Des "mex" égaux

Il est bien connu parmi les nobles roumains que la beauté d'un tableau d'entiers $a[0], a[1], a[2], \ldots a[m-1]$ est la quantité d'entiers positifs k pour lesquels on peut diviser le tableau en k sous-tableaux disjoints (séquences d'éléments consécutifs) tels que chaque élément soit contenu dans exactement un sous-tableau et que tous les sous-tableaux aient le même plus petit élément exclu (en anglais, "minimum excluded element", d'où l'abréviation "mex"). Le plus petit élément exclu (mex) d'un tableau d'entiers est le plus petit entier strictement positif **(supérieur à** 0**)** qui n'apparaît pas dans le tableau.

On vous donne un tableau d'entiers $v[0], v[1], \ldots v[n-1]$ ainsi que q requêtes de la forme (l_i, r_i), où $0 \leq l_i \leq r_i < n$ pour tout $0 \leq i < q$.

Pour chaque requête, vous devez déterminer la beauté du tableau $v[l_i], v[l_i+1], \ldots, v[r_i]$.

Détails de l'implémentation

Vous devez implémenter la procédure suivante :

```
std::vector<int> solve(
    int n, std::vector<int>& v,
    int q, std::vector<std::pair<int, int>>& queries);
```

- n : la taille du tableau d'entiers
- v : un tableau de longueur n, le tableau initial
- q : le nombre de requêtes
- $queries$: un tableau de longueur q décrivant les requêtes
- Cette procédure doit retourner un vecteur de q entiers contenant la réponse à chaque requête.
- Cette procédure est appelée exactement une fois pour chaque cas de test.

Limites

- $1 \leq n \leq 600\,000$
- $1 \leq q \leq 600\,000$
- $1 \leq v[i] \leq 400\,000$ pour tout $0 \leq i < n$
- $0 \leq l_i \leq r_i < n$ pour tout $0 \leq i < q$

Sous-tâches

1. (4 points) $1 \leq n \leq 10$, $1 \leq q \leq 100$
2. (6 points) $1 \leq n, q \leq 100$
3. (17 points) $1 \leq n, q \leq 1\,000$
4. (10 points) $1 \leq n, q \leq 100\,000$ et $1 \leq v[i] \leq 2$ pour tout $0 \leq i < n$
5. (30 points) $1 \leq n, q \leq 75\,000$
6. (33 points) Aucune contrainte supplémentaire.

Exemples

Exemple 1

Considérez l'appel suivant :

```
solve(10, {1, 1, 2, 2, 3, 3, 1, 2, 3, 4}, 2, {{0, 5}, {0, 8}})
```

Dans cet exemple, $n = 10$ et il y a 2 requêtes pour lesquelles :

- $l_0 = 0$ et $r_0 = 5$
- $l_1 = 0$ et $r_1 = 8$

Pour la première requête, on peut diviser l'intervalle en un seul sous-tableau, qui va de la position `0` à la position `5`.

Dans la deuxième requête, k peut être égal à 1 ou à 2.
Une possibilité de division en un seul sous-tableau consiste à choisir le sous-tableau allant de la position 0 à la position 8. Une possibilité de division en deux sous-tableaux consiste à choisir le sous-tableau allant de la position 0 à la position 5, et celui allant de la position 6 à la position 8.

La réponse à la première requête est 1 et celle à la deuxième est 2, donc l'appel à `solve` retournera `{1, 2}`.

Grader d'exemple

Le grader d'exemple lit les entrées dans le format suivant :

- ligne 1 : n q
- ligne 2 : $v[0]$ $v[1]$ $\ldots$ $v[n-1]$
- ligne $3 + i$: l_i r_i pour tout $0 \leq i < q$

et affiche q lignes, correspondant au résultat de l'appel à la fonction `solve` avec les paramètres donnés.