Codeforces Beta Round #17

A. Noldbach problem

time limit per test: 2 seconds
memory limit per test: 64 megabytes
input: standard input
output: standard output

Nick is interested in prime numbers. Once he read about *Goldbach problem*. It states that every even integer greater than 2 can be expressed as the sum of two primes. That got Nick's attention and he decided to invent a problem of his own and call it *Noldbach problem*. Since Nick is interested only in prime numbers, Noldbach problem states that at least k prime numbers from 2 to n inclusively can be expressed as the sum of three integer numbers: two neighboring prime numbers and 1. For example, 19 = 7 + 11 + 1, or 13 = 5 + 7 + 1.

Two prime numbers are called neighboring if there are no other prime numbers between them.

You are to help Nick, and find out if he is right or wrong.

Input

The first line of the input contains two integers n ($2 \le n \le 1000$) and k ($0 \le k \le 1000$).

Output

Output YES if at least k prime numbers from 2 to n inclusively can be expressed as it was described above. Otherwise output NO.

Examples

input
27 2
output
YES

input
45 7
output
NO

Note

In the first sample the answer is YES since at least two numbers can be expressed as it was described (for example, 13 and 19). In the second sample the answer is NO since it is impossible to express 7 prime numbers from 2 to 45 in the desired form.

B. Hierarchy

time limit per test: 2 seconds
memory limit per test: 64 megabytes
input: standard input
output: standard output

Nick's company employed n people. Now Nick needs to build a tree hierarchy of «supervisor-surbodinate» relations in the company (this is to say that each employee, except one, has exactly one supervisor). There are m applications written in the following form: «*employee a_i is ready to become a supervisor of employee b_i at extra cost c_i*». The qualification q_j of each employee is known, and for each application the following is true: $q_{a_i} > q_{b_i}$.

Would you help Nick calculate the minimum cost of such a hierarchy, or find out that it is impossible to build it.

Input

The first input line contains integer n ($1 \le n \le 1000$) — amount of employees in the company. The following line contains n space-separated numbers q_j ($0 \le q_j \le 10^6$)— the employees' qualifications. The following line contains number m ($0 \le m \le 10000$) — amount of received applications. The following m lines contain the applications themselves, each of them in the form of three space-separated numbers: a_i, b_i and c_i ($1 \le a_i, b_i \le n$, $0 \le c_i \le 10^6$). Different applications can be similar, i.e. they can come from one and the same employee who offered to become a supervisor of the same person but at a different cost. For each application $q_{a_i} > q_{b_i}$.

Output

Output the only line — the minimum cost of building such a hierarchy, or -1 if it is impossible to build it.

Examples

input
4 7 2 3 1 4 1 2 5 2 4 1 3 4 1 1 3 5
output
11

input
3 1 2 3 2 3 1 2 3 1 3
output
-1

Note

In the first sample one of the possible ways for building a hierarchy is to take applications with indexes 1, 2 and 4, which give 11 as the minimum total cost. In the second sample it is impossible to build the required hierarchy, so the answer is -1.

C. Balance

time limit per test: 3 seconds
memory limit per test: 128 megabytes
input: standard input
output: standard output

Nick likes strings very much, he likes to rotate them, sort them, rearrange characters within a string... Once he wrote a random string of characters a, b, c on a piece of paper and began to perform the following operations:

- to take two adjacent characters and replace the second character with the first one,
- to take two adjacent characters and replace the first character with the second one

To understand these actions better, let's take a look at a string «abc». All of the following strings can be obtained by performing one of the described operations on «abc»: «bbc», «abb», «acc». Let's denote the *frequency of a character* for each of the characters a, b and c as the number of occurrences of this character in the string. For example, for string «abc»: $|a| = 1$, $|b| = 1$, $|c| = 1$, and for string «bbc»: $|a| = 0$, $|b| = 2$, $|c| = 1$.

While performing the described operations, Nick sometimes got *balanced strings*. Let's say that a string is balanced, if the frequencies of each character differ by at most 1. That is $-1 \le |a| - |b| \le 1$, $-1 \le |a| - |c| \le 1$ и $-1 \le |b| - |c| \le 1$.

Would you help Nick find the number of different balanced strings that can be obtained by performing the operations described above, perhaps multiple times, on the given string s. This number should be calculated modulo 51123987.

Input

The first line contains integer n ($1 \le n \le 150$) — the length of the given string s. Next line contains the given string s. The initial string can be balanced as well, in this case it should be counted too. The given string s consists only of characters a, b and c.

Output

Output the only number — the number of different balanced strings that can be obtained by performing the described operations, perhaps multiple times, on the given string s, modulo 51123987.

Examples

input
4 abca
output
7

input
4 abbc
output
3

input
2 ab
output
1

Note

In the first sample it is possible to get 51 different strings through the described operations, but only 7 of them are balanced: «abca», «bbca», «bcca», «bcaa», «abcc», «abbc», «aabc». In the second sample: «abbc», «aabc», «abcc». In the third sample there is only one balanced string — «ab» itself.

D. Notepad

time limit per test: 2 seconds
memory limit per test: 64 megabytes
input: standard input
output: standard output

Nick is attracted by everything unconventional. He doesn't like decimal number system any more, and he decided to study other number systems. A number system with base b caught his attention. Before he starts studying it, he wants to write in his notepad all the numbers of length n without leading zeros in this number system. Each page in Nick's notepad has enough space for c numbers exactly. Nick writes every suitable number only once, starting with the first clean page and leaving no clean spaces. Nick never writes number 0 as he has unpleasant memories about zero divide.

Would you help Nick find out how many numbers will be written on the last page.

Input

The only input line contains three space-separated integers b, n and c ($2 \le b < 10^{10^6}$, $1 \le n < 10^{10^6}$, $1 \le c \le 10^9$). You may consider that Nick has infinite patience, endless amount of paper and representations of digits as characters. The numbers doesn't contain leading zeros.

Output

In the only line output the amount of numbers written on the same page as the last number.

Examples

input
2 3 3
output
1

input
2 3 4
output
4

Note

In both samples there are exactly 4 numbers of length 3 in binary number system. In the first sample Nick writes 3 numbers on the first page and 1 on the second page. In the second sample all the 4 numbers can be written on the first page.

E. Palisection

time limit per test: 2 seconds
memory limit per test: 128 megabytes
input: standard input
output: standard output

In an English class Nick had nothing to do at all, and remembered about wonderful strings called *palindromes*. We should remind you that a string is called a palindrome if it can be read the same way both from left to right and from right to left. Here are examples of such strings: «eye», «pop», «level», «aba», «deed», «racecar», «rotor», «madam».

Nick started to look carefully for all palindromes in the text that they were reading in the class. For each occurrence of each palindrome in the text he wrote a pair — the position of the beginning and the position of the ending of this occurrence in the text. Nick called each occurrence of each palindrome he found in the text *subpalindrome*. When he found all the subpalindromes, he decided to find out how many different pairs among these subpalindromes cross. Two subpalindromes cross if they cover common positions in the text. No palindrome can cross itself.

Let's look at the actions, performed by Nick, by the example of text «babb». At first he wrote out all subpalindromes:

- «b» — 1..1
- «bab» — 1..3
- «a» — 2..2
- «b» — 3..3
- «bb» — 3..4
- «b» — 4..4

Then Nick counted the amount of different pairs among these subpalindromes that cross. These pairs were six:

1. 1..1 cross with 1..3
2. 1..3 cross with 2..2
3. 1..3 cross with 3..3
4. 1..3 cross with 3..4
5. 3..3 cross with 3..4
6. 3..4 cross with 4..4

Since it's very exhausting to perform all the described actions manually, Nick asked you to help him and write a program that can find out the amount of different subpalindrome pairs that cross. Two subpalindrome pairs are regarded as different if one of the pairs contains a subpalindrome that the other does not.

Input

The first input line contains integer n ($1 \le n \le 2 \cdot 10^6$) — length of the text. The following line contains n lower-case Latin letters (from a to z).

Output

In the only line output the amount of different pairs of two subpalindromes that cross each other. Output the answer modulo 51123987.

Examples

input
4 babb
output
6

input
2 aa
output
2