UVa Online Judge

12629 Rectangle XOR Game

There is a '01' matrix with n rows and m columns, having at least one 1. Rows are numbered 1 to n from top to bottom, columns are numbered 1 to m from left to right.

Two players move in turn. In each move, the player selects a rectangle $(x_1, y_1) - (x_2, y_2)$ ($1 \le x_1 \le x_2 \le n$, $1 \le y_1 \le y_2 \le m$) and flips all the numbers ($0 \to 1$, $1 \to 0$) in it (i.e. the flip all positions (x, y) where $x_1 \le x \le x_2$, $y_1 \le y \le y_2$)). The only restriction is that the top-left corner (i.e. (x_1, y_1)) must be changing from 1 to 0.

After a player's move, if the matrix contains only zeros, he wins.

Your task is to determine whether the first player can win, if both players play perfectly.

If he can win, count how many different first moves can lead to victory, and print the lexicographically smallest one (i.e. x_1 should be as smallest as possible, then y_1, x_2, y_2)

Input

The input contains at most 400 test cases. Each case begins with two integers n and m ($1 \le n, m \le 100$). The next n lines contain the matrix. Each line contains m integers (Either '0' or '1'). There will be at least one '1' in the matrix.

The input ends with EOF

Output

For each test case, if the first player cannot win, print a single string 'No' (without quotes), otherwise print five integers c x_1 y_1 x_2 y_2, where c is the number of winning first moves, and $(x_1, y_1) - (x_2, y_2)$ is the lexicographically smallest first move.

Explanation below:

Case 1:

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

The first player loses, because he has 4 moves, leading to:

```
0 0
1 0

0 0
1 1

0 1
0 0

0 1
0 1
```

The second player can flip all 1's to 0's.

Case 2:

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

This is winning, because the first player can flip the whole board and turn to the losing game analyzed above.

Case 3:

```
2 2
1 1
1 1
```

There are 3 winning moves, leading to the following games:

```
0 0
0 0

1 0
1 1

1 1
0 1
```

The lexicographically first one is $(1, 1) - (2, 2)$.

Sample Input

```
2 2
0 1
1 0
2 2
1 0
0 1
2 2
1 1
1 1
1 7
0 1 1 1 1 0 0
```

Sample Output

```
No
1 1 1 2 2
3 1 1 2 2
3 1 2 1 5
```