A1. Binary Table (Easy Version)

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

This is the easy version of the problem. The difference between the versions is in the number of possible operations that can be made. You can make hacks if and only if you solved both versions of the problem.

You are given a binary table of size $n \times m$. This table consists of symbols 0 and 1.

You can make such operation: select 3 different cells that belong to one 2×2 square and change the symbols in these cells (change 0 to 1 and 1 to 0).

Your task is to make all symbols in the table equal to 0. You are allowed to make at most $3nm$ operations. **You don't need to minimize the number of operations.**

It can be proved that it is always possible.

Input

The first line contains a single integer t ($1 \le t \le 5000$) — the number of test cases. The next lines contain descriptions of test cases.

The first line of the description of each test case contains two integers n, m ($2 \le n, m \le 100$).

Each of the next n lines contains a binary string of length m, describing the symbols of the next row of the table.

It is guaranteed that the sum of nm for all test cases does not exceed 20000.

Output

For each test case print the integer k ($0 \le k \le 3nm$) — the number of operations.

In the each of the next k lines print 6 integers $x_1, y_1, x_2, y_2, x_3, y_3$ ($1 \le x_1, x_2, x_3 \le n, 1 \le y_1, y_2, y_3 \le m$) describing the next operation. This operation will be made with three cells (x_1, y_1), (x_2, y_2), (x_3, y_3). These three cells should be different. These three cells should belong into some 2×2 square.

Example

input

```
5
2 2
10
11
3 3
011
101
110
4 4
1111
0110
0110
1111
5 5
01011
11001
00010
11011
10000
2 3
011
101
```

output

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

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

Note

In the first test case, it is possible to make only one operation with cells $(1, 1)$, $(2, 1)$, $(2, 2)$. After that, all symbols will be equal to 0.

In the second test case:

- operation with cells $(2, 1)$, $(3, 1)$, $(3, 2)$. After it the table will be:

```
011
001
000
```

- operation with cells $(1, 2)$, $(1, 3)$, $(2, 3)$. After it the table will be:

```
000
000
000
```

In the fifth test case:

- operation with cells $(1, 3)$, $(2, 2)$, $(2, 3)$. After it the table will be:

```
010
110
```

- operation with cells $(1, 2)$, $(2, 1)$, $(2, 2)$. After it the table will be:

```
000
000
```

A2. Binary Table (Hard Version)

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

This is the hard version of the problem. The difference between the versions is in the number of possible operations that can be made. You can make hacks if and only if you solved both versions of the problem.

You are given a binary table of size $n \times m$. This table consists of symbols 0 and 1.

You can make such operation: select 3 different cells that belong to one 2×2 square and change the symbols in these cells (change 0 to 1 and 1 to 0).

Your task is to make all symbols in the table equal to 0. You are allowed to make at most nm operations. **You don't need to minimize the number of operations.**

It can be proved, that it is always possible.

Input

The first line contains a single integer t ($1 \le t \le 5000$) — the number of test cases. The next lines contain descriptions of test cases.

The first line of the description of each test case contains two integers n, m ($2 \le n, m \le 100$).

Each of the next n lines contains a binary string of length m, describing the symbols of the next row of the table.

It is guaranteed, that the sum of nm for all test cases does not exceed 20000.

Output

For each test case print the integer k ($0 \le k \le nm$) — the number of operations.

In the each of the next k lines print 6 integers $x_1, y_1, x_2, y_2, x_3, y_3$ ($1 \le x_1, x_2, x_3 \le n, 1 \le y_1, y_2, y_3 \le m$) describing the next operation. This operation will be made with three cells (x_1, y_1), (x_2, y_2), (x_3, y_3). These three cells should be different. These three cells should belong to some 2×2 square.

Example

input

```
5
2 2
10
11
3 3
011
101
110
4 4
1111
0110
0110
1111
5 5
01011
11001
00010
11011
10000
2 3
011
101
```

output

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

Note

In the first test case, it is possible to make only one operation with cells $(1, 1)$, $(2, 1)$, $(2, 2)$. After that, all symbols will be equal to 0.

In the second test case:

- operation with cells $(2, 1)$, $(3, 1)$, $(3, 2)$. After it the table will be:

  ```
  011
  001
  000
  ```
- operation with cells $(1, 2)$, $(1, 3)$, $(2, 3)$. After it the table will be:

  ```
  000
  000
  000
  ```

In the fifth test case:

- operation with cells $(1, 3)$, $(2, 2)$, $(2, 3)$. After it the table will be:

  ```
  010
  110
  ```
- operation with cells $(1, 2)$, $(2, 1)$, $(2, 2)$. After it the table will be:

  ```
  000
  000
  ```

B. Graph Subset Problem

time limit per test: 1 second

memory limit per test: 256 megabytes

input: standard input
output: standard output

You are given an undirected graph with n vertices and m edges. Also, you are given an integer k.

Find either a clique of size k or a non-empty subset of vertices such that each vertex of this subset has at least k neighbors in the subset. If there are no such cliques and subsets report about it.

A subset of vertices is called a clique of size k if its size is k and there exists an edge between every two vertices from the subset. A vertex is called a neighbor of the other vertex if there exists an edge between them.

Input

The first line contains a single integer t ($1 \le t \le 10^5$) — the number of test cases. The next lines contain descriptions of test cases.

The first line of the description of each test case contains three integers n, m, k ($1 \le n, m, k \le 10^5$, $k \le n$).

Each of the next m lines contains two integers u, v $(1 \le u, v \le n, u \neq v)$, denoting an edge between vertices u and v.

It is guaranteed that there are no self-loops or multiple edges. It is guaranteed that the sum of n for all test cases and the sum of m for all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case:

If you found a subset of vertices such that each vertex of this subset has at least k neighbors in the subset in the first line output 1 and the size of the subset. On the second line output the vertices of the subset in any order.

If you found a clique of size k then in the first line output 2 and in the second line output the vertices of the clique in any order.

If there are no required subsets and cliques print -1.

If there exists multiple possible answers you can print any of them.

Example

input

```
3
5 9 4
1 2
1 3
1 4
1 5
2 3
2 4
2 5
3 4
3 5
10 15 3
1 2
2 3
3 4
4 5
5 1
1 7
2 8
3 9
4 10
5 6
7 10
10 8
8 6
6 9
9 7
4 5 4
1 2
2 3
3 4
4 1
1 3
```

output

```
2
4 1 2 3
1 10
1 2 3 4 5 6 7 8 9 10
-1
```

Note

In the first test case: the subset $\{1, 2, 3, 4\}$ is a clique of size 4.

In the second test case: degree of each vertex in the original graph is at least 3. So the set of all vertices is a correct answer.

In the third test case: there are no cliques of size 4 or required subsets, so the answer is -1.

C. Greedy Shopping

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

You are given an array $a_1, a_2, \ldots, a_n$ of integers. This array is **non-increasing**.

Let's consider a line with n shops. The shops are numbered with integers from 1 to n from left to right. The cost of a meal in the i-th shop is equal to a_i.

You should process q queries of two types:

- 1 x y: for each shop $1 \leq i \leq x$ set $a_i = max(a_i, y)$.
- 2 x y: let's consider a hungry man with y money. He visits the shops from x-th shop to n-th and if he can buy a meal in the current shop he buys one item of it. Find how many meals he will purchase. The man can buy a meal in the shop i if he has at least a_i money, and after it his money decreases by a_i.

Input

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

The second line contains n integers $a_1, a_2, \ldots, a_n$ $(1 \leq a_i \leq 10^9)$ — the costs of the meals. It is guaranteed, that $a_1 \geq a_2 \geq \ldots \geq a_n$.

Each of the next q lines contains three integers t, x, y ($1 \leq t \leq 2$, $1 \leq x \leq n$, $1 \leq y \leq 10^9$), each describing the next query.

It is guaranteed that there exists at least one query of type 2.

Output

For each query of type 2 output the answer on the new line.

Example

input

```
10 6
10 10 10 6 6 5 5 5 3 1
2 3 50
2 4 10
1 3 10
2 2 36
1 4 7
2 2 17
```

output

```
8
3
6
2
```

Note

In the first query a hungry man will buy meals in all shops from 3 to 10.

In the second query a hungry man will buy meals in shops 4, 9, and 10.

After the third query the array $a_1, a_2, \ldots, a_n$ of costs won't change and will be $\{10, 10, 10, 6, 6, 5, 5, 5, 3, 1\}$.

In the fourth query a hungry man will buy meals in shops 2, 3, 4, 5, 9, and 10.

After the fifth query the array a of costs will be $\{10, 10, 10, 7, 6, 5, 5, 5, 3, 1\}$.

In the sixth query a hungry man will buy meals in shops 2 and 4.

D. INOI Final Contests

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

Today is the final contest of INOI (Iranian National Olympiad in Informatics). The contest room is a row with n computers. All computers are numbered with integers from 1 to n from left to right. There are m participants, numbered with integers from 1 to m.

We have an array a of length m where a_i ($1 \leq a_i \leq n$) is the computer behind which the i-th participant wants to sit.

Also, we have another array b of length m consisting of characters 'L' and 'R'. b_i is the side from which the i-th participant enters the room. 'L' means the participant enters from the left of computer 1 and goes from left to right, and 'R' means the participant enters from the right of computer n and goes from right to left.

The participants in the order from 1 to m enter the room one by one. The i-th of them enters the contest room in the direction b_i and goes to sit behind the a_i-th computer. If it is occupied he keeps walking in his direction until he reaches the first unoccupied computer. After that, he sits behind it. If he doesn't find any computer he gets upset and gives up on the contest.

The madness of the i-th participant is the distance between his assigned computer (a_i) and the computer he ends up sitting behind. The distance between computers i and j is equal to $|i - j|$.

The values in the array a **can be** equal. There exist $n^m \cdot 2^m$ possible pairs of arrays (a, b).

Consider all pairs of arrays (a, b) such that no person becomes upset. For each of them let's calculate the sum of participants madnesses. Find the sum of all these values.

You will be given some prime modulo p. Find this sum by modulo p.

Input

The only line contains three integers n, m, p ($1 \le m \le n \le 500, 10^8 \le p \le 10^9 + 9$).

It is guaranteed, that the number p is prime.

Output

Print only one integer — the required sum by modulo p.

Examples

input
```
3 1 1000000007
```
output
```
0
```

input
```
2 2 1000000009
```
output
```
4
```

input
```
3 2 998244353
```
output
```
8
```

input
```
20 10 1000000009
```
output
```
352081045
```

Note

In the first test, there are three possible arrays a: $\{1\}$, $\{2\}$, and $\{3\}$ and two possible arrays b: $\{\mathtt{L}\}$ and $\{\mathtt{R}\}$. For all six pairs of arrays (a, b), the only participant will sit behind the computer a_1, so his madness will be 0. So the total sum of madnesses will be 0.

In the second test, all possible pairs of arrays (a, b), such that no person becomes upset are:

- $(\{1, 1\}, \{\mathtt{L}, \mathtt{L}\})$, the sum of madnesses is 1;
- $(\{1, 1\}, \{\mathtt{R}, \mathtt{L}\})$, the sum of madnesses is 1;
- $(\{2, 2\}, \{\mathtt{R}, \mathtt{R}\})$, the sum of madnesses is 1;
- $(\{2, 2\}, \{\mathtt{L}, \mathtt{R}\})$, the sum of madnesses is 1;
- all possible pairs of $a \in \{\{1, 2\}, \{2, 1\}\}$ and $b \in \{\{\mathtt{L}, \mathtt{L}\}, \{\mathtt{R}, \mathtt{L}\}, \{\mathtt{L}, \mathtt{R}\}, \{\mathtt{R}, \mathtt{R}\}\}$, the sum of madnesses is 0.

So, the answer is $1 + 1 + 1 + 1 + 0 \ldots = 4$.

E. Cheat and Win

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

Let's consider a $(10^9 + 1) \times (10^9 + 1)$ field. The rows are numbered with integers from 0 to 10^9 and the columns are numbered with integers from 0 to 10^9. Let's define as (x, y) the cell located in the x-th row and y-th column.

Let's call a cell (x, y) good if $x \& y = 0$, there $\&$ is the bitwise and operation.

Let's build a graph where vertices will be all good cells of the field and we will make an edge between all pairs of adjacent by side good cells. It can be proved that this graph will be a tree — connected graph without cycles. Let's hang this tree on vertex $(0, 0)$, so we will have a rooted tree with root $(0, 0)$.

Two players will play the game. Initially, some good cells are black and others are white. Each player on his turn chooses a black good cell and a subset of its ancestors (possibly empty) and inverts their colors (from white to black and vice versa). The player who can't move (because all good cells are white) loses. It can be proved that the game is always finite.

Initially, all cells are white. You are given m pairs of cells. For each pair color all cells in a simple path between them as black. Note that we do not invert their colors, we paint them black.

Sohrab and Mashtali are going to play this game. Sohrab is the first player and Mashtali is the second.

Mashtali wants to win and decided to cheat. He can make the following operation multiple times before the game starts: choose a cell and invert colors of all vertices on the path between it and the root of the tree.

Mammad who was watching them wondered: "what is the minimum number of operations Mashtali should do to have a winning strategy?".

Find the answer to this question for the initial painting of the tree. It can be proved that at least one possible way to cheat always exists.

Input

The first line contains one integer m ($1 \le m \le 10^5$).

Each of the next m lines contains four integers x_1, y_1, x_2, y_2 ($0 \le x_i, y_i \le 10^9$, $x_i \& y_i = 0$). You should color all cells on the path between vertices (x_1, y_1) and (x_2, y_2) as black.

Output

Print a single integer — the minimum number of cheating operations the second player can do.

Examples

input

```
1
7 0 0 7
```

output

```
1
```

input

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

output

```
3
```

input

```
2
0 1 0 8
1 0 8 0
```

output

```
0
```

Note

In the first test, you can make one cheating operation with the root of the tree. After that, the second player can win because he can use a symmetric strategy.

In the second test, you can make cheating operations with cells $(0, 2), (0, 0), (3, 4)$.

In the third test, the second player already has the winning strategy and doesn't need to make any cheating operations.