

Krzysztof Dulęba

**Zadanie 1:**

Skonstruować nieskończony ciąg liter nad alfabetem  $\{a, b, c\}$  nie zawierający żadnego kwadratu, tj. podśłowa postaci  $ww$ ,  $w \neq \epsilon$ .

**Rozwiązanie:**

Skonstruuję ciąg słów  $x_n$  taki, że słowo  $x_n$  będzie prefiksem  $x_{n+1}$  oraz każde  $x_n$  nie będzie zawierało żadnego kwadratu. Wtedy będzie istnieć  $x = \lim_{n \rightarrow \infty} x_n$ , które również nie będzie zawierać żadnego kwadratu. Istotnie, gdyby  $x$  zawierało podśłowo postaci  $ww$ ,  $w \neq \epsilon$ , to pewien dostatecznie długi prefiks  $x_n$  również zawierałby podśłowo  $ww$ .

Niech  $x_1 = a$ . Do otrzymania słowa  $x_{n+1}$  zastosujemy podstawienie

$$a \rightarrow abc$$

$$b \rightarrow ac$$

$$c \rightarrow b$$

Powiedzmy w skrócie, że  $x_n = f(x_{n-1})$ . Co istotne, tak jak każde podstawienie,  $f$  jest homomorfizmem słów, tzn.  $f(vw) = f(v)f(w)$ . Pozostaje udowodnić postulowane własności tego ciągu.

1.  $x_n$  jest prefiksem  $x_{n+1}$ .

Zauważmy, że  $x_2 = abc$ , więc  $x_1$  jest prefiksem  $x_2$ . Załóżmy, że  $x_{n-1}$  jest prefiksem  $x_n$ . Wtedy  $x_n = x_{n-1}w$ , czyli

$$x_{n+1} = f(x_n) = f(x_{n-1}w) = f(x_{n-1})f(w) = x_n f(w)$$

Czyli faktycznie  $x_n$  będzie prefiksem  $x_{n+1}$ .

2.  $x_n$  nie zawiera podśłowa postaci  $ww$ ,  $w \neq \epsilon$ .

Dla  $n = 1$  twierdzenie zachodzi. Powiedzmy, że zachodzi dla wszystkich  $i < n$ , ale dla pewnego  $n > 1$  już nie. Wtedy  $x_n = vw w z$ .

Dla słowa  $q$  powiemy, że  $f^{-1}(q) = y$  jeśli istnieje  $y$  takie, że  $f(y) = q$ . Na przykład, gdy  $x_n = vw$ , to jeśli  $v$  ma odwrotność, to również  $w$  ją ma i na odwrót. Jest tak dlatego, że  $f(f^{-1}(v))w = vw = x_n = f(x_{n-1})$ , a zatem  $w = f(x_{n-1}) - f(f^{-1}(v)) = f((x_{n-1}) - f^{-1}(v))$ , gdzie  $-$  oznacza obcięcie prefiksu.

Zastanówmy się teraz, jakiej postaci musi być słowo  $w$ . Rozpatrzmy następujące przypadki:

- $w = aw'$ .

Wtedy łatwo widzieć, że słowo  $w$  ma odwrotność  $f^{-1}(w)$ . Jest tak dlatego, że oczywiście każdy prefiks, po którym w  $x_n$  występuje znak  $a$  ma odwrotność, a zatem odwrotność mają  $v$  i  $vw$ , czyli również samo  $w$ . Jeśli tak, to  $x_{n-1} = f^{-1}(x_n) = f^{-1}(v)f^{-1}(w)f^{-1}(w)f^{-1}(z)$ , czyli również  $x_{n-1}$  zawiera kwadrat — sprzeczność.

- $w = bw'$ .

Znak  $b$  może się pojawić w  $x_n$  w dwóch kontekstach: jako obraz  $c$  lub jako fragment obrazu  $a$ . Ten pierwszy przypadek oznacza, że  $v$  ma odwrotność, a zatem kończy się na  $b$  lub  $c$ . Teraz jeśli  $w$  kończy się na  $b$  lub  $c$ , to ma odwrotność, więc musi kończyć się na  $a$  (jeśli  $w$  ma odwrotność, to podobnie jak w poprzednim przypadku okazuje się, że  $x_{n-1}$  zawiera kwadrat). Wtedy mamy podciąg  $ab$  ( $a$  pochodzi z końcówki pierwszego  $w$ ,  $b$  z początku drugiego), a zatem dalej musi stać  $c$ . Czyli  $c$  jest drugim znakiem w  $w$ , ale to oznacza, że mamy podciąg  $bc$  ( $b$  to pierwszy znak pierwszego  $w$ , a  $c$  to drugi znak). Czyli wcześniej musi stać  $a$ , a przecież założyliśmy, że stoi tam  $b$  lub  $c$  — sprzeczność.

Jeśli  $b$  na początku  $w$  pochodzi z obrazu  $a$ , to dalej stoi  $c$ . W szczególności oznacza to, że  $v$  i  $w$  muszą kończyć się na  $a$ . Czyli  $v = v'a$ ,  $w = bcw''a$ , zaś  $x_n = vwwz = v'abcw''abcw''az = v'(abcw'')(abcw'')az$ . Zatem  $x_n$  zawiera również inny kwadrat, zaczynający się na  $a$ , co już rozważyliśmy w pierwszym przypadku.

- $w = cw'$ .

Jeśli drugie  $w$  zaczyna się na  $c$ , to pierwsze kończy się na  $a$  lub  $ab$ . W pierwszym przypadku mamy sytuację  $w = cw'a$ , czyli  $x_n = vwwz = vcw'acw'az$ . Jeśli  $v$  kończy się na  $a$  lub  $z$  zaczyna się na  $c$ , to mamy kwadraty rozważanych już postaci. Inaczej  $v$  kończy się na  $ab$ , zaś  $z$  zaczyna się na  $bc$ , czyli  $x_n = v'abcw'acw'abcz'$ . Zatem po wzięciu  $f^{-1}$  od całości tego wyrażenia dostajemy

$$x_{n-1} = f^{-1}(v')af^{-1}(w')bf^{-1}(w')af^{-1}(z')$$

Wtedy jednak  $x_{n-1}$  nie będzie miał odwrotności, gdyż jeśli  $f^{-1}(w')$  zaczyna się na  $a$ , to mamy dwie litery  $a$  pod rząd; jeśli zaczyna się na  $b$ , bo mamy dwie litery  $b$  pod rząd, a jeśli zaczyna się na  $c$ , to musi kończyć się na  $a$ , a wtedy znów mamy dwie litery  $a$  pod rząd, co nie jest możliwe — sprzeczność.

Zatem  $w$  kończy się na  $ab$ . Po  $ab$  musi występować  $c$ , zatem  $z = cz'$ , czyli  $x_n = vcw'abcw'abcz'$ , zaś  $x_{n-1} = f^{-1}(vc)f^{-1}(w')af^{-1}(w')af^{-1}(z')$  i również  $x_{n-1}$  zawiera kwadrat.

W każdym z rozpatrzonych przypadków okazało się, że  $x_n$  nie może zawierać kwadratu, zatem dowód jest zakończony.

### **Zadanie 2**

Zaprojektować algorytm, który dla danego automatu deterministycznego o  $n$  stanach, działającego nad alfabetem  $\Sigma$ , znajduje równoważny automat minimalny w czasie  $O(|\Sigma| \log n)$ .

### **Rozwiązanie:**

Znalazłem bardzo dobrą pozycję na ten temat i zamiast po prostu z niej przepisywać, podam tytuł: Describing an Algorithm by Hopcroft, David Gries, Acta Informatica vol 2. strony 97–109 (1973).

### **Zadanie 3**

Język  $L$  nazwiemy *definitywnym* jeśli istnieje takie  $k$ , że przynależność dowolnego słowa do języka  $L$  zależy tylko od jego  $k$  pierwszych liter. Oczywiście każdy język definitywny jest regularny, ale nie na odwrót.

Zaprojektować algorytm, który dla dowolnego deterministycznego automatu  $A$  rozstrzyga, czy  $L(A)$  jest definitywny w czasie wielomianowym względem rozmiaru  $A$ .

### **Rozwiązanie:**

Język  $L$  nie jest definitywny wtedy i tylko wtedy, gdy dla każdego  $n$  istnieje słowo  $a \in L$  długości  $|a| \geq n$ , które zostanie zaakceptowane przez automat  $A$  nie wcześniej, niż po przeczytaniu  $n$  pierwszych znaków.

Dla automatu skończonego to zachodzi tylko wtedy, gdy w grafie reprezentującym  $A$  istnieje cykl zawierający wierzchołek pewnej ścieżki od pewnego stanu początkowego do pewnego stanu końcowego. Ponadto z tego cyklu musi być osiągalny jakiś stan nieakceptujący. Wtedy bowiem można dowolnie dużo razy przejść po tym cyklu, a słowo nadal może nie być zaakceptowane, po czym opuścić go i dość do stanu akceptującego.

Będziemy korzystać z następujących procedur, których kod jest zapisany w pseudojęzyku poniżej.

```
function koniec(x)
  {oznacz x jako odwiedzony}
  foreach y (nieodwiedzony wierzchołek osiągalny z x)
    if (x to wierzchołek nieakceptujący) then return true;
    if (koniec(y)) then return true;
  return false;
endfunction
```

```

function cykl(x, root)
  {oznacz x jako odwiedzony}
  foreach y (nieodwiedzony wierzchołek osiągalny z x)
    if (x == root) then return true;
    if (koniektab[x] == true) return false;
    if (cykl(y, root)) then return true;
  return false;
endfunction

```

```

function main()
  foreach x (wierzchołek)
    koniektab[x] = koniec(x);
  foreach x (wierzchołek)
    if (cykl(x, x)) return "niedefinitywny";
  return "definitywny";
endfunction

```

Ich znaczenie jest następujące: koniec sprawdza, czy po wejściu do danego wierzchołka można jeszcze osiągnąć stan nieakceptujący (wtedy zwraca false). Jeśli jednak po osiągnięciu tego wierzchołka słowo już na pewno zostanie zaakceptowane, zwracana jest wartość true.

cykl sprawdza, czy istnieje ścieżka od wierzchołka x do wierzchołka root. W szczególności wywołany z funkcji main szuka cyklu.

#### Zadanie 4

Rozważamy wyrażenia Boole'owskie zbudowane ze stałych 0, 1, nawiasów (,) i operacji  $\wedge$  i  $\vee$ , z oczywistą semantyką. Czy istnieje automat skończony nad alfabetem  $\{0, 1, (, ), \wedge, \vee\}$ , który po przeczytaniu takiego wyrażenia Boole'owskiego przyjmuje stan akceptujący wtedy i tylko wtedy, gdy wyrażenie ma wartość 1?

#### Rozwiązanie:

Taki automat nie istnieje. Przyjmijmy jednak, że istnieje, i że ma  $n$  stanów. Dla uproszczenia rozumowania zakładam, że automat ten jest deterministyczny (co nie wpływa jednak na ogólność rozumowania). Rozważmy następujące napisy:

$$v_{k,l} = ({}^l1 \vee ({}^k0 \vee 0$$

dla  $k = 2n, \dots, 5n$  oraz  $l = 7n - k$ . Ponieważ jest tylko  $n$  stanów, to po przeczytaniu pewnych trzech różnych napisów  $v_{k,7n-k}$ ,  $v_{k',7n-k'}$ ,  $v_{k'',7n-k''}$  automat będzie w tym samym stanie. W szczególności któraś z par  $(k, k')$ ,  $(k, k'')$  lub  $(k', k'')$

będzie miała różnicę większą co do modułu od 1. Przyjmijmy, że jest to para  $(k, k')$ . Niech  $m = \lfloor \frac{k+k'}{2} \rfloor$ . Do obu napisów  $v_{k,7n-k}$  i  $v_{k',7n-k'}$  dołączmy napis  $w_m = )^m \wedge 0)^{7n-m}$ . Wtedy w obu przypadkach automat nadal będzie w tym samym stanie. Ale przecież jedno z wyrażeń  $v_{k,7n-k}w_m$ ,  $v_{k',7n-k'}w_m$  ma wartość 1, a drugie 0, oraz oba napisy to poprawne wyrażenia Boole'owskie — sprzeczność.

### Zadanie 5

Mówimy, że słowo  $w$  synchronizuje stany automatu deterministycznego, jeśli istnieje taki stan  $q_0$ , że startując z dowolnego stanu i czytając słowo  $w$ , automat dojdzie zawsze do stanu  $q_0$ , tzn  $(\forall q \in Q) q \xrightarrow{w} q_0$ .

Znaleźć najkrótsze słowo synchronizujące dla automatu nad alfabetem  $\{a, b\}$  o zbiorze stanów  $\{0, 1, \dots, k-1\}$  i funkcji przejścia

$$\begin{aligned} i &\xrightarrow{a} i+1 \pmod k && \text{dla } i = 0, 1, \dots, k-1 \\ i &\xrightarrow{b} i && \text{dla } i = 0, 1, \dots, k-2 \\ k-1 &\xrightarrow{b} 0 \end{aligned}$$

### Rozwiązanie

Zadanie synchronizacji możemy przeformułować w następującym języku: niech  $A$  będzie automatem skończonym. Wówczas tworzymy automat  $A'$ , którego stanami są wszystkie podzbiory stanów  $A$ , zaś funkcja przejścia określona jest  $Q \xrightarrow{x} \bigcup_{q \in Q} \{y : q \xrightarrow{x} y\}$ . Stanem początkowym w  $A'$  jest zbiór wszystkich stanów  $A$ , zaś stanami akceptującymi są wszystkie zbiory pojedynczych stanów  $A$ . Jest jasne, że słowo  $w$  synchronizuje  $A$  wtedy i tylko wtedy, gdy jest akceptowane przez  $A'$ . Chcemy więc znaleźć najkrótsze słowo akceptowane przez  $A'$ .

W naszym przypadku łatwo widzieć, że ostatnią literą najkrótszego słowa  $w$  musi być  $b$ , gdyż  $a$  nie synchronizuje żadnych dwóch stanów, a jedynie cyklicznie je zamienia ze sobą. Jeśli więc na końcu  $b$  stoi  $b$ , to stanem  $q_0$  będzie stan 0.

Skoro wiemy już, o który stan końcowy w  $A'$  nam chodzi, to jeszcze raz przeformułujmy zadanie. Teraz niech  $A''$  będzie automatem o stanach jak  $A'$ , z odwróconą relacją przejścia, stanem początkowym  $\{q_0\}$  i stanem akceptującym będącym zbiorem wszystkich stanów  $A$  (czyli stanem początkowym  $A'$ ). Interesuje nas teraz najkrótsze  $v$  słowo akceptowane w  $A''$ . To słowo  $v$  jest lustrzanym odbiciem szukanego słowa  $w$ .

Tutaj mamy następujące przejścia:  $Q \xrightarrow{b} Q \cup \{k-1\}$  jeśli  $0 \in Q$ ,  $Q \xrightarrow{b} Q$  jeśli  $0 \notin Q$ ,  $Q \xrightarrow{a} \bigcup_{q \in Q} \{q-1\}$ .

Z poprzednich rozważań wiemy już, że pierwszą literą  $v$  jest  $b$ . Znajdujemy się wtedy w stanie  $Q_1 = \{0, k-1\}$ . Ponieważ  $b$  nie będzie już zmieniało tego stanu, dalej musi stać ciąg  $a^n$  dla pewnego  $n$ . Po tym  $a^n$  będzie z kolei  $b$ . Jeśli to  $b$  ma w ogóle zmienić stan (a powinno, gdyż inaczej można je wyciąć otrzymując krótsze słowo), to muszą być spełnione dwa warunki:  $0 \in Q_1 a^n$  oraz  $k-1 \notin Q_1 a^n$ . Ale

tak będzie tylko dla  $n = ik - 1$ . Zatem  $n = k - 1$ , gdyż takie  $n$  jest najkrótsze. Potem stoi  $b$  i znajdujemy się teraz w stanie  $Q_2 = \{1, 0, k - 1\}$ .

Powtarzając to rozumowanie stwierdzamy, że jeśli dalej stoi  $a^n$ , a po nim  $b$ , to musi być tak, że  $0 \in Q_2a^n$  oraz  $k - 1 \notin Q_2a^n$ , co znów zachodzi tylko dla  $n = ik + 1$ . Kontynuując w ten sposób dojdziemy do stanu końcowego, a  $v$  będzie miało postać  $v = ba^{k-1}ba^{k-1} \dots ba^{k-1}b$ , przy czym  $b$  występuje w tym ciągu  $k - 1$  razy.

Tak utworzone  $v$  jest najkrótsze, gdyż w powyższej konstrukcji wykonywane były tylko i wyłącznie te operacje, które były konieczne do dojścia do stanu końcowego.  $v$  jest palindromem, więc szukane  $w = v = ba^{k-1}ba^{k-1} \dots ba^{k-1}b$ .