# Repainting the Table

Time limit: 1.5 seconds
Memory limit: 512 megabytes

**This problem can only be solved using C++ or Python3.**

Zakhar loves to play chess. He has grown bored of the usual black-and-white chessboards, so he thought about how to diversify the game.

A chessboard is a square table with $n$ rows and $n$ columns. We number the rows with integers from 0 to $n-1$ from top to bottom, and the columns with integers from 0 to $n-1$ from left to right. Zakhar calls the board $k$-colored if a cell is painted in a color equal to the remainder of the division of the number of the diagonal going up-right from the cell by $k$. The diagonals are numbered with consecutive integers starting from zero, beginning at the top left corner of the table. In other words, the number of the diagonal to which the cell $(i, j)$ belongs is equal to $(i + j) \mod k$.

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 2 | 0 | 1 |
| 1 | 1 | 2 | 0 | 1 | 2 |
| 2 | 2 | 0 | 1 | 2 | 0 |
| 3 | 0 | 1 | 2 | 0 | 1 |
| 4 | 1 | 2 | 0 | 1 | 2 |

Figure of a $k$-colored board for $n = 5, k = 3$.

Zakhar found a board at his dacha and now wants to obtain a $k$-colored board for some positive integer value of $k$. Some $t$ cells of the table are already painted, with cell $(r[i], c[i])$ painted in color $v[i]$, and Zakhar may need to repaint them to the correct color. Unpainted cells do not concern Zakhar because applying paint is easier than repainting an already existing color.

Determine the minimum number of repaints that Zakhar will need to obtain a $k$-colored board for some positive integer value of $k$.

## Solution Format

This is an unusual problem. It has a testing format with a grader, where you only need to implement the function `solve` in your solution. This function will be called by the testing program of the jury (the grader), and the returned value of the function will be accepted as the solution to the problem.

In particular, this means that there should be **no input or output** in the code you submit. If you code in C++, your code **must not** contain a function `main`. If necessary, you can implement any number of helper functions, structures, classes, and global variables, but all the code of your solution must be in one file.

For a solution in C++, you must implement the following function:

```
int solve(int n, int t, std::vector<int> r, std::vector<int> c, std::vector<int> v)
```

For a solution in Python3 or Pypy3, you must implement the following function:

```
def solve(n, t, r, c, v):
```

Here $n$ and $t$ are integers, and $r$, $c$, and $v$ are lists of integers of length $t$.

Note that the jury does not guarantee the possibility of achieving full points on Python3 or Pypy3.

In both languages, the function `solve` takes the following arguments:

- $n$ ($1 \le n \le 10^6$) — the number of rows and columns in the table.

- $t$ ($1 \le t \le 10^6$) — the number of painted cells.
- $r$ ($0 \le r[i] \le n-1$) — an array of size $t$ consisting of the row numbers of the painted cells.
- $c$ ($0 \le c[i] \le n-1$) — an array of size $t$ consisting of the column numbers of the painted cells.
- $v$ ($0 \le v[i] \le 10^9$) — an array of size $t$ consisting of the colors of the painted cells.

**All painted cells are numbered in 0-indexing**, so for any $i$ ($0 \le i < t$), the $i$-th painted cell is located at the intersection of the $r[i]$-th row and the $c[i]$-th column and has color $v[i]$. It is guaranteed that all pairs $(r[i], c[i])$ are distinct.

The function `solve` should return a single integer — the minimum number of originally painted cells that will need to be repainted to make the board $k$-colored.

It is guaranteed that the function `solve` will be called exactly once during the program's execution.

## Testing

You are provided with template files where you can write your solutions: `table.cpp` and `table.py`. Also, in C++, you have been provided with a header file `table.h` containing the definition of the function `solve`.

For your convenience, graders are provided — files `grader.cpp` and `grader.py`. In these files, reading the input from the standard input, the execution of the function `solve`, and the output of the returned value of the function `solve` to the standard output are implemented. In the testing system, these grader files may differ.

To compile your C++ code written in the file `table.cpp`, use the command

```
g++ -std=c++20 grader.cpp table.cpp -o grader
```

After executing this command, an executable file named `grader` or `grader.exe` will be created, depending on your operating system, which can be run to input a test in the specified format.

To run your Python code written in the file `table.py`, use the command `python3 grader.py`, and then you can input a test in the specified format.

If compilation via commands causes you difficulties, for local testing, you can copy the implementation of input and output from the files `grader` into the files `table` and run the files `table.cpp` or `table.py`. However, before submitting your solution to the testing system, you will need to remove these input and output implementations (in particular, you will need to remove the function `main` if you code in C++).

## Input

The grader provided to you reads input data in the following format:

The first line contains two integers $n$ and $t$ ($1 \le n, t \le 10^6$) — the number of rows and columns in the table, as well as the number of already painted cells.

The second line contains $t$ integers $r[i]$ ($0 \le r[i] \le n-1$) — the row numbers of the painted cells.

The third line contains $t$ integers $c[i]$ ($0 \le c[i] \le n-1$) — the column numbers of the painted cells.

The fourth line contains $t$ integers $v[i]$ ($0 \le v[i] \le 10^9$) — the colors of the already painted cells.

## Output

The grader outputs the returned value of the function `solve`.

## Example

| test | answer |
|---|---|
| 5 4<br>0 0 1 4<br>0 1 0 0<br>2 1 4 1 | 2 |

## Note

The example shows a sample input for the grader provided to you. Initially, four cells are painted: $(0, 0)$ in color 2, $(0, 1)$ in color 1, $(1, 0)$ in color 4, and $(4, 0)$ in color 1. It is optimal to choose $k = 3$, and the drawing of the table is provided in the problem statement.

Two cells need to be repainted: $(0, 0)$ from color 2 to color 0, and $(1, 0)$ from color 4 to color 1.

## Scoring

The tests for this problem consist of four groups. Points for each group are awarded only if all tests of the group and all tests of some of the previous groups are passed.

| Group | Points | Additional constraints | | Required groups | Comment |
|---|---|---|---|---|---|
| | | $n$ | $t$ | | |
| 0 | 0 | – | – | – | Samples. |
| 1 | 17 | $n \leq 100$ | $t \leq 100$ | 0 | |
| 2 | 21 | $n \leq 1\,000$ | – | 0, 1 | |
| 3 | 34 | $n \leq 100\,000$ | $t \leq 100\,000$ | 0, 1 | |
| 4 | 28 | – | – | 0 – 3 | |