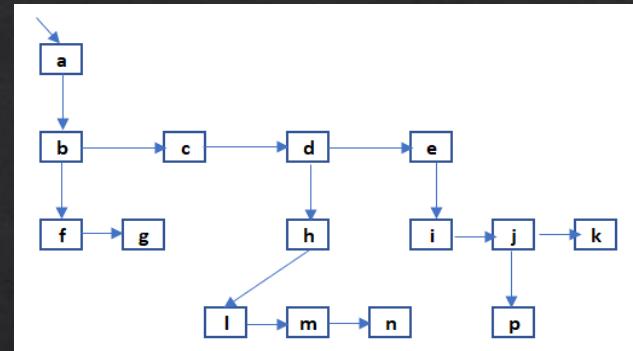


# Algoritmusok és adatszerkeztek II.

5. gyakorlat

# Szorgalmi hf megoldása

- ❖ Adott egy két pointerrel, láncoltan ábrázolt általános fa. Egy csúcsban az első gyerekre és a testvérré mutató pointerek vannak. Készítse el a következő kereső algoritmust: megadunk egy kulcsot, ha a fában van ilyen kulcs, akkor kiírja a csúcsig vezető úton a szülők kulcsait, Elképzelhető, hogy az adott csúcs több helyen is szerepel a fában, akkor mindegyik előforduláshoz írja ki az útvonalat! Az utat a gyökértől indulva, a csúcsig kell kiírni, csak a szülő Node-ok kulcsait.
- ❖ Érdekesség képpen oldjuk meg úgy is, hogy a Node-okban van egy szülő (parent) pointer is. Ha megtaláljuk a keresett kulcsot egy Node-ban, akkor a parent pointerek segítségével írjuk ki az útvonalat.
- ❖ *Megjegyzés: a feladat valós probléma. Például amikor az operációs rendszerek könyvtár rendszerében keresünk egy adott fájl bejegyzést: hol fordul elő egy adott nevű fájl? Lehet több alkönyvtárban is ugyanolyan nevű fájl. Írjuk ki az elérési útvonalakat.*



Keresett kulcs: „n”

Elérési út: a/d/h

Keresett kulcs: „g”

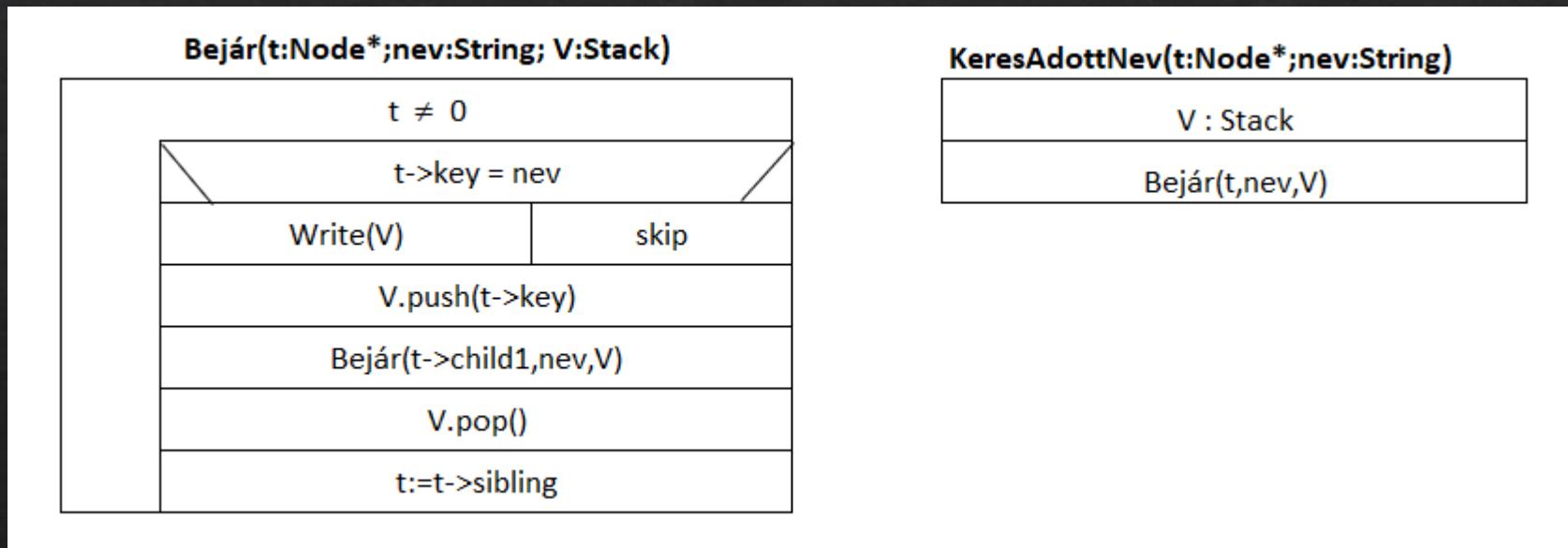
Elérési út: a/b

Keresett kulcs: „a”

Elérési út: (üres szöveg)

# Szorgalmi házi feladat megoldása

- ❖ Ötlet: egy veremben tároljuk az elérési útvonalat.
- ❖ Write(V) kiírja a veremben lévő útvonalat: alulról felfelé haladva, anélkül, hogy a verem tartalma megváltozna.



# Gráfokkal kapcsolatos alapfogalmak

- ❖ **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.

# Irányítatlan / irányított gráf

## ❖ Irányítatlan gráf definíciója:

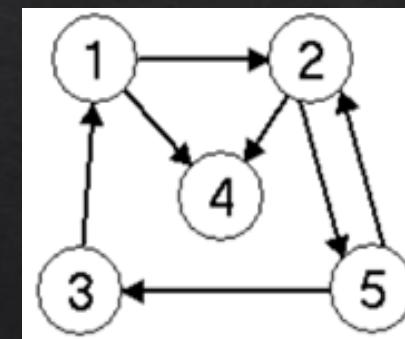
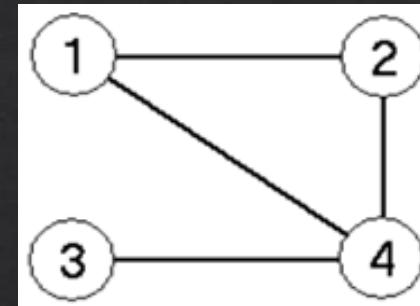
- ❖ A  $G = (V, E)$  gráf irányítatlan, ha tetszőleges  $(u, v) \in E$  ére  $(u, v) = (v, u)$ .
- ❖ Azaz, ha  $(u, v)$  létezik, akkor  $(v, u)$  él is létezik, minden két élt ábrázolni kell!

## ❖ 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 fordította a  $(v, u)$  élt, és viszont.

## ❖ Út definíciója:

- ❖ A  $G = (V, E)$  gráf csúcsainak egy  $\langle u_0; u_1; \dots; u_n \rangle$  ( $n \in N$ ) 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 föltesszük, 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 kisebtűivel is:  $a=1\dots v=26$  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.
- ❖ Ábrázolási módok:
  - ❖ Szöveges megadás
  - ❖ Szomszédossági mátrix
  - ❖ Szomszédossági lista

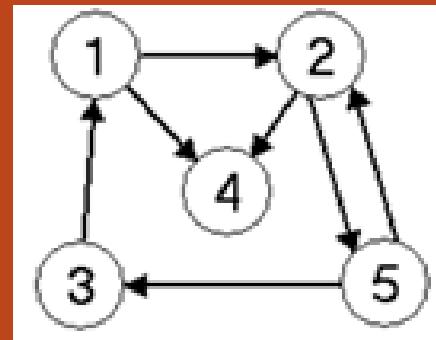
# A gráf szöveges megadása

- ◆ Egy gráfot megadhatunk szöveges leírással, vagy rajzzal szemléltethetjük. A tárgy a következő szöveges leírást használja:
- ◆ Irányított gráf esete:

**Szöveges megadás:**

$1 \rightarrow 2; 4.$   
 $2 \rightarrow 4; 5.$   
 $3 \rightarrow 1.$   
 $5 \rightarrow 2; 3.$

**A gráf képe:**



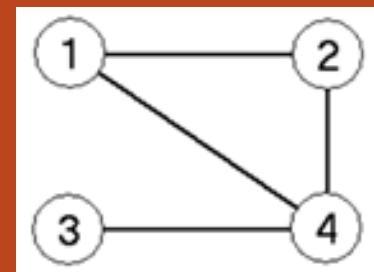
## A gráf szöveges megadása

- ❖ Irányítatlan gráf esete:

**Szöveges megadás:**

1 – 2;4.  
2 – 4.  
3 – 4.

**A gráf képe:**



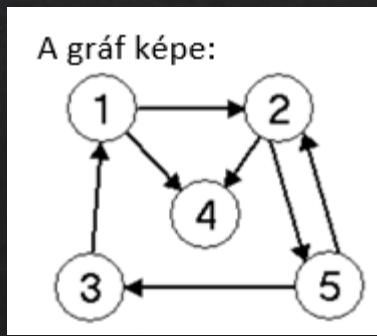
- ❖ Figyeljük meg, hogy a szöveges megadás az  $(u,v) = (v,u)$  élt csak egyszer adja meg.

# 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 feldolgozáskor 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.

# Szomszédossági mátrix

- ❖ 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  $|E| \in O(n^2)$ .



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

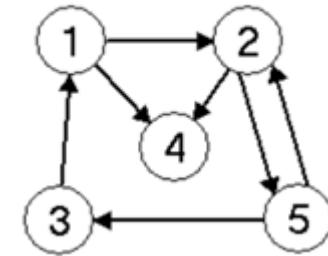
# Szomszédossági lista

- Szomszédossági listás ábrázolás (éllistás ábrázolás): 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, 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.  $|V| = n$
- Úgynévezett ritka gráfok esetén használjuk, amikor  $|E| \in O(n)$

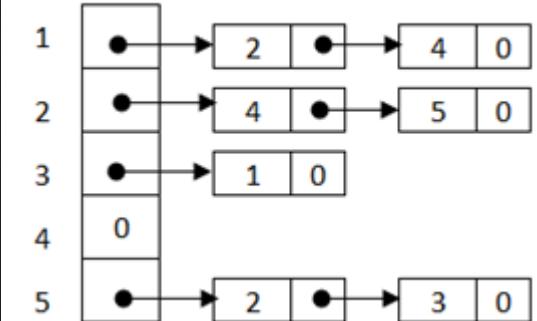
Az Edge típus UML leírása:

Edge
+v : N
+next: Edge*

A gráf képe:



Szomszédossági lista:



# Példa – irányított gráf

## Irányított gráf

Szöveges megadás:

1 → 2;4.

2 → 4;5.

3 → 1.

5 → 2;3.

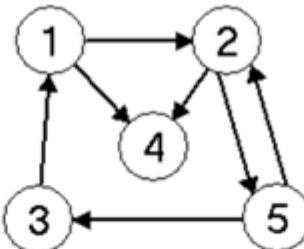
Csúcsmátrix:

	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

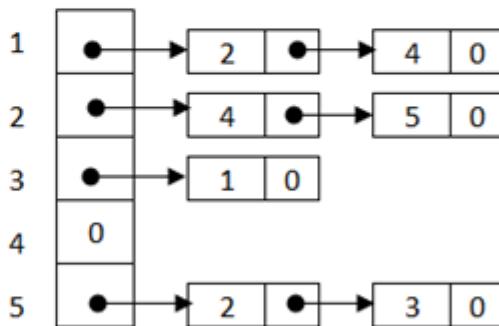
$A[i,j] = 1 \leftrightarrow (i,j) \in E$

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.

# Példa – irányítatlan gráf

## Irányítatlan gráf

Szöveges megadás:

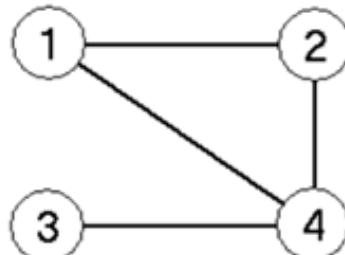
- 1 – 2;4.
- 2 – 4.
- 3 – 4.

Csúcsmátrix:

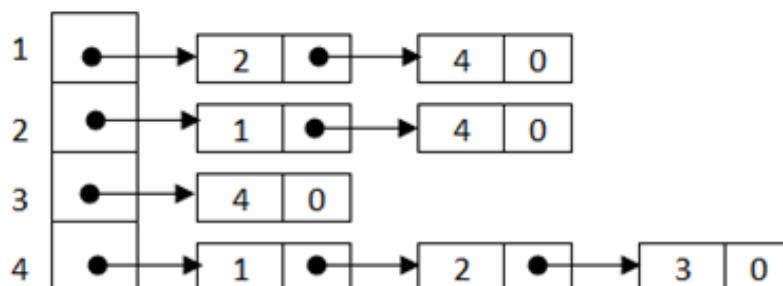
1	2	3	4	
1	0	1	0	1
2	1	0	0	1
3	0	0	0	1
4	1	1	1	0

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 minden két irányban szerepelnek.

# Ábrázolással kapcsolatos gyakorló feladatok

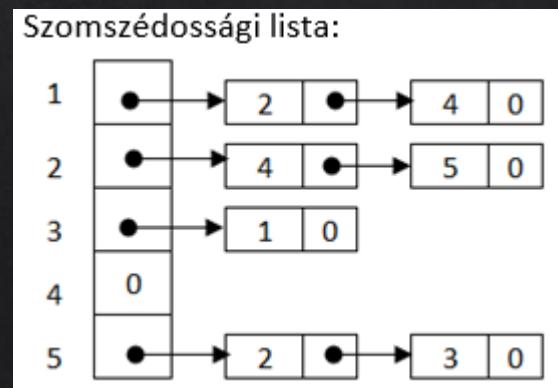
Feladatok:

1. Szomszédossági listás (éllistás) ábrázolás felépítése csúcsmátrixból
2. Befok-kifok előállítása szomszédossági listával ábrázolt gráfon
3. Transzponált gráf felépítése
  - a) Új gráf építése
  - b) Helyben (eredeti gráfból)
4. Különbség gráf
5. Komplementer gráf elkészítése

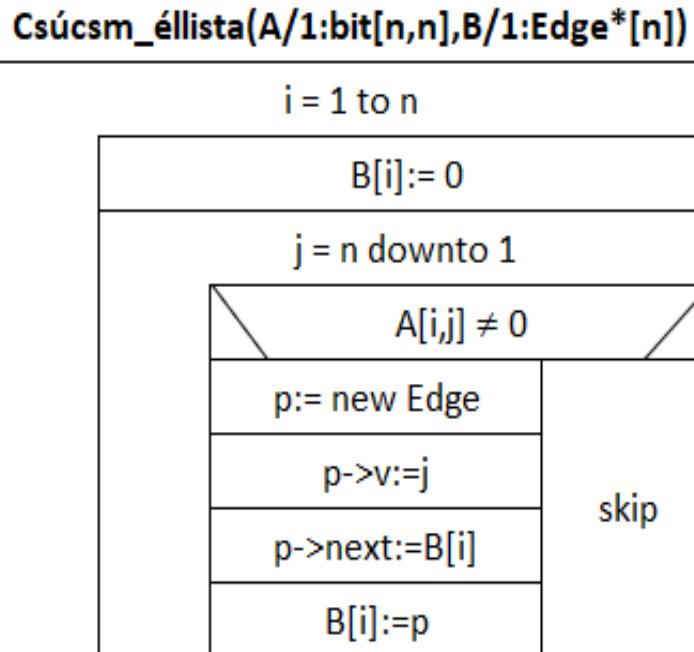
## Szomszédossági listás (éllistás) ábrázolás felépítése csúcsmátrixból

- ❖ Feladat:
- ❖ 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 mindenkor a lista végére kellene fűzni, hogy növekvően rendezett listát kapunk. Ezért kellene egy plusz pointer, ami mindenkor a lista utolsó elemére mutat. Ha a sort fordítva,  $n..1$  irányban dolgozzuk fel, akkor viszont mindenkor a lista elejére kellene fűzni az új elemet, így nincs szükség a lista végének nyilvántartására.

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



## Megoldás struktogramja

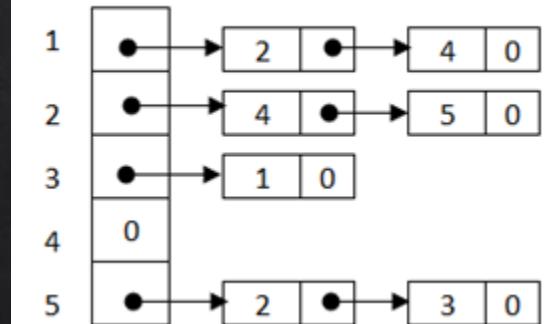


*i-dik csúcs éllistájának felépítése  
pointer null-ra állítása  
fordítva haladunk, így ha mindig az éllista elejére  
vesszük fel az új elemet, csúcs szerint  
növekvően rendezett listát kapunk.  
(i,j) élt találtunk a gráfban:  
új listaelem létrehozása, befűzése az éllista  
elejére.*

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

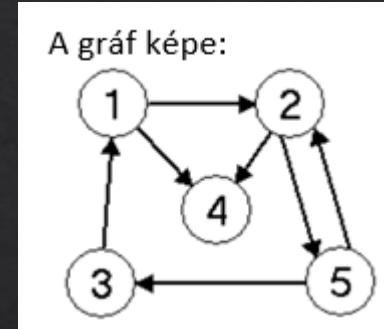
Szomszédossági lista:



Műveletigény: 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 hajtódiik végre.

## Befok-kifok előállítása szomszédossági listával ábrázolt gráfon

- ❖ Feladat:
- ❖ 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.
- ❖  $\text{Befok}[i] =$  hány él mutat i csúcsba,  
 $\text{Kifok}[i] =$  hány él indul i csúcsból.  
A példa gráfra  $\text{Befok}[1]=1$  és  $\text{Kifok}[1]=2$  lenne.  
Műveletigény:  $\Theta(n+m)$ , ahol  $n=|V|$  és  $m=|E|$ .
- ❖ Megoldás ötlete: feltöljük 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.

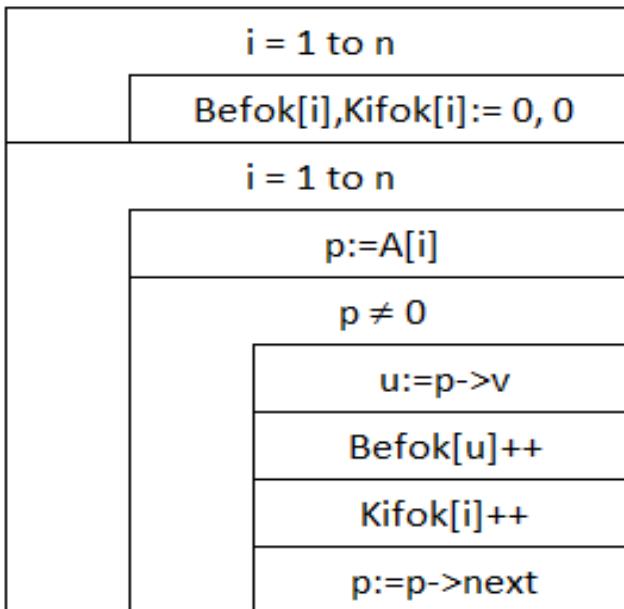


Szomszédossági lista:

1	•	→	2	•	→	4	0
2	•	→	4	•	→	5	0
3	•	→	1	0			
4	0						
5	•	→	2	•	→	3	0

## Megoldás struktogramja

**Befok\_kifok(A/1:Edge\*[n],Befok:N[n],Kifok:N[n])**



*Eredmény tömbök feltöltése nullával.*

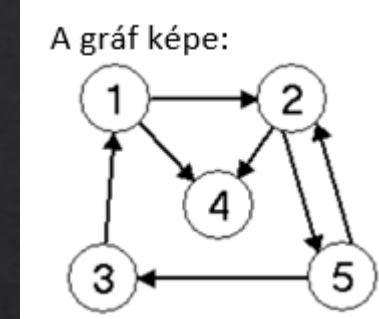
*Éllista bejárása  $p$  pointerrel.*

*$(i, u)$  élt találtunk:*

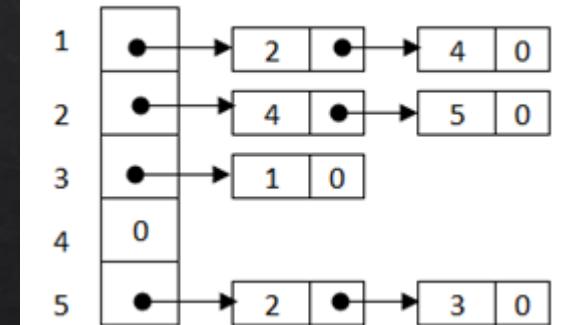
*$u$  befokát,*

*$i$  kifokát növeljük.*

*$p$  pointer tovább lép.*



Szomszédossági lista:

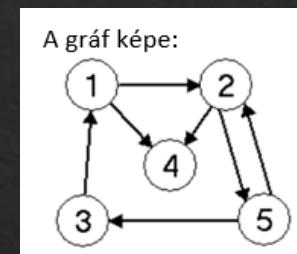


Műveletigény: az első ciklus műveletigénye  $\Theta(n)$ , a második ciklus első lépése  $n$ -szer fog végrehajtódni, míg a belső ciklus az élek számával arányos, azaz  $\Theta(m)$  műveletigényű, azaz összességében:  $\Theta(n+m)$

## Transzponált gráf felépítése

- ❖ Feladat:
- ❖ Adott egy irányított gráf szomszédossági listás ábrázolása az  $A[1 : Edge^*[n]]$  tömbben. Állítsuk elő a gráf transzponáltját az  $AT[1 : Edge^*[n]]$  tömbben. Az éllisták legyenek csúcs szerint rendezettek minden két 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:  $O(n+m)$ , ahol  $n=|V|$  és  $m=|E|$ .
- ❖ Megoldás ötlete: feltöljük null pointerrel az  $AT$  tömböt. Bejárjuk a gráf éllistáit. Az  $A[i]$  éllista feldolgozása közben  $(i,p->v)$  éleket dolgozunk fel ( $p$  pointer mutat az éppen feldolgozott lista elemre), azaz a transzponált gráfot ábrázoló adatszerkezetbe egy  $(p->v,i)$  élt kell felvennünk.
- ❖ Rendezettség kérdése: ha  $i$ -vel  $1..n$  irányban járjuk be az  $A[]$  tömböt, akkor a transzponált gráfban mindig a  $p->v$  csúcs éllistájának végére kellene fűzni az új listaelemet. Ha minden esetben elmegeünk 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 minden a lista elejére kell felvenni az új élt, így nem nő a tárigény, és a kívánt műveletigény is megvalósul.

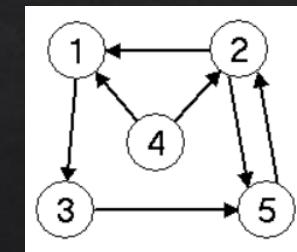
Eredeti gráf



Szomszédossági lista:

	1	2	3	4	5
1	•	2	•	4	0
2	•	4	•	5	0
3	•	1	0		
4	0				
5	•	2	•	3	0

Transzponált gráf

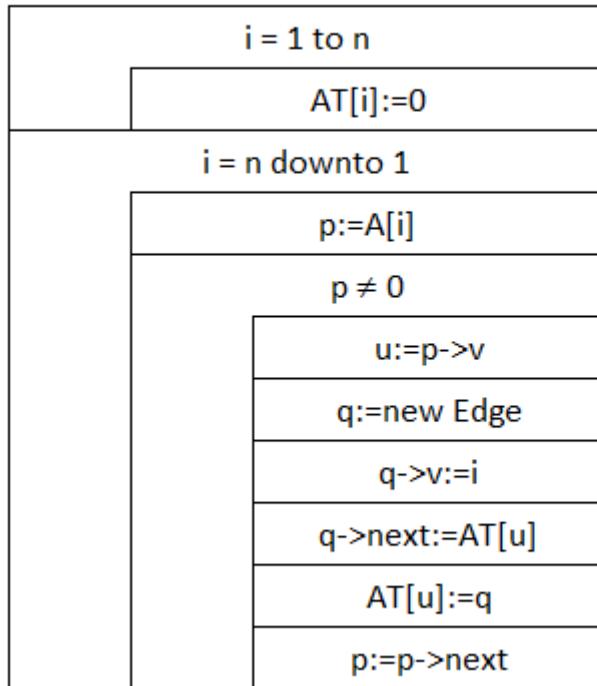


AT

	1	2	3	4	5
1	•	3	0		
2	•	1	•	5	0
3	•	5	0		
4	•	1	•	2	0
5	•	2	0		

## Megoldás struktogramja

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



*AT pointer tömb feltöltése null értékkel.*

*Visszafelé haladunk az eredeti gráf csúcsein.*

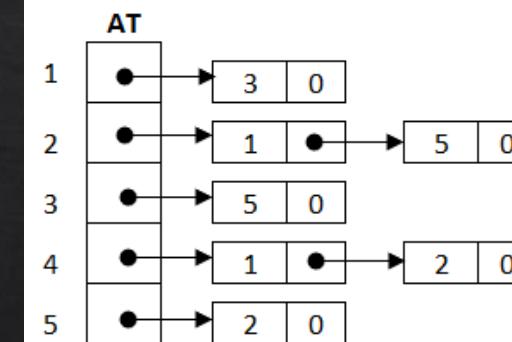
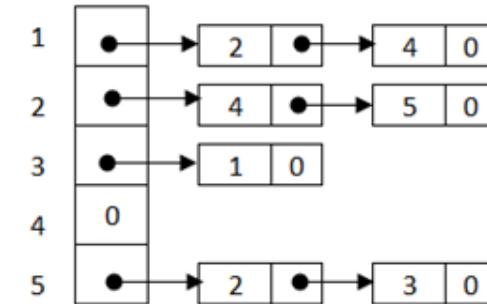
*i csúcs éllistájának bejárása p pointerrel.*

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

*(u,i) élt kell létrehozni a transzponált  
gráfban. Új listaelemet hozunk létre, kitöljük,  
majd befűzzük az u csúcs éllistájának  
elejére.*

*A listát bejáró pointert tovább léptetjük.*

Szomszédossági lista:

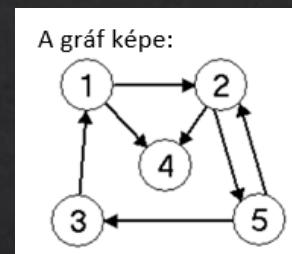


Műveletigény: első ciklus  $\Theta(n)$ , második ciklus  $\Theta(n+m)$ , tehát összességében az algoritmus  $\Theta(n+m)$ .

## Transzponált gráf felépítése „helyben”

- ❖ 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 listaelementet 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.

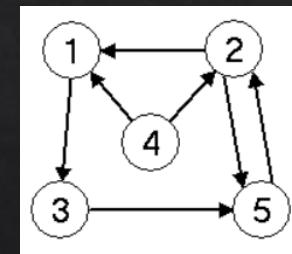
Eredeti gráf



Szomszédossági lista:

	1	2	3	4	5
1	•	2	•	4	0
2	•	4	•	5	0
3	•	1	0		
4	0				
5	•	2	•	3	0

Transzponált gráf

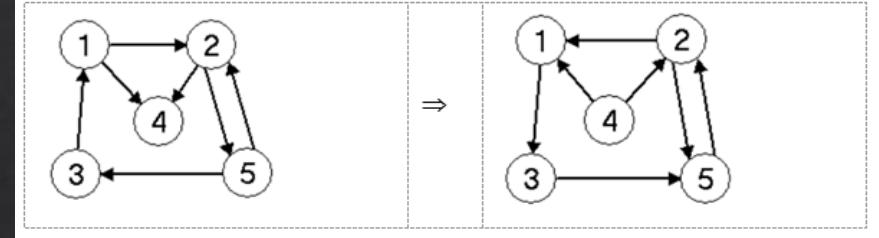


AT

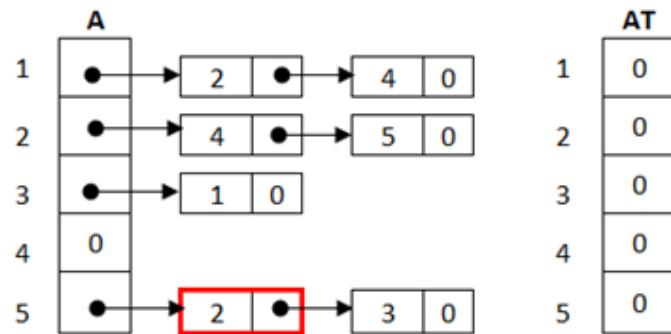
	1	2	3	4	5
1	•	3	0		
2	•	1	•	5	0
3	•	5	0		
4	•	1	•	2	0
5	•	2	0		

## Az algoritmus működésének szemléltetése

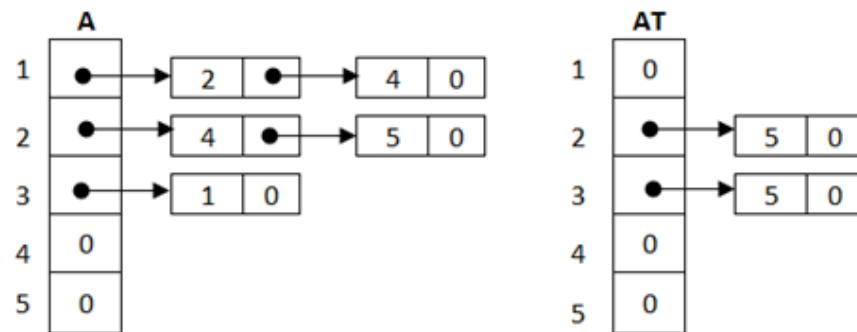
Transzponáljuk a példa gráfunkat:



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

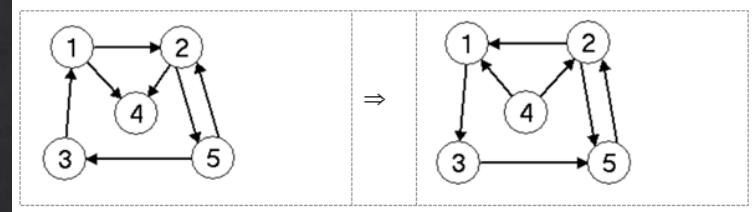


Az 5-ös csúcs éllistáját lebontottuk, a transzponált gráfban létrehoztuk a (2,5), majd (3,5) éleket.

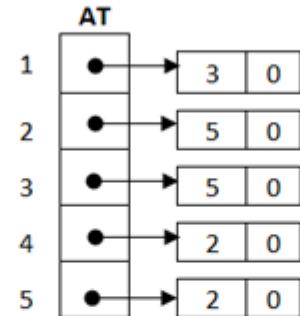
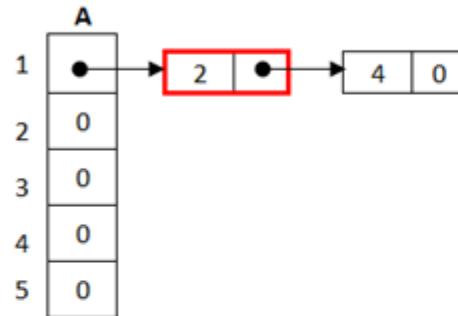


## Az algoritmus működésének szemléltetése

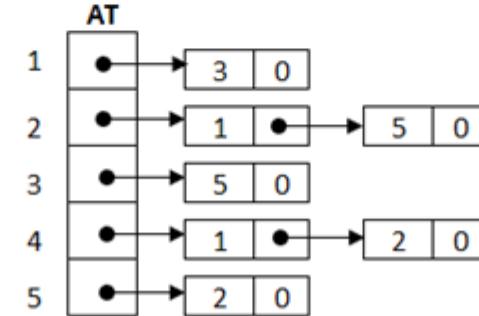
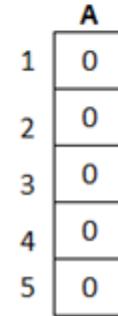
Transzponáljuk a példa gráfunkat:



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:

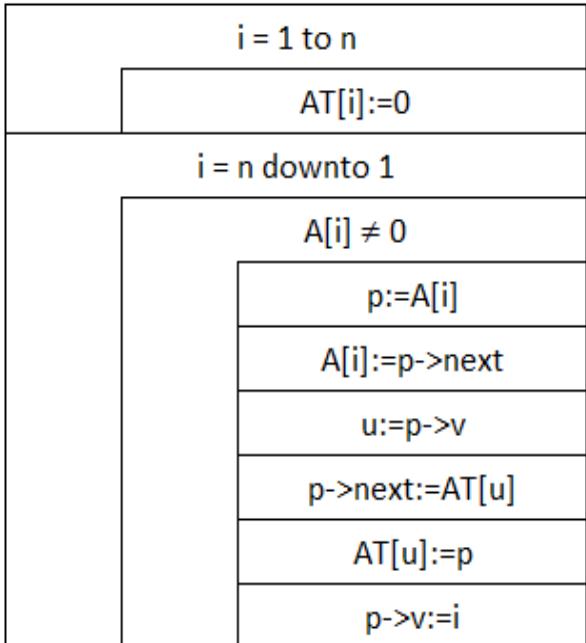


Szomszédossági lista:

1	•	→	2	•	→	4	0
2	•	→	4	•	→	5	0
3	•	→	1	0			
4	0						
5	•	→	2	•	→	3	0

## Megoldás struktogramja

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



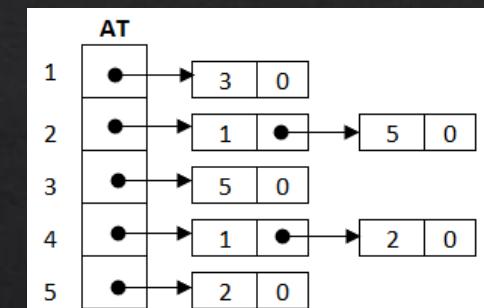
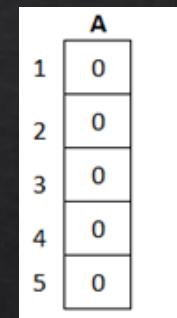
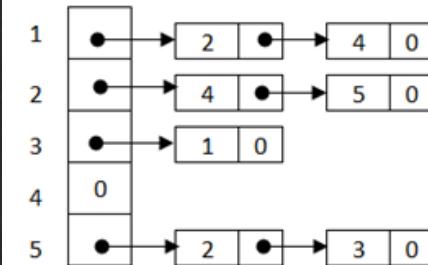
*AT pointer tömb feltöltése null értékkel.*

*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.*

Szomszédossági lista:



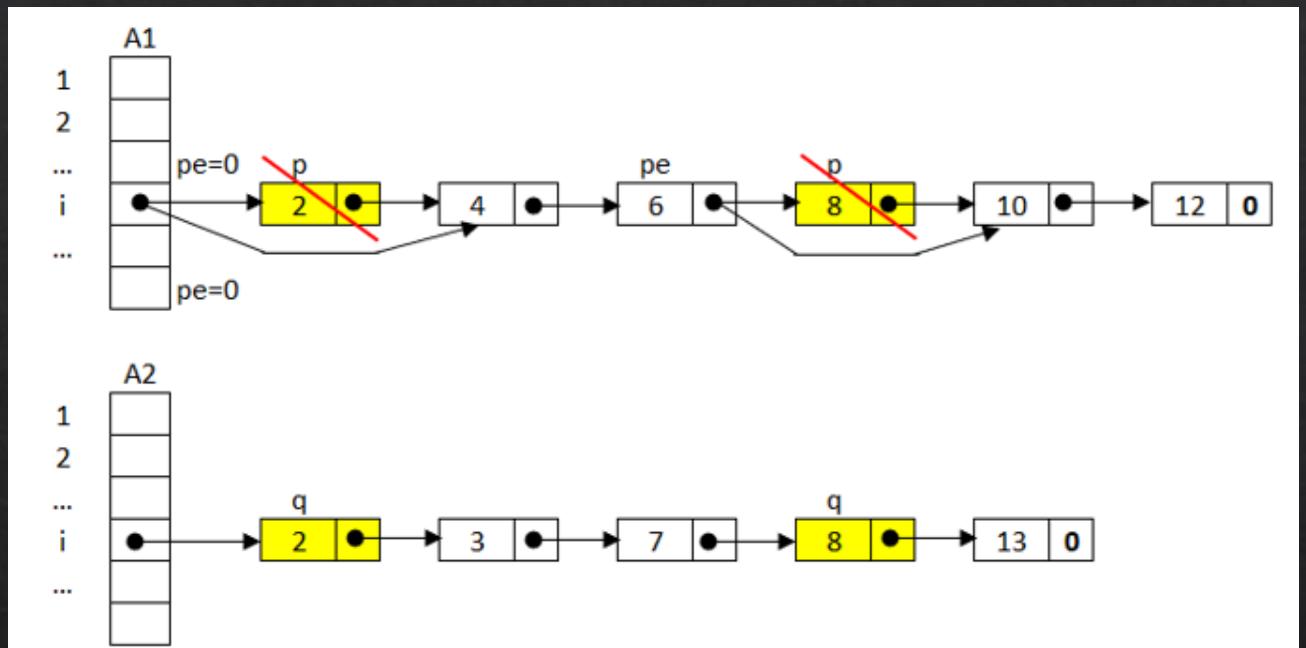
Műveletigény: az előző megoldáshoz hasonlóan könnyen látszik a  $\Theta(n+m)$  műveletigény.

## Különbség gráf

- ❖ Feladat:
- ❖ Adott két irányított gráf  $G_1$  és  $G_2$  szomszédossági listás ábrázolásban,  $G_1$  az  $A_1: \text{Edge}^*[n]$ ,  $G_2$  az  $A_2: \text{Edge}^*[n]$  tömbben. A két gráf csúcsai ugyanazok, az élek mások. Az éllisták csúcs szerint növekedően rendezett listák. Készítse el  $A_1$  tömbben a  $G_1 \setminus G_2$  gráfot.  $G_1 \setminus G_2$  gráf csúcsai ugyanazok, mint a két bemeneti gráfnak, élei pedig  $G_1$  élei közül azok, amelyek,  $G_2$  gráfban nem szerepelnek. A megoldásban használja ki, hogy az éllisták rendezettek! Műveletigény  $O(n^2)$
- ❖ Ötlet: mivel az éllisták rendezettek, a listák összefésülésével kapjuk meg a leghatékonyabb megoldást!

## Különbség gráf

- ❖ Megoldás szemléltetése az i-dik csúcsra:
- $A1[i]$  listán  $p$ ,  $A2[i]$  listán  $q$  pointerrel haladunk.
- Hárrom eset lehetséges :
  - $p \rightarrow v < q \rightarrow v$
  - $p \rightarrow v = q \rightarrow v$
  - $p \rightarrow v > q \rightarrow v$ )
- $A1[i]$  listából törlünk elemeket, a törléshez kell az előző listaelem elem címe, ez lesz majd  $pe$  pointerben. Ha az elsőt töröljük, akkor viszont  $A1[i]$  módosul!
- Ha bármelyik listán végig értünk, az összefésülés leállhat.



## Megoldás struktogramja

KülönbségGráf(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]

$p \neq 0 \wedge q \neq 0$

$p->v < q->v$

pe := p

$p := p->next$

$p->v = q->v$

r := p->next

pe = 0

A1[i]:=r      pe->next:=r

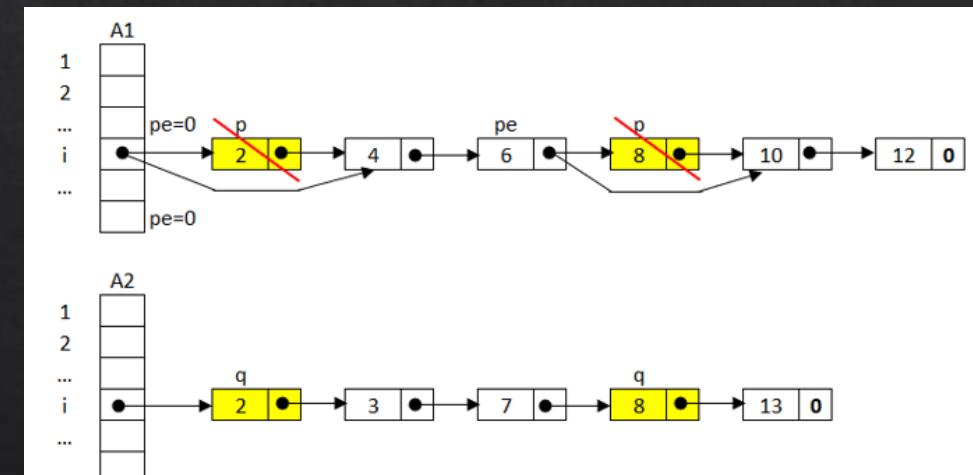
delete p

$p := r$

q := q->next

$p->v > q->v$

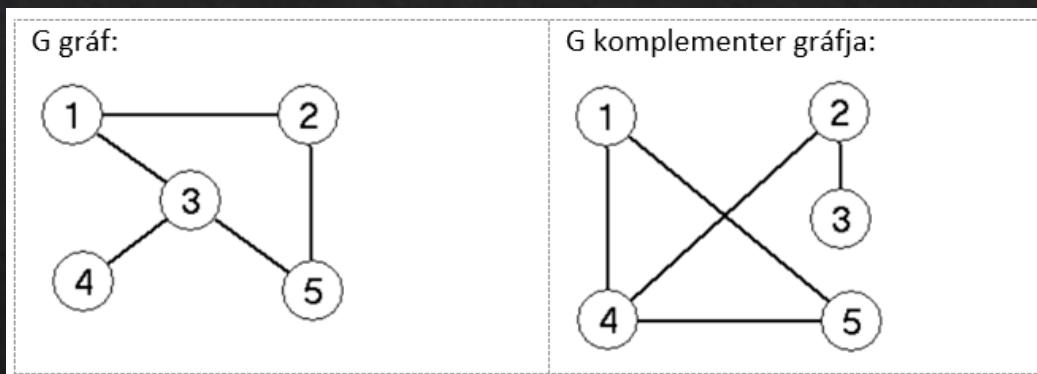
q := q->next



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

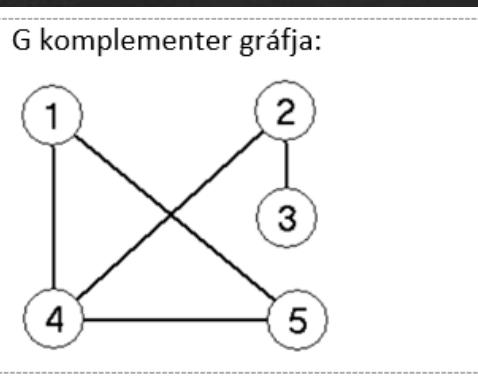
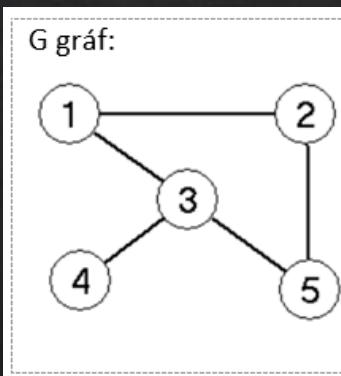
## Komplementer gráf készítése

- ❖ 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:  $O(n^2)$ ,  $O(n)$  segédmemória használható.
- ❖ Definíció: Valamely  $G=(V,E)$  irányítatlan gráf komplementer gráfja az a gráf, amelynek csúcshalmaza megegyezik a  $G$  gráf csúcshalmazával, az élhalmza pedig a  $G$  gráf élhalmazának a komplementer halmaza (a teljes gráf élhalmazára, mint alaphalmazra nézve).
- ❖ Megjegyzés: Hurokélt nem tartalmaznak a gráfok, ügyeljünk rá, hogy a komplementer gráfba se kerüljön be hurokél.



## 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 hurokélt ne hozzunk létre!

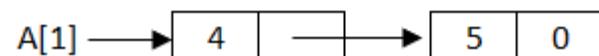


Komplementer gráfban az 1-es csúcs éleinek létrehozása:

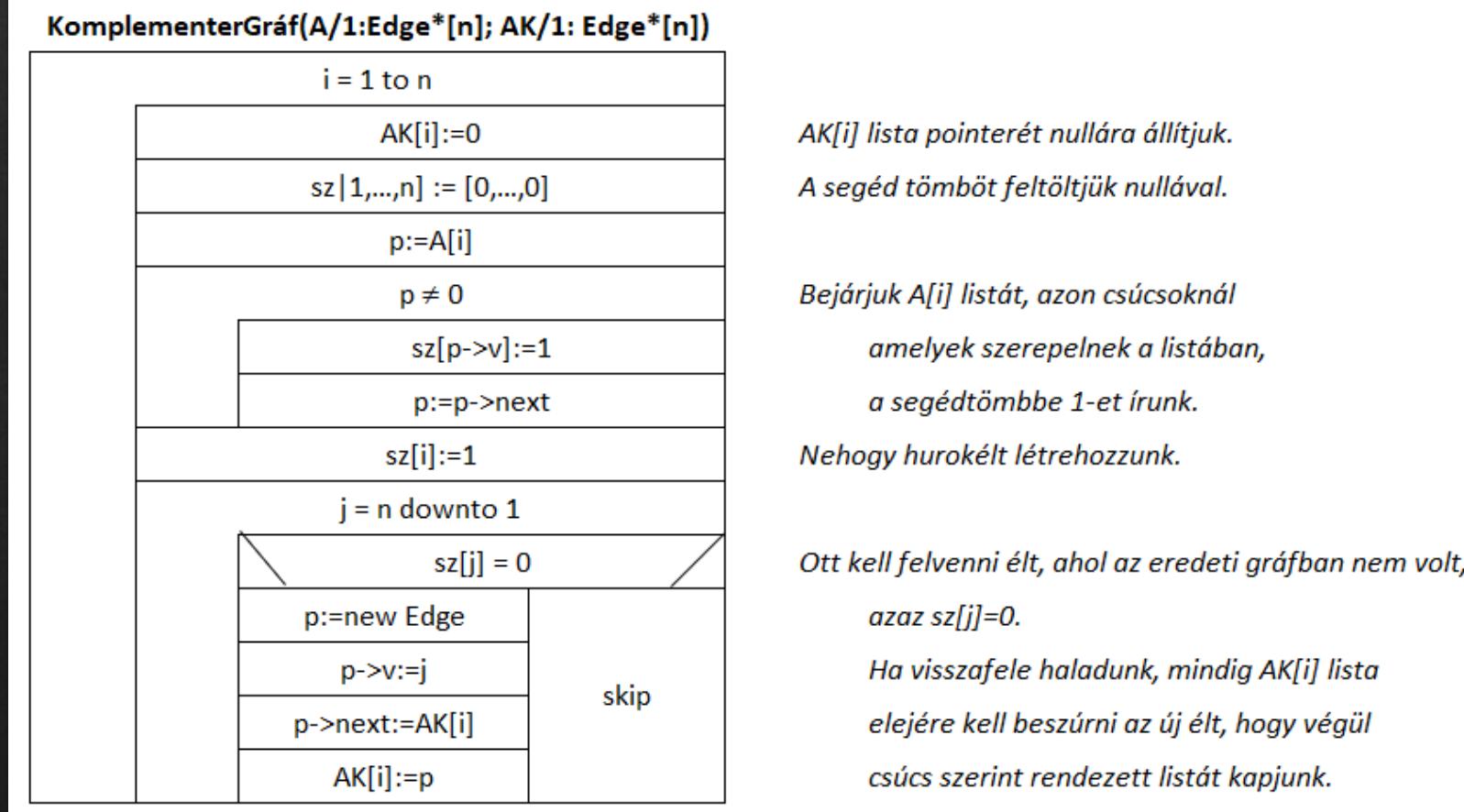
Segéd tömb feltöltése:

	1	2	3	4	5
sz:	0	1	1	0	0

Ahol nulla van, ott kell élt felvenni, kivéve a hurok élt!  
Hogy rendezett listát kapunk, a segéd tömb bejárása  
n..1 irányú lesz.

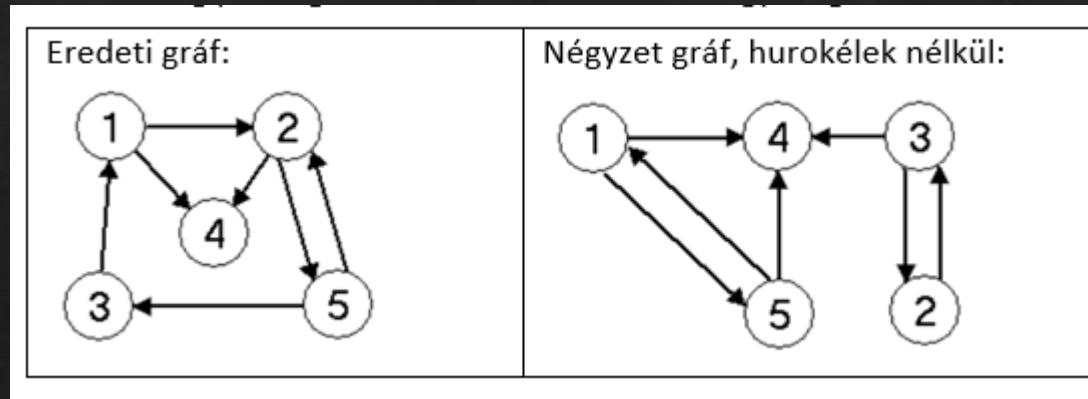


## A megoldás struktogramja



# Szorgalmi házi feladatok

- ❖  $G^2$  gráf előállítása 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.
- ❖  $G$  rát szomszédossági listával van ábrázolva, az A/1:Edge\*[n] tömbben. Az éllisták csúcs szerint rendezettek. Készítsük el hasonló ábrázolásban a  $G^2$  gráfot az A2 tömbben. Műveletigény:  $O(n*m)$  ( $|V|=n$ ,  $|E|=m$ )



# Szorgalmi házi feladatok

- ❖ Abszolút nyelő csúcs keresése 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!