Seats

Vas a organizar un concurso internacional de programación en una sala rectangular que contiene HW asientos acomodados en H renglones y W columnas. Los renglones están numerados de 0 a $H-1$ y las columnas están numeradas de 0 a $W-1$. Un asiento en el renglón r y la columna c está denotado como (r, c). Invitaste a HW concursantes numerados de 0 a $HW-1$. Creaste un *acomodo de asientos* de tal manera que a cada concursante i ($0 \leq i \leq HW-1$) tiene asignado el asiento (R_i, C_i). El acomodo asigna exactamente un concursante en cada asiento.

Un conjunto de asientos S dentro del salón se dice que es **rectangular** si existen enteros r_1, r_2, c_1, y c_2 tal que satisfacen las siguientes condiciones.

- $0 \leq r_1 \leq r_2 \leq H-1$.
- $0 \leq c_1 \leq c_2 \leq W-1$.
- El conjunto rectangular S es exactamente el conjunto de asientos (r, c) tal que $r_1 \leq r \leq r_2$ y $c_1 \leq c \leq c_2$.

Un conjunto rectangular de k ($1 \leq k \leq HW$) asientos es considerado **bello** si los concursantes asignados a los k asientos del conjunto rectangular tienen numeración de 0 hasta $k-1$. La **belleza** de un *acomodo de asientos* es el número de conjuntos rectangulares bellos que contiene el *acomodo de asientos*.

Después de preparar el *acomodo de asientos* del concurso, recibiste varias peticiones donde se tiene que intercambiar los asientos de un par concursantes. Es decir, existen Q peticiones numeradas de 0 a $Q-1$ en orden cronológico. La petición j ($0 \leq j \leq Q-1$), especifica que tienes que intercambiar los asientos asignados a los concursantes A_j y B_j. Debes aceptar cada petición inmediatamente y actualizar el *acomodo de asientos*. Después de cada actualización, tu tarea es calcular la belleza del *acomodo de asientos*.

Detalles de implementación

Debes implementar el siguiente procedimiento y función:

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

- `H`, `W`: el número de renglones y el número de columnas.
- `R`, `C`: arreglo de longitud HW que representa el *acomodo de asientos* inicial.

- Este procedimiento es llamado exactamente una vez y antes de cualquier llamada a `swap_seats`.

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

- Esta función recibe una petición para intercambiar dos asientos.
- `a`, `b`: índices de los concursantes a los que se les debe intercambiar sus asientos.
- Esta función es llamada exactamente Q veces.
- Esta función debe regresar la belleza del acomodo actual después de realizar el intercambio de asientos.

Ejemplo

Sea $H = 2$, $W = 3$, $R = [0, 1, 1, 0, 0, 1]$, $C = [0, 0, 1, 1, 2, 2]$, y $Q = 2$.

El evaluador manda llamar primero `give_initial_chart(2, 3, [0, 1, 1, 0, 0, 1], [0, 0, 1, 1, 2, 2])`.

Inicialmente, el *acomodo de asientos* es el siguiente.

0	3	4
1	2	5

Supongamos que el evaluador manda llamar `swap_seats(0, 5)`. Después de la petición 0, el *acomodo de asientos* queda de la siguiente manera:

5	3	4
1	2	0

Los conjuntos de asientos que corresponden a los concursantes $\{0\}$, $\{0, 1, 2\}$, y $\{0, 1, 2, 3, 4, 5\}$ son rectangulares y bellos. Por lo tanto, la belleza del *acomodo de asientos* es 3 y la funcioón `swap_seats` debe regresar 3.

Supongamos que el evaluador llama `swap_seats(0, 5)` de nuevo. Después de la petición 1, el *acomodo de asientos* vuelve a su estado original. Los conjuntos de

asientos que corresponden a los concursantes $\{0\}$, $\{0, 1\}$, $\{0, 1, 2, 3\}$, y $\{0, 1, 2, 3, 4, 5\}$ son rectangulares y bellos. Por lo tanto, la belleza del acomodo de asientos es 4 y `swap_seats` debe regresar 4.

Los archivos `sample-01-in.txt` y `sample-01-out.txt` en el paquete zip adjuntado corresponden a este ejemplo. Otros ejemplos de entrada/salida también están disponibles en el paquete.

Restricciones

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

Subtareas

1. (5 puntos) $HW \leq 100$, $Q \leq 5\,000$
2. (6 puntos) $HW \leq 10\,000$, $Q \leq 5\,000$
3. (20 puntos) $H \leq 1\,000$, $W \leq 1\,000$, $Q \leq 5\,000$
4. (6 puntos) $Q \leq 5\,000$, $|a - b| \leq 10\,000$ para cualqueir llamada a `swap_seats`
5. (33 puntos) $H = 1$
6. (30 puntos) Sin restricciones adicionales.

Evaluador de ejemplo

El evaluador de ejemplo lee la entrada con el siguiente formato:

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

Aquí, A_j y B_j son los parámetros de `swap_seats` en la petición j.

El evaluador de ejemplo imprime tus respuestas en el siguiente formato:

- línea $1 + j$ $(0 \leq j \leq Q - 1)$: el valor retornado por `swap_seats` para la petición j