

Algoritmuselmélet 7. előadás

Katona Gyula Y.
Budapesti Műszaki és Gazdaságtudományi Egyetem
Számítástudományi Tsz.
I. B. 137/b
kiskat@cs.bme.hu

2002 Március 11.

Múltkori animációk

Java animáció: Hash-elés

Java animáció: Huffman-fa

A Lempel–Ziv–Welch-módszer

A. Lempel és J. Ziv, 1970; T. Welch 1984

Használja: GIF, v.42bis, compress; ZIP, ARJ, LHA

Nem betűnként kódol, hanem a szöveg bizonyos szavaiból *szótárat* épít. $\implies S$

- az egybetűs szavak, azaz Σ elemei mind benne vannak S -ben;
- ha egy szó benne van a szótárban, akkor annak minden kezdődarabja is benne van;
- a szótárban tárolt szavaknak fix hosszúságú kódjuk van; az $x \in S$ szó kódját $c(x)$ jelöli.

Gyakorlatban a kódok hossza ~ 12 - 15 bit.

LZW kódolás

- az összenyomni kívánt szöveget S -beli szavak egymásutánjára bontjuk
- a szavakat a szótárbeli kódokkal helyettesítjük
- Az eredeti szöveg olvasásakor egyidőben épül, bővül az S szótár
- nincs optimalizálás, de a gyakorlatban jól működik

A szótár egyik szokásos tárolási módja a **szófa** adatszerkezet.

Ha az olvasás során egy $x \in S$ szót találunk, aminek a következő Y betűvel való folytatása már nincs S -ben, akkor $c(x)$ -et kiírjuk a kódolt szövegbe.

Az xY szót felvesszük az S szótárba. A szó $c(xY)$ kódja a legkisebb még eddig az S -ben nem szereplő kódérték lesz.

Ezután az Y betűvel kezdődően folytatjuk a bemeneti szöveg olvasását.

Legyen z egy szó típusú változó, K egy betű típusú változó. A z változó értéke kezdetben az összenyomni szánt állomány első betűje. Végig teljesül, hogy $z \in S$.

- (0) Olvassuk a bemenő állomány következő betűjét K -ba.
- (1) Ha az előző olvasási kísérlet sikertelen volt (vége a bemenetnek), akkor írjuk ki $c(z)$ -t, és álljunk meg.
- (2) Ha a zK szó is S -ben van, akkor $z \leftarrow zK$, és menjünk vissza (0)-ra.
- (3) Különben (azaz ha $zK \notin S$) írjuk ki $c(z)$ -t, tegyük a zK szót S -be. Legyen $z \leftarrow K$, majd menjünk vissza (0)-ra.

A $c(x)$ kódok rögzített hosszúságúak. Ha például ez a hosszúság 12 bit, akkor az S szótárba összesen 4096 szó kerülhet. \implies
Ha a szótár betelt, nem bővítünk tovább, úgy folytatjuk.

Példa LZW-re

Legyen $\Sigma = \{a, b, c\}$ és $c(a) = 1$, $c(b) = 2$, $c(c) = 3$.

Szótár

		$a \rightarrow$	1
		$b \rightarrow$	2
		$c \rightarrow$	3
$ababcbababaaaaaaaa$	\Rightarrow	$ab \rightarrow$	4
1 2 4 3 5 8 1 10 11 1	\Rightarrow	$ba \rightarrow$	5
	\Rightarrow	$abc \rightarrow$	6
0000 0001 0011 0010 0100	\Rightarrow	$cb \rightarrow$	7
1000 0000 1001 1010 0000	\Rightarrow	$bab \rightarrow$	8
	\Rightarrow	$baba \rightarrow$	9
	\Rightarrow	$aa \rightarrow$	10
	\Rightarrow	$aaa \rightarrow$	11
	\Rightarrow	$aaaa \rightarrow$	12

Dekódolás

Elvégezhető pusztán az egybetűs szavak kódjainak, valamint a kódok képzési szabályának ismeretében, nem kell tárolni S -et.

Pl. Ha a kódolt szöveg: 1 2 4 3 5 8 1 10 11 1 és tudjuk, hogy $c(a) = 1$,
 $c(b) = 2$, $c(c) = 3 \implies$

Ez első két jel betű kódja $\implies ab \checkmark$

ab nem volt a kezdeti S -ben, de az eleje igen; \implies a kódoló algoritmus ezt
 $c(ab) = 4$ értékkel S -be tette $\implies abab \implies$

ba volt a következő szó, ami S -be került $\implies c(ba) = 5 \dots$ Ha itt tartunk

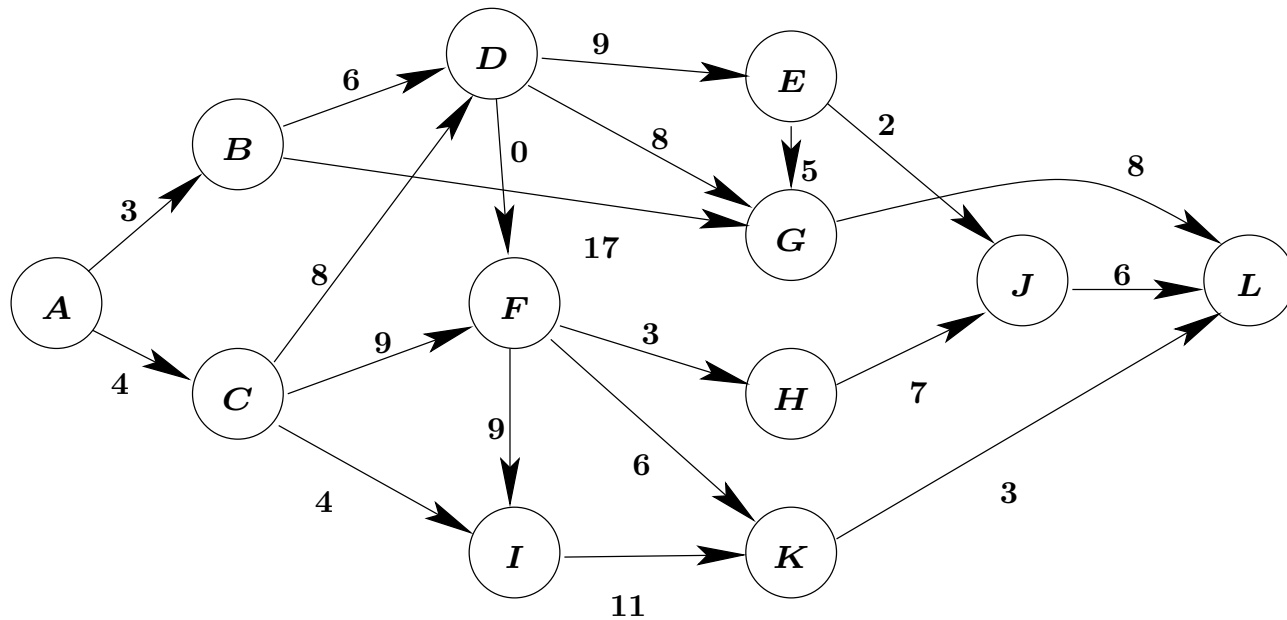
	$a \rightarrow$	1	
	$b \rightarrow$	2	a 8-as kódú szó ba^*
	$c \rightarrow$	3	alakú,
1 2 4 3 5 8 1 10 11 1	$ab \rightarrow$	4	\implies következő betű biztos b
$ababcba$	$ba \rightarrow$	5	$c(bab) = 8$
	$abc \rightarrow$	6	$ababcbabab$
	$cb \rightarrow$	7	

Java animáció: LZW-kódolás

Laczay Bálint féle GIF animáció

Gráfalgoritmusok

- irányított gráfok: $G = (V, E)$
- irányított él, irányított út, irányított kör
- élsúlyok: $c(e)$ — lehetnek negatívak is

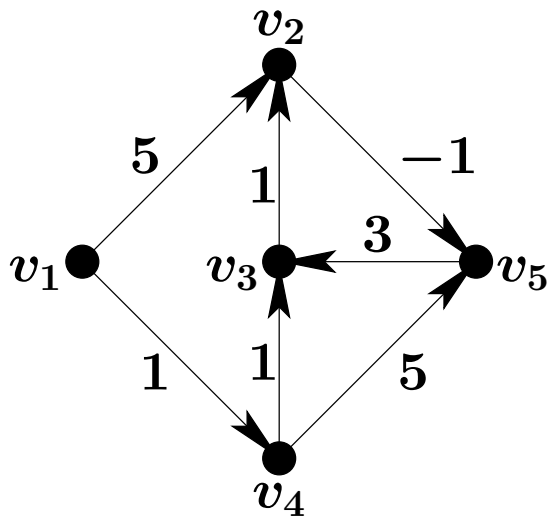


Adjacencia-mátrix

Definíció. A $G = (V, E)$ gráf *adjacencia-mátrixa* (vagy *szomszédossági mátrixa*) a következő – a V elemeivel indexelt – n -szer n -es mátrix:

$$A[i, j] = \begin{cases} 0 & \text{ha } (i, j) \notin E, \\ 1 & \text{ha } (i, j) \in E. \end{cases}$$

Írányítatlan gráfok esetén a szomszédossági mátrix szimmetrikus lesz (azaz $A[i, j] = A[j, i]$ teljesül minden i, j csúcspárra).

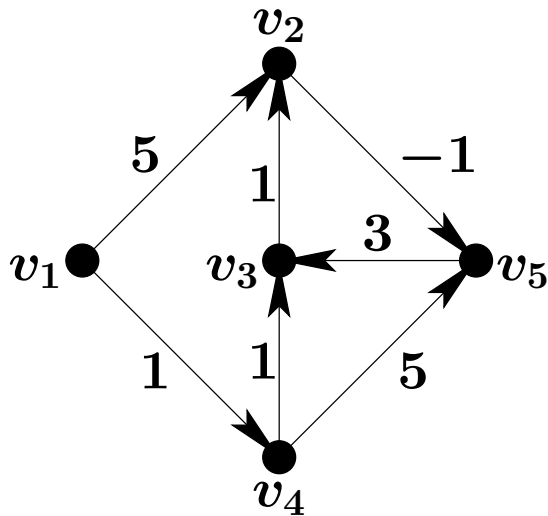


$$A = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Súlyozott adjacencia-mátrix

Ha költségek is vannak \implies

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



$$C = \begin{pmatrix} 0 & 5 & * & 1 & * \\ * & 0 & * & * & -1 \\ * & 1 & 0 & * & * \\ * & * & 1 & 0 & 5 \\ * & * & 3 & * & 0 \end{pmatrix}$$

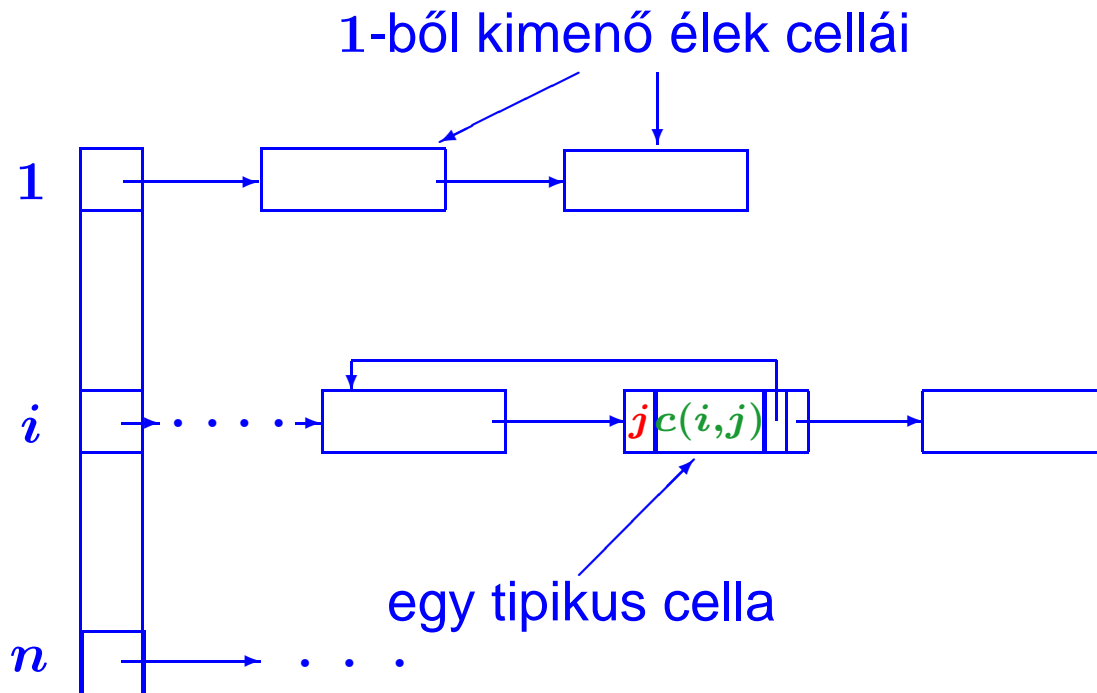
Hátránya \implies a mérete (n^2 tömbelem) teljesen független az élek számától.

Élhistás megadás

$G = (V, E)$ gráf minden csúcsához egy lista tartozik.

Az $i \in V$ csúcs listájában tároljuk az i -ből kimenő éleket, és ha kell, ezek súlyát is.

Az i listáján egy élnek a lista egy eleme (cellája) felel meg.



Az (i, j) élnek megfelelő cella tartalmazza a j sorszámot, a $c(i, j)$ súlyt (ha van), egy mutatót a következő cellára, és esetleg még egyet az előzőre is.

Tárigény: $n + e$ cella,

Irányítatlan gráfoknál:

$n + 2e$ cella

A legrövidebb utak problémája

Legyen adott egy $G = (V, E)$ irányított gráf a $c(f)$, $f \in E$ élsúlyokkal.

Feladat. Mekkora a legrövidebb út egy adott pontból egy másik adott pontba?

Feladat. Mekkora a legrövidebb út egy adott pontból az összes többibe?

Feladat. Mekkora a legrövidebb út bármely két pont között?

A G gráf egy u -t v -vel összekötő (nem feltétlenül egyszerű) $u \rightsquigarrow v$ irányított útjának a **hossza** az úton szereplő élek súlyainak összege.

Legrövidebb $u \rightsquigarrow v$ út \implies egy olyan $u \rightsquigarrow v$ út, melynek a hossza minimális a G -beli $u \rightsquigarrow v$ utak között.

u és v csúcsok (G -beli) $d(u, v)$ távolsága:

- 0, ha $u = v$;
- ∞ , ha nincs $u \rightsquigarrow v$ út
- egyébként pedig a legrövidebb $u \rightsquigarrow v$ 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!

Dijkstra módszere

Feladat. *A legrövidebb utak problémája (egy forrásból):*

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

$D[\] \implies$

- Egy a G csúcsaival indexelt tömb
- az eljárás során addig megismert legrövidebb $s \rightsquigarrow v$ utak hossza
- Felső közelítése a keresett $d(s, v)$ távolságnak
- A közelítést lépésről lépésre finomítjuk, végül $d(s, v)$ -t érjük el.

Tegyük fel, hogy a G gráf az alábbi alakú C adjacencia-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.

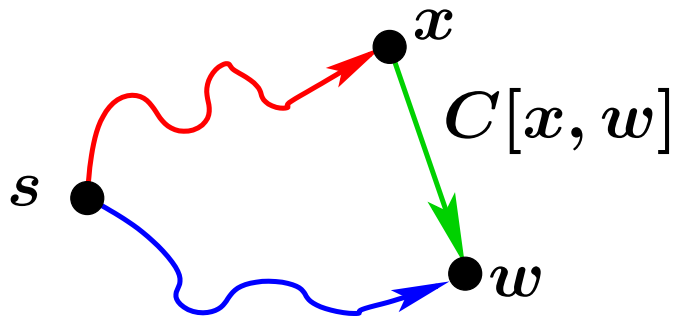
Válasszuk ki ezután az s csúcs szomszédai közül a hozzá legközelebbit, vagyis egy olyan $x \in V \setminus \{s\}$ csúcsot, melyre $D[x]$ minimális

Biztos, hogy az egyetlen (s, x) élből álló út egy legrövidebb $s \rightsquigarrow x$ út, **(az élsúlyok nemnegatívak!)**.

A **KÉSZ** halmaz azokat a csúcsokat tartalmazza, amelyeknek s -től való távolságát már tudjuk.

$\implies x$ -et betehetjük (s mellé) a KÉSZ halmazba.

Ezek után módosítsuk a többi csúcs $D[w]$ értékét, ha az eddig ismertnél rövidebb úton el lehet érni oda x -en keresztül, azaz ha $D[x] + C[x, w] < D[w]$.



Újra válasszunk ki a $v \in V \setminus \text{KÉSZ}$ csúcsok közül egy olyat, amelyre $D[v]$ minimális.

Ezen csúcs $D[\]$ -értéke már az s -től való távolságát tartalmazza.

Majd megint a $D[\]$ -értékeket módosítjuk, és így tovább, míg minden csúcs be nem kerül a KÉSZ halmazba.

Dijkstra algoritmusa adjacencia-mátrixszal

(1) KÉSZ := { s }

for minden $v \in V$ csúcsra **do**

$D[v] := C[s, v]$ (* a $d(s, v)$ távolság első közelítése *)

(2) **for** $i := 1$ **to** $n - 1$ **do begin**

Válasszunk olyan $x \in V \setminus$ KÉSZ csúcsot, melyre $D[x]$ minimális.

Tegyük x -et a KÉSZ-be.

(3) **for** minden $w \in V \setminus$ KÉSZ csúcsra **do**

$D[w] := \min\{D[w], D[x] + C[x, w]\}$ (* $d(s, w)$ új közelítése *)

end

Definíció. *különleges út:* egy $s \rightsquigarrow z$ irányított út különleges, ha a z végpontot kivéve minden pontja a KÉSZ halmazban van. A különleges úttal elérhető pontok éppen a KÉSZ-ből egyetlen éllel elérhető pontok.

Tétel. A (2) ciklus minden iterációs lépése után érvényesek a következők:

(a) KÉSZ pontjaira $D[v]$ a legrövidebb $s \rightsquigarrow v$ utak hossza.

(b) Ha $v \in \text{KÉSZ}$, akkor van olyan $d(s, v)$ hosszúságú (más szóval legrövidebb) $s \rightsquigarrow v$ út is, amelynek minden pontja a KÉSZ halmazban van.

(c) Külső (vagyis $w \in V \setminus \text{KÉSZ}$) pontokra $D[w]$ a legrövidebb különleges $s \rightsquigarrow w$ utak hossza.

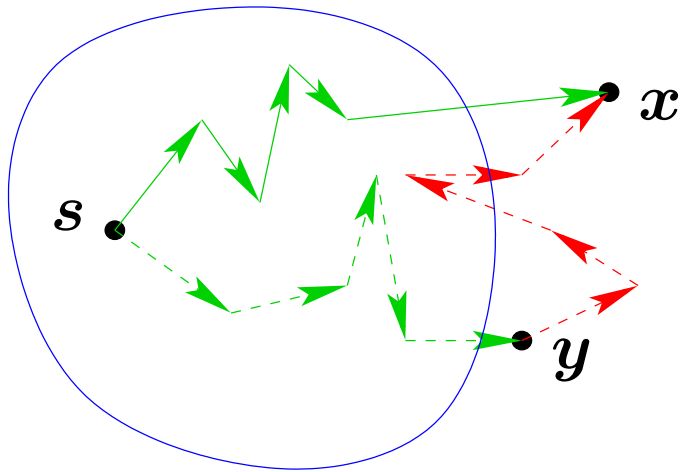
Bizonyítás: (a) Indukcióval

(2) előtt ✓

Tegyük fel, hogy igaz a j -edik iteráció után.

Belátjuk, hogy igaz a $j + 1$ -edik iteráció után is.

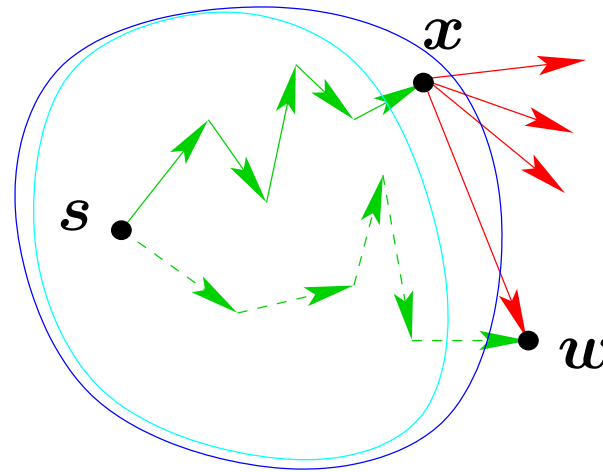
Tegyük fel, hogy az algoritmus a $j + 1$. iterációs lépésben az x csúcsot választja a KÉSZ-be.



Indirekt: mi van, ha $D[x]$ nem a $d(s, x)$ távolságot jelöli, azaz van ennél rövidebb $s \rightsquigarrow x$ út?

Ezen út „eleje” különleges \implies
 $D[y] \leq d(x, s) < D[x]$ ⚡

(b) Elég x -re \iff KÉSZ korábbi pontjaira az indukciós feltevésből
 Láttuk, hogy $d(s, x) = D[x]$, ez egy különleges $s \rightsquigarrow x$ út hossza volt a $j + 1$. iteráció előtt (itt a (c)-re vonatkozó indukciós feltevést használtuk) annak végeztével az út minden pontja KÉSZ-beli lesz.



(c) A $j + 1$. iteráció előtt

$$D[w] = \min_{v \in \text{KÉSZ} \setminus \{x\}} \{d(s, v) + C[v, w]\}.$$

Utána
$$D[w] = \min_{v \in \text{KÉSZ}} \{d(s, v) + C[v, w]\}.$$

\implies Elég megnézni, hogy $D[w]$ vagy $d(s, x) + C[x, w]$ nagyobb

Lépésszám: (2) ciklus $O(n)$, ez lefut $n - 1$ -szer $\implies O(n^2)$