goodbaidakov

DS-7. АиСД. 2.

16 янв 2022, 18:47:12
старт: 12 дек 2021, 12:14:01
финиш: 1 янв 2022, 14:14:01
длительность: 20д. 2ч.
начало: 12 дек 2021, 12:14:01
конец: 1 янв 2022, 14:14:01

B. Минимум на отрезке

Ограничение времени	10 секунд
Ограничение памяти	128Mb
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Рассмотрим последовательность целых чисел длины n. По ней с шагом 1 двигается «окно» длины k, то есть сначала в «окне» видны первые k чисел, на следующем шаге в «окне» уже будут находиться k чисел, начиная со второго, и так далее до конца последовательности. Требуется для каждого положения «окна» определить минимум в нём.

Формат ввода

В первой строке входных данных содержатся два натуральных числа n и k ($n \le 150000$, $k \le 10000$, $k \le n$) – длины последовательности и «окна», соответственно. На следующей строке находятся n чисел – сама последовательность.

Формат вывода

Выходные данные должны содержать $n - k + 1$ строк – минимумы для каждого положения «окна».

Пример

Ввод	Вывод
7 3 1 3 2 4 5 3 1	1 2 2 3 1

Примечания

Обратите внимание, что решение с непосредственным подсчётом минимума для каждого положения окна не пройдёт по времени. Один из способов решить задачу – использовать контейнер std::multiset, чтобы хранить содержимое окна и быстро получать минимум. Подробнее об std::multiset читайте здесь: http://en.cppreference.com/w/cpp/container/multiset.

Язык Python 3.7.3

Набрать здесь | Отправить файл

```
class Tree:
    def __init__(self, input):
        s = len(input)
        self.base = 1
        while self.base < s:
            self.base = self.base * 2
        self.tree = list()
        for _ in range(0, self.base * 2):
            self.tree.append(0)

        for i in range(0, s):
            self.tree[self.base + i] = input[i]

        for i in range(self.base - 1, 1, -1):
            self.tree[i] = min(self.tree[i * 2], self.tree[i * 2 + 1])

    def rec_minim(self, pointer, search_L, search_R, seg_L, seg_R):

        if search_L == seg_L and search_R == seg_R:
            return self.tree[pointer]

        left_child_seg_R = (seg_R + seg_L) // 2

        if search_R <= left_child_seg_R:
            return self.rec_minim(pointer * 2, search_L, search_R, seg_L, left_child_seg_R)

        if search_L >= left_child_seg_R + 1:
            return self.rec_minim(pointer * 2 + 1, search_L, search_R, left_child_seg_R + 1, seg_R)

        return min(self.rec_minim(pointer * 2, search_L, left_child_seg_R, seg_L, left_child_seg_R),
                   self.rec_minim(pointer * 2 + 1, left_child_seg_R + 1, search_R, left_child_seg_R + 1, seg_R))

    def minim(self, l, r):
        return self.rec_minim(1, l, r, 0, self.base - 1)

line = open('input.txt')
n, k = map(int, next(line).split())
```

Отправить

Предыдущая | Следующая