You are given integers N, P and an array a of N distinct positive integers, $A_1, A_2, \ldots, A_N$. For every **nonempty** subset $B = \{b_1, b_2, \ldots, b_m\}$ of A, you have to find $b_1 \star b_2 \star \ldots \star b_m$ and compute the sum of all of these $2^N - 1$ values modulo $10^9 + 7$. In other words, find the remainder when the sum is divided by $10^9 + 7$.

Here, $\star$ is an operation depending on value of P:

- If $P = 1$, $\star$ is addition ($+$).
- If $P = 2$, $\star$ is multiplication (`*` or $\times$).
- If $P = 3$, $\star$ is bitwise XOR (`^` or $\oplus$).

Input Format

The input consists of the integers N and P and the array A of length N.

Constraints:

- $1 \le N \le 5 \cdot 10^4$.
- $1 \le P \le 3$.
- $1 \le A_i \le 10^9$ ($1 \le i \le N$).

Output Format

Output a single integer, the specified sum modulo $10^9 + 7$.

Sample Input 1

```
N = 3
P = 1
A = [1, 2, 3]
```

Sample Output 1

```
24
```

Sample Input 2

```
N = 3
P = 2
A = [1, 2, 3]
```

Sample Output 2

```
23
```

Sample Input 3

```
N = 3
P = 3
A = [1, 2, 3]
```

Sample Output 3

```
12
```

Explanation

For sample 1, $\star$ is $+$, so the answer is $1 + 2 + 3 + (1 + 2) + (1 + 3) + (2 + 3) + (1 + 2 + 3) = 24$.

For sample 2, the answer is $1 + 2 + 3 + (1 \times 2) + (1 \times 3) + (2 \times 3) + (1 \times 2 \times 3) = 23$.

For sample 3, the answer is $1 + 2 + 3 + (1 \oplus 2) + (1 \oplus 3) + (2 \oplus 3) + (1 \oplus 2 \oplus 3) = 1 + 2 + 3 + 3 + 2 + 1 + 0 = 12$.

Anda diberi integer N, P dan suatu tatasusunan (array) A yang terdiri daripada N integer positif berbeza, $A_1, A_2, \ldots, A_N$. Bagi setiap subset **bukan kosong** $B = \{b_1, b_2, \ldots, b_m\}$ bagi A, anda perlu mencari $b_1 \star b_2 \star \ldots \star b_m$ dan hitungkan hasil tambah bagi semua $2^N - 1$ nilai tersebut modulo $10^9 + 7$. Dalam kata lain, cari bakinya apabila hasil tambah tersebut dibahagi dengan $10^9 + 7$.

Di sini, $\star$ adalah suatu operasi yang bergantung pada nilai P:

- Jika $P = 1$, $\star$ ialah operasi tambah ($+$).
- Jika $P = 2$, $\star$ ialah operasi darab (`*` atau $\times$).
- Jika $P = 3$, $\star$ ialah operasi bitwise XOR (`^` atau $\oplus$).

Format Input

Input terdiri daripada integer N dan P, dan tatasusunan A dengan panjang N.

Kekangan:

- $1 \leq N \leq 5 \cdot 10^4$.
- $1 \leq P \leq 3$.
- $1 \leq A_i \leq 10^9$ ($1 \leq i \leq N$).

Format Output

Output satu integer, iaitu hasil tambah yang dikehendaki modulo $10^9 + 7$.

Contoh Input 1

```
N = 3
P = 1
A = [1, 2, 3]
```

Contoh Output 1

```
24
```

Contoh Input 2

```
N = 3
P = 2
A = [1, 2, 3]
```

Contoh Output 2

```
23
```

Contoh Input 3

```
N = 3
P = 3
A = [1, 2, 3]
```

Contoh Output 3

```
12
```

Penjelasan

Bagi contoh 1, $\star$ ialah $+$, maka jawapannya ialah $1+2+3+(1+2)+(1+3)+(2+3)+(1+2+3)=24$.

Bagi contoh 2, jawapannya ialah $1+2+3+(1\times 2)+(1\times 3)+(2\times 3)+(1\times 2\times 3)=23$.

Bagi contoh 3, jawapannya ialah $1+2+3+(1\oplus 2)+(1\oplus 3)+(2\oplus 3)+(1\oplus 2\oplus 3)=1+2+3+3+2+1+0=12$.

Solution

The main insight is to count the contribution of each number or each bit to the sum. Call our array A and its length N.

To compute 2^k modded, we can simply multiply 2 by itself k times while modding.

P = 1 ($+$)

For each number in A, there are 2^{N-1} subsets with the number included. Therefore, the answer is $2^{N-1}(A_1) + 2^{N-1}(A_2) + \ldots + 2^{N-1}(A_N) = 2^{N-1}(A_1 + A_2 + \ldots + A_N)$.

P = 2 ($\times$)

The answer is equal to $(1 + A_1)(1 + A_2) \ldots (1 + A_N)$. The 1's in $(1 + A_i)$ represent choosing not to include A_i and the A_i's represent including it.

P = 3 (bitwise-XOR)

Consider the i-th bit (0-based index, so the i-th bit represents 2^i). For a subset S, the i-th bit is on if and only if there is an odd number of elements in S with that bit on. The elements with that bit off do not matter. Suppose there are k elements with the i-th bit on, then the contribution of the i-th bit is $2^i \cdot \left(\binom{k}{1} + \binom{k}{3} + \ldots\right) \cdot 2^{N-k}$[1]. The final answer is the sum of contributions of all bits.

Finally, since $A_i \leq 10^9$ for all i, there are at most 31 bits, so we can repeat this for each bit.

How do we compute $\binom{k}{i}$? By definition, $\binom{k}{i} = \frac{k!}{(k-i)!i!}$. Under a **prime** modulo P such as $10^9 + 7$, for all integers x where $1 \leq x \leq P - 1$, there exists a multiplicative inverse x^{-1} such that $x^{-1} \cdot x \equiv 1 \pmod{P}$. Furthermore, by Fermat's Little Theorem, you can prove that $x^{-1} \equiv x^{P-2} \pmod{P}$, which can be computed fast via binary exponentiation. Hence, we can compute $\binom{k}{i} \equiv k! \cdot ((k-i)!)^{-1} \cdot (i!)^{-1} \pmod{P}$ instead. We can also precompute factorials and their inverses, but this is optional since N is small. Modular arithmetic operations like these are very common in competitive programming, so you can search about them online. There are also templates available online like Evirir's(!) and zscoder's template.

While this is enough, we can further simplify the problem. For the i-th bit,

- If no element in A has that bit on, then the contribution of the i-th bit is zero.
- Otherwise, there is at least one element in A with that bit on. It can be proven that $\binom{k}{1} + \binom{k}{3} + \ldots = 2^{k-1}$, then

$$2^i \cdot \left(\binom{k}{1} + \binom{k}{3} + \dots\right) \cdot 2^{N-k} = 2^{i+N-k} \cdot 2^{k-1} = 2^{i+N-1}$$

Our answer is then the sum of

$$f(i) = \begin{cases} 0 & \text{if no element in } A \text{ has the } i\text{-th bit on} \\ 2^{i+N-1} & \text{otherwise} \end{cases}$$

over all i where $0 \le i \le 30$.

P.S. Sadly, while we created this problem ourselves, we did not realize that it is googleable. We apologize for missing this.

---

1. $\binom{n}{k}$ is n choose k, sometimes also written as ${}^nC_k$. ↩