

# Gráfok ábrázolása- gyakorlati anyag<sup>1</sup>

## TARTALOMJEGYZÉK

|   |    |
|---|----|
| Definíciók átismétlése .....  | 2  |
| Ábrázolási módok .....  | 2  |
| A gráf szöveges megadása .....  | 3  |
| Gráf ábrázolása a számítógépeken .....  | 3  |
| Ábrázolással kapcsolatos gyakorló feladatok .....                               | 5  |
| Szomszédossági listás (éllistas) ábrázolás felépítése csúcsmátrixból .....      | 5  |
| Befok-kifok előállítás szomszédossági listával ábrázolt gráfon .....            | 5  |
| Transzponált gráf felépítése .....  | 6  |
| Transzponálás „helyben” (haladóbb feladat) .....                                | 6  |
| Különbség gráf .....  | 8  |
| Komplementer gráf .....   | 9  |
| Szorgalmi házi feladatok .....  | 10 |
| $G^2$ gráf előállítása csúcsmátrixra, szomszédossági listára .....              | 10 |
| Abszolút nyelő csúcs keresése csúcsmátrixos ábrázolású irányított gráfban ..... | 11 |

---

<sup>1</sup> Készítette: Veszprémi Anna

## DEFINÍCIÓK ÁTISMÉTLÉSE

Mielőtt megnézzük az ábrázolással kapcsolatos feladatokat, ismételjük át az előadáson hallott, gráfokkal kapcsolatos fontosabb, ide vágó definíciókat:

### Gráf definíciója:

Gráf alatt egy  $G = (V, E)$  rendezett párost értünk, ahol  $V$  a csúcsok (vertices) tetszőleges, véges halmaza,  $E \subseteq V \times V \setminus \{(u, u) : u \in V\}$  pedig az élek (edges) halmaza. Ha  $V = \{\}$ , akkor üres gráfról, ha  $V \neq \{\}$ , akkor nemüres gráfról beszélünk.

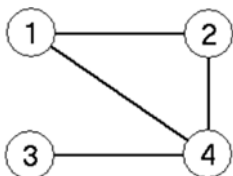
Megjegyzés: a definícióból két fontos dolog következik: a gráfokban, amelyekkel foglalkozni fogunk, nincsenek hurokélek, és nincsenek párhuzamos élek, azaz bármely két csúcs között legfeljebb egy éle lehet a gráfnak.

Az ábrázolásnál lényeges lesz, hogy a gráfunk irányított, vagy irányítatlan. Nézzük a definíciókat:

### Irányítatlan gráf definíciója:

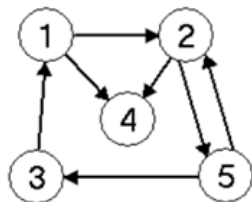
A  $G = (V, E)$  gráf irányítatlan, ha tetszőleges  $(u, v) \in E$  élre  $(u, v) = (v, u)$ .

Azaz, ha  $(u, v)$  létezik, akkor  $(v, u)$  él is létezik, mindkét élt ábrázolni kell!



### Irányított gráf definíciója:

A  $G = (V, E)$  gráf irányított, ha tetszőleges  $(u, v); (v, u) \in E$  élpárra  $(u, v) \neq (v, u)$ . Ilyenkor azt mondjuk, hogy az  $(u, v)$  él fordítottja a  $(v, u)$  él, és viszont.



### Út definíciója:

A  $G = (V; E)$  gráf csúcsainak  $(V)$  egy  $\langle u_0; u_1; \dots; u_n \rangle$   $(n \in \mathbb{N})$  sorozata a gráf egy útja, ha tetszőleges  $i \in 1..n$ -re  $(u_{i-1}, u_i) \in E$ . Ezek az  $(u_{i-1}, u_i)$  élek az út élei. Az út hossza ilyenkor  $n$ , azaz az utat alkotó élek számával egyenlő.

## ÁBRÁZOLÁSI MÓDOK

A gráfábrázolásoknál a  $G = (V, E)$  gráfról általában fölteszük, hogy  $V = \{1, \dots, n\}$ , ahol  $n = |V|$ , azaz hogy a gráf csúcsait egyértelműen azonosítják az  $1..n$  sorszámok.

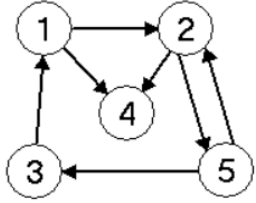
Jelölhetjük a gráf csúcsait az angol ábécé kisebb betűivel is:  $a=1..v=26$  azonosítják a gráf csúcsait.

Az ábrázolásainknál, és az ezeken futó algoritmusoknál a hatékonyság miatt nagyon lényeges, hogy a csúcs egyben egy sorszámot is jelent, azaz konstans időben tudunk tetszőleges csúcsot beazonosítani a gráfban.

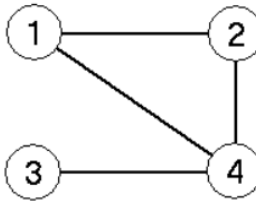
## A gráf szöveges megadása

Egy gráfot megadhatunk szöveges leírással, vagy rajzzal szemléltethetjük. A tárgy a következő szöveges leírást használja:

### G irányított gráf

|   |  |
|---|--|
| Szöveges megadás:<br>1 → 2;4.<br>2 → 4;5.<br>3 → 1.<br>5 → 2;3. | A gráf képe:<br> |
|---|--|

### G irányítatlan gráf

|   |  |
|---|--|
| Szöveges megadás:<br>1 – 2;4.<br>2 – 4.<br>3 – 4. | A gráf képe:<br> |
|---|--|

Ez utóbbinál figyeljük meg, hogy a szöveges megadás az  $(u,v) = (v,u)$  élt csak egyszer adja meg.

## Gráf ábrázolása a számítógépeken

Olyan ábrázolásra van szükség, melyet a gráfokkal kapcsolatos algoritmusok hatékonyan tudnak használni. Az algoritmusok, amelyeket tanulni fogunk, a gráf egy csúcsának feldolgozaskor a csúcs „szomszédait” fogják meglátogatni. (Egy  $u$  csúcs szomszédjainak azokat a csúcsokat tekintjük, melyekhez vezet él a gráfban, azaz ha létezik  $(u,v)$  él, akkor  $v$  az  $u$  szomszédja.) Így az ábrázolásnak ezt kell hatékonyan támogatnia.

Két ilyen gyakran használt, hatékony ábrázolást fogunk tanulni:

- Szomszédossági mátrixos (csúcsmátrixos, vagy adjacency mátrixos) ábrázolás: a gráfot egy  $n \times n$ -es bitmátrix ábrázolja ( $n = |V|$ ), legyen a neve:  $A$ .  $A \in \text{bit}^{n \times n}$ ,  $A[i,j] = 1$ , ha van  $(i,j)$  él a gráfban, 0 egyébként. Az  $i$  csúcs szomszédjainak bejárása  $\Theta(n)$  költségű: a mátrix  $i$ -dik sorát kell végig járni. Úgynevezett „sűrű” gráfoknál célszerű ezt az ábrázolást használni, amikor  $|E| \in \Theta(n^2)$ .
- Szomszédossági listás ábrázolás (éllistás listás): az  $i$  csúcs szomszédjait egy egyszerű lista (fejelem nélküli, egyirányú) ábrázolja. A lista elemek Edge típusúak. A listák kezdő pointerei (*nem fejelemei!*) egy  $n$  méretű,  $\text{Edge}^*$  típusú tömbben vannak, legyen a neve:  $A$ .  $A \in (\text{Edge}^*)^n$ . Az  $i$  csúcs szomszédjait úgy érhetjük el, hogy bejárjuk az  $A[i]$  pointerű listát. Ennek költsége:  $O(n)$ . Fontos, hogy a listák első elemére mutató pointerek egy tömbben vannak elhelyezve, így konstans időben érhetjük el a listák kezdő pointerét.

Az Edge típus UML leírása:

| Edge         |
|--------------|
| +v : N       |
| +next: Edge* |

Nem lényeges tulajdonság, de az áttekinthetőség miatt a listákat csúcs szerint növekvően rendezve szoktuk ábrázolni. Ez egy adott szomszéd keresését még gyorsíthatja is.

Nézzük meg a példa gráfok ábrázolását:

### Írányított gráf

Szöveges megadás:

1 → 2;4.

2 → 4;5.

3 → 1.

5 → 2;3.

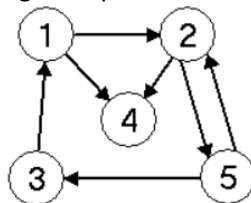
Csúcsmátrix:

|   | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 1 | 0 |
| 2 | 0 | 0 | 0 | 1 | 1 |
| 3 | 1 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 1 | 1 | 0 | 0 |

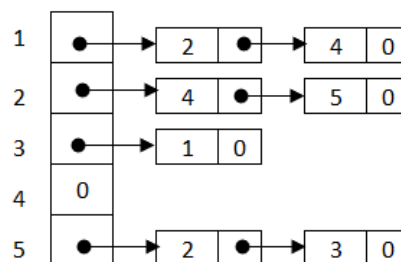
$A[i,j] = 1 \leftrightarrow (i,j) \in E$

A/1 jelzi majd, hogy A egy egytől indexelt mátrix.

A gráf képe:



Szomszédossági lista:



A[i] azon egyszerű lista első elemére mutat, amely i csúcs szomszédjait tartalmazza. A lista elemei Edge típusúak, így a tömb egy eleme, azaz A[i]: Edge\* típusú!

A/1 jelzi majd, hogy A egy egytől indexelt tömb.

### Írányítatlan gráf

Szöveges megadás:

1 – 2;4.

2 – 4.

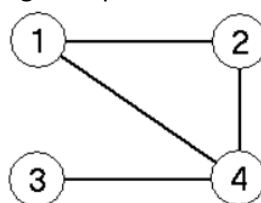
3 – 4.

Csúcsmátrix:

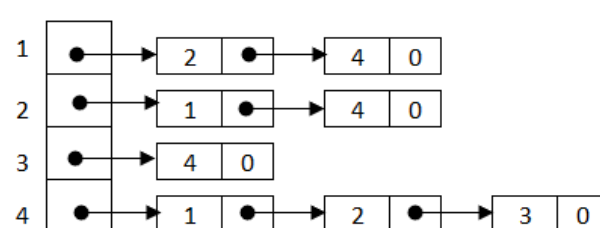
|   | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 1 |
| 2 | 1 | 0 | 0 | 1 |
| 3 | 0 | 0 | 0 | 1 |
| 4 | 1 | 1 | 1 | 0 |

Mindig szimmetrikus mátrix, így nagy gráfok esetén helytakarékosan szokták a mátrixot ábrázolni: csak a főátló alatti elemeket ábrázoljuk egy egydimenziós tömbben, sorfolytonosan elhelyezve.

A gráf képe:



Szomszédossági lista:



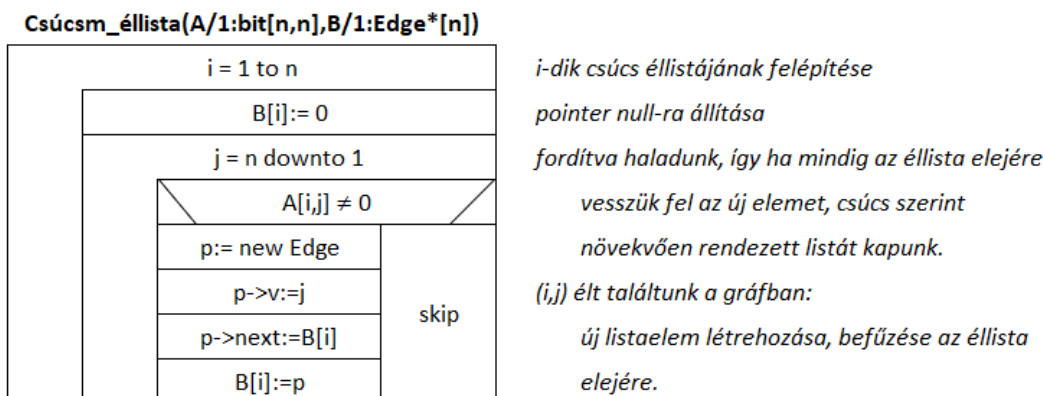
FONTOS: az élek mindkét irányban szerepelnek.

## ÁBRÁZOLÁSSAL KAPCSOLATOS GYAKORLÓ FELADATOK

### Szomszédossági listás (éllistas) ábrázolás felépítése csúcsmátrixból

Adott egy irányított gráf csúcsmátrixos ábrázolása az  $A/1 : \text{bit}[n,n]$  mátrixban. Készítsük el a gráf szomszédossági listás ábrázolását a  $B/1 : \text{Edge}^*[n]$  tömbben. Az éllisták legyenek csúcs szerint rendezettek. Műveletigény:  $O(n^2)$ , ahol  $n=|V|$ .

*Megoldás ötlete: soronként bejárjuk a mátrixot. Az  $i$ -dik sor feldolgozásakor az  $i$  csúcsból induló éllistát kell előállítani. Ha az  $i$ -dik sort  $1..n$  irányban járjuk be, akkor az új elemet mindig a lista végére kellene fűzni, hogy növekvően rendezett listát kapjunk. Ezért kellene egy plusz pointer, ami mindig a lista utolsó elemére mutat. Ha a sort fordítva,  $n..1$  irányban dolgozzuk fel, akkor viszont mindig a lista elejére kell fűzni az új elemet, így nincs szükség a lista végének nyilvántartására.*

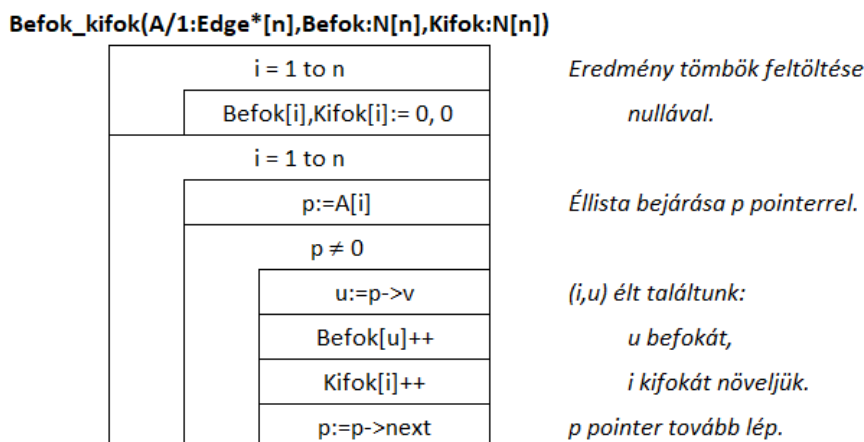


A műveletigény könnyen látszik: a külső ciklus  $n$  iterációt végez, a belső ciklus szintén, az él előállítására viszont csak akkor van szükség, ha a mátrix elem 1 értékű, ezért a belső ciklus igaz ága legfeljebb  $n^2$ -szer hajtódik végre.

### Befok-kifok előállítás szomszédossági listával ábrázolt gráfon

Adott egy irányított gráf szomszédossági listás ábrázolása az  $A/1 : \text{Edge}^*[n]$  tömbben. Adottak még a Befok/ $1 : N[n]$  és Kifok/ $1 : N[n]$  tömbök. Állítsuk elő a gráf csúcsainak befokát és kifokát a Befok és Kifok tömbökben. Befok[ $i$ ]= hány él mutat  $i$  csúcsba, Kifok[ $i$ ]= hány él indul  $i$  csúcsból. A példa gráfra Befok[1]=1 és Kifok[1]=2 lenne. Műveletigény:  $\Theta(n+m)$ , ahol  $n=|V|$  és  $m=|E|$ .

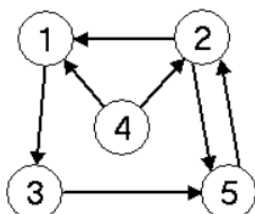
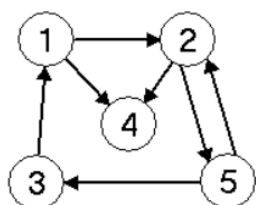
*Megoldás ötlete: feltöltjük nullával a kifok, befok tömböket. Egyszer végig járjuk az éllistákat. Amikor az  $A[i]$  éllistát dolgozzuk fel, akkor a listában szereplő élek ( $i, p \rightarrow v$ ) élt jelentenek, tehát  $i$  csúcs kifokát és  $p \rightarrow v$  csúcs befokát kell megnövelni.*



Műveletigény: az első ciklus műveletigénye  $\Theta(n)$ , a második ciklus első lépése  $n$ -szer fog végrehajtódni, míg a belső ciklus az élek számával arányos, azaz  $\Theta(m)$  műveletigényű, azaz összességében:  $\Theta(n+m)$

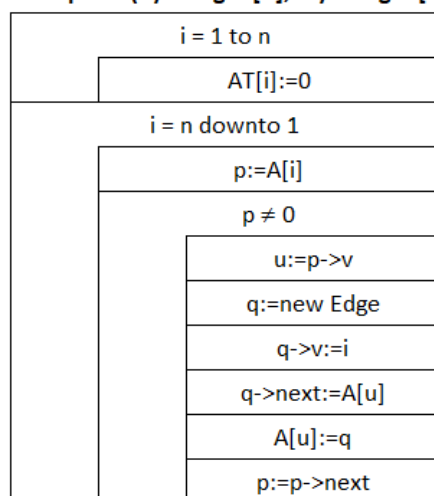
## Transzponált gráf felépítése

Adott egy irányított gráf szomszédossági listás ábrázolása az  $A/1 : \text{Edge}^*[n]$  tömbben. Állítsuk elő a gráf transzponáltját az  $AT/1 : \text{Edge}^*[n]$  tömbben. Az éllisták legyenek csúcs szerint rendezettek mindkét ábrázolásban. Irányított gráf transzponáltja: a csúcsok ugyanazok, de az élek iránya fordított, azaz ha az eredeti gráfnak volt  $(u,v)$  éle, akkor és csak akkor a transzponált gráfnak lesz  $(v,u)$  éle. Műveletigény:  $\Theta(n+m)$ , ahol  $n=|V|$  és  $m=|E|$ .



Megoldás ötlete: feltöltjük null pointerrel az  $AT$  tömböt. Bejárjuk a gráf éllistáit. Az  $A[i]$  éllista feldolgozása közben  $(i, p \rightarrow v)$  éleket dolgozunk fel ( $p$  pointer mutat az éppen feldolgozott lista elemre), azaz a transzponált gráfot ábrázoló adatszerkezetbe egy  $(p \rightarrow v, i)$  élt kell felvennünk. Rendezettség kérdése: ha  $i$ -vel  $1..n$  irányban járjuk be az  $A[]$  tömböt, akkor a transzponált gráfban mindig a  $p \rightarrow v$  csúcs éllistájának végére kellene fűzni az új listaelemet. Ha minden esetben elmegyünk a lista végére egy pointerrel, megnöveljük a futási időt. Ha nyilvántartjuk a lista végeket, plusz  $\Theta(n)$  tárigénye lenne az algoritmusnak. Viszont, ha egy ügyes trükkkel fordítva,  $n..1$  irányban járjuk be az eredeti gráf éllistáit, akkor  $i$  csökkenő, így mindig a lista elejére kell felvenni az új élt, így nem nő a tárigény, és a kívánt műveletigény is megvalósul.

### Transzponál( $A/1:\text{Edge}^*[n], AT/1:\text{Edge}^*[n]$ )



$AT$  pointer tömb feltöltése null értékkel.

Visszafelé haladunk az eredeti gráf csúcsain.

$i$  csúcs éllistájának bejárása  $p$  pointerrel.

$(i,u)$  él volt az eredeti gráfban, tehát egy

$(u,i)$  élt létrehozni a transzponált

gráfban. Új listaelemet hozunk létre, kitöltjük,

majd befűzzük az  $u$  csúcs éllistájának

elejére.

A listát bejáró pointert tovább léptetjük.

Műveletigény: első ciklus  $\Theta(n)$ , második ciklus  $\Theta(n+m)$ , tehát összességében az algoritmus  $\Theta(n+m)$ .

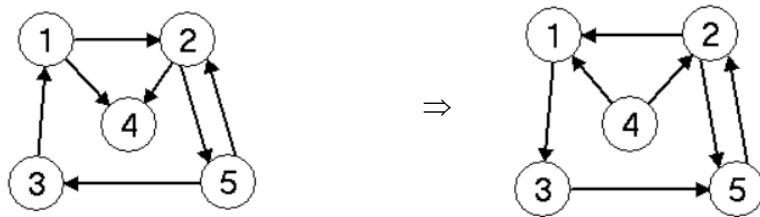
## Transzponálás „helyben” (haladóbb feladat)

Nagy gráfok esetén a memória kímélése miatt transzponálás esetén az eredeti gráfot lebontják, és annak listaelemeit felhasználva állítják elő a transzponált gráfot. Így kicsit nehezebb a feladat. Készítsük el a „helyben” transzponálás algoritmusát. Az éllisták növekvőleg rendezettek az eredeti adatszerkezetben, és azt szeretnénk, ha a transzponált gráfot leíró adatszerkezet listái is csúcs szerint növekvően rendezettek lennének. Műveletigény:  $\Theta(n+m)$ , ahol  $n=|V|$  és  $m=|E|$ .

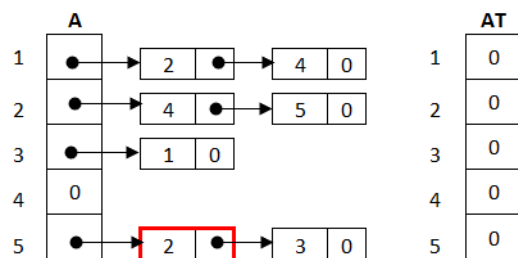
Megoldás ötlete: az előző feladatban leírtakhoz hasonlóan járunk el, csak nem új listaelemet foglalunk, hanem az eredeti listaelemet kifűzzük, átírjuk a csúcsot, és befűzzük a helyére. Ha a ciklus  $n..1$  irányú, akkor most is mindig a lista elejére kell befűzzük az új elemet.

Szemléltető ábra az algoritmus működéséhez:

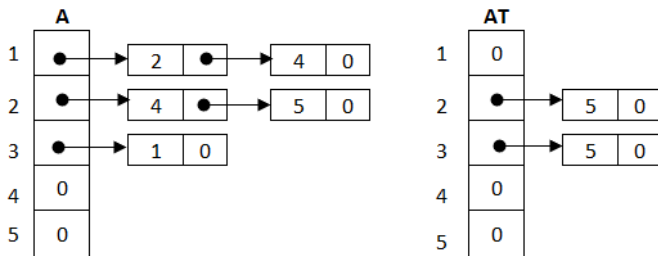
Transzponáljuk a példa gráfunkat:



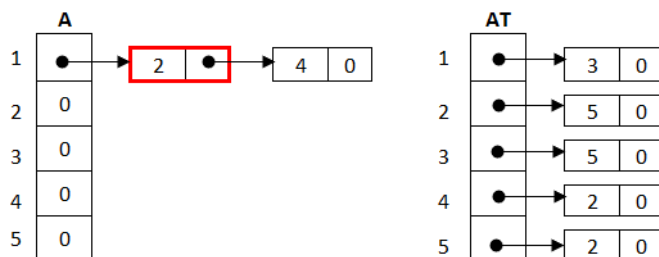
Induló helyzet. A pirossal jelzett listaelemmel indul a feldolgozás: (5,2) élből a transzponált gráfban egy (2,5) élt hozunk létre.



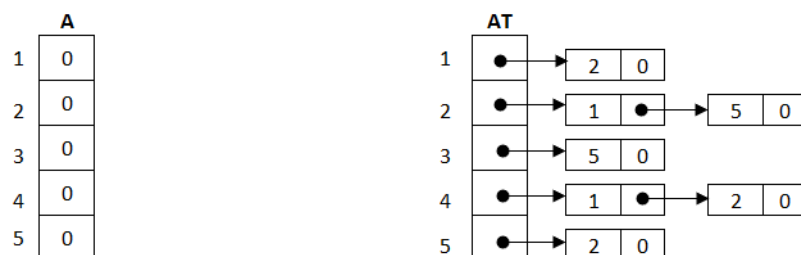
Az 5-ös csúcs éllistáját lebontottuk, a transzponált gráfban létrehoztuk a (2,5), majd (3,5) éleket.



Folytatva az algoritmust, a 3-as és 2-es csúcsok éllistáit is lebontottuk, a transzponált gráfban létrejöttek az (1,3), (4,2) és (5,2) élek. Most fogjuk látni, az n..1 irányú ciklus előnyét, a pirossal bekeretezett (1,2) élt megfordítva (2,1) éle lesz a transzponált gráfnak, és ez pont a 2-es csúcs listájának elejére illik, a következő (1,4) él fordítottja pedig a 4-es csúcs listájának elejére kerül majd.

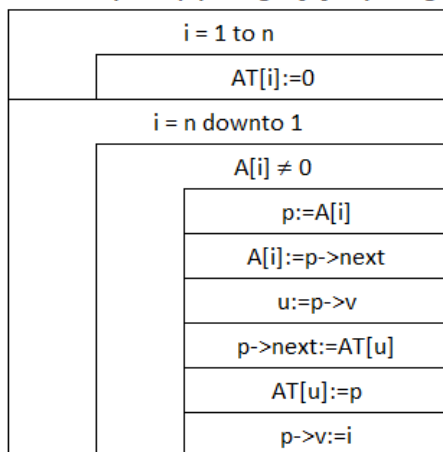


Elkészült a transzponált gráfot leíró adatszerkezet:



És most lássuk az algoritmust:

**HelybenTranszponál(A/1:Edge\*[n],AT/1:Edge\*[n])**



AT pointer tömb feltöltése null értékkel.

Visszafelé haladunk az eredeti gráf csúcsain.

Amíg A[i] lista el nem fogy:

p pointerben megjegyezzük az első listaelem címét,  
kifűzzük a lista első elemét,  
(i,u) él volt a gráfban, (u,i) élt kell létrehozni,  
p című elem befűzése u csúcs éllistájába, a  
lista elejére,  
i-be mutató élt hozunk létre.

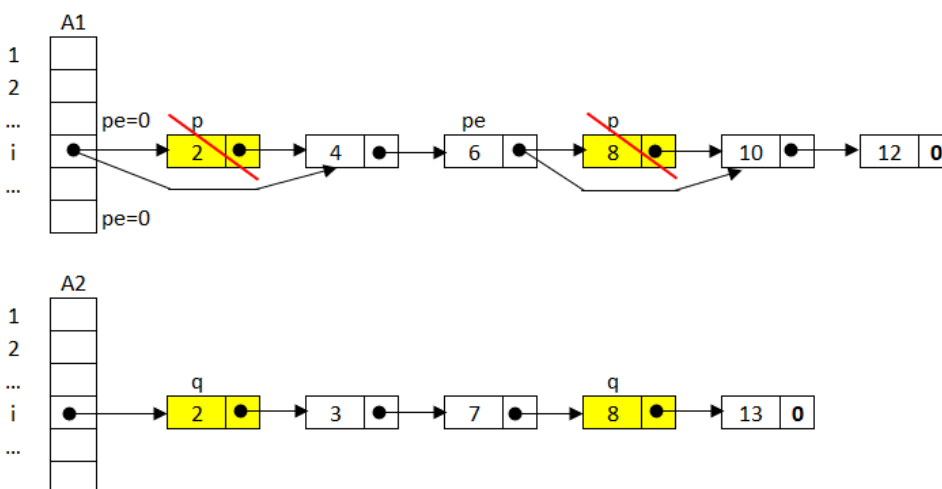
Műveletigény: az előző megoldáshoz hasonlóan könnyen látszik a  $\Theta(n+m)$  műveletigény.

### Különbség gráf

Adott két irányított gráf G1 és G2 szomszédossági listás ábrázolásban, G1 az A1: Edge\*[n], G2 az A2[n] tömbben. A két gráf csúcsai ugyanazok, az élek mások. Az éllisták csúcs szerint növekvően rendezett listák. Készítse el A1 tömbben a G1\G2 gráfot. G1\G2 gráf csúcsai ugyanazok, mint a két bemeneti gráfnak, élei pedig G1 élei közül azok, amelyek, G2 gráfban nem szerepelnek. A megoldásban használja ki, hogy az éllisták rendezettek! Műveletigény  $O(n^2)$

Ötlet: Mivel az éllisták rendezettek, a leghatékonyabb megoldást, a két lista összefésülésével kapjuk. Ennek szemléltetése az i-dik csúcsra:

- A1[i] listán p, A2[i] listán q pointerrel haladunk.
- Három eset lehetséges ( $p \rightarrow v < q \rightarrow v$ ;  $p \rightarrow v = q \rightarrow v$ ;  $p \rightarrow v > q \rightarrow v$ )
- A1[i] listából törölünk elemeket, a törléshez kell az előző listaelem elem címe, ez lesz majd pe pointerben. Ha az elsőt töröljük, akkor viszont A[i] módosul!
- Ha bármelyik listán végig értünk, az összefésülés leállhat.





A megoldás (az összefésülés algoritmusát kiemeltük):

**KülönbőségGráf(A1/1:Edge\*[n]; A2/1:Edge\*[n])**

|                       |
|-----------------------|
| i = 1 to n            |
| Összefésül(A1, A2, i) |

**Összefésül(A1/1:Edge\*[n]; A2/1:Edge\*[n]; i:N)**

|                            |  |              |
|----------------------------|--|--------------|
| pe:=0; p:=A1[i]; q:= A2[i] |  |              |
| p ≠ 0 ∧ q ≠ 0              |  |              |
| p->v < q->v                | p->v = q->v  | p->v > q->v  |
| pe := p<br>p := p->next    | r := p->next<br>pe = 0<br>A[i]:=r<br>pe->next:=r<br>delete p<br>p := r<br>q := q->next | q := q->next |

Műveletigény: egy éllista hossza  $O(n)$ , a listák összefésülése  $O(n)$ ,  $n$  csúcsra elvégezve a listák összefésülését:  $O(n^2)$

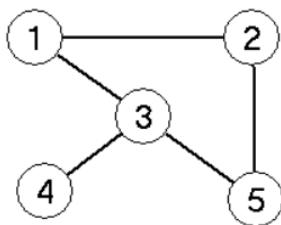
### Komplementer gráf

Adott egy irányítatlan gráf szomszédossági listás reprezentációja az A/1:Edge\*[n] tömbben. Az A tömb az éllisták első elemére mutató pointereket vagy 0 értéket tartalmaz. Az éllisták csúcs szerint növekvően rendezett listák. Készítsen algoritmust, mely az éllistákat egyszer bejárva, hasonló ábrázolással, az AK:Edge\*[n] tömbben létrehozza a komplementer gráfot. Műveletigény:  $O(n+m)$ ,  $O(n)$  segédmemória használható.

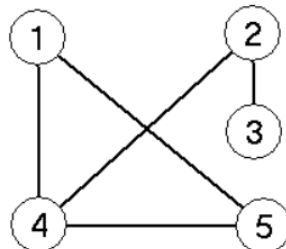
*Definíció: Valamely  $G=(V,E)$  irányítatlan gráf komplementer gráfja az a gráf, amelynek csúcshalmaza megegyezik a  $G$  gráf csúcshalmazával, az élhalmaza pedig a  $G$  gráf élhalmazának a komplementer halmaza (a teljes gráf élhalmazára, mint alaphalmazra nézve).*

*Megjegyzés: Hurokért nem tartalmaznak a gráfok, ügyeljünk rá, hogy a komplementer gráfba se kerüljön be hurokél.*

G gráf:

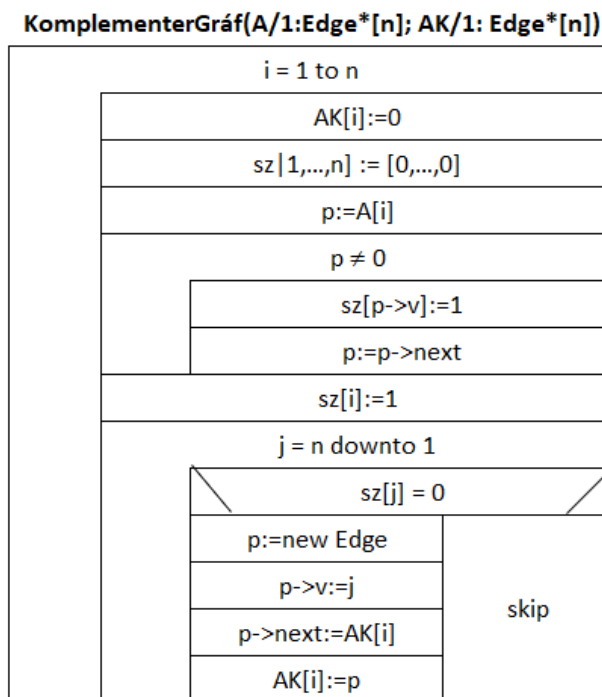


G komplementer gráfja:



Megoldás ötlete:

$A[i]$  ( $1 \leq i \leq n$ ) éllistákat bejárva, egy  $sz[1..n]$  segéd tömbbe a  $k$ -dik helyre 1-et írunk, ha  $i$  csúcsnak van  $k$  szomszédja (azaz  $(i,k)$  él van a gráfban), és 0-át, ha nincs. Ezt a segéd tömböt felhasználva könnyen előállítható a komplementer gráf  $i$  csúcsának éllistája: azokat az éleket kell felvenni, ahol  $sz[i]=0$ . Ügyelni kell két dologra: a kapott  $AK[i]$  éllista csúcs szerint rendezve legyen, valamint, hogy hurokért ne hozzunk létre!



$AK[i]$  lista pointerét nullára állítjuk.

A segéd tömböt feltöltjük nullával.

Bejárjuk  $A[i]$  listát, azon csúcsoknál amelyek szerepelnek a listában, a segéd tömbbe 1-et írunk.

Nehogy hurokért létrehozzunk.

Ott kell felvenni élt, ahol az eredeti gráfban nem volt, azaz  $sz[j]=0$ .

Ha visszafele haladunk, mindig  $AK[i]$  lista elejére kell beszúrni az új élt, hogy végül csúcs szerint rendezett listát kapjunk.

## SZORGALMI HÁZI FELADATOK

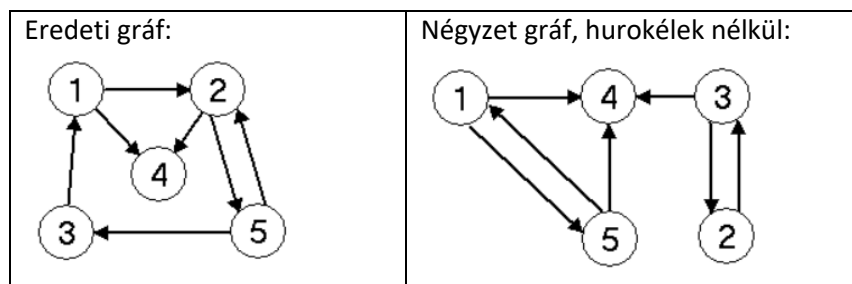
### $G^2$ gráf előállítása csúcsmátrixra, szomszédossági listára

Legyen  $G=(V,E)$  egy irányított gráf.  $G^2$  gráfnak nevezzük azt a  $G^2=(V,E^2)$  gráfot, melynek csúcsai megegyeznek az  $G$  gráf csúcsaival, élei pedig a következők:  $(u,v) \in E^2 \Leftrightarrow (u,w) \in E$  és  $(w,v) \in E$ . Azaz  $(u,v)$  éle a  $G^2$  gráfnak pontosan akkor, ha létezik  $u$ -ból  $v$ -be kettő hosszú út az eredeti gráfban. Ha hurokél keletkezne, azt ne ábrázoljuk a négyzet gráfban.

Készítsünk algoritmust, mely előállítja a  $G^2$  gráfot

- ha  $G$  csúcsmátrixszal van ábrázolva az  $A$  mátrixban,  $G^2$  keletkezzen  $A^2$  mátrixban. Műveletigény:  $O(n^3)$ .
- ha  $G$  szomszédossági listával van ábrázolva  $A$  pointer tömbbel.  $G^2$  keletkezzen  $A^2$  tömbben. Műveletigény:  $O(n*m)$

Nézzük meg példa gráfunk esetén mi lenne a négyzet gráf?



### Abszolút nyelő csúcs keresése csúcsmátrixos ábrázolású irányított gráfban

Legyen  $G=(V,E)$  egy irányított gráf. Csúcsmátrixszal ábrázolva az  $A$  mátrixban. Határozzuk meg van-e abszolút nyelő csúcsa a gráfnak, ha van, adjuk is meg a csúcsot. A gráf  $u$  csúcsát abszolút nyelőnek nevezzük, ha az  $u$  csúcs befoka  $n-1$ , kifoka pedig  $0$ . Azaz minden más csúcsból létezik  $u$ -ba mutató él, viszont  $u$ -ból nem indul ki egyetlen él sem. A feladat tehát a következő: egy olyan  $i$  indexű sort kell találni a mátrixban, hogy a sor csak nullát tartalmaz, de ugyanakkor az  $i$ -dik oszlop a főátlót kivéve csupa  $1$ -est tartalmaz. Könnyű találni  $O(n^2)$  műveletigényű algoritmust: keresünk egy csupa nulla sort (ez  $O(n^2)$ ), ha találtunk, ellenőrizzük a megfelelő oszlopot, hogy a főátlót kivéve csupa egyes-e, ez  $O(n)$ , összességében  $O(n^2)$ . Oldjuk meg  $O(n)$  műveletigénnyel!