

ASSIGNMENT-4(AP)

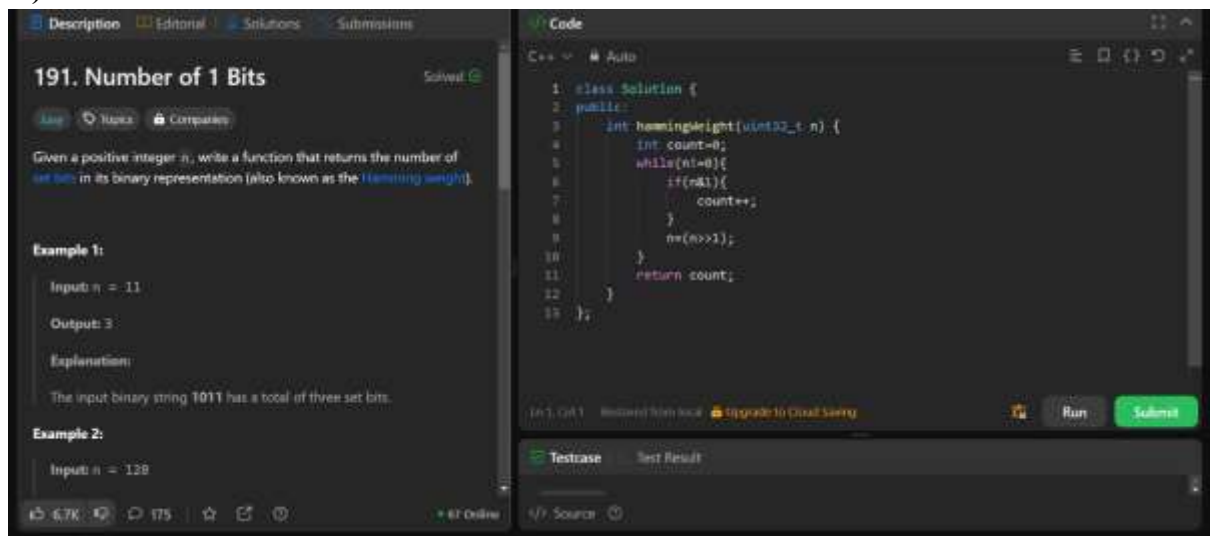
Name: Saharsh Kumar

UID: 22BCS14059

Section: 22BCS_FL_IOT-604

Group: A

1.) Number of 1 Bits



The screenshot shows the LeetCode interface for the problem '191. Number of 1 Bits'. The problem description states: 'Given a positive integer n, write a function that returns the number of 1s in its binary representation (also known as the Hamming weight)'. Example 1 shows input n = 11, output 3, with explanation: 'The input binary string 1011 has a total of three set bits.' Example 2 shows input n = 128. The C++ code on the right implements the solution using a while loop and bitwise AND operation.

```
1 class Solution {
2 public:
3     int hammingWeight(uint32_t n) {
4         int count=0;
5         while(n!=0){
6             if(n&1){
7                 count++;
8             }
9             n=(n>>1);
10        }
11        return count;
12    }
13};
```

```
int hammingWeight(uint32_t n) {  
    int count=0;  
    while(n!=0){  
        if(n&1){  
            count++;  
        }  
        n=(n>>1);  
    }  
    return count;  
}
```

2.) Longest Nice Substring

The screenshot shows the LeetCode interface for problem 1763. The left pane contains the problem description, which defines a 'nice' string as one where every letter's uppercase and lowercase forms both appear. It asks for the longest such substring, or the earliest one if there are multiple. An example shows 'YazsAxy' resulting in 'aA'. The right pane shows a C++ solution using a recursive function that iterates through the string, splitting it at each character and comparing the longest nice substrings of the two resulting parts. The bottom status bar indicates the solution is 'Accepted' with a runtime of 0 ms.

1763. Longest Nice Substring

A string *s* is *nice* if, for every letter of the alphabet that *s* contains, it appears **both** in uppercase and lowercase. For example, "aBb" is nice because 'A' and 'a' appear, and 'B' and 'b' appear. However, "abB" 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:

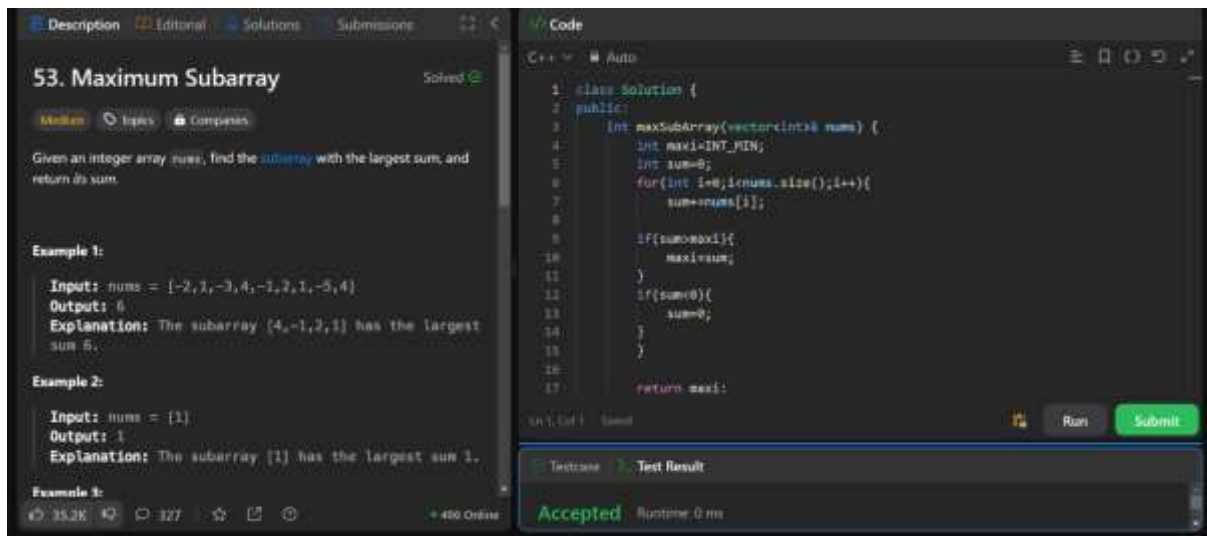
Input: *s* = "YazsAxy"
Output: "aA"
Explanation: "aA" is a nice string because 'A/a' is the only letter of the alphabet in *s*, and both 'A' and 'a' appear.

```
class Solution {
public:
    string longestNiceSubstring(string s) {
        if (s.size() < 2) return "";
        unordered_set<char> set(begin(s), end(s));
        for (int i = 0; i < s.size(); i++) {
            if (set.find((char) toupper(s[i])) == end(set) || set.find((char) tolower(s[i])) == end(set)) continue;
            string s1 = longestNiceSubstring(s.substr(0, i));
            string s2 = longestNiceSubstring(s.substr(i + 1));
            return s1.size() >= s2.size() ? s1 : s2;
        }
        return s;
    }
};
```

Accepted Runtime: 0 ms

```
string longestNiceSubstring(string s) {
    if(s.size() < 2) return "";
    unordered_set<char> set(begin(s), end(s));
    for(int i = 0; i < s.size(); i++) {
        if(!set.count((char)(s[i] ^ 32))) { // 'a' ^ 32 = 65 and 'A' ^ 32 = 97 // by XOR with 32
            we can change b/w lower and uppercase
            string s1 = longestNiceSubstring(s.substr(0, i));
            string s2 = longestNiceSubstring(s.substr(i + 1));
            return s1.size() >= s2.size() ? s1 : s2;
        }
    }
    return s;
}
```

3.) Maximum Subarray



```
int maxSubArray(vector<int>& nums) {
    int maxi=INT_MIN;
    int sum=0;
    for(int i=0;i<nums.size();i++){
        sum+=nums[i];

        if(sum>maxi){
            maxi=sum;
        }
        if(sum<0){
            sum=0;
        }
    }

    return maxi;
}
```

4.) Search a 2D Matrix II

240. Search a 2D Matrix II

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.

Example 1:

1	4	7	11	15
2	5	8	12	19
3	6	9	16	22

Code:

```
1 class Solution {
2 public:
3     bool searchMatrix(vector<vector<int>>& matrix, int target) {
4         int m = matrix.size(), n = m ? matrix[0].size() : 0, r = 0, c = m - 1;
5         while (r < m && c >= 0) {
6             if (matrix[r][c] == target) {
7                 return true;
8             }
9             matrix[r][c] > target ? c-- : r++;
10        }
11        return false;
12    }
13 }
```

Testcase: Test Result

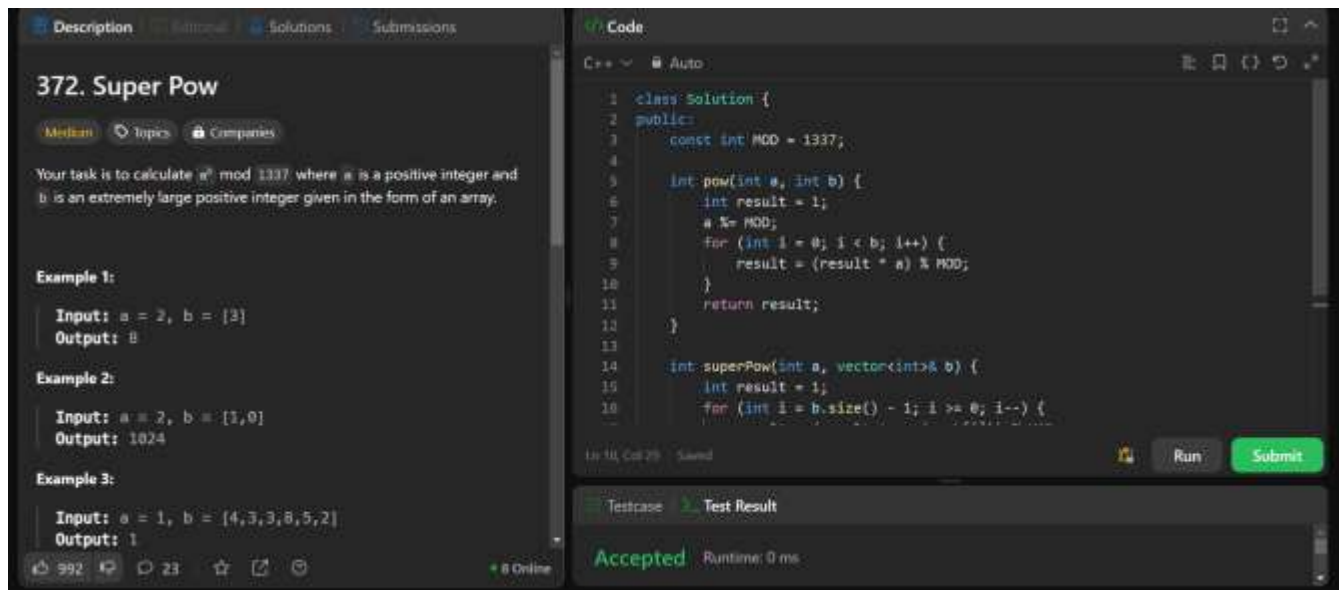
Accepted Runtime: 3 ms

Case 1 **Case 2**

Input

```
bool searchMatrix(vector<vector<int>>& matrix, int target) {
    int m = matrix.size(), n = m ? matrix[0].size() : 0, r = 0, c = n - 1;
    while (r < m && c >= 0) {
        if (matrix[r][c] == target) {
            return true;
        }
        matrix[r][c] > target ? c-- : r++;
    }
    return false;
}
```

5.) Super Pow



```
const int MOD = 1337;
```

```
int pow(int a, int b) {
    int result = 1;
    a %= MOD;
    for (int i = 0; i < b; i++) {
        result = (result * a) % MOD;
    }
    return result;
}

int superPow(int a, vector<int>& b) {
    int result = 1;
    for (int i = b.size() - 1; i >= 0; i--) {
        result = (result * pow(a, b[i])) % MOD;
        a = pow(a, 10);
    }
    return result;
}
```

```
}
```

6.) Reverse Bits

The screenshot shows the LeetCode interface for problem 190, "Reverse Bits". The left panel contains the problem description and notes. The right panel shows a C++ solution code.

190. Reverse Bits

Reverse bits of a given 32 bits unsigned integer.

Note:

- Note that in some languages, such as Java, there is no unsigned integer type. In this case, both input and output will be given as a signed integer type. They should not affect your implementation, as the integer's internal binary representation is the same, whether it is signed or unsigned.
- In Java, the compiler represents the signed integers using [two's complement notation](#). Therefore, in **Example 2** above, the input represents the signed integer -3 and the output represents the signed integer -1873741825.

Example 1:

5.3K 98 84 Online

Code

```
1 class Solution {
2 public:
3     uint32_t reverseBits(uint32_t n) {
4         long int ans=0;
5         string n1="";
6         for(int i=0;i<32;n/=2,i++){
7             n1 = to_string((n%2)) + n1;
8         }
9         for(int i =31;i>=0;i--){
10             ans+= pow(2,i)*((int)(n1[i])-48);
11             n1.erase(i);
12         }
13     }
14 }
```

Run Submit

Accepted Runtime: 0 ms

Case 1 Case 2

Input

```
uint32_t reverseBits(uint32_t n) {
    long int ans=0;
    string n1="";
    for(int i=0;i<32;n/=2,i++){
        n1 = to_string((n%2)) + n1;
    }
    for(int i =31;i>=0;i--){
        ans+= pow(2,i)*((int)(n1[i])-48);
        n1.erase(i);
    }
    return ans;
}
```