Wojciech Rytter
autor zadania, opracowanie

Piotr Sankowski
program wzorcowy

Wirusy

Komisja Badania Wirusów Binarnych wykryła, że pewne ciągi zer i jedynek są kodami wirusów. Komisja wyodrębniła zbiór wszystkich kodów wirusów. Ciąg zer i jedynek nazywamy **bezpiecznym**, *gdy żaden jego segment (tj. ciąg kolejnych wyrazów) nie jest kodem wirusa. Komisja dąży do ustalenia, czy istnieje nieskończony, bezpieczny ciąg zer i jedynek.*

Przykład

Dla zbioru kodów $\{011, 11, 00000\}$ *nieskończonym, bezpiecznym ciągiem jest* $010101\ldots$. *Dla zbioru kodów* $\{01, 11, 00000\}$ *nie istnieje nieskończony, bezpieczny ciąg zer i jedynek.*

Zadanie

Napisz program, który:

- *wczyta kody wirusów z pliku tekstowego* `WIR.IN`,
- *stwierdzi, czy istnieje nieskończony, bezpieczny ciąg zer i jedynek,*
- *zapisze wynik do pliku tekstowego* `WIR.OUT`.

Wejście

W pierwszym wierszu pliku wejściowego `WIR.IN` *znajduje się jedna liczba całkowita* n *będąca liczbą wszystkich kodów wirusów.*

W każdym z kolejnych n *wierszy znajduje się jedno niepuste słowo złożone ze znaków* `0` *i* `1` *— kod wirusa. Sumaryczna długość wszystkich słów nie przekracza 30000.*

Wyjście

W pierwszym i jedynym wierszu pliku wyjściowego `WIR.OUT` *powinno znajdować się słowo:*

- `TAK` *— jeżeli istnieje nieskończony, bezpieczny ciąg zer i jedynek,*
- `NIE` *— w przeciwnym przypadku.*

Przykład

Dla pliku wejściowego `WIR.IN`*:*

```
3
01
11
00000
```

poprawną odpowiedzią jest plik tekstowy `WIR.OUT`*:*
`NIE`

ROZWIĄZANIE

Algorytm teorio-grafowy

W rozwiązaniu zadania najważniejszą rolę odgrywają sufiksy i prefiksy wirusów. Jeśli słowo α można zapisać jako złożenie trzech słów $\alpha = u \cdot v \cdot w$, to u nazywamy prefiksem, a w sufiksem α. Na przykład prefiksami słowa *rytter* są $\{\epsilon, r, ry, ryt, rytt, rytte, rytter\}$, a sufiksami $\{\epsilon, r, er, ter, tter, ytter, rytter\}$, gdzie ϵ oznacza słowo puste o długości zero.

Dla słowa α i zbioru słów-wirusów W, przez $max_pref(\alpha)$ oznaczmy najdłuższy sufiks α, który jest jednocześnie prefiksem pewnego wirusa z W. Jeśli chcemy sprawdzić, czy dany ciąg binarny nie zawiera wirusa, to wystarczy przeglądać ten ciąg znak po znaku i pamiętać jedynie ostatni **istotny** fragment. Tym istotnym fragmentem jest maksymalny prefiks wirusa, zatem po wczytaniu ciągu $a_1a_2a_3 \ldots a_k$ pamiętamy $max_pref(a_1a_2a_3 \ldots a_k)$. Słowo $a_1a_2a_3 \ldots a_n$ jest *bezpieczne*, gdy każdy z pośrednich istotnych fragmentów jest bezpieczny, tzn. gdy
$max_pref(a_1)$, $max_pref(a_1a_2)$, $max_pref(a_1a_2a_3)$, $\ldots$, $max_pref(a_1a_2a_3 \ldots a_n)$
są bezpieczne (nie są wirusami).
Motywuje to wprowadzenie pomocniczego grafu G, którego ścieżki odpowiadają dokładnie wszystkim możliwym bezpiecznym słowom i którego węzłami są wszystkie bezpieczne prefiksy wirusów. Od węzła x prowadzimy skierowaną krawędź do y, gdy dla pewnej litery a (gdzie $a = 0$ lub $a = 1$) po dopisaniu jej do x otrzymamy słowo takie, że maksymalnym jego sufiksem, który jest prefiksem pewnego wirusa, jest y. Oznaczmy y przez $(\sigma(x, a))$. Można to zapisać bardziej formalnie:

- $\sigma(x, a) = max_pref(xa)$, gdy xa jest bezpieczny,
- $\sigma(x, a) =$ **NIL**, gdy xa nie jest bezpieczny (**NIL** oznacza brak krawędzi).

Rysunek 2 przedstawia graf G dla przykładowego zbioru wirusów

$$W = \{000,\ 0110,\ 100101, 1010, 111\}.$$

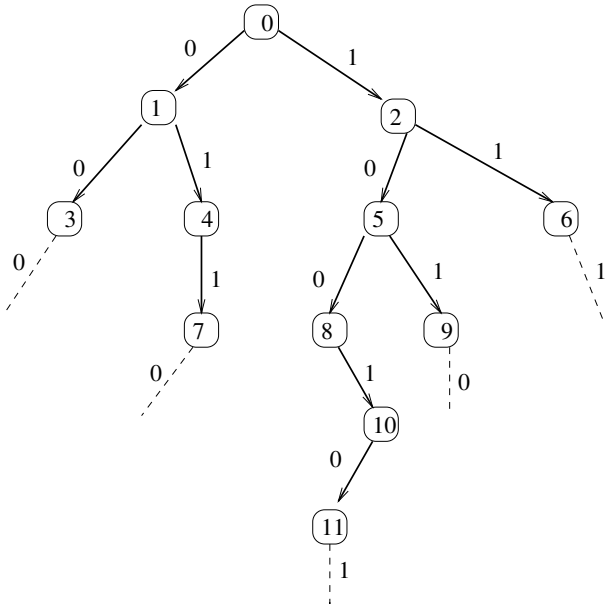
Grafy tego typu odpowiadają w *teorii obliczeń* tzw. *automatom skończonym*. Zbiór W jest bezpieczny wtedy i tylko wtedy, gdy w grafie G istnieje cykl skierowany osiągalny z węzła startowego odpowiadającego słowu pustemu. Każda ścieżka w tym grafie zaczynająca się od węzła startowego wyznacza ciąg bezpieczny.

Konstrukcja drzewa bezpiecznych prefiksów wirusów

Pierwszym krokiem w konstrukcji grafu G jest zbudowanie drzewa właściwych prefiksów wirusów, tzn. takich prefiksów słów ze zbioru W, które nie są równe żadnemu wirusowi. Drzewo dla przykładowego zbioru W jest przedstawione na rysunku 1.

Uwaga: Załóżmy początkowo, że żaden wzorzec nie jest właściwym podsłowem innego wzorca.

Rys. 1 Początkowa "aproksymacja" grafu G: drzewo T bezpiecznych prefiksów wirusów. Linie przerywane oznaczają przejścia zabronione. Brakujące niezabronione przejścia są uzupełnione w grafie G.



Wierzchołki są ponumerowane, jednakże każdy węzeł utożsamiamy z pewnym prefiksem wzorca. Załóżmy, że $syn(x, a)$ oznacza węzeł xa, jeśli xa jest bezpiecznym prefiksem wzorca. Jeśli $xa \in W$, to $syn(x, a) = N$, gdzie N jest specjalną wartością oznaczającą, że krawędzi etykietownej symbolem a z węzła x nigdy być nie może. W pozostałych przypadkach $syn(x, a) =$ **NIL**.

Właściwa konstrukcja grafu G

Algorytm korzysta istotnie z drzewa prefiksów T zbudowanego wcześniej i opisanego funkcją syn.

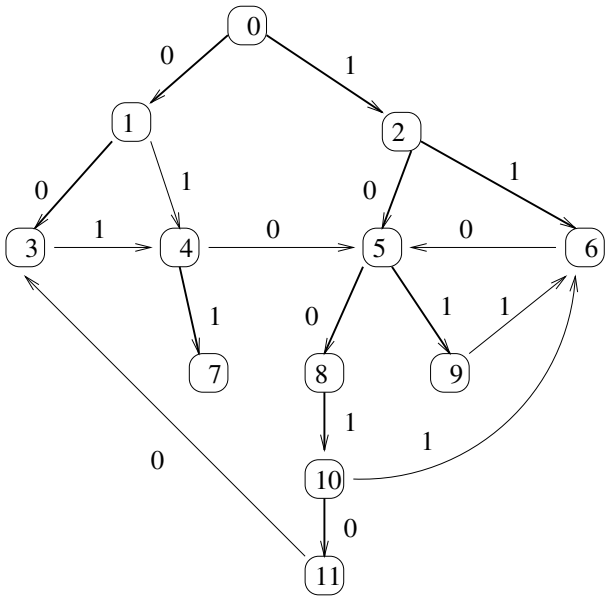
Algorytm
Inicjujemy $\sigma(\epsilon, a) = \epsilon$, $\sigma(\alpha, a) = NIL$, dla a=0,1 i $\alpha \neq \epsilon$ (słowa α odpowiadają węzłom w drzewie).
Przechodząc drzewo T wszerz, dla każdego węzła $x \neq root$ wykonaj:

```
1: niech y oraz a będą takie, że  x = syn(y,a);
2: if σ(y,a) ≠ NIL then begin
3:    temp := σ(y,a); σ(y,a) := x;
4:    for b:=0 to 1 do begin
```

```
5:        if syn(x,b)=N ∨ syn(temp,b)=N then
6:            σ(x,b):=NIL
7:        else if syn(temp,b) ≠ NIL then
8:            σ(x,b):=syn(temp,b)
9:        else σ(x,b):=σ(temp,b)
10:    end
11: end
```

Na zakończenie algorytmu należy ograniczyć graf G do jego podgrafu indukowanego przez wierzchołki osiągalne z wierzchołka startowego.
Uwaga: Założenie, że żaden wirus nie jest podsłowem innego wirusa było użyteczne przy wprowadzeniu pomocniczego drzewa prefiksów. Algorytm konstrukcji grafu automatycznie wykrywa takie sytuacje, tak więc założenie to nie jest istotne.

Rys. 2 Graf G dla przykładowego zbioru wirusów. Pogrubione krawędzie (przejścia) są pozostałością z drzewa prefiksów T, pozostałe krawędzie powstały w wyniku działania algorytmu konstrukcji G.



Sprawdzanie acykliczności

Aby stwierdzić, czy w grafie jest cykl należy sprawdzić, czy wierzchołki grafu można posortować topologicznie. Można zastosować dwie alternatywne metody:

- sortowanie topologiczne przy pomocy DFS,
- sortowanie topologiczne poprzez usuwanie wierzchołków ze stopniem wejściowym równym zero.

Nie opisujemy dokładnie tych metod ponieważ sprawdzanie acykliczności grafu wystąpiło już w wielu poprzednich zadaniach olimpijskich.
Dla naszego przykładowego zbioru wirusów nieskończona ścieżka "idąca" po węzłach grafu G.

$$0,\ 2,\ 5,\ (8,\ 10,\ 11)^{\infty}$$

odpowiada nieskończonemu bezpiecznemu słowu:

$$100(100)^{\infty}$$

Algorytm alternatywny

Wprowadzamy dwie operacje na zbiorze W. Niech $x, y \in W$.

Usuwanie y. Jeśli x jest podsłowem y, to y usuwamy z W.

Skracanie y. Jeśli $y = za$, gdzie $a = 0 \ \vee \ a = 1$, oraz $x = u\bar{a}$ i u jest sufiksem z, to $y{:=}z$.

Symbol $\bar{a}$ oznacza $1 - a$.
Na przykład jeśli $x = 0010$, $y = 001010011$, to skracając y dostajemy $y = 00101001$.
Operację skracania możemy zastosować dlatego, że jeśli w pewnym *nieskończonym* ciągu wystąpi skrócone y, to albo w tym ciągu wystąpi samo y, albo x.
Niech W' będzie zbiorem powstałym z W za pomocą jednej z tych operacji, wtedy:

$$W \text{ jest bezpieczny} \Leftrightarrow W' \text{ jest bezpieczny.}$$

Łatwo zauważyć, że W jest niebezpieczny wtedy i tylko wtedy, gdy po pewnej liczbie operacji otrzymamy zbiór zawierający oba pojedyncze symbole jako słowa. Natomiast, jeśli w wyniku dowolnego ciągu operacji otrzymamy zbiór zawierający słowo o długości większej niż jeden, to W jest bezpieczny.

Można wówczas skonstruować dowolnie długi bezpieczny ciąg postępując w następujący sposób: zaczynamy od ciągu pustego i dokładamy po jednym symbolu, do chwili gdy otrzymujemy ciąg niebezpieczny (gdy sufiks należy do w); zamieniamy ostatnio dodany symbol na przeciwny i kontynuujemy naszą konstrukcję.

Nasz algorytm polega na stosowaniu w dowolnej kolejności jednej z powyższych operacji, aż otrzymany zbiór zawierający obie litery $0, 1$, albo zbiór zawierający dłuższe słowo, który to zbiór nie da się dalej zmienić za pomocą jednej z naszych operacji. W pierwszym przypadku W jest niebezpieczny. Dla naszego przykładowego zbioru wirusów W z rysunku 1 od razu widać, że jest on bezpieczny, ponieważ żadna z dwóch operacji się do niego nie stosuje.

Przykład

Rozważmy przykład zbioru, który nie jest bezpieczny. W poniższym ciągu przekształceń podkreślamy słowo biorące udział w przekształceniu, słowo to jest usuwane lub skracane o jeden symbol od końca.

$$\{000, \underline{010}, 11\} \to \{\underline{000}, 01, 11\} \to \{00, \underline{01}, 11\} \to \{\underline{00}, 0, 11\} \to \{0, \underline{11}\} \to \{0, 1\}.$$

Zatem $\{000, 010, 11\}$ jest niebezpieczny, gdyż końcowy zbiór $\{0, 1\}$ jest niebezpieczny.

TESTY

Do sprawdzania rozwiązań zawodników użyto 10 testów:

- WIR1-3.IN — małe testy poprawnościowe,
- WIR4.IN — średni test poprawnościowy,
- WIR5.IN — test wydajnościowy sprawdzający metodę wyszukiwania cyklu, tylko bardzo długie cykle,
- WIR6.IN — mały test wydajnościowy, jeden cykl w automacie,
- WIR7.IN — duży test wydajnościowy, duża liczba możliwych nieskończonych słów,
- WIR8.IN — przypadek pesymistyczny dla iloczynu automatów, dużo krótkich słów,
- WIR9.IN — duży test wydajnościowy, mała liczba możliwych nieskończonych słów,
- WIR10.IN — średni test wydajnościowy, tylko jeden cykl w automacie.