CODEFORCES β
Sponsored by Telegram

Educational Codeforces Round 89 (Rated for Div. 2)

A. Shovels and Swords

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

Polycarp plays a well-known computer game (we won't mention its name). In this game, he can craft tools of two types — shovels and swords. To craft a shovel, Polycarp spends two sticks and one diamond; to craft a sword, Polycarp spends two diamonds and one stick.

Each tool can be sold for exactly one emerald. How many emeralds can Polycarp earn, if he has a sticks and b diamonds?

Input

The first line contains one integer t ($1 \le t \le 1000$) — the number of test cases.

The only line of each test case contains two integers a and b ($0 \le a, b \le 10^9$) — the number of sticks and the number of diamonds, respectively.

Output

For each test case print one integer — the maximum number of emeralds Polycarp can earn.

Example

input

```
4
4 4
1000000000 0
7 15
8 7
```

output

```
2
0
7
5
```

Note

In the first test case Polycarp can earn two emeralds as follows: craft one sword and one shovel.

In the second test case Polycarp does not have any diamonds, so he cannot craft anything.

B. Shuffle

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

You are given an array consisting of n integers a_1, a_2, ..., a_n. Initially $a_x = 1$, all other elements are equal to 0.

You have to perform m operations. During the i-th operation, you choose two indices c and d such that $l_i \le c, d \le r_i$, and swap a_c and a_d.

Calculate the number of indices k such that it is possible to choose the operations so that $a_k = 1$ in the end.

Input

The first line contains a single integer t ($1 \le t \le 100$) — the number of test cases. Then the description of t testcases follow.

The first line of each test case contains three integers n, x and m ($1 \le n \le 10^9$; $1 \le m \le 100$; $1 \le x \le n$).

Each of next m lines contains the descriptions of the operations; the i-th line contains two integers l_i and r_i ($1 \le l_i \le r_i \le n$).

Output

For each test case print one integer — the number of indices k such that it is possible to choose the operations so that $a_k = 1$ in the end.

Example

input

```
3
6 4 3
```

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

output

```
6
2
3
```

Note

In the first test case, it is possible to achieve $a_k = 1$ for every k. To do so, you may use the following operations:

1. swap a_k and a_4;
2. swap a_2 and a_2;
3. swap a_5 and a_5.

In the second test case, only $k = 1$ and $k = 2$ are possible answers. To achieve $a_1 = 1$, you have to swap a_1 and a_1 during the second operation. To achieve $a_2 = 1$, you have to swap a_1 and a_2 during the second operation.

C. Palindromic Paths

time limit per test: 1.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a matrix with n rows (numbered from 1 to n) and m columns (numbered from 1 to m). A number $a_{i,j}$ is written in the cell belonging to the i-th row and the j-th column, each number is either 0 or 1.

A chip is initially in the cell $(1, 1)$, and it will be moved to the cell (n, m). During each move, it either moves to the next cell in the current row, or in the current column (if the current cell is (x, y), then after the move it can be either $(x + 1, y)$ or $(x, y + 1)$). The chip cannot leave the matrix.

Consider each path of the chip from $(1, 1)$ to (n, m). A path is called *palindromic* if the number in the first cell is equal to the number in the last cell, the number in the second cell is equal to the number in the second-to-last cell, and so on.

Your goal is to change the values in the minimum number of cells so that **every** path is *palindromic*.

Input

The first line contains one integer t ($1 \le t \le 200$) — the number of test cases.

The first line of each test case contains two integers n and m ($2 \le n, m \le 30$) — the dimensions of the matrix.

Then n lines follow, the i-th line contains m integers $a_{i,1}$, $a_{i,2}$, ..., $a_{i,m}$ ($0 \le a_{i,j} \le 1$).

Output

For each test case, print one integer — the minimum number of cells you have to change so that every path in the matrix is palindromic.

Example

input

```
4
2 2
1 1
0 1
2 3
1 1 0
1 0 0
3 7
1 0 1 1 1 1 1
0 0 0 0 0 0 0
1 1 1 1 1 0 1
3 5
1 0 1 0 0
1 1 1 1 0
0 0 1 0 0
```

output

```
0
3
4
4
```

Note

The resulting matrices in the first three test cases:

$$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 \end{pmatrix}$$

D. Two Divisors

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

You are given n integers $a_1, a_2, \dots, a_n$.

For each a_i find its **two divisors** $d_1 > 1$ and $d_2 > 1$ such that $\gcd(d_1 + d_2, a_i) = 1$ (where $\gcd(a, b)$ is the greatest common divisor of a and b) or say that there is no such pair.

Input

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

The second line contains n integers $a_1, a_2, \dots, a_n$ ($2 \le a_i \le 10^7$) — the array a.

Output

To speed up the output, print two lines with n integers in each line.

The i-th integers in the first and second lines should be corresponding divisors $d_1 > 1$ and $d_2 > 1$ such that $\gcd(d_1 + d_2, a_i) = 1$ or -1 and -1 if there is no such pair. If there are multiple answers, print any of them.

Example

input

```
10
2 3 4 5 6 7 8 9 10 24
```

output

```
-1 -1 -1 -1 3 -1 -1 -1 2 2
-1 -1 -1 -1 2 -1 -1 -1 5 3
```

Note

Let's look at $a_7 = 8$. It has 3 divisors greater than 1: 2, 4, 8. As you can see, the sum of any pair of divisors is divisible by 2 as well as a_7.

There are other valid pairs of d_1 and d_2 for $a_{10} = 24$, like $(3, 4)$ or $(8, 3)$. You can print any of them.

E. Two Arrays

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

You are given two arrays $a_1, a_2, \dots, a_n$ and $b_1, b_2, \dots, b_m$. Array b is sorted in ascending order ($b_i < b_{i+1}$ for each i from 1 to $m - 1$).

You have to divide the array a into m consecutive subarrays so that, for each i from 1 to m, the minimum on the i-th subarray is equal to b_i. Note that each element belongs to exactly one subarray, and they are formed in such a way: the first several elements of a compose the first subarray, the next several elements of a compose the second subarray, and so on.

For example, if $a = [12, 10, 20, 20, 25, 30]$ and $b = [10, 20, 30]$ then there are two good partitions of array a:

1. $[12, 10, 20], [20, 25], [30]$;
2. $[12, 10], [20, 20, 25], [30]$.

You have to calculate the number of ways to divide the array a. Since the number can be pretty large print it modulo 998244353.

Input

The first line contains two integers n and m ($1 \le n, m \le 2 \cdot 10^5$) — the length of arrays a and b respectively.

The second line contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 10^9$) — the array a.

The third line contains m integers $b_1, b_2, \dots, b_m$ ($1 \le b_i \le 10^9; b_i < b_{i+1}$) — the array b.

Output

In only line print one integer — the number of ways to divide the array a modulo 998244353.

Examples

input

```
6 3
12 10 20 20 25 30
10 20 30
```

output

```
2
```

input

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

output

```
0
```

input

```
8 2
1 2 2 2 2 2 2 2
1 2
```

output

```
7
```

F. Jog Around The Graph

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

You are given a simple weighted connected undirected graph, consisting of n vertices and m edges.

A path in the graph of length k is a sequence of $k+1$ vertices $v_1, v_2, \dots, v_{k+1}$ such that for each i $(1 \le i \le k)$ the edge (v_i, v_{i+1}) is present in the graph. A path from some vertex v also has vertex $v_1 = v$. Note that edges and vertices are allowed to be included in the path multiple times.

The weight of the path is the total weight of edges in it.

For each i from 1 to q consider a path from vertex 1 of length i of the maximum weight. What is the sum of weights of these q paths?

Answer can be quite large, so print it modulo $10^9 + 7$.

Input

The first line contains a three integers n, m, q ($2 \le n \le 2000$; $n - 1 \le m \le 2000$; $m \le q \le 10^9$) — the number of vertices in the graph, the number of edges in the graph and the number of lengths that should be included in the answer.

Each of the next m lines contains a description of an edge: three integers v, u, w ($1 \le v, u \le n$; $1 \le w \le 10^6$) — two vertices v and u are connected by an undirected edge with weight w. The graph contains no loops and no multiple edges. It is guaranteed that the given edges form a connected graph.

Output

Print a single integer — the sum of the weights of the paths from vertex 1 of maximum weights of lengths $1, 2, \dots, q$ modulo $10^9 + 7$.

Examples

input

```
7 8 25
1 2 1
2 3 10
3 4 2
1 5 2
5 6 7
6 4 15
5 3 1
1 7 3
```

output

```
4361
```

input

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

output

```
60
```

input

```
15 15 23
13 10 12
11 14 12
2 15 5
4 10 8
10 2 4
10 7 5
3 10 1
5 6 11
1 13 8
9 15 4
4 2 9
11 15 1
11 12 14
10 8 12
3 6 11
```

output

```
3250
```

input

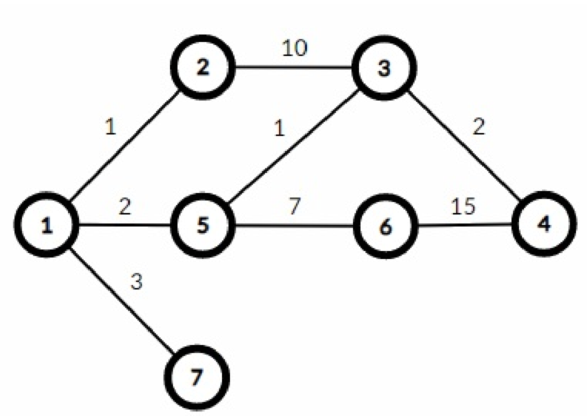
```
5 10 10000000
2 4 798
1 5 824
5 2 558
4 1 288
3 4 1890
3 1 134
2 3 1485
4 5 284
3 5 1025
1 2 649
```

output

```
768500592
```

Note

Here is the graph for the first example:



Some maximum weight paths are:

- length 1: edges $(1, 7)$ — weight 3;
- length 2: edges $(1, 2), (2, 3)$ — weight $1 + 10 = 11$;
- length 3: edges $(1, 5), (5, 6), (6, 4)$ — weight $2 + 7 + 15 = 24$;
- length 4: edges $(1, 5), (5, 6), (6, 4), (6, 4)$ — weight $2 + 7 + 15 + 15 = 39$;
- $\dots$

So the answer is the sum of 25 terms: $3 + 11 + 24 + 39 + \dots$

In the second example the maximum weight paths have weights 4, 8, 12, 16 and 20.

G. Construct the String

time limit per test: 4 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Let's denote the function $f(s)$ that takes a string s consisting of lowercase Latin letters and dots, and returns a string consisting of lowercase Latin letters as follows:

1. let r be an empty string;
2. process the characters of s from left to right. For each character c, do the following: if c is a lowercase Latin letter, append c at the end of the string r; otherwise, delete the last character from r (if r is empty before deleting the last character — the function crashes);
3. return r as the result of the function.

You are given two strings s and t. You have to delete the minimum possible number of characters from s so that $f(s) = t$ (and the function does not crash). Note that you aren't allowed to insert new characters into s or reorder the existing ones.

Input

The input consists of two lines: the first one contains s — a string consisting of lowercase Latin letters and dots, the second one contains t — a string consisting of lowercase Latin letters ($1 \le |t| \le |s| \le 10000$).

Additional constraint on the input: it is possible to remove some number of characters from s so that $f(s) = t$.

Output

Print one integer — the minimum possible number of characters you have to delete from s so $f(s)$ does not crash and returns t as the result of the function.

Examples

input

```
a.ba.b.
abb
```

output

```
2
```

input

```
.bbac..a.c.cd
bacd
```

output

```
3
```

input

```
c..code..c...o.d.de
code
```

output

```
3
```