

Name: Mohammad Ashraf Khan

UID: 28BCS11123

Section: 28BCS-FS-KRG-3A

__/__/__

Q1. Test cases:

Input 1: \rightarrow Output 1: 8

$A = [1, 3, 5]$

Input 2: \rightarrow Output 2: 2

$A = [2, 3]$

Problem statement:

We define $f(x, y)$ as number of different corresponding bits in the binary representation of x and y . For example, $f(2, 7) = 2$, since the binary representation of 2 and 7 are 010 and 111 respectively. The first and third bit differs, so $f(2, 7) = 2$.

You are given an array of N positive integers, $A_1, A_2, A_3, \dots, A_n$. Find the sum of $f(A_i, A_j)$ for all pairs (i, j) such that $1 \leq i, j \leq N$. Return the answer modulo $10^9 + 7$.

Constraint:

$1 \leq N \leq 10^5$

$1 \leq A[i] \leq 2^{31} + 1$

Soln.

Approach:

count total different bits for all ordered pairs (i, j) .

as in problem statement. $f(2, 7)$:

2 \rightarrow 010	0th bit	1st bit	2nd bit
7 \rightarrow 111	2 \rightarrow 0 7 \rightarrow 1	2 \rightarrow 1 7 \rightarrow 1	2 \rightarrow 0 7 \rightarrow 1
	different	same	diff

Total different bits $f(2, 7) = 2$

Bitwise intuition:

Instead of comparing every pair (which would be too slow) we solve (bit by bit).

For a fixed bit position k :

- $\text{count}_1 = \text{no. of having bit } k \text{ set} = A$
- $\text{count}_0 = \text{no. of having bit } k \text{ unset}$

and for this bit:

- Every pair where one number has bit = 1 (set) and the other has bit = 0 (unset)

contributes 1 to answer.

Number of such ordered pairs:

$$S = (\text{count}_1 \times \text{count}_0)$$

But since (i, j) and (j, i) both counted:

$$\text{contributions} = 2 \times \text{count}_1 \times \text{count}_0$$

C++ Implementation:

```
#include <bits/stdc++.h>
```

```
using namespace std;
```

```
#define MOD 1000000007;
```

```
long long solve(vector<int> &A)
```

```
{  
    long long N = A.size();
```

```
    long long ans = 0;
```

```
    for (int bit = 0; bit < 31; bit++)
```

```
    {  
        long long count1 = 0;
```

```
        for (int i = 0; i < N; i++)
```

```
        {  
            if (A[i] & (1LL << bit))
```

```
                count1++;
```

```
        }
```

```
        long long count0 = N - count1;
```

```
        long long contribution = (2LL * count1 % MOD *  
                                count0 % MOD) % MOD;
```

```
        ans = (ans + contribution) % MOD;
```

```
    }  
    return ans;
```

```

int main() {
    ios::sync_with_stdio(false);
    cin.tie(NULL); int N; cin >> N;
    vector<int> A(N);
    for (int i = 0; i < N; i++) {
        cin >> A[i];
    }
    cout << solve(A) << "\n";
    return 0;
}

```

Time complexity :

$$O(31 \times N) = O(N)$$

dry Run for ~~A = [2, 3]~~ A = [2, 3]

2 → 10

3 → 11

bit 0

~~2 → 10~~ 2 → 10

count1 = 1

3 → 11

count0 = 1

$$\text{contribution} = 2 \times \text{count1} \times \text{count0} \\ = 2 \times 1 \times 1 = 2$$

bit 1

2 → 10

count1 = 2

3 → 11

count0 = 0

$$\text{contribution} = 2 \times 2 \times 0 = 0$$

Higher bits 2 to 30.

~~2 → 10~~

count1 = 0

count0 = ...

$$\text{contribution} = 2 \times 0 \times (...) = 0$$

$$\text{ans} = 2 + 0$$

$$= 2 \quad \checkmark$$