Increasing

	На пълното решение	На подзадачите
Тагове	Динамично оптимиране Оптимизация на вътрешен цикъл Сегментно дърво Дърво на Фенуик	Пълно изчерпване

Анализ

Преди да започна ще спомена, че долуописаните сметки се смятат под модула от условието. Няма да го казвам на всяка една от тях, защото големината на анализа ще стане поне двойна ☺.

Подзадача №1

В тази подзадача са тестовите примери. Тя е за обратна връзка от системата.

Подзадача №2

За всяка възможна подредица се проверява дали е нарастваща.

Постигната сложност: $O(2^N)$ или $O(2^N \times N)$.

Имплементация: `First_10p.cpp`

Подзадача №3

Един възможен подход за подзадачата е да се разбият всички търсени подредици на две части, съответно лява част, състояща се само от единици, и дясна част, състояща се само от двойки. Среден елемент в подредицата ще наричаме най-дясната единица в такава подредица. Нека $ones_i$ е броят елементи, с индекс $\leq i$, които са единици, а $twos_i$ е броят елементи, с индекс $\geq i$, които са двойки. За всяка възможна единица се проверя колко подредици я включват като среден елемент. За единицата на i-та позиция, отговорът би бил $2^{ones_i - 1} \times 2^{twos_{i+1}}$. Забележете, в подредиците, образувани само от двойки, няма среден елемент. За тях може да се използва фиктивен такъв на позиция 0.

Постигната сложност: $O(N)$.

Имплементация: `Second_12p.cpp`

Подзадача №4

Ако погледнем задачата острани, в нея се търси брой на нещо си под някакъв модул. Това означава две неща – или е комбинаторика, или е динамично. Състезателите на контролното състезание бяха късметлии, защото в тази задача нямаше и грам комбинаторика ☺ (е, освен трета подзадача).

Всъщност, какъв би бил стейта на най-лесното динамично? Най-наивният такъв би бил `f(ix, value)`, като `f(ix, value)` е равно на броя ненамаляващи подредици, с последен елемент в тях `value`. За изчисляване на стойността се разглеждат два варианта, съответно дали в редицата участва елемента на позиция `ix`, или не.

Постигната сложност: $O(N \times a_i)$.

Имплементация: `Third_18p.cpp`

Подзадача №5

За тази подзадача има два варианта за решение. Първият е да се компресират числата в редицата в решението от горната подзадача. Вторият вариант е да се лишим от втория параметър на динамичното по следния начин:

Нека dp_i е броят на нарастващи подредици, които завършват на i-тия елемент в редицата. За изчисление на динамичното се фиксира предпоследен елемент в нарастващата подредица, завършваща на i-тия елемент, с вложен цикъл. По-формално казано $dp_i = 1 + \sum_{j=1, a_j \leq a_i}^{i-1} dp_j$. Единицата в сметката е за подредицата, която включва единствено i-тия елемент от редицата.

Постигната сложност: $O(N^2)$.

Имплементация: `Forth_39p.cpp`

Решение, което изкарва точките за описаните подзадачи до момента

До тук добре. Може би се питате „Какво може още да се направи?“. Всъщност, нека позавъртим малко нещата.

Нека сме компресирали числата и максималното от тях е равно на max. Последователно обхождаме числата в редицата отляво-надясно. Нека сме стигнали до елемент на позиция i и поддържаме cnt_x, равно на броя на нарастващите подредици, които завършват на елемент със стойност x. Разбира се, не е задължително x да е на i-та позиция. Ако сме успели да изчислим тези стойности за $i = N$, тогава отговорът би бил $cnt_1 + cnt_2 + cnt_3 + \cdots + cnt_{max}$. Сега въпросът е как се променя cnt, като добавим $i + 1$-вия елемент. Очевидно, само бройката на подредиците, които завършват на a_{i+1} ще се промени. По-точно ако използваме означенията от анализа на пета подзадача, $cnt_{a_{i+1}} \coloneqq cnt_{a_{i+1}} + dp_{i+1}$, където с $\coloneqq$ е означен знак за присвояване. Следващия въпрос е как може да се намери стойността на dp_{i+1} по-елегантно. Тя всъщност би била равна на $1 + cnt_1 + cnt_2 + cnt_3 + \cdots + cnt_{a_{i+1}}$. Така с вложен цикъл намираме колко трябва да се добави в $cnt_{a_{i+1}}$.

Постигната сложност: $O(N \times max)$.

Имплементация: `Slow_51p.cpp`

Подзадача №6 и Подзадача №7

В решението за 100 точки е основополагаща гореописаната идея, само трябва да се забърза малко. Може да се каже, че задачата се свежда до следните две заявки върху редицата cnt:

1) Въпрос за стойността на сбора на елементите с номера от 1 до дадено число x.
2) Заявка за добавяне на дадена стойност $value$ на елемент на дадена позиция x.

Естествено, вместо с вложени цикли, тези заявки може да се поддържат с дърво на Фенуик или със сегментно дърво. Разбира се, за решението на седма подзадача е нужна компресия на числата в редицата.

Постигната сложност: $O(N\ log_2\ N)$

Имплементации:

- Без компресия с дърво на Фенуик: `WithoutCompression_71p.cpp`
- С компресия с дърво на Фенуик: `author_100p.cpp`
- С компресия със сегментно дърво: `SegTree_100p.cpp`

Автор: Борис Михов