1763. Longest Nice Substring

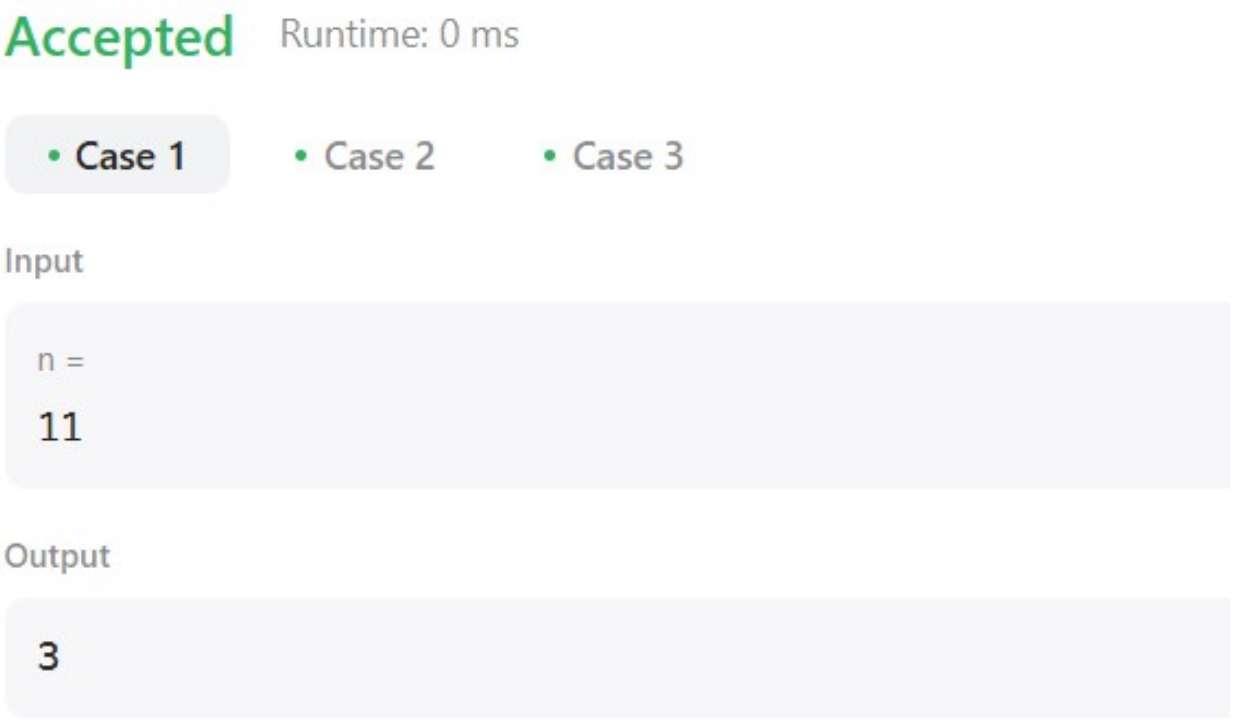
```
string longestNiceSubstring(string s) {
        int n = s.size();
        if (n < 2) return "";

        unordered_set<char> st(s.begin(), s.end());
        for (int i = 0; i < n; ++i) {
            if (st.count(tolower(s[i])) &&
st.count(toupper(s[i]))) {
                continue;
            }
            string left = longestNiceSubstring(s.substr(0,
i));
            string right = longestNiceSubstring(s.substr(i +
1));
            return left.size() >= right.size() ? left :
right;
        }

        return s;
    }
```

190. Reverse Bits

```
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;
    }
```

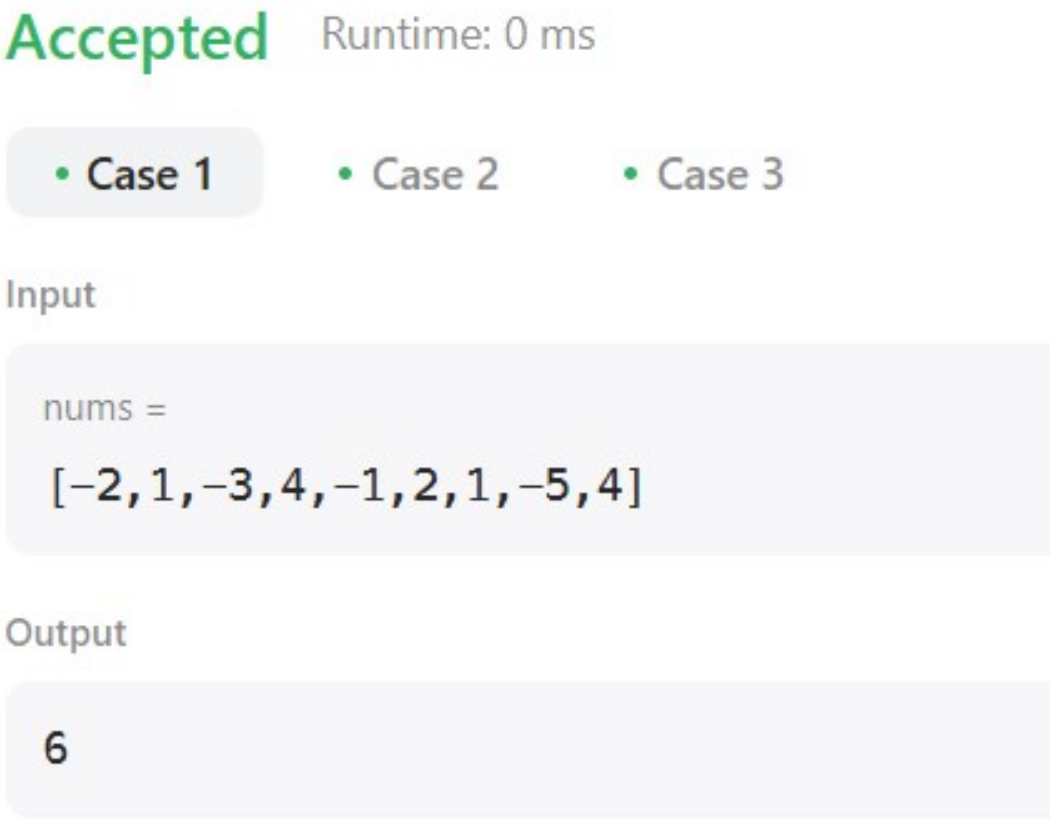
191. Number of 1 Bits

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



53. Maximum Subarray

```
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;
    }
```



240. Search a 2D Matrix II

```
bool searchMatrix(vector<vector<int>>& matrix, int
target) {
        int rows = matrix.size(), cols =
matrix[0].size();
        int row = 0, col = cols - 1;

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