**Анализ – sandwiching**

**Първа подгрупа**

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

*Сложност О(N!\*N)*

**Втора подгрупа**

С битови маски фиксираме i-тата, за които =i. След това, ако в една маска броят на сетнатите битове (нека го означим с cntr) е >=L и <=R, можем да видим броя пермутации за елементи N-cntr, които нямат елементи от типа =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 =i, и след това в отделен цикъл ще въртим битовите маски.

*Сложност O((N-L)!\*(N-L)+2N\*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=. 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 е събитието, при което q=. От тук може да се види, че броят *derangements* за N елемента е =N!-|U U U … U |. Но ако направим 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 К елемента са фиксирани и това става по следния начин: \*(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!. Можем да забележим, че за преход от към трябва да умножим по (i+1) и да прибавим . Тоест =\*i+.

*Сложност O(N)*

**Четвърта подгрупа**

В тази подгрупа е необходимо да разширим идеята от миналата подзадача до произволно К. 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 \*, като намирането на комбинациите става с двумерно ДП.

*Сложност О(N2)*

**Шеста подгрупа**

За да решим задачата, трябва да оптимизираме намирането на комбинациите. Можем да забележим, че ако използваме стандартната формула за комбинации (), преходът от i към i+1 не променя особено много в числителя и знаменателя. Това ни позволява с минимални усилия да изчислим за всяко i<=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 109+7, което е просто число). Нека разгледаме един пример: ако искаме да разделим на 2 по модул 109+7, можем вместо деление на 2 да извършим произведение по 5\*108+4. Така ако искаме например 12/2, ще направим (12\*500000004)%1000000007, което ще даде резултат 6. Добре, но как да намерим този *inverse*? Важното свойство, което той притежава, е че . Но според малката теорема на Ферма, . От тук можем да намерим *inverse*-а директно: . Следователно, ще използваме бързо степенуване и за всяко число i от 2 до N ще намерим неговият *inverse* чрез формулата . Така извършваме всички изчисления в рамките на един единствен цикъл.

*Сложност О(N)*

**Пета подгрупа**

Също като в пета подгрупа, но не е необходимо линейното пресмятане на обратните елементи на факториелите по модул.

*Сложност О(N\*log(MOD))*

*Автори: Преслав Тошев и Калоян Върбанов*