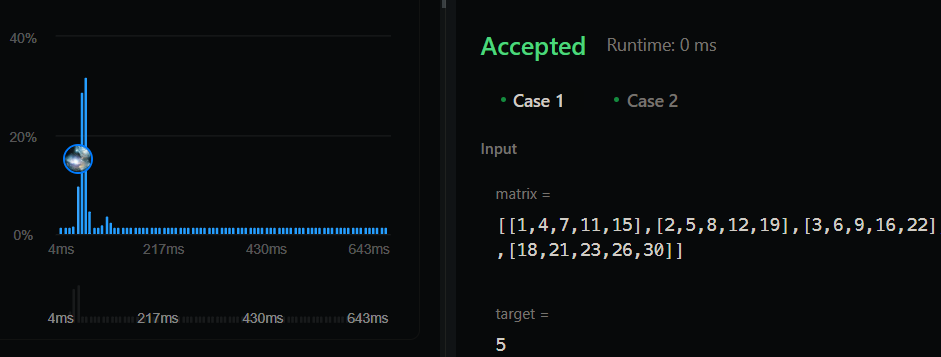
        }

        return false;

    }

};

**Output:**

****

1. **372.Super Pow**

**Code:**

class Solution {

    const int base = 1337;

    int powmod(int a, int k)

    {

        a %= base;

        int result = 1;

        for (int i = 0; i < k; ++i)

            result = (result \* a) % base;

        return result;

    }

public:

    int superPow(int a, vector<int>& b) {

        if (b.empty()) return 1;

        int last\_digit = b.back();

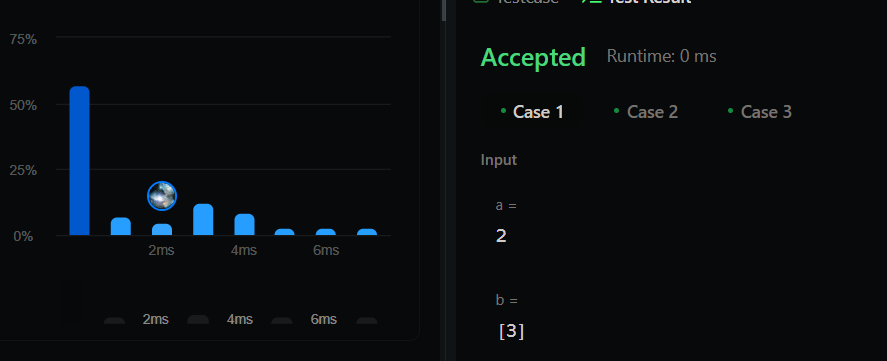
        b.pop\_back();

        return powmod(superPow(a, b), 10) \* powmod(a, last\_digit) % base;

    }

};

**Output:**

****