Rozwiązanie wzorcowe nr 1

Autor: Marcin Pilipczuk

Definicja 1. *Liczbą pojedynczą nazywamy liczbę całkowitą dodatnią, która w zapisie dziesiętnym składa się tylko z k cyfr a, gdzie $k \geqslant 1$, $1 \leqslant a \leqslant 9$. Taką liczbę będziemy w skrócie zapisywać jako $M(k, a)$.*

Oznaczenie 1. *Liczbę podwójną, składającą się z $k \geqslant 1$ cyfr $1 \leqslant a \leqslant 9$, po których następuje $l \geqslant 1$ cyfr $0 \leqslant b \leqslant 9$ $(a \neq b)$, oznaczamy $B(k, a, l, b)$.*

Wniosek 1. *Jeśli $a < b$ to $B(k, a, l, b) = M(k+l, a) + M(l, b-a)$. Jeśli $a > b$, to $B(k, a, l, b) = M(k + l, a) - M(l, a - b)$.*

Ustalmy zapytanie w zadaniu $p \geqslant 1$. Niech $N(k, a) = M(k, a) \mod p$.

Twierdzenie 1. *W ciągu $N(k, a)_{k \geqslant 1}$ istnieje wartość $k_0(a) \leqslant p$ i długość cyklu $c(a) \leqslant p$, że dla każdego $k \geqslant k_0$ zachodzi $N(k, a) = N(k + c(a), a)$. Dodatkowo $k_0(a) + c(a) \leqslant p + 1$*

Dowód. Zauważmy, że $M(k, a) \cdot 10 + a = M(k+1, a)$. Czyli $N(k+1, a)$ zależy tylko do $N(k, a)$. W związku z tym jeśli $N(k, a) = N(l, a)$, to dla każdego $i \geqslant 0$ mamy $N(k + i, a) = N(l + i, a)$. $N(k, a)$ przyjmuje tylko p wartości, w związku z tym po dla $k = 1, 2, \ldots, p + 1$ jakaś wartość się powtórzy. Niech pierwsze powtórzenie będzie na pozycjach $k_0(a)$ i $k_0(a) + c(a)$. Na mocy poprzednich obserwacji, od pozycji $k_0(a)$ ciąg $N(k, a)$ będzie cykliczny z okresem $c(a)$. Jako, że powtórka będzie na pozycji nie później niż $p + 1$, to $k_0(a) \leqslant p$.

Wniosek 2. *Rozważamy problem $N(k, a) = x$ — mamy dane a, x, szukamy k. To albo takie k nie istnieje, albo istnieje dokładnie jedno rozwiązanie $k < k_0(a)$, albo rozwiązania tworzą ciąg arytmetyczny $k + i \cdot c(a)$.*

Wniosek 3. *W czasie $O(9p)$ jesteśmy w stanie stworzyć strukturę danych wielkości $O(9p)$, która pozwala w czasie $O(1)$ rozwiązać równanie $N(k, a) = x$ z niewiadomą k.*

Dowód. Dla każdej cyfry a śledzimy ciąg $N(k, a)$ na tablicy $A[0 \ldots p - 1]$. Na początku tablica jest wyzerowana. Kolejno liczymy $N(k, a)$ dla kolejnych k i wpisujemy warotści k w pola $A[N(k, a)]$. W momencie, jeśli w miejscu zapisu coś już było, znaleźliśmy najmniejsze możliwe wartości $k_0(a)$ i $c(a)$. Zapamiętujemy te wartości oraz tablicę A dla cyfry a. Rozwiązujemy równanie przez sprawdzenie wartości $A[x]$. Jeśli jest to 0, nie ma rozwiązań. Jeśli $A[x] < k_0(a)$, rozwiązaniem jest tylko $A[x]$. Jeśli $A[x] \geqslant k_0(a)$ rozwiązaniem jest ciąg $A[x] + c(a) \cdot i$.

Wniosek 4. *Możemy rozszerzyć powyższą strukturę danych o operacje „największe rozwiązanie mniejsze niż y" i „najmniejsze rozwiązanie większe niż y" (z możliwą odpowiedzią „nie ma").*

Fakt 1. *Jeśli $B(k, a, l, b) = np$, gdzie $n > 1$, oraz d jest pierwszą cyfrą w zapisie dziesiętnym liczby p, a m jest długością zapisu dziesiętnego liczby p, to $a > d$ lub $k + l > m$.*

Dowód. $k + l \geqslant m$, bo $B(k, a, l, b) > p$. Jeśliby $k + l = m$, to jako że $B(k, a, l, b) - p \geqslant p$, pierwsza cyfra $B(k, a, l, b)$ musi być przynajmniej dwa razy większa od pierwszej cyfry p.

Twierdzenie 2. *Potrafimy odpowiedzieć na zapytanie p w czasie $O(81p)$ i pamięci $O(9p)$.*

Dowód. Skonstruujmy strukturę danych z wniosku 4. Wiemy, że rozwiązanie jest postaci $M(k+l, a) + M(l, b - a)$ lub $M(k + l, a) - M(l, a - b)$. Algorytm działa w następujący sposób: dla każdej reszty $0 \leqslant r < p$, dla każdej cyfry $1 \leqslant a \leqslant 9$, dla każdej cyfry $0 \leqslant b \leqslant 9$, $b \neq a$, próbujemy znaleźć najmniejsze rozwiązanie w postaci jak wyżej, dla $N(k + l, a) = r$.

Robimy to w sposób następujący. Wiemy, jaką resztę z dzielenia przez p musi mieć $M(l, b-a)$ lub $M(l, a-b)$. Znajdujemy najmniejsze możliwe takie l. Znajdujemy teraz najmniejsze możliwe takie $k+l$, że $k+l > l$ i $k+l \geqslant m$ (m - długość w zapisie dziesiętnym liczby p, d - pierwsza cyfra zapisu dziesiętnego liczby p) jeśli $d < a$, zaś $k+l > m$ dla $d \geqslant a$. Teraz, jeśli $b < a$, znajdujemy największe nowe l', że $l' < k+l$ i $N(l, a-b) = N(l', a-b)$. Jest to najmniejsze rozwiązanie dla cyfr a i b i $N(k+l, a) = r$, większe od p: wybieramy najmniejsze możliwe $k+l$, po czym, jeśli $b < a$, próbujemy odjąć jak najwięcej.

Biorąc minimum ze wszystkich rozwiązań dla każdego r, a, b, znajdujemy odpowiedź.

Implementacja tego rozwiązania znajduje się w pliku $BipartiteNumbers.java$.

Rozwiązanie wzorcowe nr 2

Autor: Michał Pilipczuk

DEFINICJA.1. *Liczbą pojedyńczą* oznaczoną przez $M(k,a)$ oznaczymy liczbę postaci $\overline{\underbrace{aa\ldots a}_{k}}$ gdzie a jest cyfrą różną od 0 a $k \in \mathbb{Z}^+$.

DEFINICJA.2. *Liczbą podwójną* oznaczoną przez $B(k,a,l,b)$ oznaczymy liczbę postaci $\overline{\underbrace{aa\ldots a}_{k}\underbrace{bb\ldots b}_{l}}$ gdzie a jest cyfrą różną od 0, b cyfrą różną od a, zaś $k,l \in \mathbb{Z}^+$.

FAKT.1. Jeśli w liczbie podwójnej $B(k,a,l,b)$ zachodzi $a < b$ to $B(k,a,l,b) = M(k+l,a) + M(l,b-a)$. Jeśli zaś $a > b$ to $B(k,a,l,b) = M(k+l,a) - M(l,a-b)$.

Dowód: Poprzez zastosowanie dodawania bądź odejmowania pod kreską od razu uzykujemy tezę.

WNIOSEK.1. Niech p zapytaniem w zadaniu (liczbą dla której mamy podać odpowiedź). Wówczas problem znaleźienia najmniejszej $B(k,a,l,b) > p$ podzielnej przez p sprowadza się do znalezienia dwóch liczb postaci $M(k,a)$ i $M(l,b)$ dających modulo p reszty zgodne bądź przeciwne tak, by liczby te sumowały się bądź odejmowały (zależnie od tego czy reszty są odpowiednio przeciwne czy zgodne) do poprawnej liczby podwójnej i by ta liczba podwójna była podzielna przez p i od p większa.

DEFINICJA.3. Oznaczmy $N(k,a) = M(k,a) mod p$.

FAKT.2. Ciąg liczb $N(0,a), N(1,a), N(2,a), \ldots$ jest od pewnego miejsca i po $p+1$ pierwszych elementach pierwszy cykl się już skończy.

Dowód: Zauważmy, że $N(k+1,a) = (10 \cdot N(k,a) + a) mod p$, więc kolejne wyrazy ciągu są rekurencyjnie związane tylko z jednym poprzednim wyrazem. W związku z tym jeśli w dwóch miejscach wyrazy $N(x,a)$ i $N(y,a)$ będą równe (x różne od y), to ciąg iterowany od x i od y będzie taki sam, więc ciąg będzie miał okres $|x-y|$. Z zasady szufladkowej Dirichleta wśród $p+1$ pierwszych wyrazów ciągu znajdą się dwa takie same, więc po co najwyżej $p+1$ wyrazach zakończy się pierwszy cykl.

DEFINICJA.4. Oznaczmy przez $len(a)$ długość zapisu dzieśiętnego liczby a.

FAKT.3. Dla każdego $k > 9$ zachodzi $len(ka) > len(a)$.

Dowód Funkcja $len(x)$ jest oczywiście niemalejąca, więc wystarczy sprawdzić tezę dla $k = 10$. Wtedy $10 \cdot a$ powstaje w wyniku dopisania zera na koniec a, więc długość $10 \cdot a$ jest większa o 1 od długości a.

FAKT.3. Załóżmy, że rozważamy liczby podwójne długości s, przy czym wcześniej w trakcie algorytmu przetworzyliśmy już wszystkie długości mniejsze od s. Wówczas możemy używając sumarycznie w czasie algorytmu pamięci $O(18p)$ za każdym razem w czasie $O(1)$ uzyskać odpowiedź na pytania:

1. Jaka jest najmniejsza liczba $k < s$, że $N(k,a) = r$ dla ustalonych r i a?
2. Jaka jest największa liczba $k < s$, że $N(k,a) = r$ dla ustalonych r i a?

Sposób: Do odpowiedzi na te pytania służą tablice `MN[0..p][1..9]` oraz `MX[0..p][1..9]`. W komórce `MN[x][a]` trzymamy najmniejsze takie k, że $N(k,a) = x$, zaś w komórce `MX[x][a]` największe. Na początku wszystkie komórki obu tablic ustawione są na -1 co oznacza, że dana reszta jeszcze nie wystąpiła. Możemy dla każdej przetwarzanej długości liczby podwójnej aktualizować w czasie $O(18)$ obie tablice zaś każde zapytanie zajmuje czas $O(1)$.

ALGORYTM

1. Sprawdzamy w czasie $O(1)$ czy liczby $2p, 3p, \ldots, 9p$ nie są podwójne poprzez wpisanie ich cyfr do tablicy i przeanalizowanie. Jeśli tak, to zwracamy najmniejszą z nich, w przeciwnym razie idziemy do punktu 2.

2. Po kolei przeglądamy możliwe długości liczb podwójnych. Niech s to aktualnie przetwarzana wartość. Najpierw obliczamy wyrazy $N(s,a)$ gdzie a jest cyfrą na podstawie poprzednio wyliczonych za pomocą rekurencji z dowodu faktu 2. Jeśli $s \leqslant len(p)$ to tylko aktualizujemy tablice `MX` i `MN`. W przeciwnym razie dla kolejnych cyfr a przebiegających zbiór od 1 do 9 sprawdzamy dla kolejnych możliwych b z przedziału od 1 do a czy istnieje liczba $N(t,b)$, że $N(t,b) = N(s,a)$. Odtwarzając wyniki z tablicy `MX` zawsze uzyskamy największą taką liczbę. Jeśli takie t, b istnieją to zapamiętujemy taką parę dla której $M(s,a) - M(t,b)$ jest najmniejsze możliwe. Następnie analogicznie rozważamy kolejne b z przedziału od 1 do $9 - a$ i szukamy najmniejszego takiego t, że $N(t,b) = p - N(s,a)$. Zapamiętujemy wynik dla takiej pary t, b, dla której $M(s,a) + M(t,b)$ jest najmniejsza możliwa używając tablicy `MN`. Jeśli podczas obu sprawdzeń tzn. odjęcia liczby pojedyńczej bądź dodania liczby pojedyńczej uzyskaliśmy możliwą parę t, b, to wynikiem jest najmniejsza utowrzona przez znalezione pary liczba podwójna $B(s - t, a, t, \pm(a - b))$ i kończymy iterację. W przeciwnym razie aktualizujemy tablice `MX` i `MN` i przechodzimy do następnej rozważanej długości.

Dowód poprawności: Gdyby najmniejsza możliwa liczba podwójna podzielna przez p i większa od niej była postaci $2p, 3p, \ldots, 9p$ to zostanie znaleziona w punkcie 1 algorytmu. W przeciwnym razie z faktu 3 wynika że długość najmniejszej wielokrotności będzie większa niż długość p, więc z wniosku 1 jak i z budowy algorytmu wynika, że znaleziona liczba podwójna jest podzielna przez p jak i jest najmniejsza możliwa (bo minimalizujemy jej długość a następnie w każdej iteracji nią samą). Iteracje się zakończą po co najwyżej $p + 1$ krokach, gdyż z zasady szufladkowej Dirichleta rozważając tylko liczby postaci $B(k, 1, l, 0)$ z faktu 2 wynika, że po co najwyżej $p + 1$ krokach wśród liczb $M(k, 1)$ znajdziemy dwie o takiej samej reszcie modulo p, których rożnica da nam liczbę podówjną o co najwyżej p cyfrach, więc warunek stopu też jest zachowany.

Szacowanie złożoności: Złożoność pamięciowa wynosi $O(18p)$ ze względu na tablice `MN` i `MX`. Pierwszy krok algorytmu ma złożoność obliczeniową $O(1)$ zaś każda iteracja w drugim $O(81)$(gdyż muszę rozważyć każdą parę cyfr a, b), zaś iteracje wykonują się $O(p)$ razy. Zatem sumaryczna złożoność obliczeniowa wynosi $O(81p)$.

Testy

Zostało stworzonych 10 testów każdy zawierający od 10 do 15 możliwych przypadków. Ponieważ danymi wejściowymi do zadania jest jedna liczba od 1 do 100.000 to najłatwiej zweryfikować programy wzrocowe poprzez porównanie rozwiązań dla wszystkich możliwych danych wejściowych. Rozwiązanie zaimplementowane używając opisanego algorytmu znajduje się w pliku `BipartiteNumbers.cpp` i wygenerowanie wszystkich możliwych rozwiązań zajmuje mu kilka minut.