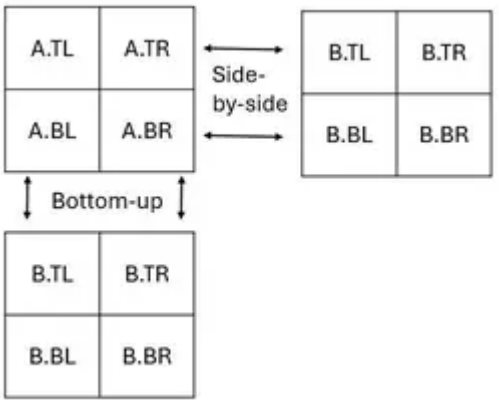
Tiling

The square of Tirana is being decorated with a mosaic of square tiles of unit size.

Each corner of a tile is colored black or white, written as 0 and 1. A tile is described by four bits TL, TR, BL, BR: the colors of its top-left, top-right, bottom-left, and bottom-right corners.

Two tiles may be placed next to each other only if the colors at both ends of their common side match:

- if tile A is directly to the left of tile B, then $A.TR = B.TL$ and $A.BR = B.BL$;
- if tile A is directly above tile B, then $A.BL = B.TL$ and $A.BR = B.TR$.



A mosaic is valid if every two horizontally or vertically adjacent tiles match.

You are given k allowed tile types and the complete first row of a narrow mosaic with n rows. The width of the mosaic in tiles is

$$w = \max\left(1, \lceil \log_2 n \rceil\right).$$

Each tile type may be used any number of times. Determine whether the given first row can be extended to a valid mosaic of n rows and w columns. If it can, output one such mosaic.

Input format

The first line contains two integers n and k — the number of rows of the mosaic and the number of allowed tile types.

Each of the next k lines contains four integers TL, TR, BL, BR describing one tile type. Tile types are numbered from 1 to k in the order given. Several tile types may have identical corner colors.

The last line contains w integers $a_1, a_2, \ldots, a_w$ — the tile types already placed in the first row, from left to right. **The given first row is not necessarily valid.**

Output format

If the mosaic cannot be completed, print `NO`. This includes the case where the given first row itself is not valid.

Otherwise, print `YES`, followed by n lines describing a valid mosaic from top to bottom. Each of these lines must contain w tile types. The first of these lines must be exactly $a_1, a_2, \ldots, a_w$. If there are several valid mosaics, print any of them.

Constraints

- $1 \leq n \leq 1000$
- $1 \leq k \leq 16$
- Each of the four values on a tile line is 0 or 1.
- $1 \leq a_j \leq k$ for every j.

Subtasks

Subtask	Points	Additional constraints
1	10	$n \leq 4$
2	15	$n \leq 8$
3	20	$n \leq 100$
4	20	$n \leq 300$
5	35	No additional constraints.

Examples

Example 1

Input:

```
4 4
0 0 0 0
0 1 0 1
1 0 1 0
1 1 1 1
2 3
```

Output:

```
YES
2 3
2 3
2 3
2 3
```

Here $n = 4$, so $w = \lceil \log_2 4 \rceil = 2$ and the mosaic is two tiles wide.

Tile 2 has $TL = 0$, $TR = 1$, $BL = 0$, $BR = 1$ and tile 3 has $TL = 1$, $TR = 0$, $BL = 1$, $BR = 0$. In the first row, tile 2 is to the left of tile 3, and their common side matches because $2.TR = 1 = 3.TL$ and $2.BR = 1 = 3.BL$.

The bottom boundary of that row reads $0, 1, 0$, which is also its top boundary, so the same row may be repeated as often as needed. It is the only completion: a top boundary of $0, 1, 0$ allows only tile 2 in the first column and only tile 3 in the second.

Example 2

Input:

```
3 2
0 0 1 1
1 1 0 0
1 1
```

Output:

```
YES
1 1
2 2
1 1
```

Here $n = 3$, so $w = \lceil \log_2 3 \rceil = 2$.

Tile 1 has both top corners 0 and both bottom corners 1; tile 2 is the opposite, with both top corners 1 and both bottom corners 0. The first row holds tile 1 twice, and the common side matches because $1.TR = 0 = 1.TL$ and $1.BR = 1 = 1.BL$.

The bottom boundary of the first row reads $1, 1, 1$, so every tile of the second row must have both top corners equal to 1, and only tile 2 fits. The bottom boundary of the second row reads $0, 0, 0$, so the third row must again consist of tile 1. This is the only possible completion.

Example 3

Input:

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

Output:

```
NO
```

Here $n = 2$, so $w = \lceil \log_2 2 \rceil = 1$ and the mosaic is one tile wide.

The only tile type has both top corners 0 and both bottom corners 1. A tile placed below the first row would need both top corners equal to 1, and no tile type has that, so the second row cannot be filled and the answer is NO. A row that is valid on its own is not enough: every row must also fit the row above it.

Example 4

Input:

```
5 2
0 0 1 1
1 1 0 0
1 1 1
```

Output:

```
YES
1 1 1
2 2 2
1 1 1
2 2 2
1 1 1
```

Here $n = 5$, so $w = \lceil \log_2 5 \rceil = 3$: the logarithm is rounded up, so five rows need three columns.

The tiles are the same as in Example 2 and the first row holds tile 1 three times. As before, each row forces the next one, so the rows alternate between tile 1 and tile 2 and the fifth row is tile 1 again.

Example 5

Input:

```
5 3
0 0 0 0
0 0 0 1
0 0 1 0
1 1 1
```

Output:

```
YES
1 1 1
1 1 1
1 1 1
1 1 1
1 1 1
```

Here $n = 5$ and $w = 3$. Tile 1 has all four corners 0, tile 2 has only $BR = 1$, and tile 3 has only $BL = 1$.

Every tile type has both top corners equal to 0, so a tile can be placed only below a tile whose bottom corners are both 0, that is, below tile 1. A row that contains tile 2 or tile 3 has a 1 on its bottom boundary, so nothing can be placed below it, and such a row can only be the last one. The first four rows must therefore consist of tile 1 only.

The last row has more freedom: any row whose own sides match fits there, for example $1\ 1\ 2$, $1\ 2\ 3$ or $3\ 2\ 3$, which gives 8 valid completions in total. Filling every row with tile 1 is one of them, and any of the other seven would be accepted as well. A solution that puts tile 2 or tile 3 into an earlier row reaches a dead end, so picking each row greedily does not work.