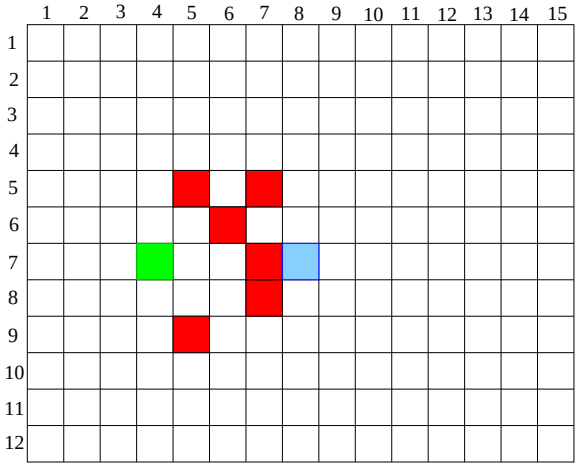
Algoritmizálás

Horváth Gyula
Szegedi Tudományegyetem
Természettudományi és Informatikai Kar
horvath@inf.u-szeged.hu

6. Gráfos feladatok

6.1. Feladat: Kiút keresés labirintusban.

Egy négyzetrácsos hálózattal leírható labirintusban adott start helyről adott cél helyre kell eljutni a lehető legrövidebb úton. A labirintusban minden mező vagy fal, vagy közlekedésre használható folyosó egy részlete. Egy lépésben szomszédos mezőre léphetünk, balra, jobbra, felfelé vagy lefelé.



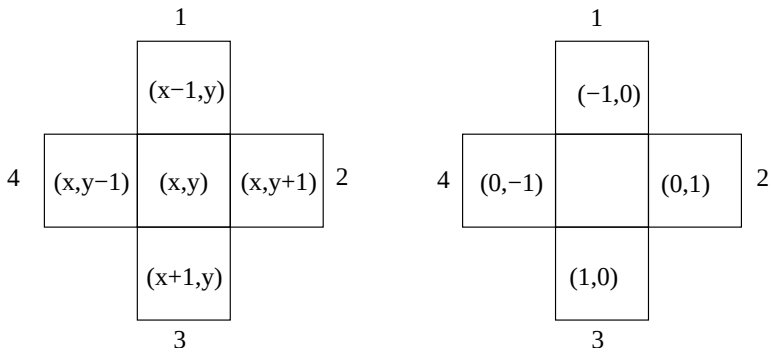
1. ábra.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1		8	7	6	7	8									
2	8	7	6	5	6	7	8								
3	7	6	5	4	5	6	7	8							
4	6	5	4	3	4	5	6	7	8						
5	5	4	3	2		6		8							
6	4	3	2	1	2										
7	3	2	1	0	1	2		8							
8	4	3	2	1	2	3		7	8						
9	5	4	3	2		4	5	6	7	8					
10	6	5	4	3	4	5	6	7	8						
11	7	6	5	4	5	6	7	8							
12	8	7	6	5	6	7	8								

Megoldás

Modell:

Tfh. a labirintus sorainak száma n, oszlopainak száma m. Tekintsük azt a $G = (V,E)$ gráfot, ahol
$V = \{(x,y) : 1 \leq x \leq n \wedge 1 \leq y \leq m \wedge (x,y)\text{nem fal}\}$
$(x,y) \rightarrow (\overline{x},\overline{y}) \in E$ akkor és csak akkor, ha $|x-\overline{x}|+|y-\overline{y}| = 1$ valamint (x,y) és $(\overline{x},\overline{y})$ nem fal.
Vagyis az (x,y) mezőről egyetlen lépéssel juthatunk az $(\overline{x},\overline{y})$ mezőre.
Tehát a feladat nem más, mint egy $(Xk,Yk) \rightsquigarrow (Xc,Xc)$ legrövidebb út keresése, ha a start mező (Xk,Yk) a célmező pedig (Xc,Xc).
A feladat tehát megoldható szélességi kereséssel, a labirintus gráf számított-gráf ábrázolását használva.



2. ábra. A lehetséges lépések

```
1  program Labirintus;
2  const
3    MaxMN=200;              {sorok és oszlopok max. száma }
4    Lep:array[1..4] of record x,y:-1..1 end={a lépések előre}
5      ((x:-1;y:0), (x:0;y:1), (x:1;y:0), (x:0;y:-1) );
6    VLep:array[1..4] of record x,y:-1..1 end={a lépések vissza}
7      ((x:1;y:0), (x:0;y:-1), (x:-1;y:0), (x:0;y:1) );
8    SorMeret=MaxMN*MaxMN;
9  type
10   Koord=0..MaxMN;
11   Pozicio=record x,y:Byte end;
12   Labirint=array[0..MaxMN, 0..MaxMN] of integer;
13   Sor=record Tar:array[1..SorMeret] of Pozicio; eleje,vege:Word end;
14 var
15   M,N:Word;               {a sorok és oszlopok száma }
16   K:Word;                 {a falak száma}
17   Xk,Yk:1..MaxMN;         {a kezdő mező koordinátái}
18   Xc,Yc:1..MaxMN;         {a cél mező koordinátái}
19   L:Labirint;             {a labirintus mátrixa }
20   S:Sor;                  {Sor a bejáráshoz}
21   x,y:Word;

22 procedure Beolvas;
23 {Global: M,N,}
24   var
25     Bef:Text;
26     i,x,y:Word;
27   begin
28     Assign(Bef,'labirint.be'); Reset(Bef);
29     ReadLn(Bef, M, N, K);
30     for x:=1 to M do
31       for y:=1 to N do L[x,y]:=0;
32     ReadLn(Bef, Xk, Yk, Xc, Yc);
33     for i:=1 to K do begin
34       ReadLn(Bef, x,y);
35       L[x,y]:=-1;
36     end{for K};
37
38     Close(Bef);
39   end{Beolvas};

40 procedure SorLetesit;
41 begin
42   S.eleje:=1; S.vege:=1;
43 end{SorLetesit};
44 procedure Sorba(p:Pozicio);
45 begin
46   S.Tar[S.vege]:=P;
47   Inc(S.vege);
48 end{Sorba};
49 procedure Sorbol(Var p:Pozicio);
50 begin
51   p:=S.Tar[S.eleje];
52   inc(S.eleje);
53 end{Sorbol};
54 Function NemUres():Boolean;
55 begin
56   NemUres:=S.eleje<S.vege;
```

```
57  end{NemUres};

58  procedure Bejar(x,y:Koord);
59  {Global: Xc,Yc,L}
60    var
61      i:Word;
62      p,q:Pozicio;
63    begin{Bejar}
64      SorLetesit;
65      p.x:=x; p.y:=y;
66      Sorba(p);
67      while NemUres Do begin
68        Sorbol(p);
69        if (p.x=Xc)And(p.y=Yc) then Break; {célba értünk}
70        for i:=1 to 4 Do begin         {lépés a négy lehetséges szomszéd felé}
71          q.x:=p.x+Lep[i].x;
72          q.y:=p.y+Lep[i].y;
73          if L[q.x, q.y]=0 then begin {itt még nem jártunk}
74            L[q.x,q.y]:=i+1;          {a lépés bejegyzése }
75            Sorba(q);                 {q-t betesszük az aktív pontok sorába}
76          end;
77        end{for i};
78      end{while};
79  end{Bejar};

80  procedure UtKiIro;
81  {Global: Xc,Yc, L}
82  Var KiF:Text;
83    Ut:array[1..MaxMN] of 1..4;
84    Tav,x,y,i:integer;
85  begin
86    Assign(Kif,'labirint.ki'); Rewrite(Kif);
87    if L[Xc,Yc]>0 then begin    {van út a célba}
88      Tav:=0;
89      x:=Xc; y:=Yc;
90       while L[x,y]>1 do begin
91         lepes:=L[x,y]-1;
92         x:=x+VLep[lepes].x;
93         y:=y+VLep[lepes].y;
94         inc(Tav);
95         Ut[Tav]:=lepes;
96       end{while};
97       WriteLn(Kif, Tav);{az út hosszának kiíratása}
98       for i:=Tav downto 1 do
99         write(KiF,Ut[i],' ');
100      writeLn(KiF);
101   end Else
102     WriteLn(Kif, 'Nincs út');
103   Close(Kif);
104 end{UtKiIro};

105 begin{Program}
106   Beolvas;
107
108   for x:=1 to M Do begin              {inicializálás}
109     L[x,0]:=1; L[x,M+1]:=-1;          {keret a tábla köré}
110   end;
111   for y:=1 to N Do begin
```

```
112       L[0,y]:=1; L[N+1,y]:=-1;
113     end;
114     L[Xk, Yk]:=1;                          {a kezdőpontban már jártunk}
115
116     Bejar(Xk, Yk);
117
118     UtKiIro;
119   end.
```

6.2. Feladat: Mérőkannák

Egy gazdának két kannája van, az egyik A literes, a másik pedig B. Szeretne kimérni pontosan L liter vizet. Az alábbi műveleteket lehet végezni a kimérés során:

1. Az A-literes kanna teletöltése
2. A B-literes kanna teletöltése
3. Az A-literes kanna kiürítése
4. A B-literes kanna kiürítése
5. Áttöltés az A-literesből a B-literesbe (amíg az tele nem lesz, ill. van A-ban)
6. Áttöltés a B-literesből az A-literesbe (amíg az tele nem lesz, ill. van B-ben)

Adjunk olyan algoritmust, amely meghatároz egy (legkevesebb lépésből álló) kimérést!

Bemenet

A `kimer.be` szöveges állomány első sora a két egész számot tartalmaz, a két kanna ürtartalmát: $AB(0 < A, B <= 200)$. A második sor a kimérendő mennyiséget tartalmazza.

Kimenet

A `kimer.ki` szöveges állomány első sora a kimérés lépéseinek (minimális) m számát tartalmazza. A második sor pontosan m egész számot tartalmazzon egy-egy szóközzel elválasztva, a kimérés lépéseit.

Példa bemenet és kimenet

bemenet

```
9 4
6
```

kimenet

```
8
1 5 4 5 4 5 1 5
```

kezdő állapot: (0,0)
1 öntéssel kapható állapotok: (9,0) (0,4)
2 öntéssel kapható állapotok: (9,4) (5,4) (4,0)
3 öntéssel kapható állapotok: (5,0) (4,4)
4 öntéssel kapható állapotok: (1,4) (8,0)
5 öntéssel kapható állapotok: (1,0) (8,4)
6 öntéssel kapható állapotok: (0,1) (9,3)
7 öntéssel kapható állapotok: (9,1) (0,3)
8 öntéssel kapható állapotok: (6,4) (3,0)
9 öntéssel kapható állapotok: (6,0) (3,4)
10 öntéssel kapható állapotok: (2,4) (7,0)
11 öntéssel kapható állapotok: (2,0) (7,4)
12 öntéssel kapható állapotok: (0,2) (9,2)

Megoldás

Minden állapot azonosítható egy (x,y) számpárral ($0 \leq x \leq A$, $0 \leq y \leq B$).
A kezdő állapot $(0,0)$.
Az öntögetés során tudnunk kell, hogy adott (x,y) állapotot elértük-e már?
Továbbá, ha elő is kell állítani egy öntési sorrrendet, akkor most nem elég azt tudni, hogy egy állapot melyik lépéssel keletkezett. Valóban, pl. ha a $(6,4)$ állapot az 5. lépéssel keletkezett, akkor az előző állapot lehetett $(9,1),(8,2)$ vagy $(7,3)$. Tehát minden elért állapothoz tároljuk azt az állapotot, amelyből keletkezett. Vegyünk egy EA tömböt, és $EA[x,y].x = 0$, ha az (x,y) állapotot még nem értük el, egyébként az állapotot, amelyből keletkezett.

```
1  program MeroKannak;
2  const
3    MaxV=200;
4    SorMeret=5000;
5  type
6    Allapot= record x,y : integer end;
7    Sor=record
8          eleje,vege:Word; Tar:array[1..SorMeret] of Allapot;
9        end;
10 var
11   A,B,L:Word;
12   EA: array[0..MaxV,0..MaxV] of Allapot;
13   S:Sor;
14
15 procedure Sorba(var S:Sor;v:Allapot);
16 begin
17     S.Tar[S.vege]:=v; Inc(S.vege);
18 end;
19 procedure Sorbol(var S:Sor;var v:Allapot);
20 begin
21   v:=S.Tar[S.eleje]; Inc(S.eleje);
22 end;

23 procedure Beolvas;
24   var
25     BeF: Text;
26 begin
27     Assign(BeF,'kimer.be'); Reset(BeF);
28     ReadLn(BeF,A,B);
29     ReadLn(BeF,L);
30     Close(BeF);
31 end{Beolvas};

32 procedure Szamit;
```

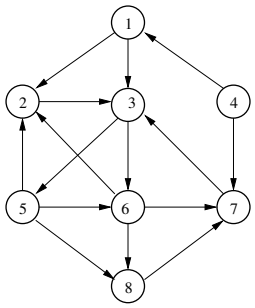
```
33  var
34    x,y:Word;
35    U,V:Allapot;
36  begin{}
37    for x:=0 to A do
38      for y:=0 to B do EA[x,y].x:=-1;
39    S.eleje:=1;S.vege:=1;
40    EA[0,0].x:=0;
41    U.x:=0; U.y:=0;
42    Sorba(S,U);
43    while (S.eleje<S.Vege) do begin
44      Sorbol(S,U);
45      if (U.x=L) then break; {kész, a kímérendő van a A-kannában}
46      if EA[A,U.y].x<0 then begin {1. lépés}
47         V.x:=A; V.y:=U.y;
48         EA[V.x,V.y]:=U;
49         Sorba(S,V);
50      end;
51      if EA[U.x,B].x<0 then begin {2. lépés}
52         V.x:=U.x; V.y:=B;
53         EA[V.x,V.y]:=U;
54         Sorba(S,V);
55      end;

56      if EA[0,U.y].x<0 then begin {3. lépés}
57         V.x:=0; V.y:=y;   EA[V.x,V.y]:=U;
58         Sorba(S,V);
59      end;
60      if EA[U.x,0].x<0 then begin {4. lépés}
61         V.x:=U.x; V.y:=0; EA[V.x,V.y]:=U; Sorba(S,V);
62      end;
63      if U.x+U.y<=B then begin  {öntés A-ból B-be}
64        V.x:=0; V.y:=U.x+U.y;
65      end else begin
66        V.x:=U.x-(B-U.y); V.y:=B;
67      end;
68      if EA[V.x,V.y].x<0 then begin {5. lépés}
69         EA[V.x,V.y]:=U;   Sorba(S,V);
70      end;
71      if U.x+U.y<=A then begin  {öntés B-ből A-ba}
72        V.y:=0; V.x:=U.x+U.y;
73      end else begin
74        V.y:=U.y-(A-U.x); V.x:=A;
75      end;
76      if EA[V.x,V.y].x<0 then begin {6. lépés}
77         EA[V.x,V.y]:=U;   Sorba(S,V);
78      end;
79     end {while az S sor nem üres };
80  end{Szamit};

81  procedure KiIr;
82  const
83    maxA=10000;
84  var KiF:Text;
85    x,y,m,i,lepes:integer;
86    Lehet:boolean;
87    szep:string;
88    U,V:Allapot;
```

```
89    Lepsor:array[1..MaxA]of Allapot;
90  begin
91      Assign(KiF,'kimer.ki'); Rewrite(KiF);
92      lehet:=false;
93      for y:=0 to B do if EA[L,y].x>0 then begin
94        lehet:=true; U.y:=y;
95        break;
96      end;
97      if not lehet then begin
98        writeln(KiF,0);
99        close(KiF); exit;
100     end;
101     U.x:=L; m:=0;
102     while (U.x<>0)or(U.y<>0) do begin
103       inc(m);
104       LepSor[m]:=U;
105       U:=EA[U.x,U.y];
106     end;
107     U.x:=0; U.y:=0;
108     writeln(KiF,m);
109     szep:='';
110     for i:=m downto 1 do begin
111       V:=LepSor[i];
112       if (V.x=A)and(U.y=V.y) then begin
113         lepes:=1;
114       end else if (U.x=V.x)and(V.y=B) then begin
115           lepes:=2;
116       end else if (V.x=0)and(U.y=V.y) then begin
117           lepes:=3;
118       end else if (U.x=V.x)and(V.y=0) then begin
119           lepes:=4;
120       end else if (V.x=0)and(V.y=U.x+U.y) or (V.x=U.x-(B-U.y))and(V.y=B) then begin
121           lepes:=5;
122       end else begin
123           lepes:=6;
124       end;
125       write(KiF,szep,lepes);
126       szep:='␣';
127       U:=V;
128     end{for i-};
129     writeln(KiF);    Close(KiF);
130 end{KiIr};

131
132 begin
133   Beolvas;
134   Szamit;
135   KiIr;
136 end.
```

6.3. Gráf adattípus, gráfok ábrázolása

6.3.1. Definíciók

1. **Irányítatlan gráf**:$G = (V,E)$
E rendezetlen $\{a,b\}, a,b \in V$ párok halmaza.
2. **Irányított gráf**:$G = (V,E)$
E rendezett (a,b) párok halmaza; $E \subseteq V \times V$.
3. **Multigráf**:



3. ábra. Példa gráf

$G = (V, E, Ind, Érk)$, $Ind, Érk : E \to V$
$Ind(e)$ az e él induló, $Érk(e)$ az érkező pontja.
Címkézett (súlyozott) gráf:$G = (V, E, C)$
$C : E \to Címke$
Minden irányítatlan $G = (V, E)$ gráf olyan irányított gráfnak tekinthető, amelyre teljesül, hogy $(\forall p, q \in V)((p, q) \in E \Rightarrow (q, p) \in E)$.
Jelölések
$KiEl(G, p) = \{q \in V : (p, q) \in E\}$
$BeEl(G, p) = \{q \in V : (q, p) \in E\}$
$KiFok(G, p) = |Ki(G, p)|$
$BeFok(G, p) = |Be(G, p)|$

6.3.2. Műveletek gráfokon

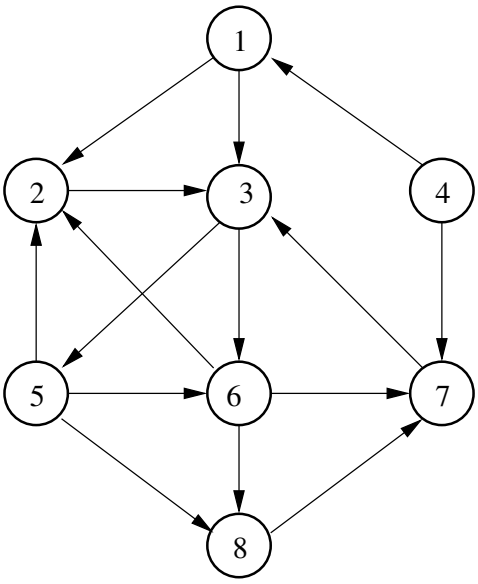
A továbbiakban feltételezzük, hogy a gráf pontjait természetes számokkal azonosítjuk, pontosabban $V \subseteq \{1, \ldots, n\}$
Milyen műveleteket akarunk gráfokon végezni?

- Pontokszama
- Elekszama
- KiFok(p)
- BeFok(p)
- PontBovit(p)
- PontTorol(p)
- ElBovit(p, q)
- ElTorol(p, q)
- VanEl(p, q)
- for q in KiEl(p) do M(p,q)

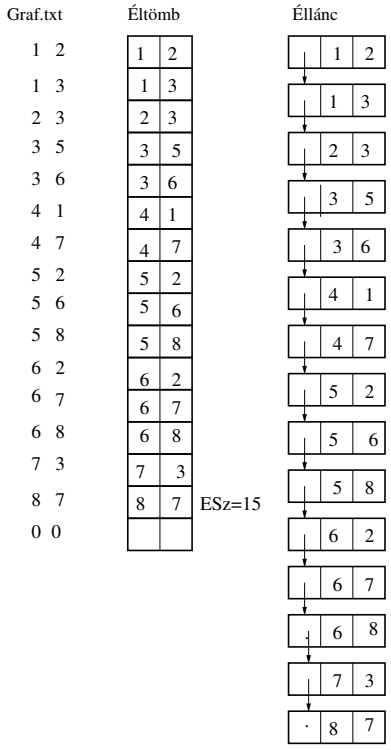
6.3.3. Gráfok ábrázolásai

Szempontok az adatszerkezet megválasztásához.

1. Az adott probléma megoldásához ténylegesen mely műveletek szükségesek.
2. Melyek a releváns műveletek, amelyek alapvetően befolyásolják az algoritmus futási idejét.
3. A tárigény az adott probléma esetén.



4. ábra. Példa gráf



5. ábra. Élhalmaztömb és élhalmazlánc

1. **Élhalmaztömb és élhalmazlánc**

2. **Kapcsolatmátrix (szomszédsági mátrix)**

G	1	2	3	4	5	6	7	8	9
1		1	1						
2			1						
3					1	1			
4	1						1		
5		1				1		1	
6		1					1	1	
7			1						
8							1		
9									

6. ábra. Kapcsolatmátrix

```
1  const
2    maxP=1000;   //a pontok max. száma
3  type
4    Graf=array[1..maxP, 1..maxP] of boolean;
5    CGraf=array[1..maxP, 1..maxP] of integer;
```

(p,q) akkor és csak akkor éle a gráfnak, ha $G[p,q] = true$.
Címkézett gráf esetén választani kell egy $nem \in Cimke$ értéket, amely nem fordul elő semmilyen él címkéjeként. (p,q) akkor és csak akkor éle a címkézett gráfnak, ha $G[p,q] != nem$, és a (p,q) él címkéjének értéke $G[p,q]$.

3. **Élhalmaz-tömb**

G	1	2	3	4	5	6	7	8	9	KiFok
1	2	3	0							2
2	3	0								1
3	5	6	0							2
4	1	7	0							2
5	2	6	8	0						3
6	2	7	8	0						3
7	3	0								1
8	7	0								1
9	0									0

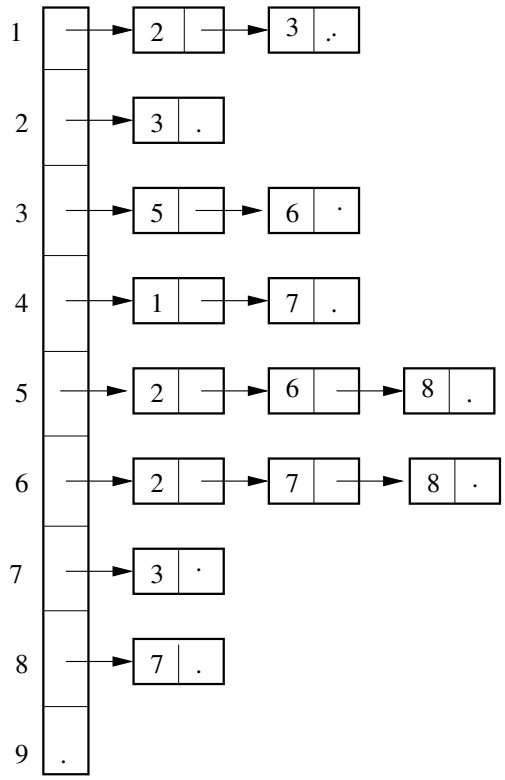
7. ábra. Gráf ábrázolása élhalmaz-tömbbel

```
1  const
2    maxP=1000; //a pontok max. száma
3  type
4    Graf=array[1..naxP, maxP] of integer;
5    KiFok=array[1..naxP] of integer;
```

A $Ki(G,p)$ halmaz bejárása:

```
1    for i:=1 to KiFok[p] do begin
2      q:=G[p,i];
3      M(p,q);
4    end;
```

4. **Élhalmaz-lánc**



8. ábra. Gráf ábrázolása élhalmaz-lánccal

a.) Statikus élhalmaz-lánc

```
1  const
2    maxP=1000;    //a pontok max száma
3    maxE=100000; //az élek max. száma
4  type
5    Lanc=longint;
6    Cella=record pont.integer; csat:Lanc end;
7    Graf=array[1..naxP] of Lanc;
8    Elek:array[1..maxE] of Cella;
```

b.) Dinamikus élhalmaz-lánc

```
1  const
2    maxP=1000;
3  type
4    Lanc=^Cella;
5    Cella=record pont.integer; csat:Lanc end;
6    Graf=array[1..naxP] of Lanc;
```

5. **Számított gráf**

Az élek halmazát explicite nem tároljuk, mert van olyan számítási eljárás, amely bármely két $p,q \in V$-re kiszámítja $VanEl(p,q)$-t.

Vagy, van olyan számítási eljárás, amely minden $p \in V$-re kigenerálja a $Ki(G,p)$ halmaz elemeit.

6.4. Elemi gráfalgoritmusok

6.4.1. Utak

Legyen $G = (V,E)$ irányított (irányítatlan) gráf.

6.1. definíció. *$p,q \in V$-re egy p-ből q-ba vezető út G-ben, jele: $\pi : p \rightsquigarrow q$, olyan*
$\pi = \langle p_0, p_1, \cdots, p_k \rangle$ pontsorozat, ahol
$p_i \neq p_j$, ha $i \neq j$, $p = p_0$ és $q = p_k$, továbbá
$p = q = p_0$, vagy $(\forall i \in \{1,\ldots,k\})\ ((p_{i-1}, p_i) \in E)$.

6.2. definíció. *A $\pi = p \rightsquigarrow q$ út hossza, $|\pi| = |p \rightsquigarrow q| = k$*

6.3. definíció. *p-ből q-ba vezető legrövidebb út hossza, p és q* távolsága*:*

$$\delta(p,q) = \begin{cases} \infty & \texttt{ha nincs } p \rightsquigarrow q \\ Min\{|\pi : p \rightsquigarrow q|\} & \pi : p \rightsquigarrow q \end{cases} \tag{1}$$

6.4. definíció. *A $\pi = \langle p_0, p_1, \cdots, p_k \rangle$ pontsorozatot a p_0-ból p_k-ba vezető* sétának *nevezzük, ha $(\forall i \in \{1,\ldots,k\})(p_{i-1}, p_i) \in E$*

6.5. definíció. *Ha $G = (V,E,C)$ élei a $C : E \to \mathbb{R}$ függvénnyel súlyozottak, akkor a $p \rightsquigarrow q$ út hossza*
$|p \rightsquigarrow q| = \sum_{i=1}^{k} C(p_{i-1}, p_i)$.
A p és q pont távolsága:

$$\delta(p,q) = \begin{cases} \infty & \texttt{ha nincs } p \rightsquigarrow q \\ Min\{|\pi : p \rightsquigarrow q|\} & \pi : p \rightsquigarrow q \end{cases} \tag{2}$$

6.6. definíció. *A $G = (V,E)$ (irányítatlan) gráfnak az $F = (\overline{V}, \overline{E})$ gráf a $r \in V$ gyökerű* feszítőfája*, ha*
1. F részgráfja G-nek ($\overline{V} \subseteq V, \overline{E} \subseteq E,$) és fa.
2. ($\forall p \in V$) ha van $r \rightsquigarrow p$ G-ben, akkor és csak akkor, ha van $r \rightsquigarrow p$ F-ben.

6.7. definíció. *A $G = (V,E)$ (irányítatlan, súlyozott) gráfnak az $F = (\overline{V}, \overline{E})$ gráf a $r \in V$ gyökerű* legrövidebb utak feszítőfája (LUF)*, ha*
1. F r-gyökerű feszítőfája G-nek, és
2. $\forall p \in V$-ra $\delta_G(r,p) = \delta_F(r,p)$.

Útproblémák
1. Adott $p,q \in V$-re van-e $p \rightsquigarrow q$ út?

2. Adott p-re az $Elr(p) = \{q : p \rightsquigarrow q\}$ halmaz kiszámítása.

3. Adott $p,q \in V$-re $\delta(p,q)$ és egy $p \rightsquigarrow q$ legrövidebb út kiszámítása.

4. Egy pontból induló legrövidebb utak : adott p-re minden q-ra $\delta(p,q)$ és egy $p \rightsquigarrow q$ legrövidebb út kiszámítása.

5. Minden p,q pontpárra $\delta(p,q)$ és egy $p \rightsquigarrow q$ legrövidebb út kiszámítása.

6.4.2. Szélességi keresés

Bemenet: $G = (V,E)$ (irányított vagy irányítatlan) gráf és egy $r \in V$ pont.
Kimenet: $D : V \to \mathbb{N}$, $Apa : V \to V$, hogy
$D(p) = \delta(r,p)$ és
az $F = (\overline{V},\overline{E})$ gráf, ahol
$\overline{V} = \{p : Apa(p) \neq null \vee p = r\}$,
$\overline{E} = \{(p,q) : Apa(q) = p \wedge p \neq null\}$
r-gyökerű LUF.

```
1  procedure SzeltBejar(p:PontTip);
2  //p–ből induló legrövidebb utak meghatározása
3  {Global:G, N, D, Apa }
4    var
5      S:SorTip;
6      u,v:PontTip;
7      i:Word;
8    begin{SzeltBejar}
9      for u:=1 to N do D[u]:=Inf;      { inicializálás}
10     D[p]:=0; Apa[p]:=0;
11     Uresit(S);
12     Sorba(S,p);
13
14     while NemUres(S) do begin
15       SorBol(S,u);
16       for i:=1 To KiFok[u] Do Begin { u –> v élre }
17         v:=G[u,i];
18         if D[v]=Inf then begin      { ha v meg nem elért }
19           Apa[v]:=u;
20           D[v]:=D[u]+1;
21           Sorba(S,v);
22         end;
23       end{for u–>v};
24     end{while};
25   end{SzeltBejar};
```

6.4.3. Mélységi keresés

```
1  procedure MelyBejar(p:PontTip);
2  //p–ből elérhető pontok meghatározása
3  {Global:G, N, Apa }
4    var
5      q:PontTip;
6      i:Word;
7    begin{MelyBejar}
8      for i:=1 To KiFok[p] Do Begin { p –> q élre }
9          q:=G[u,i];
10         if Apa[q]<0 then begin      { ha q meg nem elért }
11           Apa[q]:=p;
12           MelyBejar(q);
13         end;
14       end{for p–>q};
15   end{MelyBejar};
```

6.5. Feladat: Terv

Egy nagyszabású építkezés terve n különböző munka elvégzését írja elő. Minden munkát egy nap alatt lehet elvégezni. Egy napon több munkát is el lehet végezni párhuzamosan, feltéve, hogy a megelőzési feltételt betartjuk. Ez ezt jelenti, hogy vannak olyan

előírások, hogy egy adott a munka elvégzése meg kell, hogy előzze más adott b munka elégzését. Tehát a b munkát csak akkor lehet elkezdeni, ha már az a munkát befejeztük.

Kiszámítandó, hogy a teljes építkezést legkevesebb hány nap alatt lehet befejezni!

Bemenet

A `terv.be` szöveges állomány első sora két egész számot tartalmaz, az elvégzendő munkák n számát ($1 \leq n \leq 200$), és a megelőzési előírások m számát ($0 \leq m \leq 10000$).

Kimenet

A `terv.ki` szöveges állomány első sora az összes munka elvégzéséhez szükséges napok k számát tartalmazza. Ha a megelőzési előírások miatt nem lehet elvégezni az összes munkát, akkor az első és egyetlen sorba a 0 számot kell írni. A további k sor mindegyike egy napon elvégzendő munkák sorszámait tartalmazza egy-egy szóközzel elválasztva.

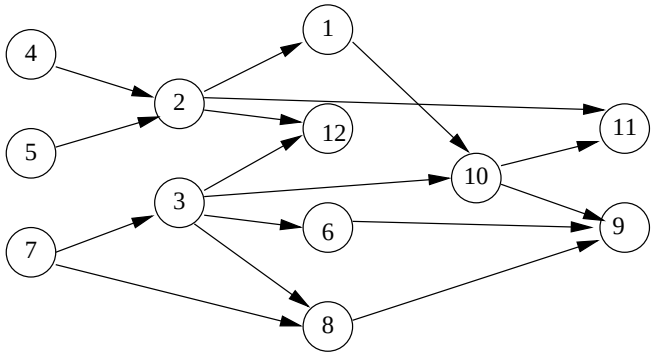
Példa bemenet és kimenet

bemenet

```
12 16
4 2
5 2
7 3
7 8
2 1
2 11
2 12
3 12
3 10
3 6
3 8
1 10
6 9
8 9
10 11
10 9
```

kimenet

```
5
4 5 7
2 3
1 12 6 8
10
11 9
```



9. ábra. A példa bemenet ábrája.

Megoldás

Elvi algoritmus.
Legyen $G = (V,E)$ az a gráf, amelynek pontjai a munkák és élei a megelőzési feltételek.

$M_1 := \{v \in V : BeFok(v) = 0\}$; {az első napi munkák }
while ($M_1 \neq \emptyset$) **and** ($V \neq \emptyset$)**do begin**

KiIr(M_1);
$V := V - M_1$;
töröljük az E élekből minden olyan $p \to q$ élet, amelyre $p \in M_1$;
$M_1 := \{v \in V : BeFok(v) = 0\}$;
end

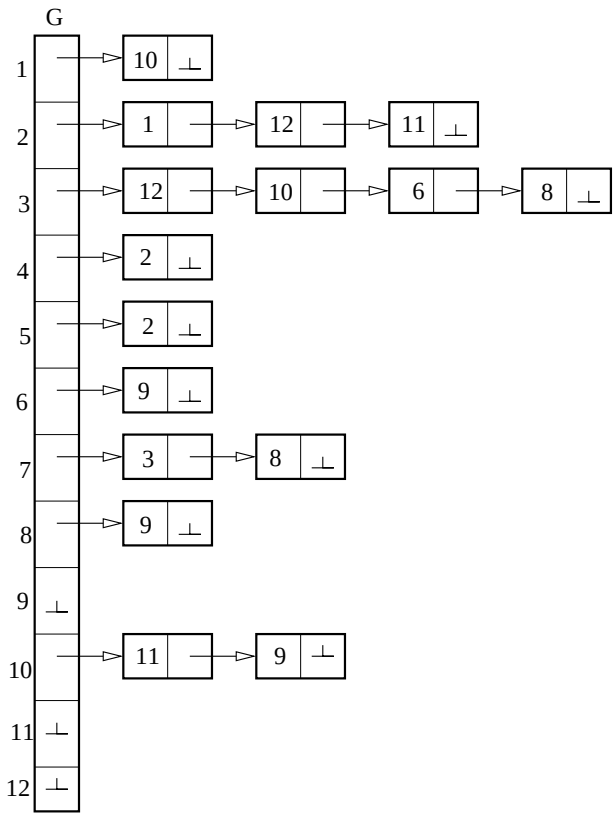
Akkor és csak akkor van megoldás, ha a megelőzési feltétel nem tartalmaz kört. Hogyan deríthető ez ki a fenti algoritmus során? Ha van kör, akkor a fenti algoritmusban az ismétlés úgy ér véget, hogy $M_1 = \emptyset$ de $V \neq \emptyset$. Fordítva is igaz, tehát ha az ismétlés úgy ér véget, hogy utána $V \neq \emptyset$, akkor van kör, mert nincs 0-befokú pont a megmaradtak között. Tehát elég megszámolni, hogy hány pont kerül bele az M_1 halmazokba.

A megelőzési előírások (gráfjának) ábrázolása.

Milyen műveleteket kell végezni?

Adott u-ra (hatékonyan) meg kell tudni adni mindazon v-ket, amelyeket u megelőz.

Jelölje ezek halmazát $Ki[u]$. Vegyünk egy G tömböt, amelynek $G[u]$ eleme olyan lánc fejére mutat, amely lánc a $Ki[u]$ halmaz elemeit tartalmazza.



10. ábra. A példa megelőzési előírásainak ábrázolása láncokban.

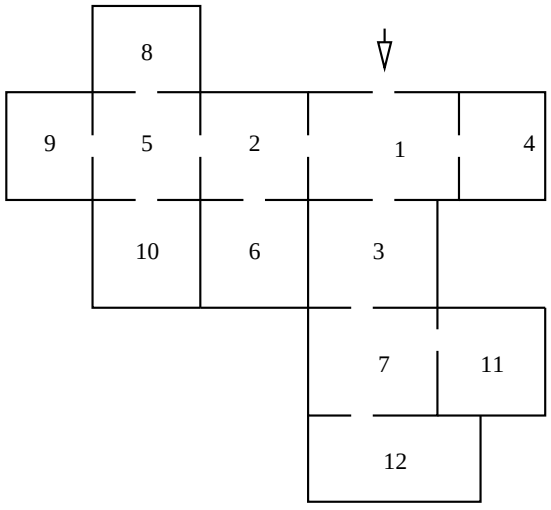
```
1  program Terv;
2  const
3    MaxN=10000;
4  type
5    Pont=1..MaxN;
6    Lanc=^Cella;
7    Cella=record oda:Pont; csat:Lanc end;
8    Graf=array[1..MaxN] of Lanc;
9  var
10   N,M:Word;
11   G:Graf;
12   Beoszt:array[1..2∧MaxN] of 0..MaxN;
```

```
13    BeFok:array[1..MaxN] of 0..MaxN;
14    Nap:Word;

15  procedure Beolvas;
16  var
17    BeF:Text;
18    i,u,v:integer;
19    Guv:Lanc;
20  begin
21    Assign(BeF, 'terv.be'); Reset(BeF);
22    ReadLn(BeF, N, M);
23
24    for u:=1 to N do begin
25      G[u]:=Nil;
26      Befok[u]:=0;
27    end;
28    for i:=1 to M do begin
29      Readln(BeF,u,v);
30      New(Guv);          {új cela létesítése}
31      Guv^.oda:=v;       {az u->v él felvétele a láncba: }
32      Guv^.csat:=G[u];{a Guv cella bekapcsolása a G[u]-lánc elejére}
33      G[u]:=Guv;
34    end{for i};
35
36    Close(BeF);
37  end{Beolvas};

38  procedure Szamit;
39  var
40    i,eleje,vege, p,q:Word;
41    El:Lanc;
42  begin
43    Nap:=0; vege:=0;
44    for i:=1 to N do {a 0-megelőzöjüek halmazának kiszámítása}
45      if Befok[i]=0 then begin
46        Inc(vege);
47        Beoszt[vege]:=i;
48      end;
49    Inc(vege);
50    Beoszt[vege]:=0;
51    eleje:=1;

52    while True do begin
53      Inc(Nap);
54      while Beoszt[eleje]<>0 do begin
55        p:=Beoszt[eleje];
56        Inc(eleje);
57        El:=G[p];
58        while El<>nil do begin      {a p->q élek feldolgozása}
59          q:=El^.oda; El:=El^.csat;
60          Dec(BeFok[q]);
61          if BeFok[q]=0 then begin{q felvétele az akt. napra}
62            Inc(vege);
63            Beoszt[vege]:=q;
64          end;
65        end{while El};
66      end{while eleje};
67      if vege=eleje then Break;
```

```
68      Inc(vege);
69      Beoszt[vege]:=0;
70      Inc(eleje);
71    end{while};
72  end{Szamit};

73  procedure KiIr;
74    var i,ind:Word;
75      KiF:Text;
76    begin
77      Assign(KiF,'terv.ki'); Rewrite(KiF);
78      if Jok<>N then Nap:=0;
79      WriteLn(KiF,Nap);
80      ind:=1;
81      for i:=1 to Nap do begin
82        repeat
83          Write(KiF,Beoszt[ind],'␣');
84          Inc(ind)
85        until Beoszt[ind]=0;
86        Inc(ind);
87        WriteLn(KiF);
88      end{for i};
89      Close(KiF);
90    end{KiIr};
91
92  begin{Program}
93    Beolvas;
94    Szamit;
95    KiIr;
96  end.
```

6.6. Feladat: Első Képtárlátogatás

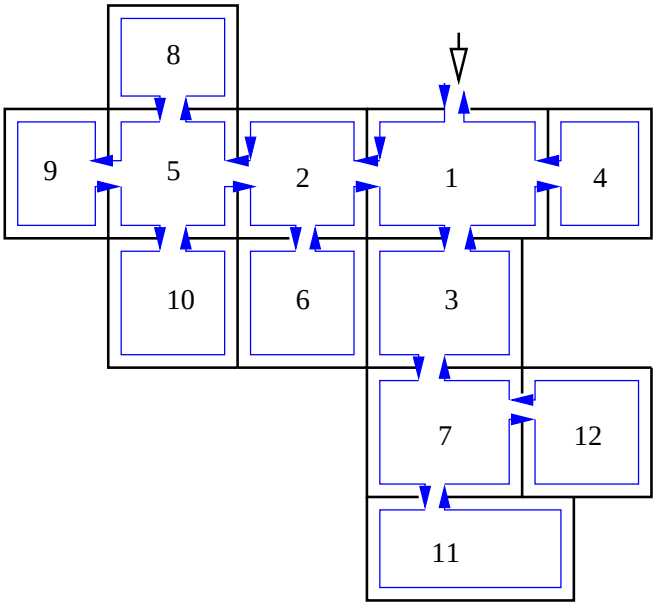
Egy sok teremből álló képtárban teszünk látogatást. Tudjuk, hogy a főbejárattól a képtár bármely termébe pontosan egy útvonalon keresztül lehet eljutni.
Adjunk olyan bejárási stratégiát, amely biztosítja, hogy minden terembe eljutunk (minden képet pontosan egyszer nézünk meg)!



11. ábra. Egy képtár alaprajza

Megoldás

A fal mellett mindig jobbra haladjunk. A lényeg, hogy adott teremből közvetlenül melyik termekbe tudok átmenni. Az alaprajz



12. ábra.

ábráján kössük össze egyenes szakasszal az olyan teremsorszám-párokat, amelyek között van ajtó (tehát közvetlenül át lehet menni egyikből a másikba).

6.7. Feladat: Második Képtárlátogatás

Ha a képtár nem teljesíti azt a feltételt, hogy bármely két terem között pontosan egy útvonal létezik (a gráfja fa), akkor nem alkalmazható a "fal mellett mindig jobbra" el. Az ábrán látható képtár esetén a 3. terembe nem jutnánk el. Fogalmazzuk meg pontosan a megoldandó feladatot.
Bemenetként adott egy képtár termeinek leírása, amely tartalmazza minden teremre, hogy abból közvetlenül melyik termekbe lehet átmenni. Feltesszük, hogy n terem esetén a termeket az $1,\ldots,n$ számokkal azonosítjuk, a bejáratot tartalmazó terem azonosítója 1.
A kimenet a termek sorszámaiból álló olyan $\langle a_1, a_2, \cdots, a_m \rangle$ számsorozat, amely megadja, hogy a bejárattól indulva milyen sorrendben kell haladnunk a képtárban, hogy minden terembe eljussunk, és a sétát a bejáratot tartalmazó 1. teremben fejezzük be.
Tehát $\langle a_1, a_2, \cdots, a_m \rangle$ számsorozatra az alábbiak teljesülnek:

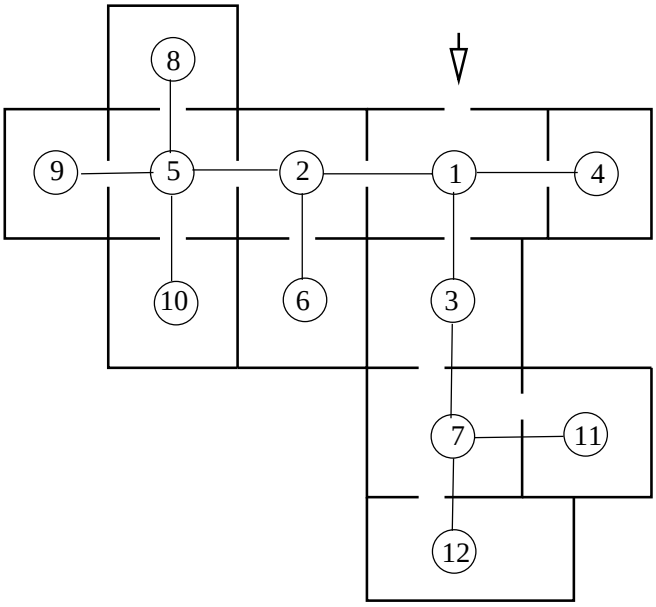
- $a_1 = 1$ és $a_m = 1$
- Minden i-re az a_i teremből van ajtó az a_{i+1} terembe ($1 \leq i < m$).
- Minden 1 és n közötti szám legalább egyszer előfordul a sorozatban.

Bemenet

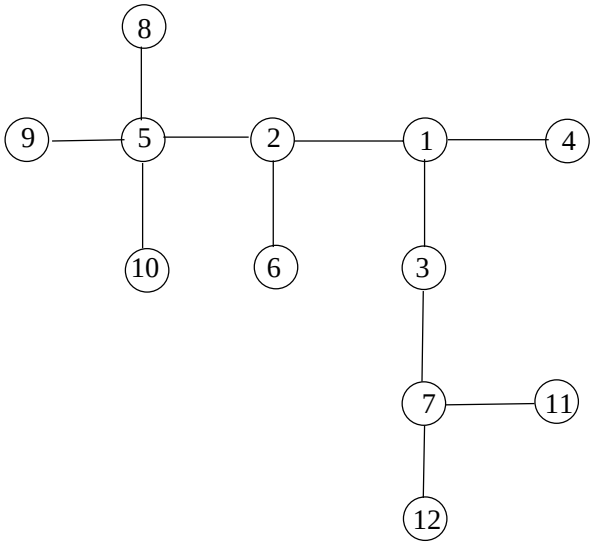
A `keptar.be` szöveges állomány első sora a termek n ($1 \leq n \leq 500$)számát tartalmazza. A következő n sor mindegyike egy terem szomszédos termeit adja meg. Az $i+1$-edik sorban azon termek sorszámai vannak felsorolva (egy-egy szóközzel elválasztva) 0-val zárva, amelyekbe nyílik ajtó az i-edik teremből.

Kimenet

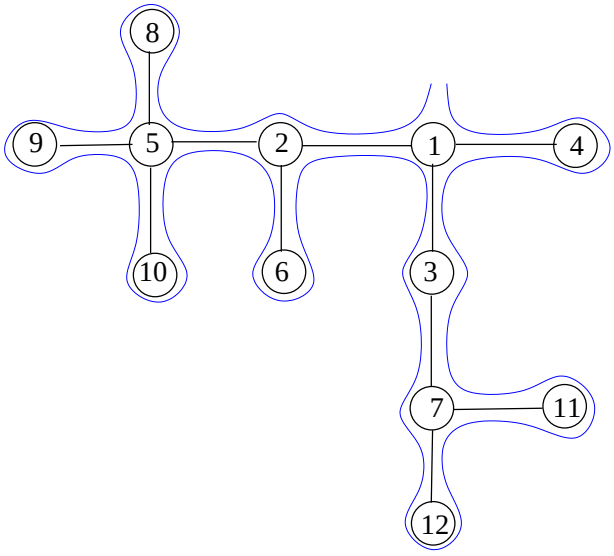
A `keptar.ki` szöveges állomány egy megoldást tartalmazzon. Több megoldás esetén bármelyik megadható.



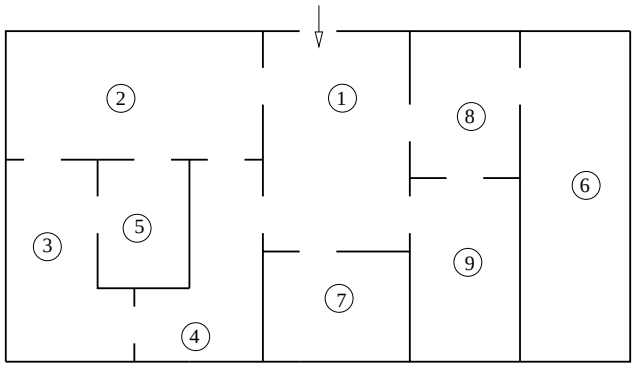
13. ábra.



14. ábra.



15. ábra. A "fal mellett mindig jobbra" elv alkalmazása.



16. ábra.

Példa bemenet és kimenet

bemenet

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

kimenet

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

Megoldás

- Mit kell tudnunk?

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

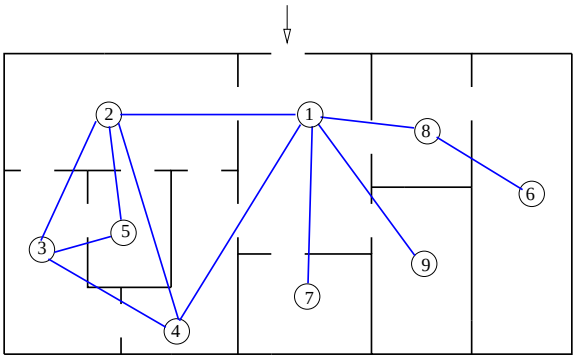
Minden p teremre $Honnan(p)$ legyen annak a teremnek a száma, amelyből először lépünk a p terembe. Kezdetben legyen minden p-re $Honnan(p) = 0$. Így ha a p teremben vagyunk, és a q terembe vezet ajtó, a $Honnan(q)$ értéke alapján el tudjuk dönteni, hogy jártunk-e q-ban.

Az algoritmus:

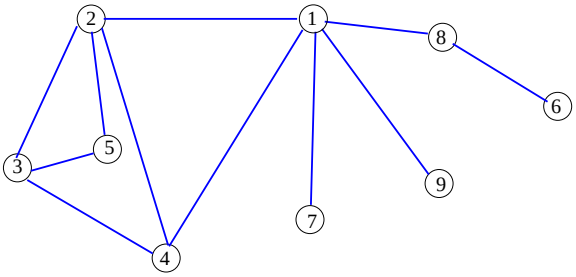
```
at = 1;                    // az aktuális terem
Honnan(1) = -1;            //az utcáról léptünk a főbejárat termébe
while(at != -1) do begin //amíg vissza nem értünk a bejárat/kijárathoz
   if (at-nek van q benemjárt szomszédos terme) then begin
      Honnan(q) = at;
      at = q;
   end else
      at = Honnan(at);
end
```

```
program Keptar;
const
  MaxN=500;
var
  N,i, at:integer;
  G:array[1..MaxN, 1..MaxN] of integer;
  Honnan,Ki,Fok:array[1..MaxN] of integer;
  KiF:Text;
procedure BeOlvas;
  var i,x,y:integer;
    BeF:Text;
  begin
    Assign(BeF, 'keptar.be'); Reset(BeF);
    ReadLn(BeF,N);
    for i:=1 to n do begin
      Fok[i]:=0;
      Read(BeF,x);
      while (x<>0) do begin
        inc(Fok[i]);
        G[i,Fok[i]]:=x;
        read(BeF, x);
      end;
    end;
    Close(BeF);
  end {Beolvas};

begin{Program}
  Beolvas;
  Assign(KiF,'keptar.ki'); Rewrite(KiF);
  for i:=1 to n do begin
    Honnan[i]:=0;
    Ki[i]:=0;
  end;
  at:=1; Honnan[1]:=-1;

  while at>0 do begin
    write(KiF, at,'␣');
    repeat
      inc(Ki[at]);
    until (Ki[at]>Fok[at])or(Honnan[G[at,Ki[at]]]=0);
    if Ki[at]<=Fok[at] then begin
      Honnan[G[at,Ki[at]]]:=at;
      at:=G[at,Ki[at]];
    end else
      at:=Honnan[at];
  end{while};
```

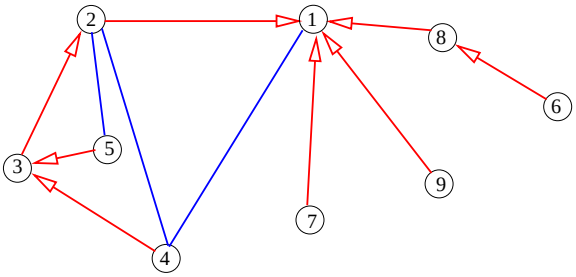
```
46
47      close(KiF);
48  end.
```



17. ábra.



18. ábra. A képtár gráfja



19. ábra. Egy teljes körséta: 1 2 3 4 3 5 3 2 1 7 1 9 1 8 6 8 1. A $p \rightarrow Honnan[p]$ élek egy feszítőfát adnak.

Rekurzív megvalósítás

```
1   program Keptar1;
2   const
3     MaxN=500;
4   var
5     N,M:Word;
6     G:array[1..MaxN, 1..MaxN] of integer;
7     Apa,Fok:array[1..MaxN] of integer;
8     i:integer;
9     KiF:Text;
```

```
10  procedure BeOlvas;
11    var i,x,y:integer;
12      BeF:Text;
13    begin
14      Assign(BeF, 'keptar.be'); Reset(BeF);
15      ReadLn(BeF,N);
16      for i:=1 to n do begin
17        Fok[i]:=0;
18        Read(BeF,x);
19        while (x<>0) do begin
20          inc(Fok[i]);
21          G[i,Fok[i]]:=x;
22          read(BeF, x);
23        end;
24      end;
25      Close(BeF);
26    end {Beolvas};

27  procedure MelyBejar(p:integer);
28  {Global: G, Fok, Apa}
29  var
30    i,q:integer;
31  begin
32    for i:=1 to Fok[p] do begin
33      q:=G[p,i];
34      if Apa[q]<0 then begin {q-ban még nem jártunk}
35        Apa[q]:=p;           {p-ből jöttünk q-ba}
36        write(KiF,' ',q);    {átlépünk q-ba}
37        MelyBejar(q);        {q-ból elérhetők bejárása}
38        write(KiF,' ',p);    {vissza kell menni p-be}
39      end;
40    end{for i};
41  end{MelyBejar};

42  begin{Program}
43    Beolvas;
44    Assign(KiF,'keptar1.ki'); Rewrite(KiF);
45    for i:=1 to n do begin
46      Apa[i]:=-1;
47    end;
48    Apa[1]:=0;
49    write(KiF,1);
50    MelyBejar(1);
51    close(KiF);
52  end.
```

6.8. Feladat: Iskolahálózat (IOI'96)

Néhány iskola számítógépes hálózatba van kötve. Az iskolák a szabadon terjeszthető programok elosztására megállapodást kötöttek: minden iskola egy listát vezet azokról az iskolákról ("szomszédairól"), amelyek számára a programokat továbbküldi. Megjegyzés: ha **B** iskola szerepel az **A** iskola terjesztési listáján, akkor **A** nem feltétlenül szerepel **B** terjesztési listáján.

Írjon programot, amely meghatározza azt a legkisebb számot, ahány iskolához egy új programot el kell juttatni ahhoz, hogy - a megállapodás alapján, a hálózaton keresztül terjesztve - végül minden iskolába eljusson.

Bemenet

Az `INPUT.TXT` állomány első sora egy egész számot, a hálózatba kapcsolt iskolák n számát tartalmazza ($2 \leq n \leq 200$). Az iskolákat az első n pozitív egész számmal azonosítjuk. A következő n sor mindegyike egy-egy terjesztési listát ír le. Az $i+1$-edik

sor az *i*-edik iskola terjesztési listáján szereplő iskolák azonosítóit tartalmazza. Minden lista végén egy 0 szám áll. Az üres lista csak egy 0 számból áll.

Kimenet

A program az eredményt, amely két sorból áll, az `OUTPUT.TXT` nevű fájlba kell írja. Az első sor egy pozitív egész számot, azt a legkisebb m számot, ahány iskolához egy új programot el kell juttatni ahhoz, hogy - a megállapodás alapján, a hálózaton keresztül terjesztve - végül minden iskolába eljusson. A második sorban kell megadni az m kiválasztott iskolát, egy-egy szóközzel elválasztva.

Példa bemenet és kimenet

bemenet	**kimenet**
13 2 0 3 0 1 4 9 0 1 0 10 6 0 7 0 5 8 0 13 0 10 0 11 12 0 9 0 13 0 12 0	2 2 5

Megoldás

Jelölje $G = (V,E)$; $V = \{1,\ldots,n\}$ az iskolahálózat gráfját.
Legyen $D \subseteq V$ egy megoldáshalmaz, tehát

$$(\forall q \in V)(\exists p \in D)(p \rightsquigarrow q)$$

A D halmazt lépésenként építhetjük:
Adott $p \in V$ pontra jelölje $Eler(p)$ a p pontból elérhető pontok halmazát:

$$Eler(p) = \{q : p \rightsquigarrow q\}$$

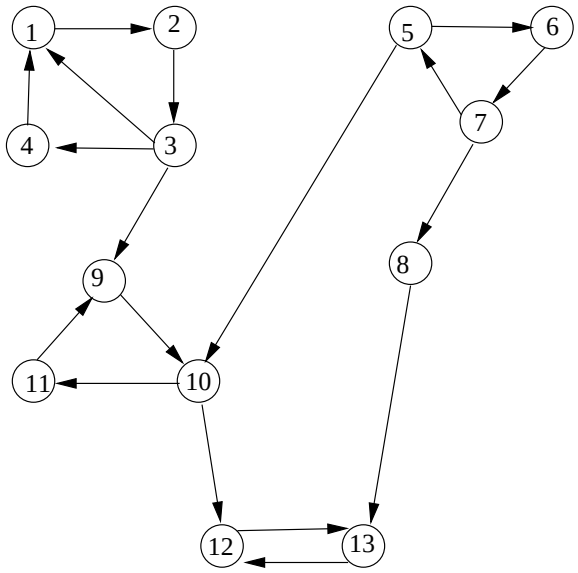
Terjesszük ki ezt halmazokra:

$$Eler(D) = \bigcup_{p \in D} Eler(p)$$

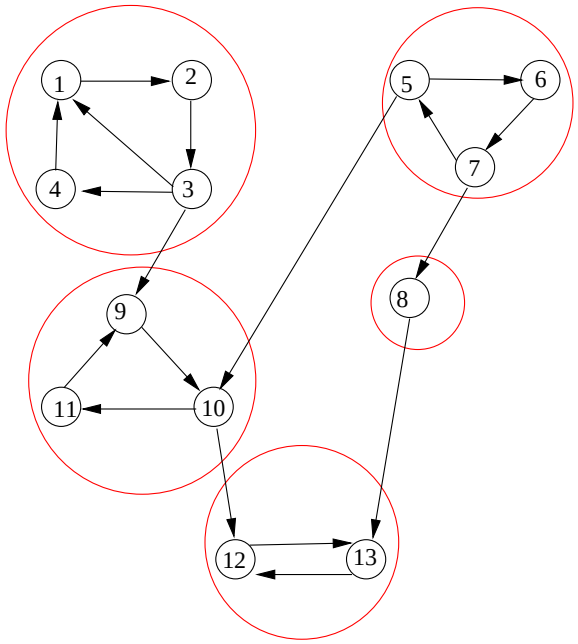
begin
  $D := \emptyset$;
  $DbolElert := \emptyset$;
  for $p \in V$ **do**
    if $p \notin DbolElert$ **then begin**
      $D := D - Eler(p) \bigcup \{p\}$
      $DbolElert := DbolElert \bigcup Eler(p)$
    end
end

Az $Eler(p)$ halmaz elemei mélységi bejárással kiszámíthatók.
Az algoritmus futási ideje legrosszabb esetben $O(n^3)$.
Az algoritmus legrosszabb esete, ha a bemenetben p szomszédai: $\{1,2,\ldots,p-1\}$. Ha n 1000 is lehet, akkor ez az algoritmus biztosan nem elég gyors megoldás.

Gyorsítási ötlet: elsőször rakjuk a pontokat olyan sorrendbe, hogy ha $p \rightsquigarrow q$, de nincs $q \rightsquigarrow p$, akkor p előbb álljon a sorozatban, mint q.

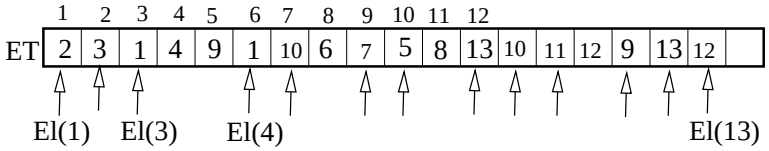


20. ábra.



21. ábra.

Egy ilyen kívánt sorrendet egyetlen mélységi bejárással előállíthatunk:
a p pontra hívott (rekurzív) mélységi bejárásban, miután p-ből kiinduló összes élet bejártunk, rakjuk p-t a sorozat elejére. (Tehát a kívánt sorozatban a pontok elhagyási idő szerint csökkenő sorrendben lesznek.)
Az így módosított algoritmus két mélységi bejárást végez, tehát futási ideje $O(E)$



22. ábra. A hálózat ábrázolása. A p iskola szomszédjainak listája a ET tömb $El[p],\ldots,El[p+1]-1$ indexű elemeiben vannak.

```
program Halozat;
const
  maxN=2000;    {a pontok maximális száma}
  maxE=100000; {az élek maximális száma}
type
  Pont=1..maxN;
  Paletta=(Feher, Szurke, Fekete);
var
  ET:array[1..maxE] of 0..MaxN;
  El:array[1..MaxN+1] of longint;
  N:Longint;                  {a pontok száma}
  Szin:array[1..maxN] of Paletta;
  Ver:array[1..maxN] of Pont; {verem}
  Vm:Longint;                 {verem-mutató}
  D:array[1..MaxN] of Pont;
  DSzam:Integer;
```

```
procedure BeOlvas;
  var i,p,q:Word;
    BeF:Text;
  begin
    Assign(BeF, 'halozat.be'); Reset(BeF);
    ReadLn(BeF,N);
    El[1]:=1; i:=1;
    for p:=1 to N do begin
      Read(BeF,q);
      while q<>0 do begin
        ET[i]:=q;
        inc(i);
        Read(BeF,q);
      end;
      readln(BeF);
      El[p+1]:=i;
    end;
    Close(BeF);
  end {Beolvas};
```

```
Procedure MelyBejar(u:Pont; elso:boolean);
{Global: G, Szin, Ver}
Var
  i,v:longint;
begin
  Szin[u]:=Szurke;
```