Algoritmusok és adatszerkezetek 2.

6. előadás – 2024. október 14.

Gráfok bejárása

- Gráfokkal kapcsolatos algoritmikus feladatokban gyakran van szükségünk arra, hogy az élek mentén lépdelve valamilyen szisztematikus módon végigjárjuk a gráf csúcsait.
- Az erre a célra való **bejáró algoritmusok** központi jelentőségűek a gráfalgoritmusok között.
- Bejáró módszerek adják a gráf-struktúrákon működő összetett algoritmusok vezérlési szerkezetét azzal, hogy szabályozzák a sorrendet, amely szerint a csúcsokhoz kötődő munkákat elvégezzük.
- Két általános, tetszőleges gráfra működő módszer: a **mélységi bejárás** vagy keresés (depth-first-search, DFS) és a **szélességi bejárás** vagy keresés (breadth-first-search, BFS).

Gráfok szélességi bejárása

- A mélységi bejáráshoz hasonlóan a feladat a gráf **csúcspontjainak** szisztematikus **feltérképezése** és közben az **élek kategorizálása**
- A lámpagyújtogató példájával: a lámpagyújtogató nem gyújt fel lámpát addig, amíg az aktuális csúcspontból látható összes fel nem gyújtott lámpát fel nem térképezi.
- A feltérképzett csúcsokat megjegyzi, és ezen **megjegyzett csúcsok közül választja ki a következő lámpát felgyújtásra**
- Látszik, hogy a szélességi bejárás esetében szükségünk van egy „memóriára”, ahol a feldolgozandó csúcspontokat tárolhatjuk
- A feladatok legegyszerűbb szervezését egy **sor** (FIFO) adatszerkezet biztosítja

A szélességi bejárás tehát egy sort (FIFO-listát) alkalmaz: Ebbe rakjuk be az éppen meglátogatott csúcs szomszédjait, hogy majd a megfelelő időben az ő meglátogatásukra is sort keríthessünk.

A módszer általános lépésének lényege, hogy

- vesszük a sor elején levő x csúcsot,
- x-et töröljük a sorból,
- betesszük a sorba (nyilván a sor végére) x azon y szomszédait, amelyeket eddig még nem láttunk.

Az algoritmus keretét adó $bejár$ eljárás tulajdonképpen ugyanaz, mint a mélységi bejárásé volt. A $bejárva[1{:}\,n]$ tömb szolgál itt is a már látott csúcsok megjelölésére.

bejár (* elvégzi a G irányított gráf szélességi bejárását *)

$v := 1$ **to** n **do**

bejárva$[v]$:= hamis ;

$v := 1$ **to** n **do**

if bejárva$[v]$ = hamis **then**

szb(v)

szb (v : csúcs)

bejárva$[v]$:= igaz ;

sorba(v, Q) ;

whil e Q nem üres do begin

$x :=$ első(Q) ;

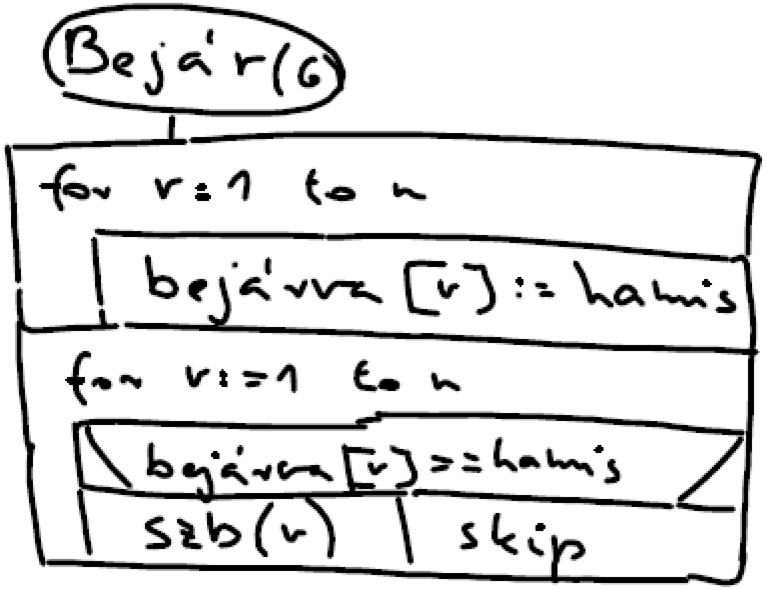
for minden $x \to y \in E$ élre **do**

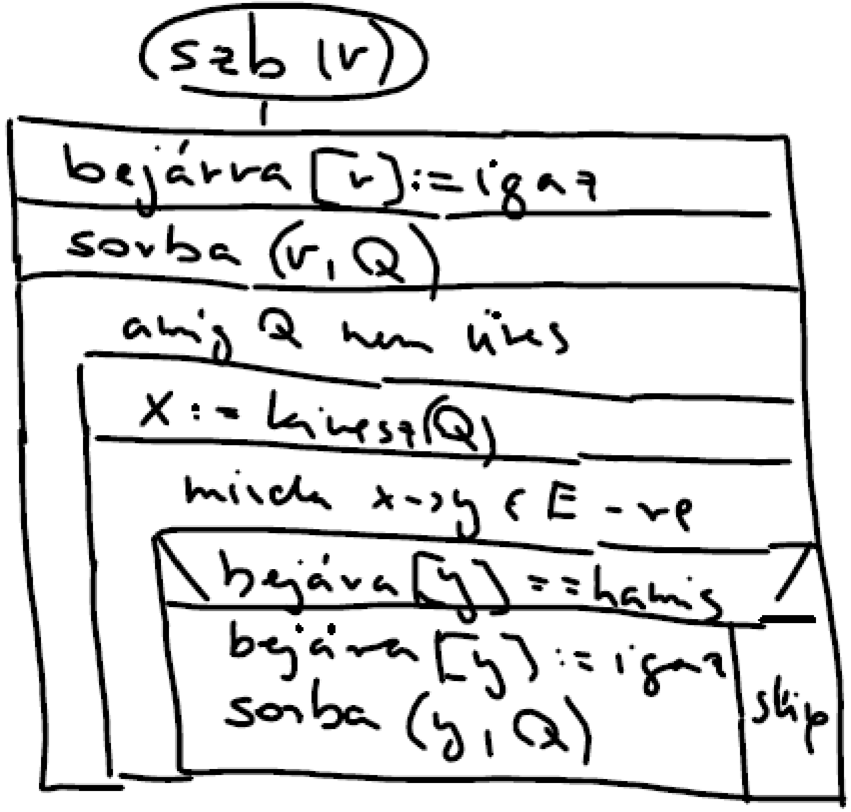
if bejárva$[y]$ = hamis **then begin**

bejárva$[y]$:= igaz;

sorba(y, Q)

end



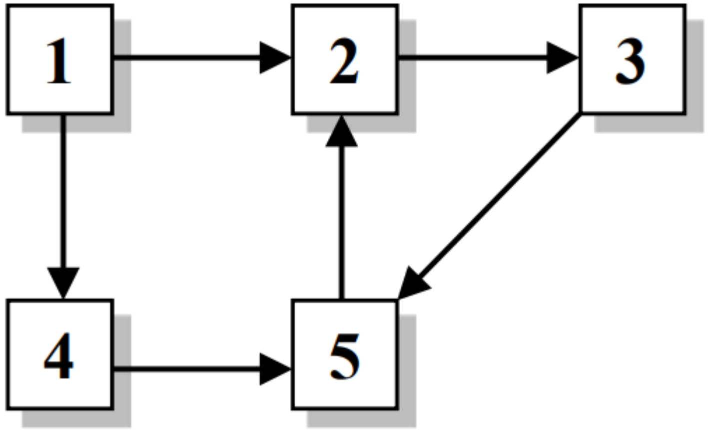


A szélességi bejárás eredménye egy **szélességi erdő** – ha a kezdőpontból minden csúcs irányított úton elérhető, akkor az erdő egyetlen szélességi fából áll.

Vegyük észre, hogy

- a szélességi bejárás egyetlen élet sem azonosít előreélként,
- ha minden éleket súlyozottnak tekintenénk egyforma súllyal, akkor a szélességi bejárás megoldja a legrövidebb utak problémáját a kiindulási (forrás) csúcsból,
- szokás ezért egy $d[1 \dots n]$ tömbben nyilvántartani, hogy egy csúcs milyen távol van a forrás csúcstól: egy csúcsból „meglátott” csúcsok eggyel nagyobb távolság értéket kapnak, mint az aktuális csúcs (ahonnan megláttuk őket)
- szokás továbbá egy $\Pi[1 \dots n]$ tömbben dokumentálni a szélességi fa felépítését: az meglátott csúcsokhoz az aktuális csúcs sorszámát jegyezzük fel a tömbben

Szélességi bejárás - példa



	S	ELÉRT	d					Π				
			1	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**4**	**5**
1	1	1	0	∞	∞	∞	∞	*NIL*	*NIL*	*NIL*	*NIL*	*NIL*
2	2, 4	1, 2, 4	0	**1**	∞	**1**	∞	*NIL*	**1**	*NIL*	**1**	*NIL*
3	4, 3	1, 2, 3, 4	0	1	**2**	1	∞	*NIL*	1	**2**	1	*NIL*
4	3, 5	1, 2, 3, 4, 5	0	1	2	1	**2**	*NIL*	1	2	1	**4**
5	5	1, 2, 3, 4, 5	0	1	2	1	2	*NIL*	1	2	1	4
6		1, 2, 3, 4, 5	0	1	2	1	2	*NIL*	1	2	1	4

Legrövidebb utak problémája egy forrásból

- Adott egy $G = (V, E)$ irányított gráf a $c(f)$, $f \in E$ élsúlyokkal.
- Kérdés, hogy mekkora a „legrövidebb” (legkisebb összsúlyú) út
  - egy adott pontból egy másik adott pontba vagy
  - egy adott pontból az összes többibe vagy
  - bármely két pont között.
- Meglepő lehet: az első két probléma ugyanolyan nehéz

A probléma pontos megfogalmazása:

- A G gráf egy u-t v-vel összekötő (nem feltétlenül egyszerű) $u \hookrightarrow v$ irányított útjának a **hossza** az úton szereplő élek súlyainak összege.
- **Legrövidebb** $u \hookrightarrow v$ **úton** egy olyan $u \hookrightarrow v$ utat értünk, amelynek a hossza minimális a G-beli $u \hookrightarrow v$ utak között.
- Az u és v csúcsok (G-beli) $d(u, v)$ **távolsága**:
  - 0, ha $u = v$;
  - ∞, ha nincs $u \hookrightarrow v$ út;
  - egyébként pedig a legrövidebb $u \hookrightarrow v$ út hossza.
- (Vigyázat, itt u és v nem felcserélhető: ha az egyik csúcs valamilyen távol van a másiktól, akkor nem biztos, hogy a másik is ugyanolyan távol van az egyiktől!)

Ha u és v egy olyan irányított körön vannak, amelynek az összsúlya negatív, akkor ezen körözve akármilyen kicsi (azaz nagy abszolút értékű negatív) úthosszat elérhetünk. *Természetes kikötés tehát, hogy G ne tartalmazzon negatív összsúlyú irányított kört.*

Nemnegatív élsúlyok esete

Jelöljünk ki a G gráfban egy $s \in V$ csúcsot forrásnak. Célunk, hogy az $u \in V$ pontoknak az s-től való távolságát meghatározzuk. Tegyük fel továbbá, hogy a $c(f)$ élsúlyok nemnegatívak. Ekkor nyilván nincs az előbbi értelemben vett negatív kör.

Ekkor a **legrövidebb utak problémája (egy forrásból)**:

Adott egy $G = (V, E)$ irányított gráf, a $c: E \to R^+$ nemnegatív értékű súlyfüggvény, és egy $s \in V$ csúcs (a forrás). Határozzuk meg minden $v \in V$-re a $d(s, v)$ távolságot.

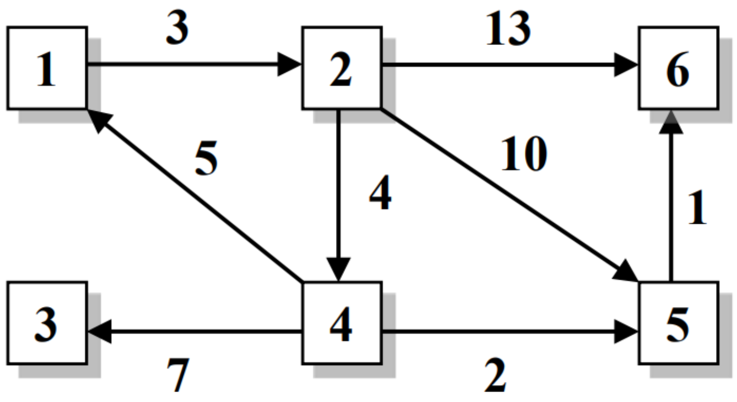
Dijkstra módszere

- Egy a G csúcsaival indexelt $d[\,]$ tömböt használunk.
- A $d[v]$ értékre úgy gondolhatunk, hogy az minden időpillanatban az eljárás során addig megismert legrövidebb $s \hookrightarrow v$ utak hosszát tartalmazza.
- A $d[v]$ mennyiség mindenkor felső közelítése lesz a keresett $d(s,v)$ távolságnak. A közelítést lépésről lépésre finomítva végül a kívánt értékeket kapjuk.
- Tegyük fel, hogy a G gráf az alábbi alakú C szomszédsági mátrixával adott:

$$C[v,w] = \begin{cases} 0 & \text{ha } v = w, \\ c(v,w) & \text{ha } v \neq w \text{ és } (v,w) \text{ éle } G\text{-nek}, \\ \infty & \text{különben.} \end{cases}$$

- Kezdetben $d[v] := C[s, v]$ minden $v \in V$ csúcsra.
- Válasszuk ki ezután az s csúcs szomszédai közül a hozzá legközelebbit, vagyis egy olyan $x \in V \setminus \{s\}$ csúcsot, melyre $d[x] (= C[s, x])$ minimális.
- Ekkor biztos, hogy az egyetlen (s, x) élből álló út egy legrövidebb $s \hookrightarrow x$ út, hiszen bármerre másfele indulnánk el s-ből, már az első él miatt legalább ilyen hosszú utat kapnánk (az élsúlyok nemnegatívak!).
- Tehát x-et betehetjük (s mellé) a $KÉSZ$ halmazba.
- A $KÉSZ$ halmaz azokat a csúcsokat tartalmazza, amelyeknek s-től való távolságát már tudjuk.
- Ezek után módosítsuk a többi csúcs $d[w]$ értékét, ha az eddig ismertnél rövidebb úton el lehet érni oda x-en keresztül, azaz ha $d[x] + C[x, w] < d[w]$.
- Most újra válasszunk ki a $v \in V \setminus KÉSZ$ csúcsok közül egy olyat, amelyre $d[v]$ minimális. Ezen csúcs $d[\ ]$-értéke már az s-től való távolságát tartalmazza az előzőhöz hasonló okok miatt (ezt később bebizonyítjuk).
- Majd megint a $d[\ ]$-értékeket módosítjuk, és így tovább, míg minden csúcs be nem kerül a $KÉSZ$ halmazba.

Dijskstra algoritmus - példa



	KÉSZ	*d*						*Π*					
		1	2	3	4	5	6	1	2	3	4	5	6
1		0	∞	∞	∞	∞	∞	*NIL*	*NIL*	*NIL*	*NIL*	*NIL*	*NIL*
2	1	0	3	∞	∞	∞	∞	*NIL*	1	*NIL*	*NIL*	*NIL*	*NIL*
3	1, 2	0	3	∞	7	13	16	*NIL*	1	*NIL*	2	2	2
4	1, 2, 4	0	3	14	7	9	16	*NIL*	1	4	2	4	2
5	1, 2, 4, 5	0	3	14	7	9	10	*NIL*	1	4	2	4	5
6	1, 2, 4, 5, 6	0	3	14	7	9	10	*NIL*	1	4	2	4	5
7	1, 2, 4, 5, 6, 3	0	3	14	7	9	10	*NIL*	1	4	2	4	5

Bellman—Ford-algoritmus

- A feladat ugyanaz: **legrövidebb utak keresése irányított gráfon egy forráscsúcsból**
- Ami különbözik: **negatív élsúlyok is előfordulhatnak a gráfon** (negatív összköltségű körök – nyilván – továbbra sem)
- Gyakorlati példa: elektromos vagy hibrid gépkocsi üzemanyag-szükséglete dimbes-dombos tájon (lejtőn negatív fogasztás is előfordulhat)
- A negatív élsúlyok előfordulása sokat ront a műveletigényen: Dijkstra mohó algoritmusa (akár) $n \log n$-es, a Bellman—Ford-algoritmus n^3-ös

- Gondoljuk meg: ha a súlyozott élű G irányított gráfban nincs negatív összhosszúságú irányított kör, akkor bármely két pontja között van olyan legrövidebb út is, ami egyszerű (azaz nem tartalmaz ismétlődő csúcsot). Következésképpen van nem több, mint $n-1$ élből álló legrövidebb út is, ahol n a G csúcsainak száma.
- Tegyük fel itt is, hogy a $G = (V, E)$ súlyozott élű irányított gráf a C szomszédsági mátrixával adott (ebben a diagonális elemek nullák, az i, j helyzetű elem a $c(i, j)$ élsúly, ha $i \to j$ éle G-nek, a többi elem pedig ∞). Az egyszerűség kedvéért tegyük még fel, hogy $V = \{1,2, \dots, n\}$ és $s = 1$.
- A szakirodalomban többnyire R. E. Bellmannak és L. R. Fordnak tulajdonított algoritmus fokozatos közelítéssel határozza meg az s-ből a többi csúcsba vivő legrövidebb utak hosszát.

A módszer egy $T[1:n-1,1:n]$ táblázat (kétdimenziós tömb) sorról sorra haladó kitöltése. Azt szeretnénk elérni, hogy végül minden i, j-re ($1 \leq i \leq n-1$, $1 \leq j \leq n$) teljesüljön, hogy

(A) $T[i,j] =$ {a legrövidebb olyan $1 \hookrightarrow j$ irányított utak hossza, melyek legfeljebb i élből állnak}.

A feladatban foglalt állítás szerint ekkor $T[n-1,j]$ a legrövidebb $1 \hookrightarrow j$ utak hosszát tartalmazza. A módszert úgy tekinthetjük, hogy a keresett minimális távolságokat $n-1$ menetben közelítjük.

- Az első menet, a $T[1,j]$ sor kitöltése kézenfekvő, hiszen nyilván $T[1,j] = C[1,j]$.
- Tegyük fel ezután, hogy az i-edik sort már kitöltöttük, azaz a $T[i,1], T[i,2], \dots, T[i,n]$ értékekre (A) igaz. Ekkor

  (B) $T[i+1,j] := \min\{T[i,j], \min\limits_{k \neq j}\{T[i,k] + C[k,j]\}\}$

  adja az $i+1$. sor helyes értékeit.

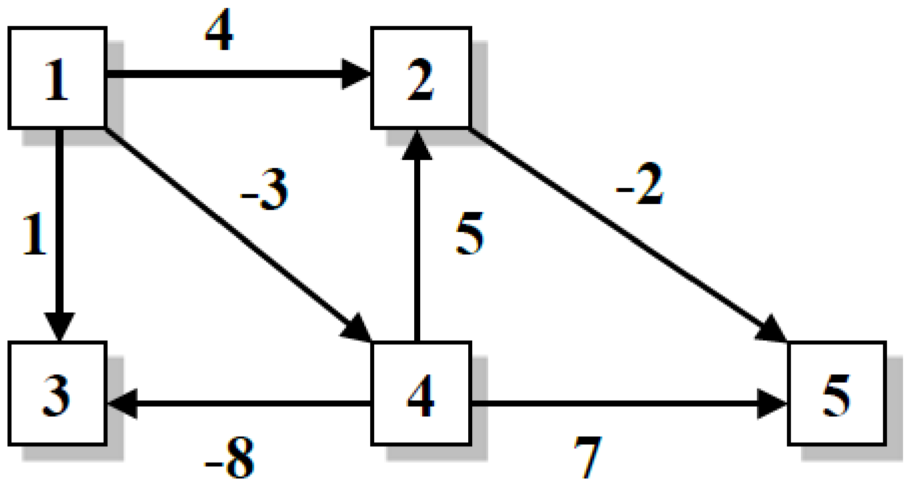
Ugyanis egy legfeljebb $i+1$ élből álló $\pi = 1 \hookrightarrow j$ út kétféle lehet:

(1) Az útnak kevesebb, mint $i+1$ éle van. Ekkor ennek a hossza szerepel $T[i,j]$-ben.

(2) Az út éppen $i+1$ élből áll. Legyen l a π út utolsó előtti pontja. Ekkor a π út $1 \hookrightarrow\ l$ szakasza i élből áll, és a π minimalitása miatt minimális hosszúságú a legfeljebb i élű $1 \hookrightarrow l$ utak között. A hossza tehát a T már elkészült darabjára tett feltevésünk miatt $T[i,l]$. Eszerint a π hossza $T[i,l] + C[l,j]$.

Tehát a π hossza szerepel a (B) jobboldalán azon mennyiségek között, amelyek minimumát vesszük; tehát egyenlő a minimummal.

Bellman—Ford algoritmus - példa



	d					Π				
	1	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**4**	**5**
1	0	∞	∞	∞	∞	*NIL*	*NIL*	*NIL*	*NIL*	*NIL*
2	0	**4**	**1**	**-3**	∞	*NIL*	**1**	**1**	**1**	*NIL*
3	0	**2**	**-11**	-3	**7**	*NIL*	**4**	**4**	1	**4**
4	0	2	-11	-3	**0**	*NIL*	4	4	1	**2**