Wojciech Rytter
treść zadania, opracowanie

Grzegorz Jakacki
program wzorcowy

Liczba wyborów symetrycznych

Dane są dwa ciągi słów (x_i) oraz (y_i), gdzie: $1 \leqslant i \leqslant n$ oraz $1 \leqslant n \leqslant 30$. Kolejno dla każdego i od 1 do n wybieramy jedno z dwóch słów: x_i albo y_i i wybrane słowa składamy (kolejne wybrane słowo dopisujemy do poprzednich z prawej strony).
Wybór *składa się z n decyzji, czy w kolejnym kroku wybrać odpowiedni i-ty wyraz z pierwszego, czy z drugiego ciągu słów. Bardziej formalnie: wybór jest n wyrazowym ciąggiem liczb 1 lub 2. Różne wybory mogą dać w wyniku to samo słowo.*
Wybór nazywamy **symetrycznym**, *gdy jego wynikiem jest palindrom — tzn. takie słowo, które nie zmienia się, gdy je czytamy od strony prawej do lewej.*

Zadanie

Napisz program, który:

- *wczytuje z pliku tekstowego LIC.IN liczbę n i dwa n-wyrazowe ciągi słów (x_i) i (y_i),*
- *oblicza liczbę wszystkich wyborów symetrycznych dla tych dwóch ciągów słów,*
- *zapisuje wynik w pliku tekstowym LIC.OUT.*

Wejście

W pierwszym wierszu pliku tekstowego LIC.IN jest zapisana jedna liczba całkowita dodatnia $n \leqslant 30$.
W następnych n wierszach są zapisane kolejne słowa ciągu (x_i) — każde w osobnym wierszu.
W następnych n wierszach są zapisane — w taki sam sposób — kolejne słowa ciągu (y_i).
Wszystkie słowa składają się wyłącznie z małych liter alfabetu angielskiego od a do z i mają długość nie mniejszą niż 1 i nie większą niż 400.

Wyjście

W pliku tekstowym LIC.OUT należy zapisać jedną liczbę całkowitą nieujemną, a mianowicie liczbę wszystkich wyborów symetrycznych.

Przykład

Dla pliku LIC.IN:

```
5
ab
a
a
ab
a
a
```

```
baaaa
a
a
ba
```

poprawnym rozwiązaniem jest plik LIC.OUT:

```
12
```

Twój program powinien szukać pliku LIC.IN w katalogu bieżącym i tworzyć plik LIC.OUT również w bieżącym katalogu. Plik zawierający napisany przez Ciebie program w postaci źródłowej powinien mieć nazwę LIC.???, gdzie zamiast ??? należy wpisać co najwyżej trzyliterowy skrót nazwy użytego języka programowania. Ten sam program w postaci wykonalnej powinien być zapisany w pliku LIC.EXE

UWAGA DO TREŚCI ZADANIA

W treści zadania znalazła się pewna niedokładność. Ostatnie zdanie punktu „Wejście" miałoby sens bliższy zamierzonemu przez autora zadania, gdyby brzmiało:

> *Wszystkie słowa składają się wyłącznie z małych liter alfabetu angielskiego od a do z i mają łączną długość nie mniejszą niż 1 i nie większą niż 400.*

ROZWIĄZANIE

Zadanie jest dosyć proste, łatwo rozwiązuje się je przez sprowadzenie do problemu obliczenia liczby wszystkich ścieżek prowadzących od wierzchołka początkowego **start** do wierzchołka końcowego **koniec** w pewnym grafie acyklicznym G. Jak się okaże, bezpośrednia konstrukcja grafu G w programie nie jest konieczna.

Rozwiązywanie zadania w sposób „naiwny" przez zbadanie wszystkich możliwych wyborów prowadzi do programu działającego w przypadku pesymistycznym zbyt długo, aby można było taki program zaakceptować (liczba możliwych wyborów jest wykładnicza).

UWAGA: Jeśli zastąpimy liczenie wszystkich wyborów przez liczenie wszystkich różnych słów symetrycznych, które możemy otrzymać, to zadanie staje się dużo trudniejsze. Nie jest znany żaden wielomianowy algorytm dla takiej wersji zadania.

Na wejściu dostajemy ciąg słów $x_1, \ldots, x_n$, $y_1, \ldots, y_n$. Załóżmy, że N jest sumą długości wszystkich słów. Słowa pamiętamy w tablicy T, jako jedno długie słowo $x_1y_1x_2y_2 \ldots x_ny_n$. Dodatkowo przechowujemy informacje o tym, gdzie w tablicy T są końce i początki słów x_i, y_i.

Chociaż nasze zadanie ma charakter tekstowy, to sprowadzimy je do problemu teoriografowego.

Wyobraźmy sobie, że mamy dwa wskaźniki przesuwające się po tablicy T. Jeden z nich „wędruje" od strony lewej do prawej, a drugi od prawej do lewej. Wskaźniki

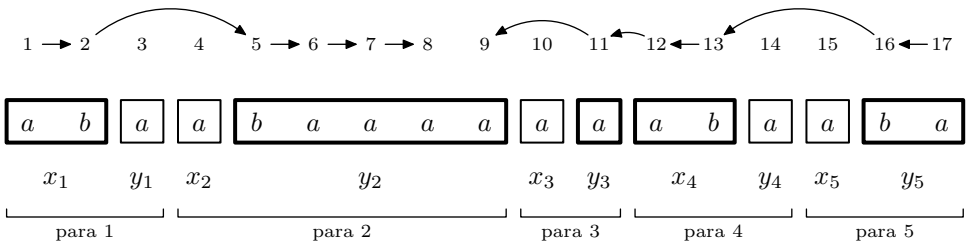
służą do wybierania po jednym słowie z każdej pary i przesuwają się w sposób zsynchronizowany, tak że tyle samo liter jest pobieranych zarówno z lewej, jak i z prawej strony. Przesuwając wskaźniki sprawdzamy jednocześnie symetrię tekstu. W momencie, gdy wskaźniki się spotkają wiemy, że znaleźliśmy słowo symetryczne i dokonaliśmy wyboru z każdej pary x_k, y_k. Wskaźniki przesuwamy tylko wtedy, gdy kolejne litery są identyczne (rozszerzamy symetrię od zewnątrz do środka).

Rozważmy skierowany acykliczny graf $G = (V, E)$, którego budowę omówimy na przykładzie z treści zadania. W tym celu posłużymy się rys. 1. Zbiór V składa się ze wszystkich par (i, j), gdzie $1 \leqslant i < j \leqslant N$, zwanych **wierzchołkami wewnętrznymi**, oraz dodatkowych wierzchołków *start* i *koniec*.

Każdy wewnętrzny wierzchołek jest parą (i, j) interpretowaną jako

(pozycja lewego wskaźnika, pozycja prawego wskaźnika)

Rys. 1 Załóżmy, że z każdej pary słów wybieramy słowo w pogrubionej ramce. Jest to wybór symetryczny. Odpowiada to przykładowej ścieżce w grafie G przedstawionej pod rysunkiem.



$$start \rightsquigarrow (1,17) \rightsquigarrow (2,16) \rightsquigarrow (5,13) \rightsquigarrow (6,12) \rightsquigarrow (7,11) \rightsquigarrow (8,9) \rightsquigarrow koniec$$

Tablica T zawiera słowo $ab|a|a|baaaa|a|a|ab|a|a|ba$, gdzie symbol '|' jedynie pokazuje granice między poszczególnymi słowami, ale w rzeczywistości nie ma go w tablicy. Zatem tablica ta ma długość $N = 17$.

Lewy wskaźnik przesuwa się zgodnie ze strzałkami $\rightarrow$ i $\curvearrowright$, a prawy zgodnie z $\leftarrow$ i $\curvearrowleft$. W grafie G mamy krawędź $(1, 17) \rightsquigarrow (2, 16)$ gdyż 2-gi i 16-ty symbol są identyczne, a przy tym jesteśmy wewnątrz pojedynczych słów. Natomiast krawędź $(2, 16) \rightsquigarrow (5, 13)$ bierze się stąd, że podobnie jak poprzednio, odpowiednie symbole są identyczne oraz pozycje 2 i 16 są końcami pojedynczych słów (czytając od lewej do prawej — pozycja 2 — i od prawej do lewej — pozycja 16), natomiast pozycje 5 i 13 są początkami pojedynczych słów z następnych par w kolejności.

Łatwo zauważyć następujący fakt:

Fakt 1: Liczba wyborów symetrycznych jest równa liczbie wszystkich ścieżek prowadzących od wierzchołka *start* do wierzchołka *koniec* w grafie acyklicznym G.

Liczbę ścieżek liczymy następująco. Przechodzimy graf w porządku topologicznym, to znaczy takim porządku, że dla każdego wierzchołka wszystkie jego poprzedniki

są odwiedzane wcześniej niż on sam. Liczymy liczbę ścieżek od wierzchołka *start* do każdego wierzchołka v, jako sumę liczb ścieżek dla jego bezpośrednich poprzedników. Wierzchołek $v = start$ ma jedną ścieżkę od siebie samego do siebie samego.

W programie oprócz tablicy T mamy tablice *nr_slowa* i *skok*, które służą do wskazywania początków i końców pojedynczych słów x_k, y_k w tablicy T oraz pomagają wyznaczać krawędzie grafu G. Sam graf jest przechowywany tylko pośrednio — dla każdego wierzchołka, do którego istnieje ścieżka z wierzchołka *start* pamięta się obliczaną przez program liczbę wszystkich takich ścieżek oraz liczbę poprzedników wierzchołka (wykorzystywaną przy przechodzeniu grafu w porządku topologicznym). Istotnym problemem technicznym jest duży rozmiar tablicy, w której przechowuje się powyższe wartości (tablica G w programie). Wykorzystując fakt, że jest to tablica trójkątna (używamy tylko komórek po jednej stronie przekątnej tablicy), można zmniejszyć rozmiar potrzebnej pamięci. Nie konstruuje się krawędzi grafu. Fakt istnienia krawędzi każdorazowo wyznacza się na podstawie zawartości tablic T, *nr_slowa* i *skok*.

TESTY

Do sprawdzania rozwiązań zawodników użyto 12-tu testów LIC0.IN–LIC11.IN.

- LIC0.IN — test z treści zadania;
- LIC1.IN–LIC8.IN — testy sprawdzające poprawność przyjętej metody rozwiązania;
- LIC9.IN–LIC11.IN — testy sprawnościowe. Ich celem było sprawdzenie szybkości działania algorytmu.