Codeforces Round #137 (Div. 2)

Problema B: *Cosmic Tables*

Prof. Edson Alves – UnB/FGA

Codeforces 222B – Cosmic Tables

The Free Meteor Association (FMA) has got a problem: as meteors are moving, the Universal Cosmic Descriptive Humorous Program (UCDHP) needs to add a special module that would analyze this movement.

UCDHP stores some secret information about meteors as an $n \times m$ table with integers in its cells. The order of meteors in the Universe is changing. That's why the main UCDHP module receives the following queries:

- *The query to swap two table rows;*
- *The query to swap two table columns;*
- *The query to obtain a secret number in a particular table cell.*

As the main UCDHP module is critical, writing the functional of working with the table has been commissioned to you.

Entrada e saída

Input

The first line contains three space-separated integers n, m and k $(1 \le n, m \le 1000, 1 \le k \le 500000)$ – the number of table columns and rows and the number of queries, correspondingly.

Next n lines contain m space-separated numbers each – the initial state of the table. Each number p in the table is an integer and satisfies the inequality $0 \le p \le 10^6$.

Next k lines contain queries in the format "$s_i x_i y_i$", where s_i is one of the characters "`c`", "`r`" or "`g`", and x_i, y_i are two integers.

- *If $s_i =$ "`c`", then the current query is the query to swap columns with indexes x_i and y_i $(1 \le x, y \le m, x \ne y)$;*
- *If $s_i =$ "`r`", then the current query is the query to swap rows with indexes x_i and y_i $(1 \le x, y \le n, x \ne y)$;*
- *If $s_i =$ "`g`", then the current query is the query to obtain the number that located in the x_i-th row and in the y_i-th column $(1 \le x \le n, 1 \le y \le m)$.*

Entrada e saída

The table rows are considered to be indexed from top to bottom from 1 to n, and the table columns – from left to right from 1 to m.

Output

For each query to obtain a number ($s_i =$ "g") print the required number. Print the answers to the queries in the order of the queries in the input.

Exemplos de entrada e saída

Entrada	Saída
3 3 5 1 2 3 4 5 6 7 8 9 g 3 2 r 3 2 c 2 3 g 2 2 g 3 2	8 9 6

Solução com complexidade $O(k)$

- Trocar efetivamente os elementos de duas linhas de lugar tem complexidade $O(m)$
- De forma equivalente, a troca de duas colunas tem complexidade $O(n)$
- Assim, no pior caso o algoritmo teria complexidade $O(k \times \max(n, m))$, o que extrapolaria o limite de tempo, dadas as restrições do problema
- A solução do problema depende, portanto, de um processamento mais eficiente das consultas que envolvem trocas de linhas e de colunas
- De fato, estas consultas podem ser respondidas em $O(1)$

Solução com complexidade $O(k)$

- A cada troca de linhas ou de colunas, a nova matriz obtida tem as mesmas linhas e colunas da matriz A, porém em ordem distinta
- A ideia, portanto, é utilizar duas permutações, denominadas `rs` e `cs`, que registrem a ordem em que as linhas e as colunas da matriz A foram rearranjadas até uma consulta de elemento
- Inicialmente, ambas permutações são identidades, isto é, `rs[i] = i` e `cs[j] = j` para $i \in [1, n]$ e $j \in [1, m]$
- Com estas permutações, a troca de linhas ou de colunas é feita apenas pela troca dos elementos nas respectivas permutações, mantendo a matriz A inalterada
- Nas consultas de elementos, basta utilizar os índices registrados nas permutações para localizar o elemento correto na matriz A

Solução com complexidade $O(k)$

```
7  vector<int>
8  solve(const vector<vector<int>>& A, int N, int M, const vector<Query>& qs)
9  {
10     vector<int> rs(N + 1), cs(M + 1), ans;
11
12     iota(rs.begin(), rs.end(), 0);
13     iota(cs.begin(), cs.end(), 0);
14
15     for (const auto& q : qs)
16     {
17         switch (q.c.front()) {
18         case 'c':
19             swap(cs[q.x], cs[q.y]);
20             break;
```

Solução com complexidade $O(k)$

```
22          case 'r':
23              swap(rs[q.x], rs[q.y]);
24              break;
25
26          default:
27              ans.push_back(A[rs[q.x]][cs[q.y]]);
28          }
29      }
30
31      return ans;
32  }
```