Equalmex

Unter rumänischen Adligen ist es allgemein bekannt, dass die Schönheit eines ganzzahligen Arrays $a[0], a[1], a[2], \ldots a[m-1]$ der Anzahl positiver ganzer Zahlen k entspricht, für die man das Array in k disjunkte Teilarrays (Sequenzen aufeinanderfolgender Elemente) aufteilen kann, sodass jedes Element genau in einem Teilarray enthalten ist und alle Teilarrays dasselbe minimale nicht enthaltene (minimal excluded) Element haben. Das minimale nicht enthaltene Element eines ganzzahligen Arrays ist die kleinste positive ganze Zahl **(größer als** 0**)**, die nicht im Array vorkommt.

Gegeben ist ein ganzzahliges Array $v[0], v[1], \ldots v[n-1]$ sowie q Abfragen der Form (l_i, r_i) ($0 \leq l_i \leq r_i < n$, $0 \leq i < q$).

Für jede Abfrage soll die Schönheit des Arrays $v[l_i], v[l_i+1], \ldots, v[r_i]$ bestimmt werden.

Implementierungsdetails

Implementiere die folgende Funktion

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

- n: Größe des ganzzahligen Arrays
- v: Array der Länge n, das Ausgangsarray
- q: Anzahl der Abfragen
- $queries$: Array der Länge q, das die Abfragen beschreibt
- Diese Funktion soll einen `vector` von q ganzer Zahlen zurückgeben, der die Antwort für jede Abfrage enthält.
- Diese Funktion wird genau einmal pro Testfall aufgerufen.

Beschränkungen

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

Teilaufgaben

1. (4 Punkte) $1 \leq n \leq 10$, $1 \leq q \leq 100$
2. (6 Punkte) $1 \leq n, q \leq 100$
3. (17 Punkte) $1 \leq n, q \leq 1,000$
4. (10 Punkte) $1 \leq n, q \leq 100,000$ und $1 \leq v[i] \leq 2$ für alle $0 \leq i < n$
5. (30 Punkte) $1 \leq n, q \leq 75,000$
6. (33 Punkte) Keine weiteren Beschränkungen.

Beispiele

Beispiel 1

Betrachte den folgenden Funktionsaufruf

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

In diesem Beispiel ist $n = 10$ und es gibt 2 Abfragen, für die gilt:

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

Für die erste Abfrage kann man das Intervall nur in ein einziges Teilarray aufteilen, nämlich von Position `0` bis Position `5`.

Bei der zweiten Abfrage kann k entweder 1 oder 2 sein. Eine Möglichkeit, in 1 Teilarray aufzuteilen, ist das Teilarray von Position 0 bis Position 8. Eine Möglichkeit, in 2 Teilarrays aufzuteilen, ist von Position 0 bis 5 und von 6 bis 8.

Die Antwort auf die erste Abfrage ist 1, und auf die zweite Abfrage 2. Also soll der Aufruf von `solve` folgendes zurückgeben: `{1, 2}`.

Beispiel-Grader

Der Beispiel-Grader liest die Eingabe im folgenden Format:

- Zeile 1: n q
- Zeile 2: $v[0]$ $v[1]$ $\ldots$ $v[n-1]$
- Zeile $3 + i$: l_i r_i für alle $0 \leq i < q$

und gibt q Zeilen aus, das Ergebnis des Aufrufs der Funktion `solve` mit den entsprechenden Parametern.