Imran Khan

UID: 23BCS12475

_/_/_

Q1. Test cases:

Input1: → output: 5

A=[1,2,3] : 5

Input2: → output 2:

A= [2, 2] 2

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, $A_1, A_2, A_3, \ldots, A_n$. Find the sum of $f(A_i, A_j)$ for all pairs (i,j) such that $1 \le i, j \le N$. Return the answer modulo 10^9+7.

Constraint:

$1 <= N <= 10^5$

$1 <= A[i] <= 2^{31}-1$

Soln: Approach:

count total different bits for all ordered pairs(i

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

		0th bit	1st bit	2nd bit
2 →	010	2 → 0 } different	2 → 1 } same	2 → 0 } di
7 →	111	7 → 1	7 → 1	7 → 1

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

Bitwise intuition:

Instead of comparing every pair (whichch would be too slow) we solve it (bit bit)

For a fixed bit position k:

- count1 = no. of having bit k set
- count0 = no. of having bit k 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:

count1 × count0

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

contributions = 2 × count1 × count0

C++ Implementation:

```
#include<bits/stdc++.h>
using namespace std;
#define MOD 1000000007;
longlong 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]

2 → 10
3 → 11

bit 0   2 → 1 0   count 1 = 1
3 → 1 1   count 0 = 1

contribution = 2x count1 x count0
= 2x 1x 1 = 2

bit 1   2 → 1 0   count1 = 2
3 → 1 1   count0 = 0

contribution = 2x 2x 0 = 0

Higher bits 2 to 30.
count1 = 0
count0 = ...

contribution = 2x 0 x (...) = 0

ans = 2+0
= 2 ✓