Rooks

You are given an $N \times M$ matrix A where each element from 1 to $N \times M$ appears exactly once. You are given a rook that is on the cell that contains the element with value 1. You are also given a K.

The rook can jump from a cell (r, c) to a cell (r', c') if both of the following conditions hold:

- $(r, c) \neq (r', c')$, so we have to move to a different cell,
- either $r = r'$ or $c = c'$, so we move only in one row or one column,
- $0 < A[r'][c'] - A[r][c] \leq K$.

Find for each cell of the matrix the minimum number of moves it takes the rook to reach it from the cell containing 1, or if it is not reachable at all.

Implementation Details

You need to implement the following function:

```
int32[][] calculate_moves(int32[][] A, int32 K)
```

- A: array of length N of arrays of length M describing the board.
- K: the movement constraint.
- The function should return an array of length N of arrays of length M containing the minimum number of moves needed to reach that cell from cell 1. If a cell is unreachable, the value for that cell should be equal to -1.

Constraints

- $1 \leq N, M \leq 2\,500$
- $1 \leq K \leq N \cdot M$
- $1 \leq A[i][j] \leq N \cdot M$
- Each integer from 1 to $N \cdot M$ appears in A exactly once.

Subtasks

Subtask	Score	Additional Constraints
1	9	$N = 1$
2	15	$N, M \leq 100$
3	11	$K = 1$
4	19	$K = N \cdot M$
5	15	$N, M \leq 500$
6	31	No further constraints.

Examples

Example 1

Consider the following call:

```
calculate_moves([[8, 2, 4, 20, 5],
                 [14, 13, 1, 19, 7],
                 [15, 18, 12, 6, 11],
                 [10, 9, 3, 16, 17]], 5)
```

We have $N = 4, M = 5, K = 5$.

The cell containing 1, which we start from, is $A[1][2]$. Therefore in the array R returned by `calculate_moves`, value $R[1][2]$ should be equal to 0, as we don't need to make any moves to reach that cell.

To reach $A[3][0] = 10$, we can go from $A[1][2] = 1$ to $A[0][2] = 4$ (as they are in the same column, and $4 - 1 = 3 \leq 5$), then from $A[0][2] = 4$ to $A[0][0] = 8$ (as they are in the same row, and $8 - 4 = 4 \leq 5$), then finally from $A[0][0]$ to $A[4][0]$ (as they are in the same column and $10 - 8 = 2$). Therefore in the array R, value $R[4][0]$ should be equal to 3.

Notice that it is not possible to reach $A[0][1] = 2$ in any way, so the value $R[0][1]$ should be equal to -1.

The procedure should return:

```
[[2, -1, 1, 6, 2],
 [4, -1, 0, 5, 3],
 [4, 5, 5, -1, 4],
 [3, -1, 1, -1, -1]]
```

Sample Grader

Input format:

```
N M K
A[0][0] A[0][1] ... A[0][M-1]
A[1][0] A[1][1] ... A[1][M-1]
...
A[N-1][0] A[N-1][1] ... A[N-1][M-1]
```

Output format:

```
R[0][0] R[0][1] ... R[0][M-1]
R[1][0] R[1][1] ... R[1][M-1]
...
R[N-1][0] R[N-1][1] ... R[N-1][M-1]
```

Here, R is the array returned by `calculate_moves`.