**АНАЛИЗ**

*Тагове: пермутации, динамично програмиране, бързо смятане, цикличност по модул*

Задачата има доста просто условие – намерете броя на пермутациите на ***N*** числа, които са *красиви*, т.е. имат поне една двойка съседни числа във вида .

Първата подзадача е за 9 точки. Тя е за наивната идея – генерираме всички пермутации на ***N*** числа и проверяваме всяка от тях дали е *красива*. Единственият триков момент е, че модулът може да бъде много малък, затова все пак трябва да сметнем отговора по модул ***M***. Сложността е .

Втората подзадача е за 14 точки. Тук ще оптимизираме решението с пълно изчерпване. Нека да мислим, че генерираме пермутации, използвайки стандартната рекурсия. Стандартен подход за оптимизация, когато имаме такава ситуация, е да видим, че, когато сме на конкретно число, не ни интересуват какъв е бил точният ред на числата преди това, единствено какви са били те. Така че се интересуваме единствено от множеството числа, които сме използвали и текущото число. Можем да използваме това, за да направим мемоизация на рекурсията: , като *mask* показва какво множество числа вече сме използвали и, че *last* е последното използвано число. Това *dp* ще ни казва броя на *красивите* пермутации, които имат определено множество от числа на първите *k* позиции, на *k*-та позиция числото е *last* и имат свойството, че не е имало позиция от 1 до (*k*-1) във вида . За да пресметнем определено *dp*, имаме два случая. Първият е да използваме числото *(last+1)* (ако можем) на следващата позиция. Това ще направи пермутацията *красива*, независимо от това какви са числата след това. Затова просто трябва да ги сложим в определен ред. Това означава, че в този случай трябва просто да сметнем някакъв факториел, вземащ предвид останалите числа. Другият случай е по-лесен. Ако използваме някое друго число, то свойството, че няма позиция във вида се запазва, така че просто ни трябва стойността на някое друго състояние на *dp*-то. В това решение, всички сметки трябва да бъдат направени по модул ***M***. Сложността тук е малко по-добра – .

Третата подзадача е за 11 точки. Използваме друг стандартен подход. Когато имаме само едно число на вход и има подзадачи с малки ограничения, можем да преизчислим някои от отговорите. Но тук отговорът трябва да се изчисли по някакъв модул, научаван по време на входа. Това не е проблем, понеже ограниченията за тази подзадача са сравнително малки. Една проста сметка може да ни покаже, че 20! е достатъчно малко, че да се побере в *long long*. Така че, можем да използваме предишното оптимизирано пълно изчерпване, за да сметнем отговорите до ***N***=20 без да използваме модул. Авторът допълнително е оптимизирал преизчисляването. Можем да забележим, че когато смятаме отговорът за определено ***N***, отговорите за всички по-малки ***N*** са някои от стейтовете на *dp*-то. Така че можем да пуснем предишното решение само веднъж, за да научим всички отговори, които ни трябват. По този начин, преизчисляването би трябвало да работи по-малко от минута на модерен компютър. Сложността на преизчисляването е и решението е просто да отпечатаме вече изчислен отговор по въведения модул.

Четвъртата подзадача е за 43 точки. Това е най-важната подзадача, в която ще опишем линейно решение за тази задача. Ще мислим за подобна задача – ще преброим броят на пермутациите, които не са *красиви*. Нека да ги наричаме *грозни* за простота. Очевидно, ако знаем броят на *грозните* пермутации на ***N*** числа, то *красивите* пермутации могат да бъдат сметнати, като извадим от ***N***!. Нека означим с – броят на *грозните* пермутации на ***N*** числа. Има два случая как можем да направим *грозни* пермутации на ***N*** числа, използвайки пермутации на (***N***-1) числа. Нека да си мислим за произволна пермутация *p* на (***N***-1) числа. Ако тази пермутация е *грозна*, то трябва да поставим числото ***N*** по такъв начин, че пермутацията да остане *грозна*. Има ***N*** възможни позиции по принцип, но една от тях е невъзможно в нашия случай – позицията след числото (***N***-1). Така че *грозните* пермутации на ***N*** числа, които могат да бъдат направени по този начин са . Сега ще разгледаме по-трудния случай – нека *p* е *красива* пермутация. Очевидно, трябва да сложим ***N*** на такова място, че да разкъсаме двойките във вида . Но ако има две или повече такива двойки, е невъзможно да ги разкъсаме с поставянето на едно число. Така че трябва *p* да е *красива* със само една двойка във вида . Нека да наречем такива пермутации *супер красиви*. Ако имаме *супер красива* пермутация, то имаме само едно място, където трябва да поставим ***N*** – между *x* и *(x+1)*. Можем да забележим, че *x* може да е всяко от числата от 1 до (***N***-2) (не може да бъде (***N***-1)). Затова, слагайки ***N*** и постигайки , ние няма да имаме двойка във вида и това означава, че *супер красиви* пермутации *p* могат да станат *грозни* по точно един начин. Последно, трябва да намерим какъв е броят на *супер красивите* пермутации на (***N***-1) числа. Един подход, който няма да обсъждаме, е да направим друго *dp* за тази бройка, което ще бъде взаимно рекурентно с първото *dp*. Нека да пресметнем броят на *супер красивите* пермутации на (***N***-1) числа, които имат двойката (1, 2). Това може да бъде постигнато по следния начин. Можем да поставим 2 на позицията точно след 1 в някоя *грозна* пермутация на числата в (*S* има (***N***-2) числа). Освен това, ако имаме пермутация *q* на числата в *S*, за да я направим *супер красива* с двойката (1, 2), единственият начин е да поставим 2 след 1 в *q*. След това, трябва да осигурим, че няма други двойки във вида . Когато поставим 2 след 1, ще имаме (може да нямаме трето число *y*, ако 1 е било накрая). Ако *y* е 3, то новата пермутация няма да бъде *супер красиви*. Така че искаме *q* да няма никоя двойка от множеството . Това е еквивалентно на броят на *грозните* пермутации на (***N***-2) числа. Нека да обобщим. Броят на *супер красивите* пермутации на (***N***-1) числа, които имат двойката за фиксирано *x* е за всички *x* в . Това означава, че броят на *супер красивите* пермутации на (***N***-1) числа е . В крайна сметка, броят на *грозните* пермутации на ***N*** числа е: , проста рекурентна зависимост. Лесно е да се види, че . Това означава, че можем да сметнем по модул ***M*** без проблем едновременно с ***N***!. Крайният отговор ще е по модул ***M***. Сложността е линейна - .

Последната подзадача е за 23 точки. Понеже коефициентите на рекурентната зависимост зависят от ***N***, повечето от стандартните подходи за бързо смятане не работят – намиране на формула, умножение на матрици и т.н. Тук е моментът да използваме това, че само смятаме отговорът по модул ***M***. Можем да направим следното наблюдение – понеже коефициентите зависят от ***N***, то те се повтарят периодично с период ***M***. Нека да погледнем момента, когато почват да се повтарят. Имаме:

…

Подобно, ще имаме:

Сега ясно можем да видим следната формула в общия случай:

Това е причината, модулът да не е много голям – максимално до 107. Можем да използваме предишното решение за изчисляване на и да използваме горната формула за смятане на *dp*-то за големи ***N***. Трябва да отбележим, че за смятането на остатъка на степента, най-лесният начин е да използваме бързо степенуване. Също така, факториелът за ***N*** ≥ ***M*** по модул ***M*** е нула. Крайната сложност на решението е .

Интересен факт е, че по модул ***M*** за всички ***M***, но това е трудно да се докаже и разбира се не се очаква състезателите да го забележат.

*Автор: Илиян Йорданов*