Problem A. Best Subsequence

Input file:	*standard input*
Output file:	*standard output*
Time limit:	7 seconds
Memory limit:	1024 mebibytes

We call W a *k-best number* of sequence $B_{1\ldots m}$ if there exists a sequence $C_{1\ldots k}$ satisfying the following two conditions:

1. $C_{1\ldots k}$ is a subsequence of $B_{1\ldots m}$.
2. $\forall i \in [1, k], C_i + C_{(i \bmod k)+1} \le W$.

Given a sequence $A_{1\ldots n}$, you need to answer Q questions. Each question consists of three integers L, R, K, and you need to calculate the minimum K-best number of $A_{L\ldots R}$.

Recall that C is a subsequence of B if and only if we can obtain C by removing some elements of B (possibly none or all).

Input

The first line contains two integers n and Q ($1 \le n, Q \le 10^5$).

The second line contains n integers $A_{1\ldots n}$ ($0 \le A_i \le 10^9$).

Then Q lines follow. Each of them contains three integers L, R, K, representing a question ($1 \le L \le R \le n$; $1 \le K \le R - L + 1$).

Output

For each question, output a single line with a single integer: the answer.

Example

standard input	standard output
5 3 2 6 1 5 4 1 5 4 1 3 2 1 5 3	8 3 6

Problem B. Binary Search Tree

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

This problem is about Binary Search Trees (BST), a basic data structure. The structure is a rooted binary tree which stores values in its nodes. If node x contains value a, all values in the left subtree of x are less than a, and all values in the right subtree of x are greater than a.

In order to unify the details, we provide an implementation of finding a value a in a BST rooted at node x:

```
void find(x, a) {
    if (x == 0 || w[x] == a) return;
    if (w[x] > a) find(l[x], a);
    else find(r[x], a);
}
```

Here, $l[x]$ is the left child of x, $r[x]$ is the right child of x, and $w[x]$ is the value of x. Specifically, if x does not have a left child (right child), $l[x]$ ($r[x]$) is 0.

We define $A(root, a)$ as the array of all nodes visited by find$(root, a)$. We also define the cost of find$(root, a)$ as

$$\sum_{v \in A(root,a)} w[v].$$

Now there are n empty BSTs and m operations. Your task is to process these operations quickly. There are two different kinds of operations:

- "`1` l r w". For each $i \in [l, r]$, insert an integer w into the i-th BST. It is guaranteed that w is not present in these BSTs. Insertion starts at the root, goes the same as find, but instead of making the last find$(0, w)$ call, creates a new node with value w there and returns.
- "`2` x a". Calculate the cost of finding a in the x-th BST.

Input

The first line contains two integers, n and m ($1 \le n, m \le 2 \cdot 10^5$), indicating the number of BSTs and the number of operations.

Then m lines follow. Each line contains description of an operation and is formatted as either "`1` l r w" ($1 \le l \le r \le n$; $1 \le w \le 10^9$) or "`2` x a" ($1 \le x \le n$; $1 \le a \le 10^9$).

It is guaranteed that all inserted numbers (w in operations of the first kind) are different from each other.

Output

For each operation of the second kind, output a single line with a single integer: the cost.

Example

standard input	standard output
3 9 1 1 2 2 1 1 3 1 1 2 3 3 2 1 2 2 1 4 2 2 2 2 2 4 2 3 2 2 3 4	2 2 2 5 4 4

Problem C. Circle

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	512 mebibytes

There are two points A and B and an obstacle circle O on a Cartesian plane.

Now, you need to choose a point C on the boundary of O and then move both points A and B to point C. While moving, the path of either point A or B can only go outside circle O or touch its boundary.

Your goal is to minimize the total moving distance, that is, the sum of the moving distances of A and B.

Input

The first line contains a single integer t ($1 \le t \le 10^6$), the number of test cases.

Each test case is given on a single line and contains seven integers $x_1, y_1, x_2, y_2, x_3, y_3, r$, where $-10^3 \le x_1, y_1, x_2, y_2, x_3, y_3 \le 10^3$ and $1 \le r \le 10^3$. Here, $A = (x_1, y_1)$, $B = (x_2, y_2)$, and O is a circle centered at (x_3, y_3) with radius r. It is guaranteed that neither A nor B is strictly inside O.

Output

For each test case, output a single line with a single real number: the answer rounded to the third decimal place. It is guaranteed that the fourth decimal place is neither 4 nor 5.

Example

standard input	standard output
3 0 0 2 2 1 1 1 0 0 2 2 1 0 1 0 0 2 2 1 -1 1	3.571 2.927 3.116

Problem D. Composite Sequence

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

A sequence S of positive integers is a *composite sequence* if and only if there is a non-empty subsequence T of S such that the sum of all integers in T is a composite number.

Given S, your task is to check whether S is a composite sequence.

Note that 1 is not a composite number.

Recall that T is a subsequence of S if and only if we can obtain T by removing some elements of S (possibly none or all).

Input

The first line contains a single integer n ($1 \le n \le 10^5$), the size of S.

The second line contains n integers $S_1, S_2, \ldots, S_n$: the elements of S ($1 \le S_i \le 10^9$).

Output

If S is a composite sequence, output "`Yes`". Otherwise, output "`No`".

Examples

standard input	standard output
2 5 7	Yes
1 97	No

Problem E. Four XOR

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Given a sequence $A_{1\ldots n}$ of distinct integers, you need to answer whether there exist four indices x, y, z, w such that $1 \le x < y < z < w \le n$ and $A_x \oplus A_y \oplus A_z \oplus A_w = 0$.

Recall that $x \oplus y$ means the bitwise exclusive-or between x and y, sometimes expressed as x xor y.

Input

The first line contains a single integer n ($4 \le n \le 10^5$).

The second line contains n integers $A_{1\ldots n}$ ($0 \le A_i \le 10^5$). It is guaranteed that all A_i are distinct.

Output

Output "`Yes`" if there are four indices satisfying the conditions, or "`No`" otherwise.

Examples

standard input	standard output
5 1 2 3 4 5	Yes
5 1 2 4 8 16	No
5 1 3 4 8 9	No

Problem F. Game

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Now, you are playing a simple game. Given an array A of length n, your task is to control a robot to move or stop in this array.

Initially, the position of the robot is randomly selected: the probability for selecting position $i \in [1, n]$ is $\frac{1}{n}$. In each turn, you know the current position, and need to make a decision between two action choices:

- **Stop**. If this action is selected, the game ends immediately. When the robot stops at position i, your score is A_i.
- **Move**. If this action is selected and the robot is at position i, with a 50% chance, the robot will move to $i - 1$, and with another 50% chance, it will move to $i + 1$. Note that when the robot is at position 1 or n, you cannot select this action.

Since the second action can be selected only when the robot is not at either end of the array, we can prove that, for any strategy, $\lim_{m \to +\infty} f(m) = 0$, where $f(m)$ represents the probability that the game continues after m turns.

Your task is to maximize the expected score of the game.

Input

The first line contains a single integer n ($1 \le n \le 5 \cdot 10^5$).

The second line contains n integers $A_1, A_2, \ldots, A_n$ ($1 \le A_i \le 10^{12}$).

Output

Output a single line with a single integer: the maximum possible expected score as a fraction modulo 998 244 353. In other words, it can be proven that the answer can be expressed as a rational number P/Q where Q is coprime with 998 244 353, and you must output $(P \cdot Q^{-1}) \bmod 998\,244\,353$.

Examples

standard input	standard output
3 3 1 2	499122179
6 6 1 2 5 3 4	582309211

Problem G. GPA

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

In this semester, Alice took n courses. Now, she has finished all the final exams. And she will get her grades in the following n days.

On the i-th day, Alice will know her grade A_i of the i-th course. If A_i is strictly less than the average grade of the first $i-1$ courses, Alice will be sad on that day.

Now Bob is hacking into the university's database. Bob can choose a set S of courses (S can be empty). And then for each course i in S, he can change Alice's grade from A_i to B_i.

Bob wants to minimize the number of days when Alice will be sad. Now you need to help him to decide which courses' grades he should modify.

Note that Alice will always be happy on the first day.

Input

The first line contains a single integer n ($1 \le n \le 4000$).

Then n lines follow. The i-th of these lines contains two integers, A_i and B_i ($0 \le A_i, B_i \le 400$).

Output

Output the minimum number of days when Alice will be sad.

Example

standard input	standard output
4 1 2 2 3 1 2 1 1	1

Problem H. Local Maxima

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	512 mebibytes

Given an $n \times m$ integer matrix A, a *local maximum* of A is a location (i, j) ($1 \le i \le n$ and $1 \le j \le m$) such that $A_{i,j}$ is no smaller than any other integer on the i-th row or on the j-th column.

For example, in the 3×3 matrix

$$\begin{bmatrix} 2 & 5 & 4 \\ 2 & 1 & 6 \\ 2 & 2 & 2 \end{bmatrix},$$

there are three local maxima: locations $(1, 2)$, $(2, 3)$, and $(3, 1)$ with values 5, 6, and 2, repectively.

An $n \times m$ integer matrix A is *good* if and only if it satisfies the following two conditions:

- There is exactly one local maximum in A.
- Each integer from 1 to $n \times m$ occurs exactly once in A.

Given n, m, and a prime number P, your task is to count the number of good matrices of size $n \times m$ modulo P.

Input

The first line contains three integers, n, m, and P, where $1 \le n, m \le 3000$ and $10^8 \le P \le 10^9 + 7$. It is guaranteed that P is prime.

Output

Output a single line with a single integer: the number of good matrices modulo P.

Examples

standard input	standard output
2 2 1000000007	16
4 3 1000000007	95800320
100 100 998244353	848530760

Problem I. Maximum Subsequence

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	1024 mebibytes

For a sequence $a_{1\ldots n}$, define $f(a)$ as

$$f(a) = \max_{1 \le l \le r \le n} \sum_{i=l}^{r} a_i.$$

Given a sequence $b_{1\ldots n}$, you need to permute $b_{1\ldots n}$ to get $b'_{1\ldots n}$ and minimize $f(b')$.

Input

The first line contains a single integer n ($1 \le n \le 16$).

The second line contains n integers $a_{1\ldots n}$ ($|a_i| \le 10^5$).

Output

Output the minimum possible $f(b')$.

Examples

standard input	standard output
4 1 -1 1 1	2
6 4 -4 5 -20 6 7	9

Problem J. Miner

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

There are n different minerals in a mine cave. The mine cave can be regarded as a coordinate axis, and the i-th mineral can be mined from any position in range $[l_i, r_i]$.

You are a miner in this mine cave. On each day, the foreman gives you a task of mining minerals. A task is a non-empty set of different minerals (there are $2^n - 1$ different tasks), and your goal is to collect all minerals in this set.

There are m safe positions a_i in the mine cave. A task is *easy* if and only if you can select a safe position a_p and find all required minerals there.

Now, you want to count the number of easy tasks.

Input

The first line contains two integers n and m ($1 \le n, m \le 10^5$).

Then n lines follow. Each of them contains two integers l_i and r_i ($1 \le l_i \le r_i \le 10^9$).

Then m lines follow. Each of them contains a single integer a_i ($1 \le a_i \le 10^9$).

Output

Output a single line with a single integer: the number of easy tasks modulo 998 244 353.

Example

standard input	standard output
3 2 7 11 1 5 3 8 4 7	5