**Задача B2. Билетчета – Анализ**

Основната идея на решението е да използваме динамично програмиране. Решението само с динамично програмиране обаче може да реши задачата само до N=10000 за под 1 сек. За това добавяме още една оптимизация за да получим решение до N=99999. Първо ще опишем базовото решение с динамично програмиране, след което ще опишем и оптимизацията.

**1. Динамично програмиране.**

Основния проблем на задачата е *„може ли да построим израз равен на K от числото X“.* Трябва да решим тази задача за всички числа от 1 до N. За да решим задачата с динамично програмиране обаче, ще решим малко по-общия проблем: *„кои са всички възможни стойности на изрази, които може да построим от числото X“ за X=1..N.*

С новото условие забелязваме следната рекурентна зависимост – всеки израз който може да се построим от X, се състои от операция, и два под-израза построени от **по-малки** части на числото Х. (разбира се тук има няколко изключения – унарен минус, самото число X и част от X, която започва с водеща нула – но те се добавят лесно). Например за числото 2342 имаме следния възможен израз - 23+(4\*2). Израза 23+(4\*2) е построен от операцията +, и изразите 23 и 4\*2. А 23 и 4\*2 са измежду възможните изрази които могат да бъдат построени от по-малките части на 2342 – 23 и 42. Рекурентната зависимост става видима и ако гледаме на изразите като двоични дървета:

**+**

**23**

**\***

**4**

**2**

Тази зависимост дава идеята за следния рекурсивен алгоритъм за намиране на всички възможни стойности на изрази получени от числото X (за улеснение приемаме, че може да почваме с водеща нула):

vuzmozni(X)

Нека x1x2..xn е десетичния запис на X

result := {X} // Множество с единствен елемент X

for i := 1 ... n

a := x1x2..xi

b := xi+1xi+2..xn

for l in vuzmozni(a)

for r in vuzmozni(b)

result =

if (r != 0 && l % r == 0)

result =

for v in result

result =

В горе-описания алгоритъм за всяко възможно разбиване на числото X=x1x2..xn на 2 половини - a := x1x2..xi и b := xi+1xi+2..xn, за всички възможни стойности l и r, които могат да се построят от двете половини – а и b респективно, и за всяка валиднa операция ◦ между l и r, добавяме l◦r към възможните изрази за Х. Също за всяка намерена възможна стойност v за Х трябва да добавим и -v за X, тъй като винаги може да сложим унарен минус отпред на всеки израз.

За да превърнем рекурсивното решение в динамично, е достатъчно да превърнем vuzmozni в вектор от множества, които се изчислява от 1 до N:

Нека vuzmozni е вектор от множества с размер N

for X := 1 ... N

Нека x1x2..xn е десетичния запис на X

vuzmozni[X] := {X} // Множество с единствен елемент X

for i := 1 ... n

a := x1x2..xi

b := xi+1xi+2..xn

for l in vuzmozni[a]

for r in vuzmozni[b]

vuzmozni[X]=

if (r != 0 && l % r == 0)

vuzmozni[X] =

for v in vuzmozni[X]

vuzmozni[X] =

Така описаното решение работи за под 1 сек до около N=10000.

Накрая за да изчислим отговора на оригиналната задача, е достатчно да преброим в колко от множествата vuzmozni[i] за i=1...N присъства К.

**1. Oптимизация до N=99999**

За да решим задачата до N=99999, е достатчно да забележим, че няма нужда да изчисляваме **цялото** множество от всички възможни стойности за числата от 10001 до 99999. За тези числа може директно да проверим дали К е възможна стойност. Това е възможно тъй като всяко от тези числа е 5-цифрено, и както и да го разбием на 2 половини, те винаги ще са най-много 4-цифрени. Тоест и двете половини ще бъдат по-малки от 10000, и за двете половини може да използваме резултата от динамичното програмиране изчислен до 10000.

Тогава ако имаме вектора vuzmozni изчислен от динамичното по-горе до 10000, и дадено 5-цифрено число Х и дадено К, може да проверим дали К може да се получи от Х ето така:

Нека vuzmozni е вектора изчилсен от динамичното до 10000

Нека x1x2..xn е десетичния запис на X

result = false;

for i := 1 ... n

a := x1x2..xi

b := xi+1xi+2..xn

for l in vuzmozni[a]

if

result = true;

break;

if

result = true;

break;

if l != 0 && K % l == 0 &&

result = true;

break;

if l != 0 && K != 0 &&

result = true;

break;

Накрая даваме псевдокода на цялото решение:

// Вход – N и К

otgovor := 0

Нека vuzmozni е вектор от множества с размер N

for X := 1 ... min(N, 10000)

Нека x1x2..xn е десетичния запис на X

vuzmozni[X] := {X} // Множество с единствен елемент X

for i := 1 ... n

a := x1x2..xi

b := xi+1xi+2..xn

for l in vuzmozni[a]

for r in vuzmozni[b]

vuzmozni[X]=

if (r != 0 && l % r == 0)

vuzmozni[X] =

for v in vuzmozni[X]

vuzmozni[X] =

if

otgovor ++

for X := 10001 ... N

Нека x1x2..xn е десетичния запис на X

for i := 1 ... n

a := x1x2..xi

b := xi+1xi+2..xn

vuzmozno = false

for l in vuzmozni[a]

if

vuzmozno = true;

break;

if

vuzmozno = true;

break;

if l != 0 && K % l == 0 &&

vuzmozno = true;

break;

if l != 0 && K != 0 &&

vuzmozno = true;

break;

if vuzmozno

otgovor ++

break

print(otgovor)

Като забележка ще добавим, че не е нужно стейтът да е стринг или да кодираме цифрите, заради водещите нули. Следният пример илюстрира защо това все пак е нужно. Ако имаме числото 203, то можем да получим от него 2 много лесно – 2+0\*3, но ако не разглеждаме водеща нула, то при разбиване 2|03, ще разгледаме 03 като 3 и няма да видим, че можем да получим нула. Може да се забележи, че този проблем единствено се отразява на факта, че можем да получим 0 от число, от което не бихме очаквали. Затова в авторовите решения единствено се добавя подходяща нула, за да се отразява тази възможност.

*Автор: Димо Бунов*