АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА КОНТРОЛНИ

На пръв поглед задачата много наподобява типа задачи, които се решават с техниката „динамично програмиране". В този случай обаче ограниченията на входните данни не позволява директно прилагане на този метод, с изключение на тези тестове, в които всички числа са по-малки от 1000.

Ограничението за **N** е сравнително малко (едва 36) и навежда на мисълта за използването на някакъв вид пълно изчерпване. Наивно реализиране на тази идея не би донесла много точки, защото броят на всички възможни комбинации е $\mathbf{2^N}$ и не би влязла във времевото ограничение. Въпреки това с малко изборетателност можем да вкараме тази идея в действие. Без притеснение можем да разделим входните данни на две приблизително равни части (ако **N** е нечетно, първата част да съдържа един елемент по-малко от втората). Лесно може да се генерират всички възможни суми от елементи, намиращи се в първата част, и да се запомнят в масив. След това се генерират всички възможни суми от елементи, намиращи се във втората част. За всяка от тях трябва да намерим колко е броят на сумите, образувани от еленетите на първата част, такива, че сборът на двете да е по-голям или равен на **T**. Това може да се пресметне с изпозването на метода „двоично търсене". За тази цел първо трябва сумите, получени от елементите от първата част, да се сортират в нарастващ ред. Така за всяка сума от втората част ще тъсим в сортираната последователност индекса на най-малкото число, такова че сборът му с текущата сума да е по-голям или равен на **T**. Всички числа в наредената последователност с индекс по-голям от намерения също ще са подходящи кандидати за сбор с текущата сума от втората последователност.

Разделяйки началната информация на две независими множества увеличава значително ефективността. Броят на всички възможни суми в първата и втората част е $\mathbf{2^{N/2}}$, което за **N = 36** е достатъчно за да влезе в ограниченията по памет и време. Сложността на соритането при използване на вградената функция е $\mathbf{O(2^{N/2}*log(2^{N/2}))}$, а за всяка сума от втората част на входа трябва да направим приблизително $\mathbf{O(log(2^{N/2}))}$ итерации, за да намерим броя на сумите, образувани от еленетите на първата част, такива че сборът на двете да е по-голям или равен на **T**. Следователно общата сложност на решението е $\mathbf{O(2^{N/2}*log(2^{N/2})) = O(2^{N/2}*(N/2))}$.

В примерната реализация за генерирането на всички възможни суми са използвани битови маски. Всяка битова маска показва кои елементи трябва да вземем при формирането на текущата сума – ако **i**-я бит е 0, не взимаме **i**-я елемент, а ако е 1 – го взимаме като събираемо в текущата сума. Лесно се доказва, че всъщност битовите маски са двоичните представяния на числата в инетервала $\mathbf{[0, 2^N)}$, където е **N** е броят на елементите, които разглеждаме като потенциални събираеми. За бърза проверка дали **i**-я бит е **0** или **1**, както и за бързо пресмятане на $\mathbf{2^N}$, се използват побитови операции.

Автор: Николай Хубанов