Problem D
Set Game

Time limit: 6 seconds
Memory limit: 2048 megabytes

Problem Description

Alice and Bob are playing a game related to set operations. The goal of the game is simple: Alice wants to maximize the sum of the elements in the set, while Bob wants to minimize it.

The game starts with a set containing some elements, and two positive integers p and q are given.

At Alice's turn, she will receive a number k. Each time, she can select two distinct numbers (say a and b) from the set, and add $ap + bq$ to the set. She can perform this operation k times.

When it's Bob's turn, he will also receive a number k. Each time, he can select a number from the set and remove it. He can perform this operation k times as well. We guarantee that after removing elements, the set will always contain at least two elements.

Note that each element in the set appears only once; if an element is generated multiple times, it is counted only once.

We assume that both Alice and Bob play using the optimal strategy. Given the initial set, the values of p and q, and the record of the game process, write a program to calculate the sum of the elements in the set after each round. Because the sum can be very large, output the result modulo $10^9 + 7$.

Input Format

The first line contains two integers n and m — the number of set elements and the number of rounds.

The second line contains two integers p and q, which have the same meaning as described.

The third line contains n integers $a_1, a_2, ...a_n$ — the elements in the initial set.

Each of the following m lines contains the information of rounds. Each line of the round is one of the following:

- 1 k — Alice's turn, with k operations.
- 2 k — Bob's turn, with k operations.

Output Format

Output m lines. In the i-th line, output the sum of the elements in the set after i-th round, with modulo $10^9 + 7$.

Technical Specification

- $2 \leq n \leq 10^5$
- $1 \leq m \leq 10^5$
- $1 \leq p,\ q \leq 10^9$
- $1 \leq a_i \leq 10^9$
- $\forall i \neq j,\ a_i \neq a_j$
- $1 \leq k \leq 10^9$

Sample Input 1

```
5 5
2 3
1 4 2 5 7
1 2
1 1
2 2
2 3
1 3
```

Sample Output 1

```
157
540
50
7
279
```

Note

Here are some properties of modulo operations:

1. $(a + b) \mod m = ((a \mod m) + (b \mod m)) \mod m$
2. $(a - b) \mod m = ((a \mod m) - (b \mod m)) \mod m$
3. $(a \times b) \mod m = (a \mod m) \times (b \mod m) \mod m$

To compute modulo division, we need to first compute the modular inverse. Here's an introduction to the modular inverse:

The expression $a/b \mod m$ can be viewed as $a \times b^{-1} \mod m$, where we need to compute the value of $b^{-1} \mod m$, also known as the modular inverse.

According to Fermat's Little Theorem: $b^{m-1} = 1 \mod m$ if b is not a multiple of m and m is a prime number. This implies that $b^{m-2} = b^{-1} \mod m$. Therefore, to compute $b^{-1} \mod m$, we can calculate $b^{m-2} \mod m$. Since the modulo m mentioned in the problem is $10^9 + 7$ which is a prime number, we have $b^{m-1} = 1 \mod m$ for every $b \in [1, m)$.

To compute $b^{m-2} \mod m$, we can use fast exponentiation and be cautious of integer overflow issues during implementation.

```
// Compute a^n mod m
int fpow(int a, int n, int m) {
  // a^0 = 1 mod m
  if (n == 0) return 1;

  // a^1 = a mod m
  if (n == 1) return a;

  // Compute k = a^{n/2}
  int k = fpow(a, n / 2, m);

  // If n is even, a^n = k * k mod m
  if (n % 2 == 0)
    return 1ll * k * k % m;

  // Else if n is odd, a^n = k * k * a mod m
  // Be aware of the integer overflow issue
  return 1ll * k * k % m * a % m;
}

int main() {
  // Define modulo
  const int m = 1'000'000'007;

  // Any value in [1, m)
  int b = 186265;

  // Compute b_inv (b^{-1}) by Fermat's Little Theorem
  // b^{m-2} = b^{-1} mod m
  int b_inv = fpow(b, m - 2, m);

  // b * b_inv modulo m should be 1
  assert(1ll * b * b_inv % m == 1);
}
```