АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА ПОКРИВАНЕ

С цел избягване на дългите описания в изложението по-нататък, нека въведем следното определение:

Нека $p=(p_1, p_2,\ldots., p_N)$ е пермутация на числата *1, 2, 3, ..., N*. Масивът $L=(L_1, L_{2,\ldots..,}L_N)$, където L_i е броя на покриваните от p_i елементи, ще наричаме *L-масив* на пермутацията *p* и ще означаваме с *L(p).*

Формално можем да дефинираме задачата така:

Да се генерира пермутация *p* на числата от 1 до *N*, така че *L(p)* да съвпада с въведения масив.

Решение с пълно изчерпване

Това е най-простото решение, което може да ни хрумне. Генерираме последователно всички пермутации на числата от 1 до *N*, за всяка пресмятаме нейния *L* масив и го сравняваме с въведения. Ако двата масива съвпаднат, то извеждаме съответната пермутация.

Това е решение от порядъка на *O(N!*f(N))*, където *f(N)* е сложността на пресмятането на *L-масива* на поредната пермутация. Това пресмятане можем да извършим по два начина:

- Обхождаме пермутацията отляво надясно и за всяко число търсим първото, надясно от него, което е по-голямо (то го покрива). В един масив натрупваме за всяко число колко елемента покрива и, когато обходим целия масив, ще сме получили *L-масива* на пермутацията. В този случай $f(N)=O(N^2)$,а цялото решение е със сложност $O(N!*N^2)$.
- Вторият начин е значително по-хитър – изчисляването на *f(N)* се извършва линейно по *N*. Това става като се използва стек, в който стоят елементите от масива, които чакат да бъдат „покрити”. Движим се отляво надясно по масива, като всеки елемент попада в този стек и след като бъде покрит се изважда от него, а към бройката покривани елементи на този, който го покрива, се добавя 1. Цялото решение е със сложност *O(N!*N).*

За съжаление *N!* расте толкова бързо, че чрез системата е невъзможно да се определи кой от двата начина е използван. Въпреки това си заслужава да запомните втория начин – той е класика за използването на стек в случаи, когато елементи на списък трябва да изчакат да се появи следващ ги елемент, който се намира в някакво стноше с тях.

Такива решения са реализирани във файлове **covern!n2.cpp** и **covern!n.cpp** от Руско Шиков**,** решават тестовете, в които $N\leq 10$ и ще получат 10 точки.

Решение с изчерпване с връщане назад (backtracking)

Друго изчерпващо решение може да бъде направено, използвайки техниката backtracking (изчерпване с връщане назад), чиято същност е рекурсивното построяване на търсената пермутация с подходящо „рязане“ на частичните решения, за които става ясно, че няма да доведат до правилно пълно решение. Задачата позволява доста добро „рязане“ и, въпреки, че сложността теоретично е *O(N!)*, това решение ще мине за тестовете, в които $N\leq 20$ и ще получи 50 точки. То е реализирано от Антон Шиков във файл **coverbtrck.cpp**.

Решение със сложност $O(N^2)$

Идеята на това решение е да добавяме елементите на пермутацията един по един отляво надясно, като в един масив поддържаме пермутация на числата от 1 до m, удовлетворяваща условието за покриване, където m е броят на числата в масива към текущия момент.

Когато добавяме нов елемент, знаем колко числа трябва да покрие той (тази бройка може да бъде и 0). Стойността на числото, което добавяме, смятаме линейно като тръгнем от края към началото на текущия масив. Лесно се вижда дали поредното число, което разглеждаме е покрито или не (трябва да няма нищо по-голямо от него до края масива, за да не е покрито). Освен това непокритите числа вървят в нарастващ ред при движението от края към началото. Като отброим толкова непокрити числа, колкото показва съответната бройка в *L-масива*, изчисляваме стойността на новия елемент, който добавяме, по формулата 1 + последното число, което ще покрие този елемент. Ако няма да покрие нито едно число (т.е. съответната стойност в *L-масива* е 0), то новият елемент получава стойност 1. Това може да доведе до повторения в нашия масив. Оправяме повторенията като увеличим с 1 всички числа по-големи или равни на новодобавената стойност. Както лесно може да се съобрази, това не променя отношението на покриване между елементите на масива.

Сложността на този алгоритъм е $O(N^2)$, той е реализиран във файл **cover.cpp** от Йордан Чапъров, ще мине на тестовете, в които $N \leq 10\,000$ и ще получи 100 точки.

Същата задача, под име towers, беше дадена и за групи А и В, но с ограничение $N \leq 1000\,000$. Който се интересува, може да опита да я реши с това ограничение или да разгледа анализа на задача towers от темата за групи А и В.

Автори: *Руско Шиков*
Александър Георгиев
Йордан Чапъров
Антон Шиков