Hírlánc

Egy baráti társaság hírláncot alkot. Ez azt jelenti, hogy minden tag megadhat egy másik tagot, akinek továbbit minden hírt, amit megkap. A hírlánc azonaban még nem teljes, van olyan tag, aki még nem mondta meg, hogy kinek továbbítja a hírt. Minden alkalommal, amikor egy tag bejelenti, hogy ezek után kinek továbbítja a hírt, meg kell mondani, hogy hány tagnak kell eljutatni egy hírt, hogy az mindenkihez eljusson. Minden tag legfeljebb egyszer jelentheti be, hogy kinek továbbít hírt és lehet olyan tag is aki nem ad meg senkit, akinek hírt továbbítana.

A megoldáshoz a `adatok` modul három művelete használható.

Könyvtári műveletek

kezd : A tagok n számát adja. Ezt kell először hívni.

ujadat(a, b) : az `a` és `b` változókban ad meg egy új kapcsolatot. Tehát az `a` tag ezek után a `b` tagnak továbbítja a hírt. Kezdetben senki nem továbbít hírt.

valasz(db) : Minden `ujadat` hívás után meg kell hívni a `valasz` függényt, aminek az aktuális patramétere azon tagok száma legyen, ahány taghoz el kell juttatni egy új hírt, hogy azt mindenki megkapja. Az utolsó kérdésre adott `valasz` hívásra automatikusan befejeződik a program.

A adatok modul műveletei Pascal nyelv esetén

- `function kezd:longint;`
- `procedure ujadat(var x:longint; var y: longint): longint;`
- `procedure valasz(x: longint);`

A adatok modul műveletei C/C++ nyelv esetén

- `int kezd();`
- `void ujadat(int &x, int &y);`
- `void valasz(int x);`

Feltételek és korlátozások

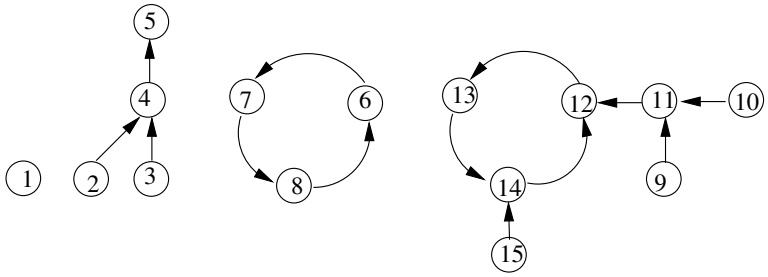
- A tagok n számára $2 \leq n \leq 100000$ teljesül.
- Minden `ujadat(a,b)` hívásra teljesül, hogy $1 \leq a \neq b \leq n$. Továbbá minden a-ra legfeljebb egy olyan `ujadat` hívás lesz, amelynek első paramétere a.
- Programod nem írhat és nem olvashat egyetlen fájlt sem, beleértve a standard bemenetet és kimenetet!
- Ha a program nem válaszol egy `ujadat` hívás után a `valasz` végrejatásával, akkor `Protokoll hiba` űzenettel befejeződik a program. Ugyancsak befejeződik a program hibás válasz esetén.

Gyakorlás

A könyvtári modul úgy használható, hogy a standard bemenet első sorába a tagok n számát kell írni. A további sorok mindegyike két egész számot tartalmazzon, `a b`, ahol `a b` egy új kapcsolat. A bemeneti adatsort a `0 0` sor zárja.
Időlimit: 0.1 mp
Memórialimit: 32 MiB

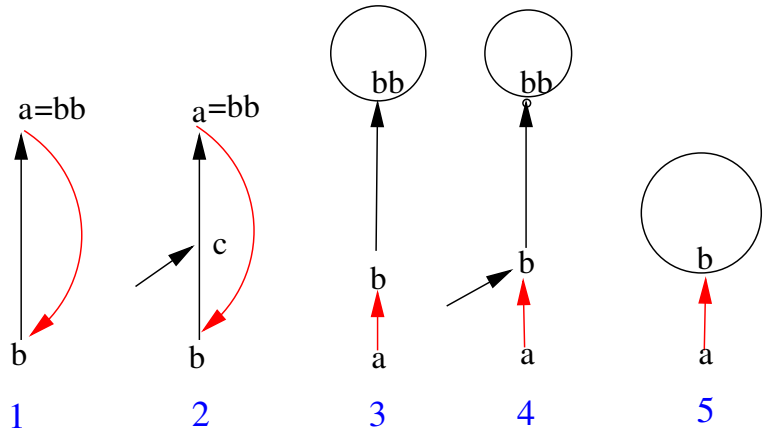
Megoldás

Tekintsük azt az $F : \{1, \ldots, n\} \to \{0, 1, \ldots, n\}$ függvényt, amelyre teljesül, hogy minden a-ra $F(a) = b$, ha az `a b` pár szerepelt már eddig egy `ujadat` műveletben paraméterként, egyébként $F(a) = 0$.
Tekintsük az F függvény GF gráfját, azaz a gráf pontjai 1,...,n és $a \to b$ akkor és csak akkor (irányított) él, ha $F(a) = b > 0$.
A függvény gráfja alapván meg tudjok mondani, hogy milyen választ kell adni:
A válasz a 0-befokú pontok száma, plusz azon körök száma, amelyekben minden pont befoka 1 (nincs farok). Például, ha a bementben eddig a (2,4), (3,4), (4,5),(6,7),(7,8),(8,6),(9,11),(10,11),(11,12),(12,13),(13,14) és (15,14) párok voltak, akkor a gráfot az alábbi ábra szemlélteti. A válasz 7, az 1,2,3,6,9,10 és 15 pontok.
Minden p pontra jelölje $LancVeg(p)$ azt a q pontot, amelyet az F függvény ismételt alkalmazásával kapunk,



1. ábra.

amire teljesül, hogy vagy $F(q) = 0$, vagy q körben van. Látható, hogy $LancVeg(p)$ létezik, és egyértelműen meghatározott minden pontra.
Tegyük fel, hogy az eddig beolvasott és feldolgozott adatok alapján a válasz V, azaz ennyi a 0-befokú pontok plusz a farok-nélküli körök száma, és az `a b` pár az új adat. Legyen $bb = LancVeg(b)$. A következő öt esetet kell megkülönböztetni:



2. ábra.

1. 1. eset: $Befok(b) = 0$, $LancVeg(b) = a$ és nincs a b-ből a-ba vetező F-úton olyan pont, amelynek befoka nagyobb, mint 1. Ekkor a válasz a változatlan V. Jelöljük meg a b-ből a-ba vezető úton lévő p pontokat, hogy $Korben[p] = 1$;

2. 2. eset: $Befok(b) = 0$, $LancVeg(b) = a$ és van olyan p pont a b-ből a-ba vetező F-úton , amelynek befoka nagyobb, mint 1. Ekkor a válasz $V - 1$. Jelöljük meg a b-ből a-ba vezető úton lévő p pontokat, hogy $Korben[p] = 2$, mivel van a keletkező körbe futó lánc.

3. 3. eset: $Befok(b) = 0$, $LancVeg(b) \neq a$. Ekkor a válasz $V - 1$.

4. 4. eset: $Befok(b) \neq 0$, $Korben(b) \neq 1$ (b vagy nincs körben, vagy olyan körben van, amelybe vezet farok). Ekkor a válasz V.

5. 5. eset: $Befok(b) \neq 0$, $Korben(b) = 1$. Ekkor a válasz $V - 1$. Jelöjük meg a b-t tartalmazó körben lévő p pontokat, hogy $Korben[p] = 2$ (eddig $Korben[p] = 1$ volt).

Ha a `LancVeg(b)` algoritmusát a naiv módon valósítjuk meg, tehát az alábbi algoritmussal, akkor a futási idő legrosszabb esetben négyzetesen függ n-től.

```
bb=b
ciklus amíg Korben[bb]=0 és F[bb]>0
    bb=F[bb]
ciklus vége
```

Példaul, ha a bemenet a `(2,1), (3,2), ... , (n, n-1)` párokat tartalmazza.
Az algoritmus lényegesen gyorsítható, ha a b-ből bb-be vezető úton minden pontot az `Elore` kapcsolattal bb-hez kapcsolunk. A további keresések soran ezt használjuk. Tehát az így módosított LancVeg algoritmus:

```
bb=b
ciklus amíg Korben[bb]=0 és F[bb]>0
    ha Elore[bb]>0 akkor
        bb=Elore[bb]
    egyébként
        bb=F[bb]
    elágazás vége
ciklus vége
ha b!=bb akkor
    x=b
    ciklus
        Elore[x]=bb
        x=F[x]
    amíg x!=bb
elágazás vége
```

Megvalósítás C++ nyelven

```
1  //hírlánc úttömörítéses algoritmusa
2  #include <iostream>
3  #include <list>
4  #include "adatok.h"
5  #define maxN 100001
6
7  using namespace std;
8
9  int BeFok[maxN];
10 int F[maxN];
11 int Korben[maxN];
12 int Elore[maxN];
13
14 int N;
15 int V;
16 void Init(){
17    for(int x=1;x<=N;x++){
18       F[x] = 0;
19       BeFok[x]=0;
20       Korben[x]=0;
21       Elore[x]=0;
22    }
23    V=N;
24 }
25 int LancVeg(int a){
26    int aa=a, x;
27    while(Korben[aa]==0 && F[aa]!=0){
28       if (Elore[aa]>0)
29          aa=Elore[aa];
30       else
31          aa=F[aa];
32    }
33    if(a!=aa){
34       x=a;
35       do{
36          Elore[x]=aa;
37          x=F[x];
38       }while(x!=0 && x!=aa);
39    }
40    return aa;
41 }
42 void szamol(int a, int b){
43    int bb=LancVeg(b);
44    if(BeFok[b]==0){
45       if(bb==a){//1. vagy 2. eset
46          int x=b;
47          bool farok=false;
48          do{
49             Korben[x]=1;
50             x=F[x];
51             if(!farok || BeFok[x]>1) farok=true;
52          }while(x!=0);
53          if(farok){//2. eset
```

```
54              V--;
55              x=b;
56              do{
57                 Korben[x]=2;
58                 x=F[x];
59              }while(x!=0);
60           }
61        }else{//3. eset
62           V--;
63        }
64     }else{//BeFok[b]>0
65        if(Korben[b]==1){//5. eset
66           int x=b;
67           do{
68              Korben[x]=2;
69              x=F[x];
70           }while(x!=b);
71           V--;
72        }//4. eset
73     }
74     F[a]=b;
75     BeFok[b]++;
76  }
77
78  int main(){
79     N=kezd();
80     Init();
81     int a,b;
82     for(;;){
83        ujadat(a,b);
84        szamol(a,b);
85        valasz(V);
86     }
87     return 0;
88  }
```