Algoritmusok és adatszerkezetek I. 4. Előadás

Gráf ábrázolások: Szomszédossági csúcsmátrix és éllista, ezek tárigénye. Szélességi gráfkeresés.

Tartalom

Gráfelméleti alapfogalmak

- Gráfok használata
  - Pl. hálózatokat, folyamatokat modellezhetünk

1. Definíció. **Gráf** alatt egy $G = (V, E)$ rendezett párost értünk, ahol

- V a csúcsok (vertices) tetszőleges, véges halmaza,
- $E \subseteq V \times V \setminus \{(u; v) : u \in V\}$ pedig az élek (edges) halmaza.
- Ha $V = \{\}$, akkor üres gráfról,
- ha $V \neq \{\}$, akkor nemüres gráfról beszélünk.

- Megjegyzés:
  - Már a definíció szintjén kizárjuk a gráfokból párhuzamos éleket és a hurokéleket.
    - Nincs ugyanis semmilyen eszközünk arra, hogy két $(u; v)$ élet megkülönböztessünk (párhuzamos élek),
    - az $(u; u)$ alakú, ún. hurokéleket pedig expliciten kizártuk
  - ➢ Így a továbbiakban gráf alatt tulajdonképpen **egyszerű gráfot** értünk.

Gráfelméleti alapfogalmak

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

3. Definíció. 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.

- Megjegyzés:
  - Az irányítatlan gráfoknál tetszőleges (u, v) éllel együtt (v, u) is a gráf éle, hiszen ez a két él egyenlő.
  - Irányított gráfoknál általában lesz a gráfnak olyan (u, v) éle, hogy ennek fordítottja (v, u) nem éle a gráfnak.

4. Definíció. A $G = (V, E)$ gráf csúcsainak (V) egy $<u_0, u_1, \ldots u_n>$ $(n \in \mathbb{N})$ sorozata a gráf egy **út**ja, 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.

Gráfelméleti alapfogalmak

5. Definíció. Tetszőleges $<u_0, u_1, \ldots u_n>$ út **rész-útja** $0 \le i \le j \le n$ esetén az $<u_i, u_{i+1}, \ldots u_j>$ út

- A **kör:** olyan út
  - kezdő és végpontja (csúcsa) azonos
  - a hossza > 0
  - az élei páronként különbözőek
- Az **egyszerű kör:** olyan kör
  - csak a kezdő és a végpontja azonos

Gráfelméleti alapfogalmak

- Tetszőleges **út akkor tartalmaz kört**
  - ha van olyan rész-útja, ami kör
- **Körmentes út:** olyan út
  - ami nem tartalmaz kört
- **Körmentes gráf** : olyan gráf
  - amiben csak körmentes utak vannak

- Megjegyzés:
  - Az utak köröket is tartalmazhatnak! A fentiek szerint tetszőleges kör hossza ≥ 2.

Gráfelméleti alapfogalmak

6. Definíció. **DAG** alatt irányított, körmentes gráfot értünk (directed acyclic graph).

- A DAG-ok modellezhetnek például összetett folyamatokat, ahol a gráf csúcsai elemi műveletek, az élei pedig az ezek közötti rákövetkezési kényszerek.

7. Definíció. Tetszőleges $G = (V, E)$ **irányított gráf irányítatlan megfelelője** az a $G' = (V, E')$ irányítatlan gráf, amire $E' = \{(u, v) : (u, v) \in E \vee (v, u) \in E\}$.

8. Definíció. A G **irányítatlan gráf összefüggő**, ha G tetszőleges csúcsából bármelyik csúcsába vezet út.
A G **irányított gráf összefüggő**, ha az irányítatlan megfelelője összefüggő.

9. Definíció. Az irányítatlan, körmentes, összefüggő gráfokat szabad fáknak, más néven **irányítatlan fák**nak nevezzük.

Gráfelméleti alapfogalmak

10. Definíció. Az u csúcs a G irányított gráf **generátor csúcsa**, ha u-ból a G tetszőleges v csúcsa elérhető, azaz létezik $u \rightsquigarrow v$ út.

11. Tulajdonság. Ha a G irányított gráfnak van generátor csúcsa, akkor összefüggő, de fordítva nem igaz az állítás.

12. Definíció. T **gyökeres fa**, más néven **irányított fa**, ha T olyan irányított gráf, aminek van generátor csúcsa, és a T irányítatlan megfelelője körmentes.
Ilyenkor a generátor csúcsot a fa gyökér csúcsának is nevezzük.

13. Tulajdonság. Tetszőleges (gyökeres vagy szabad) nemüres fának pontosan eggyel kevesebb éle van, mint ahány csúcsa.

14. Definíció. A $G = (V, E)$ gráfnak **részgráf**ja a $G' = (V', E')$ gráf, ha $V' \subseteq V \land E' \subseteq E$, és mindkét gráf irányított, vagy mindkettő irányítatlan.
A G gráfnak **valódi részgráfja** a G' gráf, ha G-nek részgráfja G′, de $G \neq G'$.

Gráfelméleti alapfogalmak

15. Definíció. Két (rész)gráf **diszjunkt**,

- ha nincs közös csúcsuk (és ebből következően közös élük sem)

16. Definíció. A G gráf **összefüggő komponense** a G' gráf,

- ha G-nek részgráfja G' és G' összefüggő
- de G-nek nincs olyan összefüggő részgráfja, aminek G' valódi részgráfja.

17. Tulajdonság. Tetszőleges gráf vagy összefüggő vagy felbontható (egymástól diszjunkt) összefüggő komponensekre (amelyek együtt kiadják a teljes gráfot)

18. Definíció. A G gráf **erdő**, ha összefüggő komponensei fák (vagy egyetlen fából áll).

19. Tulajdonság. A G irányítatlan gráf erdő $\Leftrightarrow$ G körmentes.
A G irányított gráf erdő $\Leftrightarrow$ G irányítatlan megfelelője körmentes,
és G mindegyik összefüggő komponensének van generátor csúcsa.

Gráfábrázolás

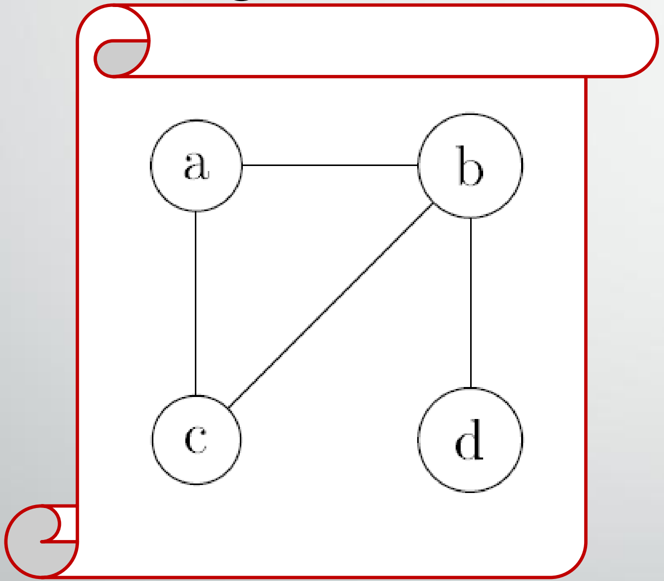
- Jelölések
  - $G = (V, E)$ gráfról általában föltesszük, hogy $V = \{v_1, . . ., v_n\}$, ahol $n \in \mathbb{N}$, azaz hogy a gráf csúcsait egyértelműen azonosítják az $1..n$ sorszámok.
  - a csúcsok sorszámait a szemléletesség kedvéért gyakran az angol ábécé kisbetűivel jelöljük
- Grafikus ábrázolás
  - Csúcsok: kis körök
  - Élek
    - irányított gráfoknál: a körök közti nyilak
    - irányítatlan esetben: a köröket összekötő vonalak
  - A csúcsok sorszámát (illetve az azt reprezentáló betűt) általában a körökbe írjuk

Gráfábrázolás

- Szöveges ábrázolás
  - irányítatlan gráfoknál: „u - v_{u1}; . . . v_{uk}." jelentése:
    - a gráfban az u csúcsnak szomszédai: v_{u1}; . . . v_{uk} csúcsok
    - Azaz: (u, v_{u1}), . . .; (u, v_{uk}) élei a gráfnak.
  - irányított gráfoknál: „u -> v_{u1}; . . . v_{uk}." jelentése:
    - a gráfban az u csúcsból (u, v_{u1}), . . .; (u, v_{uk}) irányított élek indulnak ki
    - Azaz: az u csúcs rákövetkezői (gyerekei): v_{u1}; . . . v_{uk} csúcsok

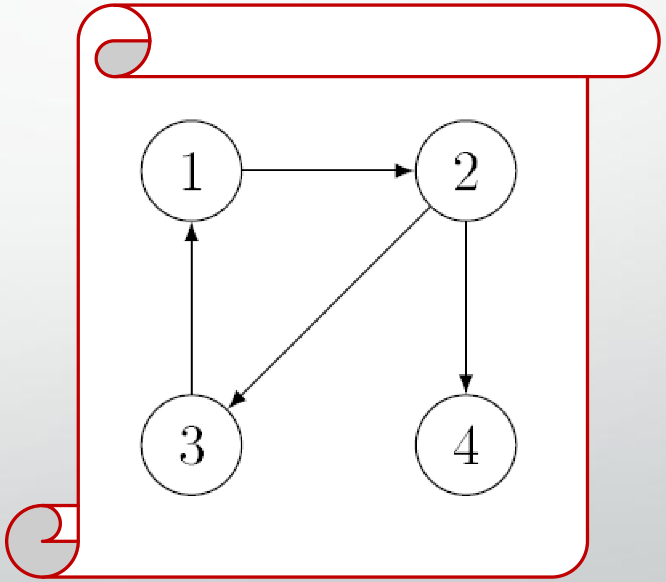
Gráfábrázolás példa

- Irányítatlan gráf grafikus és szöveges ábrázolása



a – b ; c.
b – c ; d.

- Irányított gráf grafikus és szöveges ábrázolása

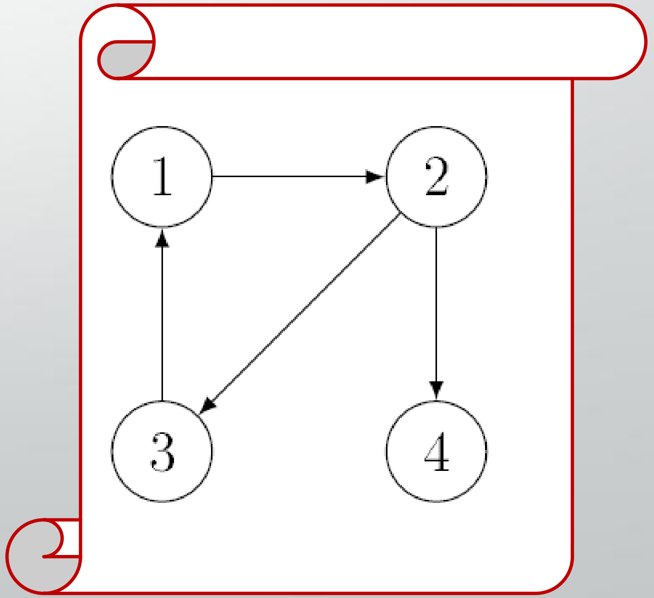


$1 \rightarrow 2.$
$2 \rightarrow 3\ ;\ 4.$
$3 \rightarrow 1.$

Szomszédossági mátrixos (adjacency matrix), más néven csúcsmátrixos reprezentáció

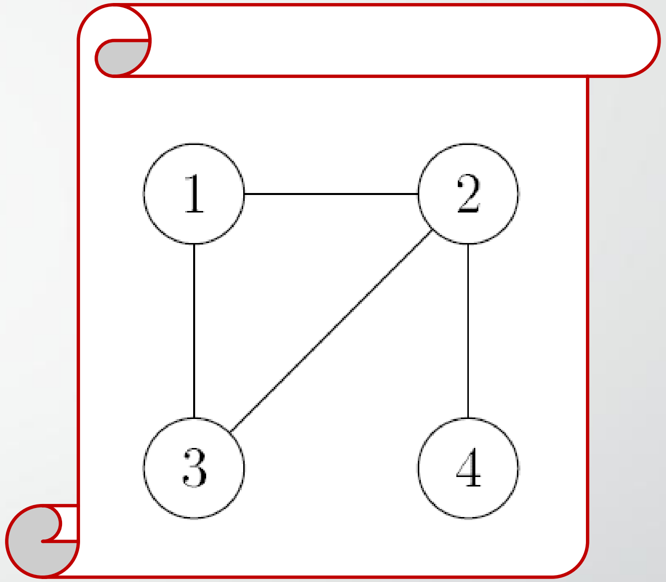
- A $G = (V, E)$ gráfot ($V = \{v_1, \ldots, v_n\}$) egy ***A*/1 : *bit*[*n*, *n*]** mátrix reprezentálja, ahol:
  - $n = |V|$ a csúcsok száma
  - 1..*n* a csúcsok sorszámai, azaz azonosító indexei,
  - **type** *bit* **is** {0, 1}
  - tetszőleges $i, j \in 1..n$ csúcssorszámokra
    - $A[i, j] = 1 \Leftrightarrow (v_i, v_j) \in E$
    - $A[i, j] = 0 \Leftrightarrow (v_i, v_j) \notin E$

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



Szomszédossági mátrixos reprezentáció

- A főátlókban mindig nullák vannak
  - ➢ csak egyszerű gráfokkal foglalkozunk (nincsenek hurokélek)
- Irányítatlan esetben a szomszédossági mátrixos reprezentáció mindig szimmetrikus
  - ➢ $(v_i, v_j) \in E \Leftrightarrow (v_j, v_i) \in E$
  - ➢ $\forall\ v_i$ és v_j csúcsokra $A[i,j] = A[j,i]$
- ➢ Elég alsóháromszög (felsőháromszög) mátrixban ábrázolni
  - Főátló nélkül
  - Pl. sorfolytonosan
  - Helyfoglalás:
    - 2. sor 1 elem, 3. sor 2 elem, … utolsó sor (n - 1) elem.
    - ➢ n^2 bit helyett csak 1 + 2 + … + (n - 1) = n * (n - 1)/2 bit



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

Mátrix- tömb megfeleltetés

- $A/1 : bit[n, n]$ mátrix -> $B: bit[n * (n - 1)/2]$ tömb
  - $a_{ij} = A[i, j]$ jelöléssel
  - $<a_{21}, a_{31}, a_{32}, a_{41}, a_{42}, a_{43}, \dots , a_{n1}, \dots , a_{n(n-1)}>$
- Innét
  - $A[i, j] = B[(i-1)*(i-2)/2+(j-1)]$ ha $i > j$ (az alsóháromszög mátrixban)
  - $A[i, j] = A[j, i]$ ha $i < j$ ($A[i, j]$ a felsőháromszög mátrixban)
  - $A[i, i] = 0$ ($A[i, i]$ a főátlón van)

$a_{ij} = A[i, j]$ elem helyének meghatározása a B tömbben

- Azt kell megszámolnunk, hogy hány elem előzi meg sorfolytonosan az a_{ij} elemet a B tömbben
- a_{ij} indexe a B-ben = az a_{ij} -t megelőző elemek számával
  - Mivel a B tömböt nullától indexeljük
- Az alsóháromszög mátrixban az a_{ij} elemet az alábbi elemek előzik meg sorfolytonosan:

$$
\begin{array}{l}
a_{21},\\
a_{31}, a_{32},\\
a_{41}, a_{42}, a_{43},\\
\ldots,\\
a_{(i-1)1}, \ldots, a_{(i-1)(i-2)},\\
a_{i1}, \ldots, a_{i(j-1)}
\end{array}
$$

- Ez pedig összesen $(1+2+3+\ldots+(i-2))+(j-1) = (i-1)*(i-2)/2+(j-1)$ elem

Műveletidő csúcsmátrixos ábrázolásnál

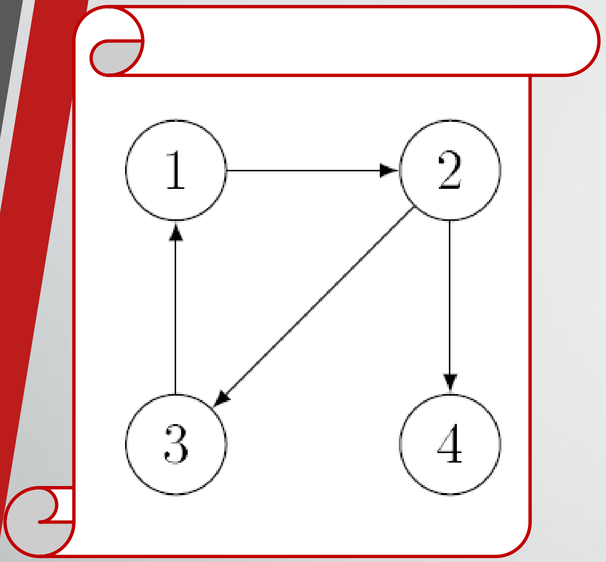
- $(v_i, v_j) \in E$ kérdés: $\Theta(1)$
  - Olyan algoritmusoknál előnyös, ahol gyakori ez a művelet
- Adott csúcs (irányított gráfoknál) gyerekeinek, vagy (irányítatlan gráfoknál) szomszédainak felsorolása: *n* lépés
  - Ez általában lényegesen több, mint ahány gyerek vagy szomszéd ténylegesen van

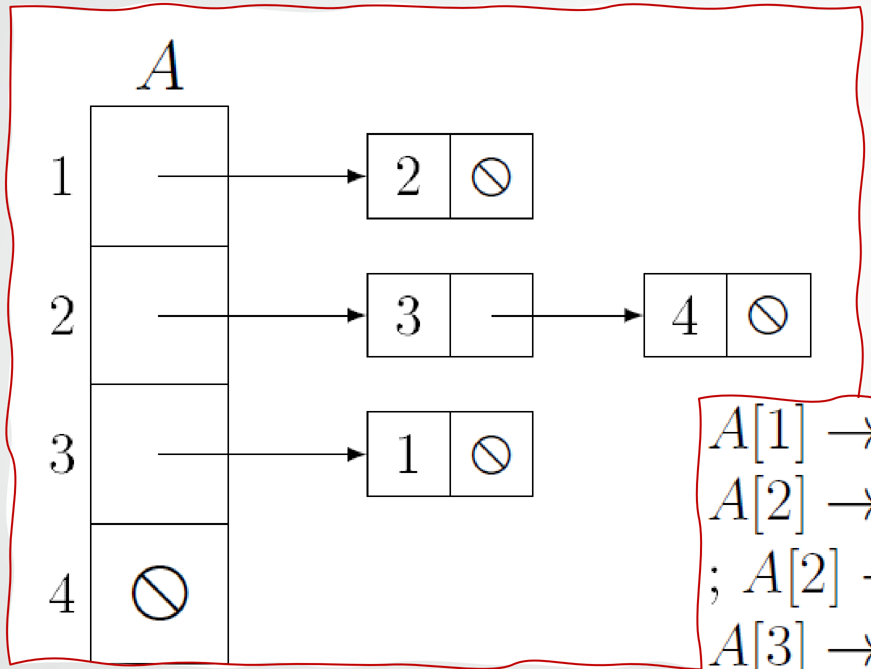
Szomszédossági listás (adjacency list) reprezentáció

$Edge$
$+v : \mathbb{N}$ $+next : Edge*$

- *A* $G = (V, E)$ gráfot ($V = \{v_1, \ldots, v_n\}$) ***A*/1 : *Edge**[*n*]** pointertömb reprezentálja, ahol:
  - Irányítatlan gráf esetében a v_i csúcs szomszédjainak sorszámait az *A*[*i*] S1L tartalmazza (i ∈1..*n*)
    - A v_i csúcs szomszédjainak indexeit tehát az *A*[*i*] lista elemeinek *v* attribútumai tartalmazzák
    - Így az *A*[*i*] lista elemei a v_i csúcshoz kapcsolódó éleknek felelnek meg
    - ➢ Minden élet kétszer ábrázolunk,
      - pl. a v_i csúcsnak szomszédja v_j -> a v_j csúcsnak is szomszédja v_i
  - Irányított gráfok esetén hasonló a reprezentáció, de az *A*[*i*] S1L csak az v_i csúcs gyerekeinek (más néven közvetlen rákövetkezőinek) sorszámait tartalmazza (i ∈1..*n*)
    - ➢ Mindegyik élet csak egyszer kell ábrázolnunk

Szomszédossági lista példa





$A[1] \to v = 2$; $A[1] \to next = \oslash$
$A[2] \to v = 3$; $A[2] \to next \to v = 4$
; $A[2] \to next \to next = \oslash$
$A[3] \to v = 1$; $A[3] \to next = \oslash$
$A[4] = \oslash$

- A szomszédossági listás ábrázolásnál S1L-ek helyett természetesen másfajta listákat is alkalmazhatunk.

Műveletidő szomszédossági listás ábrázolásnál

- $(v_i, v_j) \in E$ kérdés
  - meg kell keresnünk a j indexet az $A[i]$ listán
  - ➢ algoritmusoknál, ahol gyakori ez a művelet: érdemes a csúcsmátrixos reprezentációt választani
- Adott csúcs (irányított gráfoknál) gyerekeinek, vagy (irányítatlan gráfoknál) szomszédainak felsorolása
  - pontosan annyi lépésre van szükségünk, mint ahány gyerek vagy szomszéd ténylegesen van
  - A legtöbb gráfalgoritmusnak ez a leggyakoribb művelete
    - ➢ ezt a reprezentációt gyakran előnyben részesítjük a többi ábrázolással szemben.

Számítógépes gráfábrázolások tárigénye

- A $G = (V, E)$ gráfban:
  - csúcsok száma: $n = |V|$
  - élek száma: $m = |E|$
  - $0 \leq m \leq n * (n - 1) \leq n^2$

➢ $m \in O(n^2)$

- Gráfok osztályozása:
  - ritka gráf (sparse graph):
    - $m \in O(n)$
  - sűrű gráf (dense graph):
    - $m \in (n^2)$

Számítógépes gráfábrázolások tárigénye

- **Szomszédossági mátrixos** (csúcsmátrixos) ábrázolás tárigénye
  - Alapesetben n^2 bit
  - Irányítatlan gráfoknál, csak az alsóháromszög mátrixot tárolva: $n * (n - 1)/2$ bit
    - $n * (n - 1)/2 \in \Theta(n^2)$ ->

➢ Az aszimptotikus tárigény mindkét esetben $\in$ **$\Theta(n^2)$**

- **Szomszédossági listás** reprezentációnál
  - A pointertömb n db mutatóból áll
  - A szomszédossági listáknak pedig összesen m vagy $2 * m$ elemük van
    - (a gráf irányított vagy irányítatlan)

➢ Az aszimptotikus tárigény mindkét esetben **$\Theta(n + m)$**

Számítógépes gráfábrázolások tárigénye

- Ritka gráfoknál
  - (a gyakorlati alkalmazások többségénél)
  - $m \in O(n) \rightarrow \Theta(n + m) = \Theta(n)$
  - ➢ A szomszédossági listás reprezentáció tárigénye aszimptotikusan kisebb, mint a csúcsmátrixosé

- Sűrű gráfoknál
  - $m \in \Theta(n^2) \rightarrow \Theta(n + m) = \Theta(n^2)$
  - ➢ szomszédossági listás és a csúcsmátrixos reprezentáció tárigénye aszimptotikusan ekvivalens

- Teljes gráfoknál
  - A szomszédossági listáknak összesen $n*(n-1)$ elemük van
  - Egy-egy listaelem sok bitből áll

➢ Teljes vagy közel teljes gráfoknál a szomszédossági listás ábrázolás tényleges tárigénye jelentősen nagyobb lehet, mint a csúcsmátrixosé, ahol a mátrix egy-egy eleme akár egyetlen biten is elfér.

Melyik ábrázolást válasszuk?

Szomszédossági lista

nagy

ritka

Szomszédossági mátrix

kicsi

sűrű

Az absztrakt halmaz, az absztrakt sorozat és az absztrakt gráf típus

- Típuskonstruktorok bevezetése: ($\mathcal{T}$ tetszőleges típus)
  - $\mathcal{T}\{\}$: $\mathcal{T}$ típusú elemek tetszőleges, véges halmaza
  - $\mathcal{T}\langle\rangle$: $\mathcal{T}$ típusú elemek tetszőleges, véges sorozata
- *u* **from** *S* művelet: (*S* tetszőleges nemüres halmaz)
  - kiválasztjuk az *S* halmaz egy tetszőleges elemét,
  - *u*-nak értékül adjuk,
  - majd eltávolítjuk *S*-ből
- Az üres halmaz: önmagában álló {}

Az absztrakt halmaz, az absztrakt sorozat és az absztrakt gráf típus

- Sorozatok:
  - a matematikában szokásos módon, egytől indexeljük
  - az indexet alsó indexként jelöljük
  - ha $u; v : \mathcal{T}<>$, akkor az $u + v$ kifejezés a konkatenáljukat jelöli
  - $<>$ önmagában az üres sorozat
- A halmaz és a sorozat típusú változókat is deklarálni kell
  - $s : \mathcal{T} <> \rightarrow s$ üres sorozattal inicializálódik
  - $h : \mathcal{T}\,\{\} \rightarrow h$ üres halmazzal inicializálódik
  - Ha a zárójelek között megadunk - pontosvesszőkkel elválasztva - néhány $\mathcal{T}$ típusú elemet $\rightarrow$ a halmaz illetve sorozat a deklaráció kiértékelődése után ezeket fogja tartalmazni

Az absztrakt halmaz, az absztrakt sorozat és az absztrakt gráf típus

- L! $\mathcal{V}$: (vertex, azaz csúcs) absztrakt típus
  - A gráfok csúcsainak absztrakt típusa (A gráfok absztrakt algoritmusainak leírásához)
  - Mindegyik csúcshoz tetszőlegesen sok, névvel jelölt címke társítható ($name(v)$)
  - Mindegyik címkéhez tartozik valamilyen érték
    - értékadása: $name(v) := x$
    - a *name* címke (azaz a $\mathcal{V}$ halmazon értelmezett parciális függvény)
      a $name(v) := x$ értékadással a v csúcsnál az x értéket veszi fel.
- A $\mathcal{V}$ halmazt az algoritmusok implementációiban legtöbbször az $\mathbb{N}$ halmaz reprezentálja
  - egy n csúcsú gráf csúcsait pedig egyszerűen az $1..n$ vagy a $0..(n-1)$ halmaz
    - a tömböket egytől vagy nullától kezdve indexeljük

Az absztrakt halmaz, az absztrakt sorozat és az absztrakt gráf típus

- A címkéket ui. gyakran tömbök reprezentálják
  - A láthatóságuk, a hatáskörük és az élettertamuk is korlátozott
  - Az ebből fakadó problémákat az absztrakt algoritmus implementálásakor kell megoldani
- Az értékkel bíró címkék mellett használni fogunk egyszerű címkéket is, amikor a gráfok csúcsaihoz és/vagy éleihez egyszerű számértékeket vagy neveket társítunk

- $\mathcal{E}$: élek
- $\mathcal{G}$: élsúlyozatlan absztrakt gráfok
  - Egyszerű gráfok (nem tartalmaznak sem párhuzamos, sem hurokéleket)

$\mathcal{E}$
$+\ u, v : \mathcal{V}$

$\mathcal{G}$
$+\ V : \mathcal{V}\{\}$ $+\ E : \mathcal{E}\{\}\ //\ E \subseteq V \times V \setminus \{(u,u) : u \in V\}\ //\ \text{edges}$

Elemi gráfalgoritmusok

- Elemi gráfalgoritmusok:
  - Élsúlyozatlan gráfokon értelmezett algoritmusok
  - Szélességi gráfkeresés
- Élsúlyozatlan gráfokban tetszőleges út hossza = az út mentén érintett élek száma
- Az út tartalmazhat kört
- Irányított/irányítatlan gráf:
  - irányított gráf: Ha a gráfban tetszőleges u és v csúcsokra az $(u; v)$ élt megkülönböztetjük a $(v; u)$ éltől
  - irányítatlan gráf: ezeket definíció szerint egyenlőknek tekintjük

Szélességi gráfkeresés (BFS: Breadth-first Search)

- Értelmezése:
  - irányított és irányítatlan gráfok
- Feladata:
  - Meghatározza a start csúcsból (*s*)
  - a gráf minden, *s*-ből elérhető csúcsába
  - a legkevesebb élet tartalmazó utat
  - (ha több ilyen van, akkor az egyiket)
- Élsúlyozatlan gráfokban ezeket tekintjük
  - az *s*-ből induló *legrövidebb*, (*optimális*) utaknak

Szélességi gráfkeresés (BFS: Breadth-first Search)

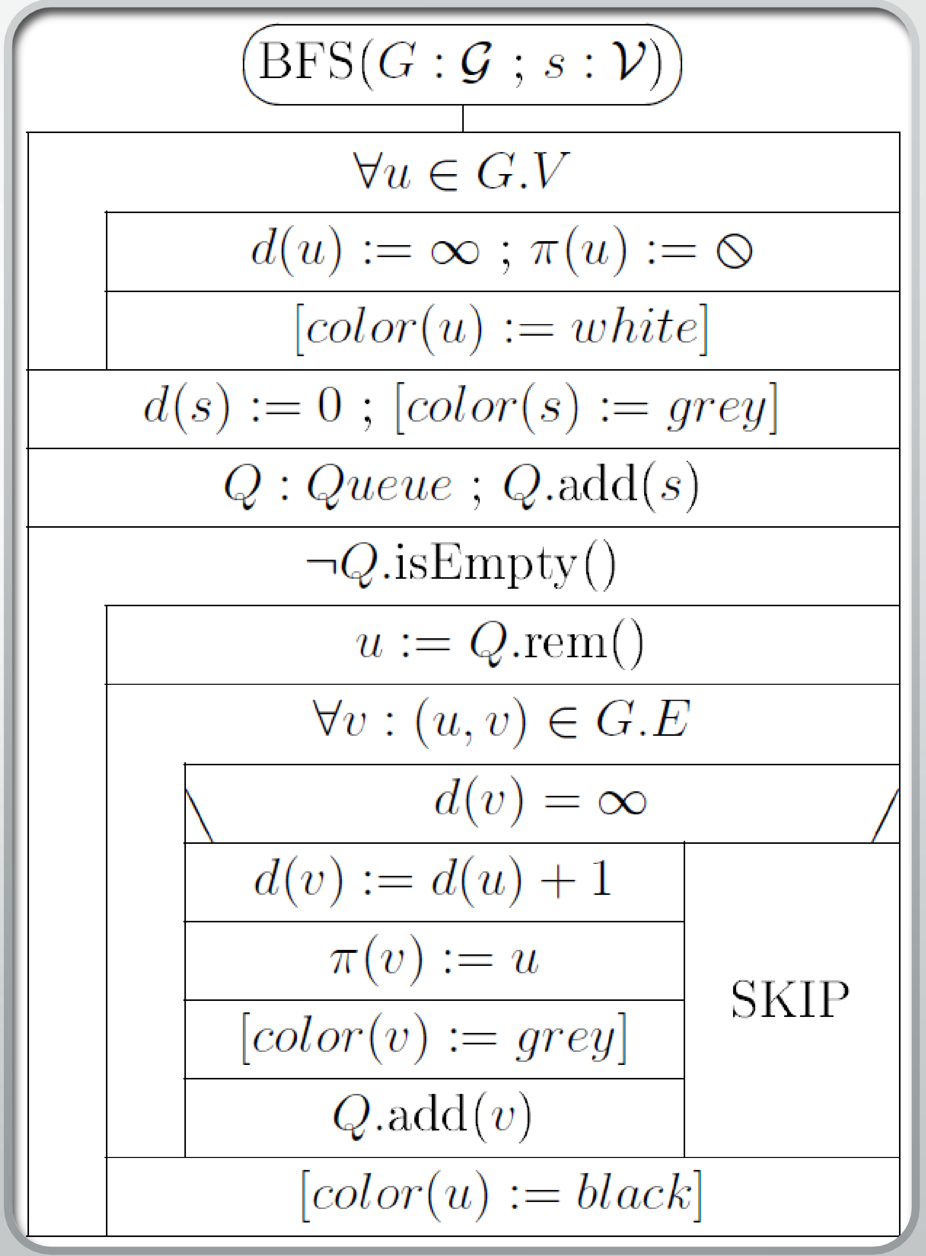
- A csúcsok fontosabb címkéi:
  - *d* - a megtalált úttal hány élen keresztül jutunk a csúcsba
  - *π* - melyik csúcsból jutunk közvetlenül a csúcsba (ki a szülője)
  - *color* - csak szemléletes tartalmuk van:
    - a **fehér** csúcsokat a gráfbejárás/keresés még nem találta meg
    - a **szürke** csúcsokat már megtalálta, de még nem dolgozta fel
    - a **fekete** csúcsokkal viszont már nincs további tennivalója
    - az értéke nem befolyásolja a program futását
      - a rá vonatkozó utasítások az algoritmusból elhagyhatók

Szélességi gráfkeresés (BFS: Breadth-first Search)

- Az *s*-től *k* távolságra levő csúcsok vannak a gráf *k*-adik szintjén
- ➢ BFS a gráfot szintenként járja be
  - először a nulladik szintet, aztán az elsőt, majd a másodikat stb.
- Minden szintet teljesen feldolgoz, mielőtt a következőre lépne, közben pedig éppen a következő szinten levő csúcsokat találja meg.
- ➢ Innét jönnek a szélességi bejárás és a szélességi keresés elnevezések.
- Ha az *u* csúcs *s*-ből nem érhető el $\rightarrow d(u) = \infty$ és $\pi(u) = \emptyset$
- $\pi(s) = \emptyset$
  - a legrövidebb $s \rightsquigarrow s$ út csak az *s* csúcsot tartalmazza
  - nincs benne él
  - ➢ *s*-nek szülője sincs

Szélességi gráfkeresés

- *Q*: Azok a csúcsok, amiket már elértünk, de még a gyerekeiket nem néztük meg
- A második, azaz a fő ciklus addig fut, amíg van már elért, de még fel nem dolgozott csúcs.
    - Kiveszi a sorból az *u* csúcsot és kiterjeszti
        - Ha valamelyik *v* gyerekcsúcsra az (*u*; *v*) él feldolgozásakor már d(*v*) $\neq \infty$
            - ➢ ez a csúcs már korábbról ismert
            - ➢ az újonnan *v*-be talált út ennél biztosan nem rövidebb
            - ➢ figyelmen kívül is hagyja
        - Ha $d(v) = \infty$
            - beállítjuk a címkéiket
            - a sor végére tesszük őket
        - Mire az *s*-től *k* távolságra levő csúcsokat feldolgozzuk, a sort már az *s*-től *k* + 1 távolságra levő csúcsok alkotják

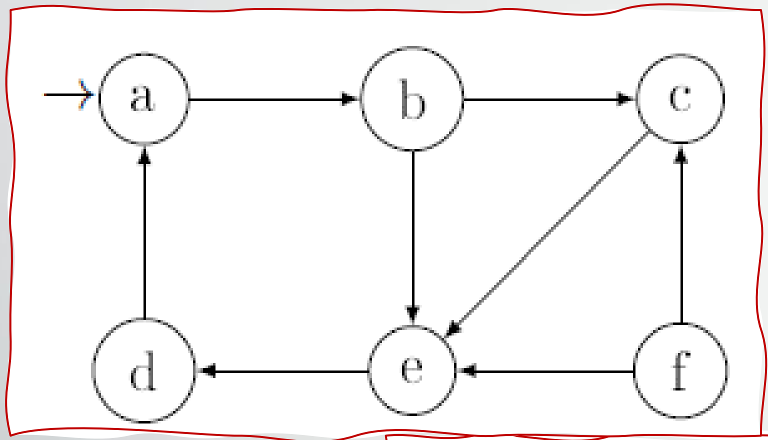


A szélességi fa (Breadth-first Tree)

- A BFS minden *s*-ből elérhető, de tőle különböző csúcsra, annak π címkéjével hivatkozik annak szülőjére, az *s*-ből a csúcsba vezető (a BFS által meghatározott) legrövidebb úton
- Több csúcsnak is lehet ugyanaz a szülője, a szülőcsúcs viszont a BFS által meghatározott legrövidebb utakon egyértelmű
- Az *s*-ből elérhető csúcsok π hivatkozásai -> egy általános fát definiálnak
  - a gyökere *s* -> $\pi(s) = \varnothing$
  - ***szélességi fának*** és ***legrövidebb utak fájának*** is nevezzük
    - fordított irányban mindegyik, az *s*-ből elérhető csúcsra a BFS által meghatározott legrövidebb utakat tartalmazza
    - Fordított ábrázolás célszerűsége
      - minden csúcsnak legfeljebb egy szülője, viszont több gyereke is lehet
      - ➢ tömörebb ábrázolás érhető el

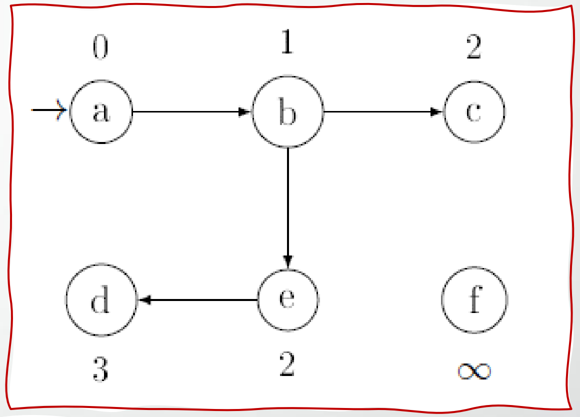
A szélességi gráfkeresés szemléltetése

- Most is és a későbbiekben is, indeterminisztikus esetekben a kisebb indexű csúcsot részesítjük előnyben.
- Kezdő csúcs: a



a → b.
b → c ; e.
c → e.
d → a.
e → d.
f → c ; e.

A gráf szélességi fája:

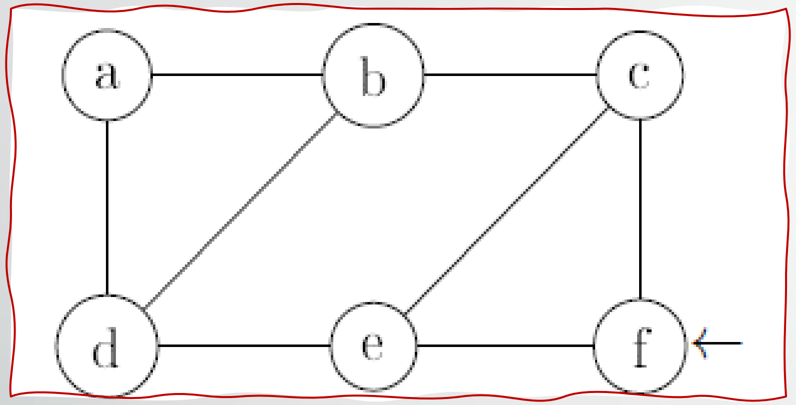


ex-pand	changes of d and π						Q : Queue
	a	b	c	d	e	f	
$d\,\mathcal{V}$	0 ⦸	∞ ⦸	∞ ⦸	∞ ⦸	∞ ⦸	∞ ⦸	⟨ a ⟩
0 a		1 a					⟨ b ⟩
1 b			2 b		2 b		⟨ c, e ⟩
2 c							⟨ e ⟩
2 e				3 e			⟨ d ⟩
3 d							⟨ ⟩

A szélességi gráfkeresés szemléltetése

- Most is és a későbbiekben is, indeterminisztikus esetekben a kisebb indexű csúcsot részesítjük előnyben.
- Start csúcs: f

A gráf szélességi fája:



a – b ; d.
b – c ; d.
c – e ; f.
d – e.
e – f.

Szélességi gráfkeresés hatékonysága

- L! $n = |G.V|$ és $m = |G.E|$
  - Az első (az inicializáló ciklus) n-szer iterál
  - A második, a fő ciklus annyiszor iterál, ahány csúcs elérhető s-ből (önmagát is számítva)
    - legfeljebb n
    - minimum 1
    - ➢ a belső ciklus legfeljebb m-szer iterál összesen
      - Maximum: ha s- ből mindegyik csúcs elérhető -> minden él sorra kerül
      - Minimum: ha s-ből nem megy ki egyetlen él sem ->egyszer sem

$$\sum: MT(n,m) \in \Theta(n+m) \text{ és } mT(n,m) \in \Theta(n)$$

Szélességi gráfkeresés implementációja szomszédossági listás és szomszédossági mátrixos gráfábrázolás esetén

- A $G = (V, E)$ gráfról föltesszük, hogy $V = \{v_1, \ldots, v_n\}$, ahol $n \in \mathbb{N}$
  - azaz a gráf csúcsait egyértelműen azonosítják az $1..n$ sorszámok
- Az absztrakt v_i csúcsok ábrázolása:
  - d és π címkéinek megfeleltetjük a $d/1$; $\pi/1 : \mathbb{N}\,[n]$ tömböket
    - ahol $d(v_i)$ reprezentációja $d[i]$ és $\pi(v_i)$ reprezentációja $\pi[i]$

Szélességi gráfkeresés implementációja szomszédossági listás és szomszédossági mátrixos gráfábrázolás esetén

- A *color* címkék reprezentálása felesleges
- A $\varnothing$ reprezentációja lehet például a 0 számérték
  - pl. $\pi[s]$ = 0 absztrakt jelentése $\pi(v_s) = \varnothing$
- BFS(A/1 : Edge*[n] ; s : 1..n ; d/1; π /1 : $\mathbb{N}$ [n]) és
  BFS(A/1 : bit [n,n] ; s : 1..n ; d/1; π /1 : $\mathbb{N}$ [n])

Ellenőrző kérdések

1. A $G[1..n]$ egy irányított gráf szomszédossági listás ábrázolása.
   - Adja meg a listaelem típus leírását!
   - Írja meg a **transzponál**(G,n,GT) eljárást, ami előállítja a gráf transzponáltjának a szomszédossági listás reprezentációját!
   - Írja meg a **kibeFokok**(G,n,be,ki) eljárást, ami minden $u \in 1..n$ csúcsra a $be[u]$ –ban kiszámítja a csúcs bemeneti fokszámát, $ki[u]$ –ban pedig a csúcs kimeneti fokszámát!
   - Mindkét eljárásban $MT(n,m) \in \Theta(n + m)$, ahol m a gráf éleinek száma

Ellenőrző kérdések

2. Szélességi gráfkeresés

- Mit számol ki a Szélességi gráfkeresés?
- Adja meg az algoritmus absztrakt struktogramját!
- A Szélességi gráfkeresés a gráf mely csúcsaiba talál optimális utat, és a végrehajtás során mikor?
- Mit tud a Szélességi gráfkeresés műveletigényéről?
- Szemléltesse az algoritmust az **a** csúcsból indítva, az **a – b ;d. b – c;d. c-e. d-e.** Irányítatlan gráfon! Rajzolja le az eredményül adódó szélességi fát is!

Köszönöm a figyelmet!

Pusztai Kinga

A bemutató Ásványi Tibor: Algoritmusok és adatszerkezetek II. eladásjegyzet: Elemi gráfalgoritmusok alapján készült.