BABEŞ-BOLYAI TUDOMÁNYEGYETEM KOLOZSVÁR
MATEMATIKAI ÉS INFORMATIKAI KAR

Kása Zoltán

Gráfalgoritmusok

2007

Mottó helyett

Königsberget vissza kellene szerezni, német ismerőseim közül senki nem lelkesedett az ötletért. Féltek. A német nagyhatalmi ambíciók látszatától féltek. A legmerészebbek is csak odáig mentek el, hogy ennek a generációnak az életében ez nem lehetséges. Akkor hát ne a németek kapják vissza a várost. Kapja vissza a németek európai énje, vagy még egyszerűbb, ha Európa lenne a kedvezményezett, s itt nyilván vita nyitható arról, hogy intézményesen meg gyakorlatilag ez mit jelenthet. Mit jelent „Európa"? Az Európai Uniót netalán? Ez távolról sem ilyen egyszerű, ámde még ha így vélekednénk is, tudnunk kell, hogy az Unió semmi jelét nem adta mostanig, hogy rendelkezne bármiféle koncepcióval Königsberg jövőjéről, vagy hogy komoly szándéka lenne kiterjeszteni rá a felségterületét, vagy ezt akár csak felvetni is. Pedig ez lenne a legegyszerűbb. Egy uniós-orosz közös protektorátus, örökös semlegességű terület formájában európai kulturális fővárossá tenni Königsberget, amely nemcsak a művészetek, a művészek és a filozófusok városállama lehetne, hanem a kereskedelem, bankrendszer és technológiai transzfer szabad kikötője, egy történelmi tárgyalóterem, az ismerkedés, együttgondolkodás és együttműködés szabadtéri színpada, egy baltikumi Hong-Kong, Oroszország és az Unió közeledésének politikai laboratóriuma.

. . .

Magam is szívesen lennék egy (nemcsak virtuális, hanem) valóságos Königsberg polgára. Mondhatnám: alig várom, hogy odaköltözhessek és Euler nyomán végigjárjam a hidakat, a helyszínen tájékozódva arról, hogy gráfelméletileg mi is velük a pontos helyzet.

> *Szőcs Géza: Kalinyin elvtárs Königsbergben, Helikon, XVII. évf. 2006, 15. (461.) szám – augusztus 10.*

Tartalomjegyzék

1. fejezet

Néhány, gráfokkal megoldható feladat

Szürke minden elmélet, barátom,
de zöld az élet aranyfája.
Goethe: Faust

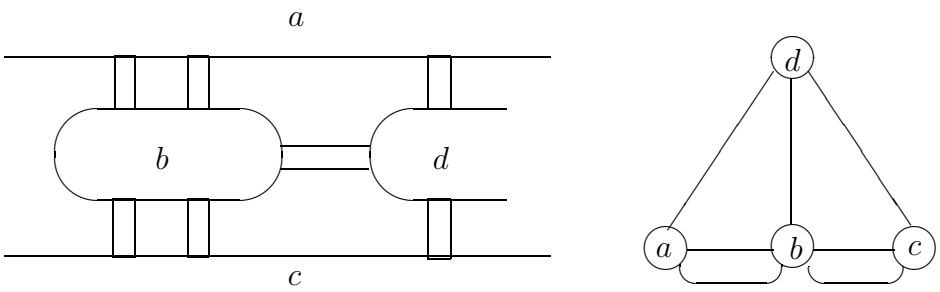
1.1. A königsbergi hidak problémája

Az első olyan feladat, amelyet gráfok segítségével oldottak meg, a königsbergi hidak problémája. A régi Poroszország Königsberg városában (ma Oroszország része, a szovjetidőkben Kalinyingrád volt) a Pregel folyón összesen hét híd van (1.1. ábra), amelyek a folyó két partját, egy szigetet és egy félszigetet kötnek össze. A város lakói olyan sétát szerettek volna tenni, ha lehet, hogy mind a hét hídon átsétáljanak, de csak egyszer mindegyiken, és visszajussanak a kiindulópontba. Nem sikerült, ezért a kor híres matematikusához, Eulerhez fordultak. Euler bebizonyította, hogy a kívánt séta nem leghetséges.

Sematikusan a hét híd a 1.2.a. ábrán látható módon ábrázolható, ahol az a, b, c és d betűk a partokat jelentik. Rajzoljuk le a négy partnak megfelelő betűket egy-egy körbe, és kössük össze ezeket annyi vonallal, ahány hiddal vannak összekötve. A 1.2.b. ábra rajzát kapjuk.

Az eredeti feladat most úgy fogalmazható át, hogy valamelyik, betűvel jelölt pontból kiindulva járjuk be ezen az ábrán az összes vonalat úgy, hogy mindegyiken csak egyszer menjünk át, és jussunk vissza a kiinduló pontba. Könnyű belátni, hogy ez nem lehetséges. Miután átmentünk egy vonalon, töröljük azt. Így, például ha a d-n áthaladunk, két vonalat törlünk, és a harmadikon már nem juthatunk át. Ez akkor is igaz, ha d-vel

1.1. ábra. Königsberg a hidakkal (*Forrás: Wikipédia*)



a **b**

1.2. ábra. Königsberg a hidakkal – sematikusan

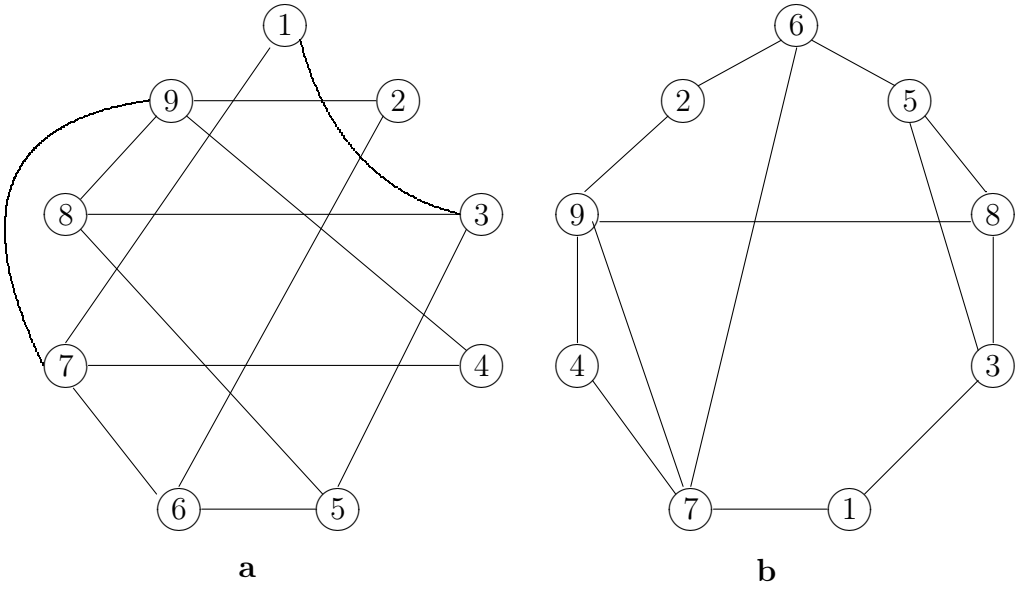
kezdünk, mert vagy újraérintve, két vonalat törlünk, és akkor már nem fejezhetjük be sétánkat ennél a pontnál, vagy itt fejezve be, nem tudtunk minden vonalat érinteni.

A 1.2.a. ábrán egy gráfnak nevezett rajz látható. Ennek segítségével könnyebben beláttuk, hogy a feladat megoldhatatlan.

A gráfnak csúcsai és élei vannak. Az élek a csúcsokat kötik össze. Ezek lehetnek egyenes szakaszok vagy tetszőleges vonalak.

1.2. Egy szórakoztató feladat

Helyezzük el az 1, 2, 3, ..., 9 számokat egy körön úgy, hogy bármely két szomszédos szám összege ne legyen osztható se 3-mal, se 5-tel, se 7-tel! Először helyezzük el a számokat sorrendben egy körön, majd kössük össze egy-egy vonallal azokat, amelyeknek összege kielégíti a feladat feltételeit (azaz nem osztható se 3-mal, se 5-tel, se 7-tel) (1.3.a ábra). Ha egy számból elindulva és mindig vonalon haladva, minden számot érintve, visszajutunk az eredeti számhoz, akkor a feladat megoldható. Próbáljuk meg ennek megfelelően átrajzolni az ábránkat! Az eredmény a 1.3.b ábra.



1.3. ábra. Számok a körön

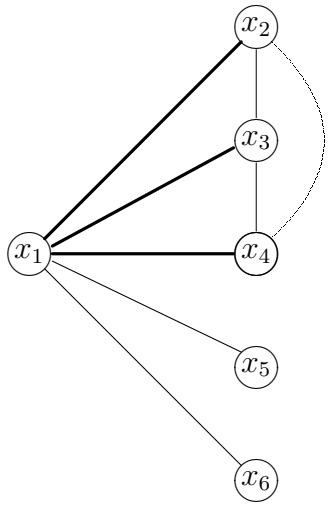
1.3. Még egy szórakoztató feladat

Bizonyítsuk be, hogy egy legalább hattagú társaságban mindig vannak hárman, akik kölcsönösen ismerik egymást vagy kölcsönösen nem ismerik egymást.

Elég, ha a feladatot pontosan 6 emberre bizonyítjuk, és legyenek ezek x_1, x_2, x_3, x_4, x_5 és x_6. Húzzunk egy piros vonalat x_i és x_j között, ha ismerik egymást és kéket, ha nem ismerik egymást. Egy olyan gráfot kapunk, amelynek legalább hat csúcsa van, és ezeket piros vagy kék élekkel

kötünk össze. Így a feladatunkat átfogalmazhatjuk: egy legalább hatcsúcsú gráfban, amelyben bármely két csúcsot piros vagy kék éllel kötünk össze, mindig van piros vagy kék háromszög (azaz, három olyan csúcs, amelyeket azonos színű élek kötnek össze).

A bizonyításhoz válasszuk például az x_1 csúcsot (1.4. ábra). A belőle kiinduló élek közül legalább három azonos színű, például piros (az ábrán vastagított vonalak). Legyenek ezek az x_1 csúcsot az x_2, x_3, x_4 csúcsokkal összekötő élek. Ha az x_2 és x_3, vagy az x_2 és x_4, vagy az x_3 és x_4 csúcsokat összekötő élek valamelyike piros, akkor találtunk piros háromszöget (x_1, x_2, x_3 vagy x_1, x_2, x_4 vagy x_1, x_3, x_4). Ha ezek között egy piros sincs, akkor mind kék, és így egy kék háromszög keletkezik (x_2, x_3, x_4).



1.4. ábra. Kölcsönös ismeretség

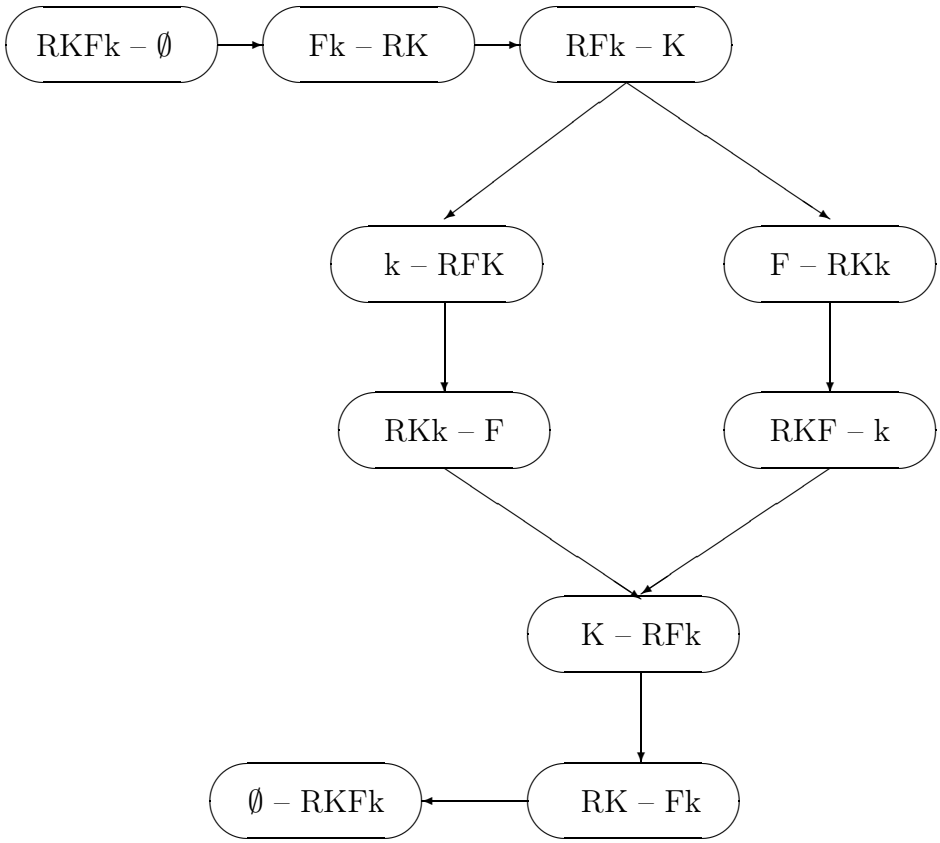
1.4. A révész, farkas, kecske és káposzta problémája

Hogyan viheti át a révész a farkast, a kecskét és a káposztát egy folyó egyik partjáról a másikra, ha csupán egyetlen csónak áll rendelkezésre, és egyszer csak egy állatot vagy a káposztát viheti magával. Egyik parton sem maradhat a kecske és a farkas révész nélkül, és hasonlóképpen a káposzta sem a kecskével.

Próbáljuk meg lerajzolni a lehetséges állapotokat (bal és jobb part „tartalma")! A következő jelöléseket használjuk: R – révész, F – farkas,

K – kecske, k – káposzta. Az állapot jelölésére megadjuk a bal és joob part tartalmát, pl. RK – Fk azt jelenti, hogy a bal parton van a révész és a kecske, a jobb parton pedig a farkas és a káposzta. Nyíllal jelöljük, ha egy adott állapotból át lehet menni egy másikba. Az összes lehetőségeket figyelembe véve, az 1.5. ábrát kapjuk. Innen rögtön látszik, hogy két megoldás van, azaz az $RKFk$ –$\emptyset$ (mindannyian a bal parton) állapotból két „út" (nyilak egymásutánisága) vezet az $\emptyset$ – $RKFk$ (mindannyian a jobb parton) állapotba.

Az eddigiektől eltérően itt az éleknek irányításuk van, ezért irányított élekről beszélünk, a megfelelő gráf pedig irányított gráf.



1.5. ábra. A révész, farkas, kecske és káposzta problémája

2. fejezet

Alapfogalmak

Hároméven aluliaknak nem ajánlott. Az apró alkotórészek könnyen beszippanhatók vagy lenyelhetők.

(A kindertojás használati utasításából)

2.1. A gráf fogalma

A gráf fogalmának értelmezéséhez szükségünk van a következő két jelölésre.

$A \times B = \{(a, b) \mid a \in A, b \in B\}$ – rendezett párok halmaza (az ún. Descartes-szorzat)

$A \otimes B = \{\{a, b\} \mid a \in A, b \in B$ vagy $a \in B, b \in A\}$ – rendezetlen párok halmaza

Megjegyzés: Itt $\{a, b\}$ nem halmaz, mivel a és b nem feltétlenül különbözőek.

A gráf

Gráfnak nevezzük a
$G = (V, E, \mathcal{G})$ rendezett hármast, ahol V *csúcsok* (vagy *szögpontok* esetleg *pontok*) nem üres halmaza, E *élek* halmaza, $\mathcal{G} : E \to V \otimes V$

Ha pontosabbak akarunk lenni, akkor a fenti jelölés helyett a következőt használjuk:

$G = (V(G), E(G), \mathcal{G}(G))$.

A G gráf *rendje* $n = |V|$, *nagysága* pedig $m = |E|$. G egy (n, m) gráf.

Ha $\mathcal{G}(e_1) = \mathcal{G}(e_2)$, akkor e_1 és e_2 *párhuzamos* vagy *többszörös* élek. Ha

$\mathcal{G}(e) = \{a, a\}$, akkor az e él *hurokél*.

Ha $\mathcal{G}(e) = \{a, b\}$, akkor azt mondjuk, hogy az a és b csúcsokat az e él köti össze, a és b *szomszédosak*, az e él *illeszkedik* az a és b csúcsokra, az a és b csúcsok az e él *végpontjai*.

Az a és b csúcsokra illeszkedő (párhuzamos) élek halmaza:

$$\mathcal{G}^{-1}(a, b) = \{e \in E \mid \mathcal{G}(e) = \{a, b\}\}.$$

Legyen x a G gráf egy csúcsa. Jelöljük $N_G(x)$-szel vagy csak $N(x)$-szel az x-szel szomszédos csúcsok halmazát:

$$N_G(x) = \{y \in V(G) \mid \exists e \in E(G), \mathcal{G}(e) = \{x, y\}\}$$

vagy

$$N_G(x) = \{y \in V(G) \mid \mathcal{G}^{-1}(x, y) \neq \emptyset.\}$$

A G gráfban az x-hez illeszkedő élek (amelyek nem hurokélek) halmaza:

$$I_G(x) = \{e \in E(G) | \exists y \in V(G), y \neq x, \mathcal{G}(e) = \{x, y\}\}$$

Az x-hez illeszkedő hurokélek halmaza:

$$L_G(x) = \{e \in E(G) | \mathcal{G}(e) = \{x, x\}\}$$

Az x csúcs *fokszáma* vagy *foka*, amelyet $\varphi(x)$-szel jelölünk, az x-hez illeszkedő élek száma (a hurokéleket kétszer számolva):

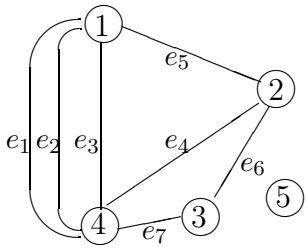
$$\varphi(x) = |I_G(x)| + 2|L_G(x)|.$$

Ha $\varphi(x) = 0$, akkor x *izolált* csúcs. Ha $\varphi(x) = 1$, akkor x *levél*.

Egy többszörös éleket és hurokéleket nem tartalmazó gráfot *egyszerű gráfnak* nevezzük. Ha G egyszerű gráf, akkor $|\mathcal{G}^{-1}(a, b)| \leq 1$ tetszőleges $a, b \in V$ csúcsokra, és $\mathcal{G}^{-1}(a, a) = \emptyset$ tetszőleges $a \in V$ csúcsra, tehát $\mathcal{G}(e) = \{a, b\}$ helyett egyszerűen írhatunk $\{a, b\}$-t, amely a megfelelő élt jelenti. Ekkor a gráf is jelölhető egyszerűbben: $G = (V, E)$.

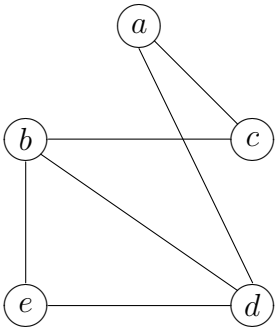
Egyszerű gráfban az x *fokszáma* vagy *foka*, amelynek jele szintén $\varphi(x)$ vagy $\varphi_G(x)$, az $N_G(x)$ halamz elemszáma: $\varphi(x) = |N_G(x)|$.

Példák.



G_1 gráf

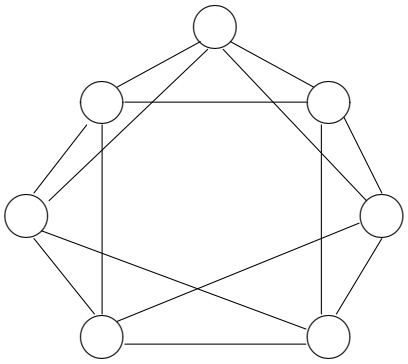
$V(G_1) = \{1, 2, 3, 4, 5\}$, $E(G_1) = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$,
$\mathcal{G}(e_1) = \mathcal{G}(e_2) = \mathcal{G}(e_3) = \{1, 4\}$, $\mathcal{G}(e_4) = \{2, 4\}$, $\mathcal{G}(e_5) = \{2, 1\}$,
$\mathcal{G}(e_6) = \{2, 3\}$, $\mathcal{G}(e_7) = \{3, 4\}$.
$\varphi(1) = 4,\ \varphi(2) = 3,\ \varphi(3) = 2,\ \varphi(4) = 5,\ \varphi(5) = 0.$



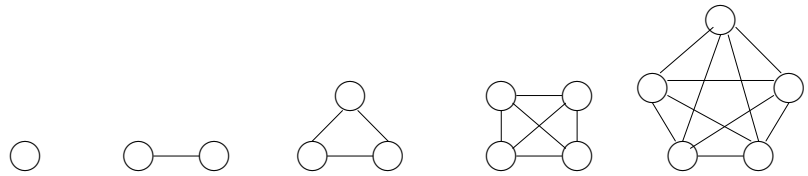
G_2 egyszerű gráf

$V(G_2) = \{a, b, c, d, e\}$,
$E(G_2) = \big\{\{a, c\}, \{a, d\}, \{b, c\}, \{b, e\}, \{b, d\}\{e, d\}\big\}$

Ha egy gráf minden fokszáma azonos, például r, akkor a gráf *reguláris* vagy *r-reguláris*. A következő gráf egy (7,14) 4-reguláris gráf.

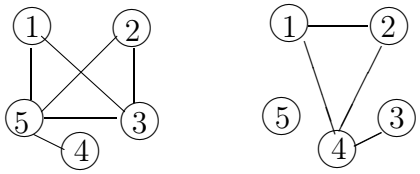


Ha egy egyszerű gráfban bármely két csúcsot él köt össze, akkor a gráf *teljes gráf*. Az n-csúcsú teljes gráf jele: K_n.



K_1 K_2 K_3 K_4 K_5

A $\overline{G} = (\overline{V}, \overline{E})$ egyszerű gráf a $G = (V, E)$ egyszerű gráf *komplementuma* vagy *komplementere*, ha $\overline{V} = V$ and $\overline{E} = \big\{\{a, b\} \mid \{a, b\} \notin E\big\}$.



G $\overline{G}$

Ha a G egyszerű gráf n-csúcsú, akkor $E(G) \cup E(\overline{G}) = E(K_n)$.

A G_1 és G_2 gráfok *izomorfak*, ha létezik egy bijektív fügvény

$$\psi \,:\, V(G_1) \to V(G_2),$$

úgy, hogy

$$\text{ha } \{a, b\} \in E(G_1), \text{ akkor } \{\psi(a), \psi(b)\} \in E(G_2).$$

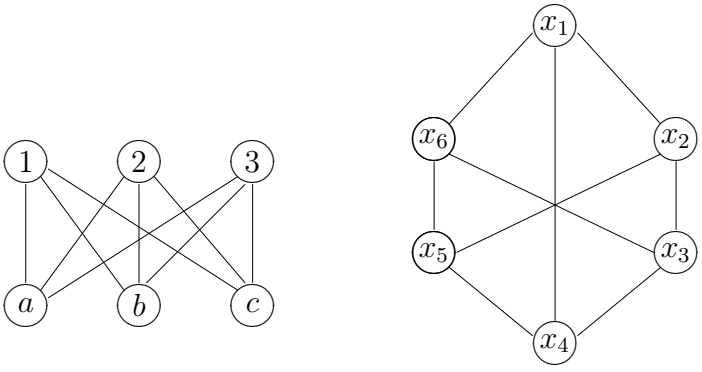
Az izomorfizmust tetszőleges gráfokra is értelmezhetjük. Két G_1 és G_2 gráf izomorf, ha létezik egy

$$\psi \,:\, V(G_1) \to V(G_2)$$

bijektív föggvény úgy, hogy

$$|\mathcal{G}_1^{-1}(a, b)| = |\mathcal{G}_2^{-1}(\psi(a), \psi(b))| \text{ minden } a, b \in V(G_1)\text{-re.}$$

Példa izomorf gráfokra.



G_1 G_2

A ψ függvény:

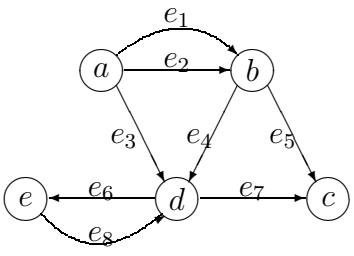
x	1	2	3	a	b	c
$\psi(x)$	x_1	x_5	x_3	x_2	x_6	x_4

Izomorf gráfokban $\varphi(x)$=$\varphi(\psi(x))$ minden $x \in V(G_1)$-re.

2.2. Irányított gráfok

Irányított gráfnak nevezzük a
$\vec{G} = (V, E, \vec{\mathcal{G}})$ rendezett hármast, ahol
V a *csúcsok* (vagy *szögpontok* vagy *pontok*) halmaza,
E az *irányított élek* halmaza és
$\vec{\mathcal{G}} : E \to V \times V$

Ha $e \in E$ és $(a, b) \in \vec{\mathcal{G}}(e)$, akkor a az e él kezdőpontja, b pedig az e él végpontja. Ha egy élnek a kezdő- és végpontja egybeesik, akkor az az él hurokél.



Ebben az irányított gráfban az e_1 és e_2 élek párhuzamosak, de e_6 és e_8 nem. Ha egy irányított gráfban nincsenek párhuzamos élek és hurokélek, akkor az egyszerű irányított gráf.

Legyen $\vec{G}$ irányított gráf. Ekkor

$$N^{\text{be}}_{\vec{G}}(y) = \{x \in V(\vec{G}) \mid \vec{\mathcal{G}}^{-1}(x, y) \neq \emptyset\}$$

az y-ba befutó élek kezdőpontjainak halmaza

$$N^{\text{ki}}_{\vec{G}}(y) = \{z \in V(\vec{G}) \mid \vec{\mathcal{G}}^{-1}(y, z) \neq \emptyset\}$$

az y-ból kifutó élek véppontjainak halmaza.
Egy irányított gráfban az x csúcs *be-foka* az x-be befutó élek száma (jelölése $\varphi^{\text{be}}(x)$), az x csúcs *ki-foka* az x-ből kifutó élek száma (jelölése $\varphi^{\text{ki}}(x)$). Ha egyszerű irányított gráfról van szó, akkor:

$$\varphi^{\text{be}}(x) = |N^{\text{be}}(x)|$$

$$\varphi^{\text{ki}}(x) = |N^{\text{ki}}(x)|.$$

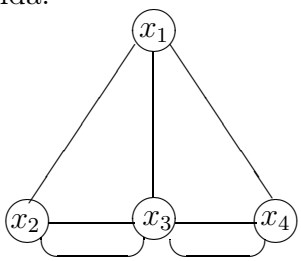
2.3. Gráfok ábrázolása

1) – geometriai ábrázolás

2) – szomszédsági (adjacencia) mátrixszal
$G = (E, V, \mathcal{G})$, $V = \{x_1, x_2, \ldots, x_n\}$
$A = (a_{ij})_{i,j=\overline{1,n}}$ a szomszédsági mátrix, ahol

$$a_{ij} = \begin{cases} |\mathcal{G}^{-1}(x_i, x_j)| & \text{ha } i \neq j \\ 2|\mathcal{G}^{-1}(x_i, x_j)| & \text{ha } i = j \end{cases}$$

Példa:



$$A = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 2 & 0 \\ 1 & 2 & 0 & 2 \\ 1 & 0 & 2 & 0 \end{pmatrix}$$

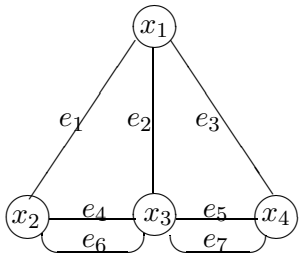
$$\varphi(x_i) = \sum_{j=1}^{n} a_{ij}, \text{ minden } i = 1, 2, \ldots, n$$

Az egyszerű gráf szomszédsági mátrixa csak 0 és 1 számokat tartalmaz. Irányított gráf esetében a definíció hasonló.

3) – illeszkedési (incidencia) mátrixszal

$G = (E, V, \mathcal{G})$, $V = \{x_1, x_2, \ldots, x_n\}, E = \{e_1, e_2, \ldots, e_m\}$
$B = (b_{ij})_{i=\overline{1,n}, j=\overline{1,m}}$,

$$b_{ij} = \begin{cases} 1 & \text{ha } x_i \text{ illeszjedik } e_j\text{-hez és } e_j \text{ nem hurokél} \\ 2 & \text{ha } x_i \text{ illeszjedik } e_j\text{-hez és } e_j \text{ hurokél} \\ 0 & \text{ha } x_i \text{ nem } e_j\text{-hez.} \end{cases}$$



$B =$

	e_1	e_2	e_3	e_4	e_5	e_6	e_7
x_1	1	1	1	0	0	0	0
x_2	1	0	0	1	0	1	0
x_3	0	1	0	1	1	1	1
x_4	0	0	1	0	1	0	1

4) – listával

a) Minden csúcsnak felsoroljuk a szomszédjait.

x_1	x_2	x_3	x_4		
x_2	x_1	x_3	x_3		
x_3	x_1	x_2	x_2	x_4	x_4
x_4	x_1	x_3	x_3		

Használhatunk láncolt listákat is.

b) A listákat egymás után írjuk egy-egy $*$-gal elválasztva, a végére két csillagot téve.

x_2	x_3	x_4	$*$	x_1	x_3	x_3	$*$	x_1	x_2	x_2	x_4	x_4	$*$	x_1	x_3	x_3

$*$	$*$

c) A $*$-okat elhagyjuk, és még egy listát használunk, amelyikben az egyes listák kezdőindexeit adjuk meg.

x_2	x_3	x_4	x_1	x_3	x_3	x_1	x_2	x_2	x_4	x_4	x_1	x_3	x_3

1	4	7	12

A második lista elemei az egyes listák kezdőelemeire mutatnak a következőképpen:

x_2	x_3	x_4	x_1	x_3	x_3	x_1	x_2	x_2	x_4	x_4	x_1	x_3	x_3
↑			↑			↑					↑		

2.4. Gráfok egyszerű tulajdonságai

1) $G = (V, E, \mathcal{G}), |E(G)| = m$, akkor $\sum_{x \in V(G)} \varphi(x) = 2m$.

2) A páratlan fokszámú csúcsok száma páros.

3) Bármely egyszerű gráfban mindig van legalább két azonos fokú csúcs.

A bizonyítást a skatulyaelv segítségével végezzük. Ha $|V(G)| = n$, akkor a fokszámok $0, 1, \ldots, n-2$ vagy $1, 2, \ldots, n-1$ lehetnek. Mert ha van izolált csúcs, nem lehet olyan, amely minden mással össze van kötve. Tehát összesen $n-1$ különböző fokszám van és n csúcs, ahonnan azonnal következik az állítás.

A gráf minimális és maximális fokszámát a következőképpen értelmezzük:

$\delta(G) = \min_{x \in V(G)} \varphi(x)$ minimális fokszám G-ben

$\Delta(G) = \max_{x \in V(G)} \varphi(x)$ maximális fokszám G-ben.

Részgráfok

A $H = (V(H), E(H), \mathcal{H})$ gráf *részgráfja* $G = (V(G), E(G), \mathcal{G})$ gráfnak, ha $V(H) \subseteq V(G)$, $E(H) \subseteq E(G)$, $\mathcal{H} = \mathcal{G}|_{E(H)}$.

Egyszerű gráfokra az értelmezés hasonló. Ha $V(H) = V(G)$, akkor H feszítő részgráf.

2.5. Séta, vonal, út

A $G = (V, E, \mathcal{G})$ gráfban a

$$W : v_0, e_1, v_1, e_2, v_2, \ldots v_{i-1}, e_i, v_i, \ldots v_{n-1}, e_n, v_n, \qquad n \geq 0$$

váltakozóan csúcsokból és élekből álló sorozat, ahol $\mathcal{G}(e_i) = \{v_{i-1}, v_i\}, i = 1, 2, \ldots, n$, *séta*. Az élek és csúcsok nem feltétlenül különbözőek. Egy ilyen sétát általában v_0–v_n sétának nevezünk. Egyszerű gráfban elég csak a csúcsokat felsorolni: $v_0, v_1, \ldots, v_n$, mivel $|\mathcal{G}^{-1}(v_{i-1}, v_i)| = 1$ minden $i = 1, 2, \ldots, n$ értékre. Itt n (a séta éleinek a száma) a séta *hossza*.

Sajátos séták:

vonal: ha élei nem ismétlődnek,

út: ha csúcsai nem ismétlődnek.

Tétel. *Bármely v_0–v_n séta tartalmaz v_0–v_n utat.*

Bizonyítás. Indukcióval a séta hossza szerint. Ha $n = 1$, akor a séta út.

Tegyük fel, hogy a tétel igaz minden n-nél kisebb értékre, és legyen

$$W : v_0, e_1, v_1, \ldots, v_{i-1}, e_i, v_i, \ldots, v_{j-1}, e_j, v_j, \ldots e_n, v_n$$

egy olyan séta, amelyik nem út. Tehát léteznek a v_i és v_j csúcsok úgy, hogy $v_i = v_j$. A $W' : v_0, e_1, v_1, \ldots, v_{i-1}, e_i, v_i, e_{j+1}v_{j+1}, \ldots e_n, v_n$ séta (amelyet a v_i–v_j séta törlésével kaptunk) rövidebb W-nél. tehát, az indukciós feltétel alapján W'-re igaz a tétel, azaz létezik v_0–v_n út, amely W-ben is út.

A u–v séta *zárt séta*, ha $u = v$. Ha egy séta nem zárt, akkor *nyított*. Hasonlóan értelmezhető a *zárt vonal* is. Ha egy zárt vonal minden csúcsa különböző, akkor *kör* a neve.

Összefüggőség

Egy gráfot *összefüggőnek* nevezünk, ha bármely két csúcsa között létezik út. Ha egy gráf nem összefüggő, akkor összefüggő *komponensekből* áll. Jelöljük $\kappa(G)$-vel a G gráf komponenseinek a számát.

Algoritmus összefüggő komponens keresésére

Legyen $x \in V(G)$. Rekurzívan értelmezzük a következő halmazokat:

$$U_0 := \{x\}$$

$$U_i := U_{i-1} \cup \left(\bigcup_{y \in U_{i-1}} N(y) \right)$$

$$\exists k : U_k = U_{k-1} = U.$$

Az U halmaz indukálta gráf egy komponens.

KOMPONENS(G, x)
1. $U := \{x\}$
2. $V := U \cup N(U)$
3. **while** $U \neq V$ **do**
4. $\quad U := V$
5. $\quad V := U \cup N(U)$
6. **return** U

Irányított gráf esetében értelmezzük a következőket:
– *irányított séta*:

$$v_0, e_1, v_1, e_2, v_2, \ldots v_{i-1}, e_i, v_i, \ldots v_{n-1}, e_n, v_n, \qquad n \geq 0,$$

ahol $\mathcal{G}(e_i) = (v_{i-1}, v_i), i = 1, 2, \ldots, n$.

Hasonlóan értelmezhetők az irányított vonal, irányított út, zárt irányított vonal, irányított kör fogalmak.

– *lánc*

$$v_0, e_1, v_1, e_2, v_2, \dots v_{i-1}, e_i, v_i, \dots v_{n-1}, e_n, v_n, \qquad n \geq 0,$$

ahol $\mathcal{G}(e_i) = (v_{i-1}, v_i)$ vagy $\mathcal{G}(e_i) = (v_i, v_{i-1})$, minden $i = 1, 2, \dots, n$ értékre. Ha elhagyjuk az élek irányítását, akkot a lánc séta lesz.

Az összefüggőségre a következőket értelmezzük: – *gyengén összefüggő irányított gráf*, ha bármely u, v csúcspárra létezik irányított u–v út vagy irányított v–u út.

– *erősen összefüggő irányított gráf*, ha bármely u, v csúcspárra létezik irányított u–v út.

– *összefüggő irányított gráf*, ha bármely csúcspár között van lánc (azaz az irányítás elhagyásával összefüggő gráfot kapunk).

2.6. Súlyozott gráfok

Gráfok esetében értelmezzük a súlyozott gráf fogalmát. *Súlyozott gráf* $G = (V, E, \vec{\mathcal{G}}, \mathcal{W})$, ahol

$$\mathcal{W} : E \to \mathbf{R}.$$

Egy súlyozott P út hossza $l(P) = \sum_{e_i \in P} \mathcal{W}(e_i)$.

Ha egy gráf nem súlyozott, mindig tekinthető annak, hiszen minden élhez hozzárendelhetjük az 1 értéket, azaz $\mathcal{W}(e) = 1$ bármely e élre.

Ezek hasonlóképpen értelmezhetők az irányított gráfokban is.

Két csúcs közti $d(u, v)$ távolság értelmezése: a legrövidebb u–v út hossza.

$$d(u, v) = \min_{P \; u\text{–}v \text{ út}} l(P)$$

Gráf *távolsági mátrix*: $G = (V, E)$, $V = \{v_1, v_2, \dots, v_n\}$:

$$D = \left(d_{ij}\right)_{i,j=\overline{1,n}} \text{ ahol } d_{ij} = d(v_i, v_j)$$

Warshall-algoritmus távolsági mátrix meghatározására

Kezdetben értelmezzük a szomszédsági mátrix következő változatát:

$D_0 := (d_{ij}^{(0)})_{i,j=\overline{1,n}}$

$$\text{ahol } d_{ij}^{(0)} = \begin{cases} \mathcal{W}(v_i, v_j) & \text{ha } \{v_i, v_j\} \in E \\ 0 & i = j \\ \infty & \text{ha } \{v_i, v_j\} \notin E, i \neq j \end{cases}$$

Majd sorra kiszámítjuk a $D_1, D_2, \dots D_k \dots$ mátrixokat minden $k > 0$ értékre, ahol $D_k := (d_{ij}^{(k)})_{i,j=\overline{1,n}}$:

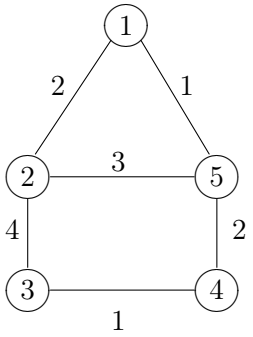
$d_{ij}^{(k)} := \min\left(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right)$ minden $i, j = 1, 2, \dots, n$ értékre.

Létezik egy olyan k_0 érték amelyre $D_{k_0-1} = D_{k_0}$, ez lesz a D távolsági mátrix.

Az algoritmus a következő:

WARSHALL(D_0)
1. $D := D_0$
2. **for** $k := 1$ **to** n **do**
3. **for** $i := 1$ **to** n **do**
4. **for** $j := 1$ **to** n **do**
5. $d_{ij} := \min(d_{ij}, d_{ik} + d_{kj})$
6. **return** D

Példa.



$$D_0 = \begin{pmatrix} 0 & 2 & \infty & \infty & 1 \\ 2 & 0 & 4 & \infty & 3 \\ \infty & 4 & 0 & 1 & \infty \\ \infty & \infty & 1 & 0 & 2 \\ 1 & 3 & \infty & 2 & 0 \end{pmatrix} \qquad D = \begin{pmatrix} 0 & 2 & 4 & 3 & 1 \\ 2 & 0 & 4 & 5 & 3 \\ 4 & 4 & 0 & 1 & 3 \\ 3 & 5 & 1 & 0 & 2 \\ 1 & 3 & 3 & 2 & 0 \end{pmatrix}$$

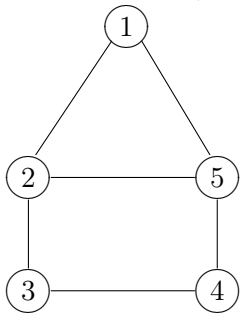
Gráfok csúcsainak bejárása

A körmentes összefüggő gráfot *fának* nevezzük. A tetszőleges körmentes gráf neve *liget*.

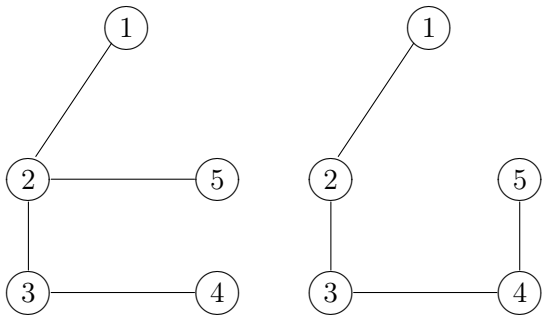
Ha egy fában kitüntetünk egy csúcsot, akkor *gyökeres fáról* beszáélünk. Ebben az esetben úgy tekintjük, hogy a fa irányított, azaz az élek minden úton a gyökértől a levél felé irányítottak.

Ha egy gráfnak minden csúcsát meg szeretnénk vizsgálni, akkor a gráf bejárásáról beszélünk. Két ismert gráfbejárási algoritmus létezik:

– *szélességi bejárás* (BFS – breadth-first search)
– *mélységi bejárás* (DFS – depth-first search)

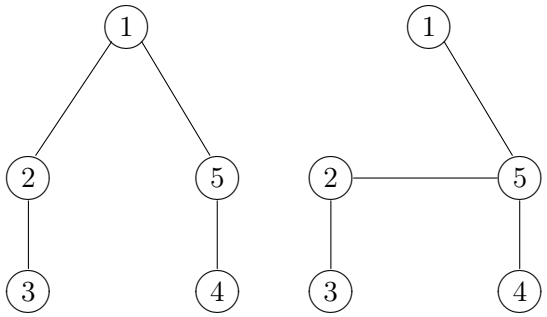


A. Szélességi bejárás – a csúcsokat úgy járjuk be, mint ahogy a hullám terjed.



3, 2, 4, 1, 5 (3-ból 2-be és 4-be, majd 2-ből 1-be és 5-be) vagy
3, 2, 4, 1, 5 (3-ból 2-be és 4-be, majd 2-be és 1-be, 4-ből 5-be)

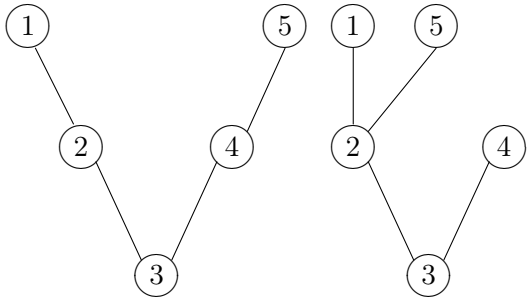
B. Mélységi bejárás – a csúcsokat úgy járjuk be, mint ahogy sétálunk.



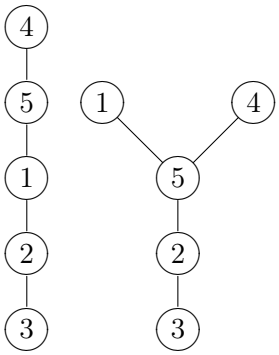
3, 2, 1, 5, 4
vagy

3, 2, 5, 1, (5,) 4
A bejárásokat fákkal ábrázolhatjuk.

A



B



A – „széles" fák
B – „mély" fák

3. fejezet

Legrövidebb utak

– Mennyire van ide Udvarhely?
– Légvonalban 10 km, de tudok egy rövidebb utat.

(székely vicc)

3.1. Legrövidebb utak nem irányított gráfokban

3.1.1. Moore algoritmusa

Ez az algoritmus nem súlyozott gráfokban meghatározza a legrövidebb utakat egy adott csúcsból az összes többi csúcsba. Az algoritmus A mélségi bejáráson alapul.

Legyen $G = (V, E)$ Egy adott gráf. A következő jelöléseket használjuk:
– u a kezdőcsúcs,
– $l(v)$ a v-nek az u-tól való távolsága,
– $p(v)$ a legrövidebb uton a v-t megelőző csúcs,
– Q egy sor (elemet hozzáadni egyik végén lehet, elemet kivenni a másikon).

Az algoritmusban használjuk még a következő műveleteket:
$v \to Q$ jelentése: beírja a v elemet a Q sorba,
$Q \to v$ jelentése: kiveszi a sor egy elemét (és ki is törli onnan), amelyet v-vel jelöl.

a) Algoritmus legrövidebb távolságok keresésére az u csúcsból kiindulva

MooreTávolság(G, u)
1. $l(u) := 0$

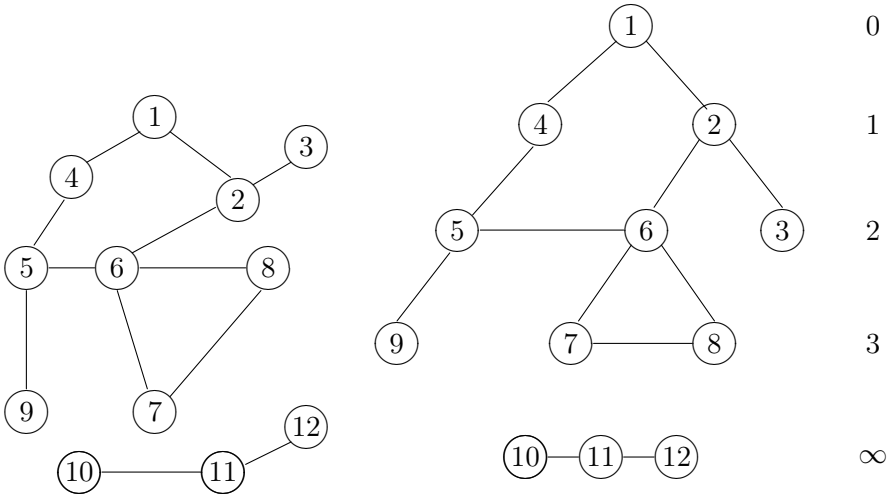
```
2.  for minden v ∈ V(G), v ≠ u csúcsra do
3.      l(v) := ∞
4.  legyen Q egy üres sor
5.  u → Q
6.  while Q ≠ ∅ do
7.          Q → x
8.          for minden y ∈ N(x) do
9.              if l(y) = ∞ then
10.                 p(y) := x
11.                 l(y) := l(x) + 1
12.                 y → Q
13. return l, p
```

b) Algoritmus a legrövidebb u–v utak keresésére

```
MooreÚt(l, p, v)
1.  k := l(v)
2.  u_k := v
3.  while k ≠ 0 do
4.          u_{k-1} := p(u_k)
5.          k := k − 1
6.  return u
```

Az eredményt az $u_0, u_1, \ldots, u_k$ vektor tartalmazza.

Irányított gráfok esetében: $N(x)$ helyett $N^{\text{ki}}(x)$



	1	2	3	4	5	6	7	8	9	10	11	12
l	0	1	2	1	2	2	3	3	3	∞	∞	∞
p		1	2	1	4	2	6	6	5			

Példa.
Ha $u = 1$ és $v = 8$, akkor $k := 3$, és az algoritmus lépései:
$u_3 := 8$, $u_2 := 6$, $u_1 := 2$, $u_0 := 1$.
Tehát a legrövidebb út 1 és 8 között: 1, 2, 6, 8.

3.2. Legrövidebb utak súlyozott gráfokban

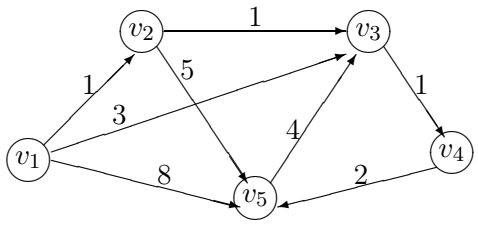
3.2.1. Legrövidebb utak egy csúcsból minden csúcsba

3.2.1.1. Dijkstra algoritmusa

DIJKSTRA(G, u)
1. $S := \{u\},\ T := V \setminus S,\ l(u) := 0$
2. **for** minden $v \in V,\ v \neq u$ **do**
3. $\quad l(v) := \infty$
4. $x := u$
5. **while** $T \neq \emptyset$ **do**
6. $\quad$ **for** minden $v \in N(x) \cap T$ **do**
7. $\quad\quad$ **if** $l(v) > l(x) + \mathcal{W}(x, v)$ **then**
8. $\quad\quad\quad l(v) := l(x) + \mathcal{W}(x, v)$
9. $\quad\quad\quad p(v) := x$
10. $\quad$ Legyen $x \in T$: $l(x) = \min_{y \in T} l(y)$
11. $\quad S := S \cup \{x\}$, $T := T \setminus \{x\}$
12. **return** l, p

Irányított gráfok esetében a 6. sorban $N(x)$ helyett $N^{\text{ki}}(x)$-et írunk.

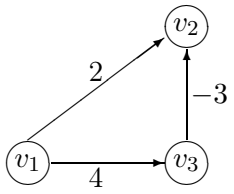
A legrövidebb utat ugyanaz az algoritmus adja meg, mint a Moore-algoritmus esetében.



A u kezdőcsúcs különböző értékeire, a következőket kapjuk.

	v_i	v_1	v_2	v_3	v_4	v_5
$u = v_1$	l_i	0	1	2	3	5
	p_i	$-$	v_1	v_2	v_3	v_4
$u = v_2$	l_i	∞	0	1	2	4
	p_i	$-$	$-$	v_2	v_3	v_4
$u = v_3$	l_i	∞	∞	0	1	3
	p_i	$-$	$-$	$-$	v_3	v_4
$u = v_4$	l_i	∞	∞	6	0	2
	p_i	$-$	$-$	v_5	$-$	v_4
$u = v_5$	l_i	∞	∞	4	5	0
	p_i	$-$	$-$	v_5	v_3	$-$

A Dijsktra-algoritmus nem működik negatív súlyok esetén. Erre példa lehet a következő gráf:



3.2.1.2. Ford algoritmusa

Legyenek $V = \{v_1, v_2, \ldots, v_n\}$ a súlyozott gráf csúcsai.

FORD(G)
1. $i := 1$
2. $l(v_1) := 0$
3. $l(v_i) := \infty$, minden $i = 2, 3, \ldots, n$.
4. **while** $i \leq n$ **do**
5. minden $v \in N(v_i) : l(v) := \min\big(l(v_i), l(v_i) + \mathcal{W}(v_i, v)\big)$
6. **if** $l(v)$ módosult és ($v = v_j$, ahol $j < i$)
7. **then** $i := j - 1$
8. $i := i + 1$
9. **return** l

Részletesebben:

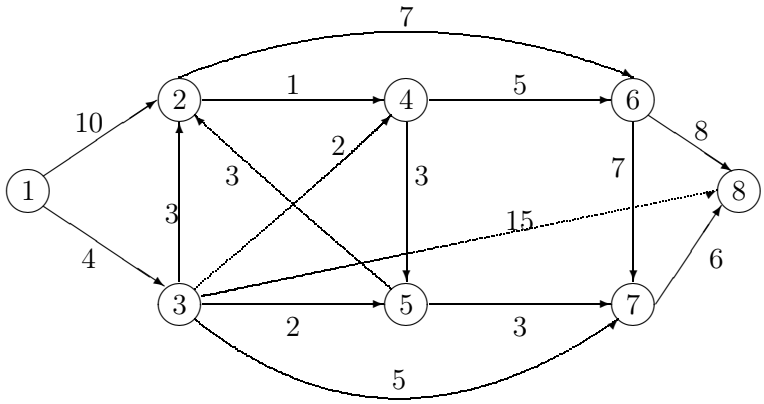
A p vektor egy eleme itt is mindig az előző csúcsra mutat. Ha x és y között nincs él, akkor $\mathcal{W}(x, y) = \infty$ értéket veszünk.

FORD(G)
1. $l(v_1) := 0$
2. $l(v_i) := \infty$, minden $i = 2, 3, \ldots, n$.
3. $i := 1$

```
4.  while i ≤ n do
5.      j := 1
6.      while j ≤ n do
7.          if l(v_j) − l(v_i) > W(v_i, v_j) then
8.              l(v_j) := l(v_i) + W(v_i, v_j)
9.              p(v_j) := v_i
10.             if j < i then
11.                 i := j − 1
12.                 j := n
13.         j := j + 1
14.     i := i + 1
15. return l, p
```

Irányított gráfokra $N(v_i)$ helyett $N^{\text{ki}}(v_i)$ szerepel.

Példa. Adott a következő súlyozott gráf:



amelynek szomszédsági mátrixa:

$$\begin{pmatrix} 0 & 10 & 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 7 & 0 & 0 \\ 0 & 3 & 0 & 2 & 2 & 0 & 5 & 15 \\ 0 & 0 & 0 & 0 & 3 & 5 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 7 & 8 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$u = 1$-re a Ford-algoritmus a következőt adja:

i :	1	2	3	4	5	6	7	8
l_i:	0	7	4	6	6	11	9	15
p_i:	0	3	1	3	3	4	3	7

Ez az algoritmus akkor is múködik, ha bizonyos élek negatív súlyúak, amennyiben a gráfban nincs negatív hosszúságú kör.

3.2.2. Legrövidebb utak minden csúcsból egy csúcsba

3.2.2.1. A Bellman–Kalaba-algoritmus

A módszer a szomszédsági mátrixot használja. Kiválasztjuk azt a csúcsot, amelybe a legrövidebb utakat keressük. Ennek a csúcsnak megfelel a mátrix egy oszlopa. Jelöljük ezt a oszlopot a következőképpen: $V^{(1)} = \left(V_i^{(1)}\right)_{i=\overline{1,n}}$. A szomszédsági mátrix: $A = \left(a_{ij}\right)_{i,j=\overline{1,n}}$, ahol $a_{ij} = d_{ij}^{(0)}$, azaz:

$$a_{ij} = \begin{cases} \mathcal{W}(v_i, v_j) & \text{ha } \{v_i, v_j\} \in E(G) \ \ (\text{vagy } (v_i, v_j) \in E(\vec{G})) \\ 0 & \text{ha } i = j \\ \infty & \text{ha } \{v_i, v_j\} \notin E(G) \ \ (\text{vagy } (v_i, v_j) \notin E(\vec{G})) \end{cases}$$

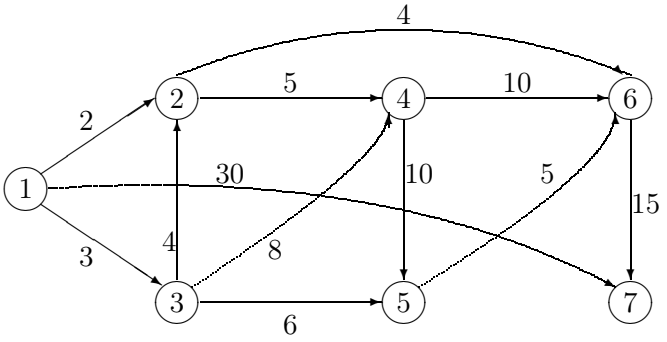
Kiszámítjuk a következő vektorokat ($k = 1, 2, \ldots$):

$$V_i^{(k)} = \min_{j=\overline{1,n}} \left(a_{ij} + V_j^{(k-1)}\right) \text{ for } i = 1, 2, \ldots, n$$

ameddig $V^{(l)} = V^{(l-1)}$ egy bizonyos l értékre.

Példa.

Keressük meg a következő gráfban a 7-es csúcsba befutó legrövidebb utakat minden csócsból. Tehát a $V^{(1)}$ vektor a mátrix 7-es oszlopa.



	1	2	3	4	5	6	7	$V^{(1)}$	$V^{(2)}$	$V^{(3)}$	$V^{(4)}$
1	0	2	3	∞	∞	∞	30	30	30	21	21
2	∞	0	∞	5	∞	4	∞	∞	19	19	19
3	∞	4	0	8	6	∞	∞	∞	∞	23	23
4	∞	∞	∞	0	10	10	∞	∞	25	25	25
5	∞	∞	∞	∞	0	5	∞	∞	20	20	20
6	∞	∞	∞	∞	∞	0	15	15	15	15	15
7	∞	∞	∞	∞	∞	∞	0	0	0	0	0

Egy u–v utat, az előző algoritmusokhoz hasonlóan, egy p vektor segítségével határozhatunk meg. Kezdetben $p(i) := i$ minden $i = 1, 2, \ldots, n$ értékre. Ha a $\min\limits_{j=\overline{1,n}} \left(a_{ij} + V_j^{(k-1)}\right)$ minimumot a $j = \ell$ értékre kapjuk meg, akkor, ha $i <> \ell$ a $p_i := \ell$ lesz.

BellmanKalaba$(A, V^{(1)})$
1. **for** $i = 1, 2, \ldots, n$ **do**
2. $\quad p_i := i$
3. $i := 1$
4. **repeat**
5. $\quad i := i + 1$
6. $\quad$ **for** $k := 1, 2, \ldots n$ **do**
7. $\quad\quad min := a_{k1} + V_1^{(i-1)}$
8. $\quad\quad p_i := 1$
9. $\quad\quad$ **for** $j := 2, 3, \ldots n$ **do**
10. $\quad\quad\quad$ **if** $min > a_{kj} + V_j^{(i-1)}$ **then**
11. $\quad\quad\quad\quad min := a_{kj} + V_j^{(i-1)}$
12. $\quad\quad\quad\quad$ **if** $i \neq j$ **then** $p_i := j$
13. $\quad\quad V_k^{(i)} := min$
14. **until** $V^{(i)} = V^{(i-1)}$
15. **return** $V^{(i)}, p$

Egy v_i–v_j utat (a fenti algoritmusban az első vektor a j oszlop) a következő algoritmussal határozzuk meg:

$k := 1$
$u_k := i$
while $p_k \neq j$ **do**
$\quad u_{k+1} := p_{u_k}$
$\quad k := k + 1$

Előbbi példánkban $p = (2, 6, 2, 6, 6, 7, 7)$. Az 1 és 7 csúcsok között a legrövidebb út a következő csúcsokból áll:

$u_1 := 1,\ u_2 := p_1 = 2,\ u_3 := p_2 = 6,\ u_4 := p_6 = 7$. Vagyis a megfelelő út: 1,2,6,7.

3.2.3. Legrövidebb utak minden csúcsból minden csúcsba

A távolsági mátrix meghatározására egy Warshall-típusú algoritmust használunk. Hogy meghatározhassuk nemcsak a távolságokat, hanem az utakat is, egy P mátrixot használunk az előző csúcsok megőrzésére.

3.2.3.1. A Floyd–Warshall-algoritmus

Kezdetben $p_{ij} := i$ ha $d_{ij} \neq \infty$ és $i \neq j$; más esetekben $p_{ij} := 0$.

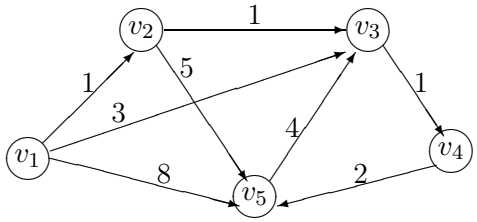
FloydWarshall(D_0)
1. $D := D_0$
2. **for** $k := 1$ **to** n **do**
3. **for** $i := 1$ **to** n **do**
4. **for** $j := 1$ **to** n **do**
5. **if** $d_{ij} > d_{ik} + d_{kj}$ **then**
6. $d_{ij} := d_{ik} + d_{kj}$
7. $p_{ij} := p_{kj}$
8. **return** D, p

Egy u_x–u_y utat a következő algoritmussal határozzuk meg:

1. $k := n$:
2. $u_k := y$
3. **while** $u_k \neq x$ **do**
4. $u_{k-1} := p_{xu_k}$
5. $k := k - 1$

A keresett út: $u_k, u_{k+1}, \ldots, u_n$.

Példa.



Az előbbi gráf szomszédsági mátrixa és a megfelelő P mátrix kezdeti értéke:

$$D_0 = \begin{pmatrix} 0 & 1 & 3 & \infty & 8 \\ \infty & 0 & 1 & \infty & 5 \\ \infty & \infty & 0 & 1 & \infty \\ \infty & \infty & \infty & 0 & 2 \\ \infty & \infty & 4 & \infty & 0 \end{pmatrix} \qquad P_0 = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 2 & 0 & 2 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 4 \\ 0 & 0 & 5 & 0 & 0 \end{pmatrix}$$

Az algoritmus eredménye a D és P mátrixok:

$$D = \begin{pmatrix} 0 & 1 & 2 & 3 & 5 \\ \infty & 0 & 1 & 2 & 4 \\ \infty & \infty & 0 & 1 & 3 \\ \infty & \infty & 6 & 0 & 2 \\ \infty & \infty & 4 & 5 & 0 \end{pmatrix} \qquad P = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ 0 & 0 & 2 & 3 & 4 \\ 0 & 0 & 0 & 3 & 4 \\ 0 & 0 & 5 & 0 & 4 \\ 0 & 0 & 5 & 3 & 0 \end{pmatrix}$$

3.2.4. Az algoritmusok bonyolultsága

Felmerülhet a kérdés, hogy melyik jobb a bemutatott algoritmusok közül? Hogyan határozhatjuk meg a bonyolultságukat?

Logikus lenne, hogy az algoritmus bonyolultságán a feladat megoldásához szükséges időt értsük. Mivel azonban az algoritmus különféle nyelveken és gépeken valósítható meg, a megfelelő program percekben, órákban mért futási ideje nem lehet alapja az összehasonlításnak. Ehelyett az algoritmus jól megválasztott műveleteinek vagy lépéseinek a számát vizsgáljuk. Gyakorlatilag azonban kevés olyan eset van, amikor pontosan meg lehet határozni egy algoritmus lépésszámát adott nagyságú bemenetre. Ezért a lépésszámot vagy az algoritmus szempontjából fontos műveletek számát a *legrosszabb esetben* vizsgáljuk minden azonos nagyságú bemenet esetében. Gráfok esetében azonos csúcsszámú vagy azonos élszámú gráfokat vizsgálunk. A következőkben bonyolultságon mindig ilyen értelemben vett bonyolultságot értünk. A bonyolultság a bemeneti adatok nagyságán értelmezett függvény.

3.2.4.1. Az $O\big(f(n)\big)$ jelölés

Legyenek adottak az $f, g : \mathbf{N} \to \mathbf{R}^+$ függvények. Ha létezik egy C pozitív valós szám és egy n_0 természetes szám úgy, hogy

$$f(n) \leq Cg(n) \quad \text{ha} \quad n \geq n_0,$$

akkor az f függvény *rendje* kisebb vagy egyenlő, mint a g rendje, és ezt így írjuk:

$$f(n) = O\big(g(n)\big).$$

Ha $f(n) = O\big(g(n)\big)$ és $g(n) = O\big(f(n)\big)$, akkor az f és g függvények azonos

rendűek, és ennek jelölése: $f(n) = \Theta\big(g(n)\big)$.

Példák. A közismert mátrixszorzás bonyolultsága, ha mindkettő $n \times n$ típusú, $O(n^3)$. Itt alapműveletnek két elem szorzatát vesszük.

n-elemű lista esetében a szekvenciális keresés bonyolultsága $O(n)$, a buborékos rendezés esetében pedig $O(n^2)$ (mindkét esetben alapműveletként két elem összehasonlítást vesszük).

3.2.4.2. A legrövidebb utak algoritmusainak bonyolultsága

A Floyd–Warshall-algoritmus kivételével a bemutatott algoritmusok vagy egy csúcsból minden csúcsba vagy minden csúcsból egybe keresik a legrövidebb utakat. Ezért, hogy egységesen kezelhessük őket, beleértve a Floyd–Warshall-algoritmust is, átszámítjuk a bonyolultságot a „minden csúcsból minden csúcsba esetre", ami azt jelenti, hogy szorzunk n-nel.

A Moore- és Dijkstra-algoritmusoknál egy-egy **while** és **for** ciklus van egymásba ágyazva. Mivel a gráf n-csúcsú, az elsőt legfennebb n-szer, a másodikat pedig legfennebb $(n-1)$-szer végezzük el. Ezért a bonyolultság $O(n^2)$, mivel $n(n-1)$ másodfokú.

algoritmus	bonyolultság	bonyolultság „minden csúcsból minden csúcsba"
Moore	$O(n^2)$	$O(n^3)$
Dijkstra	$O(n^2)$	$O(n^3)$
Ford	$O(n^3)$	$O(n^4)$
Bellman–Kalaba	$O(n^3)$	$O(n^4)$
Floyd–Warshall	$O(n^3)$	$O(n^3)$

4. fejezet

A kritikus út

A vonatjegy iránt a vonat indulása előtt 30 perccel támasztható igény. A vonatjegy iránt támasztható igény a vonat indulása előtt 5 perccel megszűnik.

(Nyomtatott tájékoztató a püspökladányi vasútállomáson, 1971)

Ha egy bonyolult, több tevékenységből álló feladatot kell megoldanunk úgy, hogy ismerjük az egyes tevékenységek elvégzéséhez szükséges időt, valamint a tevékenységek egymásutániságát, és szeretnénk a feladatot minél hamarabb befejezni, akkor igénybe vehetjük a gráfelmélet eredményeit. A feladathoz rendelt gráf ebben az esetben sajátos, amelyben egy kezdőcsúcs és egy végcsúcs közötti leghosszabb uton is el kell végezni minden tevékenységet, és ezen nem lehet várakozni, ha nem akarjuk negatívan befolyásolni az eredményt.

A feladat gráfmodelljét kétféleképpen lehet felépíteni. Egyik esetben a tevékenységeknek a gráfban irányított élek felelnek meg, a csúcsok pedig eseményeket jelentenek (a befutó tevékenységek végetérte után kezdhetők el a kifutó éleknek megfelelő tevékenységek). A másik modellben a csúcsok felelnek meg a tevékenységeknek, míg az élek csupán a tevékenységek közti kapcsolatokat jelölik.

4.1. Első modell: a tevékenységeket élekkel jelöljük

Tevékenységi gráfnak nevezzük azt a $\vec{G} = (V, E, \mathcal{W})$, $V = \{v_1, v_2, \ldots, v_n\}$, összefüggő, irányított kört nem tartalmazó irányított gráfot, amelyre fennállnak a következők:

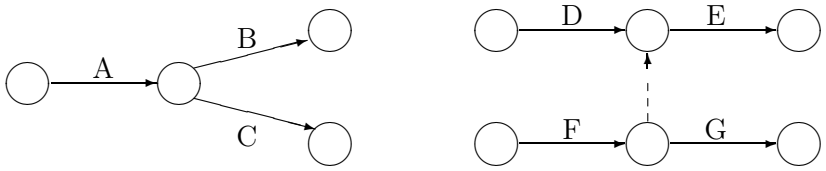
– az élek tevékenységet jelölnek, az élekhez rendelt súlyok a megfelelő tevékenységek végrehajtásához szükséges időtartamot jelölik, és az időtartam

jelölése: $d_{ij} = \mathcal{W}(v_i, v_j)$,

– létezik egy *kezdőcsúcs*, legyen ez v_1, amelyre $N^{\text{be}}(v_1) = \emptyset$ (azaz, egyetlen él se fut bele),

– létezik egy *végcsúcs*, legyen ez v_n, amelyre $N^{\text{ki}}(v_n) = \emptyset$ (azaz, egyetlen él se fut ki belőle).

A tevékenységek közti kapcsolatot a következő ábrával szemléltethetjük:



Példánkban az A tevékenységet be kell fejezni mielőtt megkezdődnének a B és C tevékenységek. Lehetnek 0 időtartamú (azaz fiktív) tevékenységek is, amelyek csupán arra szolgálnak, hogy bizonyos tevékenységi sorrendet megváltossanak (ezeket szaggatott vonallal jelöljük). Az E tevékenységet csak akkor lehet elkezdeni, ha már befejeztük a D és F tevékenységeket (itt van egy fiktív tevékenység), viszont a G megkezdése csak az F bevégeztétől függ.

Mennyi ideig tart az egész feladat elvégzése? Ez megfelel a kezdő- és a végcsúcs közti leghosszabb út hosszának. Mivel a gráf nem tartalmaz irányított köröket, a feladat megoldására alkalmazhatjuk a legrövidebb utak esetében használt algoritmusokat, ha minimum helyett mindenhol maximumot veszünk. De, mivel sajátos gráfról van szó, használhatunk egyszerűbb algoritmusokat is.

Szintekre bontás

A tevékenységi gráf csúcsait szintekre bonthatjuk. A kezdőcsúcs szintje 1. Ha létezik a (v_i, v_j) irányított él, akkor a v_i szintje kisebb, mint a v_j szintje.

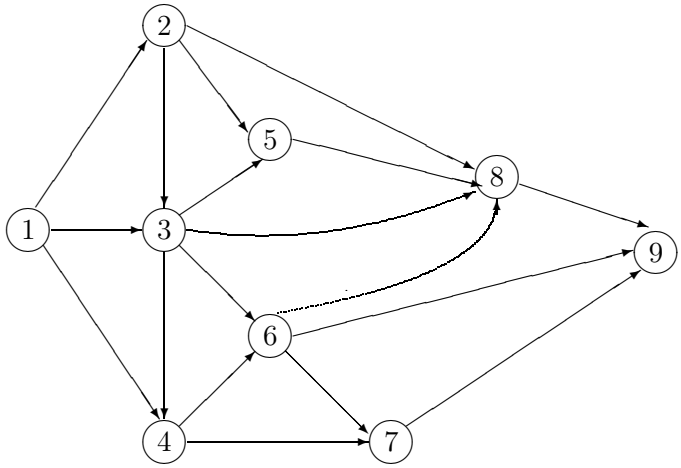
A következő algoritmus megoldja a szintekre bontást, felhasználva a NEXT rekurzív eljárást:

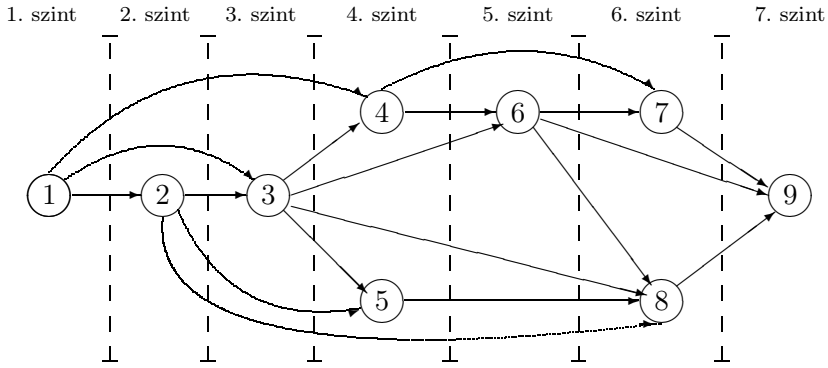
```
SZINTEKREBONTÁS(G)
for i = 1 to n do
    l(i) := 1
for i = 1 to n do
    call NEXT(G, l, i)
return(l)
```

ahol

```
NEXT(G, l, i)
for j = 1 to n do
```

if $(v_i, v_j) \in E$ **and** $\big(l(j) \leq l(i)\big)$ **then**
$l(j) := l(i) + 1$
if $j < i$ **then call** NEXT(G, l, j)
return l

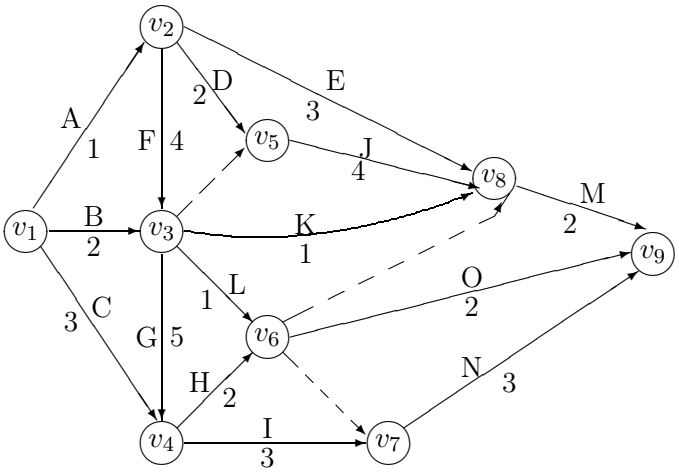




Példaként lássunk egy konkrét feladatot.

tevékenység	előző tevékenységek	időtartam
A	–	1
B	–	2
C	–	3
D	A	2
E	A	3
F	A	4
G	B, F	5
H	C, G	2
I	C, G	3
J	B, F, D	4
K	B, F	1
L	B, F	1
M	E, H, J, K, L	2
N	H, I, L	3
O	H, L	2

A megfelelő tevékenységi gráf a következő:



A tevékenységi gráf előállítása nem mindig egyszerű, hisz eredetileg csak az éleket ismerjük, és meg kell határoznunk a csúcsokat a megfelelő precedenciáknak megfelelően.

Ha a $v_1, v_2, \ldots, v_n$ csúcsok ebben a sorrendben vannak szintekre bontva, akkor a következő algoritmus (a kritikus út módszere, angolul CPM – Critical Path Method) megadja a t_i és t_i^* időpontokat, amelyek a tevékenységi gráf v_i csúcsához (eseményéhez) kapcsolódnak. Ha a feladat elvégzése kezdetének a 0 időpontot tekintjük, akkor t_i azt a *legkorábbi*, t_i^* pedig azt a *legkésőbbi* időpontot jelenti, amikor a v_i-ből kinduló tevékenységeket el lehet kezdeni.

CPMcsúcs(G)
$t_1 := 0$
for $j = 2, 3, \ldots n$ **do**

$$t_j := \max_{v_i \in N^{\text{be}}(v_j)} (t_i + d_{ij})$$

$t_n^* := t_n$
for $i = n-1, n-2 \ldots 1$ **do**

$$t_i^* := \min_{v_j \in N^{\text{ki}}(v_i)} (t_j^* - d_{ij})$$

return t, t^*

Az előbbi példánk esetében:

csúcs	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9
t_i	0	1	5	10	5	12	13	12	16
t_i^*	0	1	5	10	10	13	13	14	16

Értelmezünk néhány ún. *időtartalékot* minden tevékenységre, amelyek értékes információkat tartalmaznak a feladat elvégzésére.

$R_t(v_i, v_j) = t_j^* - t_i - d_{ij}$ a *teljes időtartalék*. A (v_i, v_j) tevékenységet ennyi idővel lehet később kezdeni anélkül, hogy ez befolyásolná az egész feladat elvégzésének időtartamát.

$R_f(v_i, v_j) = t_j - t_i - d_{ij}$ a *szabad időtartalék*. A (v_i, v_j) tevékenységet ennyi idővel lehet később kezdeni anélkül, hogy ez befolyásolná a t_j időpontot.

$R_s(v_i, v_j) = \max(t_j - t_i^* - d_{ij}, 0)$ a *biztos időtartalék*. A (v_i, v_j) tevékenységet ennyi idővel lehet később befejezni anélkül, hogy ez befolyásolná az egész feladat elvégzésének időtartamát.

Azok a tevékenységek, amelyekre mindhárom időtartalék 0, a *kritikus uton* vannak és *kritikus tevékenységeknek* nevezzük őket. Ezen tevékenységek esetében bármilyen késlekedés az egész időtartam róvására megy.

Előbbi példánk esetében:

tevékenység	időtartam	teljes időtartalék: R_t	szabad időtartalék: R_f	biztos időtartalék: R_s
A	1	0	0	0
B	2	3	3	3
C	3	7	7	7
D	2	7	2	2
E	3	10	8	8
F	4	0	0	0
G	5	0	0	0
H	2	1	0	0
I	3	0	0	0
J	4	5	3	0
K	1	8	6	6
L	1	7	6	6
M	2	2	2	0
N	3	0	0	0
O	2	2	2	1

Módosíthatjuk a Floyd–Warshall-algoritmust, hogy leghosszabb távolságokat számoljon, majd ezt felhasználjuk a t és t^* időpontok kiszémítására. Újraértelmezzük a D_0 mátrixot:

$$d_{ij}^{(0)} = \begin{cases} \mathcal{W}(v_i, v_j) & \text{ha } (v_i, v_j) \in E \\ 0 & \text{ha } i = j \\ -\infty & \text{ha } (v_i, v_j) \notin E \end{cases}$$

Az algoritmus a következő:

FWMax(D_0)
$D := D_0$
for $k := 1$ **to** n **do**
  for $i := 1$ **to** n **do**
    for $j := 1$ **to** n **do**
      if $d_{ij} < d_{ik} + d_{kj}$ **then** $d_{ij} := d_{ik} + d_{kj}$
return(D)

A legkorábbi és legkésőbbi időpontok kioszámítására a következő algoritmust használjuk.

for $i = 1, 2, \ldots, n$ **do**
  $t_i := d_{1i}$
for $i = 1, 2, \ldots, n$ **do**
  $t_i^* := d_{1n} - d_{in}$

Példánk esetében:

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9
v_1	0	1	5	10	5	12	13	12	16
v_2	$-\infty$	0	4	9	4	11	12	11	15
v_3	$-\infty$	$-\infty$	0	5	0	7	8	7	11
v_4	$-\infty$	$-\infty$	$-\infty$	0	$-\infty$	2	3	2	6
v_5	$-\infty$	$-\infty$	$-\infty$	$-\infty$	0	$-\infty$	$-\infty$	4	6
v_6	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	0	0	0	3
v_7	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	0	$-\infty$	3
v_8	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	0	2
v_9	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	0

A t_i és t_i^* időpontok tehát a következők:

vertex	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9
t_i	0	1	5	10	5	12	13	12	16
t_i^*	0	1	5	10	10	13	13	14	16

Ha a tevékenységek időtartama nem állapítható meg pontosan, ahogy ez a gyakorlatban többnyire történni szokott, akkor a legkisebb (optimista), illetve a legnagyobb (pesszimista) becsült időtartamot vesszük, azaz időtartam helyett egy időintervallumot. Ebben az esetben a feladat megoldásához szükséges idő meghatározására a PERT módszert használjuk (PERT = *Programme Evaluation and Review Technique* vagy *Programme Evolution Research Task*). Az időtartam ebben az esetben a megfelelő (optimista és pesszimista időtartamok által meghatározott) intervallumba eső véletlenszerű érték. A kapott kritikus út és a kritikus tevékenységek bizonyos valószínűséggel értendők.

4.2. Második modell: a tevékenységeket csúcsokkal jelöljük

A tevékenységi gráfot a következőképpen módosítjuk. A csúcsok tevékenységeknek felelnek meg, az élek pedig ezek egymásutániságát jelölik. Ezért ennek a tevékenységi gráfnak az előállítása sokkal egyszerűbb, nem más rutinmunkánál. Tekintsük itt is, hogy a csúcsok szintekre vannak bontva, méghozzá a $v_1, v_2, \ldots, v_n$ sorrendben. Ebben az esetben a kezdő- és végcsúcs egy-egy fiktív (nulla időtartamú) tevékenységnek felel meg. Minden csúcs esetében értelmezzük a következőket:

$t_m(v_i)$	v_i	$t^*_m(v_i)$
$t_M(v_i)$	d_i	$t^*_M(v_i)$

d_i – a v_i tevékenység időtartama,
$t_m(v_i)$ – legkorábbi időpont, amikor a v_i tevékenység megkezdődhet,
$t^*_m(v_i)$ – legkorábbi időpont, amikor a v_i tevékenység befejeződhet,
$t_M(v_i)$ – legkésőbbi időpont, amikor a v_i tevékenység megkezdődhet,
$t^*_M(v_i)$ – legkésőbbi időpont, amikor a v_i tevékenység befejeződhet.

A következő algoritmussal kiszámíthatjuk ezen időpontokat.

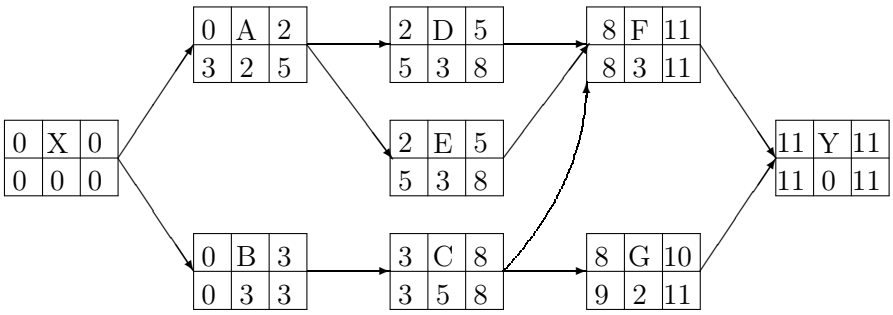
CPMÉL(D)
$t_m(v_1) := 0$
$t^*_m(v_1) := d_1$
for $j := 2, 3 \ldots, n$ **do**
  $t_m(v_j) := \max_{v_i \in N^{\text{be}}(v_j)} t^*_m(v_i)$
  $t^*_m(v_j) := t_m(v_j) + d_j$
$t^*_M(v_n) := t^*_m(v_n)$
$t_M(v_n) := t^*_M(v_n) - d_n$
for $i := n-1, n-2, \ldots, 1$ **do**
  $t^*_M(v_i) := \min_{v_j \in N^{\text{ki}}(v_i)} t_M(v_j)$
  $t_M(v_i) := t^*_M(v_i) - d_i$
return $t_m, t^*_m, t_M, t^*_M,$

Egy v tevékenység kritikus, ha $t_m(v) = t_M(v)$ (és természetesen $t^*_m(v) = t^*_M(v)$ is). Egy olyan utat, amelyen minden tevékenység kritikus, kritikus útnak nevezünk.

Oldjuk meg a következő feladatot.

tevékenység	előző teveékenységek	időtartam
A	–	2
B	–	3
C	B	5
D	A	3
E	A	3
F	C, D, E	3
G	C	2

A megoldást a következő ábra mutatja.



A feladat megoldásának teljes időtartama 11 időegység. A kritikus út X, B, C, F, Y (ahol X és Y fiktív tevékenységek).

5. fejezet

Euler- és Hamilton-gráfok

Euleri gráf: minden foka páros,
és a tétel mindörökre áll most;
gráfokról ez állítás
a világnak ősforrás.

(B. Zelinka: A gráfelmélet himnusza,
ford. Ádám András)

Utunkban, te nemes lovag, segíts meg.
Hajrá, fogyjon az út, társak, siessünk!

(Janus Pannonius: Búcsú Váradtól,
ford. Áprily Lajos)

5.1. Euler-gráfok

Egy vonal *Euler-vonal*, ha a gráf minden élét tartalmazza. Ha egy ilyen vonal zárt, akkor *zárt Euler-vonalról* beszélünk. Egy gráfot *Euler-gráfnak* nevezünk, ha van benne zárt Euler-vonal. Emlékeztetünk, hogy egy gráf tartalmazhat párhuzamos éleket és hurkoksat is.

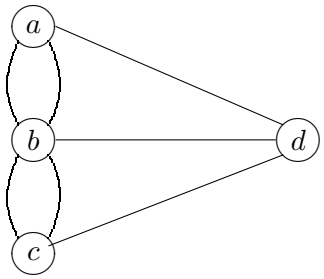
1. tétel. *Egy izolált csúcsokat nem tartalmazó gráf akkor és csakis akkor Euler-gráf, ha összefüggő és minden csúcsának fokszáma páros.*

Bizonyítás. a) Ha a gráf Euler-gráf, akkor van benne zárt Euler-vonal, tehát összefüggő. A zárt vonal minden csúcs érintésekor pontosan két élt használ, tehát a minden fokszám páros.

b) Tekintsünk egy összefüggő gráfot, amelyben minden csúcs fokszáma páros. Az élek számára vonatkozó indukcióval bizonyítjuk, hogy ekkor létezik zárt Euler-vonal. Ha az n-csúcsú gráf egyetlen n hosszúságú kört tartalmaz, akkor a tétel állítása igaz. Legyen a gráfnak ennél több éle. Mivel a gráf összefüggő és minden csúcsa páros fokszámú, könnyen találhatunk benne egy zárt

vonalat. Nem kell mást tennünk csak bejárnunk az éleket, vigyázva arra, hogy minden csúcsban olyan élen haladjunk tovább, amelyet még nem érintettünk. Mivel minden fokszám púros, ssak akkor nem tudjuk folytatni az eljárást, ha már visszaértünk a kiinduló csúcsba. Ha ez a zárt vonal tartalmazza a gráf minden élét, akkor tételünket bebizonyítottuk. Ha nem, akkor elhagyjuk a kapott zárt vonal éleit a gráfból. Az így kapott gráfban az élek száma kisebb lesz, minden fokszám szintén páros, de a gráf esetleg nem összefüggő. Ekkor minden komponensének kevesebb éle van, mint az eredeti gráfnak, tehát, az indukciós feltevés alapján minden komponensére igaz a tétel, azaz minden komponensében van zárt Euler-vonal. Legyenek ezek $\mathcal{E}_1, \mathcal{E}_2, \ldots \mathcal{E}_k$, ha k komponens van. A gráfból törölt vonal sorra érinti ezeket a zárt vonalakat, és velük együtt az eredeti gráf egy zárt Euler-vonalát képezi. És ezzel bebizonyítottuk a tételt. □

A tétel értelmében a königsbergi hidak problémájának megfelelő gráf nem Euler-gráf, mivel minden csúcsának fokszáma páratlan. ($\varphi(a) = \varphi(c) = \varphi(d) = 3$ és $\varphi(b) = 5$).



2. tétel. *Egy összefüggő gráf akkor és csakis akkor tartalmaz (nyitott) Euler-vonalat, ha két, páratlan fokszámú csúcsán kivül minden más csúcs fokszáma páros.*

Bizonyítás. Egy új él hozzáadásával a két páratlan fokszámú csúcs között minden fokszám páros lesz, tehát lesz a gráfban zárt Euler-vonal (1). tétel alapján). Elhagyja belőle a hozzáadott élt, egy (nyitott) Euler-vonalat kapunk. □

3. tétel. *Ha egy összefüggő G gráfban a páratlan fokszámű csúcsok száma $2k (k \geq 1)$, akkor létezik a G gráfban k darab élfüggetlen vonal, amelyek együtt lefedik a gráf éleit.*

Bizonyítás. Két vonal élfüggetlen, ha nincs közös élük. A $2k$ páratlan fokszámú csúcsot össze lehet kötni k olyan éllel, amelyek nem szomszédosak. Ekkor nminden fokszám páros, és az 1. tétel alapján a bizonyítás nyilvánvaló, mivel a zárt Euler-vonalból elhagyja a nem szomszédos k élt, a zárt Euler-vonal k élfüggetlen vonallá esik szét. □

Fleury[1] algoritmusa

Az algoritmus segítségével megkereshetünk egy Euler-vonalat (páratlan fokú csúcsból indulunk), vagy akár zárt Euler-vonalat is (tetszőleges csúcsból indulunk).

Egy élt *hídnak* nevezünk, ha elhagyásával eggyel nő a gráf összefüggő komponenseinek a száma.

A Fleury-algoritmus esetében elindulunk egy csúcsból, kiválasztunk egy élt, majd töröljük azt, a következő csúcsban hasonlóan váálasztunk egy élt, vigyázva arra, hogy hidat csak akkor válasszunk, ha más lehetőségünk nincs. Minden egyszer bejárt élt törlünk.

FLEURY(G, u)
1. válasszunk ki egy u-hoz illeszkedő e élt, amelynek másik végpontja v
2. $i := 1$
3. $w_i := e$
4. töröljük ki a gráfból az e élt
5. $u := v$
6. **while** van még él a gráfban **do**
7. válasszunk ki egy u-hoz illeszkedő e élt, amelynek másik végpontja v, és amelyik csak akkor lehet híd, ha nincs más lehetőség
8. $i := i + 1$
9. $w_i := e$
10. töröljük ki a gráfból az e élt
11. $u := v$
12. **return** $w_k, k = 1, 2, \ldots, i$

Példa.

Az 1. tétel alapján könnyen tervezhetünk egy más algoritmust is.

Irányított gráfok esetében a következő eredményeket fogalmazhatjuk meg.

4. tétel. *Egy összefüggő irányított gráfban akkor és csakis akkor létezik irányított zárt Euler-vonal, ha minden v csúcsra igaz, hogy: $\varphi^{be}(v) = \varphi^{ki}(v)$.*

5. tétel. *Egy összefüggő irányított gráfban akkor és csakis akkor van (nyitott) irányított Euler-vonal, ha van két olyan u és v csúcsa, amelyekre $\varphi^{ki}(u) = \varphi^{be}(u) + 1$, $\varphi^{ki}(v) = \varphi^{be}(v) - 1$, és a gráf minden más x csúcsára $\varphi^{be}(x) = \varphi^{ki}(x)$. A vonal u-val kezdődik és v-vel végződik.*

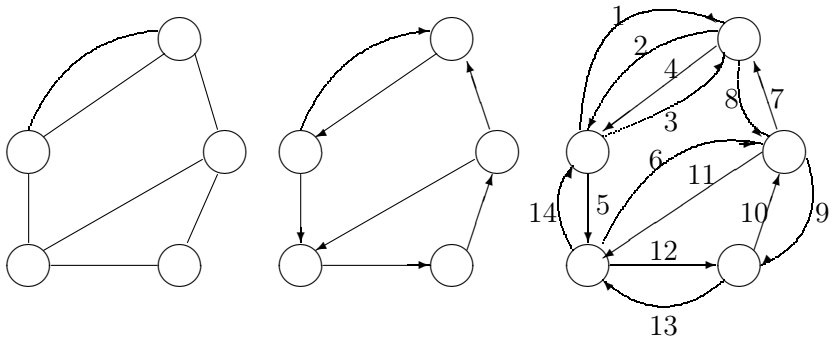
Egy érdekes eredmény nem irányított gráfokra:

6. tétel. *Egy összefüggő gráf éleit be lehet járni úgy, hogy minden élt pontosan kétszer érintsünk, méghozzá úgy, hogy mindkét irányba egyszer, és visszaérjünk a kiinduló csúcsba.*

[1]Henri Fleury, ma már alig ismert francia matematikus, aki 1883-ban közölte algoritmusát.

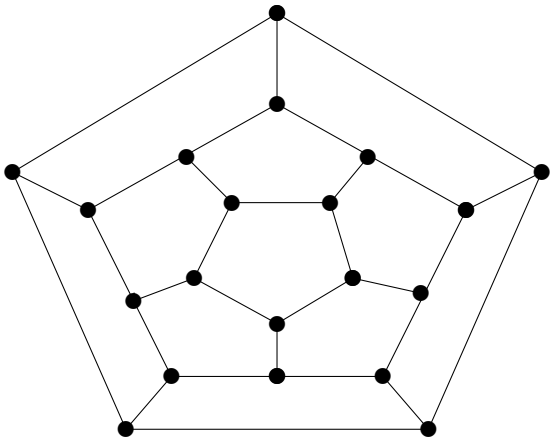
Bizonyítás. Tetszőlegesen irányítjuk a gráf éleit, majd minden (u, v) él esetében hozzáadunk egy új (v, u) élt. Az eredményül kapott irányított gráf kielégíti a 4. tétel követelményeit, tehát létezik benne zárt irányított Euler-vonal. Ez pedig megfelel a kért bejárásnak az eredeti gráfban. □

Példa. Az irányított gráfban sorszámmal jelöltünk egy lehetséges bejárást.



5.2. Hamilton-gráfok

1857-ben Sire William R. Hamilton ír matematikus kitalálta a dodekaéderjátékot (a dodekaéder 20 csúcsú, 30 élű, 12 lapú szabályos test, amelynek minden lapja szabályos ötszög). Minden csúcs egy várost jelentett, és a feladat szerint be kellett járni minden várost, de csak egyszer érintve mindegyiket, és visszajutni a kiindulópontba. A játéknak megfelelő gráf, amelyet úgy kaphatunk meg, hogy a dodakaéder egyik lapját széthúzzuk, majd erre vetítjük a csúcsokat és éleket, a következő.

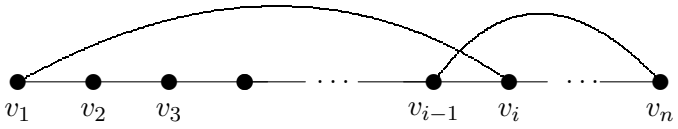


Egy utat *Hamilton-útnak* nevezünk, ha tartalmazza a gráf összes csúcsát.

Egy kör, amelyik tartalmazza a gráf minden csúcsát, *Hamilton-kör*. Egy gráf *Hamilton-gráf*, ha tartalmaz Hamilton-kört. A Hamilton-gráfok vizsgálata sokkal bonyolultabb, mint az Euler-gráfoké.

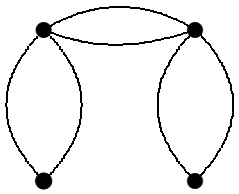
7. tétel. [Ore] *Ha egy legalább $n \geq 3$ csúcsú egyszerű gráfban bármelyik kér nem szomszédos u, v csúcsra igaz, hogy $\varphi(u) + \varphi(v) \geq n$, akkor a gráfban van Hamilton-kör.*

Bizonyítás. A bizonyítást reductio ad absurdummal végezzük. Az összes olyan n-csúcsú egyszerű gráfok közül, amelyek teljesítik a tétel feltételeit, és mégsem tartalmaznak Hamilton-kört, tekintsünk egy maximális élszámút. Ezért hozzáadva egy $\{u, v\}$ élt ez a gráf tartalmazni fog egy Hamilton-kört. Így az eredeti gráfban van egy Hamilton-út, legyen ez $u = v_1, v_2, \ldots, v_n = v$.



Ezen a Hamilton-uton kell lennie két szomszédos csúcsnak, v_{i-1} és v_i úgy, hogy v_1 szomszédos legyen v_i-vel, v_n pedig v_{i-1}-vel, különben $\varphi(v_1) \leq n - 1 - \varphi(v_n)$, ami ellentmondásban lenne a tétel feltételével. De ekkor létezik egy $v_1, v_2, \ldots, v_{i-2}, v_{i-1}, v_n, v_{n-1}, \ldots v_{i+1}, v_i, v_1$ Hamilton-kör is, ami ellentmondás. □

A következő gráf példa arra, hogy a tétel tetszőleges gráfra nem igaz. A nem szomszédos csúcsok fokszámának összege legalább 4, és a gráf mégsem tartalmaz Hamilton-kört.



8. tétel. [Dirac Gábor] *Ha egy legfeljebb $2k$-csúcsú egyszerű gráfban minden csúcs fokszáma legalább k ($k > 1$), akkor a gráfban van Hamilton-kör.*

1. bizonyítás. Alkalmazzuk Ore tételét!

2. bizonyítás. A bizonyításhoz szükségünk lesz a következő két lemmára.

1. lemma. *Ha egy legfeljebb $2k$-csúcsú egyszerű gráfban minden csúcs fokszáma legalább $k \geq 1$, akkor a gráf összefüggő.*

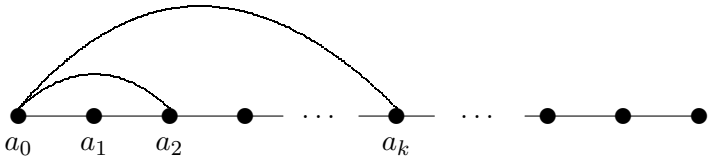
Bizonyítás. Legyen egy olyan legfeljebb $2k$-csúcsú egyszerű gráf, amelyben minden csúcs fokszáma legalább k, de mégsem összefüggő. Ekkor a gráf legalább két komponensből áll, és ezek közül legalább egyiknek legfeljebb k

csúcsa van. Ebben a komponensben a legnagyobb fokszám $k-1$ lehet, ami ellentmondás a tétel feltevésével, tehát a gráf csak összefüggő lehet. □

Adjunk példát arra, hogy tetszőleges gráfra a lemma nem igaz.

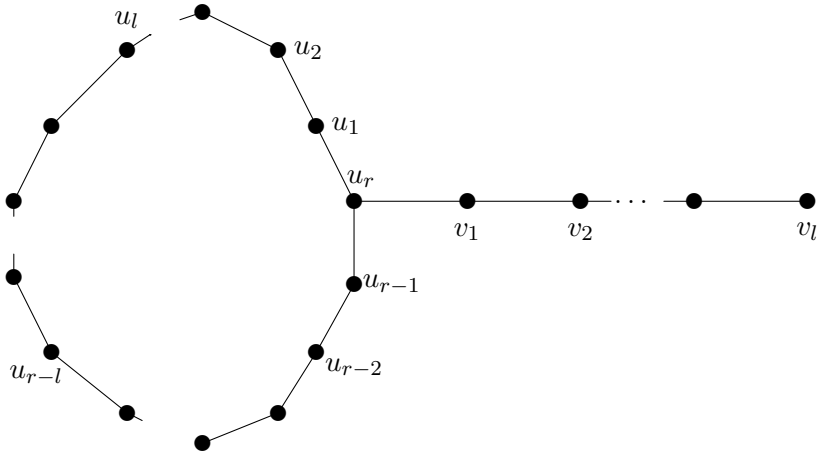
2. lemma. *Ha egy egyszerű gráfban minden csúcs fokszáma legalább $k > 1$, akkor a gráfban van egy legalább $k+1$ hosszúságú kör.*

Bizonyítás. Legyen $a_0, a_1, a_2 \ldots$ egy leghosszbbb út a gráfban.



Mivel minden v csúcsra igaz, hogy $\varphi(v) \geq k$, a_0-nak is a fokszáma legalább k. De az $a_0, a_1, a_2, \ldots$ egy leghosszabb út, ezért a_0 csak az uton levő csúcsokkal lehet szomszédos. a_0 fokszáma legalább k, ezért a legrosszabb esetben is szomszédos az $a_1, a_2, \ldots a_k$ csúcsokkal. Ekkor $a_1, a_2, \ldots a_k, a_1$ egy $k+1$ hosszúságú kör. □

A 8. tétel bizonyítása Az 1. alapján a gráf összefüggő. A 2. lemma alapján a gráfban van egy $r \geq k+1$ hosszúságú kör.



Ha ez a kör nem Hamilton-kör ($r < 2k$), akkor egyik csúcsából (legyen ez u_r), létezik egy leghosszabb út, $u_r, v_1, \ldots, v_l$, amelynek csúcsai nincsenek a körön. Mivel ez egy leghosszbb út, és $\varphi(v_l) \geq k$, a v_l csúcs csak ezen az uton vagy a körön lévő csúcsokkal szomszédos, összesen legalább k-val. De v_l nem lehet szomszédos sem az $u_1, u_2 \ldots, u_l$ csúcsokkal sem pedig az $u_{r-l}, u_{r-l+1}, \ldots u_r$ csúcsokkal (mert akkor egy hosszabb kör keletkezne). A v_l csúcs ugyanakkor nem lehet szomszédos az $u_{l+1}, u_{l+2}, \ldots, u_{r-l-1}$ csúcsok közül két szomszédossal, mert akkor szintén hosszabb kör keletkezne, ha a két ilyen csúcs közötti élt

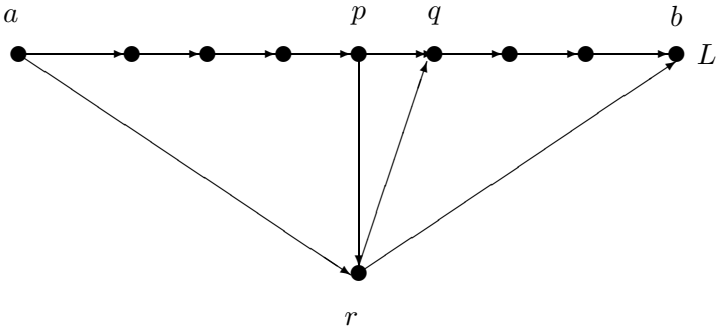
a két éllel helyettesítenénk. Tehát a v_l csúcsnak a körön legfeljebb $\frac{r-2l}{2}$ szomszédja lehet. De, másfelől, v_l-nek legalább $k-l$ szomszédja kell, hogy legyen körön (mert az uton legfeljebb l lehet). Így $\frac{r-2l}{2} \geq k-l$, azaz $r \geq 2k$, ami nyilvánvalóan ellentmondás. □

9. tétel. *Legyen G egy $n \geq 2$ csúcsú egyszerű gráf. Ha $\varphi(v) \geq \frac{n-1}{2}$ bármelyik v csúcsra, akkor G tartalmaz Hamilton-utat.*

Bizonyítás. Legyen G' az a gráf, amelyet úgy kapunk G-ből, hogy hozzáadunk egy új, x csúcsot, amelyet összekötünk a eredeti gráf minden csúcsával. A G' gráfban $\varphi(x) = n$ és $\varphi(v) \geq \frac{n-1}{2} + 1 = \frac{n+1}{2}$ mindne v-re G-ből, de G' csúcsainak a száma $n+1$, így a 8. tétel alapján G'-ben Hamilton-kör. Ha töröljük az x csúcsot a körből, egy G-beli Hamilton-utat kapunk. □

10. tétel. [Rédei] *Egy legalább kétcsúcsú teljes gráf éleinek tetszőleges irányításával kapott gráfban mindig van irányított Hamilton-út.*

Bizonyítás. Tekintsünk a teljes gráfban az élek irányítása után egy leghosszabb L irányított utat:



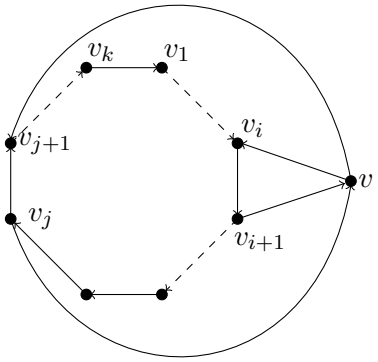
Ha ez nem Hamilton-út, akkor létezik egy r csúcs a gráfban, amelyik nincs rajta ezen az uton. Ekkor létezik az (a, r) irányított él (különben, ha az irányítás (r, a), akkor L nem a leghosszabb irányított út), hasonlóképen létezik az (r, b) él is. Ekkor azonban léteznie kell a (p, r) és (r, q) irámyított éleknek, de ekkor az $a, \ldots p, r, q, \ldots, b$ irányított út hosszabb mint L, ami ellentmondás. Tehát L Hamilton-út. □

11. tétel. *Ha egy legalább háromcsúcsú teljes gráf éleit úgy irányítjuk, hogy minden csúcsához illeszkedjék kifutó és befutó él is, akkor az így kapott gráfban van irányított Hamilton-kör.*

Bizonyítás. A Rédei-tétel alapján (10. tétel) a gráfban van irányított Hamilton-út. Legyen ez $v_1, v_2 \ldots, v_n$. Ha a v_1 és v_n csúcsok között az irányítás v_n-ből v_1-be van, akkor van a gráfban irányított Hamilton-kör. Ha pedig

fordítva, v_1-ből v_n-be van irányítva az él, akkor mivel v_1-hez is kell illeszkednie legalább egy befutó élnek, létezik egy (v_k, v_1) irányított él. Ezért van a gráfban irányított kör: $v_1, v_2 \ldots, v_k, v_1$.

Tekintsünk egy leghosszabb irányított kört, legyen ez $v_1, v_2 \ldots, v_k, v_1$.



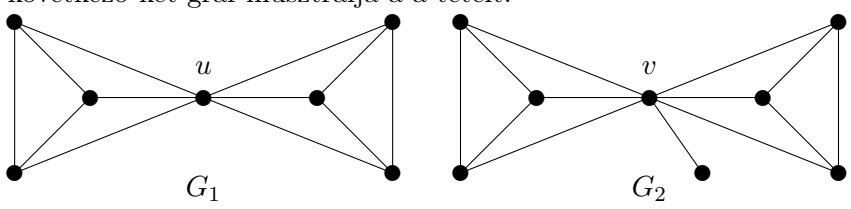
Ha ez a kör nem tartalmazza a gráf minden csúcsát, akkor létezik egy körön kívüli v csúcs. Ez össze van kötve a kör minden csúcsával valamilyen irányítású éllel. Ha létezik két (v_i, v) és (v, v_{i+1}) él, akkor ellentmondáshoz jutunk, hisz keletkezik egy hosszabb irányított kör: $v_1, v_2 \ldots, v_i, v, v_{i+1} \ldots v_n, v_1$. Tehát ilyen eset nem lehet. De, mivel az sem lehet, hogy minden v-hez illeszkedő él befutó vagy kifutó legyen, kell lennie két (v, v_i) és (v_{i+1}, v) élnek is, de ekkor feltétlenül léteznie kell a (v, v_{j+1}) és (v_j, v) éleknek is, ami újból ellentmondáshoz vezet, hiszen ekkor az előző esethez hasonlóan (v_j, v_{j+1}) helyett a (v_j, v) és (v, v_{j+1}) éleket véve, hosszabb kört kapunk. Ezzel a tételt bebizonyítottuk. □

A tétel szerint a gráf egy adott köre mindig bővíthető egy csúccsal, tehát lényegében kapunk egy algoritmust is a Hamilton-kör megkeresésére.

A következő tétel arra ad elégséges feltételt, hogy egy gráf ne tartalmazzon Hamilton-utat vagy -kört.

12. tétel. *Ha egy G gráf k csúcs kitörlése után több mint k komponenre bomlik, akkor G nem tartlmaz Hamilton-kört. Ha k csúcs törlésével G több mint $k+1$ komponensre bonmlik, akkor G még Hamilton-utat sem tartalmaz.*

A következő két gráf illusztrálja a a tételt.



Hamilton-út és -kör keresése

A latin négyzet segítségével irányított Hamilton-utakat és -köröket kereshetünk. A latin négyzet hasonlít a szomszédsági mátrixhoz, de itt a mátrix elemei maguk az élek, pontosan azok végpontjai.

Legyen $\vec{G} = (V, E)$ egy egyszerű irányított gráf. Értelmezzük az n-csúcsú $\vec{G}$ gráf latin négyzetét a következőképpen.

$L = (l_{ij})_{i,j=\overline{1,n}}$, ahol

$$l_{ij} = \begin{cases} v_i v_j & \text{ha} (v_i, v_j) \in E \\ 0 & \text{ha} (v_i, v_j) \notin E \end{cases}$$

Értelmezzük a következő mátrixot is.

$L^* = (l^*_{ij})_{i,j=\overline{1,n}}$, ahol

$$l^*_{ij} = \begin{cases} v_j & \text{ha} (v_i, v_j) \in E \\ 0 & \text{ha} (v_i, v_j) \notin E \end{cases}$$

A következő szorzás segítségével kereshetjük meg a Hamilton-utakat és köröket.

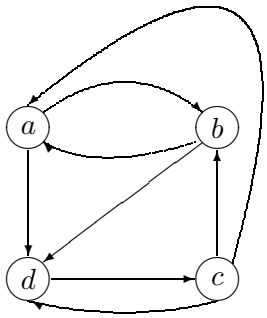
$L^{(2)} = L \otimes L^*$

$L^{(k)} = L^{(k-1)} \otimes L^*$, ha $k \geq 2$,

ahol $l^{(k)}_{ij} = \{ l^{(k-1)}_{i1} \cdot l^*_{1j}, l^{(k-1)}_{i2} \cdot l^*_{2j}, \ldots, l^{(k-1)}_{in} \cdot l^*_{nj} \}$.

Itt $l^{(k-1)}_{i1} \cdot l^*_{1j}$ a két elem, $l^{(k-1)}_{i1}$ és l^*_{1j}, konkatenálását (azaz egymásmellé helyezését) jelenti, és ez megfelel egy sétának a gráfban. Amennyiben az $l^{(k-1)}_{i1}$ elem halmaz, akkor a konkatenálás mindegyik elemére vonatkozik. Ha konkatenálásnál egyik elem 0, akkor az eredmény is az. Mivel Hamilton-utakat keresünk, azok az elemek nem érdekelnek, amelyekben ismétlődnek csúcsok, hisz akkor a séta nem út. Ezért ezekben az esetekben az illető elemet 0-val helyettesítjük. Így az $L^{(n-1)}$ elemei irányított Hamilton utaknak felelnek meg. Amennyiben az irányított Hamilton-köröket szeretnénk megkeresni úgy megengedjük, hogy szorzáskor az n-dik hatványnál az elemek első és utolsó betűje megegyezzék.

Példa. Példánkban az mátrix sorait és oszlopait az a, b, c, d csúcsokkal indexeljük (az első sor a-nak, a második b-nek, a harmadik c-nek, a negyedik pedig d-nek felel meg). Az egyes helyeken halmazok helyett egymás alá írt elemeket használunk.



A megfelelő latin négyzetek (mátrixok) és azok hatványai a következők:

$$L = \begin{pmatrix} 0 & ab & 0 & ad \\ ba & 0 & 0 & bd \\ ca & cb & 0 & cd \\ 0 & 0 & dc & 0 \end{pmatrix} \qquad L^* = \begin{pmatrix} 0 & b & 0 & d \\ a & 0 & 0 & d \\ a & b & 0 & d \\ 0 & 0 & c & 0 \end{pmatrix}$$

$$L^{(2)} = \begin{pmatrix} 0 & 0 & adc & abd \\ 0 & 0 & bdc & bad \\ cba & cab & 0 & \left\{ \begin{matrix} cad \\ cbd \end{matrix} \right\} \\ dca & dcb & 0 & 0 \end{pmatrix}$$

$$L^{(3)} = \begin{pmatrix} 0 & adcb & abdc & 0 \\ bdca & 0 & badc & 0 \\ 0 & 0 & 0 & \left\{ \begin{matrix} cbad \\ cabd \end{matrix} \right\} \\ dcba & dcab & 0 & 0 \end{pmatrix}$$

$$L^{(4)} = \begin{pmatrix} \left\{ \begin{matrix} adcba \\ abdca \end{matrix} \right\} & 0 & 0 & 0 \\ 0 & \left\{ \begin{matrix} bdcab \\ badcb \end{matrix} \right\} & 0 & 0 \\ 0 & 0 & \left\{ \begin{matrix} cbadc \\ cabdc \end{matrix} \right\} & 0 \\ 0 & 0 & 0 & \left\{ \begin{matrix} dcbad \\ dcabd \end{matrix} \right\} \end{pmatrix}$$

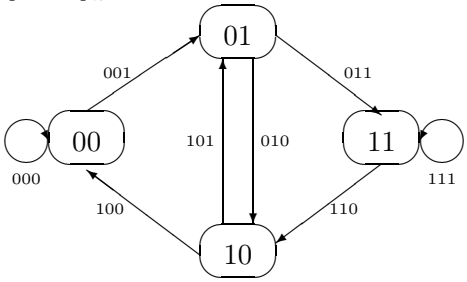
Az L^3 elemei szerint a gráfban 8 Hamilton-út van, L^4 szerint pedig két Hamilton-kör (a mátixban ezek mindegyike négyszer jelenik meg, hisz egy kör

bármelyik csúccsal kezdődhet).

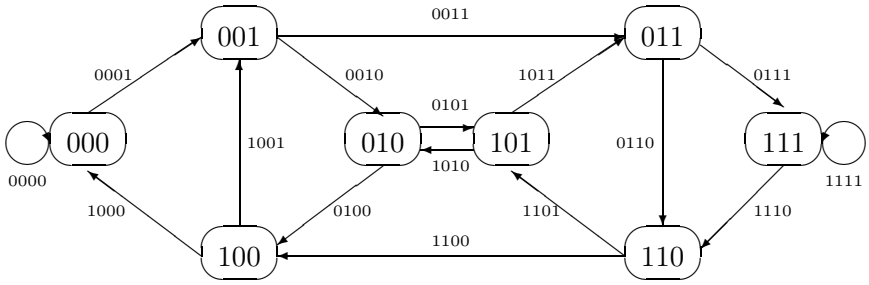
5.3. De Bruijn-gráfok

Legyen $\mathcal{A}$ egy n-betűs ábécé, $\mathcal{A}^k$ pedig az $\mathcal{A}$ fölötti összes k hosszúságú szavak halmaza.

De Bruijn-gráfnak nevezzük azt a $B(n,k) = (V(n,k), E(n,k))$ gráfot, ahol $V(n,k) = \mathcal{A}^k$ a csúcsok halmaza, $E(n,k) = \mathcal{A}^{k+1}$ az élek halmaza, és létezik irányított él egy $x_1x_2\ldots x_k$ csúcsból egy $y_1y_2\ldots y_k$ csúcsba, ha $x_2x_3\ldots x_k = y_1y_2\cdots y_{k-1}$.



A $B(2,2)$ De Bruijn-gráf.



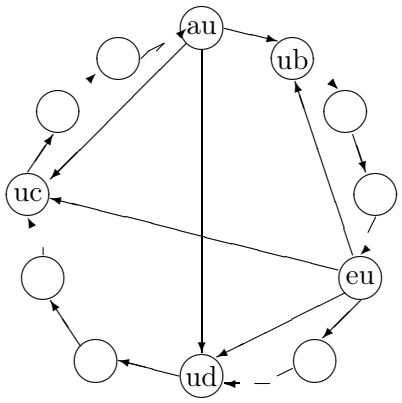
A $B(3,2)$ De Bruijn-gráf.

A $B(n,k)$ De Bruijn-gráfban létezik zárt irányított Euler-vonal, hisz minden csúcsába a befutó élek száma azonos a kifutó élek számával, és ez n. Ugyanakkor létezik irányított Hamilton-kör is, hisz egy zárt irányított Euler-vonalnak a $B(n,k)$ gráfban megfelel egy irányított Hamilton-kör a $B(n,k+1)$ gráfban. Például a 000, 001, 011, 111, 110, 101, 010, 100 élsorozat zárt irányított Euler-vonal $B(2,2)$-ben, és egyben Hamilton-út $B(2,3)$-ban, amely folytatható a 000 csúccsal, hogy Hamilton-kör legyen.

13. tétel. *Ha a $B(n,k)$ gráfból, ahol $n > 2$, elhagyjuk egy irányított Hamilton-kör éleit, a kapott gráf összefüggő marad.*

Bizonyítás. Legyenek $a, b, c, d, e \in \mathcal{A}$, $u \in \mathcal{A}^{k-1}$ és két szomszédos au és ub csúcs az adott Hamilton-körön. A De Bruijn-gráfok értelmezése szerint léteznek az uc, ud és eu csúcsok úgy, hogy léteznek az (au, uc), (au, ud), (eu, ub) élek, amelyek nincsenek az adott Hamilton-körön, és az (eu, uc) és (eu, ud) élek

közül legfeljebb egyik lehet az adott Hamilton-körön. Tehát, az *au* és *ub* csúcsok között van lánc a Hamilton-kör éleinek törlése után. Ez a Hamilton-kör bármelyik két csúcsára igaz, tehát a gráaf a Hamilton-kör éleinek törlés után is összefüggő marad.



□

Ha $n > 2$, akkor a $B(n, k)$ gráfban egy Hamilton-út mindig folytatható úgy, hogy zárt Euler-vonal keletkezzék. Ha $n = 2$, akkor ez az állítás nem igaz, amint az könnyen belátható.

6. fejezet

Fák és ligetek

Azért van a bináris fa gyökere felül, és a levelei alul, mert akik kitalálták, soha nem jártak a természetben, és nem láttak igazi fát.

(falfirka)

6.1. Alaptulajdonságok

Legyen $G = (V, E, \mathcal{G})$ egy tetszőleges gráf, ahol $m = |E|$, $n = |V|$, és $k = \kappa(G)$ az összefüggő komponenseinek a száma. A gráf *ciklomatikus száma* $\nu(G) = m - n + k$.

14. tétel. *Legyen G egy gráf, G' pedig egy olyan gráf, amelyet G-ből kapunk egy él hozzáadásával.*

a) Ha az új él hurok vagy ugyanannak a komponensnek két csúcsát köti össze, akkor $\nu(G') = \nu(G) + 1$.

b) Ha az új él két különböző komponens két csúcsát köti össze, akkor $\nu(G') = \nu(G)$.

Bizonyítás. Az a) esetben egy él hozzáadásával csak az élek száma nő, ezért $\nu(G') = \nu(G) + 1$. A b) esetben az élek száma eggyel nő, ugyanakkor eggyel csökken a komponensek száma, tehát $\nu(G') = \nu(G)$. □

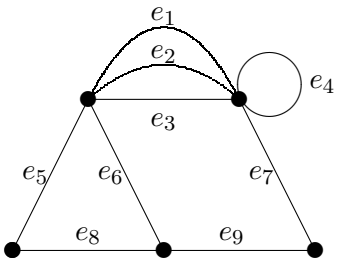
Legyen $E = \{e_1, e_2, \ldots, e_m\}$ a gráf éleinek halmaza. Egy kör ábrázolható egy (karakterisztikus) vektor segítségével. Egy kör karakterisztikus vektora $\mathbf{v} = (v_1, v_2, \ldots v_m)$, ahol $v_i = 1$, ha az e_i él rajta van a körön, és $v_i = 0$ különben. Értelmezhatjük két ilyen vektor összeadását: $\mathbf{v} = \mathbf{v^1} + \mathbf{v^2}$, ahol

$$v_i = \begin{cases} 1 & \text{ha } v_i^1 + v_i^2 = 1 \\ 0 & \text{ha } v_i^1 + v_i^2 = 2 \text{ vagy } v_i^1 + v_i^2 = 0 \end{cases}$$

A $c_1, c_2, \ldots c_p$ körök függetlenek, ha a megfelelő $\mathbf{v^1}$, $\mathbf{v^2}$, ..., $\mathbf{v^p}$ karakterisztikus vektorok függetlenek, azaz ha tetszőleges $\alpha_i \in \{0,1\}$ értékekre

$$\alpha_1\mathbf{v^1} + \alpha_2\mathbf{v^2} + \ldots + \alpha_p\mathbf{v^p} = \mathbf{0} \;\Rightarrow\; \alpha_1 = 0, \alpha_2 = 0, \ldots \alpha_p = 0,$$

ahol $\mathbf{0} = (0, 0, \ldots, 0)$.



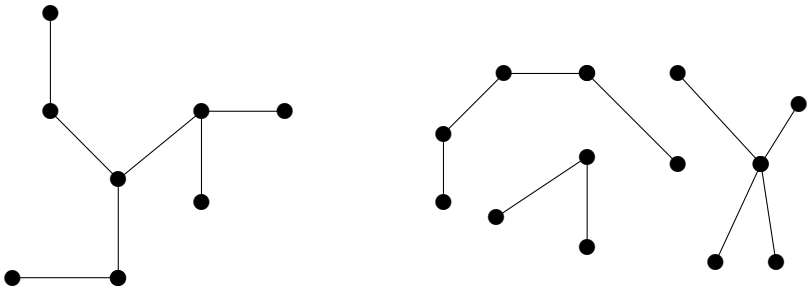
Az $[e_1, e_6, e_9, e_7]$ kör karakterisztiksu vektora $(1,0,0,0,0,1,1,0,1)$. Az $[e_5, e_6, e_8]$, $[e_6, e_3, e_7, e_9]$ és $[e_3, e_5, e_8, e_9, e_7]$ körök nem függetlenek, mert a megfelelő karakterisztikus vektorokra:
$(0,0,0,0,1,1,0,1,0) + (0,0,1,0,0,1,1,0,1) + (0,0,1,0,1,0,1,1,1) = 0$.

Egy maximális független körhalmaz[1] *fundamentális körrendszert* vagy *alapkörrendszert* képez. Bármely kör, amely nem eleme egy alapkörrendszernek, kifejezhető az alapkörrendszer elemeinek lineáris kombinációjaként.

$\nu(G)$ a G alapkörrendszere elemeinek a száma. Ha $\nu(G) = 0$, akor a G gráfban nincs kör.

Ha egy gráf körmentes, akkor a neve *liget* (vagy *erdő*). Egy összefüggő körmentes gráfot *fának* nevezünk. A liget több fából áll. A ligetre fennáll, hogy $\nu(G) = 0$.

Ligetben vagy fában az elsőfokú csúcsokat *levélnek* nevezzük. Minden fában legalább két levél van.

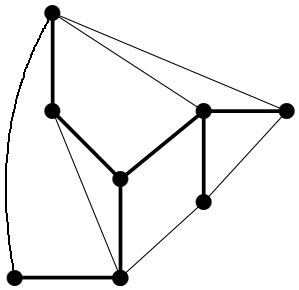


fa liget

A G gráf *feszítőfája* vagy *faváza* a G olyan részgráfja, amely fa és tartalmazza a G gráf minden csúcsát (azaz olyan feszítő részgráf, amely fa). Fában minden

[1]Itt a maximális azt jelenti, hogy tetszőleges kör hozzáadásával a rendszerhez, az már nem lesz független.

él híd.



feszítőfa

15. tétel. *Legyen G egy n-csúcsú gráf. A következő állítások egyenértékűek,* és a fákat jellemzik.

(1) G összefüggő és körmentes.

(2) G körmentes és $n-1$ éle van.

(3) G összefüggő és $n-1$ éle van.

(4) G körmentes, de bármely két nem szomszédos csúcsának összekötésével kör keletkezik.

(5) G összefüggő, de bármely élének törlésével szétesik két komponensre.

(6) A gráf bármely két csúcsát pontosan egy út köti össze.

Bizonyítás. *(1) ⇒ (2)*: Ha G összefüggő, akkor $k = \kappa(G) = 1$ (egyetlen komponensből áll), és mivel körmentes, következik, hogy $\nu(G) = m-n+1 = 0$, ahonnan $m = n-1$.

(2) ⇒ (3): G körmentes, tehát $\nu(G) = m-n+k = 0$ és, mivel $m = n-1 \Rightarrow k = 1$, tehát a gráf összefüggő.

(3) ⇒ (4): Mivel a gráf összefüggő, ezért $k = 1$, és $m = n-1 \Rightarrow \nu(G) = 0$. Tehát G körmentes. Új él hozzáadásával $\nu(G)$-ben csak m értéke nő, tehát $\nu(G) = 1$, ami azt jelenti, hogy egy kör keletkezik.

(4) ⇒ (5): Ha G nem lenne összefüggő, akkor két komponense egy-egy csúcsát összekötve, nem jelenik meg kör, amely ellentmondás *(4)*-gyel, tehát G-nek összefüggőnek kell lennie.

G körmentes és összefüggő, tehát $\nu(G) = m-n+1 = 0$, innen $m = n-1$. Ha törlünk egy élt, akkor m értéke csökken, tehát k-nsk 2-vel kell egyenlőnek lennie, hogy $\nu(G) = m-n+k = 0$ legyen. Ezért egy él törlésével a gráf szétesik két komponensre.

(5) ⇒ (6): Mivel G összefüggő, ha bármely két csúcsa között két különböző út lenne, akkor lenne benne kör, de ekkor a körből elhagyva egy élt, a gráf nem esne szét két komponenre, tehát ellentmondáshoz jutunk.

(6) ⇒ (1): Ha a gráf bármely két csúcsa között van egy út, akkor a gráf összefüggő. Ha pedig G tartalmazan kört, akkor a kör bármely két csúcsa között két út lenne, ami ellentmondás. □

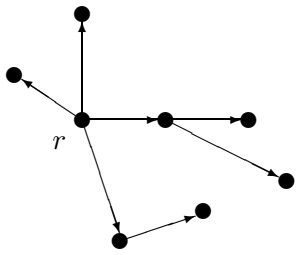
16. tétel. *Ha egy gráf T részgráfja a következő tulajdonságok közül bármely hárommal rendelkezik, akkor T feszítőfa.*

(1) T összefüggő.
(2) T körmentes.
(3) T-nek n csúcsa van.
(4) T-nek $n-1$ éle van.

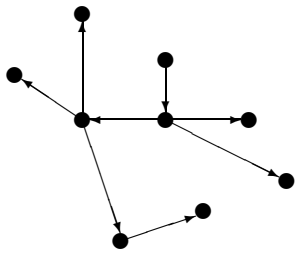
Megjegyzés. A *(2)* és *(4)* tulajdonságok együtt biztosítják, hogy T feszítőfa legyen, nem kell egy harmadik tulajdonság.

Gyökeres fák

A gyökeres fa olyan irányított élű fa, amelyben kijelöltünk egy *gyökérnek* nevezett r csúcsot, azzal a tulajdonsággal, hogy bármely v csúcsára igaz legyen, hogy létezik r–v irányított út. Ha egy fában kijelöltünk egy csúcsot gyökérnek, akkor az éleket nem is fontos irányítani, hisz mindig úgy tekinthető, hogy a gyökértől a levelek felé vannnak irányítva. Egy él kezdőcsúcsa az *ős*, a végcsúcsa pedig a *leszármazott*. A gyökér sohasem tekinthető levélnek, akkor sem, ha a foka egyenlő eggyel.



gyökeres fa



nem gyökeres fa (de lehetne)

Példánk második gráfja nem gyökeres fa, mert nincs kijelölt gyökere, de van olyan csúcsa, amelyet ha kijelölnénk, akkor teljesülnének az értelmezés kikötései.

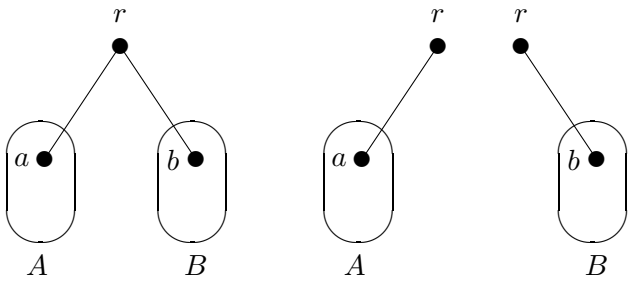
Bináris fák

A bináris fák olyan sajátos gyökeres fák, amelyeknek élei nem irányítottak, ennek ellenére mindig úgy tekintjük, mintha azok a gyökértől a levelek fele lennének irányítva. A bináris fákat rekurzívan értelmezzük.

1. Egy csúcs bináris fa, és gyökér a neve.

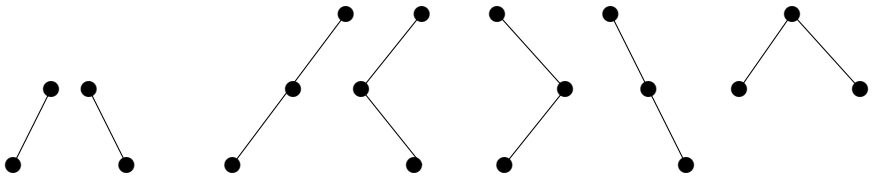
2. Ha az A és B, a és b gyökerű bináris fák, akkor bináris fák a következők is, amelyekben A bal oldali részfa, míg B jobb oldali részfa:
– egy r gyökerű fa, amelyben r egy-egy éllel kapcsolódik a-hoz és b-hez,
– egy r gyökerű fa, amelyben r egy éllel kapcsolódik a-hoz,
– egy r gyökerű fa, amelyben r egy éllel kapcsolódik b-hez.

Grafikusan ábrázolva:



A bináris fákat mindig úgy rajzoljuk le, hogy felül van a gyökére, alatta a többi csúcs. Amint az értelmezésből is látszik, a bináris fáknál megkülönböztetjük a bal és a jobb oldali részfákat. Ha ezeket felcseréljük, akkor más bináris fát kapunk, annak ellenére, hogy ezek mint gráfok, izomorfak.

Példaként felsoroljuk az összes két- és háromcsúcsú bináris fát.



n=2 n=3

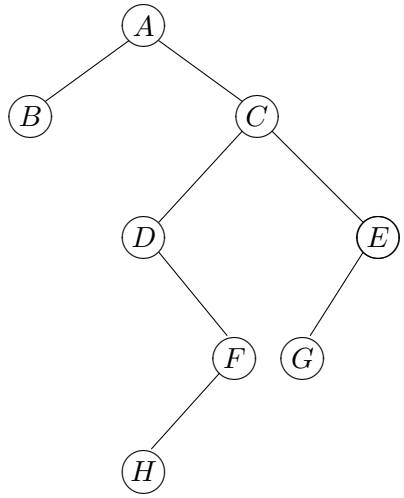
A bináris fa csúcsait a többféle módszerrel járhatjuk be:

– gyökérkezdő (preorder) bejárás: először megvizsgáljuk a gyökeret, majd bejárjuk a bal, azután pedig a jobb oldali részfát,

– gyökérközepű (inorder) bejárás: először bejárjuk a bal oldali részfát, azután megvizsgáljuk a gyökeret, majd bejárjuk a jobb oldali részfát,

– gyökőrvégző (postorder) bejárás: először bejárjuk a bal, azután a jobb oldali részfát, majd megvizsgáljuk a gyökeret.

Példa.



preorder: A, B, C, D, F, H, E, G

inorder: B, A, D, H, F, C, G, E

postorder: B, H, F, D, G, E, C, A

A különféle bejárásokat a következő eljárásokkal írhatjuk le, ahol B jelöl egy bináris fát, B_L a bal oldali, B_R pedig a jobb oldali részfáját:

PREORDER(B)
1. **if** B nem üres **then**
2. **return** B gyökere
3. PREORDER(B_L)
4. PREORDER(B_R)

INORDER(B)
1. **if** B nem üres **then**
2. INORDER(B_L)
3. **return** B gyökere
4. INEORDER(B_R)

POSTORDER(B)
1. **if** B nem üres **then**
2. POSTORDER(B_L)
3. POSTORDER(B_R)
4. **return** B gyökere

6.2. Gazdaságos feszítőfák

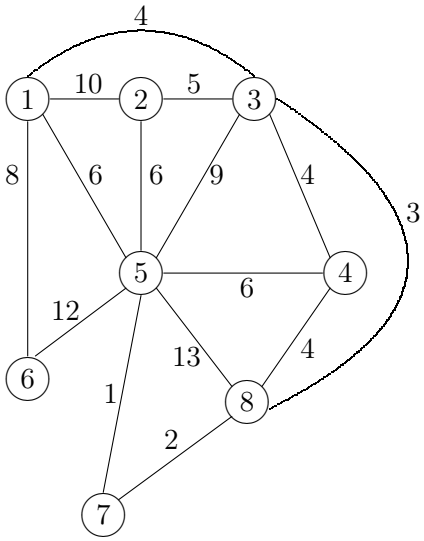
Súlyozott gráfban egy feszítőfa értéke az éleihez rendelt súlyok összege. Adott súlyozott gráfban keressük a legkisebb értékű feszítőfát, amelyet *minimális fes-*

zítőfának nevezünk. Két algoritmust mutatunk be minimális feszítőfa megkeresésére.

6.2.1. Kruskal algoritmusa

A gráf éleit súlyuk szerint növekvő sorrendbe rendezzük. Az első él a sorból bekerül a leendő gazdaságos favázba (az alábbi algoritmusban a leendő favázba bekerülő éleket megcsillagozzuk). Kezdetben a gráf minden csúcsa egy-egy halmazt képez. Egy él akkor kerül be a favázba, ha végpontjai különböző halmazból valók, és ekkor a két megfelelő halmazt egyesítjük. Az algoritmus akkor ér véget, amikor a gráf minden csúcsa egy halmazban van.

Példa.



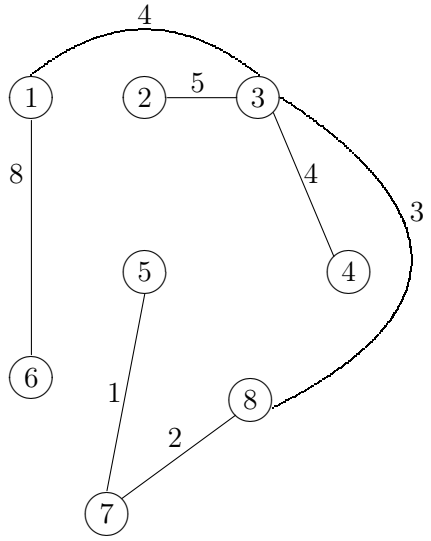
Az első oszlopban az élek vannak, a másodikban a megfelelő súly értéke, a harmadikban csillag, ha az él bekerült a favázba, a negyedik oszlopban pedig a csúcshalmazok.

```
                    {1}, {2}, {3}, {4}, {5}, {6}, {7}, {8}

{5,7}     1   *     {5,7}, {1}, {2}, {3}, {4}, {6}, [8}
{7,8}     2   *     {5,7,8}, {1}, {2}, {3}, {4}, {6}
{3,8}     3   *     {3,5,7,8}, {1}, {2}, {4}, {6}
{1,3}     4   *     {1,3,5,7,8}, {2}, {4}, {6}
{3,4}     4   *     {1,3,4,5,7,8}, {2}, {6}
{4,8}     4
{2,3}     5   *     {1,2,3,4,5,7,8}, {6}
{1,5}     6
{2,5}     6
```

```
{4,5}     6
{1,6}     8   *    {1,2,3,4,5,6,7,8}
{3,5}     9
{1,2}    10
{5,6}    12
{5,8}    13
```

A csillaggal megjelölt élek a gazdaságos faváz élei. Maga a faváz a következő:



Az algoritmus leírásához tekintsük az élek $E = \{e_1, e_2, \ldots, e_m\}$ halmazát úgy, hogy $\mathcal{W}(e_i) \leq \mathcal{W}(e_{i+1})$, minden $i = 1, 2, \ldots, m-1$ értékre (azaz, az élek súlyuk szerint növekvő sorrendben vannak indexelve). Halmazok helyett egy $h = (h_1, h_2, \ldots, h_n)$ vektort használunk (n a csúcsok száma), amelynek elemei kezdetben egyenlőek az indexükkel, ami arra utal, hogy különböző halmazok elemei. Amikor két halmazt egyesítünk, a megfelelő h_i értékeket egyenlővé tesszük (egyik halmaz elemeinek h_i értékeit a másik halmaz h_i értékeire állítjuk.).

KRUSKAL(E)
1. **for** j=1,2, . . . , n **do**
2. $\quad h_j := j$
3. $i := 1$
4. **while** h elemei különbözőek **do**
5. $\quad$ **if** (e_i végpontjai v_k, v_l) és ($h_k \neq h_l$) **then**
6. $\quad\quad$ **return** e_i
7. $\quad\quad$ **for** j:=1, 2,. . . , n **do**
8. $\quad\quad\quad$ **if** $h_j = h_l$ **then**
9. $\quad\quad\quad\quad h_j := h_k$

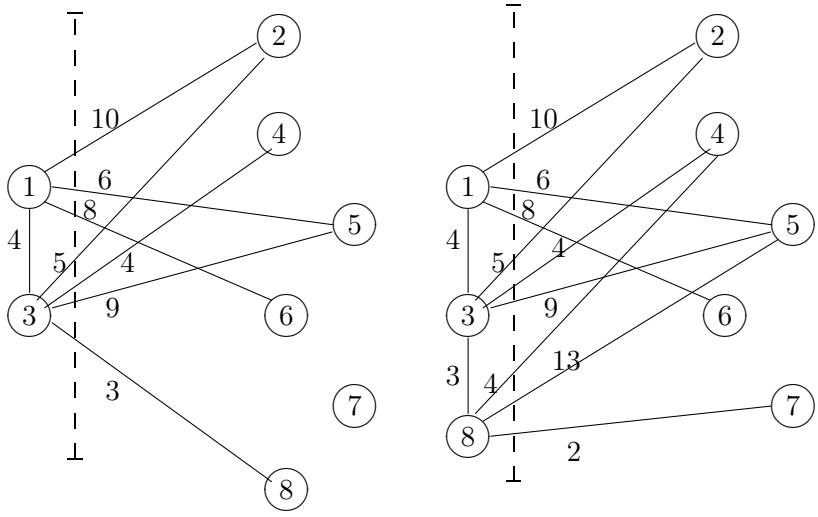
10. i:=i+1

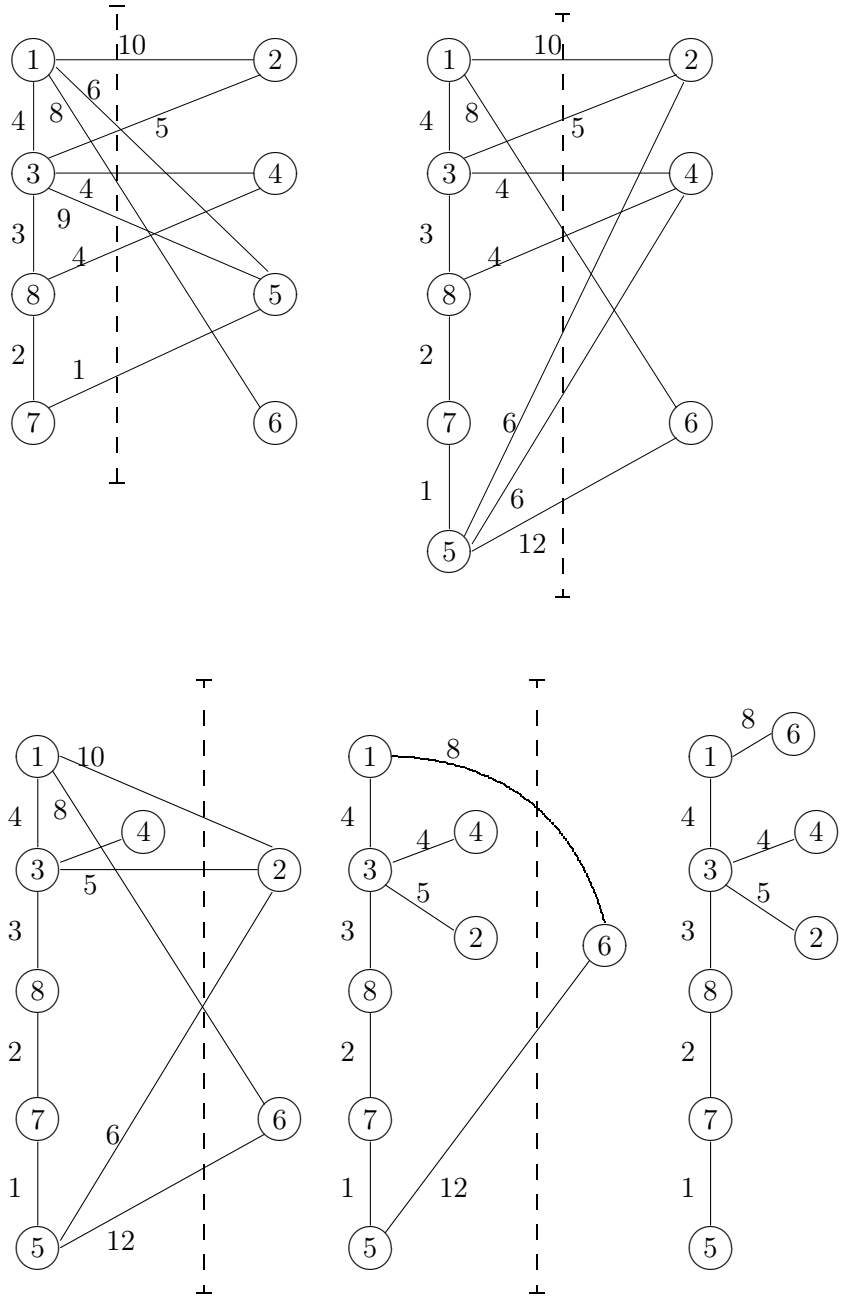
6.2.2. Prim algoritmusa

Az algoritmus alapötlete az, hogy egy adott x csúcshoz illeszkedő összes él közül a legkisebb súlyú mindenképpen benne van a gazdaságos favázban. Ha nem így lenne, akkor a gazdaságos favázhoz hozzáadva ezt az élt, egy kör keletkezne. Ebből a körből elhagyva az x-hez illeszkedő minden más élt, egy kisebb értékű favázat kapunk, ami ellentmondás.

Tetszőleges csúccsal kezdünk. Ezt a csúcsot tegyük be az A halmazba, a többi pedig legyen a B halmazban. (Minden lépésben $A \cup B = V$.) Tekintsük a két halmazt összekötő éleket. Válasszuk ki közülük a legkizebb súlyút. Tegyük át az A halmazba ennek az élnek a B halmazba eső végpontját. Folytassuk mindaddig amíg minden csúcs átkerül A-ba. Az algoritmus során kiválasztott élek a gazdaságos faváz élei lesznek.

Az előbbi példa esetében az algoritmus lépései a következők. Az első lépést elhagyhatjuk, ha egyből a legkisebb súlyú éllel indulunk, és annak mindkét végpontját betesszük az A halmazba.





Legyen $G = (V, E, \mathcal{W})$ egy súlyozott egyszerű gráf. Az algoritmus kezdőcsúcsként az x-et használja.

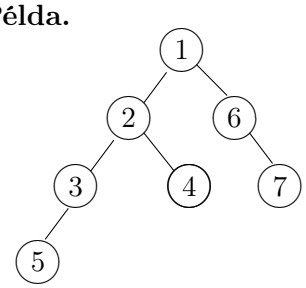
PRIM(G, x)
1. $A := \{x\}$
2 $B := V \setminus A$
3. **while** $A \neq V$ **do**

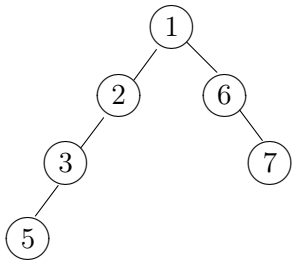
4. legyen $\{a, b\} \in E$, $a \in A$, $b \in B$ a legkisebb súlyú él az összes A és B közötti él közül
5. **return** $\{a, b\}$
6. $A := A \cup \{b\}$
7. $B := B \setminus \{b\}$

6.3. A Prüfer-kód

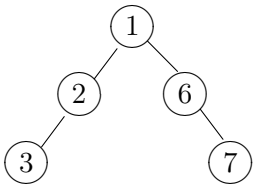
Egy n-csúcsú gyökeres fának a Prüfer-kódja egy $n-1$ természetes számból álló sorozat. Címkézzük meg a fa csúcsait tetszőleges módon az $1, 2, \ldots, n$ természetes számokkal. Válasszuk ki a levelek közül a legkisebb címkéjűt, töröljük ki a fából, majd írjuk be a sorozatba az őse címkéjét. Hasonlóképpen járjunk el a következőkben mindaddig, amíg van levél a fában. Az eredményül kapott sorozat a fa Prüfer-kódja. A kód utolsó eleme a gyökér címkéje.

Példa.

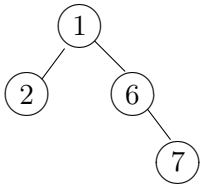




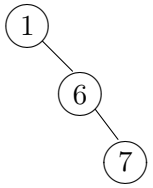
2



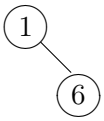
2,3



2,3,2



2,3,2,1



2,3,2,1,6

2,3,2,1,6,1

A következő algoritmus egy n-csúcsú F fa Prüfer-kódját adja meg. A fa csúcsait az $1, 2, \ldots, n$ számokkal címkézzük meg.

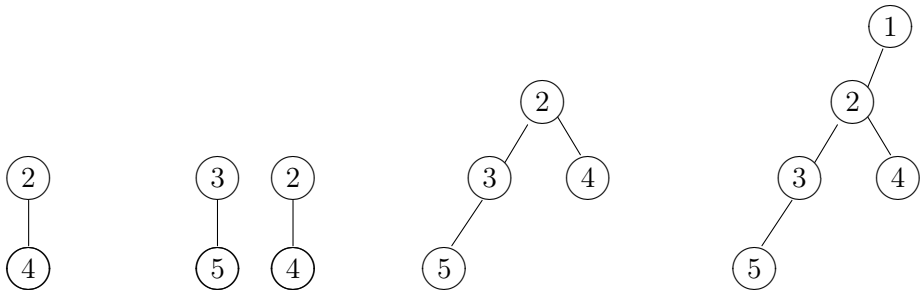
PRÜFERKÓDOLÁS(F)
1. legyen K üres sorozat
2. **while** F nemcsak gyökérből áll **do**
3. legyen v a legkisebb címkéjű levél F-ben
4. írjuk be K-ba v ősét
5. töröljük v-t F-ből
6. **return** K

Visszakódolás

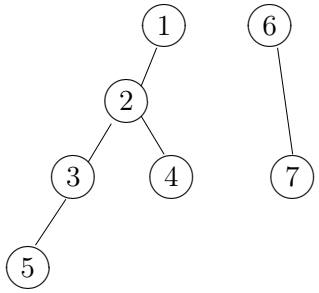
A Prüfer-kód egy véges számsorozat. Legyen a az első természetes szám a sorozatban. Keressük meg azt a legkisebb b természetes számot, amely nincs benne a sorozatban. Rajzoljuk élt a fában a-ból b-be. Töröljük ki sorozat első elemét (az a-t), és írjuk be a végére a b-t. Folytassuk mindaddig, amíg elfogynak az eredeti sorozat elemei.

Példa.

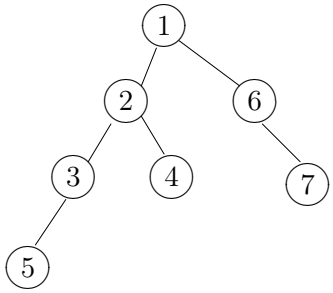
$2, 3, 2, 1, 6, 1 \parallel 4$
$3, 2, 1, 6, 1, 4 \parallel 5$
$2, 1, 6, 1, 4, 5 \parallel 3$
$1, 6, 1, 4, 5, 3 \parallel 2$
$6, 1, 4, 5, 3, 2 \parallel 7$
$1, 4, 5, 3, 2, 7 \parallel 6$
$4, 5, 3, 2, 7, 6$



1 2 3 4



5



6

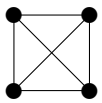
A küvetkező algoritmus a visszakódolást végzi el. Bemenetként a K sorozatot kapja, valamint a fa csúcsainak n számát.

PrüferDekódolás(K, n)
1. legyen F egy üres gráf
2. **for** $i = 1, 2, \ldots, n-1$ **do**
3. legyen x a K sorozat első eleme
4. legyen y a legkisebb természetes szám, amely nincs benne K-ban
5. rajzoljunk egy (x, y) élt F-be
6. töröljük x-et a K elejéről, és adjuk hozzá a végére y-t
7. **return** F

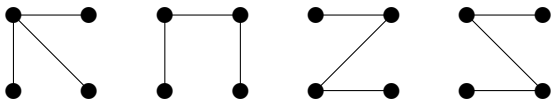
Egy n-csúcsú fa Prüfer-kódja $n-1$ elemből áll. Az utolsó elem mindig a gyökér címkéje. Az első $n-2$ elem bármelyik lehet az első n természetes számból. Ezért összesen n^{n-2} ilyen ismétlődéses variáció van. Ez pedig megegyezik az összes címkézett n-csúcsú fával (lényegtelen, hogy melyik csúcs a gyökér). Ez az ún. *Cayley-tétel*, amelyet a következőképpen is kijelenthetünk.

17. tétel (Cayley). *Ha egy n-csúcsú teljes gráf csúcsait az első n természetes számmal címkézzük meg, akkor a gráfnak n^{n-2} különböző feszítő fája van.*

Példa. A K_4 gráf esetében $4^2 = 16$ különböző feszítő fa van.



K_4



a) *b*) *c*) *d*)

Az *a*), *b*), *c*) és *d*) típusok mindegyikéből pontosan 4 van.

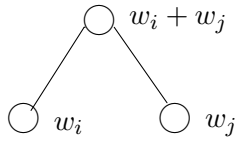
6.4. Huffman algoritmusa

Tekintsünk egy gyökeres fát, amelynek $v_1, v_2, \ldots, v_k$ levelei rendre a $w_1, w_2, \ldots, w_k$ súlyokkal rendelkeznek. Ha a gyökértől egy v_j levélig az út

hosszát l_j-vel jelöljük, akkor értelmezzük a $\sum_{j=1}^{k} w_j l_j$ értéket, amelynek neve *súlyozott úthossz.*

Feladatunk, hogy adott véges számsorozathoz mint levelekhez rendelt súlyokhoz, keressünk minimális súlyozott úthosszú bináris fát. Erre Huffman a következő ötletes algoritmust ajánlotta.

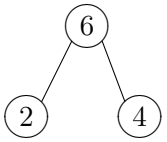
Válasszuk ki a sorozatból a két legkisebbet, legyenek ezek w_i és w_j, töröljük ki őket a sorozatból, majd adjuk hozzá a sorozathoz a $w_i + w_j$ számot, aztán pedig adjuk hozzá a keresendő fához a következő részfát:



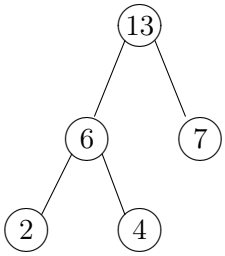
Folytassuk az eljárást mindaddig, amíg a sorozat egyetlen számmá zsugorodik. Az így kapott bináris fa a keresett minimális súlyozott úthosszú bináris fa.

Példa.

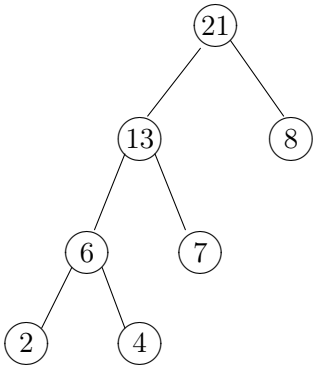
A levelekhez rendelt súlyok a következők: $2, 4, 7, 8, 13, 19, 20$.



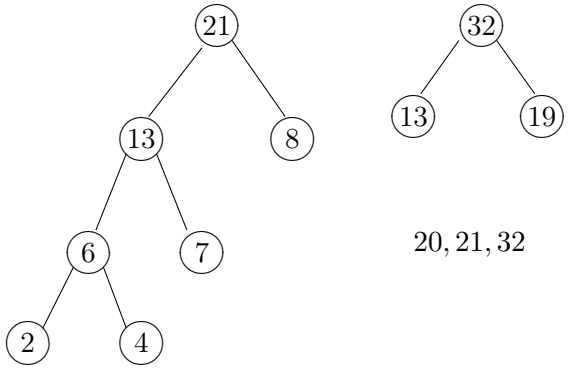
$6, 7, 8, 13, 19, 20$



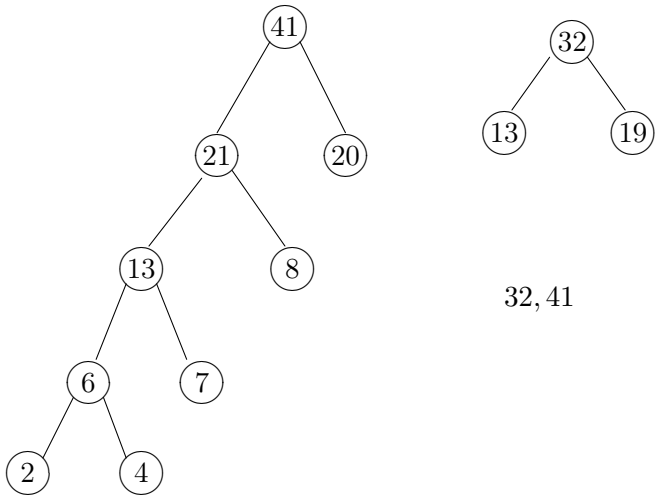
$8, 13, 13, 19, 20$



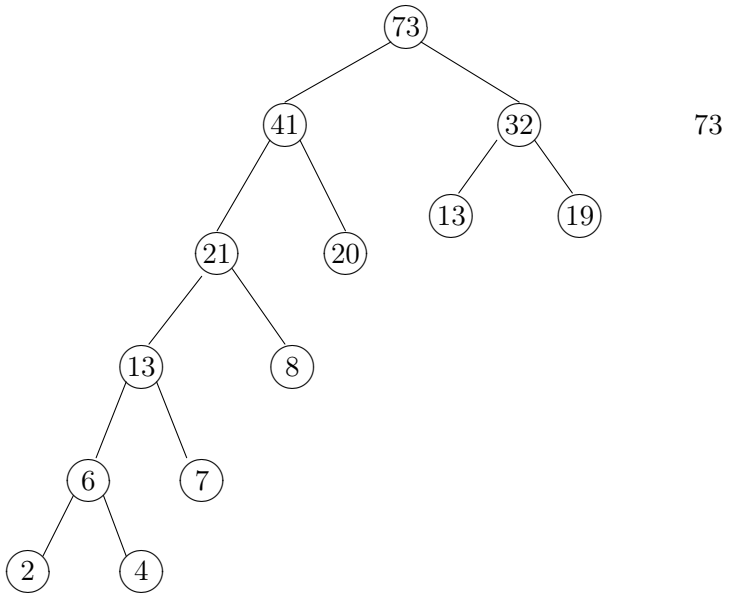
$13, 19, 20, 21$



$20, 21, 32$



$32, 41$

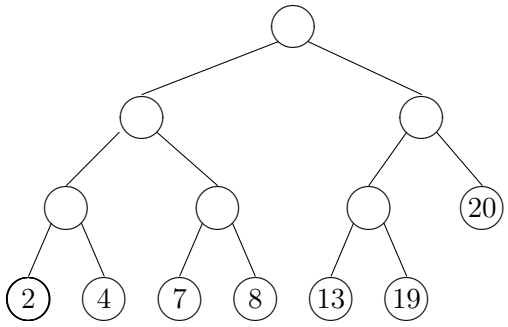


73

A minimális súlyozott úthosszú ebben az esetben:

$$2 \cdot 5 + 4 \cdot 5 + 7 \cdot 4 + 8 \cdot 3 + 20 \cdot 2 + 13 \cdot 2 + 19 \cdot 2 = 186,$$

amely minimális. Bármilyen más, ugyanazokkal a súlyozott levelekkel rendelkező bináris fa esetében a súlyozott úthossz ennél nem kisebb.
Például a következő fa esetében



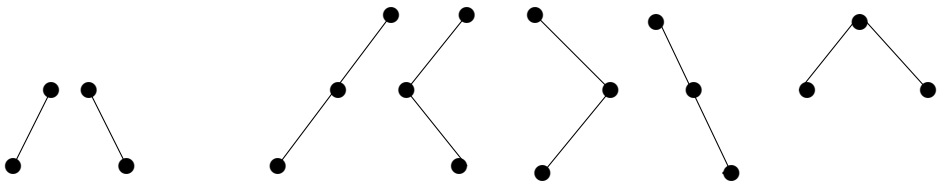
a súlyozott úthossz: $(2 + 4 + 7 + 8 + 13 + 19) \cdot 3 + 20 \cdot 2 = 199$.

A következő algoritmus bemenete a $W = (w_1, w_2, \ldots, w_k)$ sorozat, amelynek elemei a levelekhez rendelt súlyok.

HUFFMAN(W)
1. legyen F egy üres fa
2. **while** W egynél több elemet tartalmaz **do**
3. válasszuk ki a W legkisebb elemét, legyenek ezek u és v
4. rajzoljuk be F-be az $\{u, u+v\}$ és $\{v, u+v\}$ éleket
5. töröljük ki W-ből u-t és v-t, majd írjuk be helyettük az $u+v$-t
5. **return** F

6.5. Bináris fák száma

Jelöljük b_n-nel az n csúcsú bináris fák számát. Ekkor $b_1 = 1$, $b_2 = 2$, $b_3 = 5$ (lásd az ábrát). Legyen $b_0 = 1$. (Később látni fogjuk, hogy ez jó választás.)



n=2 n=3

Ha rögzítjük egy n csúcsú bináris fa gyökerét, akkor még $n-1$ csúcs marad a bal és jobb részfában összesen. Ha k csúcs van a bal oldali, $n-1-k$ pedig a jobb oldali részfában, akkor összesen $b_k b_{n-1-k}$ ilyen bináris fa létezik. Összegezve $k = 0, 1, \ldots, n-1$ értékekre, pontosan b_n-t kapjuk. Tehát tetszőleges $n \geq 1$ természetes számra a b_n-ben megoldandó rekurzív egyenlet a következő:

$$b_n = b_0 b_{n-1} + b_1 b_{n-2} + \cdots + b_{n-1} b_0. \tag{6.1}$$

Ez még így is írható:

$$b_n = \sum_{k=0}^{n-1} b_k b_{n-1-k}.$$

A fenti rekurzív egyenlet mindkét oldalát z^n-nel szorozva, majd n szerint összegezve, a következőt kapjuk:

$$\sum_{n=1}^{\infty} b_n z^n = \sum_{n=1}^{\infty} \left(\sum_{k=0}^{n-1} b_k b_{n-1-k} \right) z^n. \tag{6.2}$$

Legyen $B(z) = \sum_{n=0}^{\infty} b_n z^n$ a b_n számok generátorfüggvénye. Az (6.1) összefüggés bal oldala éppen $B(z) - 1$ (mivel $b_0 = 1$). A jobb oldal nagyon hasonlít két generátorfüggvény szorzatához. Hogy észrevegyük, melyik két függvényről van szó, használjuk a következő jelölést:

$$A(z) = zB(z) = \sum_{n=0}^{\infty} b_n z^{n+1} = \sum_{n=1}^{\infty} b_{n-1} z^n.$$

Ekkor az (6.2) jobb oldala éppen $A(z)B(z)$, ami egyenlő $zB^2(z)$-vel. Innen

$$B(z) - 1 = zB^2(z), \quad B(0) = 1.$$

Oldjuk meg ezt az egyenletet $B(z)$-ben! Ekkor

$$B(z) = \frac{1 \pm \sqrt{1-4z}}{2z}\ .$$

Mivel $B(0) = 1$ csak a negatív jel megfelelő.
$B(z)$ kifejtésében alkalmazzuk az általánosított binomiális képletet[2]:

[2]A binomiális képlet általánosítható tetszőleges valós r-re is, vagyis

$$(1+z)^r = \sum_{n=0}^{\infty} \binom{r}{n} z^n.$$

$$\begin{aligned}
B(z) &= \frac{1}{2z}\left(1-\sqrt{1-4z}\right) = \frac{1}{2z}\left(1-(1-4z)^{1/2}\right) \\
&= \frac{1}{2z}\left(1-\sum_{n=0}^{\infty}\binom{1/2}{n}(-4z)^n\right) = \frac{1}{2z}\left(1-\sum_{n=0}^{\infty}\binom{1/2}{n}(-1)^n 2^{2n} z^n\right) \\
&= \frac{1}{2z} - \binom{1/2}{0}\frac{2^0 z^0}{2z} + \binom{1/2}{1}\frac{2^2 z}{2z} - \cdots - \binom{1/2}{n}(-1)^n\frac{2^{2n}z^n}{2z} + \cdots \\
&= \binom{1/2}{1}2 - \binom{1/2}{2}2^3 z + \cdots - \binom{1/2}{n}(-1)^n 2^{2n-1} z^{n-1} + \cdots \\
&= \sum_{n\geq 0}\binom{1/2}{n+1}(-1)^n 2^{2n+1} z^n = \sum_{n=0}^{\infty}\frac{1}{n+1}\binom{2n}{n} z^n.
\end{aligned}$$

Innen $b_n = \dfrac{1}{n+1}\binom{2n}{n}$, amely azonos a C_n-nel jelölt ún. Catalan-számmal.

Megjegyzés. Az utolsó átalakításnál felhasználtuk a következő, könnyen bizonyítható összefüggést:

$$\binom{1/2}{n+1} = \frac{(-1)^n}{2^{2n+1}(n+1)}\binom{2n}{n}.$$

Itt az $\binom{r}{n}$ a kombináció általánosítása valós r-re, vagyis

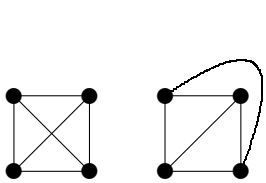
$$\binom{r}{n} = \begin{cases} \dfrac{r(r-1)(r-2)\ldots(r-n+1)}{n(n-1)\ldots 1}, & \text{ha } n>0, \\ 1, & \text{ha } n=0, \\ 0, & \text{ha } n<0. \end{cases}$$

7. fejezet

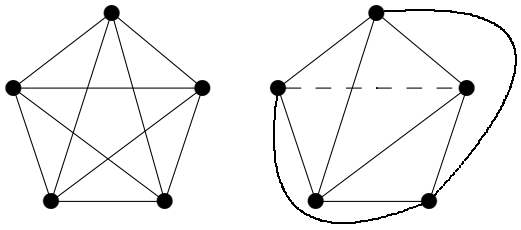
Síkba rajzolható gráfok

Erdős Pál világhírű matematikusnak volt egy szellemes mondása arról, hogy Istennek van egy könyve, a Transzfinit Könyv, amelyben a matematikai tételek mindegyike benne van, méghozzá a legszebb bizonyítással. Az Euler-képlet egyik alábbi bizonyítása minden bizonnyal ebből a könyvből van.

Egy gráf *síkba rajzolható*, ha lerajzolható a síkban úgy, hogy élei a csúcsokon kívül nem metszik egymást. Egy síkba rajzolható gráfot röviden *síkgráfnak* is nevezünk.

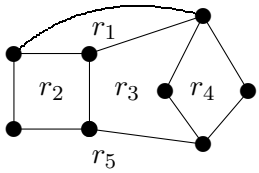


K_4



K_5

A fenti ábrán K_4 lerajzolható úgy, hogy élei ne messék egymást, de K_5 már nem.



$n = 8$
$m = 11$
$r = 5$

A síkgráf élei és csúcsai tartományokat határoznak meg (például r_1, r_2, r_3, r_4 és r_5 a fenti gráfban). Ezek közül egy végtelen, a többi véges.

18. tétel. *Egy síkgráfban a véges tartományokat határoló élek alapkörrendszert alkotnak.*

Bizonyítás. A bizonyítást a tartományok száma szerinti indukcióval végezzük. Minden új tartományt legalább egy új él is határol. Tehát, ha a tartományok száma eggyel nő, akkor eggyel nő az alapkörök száma is. □

Tehát a tartományok száma eggyel nagyobb, mint a ciklomatikus szám (a végtelen tartomány kiatt), azaz $r = \nu(G) + 1$. Mivel $\nu(G) = m - n + 1$, kijelenthetjük a következő tételt.

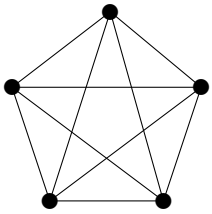
19. tétel (Euler). *Ha egy összefüggő gráfnak n csúcsa, m éle és r tartománya van, akkor*

$$n - m + r = 2.$$

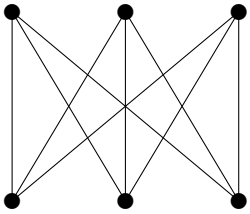
Más bizonyítés: Vizsgáljuk meg az $n-m+r$ értéket egy tetszőleges síkgráfban. Ha kitörlünk egy körből egy élt, akkor az élek száma is és a tartományok száma is eggyel csökken, a csúcsok száma változatlan marad. Tehát az $n-m+r$ kifejezés állandó bármely síkgráf esetében. Addig folytatjuk a körön levő élek törlését, amíg fát nem kapunk. Ebben az esetben is $n - m + r$ értéke ugyanaz az állandó. De fa esetében ez könnyen kiszámíthatü, mivel $m = n - 1$ és $r = 1$ (csak a végtelen tartomány maradt). Így $n - m + r = n - (n - 1) + 1 = 2$. Tehát az állandó értéke 2. Így $n - m + r = 2$.

A `http://www.ics.uci.edu/~eppstein/junkyard/euler/` címen a tételnek 19 bizonyítása található.

Euler képletét felhasználhatjuk arra, hogy bebizonyítsuk, hogy a K_5 éd $K_{3,3}$ gráfok nem síkgráfok.



K_5



$K_{3,3}$

20. tétel. *K_5 nem síkgráf.*

Bizonyítás. Mivel $n = 5$, $m = 10$, és minden tartományt legalább három él határol:

$$3r \leq 2m \quad \Rightarrow \quad r \leq \frac{2m}{3} = \frac{20}{3}.$$

De r egész szám lévén, $r \leq 6$. Ka K_5 síkgráf, akkor érvényes rá Euler képlete, azaz $r = 2 - n + m == 2 - 5 + 10 = 7$, ami ellentmondás. □

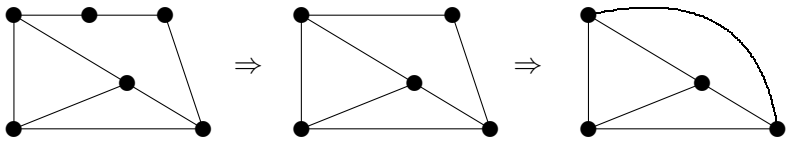
21. tétel. *$K_{3,3}$ nem síkgráf.*

Bizonyítás. Mivel $n = 6$, $m = 9$. és minden tartományt legalább négy él határol:

$$4r \leq 2m \quad \Rightarrow \quad r \leq \frac{m}{2} = \frac{9}{2}.$$

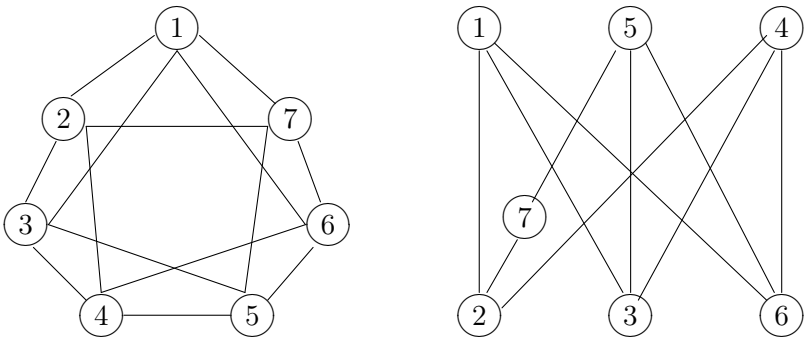
De r egész szám, ezért $r \leq 4$. Ha $K_{3,3}$ síkgráf, akkor Euler képlete alapján $r = 2 - n + m = 2 - 6 + 9 = 5$, ami ellentmondás. □

Értelmezzük a gráf *összevonását* mint azt a műveletet, amelynek során elhagyunk a gráfból egy kétfokú csúcsot, és a két hozzá illeszkedő élt egyetlen eggyel helyettesítjük, amely összeköti az elhagyott élek másik végpontjait.

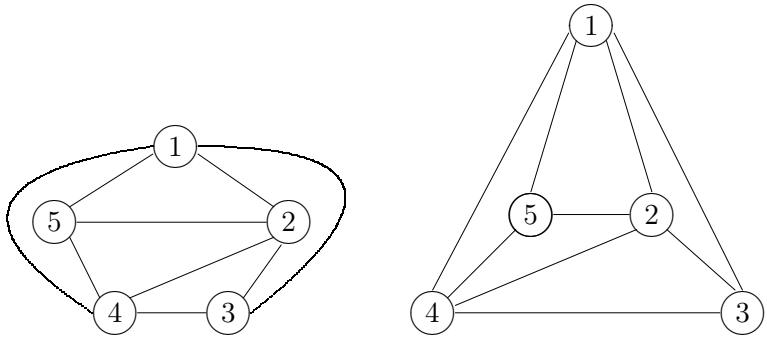


22. tétel (Kuratowski). *Egy összefüggő G gráf akkor és csakis akkor síkgráf, ha nem tartalmaz egyetlen olyan részgráfot sem, amely a K_5 vagy $K_{3,3}$ gráfok valamelyikévé vonható össze.*

Az alábbi gráfban. miután tötöljük az $\{1,7\},\{6,7\}$ valamint a $\{2,3\},\{4,5\}$ éleket, a kapott gráf könnten összevonható $K_{3,3}$ gráffá, tehát az eredeti gráf nem síkgráf.



Minden síkgráf lerajzolható úgy is, hogy élei egyenes szakaszok legyenek. Például:



23. tétel. *Egy összefüggő egyszerű síkgráfban, ahol a csúcsok száma $n \geq 3$, igazak a következők:*

a) $r \leq 2(n-2)$

b) $m \leq 3(n-2)$.

Bizonyítás. *a*) Minden tartományt legalábbb három él határol, tehát $3r \leq 2m$. Euler képletéből

$$3r \leq 2m = 2(n+r-2), \quad \text{és innen} \quad r \leq 2n-4$$

b) Az *a*) pontbeli eredményt használva $m = n+r-2 \leq 3n-6$. □

24. tétel. *Egy egyszerű síkgráfban mindig létezik olyan csúcs, amelynek foka legfennebb 5.*

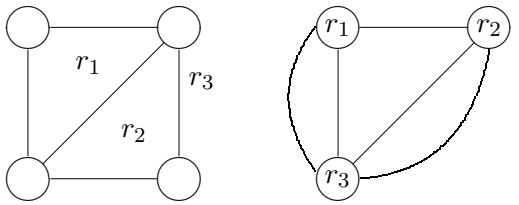
Bizonyítás. Ha minden csúcs fokszáma legalább 6, akkor

$$2m = \sum_{v \in V(G)} \deg(v) \geq 6n.$$

Ebből $m \geq 3n$ következik, amely ellentmondás az előbbi tétel eredményével, azaz azzal, hogy $m \leq 3n-6$. □

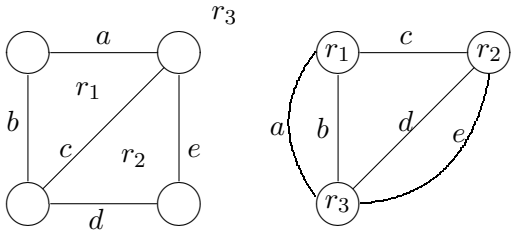
Duális gráfok

Értelmezzük egy G síkgráfnak a G^*-gal jelzett *duálisát*. A G gráf minden tartományának megfeleltetjük a G^* egy-egy csúcsát. G^*-ben két csúcs p párhuzamos éllel van összekötve, ha G-ben a megfelelő két tartománynak p közös határéle van.



G G^*

Egy gráfban egy élhalmaz *(él)vágat*, ha törlésével a gráf szétesik, azaz nem lesz összefüggő. Jelöljük az a, b, c, d, e betűkkel a fenti G gráf éleit. Két tartomány közös élének megfelelő élt G^*-ban ugyanazzal a betűvel jelöljük.



G G^*

A G gráf egy köre G^*-ban élvágat és fordítva, G^* minden élvágatának megfelel G-nek egy köre.

Ennek alapján általánosabban értelemzhetjük a dualitást.

Két gráf, G és G^* egymás duálisa, ha létrehozható éleik között olyan kölcsönös és egyértelmű megfeleltetés, hogy G bármelyik körének élei G^* élvágatnak felelnek meg, és fordítva, G^* bármely élvágatának élei G-ben kört alkotnak.

25. tétel. *Egy összefüggő gráf akkor és csakis akkor síkgráf, ha létezik duálisa.*

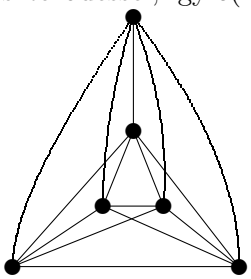
Keresztezési szám

Egy gráf *keresztezési száma* az legkisebb természetes szám, amely megegyezik a gráf összes lerajzolásai közül a lehető legkisebb élkereszteződések számával. Jelöljük ezt $c(G)$-vel (angolul *crossing number*). Síkgráf keresztezési száma 0. Amint már láttuk, $c(K_5) = 1$ és $c(K_{3,3}) = 1$.

26. tétel. $c(K_6) = 3$.

Bizonyítás. A K_6 gráf helyett értelmezzünk egy új gráfot: K_6 egy lerajzolásában bármely két élének keresztezési pontját tekintsük az új gráf egy-egy csúcsának, a megfelelő élekkel együtt. Ekkor az így kapott gráf síkgráf, és $n' = 6 + c$ csúcsa és $m' = 15 + 2c$ éle van (mivel K_6-ban 6 csúcs és 15 él van). Ezért $m' \leq 3n' - 6$, tehát $15 + 2c \leq 18 + 3c - 6$, azaz $c \geq 3$. De K_6 lerajzolható

3 élkereszteződéssel, így $c(K_6) = 3$.



□

Hasonló módon bizonyítható, hogy

$$c(K_n) \geq \frac{(n-3)(n-4)}{2}, \quad n \geq 3.$$

Felső határként ismeretes:

$$c(K_n) \leq \frac{1}{4} \left\lfloor \frac{n}{2} \right\rfloor \left\lfloor \frac{n-1}{2} \right\rfloor \left\lfloor \frac{n-2}{2} \right\rfloor \left\lfloor \frac{n-3}{2} \right\rfloor$$

Ha $n \leq 10$, akkor egyenlőség áll fenn. Ez sejtés tetszőleges n-re is.

Hasonlóképpen $K_{m,n}$-re

$$c(K_{m,n}) \leq \left\lfloor \frac{m}{2} \right\rfloor \left\lfloor \frac{m-1}{2} \right\rfloor \left\lfloor \frac{n}{2} \right\rfloor \left\lfloor \frac{n-1}{2} \right\rfloor.$$

$1 \leq \min(m, n) \leq 10$ esetében egyenlőség áll fenn.

8. fejezet

Folyamfeladatok

A folyamok mind az óceánba ömlenek. Menjenek, és hadd menjenek a többiek is. A nagy víz kiömlik, kitépi magát medréből és ömlik a korok, a fajok, a különböző lelkek törvénye szerint. A meder más, a víz ugyanaz – ömöljetek az óceánba.

(Ramakrisna mondásai, ford. Kemény Ildikó in Hamvas Béla: Anthologia humana)

8.1. A hálózati folyamokról

Szükségünk lesz a következő jelölésekre. Legyen X tetszőleges halmaz, $\mathbf{Z}$ pedig az egész számok halmaza. Ha adott egy tetszőleges $g : X \times X \to \mathbf{Z}$ függvény, akkor kiterjesztjük halmazokra a következőképpen.

$$\text{Ha } A, B \subseteq X, \text{ akkor } g(A,B) = \sum_{\substack{x \in A \\ y \in B}} g(x,y).$$

Tulajdonságok:

1) $g(A, B \cup C) = g(A,B) + g(A,C) - g(A, B \cap C)$
2) $g(A \cup B, C) = g(A,C) + g(B,C) - g(A \cap B, C)$
3) Ha $B \cap C = \emptyset$, akkor
   $g(A, B \cup C) = g(A,B) + g(A,C)$
   $g(B \cup C, A) = g(B,A) + g(C,A)$
4) Ha $f : X \times X \to \mathbf{Z}$, $g : X \times X \to \mathbf{Z}$, $h : X \times X \to \mathbf{Z}$ and $f = g + h$, akkor
   $f(A,B) = g(A,B) + h(A,B)$ for $A, B \subseteq X$.

HA $A = \{a\}$, akkor $f(\{a\}, B)$ helyett egyszerűen csak $f(a,B)$-t írunk. A következőkben a halmazokat nagy-, az elemeiket pedig kisbetűkkel jelöljük.

Egy *(közlekedési) hálózat* egy (V, E) irányított gráf a következő tulajdonságokkal:

a) $\exists s \in V : N^{\text{be}}(s) = \emptyset$, s *forrás*,

b) $\exists t \in V : N^{\text{ki}}(t) = \emptyset$, t *nyelő*,

c) Értelmezzük a *kapacitásfüggvényt* a következőképpen: $\alpha : V \times V \to \mathbf{Z}_+$ úgy, hogy

$$\alpha(x, y) > 0 \text{ if } (x, y) \in E,$$
$$\alpha(x, y) = 0 \text{ if } (x, y) \notin E.$$

A hálózat jelölése $H = (V, E, \alpha, s, t)$.

Folyamnak nevezzük az $f : V \times V \to \mathbf{Z}_+$ függvényt, amely a következő tulajdonságokkal rendelkezik:

1° $f(x, y) \leq \alpha(x, y)$ (*kapacitás-megszorítás*),

2° $f(x, V) = f(V, x)$ for all $x \in V \setminus \{s, t\}$ (*egyensúly-feltétel*).

A $v(f) = f(s, V)$ értéket az f *folyam értékének* nevezzük. Egy folyam *telít* egy élt, ha azon az élen a folyam értéke egyező a kapacitással. Ilyenkor ez az él *telített*.

27. tétel. *$v(f) = f(s, V) = f(V, t)$ bármely f folyamra.*

Bizonyítás.

$$\sum_{x \in V} \big(f(V, x) - f(x, V)\big) = f(V, V) - f(V, V) = 0,$$

másfelől

$$\sum_{x \in V} \big(f(V, x) - f(x, V)\big) = \underbrace{f(V, s)}_{=0} - f(s, V) +$$
$$+ \sum_{\substack{x \in V \\ x \neq s, t}} \underbrace{\big(f(V, x) - f(x, V)\big)}_{=0} +$$
$$+ f(V, t) - \underbrace{f(t, V)}_{=0} = f(V, t) - f(s, V).$$

□

Ennek a tételnek az alapján átfogalmazhatjuk a folyam kikötéseit:

1° $f(x, y) \leq \alpha(x, y)$,

$$2^\circ \; f(x, V) - f(V, x) = \begin{cases} v(f), & \text{ha } x = s \\ 0, & \text{ha } x \neq s, x \neq t \\ -v(f), & \text{ha } x = t \end{cases}$$

Feladat.

Keressünk maximális értékű folyamot egy adott hálózatban. A maximális értékű folyamot egyszerűen **maximális folyamnak** nevezzük.

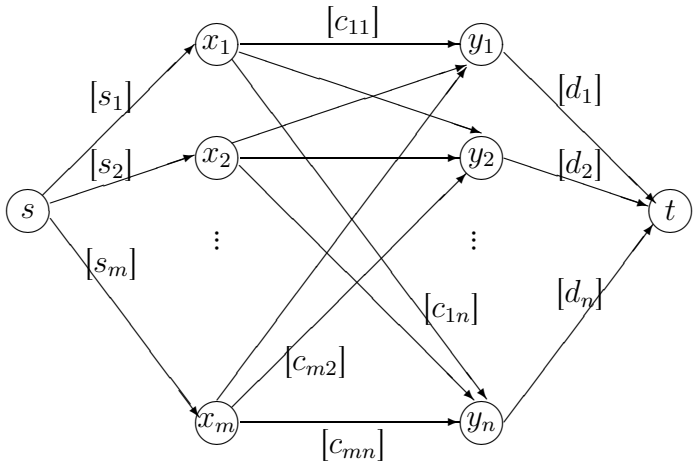
Példák.

1) *Tengeri szállítás*

Adott m kikötő, $x_1, x_2, \ldots x_m$, amelyekben bizonyos áru rendre s_1, s_2, $\ldots s_m$ mennyiségben fordul elő, és n kikötő, $y_1, y_2, \ldots, y_n$, ahol az illető árúból rendre $d_1, d_2, \ldots, d_n$ mennyiséget igényelnek. A feladat az, hogyan lehet megoldani minél több áru elszállítását, ha tudjuk, hogy egy adott x_i kikötőből egy y_j kikötőbe adott időintervallumban c_{ij} mennyiség szállítható (a hajó kapacitása.

A feladathoz rendeljünk hozzá egy irányított gráfot, amelynek $x_1, x_2, \ldots x_m$ és $y_1, y_2, \ldots, y_n$ a csúcsai, irányított él van minden x_i-ből minden y_j-be. Majd adjunk hozzá két új csúcsot, s-t és t-t. Húzzunk élt s-ből minden x_i-be és minden y_j-ből t-be. Minden élhez remndeljünk bizonyos kapacitást: az (s, x_i) élek kapacítása s_i, az (y_i, t) élek kapacítása d_i, az (x_i, y_j) éleké pedig c_{ij}.

A feladat: keressünk maximális folyamot az így értelmezett hálózatban!

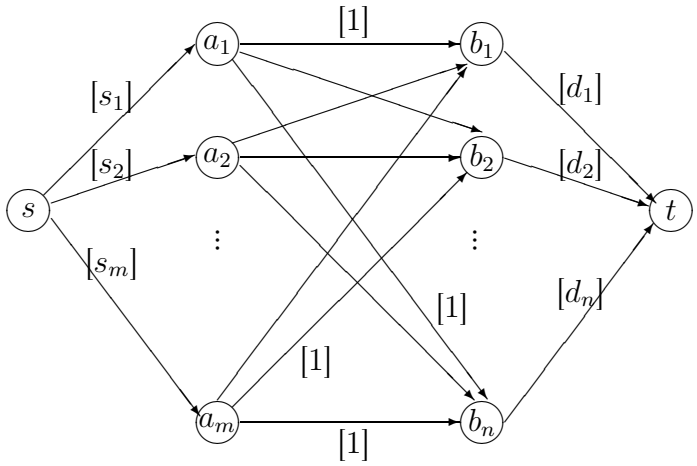


2) *Családi kirándulás*

Az $a_1, a_2, \ldots, a_m$, amelyek családok rendre $s_1, s_2, \ldots, s_m$ tagúak, kirándulni szeretnének autóbuszokkal. Összesen n busz áll rendelkezésükre, ezek $b_1, b_2, \ldots, b_n$, amelyek rendre a $d_1, d_2, \ldots, d_n$ személyt szállíthatnak. A következő feladatot kell megoldanunk: lehetsége-e úgy megszervezni a kirándulást, hogy a családtagok mind különböző buszokba kerüljenek?

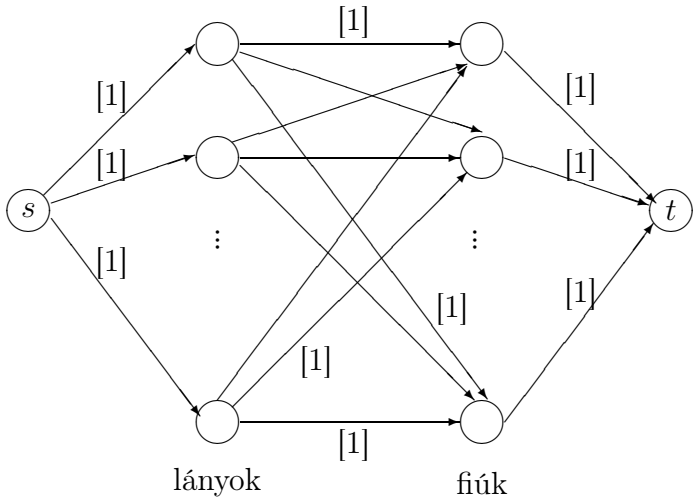
A feladathoz rendelt gráf hasonló az előző feladat gráfjához. A gráf csúcsai $a_1, a_2, \ldots, a_m$ és $b_1, b_2, \ldots, b_n$. Minden a_i csúcsból van irányított él minden b_j csúcsba, méghozzá 1 kapacitással. Addjunk hozzá még két új csúcsot, az s-t és t-t úgy, hogy legyen irányított él s-ből minden a_i csúcsba s_i kapacitással, és minden b_j-ből t-be d_j kapacitással. Ebben a hálózatban

keresünk maximális folyamot. Feldatunknak akkor van megoldása, ha létezik olyan folyam, amely telíti az s-ből kifutó éleket.



3) *Kollégiumi táncmulatság*

Meg lehet-e szervezni egy kollégiumban egy olyan táncot, hogy minden lány olyan fiúval táncoljon, akit ismer? A feladathoz rendelt gráf hasonló az előbbiekkez, ebben van irányított él minden lánytól azokhoz a fiúkhoz, akiket ismer. Van irányított él s-ből minden lányhoz, és minden fiútól t-be. Minden élen a kapacitás értéke 1. A feladatnak akkor van megoldása, ha létezik olyan maximális folyam, amely telíti az s-ből kifutó éleket.



A vágat

Ha egy hálózatban van a csúcsoknak egy olyan $A \subseteq V$, $\overline{A} = V \setminus A$ partíciója, hogy $s \in A$ és $t \in \overline{A}$, akkor ezt $(A, \overline{A})$ *vágatnak* nevezzük. Az $(A, \overline{A})$ vágat *kapacitása* egyenlő az éleihez rendelt kapacitások összegével:

$$\alpha(A, \overline{A}) = \sum_{\substack{x \in A \\ y \in \overline{A}}} \alpha(x, y).$$

A legkisebb kapacitású vágatot egy adott hálózatban *minimális vágatnak* nevezzük.

28. tétel. *Ha $(A, \overline{A})$ vágat egy hálózatban, akkor tetszóleges f folyamra*

$$v(f) = f(A, \overline{A}) - f(\overline{A}, A) \leq \alpha(A, \overline{A}).$$

Bizonyítás. Igazak a következők:
$f(s, V) - f(V, s) = v(f)$,
$f(x, V) - f(V, x) = 0$ for $x \neq s, t$.
Adjuk össze ezeket az egyenlőségeket minden $x \in A$ csúcsra:

$$v(f) = \sum_{x \in A} \big(f(x, V) - f(V, x)\big) = f(A, V) - f(V, A),$$

de $V = A \cup \overline{A}$ és $A \cap \overline{A} = \emptyset$, és ekkor:

$$v(f) = f(A, V) - f(V, A) = f(A, A \cup \overline{A}) - f(A \cup \overline{A}, A)$$

$$= f(A, A) + f(A, \overline{A}) - f(A, A) - f(\overline{A}, A) = f(A, \overline{A}) - f(\overline{A}, A) \leq \alpha(A, \overline{A}),$$

mivel $f(\overline{A}, A) \geq 0$ mindig. □

29. tétel (Ford–Fulkerson). *Egy hálózatban egy maximális folyam értéke egyenlő a minimális vágatkapacitással.*

Bizonyítás. Az előző tétel alapján elegendő bizonyítani, hogy létezik egy f maximális folyam és egy $(A, \overline{A})$ vágat, amelyekre:
$f(A, \overline{A}) = \alpha(A, \overline{A})$ és
$f(\overline{A}, A) = 0$.
Építsük fel rekurzívan az A halmazt!
1) Először legyen s az A egyetlen eleme.
2) Ha létezik egy olyan (x, y) irányított él, amelyre fennáll, hogy $x \in A$, $y \notin A$ és $f(x, y) < \alpha(x, y)$, akkor tegyük be y-t az A-ba.

3) Ha létezik egy olyan (y, x) irányított él, amelyre fennáll, hogy $x \in A$, $y \notin A$ és $f(y, x) > 0$, akkot tegyük be y-t az A-ba.

Azt állítjuk, hogy amikor már nem tudjuk folytatni a fenti lépések egyikét sem, akkor $t \in \overline{A}$. Ha nem így lenne, akkor létezne egy $x_1, x_2, \ldots, x_r$ csúcsok sorozata úgy, hogy $x_1 = s$, $x_r = t$ és minden $i = 1, 2, \ldots r-1$ értékre

$(x_i, x_{i+1}) \in E$ és $f(x_i, x_{i+1} < \alpha(x_i, x_{i+1})$, vagy
$(x_{i+1}, x_i) \in E$ és $f(x_{i+1}, x_i) > 0$.

Használjuk a következő jelöléseket:

$$\varepsilon_1 = \min_{\forall i:\ (x_i, x_{i+1}) \in E} \big(\alpha(x_i, x_{i+1}) - f(x_i, x_{i+1})\big)$$

$$\varepsilon_2 = \min_{\forall i:\ ;\ (x_{i+1}, x_i) \in E} f(x_{i+1}, x_i).$$

És legyen $\varepsilon = \min\{\varepsilon_1, \varepsilon_2\}$. Most értelmezhetjük a következő f^* folyamot:

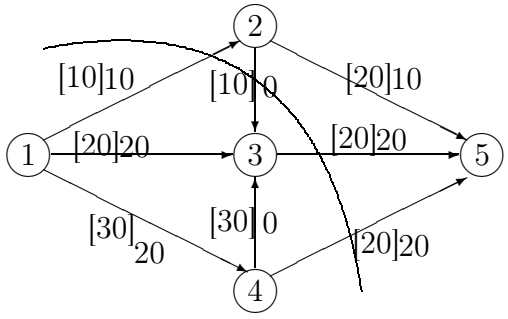
1) $f^*(x_i, x_{i+1}) = f(x_i, x_{i+1}) + \varepsilon$ ha $(x_i, x_{i+1}) \in E$
2) $f^*(x_{i+1}, x_i) = f(x_{i+1}, x_i) - \varepsilon$ ha $(x_{i+1}, x_i) \in E$
3) $f^*(x, y) = f(x, y)$ minden olyan élre, amelynek végpontjai nincsenek az $x_1, x_2, \ldots, x_r$ sorozatban.

Könnyű ellenőrizni, hogy ez a függvény ténylegesen folyam, méghozzá a $v(f^*) = v(f) + \varepsilon$ értékkel, ami ellentmondás, hisz feltételezésünk szerint f maximális volt.

Tehát, az A felépítése alapján megállapíthatjuk, hogy

$\forall (x, y) \in (A, \overline{A}):$ $\quad f(x, y) = \alpha(x, y)$
$\forall (y, x) \in (\overline{A}, A):$ $\quad f(y, x) = 0$, és ezzel bebizonyítottuk tételünket.

Példa.



A fenti példában a kapacitásokat szögletes zárójelbe írtuk, a folyam értékét egy adott élen pedig zárójel nélkül. A folyam értéke 50, és ez a

folyam maximális, hisz létezik egy $A = \{1, 3, 4\}$ és $\overline{A} = \{2, 5\}$ halmazokból álló vágat, amelynek a kapacitása szintén 50. Minden A-ból $\overline{A}$-ba mutató $(1, 2), (3, 5), (4, 5)$ irányított él telített, és az egyetlen $\overline{A}$-ból A-ba mutató $(2, 3)$ irányított élen a folyam 0 értékű. □

Ezt a tételt szokás még *maximális folyam–minimális kapacitás-tételnek* is nevezni.

8.2. A Ford–Fulkerson-algoritmus

A 29. tétel alapján tervezhetünk egy algoritmust a maximális folyam megkeresésére. Az algoritmus lépésenként címkézi a csúcsokat. Egy y csúcs címkéje $(x^+, \Delta y)$ vagy $(x^-, \Delta y)$, ahol x^+ azt jelenti, hogy az (x, y) élen a folyam Δy értékkel növelhető, x^- pedig azt, hogy az (y, x) élen csökkenthető Δy értékkel.

Kezdetben úgy tekintjük, hogy a folyam értéke minden élen 0.

Címkézzük meg az s csúcsot a $(-, \infty)$ címkével.

Ha $(x, y) \in E$, x meg van címkézve, y pedig nincs, és $f(x, y) < \alpha(x, y)$, akkor címkézzük meg y-t az $(x^+, \Delta y)$ címkével, ahol $\Delta y = \min\big(\Delta x, \alpha(x, y) - f(x, y)\big)$.

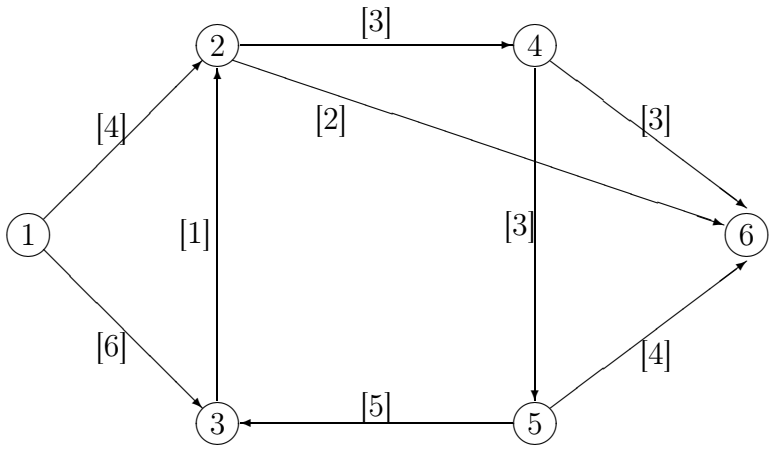
Ha $(y, x) \in E$, x meg van címkézve, y pedig nincs, és $f(y, x) > 0$, akkor címkézzük meg az y csúcsot $(x^-, \Delta y)$-vel, ahol $\Delta y = \min\big(\Delta x, f(y, x)\big)$.

Ha meg tudjuk címkézni a t csúcsot, akkor van egy lánc s és t között, amelyen módosítható a folyam Δt értékkel. A címke első eleme megmutatja a csúcsot, amelyik az él másik végpontja, a $+$ vagy $-$ jel pedig azt, hogy a folyam értékét ezen az élen növeljük vagy csökkentjük. Folytatjuk a következő csúccsal, és így tovább. Módosítás után folytatjuk újabb címkézéssel.

Ha nem tudjuk megcímkézni t-t, akkor a folyam maximális, a címkézett csúcsok a megfelelő $(A, \overline{A})$ minimális vágat A halmazát, a címkézetlenek pedig az $\overline{A}$ halmazát képezik.

Azt a láncot, amelyen módosítani (javítani) lehet a folyamot, *javítóláncnak (javítóútnak)* nevezzük.

Példa.



A címkézés folyamata

Kezdetben a folyam minden élen 0. A következő címkézést végezzük:

csúcs	1	2	4	6
címke	$(-,\infty)$	$(1^+,4)$	$(2^+,3)$	$(4^+,4)$

A folyam a következőképpen módosul: $f(4,6)=3$, $f(2,4)=3$, $f(1,2)=3$. Miután töröljük a címkéket, újrakezdjük.

csúcs	1	2	6
címke	$(-,\infty)$	$(1^+,1)$	$(2^+,1)$

A folyam a következőképpen módosul: $f(2,6)=1$, $f(1,2)=4$. Miután ismét töröljük a címkéket, újrakezdjük.

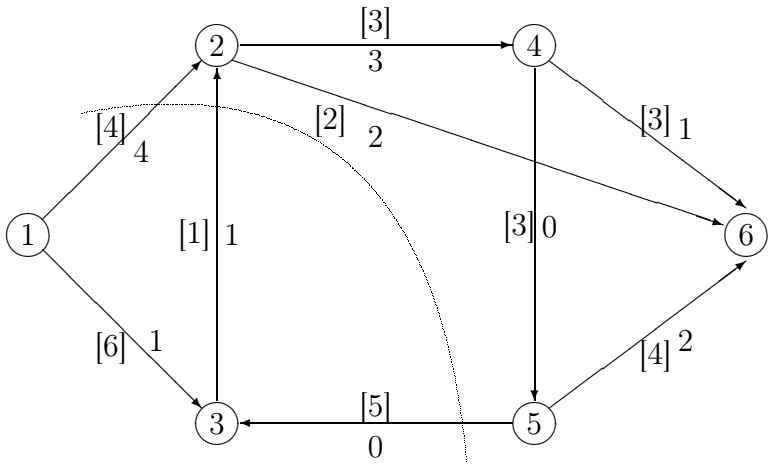
csúcs	1	3	2	6
címke	$(-,\infty)$	$(1^+,6)$	$(3^+,1)$	$(2^+,1)$

A folyam a következőképpen módosul: $f(2,6)=2$, $f(3,2)=1$, $f(1,3)=1$. Folytatjuk, újabb címketörlések után.

csúcs	1	3
címke	$(-,\infty)$	$(1^+,5)$

Mivel nem tudjuk t-t megcímkézni, a folyam maximális. A minimális vágat: $A=\{1,3\}$, $\overline{A}=\{2,4,5,6\}$. A folyam értéke 5.

Az eredmény:

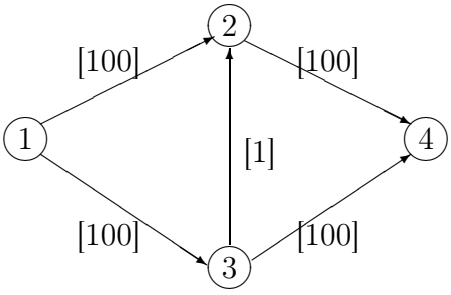


FORD–FULKERSON()
1.
2.

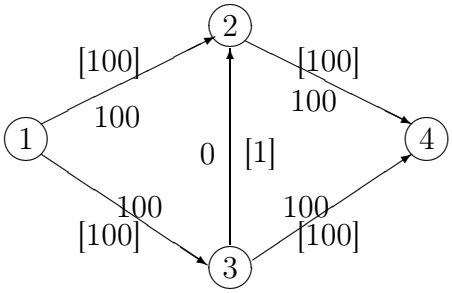
A Ford–Fulkerson-algoritmus akkor is működik, ha a kapacitások racionális számok, de nem alkalmazható, ha a kapacitások valós számok. (Ennek bizonyítását *lásd* [2]-ben.)

8.3. A Ford–Fulkerson-algoritmus elemzése

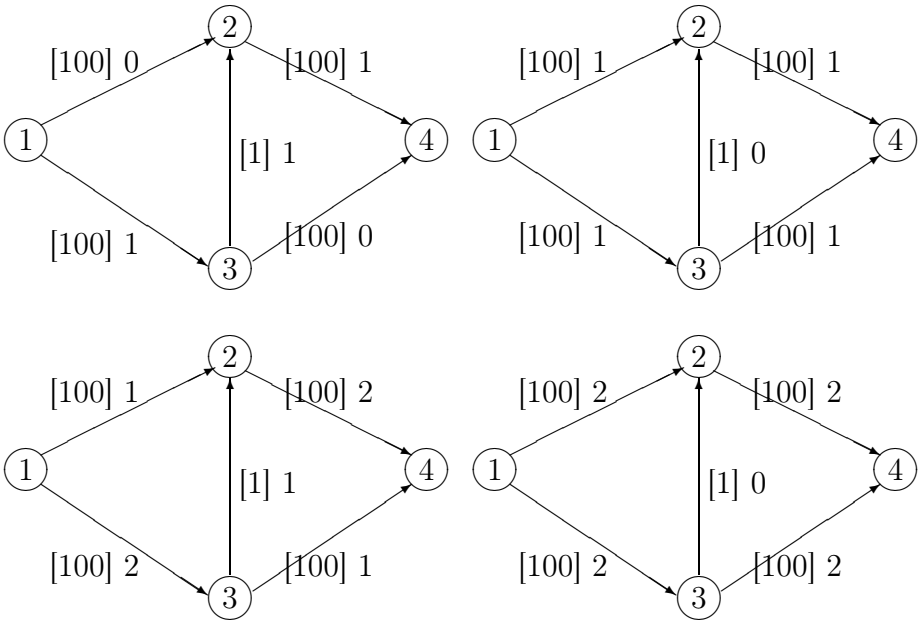
Tekintsük a következő példát:



A Ford–Fulkerson-algoritmust alkalmazva a megoldást két lépésben megkaphatjuk. Nulla értékű folyammal indulva, az $1, 2, 4$ úton növelni lehet a folyamot 100-zal, aztán hasonlóképpen az $1, 3, 4$ úton is. Tehát a maximális folyam értéke 200.



De az algoritmus ugyanúgy választhatja először az $1, 3, 2, 4$ utat, amelyen a folyam 1-gyel növelhető. Ezután, a $1, 2, 3, 4$ láncot használva, a folyamot szintén 1-gyel lehet növelni.



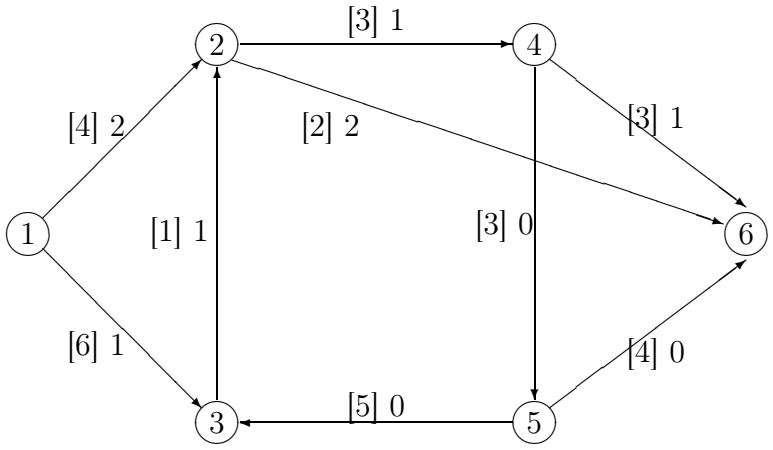
Így folytatva, az eredményt 200 lépésben kapjuk meg. Az algoritmus bonyolultsága ily módon függ a folyam értékétől. Ha $m = |E(G)|$, a hálózat éleinek száma, $v(f)$ pedig a folyam értéke, akkor az algoritmus bonyolultsága $O(mv(f))$ vagy $O(n^2v(f))$, ha n a hálózat csúcsainak a száma (pszeudopolinomiális algoritmus).

Az algoritmus javítható, ha minden alkalommal a lehetséges láncok közül a legrövidebbet (legkevesebb élből állót) választjuk. Ha a javítóláncot szélességi kereséssel határozzuk meg, akkor mindig a legrövidebbet kapjuk. A Ford–Fulkerson-algoritmusnak ezt a változatát *Edmonds–Karp-algoritmusnak* nevezzük. Ennek a bonyolultsága $O(nm^2)$ vagy $O(n^5)$.

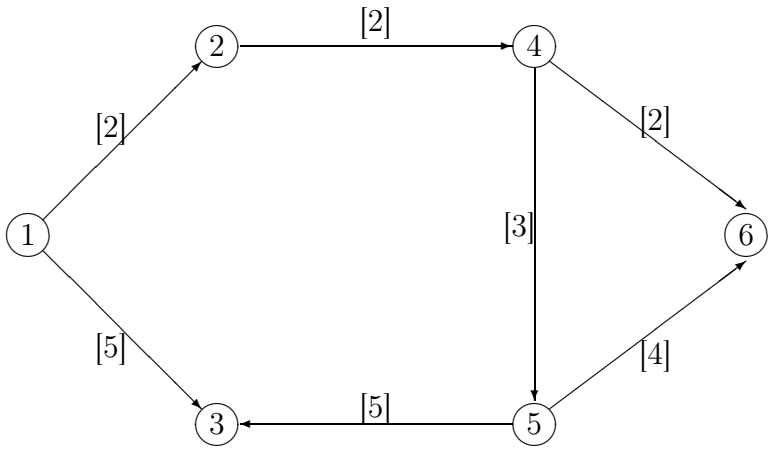
A reziduális hálózat

Legyen $H = (V, E, \alpha, s, t)$ egy hálózat, f pedig egy folyam ebben a hálózatban, és rendeljük hozzá az $H_f = (V, E', \alpha', s, t)$ *reziduális hálózatot*, ahol $\alpha' = \alpha - f$, és E'-et E-ből úgy kapjuk, hogy elhagyjuk a 0 kapacitású éleket.

A következő hálózat és folyam:



reziduális hálózata:



Ha f egy hálózati folyam, f' a hozzárendelt reziduális hálózat folyama, akkor $f + f'$ az eredeti hálózat folyama, amelynek értéke $v(f) + v(f')$.

A reziduális hálózat segítségével be lehet bizonyítani az Edmonds–Karp-algoritmus helyességét.

8.4. Általánosított folyam

A hálózatot úgy lehet általánosítani, hogy a kapacitás mellett megadunk egy alsó határt is minden élen, és folyam a két érték közé kell, hogy essen.

Legyen $H^* = (V, E, \beta, \alpha, s, t)$ általánosított hálózat, ahol V, E, s, t elentése ugyanaz mint eddig. A β és α föggvények $V \times V$-n értelmezettek és nem negatív egész értékűek. Ezek alsó és felső kapacitásfüggvények: $\beta \leq \alpha$ és

1) $\alpha(x, y) > 0$ ha $(x, y) \in E$

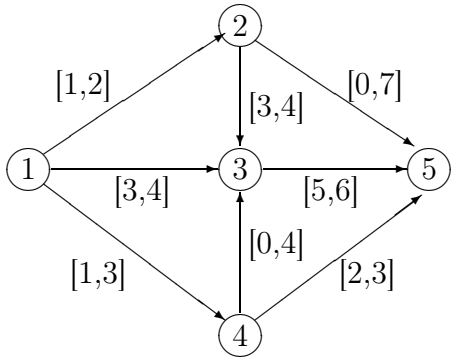
2) $\alpha(x, y) = 0$ ha $(x, y) \notin E$.

Az *általánosított folyam* egy általánosított hálózatban az $f : V \times V \to \mathbf{Z}_+$ függvény, amely a következő tulajdonságokkal rendelkezik:

1) $\beta(x, y \leq f(x, y) \leq \alpha(x, y)$ ha $x, y \in V$

2) $f(V, x) = f(x, V)$ ha $x \in V \setminus \{s, t\}$.

Az általánosított folyam értéke $v(f) = f(s, V) = f(V, t)$. Célunk, hogy maximális értékű általánosított folyamot keressünk, ha ilyen létezik. A klasszikus folyamfeladatnál mindig létezett maximális folyam, hisz a nulla értékű folyam létezése biztosított. Itt azonban meg kell vizsgálnunk, milyen feltétel mellett létezik általánosított folyam. A következő példa esetében nem létezik általánosított folyam, hisz a 2 csúcsba maximálisan 2 érték futhat be, és minimálisan 3-nak kell kimennie (3+0).



Hogy válaszoljunk arra a kérdésre, hogy mikor létezik általánosított folyam, visszavezetjük a feladatot egy klasszikus folyamfeladatra. Rendeljünk hozzá az $H = (V, E, \alpha, s, t)$ általánosított hálózathoz egy $H^* = (V^*, E^*, \alpha^*, a, z)$ klasszikus hálózatot a következőképpen:

$$V^* := V \cup \{a, z\}$$

$$E^* := E \cup \big\{(a,x) \mid N_G^{\mathrm{be}}(x) \neq \emptyset\big\} \cup \big\{(x,z) \mid N_G^{\mathrm{ki}}(x) \neq \emptyset\big\} \cup \big\{(t,s)\big\}$$

$$\alpha^*(x,y) := \alpha(x,y) - \beta(x,y), \quad \forall x,y \in V$$

$$\alpha^*(a,x) := \beta(V,x)$$

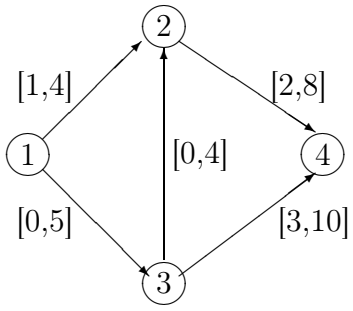
$$\alpha^*(x,z) := \beta(x,V)$$

$$\alpha^*(t,s) := \infty$$

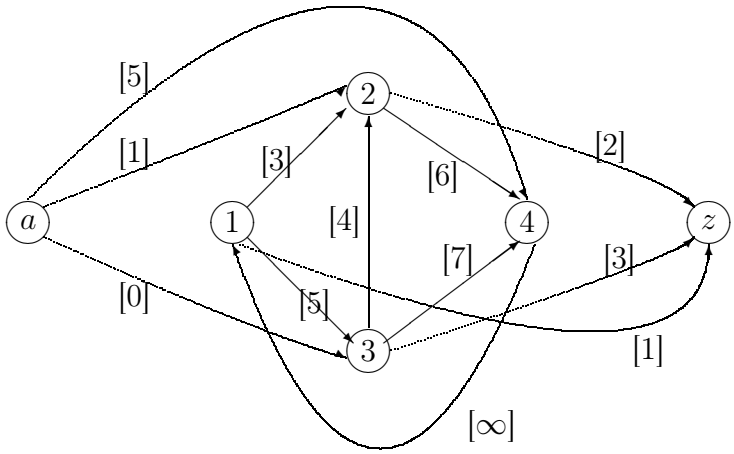
Emlékeztetünk arra, hogy

$$\beta(V,x) = \sum_{y \in N_G^{\mathrm{be}}(x)} \beta(y,x), \qquad \beta(x,V) = \sum_{y \in N_G^{\mathrm{ki}}(x)} \beta(x,y)$$

Egy példa általánosított hálózatra:



A hozzárendelt klasszikus hálózat:



A következő tétel választ ad a feltett kérdére, hogy mikor létezik általánosított folyam.

30. tétel. *A H általánosított hálózatban akkor és csakis akkor létezik általánosított folyam, ha a H^* hozzárendelt hálózatban létezik olyan maximális folyam, amely telíti az a csúcsból kiinduló éleket.*

Bizonyítás. I. Legyen f^* olyan folyam H^*-ban, amelyik telíti az a csúcsból kiinduló éleket. De, mivel

$$\alpha^*(a, V^*) = \beta(V, V) = \alpha(V^*, z),$$

ez a folyam telíteni fogja a z csúcsba befutó éleket is.

Definiáljuk az $f := f^* + \beta$ függvényt, és bebizonyítjuk, hogy ez általánosított folyam az eredeti hálózatban.

1. *Kapacitás-feltétel.*

A

$$0 \leq f^*((x, y) \leq \alpha^*(x, y) = \alpha(x, y) - \beta(x, y) \quad \text{ha } x, y \in V,$$

összefüggésből következik:

$$\beta(x, y) \leq f(x, y) \leq \alpha(x, y) \quad \text{ha } x, y \in V$$

2. *Egyensúly-feltétel.*

Ha $x \in V \setminus \{s,t\}$, akkor

$$f(V,x) - f(x,V) = f^*(V,x) + \beta(V,x) - f^*(x,V) - \beta(x,V)$$

De

$$\beta(V,x) = \alpha^*(a,x) = f^*(a,x) \quad \text{ha } x \in V \setminus \{s,t\} \quad \text{(telítés)}$$

$$\beta(x,V) = \alpha^*(x,z) = f^*(x,z) \quad \text{ha } x \in V \setminus \{s,t\} \quad \text{(telítés)}$$

és mivel $V^* = V \cup \{a,z\}$, következik

$$f^*(V,x) + \beta(V,x) = f^*(V,x) + f^*(a,x) = f^*(V^*,x)$$

$$f^*(x,V) + \beta(x,V) = f^*(x,V) + f^*(x,z) = f^*(x,V^*)$$

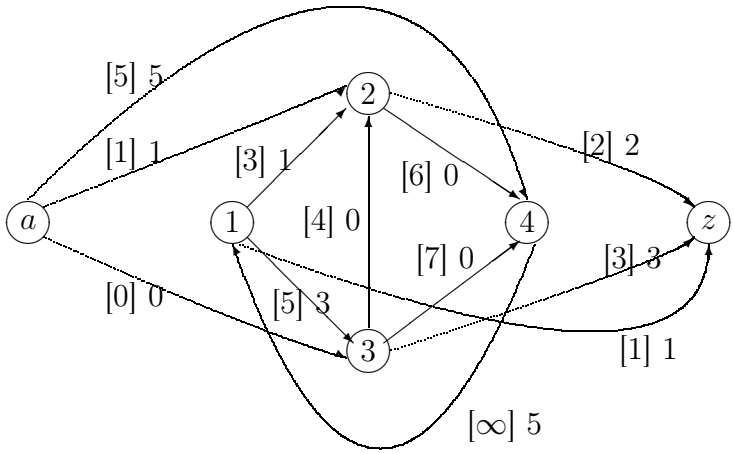
Mivel f^* folyam a H^* hálózatban, azt kapjuk, hogy:

$$f(V,x) - f(x,V) = f^*(V^*,x) - f^*(x,V^*) = 0 \quad \text{ha } x \in V \setminus \{s,t\}.$$

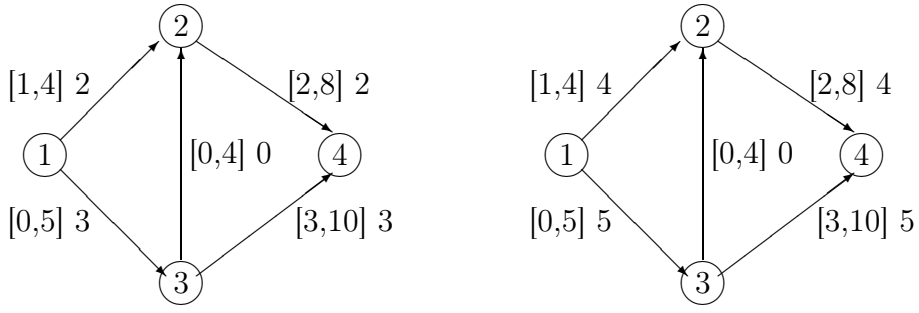
Tehát f általánosított folyam az eredeti hálózatban, tehát létezik általánosított folyam.

II. Fordítva, ha f létezik, akkor hasonló módon be lehet bizonyítani, hogy f^* olyan folyam, amely telíti az a csúcsból kifutó éleket. □

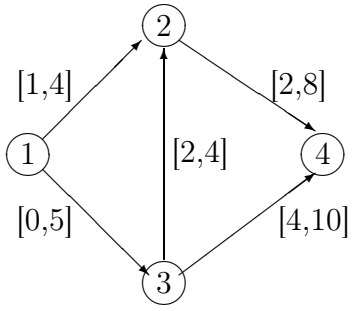
Egy maximális folyam H^*-ban:



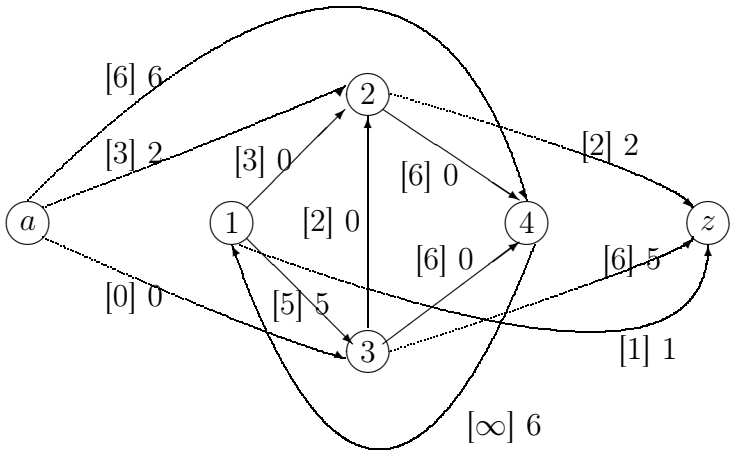
This flow saturates the arcs from a, so the function $f := f^* - \beta$ computed on the arcs of the network G will be a generalized flow (figure in the left). The maximum generalized flow, obtained from this, after applying the Ford-Fulkerson method, if we do not consider the lower capacity, will be the one in the right figure.



Példa általánosított hálózatra, amelyben nincs általánosított folyam.



A hozzárendelt hálózat és maximális folyam:



A folyam maximális, amit az $A = \{a, 1, 2, 4\}$, $\overline{A} = \{3, z\}$ vágat bizonyít. Mivel ez a folyam nem telíti a z-be befitó éléket (a $(3, z)$ él nem telített), ezért nem létezik általánosított folyam.

8.5. Minimális költségű maximális folyam

Let us consider a flow network $G = (V, E, \alpha, s, t)$ and a cost function $c : V \times V \to \mathbf{R}_+$ (where $\mathbf{R}_+$ is the set of nonnegative real numbers). If f

is a flow in G then the *cost of the flow* f is defined as:

$$\mathcal{C}(f) := \sum_{x,y \in V} c(x,y) f(x,y)$$

Usually $c(x,y) = 0$ for $(x,y) \notin E$, but by our definition this is not important (because in this case the flow is equal to 0).

We are interesting in minimum cost maximum flow. How can we know if a given maximum flow is or not of minimum cost? We solve this problem by attaching to it a weighted digraph in which the absence of a negative length cycle will correspond to a minimum cost maximum flow.

For a network $G = (V, E, \alpha, s, t)$ a cost function c and a maximum flow f let us define the following weighted digraph $\overline{A}G$ with the same vertex set as in G:

• If on the arc (x,y) in G the flow $f(x,y) = 0$, then in $\overline{A}G$ put an arc (x,y) with the weight

$$\mathcal{W}(x,y) := c(x,y)$$

• If on the arc (x,y) in G the flow $f(x,y) = \alpha(x,y)$, then in $\overline{A}G$ put an arc (y,x) with the weight

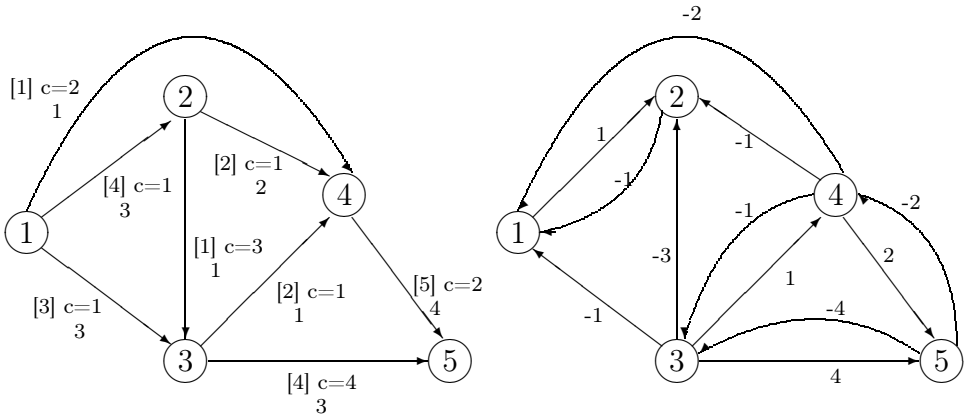
$$\mathcal{W}(y,x) := -c(x,y)$$

• If on the arc (x,y) in G the flow $0 < f(x,y) < \alpha(x,y)$, then in $\overline{A}G$ put the both arcs (x,y) and (y,x) with the weights:

$$\mathcal{W}(x,y) := c(x,y) \qquad \text{and}$$

$$\mathcal{W}(y,x) := -c(x,y).$$

An example:



The cost of this flow is $\mathcal{C}(f) = 3{\cdot}1+3{\cdot}1+2{\cdot}1+1{\cdot}3+1{\cdot}2+1{\cdot}1+3{\cdot}4+4{\cdot}2 = 34$.

31. tétel (Busacker–Saaty). *A maximum flow in the network G is of minimum cost if and only if the corresponding weighted graph $\overline{A}G$ has no negative length directed cycle.*

Bizonyítás. I. Let us prove that if f is a minimum cost flow in G, then in $\overline{A}G$ there exists no negative length cycles. This assertion is equivalent to the following: *if in $\overline{A}G$ there exists a negative length cycle, then the corresponding flow f in G in not of minimum cost.*

Let K be a negative length cycle in $\overline{A}G$, and let us define the following function (on arcs only):

$\overline{A}f(x,y) := f(x,y)+1$ if (x,y) is on the cycle K and $\mathcal{W}(x,y) > 0$,
$\overline{A}f(x,y) := f(x,y)-1$ if (y,x) is on the cycle K and $\mathcal{W}(y,x) < 0$,
$\overline{A}f(x,y) := f(x,y)$ otherwise

Can be proved that this function $\overline{A}f$ is a flow in G too. Let us use the following notation:

$$f_c(A) = \sum_{x,y)\in A} c(x,y)f(x,y) \qquad \text{where } A \subseteq E$$

Can be proved the following:

- if $A \cap B = \emptyset$ then $f_c(A \cup B) = f_c(A) + f_c(B)$
- if $\overline{A}f = f + 1$ then $\overline{A}f_c(A) = f_c(A) + c(A)$ where $c(A) := \sum_{(x,y)\in A} c(x,y)$

For simplicity let us denote by K too the set of arcs of the cycle K. Then $E = (E \setminus K) \cup K$ and $(E \setminus K) \cap K = \emptyset$. Then

$$\overline{A}f_c(E) = \overline{A}f_c(E \setminus K) + \overline{A}f_c(K) = f_c(E \setminus K) + f_c(K) + c(K)$$

$= f_c(E)+c(K) < f_c(E)$ becuase $c(K) < 0$ is the length of the cycle K

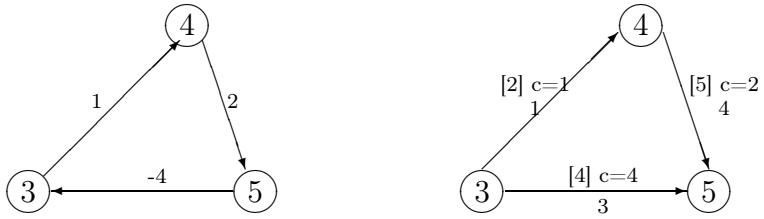
which is contradiction with the minimality of f.

II. Conversely, can be proved, in the same way, the following: if f is no a minimum flow, then in $\overline{A}G$ there exists a negative length cycle.

□

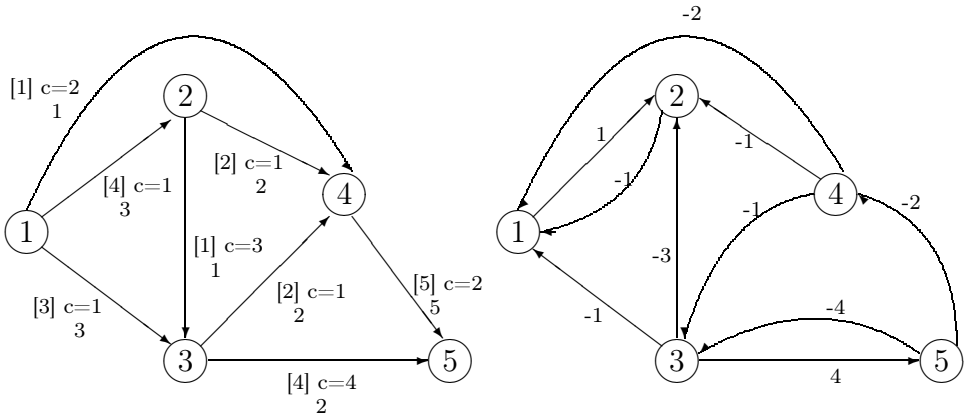
In our example a directed cycle with length -1 exists: 3,4,5,3 wich

correspond to two paths in the original network (3,4,5 and 3,5).



This means that the cost of the flow can be descreased by 1, if we increase the value of the flow on the path 3,4,5 by 1, and descrease on the path 3,5 (here only one edge) by 1.

The new flow and the corresponding weighted graph are the following:



The cost of this flow is 33 (the modifications are: $+1 \cdot 1 + 1 \cdot 2 - 1 \cdot 4$). In this new graph there is no directed cycle with negative length.

To find the negative length cycles, we can use the Floyd–Warshall algorithm. This algorithm doesn't give us the distance between vertices if there are negative weights in graph, but we can check if there are or not negative length cycles. If the result of this algorithm is the matrix D and there exists a $d_{ii} < 0$ for some i, then in the graph there is at least a negative length cycles.

For our above example the adjacency matrix is:

$$D_0 = \begin{pmatrix} 0 & 1 & \infty & \infty & \infty \\ -1 & 0 & \infty & \infty & \infty \\ -1 & -3 & 0 & 1 & 4 \\ -2 & -1 & -1 & 0 & 2 \\ \infty & \infty & -4 & -2 & 0 \end{pmatrix}$$

and the result of the Floyd-Warshall algorithm is

$$D = \begin{pmatrix} 0 & 1 & \infty & \infty & \infty \\ -1 & 0 & \infty & \infty & \infty \\ -5 & -4 & -1 & 0 & 2 \\ -6 & -5 & -2 & -1 & 1 \\ -9 & -8 & -5 & -4 & -2 \end{pmatrix}$$

Because of the negative values on the main diagonal, in graph there exists a cycle with negative length.

But we can modify the Floyd–Warshall algorithm to find the shortest paths even in the case of negative weights and negative length cycles. When a negative value on the main diagonal appears we will take it 0. In this case the negative length cycles does not influence any more the length of the paths.

```
D := D_0
cycle := 0
for k := 1 to n do
    for i := 1 to n do
        for j := 1 to n do
            if d_ij > d_ik + d_kj then
               d_ij := d_ik + d_kj
               if d_ij < 0 and i = j then d_ij := 0
                                          cycle := i
               endif
            endif
        endfor
    endfor
endfor
```

For our above example:

$$D = \begin{pmatrix} 0 & 1 & \infty & \infty & \infty \\ -1 & 0 & \infty & \infty & \infty \\ -5 & -4 & 0 & 0 & 3 \\ -6 & -5 & -2 & 0 & 2 \\ -8 & -7 & -4 & -3 & 0 \end{pmatrix}$$

If at the end of the algorithm $cycle > 0$, then the graph contains negative length cycles, and vertex v_{cycle} is on that cycle.

To find the negative length cycle let use the following algorithm (in which $d0_{ij}$ is the general element of D_0):

$i := cycle$
$j := \min\limits_{1 \le s \le n} (d0_{is} + d_{si})$ if $j \neq i$. The arc (v_i, v_j) is on the cycle.
while $j \neq i$ **do**
  $k := \min\limits_{1 \le s \le n} (d0_{js} + d_{si})$. The arc (v_j, v_k) is on the cycle.
  $j := k$
endwhile

Resuming the flow problems

– flow networks: maximum flow

methods:	Ford–Fulkerson	$O(n^2 v(f))$
	Edmonds–Karp	$O(n^5)$
	preflow	$O(n^4)$

– minimum cost maximum flow

– generalized flow networks: generalized (or compatible) flow
maximum generalized flow
minimum generalized flow

9. fejezet

Páros gráfok – optimális hatásfokú foglalkoztatás

Hát én immár kit válasszak,
virágom, virágom?
(Tavaszi szél – magyar népdal)

10. fejezet

Szélsőérték feladatok – extrémgráfok

Az egész természet a legkisebb megrezzenést is megérzi: s a tenger egyetlen kavics bedobásától megváltozik. Így van ez a kegyelemben is: a legkisebb tett is az egészre hat ki. Így tehát minden fontos.

(Pascal: Gondolatok, ford. Hamvas Béla)

11. fejezet

A gráfelmélet történetéből

Rakjuk le, hangyaszorgalommal, amit
Agyunk az ihlett órákban teremt
S ha összehordtunk minden kis követ,
Építsük egy újabb kor Bábelét,
Míg oly magas lesz, mint a csillagok.

(Vörösmarty: Gondolatok a könyvtárban)

12. fejezet

A gráfelmélet himnusza

A szám és a zene a legszorosabban kapcsolódik.
(Novalis: Hang és tánc, ford. Kemény Ildikó
in Hamvas Béla: Anthologia humana)

Az ismert cseh matematikus, gráfelmélész, Bohdan Zelinka versét több nyelvre is lefordították. A magyar fordítást kottával együtt közöljük.

A gráfelmélet himnusza

Szöveg: *Bohdan Zelinka*
Zene: *Zdenek Ryjaček*

Ádám András fordítása

Állott hét híd a Pregel folyóján,
akkortájt ez nem csekélység volt ám;
Königsbergben büszke sok tanácsos,
ennyi híddal hogy ékes a város.

Alkonyatkor kavarog a népség,
és fejükben hánytorog a kétség:
hogy’ lehetne jó utat találni,
minden hídon egyszer általjárni.

Mind a hét híd egyszer essen útba,
séta végén otthon lenni újra;
de a jó út valahol hibázik,
egy híd mindig fölös vagy hiányzik.

Refrén:
Euleri gráf: minden foka páros,

és a tétel mindörökre áll most;
gráfokról ez állítás
a világnak ősforrás.

Él egy ember, gondoljunk csak rája,
itt minálunk, nincs tudásban párja;
úgy érti a számolást és mérést,
hogy elébe kell tárni a kérdést.

Euler mester fejét búsan rázza:
„Oly talány ez, nincsen megoldása;
nincs oly út, mint uraságtok kérik,
amely minden hidat egyszer érint.

Refrén:
Euleri gráf: . . .

Érckemény a tudományos tétel,
mit sem kezdhet ellene a kétely;
árad a víz, szilárd a híd rajta,
még erősb a tudomány hatalma."

Háború jött a Pregel folyóra,
minden hídját ízzé-porrá szórta;
nemzedékek hosszú során fénylik
Euler és a folyó neve végig.

Refrén:
Euleri gráf:. . .

Euler híre nem ér addig véget,
míg csak élni fog a gráfelmélet;
s egyik évre amint jön a másik,
az elmélet mind jobban virágzik.

Jó kollégák, töltsük meg a kelyhet,
Áldomásra mind emeljük feljebb:
nekünk a gráfelmélet oly drága,
hadd teremjen sok-sok szép virága!

13. fejezet

Hogyan rajzoljunk gráfokat LaTeX-ben

Szójegyzékek

magyar–román–*angol*

be-fok – grad interior – *in-degree*
csúcs, szögpont – nod, vârf – *node, vertex*
duális gráf – graf dual – *dual graph*
egyszerű gráf – graf simplu – *graph*
él – muchie – *edge*
fa – arbore – *tree*
faváz, feszítőfa – arbore de acoperire, arbore parţial – *spanning tree*
feszítőfa, faváz – arbore de acoperire, arbore parţial – *spanning tree*
feszítő részgráf – graf parţial – *spanning subgraph*
fok – grad – *degree*
folyam – flux – *flow*
forrás – sursă – *source*
gráf nagysága – dimensiunea grafului – *size of graph*
gráf rendje – ordinul grafului – *order of graph*
hurok – buclă – *loop*
incidencia mátrix, illeszkedési mátrix – matrice de incidenţă – *incidence matrix*
irányított él – arc – *arc*
irányított gráf – graf orientat – *digraph, directed graph*
irányított kör – circuit elementar – *cycle in digraph, directed cycle*
irányított séta – drum – *walk in digraph*
irányított út – drum elementar – *directed path, path in digraph*
irányított vonal – drum simplu – *trail in digraph*
irányított zárt séta – circuit – *closed walk in digraph*
irányított zárt vonal – circuit simplu – *circuit in digraph, directed circuit*
kapacitás – capacitate – *capacity*
keresztezési szám – număr de încrucişare – *crossing number*

ki-fok – grad exterior – *out-degree*
kiegészítő gráf, komplementer gráf – graf complementar – *complement of a graph*
komplementer gráf, kiegészítő gráf – graf complementar – *complement of a graph*
komponens – componentă – *component*
kör – ciclu elementar – *cycle*
közlekedési hálózat – reţea de transport – *network*
kritikus út – drum critic – *critical path*
kromatikus szám – număr cromatic – *chromatic number*
lánc – lanţ în graf orientat – *semipath*
liget – pădure – *forest*
mélységi bejárás (keresés) – căutare în adâncime – *depth-first search*
mohó algoritmus – algoritm greedy – *greedy algorithm*
(multi)gráf – multigraf – *multigraph*
nyelő – puţ – *sink*
összefüggő gráf – graf conex – *connected graph*
páros gráf – graf bipartit – *bipartite graph*
párosítás – cuplaj – *matching*
reguláris gráf – graf regular – *regular graph*
részgráf – subgraf, graf parţial – *subgraph*
séta – lanţ – *walk*
síkgráf – graf planar – *planar graph*
szélességi bejárás (keresés) – căutare în lăţime – *breadth-first search*
szögpont, csúcs – vârf, nod – *vertex, node*
szomszédsági mátrix – matrice de adiacenţă – *adjacency matrix*
teljes gráf – graf complet – *complete graph*
út – lanţ elementar – *path*
vágat – tăietură – *cut*
vonal – lanţ simplu – *trail*
zárt séta – ciclu – *closed walk*
zárt vonal – ciclu simplu – *circuit*

román–magyar–*angol*

algoritm greedy – mohó algoritmus – *greedy algorithm*
arbore – fa – *tree*
arbore de acoperire, arbore parţial – faváz, feszítőfa – *spanning tree*
arc – irányított él – *arc*
buclă – hurok – *loop*
capacitate – kapacitás – *capacity*
căutare în adâncime – mélységi bejárás – *depth-first search*
căutare în lăţime – szélességi bejárás – *breadth-first search*
ciclu – zárt séta – *closed walk*
ciclu elementar – kör – *cycle*
ciclu simplu – zárt vonal – *circuit*
circuit – irányított zárt séta – *closed walk in digraph*
circuit elementar – irányított kör – *cycle in digraph, directed cycle*
circuit simplu – irányított zárt vonal – *circuit in digraph, directed circuit*
componentă – komponens – *component*
cuplaj – párosítás – *matching*
dimensiunea grafului – gráf nagysága – *size of graph*
drum – irányított séta – *walk in digraph*
drum critic – kritikus út – *critical path*
drum elementar – irányított út – *directed path, path in digraph*
drum simplu – irányított vonal – *trail in digraph*
flux – folyam – *flow*
grad – fok – *degree*
grad exterior – ki-fok – *out-degree*
grad interior – be-fok – *in-degree*
graf bipartit – páros gráf – *bipartite graph*
graf complementar – komplementer gráf, kiegészítő gráf – complement of a graph
graf complet – teljes gráf – *complete graph*
graf conex – összefüggő gráf – *connected graph*
graf dual – duális gráf – *dual graph*
graf orientat – irányított gráf – *digraph, directed graph*
graf parţial – feszítő részgráf – *spanning subgraph*
graf planar – síkgráf – *planar graph*
graf regular – reguláris gráf – *regular graph*
graf simplu – egyszerű gráf – *graph*
lanţ – séta – *walk*

lanţ elementar – út – *path*
lanţ în graf orientat – lánc – *semipath*
lanţ simplu – vonal – *trail*
matrice de adiacenţă – szomszédsági mátrix – *adjacency matrix*
matrice de incidenţă – illeszkedési (incidencia) mátrix – *incidence matrix*
muchie – él – *edge*
multigraf – (multi)gráf – *multigraph*
nod, vârf – csúcs, szögpont – *node, vertex*
număr cromatic – kromatikus szám – *chromatic number*
număr de încrucişare – keresztezési szám – *crossing number*
ordinul grafului – gráf rendje – *order of graph*
pădure – liget – *forest*
puţ – nyelő – *sink*
reţea de transport – nyelő – *network*
subgraf – részgráf – *subgraph*
sursă – forrás – *source*
tăietură – vágat – *cut*
vârf, nod – szögpont, csúcs – *vertex, node*

angol–magyar–*román*

adjacency matrix – szomszédsági mátrix – *matrice de adiacenţă*
arc – irányított él – *arc*
bipartite graph – páros gráf – *graf bipartit*
breadth-first search – szélességi bejárás – *căutare în lăţime*
capacity – kapacitás – *capacitate*
chromatic number – kromatikus szám – *număr cromatic*
circuit – zárt vonal – *ciclu simplu*
circuit in digraph, directed circuit – irányított zárt vonal – *circuit simplu*
closed walk – zárt séta – *ciclu*
closed walk in digraph – irányított zárt séta – *circuit*
complement of a graph – komplementer gráf, kiegészítő gráf – *graf complementar*
complete graph – teljes gráf – *graf complet*
component – komponens – *componentă*
connected graph – összefüggő gráf – *graf conex*
critical path – kritikus út – *drum critic*
crossing number – keresztezési szám – *număr de încrucişare*
cut – vágat – *tăietură*
cycle – kör – *ciclu elementar*
cycle in digraph, directed cycle – irányított kör – *circuit elementar*
degree – fok – *grad*
depth-first search – mélységi bejárás – *căutare în adâncime*
digraph, directed graph – irányított gráf – *graf orientat*
directed circuit, circuit in digraph – irányított zárt vonal – *circuit simplu*
directed cycle, cycle in digraph – irányított kör – *circuit elementar*
directed graph, digraph – irányított gráf – *graf orientat*
directed path, path in digraph – irányított út – *drum elementar*
dual graph – duális gráf – *graf dual*
edge – él – *muchie*
flow – folyam – *flux*
forest – liget – *pădure*
graph – egyszerű gráf – *graf simplu*
greedy algorithm – mohó algoritmus – *algoritmul greedy*
incidence matrix – illeszkedési (incidencia) mátrix – *matrice de incidenţă*
in-degree – be-fok – *grad interior*

loop – hurok – *buclă*
matching – párosítás – *cuplaj*
multigraph – (multi)gráf – *multigraf*
network – közlekedési hálózat – *reţea de transport*
node, vertex – csúcs, szögpont – *nod, vîrf*
order of graph – gráf rendje – *ordinul grafului*
out-degree – ki-fok – *grad exterior*
path – út – *lanţ elementar*
path in digraph, directed path – irányított út – *drum elementar*
planar graph – síkgráf – *graf planar*
regular graph – reguláris gráf – *graf regular*
semipath – lánc – *lanţ în graf orientat*
sink – nyelő – *puţ*
size of graph – gráf nagysága – *dimensiunea grafului*
source – forrás – sursă
spanning subgraph – feszítő részgráf – *graf parţial*
spanning tree – faváz, feszítőfa – *arbore de acoperire, arbore parţial*
subgraph – részgráf – *subgraf*
trail – vonal – *lanţ simplu*
trail in digraph – irányított vonal – *drum simplu*
tree – fa – *arbore*
vertex, node – szögpont, csúcs – *vârf, nod*
walk – séta – *lanţ*
walk in digraph – irányított séta – *drum*

Irodalomjegyzék

[1] Andrásfai Béla: *Ismerkedés a gráfelmélettel*. Tankönyvkiadó, Budapest, 1971. Angolul: *Introductory Graph Theory*, Akadémiai Kiadó, Budapest, Adam Hilger Ltd., Bristol, Pergamon Press Inc. Elmsford, New York, 1977.

[2] Andrásfai Béla: *Gráfelmélet (folyamok, mátrixok)*. Akadémiai Kiadó, Budapest, 1983.

[3] Andrásfai Béla: *Gráfelmélet*. Polygon, Szeged, 1994.

[4] Baase, Sara: *Computer Algorithms: Introduction to Design and Analysis*. Addison-Wesley Publ. Co., 1988.

[5] Bege Antal – Kása Zoltán: *Algoritmikus kombinatorika és számelmélet*. Presa Universitară Clujeană/Kolozsvári Egyetemi Kiadó, 2006.

[6] Berge, Claude: *Théorie des graphes et ses applications*. Dunod, Paris, 1967. Románul: *Teoria grafurilor şi aplicaţii*, Editura Tehnică, Bucureşti, 1969.

[7] Berge, Claude: *Graphes et hypergraphes*. Dunod, Paris, 1970.

[8] Chartrand, Gary – Oellermann, Ortrud R.: *Applied and Algorithmic Graph Theory*. McGraw-Hill, Inc., 1993.

[9] Ciurea, Eleonor: *Algoritmi. Introducere în algoritmica grafurilor*. Editura Tehnică, 2001.

[10] Comtet, L.: *Advanced Combinatorics. The Art of Finite and Infinite Expansions*. D. Reidel Publ. Co., Dordrecht–Boston, 1974.

[11] Cormen, Thomas H. – Leiserson, Charles E. – Rivest, Ronald L. – Stein, Clifford: *Introduction to Algorithms*. Mit

Press–McGraw-Hill, 2001. Magyarul: *Új algoritmusok*, Scolar Kiadó, Budapest, 2003.

[12] Cseke Vilmos: *A gráfelmélet és alkalmazásai*. Tudományos Könyvkiadó, Bukarest, 1972.

[13] Graham, Ronald L. – Knuth, Donald E. – Patashnik, Oren: *Concrete Mathematics. A Foundation for Computer Science*. Addison-Wesley, 1994. Magyarul: *Komkrét matematika*, Műszaki Könyvkiadó, Budapest, 1998.

[14] Hajnal Péter: *Összeszámlálási problémák*. Polygon, Szeged, 1997.

[15] Hopcropft, John E. – Ullmann, Jeffrey D.: *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley Publ. Co., 1979.

[16] Ionescu Clara – Zsakó Ioan: *Structuri arborescente cu aplicaţiile lor*. Editura Tehnică, Bucureşti, 1990.

[17] Ionescu, H. – Dinescu, C. – Săvulescu, B.: *Probleme ale cercetării operaţionale*. Ed. Didactică şi Pedagogică, Bucureşti, 1972.

[18] Iványi Antal (szerk.): *Conference of Young Programmers and Mathematicians*. Eötvös Loránd University, Budapest, 1984.

[19] Iványi Antal (szerk.): *Third Conference of Program Designers*. Eötvös Loránd University, Budapest, 1984.

[20] Katona Gyula Y. – Recski András: *Gráfelmélet és algoritmuselmélet*. ELTE-jegyzet, Budapest, 1994.

[21] Katona Gyula Y. – Recski András – Szabó Csaba: *Gráfelmélet, algoritmuselmélet és algebra*. BME-jegyzet, Budapest, 1997.

[22] Knuth, Donald E.: *The Art of Computer Programming, Vol 1: Fundamental Algorithms. Second edition*. Addison-Wesley, 1981. Magyarul: *A számítógép-programozás művészete 1. Alapvető algoritmusok*, Második kiadás, Műszaki Könyvkiadó, Budapest, 1994.

[23] Kása Zoltán: *Combinatorică cu aplicaţii*. Presa Universitară Clujeană, Cluj, 2003.

[24] Livovschi, Leon – Georgescu, Horia – Popovici, Constantin P. – Ţăndăreanu, Nicolae: *Bazele informaticii.* Ed. Didactică şi Pedagogică, Bucureşti, 1981.

[25] Lovász László: *Combinatorial Problems and Exercises.* Akadémiai Kiadó, Budapest & North-Holland, Amsterdam, 1979. Magyarul: *Kombinatorikai problémák és feladatok*, Typotex Kiadó, Budapest, 1999.

[26] Lovász László – Pelikán József – Vesztergombi Katalin: *Diszkrét matematika.* Typotex, Budapest, 2006.

[27] Ore, Øystein: *Graphs and their uses.* Yale University, 1963. Magyarul: *A gráfok és alkalmazásaik.* Gondolat Kiadó, Budapest, 1972. Románul: *Grafuri şi aplicaţii.* Ed. Tehnică, Bucureşti, 1975.

[28] Pach János: Sík mezőben hármas út. *MTA Közgyülési Előadások (szerk. Potó J.),* Magyar Tudományos Akadémia, Budapest, 1999., 131–138. p.

[29] Riordan, J.: *An Introduction to Combinatorial Analysis.* John Wiley & Sons. Inc., 1958.

[30] Riordan, J.: *Combinatorial Identities.* John Wiley & Sons. Inc., 1968.

[31] Tomescu, Ioan: *Introducere în combinatorică.* Editura Tehnică, Bucureşti, 1972.

[32] Tomescu, Ioan: *Probleme de combinatorică şi teoria grafurilor.* Editura Didactică şi Pedagogică, Bucureşti, 1981.

[33] Vilenkin, N. J.: *Kombinatorika.* Műszaki Könyvkiadó, Budapest, 1971.

[34] Wilf, H. S.: *Generatingfunctionology.* Academic Press Inc., Boston, MA, 1996.

Tárgymutató