Посадочные места

Вы планируете проводить международное соревнование по программированию в прямоугольном зале, в котором HW посадочных мест, они организованы в виде прямоугольника, содержащего H строк и W столбцов. Строки пронумерованы от 0 до $H-1$, а столбцы пронумерованы от 0 до $W-1$. Посадочное место в строке r и столбце c обозначается как (r, c). Вы пригласили HW участников, пронумерованных от 0 до $HW-1$. Вы также подготовили распределение участников по посадочным местам, в соответствии с которым участник с номером i $(0 \leq i \leq HW-1)$ занимает посадочное место (R_i, C_i). В соответствии с распределением на каждом посадочном месте размещается ровно один участник.

Множество S посадочных мест в зале называется **прямоугольником**, если для некоторых целых чисел r_1, r_2, c_1 и c_2 выполнены следующие условия:

- $0 \leq r_1 \leq r_2 \leq H-1$.
- $0 \leq c_1 \leq c_2 \leq W-1$.
- S состоит в точности из всех посадочных мест (r, c), для которых $r_1 \leq r \leq r_2$ и $c_1 \leq c \leq c_2$.

Прямоугольник, состоящий из k посадочных мест $(1 \leq k \leq HW)$, называется **красивым**, если на местах из этого прямоугольника размещаются в точности участники с номерами от 0 до $k-1$. **Красотой** распределения участников по посадочным местам называется количество красивых прямоугольников для данного распределения.

После подготовки вашего распределения участников по местам, вы выполняете несколько запросов обмена двух участников посадочными местами. А именно, дано Q таких запросов, пронумерованных от 0 до $Q-1$ в хронологическом порядке. Запрос с номером j $(0 \leq j \leq Q-1)$ состоит том, что участники с номерами A_j и B_j меняются посадочными местами. Вы немедленно обрабатываете каждый запрос и обновляете распределение участников по посадочным местам. После каждого обновления вам требуется вычислить красоту текущего распределения участников по посадочным местам.

Детали реализации

Вам необходимо реализовать следующие процедуры и функции:

```
give_initial_chart(int H, int W, int[] R, int[] C)
```

- H, W: количество строк и количество столбцов.
- R, C: массивы длины HW, задающие исходное распределение участников по посадочным местам.
- Процедура будет вызвана ровно один раз до любого вызова swap_seats.

```
int swap_seats(int a, int b)
```

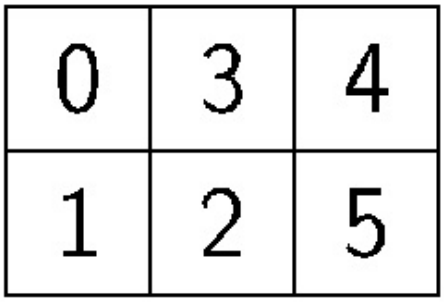
- Эта функция описывает запрос обмена двух участников посадочными местами.
- a, b: участники, которые меняются посадочными местами.
- Эта функция будет вызвана Q раз.
- Функция должна вернуть красоту распределения участников по посадочным местам после обмена.

Пример

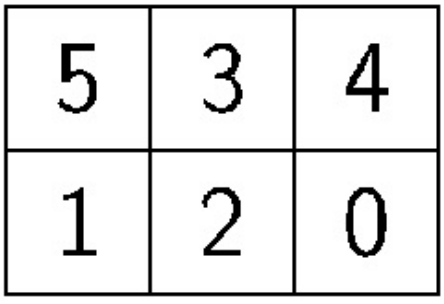
Пусть $H = 2$, $W = 3$, $R = [0, 1, 1, 0, 0, 1]$, $C = [0, 0, 1, 1, 2, 2]$ и $Q = 2$.

Проверяющий модуль (grader) вызывает `give_initial_chart(2, 3, [0, 1, 1, 0, 0, 1], [0, 0, 1, 1, 2, 2])`.

Исходно распределение участников по посадочным местам выглядит следующим образом.



Пусть затем проверяющий модуль вызывает `swap_seats(0, 5)`. После запроса 0 распределение участников по посадочным местам выглядит следующим образом.



Множества посадочных мест, соответствующие множествам участников $\{0\}$, $\{0, 1, 2\}$ и $\{0, 1, 2, 3, 4, 5\}$, являются красивыми прямоугольниками. Следовательно красота этого распределения участников по посадочным местам равна 3, и функция `swap_seats` должна вернуть 3.

Пусть теперь проверяющий модуль снова вызывает `swap_seats(0, 5)`. После запроса 1 распределение участников по посадочным местам возвращается к исходному состоянию. Множества посадочных мест, соответствующие множествам участников $\{0\}$, $\{0, 1\}$, $\{0, 1, 2, 3\}$ и $\{0, 1, 2, 3, 4, 5\}$, являются красивыми прямоугольниками. Таким образом, красота этого распределения участников по посадочным местам равна 4, и функция `swap_seats` должна вернуть 4.

Файлы `sample-01-in.txt` и `sample-01-out.txt` в приложенном zip-архиве соответствуют этому примеру. В архиве есть также другие примеры ввода и вывода.

Ограничения

- $1 \leq H$
- $1 \leq W$
- $HW \leq 1\,000\,000$
- $0 \leq R_i \leq H - 1$ $(0 \leq i \leq HW - 1)$
- $0 \leq C_i \leq W - 1$ $(0 \leq i \leq HW - 1)$
- $(R_i, C_i) \neq (R_j, C_j)$ $(0 \leq i < j \leq HW - 1)$
- $1 \leq Q \leq 50\,000$
- $0 \leq a \leq HW - 1$ для всех вызовов `swap_seats`
- $0 \leq b \leq HW - 1$ для всех вызовов `swap_seats`
- $a \neq b$ для всех вызовов `swap_seats`

Подзадачи

1. (5 баллов) $HW \leq 100$, $Q \leq 5\,000$
2. (6 баллов) $HW \leq 10\,000$, $Q \leq 5\,000$
3. (20 баллов) $H \leq 1\,000$, $W \leq 1\,000$, $Q \leq 5\,000$
4. (6 баллов) $Q \leq 5\,000$, $|a - b| \leq 10\,000$ для всех вызовов `swap_seats`
5. (33 балла) $H = 1$
6. (30 баллов) Нет дополнительных ограничений

Пример проверяющего модуля

Пример проверяющего модуля считывает входные данные в следующем формате:

- строка 1: H W Q

- строка $2 + i$ ($0 \leq i \leq HW - 1$): R_i C_i
- строка $2 + HW + j$ ($0 \leq j \leq Q - 1$): A_j B_j

Здесь A_j и B_j — параметры вызова `swap_seats` для запроса j.

Пример проверяющего модуля выводит ваши ответы в следующем формате:

- строка $1 + j$ ($0 \leq j \leq Q - 1$) : возвращаемое значение `swap_seats` для запроса j