Krzysztof Sikora
Treść zadania, Opracowanie

Marcin Stefaniak
Program

Ciągi bez zająknięć

Rozważamy ciągi liter. Powiemy, że ciąg $x_1x_2\ldots x_n$ zawiera **zająknięcie**, *jeśli napotykamy w nim dwa, następujące bezpośrednio po sobie, wystąpienia takiego samego podciągu. Tzn. jeśli dla pewnych i i j ($1 \leqslant i < j \leqslant \frac{n+i+1}{2}$) zachodzi: $x_i = x_j$, $x_{i+1} = x_{j+1}$, $\ldots$, $x_{j-1} = x_{2j-i-1}$.*

Interesują nas n-elementowe ciągi bez zająknięć o minimalnej liczbie liter.

Przykłady

Dla $n = 3$ wystarczą dwie litery, powiedzmy a i b. Mamy dokładnie dwa 3-elementowe ciągi bez zająknięć złożone z takich liter: aba i bab. Dla $n = 5$ potrzebne są już 3 różne litery. Na przykład abcab jest trójliterowym ciągiem bez zająknięć. W ciągu babab mamy dwa zająknięcia: ba i ab.

Zadanie

Napisz program, który:

- *wczyta długość ciągu n,*
- *obliczy n-elementowy ciąg bez zająknięć o minimalnej liczbie różnych liter,*
- *wypisze wynik.*

Wejście

W pierwszym wierszu standardowego wejścia zapisana jest jedna dodatnia liczba całkowita n, $1 \leqslant n \leqslant 10\,000\,000$.

Wyjście

Twój program powinien pisać na standardowe wyjście. W pierwszym wierszu powinna zostać wypisana jedna dodatnia liczba całkowita k, równa minimalnej liczbie różnych liter, które muszą wystąpić w n-elementowym ciągu nie zawierającym zająknięć.

W drugim wierszu należy wypisać obliczony ciąg bez zająknięć, jako słowo złożone z n małych liter alfabetu angielskiego, od litery `a` *do k-tej litery alfabetu. Jeżeli istnieje wiele takich ciągów, Twój program powinien wypisać jeden z nich.*

Możesz przyjąć, że 26 liter wystarczy.

Przykład

Dla danych wejściowych:
```
5
```
poprawnym wynikiem jest:
```
3
abcab
```

Rozwiązanie

Analiza problemu

Słowa bezkwadratowe i silnie bezsześcienne

Rozpoczniemy od kilku definicji, które są niezbędne do zrozumienia rozwiązania wzorcowego.

Definicja 1 Słowo nazywamy *bezkwadratowym*, jeśli nie zawiera podsłowa postaci xx, gdzie x jest niepustym, skończonym słowem.

Łatwo zauważyć, że nad alfabetem dwuliterowym ($\{a,b\}$) nie istnieje słowo bezkwadratowe dłuższe niż 3, ponieważ dowolny ciąg długości 4 zawiera *aa* lub *bb* albo jest postaci *abab* lub *baba*. Z tego powodu dowolne bezkwadratowe słowo o długości większej niż 3 musi być słowem nad alfabetem co najmniej trzyliterowym. Norweski matematyk Axel Thue udowodnił, że istnieje nieskończenie długie słowo bezkwadratowe nad alfabetem trzyliterowym i pokazał, jak je znaleźć.

Definicja 2 Słowo nazywamy *silnie bezsześciennym*, jeśli nie zawiera podsłowa postaci xxa, gdzie x jest niepustym, skończonym słowem i a jest pierwszym symbolem x.

Oczywiście, jeśli słowo jest bezkwadratowe, to jest także silnie bezsześcienne.

Słowo Thuego-Morsa

Definicja 3 *Słowem Thuego-Morsa* nad alfabetem $\{a,b\}$ nazywamy ciąg $\alpha = (s_n)_{n\in\mathbb{N}}$ taki, że

$$s_n = \begin{cases} a & \text{jeśli } n \text{ ma parzystą liczbę jedynek w zapisie binarnym,} \\ b & \text{jeśli } n \text{ ma nieparzystą liczbę jedynek w zapisie binarnym.} \end{cases}$$

W konsekwencji powstaje nieskończone słowo nad alfabetem dwuelementowym

$$\alpha = abbabaabbaabab\ldots$$

Podamy teraz kilka własności słowa Thuego-Morsa. Dla prostoty dowodów załóżmy, że litery a i b są „przeciwne", czyli $a = -b$ i $b = -a$. Łatwo zauważyć, patrząc na binarną reprezentacje n, że:

Własność 4 $s_{2n} = s_n$

Własność 5 $s_{2n+1} = -s_n$

Z własności 4 i 5 mamy:

Własność 6 *Jeśli $s_j = s_{j+1}$, to j jest nieparzyste.*

Dowód Załóżmy, że $s_j = s_{j+1}$ i j jest parzyste. Wtedy $s_j = s_{j/2}$ i $s_{j+1} = -s_{j/2}$, czyli $s_j \neq s_{j+1}$. Sprzeczność. ■

Teraz możemy łatwo pokazać, że:

Własność 7 *Nie istnieje j takie, że $s_j = s_{j+1} = s_{j+2}$.*

Z własności 4 wynika, że

$$(s_{2n})_{n\in\mathbb{N}} = \alpha,$$

a z własności 5, że

$$(-s_{2n+1})_{n\in\mathbb{N}} = \alpha.$$

Oznaczmy ciąg składający się z liter na pozycjach parzystych przez P, a ciąg składający się z liter na pozycjach nieparzystych przez N.

Własność 8 *W 5 kolejnych literach ciągu α istnieją 2 kolejne litery, które są takie same.*

Dowód Gdyby tak nie było, mielibyśmy jedną z dwóch sytuacji: *ababa* lub *babab*. Wtedy podciąg złożony z pierwszej, trzeciej i piątej litery byłby postaci *aaa* lub *bbb* i należał albo do P, albo do N, w zależności od parzystości położenie ciągu w słowie Thuego-Morsa. Teraz P lub N zawierałby 3 kolejne takie same litery, co jest niemożliwe. ■

Twierdzenie 9 *Słowo Thuego-Morsa jest silnie bezsześcienne, tzn. nie istnieją $i \geqslant 0$, $j > 0$ takie, że $s_{i+k} = s_{i+j+k}$ dla każdego k, $0 \leqslant k \leqslant j$.*

Dowód Załóżmy, że takie wartości i i j istnieją, wtedy:

1. j nie może być równe 1 (z własności 7).
2. j nie może być nieparzyste i większe niż 1, gdyż wtedy jest co najmniej 7 liter między i i $i+2j$ (włącznie) i z własności 8 wynika, że istnieją dwie kolejne takie same litery. Z faktu, iż litery od i to $i+j$ są takie same jak od $i+j$ do $i+2j$, mamy, że istnieją dwie różne takie pary, przesunięte o j. Ponieważ j jest nieparzyste, jedna z tych par zaczyna się na parzystej pozycji, co nie jest możliwe (z własności 6).
3. Zatem j musi być parzyste. Załóżmy, że jest minimalne. Teraz albo P, albo N spełnia równania z twierdzenia dla $j/2$ w miejsce j. Ponieważ j jest parzyste, mamy sprzeczność, bo j było minimalne. Nie może być także nieparzyste, co zostało udowodnione wcześniej. ■

Lemat 10 *Istnieje słowo bezkwadratowe β nad alfabetem czteroliterowym.*

Dowód Skonstruujmy alfabet mocy 4 o elementach będących parami symboli z alfabetu zdefiniowanego wcześniej:

$$\Sigma = \{[aa], [ab], [ba], [bb]\}.$$

Zdefiniujmy teraz ciąg $\beta = (d_n)_{n\in\mathbb{N}}$ taki, że

$$d_n = [s_n s_{n+1}] \text{ dla każdego } n \geqslant 0,$$

gdzie s_n to litery ciągu Thuego-Morsa. Ciąg β jest bezkwadratowy. Gdyby tak nie było, istniałoby podsłowo yy, w którym

$$y = d_j \ldots d_{j+t-1} = d_{j+t} \ldots d_{j+2t-1}$$

dla pewnego $t \geqslant 1$. Wtedy mielibyśmy

$$[s_j s_{j+1}] \ldots [s_{j+t-1} s_{j+t}] = [s_{j+t} s_{j+t+1}] \ldots [s_{j+2t-1} s_{j+2t}],$$

co daje $s_{j+i} = s_{j+t+i}$ dla $0 \leqslant i \leqslant t$. Wtedy słowo Thuego-Morsa zawiera

$$(s_j \ldots s_{j+t-1})^2 s_j,$$

co jest sprzeczne z twierdzeniem 9. ■

Teraz pokażemy, jak przekształcić ciąg β w bezkwadratowy ciąg γ nad trzyliterowym alfabetem. Konstrukcja opiera się na następującej obserwacji: jeśli oznaczymy litery alfabetu Σ jako

$$[aa] = 1,\ [ab] = 2,\ [ba] = 3,\ [bb] = 4,$$

to łatwo można zobaczyć, że w β symbol 1 musi być przed 1 lub 2. W rzeczywistości 1 nie może występować po 1, gdyż β jest bezkwadratowe. Więc 1 musi być przed 2. Przeprowadzając podobne rozumowanie dla pozostałych symboli otrzymujemy:

Lemat 11 *W słowie* β*:*

- 1 *występuje po* 3 *i przed* 2.
- 4 *występuje po* 2 *i przed* 3.

Teraz, gdy zastąpimy wszystkie wystąpienia 1 i 4 w ciągu β przez 5, otrzymamy ciąg γ.

Twierdzenie 12 *Słowo* γ *jest bezkwadratowe.*

Dowód Gdyby tak nie było, istniałoby podsłowo yy.

Gdyby długość y była równa 1, yy byłoby postaci 55 (22 i 33 nie mogą się pojawić). To pociągałoby za sobą istnienie 11, 14, 41 lub 44 w ciągu β, co byłoby sprzeczne z lematem 11.

Gdyby długość y wynosiła co najmniej 2, wówczas każde wystąpienie 5 w y znajdowałoby się pomiędzy dwoma symbolami. Wtedy moglibyśmy dokonać jednoznacznej rekonstrukcji słowa (nazwijmy je x), z którego powstał y. Wówczas β zawierałoby xx, co jest sprzeczne z lematem 10. ■

Algorytm

Rozwiązanie wzorcowe rozpatruje 4 przypadki:

- dla $n = 1$ wielkość alfabetu wynosi 1, a poszukiwanym słowem jest *a*,
- dla $n = 2$ wielkość alfabetu wynosi 2, a poszukiwanym słowem jest *ab*,
- dla $n = 3$ wielkość alfabetu wynosi 2, a poszukiwanym słowem jest *aba*,
- dla $n \geqslant 4$ wielkość alfabetu wynosi 3, a poszukiwane słowo otrzymywane jest przez generowanie kolejnych liter ciągu Thuego-Morsa i przekształcanie ich zgodnie z regułami podanymi wcześniej.

Algorytm działa w czasie $O(n)$ i wykorzystuje pamięć o stałym rozmiarze.

Testy

Zadanie sprawdzane było na zestawie 20 danych testowych, dla n równych 1, 2, 3, 4, 100, 300, 1000, 3000, 10000, 30000, 100000, 200000, 400000, 800000, 1000000, 2000000, 4000000, 6000000, 8000000 i 10000000.