Solution 1

Author: Marcin Pilipczuk

Definition 1. *A monopartite number is a positive integer whose decimal notation contains of k digits a, where $k \geq 1$, $1 \leq a \leq 9$. We will denote such a number by $M(k, a)$.*

Notation 1. *We will denote a bipartite number, whose decimal notation contains of $k \geq 1$ digits $1 \leq a \leq 9$ followed by $l \geq 1$ digits $0 \leq b \leq 9$ $(a \neq b)$ by $B(k, a, l, b)$.*

Corollary 1. *If $a < b$ then $B(k, a, l, b) = M(k+l, a) + M(l, b-a)$. If $a > b$, then $B(k, a, l, b) = M(k + l, a) - M(l, a - b)$.*

Let us fix question $p \geq 1$. Let $N(k, a) = M(k, a) \mod p$.

Theorem 1. *Let's consider sequence $N(k, a)_{k \geq 1}$. There exists $k_0(a) \leq p$ and period length $c(a) \leq p$ such that for every integer $k \geq k_0$ we have $N(k, a) = N(k + c(a), a)$. Moreover, $k_0(a) + c(a) \leq p + 1$*

Proof. Notice that $M(k, a) \cdot 10 + a = M(k + 1, a)$. Therefore $N(k + 1, a)$ depends only on $N(k, a)$, so if $N(k, a) = N(l, a)$, then for every integer $i \geq 0$ we have $N(k + i, a) = N(l + i, a)$. $N(k, a)$ attains only p values $(0, 1, \ldots, p - 1)$, so for $k = 1, 2, \ldots, p + 1$ some value $N(k, a)$ will occure at least twice. Let's denote the position of first repetition by $k_0(a)$ and $k_0(a) + c(a)$. So, starting at position $k_0(a)$ the sequence $N(k, a)$ will be periodic with period $c(a)$. Since the first repetition will be at last at position $p + 1$, so $k_0(a) \leq p$.

Corollary 2. *Let's consider the following equation: $N(k, a) = x$, where a and x are given and k is the variable. Then such k doesn't exists or there is only one such solution k satisfying $k < k_0(a)$ or all the solutions form arithmetic sequence $k + i \cdot c(a)$.*

Corollary 3. *In time $O(9p)$ we can create data structure of size $O(9p)$, which will in constant time solve equations $N(k, a) = x$ with variable k.*

Proof. For every digit a we can follow sequence $N(k, a)$ using array $A[0 \ldots p - 1]$. At the beginning, the array is filled with zeros. We calculate $N(k, a)$ for $k = 1, 2, 3, \ldots$ and we write k into cell $A[N(k, a)]$. When we discover that $A[N(k, a)]$ was positive already, we found $k_0(a)$ and $c(a)$. We save these values and tha array A for digit a. Having these values, solving equation $N(k, a) = x$ is easy. If $A[x] = 0$, then there is no solution. If $A[x] < k_0(a)$ then there is only one solution $A[x]$. In other case, the sequence $A[x] + c(a) \cdot i$ is the solution.

Corollary 4. *We can enhance aforementioned structure with operations ,,greatest solution lesser than y" and ,,smallest solution greater than y".*

Fact 1. *If $B(k, a, l, b) = np$, where $n > 1$, and d is thefirst digit in the decimal notation of p, and m is the length of hte decimal notation of p, then $a > d$ or $k + l > m$.*

Proof. $k + l \geq m$, since $B(k, a, l, b) > p$. If $k + l = m$, then since $B(k, a, l, b) - p \geq p$, the first digit of $B(k, a, l, b)$ should be at least twice as big as first digit of p.

Theorem 2. *We can solve question p in time $O(81p)$ and memory $O(9p)$.*

Proof. Firstly, we create data structure known from Corollary 4. We know, that the solution is in form $M(k + l, a) + M(l, b - a)$ or $M(k + l, a) - M(l, a - b)$. Our algoritm will work in the following schema: for every remainder $0 \leq r < p$, for every digit $1 \leq a \leq 9$, for every digit $0 \leq b \leq 9$, $b \neq a$, we will try to find smallest possible solution with additionally $N(k+l, a) = r$.

We do it as follows. Since $N(k+l, a) = r$, we know what should be $N(l, b-a)$ and $N(l, a-b)$. Using our data structure, we find smallest possible such l. Now we find smallest possible $k+l$, that $k+l > l$ and $k+l \geq m$ (where m is the length of the decimal notation of p, d is the first digit of the decimal notation of p) if $d < a$, and $k+l > m$ if $d \geq a$. Now, if $b < a$, we find greatest possible l' that $l' < k+l$ and $N(l, a-b) = N(l', a-b)$; if $b > a$, let $l' = l$. Now $B(k, a,' l, b)$ is the smallest sollution satisfying $N(k+l', a) = r$, because we chose smallest possible $k+l$ and then if $b < a$ we tried to substract as much as possible.

By taking minimum of solutions for all r, a, b we get the general solution.

This solution is implemented in file $BipartiteNumbers.java$.

Solution 2

Author: Michal Pilipczuk

DEFINITION.1. A *monopartite number*, denoted by $M(k, a)$, will be a number with form: $\underbrace{\overline{aa\ldots a}}_{k}$ where a is a nonzero digit and $k \in \mathbb{Z}^+$.

DEFINITION.2. A *bipartite number*, denoted by $B(k, a, l, b)$, will be a number with form: $\underbrace{\overline{aa\ldots a}}_{k}\underbrace{\overline{bb\ldots b}}_{l}$ where a is a nonzero digit, b is a digit not equal to a and $k, l \in \mathbb{Z}^+$.

FACT.1. If for a bipartite number $B(k, a, l, b)$ condition $a < b$ is fulfilled then $B(k, a, l, b) = M(k + l, a) + M(l, b - a)$. If $a > b$ then $B(k, a, l, b) = M(k + l, a) - M(l, a - b)$.

Proof: Using school-based adding and subtracting algorithms the thesis is obvious.

CONCLUSION.1. Let p be the number we are asked in the problem (a number for which the smallest greater bipartite multiplication is to be determined). Assuming this, problem of finding smallest $B(k, a, l, b) > p$ divisible by p is the same as problem of finding two monopartite numbers $M(k, a)$ i $M(l, b)$ having the same or opposite remainders modulo p, thus those numbers will sum or subtract (respectively if the remainders are opposite or the same) to a proper bipartite number an this bipartite number will be smallest possible, greater than p and divisible by p.

DEFINITION.3. Let $N(k, a) = M(k, a) mod p$.

FACT.2. A sequence of numbers $N(0, a), N(1, a), N(2, a), \ldots$ is from a certain element cyclic and after $p + 1$ elements the first cycle will be ended.

Proof: Obviously $N(k + 1, a) = (10 \cdot N(k, a) + a) mod p$, so consecutive elements of the sequence are recursively determined only by previous elements. Because of this, if for two different indexes x, y the elements $N(x, a)$ and $N(y, a)$ will be equal, then the sequence will be built from x and from y in the same way, so there will be a cycle with length $|x - y|$. From pigeons' holes theorem after first $p+1$ elements of the sequence there will be two same, So after at most $p + 1$ elements the first cycle will be ended.

DEFINITION.4. Let $len(a)$ be a length of decimal representation of a.

FACT.3. For every $k > 9$ a inequality occurs: $len(ka) > len(a)$.

Proof: Function $len(x)$ is obviously non-decreasing, so the thesis should be only proven for $k = 10$. In this situation $10 \cdot a$ is constructed by adding a zero on the end of decimal representation of a, so length $10 \cdot a$ is greater than the length of a.

FACT.3. Let as assume, that we consider all bipartite numbers of length s and that all smaller lengths were considered previously. Then we can using in the whole algorithm memory complexity $O(18p)$ answer in time complexity $O(1)$ the questions:

1. What is the smallest number $k < s$, for which $N(k, a) = r$ for given r and a?
2. What is the largest number $k < s$, for which $N(k, a) = r$ for given r and a?

Method: To answer these questions we will use arrays `MN[0..p][1..9]` and `MX[0..p][1..9]`. In cell `MN[x][a]` there will be stored the smallest already found k so $N(k, a) = x$, and in cell

`MX[x][a]` the largest. At the beginning the arrays are filled with value -1 which means that no remainder has been yet found. When considering a length s of a bipartite number we can in time complexity $O(18)$ actualize both arrays and every question has time complexity $O(1)$.

ALGORITHM

1. In time complexity $O(1)$ we find out if numbers $2p, 3p, \ldots, 9p$ are bipartite by analyzing their digits put into an array. If yes, then the smallest bipartite will be an answer. Otherwise we go to point 2.

2. We consider consecutive possible lengths of bipartite numbers beginning with 1. Let s be considered length at the moment. Firstly, we count numbers $N(s, a)$ for every digit a using recursive rule from proof of fact 2. After this, if $s \leq len(p)$ then we only actualize arrays `MX` i `MN`. Otherwise, for consecutive digits a from 1 to 9 and for consecutive digits b from 1 to a we check if there exists such a $N(t, b)$ that $N(t, b) = N(s, a)$. Using array `MX` we will always obtain the largest such a number. If such t, b exist then we memorize such a pair t, b for which $M(s, a) - M(t, b)$ is the smallest possible. After that analogically we consider all pairs (a, b) where a is from 1 to 9 and b is from 1 to $9 - a$ and we seek the smallest such a t that $N(t, b) = p - N(s, a)$ using array `MN`. We memorize such pair t, b for which $M(s, a) + M(t, b)$ is smallest possible. If after both checks i.e. subtracting a monopartite number or adding a monopartite number we obtained a possible pair t, b, then the answer is a found pair that builds the lowest bipartite number $B(s - t, a, t, \pm(b - a))$ and we end iterations. Otherwise we just actualize arrays `MX` i `MN` and continue to the next possible length.

Proof of correctness: If smallest possible bipartite number divisible by p and greater than p will have form $2p, 3p, \ldots, 9p$ then it'll be found in point 1 of the algorithm. Otherwise form fact 3 we conclude that the length of smallest possible bipartite multiplication of p will be greater than of p, so from conclusion 1 as well as from the construction of the algorithm it can be easily obtained that found bipartite number is divisible by p and smallest possible (as we minimize its length and then in every iteration it itself). Iterations will end after at most $p + 1$ steps, as from pigeons' holes theorem for $B(k, 1, l, 0)$ from fact 2 we can conclude, that after at most $p+1$ steps from numbers $M(k, 1)$ we will find two with same remainder modulo p, which subtraction is a bipartite number divisible by p. Therefore the stop condition is fulfilled as well.

Complexity estimation: The memory complexity is $O(18p)$ because of the use of arrays `MN` i `MX`. The first step of the algorithm has time complexity $O(1)$ and every iteration in the second step $O(81)$(because every pair of digits (a, b) needs to be considered). The iterations will be processed at most $O(p)$ times, therefore total time complexity is $O(81p)$.

Tests

10 test have been created, each consisting of from 10 to 15 cases. As input data is a single positive integer number up to 100.000 then the best method of verification of tests and model solutions is to check their correctness for all possible input numbers. The solution based on described algorithm is implemented in file `BipartiteNumbers.cpp` and it should take a few minutes for it to generate output for every test case from 1 to 100.000.