Task: ROW
Equivalent programs

Olimpiada Informatyczna

XXIII OI, Stage III, Day 0. Source file `row.*` Available memory: 128 MB. *12.04.2016*

Byteasar got a new computer, and is learning to program it. A program is a sequence of commands, of which there are k kinds, numbered from 1 to k for convenience. Some pairs of commands are *commutative*, i.e., if they are next to each other in the program (in any order), then swapping them results in an *equivalent* program with the same behavior. The remaining pairs of instructions that do not have this property are called *noncommutative*. So far, Byteasar wrote two programs consisting of n commands each, and is wondering if they are equivalent. Help him out!

Input

In the first line of the standard input, there are three integers n, k, and m, separated by single spaces, which denote respectively the common number of commands of the programs, the number of command types of the computer, and the number of noncommutative pairs of instructions.

The m lines that follow specify these pairs: each contains two integers a and b ($1 \leq a < b \leq k$), separated by a single space, such that the pair of instructions no. a and b is noncommutative. You may assume that every pair of instructions appears on the input at most once.

Finally, the last two lines summarize the programs: Each of these lines contains a sequence of n integers $c_1, c_2, \ldots, c_n$ ($1 \leq c_i \leq k$), separated by single spaces, which are the numbers of successive commands of the respective program.

Output

In the one and only line of the standard output, a single word should be printed: `TAK` (Polish for *yes*) if the programs given on input are equivalent and `NIE` (Polish for *no*) if they are not.

Example

For the input data:

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

the correct result is:

```
TAK
```

whereas for the following input data:

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

the correct result is:

```
NIE
```

Explanation of the first example: The following sequence of swaps transforms the first program into the second: (2,3), (4,5), (3,4), where $(i, i+1)$ stands for swapping the i-th and the $(i+1)$-th command in the current program, i.e., after previous swaps.

Sample Grading Tests:

- `1ocen:` $n = 50$, $k = 50$, $m = 1$; programs: $(1, 2, \ldots, 49, 50)$ and $(50, 49, \ldots, 2, 1)$; answer: `NIE`.
- `2ocen:` $n = 99\,999$, $k = 3$, $m = 1$; noncommutative commands: 1 and 2, programs: $(1, 2, 3, 1, 2, 3, \ldots, 1, 2, 3)$ and $(3, 1, 2, 3, 1, 2, \ldots, 3, 1, 2)$; answer: `TAK`.
- `3ocen:` $n = 100\,000$, $k = 1000$, $m = 50\,000$; programs: $(13, 13, 13, \ldots, 13)$ and $(37, 37, 37, \ldots, 37)$; answer: `NIE`.

Grading

The set of tests consists of the following subsets. Within each subset, there may be several test groups. All test satisfy the following conditions: $1 \leq n \leq 100\,000$, $1 \leq k \leq 1000$, $0 \leq m \leq 50\,000$.

Subset	**Property**	**Score**
1	$n \leq 5$	5
2	$k \leq 2$	5
3	$n \leq 1000$	25
4	no additional conditions	65