

Анализ – sandwiching

Първа подгрупа

Генерираме всяка пермутация и проверяваме колко е броят на елементите, за които $p_i=i$.

Сложност $O(N!*N)$

Втора подгрупа

С битови маски фиксираме i -тата, за които $p_i=i$. След това, ако в една маска броят на сетнатите битове (нека го означим с $cntr$) е $\geq L$ и $\leq R$, можем да видим броя пермутации за елементи $N-cntr$, които нямат елементи от типа $p_i=i$, и да ги прибавим към отговора. За по-бързо изпълнение, ще изчислим предварително за всеки брой елементи от 0 до $N-L$ колко са пермутациите, в които няма i , за което $p_i=i$, и след това в отделен цикъл ще въртим битовите маски.

Сложност $O((N-L)!*(N-L)+2^N*N)$

Трета подгрупа

Тук, както си личи от подусловието, ни е нужно да намерим бърз начин за намиране на броя пермутации на N елемента, в които не съществува i , такова че $i=p_i$. Такъв тип пермутации се наричат *derangements* и съществуват много алгоритми за намиране на бройката им. Сега ще обсъдим най-лесният за имплементиране, който се основава на принципа за включване-изключване. Нека E_q е събитието, при което $q=p_q$. От тук може да се види, че броят *derangements* за N елемента е $D_N=N! - |E_1 \cup E_2 \cup E_3 \cup \dots \cup E_n|$. Но ако направим $N! - |E_1| - |E_2| - |E_3| - \dots - |E_n|$, ще повторим голяма част от събитията, понеже е възможно в една пермутация повече от един елемент да си е на мястото. Затова ще трябва да добавим броя пермутации, в които два елемента са си на мястото. Но следвайки нашата логика, по този начин ще прибавим броя на пермутациите, в които три елемента са си на мястото. От тук се получава шаблон, при който броят пермутации с четен брой фиксирани елементи ще трябва да се прибавят, а тези с нечетен да се извадят. Бързо можем да намерим броя пермутации, в които поне K елемента са фиксирани и това става по следния начин: $C_N^K * (N-K)!$. От тук можем да съкратим $(N-K)!$ и да получим, че броят е всъщност $N!/K!$. Тоест ние трябва да намерим $N! - N! \sum (-1)^K / K!$. Можем да забележим, че за преход от D_i към D_{i+1} трябва да умножим D_i по $(i+1)$ и да прибавим $(-1)^{i+1}$. Тоест $D_i = D_{i-1} * i + (-1)^i$.

Сложност $O(N)$

Четвърта подгрупа

В тази подгрупа е необходимо да разширим идеята от миналата подзадача до произволно K . Това може да постигнем лесно като използваме триъгълникът на Паскал за пресмятане на комбинациите. Така за всяко число i между L и R ще трябва да намерим $C_N^i * D_{N-i}$, като намирането на комбинациите става с двумерно ДП.

Сложност $O(N^2)$

Шеста подгрупа

За да решим задачата, трябва да оптимизираме намирането на комбинациите. Можем да забележим, че ако използваме стандартната формула за комбинации ($C_N^K = \frac{N!}{(N-K)! \cdot K!}$), преходът от i към $i+1$ не променя особено много в числителя и знаменателя. Това ни позволява с минимални усилия да изчислим C_N^i за всяко $i \leq N$ линейно. В тази идея има един проблем – имаме числа в знаменател. Тъй като отговорът се смята по модул, делението не е възможна операция (в модулната аритметика няма деление). За да заобиколим този проблем, ще трябва да намерим *modular multiplicative inverse* на всяко число в знаменател. Накратко казано, този *inverse* ще ни позволи да разделяме числа чрез умножение (важно е да се отбележи, че *inverse* съществува само ако делителят и модулът са взаимно прости - затова в задачата се използва модул 10^9+7 , което е просто число). Нека разгледаме един пример: ако искаме да разделим на 2 по модул 10^9+7 , можем вместо деление на 2 да извършим произведение по $5 \cdot 10^8+4$. Така ако искаме например $12/2$, ще направим $(12 \cdot 500000004) \% 1000000007$, което ще даде резултат 6. Добре, но как да намерим този *inverse*? Важното свойство, което той притежава, е че $a * x \equiv 1 \pmod{p}$. Но според малката теорема на Ферма, $a^{p-1} \equiv 1 \pmod{p}$. От тук можем да намерим *inverse*-а директно: $x = a^{p-2}$. Следователно, ще използваме бързо степенуване и за всяко число i от 2 до N ще намерим неговият *inverse* чрез формулата $x_i = i^{10^9+5}$. Така извършваме всички изчисления в рамките на един единствен цикъл.
Сложност $O(N)$

Пета подгрупа

Също като в пета подгрупа, но не е необходимо линейното пресмятане на обратните елементи на факториелите по модул.
Сложност $O(N \cdot \log(MOD))$

Автори: Преслав Тошев и Калоян Върбанов