Wojciech Rytter
Treść zadania, Opracowanie

Przemysław Horban
Program

Dostępna pamięć: 128 MB.

OI, Etap III, dzień drugi, 15.04.2010

Jedynki

Niech x będzie ciągiem zer i jedynek. **Skrajnie samotną jedynką** *(w skrócie SKS jedynką) w x jest skrajna (ostatnia lub pierwsza) jedynka, która dodatkowo nie sąsiaduje bezpośrednio z żadną inną jedynką. Na przykład, w ciągu* `10001010` *są dwie SKS jedynki, w ciągu* `1101011000` *nie ma żadnej SKS jedynki, a w ciągu* `1000` *jest tylko jedna SKS jedynka.*

Oznaczmy przez $sks(n)$ sumaryczną liczbę SKS jedynek w reprezentacjach binarnych liczb od 1 do n. Na przykład, $sks(5) = 5$, $sks(64) = 59$, $sks(128) = 122$, $sks(256) = 249$.

Chcemy przetwarzać bardzo duże liczby. Będziemy je więc reprezentować w postaci **zwartej**. *Jeśli x jest dodatnią liczbą całkowitą, $(x)_2$ jest jej zapisem binarnym (zaczynającym się od 1), to zwartą reprezentacją x jest ciąg $REP(x)$ złożony z dodatnich liczb całkowitych, które odpowiadają długościom kolejnych bloków takich samych cyfr. Na przykład:*

$$REP(460\,288) = REP(1110000011000000000_2) = (3,5,2,9)$$
$$REP(408) = REP(110011000_2) = (2,2,2,3)$$

Twoim zadaniem jest napisanie programu, który oblicza ciąg $REP(sks(n))$ na podstawie $REP(n)$.

Wejście

Pierwszy wiersz standardowego wejścia zawiera jedną liczbę całkowitą k ($1 \leqslant k \leqslant 1\,000\,000$), oznaczającą długość zwartej reprezentacji dodatniej liczby całkowitej n. Drugi wiersz zawiera k liczb całkowitych $x_1, x_2, \ldots, x_k$ ($0 < x_i \leqslant 1\,000\,000\,000$) pooddzielanych pojedynczymi odstępami. Ciąg liczb $x_1, x_2, \ldots, x_k$ jest zwartą reprezentacją dodatniej liczby całkowitej n. Dodatkowo możesz założyć, że $x_1 + x_2 + \ldots + x_k \leqslant 1\,000\,000\,000$, czyli $0 < n < 2^{1\,000\,000\,000}$.

Wyjście

Twój program powinien wypisać na standardowe wyjście dwa wiersze. W pierwszym z nich powinna znajdować się jedna dodatnia liczba całkowita l. W drugim wierszu powinno znaleźć się l dodatnich liczb całkowitych $y_1, y_2, \ldots, y_l$, pooddzielanych pojedynczymi odstępami. Ciąg $y_1, y_2, \ldots, y_l$ powinien być zwartą reprezentacją liczby $sks(n)$.

Przykład

Dla danych wejściowych:	*poprawnym wynikiem jest:*
`6`	`5`
`1 1 1 1 1 1`	`1 1 2 1 1`

Wyjaśnienie do przykładu: *Ciąg $1, 1, 1, 1, 1, 1$ jest zwartą reprezentacją $101010_2 = 42$, $sks(42) = 45$, natomiast $45 = 101101_2$ ma zwartą reprezentację $1, 1, 2, 1, 1$.*

Rozwiązanie

Wprowadzenie

W tym zadaniu będziemy mieli do czynienia z bardzo elementarną kombinatoryką i nietrywialną algorytmiką, wymuszoną zwartymi reprezentacjami.

Najprościej jest obliczyć osobno liczbę $ls(n)$ liczb nie większych niż n, które mają lewe samotne jedynki, oraz liczbę $ps(n)$ tych, które mają prawe samotne jedynki. Następnie należy odjąć (trywialną do obliczenia) liczbę $cs(n)$ tych liczb nie większych niż n, które mają samotne jedynki będące zarazem lewostronnymi i prawostronnymi (są to wyłącznie liczby mające zapis binarny postaci $(1000\ldots0)_2$). W ten sposób otrzymujemy wzór:

$$wynik \;=\; sks(n) \;=\; ls(n) + ps(n) - cs(n). \tag{1}$$

Jak znaleźć wzory na $ls(n)$ i $ps(n)$? Można zacząć od chwili eksperymentowania, a potem po prostu wpaść na dobre wzory (przecież na zawodach nie trzeba niczego udowadniać). Często, przynajmniej w szczególnych przypadkach, wzory bywają łatwe do dostrzeżenia, wystarczy obliczyć wynik ręcznie dla małych danych i od razu je „widać".

Przykładowo, nietrudno zauważyć (będzie to także szczególnym przypadkiem twierdzeń z następnej sekcji), że zachodzi

$$ls(2^t - 1) \;=\; ps(2^t - 1) \;=\; 2^{t-1}. \tag{2}$$

Wyprowadzenie wzorów

Oznaczmy $t = t(n) = \lfloor \log_2 n \rfloor$. Wówczas zapis binarny liczby n ma $t+1$ bitów. Niech $k = k(n)$ oznacza rozmiar zwartej reprezentacji liczby n.

Zacznijmy od oczywistego wzoru na $cs(n)$.

Twierdzenie 1.

$$cs(n) = t + 1. \tag{3}$$

Liczbę $ls(n)$ także oblicza się całkiem łatwo — jest to liczba tych liczb nieprzekraczających n, których zapis binarny zaczyna się od 10.

Twierdzenie 2. *$ls(1) = 1$, natomiast dla $n \geqslant 2$:*

(a) jeśli zapis binarny liczby n zaczyna się od 11*, to $ls(n) = 2^t$,*

(b) w przeciwnym wypadku $ls(n) = n - 2^{t-1} + 1$.

Dowód: (a) Lewą samotną jedynkę mają wszystkie liczby $(t+1)$-cyfrowe zaczynające się od 10 (takich liczb jest 2^{t-1}) oraz pewne liczby o mniej niż $t+1$ cyfrach, których jest $ls(2^t - 1) = 2^{t-1}$, co można pokazać chociażby przez indukcję. W sumie mamy 2^t liczb.

(b) Tym razem obliczymy, ile liczb nieprzekraczających n *nie ma* lewej samotnej jedynki. Wszystkie takie liczby mają co najwyżej t cyfr, a zatem jest ich dokładnie

$$2^t - 1 - ls(2^t - 1) = 2^{t-1} - 1.$$

Stąd pozostałych liczb jest $n - (2^{t-1} - 1)$. ■

Zdecydowanie najciekawsze jest wyprowadzenie wzoru na liczbę samotnych prawostronnych jedynek. Ta część rozwiązania sprawiła także najwięcej kłopotów zawodnikom, którzy próbując ją wykonać, często grzęźli w żmudnych rachunkach.

Twierdzenie 3.

$$ps(n) = \frac{1}{2}(n + k) \tag{4}$$

(Zauważmy, że równanie (2) *to szczególny przypadek równania* (4) *dla* $k = 1$*).*

Dowód: Do powyższego wzoru można dojść różnymi drogami, jedną z nich jest wielokrotne stosowanie równania (2). Wtedy trzeba posumować pewną liczbę postępów geometrycznych (potęgi dwójki). Pokażemy inną metodę, mniej rachunkową.

Niech $Z(n)$ będzie zbiorem tych x, $x \leqslant n$, które mają samotną prawostronną jedynkę. Oznaczmy $nps(n) = n - ps(n)$; jest to liczba liczb $x \leqslant n$, które nie mają samotnej prawostronnej jedynki. Zamiast równości (4) udowodnimy następującą, równoważną równość:

$$ps(n) - nps(n) \;=\; k. \tag{5}$$

Zdefiniujemy funkcję f, która każdej liczbie mającej samotną prawostronną jedynkę przyporządkowuje liczbę jej nieposiadającą. Działamy na reprezentacjach binarnych (niezwartych). Dodatkowo, dla uproszczenia dalszych rozważań, do reprezentacji wszystkich rozważanych liczb dopiszmy jedno zero wiodące. Niech teraz $(\alpha\ 01\ 0\ldots0)_2$ będzie binarną reprezentacją x (α jest tu, być może pustym, ciągiem zer i jedynek). Wtedy

$$f(x) \;=\; f\big((\alpha 01\underbrace{0\ldots0}_{l})_2\big) \;\stackrel{\text{def}}{=}\; (\alpha 11\underbrace{0\ldots0}_{l})_2.$$

Okazuje się, że f jest bijekcją. Nietrudno dostrzec, że jest różnowartościowa. Aby pokazać, że jest „na", weźmy liczbę y, która nie ma prawej samotnej jedynki. Taka liczba musi być postaci $y = (\beta 110\ldots0)_2$. Stąd oczywiście $y = f(\beta 010\ldots0)_2$.

Niech

$$Z'(n) \;=\; \{x \in Z(n) \;:\; f(x) > n\}.$$

Ponieważ dla każdego x zachodzi $f(x) > x$, więc f odwzorowuje (w sposób różnowartościowy) zbiór $Z(n) \setminus Z'(n)$ na cały zbiór $\{1, 2, \ldots, n\} \setminus Z(n)$. Zatem wystarczy teraz udowodnić, że $|Z'(n)| = k$.

Zauważmy, że:

$$Z'(n) \;=\; \{(\alpha 01\ \underbrace{0\ldots0}_{t(n)-|\alpha|})_2 \;:\; \alpha 01 \text{ lub } \alpha 10 \text{ jest prefiksem } n \text{ w zapisie binarnym}\}.$$

Uzasadnienie powyższej równości nie jest trudne. $Z'(n)$ składa się bowiem z dokładnie takich x, że $x \leqslant n < f(x)$, czyli

$$(\alpha 01 \underbrace{0\ldots0}_{t(n)-|\alpha|})_2 \leqslant n < (\alpha 11 \underbrace{0\ldots0}_{t(n)-|\alpha|})_2.$$

Ile, wobec tego, liczb zawiera $Z'(n)$? Każda z nich odpowiada zapisowi $(n)_2$ w postaci $\alpha 01\gamma$ lub $\alpha 10\gamma$ dla pewnych ciągów zerojedynkowych α i γ [1]. Takich podziałów jest dokładnie tyle, ile bloków w reprezentacji binarnej n (bez zer wiodących), czyli $k(n)$. Stąd

$$|Z'(n)| = k = k(n).$$

Przykład.

$$\begin{aligned} Z'(0111000111000_2) \;=\; \{\; & 0100000000000_2,\; 0111000100000_2, \\ & 0110100000000_2,\; 0111000110100_2 \;\}. \end{aligned}$$

■

Arytmetyka zwartych reprezentacji

Istotnym elementem zadania jest przetwarzanie olbrzymich liczb w postaci zwartej. Ponieważ wszystko jest związane tu jakoś z potęgami dwójki, więc złożoność czasowa operacji na zwartych reprezentacjach będzie liniowa ze względu na ich rozmiar. Zauważmy, że do obliczenia wyniku — wzór (1), do którego podstawiamy równości podane w Twierdzeniach 1, 2 i 3 — potrzebujemy jedynie zaimplementować operacje: dodawania, odejmowania, dzielenia przez 2 i obliczania reprezentacji liczb postaci 2^t. Dwie ostatnie są już naprawdę proste, więc opiszemy jedynie dodawanie i odejmowanie.

Dodawanie zwartych reprezentacji zaimplementujemy, wzorując się na dodawaniu pisemnym. Zapiszmy sumowane liczby a i b jedna pod drugą, wyrównując do prawej, do dłuższej z tych liczb dopiszmy na początku jedno zero wiodące, a do krótszej — tyle zer wiodących, żeby ich długości wyrównały się. *Miejscami podziału* przy sumowaniu nazwiemy wszystkie granice między kolejnymi blokami takich samych cyfr, czy to w ramach liczby a, czy b.

Sumę $a + b$ będziemy obliczać od prawej do lewej, rozpatrując fragmenty dodawanych liczb między kolejnymi miejscami podziału. Zauważmy, że wówczas problem sprowadza nam się do zsumowania dwóch bloków, z których każdy jest złożony tylko z zer lub tylko z jedynek; musimy też wziąć pod uwagę bit (cyfrę) przeniesienia, wynikły z sumowania poprzednich bloków. W wyniku otrzymujemy nowy blok lub ewentualnie dwa bloki, a także bit przeniesienia, którego używamy przy sumowaniu kolejnej pary bloków, patrz rys. 1.

Widać wyraźnie, że stosując powyższy przepis, możemy obliczyć zwartą reprezentację liczby $a + b$ w czasie $O(k(a) + k(b))$, i co więcej, $k(a + b) = O(k(a) + k(b))$. Przykład działania tego algorytmu znajduje się na rys. 2.

[1] Przypomnijmy, że przed pierwszą jedynką w reprezentacji rozważanych liczb dopisaliśmy 0.

blok 1	**blok 2**	**bit**	**wynikowy blok**	**wynikowy bit**
0...00	0...00	0	0...00	0
0...00	0...00	1	0...01	0
0...00	1...11	0	1...11	0
0...00	1...11	1	0...00	1
1...11	1...11	0	1...10	1
1...11	1...11	1	1...11	1

Rys. 1: Tabelka przedstawiająca algorytm sumowania bloków, z których każdy jest złożony z takich samych cyfr.

$$\begin{aligned} a: &\quad 0\ 11\ 111\ 0\ 11\ 11\ 000\ 00\ 1\ 1111\ 00\ 000\ 1 \\ b: &\quad 0\ 00\ 111\ 1\ 11\ 00\ 000\ 11\ 0\ 1111\ 11\ 000\ 0 \\ a+b: &\quad 1\ 00\ 111\ 0\ 10\ 11\ 001\ 00\ 0\ 1110\ 11\ 000\ 1 \end{aligned}$$

Rys. 2: Obliczanie $a + b$ metodą blokową; $k(a) = 7$, $k(b) = 6$, $k(a + b) = 15$. Odstępy reprezentują miejsca podziału.

Jeśli umiemy dodawać zwarte reprezentacje, to odejmowanie możemy zaimplementować za pomocą dodawania. Załóżmy, że chcemy obliczyć $a - b$, przy czym wiemy, że $a - b > 0$ oraz $b > 0$. Niech $T = t(a) + 2$. Zaczynamy od obliczenia $c = a + 2^T - 1 - b$, co robimy w ten sposób, że do liczby a dodajemy liczbę $2^T - 1 - b$; jeśli $REP(b) = (x_1, \ldots, x_k)$, to $REP(2^T - 1 - b) = (T - t(b), x_1, \ldots, x_k)$. Żądanym wynikiem jest teraz $c + 1 - 2^T$. Dodanie 1 wykonujemy jak zwykłe dodawanie. Zauważmy dalej, że $c+1-2^T < a < 2^{T-1}$, więc odjęcie 2^T od $c+1$ wykonujemy poprzez usunięcie początkowej jedynki oraz wszystkich następujących po niej bezpośrednio zer (co najmniej jednego), co odpowiada usunięciu pierwszych dwóch liczb ze zwartej reprezentacji liczby $c + 1$. Kto nie wierzy, niech sprawdzi na przykładzie.

Rozwiązanie wzorcowe, wykorzystujące arytmetykę na zwartych reprezentacjach liczb, jest zaimplementowane w plikach `jed.c`, `jed1.cpp`, `jed2.pas`, `jed3.cpp` oraz, wyjątkowo krótko, w pliku `jed4.cpp`. Wykonuje ono $O(1)$ operacji o koszcie liniowym, więc jego złożoność czasowa wynosi $O(k)$.

Rozwiązania nieoptymalne

Najprostsze rozwiązanie zadania *Jedynki* polega na przejrzeniu wszystkich liczb od 1 do n i zliczeniu napotkanych SKS jedynek. Zostało ono zaimplementowane w pliku `jeds1.pas` i zdobywało na zawodach 20 punktów.

W pliku `jeds0.c` znajduje się implementacja rozwiązania korzystającego z takich samych wzorów co rozwiązanie wzorcowe, ale wykonującego operacje jedynie na zmiennych całkowitych 64-bitowych. Takie rozwiązania zdobywały na zawodach 40 punktów.

Rozwiązanie z pliku `jeds2.cpp` również wykorzystuje opisane wzory, jednakże do ich obliczania używa standardowej (a nie zwartej) arytmetyki dużych liczb. Za tego typu rozwiązania można było zdobyć na zawodach 60-70 punktów.

Testy

Rozwiązania zadania *Jedynki* były sprawdzane na 10 zestawach testowych, opisanych w poniższej tabelce. Parametr n oznacza liczbę reprezentowaną w teście, $k = k(n)$ to długość zwartej reprezentacji n, natomiast S to $t(n)+1$, czyli suma liczb x_i z wejścia. W opisach testów skróty NPJ i PJ oznaczają liczby n rozpoczynające się w zapisie binarnym odpowiednio od 11 i od 10, natomiast DW i MW oznaczają odpowiednio dużą lub małą wariancję liczb występujących w $REP(n)$.

Nazwa	k	n lub S	Opis
jed1a.in	5	$n < 10^6$	NPJ, DW — test losowy
jed1b.in	10	$S = 10$	PJ, MW — liczba $(1010\ldots)_2$
jed1c.in	1	$n = 1$	PJ, MW — przypadek brzegowy
jed2a.in	9	$n < 10^6$	NPJ, MW — test losowy
jed2b.in	2	$n = 2^{15}$	PJ, DW
jed2c.in	1	$n = 1$	PJ, MW — przypadek brzegowy
jed3a.in	10	$S = 55$	PJ, DW — test losowy, posortowany
jed3b.in	5	$S = 31$	NPJ — malejący ciąg potęg 2
jed4a.in	10	$S = 55$	NPJ, MW — test losowy
jed4b.in	3	$n = 2^{59} + 2^{58} - 1$	PJ, DW
jed5a.in	200	$S = 1\,000$	PJ, DW — test losowy, posortowany
jed5b.in	50	$S = 1\,000$	NPJ, MW — test losowy
jed6a.in	300	$S = 10\,000$	PJ, MW — test losowy
jed6b.in	100	$S = 10\,000$	NPJ, DW — test losowy
jed7a.in	500	$S = 10^5$	NPJ, DW — test losowy
jed7b.in	2	$n = 2^{100\,000}$	PJ, DW
jed8a.in	1 000	$S = 10^7$	PJ, MW — test losowy, posortowany
jed8b.in	10 000	$S = 10^7$	NPJ, DW — test losowy
jed9a.in	100 000	$S = 10^8$	PJ, MW — test losowy
jed9b.in	100 000	$S = 10^8$	NPJ, DW — test losowy
jed10a.in	1 000 000	$S = 10^9$	PJ, MW — test losowy
jed10b.in	1 000 000	$S = 10^9$	NPJ, DW — test losowy
jed10c.in	1	$n = 2^{(10^9)} - 1$	NPJ, MW — przypadek brzegowy