Krzysztof Diks
autor zadania, opracowanie

Adam Borowski
program wzorcowy

AB-słowa

Każdy niepusty ciąg, którego elementami są małe litery a i b, a także ciąg pusty nazywamy ab-słowem. Jeżeli $X = [x_1, \ldots, x_n]$ jest ab-słowem, a i, j takimi dowolnymi liczbami całkowitymi, że $1 \leqslant i \leqslant j \leqslant n$, to przez $X[i..j]$ będziemy oznaczali podsłowo X składające się z kolejnych liter $x_i, \ldots, x_j$. Powiemy, że ab-słowo $X = [x_1, \ldots, x_n]$ jest ładnie zbudowane, *jeżeli zawiera tyle samo liter a, co b i dla każdego $i = 1, \ldots, n$ podsłowo $X[1..i]$ zawiera co najmniej tyle samo liter a, co liter b.*

Podamy teraz indukcyjną definicję podobieństwa *ładnie zbudowanych ab-słów:*

- *każde dwa puste ab-słowa (tzn. nie zawierające żadnych liter) są podobne,*
- *dwa niepuste, ładnie zbudowane ab-słowa $X = [x_1, \ldots, x_n]$ i $Y = [y_1, \ldots, y_m]$ są podobne, jeżeli są tej samej długości ($n = m$) i jest spełniony jeden z następujących warunków:*
  - *$x_1 = y_1$, $x_n = y_n$ oraz $X[2..n-1]$ i $Y[2..n-1]$ są ładnie zbudowanymi ab-słowami*
  - *istnieje i, $1 \leqslant i < n$, takie, że słowa $X[1..i]$, $X[i+1..n]$ są ładnie zbudowanymi ab-słowami i*
    - *$Y[1..i]$, $Y[i+1..n]$ są ładnie zbudowanymi ab-słowami i $X[1..i]$ jest podobne do $Y[1..i]$ oraz $X[i+1..n]$ jest podobne do $Y[i+1..n]$ lub*
    - *$Y[1..n-i]$, $Y[n-i+1..n]$ są ładnie zbudowanymi ab-słowami i $X[1..i]$ jest podobne do $Y[n-i+1..n]$ oraz $X[i+1..n]$ jest podobne do $Y[1..n-i]$.*

Stopniem zróżnicowania *niepustego zbioru S ładnie zbudowanych ab-słów nazywamy największą liczbę ab-słów, które można wybrać z S tak, żeby żdane dwa wybrane słowa nie były do siebie podobne.*

Zadanie

Napisz program, który:

- *wczyta z pliku tekstowego ABS.IN elementy zbioru S;*
- *policzy stopień zróżnicowania zbioru S;*
- *zapisze wynik w pliku tekstowym ABS.OUT.*

Wejście

W pliku tekstowym ABS.IN są zapisane:

- *w pierwszym wierszu liczba n elementów zbioru S, $1 \leqslant n \leqslant 1000$;*
- *w kolejnych n wierszach elementy zbioru S, tj. ładnie zbudowane ab-słowa, po jednym w każdym wierszu; pierwsza litera każdego ab-słowa jest pierwszym symbolem w wierszu*

i między kolejnymi literami w słowie nie ma żadnych innych znaków; każde słowo ma długość co najmniej 1, a co najwyżej 200.

Wyjście

W pierwszym i jedynym wierszu pliku tekstowego ABS.OUT należy zapisać jedną liczbę całkowitą — stopień zróżnicowania S.

Przykład

Dla pliku wejściowego ABS.IN o zawartości:

```
3
aabaabbbab
abababaabb
abaaabbabb
```

poprawnym rozwiązaniem jest plik ABS.OUT o zawartości:

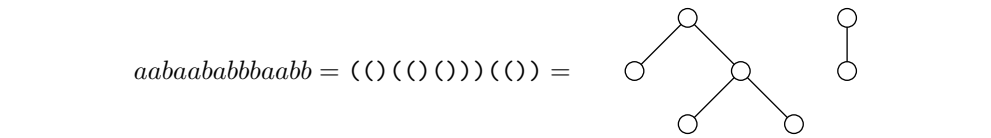
```
2
```

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

ROZWIĄZANIE

Łatwo zauważyć, że ab-słowo jest „ładnie zbudowane" wtedy i tylko wtedy, gdy po zamianie każdej litery a na nawias otwierający „(" i każdej litery b na nawias zamykający „)" otrzymamy poprawnie zbudowane **wyrażenie nawiasowe**. Każdemu poprawnie zbudowanemu wyrażeniu nawiasowemu odpowiada jednoznacznie ciąg drzew (z korzeniami) otrzymany w sposób zilustrowany na rysunku 1.

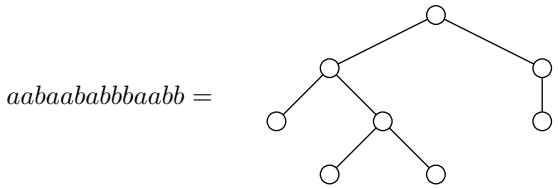
Rys. 1 Przykład odpowiedniości ab-słów, wyrażeń nawiasowych i ciągów drzew



Podczepiając kolejne drzewa w ciągu do jednego, dodatkowego wierzchołka, otrzymamy drzewo jednoznacznie reprezentujące dane ab-słowo lub równoważne mu wyrażenie nawiasowe. Nowy wierzcholek ustanawiamy korzeniem drzewa. Drzewo powstałe z dodania korzenia do ciągu drzew z rysunku 1 przedstawiono na rysunku 2.

Powiemy, że dwa drzewa (posiadające wyróżnione korzenie) T_1 i T_2 są **izomorficzne** wtedy i tylko wtedy, gdy spełniony jest jeden z następujących warunków:

Rys. 2 Ciąg (dwóch) drzew z rysunku 1 połączonych w jedno drzewo nowym wierzchołkiem (aktualnym korzeniem)



- Oba drzewa składają się tylko z pojedynczych wierzchołków.
- Jeśli $v_1, \ldots, v_k$ są synami korzenia drzewa T_1, a $u_1, \ldots, u_m$ są synami korzenia drzewa T_2, to $k = m$ (liczby synów korzeni w obu drzewach są takie same) i istnieje wzajemnie jednoznaczne przyporządkowanie synów korzenia T_1 synom korzenia T_2 takie, że poddrzewa zaczepione w odpowiadających sobie wierzchołkach są izomorficzne.

Na przykład drzewa odpowiadające wyrażeniom nawiasowym

```
( () (()) ((())) (((())())) )
( (()) (((())())) ((())) () )
( (()) (()(())) () ((())) )
```

są parami izomorficzne.

Nietrudno zauważyć, że relacja izomorfizmu drzew jest relacją:

- zwrotną (każde drzewo jest izomorficzne z samym sobą),
- symetryczną (jeśli drzewo T jest izomorficzne z drzewem T', to drzewo T' jest izomorficzne z drzewem T),
- przechodnią (jeśli drzewo T jest izomorficzne z drzewem T' i drzewo T' jest izomorficzne z drzewem T'', to drzewo T jest izomorficzne z drzewem T'').

O relacji, która jest zwrotna, symetryczna i przechodnia mówimy, że jest **relacją równoważności**.

Ponieważ ab-słowom jednoznacznie odpowiadają drzewa, to relację izomorfizmu drzew można w naturalny sposob przenieść na izomorfizm ab-słow: dwa ab-słowa są izomorficzne wtedy i tylko wtedy, gdy odpowiadające im drzewa są izomorficzne.

Analogicznie relacja podobieństwa na słowach zdefiniowana w treści zadania przenosi się w naturalny sposób na podobieństwa odpowiadających im drzew — podobieństwu podsłów odpowiada podobieństwo odpowiadających im ciagów poddrzew.

Niestety, relacja podobieństwa zdefiniowana w treści zadania nie jest relacją równoważności. Relacja podobieństwa ab-słów jest zwrotna, symetryczna, ale nie jest przechodnia, co, jak się okaże, bardzo utrudnia skonstruowanie efektywnego algorytmu dla naszego zadania.

Jeśli weźmiemy słowa (dla przejrzystości używamy nawiasów zamiast liter a i b)

```
A = ( () (()) ((())) (((()))) )
B = ( (()) (((()))) ((())) () )
C = ( (()) (((()))) () ((())) )
```

to A jest podobne do B, B jest podobne do C, ale A nie jest podobne do C!

Problem obliczania stopnia zróżnicowania danego zbioru ab-słów można wyrazić w terminach teorii grafów.

Powiemy, że wierzchołki grafu są **niezależne** wtedy i tylko wtedy, gdy nie są połączone krawędzią. Podzbiór wierzchołków grafu nazwiemy **zbiorem wierzchołków niezależnych** jeżeli wierzchołki w tym zbiorze są parami niezależne (tzn. jeżeli nie ma pomiędzy nimi żadnej krawędzi). Problem obliczania w grafie zbioru wierzchołków niezależnych o największym rozmiarze jest problemem bardzo trudnym obliczeniowo. Nie jest dla tego problemu znany żaden algorytm, który działałby w czasie wielomianowym, a istnienie takiego algorytmu jest bardzo mało prawdopodobne. Czytelnik, który znajdzie wielomianowy algorytm wyznaczający naliczniejszy zbiór wierzchołków niezależnych rozwiąże jeden z fundamentalnych problemów w informatyce, znany pod nazwą $P \stackrel{?}{=} NP$, zdobędzie sławę i pieniądze. Więcej na ten temat można znaleźć w [12].

Problem znajdowania stopnia zróżnicowania danego zbioru ab-słow można sprowadzić do obliczania rozmiaru najliczniejszego zbioru wierzchołków niezależnych w pewnym grafie. Tym grafem jest graf podobieństwa ab-słów, w którym wierzchołkami są dane ab-słowa, a dwa wierzchołki są połączone krawędzią wtedy i tylko, gdy odpowiadające im ab-słowa są podobne. Zauważmy, że gdyby relacją na słowach byla relacja izomorfizmu, to dla policzenia stopnia zróżnicowania zbioru ab-słów w tej relacji wystarczyłoby znaleźć liczbę spójnych składowych w grafie izomorfizmu, co daje się zrealizować w czasie wielomianowym.

Pan Adam Borowski pokazał, że istnieje wielomianowy algorytm, który dla dowolnego n-wierzchołkowego grafu G konstruuje n ab-słów, dla których graf podobieństwa jest właśnie grafem G. Stopień zróżnicowania takiego zbioru ab-słów jest zatem równy rozmiarowi najliczniejszego zbioru wierzchołków niezależnych w G. Wynika stąd, że gdybyśmy potrafili rozwiązać nasze zadanie w czasie wielomianowym, to rozwiązalibyśmy problem $P = NP$.*

Algorytm wzorcowy dla zadania ab-słowa działa w czasie wykładniczym. Na taką złożoność algorytmu ma wpływ sposób obliczania maksymalnego zbioru niezależnego w grafie podobieństwa. Zbiór ten jest obliczany przy pomocy metody zwanej **przeszukiwaniem z powrotami**, która w tym wypadku polega na generowaniu w systematyczny sposób wszystkich niezależnych podzbiorów i wybraniu z nich najliczniejszego. Taka metoda ma szanse powodzenia tylko wtedy, gdy graf, w którym szukamy najliczniejszego zbioru niezależnych wierzchołków, jest małego rozmiaru. Algorytm wzorcowy gwarantuje, że dla danych o rozmiarze wyspecyfikowanym w treści zadania, grafy, w których obliczane są zbiory niezależnych wierzchołków, są małych rozmiarów.

W jaki sposób to osiągamy? Nietrudno zauważyć, że dwa ab-słowa, które nie są izomorficzne, nie są do siebie podobne. Wynika stąd, że możemy zbiór wszystkich ab-słów podzielić na klasy, w których słowa są parami izomorficzne i dla każdej z nich

---

* Pojawia się tu subtelna różnica polegająca na tym, że z rozwiązania wynika algorytm, który znajduje w czasie wielomianowym rozmiar najliczniejszego zbioru wierzchołków niezależnych, a nie sam zbiór. Dysponując jednakże takim algorytmem możemy w prosty sposób wyznaczyć w czasie wielomianowym również elementy tego zbioru. Konstrukcję takiego rozszerzenia algorytmu pozostawiamy jako ćwiczenie dla Czytelnika.

policzyć oddzielnie stopień zróżnicowania. Poszukiwanym stopniem zróżnicowania całego zbioru jest suma stopni zróżnicowania poszczególnych klas. Algorytm podziału zbioru wszystkich ab-słów na klasy słów parami izomorficznych jest następujący:

- każdemu ab-słowu przyporządkowujemy sygnaturę, która jednoznacznie identyfikuje ab-słowa z tej samej klasy;
- sortujemy ab-słowa względem ich sygnatur;
- wyznaczamy klasy słów parami izomorficznych, jako ciągi kolejnych słów o takich samych sygnaturach.

Sygnaturę o potrzebnych nam własnościach można wybrać na wiele sposobów. Tutaj za sygnaturę przyjęliśmy drzewo ab-słowa przekształcone do **postaci kanonicznej** za pomocą permutowania synów każdego wierzchołka w tym drzewie.

Zdefiniujmy na drzewach następujący porządek liniowy (zob. [11]), oznaczany przez $\prec$:

(1) Jeżeli korzeń drzewa T ma mniej synów od korzenia drzewa T', to $T \prec T'$.

(2) Jeżeli dla uporządkowanych (w porządku $\prec$) ciągów synów korzeni T i T', odpowiednio $v_1, \ldots, v_k$ i $u_1, \ldots, u_k$, zachodzi $v_k = u_k, \ldots, v_{j+1} = u_{j+1}, v_j \prec u_j$ to $T \prec T'$.

Powiemy, że drzewo jest w **postaci kanonicznej**, jeśli synowie każdego wierzchołka w tym drzewie są uporządkowani nierosnąco zgodnie z porządkiem $\prec$ dla poddrzew, których są korzeniami.

Zdefiniujemy dodatkowo relację silnego podobieństwa, która pozwala na zmniejszenie rozmiaru grafu podobieństwa, w którym mamy znajdować najliczniejszy zbiór niezależnych wierzchołków.

Dwa ab-słowa A i B są **silnie podobne** wtedy i tylko wtedy, gdy dla każdego ab-słowa C, słowo A jest podobne do słowa C wtedy i tylko wtedy, gdy słowo B jest podobne do C.

Tak jak w przypadku podobieństwa, silne podobieństwo przenosi się w naturalny sposób na drzewa odpowiadające ab-słowom.

Łatwo sprawdzić, że wszystkie słowa, które są do siebie silnie podobne, są nierozróżnialne w sensie relacji podobieństwa — każdą grupę silnie podobnych słów możemy zastąpić jej jednym reprezentantem („skleić"), a stopień zróżnicowania zbioru słów nie zmieni się. Problem w tym, że nie dysponujemy szybkim algorytmem sprawdzającym silne podobieństwo słów. Spróbujemy jednak posłużyć się algorytmem, który czasami nie stwierdza silnego podobieństwa tam, gdzie ono zachodzi, ale za to jest szybki. Niedoskonałość algorytmu spowoduje, że po wykonaniu operacji sklejania pozostaną słowa, które są silnie podobne, ale nie zostały sklejone. No cóż, trudno. Okazuje się jednak, że zmniejszenie rozmiaru grafu, które dokonuje się w wyniku sklejania wierzchołków, jest wystarczające dla naszych celów.

Oto szybki algorytm „częściowego sprawdzania" silnego podobieństwa dwóch słów A i B:

- Jeśli liczba synów korzenia drzewa słowa A jest różna od liczby synów korzenia drzewa słowa B, to słowa nie są silnie podobne.

Załóżmy zatem, że liczby synów obu korzeni są takie same.

- Jeśli liczba synów jest mniejsza niż 4, synów korzeni drzew A i B sortujemy wg sygnatur zaczepionych w nich poddrzew. Jeśli odpowiadające sobie w tym porządku poddrzewa z obu drzewach A i B są silnie podobne, to słowa A i B też są silnie podobne.
- Jesli liczba synów jest większa od 3, sprawdzamy silne podobieństwo kolejnych (bez sortowania!) par poddrzew drzew słów A i B, których korzenie są synami korzeni tych drzew. Jeśli wszyskie pary poddrzew są silnie podobne, to drzewa A i B są też silnie podobne.

Oczywiście, jeśli słowa A i B są silnie podobne, to są podobne.

A oto wzorcowy algorytm obliczania stopnia zróżnicowania wejściowego zbioru ab-słów.

- Dzielimy dany zbiór ab-słów na klasy słów parami izomorficznych.
- Wewnątrz każdej klasy:
  - Znajdujemy zbiory słów silnie podobnych. Z każdego zbioru słów silnie podobnych usuwamy wszystkie słowa oprócz jednego.
  - Dla pozostałych słów budujemy graf podobieństwa.
  - W grafie podobieństwa znajdujemy największy zbiór wierzcholkow niezależnych.

Suma rozmiarów największych zbiorów wierzchołków niezależnych w każdej z klas słów parami izomorficznych jest poszukiwanym stopniem zróżnicowania wejściowego zbioru ab-słów.

TESTY

Do sprawdzania rozwiązań zawodników użyto 13-tu testów ABS0.IN–ABS12.IN. Test ABS0.IN był testem z treści zadania. W testach ABS3.IN, ABS5.IN, ABS6.IN, ABS7.IN, ABS11.IN relacja podobieństwa na zbiorze słów wejściowych pokrywała się z relacją izomorfizmu, przy czym testy ABS5.IN–ABS7.IN były testami o dużych rozmiarach (1000 słów o długościach dochodzących do 200 znaków). Pozostałe testy zaliczały tylko programy, których autorzy zauważyli, że relacja podobieństwa nie jest relacją rownoważności. W szczególności rozwiązanie dla zestawu danych ABS9.IN sprowadzało się do znalezienia najliczniejszego zbioru wierzchołków niezależnych w pewnym grafie ośmiowierzchołkowym.