Codeforces Round #254 (Div. 2)

A. DZY Loves Chessboard

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

DZY loves chessboard, and he enjoys playing with it.

He has a chessboard of n rows and m columns. Some cells of the chessboard are bad, others are good. For every good cell, DZY wants to put a chessman on it. Each chessman is either white or black. After putting all chessmen, DZY wants that no two chessmen with the same color are on two adjacent cells. Two cells are adjacent if and only if they share a common edge.

You task is to find any suitable placement of chessmen on the given chessboard.

Input

The first line contains two space-separated integers n and m $(1 \leq n, m \leq 100)$.

Each of the next n lines contains a string of m characters: the j-th character of the i-th string is either "`.`" or "`-`". A "`.`" means that the corresponding cell (in the i-th row and the j-th column) is good, while a "`-`" means it is bad.

Output

Output must contain n lines, each line must contain a string of m characters. The j-th character of the i-th string should be either "`W`", "`B`" or "`-`". Character "`W`" means the chessman on the cell is white, "`B`" means it is black, "`-`" means the cell is a bad cell.

If multiple answers exist, print any of them. It is guaranteed that at least one answer exists.

Sample test(s)

input
1 1 .
output
B

input
2 2 .. ..
output
BW WB

input
3 3 .-. --- --.
output
B-B --- --B

Note

In the first sample, DZY puts a single black chessman. Of course putting a white one is also OK.

In the second sample, all 4 cells are good. No two same chessmen share an edge in the sample output.

In the third sample, no good cells are adjacent. So you can just put 3 chessmen, no matter what their colors are.

B. DZY Loves Chemistry

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

DZY loves chemistry, and he enjoys mixing chemicals.

DZY has n chemicals, and m pairs of them will react. He wants to pour these chemicals into a test tube, and he needs to pour them in one by one, in any order.

Let's consider the danger of a test tube. Danger of an empty test tube is 1. And every time when DZY pours a chemical, if there are already one or more chemicals in the test tube that can react with it, the danger of the test tube will be multiplied by 2. Otherwise the danger remains as it is.

Find the maximum possible danger after pouring all the chemicals one by one in optimal order.

Input

The first line contains two space-separated integers n and m $(1 \le n \le 50;\ 0 \le m \le \frac{n(n-1)}{2})$.

Each of the next m lines contains two space-separated integers x_i and y_i $(1 \le x_i < y_i \le n)$. These integers mean that the chemical x_i will react with the chemical y_i. Each pair of chemicals will appear at most once in the input.

Consider all the chemicals numbered from 1 to n in some order.

Output

Print a single integer — the maximum possible danger.

Sample test(s)

input
1 0
output
1

input
2 1 1 2
output
2

input
3 2 1 2 2 3
output
4

Note

In the first sample, there's only one way to pour, and the danger won't increase.

In the second sample, no matter we pour the 1st chemical first, or pour the 2nd chemical first, the answer is always 2.

In the third sample, there are four ways to achieve the maximum possible danger: 2-1-3, 2-3-1, 1-2-3 and 3-2-1 (that is the numbers of the chemicals in order of pouring).

C. DZY Loves Physics

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

DZY loves Physics, and he enjoys calculating density.

Almost everything has density, even a graph. We define the density of a non-directed graph (nodes and edges of the graph have some values) as follows:

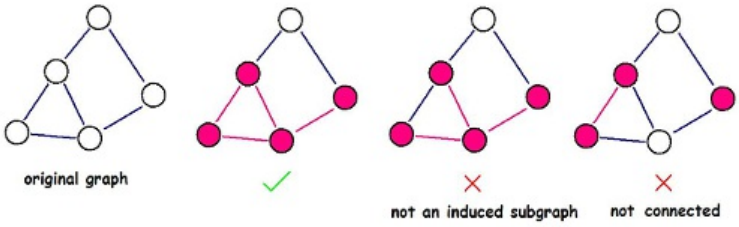
$$\begin{cases} \frac{v}{e} & (e > 0) \\ 0 & (e = 0) \end{cases}$$

where v is the sum of the values of the nodes, e is the sum of the values of the edges.
Once DZY got a graph G, now he wants to find a connected induced subgraph G' of the graph, such that the density of G' is as large as possible.

An induced subgraph $G'(V', E')$ of a graph $G(V, E)$ is a graph that satisfies:

- $V' \subseteq V$;
- edge $(a, b) \in E'$ if and only if $a \in V', b \in V'$, and edge $(a, b) \in E$;
- the value of an edge in G' is the same as the value of the corresponding edge in G, so as the value of a node.

Help DZY to find the induced subgraph with maximum density. Note that the induced subgraph you choose must be connected.



Input

The first line contains two space-separated integers n $(1 \le n \le 500)$, m $(0 \le m \le \frac{n(n-1)}{2})$. Integer n represents the number of nodes of the graph G, m represents the number of edges.

The second line contains n space-separated integers x_i $(1 \le x_i \le 10^6)$, where x_i represents the value of the i-th node. Consider the graph nodes are numbered from 1 to n.

Each of the next m lines contains three space-separated integers a_i, b_i, c_i $(1 \le a_i < b_i \le n; 1 \le c_i \le 10^3)$, denoting an edge between node a_i and b_i with value c_i. The graph won't contain multiple edges.

Output

Output a real number denoting the answer, with an absolute or relative error of at most 10^{-9}.

Sample test(s)

input
1 0 1
output
0.000000000000000

input
2 1 1 2 1 2 1
output
3.000000000000000

input
5 6 13 56 73 98 17 1 2 56 1 3 29 1 4 42 2 3 95 2 4 88 3 4 63
output

```
2.965517241379311
```

Note

In the first sample, you can only choose an empty subgraph, or the subgraph containing only node 1.

In the second sample, choosing the whole graph is optimal.

D. DZY Loves FFT

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

DZY loves Fast Fourier Transformation, and he enjoys using it.

Fast Fourier Transformation is an algorithm used to calculate convolution. Specifically, if a, b and c are sequences with length n, which are indexed from 0 to n - 1, and

$$c_i = \sum_{j=0}^{i} a_j b_{i-j}.$$

We can calculate c fast using Fast Fourier Transformation.

DZY made a little change on this formula. Now

$$c_i = \max_{j=0}^{i} a_j b_{i-j}.$$

To make things easier, a is a permutation of integers from 1 to n, and b is a sequence only containing 0 and 1. Given a and b, DZY needs your help to calculate c.

Because he is naughty, DZY provides a special way to get a and b. What you need is only three integers n, d, x. After getting them, use the code below to generate a and b.

```
//x is 64-bit variable;
function getNextX() {
    x = (x * 37 + 10007) % 1000000007;
    return x;
}
function initAB() {
    for(i = 0; i < n; i = i + 1){
        a[i] = i + 1;
    }
    for(i = 0; i < n; i = i + 1){
        swap(a[i], a[getNextX() % (i + 1)]);
    }
    for(i = 0; i < n; i = i + 1){
        if (i < d)
            b[i] = 1;
        else
            b[i] = 0;
    }
    for(i = 0; i < n; i = i + 1){
        swap(b[i], b[getNextX() % (i + 1)]);
    }
}
```

Operation `x % y` denotes remainder after division x by y. Function `swap(x, y)` swaps two values x and y.

Input

The only line of input contains three space-separated integers n, d, x ($1 \le d \le n \le 100000$; $0 \le x \le 1000000006$). Because DZY is naughty, x can't be equal to 27777500.

Output

Output n lines, the i-th line should contain an integer c_{i-1}.

Sample test(s)

input
3 1 1

output
1 3 2

input
5 4 2

output

```
2
2
4
5
5
```

input

```
5 4 3
```

output

```
5
5
5
5
4
```

Note

In the first sample, a is $[1\ 3\ 2]$, b is $[1\ 0\ 0]$, so $c_0 = max(1 \cdot 1) = 1$, $c_1 = max(1 \cdot 0, 3 \cdot 1) = 3$, $c_2 = max(1 \cdot 0, 3 \cdot 0, 2 \cdot 1) = 2$.

In the second sample, a is $[2\ 1\ 4\ 5\ 3]$, b is $[1\ 1\ 1\ 0\ 1]$.

In the third sample, a is $[5\ 2\ 1\ 4\ 3]$, b is $[1\ 1\ 1\ 1\ 0]$.

E. DZY Loves Colors

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

DZY loves colors, and he enjoys painting.

On a colorful day, DZY gets a colorful ribbon, which consists of n units (they are numbered from 1 to n from left to right). The color of the i-th unit of the ribbon is i at first. It is colorful enough, but we still consider that the colorfulness of each unit is 0 at first.

DZY loves painting, we know. He takes up a paintbrush with color x and uses it to draw a line on the ribbon. In such a case some contiguous units are painted. Imagine that the color of unit i currently is y. When it is painted by this paintbrush, the color of the unit becomes x, and the colorfulness of the unit increases by $|x - y|$.

DZY wants to perform m operations, each operation can be one of the following:

1. Paint all the units with numbers between l and r (both inclusive) with color x.
2. Ask the sum of colorfulness of the units between l and r (both inclusive).

Can you help DZY?

Input

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

Each of the next m lines begins with a integer $type$ $(1 \le type \le 2)$, which represents the type of this operation.

If $type = 1$, there will be 3 more integers l, r, x $(1 \le l \le r \le n;\ 1 \le x \le 10^8)$ in this line, describing an operation 1.

If $type = 2$, there will be 2 more integers l, r $(1 \le l \le r \le n)$ in this line, describing an operation 2.

Output

For each operation 2, print a line containing the answer — sum of colorfulness.

Sample test(s)

input

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

output

```
8
```

input

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

output

```
3
2
1
```

input

```
10 6
1 1 5 3
1 2 7 9
1 10 10 11
1 3 8 12
1 1 10 3
2 1 10
```

output

```
129
```

Note

In the first sample, the color of each unit is initially $[1, 2, 3]$, and the colorfulness is $[0, 0, 0]$.

After the first operation, colors become $[4, 4, 3]$, colorfulness become $[3, 2, 0]$.

After the second operation, colors become $[4, 5, 5]$, colorfulness become $[3, 3, 2]$.

So the answer to the only operation of type 2 is 8.