Wojciech Guzicki
Treść zadania, Opracowanie

Paweł Wolff
Program

OI, Etap I

Skarbonki

Smok Bajtazar ma n skarbonek. Każdą skarbonkę można otworzyć jej kluczem lub rozbić młotkiem. Bajtazar powrzucał klucze do pewnych skarbonek, pamięta przy tym który do której. Bajtazar zamierza kupić samochód i musi dostać się do wszystkich skarbonek. Chce jednak zniszczyć jak najmniej z nich. Pomóż Bajtazarowi ustalić, ile skarbonek musi rozbić.

Zadanie

Napisz program, który:

- *wczyta ze standardowego wejścia liczbę skarbonek i rozmieszczenie odpowiadających im kluczy,*
- *obliczy minimalną liczbę skarbonek, które trzeba rozbić, aby dostać się do wszystkich skarbonek,*
- *wypisze wynik na standardowe wyjście.*

Wejście

W pierwszym wierszu standardowego wejścia znajduje się jedna liczba całkowita n ($1 \leqslant n \leqslant 1\,000\,000$) — tyle skarbonek posiada smok. Skarbonki (jak również odpowiadające im klucze) są ponumerowane od 1 do n. Dalej na wejściu mamy n wierszy: w $(i+1)$-szym wierszu zapisana jest jedna liczba całkowita — numer skarbonki, w której znajduje się i-ty klucz.

Wyjście

W pierwszym i jedynym wierszu standardowego wyjścia należy zapisać jedną liczbę całkowitą — minimalną liczbę skarbonek, które trzeba rozbić, aby dostać się do wszystkich skarbonek.

Przykład

Dla danych wejściowych:

```
4
2
1
2
4
```

poprawnym wynikiem jest:

```
2
```

W powyższym przykładzie wystarczy rozbić skarbonki numer 1 i 4.

Rozwiązanie

Interpretacja grafowa

Zadanie ma naturalną interpretację w języku teorii grafów. Utwórzmy graf skierowany G mający n wierzchołków ponumerowanych liczbami od 1 do n. Wierzchołki o numerach i i j są połączone krawędzią (biegnącą od i do j), jeśli klucz do skarbonki i znajduje się w skarbonce j. Zauważmy, że z każdego wierzchołka wychodzi dokładnie jedna krawędź; stąd wynika, że w grafie jest dokładnie n krawędzi. Oczywiście do jednego wierzchołka może wchodzić wiele krawędzi, w takim przypadku w grafie będą także wierzchołki, do których nie wchodzi żadna krawędź. Jeśli do jednego wierzchołka wchodzi wiele krawędzi, to znaczy, że w odpowiedniej skarbonce znajduje się wiele kluczy. Jeśli zaś do jakiegoś wierzchołka nie wchodzi żadna krawędź, to znaczy, że w skarbonce o numerze równym numerowi tego wierzchołka nie ma żadnego klucza.

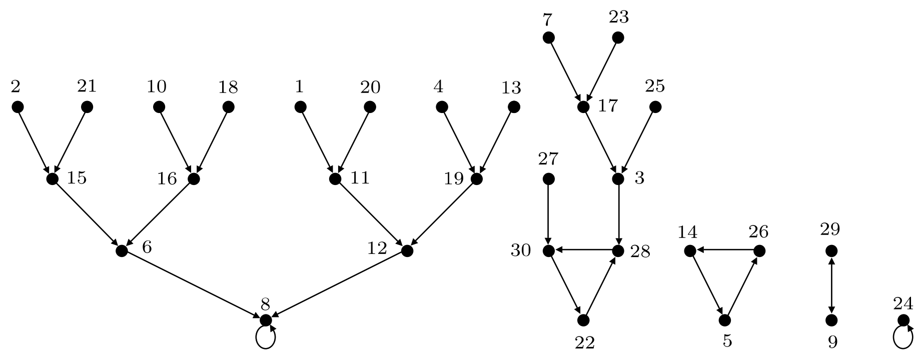
Przykład Popatrzmy na przykład takiego grafu. Przypuśćmy, że $n = 30$ oraz w pliku wejściowym (po liczbie 30) znajdują się następujące liczby:

$$11, 15, 28, 19, 26, 8, 17, 8, 29, 16, 12, 8, 19, 5, 6, 6, 3, 16, 12, 11, 15, 28, 17, 24, 3, 14, 30, 30, 9, 22.$$

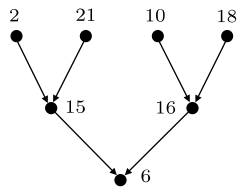
Inaczej mówiąc, w grafie G znajdują się następujące krawędzie:

1	$\rightarrow$	11	2	$\rightarrow$	15	3	$\rightarrow$	28	4	$\rightarrow$	19	5	$\rightarrow$	26
6	$\rightarrow$	8	7	$\rightarrow$	17	8	$\rightarrow$	8	9	$\rightarrow$	29	10	$\rightarrow$	16
11	$\rightarrow$	12	12	$\rightarrow$	8	13	$\rightarrow$	19	14	$\rightarrow$	5	15	$\rightarrow$	6
16	$\rightarrow$	6	17	$\rightarrow$	3	18	$\rightarrow$	16	19	$\rightarrow$	12	20	$\rightarrow$	11
21	$\rightarrow$	15	22	$\rightarrow$	28	23	$\rightarrow$	17	24	$\rightarrow$	24	25	$\rightarrow$	3
26	$\rightarrow$	14	27	$\rightarrow$	30	28	$\rightarrow$	30	29	$\rightarrow$	9	30	$\rightarrow$	22

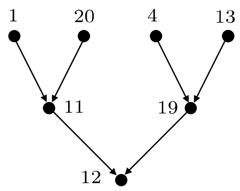
A oto rysunek tego grafu:



Zauważmy, że przedstawiony graf ma 5 składowych (czyli, mówiąc językiem potocznym, składa się z pięciu oddzielnych kawałków) i każda z tych składowych ma postać cyklu (być może długości 1), do którego dochodzą skierowane drzewa (czyli grafy bez cykli). W pierwszej składowej mamy cykl jednoelementowy złożony z wierzchołka o numerze 8, do którego dochodzą dwa drzewa binarne:



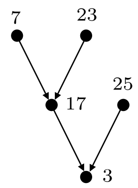
oraz



W drugiej składowej mamy cykl długości 3:

$$22 \rightarrow 28 \rightarrow 30 \rightarrow 22,$$

do którego dochodzą dwa drzewa. Jedno z nich składa się z jednego wierzchołka o numerze 27 i jest dołączone do wierzchołka o numerze 30; drugie zaś ma postać:



i jest dołączone do wierzchołka o numerze 28. Trzecia, czwarta i piąta składowa składają się z samych cykli, odpowiednio o długości 3, 2 i 1:

$$5 \rightarrow 26 \rightarrow 14 \rightarrow 5, \quad 9 \rightarrow 29 \rightarrow 9 \quad \text{oraz} \quad 24 \rightarrow 24.$$

■ *Przykład*

Graf G, w którym, jak w treści zadania, z każdego wierzchołka wychodzi dokładnie jedna krawędź, ma zawsze podobną postać: każda składowa ma jeden cykl (być może jednoelementowy), do którego są dołączone drzewa (być może zero drzew).

Zastanówmy się, dlaczego tak jest? Wybierzmy z grafu dowolny wierzchołek v i startując z niego poruszajmy się idąc wzdłuż krawędzi. Zauważmy, że nasza droga jest wyznaczona jednoznacznie, gdyż z każdego wierzchołka musimy wyjść jedyną możliwą krawędzią. Z drugiej strony, z tego samego powodu, możemy w opisany sposób poruszać się bez końca — zawsze mamy krawędź wychodzącą z wierzchołka. Zapisując kolejno odwiedzone wierzchołki otrzymujemy ciąg:

$$v = v_0 \rightarrow v_1 \rightarrow v_2 \rightarrow \ldots$$

Oczywiście któryś wierzchołek musi się w tym ciągu powtórzyć. Niech pierwszym takim wierzchołkiem będzie wierzchołek v_k. Mamy wtedy następujący ciąg, cykliczny od wyrazu v_k:

$$v = v_0 \rightarrow v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_{k-1} \rightarrow v_k \rightarrow v_{k+1} \rightarrow \ldots \rightarrow v_{k+l-1} \rightarrow v_{k+l} = v_k \rightarrow v_{k+1} \rightarrow \ldots$$

Zatem z każdego wierzchołka dochodzimy do pewnego cyklu. Łatwo spostrzec, że cykle te są rozłączne; wynika to ponownie stąd, że z każdego wierzchołka wychodzi dokładnie jedna krawędź. Wreszcie zauważmy (nietrudny dowód zostawimy jako ćwiczenie), że w każdej składowej z każdego wierzchołka dochodzimy do tego samego cyklu.

Oczywiście rozbicie którejkolwiek skarbonki odpowiadającej wierzchołkowi w cyklu pozwala otworzyć wszystkie skarbonki odpowiadające wierzchołkom składowej, w której ten cykl się znajduje. Oznacza to, że zadanie sprowadza się do policzenia składowych grafu.

Zliczanie składowych

Istnieje wiele metod znajdowania liczby składowych grafu. Wspomnimy najpierw o dwóch.

Jedna metoda polega na przeszukiwaniu grafu (w głąb lub wszerz). Wierzchołki, do których można dotrzeć z jednego wierzchołka poruszając się po krawędziach w dowolnym kierunku (także „pod prąd”), tworzą jedną składową.

Innym algorytmem jest tzw. algorytm *find - union*, polegający na sukcesywnym łączeniu w coraz większe zbiory wierzchołków połączonych krawędziami i należących do jednej składowej. Opis tego algorytmu Czytelnik może znaleźć na przykład w [17] (rozdział 22).

Jeszcze inną metodę znajdziemy w algorytmie stosowanym do tzw. sortowania topologicznego. Polega ona na usuwaniu z grafu kolejno wierzchołków, do których nie wchodzi żadna krawędź; w ten sposób każda składowa grafu zostaje zredukowana do cyklu, a cykle możemy już łatwo zliczyć. Opiszemy tę procedurę dokładniej. Przypuśćmy, że mamy dane dwie tablice:

$$Gdzie, Ile : \textbf{array}[1..n] \textbf{ of longint};$$

Tablica `Gdzie` wskazuje, w której skarbonce znajduje się dany klucz; czyli jeśli `Gdzie[5]=7`, to klucz 5 znajduje się w skarbonce 7, a więc w grafie G mamy krawędź $5 \to 7$. Tablica `Ile` wskazuje, ile krawędzi wchodzi do danego wierzchołka. Obie tablice wypełniamy początkowo na podstawie danych wczytanych z pliku wejściowego (szczegóły pominiemy).

Teraz przeglądamy kolejno wierzchołki i usuwamy z grafu te, do których nie wchodzi żadna krawędź. Usunięcie wierzchołka zaznaczamy wpisując 0 na odpowiedniej pozycji w tablicy `Gdzie`. A oto procedura usuwania wierzchołków:

```
procedure Usun;
  var
    i,j,k: longint;
  begin
    for i:=1 to n do
      if (Ile[i]=0) and (Gdzie[i]<>0) then
        begin
          j:=i;
          repeat
            k:=Gdzie[j];
            Gdzie[j]:=0;
            j:=k;
            Dec(Ile[j]);
          until (Ile[j]<>0)
        end
  end;
```

Po wykonaniu powyższej procedury w grafie G pozostaną same cykle. Tworzące je wierzchołki rozpoznajemy po tym, że odpowiadająca im wartość w tablicy `Ile` jest niezerowa. Cykle zliczamy za pomocą następującej procedury:

```
procedure ZliczCykle;
  var
    i,j : longint;
  begin
    Liczba:=0;
    for i:=1 to n do
      if Ile[i]<>0 then
        begin
          j:=Gdzie[i];
          while j<>i do
            begin
              Ile[j]:=0;
              j:=Gdzie[j]
            end;
          Inc(Liczba)
        end
  end;
```

Po zakończeniu procedury zmienna globalna `Liczba` zawiera szukaną liczbę cykli.

Zliczanie składowych — inny sposób

Chwila zastanowienia pokazuje, że składowe w grafie *G* możemy także zliczyć nie usuwając wierzchołków (i nie korzystając z tablicy `Ile`). W tym celu tworzymy tablicę:

Numer : **array**[1..*n*] **of longint**

Początkowo wypełniamy ją zerami. Następnie dla wierzchołka o numerze *i* poruszamy się po grafie zgodnie z kierunkiem krawędzi (jak wskazuje tablica `Gdzie`) i wpisujmy liczbę *i* w każde wolne (tzn. zawierające 0) miejsce w tablicy `Numer` odpowiadające numerowi odwiedzanego wierzchołka. To postępowanie może zakończyć się dwoma sposobami. Albo natkniemy się na wierzchołek, któremu już przypisaliśmy liczbę *i*, albo natkniemy się na wierzchołek, któremu była wcześniej przypisana jakaś liczba *j* różna od *i*. Pierwszy przypadek oznacza, że na naszej drodze natknęliśmy się na nowy cykl; zwiększamy więc liczbę cykli o 1. Drugi przypadek oznacza, że doszliśmy do miejsca, przez które już przechodziliśmy i z którego w dalszym ciągu doszlibyśmy do cyklu znalezionego wcześniej — takiej drogi nie musimy dalej kontynuować.

Przykład (cd.) Prześledźmy przedstawione postępowanie na przykładzie grafu *G* opisanego na początku tego rozdziału. Po zainicjowaniu tablicy `Gdzie` rozważamy kolejno wierzchołki $i = 1, 2, \ldots$ Dla $i = 1$ wpisujemy liczbę 1 w następujące miejsca tablicy `Numer`:

$$1, 11, 12, 8.$$

Następnie dla $i = 2$ liczbę 2 wpiszemy w miejsca:

$$2, 15, 6$$

i stwierdzimy, że następny wierzchołek (o numerze 8) był już odwiedzony, a cykl, do którego należy, już policzony. Liczbę 3 wpiszemy następnie w miejsca:

$$3, 28, 30, 22$$

i stwierdzimy, że znaleźliśmy nowy cykl. Liczbę 4 wpiszemy w miejsca 4 i 19 i stwierdzimy, że następny wierzchołek (o numerze 12) był już znaleziony. Liczba 5 doprowadzi nas do nowego cyklu:

$$5, 14, 26.$$

Wierzchołek 6 był już odwiedzony. Z wierzchołka 7 dojdziemy do odwiedzonego wcześniej wierzchołka 3 i tak dalej. ■ *Przykład (cd.)*

A oto algorytm oparty na opisanym pomyśle.

```
procedure Zlicz;
  var
    i,j : longint;
  begin
    Liczba:=0;
    for i:=1 to n do Numer[i]:=0;
    for i:=1 to n do
      begin
        j:=i;
        while Numer[j]=0 do
          begin
            Numer[j]:=i;
            j:=Gdzie[j]
          end;
        if Numer[j]=i then Inc(Liczba)
      end
  end
```

Podobnie jak poprzednio na zakończenie liczba cykli jest zapisana w zmiennej `Liczba`.

Przedstawione rozwiązanie zostało zaimplementowane w programie wzorcowym i działa w czasie $O(n)$.

Testy

Do zadania opracowano 16 testów. Początkowe testy służą sprawdzeniu poprawności rozwiązania w pewnych typowych sytuacjach (cykle jednoelementowe, dwuelementowe, cykle z dołączonymi drzewami, cykle bez dołączonych wierzchołków itp.). Następne testy służą sprawdzeniu szybkości działania programu. W ostatnich testach znaczna część grafu jest wygenerowana w sposób losowy.

Poniżej zamieszczamy szczegółowy opis testów:

Nazwa	**n**	**Opis**
ska1.in	32	mały test poprawnościowy: 12 wierzchołków tworzy 3 spójne składowe, reszta stanowi jednoelementowe spójne składowe (aby odpowiedź nie była zbyt małą liczbą, którą można z dużym prawdopodobieństwem zgadnąć na „chybił-trafił”)
ska2a.in	1	najmniejszy możliwy test
ska2b.in	11	mały test poprawnościowy: jeden cykl 3 elementowy, 8 cykli jednoelementowych
ska3.in	28	mały test poprawnościowy: drzewo binarne, którego korzeń jest jednoelementowym cyklem i dodatkowo 10 małych cykli
ska4.in	31	mały test poprawnościowy: kilka ścieżek zakończonych pętlami (cyklami jednoelementowymi), jeden cykl dwuelementowy i kilka cykli jednoelementowych
ska5.in	36	mały test poprawnościowy: cykl 5-elementowy z dołączonymi do niego ścieżkami, jedna ścieżka z pętlą na końcu plus kilka cykli jednoelementowych
ska6.in	2 187	średni test: cykl długości 37 z dołączonymi drzewami rozmiaru 50 plus część losowa grafu (stanowiąca odrębne składowe) rozmiaru 300
ska7.in	34 507	średni test: cykl długości 7 z dołączonymi drzewami rozmiaru 3 500 plus część losowa rozmiaru 10 000
ska8.in	455 091	duży test: 13 cykli długości od 1 do 13 z dołączonymi ścieżkami długości 5 000
ska9.in	20 100	średni test: 100 cykli długości od 1 do 100 z dołączonymi ścieżkami długości 1, oprócz jednej, która jest długości 10 001
ska10.in	300 100	duży test: 3 cykle długości 100 000 plus część losowa rozmiaru 100
ska11.in	701 000	duży test: 1 000 cykli długości 700 plus część losowa rozmiaru 1 000
ska12.in	50 000	średni test losowy
ska13.in	200 000	duży test losowy
ska14.in	500 000	duży test losowy
ska15.in	1 000 000	duży test: 990 000 cykli jednoelementowych plus część losowa rozmiaru 10 000