Seats

Vous comptez organiser une compétition internationale de programmation dans un hall rectangulaire qui a HW sièges rangés en H lignes et W colonnes.

Les lignes sont numérotées de 0 à $H-1$ et les colonnes sont numérotées de 0 à $W-1$. Le siège situé dans la ligne r et la colonne c est noté (r, c). Vous avez convoqué HW participants numérotés de 0 à $HW-1$. Vous avez aussi une configuration de places assises, affectant le participant i ($0 \leq i \leq HW-1$) au siège (R_i, C_i). La configuration est telle que chaque siège soit affecté à exactement un participant.

Un ensemble de sièges dans le hall S est dit **rectangulaire** s'il existe r_1, r_2, c_1 et c_2 vérifiant les conditions suivantes :

- $0 \leq r_1 \leq r_2 \leq H-1$.
- $0 \leq c_1 \leq c_2 \leq W-1$.
- S est exactement l'ensemble de tous les sièges (r, c) tel que $r_1 \leq r \leq r_2$ et $c_1 \leq c \leq c_2$.

Un ensemble rectangulaire de k sièges ($1 \leq k \leq HW$) est considéré **beau** si les participants affectés aux sièges de cet ensemble sont numérotés de 0 à $k-1$. La **beauté** d'une configuration de places assises est le nombre de beaux ensembles rectangulaires dans cette configuration.

Après avoir préparé votre configuration de places assises, vous recevez plusieurs requêtes pour permuter deux sièges assignés à deux participants. Plus précisément, il y a Q requêtes numérotées, par ordre chronologique, de 0 à $Q-1$. La requête j avec ($0 \leq j \leq Q-1$) consiste à permuter les sièges affectés aux participants A_j et B_j. Vous acceptez immédiatement la requête et vous mettez à jour la configuration des places assises. Après chaque mise à jour, votre objectif est de calculer la beauté de la configuration de places assises actuelle.

Détails d'implémentation

Vous devez implémenter la procédure et la fonction suivantes :

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

- `H`, `W` : le nombre de lignes et le nombre de colonnes.
- `R`, `C` : tableaux de taille HW représentant la configuration initiale des places

assises.
- Cette procédure est appelée exactement une fois avant tout appel à `swap_seats`.

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

- Cette fonction décrit une requête de permutation de deux sièges.
- `a`, `b` : les participants dont les sièges seront permutés.
- Cette fonction sera appelée Q fois.
- Cette fonction doit retourner la beauté d'une configuration de places assises après la permutation.

Exemple

Pour $H = 2$, $W = 3$, $R = [0, 1, 1, 0, 0, 1]$, $C = [0, 0, 1, 1, 2, 2]$, et $Q = 2$.

La système d'évaluation commence par appeler `give_initial_chart(2, 3, [0, 1, 1, 0, 0, 1], [0, 0, 1, 1, 2, 2])`.

Ainsi, au départ, la configuration des places assises est la suivante :

0	3	4
1	2	5

On suppose que le système d'évaluation appellera `swap_seats(0, 5)`. Après la requête `0`, la configuration des places assises devient :

5	3	4
1	2	0

Les ensembles de sièges correspondant aux participants $\{0\}$, $\{0, 1, 2\}$, et $\{0, 1, 2, 3, 4, 5\}$ sont rectangulaires et beaux. Ainsi, la beauté de cette configuration de places assises est de 3 et `swap_seats` doit retourner 3.

Si par exemple l'évaluateur (grader) appelle `swap_seats(0, 5)` une autre fois. Après la requête 1, la configuration de places assises revient à l'état initial. Les ensembles de

sièges correspondant aux participants $\{0\}$, $\{0, 1\}$, $\{0, 1, 2, 3\}$, et $\{0, 1, 2, 3, 4, 5\}$ sont rectangulaires et beaux. Ainsi, la beauté de cette configuration est de 4, et `swap_seats` devra retourner 4.

Les fichiers `sample-01-in.txt` et `sample-01-out.txt` dans le package compressé joint correspondent à cet exemple. D'autres exemples d'inputs/outputs sont également disponibles dans le package.

Contraintes

- $1 \leq H$
- $1 \leq W$
- $HW \leq 1\,000\,000$
- $0 \leq R_i \leq H - 1$ ($0 \leq i \leq HW - 1$)
- $0 \leq C_i \leq W - 1$ ($0 \leq i \leq HW - 1$)
- $(R_i, C_i) \neq (R_j, C_j)$ ($0 \leq i < j \leq HW - 1$)
- $1 \leq Q \leq 50\,000$
- $0 \leq a \leq HW - 1$ pour chaque appel à `swap_seats`
- $0 \leq b \leq HW - 1$ pour chaque appel à `swap_seats`
- $a \neq b$ pour chaque appel à `swap_seats`

Sous-tâches

1. (5 points) $HW \leq 100$, $Q \leq 5\,000$
2. (6 points) $HW \leq 10\,000$, $Q \leq 5\,000$
3. (20 points) $H \leq 1\,000$, $W \leq 1\,000$, $Q \leq 5\,000$
4. (6 points) $Q \leq 5\,000$, $|a - b| \leq 10\,000$ pour chaque appel à `swap_seats`
5. (33 points) $H = 1$
6. (30 points) Sans autres contraintes

Evaluateur d'exemples (grader)

L'évaluateur d'exemples (grader) lit l'entrée (input) dans le format suivant :

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

Ici, A_j et B_j sont les paramètres de l'appel à `swap_seats` pour la requête j.

L'évaluateur d'exemples (grader) renvoie (output) vos réponses dans le format suivant :

- ligne $1 + j$ ($0 \leq j \leq Q - 1$) : la valeur retournée par `swap_seats` pour la requête j