10. Gráf absztrakt adattípus, gráfok ábrázolása

10.1. 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**:
$G = (V,E,Ind,Érk)$, $Ind,Érk : E \to V$
$Ind(e)$ 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 $(\forall p,q \in V)((p,q) \in E \Rightarrow (q,p) \in E)$.
Minden $G = (V,E)$ gráf megadható olyan függvénnyel, amely a gráf minden $p \in V$ pontjához azon q pontok halmazát rendeli, amelyekre $(p,q) \in E$.

$$r_E : V \to 2^V; \quad r_E(p) = \{q : (p,q) \in E\}$$

Címkézett gráf esetén pedig

$$r_E : V \to 2^{V \times Cimke}; \quad r_E(p) = \{(q,s) : (p,q) \in E \wedge C(p,q) = s\}$$

Jelölések
$Ki(G,p) = \{q \in V : (p,q) \in E\}$
$Be(G,p) = \{q \in V : (q,p) \in E\}$
$KiFok(G,p) = |Ki(G,p)|$
$BeFok(G,p) = |Be(G,p)|$

10.2. Gráf absztrakt adattípus

Milyen műveleteket akarunk gráfokon végezni?
Értékhalmaz: $Graf = \{G = (V,E) : V \subseteq PontTip, E \subseteq V \times V\}$
Műveletek:

$$G : Graf,\ p,p1,p2 : PontTip,\ ir : boolean,\ I : Iterator,$$

$\{Igaz\}$	$Letesit(G,ir)$	$\{G = (\emptyset,\emptyset)\}$
$\{G = G\}$	$Megszuntet(G)$	$\{Igaz\}$
$\{G = G\}$	$Uresit(G)$	$\{G = (\emptyset,\emptyset)\}$
$\{G = G\}$	$Iranyitott(G)$	$\{\neg Iranyitott(G) \Rightarrow (p,q) \in E \Rightarrow (q,p) \in E)\}$
$\{G = G\}$	$Pontokszama(G)$	$\{= \lvert V \rvert\}$
$\{G = G\}$	$Elekszama(G)$	$\{= \lvert E \rvert\}$
$\{G = G\}$	$KiFok(G,p)$	$\{= \lvert Ki(G,p) \rvert\}$
$\{G = G\}$	$BeFok(G,p)$	$\{= \lvert Be(G,p) \rvert\}$
$\{G = (V,E)\}$	$PontBovit(G,p)$	$\{V = Pre(V) \cup \{p\} \wedge E = Pre(E)\}$
$\{G = (V,E) \wedge p \in V\}$	$PontTorol(G,p)$	$\{V = Pre(V) - \{p\} \wedge$ $E = Pre(E) - \{(p,q) : q \in Ki(G,p)\} -$ $\{(q,p) : q \in Be(G,p)\}\}$
$\{G = (V,E), p1,p2 \in V\}$	$ElBovit(G,p1,p2)$	$\{E = Pre(E) \cup \{(p1,p2)\} \wedge V = Pre(V)\}$
$\{G = (V,E), p1,p2 \in V\}$	$ElTorol(G,p1,p2)$	$\{E = Pre(E) - \{(p1,p2)\} \wedge V = Pre(V)\}$
$\{G = (V,E), p1,p2 \in V\}$	$Vanel(G,p1,p2)$	$\{= (p1,p2) \in E \wedge E = Pre(E) \wedge V = Pre(V)\}$
$\{G = G\}$	$PIterator(G,I)$	$\{\}$
$\{G = G\}$	$KiEl(G,p)$	$\{= \{q : VanEl(p,q)\}\}$
$\{G = G\}$	$KiIterator(G,p)$	$\{\}$
$\{G = G\}$	$ElIterator(G,I)$	$\{\}$

A továbbiakban feltételezzük, hogy a gráf pontjait természetes számokkal azonosítjuk, pontosabban $V \subseteq \{1,\ldots,n\}$
Java nyelven az alábbi interface-t használhatjuk.

```
public interface Graf extends Iterable<Integer>{
   public boolean Iranyitott();
   public int Pontokszama();
   public int Maxpont();
   public int Elekszama();
   public int KiFok(int p);
   public int BeFok(int p);
   public void Uresit();
   public void PontBovit(int p);
   public void PontTorol(int p);
   public void ElBovit(int p, int q);
   public void ElTorol(int p, int q);
   public boolean VanEl(int p, int q);
   public Halmaz<Integer> Ki(int p);
   public Iterator<Integer> KiIterator(int p);
   public Iterator<GrafEl> ElIterator();
}
```

Ahol a GrafEl osztály

```
public class GrafEl{
   public int ki;
   public int be;
   public GrafEl(int p, int q){
      ki=p;
      be=q;
   }
   public GrafEl(){
   }
   public String toString(){
      return Integer.toString(ki)+"->"+Integer.toString(be);
   }
}
```

Ekkor a gráf pontjainak bejárása:

```
for (int p:G)
  M(p);
```

vagy

```
Iterator<Integer> piter=G.iterator();
while (piter.hasNext()){
  int p=piter.next();
  M(p);
}
```

Adott p pont szomszédjainak (a p-ből induló élek) bejárása:

```
for (int q:G.KiEl(p))
   M(p,q);
```

vagy

```
Iterator<Integer> kiiter=G.KiEl(p).iterator();
 while (kiiter.hasNext()){
   int q=kiiter.next();
   M(p, q);
 }
```

A gráf minden $(p,q) \in E$ élének bejárása:

```
for (Iterator<GrafEl> eliter=G.ElIterator(); eliter.hasNext();){
   GrafEl el=eliter.next();
   M(el.ki, el.be);
}
```

vagy

```
Iterator<GrafEl> eliter=G.ElIterator();
   while (eliter.hasNext()){
     GrafEl el=eliter.next();
     M(el.ki, el.be);
  }
```

Az él-iteráció nyilvánvalóan megvalósítható a pont-iteráció és ki-iteráció műveletekkel, de fordítva nem.

```
for (int p:G){
   for (int q:G.Ki(p))
      M(p,q);;
   }
}
```

Címkézett (súlyozott) gráf absztrakt adattípus
Értékhalmaz:
$Graf = \{G = (V,E,C) : V \subseteq PontTip, E \subseteq V \times V, C : E \to CimkeTip\}$
Műveletek: Graf-műveletek +

$$G : Graf, P, P1, P2 : PontTip, S : CimkeTip, I : PIterator,$$

$\{G = (V,E), p1, p2 \in V\}$	$ElBovit(G, p1, p2, s)$	$\{E = Pre(E) \cup \{(p1, p2)\} \wedge C(p1, p2) = s\}$
$\{G = (V,E), (p1, p2) \in E\}$	$ElCimke(G, p1, p2, s)$	$\{s = Pre(C)(p1, p2) \wedge E = Pre(E)\}$
$\{G = (V,E), (p1, p2) \in E\}$	$ElCimkez(G, p1, p2, s)$	$\{s = C(p1, p2) \wedge E = Pre(E)\}$
$\{G = G\}$	$KiCEl(G, p)$	$\{= \{(q,s) : VanEl(p,q) \wedge C(p,q) = s\}\}$
$\{G = G\}$	$ElIterator(G,I)$	$\{\}$

Ha a CimkeTip típuson alapértelmezett lineáris rendezési reláció, akkor a címkézett gráfot *súlyozott gráfnak* nevezzük. Ekkor az élek halmazán is alapértelmezett az a rendezés, ami az él súlya szerinti rendezés. Tehát $(p_1, q_1) \leq (p_2, q_2) \Leftrightarrow$ ha $C(p_1, q_1) \leq C(p_2, q_2)$.

```
public interface CGraf<Cimke> extends Graf, Iterable<Integer>{
   public void ElBovit(int p, int q, Cimke s);
   public Cimke ElCimke(int p, int q);
   public void ElCimkez(int p, int q, Cimke s);
   public Fuggveny<Integer,Cimke> KiCEl(int p);
   public Iterator<CGrafEl<Cimke>> CElIterator();
}

public interface SGraf<Suly extends Comparable<Suly>>
 extends Graf, Iterable<Integer>{
   public void ElBovit(int p, int q, Suly s);
   public Suly ElSuly(int p, int q);
   public Fuggveny<Integer,Suly> KiSEl(int p);
   public void ElSulyoz(int p, int q, Suly s);
   public Iterator<SGrafEl<Suly>> SEliterator();
}
```

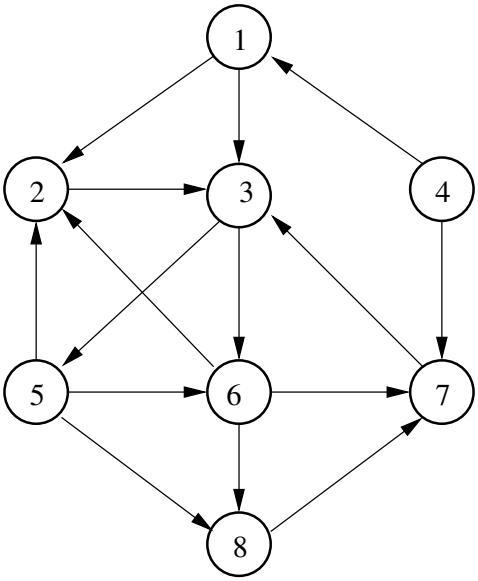
10.3. Gráfok ábrázolásai

Szempontok az adatszerkezet megválasztásához.

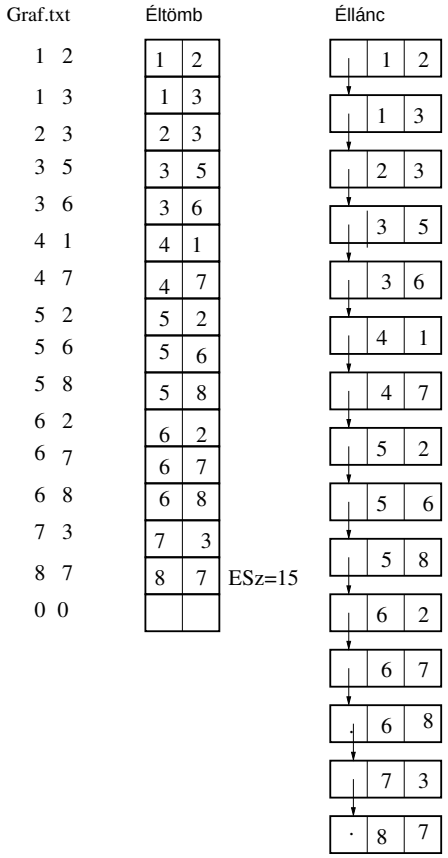
1. Az adott probléma megoldásához ténylegesen mely műveletek szükségesek.
2. Melyek a releváns műveletek, amelyek alapvetően befolyásolják az algoritmus futási idejét.
3. A tárigény az adott probléma esetén.

1. **Élhalmaztömb és élhalmazlánc**
a.) Statikus (tömbös)
Tárigény: $\Theta(|E|)$
VANEL időigénye: $\Theta(|E|)$
El-iteráció időigénye: $\Theta(|E|)$
Ki-iteráció időigénye: $\Theta(|E|)$

b.) Dinamikus
Tárigény: $\Theta(|E|)$



1. ábra. Példa gráf



2. ábra. Élhalmaztömb és élhalmazlánc

VANEL időigénye: $\Theta(|E|)$
El-iteráció időigénye: $\Theta(|E|)$
Ki-iteráció időigénye: $\Theta(|E|)$

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									

3. ábra. Kapcsolatmátrix

```
boolean[][] G;  //
Cimke[][] G;    //címkézett (súlyozott) gráf ábrázolására
```

(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 $G[p][q]$.
Multi-gráf nem ábrázolható szomszédsági mátrix-al.
Tárigény: $\Theta(n^2)$ ($n = |V|$)
VANEL időigénye: $\Theta(1)$
El-iteráció időigénye: $\Theta(n^2)$
Ki-iteráció időigénye: $\Theta(n)$

3. **Éllista**

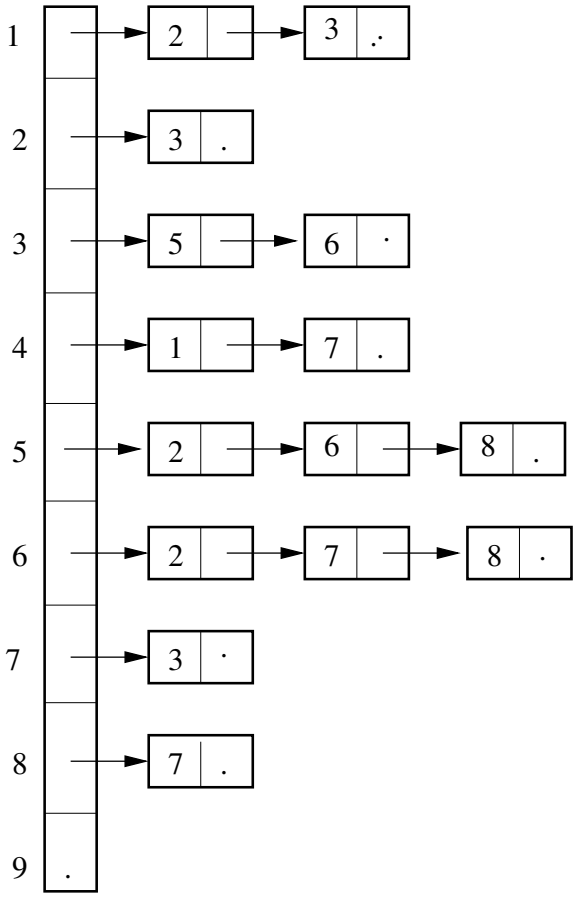
a.) Statikus

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

4. ábra. Statikus éllista

```
int[][] G;
int[] KiFok;
```

Tárigény: $\Theta(n^2)$
VANEL időigénye: $T_{lr} = O(n)$
El-iteráció időigénye: $\Theta(|E|)$
Ki-iteráció időigénye: $\Theta(|Ki(G,p)|)$
b.) Dinamikus



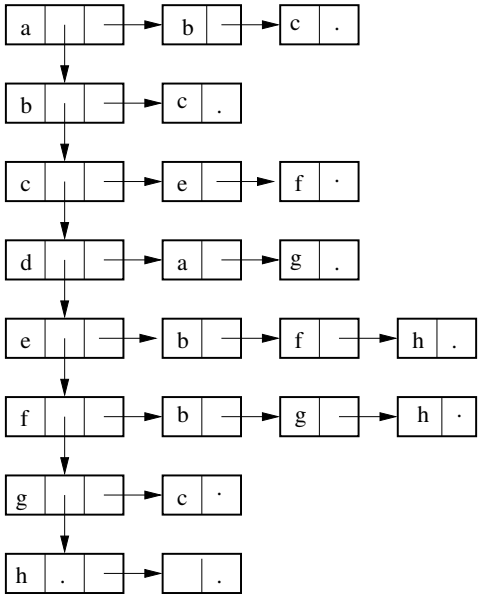
5. ábra. Dinamikus éllista

```
  Lanc<Integer>[] G;
```

Tárigény: $\Theta(|V|+|E|)$
VANEL időigénye: $T_{lr} = O(n)$
El-iteráció időigénye: $\Theta(|E|)$
Ki-iteráció időigénye: $\Theta(|Ki(G,p)|)$
4. **Általános éllista**

```
public class GrafLanc<PontTip>{
   public PontTip pont;
   public Lanc<PontTip> ki;
   public GrafLanc<PontTip> csat;
}
```

Az általános éllistás ábrázolás előnye, hogy a gráf pontjai tetszőleges (de rögzített) típusú értékek lehetnek.
Címkézett gráf esetén a ki-lánc (a vízszintes lánc) minden cellája a pont mellett egy címke értéket is tartalmaz.
Tárigény: $\Theta(|V|+|E|)$
VANEL időigénye: $T_{lr} = O(n)$
El-iteráció időigénye: $\Theta(|E|)$
Ki-iteráció időigénye: $T_{lr} = O(n) + \Theta(|Ki(G,p)|)$



6. ábra. Általános éllista

5. **Halmazfüggvény**

$$G = (V,E), \quad E \subseteq V \times V$$
$$r_E : V \to 2^V; \quad r_E(p) = Ki(G,p) = \{q : (p,q) \in E\}$$

```
Halmaz<Integer>[] G;
```

(p,q) akkor és csak akkor éle a gráfnak, ha $G[p].Eleme(q)$.
Tehát a VANEL(G,P,Q) futási ideje a Halmaz adattípus ELEME műveletének futási ideje lesz.
Ki-iteráció (a p-ből induló élek bejárása) megvalósítása:

```
for (int q:G[p])
  M(p,q);
```

Vegyük észre, hogy ha a $Ki(G,p)$ halmazokat bitvektorral ábrázoljuk, akkor éppen a szomszédsági mátrix ábrázolást kapjuk, ha tömbbel ábrázoljuk, akkor a statikus éllistás, ha lánccal, akkor a dinamikus éllistás ábrázolást kapjuk.
Általánosan, a $Ki(G,p)$ halmazok ábrázolására bármely halmaz ábrázolás használható. Tehát akár bináris (kiegyensúlyozott) keresőfát is alkalmazhatunk.
Ezért a gráfok Java megvalósítása megadható egyetlen kóddal (osztállyal) úgy, hogy a konstruktor paramétereként megadhatjuk:
a pontok számát,
a gráf irányított-irányítatlan voltát,
a $Ki(G,p)$ halmazok ábrázolási módját.
Lásd a GrafA osztályt.

6. **V keresőfában, E láncban (keresőfában)**

Ha a gráf pontjai tetszőleges egész számok (vagy olyan típusú értékek, amelyen értelmezett egy lineáris rendezési reláció), akkor a pontok V halmazát ábrázolhatjuk bináris (kiegyensúlyozott) keresőfában. Kiegyensúlyozott keresőfa használatával a műveletek futási ideje:
Tárigény: $\Theta(|V|+|E|)$
VANEL időigénye: $T_{lr} = O(\lg n)$
El-iteráció időigénye: $\Theta(|E|)$
Ki-iteráció időigénye: $O(\lg n) + \Theta(|Ki(G,p)|)$

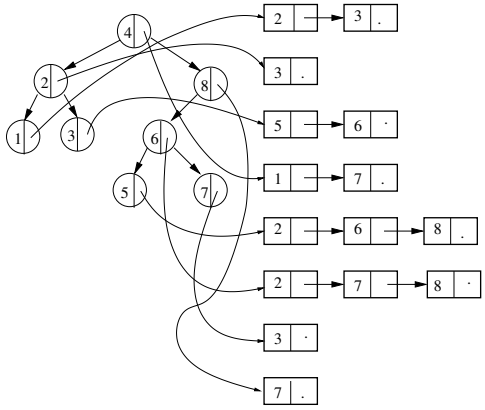
7. **Dinamikus változók összekapcsolása**

Speciális gráfok, például fák ábrázolhatók dinamikus változók (referenciák, pointerek) alkalmazásával.

8. **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.

A Graf adattípus megvalósításai:



7. ábra. Keresőfás ábrázolás

GrafA: Konstruktor paraméterezés: $GrafA(\text{int } n, \text{boolean } irany, \text{String } abr)$, ahol
n a létesítendő gráf pontjainak száma,
$irany = true$ esetén irányított, egyébként irányítatlan lesz a gráf, alapértelmezett érték $true$
abr értéke határozza meg az ábrázolás módját, alapértelmezett értéke "Lanc":

- "Matrix" a ki-pontok halmaza tömb adatszerkezetben, azaz szomszédsági mátrix az ábrázolás módja
- "Lanc" a ki-pontok halmaza lánc adatszerkezetben
- "BKF" a ki-pontok halmaza bináris keresőfa adatszerkezetben
- "AVLFa" a ki-pontok halmaza AVL-fa bináris keresőfa adatszerkezetben
- "HasitL" a ki-pontok halmaza hasítótábla láncolással adatszerkezetben
- "HasitN" a ki-pontok halmaza hasítótábla nyílt címzéssel adatszerkezetben

Az SGraf adattípus megvalósításai:
SGrafA: Konstruktor paraméterezés: $SGrafA(\text{int } n, \text{boolean } irany, \text{String } abr)$, ahol
n a létesítendő gráf pontjainak száma,
$irany = true$ esetén irányított, egyébként irányítatlan lesz a gráf, alapértelmezett érték $true$
abr értéke határozza meg az ábrázolás módját, alapértelmezett értéke "Lanc":

- "Matrix" a ki-pontok halmaza tömb adatszerkezetben, azaz szomszédsági mátrix az ábrázolás módja
- "Lanc" a ki-pontok halmaza lánc adatszerkezetben
- "BKF" a ki-pontok halmaza bináris keresőfa adatszerkezetben
- "AVLFa" a ki-pontok halmaza AVL-fa bináris keresőfa adatszerkezetben
- "HasitL" a ki-pontok halmaza hasítótábla láncolással adatszerkezetben
- "HasitN" a ki-pontok halmaza hasítótábla nyílt címzéssel adatszerkezetben

10.4. Elemi gráfalgoritmusok

10.4.1. Utak

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

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

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

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

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

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

10.5. 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:

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

10.6. 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.*

10.7. 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.

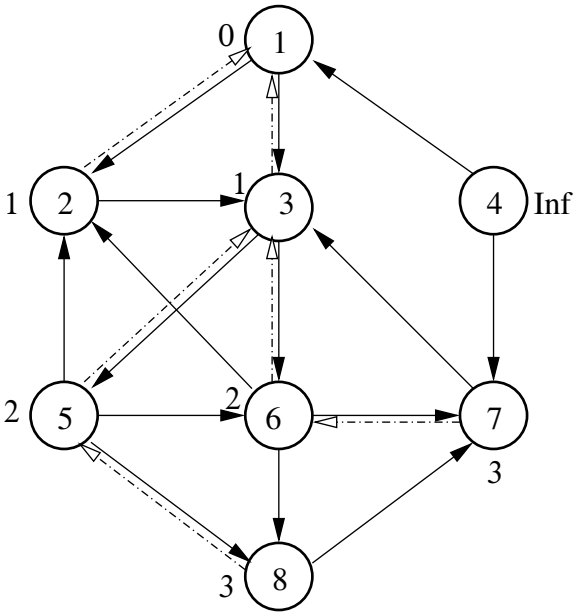
5. Minden p, q pontpárra $\delta(p,q)$ és egy $p \rightsquigarrow q$ legrövidebb út kiszámítása.

10.5. 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.

```
public class SzeltKeres{
   public int[] D;
   public int[] Apa;
   public void Bejar(Graf G, int r){
      int n=G.Maxpont();
      int u; int Inf=n;
      D=new int[n+1];
      Apa=new int[n+1];
      Sor<Integer> S=new SorT<Integer>(n);
      for (int p:G){
         Apa[p]=-1; D[p]=Inf;
      }
      D[r]=0; Apa[r]=0;
      S.SorBa(r);
```

```
      while(S.Elemszam()>0){
         u=S.SorBol();
         for (int v:G.KiEl(u))
            if (Apa[v]<0){
               Apa[v]=u;
               D[v]=D[u]+1;
               S.SorBa(v);
            }
      }
   }
}
```



8. ábra. A példa gráf szélességi bejárása az 1 pontra.

10.8. lemma. *Legyen $G=(V,E)$ irányított vagy irányítatlan gráf és $s,u,v \in V$.*
Minden $(u,v) \in E$ élre $\delta(s,v) \leq \delta(s,u)+1$.

10.9. lemma. *Ha* SZELTKERES *algoritmust az r pontra alkalmazzuk, akkor a kiszámított D-re teljesül:* $(\forall v \in V)\,(D[v] \geq \delta(r,v))$.

Bizonyítás. Az S sorba kerülés szerinti indukcióval.
i) Az első pont, ami bekerül: r, de $D[r] := 0$ és $\delta(r,r) = 0$
ii) Tfh. $S.Sorbol(u)$ után az $u \to v$ élet vizsgáljuk és $D[v] = Inf$, ami ekvivalens azzal, hogy $Apa[v] < 0$. Ekkor

$$\begin{aligned} D[v] &= D[u]+1 \\ &\geq \delta(r,u)+1 \\ &\geq \delta(r,v) \end{aligned}$$

és v-t betesszük az S sorba.

■

10.10. lemma. *Legyen a* SZELTKERES *algoritmust végrehajtásának egy pillanatában az S sor tartalma $S =< v_1, v_2, \ldots, v_k >$. Ekkor $D[v_k] \leq D[v_1]+1$ és $D[v_i] \leq D[v_{i+1}]$ $(i = 1,\ldots,k-1)$*

Következmény
Ha egy u pont előbb kerül a sorba, mint a v pont, akkor $D[u] \leq D[v]$.

Bizonyítás.
a) $u = S.Sorbol()$ után: $S =< v_2, \ldots, v_k >$
Bizonyítandó: $D[v_k] \leq D[v_2]+1$.

De $D[v_k] \leq D[v_1]+1 \leq D[v_2]+1$
b) $S.Sorba(v)$ után: $S =< v_2, \ldots, v_k, v >$
Bizonyítandó: $D[v_k] \leq D[v]$ és $D[v] \leq D[v_2]+1$.
$D[v_k] \leq D[v_1]+1 = D[v]$
$D[v_1] \leq D[v_2] \Rightarrow D[v] = D[u = v_1]+1 \leq D[v_2]+1$ ■

10.11. tétel. $(\forall v \in V)\,(\delta(r,v) = D[v])$

Bizonyítás.
a) $\delta(r,v) = \infty$ azaz nincs $r \rightsquigarrow v$ út.
Ekkor v nem kerülhet be a sorba, tehát marad a kezdetben kapott $D[v] = Inf$

b) Tfh.$\delta(r,v) \neq \infty$
Indirekt biz. : Tfh. v a legkisebb olyan $\delta(r,v)$ értékű pont, hogy $\delta(r,v) \neq D[v]$
($r \neq v$) és 2. lemma miatt: $D[v] > \delta(r,v)$ és $\delta(r,v) < \infty$
Legyen u egy olyan pont, amely közvetlenül megelőzi v-t az $r \rightsquigarrow v$ legrövidebb úton: $r \rightsquigarrow u \rightarrow v$
Mivel $\delta(r,u) < \delta(r,v)$, az ind. feltevés szerint: $D[u] = \delta(r,u)$ és $\delta(r,v) = \delta(r,u)+1$

$D[v] > \delta(r,v) = \delta(r,u)+1 = D[u]+1$

Tekintsük azt a helyzetet, amikor a SZELTKERES algoritmus az $u \rightarrow v$ élet vizsgálja.
I) $D[v] = Inf$: tehát $D[v] := D[u]+1 = \delta(r,v)+1$ Ellentmondás!
II)$D[v] < Inf$: tehát v már korábban kapott $D[v] < \infty$ értéket.
De $D[v] \leq D[u]$, mivel v már nincs a sorban, ami ellentmondás!
■

A SZELTKERES algoritmus futási ideje:
$T_{lr} = O(V+E)$,
feltéve, hogy a Ki-iteráció lineáris idejű. Ez teljesül, ha éllistás ábrázolást alkalmazunk.

10.12. Állítás. *A $G_\pi = (V_\pi, E_\pi)$ gráf, ahol*
$V_\pi = \{v : v \in V \wedge Apa[v] \neq 0 \vee v = r\}$
$E_\pi = \{(u,v) : Apa[v] = u \wedge Apa[v] \neq 0\}$
G-nek egy r-gyökerű legrövidebb utak feszítőfája.