Seven-digit numbers, which can be transformed into arithmetic expression, which is equal to 100

Yurii Lahodiuk
yura.lagodiuk@gmail.com

June 2017

As an auxiliary step, let's define a following *prefix* notation for the binary operations (for all rational numbers, $\forall a, b \in \mathbb{Q}$):

$$\left\{ f(a,b) \;\middle|\; f \in \{+,-,*,/\} \right\} \iff \left\{ a+b, a-b, a*b, \frac{a}{b} \right\} \tag{1}$$

Also, let's introduce the notation of the *left-* and *right- inverse* of binary operations ($\forall a, b, t \in \mathbb{Q}$):

$$\forall f \in \{+,-,*,/\}, f(a,b) = t \implies \begin{cases} f_L^{-1}(t,a) = b \\ f_R^{-1}(t,b) = a \end{cases} \tag{2}$$

In general, inverse operations are not bijective (e.g. the following expression $0 * x = 0$ is true for all values $x \in \mathbb{Q}$, hence the left inverse operation of multiplication in this case returns any number). Division by zero is not allowed as well, which leads to the additional restrictions. Nevertheless, let's postpone these details till the moment of implementation of a solution.

Let's consider a recursively-defined function $F(x, n)$, which returns a set of values of all possible arithmetic expressions, which can be obtained by placing binary operations and brackets between digits of the n-digit number x (unary minus is allowed as well):

$$F(x,n) = \begin{cases} \{0\}, & n = 1, x = 0 \\ \{-x, x\}, & n = 1, x \neq 0 \\ \bigcup\limits_{k=1}^{n-1} \left\{ f(a,b) \;\middle|\; f \in \{+,-,*,/\},\ (a,b) \in F\left(\left\lfloor \frac{x}{10^k} \right\rfloor, n-k\right) \times F\left(x \mod 10^k, k\right) \right\}, & n > 1 \end{cases} \tag{3}$$

where: x – considered number
n – amount of digits inside the considered number (because leading zeroes are allowed)
$F(...) \times F(...)$ – notation for the cartesian product of sets

As you can see, the solution of the problem is defined in terms of solutions of the smaller instances of the problem. Hence, *Dynamic Programming* can be used for calculation of values of the possible arithmetic expressions. Afterwards, in order to figure-out, whether the number 100 can be obtained by placing binary operations and brackets between digits of the 7-digit number x - it is just needed to check if $100 \in F(x, 7)$.

Unfortunately, the total cardinality of all memoized sets grows fast, and due to the intensive memory consumption described solution is not effective for 6- and 7- digit numbers.

Let's use the fact, that actually *we don't need to calculate the set of all values of all possible arithmetic expressions, and instead we just want to know, whether the target number is present inside this set.* Described fact leads to the following sketch of the strategy of computation:

1. Let's introduce the function $A(x, n, t)$, which indicates, whether n-digit number x can be transformed into the arithmetic expression, which evaluates to the target number t
2. For numbers with small amount of digits (when $n < 4$) it is still reasonable to check whether $t \in F(x, n)$ (hence, in this case the value of calculated directly $A(x, n, t) := t \in F(x, n)$)
3. For numbers with larger amount of digits the following approach is used:
   (a) For every $k \in \{1, 2, ..., n-1\}$ split n-digit number into k- and $(n-k)$- digit numbers: $(x \mod 10^k)$ and $\left\lfloor \frac{x}{10^k} \right\rfloor$
   (b) Assuming, that $k < (n-k)$, let's calculate the set of all possible values $F\left(x \mod 10^k, k\right)$ (according to the equation (3))
   (c) Now, for every value $y \in F(x, k)$ and for every binary operation $f \in \{+, -, *, /\}$ we need to check whether $A\left(\left\lfloor \frac{x}{10^k} \right\rfloor, n-k, f_R^{-1}(t, y)\right)$ is *true* (where $f_R^{-1}(t, y)$ denotes an inverse of the corresponding binary operation f)
   (d) A very similar schema of computation will be in case, when $k \geq (n-k)$

Below is presented the entire recurrence relation with all conditions:

$$A(x, n, t) = \begin{cases} true, & n = 1, t = x \\ false, & n = 1, t \neq x \\ true, & n \leq 4, t \in F(x, n) \\ false, & n \leq 4, t \notin F(x, n) \\ B(x, n, t, 1), & n > 4 \end{cases} \tag{4}$$

where: x – considered number
n – amount of digits inside the considered number (because leading zeroes are allowed)
t – target number

$B(x, n, t, k)$ represents an iteration over all $k \in \{1, 2, ..., n-1\}$ and defined as follows:

$$B(x, n, t, k) = \begin{cases} false, \ k = n \\ true, \ n-k \leq k, \exists a \in F\left(\left\lfloor \frac{x}{10^k} \right\rfloor, n-k\right), \exists f \in \{+, -, *, /\}, A\left(x \mod 10^k, k, f_L^{-1}(t, a)\right) = true \\ true, \ n-k > k, \exists a \in F\left(x \mod 10^k, k\right), \exists f \in \{+, -, *, /\}, A\left(\left\lfloor \frac{x}{10^k} \right\rfloor, n-k, f_R^{-1}(t, b)\right) = true \\ B(x, n, t, k+1) \end{cases} \tag{5}$$

So, in order to check, whether the number 100 can be obtained by placing binary operations and brackets between digits of the 7-digit number x - it is just needed to check if $A(x, 7, 100)$ is *true*.

The recurrence relation of $A(x, n, t)$ is defined in terms of the smaller instances of the problem, and *Dynamic Programming* can be used for calculation of its value as well. Also, in case of 7-digit numbers, within described approach - it is needed to calculate $F(x, n)$ only for such n, which are not greater than 4, which positively impacts the memory consumption.

The *Java* implementation of proposed solution allows to check $A(x, 7, 100)$ for all $x \in \{0, 1, ..., 10^7\}$ within around 25 minutes (on my laptop with 1.3 GHz processor and 512 MB heap size of the Java virtual machine).