Codeforces Round #560 (Div. 3)

A. Remainder

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a huge decimal number consisting of n digits. It is guaranteed that this number has no leading zeros. Each digit of this number is either `0` or `1`.

You may perform several (possibly zero) operations with this number. During each operation you are allowed to change any digit of your number; you may change `0` to `1` or `1` to `0`. It is possible that after some operation you can obtain a number with leading zeroes, but it does not matter for this problem.

You are also given two integers $0 \le y < x < n$. Your task is to calculate the minimum number of operations you should perform to obtain the number that has remainder 10^y modulo 10^x. In other words, the obtained number should have remainder 10^y when divided by 10^x.

Input

The first line of the input contains three integers n, x, y ($0 \le y < x < n \le 2 \cdot 10^5$) — the length of the number and the integers x and y, respectively.

The second line of the input contains one decimal number consisting of n digits, each digit of this number is either `0` or `1`. It is guaranteed that the first digit of the number is `1`.

Output

Print one integer — the minimum number of operations you should perform to obtain the number having remainder 10^y modulo 10^x. In other words, the obtained number should have remainder 10^y when divided by 10^x.

Examples

input

```
11 5 2
11010100101
```

output

```
1
```

input

```
11 5 1
11010100101
```

output

```
3
```

Note

In the first example the number will be 11010100100 after performing one operation. It has remainder 100 modulo 100000.

In the second example the number will be 11010100010 after performing three operations. It has remainder 10 modulo 100000.

B. Polycarp Training

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Polycarp wants to train before another programming competition. During the first day of his training he should solve exactly 1 problem, during the second day — exactly 2 problems, during the third day — exactly 3 problems, and so on. During the k-th day he should solve k problems.

Polycarp has a list of n contests, the i-th contest consists of a_i problems. During each day Polycarp has to choose **exactly one** of the contests he didn't solve yet and solve it. He solves **exactly k problems from this contest**. Other problems are discarded from it. If there are no contests consisting of at least k problems that Polycarp didn't solve yet during the k-th day, then Polycarp stops his training.

How many days Polycarp can train if he chooses the contests optimally?

Input

The first line of the input contains one integer n ($1 \le n \le 2 \cdot 10^5$) — the number of contests.

The second line of the input contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 2 \cdot 10^5$) — the number of problems in the i-th contest.

Output

Print one integer — the maximum number of days Polycarp can train if he chooses the contests optimally.

Examples

input

```
4
3 1 4 1
```

output

```
3
```

input

```
3
1 1 1
```

output

```
1
```

input

```
5
1 1 1 2 2
```

output

```
2
```

C. Good String

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Let's call (yet again) a string **good** if its length is even, and every character in odd position of this string is different from the next character (the first character is different from the second, the third is different from the fourth, and so on). For example, the strings `good`, `string` and `xyyx` are good strings, and the strings `bad`, `aa` and `aabc` are not good. **Note that the empty string is considered good**.

You are given a string s, you have to delete minimum number of characters from this string so that it becomes good.

Input

The first line contains one integer n ($1 \le n \le 2 \cdot 10^5$) — the number of characters in s.

The second line contains the string s, consisting of exactly n lowercase Latin letters.

Output

In the first line, print one integer k ($0 \le k \le n$) — the minimum number of characters you have to delete from s to make it good.

In the second line, print the resulting string s. **If it is empty, you may leave the second line blank, or not print it at all**.

Examples

input

```
4
good
```

output

```
0
good
```

input

```
4
aabc
```

output

```
2
ab
```

input

```
3
aaa
```

output

```
3
```

D. Almost All Divisors

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

We guessed some integer number x. You are given a list of **almost all** its divisors. **Almost all** means that there are **all divisors except** 1 **and** x in the list.

Your task is to find the minimum possible integer x that can be the guessed number, or say that the input data is contradictory and it is impossible to find such number.

You have to answer t independent queries.

Input
The first line of the input contains one integer t ($1 \le t \le 25$) — the number of queries. Then t queries follow.

The first line of the query contains one integer n ($1 \le n \le 300$) — the number of divisors in the list.

The second line of the query contains n integers $d_1, d_2, \dots, d_n$ ($2 \le d_i \le 10^6$), where d_i is the i-th divisor of the guessed number. It is guaranteed that all values d_i are **distinct**.

Output
For each query print the answer to it.

If the input data in the query is contradictory and it is impossible to find such number x that the given list of divisors is the list of **almost all** its divisors, print -1. Otherwise print the minimum possible x.

Example

input
```
2
8
8 2 12 6 4 24 16 3
1
2
```
output
```
48
4
```

E. Two Arrays and Sum of Functions

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given two arrays a and b, both of length n.

Let's define a function $f(l, r) = \sum\limits_{l \le i \le r} a_i \cdot b_i$.

Your task is to reorder the elements (choose an arbitrary order of elements) of the array b to minimize the value of $\sum\limits_{1 \le l \le r \le n} f(l, r)$. Since the answer can be very large, you have to print it modulo 998244353. Note that you should **minimize the answer but not its remainder**.

Input
The first line of the input contains one integer n ($1 \le n \le 2 \cdot 10^5$) — the number of elements in a and b.

The second line of the input contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 10^6$), where a_i is the i-th element of a.

The third line of the input contains n integers $b_1, b_2, \dots, b_n$ ($1 \le b_j \le 10^6$), where b_j is the j-th element of b.

Output
Print one integer — the minimum possible value of $\sum\limits_{1 \le l \le r \le n} f(l, r)$ after rearranging elements of b, taken modulo 998244353. Note that you should **minimize the answer but not its remainder**.

Examples

input
```
5
1 8 7 2 4
```

```
9 7 2 9 3
```

output

```
646
```

input

```
1
1000000
1000000
```

output

```
757402647
```

input

```
2
1 3
4 2
```

output

```
20
```

F1. Microtransactions (easy version)

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

The only difference between easy and hard versions is constraints.

Ivan plays a computer game that contains some microtransactions to make characters look cooler. Since Ivan wants his character to be really cool, he wants to use some of these microtransactions — and he won't start playing until he gets all of them.

Each day (during the **morning**) Ivan earns exactly one burle.

There are n types of microtransactions in the game. Each microtransaction costs 2 burles usually and 1 burle if it is on sale. Ivan has to order exactly k_i microtransactions of the i-th type (he orders microtransactions during the **evening**).

Ivan can order **any** (possibly zero) number of microtransactions of **any** types during any day (of course, **if he has enough money to do it**). If the microtransaction he wants to order is on sale then he can buy it for 1 burle and otherwise he can buy it for 2 burles.

There are also m special offers in the game shop. The j-th offer (d_j, t_j) means that microtransactions of the t_j-th type are on sale during the d_j-th day.

Ivan wants to order all microtransactions as soon as possible. Your task is to calculate the minimum day when he can buy all microtransactions he want and actually start playing.

Input

The first line of the input contains two integers n and m ($1 \le n, m \le 1000$) — the number of types of microtransactions and the number of special offers in the game shop.

The second line of the input contains n integers $k_1, k_2, \dots, k_n$ ($0 \le k_i \le 1000$), where k_i is the number of copies of microtransaction of the i-th type Ivan has to order. It is guaranteed that **sum of all k_i is not less than 1 and not greater than** 1000.

The next m lines contain special offers. The j-th of these lines contains the j-th special offer. It is given as a pair of integers (d_j, t_j) ($1 \le d_j \le 1000, 1 \le t_j \le n$) and means that microtransactions of the t_j-th type are on sale during the d_j-th day.

Output

Print one integer — the minimum day when Ivan can order all microtransactions he wants and actually start playing.

Examples

input

```
5 6
1 2 0 2 0
2 4
3 3
1 5
1 2
1 5
2 3
```

output

```
8
```

input

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

```
3 5
4 2
2 5
```

output

```
20
```

F2. Microtransactions (hard version)

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

The only difference between easy and hard versions is constraints.

Ivan plays a computer game that contains some microtransactions to make characters look cooler. Since Ivan wants his character to be really cool, he wants to use some of these microtransactions — and he won't start playing until he gets all of them.

Each day (during the **morning**) Ivan earns exactly one burle.

There are n types of microtransactions in the game. Each microtransaction costs 2 burles usually and 1 burle if it is on sale. Ivan has to order exactly k_i microtransactions of the i-th type (he orders microtransactions during the **evening**).

Ivan can order **any** (possibly zero) number of microtransactions of **any** types during any day (of course, **if he has enough money to do it**). If the microtransaction he wants to order is on sale then he can buy it for 1 burle and otherwise he can buy it for 2 burles.

There are also m special offers in the game shop. The j-th offer (d_j, t_j) means that microtransactions of the t_j-th type are on sale during the d_j-th day.

Ivan wants to order all microtransactions as soon as possible. Your task is to calculate the minimum day when he can buy all microtransactions he want and actually start playing.

Input

The first line of the input contains two integers n and m ($1 \le n, m \le 2 \cdot 10^5$) — the number of types of microtransactions and the number of special offers in the game shop.

The second line of the input contains n integers $k_1, k_2, \dots, k_n$ ($0 \le k_i \le 2 \cdot 10^5$), where k_i is the number of copies of microtransaction of the i-th type Ivan has to order. It is guaranteed that **sum of all k_i is not less than 1 and not greater than** $2 \cdot 10^5$.

The next m lines contain special offers. The j-th of these lines contains the j-th special offer. It is given as a pair of integers (d_j, t_j) ($1 \le d_j \le 2 \cdot 10^5, 1 \le t_j \le n$) and means that microtransactions of the t_j-th type are on sale during the d_j-th day.

Output

Print one integer — the minimum day when Ivan can order all microtransactions he wants and actually start playing.

Examples

input

```
5 6
1 2 0 2 0
2 4
3 3
1 5
1 2
1 5
2 3
```

output

```
8
```

input

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

output

```
20
```