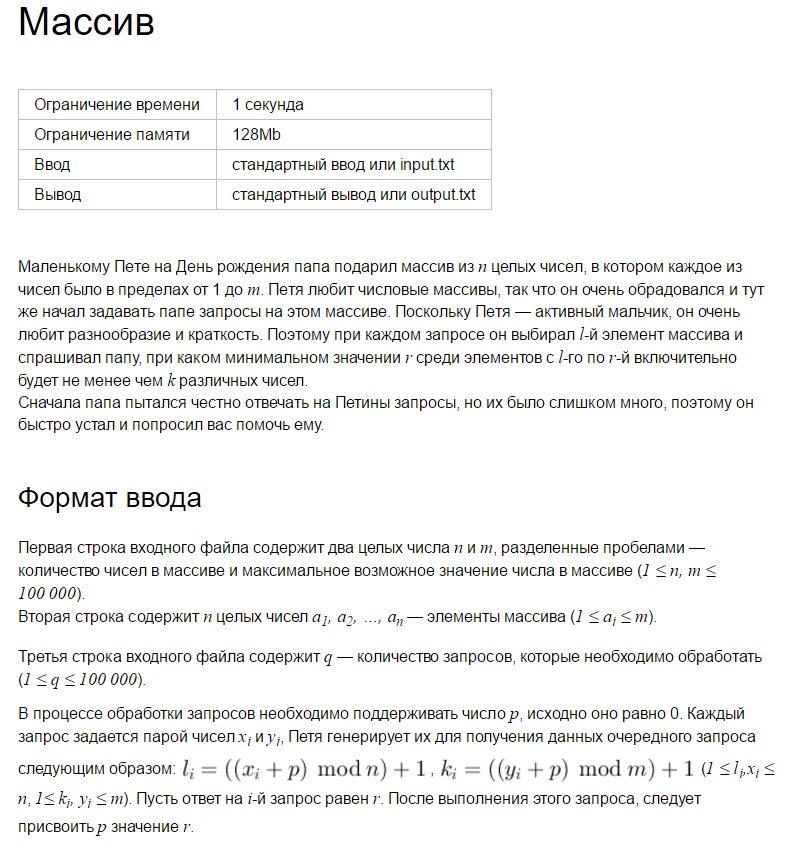
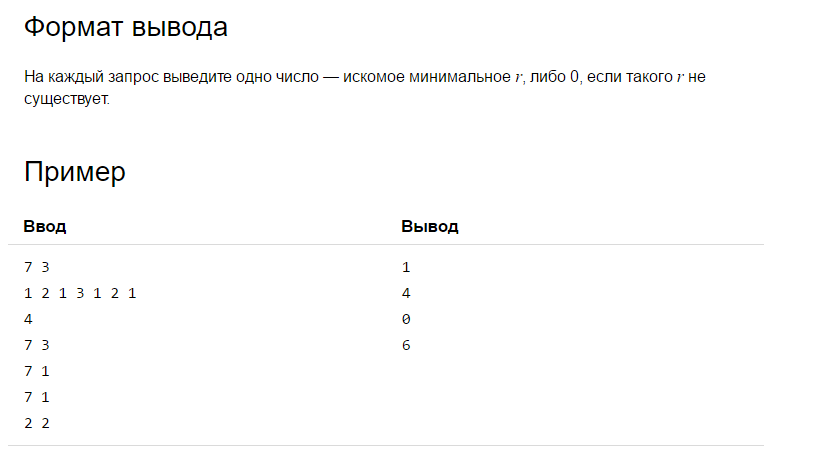
Задача «Массив»

Решение Евченко Богдана ПИ-15-1

https://github.com/olderor/advanced2





Решение.

1. Для решения задачи построим персистентное дерево отрезков. Для каждого левого индекса (то есть ) сделаем свою версию дерева отрезков, в котором будем хранить количество различных элементов на отрезке.
   1. Построим дерево отрезков, заполненное нулями, - это будет начальная версия дерева отрезков, которую будем в последствии изменять. Также, создадим массив размером , заполненный значением . В этом массиве будем хранить последний записанный индекс для значений .
   2. Для каждого нового суффикса будем обновлять предыдущую версию дерева отрезков. Пройдемся по всем элементам массива, начиная с конца. Установим значение под индексом в дереве отрезков, а также обнулим значение в дереве под индексом , то есть индекс последнего записанного элемента с таким же значением, если таковой был ( ) и обновим дерево. Запишем в массив . Таким образом, для каждого префикса будем хранить в листьях: 1 – если это первое вхождение числа в массив, 0 – если оно уже повторялось ранее. Поскольку мы проходим с конца, то для левой границы будет соответствовать полученное дерево (все элементы до нас не интересуют, их значения в дереве 0 соответственно).
2. Следовательно, для каждого узла в дереве суммируя значения его детей мы получим количество различных элементов в массиве на отрезке, а поскольку дерево персистентное, мы сможем хранить это количество и для каждой левой границы.
3. Для каждого поступающего запроса будем опускаться по дереву с нужной нам версией, отнимая количество различных элементов на рассматриваемом отрезке от данного. Так мы легко найдем правый индекс, такой что кол-во элементов на отрезке будет не менее чем . При этом, если после прохода дерева осталось положительным, то это значит, что такого не существует.

Доказательство

В пункте 1.2 каждый раз, заполняя дерево отрезков, мы записываем первое вхождение числа в массив, обнуляя все последующие повторения. Таким образом, для рассматриваемого дерева отрезков мы будем иметь самую оптимальную информацию, то есть на каждом отрезке будет храниться максимальное количество различных чисел в массиве. Это очевидно, поскольку если записать не первое вхождение числа, необходимо будет сдвинуть правый индекс соответственно вправо, что не является оптимальным решением, поскольку в задаче необходимо найти минимальное .

Итоги

Оценка памяти. Для решения задачи потребовалось: массив размером ; массив размером ; персистентное дерево отрезков, размер которого составляет , поскольку использовалась ссылочная реализация, а также память на запись изменений дерева (для каждой версии необходимо было обновлять дерево за – глубина дерева). Следовательно, полная сложность алгоритма по памяти равна .

Оценка времени работы. Для решения задачи потребовалось: считать значения массива за , построить дерево отрезков за , для каждой версии обновить дерево за , что в целом составляет . А также для каждого запроса найти ответ (поиск в дереве) за , что в целом составляет Следовательно, полная сложность алгоритма по времени работы равна.

Итог:

* затраты по памяти.
* затраты по времени.