Képtárlátogatás

Egy n teremből álló képtárban teszünk látogatást. A termeket sorszámukkal azonosítják 1-től n-ig. Minden teremben minden ajtóra rá van írva, hogy az az ajtó melyik terembe vezet. A főbejárat az 1. terembe nyílik.

Feladat

Ijunk olyan programot, amely megad egy olyan útvonalat, amely az 1. teremből indul, oda ér vissza és minden terembe eljut!

Bemenet

A standard bemenet első sora egy egész számot tartalmaz, a termen n számát ($1 \leq n \leq 2000$). A további n sor tartalmazza a képtár szerkezetét, megadva minden teremre, hogy abból melyik teremekbe nyílik ajtó. Az állomány $i+1$-edik sorában azon termek sorszámai vannak felsorolva egy-egy szóközzel elválasztva, amelyekbe nyílik ajtó az i-edik teremből. A felsorolást egy 0 szám zárja.
A bemenet olyan képtárat ír le, amelynek biztosan van olyan bejárása, amely minden terembe eljut.

Kimenet

A standard kimenet első és egyetlen sora egy alkalmas bejárást tartalmazzon, a termek sorszámait egy-egy szóközzel elválasztva. Több megoldás esetén bármelyik megadható.

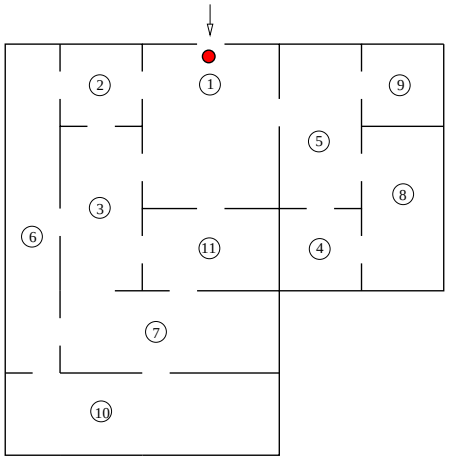
Példa

Bemenet

```
11
3 2 11 5 0
1 3 6 0
1 7 11 6 2 0
5 8 0
4 9 1 0
2 3 7 10 0
11 6 3 10 0
4 5 0
5 0
7 6 0
1 3 7 0
```

Kimenet

```
1 3 7 11 7 6 2 6 10 6 7 3 1 5 4 8 4 5 9 5 1
```



Korlátok

Időlimit: 0.1 mp.
Memórilimit: 32 MiB
Pontozás: a tesztesetek 40—

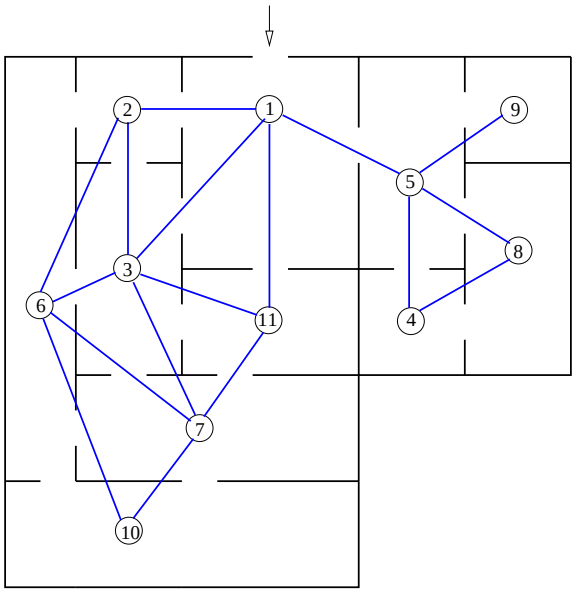
Megoldás

- Mit kell tudnunk?

1. Jártunk-e már az adott teremben?
2. Melyik teremből léptünk először az adott terembe?

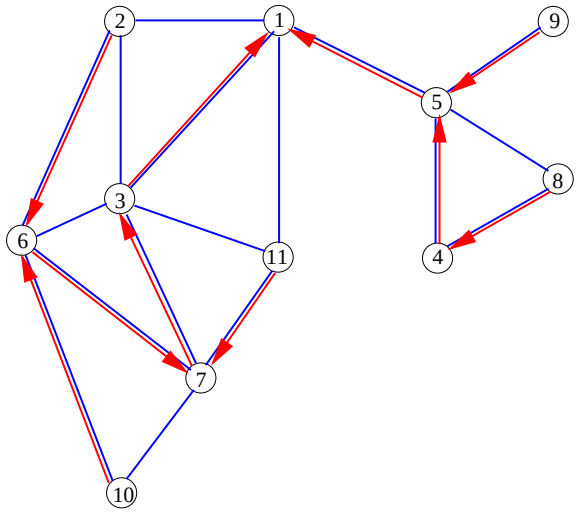
Az algoritmus:

```
hol:=1;                  // az aktuális terem
Honnan[1]:=0;            //az utcáról léptünk a föbejárat termébe
ciklus amíg (hol != 0)   //amíg vissza nem értünk a bejárathoz
   ha (hol-nak van hova benemjárt szomszédos terme) akkor
      Honnan[hova] := hol
      hol := hova;
   egyébként
      hol := Honnan[hol]
   elágazás vége
ciklus vége
```



1. ábra. A képtár gráfja

Tekintsük azt az irányítatlan gráfot, amelynek pontjai a termek, U, V akkor és csak akkor él a gráfban, ha az U teremből nyílik ajtó a V terembe.



2. ábra. A képtár gráf mélységi feszítőfája

Az 2. ábrán minden $U \to V$ piros nyíl azt mutatja, hogy az U terembe a V teremből léptünk először, tehát a V terembe kell visszalépni, ha az U teremnek nincs olyan szomszédja, amelyikben nem jártunk még. Ezt nevezzük a gráf mélységi feszítőfájának. Tehát $U \to V$ akkor és csak akkor éle a feszítőfának, ha $Honnan[U] = V$. A feszítőfa alapján egyszerűen meg tudunk adni minden ponthoz a kiindulási pontból oda vezető utat.

Megjegyezzük, hogy a fenti algoritmus alkalmazható irányított gráf bejárására is.

Megvalósítás C++ nyelven

```
1  #include <iostream>
2  #include <vector>
3  #define maxN 10001
4  using namespace std;
5  typedef vector<int> Graf[maxN];
6  Graf G;
7  int Honnan[maxN];
8  int n;
9  void Beolvas(){
10    int m,p,q;
11    cin>>n;
12    for (int i=1;i<=n;i++){
13       cin>>q;
14       while(q>0){
15          G[i].push_back(q);
16          cin>>q;
17       }
18    }
19 }
20 void NemrekurzivBejar(Graf G, int n, int p){
21    int Honnan[n+1];
22    for (int q=1;q<=n;q++)
23       Honnan[q]=-1;
24    Honnan[p]=0;
25    int hol=p,hova,i;
26
27    while(hol!=0){
28       cout<<hol<<"␣";
29       i=0;
30       while(i<G[hol].size() && Honnan[G[hol][i]]>=0) i++;
31       if (i<G[hol].size()){
32          hova=G[hol][i];
33          Honnan[hova]=hol;
34          hol=hova;
35       }else{
36          hol=Honnan[hol];//visszalépés
37       }
38    }
39    cout<<endl;
40 }
41 //rekurzív bejárás
42 void MelyBejar(int p){
43 //Globális: G,Honnan
44     cout<<p<<"␣";
45     for(int q:G[p])
46         if (Honnan[q]<0){//q-ban még nem jártunk
47             Honnan[q]=p;
48             MelyBejar(q);
49             cout<<p<<"␣";
50         }
51 }
52
53
```

```
54 int main(){
55    Beolvas();
56    for (int i=1; i<=n;i++) Honnan[i]=-1;
57    Honnan[1]=0;
58    MelyBejar(1);
59
60    return 0;
61 }
```