Gráfok, definíciók

Irányítatlan gráf: G = (V,E), ahol E rendezetlen $(a,b), a,b \in V$ párok halmaza.

Irányított gráf: G = (V,E) E rendezett (a,b) párok halmaza; $E \subseteq V \times V$.

Címkézett (súlyozott) gráf: G = (V,E,C) $C : E \rightarrow$ Címke

Minden irányítatlan G = (V,E) gráf olyan irányított gráfnak tekinthető, amelyre teljesül, hogy ha $(p,q) \in E)$ akkor $(q,p) \in E$.

Jelölések:

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

Gráfok ábrázolása

Szempontok az adatszerkezet megválasztásához.

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

Ábrázolások

- Élhalmazlánc vagy élhalmaztömb: az élek felsorolása. Tárigény: $\Theta(|E|)$, VANEL időigénye: $\Theta(|E|)$, El-iteráció időigénye: $\Theta(|E|)$, Ki-iteráció időigénye: $\Theta(|E|)$.
- Kapcsolatmátrix (szomszédsági) mátrix: $G[i,j] = 1$ ha $(i,j) \in E$, címkézett gráf esetén G a címkéket tárolhatja. Tárigény: $\Theta(n^2)$, ahol $|V| = n$, VANEL időigénye: $\Theta(1)$, El-iteráció időigénye: $\Theta(n^2)$, Ki-iteráció időigénye: $\Theta(n)$.
- Statikus éllista: egy n elemű tömb, amelynek az i-dik eleme a $Ki(G,i)$ halmazt tartalmazza láncként. Tárigény: $\Theta(|E|+n)$, VANEL időigénye: $O(n)$, El-iteráció időigénye: $\Theta(|E|)$, Ki-iteráció időigénye: $\Theta(|Ki(G,i)|)$.

Példa:

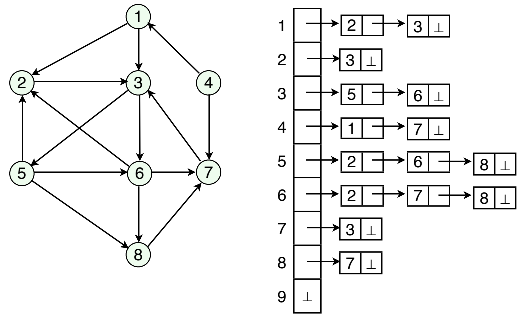
Számított gráf

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.

Alapvető definíciók

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

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



1. ábra.

A $\pi = p \rightsquigarrow q$ út hossza, $|\pi| = |p \rightsquigarrow q| = k$. Ha G = (V,E,C) élei a $C : E \rightarrow R$ függvénnyel súlyozottak, akkor a $p \rightsquigarrow q$ út hossza $|p \rightsquigarrow q| = \sum_{i=1}^{k} = C(p_{i-1}, p_i)$.

Definíció p-ből q-ba vezető legrövidebb út hossza, p és q távolsága: $\delta(p,q) = \infty$, ha nincs $p \rightsquigarrow q$ út és $Min\{|\pi| : \pi : p \rightsquigarrow q\}$ egyébként.

Definíció A $p = \{p_0, p_1, \ldots, p_k\}$ pontsorozatot a p_0-ból p_k-ba vezető sétának nevezzük, ha $(\forall i \in \{1, \ldots, k\})(p_{i-1}, p_i) \in E$.

Definíció A G=(V,E) irányított vagy irányítatlan gráfnak az $F = (\bar{V}, \bar{E})$ gráf a $r \in V$ gyökerű feszítőfája, ha

- F részgráfja G-nek ($\bar{V} \subseteq V, \bar{E} \subseteq E$) és fa.
- $(\forall p \in V)$ akkor és csak akkor van $r \rightsquigarrow p$ G-ben, ha van $r \rightsquigarrow p$ F-ben.

Definíció A G = (V,E) irányított vagy irányítatlan, esetleg súlyozott gráfnak az $F = (\bar{V}, \bar{E})$ gráf a $r \in V$ gyökerű legrövidebb utak feszítőfája (LUF), ha

- F r-gyökerű feszítőfája G-nek, és
- $\forall p \in V$-re $\delta_G(r,p) = \delta_F(r,p)$.

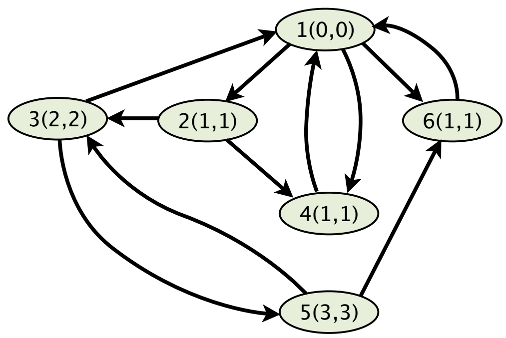
Szélességi keresés

Egy adott súlyozatlan irányított vagy irányítatlan gráf egy pontjából keressük az elérhető pontokat, és az azokhoz vezető legrövidebb utakat.

Bemenet: a $G = (V,E)$ gráf és egy kiindulási pont $r \in V$. Kimenet: a legrövidebb utak fája, egy Apa függvény által megadva, és egy d függvény, amelyre $d(p) = \delta(p)$ minden $p \in V$-re.

```
SZELTKERES
 for(p in V)
     {Apa(p):=-1
      d(p):=INF}
 d(r):=0
 Apa(r):=0
 Letesit(S:Sor)
 Sorba(S,r)
 While(Elemszam(S)>0)
```

```
{Sorbol(S,u)
 for(v in Kiel(G,u))
    {if Apa(v)<0
        Then {Apa(v):=u
              d(v):=d(u)+1
              Sorba(S,v)}}}
```



2. ábra.

Példa Adott egy sziget ahol 42 kaméleon lakik. A kaméleonok három színt vehetnek fel piros, kék és zöld. Ha két különböző színű kaméleon találkozik, akkor megijednek és mindkettő átváltoztatja a színét a harmadik színre. A szigeten a kiindulási időpontban 13,14,15 darab piros zöld és kék kaméleon van. Előfordulhat -e, hogy a szigeten minden kaméleonnak ugyanaz legyen a színe?

Megoldás: Definiáljunk egy gráfot, a gráf pontjai az (x,y,z) számhármasok, ahol minden komponens nemnegatív egész és $x+y+z=42$. Az élek a lehetséges átváltozásoknak felelnek meg. Tehát (x,y,z)-nek három szomszédja lesz $(x-1,y-1,z+2)$, $(x-1,y+2,z-1)$, $(x+2,y-1,z-1)$. A feladat, hogy meghatározzuk van -e a gráfban a $(13,14,15)$ pontból a $(42,0,0)$, $(0,42,0)$, $(0,0,42)$ pontok valamelyikébe út. Ezt megoldhatjuk szélességi kereséssel.

Példa

Erdős szám kiszámítása. Erdős Pál matematikus tiszteletére definálták az Erdős szám fogalmát. Erdős Pál Erdős száma 0, azoknak a matematikusoknak, akiknek van közös cikkük Erdőssel az Erdős számúk 1. Azoknak, akiknek nincs közös cikkük Erdős Pállal, de van közös cikkük olyan matematikussal, akinek az Erdős száma 1, az Erdős száma 2. Általában egy matematikus Erdős száma a társszerzői Erdős számának minimuma +1. Adjunk algoritmust, amely a cikkek adatbázisa alapján (feltételezhetjük, hogy adott egy TÁRS algoritmus, ami minden matematikusra visszaadja a társszerzői halmazát) meghatározza egy adott személy Erdős számát. A futási idő lineáris kell legyen $R+n$-ben, ahol n az adatbázisban a matematikusok száma R pedig a $t(i)$ értékek összege, ahol $t(i)$ az i személy társsszerzőinek száma.

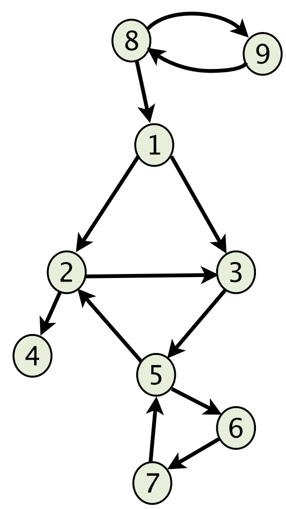
Definiálunk egy gráfot, amelyben midnenkinek a társszerzői a szomszédjai. Ebben keressük a legrövidebb Erdős szerző utat.

Mélységi keresés kibővítve

Az algoritmus elemzéséhez és egyes alkalmazásainak helyességének bizonyításához hasznos, ha minden pontra definiáljuk a d elérési és f elhagyási idejét.

```
MelykeresBovit(G)
 for(u in G)
   {szin(u):=feher
    Apa(u):=0}
 ido:=0
 for(u in G)
    {if szin(u)=feher
      then MBejarBovit(u)}
MBejarBovit(u)
 szin(u):=szurke
 ido:=ido+1
 d(u):=ido
 for(v in Kiel(G,u))
    {if szin(v)=feher
       Then {Apa(v):=u
             MBejarBovit(v)}}
 szin(u):=fekete
 ido:=ido+1
 f(u):=ido
```

- MBejarBovit(1)
- szín(1):=szürke
- d(1)=1
- Apa(2):=1
- MBejarBovit(2)
- szín(2):=szürke
- d(2)=2
- Apa(3):=2
- MBejarBovit(3)
- szín(3):=szürke
- d(3)=3



3. ábra.

- Apa(5):=3
- MBejarBovit(5)
- szín(5):=szürke
- d(5)=4
- Apa(6):=5
- MBejarBovit(6)
- szín(6):=szürke
- d(6)=5
- Apa(7):=6
- MBejarBovit(7)
- szín(7):=szürke
- d(7)=6
- MBejarBovit(7) vége, mert 5 már nem fehér
- szín(7):=fekete

- f(7)=7
- MBejarBovit(6) vége
- szín(6):=fekete
- f(6)=8
- MBejarBovit(5) vége
- szín(5):=fekete
- f(5)=9
- MBejarBovit(3) vége
- szín(3):=fekete
- f(3)=10
- Apa(4):=2
- MBejarBovit(4)
- szín(4):=szürke
- d(4)=11
- MBejarBovit(4) vége
- szín(4):=fekete
- f(4)=12
- MBejarBovit(2) vége
- szín(2):=fekete
- f(2)=13
- MBejarBovit(1) vége
- szín(1):=fekete
- f(1)=14
- MBejarBovit(8)
- szín(8):=szürke
- d(8)=15
- Apa(9):=8
- MBejarBovit(9)
- szín(9):=szürke

- d(9)=16
- MBejarBovit(9) vége
- szín(9):=fekete
- f(9)=17
- MBejarBovit(8) vége
- szín(8):=fekete
- f(8)=18

1. táblázat. A keresés során kapott értékek

pont	1	2	3	4	5	6	7	8	9
apa	0	1	2	2	3	5	6	0	8
d	1	2	3	11	4	5	6	15	16
f	14	13	10	12	9	8	7	18	17

Topologikus rendezés

Definíció Egy G =(V,E) irányított gráf topologikus rendezésén a V pontjainak egy olyan $v_1, v_2, \ldots, v_n$ $(n = |V|)$ felsorolását értjük, amelyre teljesül, hogy minden $(u, v) \in E$ élre, u előbb van a felsorolásban, mint v.

Lemma A G = (V,E) irányított gráfnak akkor és csak akkor van topologikus rendezése, ha G körmentes.

Tétel Ha a G irányított gráf körmentes, akkor pontjainak minden olyan $v_1, \ldots, v_n$ felsorolása, amelyre $f(v_i) > f(v_{i+1})$, $i = 1, \ldots, n-1$ G egy topologikus rendezése lesz bármely mélységi bejárással kiszámított f elhagyási függvényre.

Elvi algoritmus

- A mélységi keresés algoritmusát hajtsuk végre a gráfra.
- Az egyes csúcsok elhagyásakor beszúrjuk őket egy láncolt lista elejére.
- A csúcsok láncolt listája adja meg a rendezési sorrendet.

Az algoritmus futási ideje megegyezik a mélységi keresésével, azaz $\theta(V + E)$. (Feltéve, hogy a ki iteráció lineáris időben megvalósítható.)

Erősen összefüggő komponensek

Definíció $u \sim v$ ha van $u \rightsquigarrow v$ út és $v \rightsquigarrow u$ út is a gráfban.

Lemma A $\sim$ reláció egy ekvivalencia reláció, így egy osztályozást definiál.

Definíció A $\sim$ reláció által definiált osztályozás osztályai a gráf erősen összefüggő komponensei. Azaz egy u pontot tartalmazó erősen összefüggő komponens: $C(u) = \{v \in V : u \sim v\}$.

Definíció A G = (V,E) gráf transzponáltja: $G^T = (V, E^T)$, ahol $E^T = \{(p,q) : (q,p) \in E\}$.

Definíció Egy G = (V,E) gráf komponensgráfjának hívjuk azt a gráfot, amelynek csúcsai az erősen összefüggő komponensek, és C komponensből akkor vezet él egy D komponensbe, ha C valamely pontjából G-ben vezet él D valamely pontjába.

Lemma A komponensgráf körmentes.
Példa EOK komponensek használatára

Feladat: Egy faluban minden emberre ismert azoknak a halmaza, akiknek továbbmondja az általa megismert pletykát. Határozzuk meg a falu legtitoktartóbb emberét, azaz olyan embert, akinek elmondva egy hírt a legkevesebb emberhez jut el a pletyka!

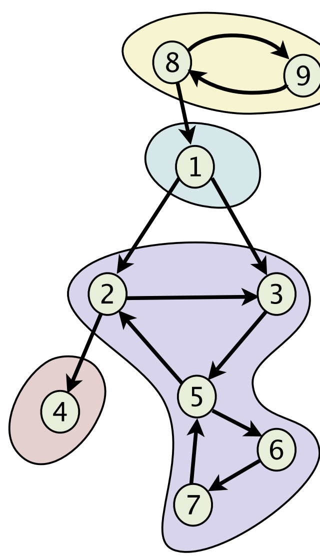
Megoldás: Definiáljunk egy irányított gráfot (a falu pletyka gráfját): a pontok az emberek A-ból megy él B-be, ha A elmondja B-nek a pletykát. A gráfban a feladat olyan pont keresése, amelybol a legkevesebb pontba vezet út.

A megoldást a következő (elvi) algoritmus szolgáltatja - határozzuk meg az erősen összefüggő komponensek komponensgráfját - vegyük azokat a komponenseket, amelyekből a komponensgráfból nem vezet ki él - a legkisebb elemszámú ilyen komponensből válasszunk ki egy pontot.

Elvi algoritmus

- 1. Számítsuk ki a MELYKERES algoritmussal az f(u) elhagyási értékeket.
- 2. A G^T transzponált gráfra alkalmazzuk a MELYKERES eljárást úgy, hogy a pontokra a MBEJÁR eljárást f szerint csökkenő sorrendben hívjuk.
- 3. A 2. pontban az egy mélységi feszítőfába kerülő pontok alkotnak egy erősen összefüggő komponenst.

Megvalósítás: A 2-es pontban nem rendezést hajtunk végre az f értékek szerint, hanem a pontokat a befejezésük időpontjában berakjuk egy verembe, és abból vesszük a pontokat a második bejárásnál.



4. ábra.

Példa:

A transzponált gráfon a Mélykeres ejjárásban a pontok sorrendje: $8, 9, 1, 2, 4, 3, 5, 6, 7$

2. táblázat. Az első bejárás során kapott értékek

pont	1	2	3	4	5	6	7	8	9
apa	0	1	2	2	3	5	6	0	8
d	1	2	3	11	4	5	6	15	16
f	14	13	10	12	9	8	7	18	17

3. táblázat. Az második bejárás során kapott értékek

pont	1	2	3	4	5	6	7	8	9
apa	0	0	5	0	2	7	5	0	8
d	5	7	9	17	8	12	11	1	2
f	6	16	10	18	15	13	14	4	3

Komponensek: (8,9), (1), (2,5,3,7,6), (4)

Legrövidebb utak súlyozott gráfokban

A feladat egy súlyozott gráfban egy adott pontból kiinduló legrövidebb utak megkeresése. Az input a súlyozott gráf és a kiindulási *s* pont. Outputként egy legrövidebb utak fáját adunk vissza, egy Apa függvény által, továbbá a legrövidebb utak hosszait egy *d* függvény által. A feladatot nemnegatív élsúlyok esetén a következő Dijsktra algoritmussal oldhatjuk meg. A pontokat egy d érték szerinti *Q* módosítható prioritási sorban tároljuk.

Dijskstra algoritmusa

```
Kezd(G,s)
 for(v in V)
    {d(v):=INF
     Apa(v):=0
     Kesz(v):=0}
 d(s):=0

Kozelit(G,u,v,Q)
 if d(v)>d(u)+c(u,v)
    then {d(v):=d(u)+c(u,v)
          Modosit(Q,v)
          Apa(v):=u}

Dijkstra(G,s)
 Kezd(G,s)
 Letesit(Q: ModPrisor)
 for(v in V)
    SorBa(Q,v)
 while(Elemszam(Q)>0)
    {SorBol(Q,u)
    Kesz(u):=1
    for(v in KiEl(G,u))
        {if Kesz(v)=0
           then Kozelit(u,v)}}
```

Példa

Hajtsuk végre az 1 pontból a Dijkstra algoritmust az alábbi gráfra. (A mátrixban a c_{ij} érték az (i,j) él hossza, ∞ ha nincs él.)

$$\begin{pmatrix} \infty & 2 & \infty & 1 & 8 & \infty \\ 2 & \infty & 1 & 4 & 1 & 5 \\ 2 & \infty & 2 & 1 & 1 & 7 \\ \infty & 1 & 4 & \infty & 2 & 7 \\ 1 & \infty & 1 & 4 & \infty & 1 \\ \infty & 1 & \infty & 4 & 2 & 1 \end{pmatrix}$$

$Kesz = \{1\}, d(2) = 2, Apa(2) = 1, d(4) = 1, Apa(4) = 1, d(5) = 8, Apa(5) = 1,$
$Kesz = \{1,4\}, d(3) = 5, Apa(3) = 4, d(5) = 3, Apa(5) = 4, d(6) = 8, Apa(6) = 4,$
$Kesz = \{1,4,2\}, d(3) = 3, Apa(3) = 2, d(6) = 6, Apa(6) = 2,$
$Kesz = \{1,4,2,3\},$
$Kesz = \{1,4,2,3,5\} d(6) = 4, Apa(6) = 5,$

Floyd Warshall algoritmus

A feladat egy súlyozott gráfban minden pontpárra a legrövidebb utak megkeresése. Az input a súlyozott gráf. Outputként egy mátrixot adunk meg, amely a legrövidebb utakat tartalmazza, továbbá egy segédmátrixot, amely minden pontpárra tartalmazza a legrövidebb út első pontját, így a mátrix alapján a legrövidebb út egyszerűen megkapható.

A feladatot dinamikus programozással oldjuk meg. Legyen $V = 1 \dots n$ és $c(i,j) = \infty$, ha $(i,j) \notin E$, legyen $G(i,j) = c(i,j)$ ha $i \neq j$ és $G(i,i) = 0$, $i = 1,\dots,n$.

Részproblémákra bontás:

$\forall i,j \in \{1,\dots,n\}$-ra és $\forall k \in \{0,1,\dots,n\}$-re legyen $D^k(i,j)$=az i-ből j-be vezető olyan utak hosszának minimuma, amelyek legfeljebb az $\{1,\dots,k\}$ belső pontokon mennek keresztül. Ekkor $D^n(i,j) = \delta(i,j)$ és $D^0(i,j) = G(i,j)$.

Rekurzív összefüggés ($k \geq 1$):

$$D^k(i,j) = \min\{D^{k-1}(i,j), D^{k-1}(i,k) + D^{k-1}(k,j)\}$$

,

Megjegyzés: A rekurzió kiszámítása egyetlen tömbben megoldható, mert $D^{k-1}(i,k) = D^k(i,k)$ és $D^{k-1}(k,j) = D^k(k,j)$

Floyd Warshall algoritmus

```
FloydWarshall(G)
for i=1 to n
   {for j=1 to n
      {D[i,j]:=G[i,j]
      if D[i,j]<INF then Elso[i,j]:=j}}
for k=1 to n
    {for i=1 to n
        {for j=1 to n
           If D[i,k]+D[k,j]<D[i,j]
              then {D[i,j]:=D[i,k]+D[k,j]
                   Elso[i,j]:=Elso[i,k]}}}
```

Futási idő: $\Theta(n^3)$

A G = (V,E) gráf tranzitív lezártja az a $G^* = (V, E^*)$ gráf, ahol $E^* = \{(u, v) : u \rightsquigarrow v\}$.

Egy lehetséges algoritmus G^* kiszámítására: vegyük azt a gráfot, amelyben minden létező él súlya 1, és alkalmazzuk a Floyd-Warshall algoritmust. Az i és j pontok között akkor és csak akkor van út G-ben, ha távolságuk nem ∞.

Megjegyzés: Azonban a Floyd-Warshall algoritmus egyszerű módosításával (Boolean értékeket használva a távolságok helyett) hatékonyabb megoldást kapunk.

Példa

Tegyük fel, hogy egy hivatalban minden hivatalnok megvesztegethető. Nem mindegyiküket lehet közvetlenül megvesztegetni a magasabb hivatalban levő embereket csak az vesztegetheti meg, aki rendelkezik megfelelő ajánlólevéllel, amit az ajánlólevél irójának megvesztegetésével lehet megszerezni. Egy jól értesült vállalkozó tudja kit mennyibe kerül megvesztegetni és azt is, hogy az egyes emberek milyen ajánlóleveleket fogadnak el. A hivatal vezetőjét szeretné megvesztegetni a lehető legkisebb teljes költség megfizetése mellett. Adjunk meg egy eljárást, amellyel meghatározhatja kiket kell megvesztegetnie!

Megoldás

Vegyünk egy gráfot, amelynek pontjai a vállalkozó és a tisztviselők. A vállalkozóból él megy azokba a tiszviselőkbe, akiket közvetlenül megvesztegethet, továbbá minden tisztviselőből él megy azokba a tisztviselőkbe, akik elfogadják az ajánlólevelét. A gráfban a csúcsoknak van súlya, a vállalkozóé 0, a tisztviselőké a megvesztegetés összege. A cél a vállalkozóból a hivatal vezetőjébe egy legrövidebb út megkeresése, ahol az út hossza a benne levő csúcsok súlyainak összege.

Ez megoldható a legrövidebb út algoritmusok módosításával is, de visszavezethetjük a legrövidebb út problémára. Minden csúcsot helyettesítsünk egy éllel, amelynek súlya a csúcs súlya. Az él kezdőpontjába mennek azok az élek, amelyek a csúcsba mentek, az él végpontjából mennek azok az élek, amelyek a csúcsból mentek. (Szemléletesen ez azt jelenti, hogy az új csúcsok a megvesztegetések kezdetei és végei, a kezdet előfeltétele az ajánlólevél, és ajánlólevelet az ember csak a vesztegetés végén kap.