Задача D. Массив

Маленькому Пете на День рождения папа подарил массив из n целых чисел, в котором каждое из чисел было в пределах от 1 до m. Петя любит числовые массивы, так что он очень обрадовался и тут же начал задавать папе запросы на этом массиве. Поскольку Петя — активный мальчик, он очень любит разнообразие и краткость. Поэтому при каждом запросе он выбирал l-й элемент массива и спрашивал папу, при каком минимальном значении r среди элементов с l-го по r-й включительно будет не менее чем k различных чисел.

Сначала папа пытался честно отвечать на Петины запросы, но их было слишком много, поэтому он быстро устал и попросил вас помочь ему.

Формат ввода

Первая строка входного файла содержит два целых числа n и m, разделенные пробелами — количество чисел в массиве и максимальное возможное значение числа в массиве ($1 \le n, m \le 100\,000$).
Вторая строка содержит n целых чисел $a_1, a_2, \ldots, a_n$ — элементы массива ($1 \le a_i \le m$).
Третья строка входного файла содержит q — количество запросов, которые необходимо обработать ($1 \le q \le 100\,000$).
В процессе обработки запросов необходимо поддерживать число p, исходно оно равно 0. Каждый запрос задается парой чисел x_i и y_i, Петя генерирует их для получения данных очередного запроса следующим образом: $l_i = ((x_i + p) \bmod n) + 1$, $k_i = ((y_i + p) \bmod m) + 1$ ($1 \le l_i, x_i \le n$, $1 \le k_i, y_i \le m$). Пусть ответ на i-й запрос равен r. После выполнения этого запроса, следует присвоить p значение r.

Формат вывода

На каждый запрос выведите одно число — искомое минимальное r, либо 0, если такого r не существует.

Решение.

1) Предположим, что все запросы даны в offline.
2) На основании исходного массива построим массив b из нулей и единиц, где 1 – означает что, число $a[i]$ не встречалось в массиве ранее.
3) Построим дерево отрезков для суммы на массиве b. С помощью этого дерева мы можем ответить на вопрос задачи, в случае, если в запросе $l=1$. Для этого достаточно пройтись по дереву отрезков вниз. Если сумма в левом сыне меньше k, идём вправо уменьшив k на значение этой суммы, иначе идём влево.
4) Чтобы можно было отвечать на запросы для любых l достаточно просто отсортировать запросы по l и отвечать на них по возрастанию. Изначально массив b заполнен нулями. При переходе на следующие значение l сделаем *update* массива b:

изменив $l+1$ индекс массива b на ноль, а предыдущее вхождение числа $a[l+1]$ (если он было) поменять на единицу (здесь мы по сути для любого суффикса забываем о том, что в массиве a были элементы не входящие в суффикс). Для того чтобы находить предыдущее значение числа в массиве a можно хранить все вхождения для каждого возможного числа а также текущие позиции в этом массиве, уменьшая их каждый раз, когда мы изменяем $b[l+1]$. Теперь для каждой версии мы можем искать ответ на запрос спуском по дереву как в пункте 3.

5) Если сделать дерево отрезков персистентным и создать версии для всех l , можно будет отвечать на запросы онлайн. Для этого нужно делать поиск описанный в пункте 3 для нужной нам версии дерева отрезков (в зависимости от l)

Оценка времени работы.

Построение дерево отрезков $O(n\times\log(n))$ – так как мы делаем *update* дерева отрезков за $O(\log(n))$ для всех l .

Поиск по дереву происходит за $O(\log(n))$ ведь мы всегда идём только в одну сторону, а высота дерева отрезков $\log(n)$. Так как имеем q запросов, то ответы на них будут занимать $O(q\times\log(n))$.

Итоговая асимптотика: $O(n\times\log(n))+O(q\times\log(n))$

Оценка используемой памяти:

Персистентное дерево отрезков занимает $O(n+q\times\log(n))$ памяти.

Для предподсчета предыдущих значений массива a используется $O(q\times\log(n))$ памяти.

Итоговая асимптотика: $O(n+q\times\log(n))+O(m+n)$