Neumann János Tehetséggondozó Program
Gráfalgoritmusok I.

Horváth Gyula
horvath@inf.elte.hu

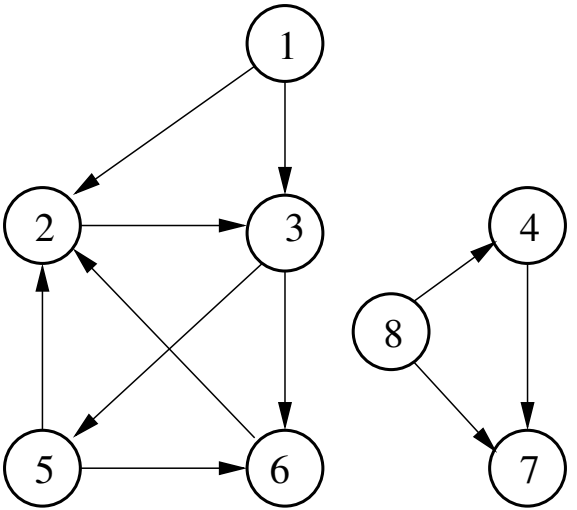
A mindennapi életben számos olyan algoritmikus problémával találkozhatunk, amelynek a megoldása megoldaható gráf modellben.
Az ilyen problémák esetén az adatok jellemző tulajdonsága, hogy az adathalmaz elemei között kapcsolatok vannak. Ezek a kapcsolatok fejezhetők ki a gráf matematikai fogalmával. A matematikai gráf fogalom azonban csak alap modell a problémát megoldó algoritmushoz. Lényeges eltérés a matematikai gráf fogalomtól, hogy az algoritmushoz a gráfot alkalmas adatszerkezetben tárolni kell, és műveleteket kell tudni végezni (hatékonyan) a gráfokon.

1. Definíciók

1.1. definíció. ***Irányított gráf*** $G = (V, E)$

V: *a gráf pontjainak halmaza*

E *a gráf éleinek halmaza, rendezett* (a, b) *párok halmaza;* $E \subseteq V \times V = \{(a, b) : a \in V, b \in V\}$.



1. ábra. Irányított gráf.

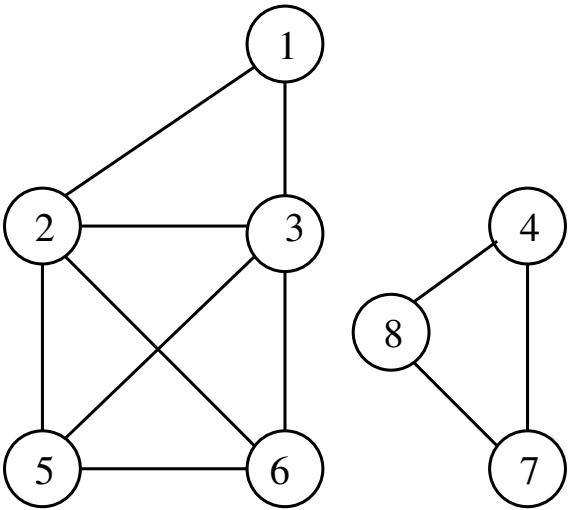
$V = \{1, \ldots, 8\}$,

$E = \{(1,2), (1,3), (2,3), (3,5), (3,6), (4,7), (5,2), (5,6), (6,2), (8,4), (8,7)\}$

1.2. definíció. ***Irányítatlan gráf*** $G = (V, E)$

V*: a gráf pontjainak halmaza*

E*: az élek halmaza, rendezetlen* $\{a, b\}, a, b \in V$ *párok halmaza.*

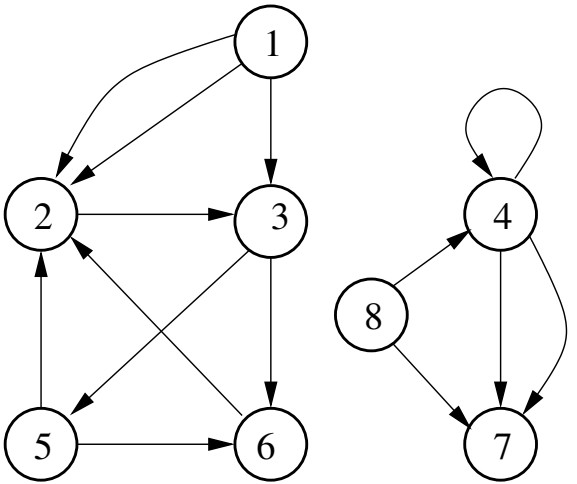


2. ábra. Irányítatlan gráf.

$V = \{1, \ldots, 8\}$,

$E = \{\{1,2\},\{1,3\},\{2,3\},\{3,5\},\{3,6\},\{4,7\},\{5,2\},\{5,6\},\{6,2\},\{8,4\},\{8,7\}\}$

1.3. definíció. ***Irányított multigráf*** $G = (V, E, Ind, Érk)$
$Ind, Érk : E \to V$ *az élek végpontjait megadó függvények*
$Ind(e)$ *az* e *él induló,* $Érk(e)$ *az érkező pontja.*

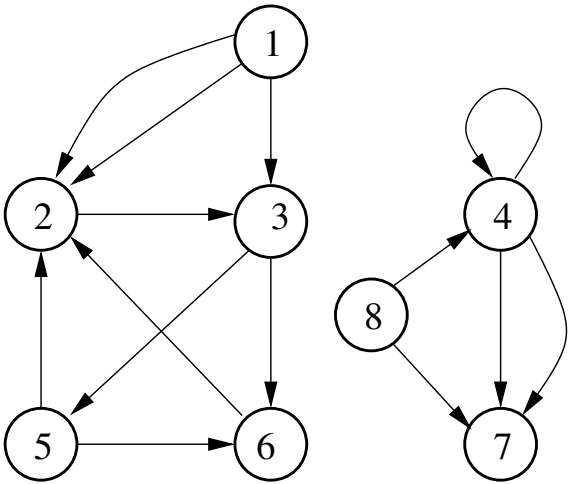


3. ábra. Irányított multigráf

1.4. definíció. ***Irányított multigráf*** $G = (V, E)$

V *: a gráf pontjainak halmaza*

E *a gráf éleinek halmaza, rendezett* (a,b) *párok multihalmaza;* $E \subseteq V \times V = \{(a,b) : a \in V, b \in V\}$.

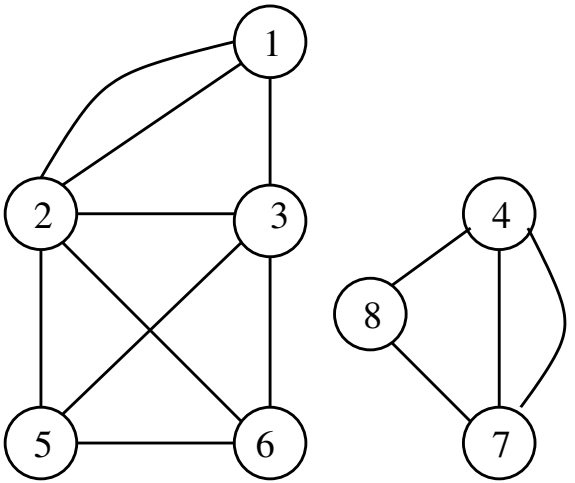


4. ábra. Irányított multigráf.

$V = \{1, \ldots, 8\}$,

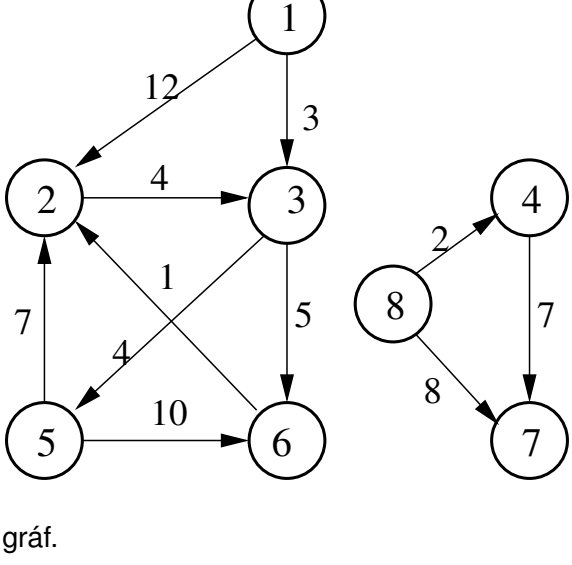
$E = \{(1,2),(1,2),(1,3),(2,3),(3,5),(3,6),(4,7),(4,7),(5,2),(5,6),(6,2),(8,4),(8,7)(4,4)\}$

1.5. definíció. ***Irányítatlan multigráf*** $G = (V, E, Ind, Érk)$
$Ind, Érk : E \to V$ *az élek végpontjait megadó függvények*
$Ind(e)$ *az* e *él induló,* $Érk(e)$ *az érkező pontja.*



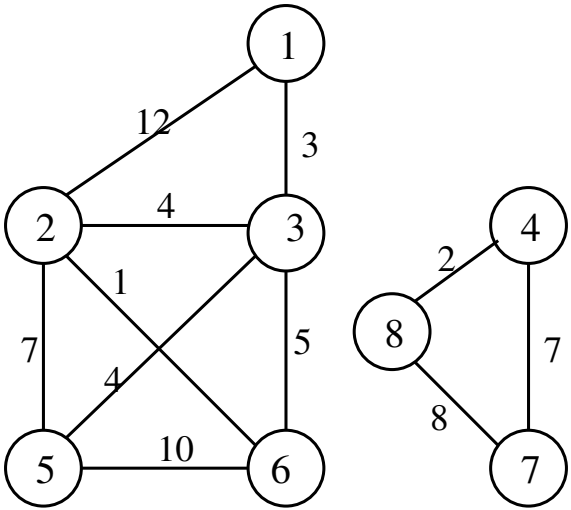
5. ábra. Irányítatlan multigráf.
$V = \{1, \ldots, 8\}$,
$E = \{\{1,2\},\{1,2\},\{1,3\},\{2,3\},\{3,5\},\{3,6\},\{4,7\},\{4,7\},\{5,2\},\{5,6\},\{6,2\},\{8,4\},\{8,7\}\}$

1.6. definíció. ***Címkézett (súlyozott) irányított gráf*** $G = (V, E, C)$
(V, E) *(irányított/irányítatlan) gráf.*
$C : E \to$ *Címke súlyfüggvény*
Másképpen:
$E \subseteq V \times V \times C = \{(a, b, c) : a \in V, b \in V, c \in C\}$.



6. ábra. Súlyozott irányított gráf.
$V = \{1, \ldots, 8\}$,
$E = \{(1,2,12), (1,3,3), (2,3,4), (3,5,4), (3,6,5), (4,7,7), (5,2,7), (5,6,10), (6,2,1), (8,4,2), (8,7,8$

1.7. definíció. ***Címkézett (súlyozott) irányítatlan gráf*** $G = (V, E, C)$
(V, E) *(irányítatlan) gráf*
$C : E \to$ *Címke súly függvény*



7. ábra. Súlyozott irányítatalan gráf.

$V = \{1, \ldots, 8\}$,
$E = \{(\{1,2\}, 12), (\{1,3\}, 3), (\{2,3\}, 4), (\{3,5\}, 4), (\{3,6\}, 5), (\{4,7\}, 7), (\{5,2\}, 7), (\{5,6\}, 10), (\{6,$

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

$KiEl(G,p) = \{q \in V : (p,q) \in E\}$: a kipontok halmaza, azaz minden olyan q pont, amelyre van $p \to q$ él a G gráfban.

$BeEl(G,p) = \{q \in V : (q,p) \in E\}$: a bepontok halmaza, azaz minden olyan q pont, amelyre van $q \to p$ él a G gráfban.

$KiFok(G,p) = |KiEl(G,p)|$: a kipontok száma

$BeFok(G,p) = |BeEl(G,p)|$: a bepontok száma

1.1. Műveletek gráfokon

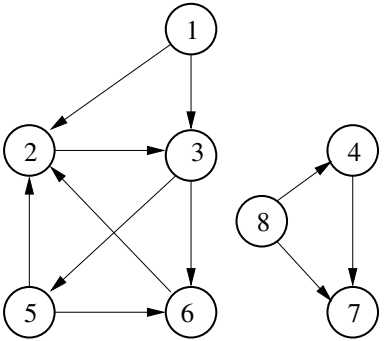
A továbbiakban feltételezzük, hogy a gráf pontjait természetes számokkal azonosítjuk, pontosabban $V = \{1, \ldots, n\}$
Milyen műveleteket akarunk gráfokon végezni?

- Pontokszáma
- Elekszáma
- KiFok(p)
- BeFok(p)
- PontBővít(p)
- PontTöröl(p)
- ÉlBővít(p, q)
- ÉlTöröl(p, q)
- VanÉl(p, q)
- for q in KiÉl(p) do M(p,q) //p-ből kinduló élek bejárása

2. 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.



8. ábra. Példa gráf

Minden gráf megadható olyan hozzárendeléssel (függvénnyel), amely minden p ponthoz megadja a $KiEl(p)$ halmazt. Multigráf esetén $KiEl(p)$ multihalmaz.

$$
\begin{aligned}
1 &\mapsto \{2,3\} \\
2 &\mapsto \{3\} \\
3 &\mapsto \{5,6\} \\
4 &\mapsto \{7\} \\
5 &\mapsto \{2,6\} \\
6 &\mapsto \{2\} \\
7 &\mapsto \{\} \\
8 &\mapsto \{4,7\}
\end{aligned}
$$

Tehát egy $G = (V, E)$ gráf, ahol a gráf pontjainak halmaza $V = \{1, \ldots, n\}$ megadható olyan tömb típussal, amelynek elemei halmazok.

Pascal:

```
1 const
2   maxP=1000;   //{a pontok max. száma}
3 type
4   Graf=array[1..maxP] of Halmaz;
```

C/C++:

```
1 #define maxP 1000
2 typedef Halmaz Graf[maxP];
```

2.1. Szöveges ábrázolás

Élek felsorolásával: Az állomány első sora két egész számot tartalmaz, a pontok n számát és az élek m számát. A további n sor mindegyike egy $a\ b$ számpárt tartalmaz ($1 \leq a, b \leq n$) , a gráf egy élét.

Kipontok soronkénti felsorolásával: Az állomány első sora egy egész számot tartalmaz, a pontok n számát. A további n sor tartalmazza az éleket. Az $p+1$-edik sorban a $KiEl(p)$ halmaz elemeit vannak felsorolva, a felsorolást a 0 zárja.

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

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

2.2. Éltömb

ind	1	1	2	3	3	4	5	5	6	8	8
érk	2	3	3	5	6	7	2	6	2	4	7

9. ábra. A példa gráf éltömb ábrázolása

Pascal:

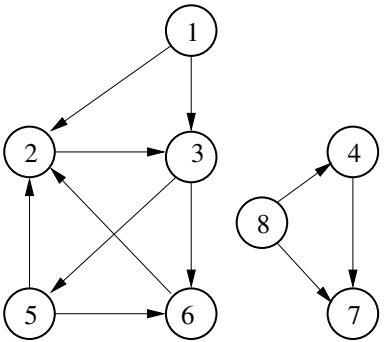
```
1  type
2    Cimketip=integer;// {a súlyok típusa}
3    Graf=array[1..maxP] of record ind,erk: 1..maxP end;
4    SGraf=array[1..maxP] of record ind,erk: 1..maxP, suly: Cimketip end;
```

C++:

```
1  typedef int Cimketip; //{ a súlyok típusa}
2  typedef struct El{
3      int ind,erk;
4  }El;
5  typedef El Graf[maxP];
6  typedef struct SEl{
7      int ind,erk;
8      Cimketip suly;
9  }SEl;
10 typedef SEl SGraf[maxP];
```

Ez az ábrázolás nem támogatja a leggyakoribb műveletek hatékony megvalósítását, ezért csak a teljesség kedvéért említjük meg. Előfordulhat, hogy közbülső ábrázolásként használjuk; először a beolvasásakor ilyen ábrázolást állítunk elő, és ebből építünk fel hatékonyabb adatszerkezetet.

2.3. Kapcsolatmátrix (szomszédsági mátrix)



10. ábra. Példa gráf

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

11. ábra. Kapcsolatmátrix

Pascal:

```
1 type
2   Graf=array[1..maxP, 1..maxP] of boolean;
3   SGraf=array[1..maxP, 1..maxP] of integer;
4 var
5   G:Graf;
```

C++:

```
1 typedef bool Graf[maxP][maxP];
2 typedef int SGraf[maxP][maxP];
3 Graf G;
```

A (p,q) pontpár akkor és csak akkor éle a gráfnak, ha $G[p,q] = true$. Tehát a vanél művelet hatékonyan megvalósítható (konstans idejű).
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 $G[p,q]$.
A $KiEl(p)$ kipontok bejárása nem hatékony, nem az elemszámmal arányos.
Pascal:

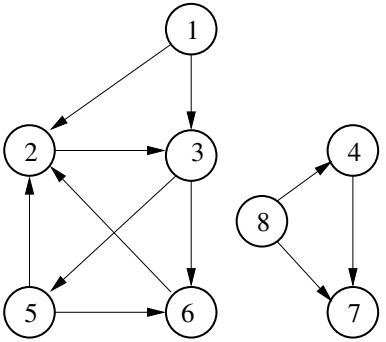
```
1   for q=1 to n do
2     if G[p,q] then M(p,q);
```

C++:

```
1   for (int q=1;q<=n;q++)
2     if (G[p][q]) M(p,q);
```

A tárigény n pontú gráf esetén n^2, ezért nagyméretű, (de kevés élet tartalmazó) gráf esetén nem alkalmazható.

2.4. Élhalmaz-tömb



12. ábra. Példa gráf

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

13. ábra. Élhalmaz-tömb

Pascal:

```
1 type
2   Graf=array[1..maxP, 1..maxP] of 0..maxP;
3 var
4   G:Graf;
5   KiFok:array[1..maxP] of integer;
```

C++:

```
1 typedef int Graf[maxP][maxP];
2 Graf G;
3 int KiFok[maxP+1];
```

A tárigény, csakúgy, mint a szomszédsági mátrixnál, n pontú gráf esetén n^2, ezért nagyméretű, (de kevés élet tartalmazó) gráf esetén nem alkalmazható.

A $KiEl(p)$ kipontok bejárása hatékony, az elemszámmal arányos.

Pascal:

```
1   for i=1 to KiFok[p] do
2     M(p,G[p,i]);
```

C++:

```
1   for (int i=1;i<=KiFok[p];i++)
2     M(p,G[p][i]);
```

A vanél művelet persze nem konstans idejű lesz:

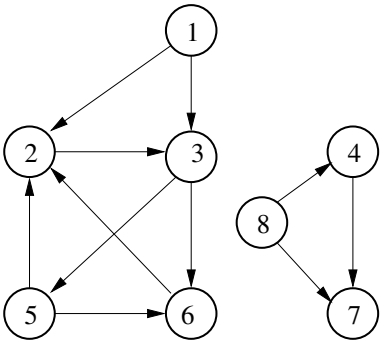
Pascal:

```
1   i:=1;
2   while (i<=KiFok[p] and G[p,i]<>q) do i:=i+1;
3   vanel:=i<=KiFok[p];
```

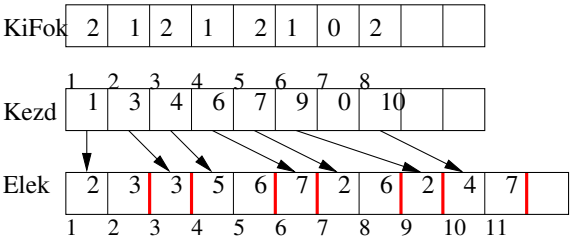
C++:

```
1   i=1;
2   while (i<=KiFok[p] && G[p][i]!=q) i++;
3   vanel=i<=KiFok[p];
```

2.5. Élvégpontok felsorolása



14. ábra. Példa gráf



15. ábra. Élhalmaz-lánc

Pascal:

```
1  const
2        maxP=1000;
3        maxE=100000;
4  var
5        KiFok:array[1..maxP] of integer;
6        Kezd:array[1..maxP] of integer;
7        Elek:array[1..maxE] of integer;
```

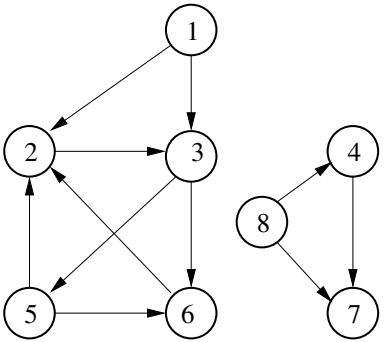
$KiEl(G,p)$ **bejárása Pascal:**

```
1   tol:=Kezd[p]-1
2   for i:=1 to KiFok[p] do begin
3     q:=Elek[tol+i];
4     //M(p,q) //{a p->q él feldolgozása}
5   end;
```

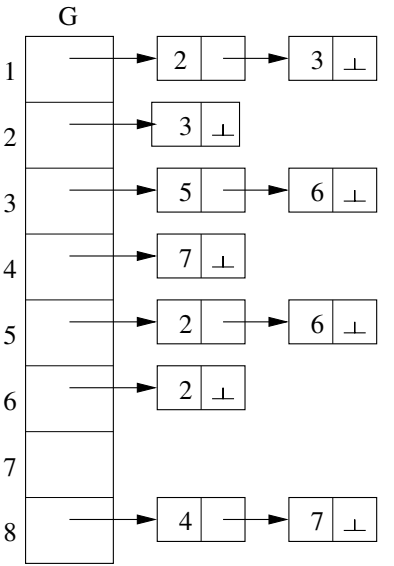
C++:

```
1  #define  maxP 10001
2  #define maxE 100000
3  int Kezd[maxP];
4  int KiFok[maxP];
5  int Elek[maxE];
6  //KiEl(G,p) bejárása
7    int tol=Kezd[p];
8    for (int i=0; i<KiFok[p];i++){
9      int q=Elek[tol+i];
10     //M(); //a p->q él feldolgozása
11   }
```

2.6. Élhalmaz-lánc



16. ábra. Példa gráf



17. ábra. Élhalmaz-lánc

2.6.1. Statikus élhalmaz-lánc

Pascal:

```
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:longint; csat:Lanc end;
7    Graf=array[1..maxP] of Lanc;
8  var
9    G:Graf;
10   Elek:array[1..maxE] of Cella;
11   n:longint; //a pontok száma
12   m:longint; //az élek száma
```

```
13  //Beolvasás és az adatszerkezet felépítése
14    szabad,i,p,q,pql:longint;
15    readln(n,m);
16    szabad:=1;
17    for i:=1 to n do G[i]:=0;
18    for i:=1 to m do begin
19      readln(p,q);              //{egy él beolvasása}
20      Elek[szabad].pont:=q;     //{becsatolás a láncba}
21      Elek[szabad].csat:=G[p];
22      G[p]:=szabad;
23      inc(szabad);
24    end;
```

$KiEl(G,p)$ bejárása

```
1   pql:=G[p];
2   while pql<>0 do begin
3     q:=Elek[pql].pont;
4     //M(p,q); //{az él feldolgozása}
5     pql:=Elek[pql].csat;
6   end;
```

C++:

```
1  typedef int Lanc;
2  typedef struct {int pont; Lanc csat;} Cella;
3  typedef Lanc Graf[maxP];
4  typedef Cella Elek[maxE];
5  Graf G; Elek El;
6  //Beolvasás és az adatszerkezet felépítése
7    int p,q,szabad=0,pql;
8    cin>>n>>m;
9    for (int i=1;i<=n;i++) G[i]=0;
10   for (int i=1;i<=m;i++){
11     cin>>p>>q;
12     Elek[szabad].pont=q;    //{becsatolás a láncba}
13     Elek[szabad].csat=G[p];
14     G[p]=szabad++;
15   }
```

$KiEl(G,p)$ bejárása

```
1    Lanc pql=G[p];
2    while (pql!=0){
3       q=El[pql].pont;
4       //M(p,q); //{a p->q él feldolgozása}
5       pql=El[pql].csat;
6    }
```

2.6.2. Dinamikus élhalmaz-lánc

Pascal:

```
const
  maxP=1000;
type
  Lanc=^Cella;
  Cella=record pont.integer; csat:Lanc end;
  Graf=array[1..naxP] of Lanc;
var
  G:Graf;
  n,m:longint;
  pql:Lanc;
  readln(n,m);
  for i:=1 to n do G[i]:=Nil;
  for i:=1 to m do begin
    readln(p,q);    //{egy él beolvasása}
    new(pql);       //{új cella létesítése}
    pql^.pont:=q;
    pql^.csat:=G[p];//{az új cella becsatolás a láncba}
    G[p]:=pql;
  end;
```

$KiEl(G,p)$ **bejárása**

```
1   pql:=G[p];
2   while pql<>Nil do begin
3     q:=pql^.pont;
4     //M(p,q); //{a p->q él feldolgozása}
5     pql:=pql^.csat;
6   end;
```

2.7. 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 $KiEl(G,p)$ halmaz elemeit.
Például, ha egy $n \times m$-es négyzetrácsos hálózat mezőin lépkedhetünk, egy lépésben a 4 szomszédos mezőre; fel, le, jobbra és balra. Tekincsük azt a $G = (V,E)$ gráfot, amelynek pontjai a hálózat mezői, azaz $V = \{(x,y) : 1 \leq x \leq n, 1 \leq y \leq m\}$, tehát a pontokat a koordinátáikkal azonosítjuk. Ekkor

$$(x,y) \rightarrow (u,v)$$

akkor és csak akkor él a gráfban, ha $|x-u| + |y-v| = 1$

3. 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

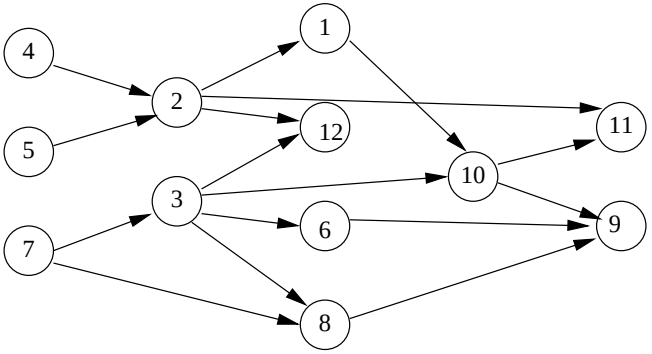
A standard bemenet első sora két egész számot tartalmaz, az elvégzendő munkák n számát ($1 \leq n \leq 20000$), és a megelőzési előírások m számát ($0 \leq m \leq 100000$).

Kimenet

A standard kimenet 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 ás egyetlen sorba a 0 számot kell írni. A további k sor mindegyike egy napon elvégzendő munkák sorszámait tartalmazza egy-egy szóközzel elválasztva. Több megoldás esetén bármelyik megadható.

Példa bemenet és kimenet

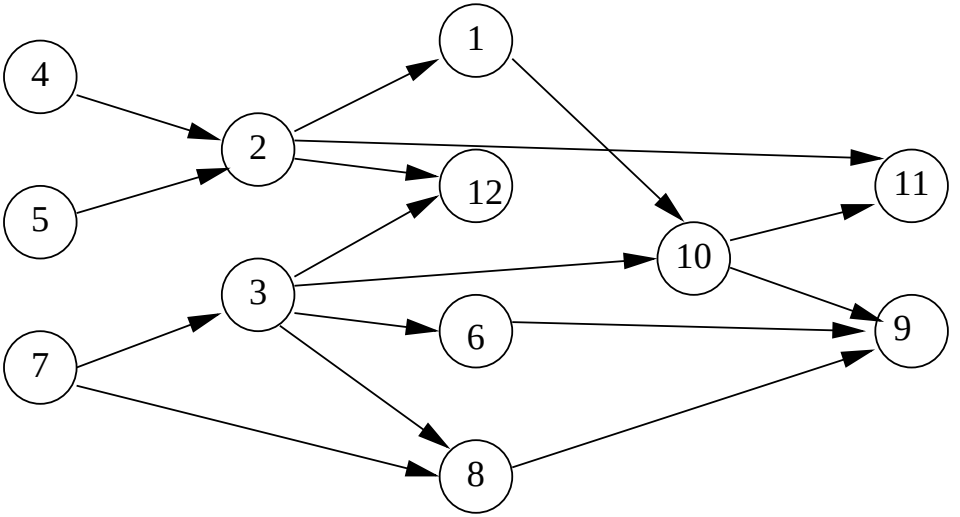
bemenet	kimenet
12 16 4 2 5 2 7 3 7 8 2 1 2 11 2 12 3 12 3 10 3 6 3 8 1 10 6 9 8 9 10 11 10 9	5 4 5 7 2 3 1 12 6 8 10 11 9



18. ábra. A példa bemenet ábrája. A gráf pontjai az elvégzendő munkák, $p \to q$ akkor és csak akkor él a gráfban, ha a p munkának meg kell előznie a q munkát.

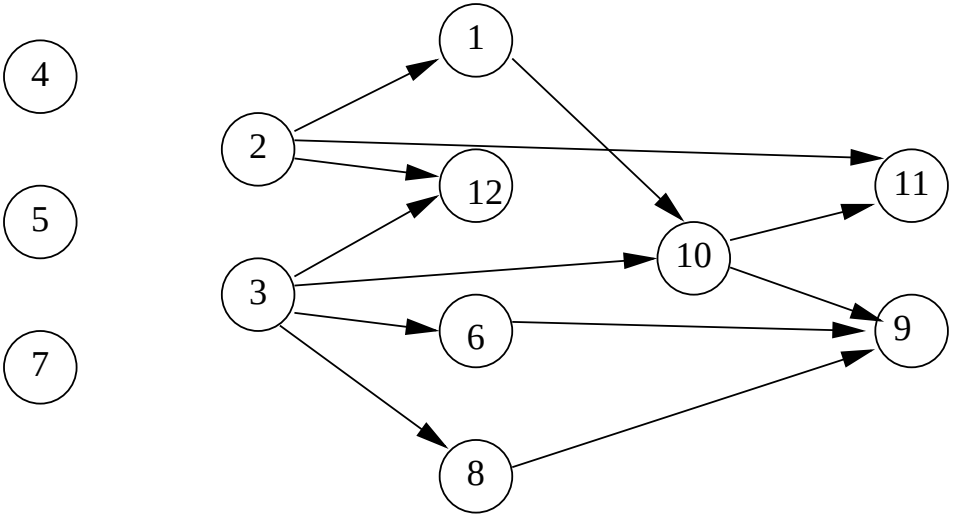
Megoldás

Megoldás



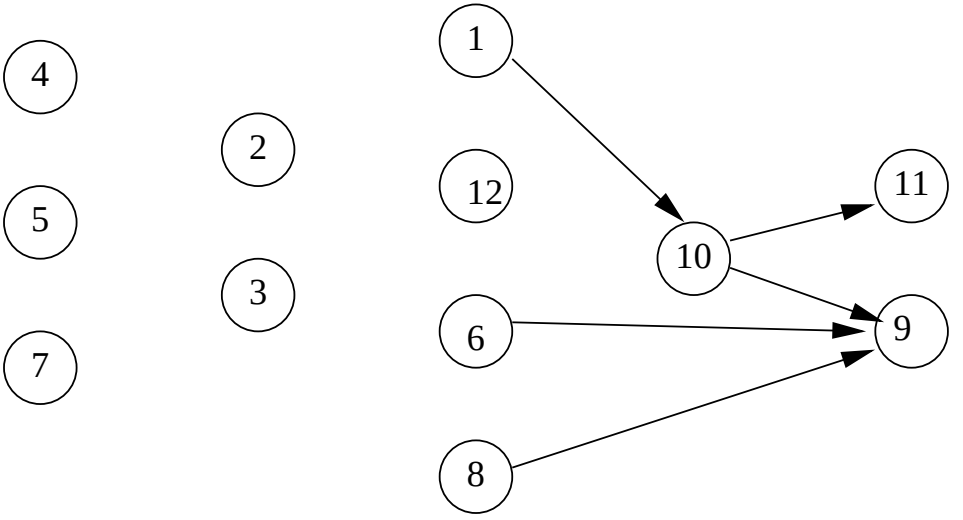
1. nap

Megoldás

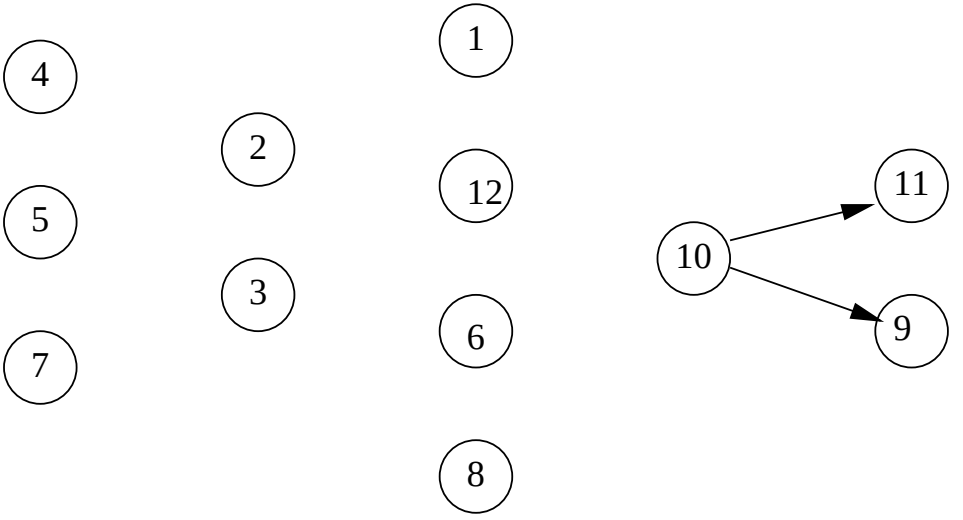


1. nap 2. nap

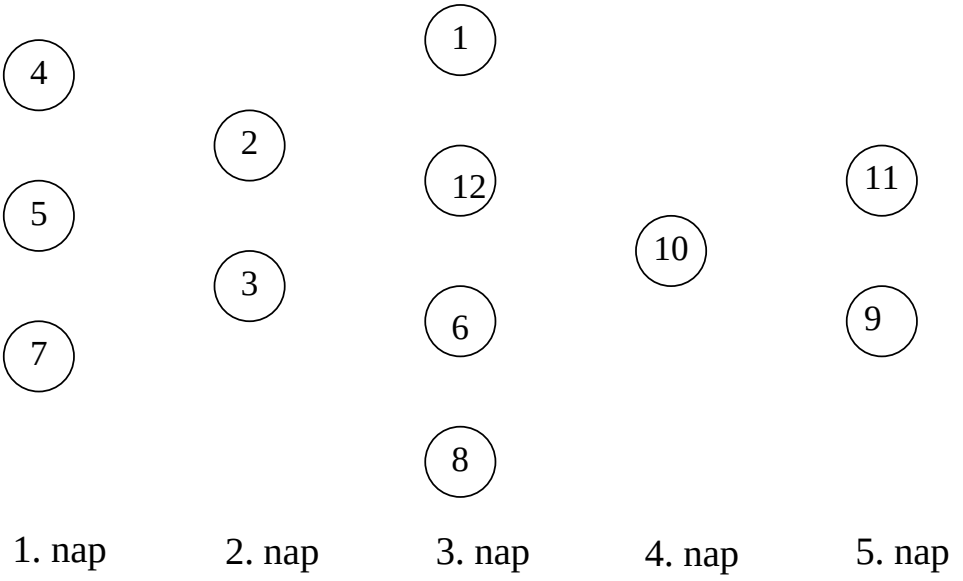
Megoldás



Megoldás



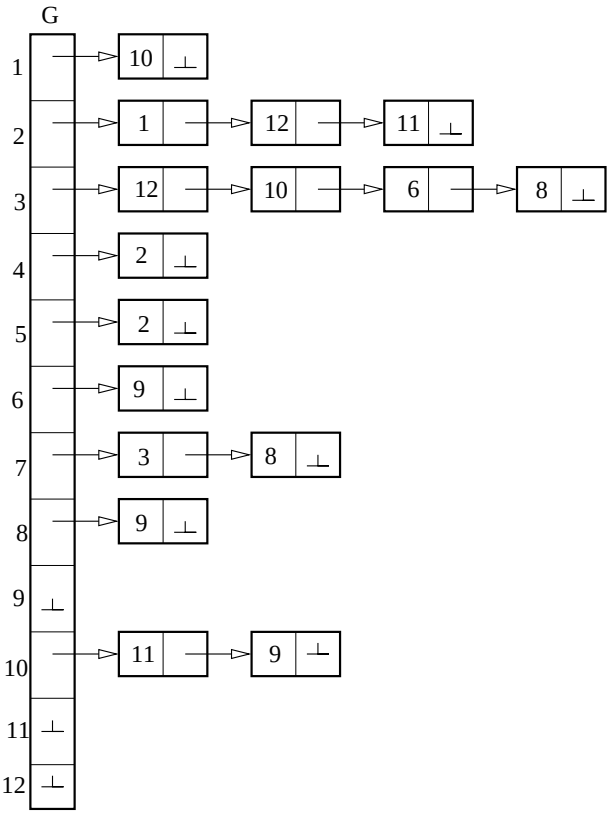
Megoldás



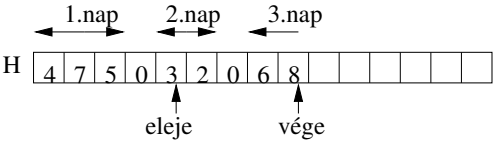
Elvi algoritmus.
Legyen $G = (V,E)$ az a gráf, amelynek pontjai a munkák és élei a megelőzési feltételek.

$H(1) := \{v \in V : BeFok(v) = 0\}$; {az első napi munkák }
$nap = 1$
while $(H(nap) \neq \emptyset)$ **and** $(V \neq \emptyset$)**do begin**
  KiIr($H(nap)$);
  $V := V - H(nap)$;
  töröljük az E élekből minden olyan $p \to q$ élet, amelyre $p \in H(nap)$;
  $nap = nap + 1$;
  $H(nap) := \{v \in V : BeFok(v) = 0\}$;
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 $H(nap) = \emptyset$ 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 H 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 $KiEl(u)$. Vegyünk egy G tömböt, amelynek $G[u]$ eleme olyan lánc fejére mutat, amely lánc a $KiEl(u)$ halmaz elemeit tartalmazza.



19. ábra. A példa megelőzési előírásainak ábrázolása láncokban.



20. ábra. A H halmazok ábrázolása

```
1  program Terv;
2  const
3    maxP=20000;    //{a pontok max száma}
4    maxE=100000;   //{az élek max. száma}
5  type
6    Pont=1..maxP;
7    Lanc=longint;
8    Cella=record pont:longint; csat:Lanc end;
9    Graf=array[1..maxP] of Lanc;
10 var
11   N,M:longint;
12   G:Graf;
13   Elek:array[1..maxE] of Cella;
14   H:array[1..2*maxP] of 0..maxP;
15   BeFok:array[1..maxP] of 0..maxP;
16   Nap:longint;
```

```
17 procedure Beolvas;
18 //Globalis: G,Elek,Befok
19 var
20   i,p,q:longint;
21   szabad:Lanc;
22 begin
23   ReadLn(N, M);
24   for p:=1 to N do begin
25     G[p]:=0;
26     Befok[p]:=0;
27   end;
28   szabad:=1;
29   for i:=1 to M do begin
30     Readln(p,q);
31   readln(p,q);                  //{egy él beolvasása}
32     Elek[szabad].pont:=q;       //{becsatolás a láncba}
33     Elek[szabad].csat:=G[p];
34     G[p]:=szabad;
35     inc(szabad);
36   end{for i};
37 end{Beolvas};
```

```
38 procedure Kilr;
39   var i,ind:longint;
40   begin
41     WriteLn(Nap);
42     ind:=1;
43     for i:=1 to Nap do begin
44       repeat
45         Write(H[ind],'␣');
46         Inc(ind)
47       until H[ind]=0;
48       Inc(ind);
49       WriteLn();
50     end{for i};
51   end{Kilr};
```

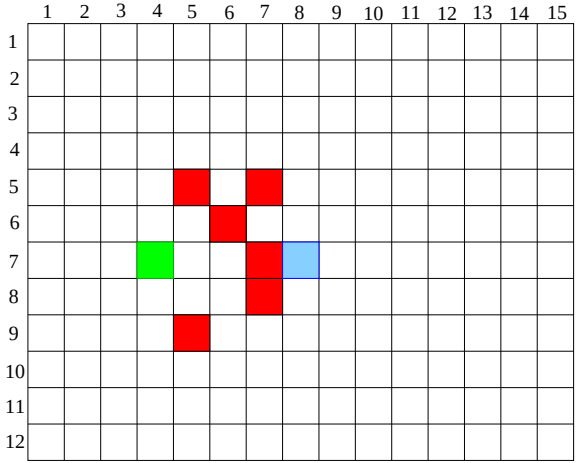
```
52 procedure Szamit;
53 var
54   i,eleje,vege, p,q:longint;
55   El:Lanc;
56 begin
57   Nap:=0; vege:=0;
58   for i:=1 to N do {a 0-megelőzöjüek halmazának kiszámítása}
59     if Befok[i]=0 then begin
60       Inc(vege);
61       H[vege]:=i;
62     end;
63   Inc(vege);
64   H[vege]:=0;
65   eleje:=1;
```

```
66    while True do begin
67      Inc(Nap);
68      while H[eleje]<>0 do begin
69        p:=H[eleje];
70        Inc(eleje);
71        El:=G[p];
72        while El<>0 do begin       {a p->q élek feldolgozása}
73          q:=Elek[El].pont;
74          El:=Elek[El].csat;       {továbblépés a láncban}
75          Dec(BeFok[q]);
76          if BeFok[q]=0 then begin{q felvétele az akt. napra}
77            Inc(vege);
78            H[vege]:=q;
79          end;
80        end{while El};
81      end{while eleje};
82      if vege=eleje then Break;
83      Inc(vege);
84      H[vege]:=0;
85      Inc(eleje);
86    end{while};
87  end{Szamit};
```

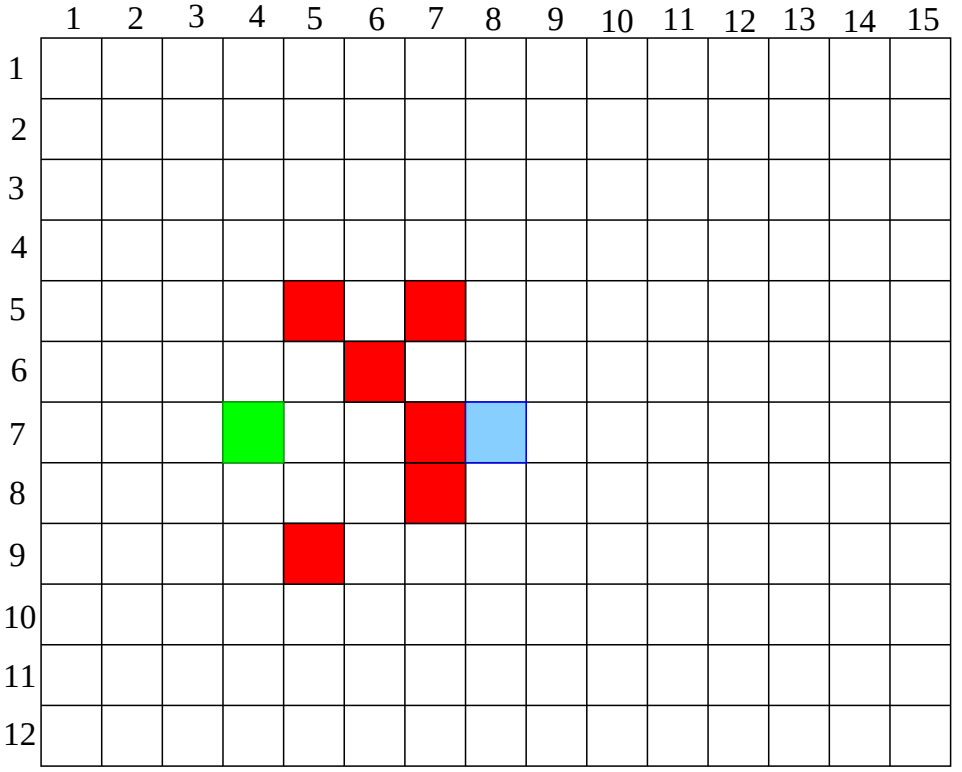
```
88  begin{Program}
89     Beolvas;
90     Szamit;
91     Kilr;
92  end.
```

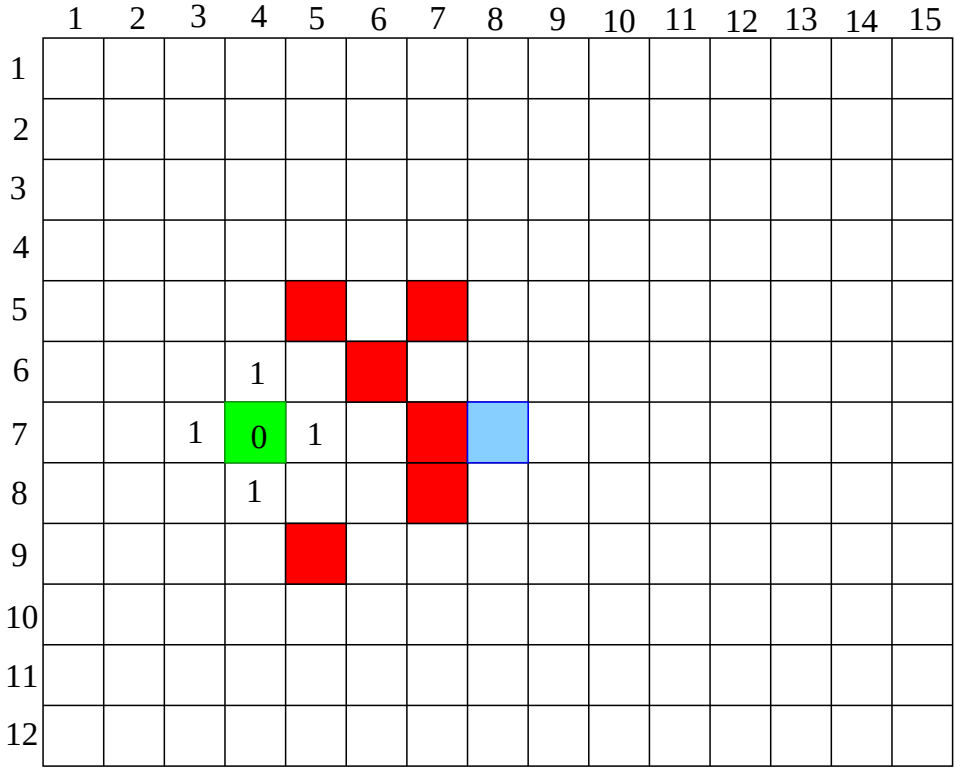
3.1. Feladat: Kiút keresés labirintusban.

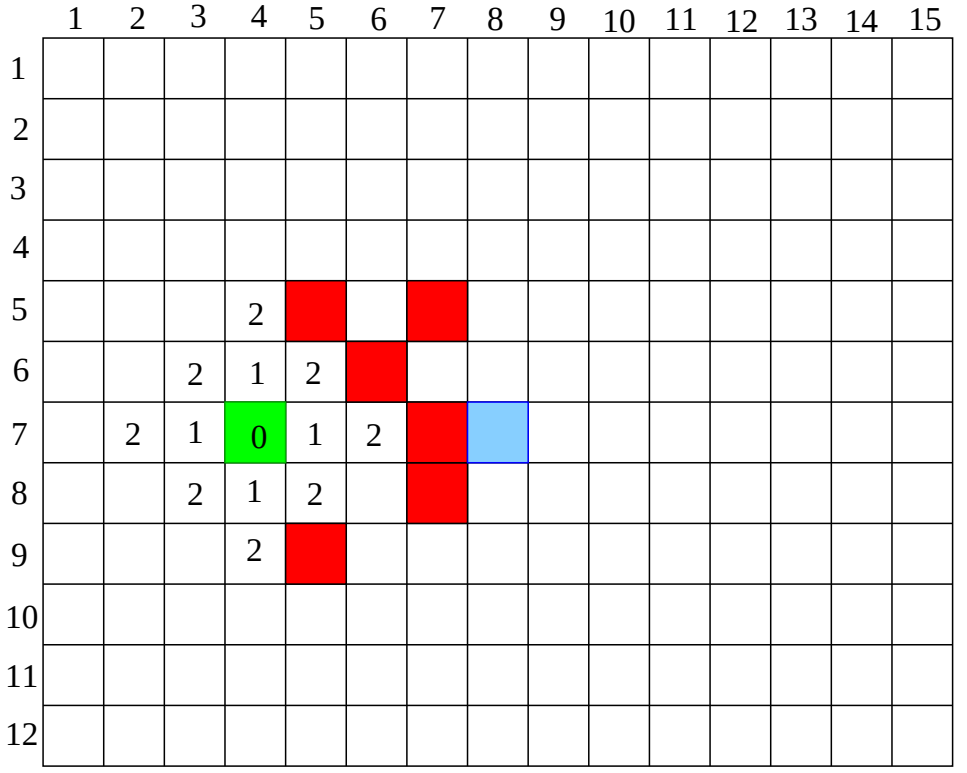
Egy négyzetrácsos hálózattal leírható labirintusban adott start helyről adott cél helyre kell eljutni a lehető legrövidebb úton. A labirintusban minden mező vagy fal, vagy közlekedésre használható folyosó egy része. Egy lépésben szomszédos mezőre léphetünk, balra, jobbra, felfelé vagy lefelé.

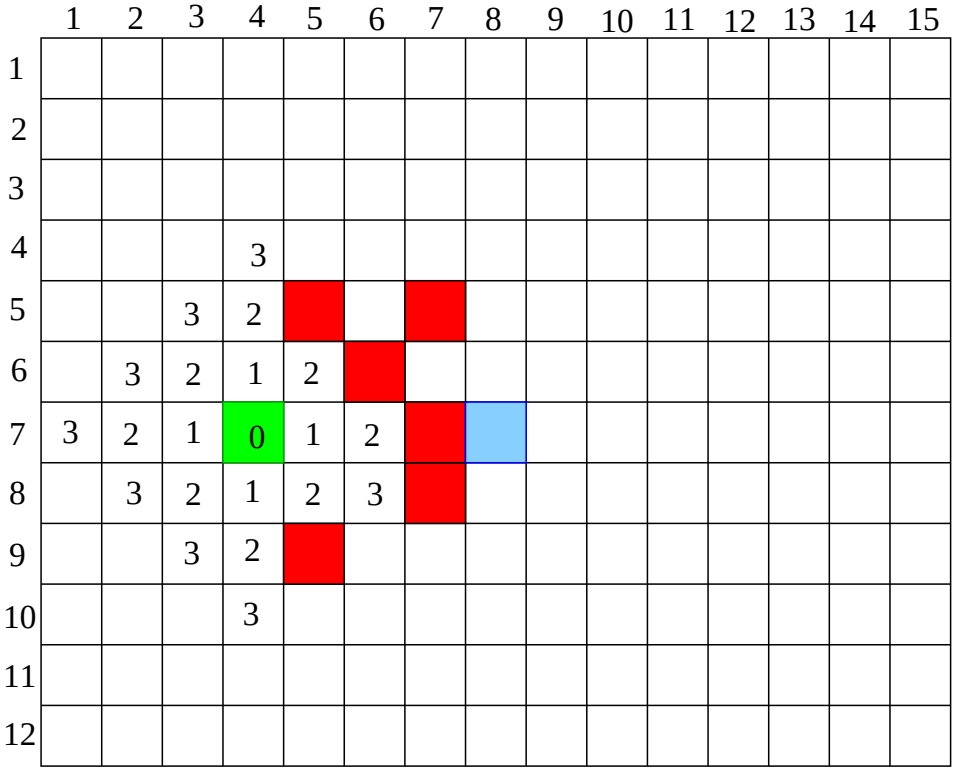


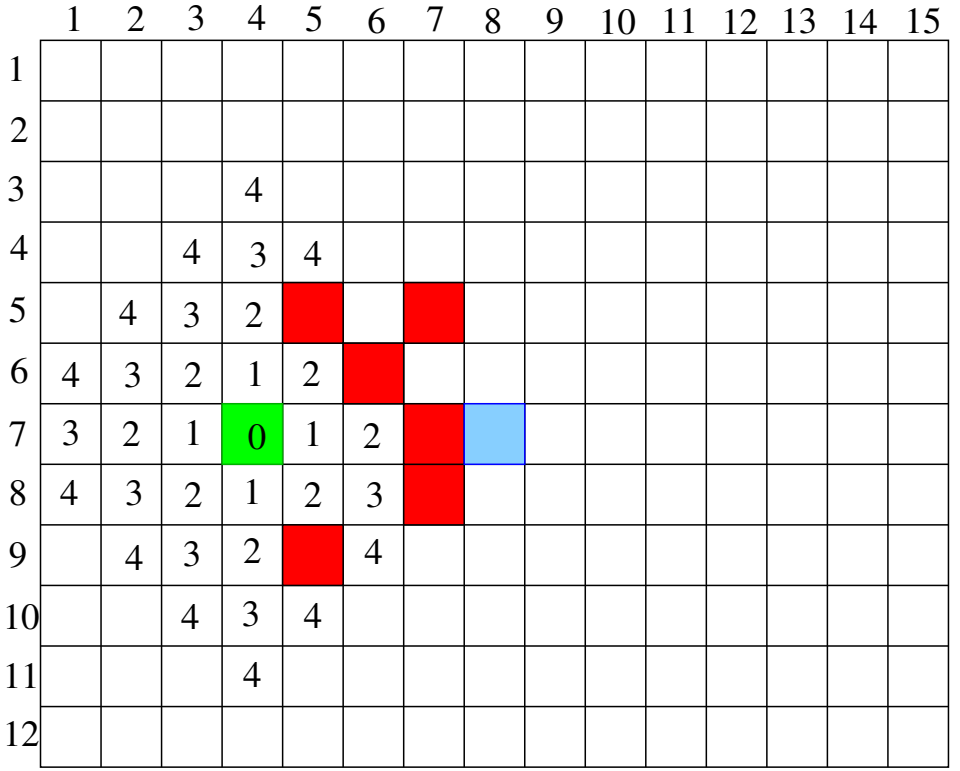
21. ábra.

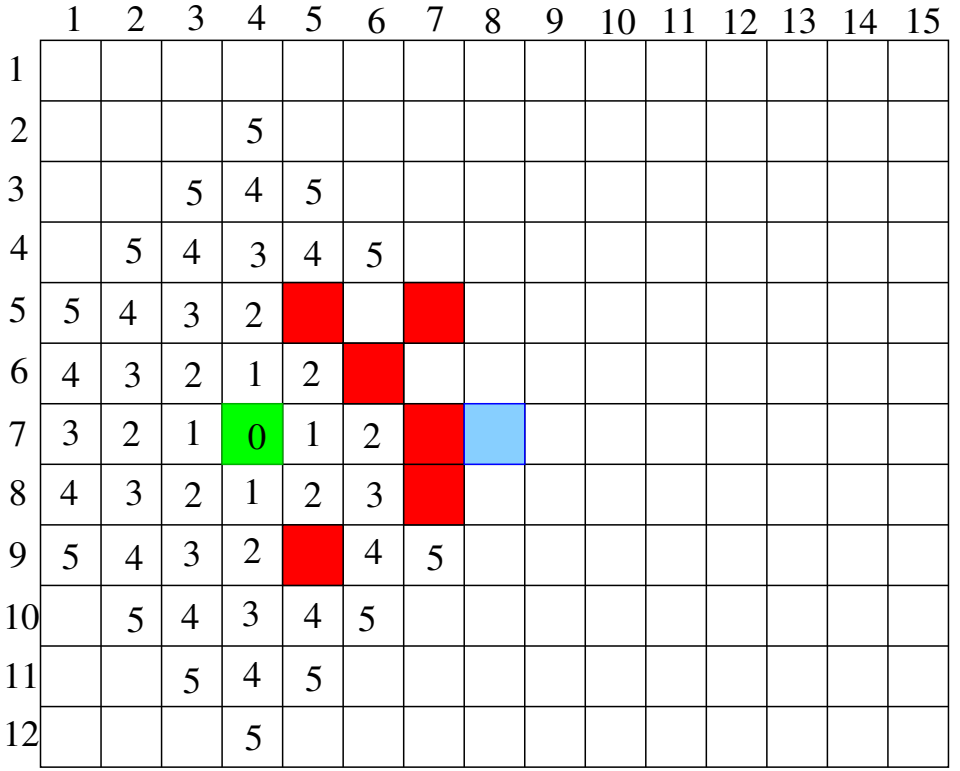

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
1
2
3
4
5
6
7
8
9
10
11
12

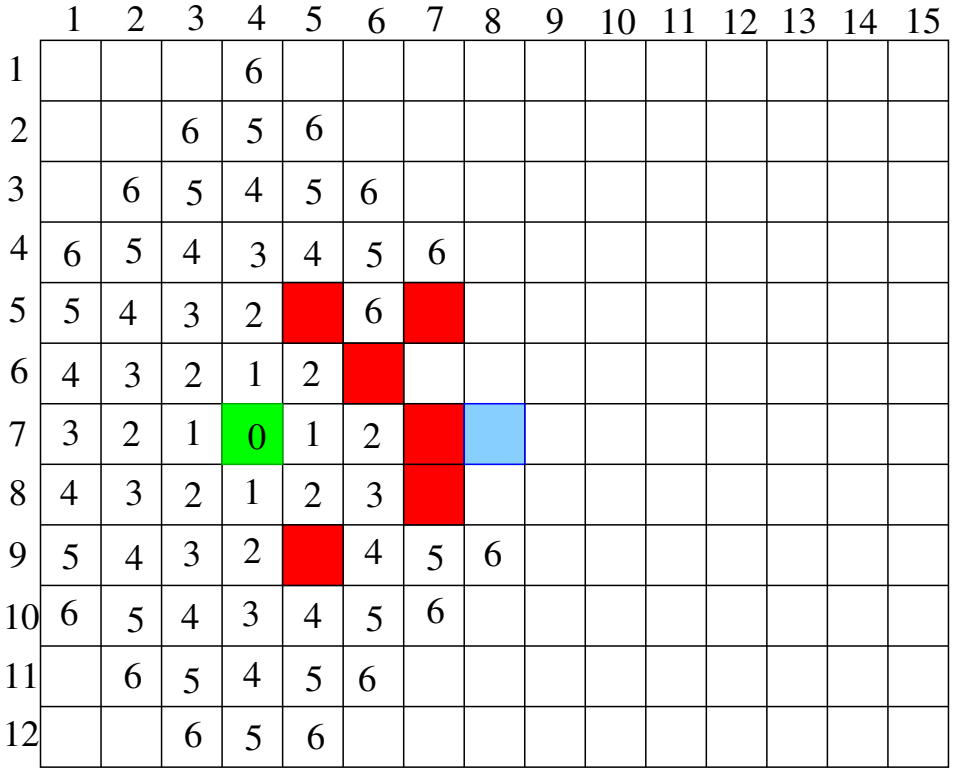

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
1 2 3 4 5 6 7 8 9 10 11 12
1
1
1 0 1
1

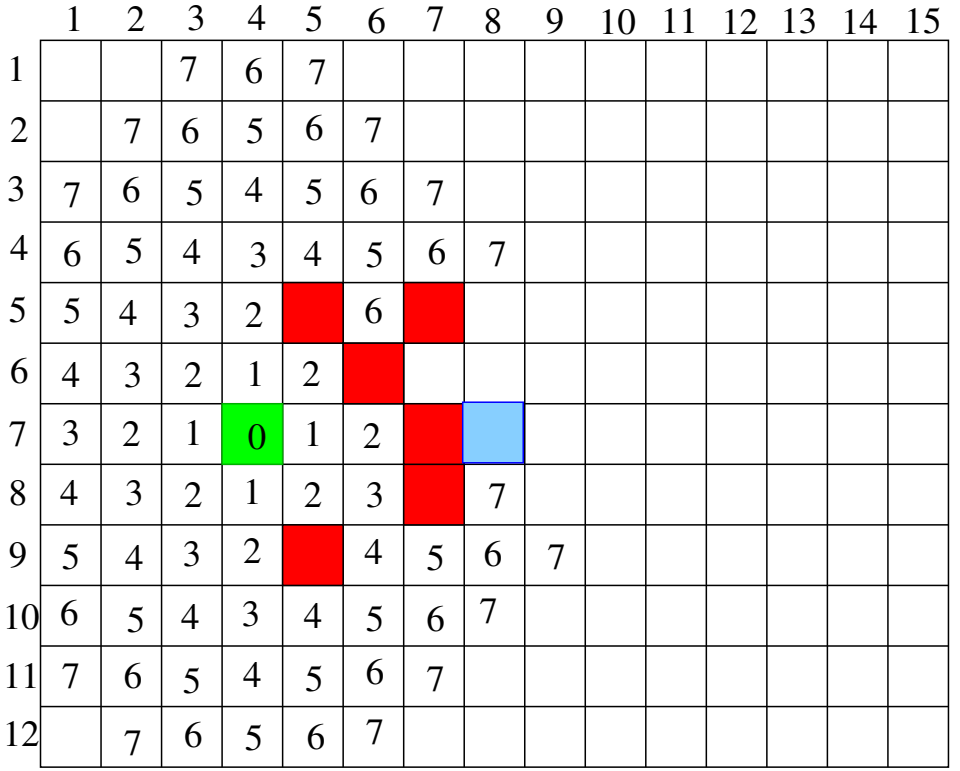

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
1
2
3
4
5
6
7
8
9
10
11
12
2
2 1 2
2 1 0 1 2
2 1 2
2

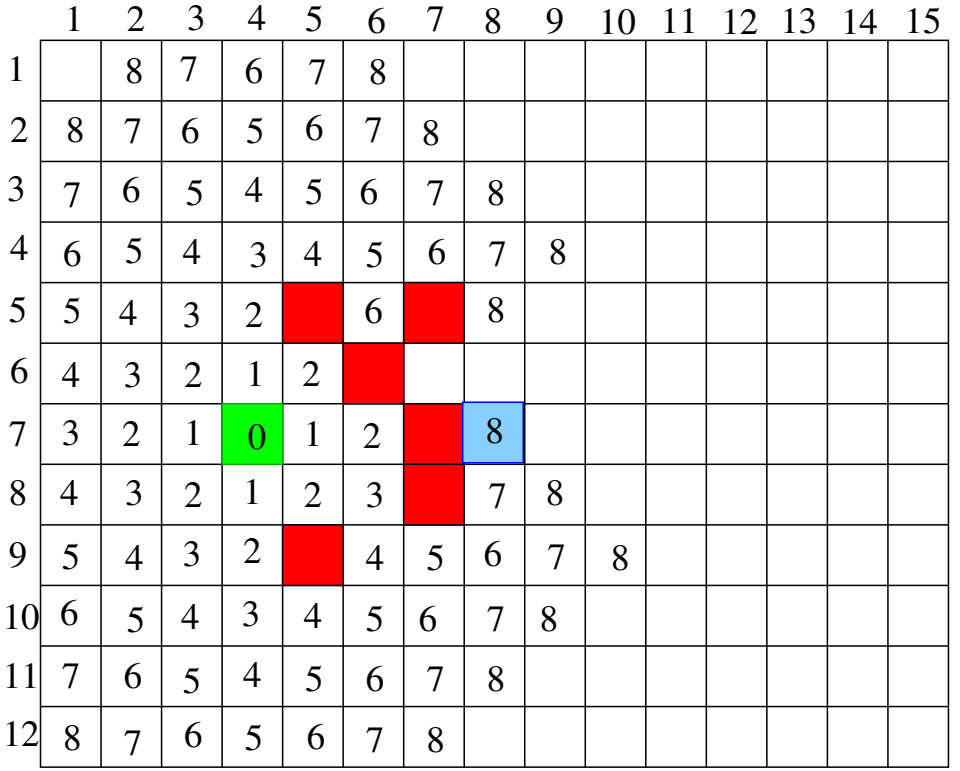

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
1
2
3
4
3
5
3 2
6
3 2 1 2
7
3 2 1 0 1 2
8
3 2 1 2 3
9
3 2
10
3
11
12


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


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


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


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


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:

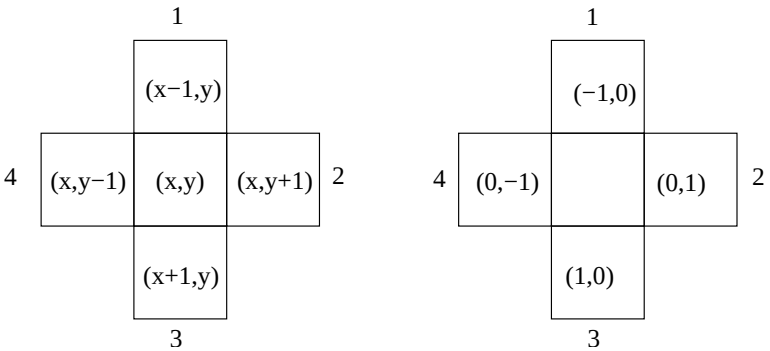
Tfh. a labirintus sorainak száma n, oszlopainak száma m. Tekintsük azt a $G = (V,E)$ gráfot, ahol $V = \{(x,y) : 1 \leq x \leq n \wedge 1 \leq y \leq m \wedge (x,y)\text{nem fal}\}$

$(x,y) \to (\bar{x},\bar{y}) \in E$ akkor és csak akkor, ha $|x-\bar{x}| + |y-\bar{y}| = 1$ valamint (x,y) és $(\bar{x},\bar{y})$ nem fal.

Vagyis az (x,y) mezőről egyetlen lépéssel juthatunk az $(\bar{x},\bar{y})$ 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 bejárással, a labirintus gráf számított-gráf árázolását használva.



22. ábra. A lehetséges lépések

3.2. Elemi gráfproblémák

3.2.1. Utak

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

3.1. 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 $(\forall i \in \{1,\ldots,k\})\ ((p_{i-1}, p_i) \in E)$.*

3.2. definíció. *A $\pi = p \rightsquigarrow q$ út hossza, $|\pi| = |p \rightsquigarrow q| = k$ (az utat alkotó élek száma)*

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

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

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

3.5. 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:

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

3.6. 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.

3.7. 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 $\delta_G(r,p) = \delta_F(r,p)$.

3.2.2. Útproblémák

Vaneút?

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

Elérhető pontok

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

Két pont közötti legrövidebb út

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

Egy ponttból induló legrövidebb utak

4. Adott p-re minden q-ra $\delta(p,q)$ és egy $p \rightsquigarrow q$ legrövidebb út kiszámítása.

Legrövidebb utak minden pontpárra

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

4. Szélességi bejárás

Bemenet: $G = (V,E)$ (irányított vagy irányítatlan) gráf és egy $r \in V$ pont.
Kiszámítandó minden p pontra az r-ből p-be vezető legrövidebb út hossza, és legrövidebb út.
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
$\overline{V} = \{p : Apa(p) \neq null \vee p = r\}$,
$\overline{E} = \{(p,q) : Apa(q) = p \wedge p \neq null\}$
Az F gráf a G gráf r-gyökerű legrövidebb utak feszítőfája (LUF).

4.1. Elvi algoritmus

Legyen $G = (V,E)$ terszőleges irányított, vagy irányítatlan gráf.
Jelölje $S(t)$ azon p pontok halmazát, amelyek t távolsára vannak r-től.

$$S(t) = \{p \in V : \delta(r,p) = t\}$$

Nyilvánvaló, hogy $S(0) = \{r\}$.
Jelölés: $D(p) = \delta(r,p)$
Nyilvánvaló, hogy minden $t > 0$-ra és minden $q \in S(t)$ van olyan $p \in S(t-1)$, hogy $(p,q) \in E$
Fordítva, ha $p \in S(t-1)$ és $(p,q) \in E$, akkor $D(q) \leq t+1$, hiszen van $t+1$ hosszú út r-ből q-ba, mert van t hosszú (legrövidebb) út r-ből p-be és van él p-ből q-ba: $r \rightsquigarrow p \to q$.
Tehát az $S(t)$ halmaz előállítható az $S(t-1)$ halmazból.

```
Inf := |V|;  //pontok száma
for p ∈ V do D(p) := Inf;
D(r) := 0;
S := {r};
repeat
    St := ∅;
    for p ∈ S do begin
        for q ∈ KiEl(p) do //  minden p → q  élre
            if D(q) = Inf then begin  //q benemjárt
                D(q) := D(p) + 1;
                Apa(q) := p;  //p-ből megyünk q-ba
                St := St + {q};
            end;
    end;
    S := St;
until S <> ∅ ;
```

A szélességi bejárás megvalósítása Pascal nyelven

```
1  procedure SzeltBejar(r:PontTip; N:longint; const G:Graf;
2      var D:array of longint; var Apa:array of longint);
3  //r–ből induló legrövidebb utak meghatározása
4    var
5      S:Sor;
6      p,q:PontTip;   i:integer;
7    begin{SzeltBejar}
8      for p:=1 to N do D[p]:=Inf;      { inicializálás}
9      D[r]:=0; Apa[r]:=0;
10     Letesit(S);
11     Sorba(S,r);
12     while Elemszam(S)>0 do begin
13       p:=SorBol(S);
14       for i:=1 To KiFok[p] Do Begin { p–>q élre }
15         q:=G[p,i];
16         if D[q]=Inf then begin       { ha q meg nem elért }
17           Apa[q]:=p;
18           D[q]:=D[p]+1;
19           Sorba(S,q);
20         end;
21       end{for p–>q};
22     end{while};
23   end{SzeltBejar};
```

A Sor adattípus megvalósítása

```
1  const
2    SorMeret=10000;
3  type
4    PontTip=integer;
5    Sor=record
6      Tar:Array[1..SorMeret] of PontTip;
7      eleje,vege:integer;
8     end;
9  procedure Letesit(var S:Sor);
10 begin
11   S.eleje:=1; S.vege:=1;
12 end{Letesit};
13 procedure Sorba(var S:Sor; p:PontTip);
14 begin
15   S.Tar[S.vege]:=p;
16   Inc(S.vege);
17 end{Sorba};
18 function SorBol(var S:Sor):PontTip;
19 begin
20   if S.eleje>=S.vege then exit;
21   SorBol:=S.Tar[S.eleje];
22   inc(S.eleje);
23 end{Sorbol};
```

```
24 function Elemszam(var S:Sor):integer;
25 begin
26   Elemszam:=S.vege-S.eleje;
27 End{Elemszam};
```

4.1.1. Egy legrövidebb út kiíratása

```
1  procedure UtKiIro(r,p:PontTip);
2  //Globalis: Apa
3  var u,i:longint;
4    Ut:array[1..maxP] of longint;
5  begin
6    u:=0;
7    while p<>r do begin
8      inc(u);
9      Ut[u]:=p;
10     p:=Apa[p];
11   end;
12   write(r);
13   for i:=u downto 1 do begin
14     write('->',Ut[i]);
15   end; writeln();
16 end;
```

4.2. Feladat: Mérőkannák

Egy gazdának két kannája van, az egyik A literes, a másik pedig B. Szeretne kimérni pontosan L liter vizet az első kannájába. 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

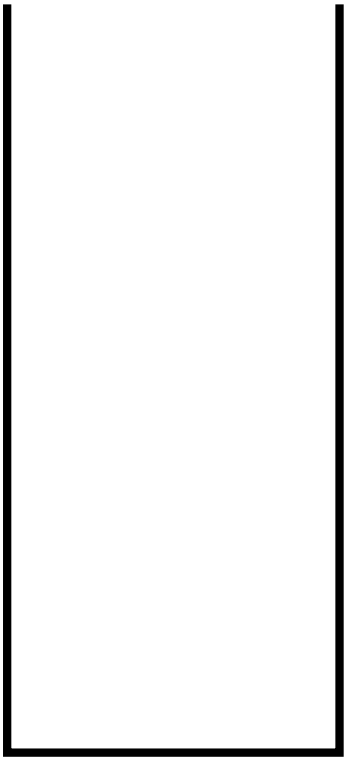
A `kimer.be` szöveges állomány első sora a két egész számot tartalmaz, a két kanna ürtartalmát: $AB(0 < A,B <= 200)$. A második sor a kimérendő mennyiséget tartalmazza.

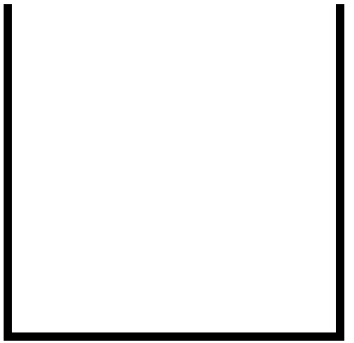
Kimenet

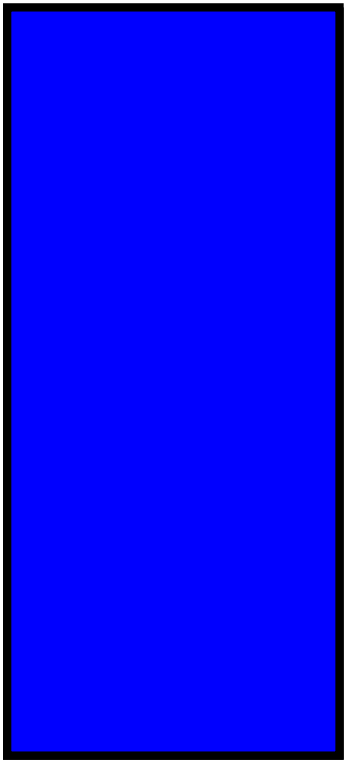
A `kimer.ki` szöveges állomány első sora a kimérés lépéseinek (minimális) m számát tartalmazza. A második sor pontosan m egész számot tartalmazzon egy-egy szóközzel elválasztva, a kimérés lépéseit.

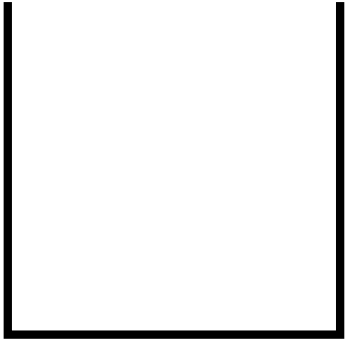
Példa bemenet és kimenet

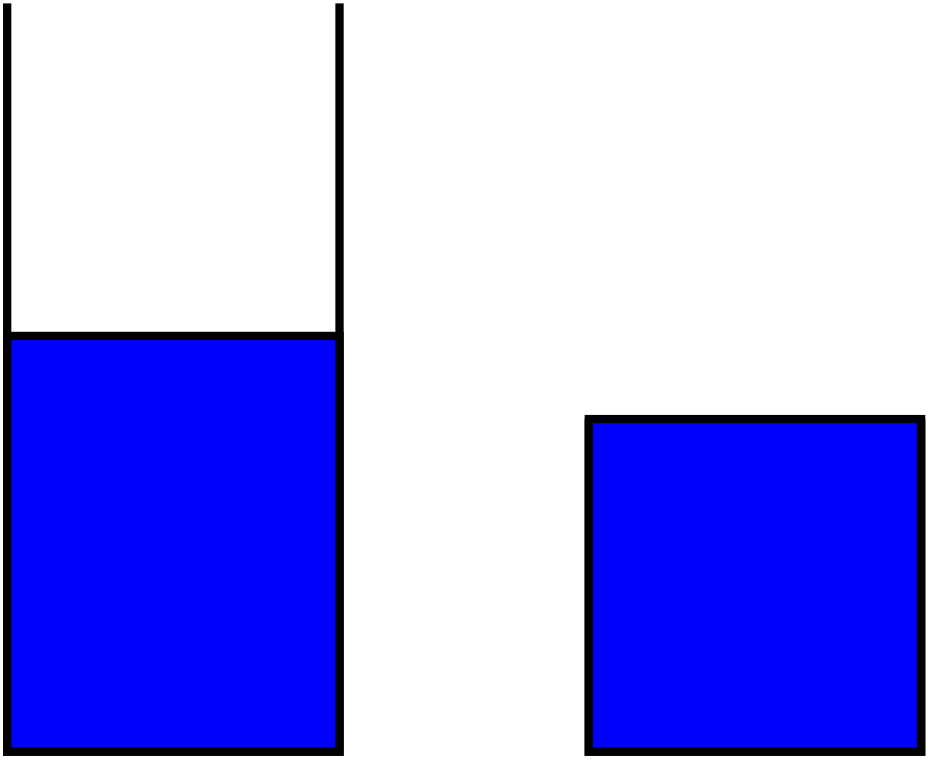
bemenet	kimenet
9 4 6	8 1 5 4 5 4 5 1 5

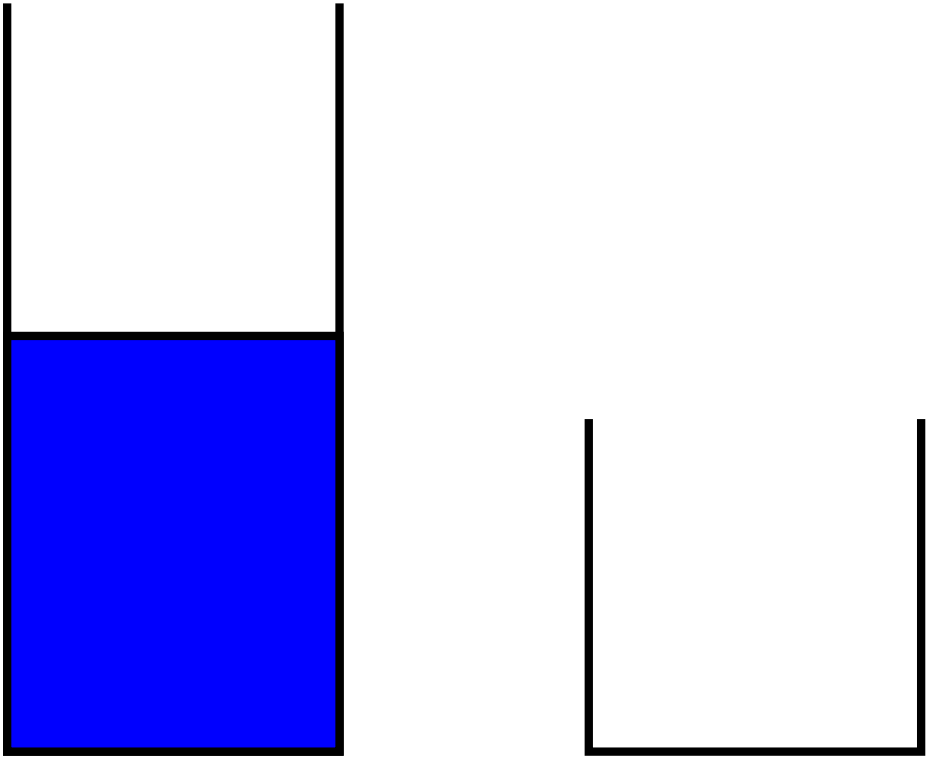


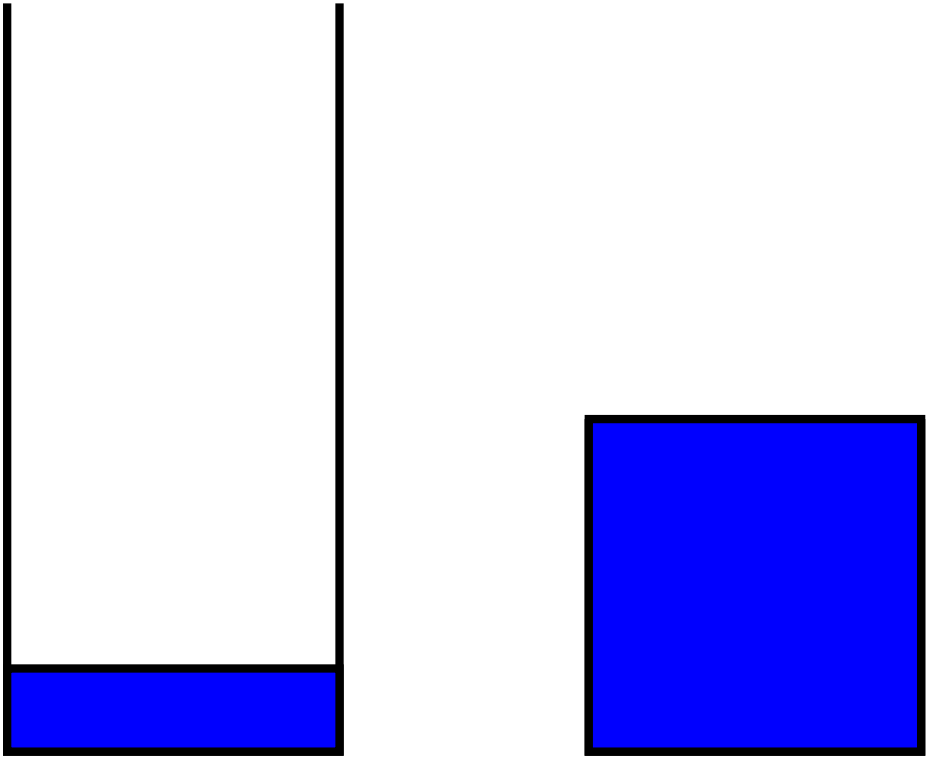


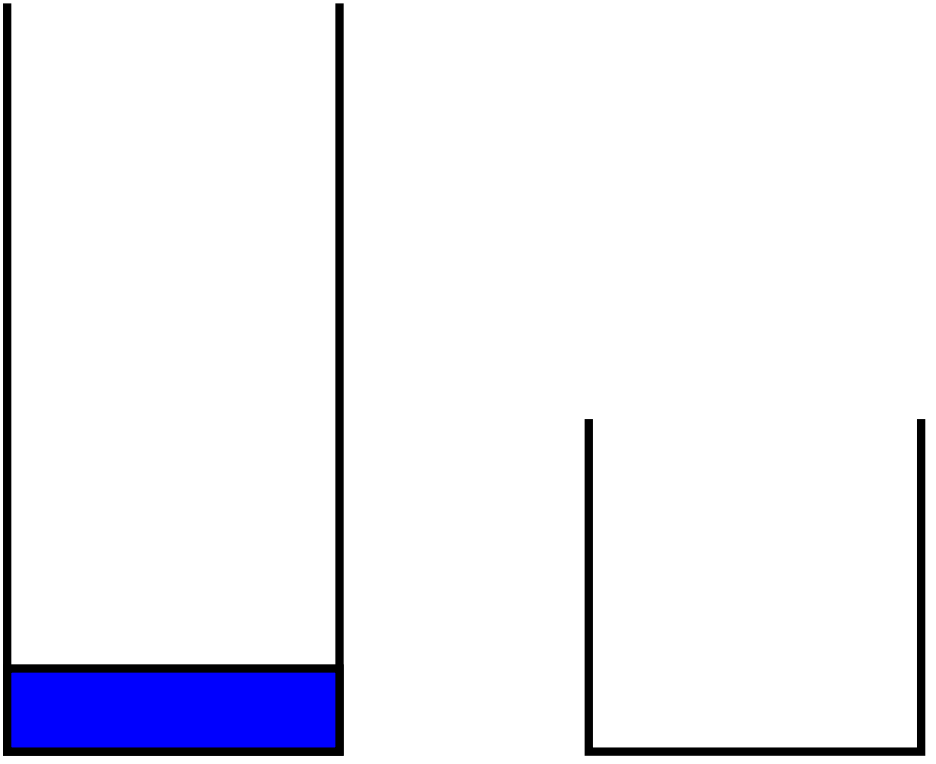


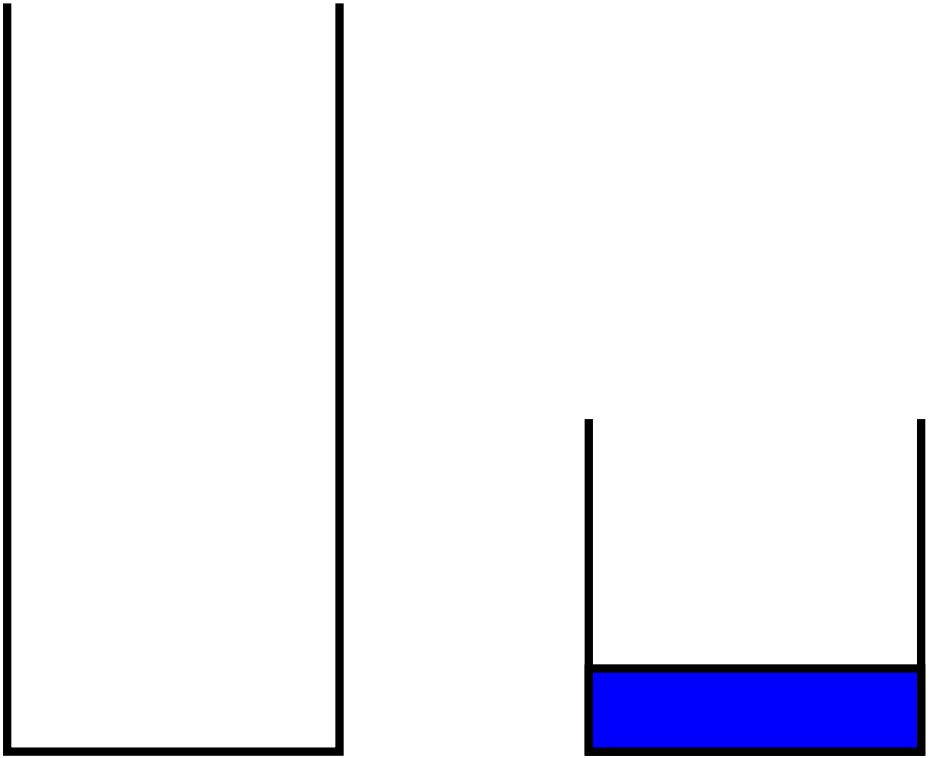


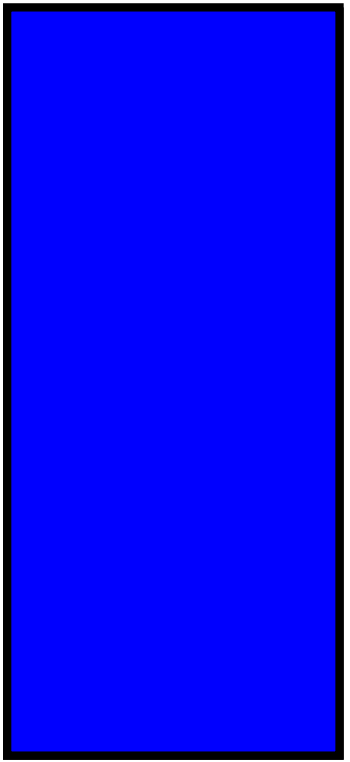


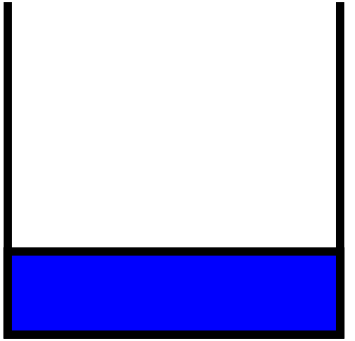


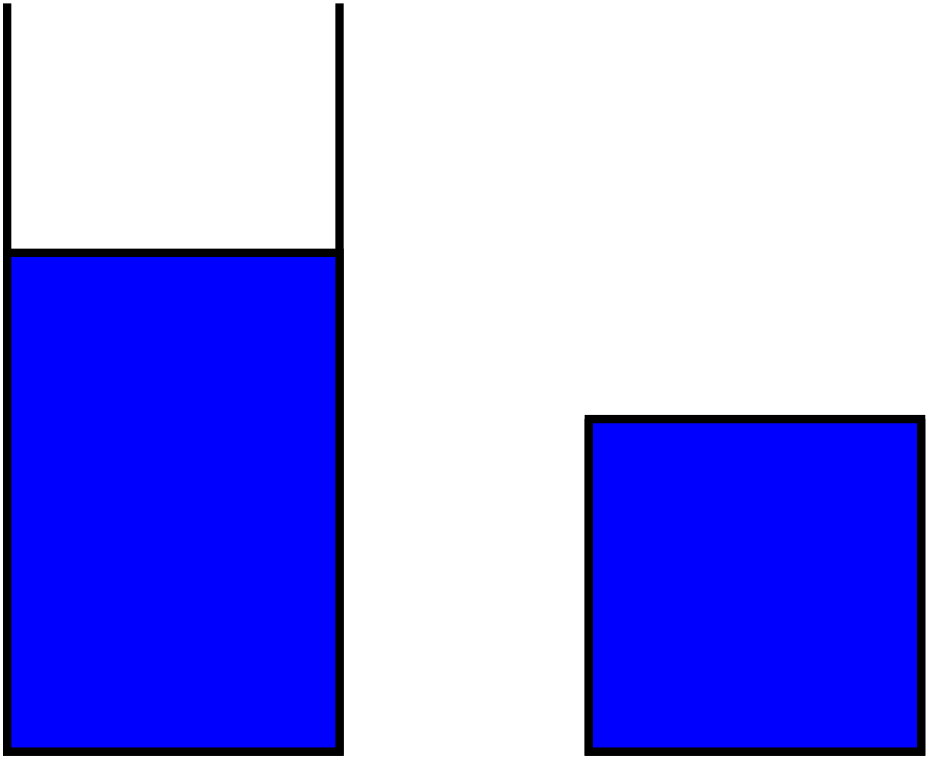












kezdő állapot: (0,0)

kezdő állapot: (0,0)
1 öntéssel kapható állapotok: (9,0) (0,4)

kezdő állapot: (0,0)
1 öntéssel kapható állapotok: (9,0) (0,4)
2 öntéssel kapható állapotok: (9,4) (5,4) (4,0)

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)

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)

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)

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)

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)

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)

Megoldás

Minden állapot azonosítható egy (x,y) számpárral ($0 \leq x \leq A$, $0 \leq y \leq B$).
A kezdő állapot $(0,0)$. Tudjuk, hogy egy adott (x,y) állapotból egyet öntéssel milyen állapotok keletkezhetnek.
Tekinsük azt a $G = (V,E)$ gráfot, amelynek pontjai az állapotok, tehát $V = \{(x,y) : 0 \leq x \leq A, 0 \leq y \leq B\}$ és $(x,y) \to (\overline{x},\overline{y})$ akkor és csak akkor él a G gráfban, ha az $(\overline{x},\overline{y})$ állapot egyetlen öntéssel megkapható az (x,y) állapotból. Miden pontból legfeljebb 6 él indul ki, és ezek végpontjai kiszámíthatók x,y függvényeként. Tehát számított gráf ábrázolás alkalmazható.
Tehát a feladat olyan $(0,0) \rightsquigarrow (x,L)$ legrövidebb út keresése a G gráfban.
A legrövidebb utak feszítőfáját az $Apa : 0..A \times 0..B \to 0..A \times 0..B$ függvénnyel ábrázoljuk. Tehát $Apa(p) = q$ azt jelenti, hogy az $(0,0) \rightsquigarrow p$ legrövidebb úton p előtt q szerepel, azaz $(0,0) \rightsquigarrow p = (0,0) \rightsquigarrow q \to p$. Ha nem létezik $(0,0) \rightsquigarrow p$ (kezdetben minden $p <> (0,0)$-ra) akkor $Apa(p).x = -1$.
Az (x,y) párok ábrázolására rekord (struct) típust használunk; `Allapot=record x,y:word end`.

```
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 procedure Letesit(var S:Sor);
11 begin
12   S.eleje:=1; S.vege:=1;
13 end{Letesit};
14 Procedure Sorba(var S:Sor;v:Allapot);
15 begin
16     S.Tar[S.vege]:=v; Inc(S.vege);
17 end;
18 function Sorbol(var S:Sor):Allapot;
19 begin
20   SorBol:=S.Tar[S.eleje]; Inc(S.eleje);
21 end;
22 function Elemszam(var S:Sor):integer;
23 begin
24   Elemszam:=S.vege-S.eleje;
25 End{Elemszam};
```

```
26 var
27   A,B,L:Word;
28   Apa: array[0..MaxV,0..MaxV] of Allapot;
29   S:Sor;
30
31 Procedure Beolvas;
32   var
33     BeF: Text;
34 begin
35     Assign(BeF,'kimer.be'); Reset(BeF);
36     ReadLn(BeF,A,B);
37     ReadLn(BeF,L);
38     Close(BeF);
39 end{Beolvas};
40
41 Procedure Kilr;
42 const
43   maxA=10000;
44 var KiF:Text;
45   x,y,m,i,lepes:integer;
46   Lehet:boolean;
47   szep:string;
48   U,V:Allapot;
49   Lepsor:array[1..MaxA]of Allapot;
50 begin
```

```
51      Assign(KiF,'kimer.ki'); Rewrite(KiF);
52      lehet:=false;
53      for y:=0 to B do if Apa[L,y].x>0 then begin
54        lehet:=true; U.y:=y;
55        break;
56      end;
57      if not lehet then begin
58        writeln(KiF,0);
59        close(KiF); exit;
60      end;
61      U.x:=L; m:=0;
62      while (U.x<>0)or(U.y<>0) do begin
63        inc(m);
64        LepSor[m]:=U;
65        U:=Apa[U.x,U.y];
66      end;
67      U.x:=0; U.y:=0;
68      writeln(KiF,m);
69      szep:='';
70      for i:=m downto 1 do begin
71        V:=LepSor[i];
72        if (V.x=A)and(U.y=V.y) then begin
73          lepes:=1;
74        end else if (U.x=V.x)and(V.y=B) then begin
75          lepes:=2;
```

```
76        end else if (V.x=0)and(U.y=V.y) then begin
77           lepes:=3;
78        end else if (U.x=V.x)and(V.y=0) then begin
79           lepes:=4;
80        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) th
81           lepes:=5;
82        end else begin
83           lepes:=6;
84        end;
85        write(KiF,szep,lepes);
86        szep:='␣';
87        U:=V;
88      end{for i-};
89      writeln(KiF);
90      Close(KiF);
91  end{Kilr};
```

```
92  procedure Szamit;
93  var
94    x,y:Word;
95    U,V:Allapot;
96  begin{}
97    for x:=0 to A do
98      for y:=0 to B do Apa[x,y].x:=-1;
99    Letesit(S);
00    Apa[0,0].x:=0;
01    U.x:=0; U.y:=0;
02    Sorba(S,U);
```

```
03    while (Elemszam(S)>0) do begin
04      U:=Sorbol(S);
05  {minden U->V élre:}
06      if (U.x=L) then break;
07      if Apa[A,U.y].x<0 then begin {1. lépés}
08         V.x:=A; V.y:=U.y;
09         Apa[V.x,V.y]:=U;
10         Sorba(S,V);
11      end;
12      if Apa[U.x,B].x<0 then begin {2. lépés}
13         V.x:=U.x; V.y:=B;
14         Apa[V.x,V.y]:=U;
15         Sorba(S,V);
16      end;
17      if Apa[0,U.y].x<0 then begin {3. lépés}
18         V.x:=0; V.y:=y;
19         Apa[V.x,V.y]:=U;
20         Sorba(S,V);
21      end;
22      if Apa[U.x,0].x<0 then begin {4. lépés}
23         V.x:=U.x; V.y:=0;
24         Apa[V.x,V.y]:=U;
25         Sorba(S,V);
26      end;
```

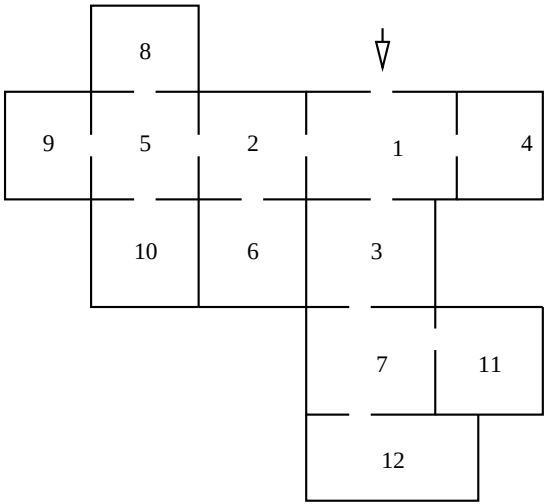
```
27      if U.x+U.y<=B then begin
28        V.x:=0; V.y:=U.x+U.y;
29      end else begin
30        V.x:=U.x-(B-U.y); V.y:=B;
31      end;
32      if Apa[V.x,V.y].x<0 then begin {5. lépés}
33         Apa[V.x,V.y]:=U;
34         Sorba(S,V);
35      end;
36      if U.x+U.y<=A then begin
37        V.y:=0; V.x:=U.x+U.y;
38      end else begin
39        V.y:=U.y-(A-U.x); V.x:=A;
40      end;
41      if Apa[V.x,V.y].x<0 then begin {6. lépés}
42         Apa[V.x,V.y]:=U;
43         Sorba(S,V);
44      end;
45     end {While S<>[] };
46  end{Szamit};
47  begin
48    Beolvas;
49    Szamit;
50    Kilr;
51  end.
```

5. Mélységi bejárás

5.1. Első képtár látogatás

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)!

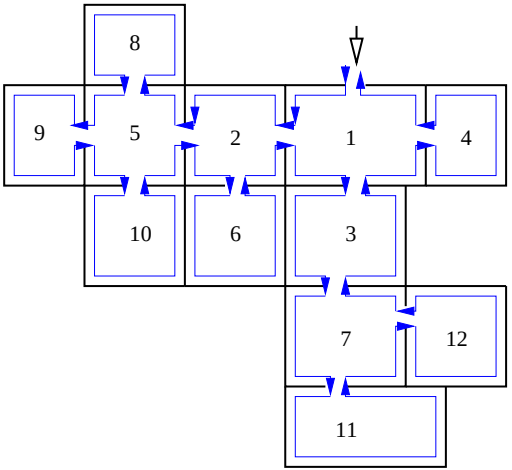


23. ábra.

Megoldás

A fal mellett mindig jobbra haladjunk.

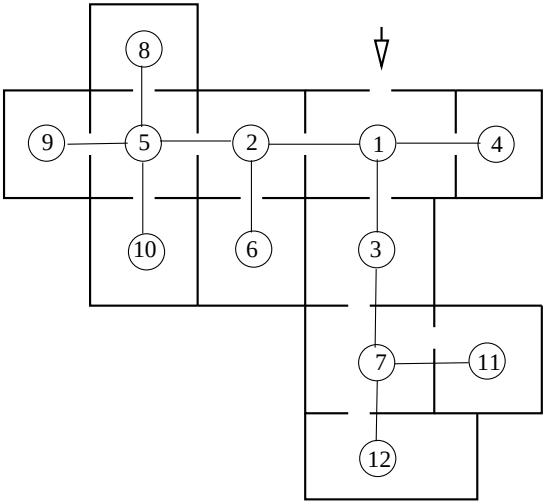
Tekintsük azt a $G = (V,E)$ gráfot, amelynek pontjai (csúcsai) a képtár termei, $V = \{1,\ldots,11\}$,



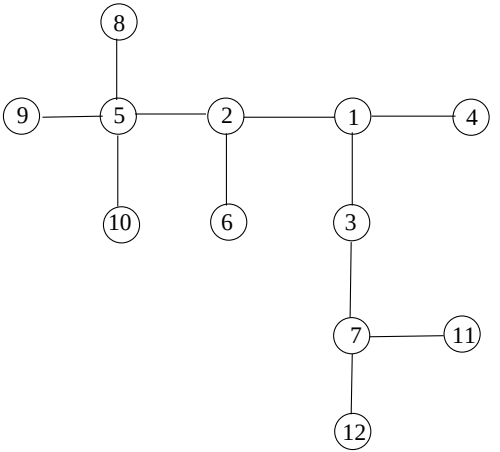
24. ábra.

$(p,q) \in E$ akkor és csak akkor, ha p teremből nyílik ajtó a q terembe.

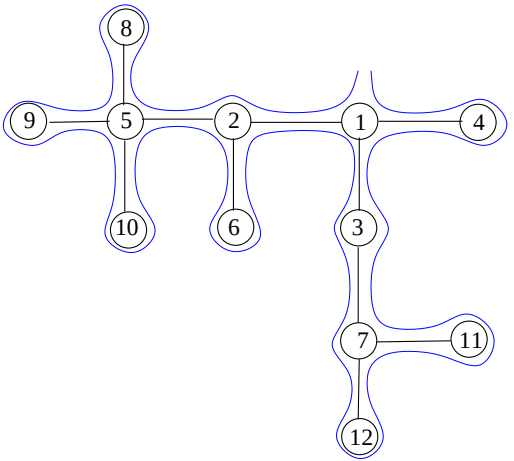
Ez a gráf fa, ezért lehet a fenti stratégiával bejárni a képtár minden termét.



25. ábra.



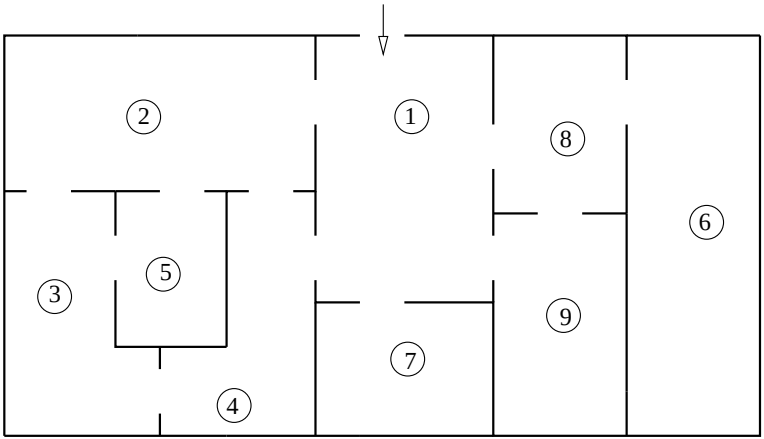
26. ábra.



27. ábra.

5.2. Második képtárlátogatás

Egy sok teremből álló képtárban teszünk látogatást. Adjuk meg egy bejárást, amely biztosítja, hogy minden terembe eljutunk és a végén visszaérkezünk a kiindulási terembe!
Minden teremben kirakták a terem sorszámát és minden ajtón rajta van, hogy melyik terembe vezet.



28. ábra. A képtár termeinek alaprajza

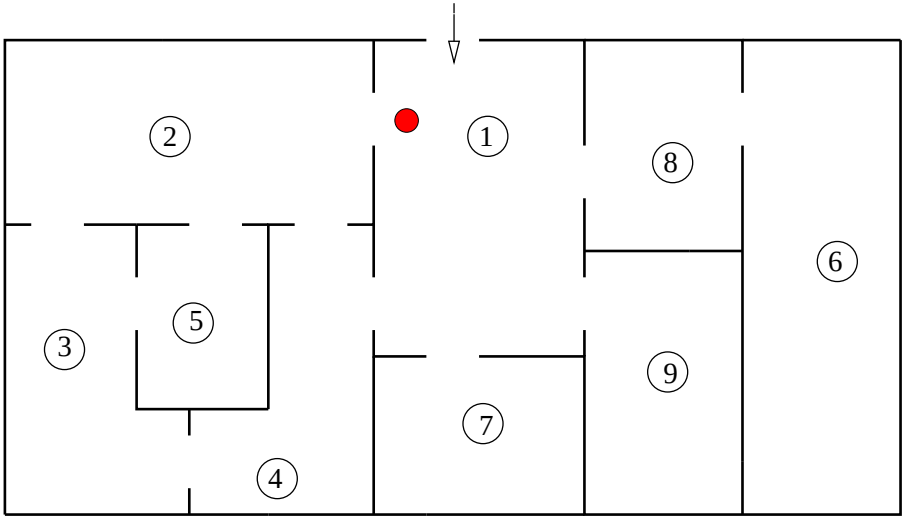
- Mit kell tudnunk?

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

- Mit kell tudnunk?

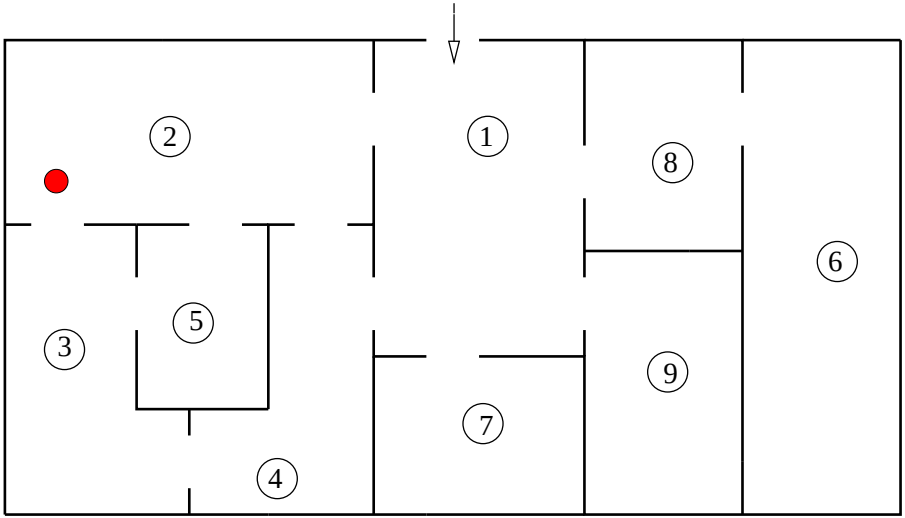
1. Jártunk-e már az adott teremben?
2. Melyik teremből léptünk először az adott terembe?

Minden p teremre $Honnan(p)$ legyen annak a teremnek a száma, amelyből először lépünk a p terembe. Kezdetben legyen minden p-re $Honnan(p) = -1$. 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.



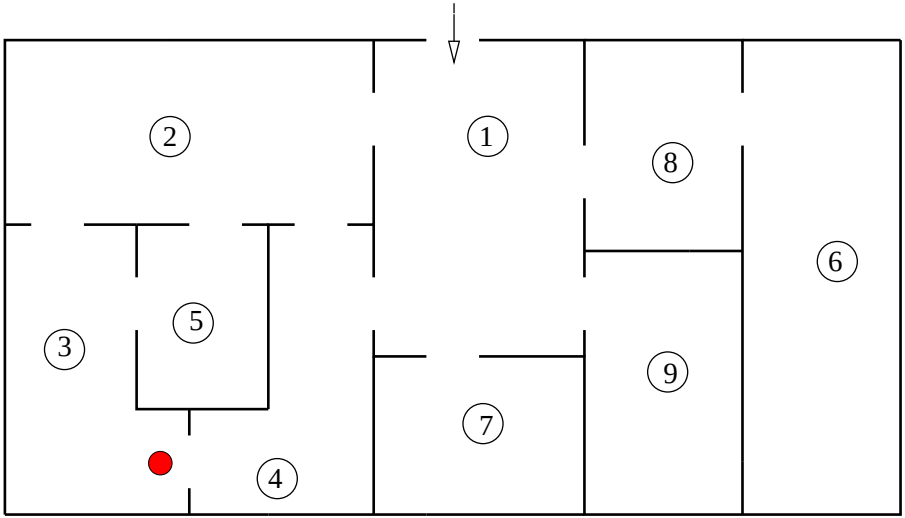
terem	1	2	3	4	5	6	7	8	9
honnan	−1								

utvonal 1



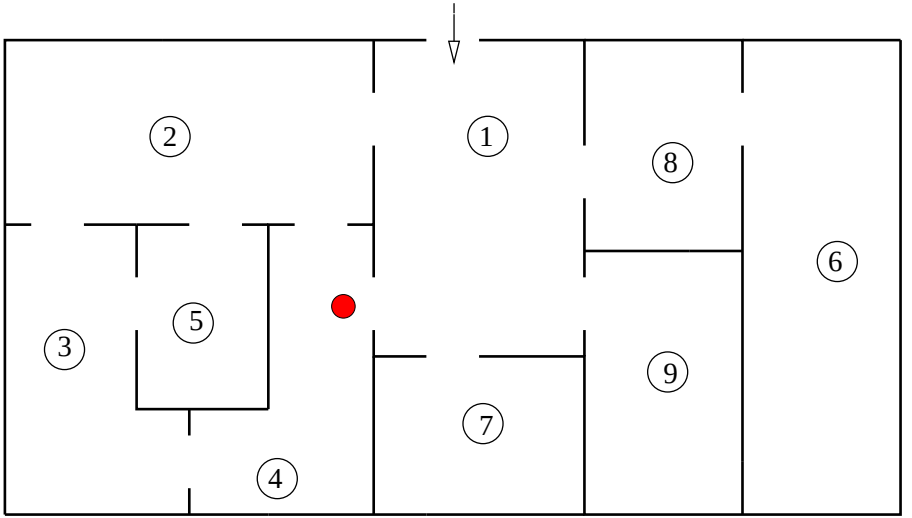
terem	1	2	3	4	5	6	7	8	9
honnan	−1	1							

utvonal 1 2



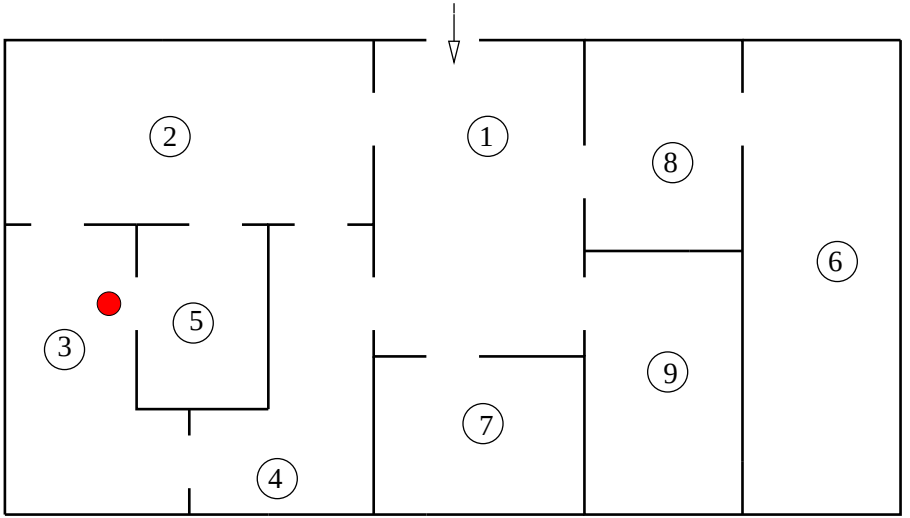
terem	1	2	3	4	5	6	7	8	9
honnan	−1	1	2						

utvonal 1 2 3



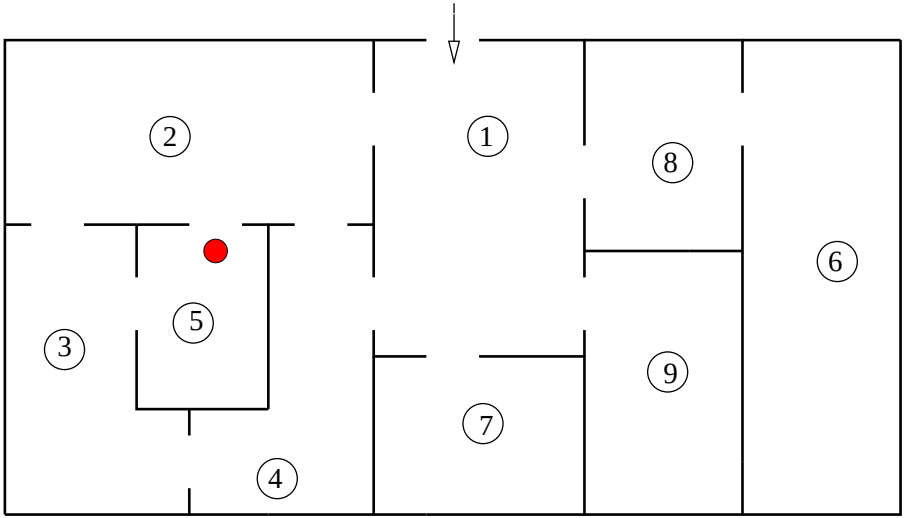
terem	1	2	3	4	5	6	7	8	9
honnan	−1	1	2	3					

utvonal 1 2 3 4



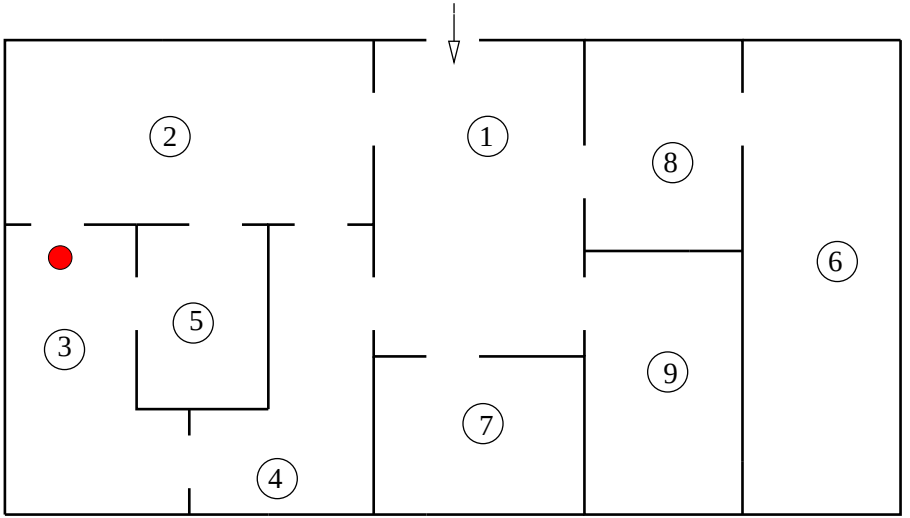
terem	1	2	3	4	5	6	7	8	9
honnan	−1	1	2	3					

utvonal 1 2 3 4 3



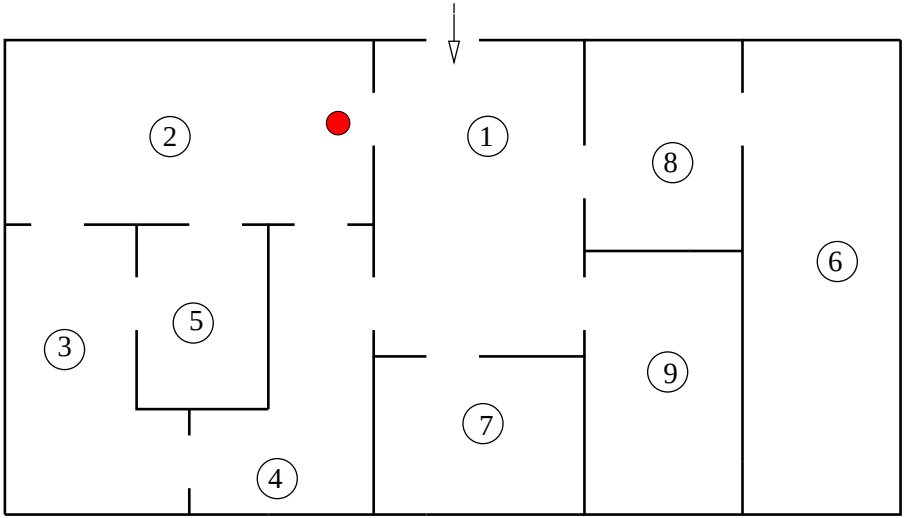
terem	1	2	3	4	5	6	7	8	9
honnan	−1	1	2	3	3				

utvonal 1 2 3 4 3 5



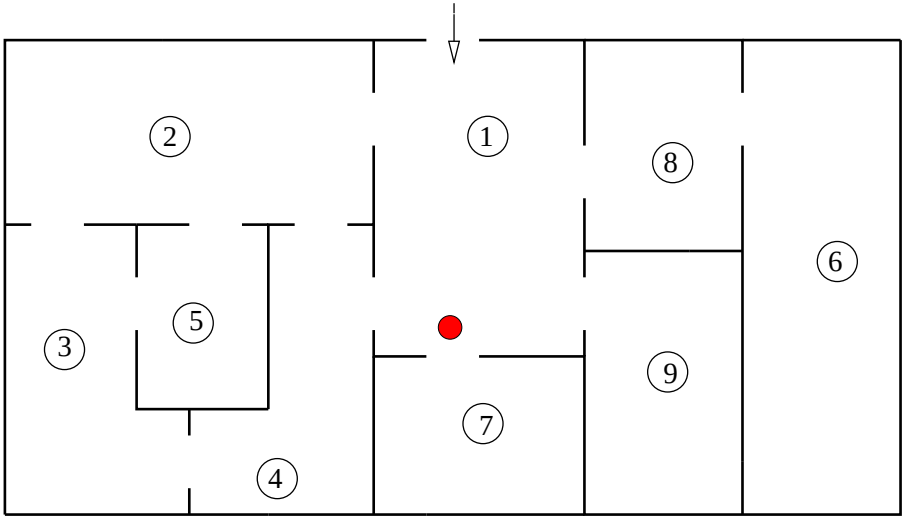
terem	1	2	3	4	5	6	7	8	9
honnan	−1	1	2	3	3				

utvonal 1 2 3 4 3 5 3



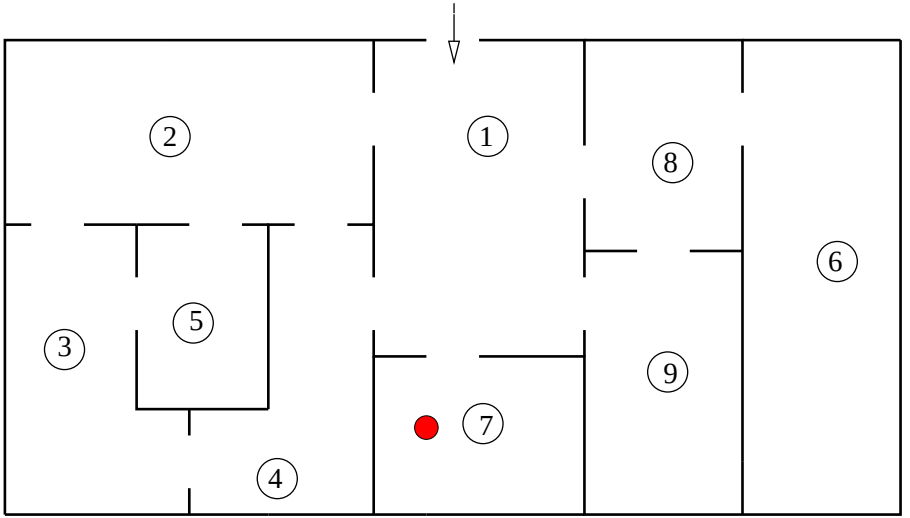
terem	1	2	3	4	5	6	7	8	9
honnan	−1	1	2	3	3				

utvonal 1 2 3 4 3 5 3 2



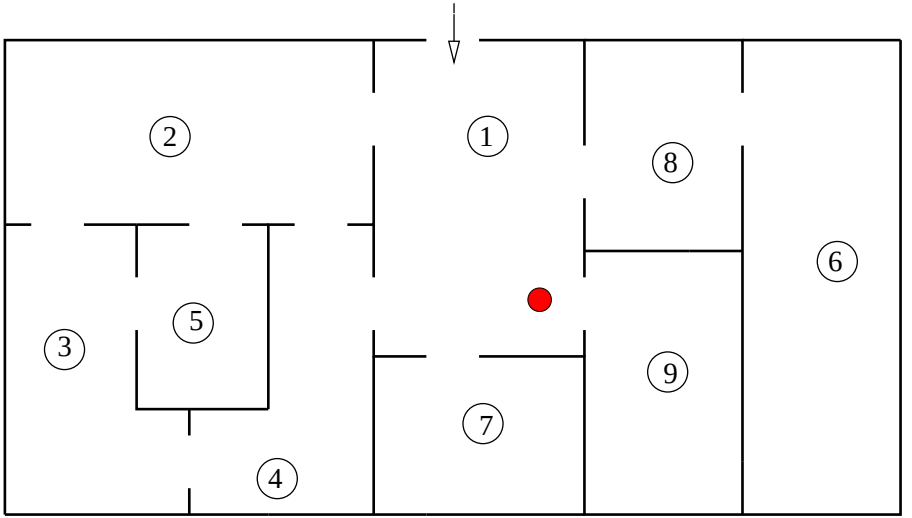
terem	1	2	3	4	5	6	7	8	9
honnan	−1	1	2	3	3				

utvonal 1 2 3 4 3 5 3 2 1



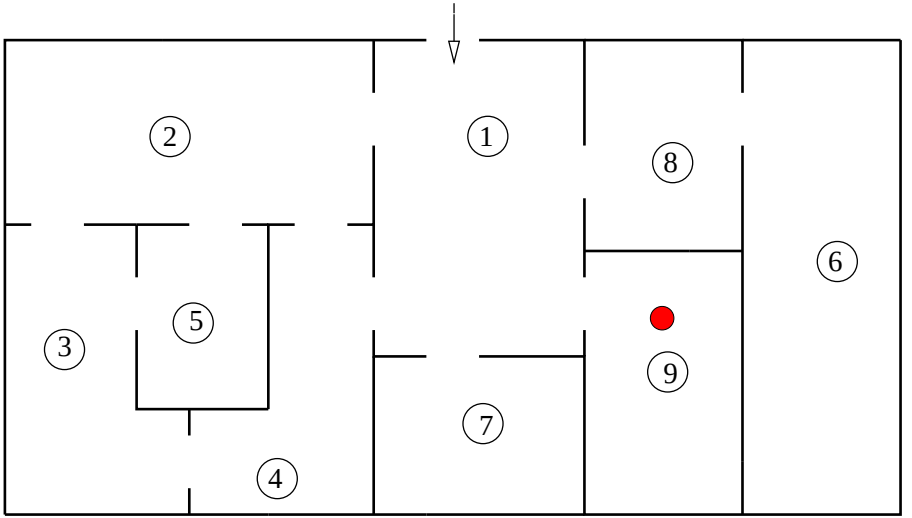
terem	1	2	3	4	5	6	7	8	9
honnan	−1	1	2	3	3		1		

utvonal 1 2 3 4 3 5 3 2 1 7



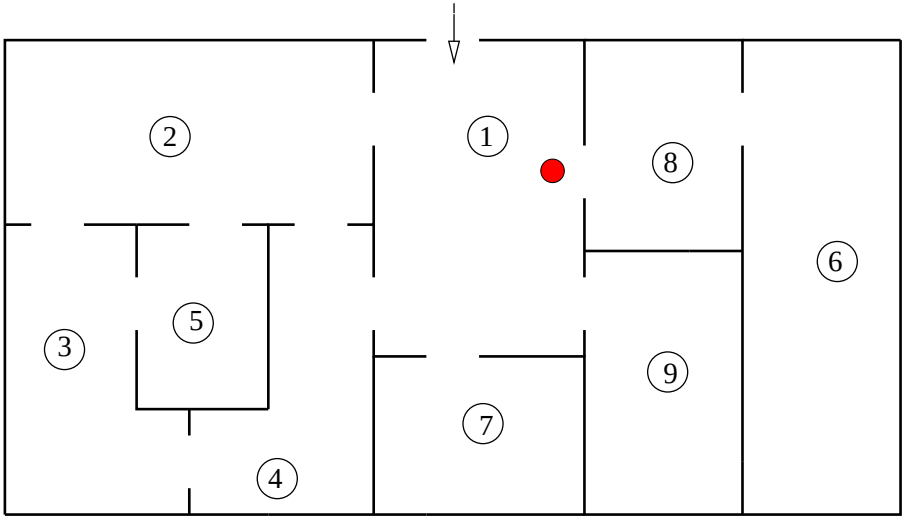
terem	1	2	3	4	5	6	7	8	9
honnan	−1	1	2	3	3		1		

utvonal 1 2 3 4 3 5 3 2 1 7 1



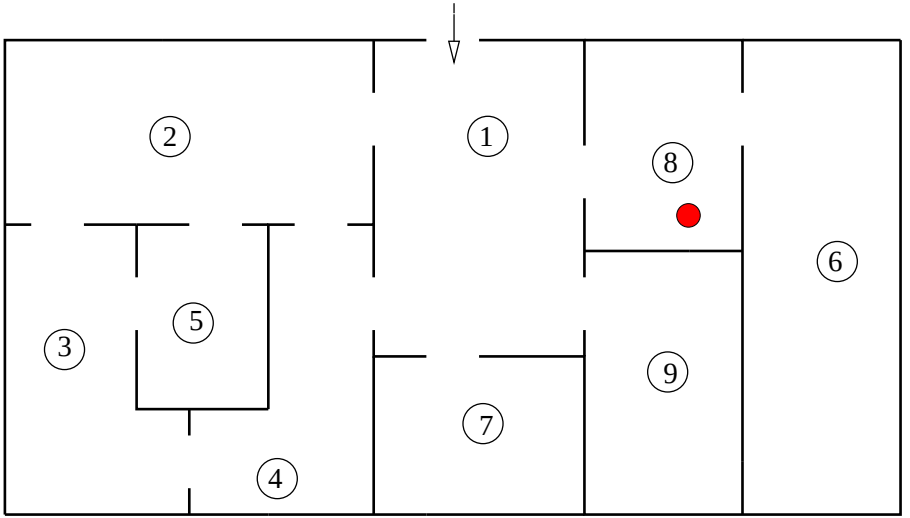
terem	1	2	3	4	5	6	7	8	9
honnan	−1	1	2	3	3		1		1

utvonal 1 2 3 4 3 5 3 2 1 7 1 9



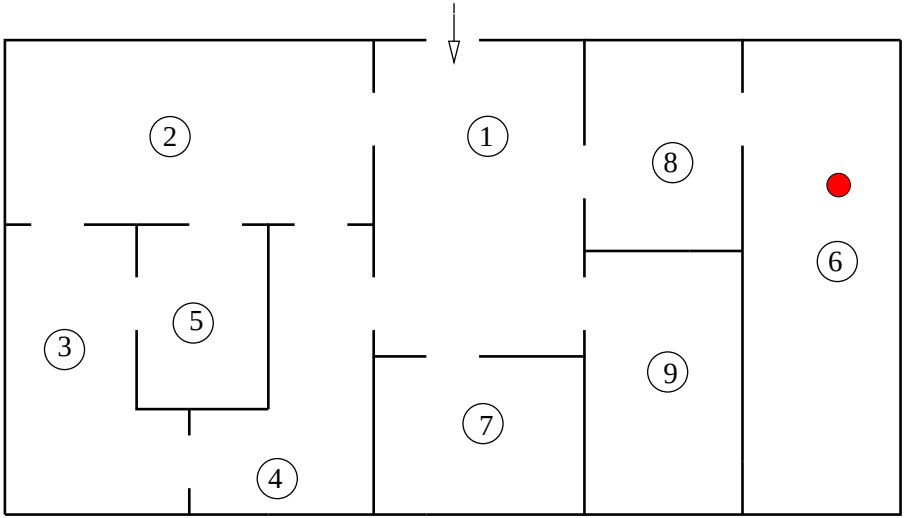
terem	1	2	3	4	5	6	7	8	9
honnan	−1	1	2	3	3		1		1

utvonal 1 2 3 4 3 5 3 2 1 7 1 9 1



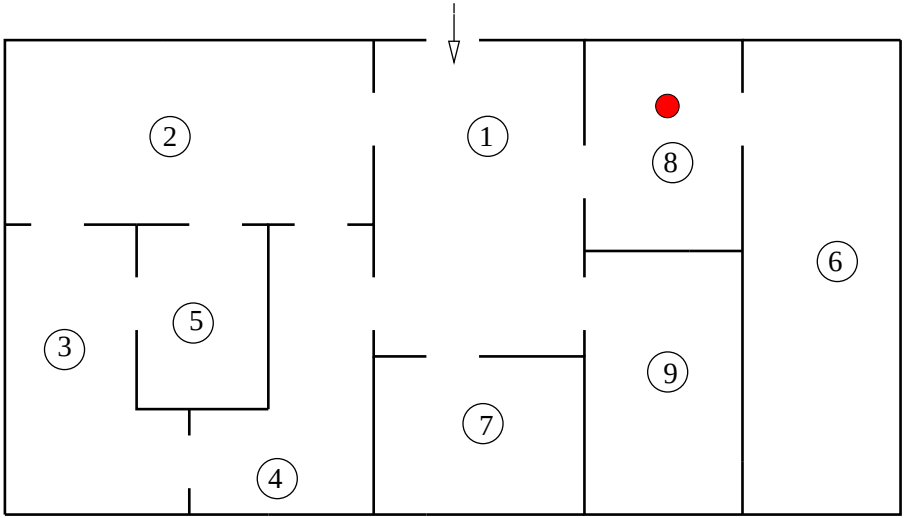
terem	1	2	3	4	5	6	7	8	9
honnan	−1	1	2	3	3		1	1	1

utvonal 1 2 3 4 3 5 3 2 1 7 1 9 1 8



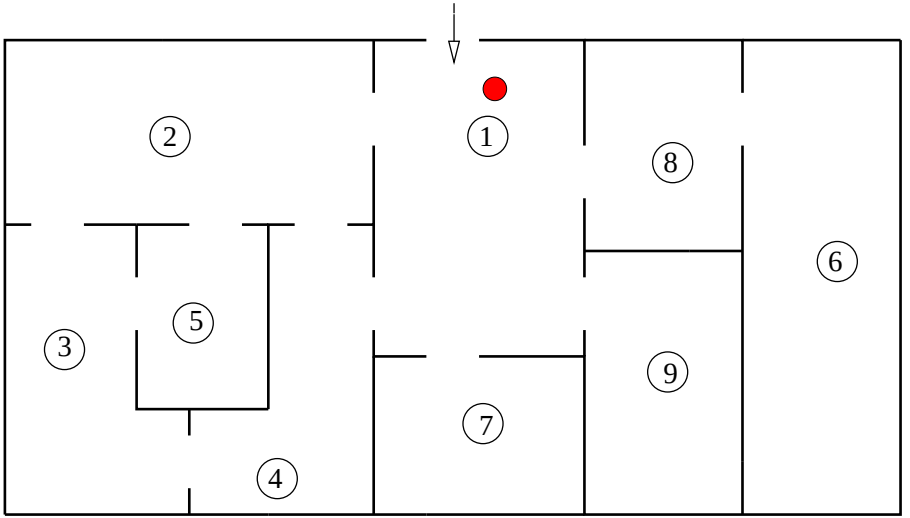
terem	1	2	3	4	5	6	7	8	9
honnan	−1	1	2	3	3	8	1	1	1

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



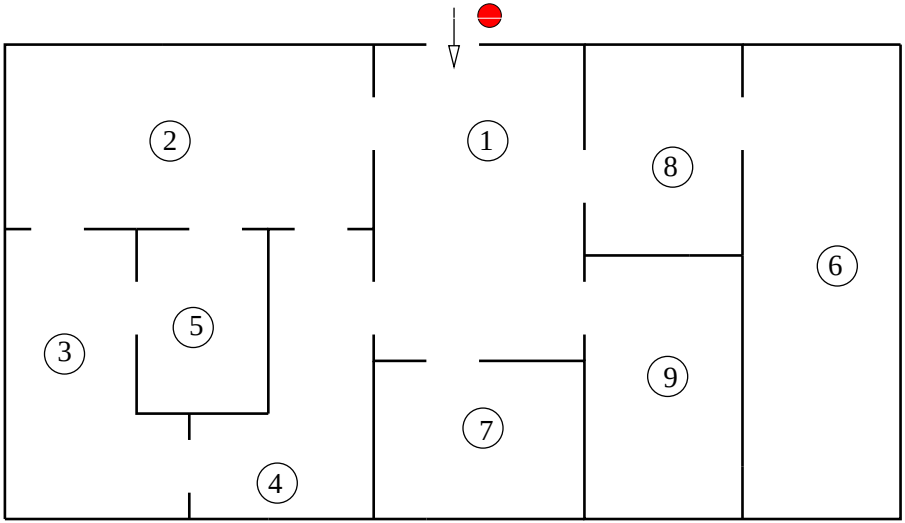
terem	1	2	3	4	5	6	7	8	9
honnan	−1	1	2	3	3	8	1	1	1

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



terem	1	2	3	4	5	6	7	8	9
honnan	−1	1	2	3	3	8	1	1	1

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



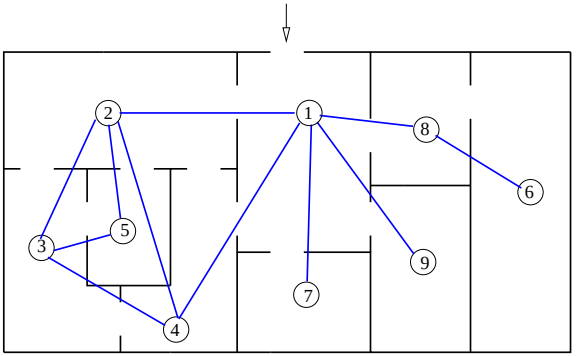
terem	1	2	3	4	5	6	7	8	9
honnan	−1	1	2	3	3	8	1	1	1

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

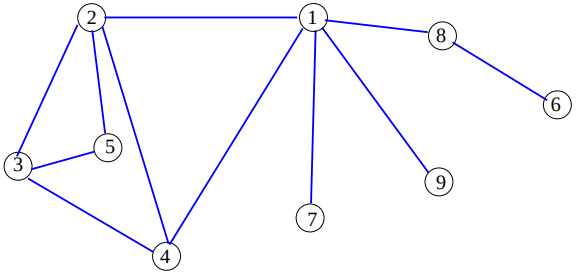
Az algoritmus:

```
1  itt := 1;          // {az aktuális terem}
2  Honnan[1] = 0;     //{az utcáról léptünk a főbejárat termébe}
3  while (itt != 0) do begin{//{amíg vissza nem értünk a bejárat/kijárathoz}
4     if (itt-nek van benemjárt szomszédos terme:hova) then begin
5        Honnan[hova] := itt;
6        itt := hova;          //{továbblépés a szomszédos terembe}
7     end else else            //{nincs benemjárt szomszédos terem}
8        itt := Honnan[itt];//{visszalépés}
9  end
```

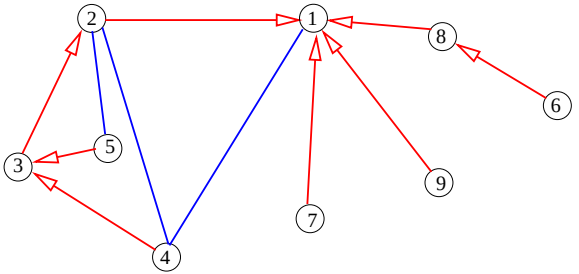
A gráf-modell: A gráf pontjai a termek, két terem között akkor és csak akkor van él, ha közöttük ajtó, amelyen át lehet menni egyikből a másikba (és fordítva).



29. ábra.



30. ábra.



31. ábra. A bejárás alkotta (mélységi) feszítőfa

5.3. Nemrekurzív megvalósítás

A gráf élhalmaz-tömb ábrázolását használva.

```
1  begin{Program}
2    Beolvas;
3    for i:=1 to n do begin
4      Honnan[i]:=0;
5      Szomszed[i]:=0;
6    end;
7    itt:=1;
8
9    while itt>0 do begin
10     write(itt,'␣');
11     repeat                          {benemjárt szomszéd keresése}
12       inc(Szomszed[itt]);
13     until (Szomszed[itt]>Fok[itt])or(Honnan[G[itt,Szomszed[itt]]]=0);
14     if Szomszed[itt]<=Fok[itt] then begin {van benemjárt szomszéd}
15       hova:=G[itt,Szomszed[itt]];
16       Honnan[hova]:=itt;            {itt–ből lépünk hova–ba}
17       itt:=hova;                    {itt–>hova átlépés}
18     end else                        {nincs benemjárt szomszéd}
19       itt:=Honnan[itt];             {visszalépés}
20   end{while};
21 end.
```

5.4. Rekurzív megvalósítás

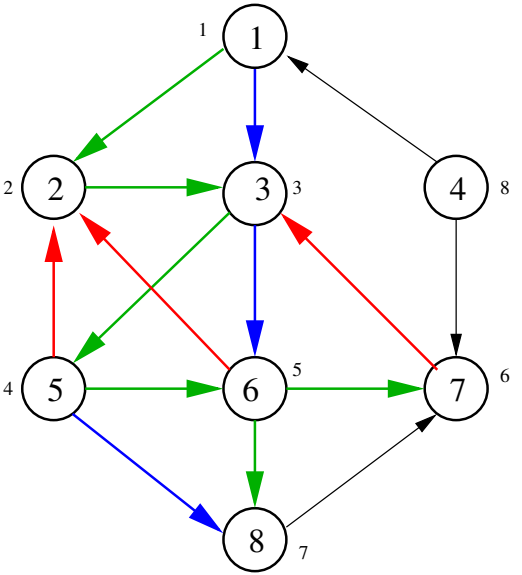
```
1  procedure MelyBejar(p:integer);
2  {Global: G, Fok, Apa}
3  var
4    i,q:integer;
5  begin
6    write('␣',p);
7    for i:=1 to Fok[p] do begin
8      q:=G[p,i];                {p->q él ?}
9      if Apa[q]<0 then begin    {q-ban még nem jártunk}
10       Apa[q]:=p;              {p-ből jöttünk q-ba}
11       MelyBejar(q);           {q-ból elérhetők bejárása}
12       write(KiF,'␣',p);       {vissza kell menni p-be}
13     end;
14   end{for i};
15 end{MelyBejar};
```

5.5. Élek osztályozása

Módisítsuk a mélységi bejárást úgy, hogy színezzük a bejárás során a gráf pontjait: kezdetben minden pont színe legyen Fehér, amikor először elérünk egy pontot, akkor színezzük Szürkére, amikor befejeztük a pontpol elérhető pontok bejárását, akkor színezzük a pontot Feketére.

```
1  procedure MelyBejar(p:integer);
2  {Global: G, Fok, Szin, Apa}
3  var
4    i,q:integer;
5  begin
6    Szin[p]:=Szurke;
7    for i:=1 to Fok[p] do begin
8      q:=G[p,i];                {p->q él ?}
9      if Szin[q]=Feher then begin {p->q faél}
10       Apa[q]:=p;              {p-ből jöttünk q-ba}
11       MelyBejar(q);           {q-ból elérhetők bejárása}
12     end else if Szinq]=Szurke then begin{p->q visszaél}
13
14     end else begin{p-q előreél vagy keresztél}
15
16     end;
17   end{for i};
18   Szin[p]:=Fekete;
19 end{MelyBejar};
```

Az $p \to q$ él vizsgálatakor, $Szin[q]$ alapján az osztályozás:
$Szin[q] = Feher$ akkor az él faél,
$Szin[q] = Szurke$ akkor az él visszaél,
$Szin[q] = Fekete$ és p-t előbb értük el, mint q-t, akkor az él előreél,
$Szin[q] = Feher$ és q-t előbb értük el, mint p-t, akkor az él keresztél.



32. ábra. Élek osztályozása: zöld: faél, piros: visszaél, kék: előreél, fekete: keresztél

Állítás: Minden G irányítatlan gráf bármely mélységi bejárása során minden él vagy faél, vagy visszél lesz.
Bizonyítás: Legyen (p,q) tetszőleges él a gráfban és tegyük fel, hogy a p pontot értük el előbb a bejárás során. q biztosan elérhető a p pontból, ezért a bejárás során mind a $p \to q$ mind a $q \to p$ élet vizsgáljuk. Ha a $p \to q$ élet vizsgáljuk előbb, akkor q színe Fehér, tehát faél lesz, ha a $q \to p$-t vizsgáljuk előbb, akkor, mivel p színe Szurke, az él visszaél lesz.

5.6. Topologikus rendezés

Egy $G == V,E)$ irányított gráf topologikus rendezésén a gráf pontjainak egy olyan

$$p_1, ..., p_n$$

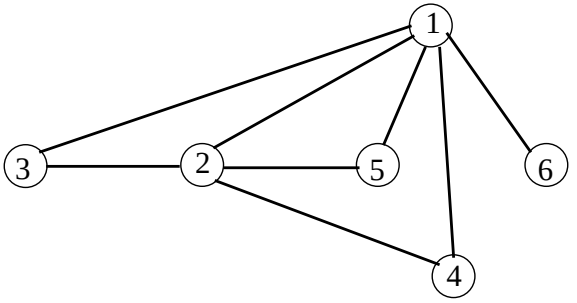
felsorolása, hogy bármely $u \to v \in E$ él $u = p_i$ és $v = p_j$ esetén $i < j$. (Tehát u előbb szerepel a felsorolásban, mint v).
Állítás: A G gráfnak akkor és csak akkor van topologikus rendezése, ha G körmentes.
Állítás: A G gráfnak akkor és csak akkor van topologikus rendezése, ha G bármely mélységi bejárása során nincs visszaél.
Állítás: Ha a mélységi bejárás során a pontokat a befejezési idő szerint (amikor a színét Fekete-re állítjuk) fordított sorrendben leírjuk, akkor egy topologikus rendezést kapunk.
Valóban, amikor a $Szin[p] := Fekete$ utasítást végrehajtjuk, akkor az összes olyan pontot már leírtuk a sorba, amelyek p-ből elérhetők.

```
1  procedure TopRend(const G:Graf; const Fok:array of longint; n:longint;
2      var T:array of longint; var Van:boolean);
3  var
4    m:longint;
5    procedure MelyBejar(p:longint);
6    {Global: G, Fok, Szin, m, T}
7    var
8      i,q:longint;
9    begin{MelyBejar};
10     Szin[p]:=Szurke;
11     for i:=1 to Fok[p] do begin
12       q:=G[p,i];              {p->q él ?}
13       if Szin[q]=Feher then begin {p->q faél}
14       MelyBejar(q);           {q-ból elérhetők bejárása}
15       if m<0 then exit;
16       end else if Szinq]=Szurke then begin{p->q visszaél}
17         m:=-1; exit;
18       end;
19     end{for i};
20     Szin[p]:=Fekete;
21     T[m]:=p;
22     dec(m);
23   end{MelyBejar};
24 begin {TopRend};
25   m:=n;
```

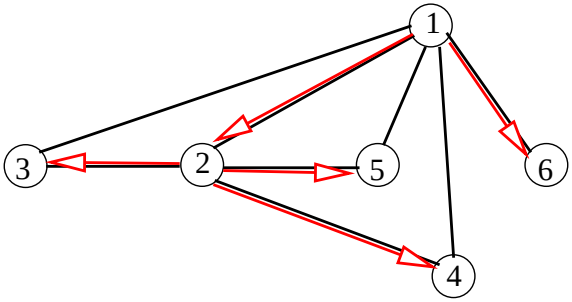
```
26    for i:=1 to n do Szin[i]:=Feher;
27    for i:=1 to n do if Szin[i]=Feher then
28     MelyBejar(i);
29    Van:=m=0;
30  end{TopRend};
```

5.7. Utcaseprő

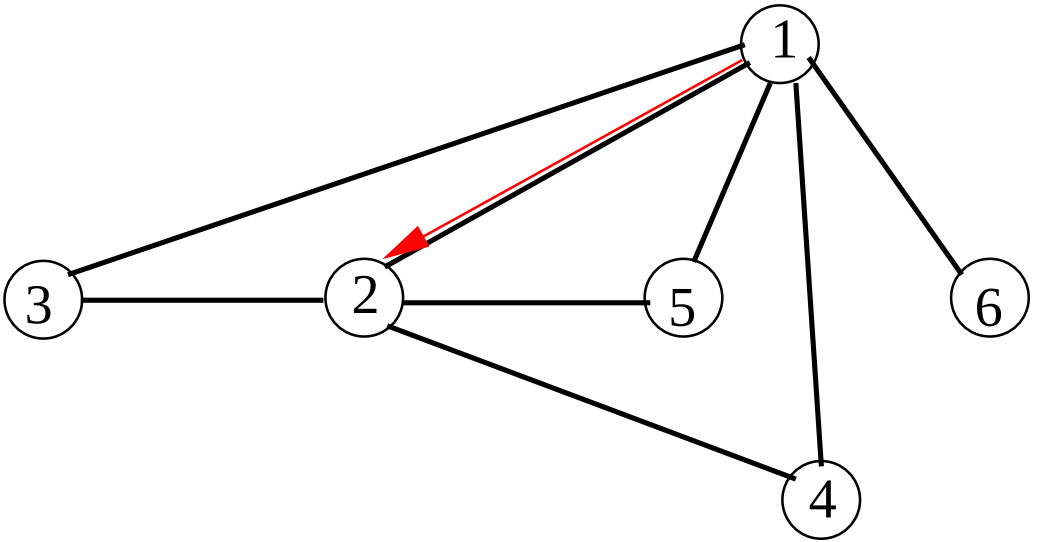
A város elhatározta, hogy minden nap végigmegy a város utcáin az egyetlen utcaseprő gépe. A város minden utcája kétirányú, és az utcaseprőnek is be kell tartania a közlekedési szabályokat. Adjunk olyan útvonalat, amelyen haladva az utcaseprő minden utcában pontosan egyszer halad végig mindkét irányban.

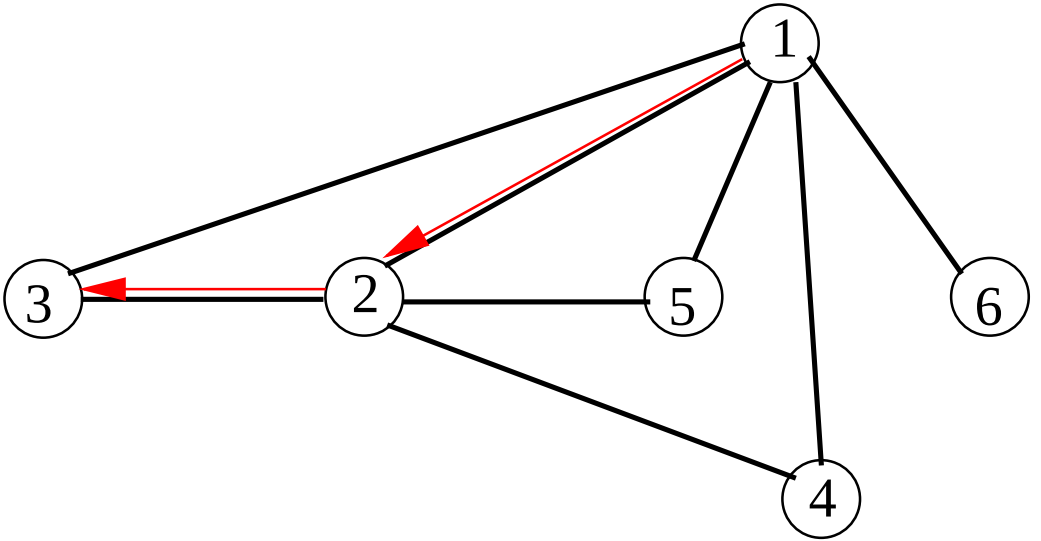


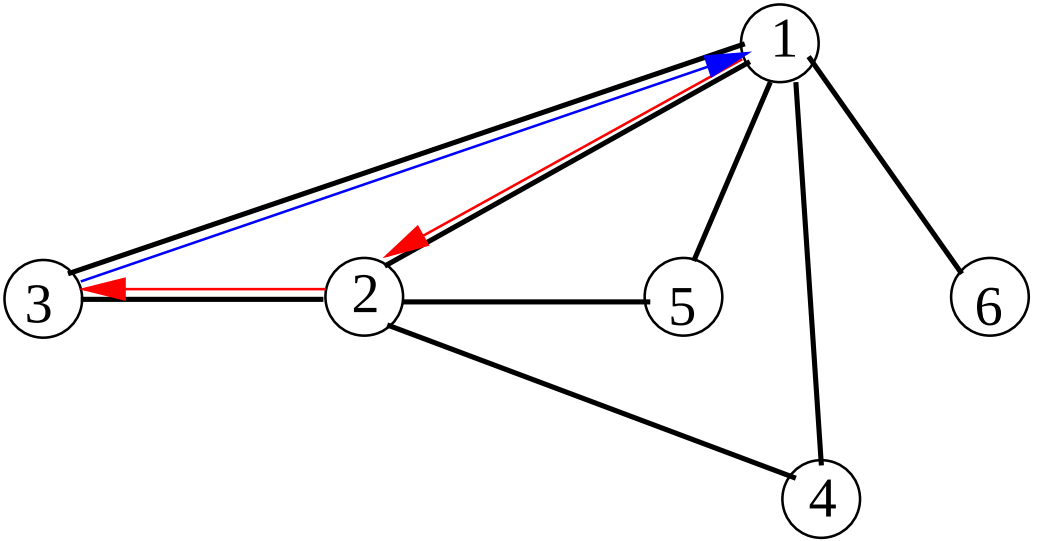
33. ábra.

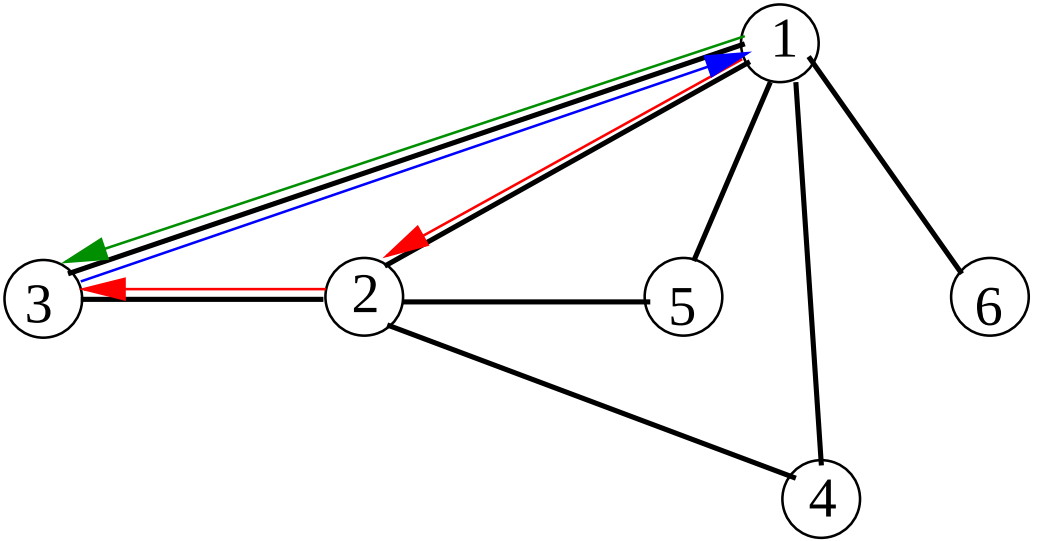


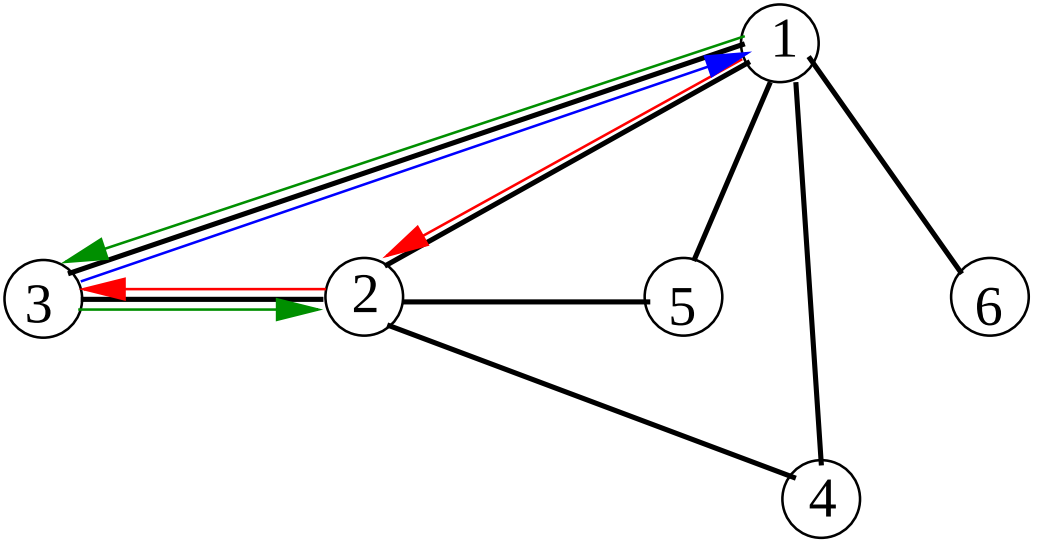
34. ábra. Mélységi felszítőfa alapján az útvonal: 1-2-3-1-3-2-4-1-4-2-5-1-5-2-1-6-1

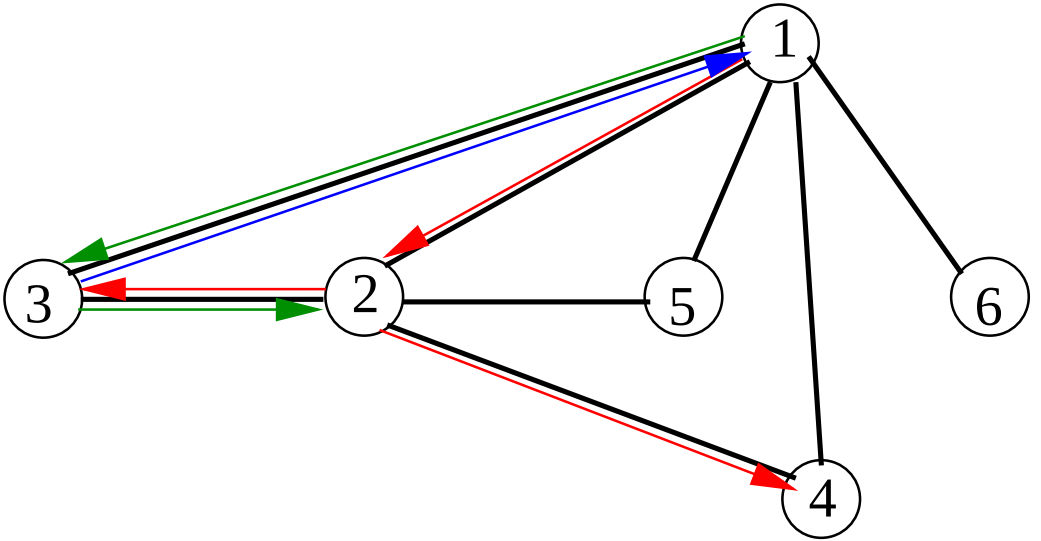

1
3
2
5
6
4

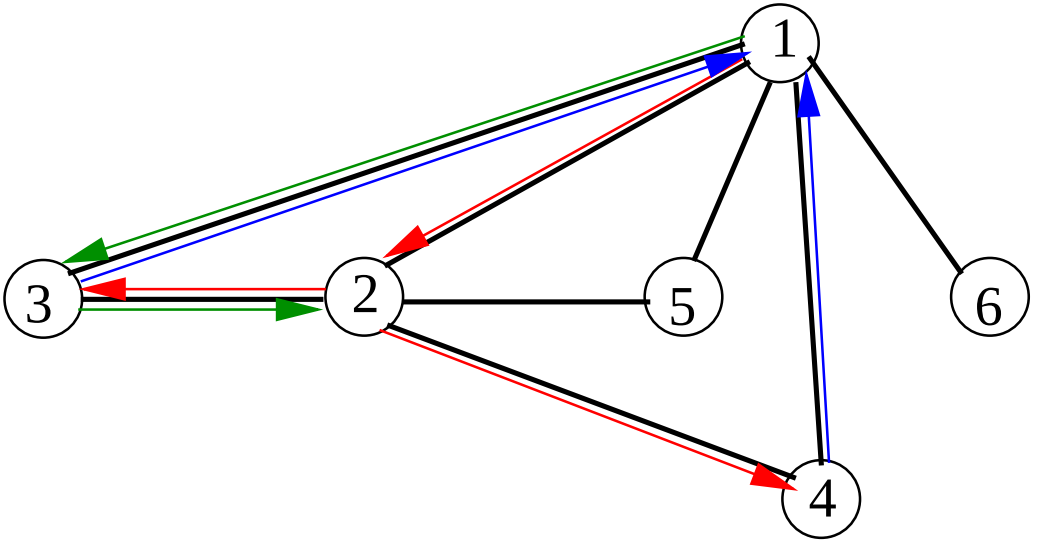

1
3
2
5
6
4

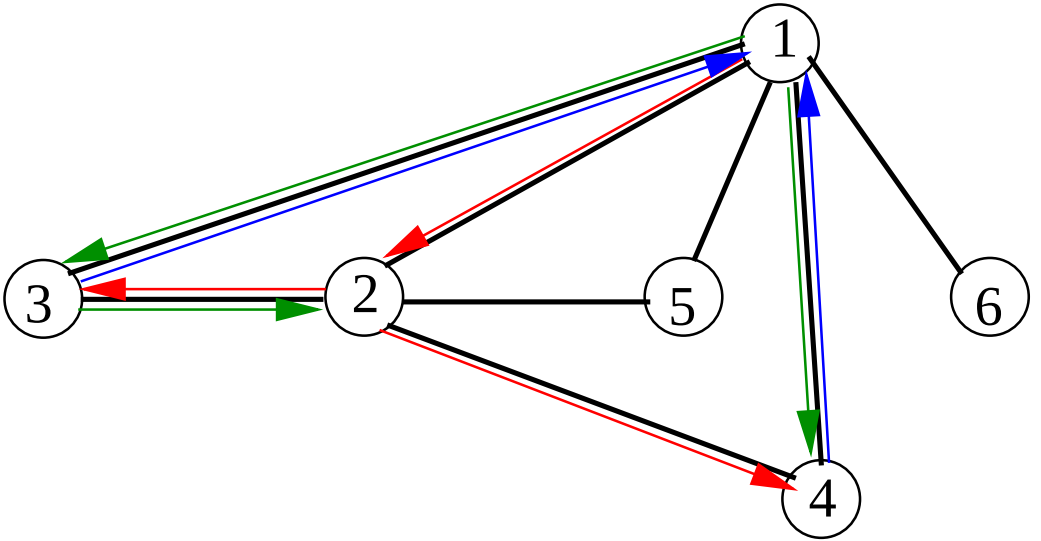

1
3
2
5
6
4

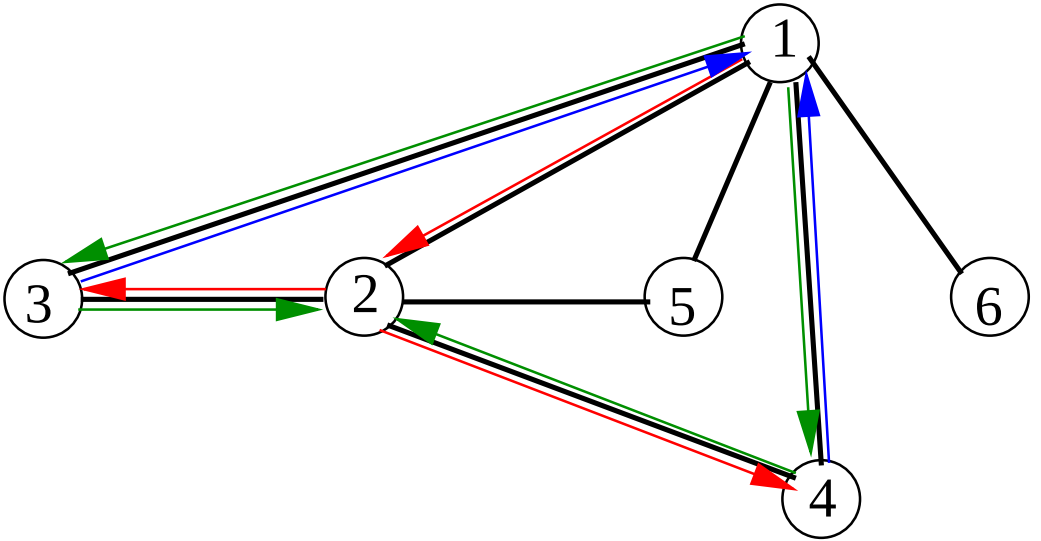

1
3
2
5
6
4

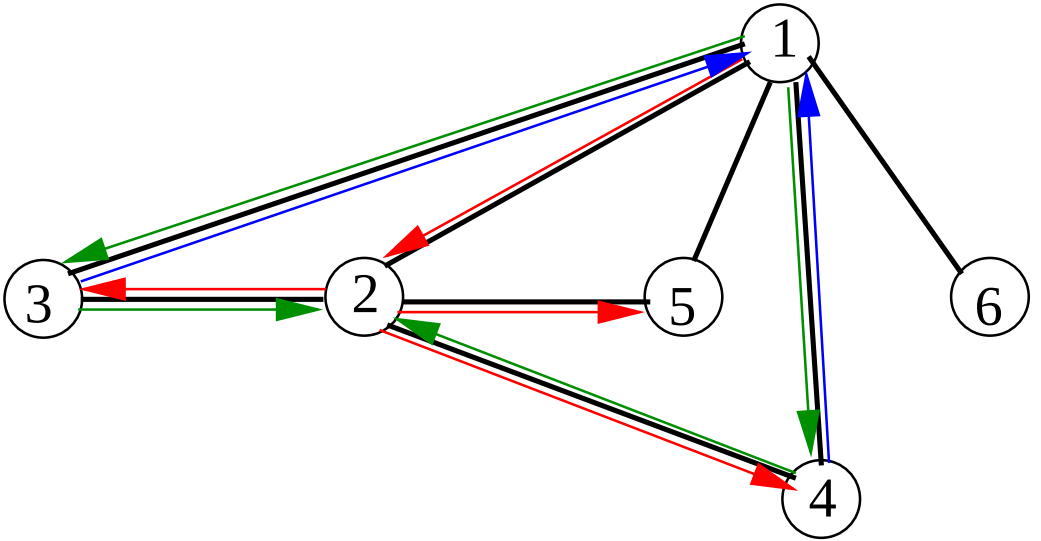

1
3
2
5
6
4

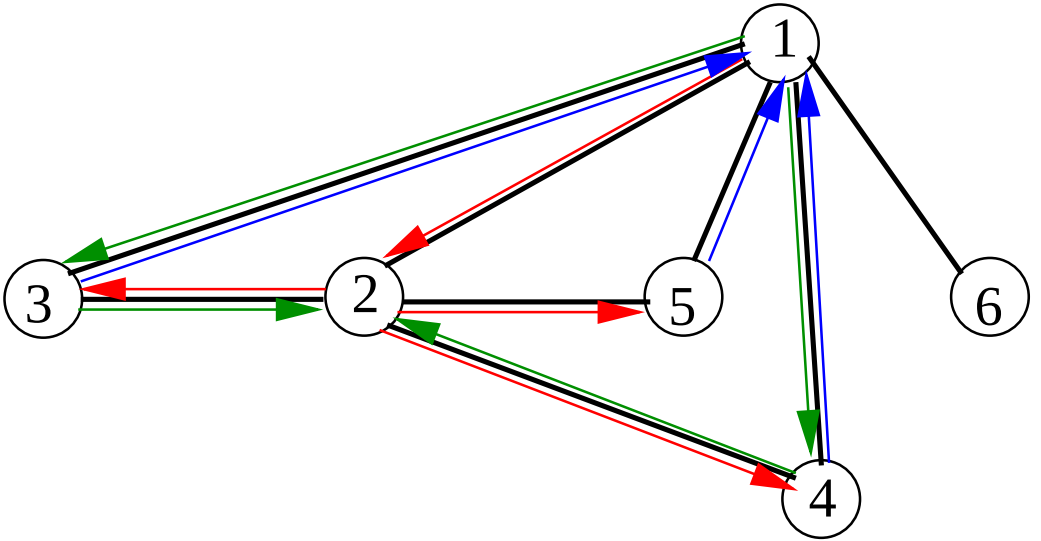

1
3
2
5
6
4

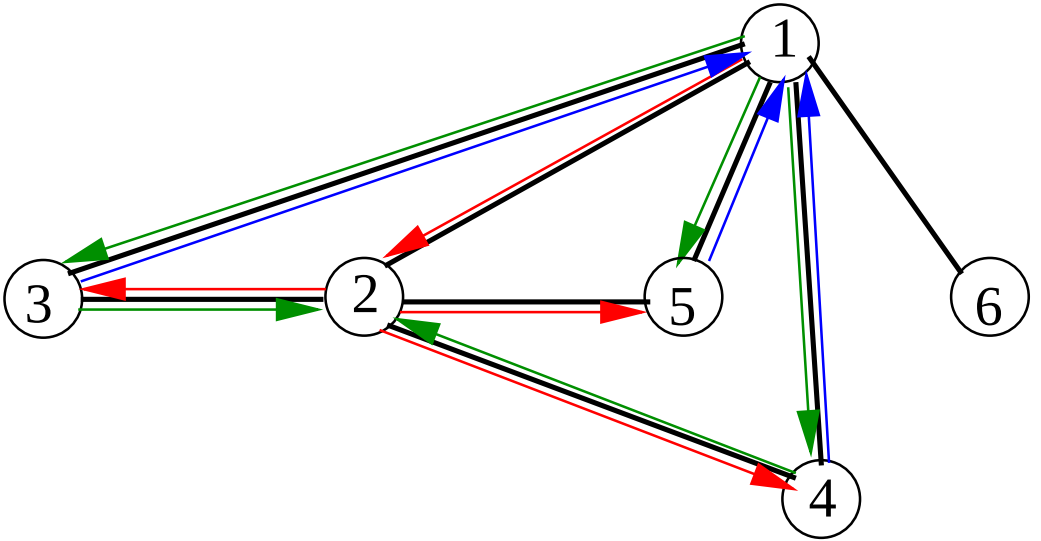

1
3
2
5
6
4

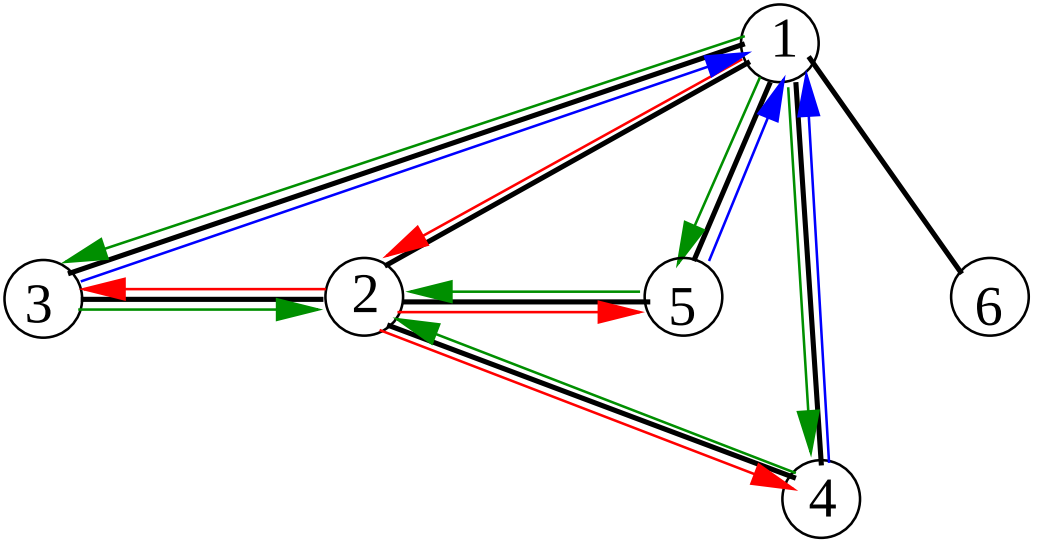

1
3
2
5
6
4

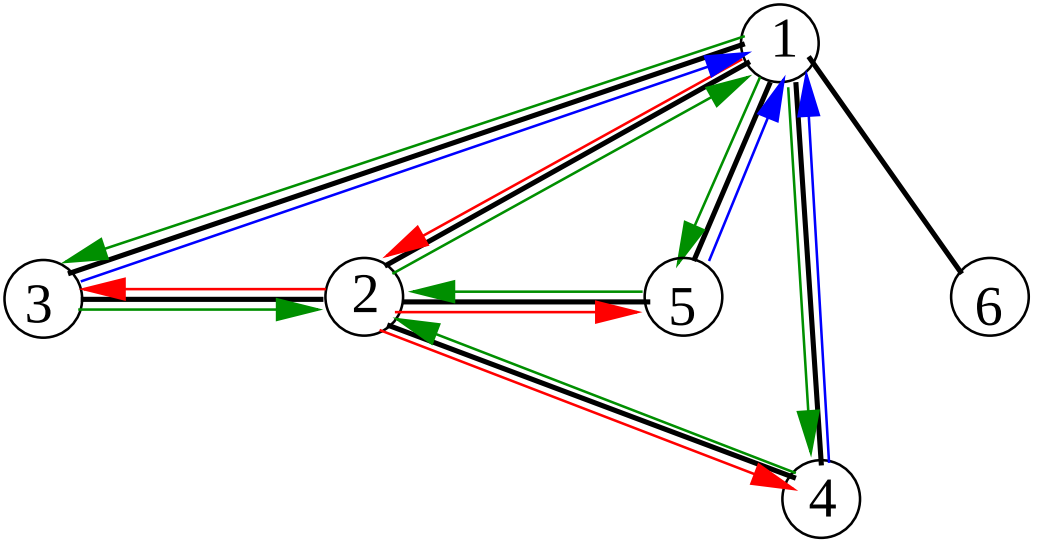

1
3
2
5
6
4

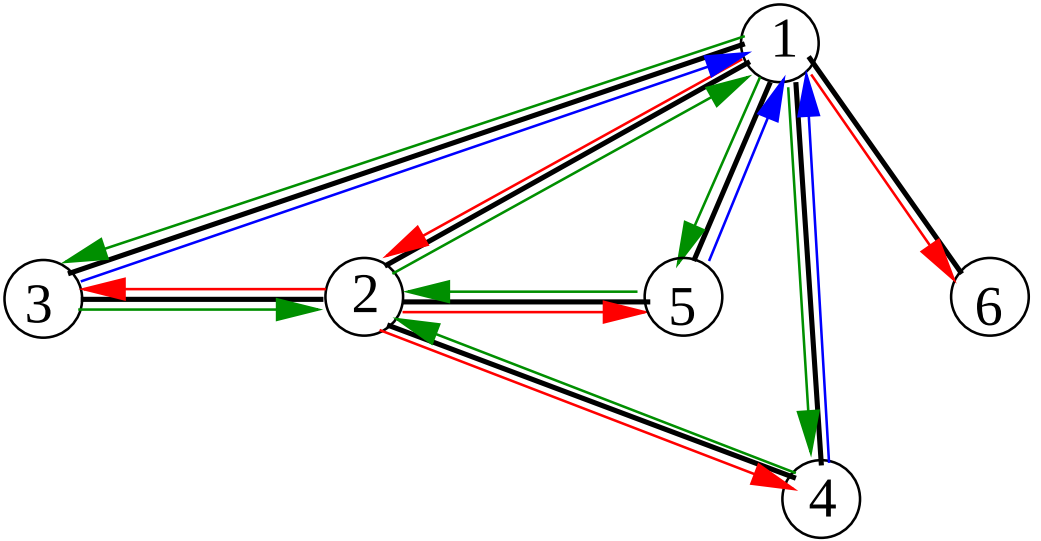

1
3
2
5
6
4

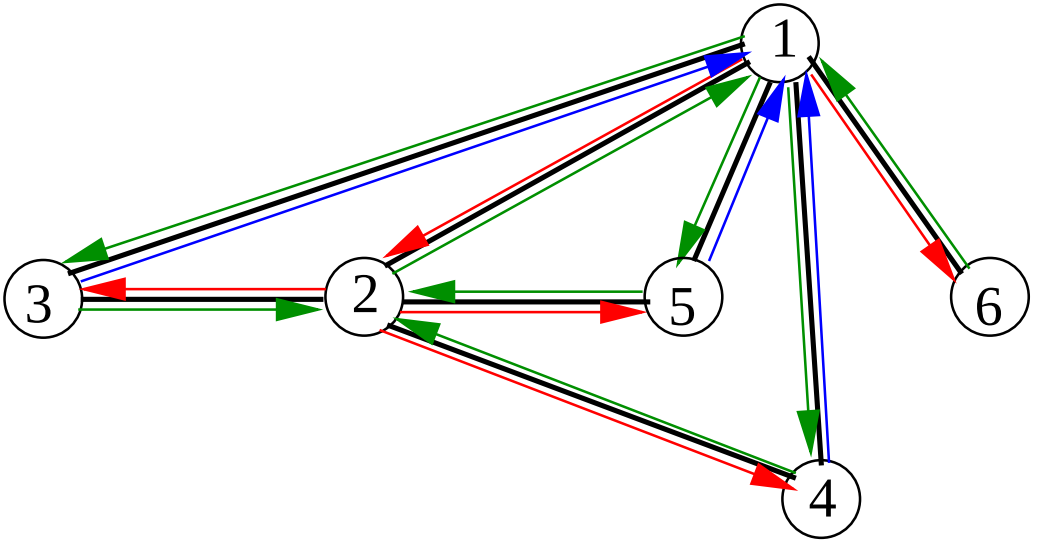

1
3
2
5
6
4


1
3
2
5
6
4


1
3
2
5
6
4


1
3
2
5
6
4


1
3
2
5
6
4


1
3
2
5
6
4

```
1  program Utcasepro;
2  const
3    MaxN=500;
4  type
5    Paletta=(Feher,Szurke,Fekete);
6  var
7    N,i:longint;
8    G:array[1..MaxN, 1..MaxN] of longint;
9    Apa,Fok:array[1..MaxN] of longint;
10   Szin:array[1..MaxN] of Paletta;
11 procedure BeOlvas;
12   var i,x:longint;
13     BeF:Text;
14   begin
15     ReadLn(N);
16     for i:=1 to n do begin
17       Fok[i]:=0;
18       Read(x);
19       while (x<>0) do begin
20         inc(Fok[i]);
21         G[i,Fok[i]]:=x;
22         read(x);
23       end;
24     end;
25   end {Beolvas};
```

```
26 procedure MelyBejar(p:longint);
27 {Global: G, Fok, Apa}
28 var
29   i,q:longint;
30 begin
31   write('␣',p);
32   Szin[p]:=Szurke;
33   for i:=1 to Fok[p] do begin
34     q:=G[p,i];
35     if Szin[q]=Feher then begin {q-ban még nem jártunk}
36       Apa[q]:=p;                {p-ből jöttünk q-ba}
37       MelyBejar(q);             {q-ból elérhetők bejárása}
38       write('␣',p);             {vissza kell menni p-be}
39     end else if (Szin[q]=Szurke)and(q<>Apa[p]) then begin
40       write('␣',q,'␣',p);         {->q->p}
41     end;
42   end{for i};
43   Szin[p]:=Fekete;
44 end{MelyBejar};
45
46 begin{Program}
47   Beolvas;
48   for i:=1 to n do Szin[i]:=Feher;
49   MelyBejar(1);
50 end.
```

6. Euler-séta, Euler-kör

6.1. definíció. *A $G = (V,E)$ (irányított vagy irányítatlan) gráfban Euler-séta olyan séta, amely a gráf minden élét pontosan egyszer tartalmazza.*

6.2. definíció. *A $G = (V,E)$ (irányított vagy irányítatlan) gráfban Euler-kör olyan séta, amely a gráf minden élét pontosan egyszer tartalmazza, és a séta első és utolsó pontja megegyezik.*

6.3. tétel. *A $G = (V,E)$ irányított gráfban akkor és csak akkor van Euler-séta, ha van G-ben olyan pont, amelyből bármely más pont elérhető, és vagy minden pont kifoka megegyezik befokával (ekkor a séta körséta),*

$$(\forall v \in V)(KiFok(v) = BeFok(v)) \tag{3}$$

vagy van olyan $a \in V$ és $b \in V$ pont, hogy

$$KiFok(a) = BeFok(a) + 1 \text{ és } BeFok(b) = KiFok(b) + 1 \tag{4}$$

$$(\forall v \in V)(v \neq a \wedge v \neq b \Rightarrow KiFok(v) = BeFok(v)) \tag{5}$$

6.4. tétel. *A $G = (V, E)$ irányítatlan gráfban akkor és csak akkor van Euler-séta, ha összefüggő, és*

$$(\forall v \in V)(Fok(v) \text{ páros}) \tag{6}$$

(ekkor a séta körséta) vagy van olyan $a \in V$ és $b \in V, a \neq b$ pont van, hogy

$$Fok(a) \text{ páratlan} \tag{7}$$

$$Fok(b) \text{ páratlan} \tag{8}$$

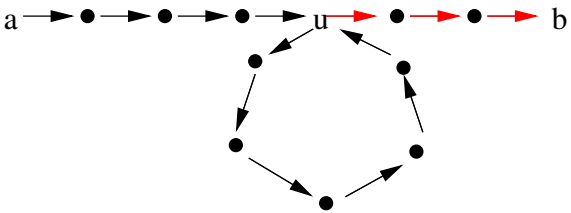
$$(\forall v \in V)(v \neq a \wedge v \neq b \Rightarrow Fok(v) \text{ páros}) \tag{9}$$

Bizonyítás irányított gráfra

Tfh. $KiFok(a) = BeFok(a) + 1$ és $BeFok(b) = KiFok(b) + 1$. Vegyünk egy olyan vak sétát, amely az a pontból indul, tetszőlegesen haladunk, töröljük azokat az éleket, amelyeken áthaladunk. Addig menjünk, ameddig olyan ponthoz érünk, amelyből már nem indul ki (benemjárt) út.

A séta biztosan a b ponban ér véget, mert a séta minden belső p pontjára $KiFok(p) = BeFok(p)$,

35. ábra.



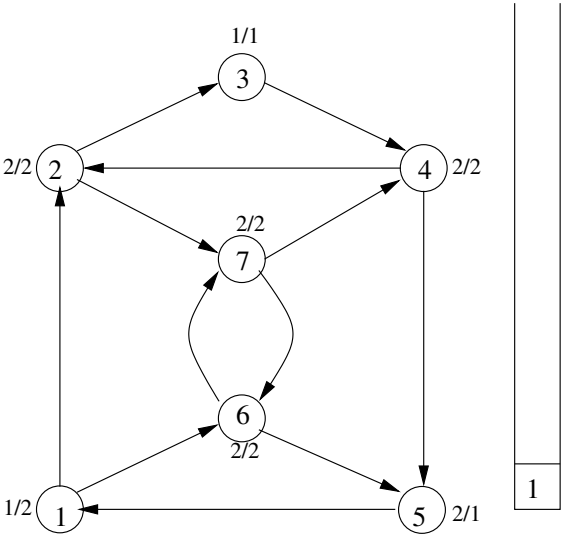
36. ábra.

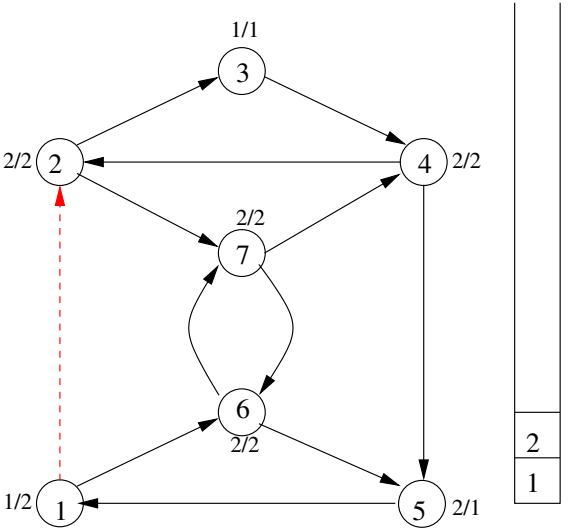
így ha ilyen pontba léptünk be, akkor abból ki is tudunk lépni.

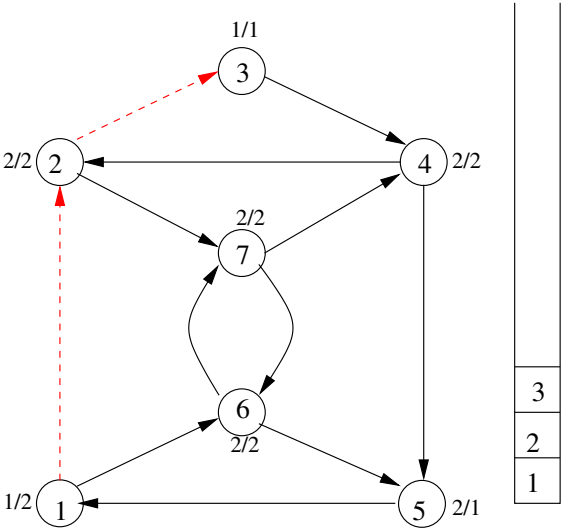
Legyen u egy (az utolsó) olyan pont az $a \rightsquigarrow b$ sétában, amelyből indul ki benem járt él. Ha nincs ilyen, akkor a séta a gráf minden élét tartalmazza.

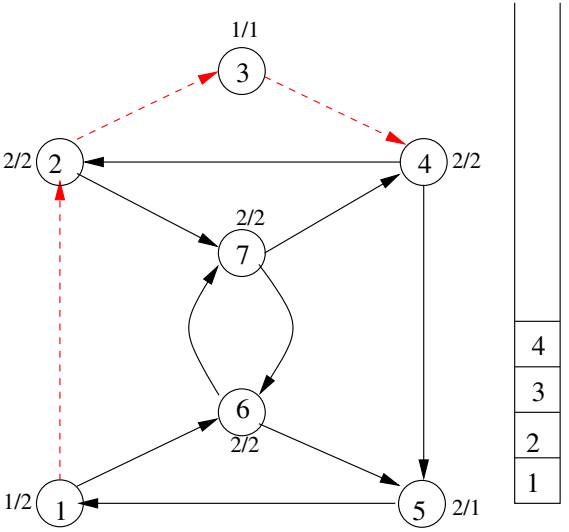
Vegyünk egy vak sétát, amely az u pontból indul. Ez biztosan az u pontban végződik. Kapcsoljuk be ezt az $a \rightsquigarrow b$ sétába.

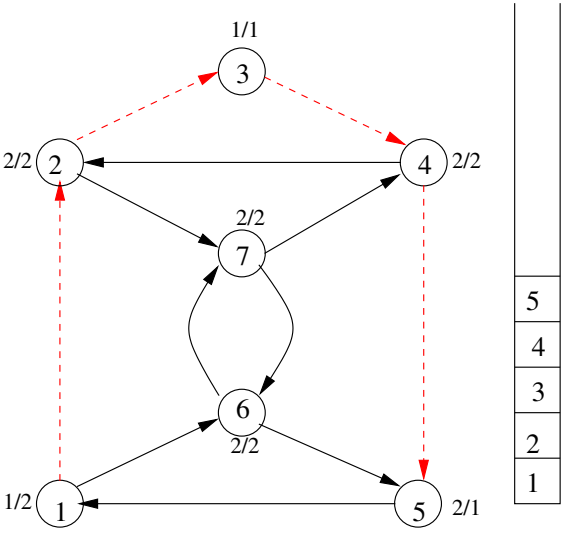
Folytassuk ezt mindaddig, amíg van benemjárt él.

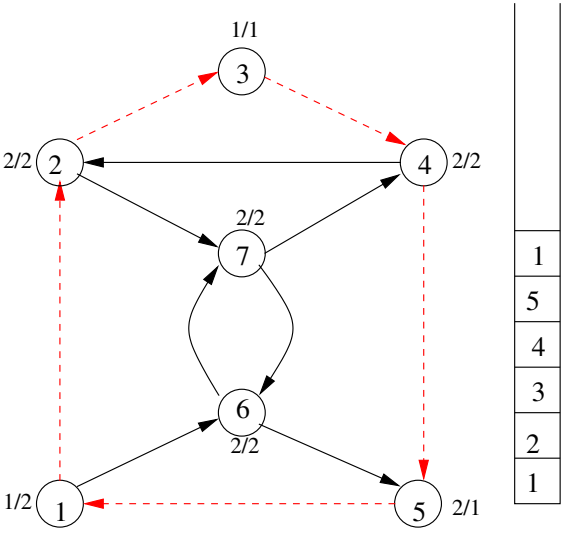

1/1
3
2/2
2
4
2/2
2/2
7
6
2/2
1/2
1
5
2/1
1

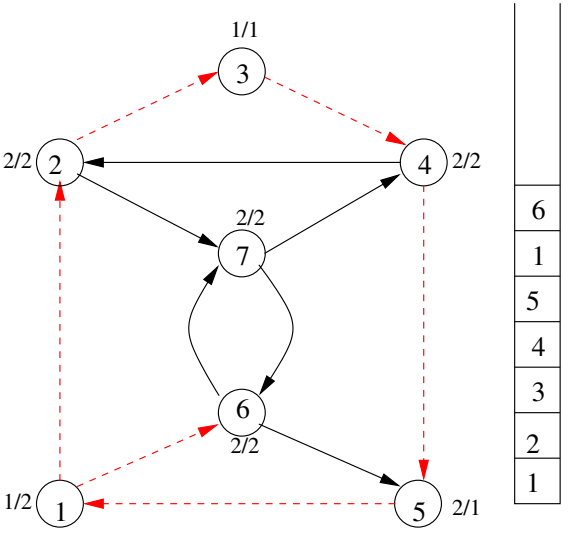

1/1
3
2/2
2
4
2/2
2/2
7
6
2/2
1/2
1
5
2/1
2
1

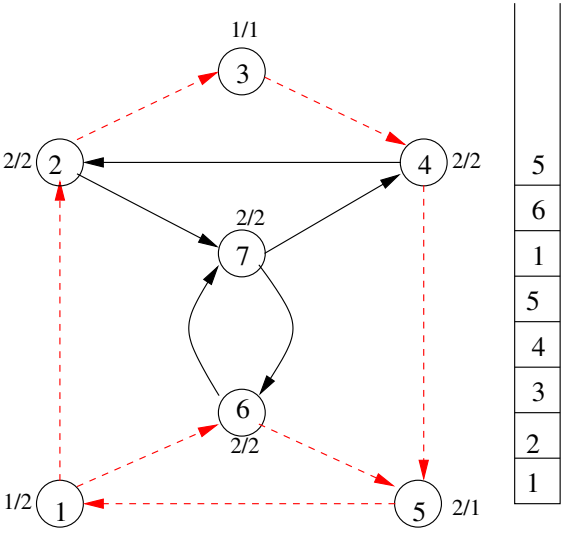

1/1
3
2/2
2
4
2/2
2/2
7
6
2/2
1/2
1
5
2/1
3
2
1

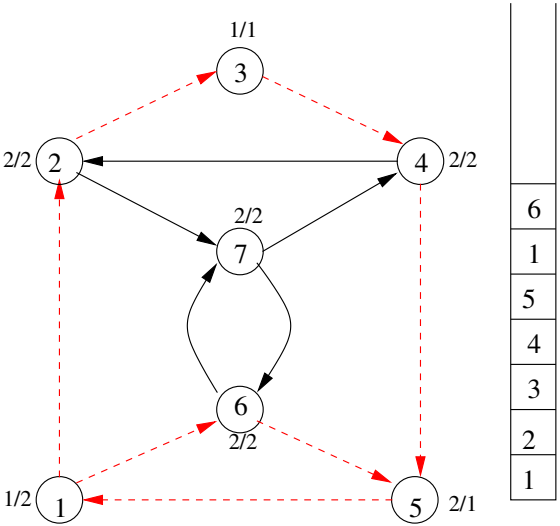

1/1
3
2/2
2
4
2/2
2/2
7
6
2/2
1/2
1
5
2/1
4
3
2
1

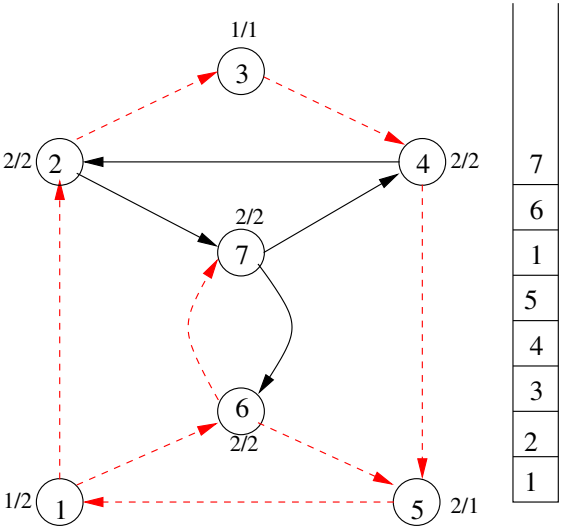

1/1
3
2/2
2
4
2/2
2/2
7
6
2/2
1/2
1
5
2/1
5
4
3
2
1

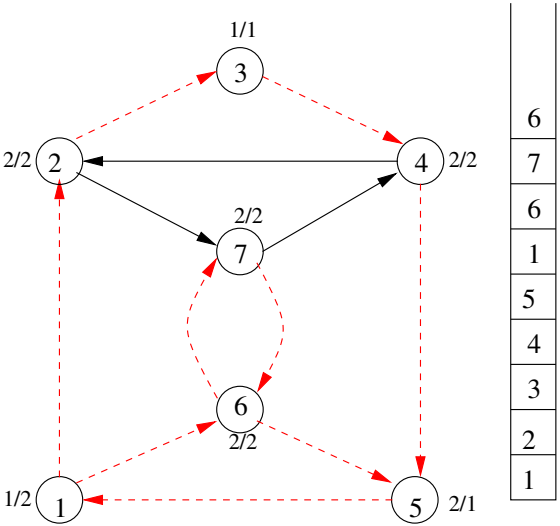

1/1
3
2/2
2
4
2/2
2/2
7
6
2/2
1/2
1
5
2/1
1
5
4
3
2
1

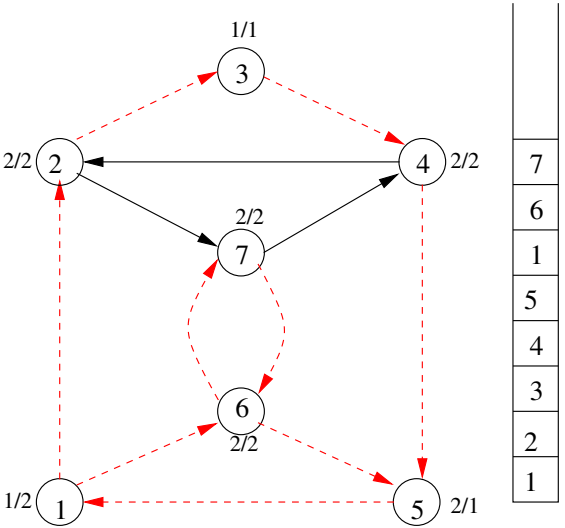

1/1
3
2/2
2
4
2/2
2/2
7
6
2/2
1/2
1
5
2/1
6
1
5
4
3
2
1


1/1
3
2/2
2
4
2/2
2/2
7
6
2/2
1/2
1
5
2/1
5
6
1
5
4
3
2
1

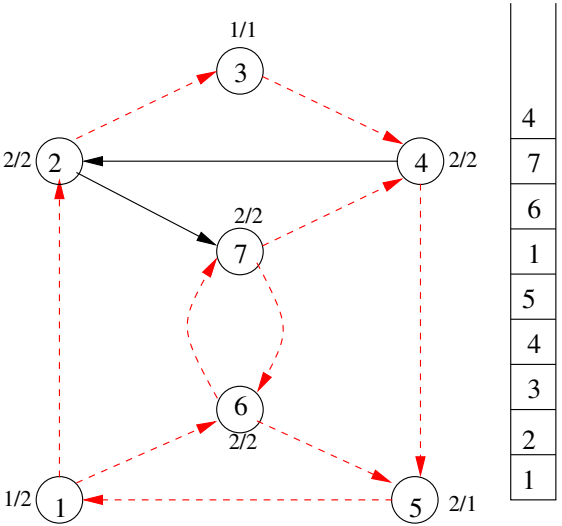

1/1
3
2/2
2
4
2/2
2/2
7
6
2/2
1/2
1
5
2/1
6
1
5
4
3
2
1


1/1
3
2/2
2
4
2/2
2/2
7
6
2/2
1/2
1
5
2/1
7
6
1
5
4
3
2
1

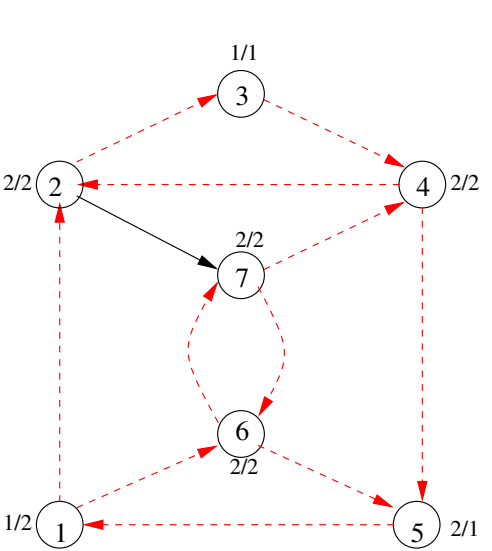

1/1
3
2/2
2
4
2/2
2/2
7
6
2/2
1/2
1
5
2/1
6
7
6
1
5
4
3
2
1



6 5

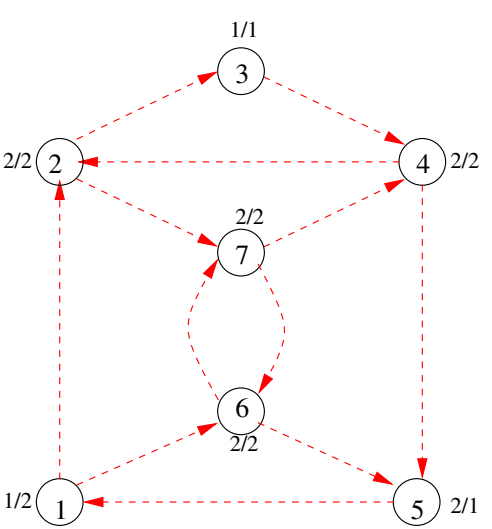


6 5



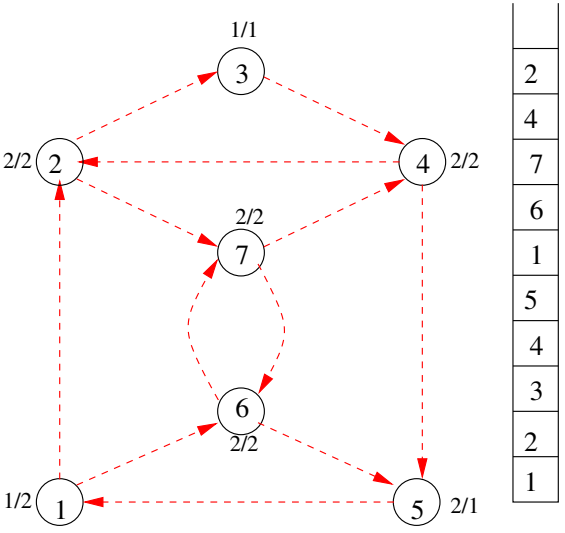
2
4
7
6
1
5
4
3
2
1

6 5

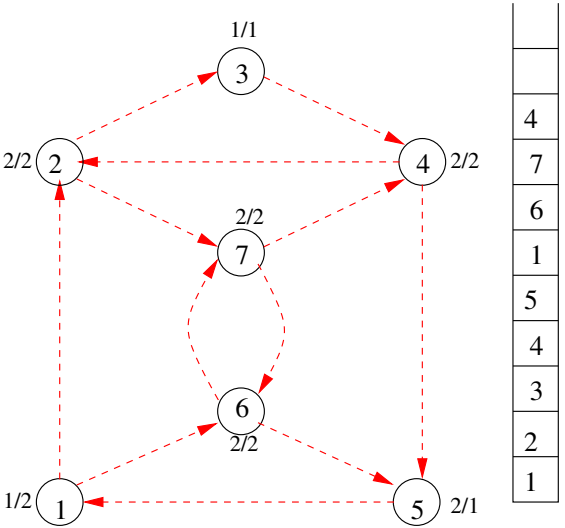


7
2
4
7
6
1
5
4
3
2
1

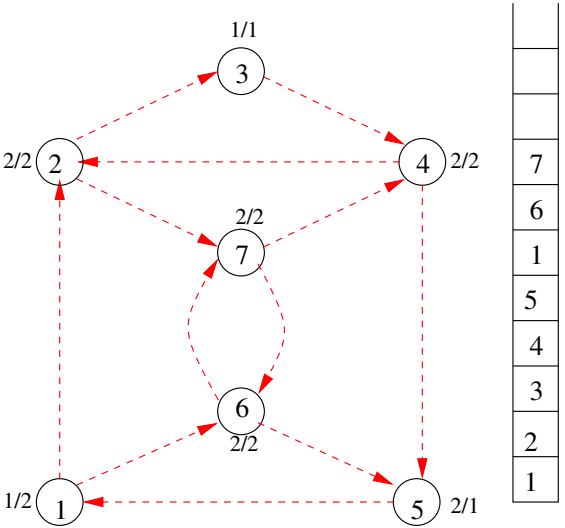
6 5



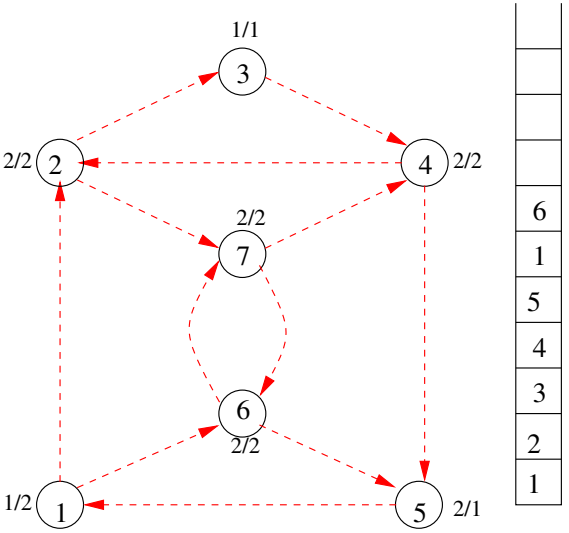
7 6 5



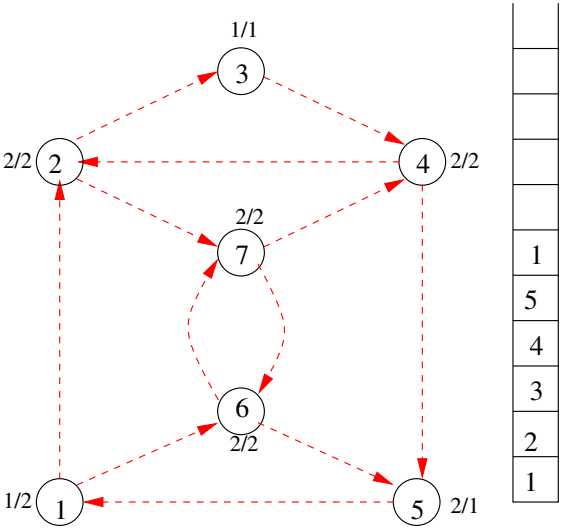
2 7 6 5



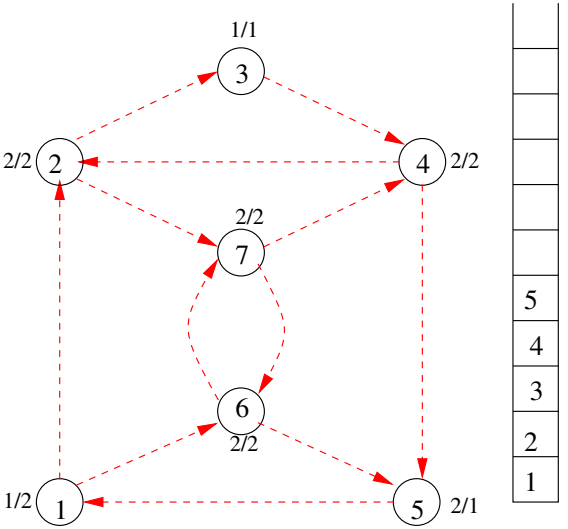
4 2 7 6 5



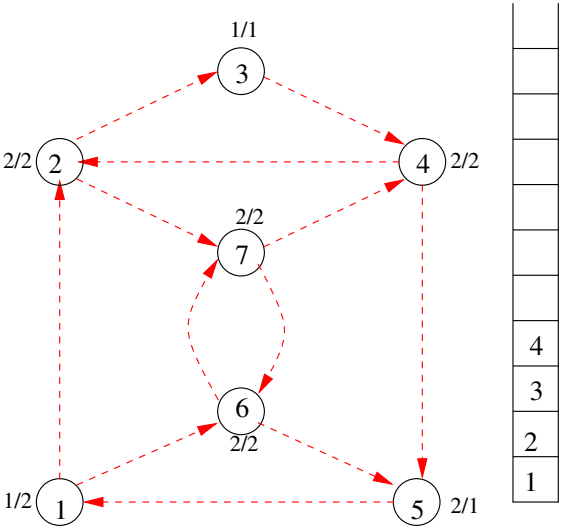
7 4 2 7 6 5



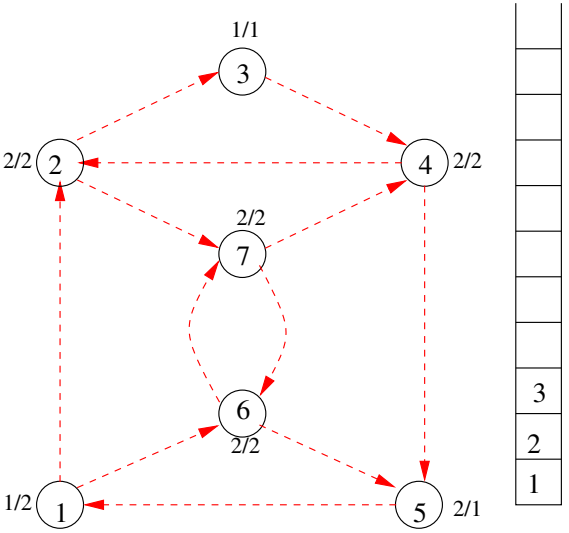
6 7 4 2 7 6 5



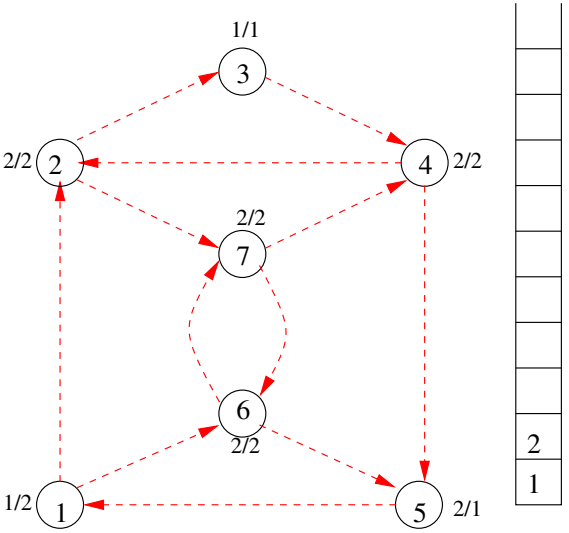
1 6 7 4 2 7 6 5



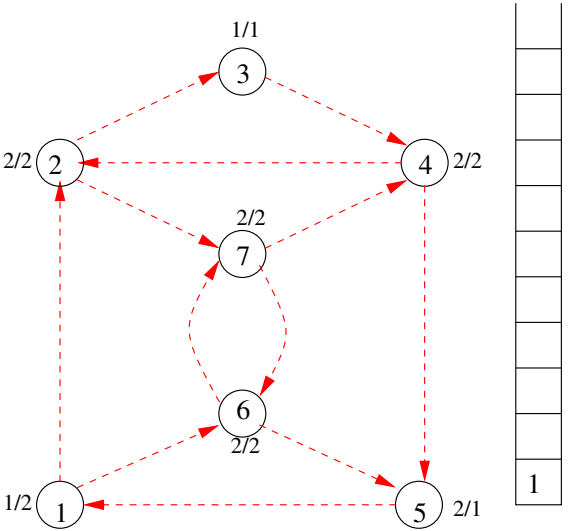
5 1 6 7 4 2 7 6 5



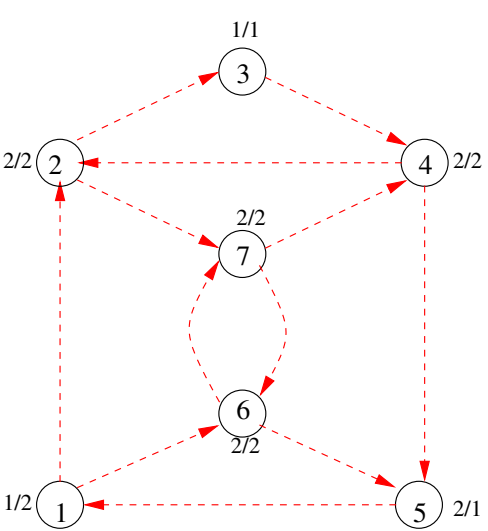
4 5 1 6 7 4 2 7 6 5



3 4 5 1 6 7 4 2 7 6 5



2 3 4 5 1 6 7 4 2 7 6 5



1 2 3 4 5 1 6 7 4 2 7 6 5

6.1. Euler-séta keresés irányított gráfra

```
1  Program EulerSeta; {Euler-séta keresés irányított gráfra}
2  Const
3    MaxP=10000;       {a pontok max. száma}
4    MaxM=100000;      {az élek max. száma}
5    Null=0;
6  Type
7    PontTip=0..MaxP;
8    Cella=Record
9             Pont:PontTip;  csat:0..MaxM;
10          End;
11   Graf=Array[1..MaxP] Of 0..MaxM; {a gráfábrázolás típusa }
12   Uttip=array[0..MaxM] of 0..MaxP;
13   VeremT=record teto:longint; Tar:array[1..MaxP] of longint end;
14 Var
15   N,                              {a pontok száma }
16   E:Longint;                      {az élek száma}
17   G:Graf;
18   KiFok,BeFok:Array[1..MaxP] of longint;
19   El:Array[1..MaxP] of Cella;  {az első aktív él: p->q=El[G[p]].pont}
20   Szabad:longint;                 {az első szabad cella}
21   Ut:UtTip;                       {a séta pontjai }
22   V:VeremT;
```

```
23 Procedure Beolvas;
24 Var BeF:Text;
25   u,v:PontTip;
26   i:Longint;
27   Guv:longint;
28 Begin
29   szabad:=1;
30   Assign(BeF, 'eulerkor.be'); Reset(BeF);
31   ReadLn(BeF, N, E);
32   For u:=1 To N Do Begin        {Inicializálás}
33     G[u]:=Null;
34     KiFok[u]:=0;
35     BeFok[u]:=0;
36   End;
37   For i:=1 To E Do Begin        {az input beolvasása}
38     ReadLn(BeF, u,v);
39     Guv:=szabad; inc(szabad);
40     El[Guv].pont:=v;  El[Guv].csat:=G[u];
41     G[u]:=Guv;
42     Inc(KiFok[u]);    Inc(BeFok[v]);
43   End{for i};
44  Close(BeF);
45 End{Beolvas};
```

```
46 Procedure KiIr;
47 Var
48   KiF:Text;
49   i:longint;
50 Begin
51   Assign(KiF,'eulerkor.ki'); Rewrite(KiF);
52   for i:=1 to E do
53     write(KiF,Ut[i],'␣');
54   WriteIn(kiF);
55   Close(KiF);
56 End{KiIr};
```

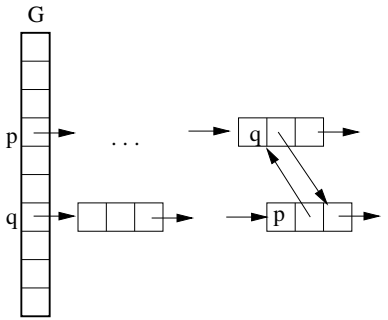
```
57 Procedure VeremBe(p:longint);
58 begin
59   inc(V.teto); V.Tar[V.teto]:=p;
60 end;
61 Procedure Torol;
62 begin
63   dec(V.teto);
64 end;
65 function Teteje:longint;
66 begin
67   Teteje:=V.Tar[V.teto];
68 end;
69 function NemUres:boolean;
70 begin
71   NemUres:=V.teto>0;
72 end;
73 Procedure Letesit();
74 begin
75   V.teto:=0;
76 end;
```

```
77 procedure EulerSetaKeres(var G:graf; a:longint; n:longint; m:longint; var U
78 {A G gráfban a-ból induló Euler-sétát készít}
79 var p:PontTip;
80     k:longint;
81 begin
82   Letesit(m);{verem létesítés}
83   VeremBe(a);
84   k:=m;
85   while NemUres do begin    {amíg a verem nem üres}
86     p:=Teteje;
87     if G[p]=Null then begin{nincs már p-ből benemjárt él}
88       Torol;               {p-t töröljük a veremből}
89       Ut[k]:=p;
90       dec(k);
91     end else begin         {tovább a p->q élen}
92       VeremBe(El[G[p]].pont);
93       G[p]:=El[G[p]].csat;
94     end;
95   end{while};
96 end{EulerUt};
```

```
97  var a,b,i:longint;
98  begin{Program}
99    Beolvas;
00    a:=0; b:=0;
01    for i:=1 to n do begin
02      if BeFok[i]=KiFok[i] then continue;
03      if BeFok[i]+1=Kifok[i] then begin
04        if a=0 then a:=i else a:=n+1;
05      end;
06      if BeFok[i]=Kifok[i]+1 then begin
07        if b=0 then b:=i else b:=n+1;
08      end;
09    end;
10    if a=0 then a:=1;
11    if (a<=n)and(b<=n) then begin
12     EulerSetaKeres(G,a,n,E,Ut);
13     Kilr;
14    end else
15     writeln('Nincs␣Euler-séta!');
16  end.
```

6.2. Euler-kör keresés irányítatlan gráfra

Az algoritmus lényege ugyan az, mint irányított esetben. Azonban a hatékony (élek számával arányos futási idejű) megvalósításhoz meg kell oldani az élek hatékony törlését. Mivel irányítatlan gráfot úgy ábrázolunk, hogy a (p,q) él esetén $p \to q$ és $q \to p$ szerepel az ábrázolásban, ezért ha a $p \to q$-t töröljük, akkor tötölni kell a $q \to p$-t is. Tehát konstans időben meg kell találni a másik irányú élet. Ezt megoldhatjuk úgy, hogy láncolt ábrázolást alkalmazunk, és minden cellában eltároljuk a másik irányú él cellájára mutató információt (címet). Továbbá, nem érdemes ténylegesen törölni, azaz a láncból eltávolítani, hanem csak megjegyezni, hogy törölve lett.



37. ábra. Gráf ábrázolása a $p \to q$ és $q \to p$ élek összekapcsolásával.

```
1  Program EulerKor; {Euler–út(kör) keresés irányítatlan gráfra}
2  Const
3    MaxP=10000;        {a pontok max. száma}
4    MaxM=100000;       {az élek max. száma}
```

```
5    Null=0;
6  Type
7    PontTip=0..MaxP;
8    Cella=Record
9             Pont:PontTip;
10            masik:longint;
11            aktiv:boolean;
12            csat:0..MaxM;
13          End;
14   Graf=Array[1..MaxP] Of 0..MaxM; {a gráfábrázolás típusa }
15   Uttip=array[0..MaxM] of 0..MaxP;
16 Var
17   N,E:Longint;        {a pontok száma, az élek száma  }
18   G:Graf;
19   El:Array[1..MaxP] of Cella;  {az első aktív él: p->q=El[G[p]].pont}
20   Szabad:longint;              {az első szabad cella}
21   Ut:UtTip;                    {a séta pontjai }
22   Fok:Array[1..MaxP] of longint;
23   a:PontTip;
```

```
24 Procedure Beolvas; Var BeF:Text;
25   u,v:PontTip;
26   i, Guv,Gvu:longint;
27 Begin
28   Assign(BeF, 'eulerkor.be'); Reset(BeF);
29   ReadLn(BeF, N, E);
30   For u:=1 To N Do Begin        {Inicializálás}
31     G[u]:=Null; Fok[u]:=0;
32   End;
33   For i:=1 To E Do Begin        {az input beolvasása}
34     ReadLn(BeF, u,v);
35     Guv:=szabad; inc(szabad); Gvu:=szabad; inc(szabad);
36     El[Guv].pont:=v;
37     El[Guv].csat:=G[u];
38     El[Guv].masik:=Guv;
39     G[u]:=Guv;
40     El[Gvu].pont:=u;
41     El[Gvu].csat:=G[v];
42     El[Gvu].masik:=Guv;
43     G[v]:=Gvu;
44     Inc(Fok[u]);  Inc(Fok[v]);
45   End{for i};
46  Close(BeF);
47 End{Beolvas};
```

```
48 procedure EulerKor(
49   var G:graf;            //a gráf
50       n:word; m:longint;//a pontok és élek száma
51       a:PontTip;         //van a-ból induló Euler-út
52   var Ut:UtTip);         //az út
53 var p:PontTip; q, k:longint;
54 begin
55   Letesit(m);{verem létesítés}
56   VeremBe(a);    k:=m;
57   while NemUres do begin {amíg a verem nem üres}
58     p:=Teteje;
59     while (G[p]<>Null)and(not El[G[p]].aktiv) do G[p]:=El[G[p]].csat;
60     if G[p]=Null then begin{nincs már p-ből benemjárt él}
61       Torol;                {p-t töröljük a veremből}
62       Ut[k]:=p;  dec(k);
63     end else begin          {tovább a p->q élen}
64       q:=El[G[p]].pont
65       VeremBe(q);
66 //       El[G[p]].aktiv:=false;
67       El[El[G[p]].masik].aktiv:=false;
68       G[p]:=El[G[p]].csat;
69     end;
70   end{while};
71 end{EulerKor};
```

```
72  Begin{Program}
73     Beolvas;
74     Elokeszit;
75     EulerKor;
76  End.
```