# A. The Brand New Function

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

Polycarpus has a sequence, consisting of $n$ non-negative integers: $a_1, a_2, ..., a_n$.

Let's define function $f(l, r)$ ($l, r$ are integer, $1 \le l \le r \le n$) for sequence $a$ as an operation of bitwise OR of all the sequence elements with indexes from $l$ to $r$. Formally: $f(l, r) = a_l \mid a_{l+1} \mid ... \mid a_r$.

Polycarpus took a piece of paper and wrote out the values of function $f(l, r)$ for all $l, r$ ($l, r$ are integer, $1 \le l \le r \le n$). Now he wants to know, how many **distinct** values he's got in the end.

Help Polycarpus, count the number of distinct values of function $f(l, r)$ for the given sequence $a$.

Expression $x \mid y$ means applying the operation of bitwise OR to numbers $x$ and $y$. This operation exists in all modern programming languages, for example, in language *C++* and *Java* it is marked as "`|`", in *Pascal* — as "`or`".

## Input

The first line contains integer $n$ $(1 \le n \le 10^5)$ — the number of elements of sequence $a$. The second line contains $n$ space-separated integers $a_1, a_2, ..., a_n$ $(0 \le a_i \le 10^6)$ — the elements of sequence $a$.

## Output

Print a single integer — the number of distinct values of function $f(l, r)$ for the given sequence $a$.

Please, do not use the `%lld` specifier to read or write 64-bit integers in C++. It is preferred to use `cin`, `cout` streams or the `%I64d` specifier.

**Sample test(s)**

input

```
3
1 2 0
```

output

```
4
```

input

```
10
1 2 3 4 5 6 1 2 9 10
```

output

```
11
```

## Note

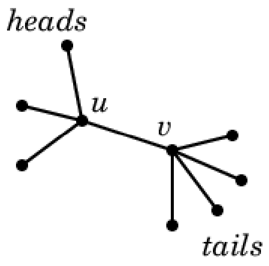
In the first test case Polycarpus will have 6 numbers written on the paper: $f(1, 1) = 1$, $f(1, 2) = 3$, $f(1, 3) = 3$, $f(2, 2) = 2$, $f(2, 3) = 2$, $f(3, 3) = 0$. There are exactly $4$ distinct numbers among them: $0, 1, 2, 3$.

## B. Hydra

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

One day Petya got a birthday present from his mom: a book called "The Legends and Myths of Graph Theory". From this book Petya learned about a *hydra* graph.

A non-oriented graph is a *hydra*, if it has a structure, shown on the figure below. Namely, there are two nodes $u$ and $v$ connected by an edge, they are the hydra's *chest* and *stomach*, correspondingly. The chest is connected with $h$ nodes, which are the hydra's *heads*. The stomach is connected with $t$ nodes, which are the hydra's *tails*. Note that the hydra is a tree, consisting of $h+t+2$ nodes.



Also, Petya's got a non-directed graph $G$, consisting of $n$ nodes and $m$ edges. Petya got this graph as a last year birthday present from his mom. Graph $G$ contains no self-loops or multiple edges.

Now Petya wants to find a hydra in graph $G$. Or else, to make sure that the graph doesn't have a hydra.

### Input

The first line contains four integers $n, m, h, t$ $(1 \le n, m \le 10^5, 1 \le h, t \le 100)$ — the number of nodes and edges in graph $G$, and the number of a hydra's heads and tails.

Next $m$ lines contain the description of the edges of graph $G$. The $i$-th of these lines contains two integers $a_i$ and $b_i$ $(1 \le a_i, b_i \le n, a \ne b)$ — the numbers of the nodes, connected by the $i$-th edge.

It is guaranteed that graph $G$ contains no self-loops and multiple edges. Consider the nodes of graph $G$ numbered with integers from 1 to $n$.

### Output

If graph $G$ has no hydra, print "`NO`" (without the quotes).

Otherwise, in the first line print "`YES`" (without the quotes). In the second line print two integers — the numbers of nodes $u$ and $v$. In the third line print $h$ numbers — the numbers of the nodes that are the heads. In the fourth line print $t$ numbers — the numbers of the nodes that are the tails. All printed numbers should be distinct.

If there are multiple possible answers, you are allowed to print any of them.

**Sample test(s)**

input
```
9 12 2 3
1 2
2 3
1 3
1 4
2 5
4 5
4 6
6 5
6 7
7 5
8 7
9 1
```

output
```
YES
4 1
5 6
9 3 2
```

input
```
7 10 3 3
1 2
2 3
1 3
1 4
2 5
4 5
```

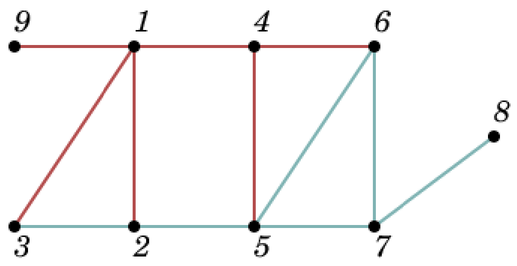
```
4 6
6 5
6 7
7 5
```

output

```
NO
```

**Note**

The first sample is depicted on the picture below:



# C. Colorado Potato Beetle

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

Old MacDonald has a farm and a large potato field, $(10^{10}+1)\times(10^{10}+1)$ square meters in size. The field is divided into square garden beds, each bed takes up one square meter.

Old McDonald knows that the Colorado potato beetle is about to invade his farm and can destroy the entire harvest. To fight the insects, Old McDonald wants to spray some beds with insecticides.

So Old McDonald went to the field, stood at the center of the central field bed and sprayed this bed with insecticides. Now he's going to make a series of movements and spray a few more beds. During each movement Old McDonald moves left, right, up or down the field some integer number of meters. As Old McDonald moves, he sprays all the beds he steps on. In other words, the beds that have any intersection at all with Old McDonald's trajectory, are sprayed with insecticides.

When Old McDonald finished spraying, he wrote out all his movements on a piece of paper. Now he wants to know how many beds won't be infected after the invasion of the Colorado beetles.

It is known that the invasion of the Colorado beetles goes as follows. First some bed on the field border gets infected. Than any bed that hasn't been infected, hasn't been sprayed with insecticides and has a common side with an infected bed, gets infected as well. Help Old McDonald and determine the number of beds that won't be infected by the Colorado potato beetle.

## Input

The first line contains an integer $n$ $(1 \le n \le 1000)$ — the number of Old McDonald's movements.

Next $n$ lines contain the description of Old McDonald's movements. The $i$-th of these lines describes the $i$-th movement. Each movement is given in the format "$d_i$ $x_i$", where $d_i$ is the character that determines the direction of the movement ("`L`", "`R`", "`U`" or "`D`" for directions "left", "right", "up" and "down", correspondingly), and $x_i$ $(1 \le x_i \le 10^6)$ is an integer that determines the number of meters in the movement.

## Output

Print a single integer — the number of beds that won't be infected by the Colorado potato beetle.

Please do not use the `%lld` specifier to read or write 64-bit integers in C++. It is preferred to use the `cin`, `cout` streams or the `%I64d` specifier.

**Sample test(s)**

input

```
5
R 8
U 9
L 9
D 8
L 2
```

output

```
101
```

input

```
7
R 10
D 2
L 7
U 9
D 2
R 3
D 10
```

output

```
52
```

# D. Cubes

time limit per test: 5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

One day Petya got a set of wooden cubes as a present from his mom. Petya immediately built a whole city from these cubes.

The base of the city is an $n \times n$ square, divided into unit squares. The square's sides are parallel to the coordinate axes, the square's opposite corners have coordinates $(0, 0)$ and $(n, n)$. On each of the unit squares Petya built a tower of wooden cubes. The side of a wooden cube also has a unit length.

After that Petya went an infinitely large distance away from his masterpiece and looked at it in the direction of vector $v = (v_x, v_y, 0)$. Petya wonders, how many distinct cubes are visible from this position. Help him, find this number.

Each cube includes the border. We think that a cube is visible if there is a ray emanating from some point $p$, belonging to the cube, in the direction of vector $-v$, that doesn't contain any points, belonging to other cubes.

## Input

The first line contains three integers $n$, $v_x$ and $v_y$ $(1 \le n \le 10^3, |v_x|, |v_y| \le |10^4|, |v_x| + |v_y| > 0)$.

Next $n$ lines contain $n$ integers each: the $j$-th integer in the $i$-th line $a_{ij}$ $(0 \le a_{ij} \le 10^9, 1 \le i, j \le n)$ represents the height of the cube tower that stands on the unit square with opposite corners at points $(i - 1, j - 1)$ and $(i, j)$.

## Output

Print a single integer — the number of visible cubes.

Please, do not use the `%lld` specifier to read or write 64-bit integers in C++. It is preferred to use the `cin`, `cout` streams or the `%I64d` specifier.

### Sample test(s)

input

```
5 -1 2
5 0 0 0 1
0 0 0 0 2
0 0 0 1 2
0 0 0 0 2
2 2 2 2 3
```

output

```
20
```

input

```
5 1 -2
5 0 0 0 1
0 0 0 0 2
0 0 0 1 2
0 0 0 0 2
2 2 2 2 3
```

output

```
15
```

# E. Matrix

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

Let's consider an $n \times n$ square matrix, consisting of digits one and zero.

We'll consider a matrix *good*, if it meets the following condition: in each row of the matrix all ones go in one group. That is, each row of the matrix looks like that $00...0011...1100...00$ (or simply consists of zeroes if it has no ones).

You are given matrix $a$ of size $n \times n$, consisting of zeroes and ones. Your task is to determine whether you can get a good matrix $b$ from it by rearranging the columns or not.

### Input

The first line contains integer $n$ $(1 \le n \le 500)$ — the size of matrix $a$.

Each of $n$ following lines contains $n$ characters "`0`" and "`1`" — matrix $a$. Note that the characters are written without separators.

### Output

Print "`YES`" in the first line, if you can rearrange the matrix columns so as to get a good matrix $b$. In the next $n$ lines print the good matrix $b$. If there are multiple answers, you are allowed to print any of them.

If it is impossible to get a good matrix, print "`NO`".

**Sample test(s)**

input

```
6
100010
110110
011001
010010
000100
011001
```

output

```
YES
011000
111100
000111
001100
100000
000111
```

input

```
3
110
101
011
```

output

```
NO
```