АНАЛИЗ НА ЗАДАЧА
red

Първа + втора подзадача

Наивното решение работи за тези подзадачи. Тук използваме булев масив, който съхранява срещаните елементи за всеки от подредиците. Така знаем кои числа можем да съберем с `long long sum` (различните), тръгвайки от дадена позиция в редицата.

Постигната сложност: $O(N^2)$
Решение: `red_40p.cpp`

Пълно решение

Вместо да разглеждаме всяка подредица поотделно, намираме броя на подредиците, в които a_i участва поне веднъж, и го умножаваме по самото a_i, събирайки после произведението с отговора. Още отсега е нужно да отбележим, че:

Ако a_L, a_{L+1}, a_{L+2}, …, a_{R-1}, a_R не съдържа числото x, то и подредиците му с начала и крайчета съответно l и r, където $L \leq l \leq r \leq R$ не съдържат числото x и те общо са $\frac{(R-L+1) \times (R-L+2)}{2}$ на брой.

Разделяме редицата на подредици, част от които не съдържат нито веднъж числото a_i и имат възможно най-голяма дължина. Изобразен е пример, където $j < i$ и $a_i = a_j$.

$$a_j,\ a_{j+1},\ a_{j+2},\ \ldots,\ a_{i-1},\ a_i,\ \ldots$$

Тук a_{j+1}, a_{j+2}, …, a_{i-1} не съдържа a_i. $\Rightarrow$ Има $\frac{(i-1-(j+1)+1) \times (i-1-(j+1)+2)}{2}$ на брой по-малки подредици измежду a_{j+1}, a_{j+2}, …, a_{i-1}. Ще използваме дадена редица като пример. Нека редицата ни е $\{1, 2, 3, 1, 2, 1\}$. Тук можем да я разделим на части $(2, 3)$, $(5, 5)$, които не съдържат $a_1 = 1$, общо $\frac{2 \times 3}{2} + \frac{1 \times 2}{2} = 4$. Редицата има общо $\frac{6 \times 7}{2} = 21$ подредици, затова $(21 - 4) \times 1 = 17$ сума от $a_1 = 1$. Прибавяме я към финалния ни отговор.

Обобщение:

За най-подходящ подход се приема итерирането през всеки елемент от дадената редица, изчислявайки сумата, създадена спрямо a_i, а после преминаване през стойностите на елементите (от 1 до 10^5) и извършвайки последна операция, описана по-горе. При решението ми за 100 точки използвах точно описаната операция, но и друго решение се счита и алгоритъма, описан в първия абзац на това описание на пълното решение.

Постигната сложност: $O(N + MAXX)$, като $MAXX = max(a_1, a_2, \ldots, a_N)$

Решение: `red_100p.cpp`

***Автор**: Радослав Радославов*