A. CQXYM Count Permutations

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

CQXYM is counting permutations length of $2n$.

A permutation is an array consisting of n distinct integers from 1 to n in arbitrary order. For example, $[2, 3, 1, 5, 4]$ is a permutation, but $[1, 2, 2]$ is not a permutation (2 appears twice in the array) and $[1, 3, 4]$ is also not a permutation ($n = 3$ but there is 4 in the array).

A permutation p(length of $2n$) will be counted only if the number of i satisfying $p_i < p_{i+1}$ is no less than n. For example:

- Permutation $[1, 2, 3, 4]$ will count, because the number of such i that $p_i < p_{i+1}$ equals 3 ($i = 1$, $i = 2$, $i = 3$).
- Permutation $[3, 2, 1, 4]$ won't count, because the number of such i that $p_i < p_{i+1}$ equals 1 ($i = 3$).

CQXYM wants you to help him to count the number of such permutations modulo 1000000007 ($10^9 + 7$).

In addition, modulo operation is to get the remainder. For example:

- $7 \mod 3 = 1$, because $7 = 3 \cdot 2 + 1$,
- $15 \mod 4 = 3$, because $15 = 4 \cdot 3 + 3$.

Input

The input consists of multiple test cases.

The first line contains an integer $t(t \geq 1)$ — the number of test cases. The description of the test cases follows.

Only one line of each test case contains an integer $n(1 \leq n \leq 10^5)$.

It is guaranteed that the sum of n over all test cases does not exceed 10^5

Output

For each test case, print the answer in a single line.

Example

input

```
4
1
2
9
91234
```

output

```
1
12
830455698
890287984
```

Note

$n = 1$, there is only one permutation that satisfies the condition: $[1, 2]$.

In permutation $[1, 2]$, $p_1 < p_2$, and there is one $i = 1$ satisfy the condition. Since $1 \geq n$, this permutation should be counted. In permutation $[2, 1]$, $p_1 > p_2$. Because $0 < n$, this permutation should not be counted.

$n = 2$, there are 12 permutations:
$[1, 2, 3, 4], [1, 2, 4, 3], [1, 3, 2, 4], [1, 3, 4, 2], [1, 4, 2, 3], [2, 1, 3, 4], [2, 3, 1, 4], [2, 3, 4, 1], [2, 4, 1, 3], [3, 1, 2, 4], [3, 4, 1, 2], [4, 1, 2, 3]$.

B. Diameter of Graph

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

CQXYM wants to create a connected undirected graph with n nodes and m edges, and the diameter of the graph must be strictly less than $k - 1$. Also, CQXYM doesn't want a graph that contains self-loops or multiple edges (i.e. each edge connects two different vertices and between each pair of vertices there is at most one edge).

The diameter of a graph is the maximum distance between any two nodes.

The distance between two nodes is the minimum number of the edges on the path which endpoints are the two nodes.

CQXYM wonders whether it is possible to create such a graph.

Input

The input consists of multiple test cases.

The first line contains an integer $t(1 \le t \le 10^5)$ — the number of test cases. The description of the test cases follows.

Only one line of each test case contains three integers $n(1 \le n \le 10^9)$, m, k $(0 \le m, k \le 10^9)$.

Output

For each test case, print YES if it is possible to create the graph, or print NO if it is impossible. You can print each letter in any case (upper or lower).

Example

input

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

output

```
YES
NO
YES
NO
NO
```

Note

In the first test case, the graph's diameter equal to 0.

In the second test case, the graph's diameter can only be 2.

In the third test case, the graph's diameter can only be 1.

C. Portal

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

CQXYM found a rectangle A of size $n \times m$. There are n rows and m columns of blocks. Each block of the rectangle is an obsidian block or empty. CQXYM can change an obsidian block to an empty block or an empty block to an obsidian block in one operation.

A rectangle M size of $a \times b$ is called a portal if and only if it satisfies the following conditions:

- $a \ge 5, b \ge 4$.
- For all $1 < x < a$, blocks $M_{x,1}$ and $M_{x,b}$ are obsidian blocks.
- For all $1 < x < b$, blocks $M_{1,x}$ and $M_{a,x}$ are obsidian blocks.
- For all $1 < x < a, 1 < y < b$, block $M_{x,y}$ is an empty block.
- $M_{1,1}, M_{1,b}, M_{a,1}, M_{a,b}$ **can be any type**.

Note that the there must be a rows and b columns, not b rows and a columns.
Note that corners can be any type

CQXYM wants to know the minimum number of operations he needs to make at least one sub-rectangle a portal.

Input

The first line contains an integer t $(t \ge 1)$, which is the number of test cases.

For each test case, the first line contains two integers n and m $(5 \le n \le 400, 4 \le m \le 400)$.

Then n lines follow, each line contains m characters 0 or 1. If the j-th character of i-th line is 0, block $A_{i,j}$ is an empty block. Otherwise, block $A_{i,j}$ is an obsidian block.

It is guaranteed that the sum of n over all test cases does not exceed 400.

It is guaranteed that the sum of m over all test cases does not exceed 400.

Output

Output t answers, and each answer in a line.

Examples

input
1 5 4 1000 0000 0110 0000 0001

output
12

input
1 9 9 001010001 101110100 000010011 100000001 101010101 110001111 000001111 111100000 000110000

output
5

Note

In the first test case, the final portal is like this:

```
1110
1001
1001
1001
0111
```

D. Mathematics Curriculum

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

Let $c_1, c_2, \ldots, c_n$ be a permutation of integers $1, 2, \ldots, n$. Consider all subsegments of this permutation containing an integer x. Given an integer m, we call the integer x *good* if there are exactly m different values of maximum on these subsegments.

Cirno is studying mathematics, and the teacher asks her to count the number of permutations of length n with exactly k *good* numbers.

Unfortunately, Cirno isn't good at mathematics, and she can't answer this question. Therefore, she asks you for help.

Since the answer may be very big, you only need to tell her the number of permutations modulo p.

A permutation is an array consisting of n distinct integers from 1 to n in arbitrary order. For example, $[2, 3, 1, 5, 4]$ is a permutation, but $[1, 2, 2]$ is not a permutation (2 appears twice in the array) and $[1, 3, 4]$ is also not a permutation ($n = 3$ but there is 4 in the array).

A sequence a is a subsegment of a sequence b if a can be obtained from b by deletion of several (possibly, zero or all) elements from the beginning and several (possibly, zero or all) elements from the end.

Input

The first line contains four integers n, m, k, p ($1 \le n \le 100, 1 \le m \le n, 1 \le k \le n, 1 \le p \le 10^9$).

Output

Output the number of permutations modulo p.

Examples

input
4 3 2 10007

output
4

input
6 4 1 769626776

output

```
472
```

input

```
66 11 9 786747482
```

output

```
206331312
```

input

```
99 30 18 650457567
```

output

```
77365367
```

Note

In the first test case, there are four permutations: $[1,3,2,4]$, $[2,3,1,4]$, $[4,1,3,2]$ and $[4,2,3,1]$.

Take permutation $[1,3,2,4]$ as an example:

For number 1, all subsegments containing it are: $[1]$, $[1,3]$, $[1,3,2]$ and $[1,3,2,4]$, and there're three different maxima 1, 3 and 4.

Similarly, for number 3, there're two different maxima 3 and 4. For number 2, there're three different maxima 2, 3 and 4. And for number 4, there're only one, that is 4 itself.

E. Train Maintenance

time limit per test: 1 second
memory limit per test: 512 megabytes
input: standard input
output: standard output

Kawasiro Nitori is excellent in engineering. Thus she has been appointed to help maintain trains.

There are n models of trains, and Nitori's department will only have at most one train of each model at any moment. In the beginning, there are no trains, at each of the following m days, one train will be added, or one train will be removed. When a train of model i is added at day t, it works for x_i days (day t inclusive), then it is in maintenance for y_i days, then in work for x_i days again, and so on until it is removed.

In order to make management easier, Nitori wants you to help her calculate how many trains are in maintenance in each day.

On a day a train is removed, it is not counted as in maintenance.

Input

The first line contains two integers n, m ($1 \le n, m \le 2 \cdot 10^5$).

The i-th of the next n lines contains two integers x_i, y_i ($1 \le x_i, y_i \le 10^9$).

Each of the next m lines contains two integers op, k ($1 \le k \le n$, $op = 1$ or $op = 2$). If $op = 1$, it means this day's a train of model k is added, otherwise the train of model k is removed. It is guaranteed that when a train of model x is added, there is no train of the same model in the department, and when a train of model x is removed, there is such a train in the department.

Output

Print m lines, The i-th of these lines contains one integers, denoting the number of trains in maintenance in the i-th day.

Examples

input

```
3 4
10 15
12 10
1 1
1 3
1 1
2 1
2 3
```

output

```
0
1
0
0
```

input

```
5 4
1 1
10000000 100000000
998244353 1
```

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

output

```
0
0
0
1
```

Note

Consider the first example:

The first day: Nitori adds a train of model 3. Only a train of model 3 is running and no train is in maintenance.

The second day: Nitori adds a train of model 1. A train of model 1 is running and a train of model 3 is in maintenance.

The third day: Nitori removes a train of model 1. The situation is the same as the first day.

The fourth day: Nitori removes a train of model 3. There are no trains at all.

F. Subsequence

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

Alice has an integer sequence a of length n and **all elements are different**. She will choose a subsequence of a of length m, and defines the value of a subsequence $a_{b_1}, a_{b_2}, \ldots, a_{b_m}$ as

$$\sum_{i=1}^{m}(m \cdot a_{b_i}) - \sum_{i=1}^{m}\sum_{j=1}^{m} f(\min(b_i, b_j), \max(b_i, b_j)),$$

where $f(i, j)$ denotes $\min(a_i, a_{i+1}, \ldots, a_j)$.

Alice wants you to help her to maximize the value of the subsequence she choose.

A sequence s is a subsequence of a sequence t if s can be obtained from t by deletion of several (possibly, zero or all) elements.

Input

The first line contains two integers n and m ($1 \le m \le n \le 4000$).

The second line contains n distinct integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i < 2^{31}$).

Output

Print the maximal value Alice can get.

Examples

input

```
6 4
15 2 18 12 13 4
```

output

```
100
```

input

```
11 5
9 3 7 1 8 12 10 20 15 18 5
```

output

```
176
```

input

```
1 1
114514
```

output

```
0
```

input

```
2 1
666 888
```

output

```
0
```

Note

In the first example, Alice can choose the subsequence $[15, 2, 18, 13]$, which has the value $4 \cdot (15 + 2 + 18 + 13) - (15 + 2 + 2 + 2) - (2 + 2 + 2 + 2) - (2 + 2 + 18 + 12) - (2 + 2 + 12 + 13) = 100$. In the second example, there are a variety of subsequences with value 176, and one of them is $[9, 7, 12, 20, 18]$.