

Algoritmusok és adatszerkezetek 2

2015/16 tavaszi félév

Előadó: Dr. Ásványi Tibor

Készítették:

Koruhely Gábor (Koru)

Szalay Richárd (Whisperity)

Lektorálta:

Dr. Ásványi Tibor

Frissítve: 2017. 02. 20.

első blokk: 10:15 - 11:00  
 szünet: 11:00 – 11:15  
 második blokk: 11:15 - 12:00

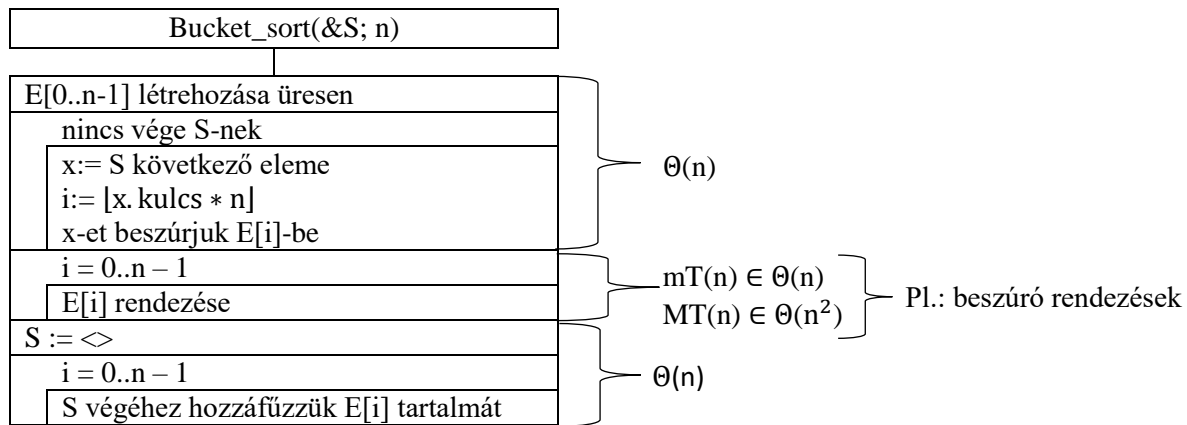
## Rendezés lineáris időben

### Edényrendezés (bucket sort)

Tételezzük fel, hogy a kulcsok:  $[0,1)$  intervallumba esnek és egyenletesen oszlanak el.

Példa számok: 0,32; 0,81; 0,89; 0,17; 0,53 (n = 5) a „k” kulcsot  $[k * n]$  edénybe tesszük

0.edény: 0,17  
 1.edény: 0,32  
 2.edény: 0,53  
 3.edény:  
 4.edény: 0,81; 0,89  
 az eredmény: 0,17; 0,32; 0,53; 0,81; 0,89



azonban hiába mehet  $n^2$ -ig a műveletigény, a legtöbb esetben (a legtöbb inputra) marad az átlagos n-es műveletigény

## Leszámláló rendezés (Counting sort)

Most tegyük fel, hogy a kulcsok  $[0..k]$ -ba esnek,  $k \in O(n)$  (azaz  $k$  legfeljebb  $n$ -nel lineáris)

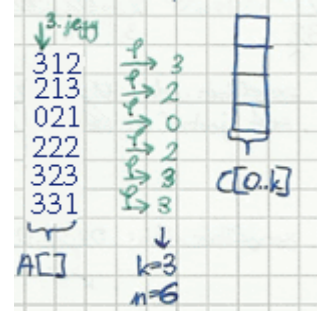
$\phi$ : kulcsfüggvény  $A[1..n]: T$   
 $\phi: T \rightarrow [0;k]$   $C[0..k]: 0..n$

Példa: ( $k = 3$ ), a számok: 312, 213, 021, 222, 323, 331  
 (a 0. lépés az init)

C[] adottodik elemkor								végén
h	0.	1.	2.	3.	4.	5.	6.	
0	0			1				1
1	0							0
2	0		1		2			2
3	0	1				2	3	3

ennyi ilyen kulcsú elem van  $\phi$  szerint

négyes számrendszer-beli számok



Counting\_sort( $A[1..n]$ ,  $k$ ,  $\phi$ ,  $B[1..n]$ )

$C[0..k]$  létrehozása

$i := 0..k$

$C[i] := 0$

$i = 1..n$

$C[\phi(A[i])] := C[\phi(A[i])] + 1$

$i = 1..k$

$C[i] := C[i] + C[i - 1]$

$i = n..1$  (-1)

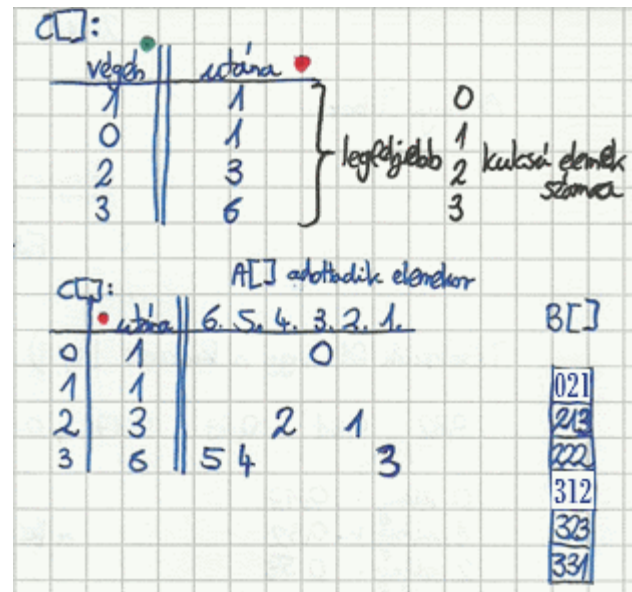
$j := \phi(A[i])$

$B[C[j]] := A[i]$

$C[j] := C[j] - 1$

végén

utána



most már a bal szélső számjegy szerint is rendezett a tömb; a többi számjegy szerinti, most már másodlagos rendezettséget megtartotta.

## Radix (számjegypozíciós) rendezés

Radix(n elem, d számjegy, (szjegyek 0..k))

$i := 1..d$

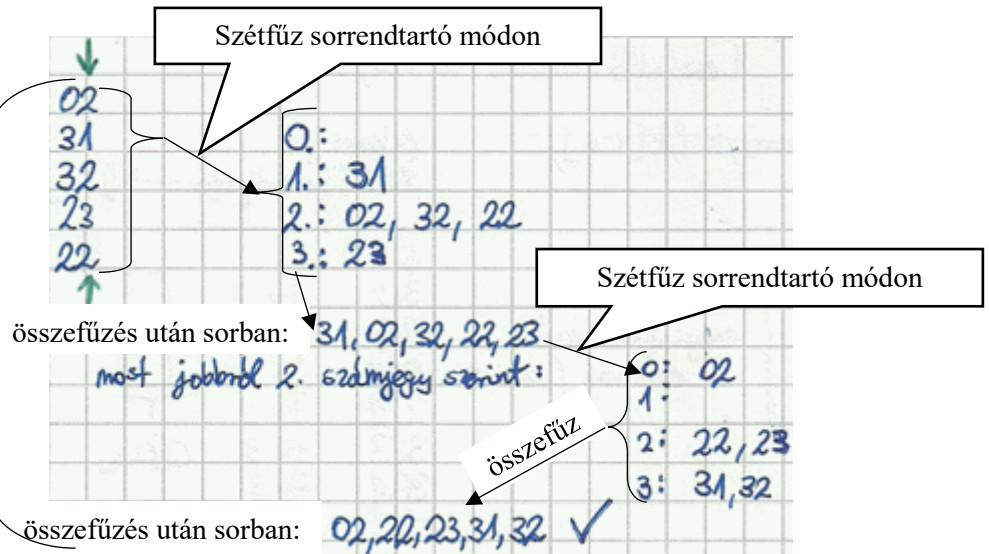
rendezzük az elemeket stabil rendezéssel  
hátról i-edik számjegyük szerint

Stabil rendezés: amely az elemeknek, egyenlő kulcsok esetén, az egymáshoz viszonyított sorrendjét nem változtatja meg (pl Counting\_sort ilyen).

$$\left. \begin{array}{l} T_{\text{radix}} \in \Theta(d * T_{\text{stabil}}(n)) \\ T_{\text{counting}}(n, k) \in \Theta(n + k) \end{array} \right\} T_{\text{Radix rendezés tömbökre}} \in \Theta(d * (n + k)) = \Theta(n)$$

ha  $d$  konstans és  $k \in O(n)$  volt

Radix rendezés  
láncolt listákra,  
stabil edény-  
rendezéssel, jobbról  
az i.-edik számjegy  
szerint ( $i := 1, \dots, d$ )



$$T_{\text{radix/edényrendezés}}(n, k) \in \Theta(d * (n + k)) = \Theta(n)$$

ha  $d$  konstans és  $k \in O(n)$

tárgy követelményei:

- Legalább elégséges gyakorlati jegy kell a vizsgázáshoz,
- vizsga
  - 3szor lehet maximum vizsgázni egy vizsgaidőszakban,
  - vizsga ugyanolyan felépítésű, mint előző félévben az algo 1

tematika:

- A tárgy honlapján megtalálható a tárgy tematikája: <http://aszt.inf.elte.hu/~asvanyi/ad/ad2programok.pdf>
- 8tétel
- nem összehasonlító rendezésekről lesz szó,
- hasítótáblák
- gráf algók
  - legrövidebb út
  - minimális feszítőfa
- veszteségmentes tömörítés
- mintaillesztés

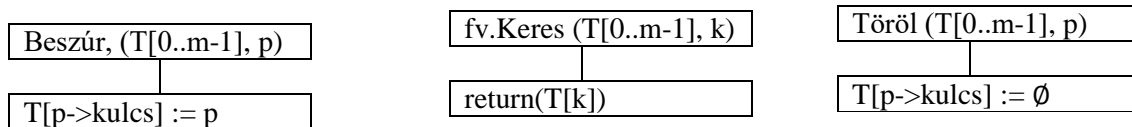
## Hasítótáblák (Hash tables)

Hasonlóan a kiegyensúlyozott fákhoz cél lenne a 3 alapl művelet (keresés, beszúrás, törlés) optimálisan fusson le.

- átlagos esetben  $\Theta(1)$ , legrosszabb esetben  $\Theta(n)$ .

Direkt – címzés (Direct addressing):

a lehetséges kulcsok az  $U = \{0, \dots, m-1\}$  (m nem túl nagy pozitív egész) univerzumba esnek  
műveletei: beszúr; keres; töröl



A tárigény miatt problémás a módszer, pl. ha a személyi szám szerint így tárolnánk a magyarokat, akkor a <10millió embernek 75 millió hely kellene, nagyon sok az üres hely.

h: kulcsuniverzumból résekre (slotokra) képez le.

(n a tárolt adatok száma)

h:  $U \rightarrow [0..m-1]$

( $|U| \gg m$  /\*>> := sokkal nagyobb\*/), ( $m \in O(n)$ )

h hasító függvény

A baj az, hogy hiába tesszük a hash szerinti helyre a kulcsokat, előbb, utóbb lesz két külön kulcs, aminél  $h(k) = h(k')$ . Ekkor kulcsütközés következik be, ez célszerűen láncolt listával elkerülhető.

Láncolt listás esetben viszont  $MT_{\text{keres}}(n) \in \Theta(n)$ ;  $mT_{\text{keres}}(n) \in \Theta(1)$ ;  $AT_{\text{keres}}(n) \in \Theta(1 + \alpha)$ , a törlés műveletigénye ugyanaz, mint a keres egyes eseteiben.

a minimális akkor lehet, ha a réshez tartozó lista első elemét keressük, vagy a lista üres.

a listák átlagos hossza a hashtábla kitöltöttségi hányadosa,  $\alpha = \frac{n}{m}$

Ha nem ellenőrizzük az esetleges duplikált kulcsokat, a beszúrás igénye  $\Theta(1)$ . Ha nem engedjük meg a duplikált kulcsokat, ugyanaz, mint a keresés-nél.

Egyszerű egyenletes hasítás: minden slotra ugyanakkora a h leképezési valószínűsége.

Ilyen hash-ek megadhatók, de elég komoly matematika van mögötte.

Osztómódszer:  $h(k) = k \bmod m$ ,

ahol m olyan prím amely messze van a 2-hatványoktól

Ha a kulcsok a [0,1) intervallumon egyenletesen oszlanak el, akkor a  $h(k) = \lfloor k * m \rfloor$  is jó hashfüggvény

Szorzó módszer:

$0 < A < 1$  konstans

$h(k) = \lfloor \{k * A\} * m \rfloor$  ( $\{x\}$  a törtrész fgv.)

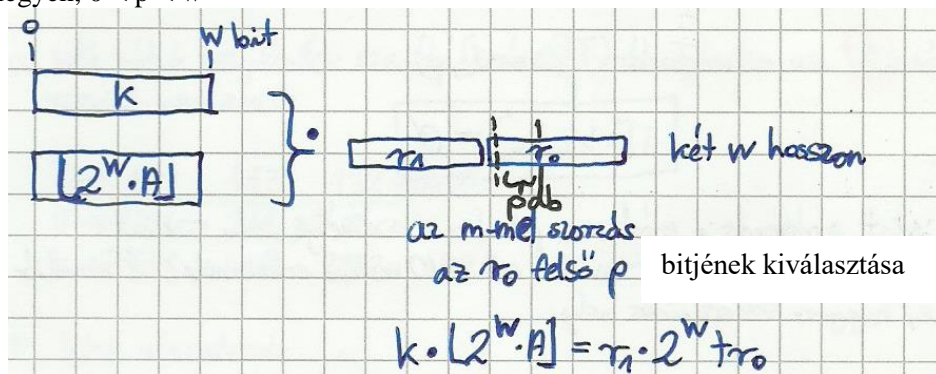
(különböző vizsgálatok szerint  $A = \frac{(\sqrt{5}-1)}{2} \approx 0,618$  .... jól szórja szét a kulcsokat.

Ez a számítás azonban lassúbb a lebegőpontos aritmetika miatt. Gyorsítás:

$m = 2^p$  a hashtábla mérete

$w = 32$

$k \in 0 \dots 2^w - 1$ ; legyen,  $0 < p < w$



A kulcsütközések feloldására a láncolt lista helyett vannak más módszerek is.

## Nyílt címzés, részletesen ld. <http://aszt.inf.elte.hu/~asvanyi/ad/ad2jegyzetReszek.pdf>

[A nyílt címzésről szóló rész itt erősen hiányos, mert töröltem a tananyag változása miatt elavult részeket. (ÁT)]

$h: U \times [0 \dots m - 1] \rightarrow [0 \dots m - 1]$  Beszúrásnál ellenőrizzük a duplikált kulcsokat.

$k$  kulcshoz  $m$  darab hash-érték fog tartozni:  $\langle h(k,0); h(k,1); \dots h(k,m-1) \rangle \leftarrow$  potenciális próbasorozat

A műveletek a potenciális próbasorozat egy prefixén mennek végig: aktuális próbasorozat

A potenciális próbasorozatnak a  $\langle 0,1,\dots,m-1 \rangle$  permutáltjának kell lennie

A próbasorozatok egyik előállítási módja pl. a lineáris próbálás:

$$h(k,i) = (h'(k) + i) \bmod m$$

## 11<sub>2</sub> EA

### Hasítótáblák

Hasítótáblák tulajdonságai:

- egy szótárt valósítanak meg
- kulcs alapján lehet benne keresni
- törölni is lehet benne
- átlagos műveleti igény konstans

a tábla (ism.):

egy tömbnek lehet tekinteni;  $0..m-1$ -ig indexelve van

speciálisan a nyílt címzésnél az adatokat slot-okban (résekben) tároljuk;

A foglalt rések mellett megkülönböztetjük a törölt illetve az üres slotokat

sikertelen a keresés, ha üres slothoz ér, vagy ha túl sokat próbálkozik ( $m$  db próba után megáll)

valójában nem egy hash függvény van, hanem  $m$  db hash függvény van

feltehetjük, hogy van  $m$  db hash függvény

$$h: (., i) : U \rightarrow 0..m-1 \quad (i \in 0..m-1)$$

$$\langle h(k,0), \dots, h(k,m-1) \rangle \text{ perm } \langle 0, \dots, m-1 \rangle$$

lineáris próba:  $h(k,i) = (h'(k) + i) \bmod m$

négyzetes próba:  $h(k,i) = (h'(k) + c_1 * i + c_2 * i^2) \bmod m$  /\*ahol  $h'(k) = h(k,0)$ \*/

ahol jól bevált ajánlás, hogy  $m$  2-hatvány,  $c_1, c_2$  pedig  $\frac{1}{2}$  ( $m = 2^p, 0 < p \in \mathbb{Z}$ )

$$i \neq j: (h(k,i) - h(k,j)) \bmod m = \left[ \frac{i*(i+1)}{2} - \frac{j*(j+1)}{2} \right] \bmod m \neq 0 \Leftrightarrow$$

$$2m \nmid i * (i + 1) - j * (j + 1) = i^2 + i - j^2 - j = i^2 - j^2 + (i - j) = (i - j)(i + j + 1)$$

$2^{p+1} \nmid (i - j)(i + j + 1)$  ezek párossága különbözik

$$a) \quad 2 \mid (i - j) \Rightarrow 2 \nmid (i + j + 1)$$

$$\text{de } 2m \nmid (i - j) \text{ hiszen } i - j < 2m-1$$

$$b) \quad 2 \mid (i + j + 1) \Rightarrow 2 \nmid (i - j)$$

Tipikusan foglalt rések hosszú összefüggő szakaszain alakulnak ki  $T[0..m-1]$ -ben elsődleges csomósodások.

de  $2m \nmid (i+j+1)$ -nek, mivel  $(i+j+1) < 2m-1$   
 azaz  $2$ -hatvány tábla esetén a négyzetes próba szépen lefedti az egész táblát

$$(h(k, i+1) - h(k, i)) \bmod m = \left[ \frac{(i+1) * (i+2)}{2} - \frac{i * (i+1)}{2} \right] \bmod m = (i+1) \bmod m$$

azaz:  $h(k, i+1) = (h(k, i) + (i+1)) \bmod m$   
 és  $h(k, 0) = h'(k)$

az általános négyzetes próbánál (ahol  $c_1$  és  $c_2$  tetszőleges)

ha  $h(k_1, 0) = h(k_2, 0)$ , akkor  $\forall i - re \ h(k_1, i) \neq h(k_2, i)$

↑  
Másodlagos csomósodás

Tehát csak  $m$  különböző próbasorozat van a lehetséges  $m!$  közül, így a (másodlagos) csomósodás fellép, de ez jobb, mint a lineáris próba elsődleges csomósodása, ahol a sorozatok menet közben gabalyodhatnak össze...

## Kettős hasítás

$$h(k, i) = [h_1(k) + i * h_2(k)] \bmod m$$

a próbasorozatok száma itt már  $\Theta(n^2)$  szemben  $\Theta(n)$ -nel.

ez – tapasztalat alapján – már majdnem ideális tud lenni

ha  $\text{luko}(h_2(k), m) = 1 \Rightarrow \langle h(k, 0), \dots, h(k, m-1) \rangle \text{ perm } \langle 0, \dots, m-1 \rangle$ -nak

hiszen  $i \neq j \Rightarrow (h(k, i) - h(k, j)) \bmod m \neq 0$

$$0 \neq (h(k, i) - h(k, j)) \bmod m \Leftrightarrow (i - j) * h_2(k) \bmod m \neq 0$$

$$\Leftrightarrow m \nmid (i - j) * h_2(k) \text{ mivel } |i - j| < m, h_2(k) \text{ pedig relatív prím (ez feltétel volt, hogy lehet biztosítani,}$$

hogy a  $h_2(k)$  és az  $m$  relatív prímekek legyenek).

a)  $m$  legyen prím, akkor  $h_1(k) := k \bmod m$

$$h_2(k) = 1 + (k \bmod m')$$

ahol:  $m' = m - 1$  v  $m' = m - 2$  például, azaz  $m$ -nél picit kisebb

b)  $m = 2^p$  és  $2 \nmid h_2(k)$

Hashelés Ideális esete: Egyenletes hasítás

ideális lenne, ha  $\langle 0..m-1 \rangle$  összes permutáció azonos valószínűséggel fordulna elő

$0 < \alpha$  telítettségi együttható  $< 1$  esetén

- sikertelen keresés várható hossza  $\leq \frac{1}{1-\alpha}$
- beszúrás várható hossza  $\leq \frac{1}{1-\alpha}$
- sikeres keresés várható hossza  $\leq \frac{1}{\alpha} * \ln\left(\frac{1}{1-\alpha}\right)$

Feltéve, hogy nincs a hasító táblában törölt rés (slot).

Ha sokat használjuk a hashtáblát, akkor feltöredezik, elfogynak az üres slotok, ekkor frissítést kell végrehajtani: beszúrásokkal új táblát létrehozni a mostani elemekből.

## Gráf algoritmusok

Szomszédosági mátrix (C)

$$G = (V, E) \quad E \subseteq V \times V$$

$$V = 1..n$$

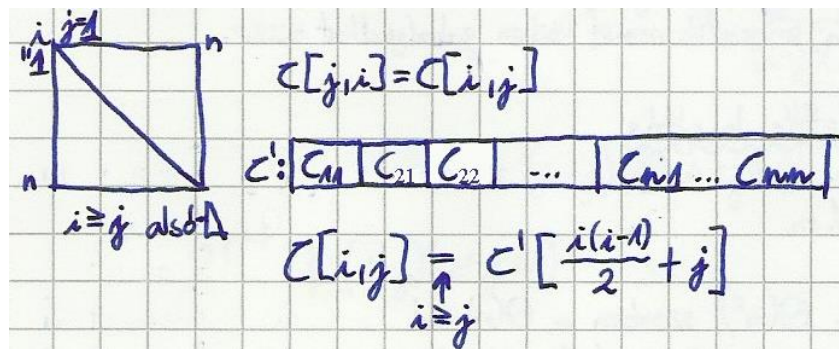
$$C[i, j] = \begin{cases} 1 & \Leftrightarrow (i, j) \in E \\ 0 & \Leftrightarrow (i, j) \notin E \end{cases}$$

ha számít az él költsége

$$w: E \rightarrow \mathbb{R}$$

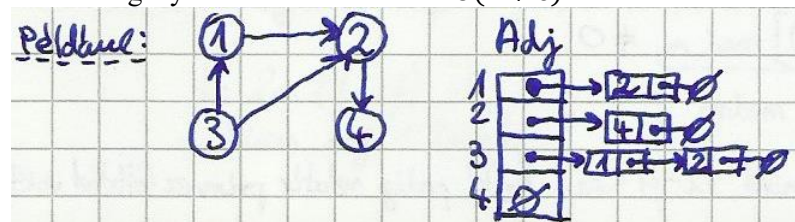
$$C[i, j] = \begin{cases} w(i, j) & \Leftrightarrow (i, j) \in E \quad (i \neq j) \\ 0 & \Leftrightarrow i = j \\ +\infty & \text{különben} \end{cases}$$

Azonban a mátrix tárolás miatt a műveletigény  $\Theta(n^2)$ , ami az irányítatlan esetben a szimmetria kihasználásával javítható (majdnem felezhető), de  $\Theta(n^2)$  marad (csak alsó háromszög mátrixot tároljuk).



Szomszédsági éllistas reprezentáció:

Memóriaigény  $n$  csúcs és  $e$  él esetén:  $\Theta(n + e)$





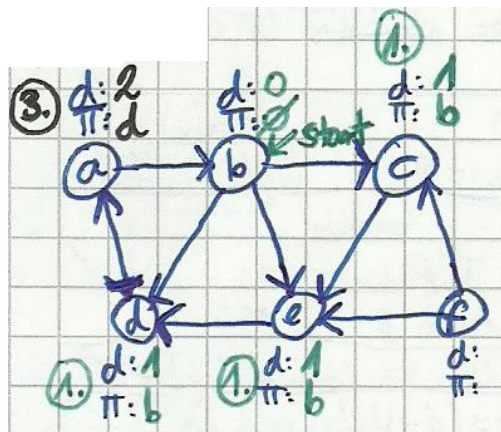
## Elemi gráf algoritmusok

## Szélességi keresés

Meghatározzuk a start csúcsból a további csúcsokba a legkevesebb él tartalmazó utat.

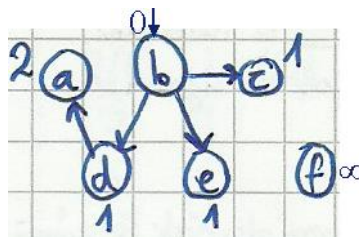
$d$  – hány élen keresztül jutunk a csúcsba

$\pi$  – melyik csúcsból jutunk a csúcsba



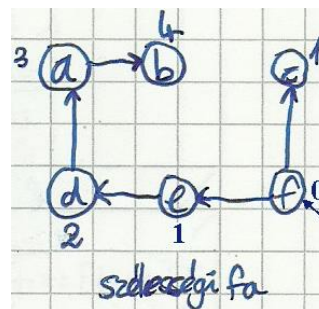
$Q = \langle b \rangle$

1. lépés:  $Q = \langle c, d, e \rangle$  /\*b szomszédai\*/
2. lépés:  $Q = \langle d, e \rangle$  /\*c rákövetkezője e, de ott már voltunk\*/
3. lépés:  $Q = \langle e, a \rangle$  /\*„a” d szomszédjai\*/
4. lépés:  $Q = \langle a \rangle$  /\*e szomszédja csak a d, de ott már voltunk\*/
5. lépés:  $Q = \langle \rangle$  /\*a sor kiürült, a bejárás véget ért



f-ből indítva, másik szemléltetési móddal:

d	a	b	c	d	e	f	Q	$\pi$	a	b	c	d	e	f
	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	0	$\langle f \rangle$		$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
f			1		1		$\langle c, e \rangle$				f		f	
c							$\langle e \rangle$							
e				2			$\langle d \rangle$					e		
d	3						$\langle a \rangle$	d						
a		4					$\langle b \rangle$		a					
b	3	4	1	2	1	0	$\langle \rangle$	d	a	f	e	f	$\emptyset$	



Az irányított gráfon a „szomszédsági” kapcsolat nem szimmetrikus.

A fenti gráfban pl.:  $a$  szomszédja  $b$ , de  $b$ -nek nem szomszédja  $a$ .

## Jelölés:

Ha  $G = (V, E)$ ,  $E \subseteq V \times V$  gráf,

$G.V$  a  $G$  csúcsainak halmaza

$G.E$  a  $G$  éleinek halmaza

$u \in G.V$  esetén

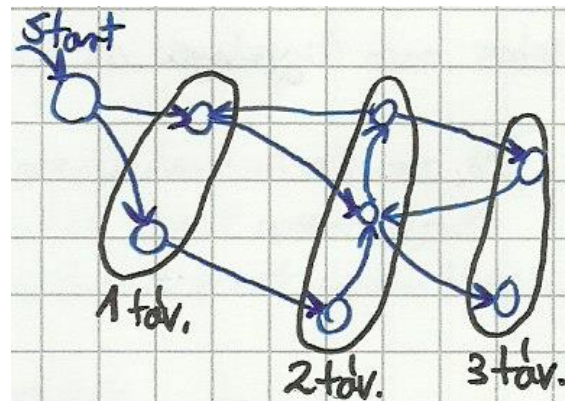
$G.Adj[u] \stackrel{\text{def}}{=} \{ v \in G.V \mid (u, v) \in G.E \}$

Feltesszük, hogy  $(u, u) \notin G.E$ , azaz nincs hurokél.

Feltesszük, hogy nincsenek párhuzamos élek sem.

}  $G$  egyszerű gráf

BFS(G,s)	
$\forall u \in G.V$	
u.d := $\infty$	
u. $\pi$ := $\emptyset$	
u.szín := fehér	
s.d := 0	
s.szín := szürke	
Q := <s>	
Q $\neq \langle \rangle$	
u := Q.sorból()	
$\forall v \in G.Adj[u]$	
v.szín = fehér	
v.d := u.d + 1	
v. $\pi$ := u	
v.szín := szürke	
Q.sorba(v)	
u.szín := fekete	



BFS műveletigénye:  $T(n, e) \in O(n + e)$ , ahol  $n = |V|$  és  $e = |E|$

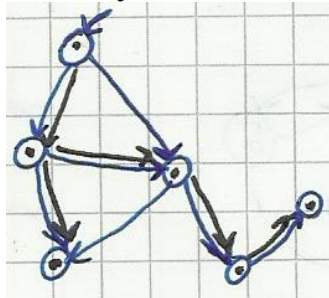
mindent legfeljebb egyszer dolgozunk fel, de előfordulhat, hogy néhány csúcs kimarad (a start csúcsból nem elérhető csúcsok és élek maradnak ki.).

DFS (Mélyégi keresés)

DFS(G)	
$\forall u \in G.V$	
u. $\pi$ := $\emptyset$	
u.szín := fehér	
ido := 0	
$\forall u \in G.V$	
u.szín = fehér	
DFS_VISIT(u,G, ido)	

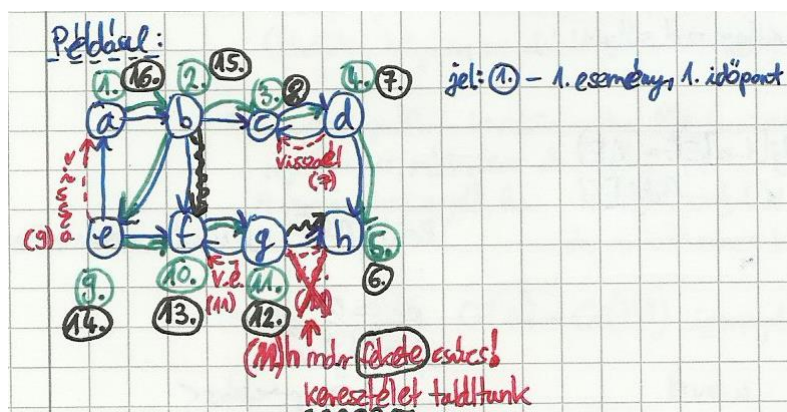
u.d = elérési idő (discovery time)

u.f = befejezési idő (finishing time)



DFS_VISIT(&u,G, &ido)	
ido := ido + 1	
u.d := ido	
u.szín := szürke	
$\forall v \in G.Adj[u]$	
v.szín = fehér	
v. $\pi$ := u	
DFS_VISIT(v,G, ido)	
v.szín = szürke	
viszrael(u,v)	
u.szín := fekete	
ido := ido + 1	
u.f := ido	

Mélyégi bejárás példa:



Ha az él már fekete csúcsba mutat, keresztélet vagy előreélet találtunk.

## Élek osztályozása

Def:

$a \rightarrow b$  faél: ez kerül a mélységi fába, e mentén járjuk be a gráfot

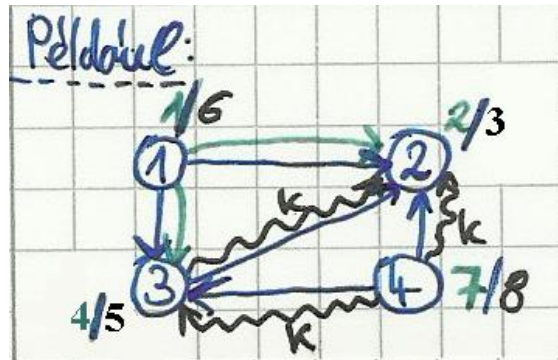
$f \leftarrow g$  visszaél (f a g őse egy mélységi fában.)

$b \rightarrow f$  előreél, az előreél két leszármazottat köt össze, ahol b-ből kettő vagy több faélból álló út vezet f-be.

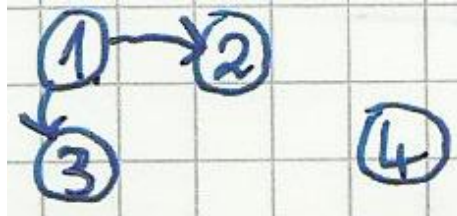
$g \rightarrow h$  keresztél, olyan két csúcs, amelyek más ágon vannak vagy másik mélységi fába mutat át

(\*)

Példa:



Azaz a keresés eredménye egy mélységi erdő



DFS műveletigénye:  $T(n,e) = \Theta(n + e)$  minden csúcsot néhányszor (3) és minden élet egyszer.

Tétel:

Ha a (\*) mélységi bejárás során egy  $(u,v)$  élet találunk:

$(u,v)$  faél  $\Leftrightarrow v.\text{szín} = \text{fehér}$

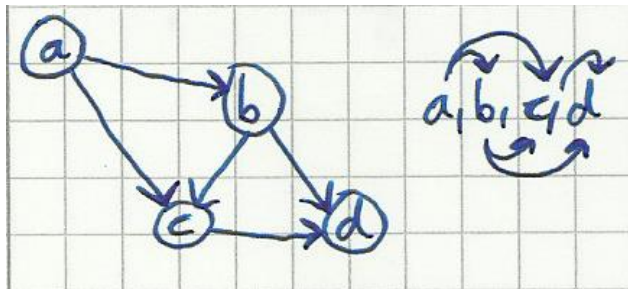
$(u,v)$  visszaél  $\Leftrightarrow v.\text{szín} = \text{szürke}$

$(u,v)$  előreél  $\Leftrightarrow v.\text{szín} = \text{fekete} \wedge u.d < v.d$

$(u,v)$  keresztél  $\Leftrightarrow v.\text{szín} = \text{fekete} \wedge u.d > v.d$

# Irányított gráfok csúcsainak topologikus rendezése

Topologikus rendezés: A csúcsok olyan sorrendje, amelyben minden él egy-egy később jövő csúcsba (szemléletesen: balról jobbra) mutat.



Tétel: Pontosán akkor  $\exists$  topologikus rendezés, ha nincs irányított kör a gráfban.

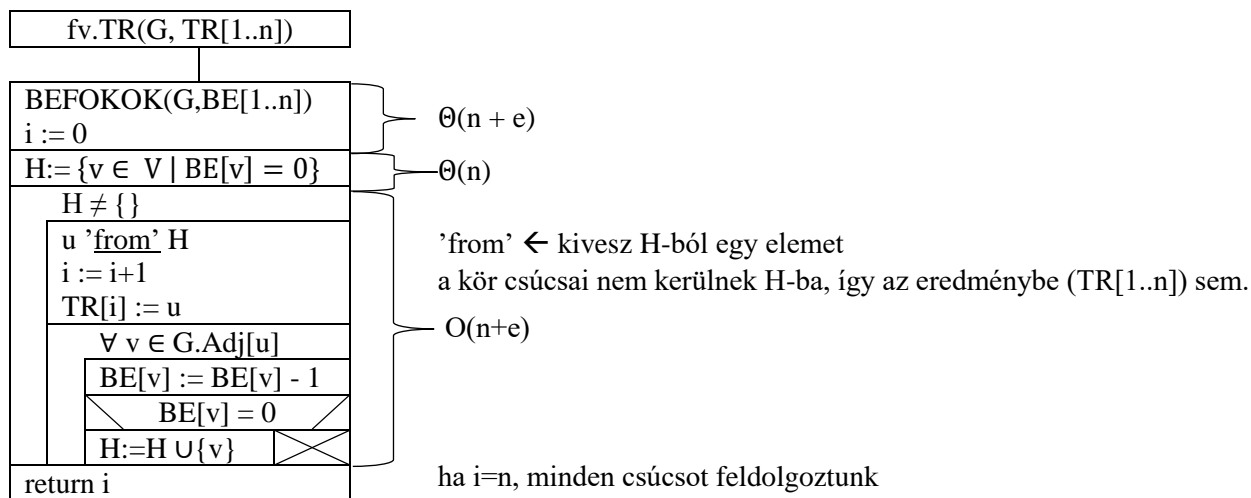
Bizonyítás:

- Ha van irányított kör a gráfban, jelölje  $\langle u_1, u_2, \dots, u_k, u_1 \rangle$ !  
Ekkor egy tetszőleges topologikus rendezésben  $u_1$  után jön valahol  $u_2$ , az után valahol  $u_3$ , és így tovább, végül is  $u_1$  után jön  $u_k$ , és  $u_k$  után  $u_1$ , ami  $\nexists$ , tehát ekkor nincs topologikus rendezés (a gráf csúcsain).
- Ha nincs irányított kör a gráfban, akkor nyilván van olyan csúcs, aminek nincs megelőzője.

Ha veszünk egy megelőzővel nem rendelkező csúcsot és töröljük a gráfból, akkor a maradék gráfban nem keletkezik irányított kör, lesz megint legalább egy olyan, amelyiknek nincs megelőzője.

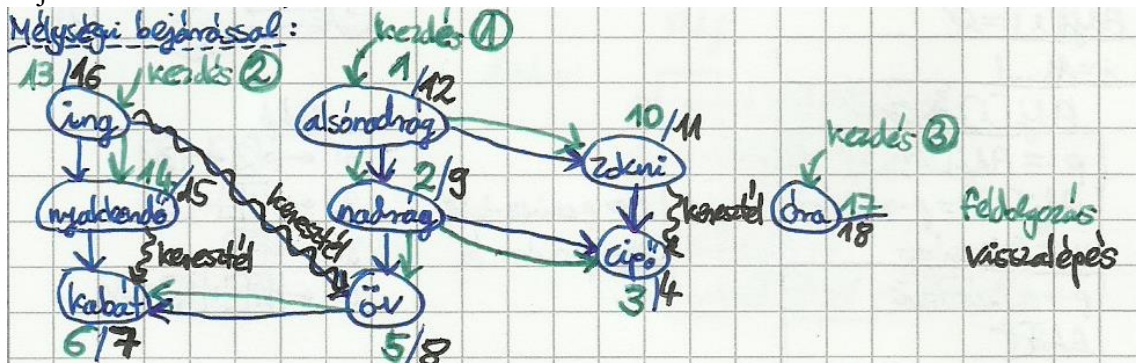
Sorban a törölt csúcsok adják a topologikus rendezést.

$$G=(V, \text{Adj}[V]) \quad \text{Adj}[u] = \{v \in V \mid (u,v) \in E\} \subseteq V$$



$$T_{\text{TR}}(n) \in O(n + e).$$

Mélységi bejárás:



A sorrend:  $\langle \text{óra}, \text{ing}, \text{nyakkendő}, \text{alsónadrág}, \text{zokni}, \text{nadrág}, \text{öv}, \text{kabát}, \text{cipő} \rangle$

Tetszőleges  $u$  csúcs, akkor kerül a TR-be, a hátulról első szabad helyre, amikor befejeztük.

Ha a mélységi bejárás visszaélt talál  $\Leftrightarrow$  irányított kör van a gráfban  $\Leftrightarrow$  nincs topologikus rendezés.

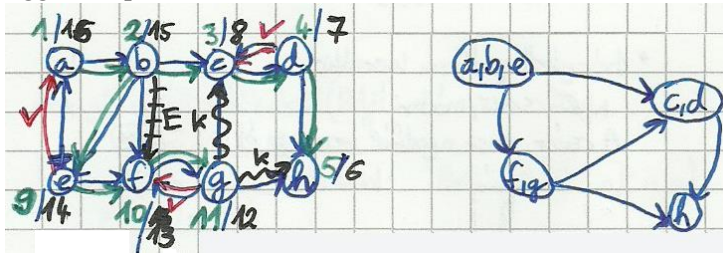


## Erősen összefüggő komponensek

/\*Minden csúcsból minden csúcsba vezet irányított út a komponensen belül\*/

A gráf erősen összefüggő komponensei:

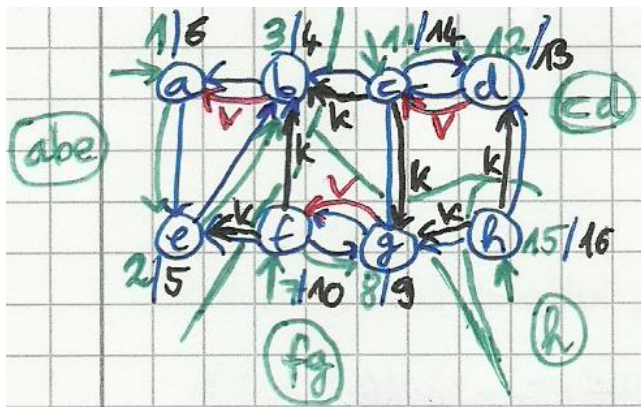
- a,b,e
- f,g
- c,d
- h



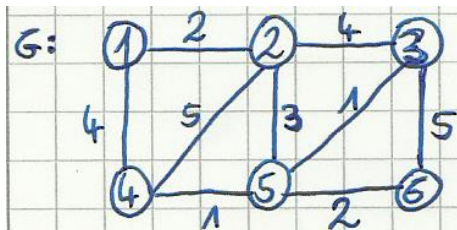
Előállítás: mélységi bejárással mintha topologikus rendezés lenne, de nem kezeljük a visszaéleket.

A sorozat:  $\langle a, b, e, f, g, c, d, h \rangle$  a csúcsok befejezés szerinti sorrendben.

Transzponáljuk a gráfot, ezen mélységi bejárás a sorrend szerint, az így talált mélységi fák a komponensek.



## Minimális feszítőfák



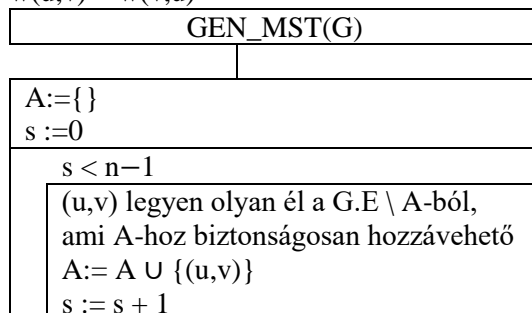
Kössünk össze minden várost, hogy minden pontba eljusson az áram, de a lehető legolcsóbban (és még sok más megfogalmazású feladat)

A lényeg minimális feszítőfa keresése.

$G=(V,E)$   $E \subseteq V \times V$   $(u,v) \in E \Leftrightarrow (v,u) \in E$  /\* $u \neq v$ , a hurokéleknek nincs értelme\*/

$w: E \rightarrow \mathbb{R}$  élsúlyok

$w(u,v) = w(v,u)$



azaz  $A \cup \{(u,v)\}$  minimális feszítőfa része marad  
 $\mathcal{P}$  invariáns  $= \exists T = (V, T_E) \text{ MST, hogy } A \subseteq T_E$

Def1:  $G=(V,E)$ ,  $\emptyset \subsetneq S \subsetneq V$  esetben  $(S, V \setminus S)$  vágás a  $G$  gráfban

Def2:  $A \subseteq E$  és  $(S, V \setminus S)$  vágás esetén a vágás elkerüli az  $A$ -t  $\Leftrightarrow A$  egyetlen éle sem keresztezi a vágást

Def3:  $(u,v)$  él keresztezi az  $(S, V \setminus S)$  vágást  $\Leftrightarrow$  az  $u \in S$  és  $v \in V \setminus S$  (vagy fordítva  $u \in V \setminus S$  és  $v \in S$ )

Tétel:  $G = (V, E)$  irányítatlan  $w: E \rightarrow \mathbb{R}$ -rel élsúlyozott gráf  
 $(S, V \setminus S)$  vágás a gráfban elkerüli  $A$ -t, ahol  $\exists T = (V, T_E)$  MST, hogy  $A \subseteq T_E$   
 $(u, v) \in E$  egy könnyű él (legkisebb költségű) a vágásban  
Ekkor  $(u, v)$  biztonságos  $A$ -ra nézve

Bizonyítás:

a)  $(u, v) \in T_E$  ✓

b)  $(u, v) \notin T_E \Rightarrow (p, q) \in T_E$ , ami keresztezi az  $(S, V \setminus S)$  vágást, ui:  $T$  feszítőfa, tehát  $T$ -ben el lehet jutni  $u$ -ból  $v$ -be.

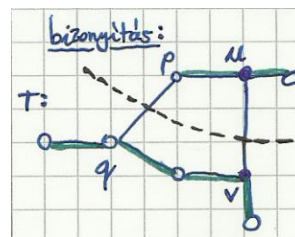
$\Rightarrow w(p, q) \geq w(u, v)$

$(p, q)$  törlésével az MST szétesik, de ha ehhez  $(u, v)$ -t hozzávesszük, akkor újra feszítőfa lesz.

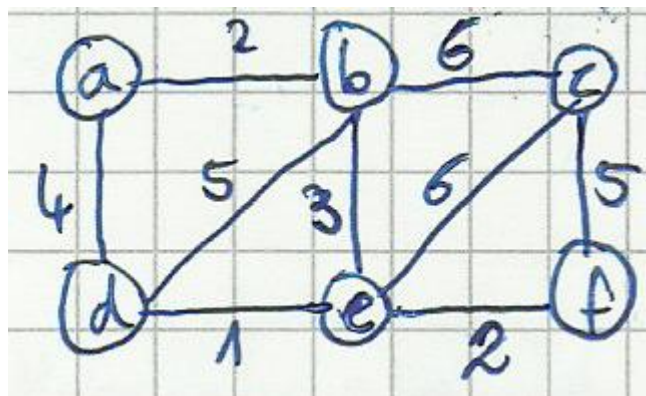
$T' = T \setminus \{(p, q)\} \cup \{(u, v)\}$

$w(T') = w(T) - w(p, q) + w(u, v) \leq w(T)$

viszont mivel  $T$  MST volt,  $w(T') \geq w(T) \Rightarrow$  azaz  $w(T') = w(T)$  és  $T'$  is MST kell, hogy legyen...



## Minimális feszítőfa ismételése példán keresztül


 $G=(V,E)$ 
 $w: E \rightarrow \mathbb{R}$ 

egy vágás:

S	V\S
{a,b,d}	{c,e,f}

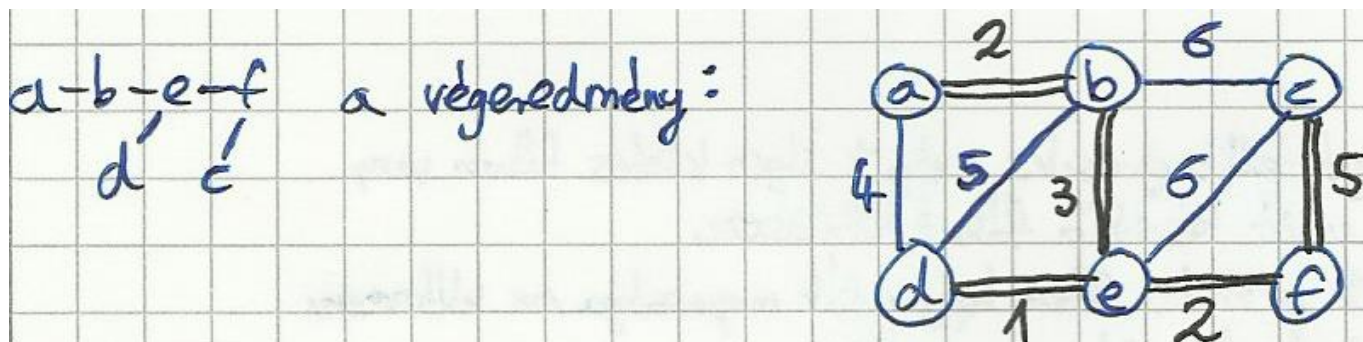
egy vágást keresztező él az  $E \cap (S \times (V \setminus S))$ 

egy eleme, ennek a halmaznak a legkisebb súlyú tagja a könnyű él

A	S	V\S
$\emptyset$	{a, b, d}	{c, e, f}
d-e	{a, b, d, e, f}	{c}
d-e c   f	{a}	{b, c, d, e, f}
a-b d-e c   f	{a, b, d, e}	{c, f}
a-b d-e-f   c	{a, b}	{c, d, e, f}

Az új vágás a két élt kerülje el, ne menjen keresztül rajtuk, majd mindig választjuk a könnyű élt.

Az eddig kiszámolt feszítőfa-részlet (A) élhalmaza diszjunkt kell, hogy legyen a vágás élével.

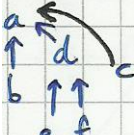


## Kruskal algoritmus

- Elsőként súly szerint sorba rendezzük a csúcsokat! Minden csúcs egy egyelemű fát képezzen!
- Majd minden lépésben hozzávesszük a minimális élt, ha külön komponenseket kötnek össze (ha nem, kihagyjuk)

0. lépés	1	2	3	4	5	6	7	8	9	végeredmény:
	d <sup>1</sup> e	a <sup>2</sup> b	e <sup>2</sup> f	b  3 e	a  4 d	<del>a<sup>2</sup>b</del> 5  3 d	c  5 f	b <sup>6</sup> c	<del>b<sup>6</sup>c</del> 5  3 e	
a b c d e f	a b c d <sup>1</sup> e f	a <sup>2</sup> b c d <sup>1</sup> e f	a <sup>2</sup> b c d <sup>1</sup> e <sup>2</sup> f	a <sup>2</sup> b c  3 d <sup>1</sup> e <sup>2</sup> f	a <sup>2</sup> b c  4  3 d <sup>1</sup> e <sup>2</sup> f kihagyjuk	<del>a<sup>2</sup>b c</del> 5  3 d <sup>1</sup> e <sup>2</sup> f kihagyjuk	a <sup>2</sup> b c  3  5 d <sup>1</sup> e <sup>2</sup> f	a <sup>2</sup> b <sup>6</sup> c  3  5 d <sup>1</sup> e <sup>2</sup> f kihagyjuk	<del>a<sup>2</sup>b<sup>6</sup>c</del> 5  3  5 d <sup>1</sup> e <sup>2</sup> f kihagyjuk	

A komponensek nyilvántartásához menetközben egy másik fát is kezelünk:

0	1 (d-e)	2 (a-b)	3 (f-d)	4 (b-e)	5	6	7 (c-f)	8	9
a b c d e f	a b c d f ↑ e	a d c ↑ ↑ b e f	a d c ↑ ↑ ↖ b e f	a ← d c ↑ ↑ ↖ b e f	(kihagyjuk)	(kihagyjuk)		(kihagyjuk)	(kihagyjuk)

MST_Kruskal(G,w)		$n-1 \leq e =  G.E  < n^2$
G.E mon. növ. rend. w szerint		$O(e \cdot \log e) = O(e \cdot \log n)$
$\forall u \in G.V$		$\Theta(n)$
Make_Set(u)		$\Theta(1)$
A := {}		
$\forall (u,v) \in G.E$ (w szerint mon. növ.)		
x := HOL_VAN(u)		$O(\log n)$ e-szer
y := HOL_VAN(v)		
$x \neq y$		
A := A ∪ {(u,v)}    Unió(x,y)		$O(n)$ (n-1-szer néhány konstans)
return(A)		

A HOL\_VAN() megállapítja, a csúcs melyik ilyen köztes fában van, ennek megfelelően az unió később a fákat köti össze. (Mindegyik fát a gyökércsúcsa azonosítja.)

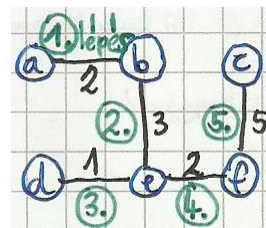
Az új élt úgy vesszük a köztes fához, hogy a fák magassága ne változzon, de a komponenshez tartozás jelezve maradjon. (Az alacsonyabb fa gyökerét a nagyobb fa gyökere alá kötjük be.)

A fák magassága legfeljebb  $\log_2 n$  ( $n = |G.V|$ )

Közös magasság esetén az uniózott fa magassága  $h+1$  lesz (h közös magasság volt)

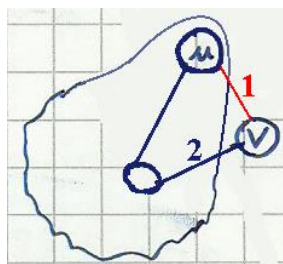
## Prim algoritmus

- Kiindulva egy csúcsból és a (csúcs, maradék) vágás közé választja a minimális élt
- Ezt hozzáveszi az eredményhez és az (eddig eredmény, maradék) vágásban keresi a könnyű élt
- addig folytatva az előző két lépést, míg Minimális feszítőfát nem kapunk.



meg kell adni, hogy hol kezd (s), de mindegy, hogy hol kezd

MST_PRIM(G, s, w)	
$\forall u \in G.V$	
u.kulcs := ∞	$\Theta(n)$
u.π := ∅	
s.kulcs := 0	$\Theta(1)$
Q : PrSor(G.V, kulcs alapján)	$\Theta(n)$
Q ≠ ∅	$\Theta(n)$
u := Q.MIN_KIVESZ()	$n \cdot O(\lg(n))$
$\forall v \in G.Adj[u]$	$\Theta(n+e)$
$v \in Q \wedge v.kulcs > w(u,v)$	$\Theta(e)$
v.kulcs := w(u, v)	$O(e) + O((e) \cdot \lg(n))$ a kupac adminisztrációja, a kulcs váltás miatt
v. π := u	



Lehet, hogy az u hozzávétele után v közelebb kerül a részleges feszítőfához.

Az egész algoritmus műveletigénye:  $O((n+e) \cdot \lg(n))$ , de a gráf összefüggő ( $e \geq n-1$ ), azaz  $O(e \cdot \lg(n))$



kezdő csúcs: a

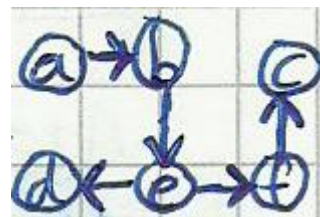
lépés	kulcs						$\pi$					
	a	b	c	d	e	f	a	b	c	d	e	f
start	0	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
a		2		4				a		a		
b	-		6		3				b		b	
e		-		1		2				e		e
d	-	-			-							
f			5		-				f			
c	nincs már több lehetséges szomszédja											
vége:	0	2	5	1	3	2	$\emptyset$	a	f	e	b	e

-: a szomszéd már korábban belekerült a fába

l: új, jobb távolság

prim optimalizációja:

Fibonacci kupacokkal le lehet csökkenteni a műveleti igényt:  $O(e+n \cdot \lg(n))$

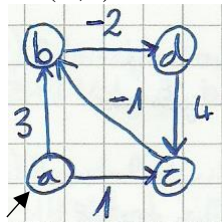


Az MST, amit kaptunk,  
irányított faként  
reprezentált  
IRÁNYÍTATLAN FA

## Legrövidebb út probléma

Adott egy tetszőleges hálózat, amiben az utaknak költségeik vannak, ezt a hálózatot egyszerű gráffal ábrázoljuk.

$G=(V,E)$   $w: E \rightarrow \mathbb{R}$



Negatív költség is lehetséges. Adott pontból az összes többi csúcsba meg szeretnénk határozni a legrövidebb utat a költségekkel, nem biztos, hogy a legkevesebb élből álló út a legrövidebb.

Mikor értelmes a kérdés:

- ha létezik a start csúcsból elérhető negatív kör, nincs legrövidebb út egyetlen a körből elérhető csúcsba sem.

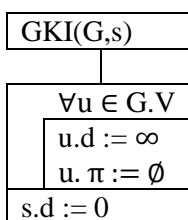
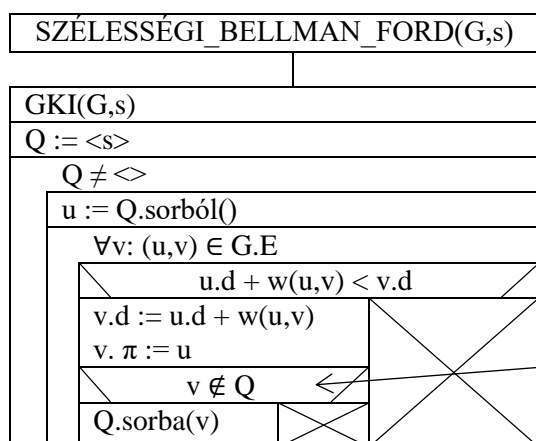
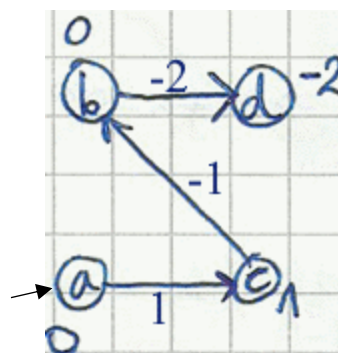
Ha a startcsúcsból csak nem negatív kör érhető el, az az útból elhagyható.

Optimális útnak tetszőleges részútja is optimális út.

Általános algoritmus: (Robert Endre Tarjan: Data Structures and Network Algorithms könyvben: "Breadth-first Scanning" algoritmus)

„Queue-based BELLMAN-FORD” (Sor alapú Bellman-Ford algo)

d	a	b	c	d	Q	$\pi$	a	b	c	d	menet
init	0	$\infty$	$\infty$	$\infty$	<a>		$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	
a		3	1		<b,c>			a	a		0.
b				1	<c,d>					b	1.
c		0			<d,b>			c			
d					<b>						2.
b				-2	<d>					b	
d					<>						3.
	0	0	1	-2			$\emptyset$	c	a	b	



a sorban levés ellenőrizhető pl. egy adattaggal, mindig bejárni a sort nem lenne hatékony.

Tulajdonság:

$\forall u \in G.V$ , hogy ha az u csúcsba vezet k élből álló  $s \rightarrow u$  optimális út  $\Rightarrow$  a k. menet elején már  $u.d = w(s \rightarrow u)$  és  $(s, \dots, u.\pi, u.\pi, u)$  egy optimális út

Lemma: s-ből nem érhető el negatív kör (amely összköltsége negatív)  $\Rightarrow$  tetszőleges u elérhető csúcsra létezik  $s \rightarrow u$  optimális út, ami legfeljebb n-1 élből áll.

T( Lemma és Tulajdonság következménye):

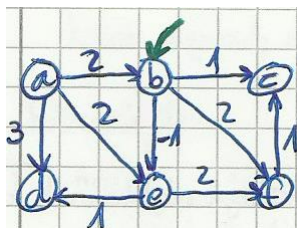
Ha s-ből nem érhető el negatív kör

$\Rightarrow$  tetszőleges u elérhető (s-ből) csúcsra létezik  $s \rightarrow u$  optimális út, és egy optimális út az n-1. menet elején már rendelkezésünkre áll (ki van számolva).

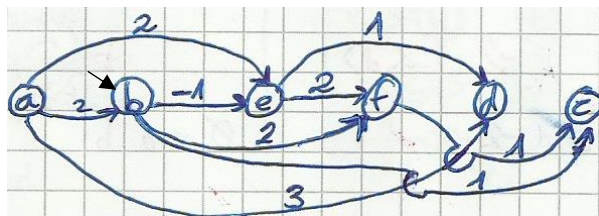
$\Rightarrow$  az n-1. menet végére kiürül a sor, az algoritmus  $O(n \cdot e)$  időben megáll.

Következmény: Ha az  $(n-1)$ . menet végén van a  $Q$  sorban elem  $\Leftrightarrow$  van  $s$ -ből elérhető negatív kör.  
 Megjegyzés: vegyünk egy ilyen  $w$  elemet. Ebből a  $\pi$  pointereken visszafele haladva megtalálható egy ismétlődő  $v$  elem, amire  $\exists k \in 1..n$   $\pi^k(v) = v$ , és  $(v, \pi^{k-1}(v), \pi^{k-2}(v), \dots, \pi(v), v)$  egy negatív kör.

### Legrövidebb utak egy forrásból, körmentes irányított gráfokon



feltétel: nincs irányított kör,  
 ekkor lehet topologikus rendezéssel optimális utakat keresni.

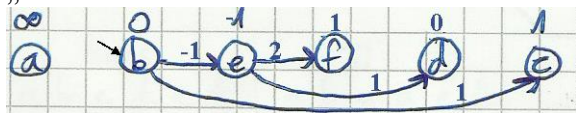


topologikus rendezés:

	d						$\pi$					
	a	b	c	d	e	f	a	b	c	d	e	f
init	$\infty$	0	$\infty$	$\infty$	$\infty$	$\infty$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
a												
b			1		-1	2		b		b	b	
e				0		1			e		e	
f												
d												
c												
vége:	$\infty$	0	1	0	-1	1	$\emptyset$	$\emptyset$	b	e	b	e

nem történik semmi, el kell jutni a start csúcsba

„a” nem volt elérhető

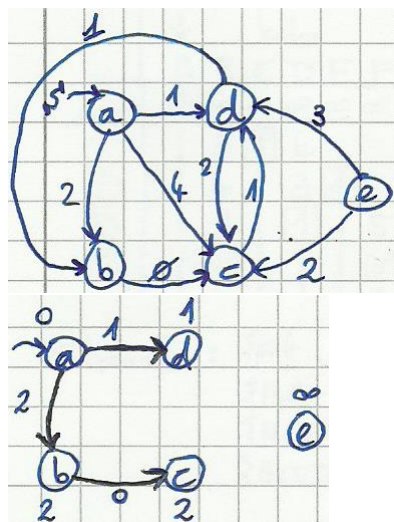


Műveletigény:  $\Theta(n+e)$  műveletigény (topologikus rendezés)  
 $\forall$  csúcs kiterjesztése ( $\Theta(n+e)$ )

Teljes műveletigénye:  $\Theta(n+e)$

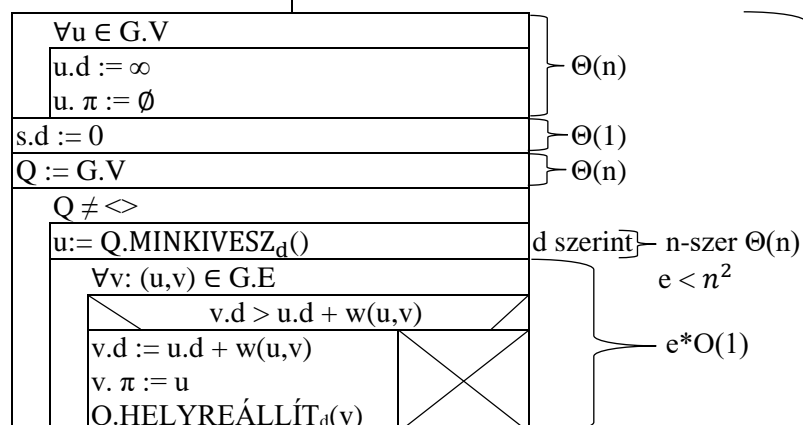
Legyen  $G=(V,E)$  akár irányított, akár irányítatlan,  
 $w: E \rightarrow \mathbb{R}_0$  nem negatív valós számok

## DIJKSTRA algoritmus



d	a	b	c	d	e	kiterjesztés	$\pi$	a	b	c	d	e
init:	0	$\infty$	$\infty$	$\infty$	$\infty$			$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
		2	4	1		a			a	a	a	
			3			d				d		
			2			b				b		
						c						
						e						
vége:	0	2	2	1	$\infty$			$\emptyset$	a	b	a	$\emptyset$

Dijkstra( $G, w, s$ )



$T(n) \in \Theta(n^2)$   
 ha a Q rendezetlen

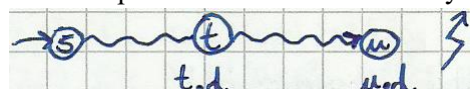
alsó index d azt jelenti, hogy a minkivesz a d értékei szerinti  
 szomszédsági éllista + bináris kupac esetén  $T(n, e) \in O((n+e) \cdot \log n)$

$Q.MINKIVESZ_d()$ :  $n \cdot O(\lg n)$   
 $Q.HELYPEÁLLÍT_d(v)$ :  $O(e \cdot \lg n)$

Áll: Amikor a csúcsot kivesszük a sorból, oda már optimális út vezet.

Bizonyítás:

start csúcsra  $\checkmark$ , a költség optimális. Tegyük fel, hogy nem igaz az Állítás! Legyen u az első ilyen csúcs, t pedig az  $s \rightarrow u$  optimális út első csúcsa amely  $\in Q$  sornak.



u-ra nem igaz, u-t válasszuk kiterjesztésre, ekkor nincs talált optimális út (de attól még létezik)

$u.d > w(s \rightarrow u)$ , mivel egyébként megtaláltuk volna az optimális utat.

$t.d \leq w(s \rightarrow t) \leq w(s \rightarrow u) < u.d$

az  $s \rightarrow t$  optimális út, mivel optimális út részútja.

$\therefore$  hiszen „t”-t megelőző csúcs kiterjesztésekor beállítottuk, de mivel  $t.d < u.d$ , ezért a következő lépés „t”-t választja kiterjesztésre  $\checkmark$

Minden csúcsból minden csúcsba legrövidebb út minden csúcsra futtatott Dijkstrával:

- ritka gráfra  $O(n \cdot (n+e) \cdot \log n) = O(n^2 \cdot \log n)$
- sűrű gráfra  $O(n^3 \cdot \log n)$ , nem jó választás...
  - ehelyett a vektorpáros Dijkstra stabil  $O(n^3)$ -öt ( $n \cdot O(n^2)$ ) fut.

## Floyd–Warshall - algoritmus

Engedjük meg a negatív élsúlyt, de ne lehessen negatív kör.

$G=(V, E)$

$G: V = 1..n$

csúsmátrixos ábrázolás

$D_{ij} := i \rightarrow j$  optimális út költsége (végtelen, ha nincs optimális út)

$\pi_{ij} := i \rightarrow j$  optimális úton  $j$  szülője

$D_{ij}^{(k)}$  = az  $i \rightarrow j$  úton az  $[1..k]$  indexű csúcsok lehetnek csak közbenső csúcsok.

$$\min\{w(i \xrightarrow[k]{v} j)\}$$

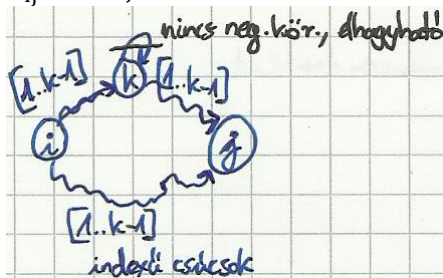
$\pi_{ij}^{(k)}$  = egy ilyen  $[1..k]$  közbensős úton a  $j$  csúcsok szülője

$$D_{ij}^{(0)} = \begin{cases} 0, & \text{ha } i = j \\ w(i,j), & \text{ha } (i,j) \in G.E \wedge i \neq j \\ \infty, & \text{ha } i \neq j \wedge (i,j) \notin G.E \end{cases} \quad (\text{közvetlen út szülője})$$

$$\pi_{ij}^{(0)} = \begin{cases} \emptyset, & \text{ha } i = j \\ i, & \text{ha } i \neq j \text{ és } (i,j) \in G.E \\ \emptyset, & \text{különben} \end{cases}$$

$$D_{ij}^{(n)} = D_{ij}$$

$$\pi_{ij}^{(n)} = \pi_{ij}$$



$$\text{ha } D_{ij}^{(k-1)} > D_{ik}^{(k-1)} + D_{kj}^{(k-1)}, \text{ akkor } D_{ij}^{(k)} = D_{ik}^{(k-1)} + D_{kj}^{(k-1)}$$

$$\text{különben: } D_{ij}^{(k)} = D_{ij}^{(k-1)}$$

hasonlóan  $\pi_{ij}^{(k)}$  is ha nem tudtuk az utat javítani  $= \pi_{ij}^{(k-1)}$

ha tudtuk  $= \pi_{kj}^{(k-1)}$

$$D_{ik}^{(k)} = D_{ik}^{(k-1)}$$

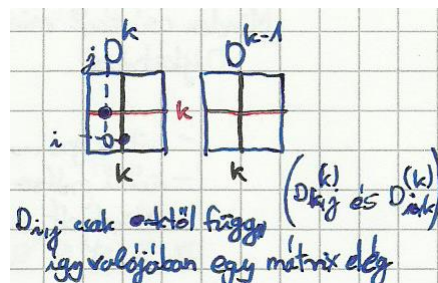
$$D_{kj}^{(k)} = D_{kj}^{(k-1)}$$

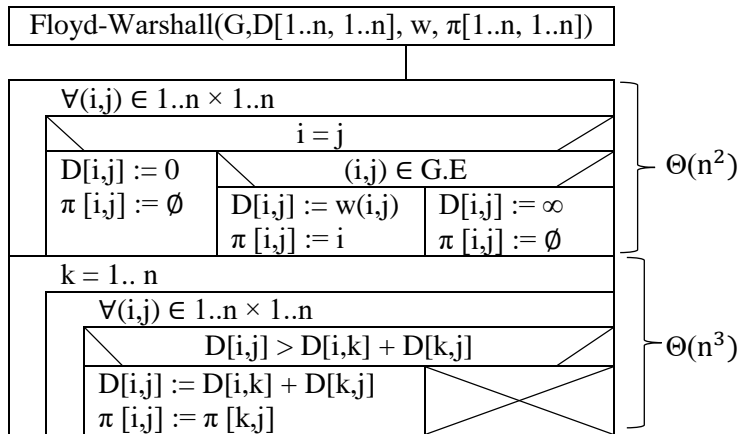
Az algoritmus lépésenként előállítja mátrixpárok sorozatát.

$$D^{(k)} = \left( D_{ij}^{(k)} \right)_{i,j=1}^n \quad k = 0, \dots, n-1, n+1 \text{ db mátrix.}$$

$D$  és  $\pi$   $k$ -edik sora és oszlopa a  $(k-1)$  és  $k$ . lépésben egyenlő:

$D$   $i$ -edik sora:  $i$ . csúcsból optimális utak hossza.





$$T_{FW} \in \Theta(n^3)$$

Sűrű gráfokon az aszimptotikus műveletigény, mint a Dijkstra, de ritka gráfokon a Dijkstra aszimptotikusan jobb (feltéve hogy  $\nexists$  negatív él, és így a Dijkstra is alkalmazható).

Megjegyzés: A negatív körök a D főátlójában negatív számként jelennek meg.

### Tranzitív lezárt algoritmus (Warshall algoritmus)

$T_{ij}$ : van-e út i-ből j-be (megállapítja, hogy van-e a két csúc között út (csak az számít, hogy van-e, hogy milyen költségű az nem számít))

$T_{ij}^{(k)} \Leftrightarrow i \xrightarrow{\vee}^k j$  megint csak legfeljebb az  $1..k$  csúcsokat érintő út.

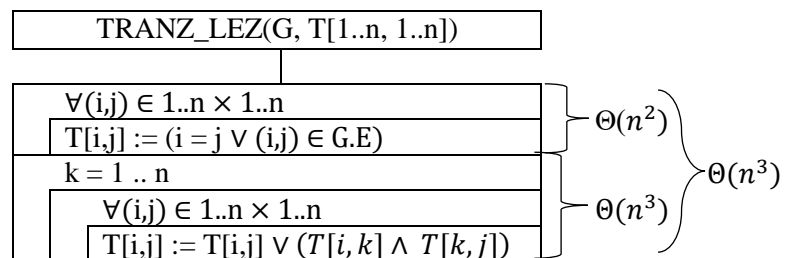
$T_{ij}^{(0)} = \begin{cases} \uparrow & , \text{ ha } i = j \vee (i,j) \in G.E \\ \downarrow & , \text{ különben} \end{cases}$

$$T_{ij}^{(k)} = T_{ij}^{(k-1)} \vee (T_{ik}^{(k-1)} \wedge T_{kj}^{(k-1)})$$

$$T_{ij}^{(n)} = T_{ij}$$

$$T_{ik}^{(k)} = T_{ik}^{(k-1)}$$

$$T_{kj}^{(k)} = T_{kj}^{(k-1)}$$



Sűrű gráfokra sokkal jobb, mint  $n$  db szélességi bejárás.

Ritka gráfokra ( $e \in O(n)$ ) viszont  $n$  darab szélességi bejárás műveletigénye:  $O[n * (n + e)] = O(n^2)$  aszimptotikusan kisebb, mint a  $\Theta(n^3)$  Warshall algo-nál.

$$e \in O(n)$$

Sűrű gráfokra  $T_{n*DFS}(n) \in O(n * (n + e)) = O(n^3)$ .

$$e \in \Theta(n^2)$$

Ez hasonló a Warshall-hoz, de annak kisebb a konstans szorzója.

## Mintaillesztési feladat

	1	2	3	4	5	6	7	8	9	10	11
T[1..11] =	A	B	A	B	B	A	B	A	B	A	B
	B	A	B	A							
		B	A	B	A						
			B	A	B	A					
				B	A	B	A				
					B	A	B	A			
						B	A	B	A		
							B	A	B	A	
								B	A	B	A

P[1..4] = BABA

piros betű: annál a betűnél található az első nem egyező karakter

zöld betű: egyező karakter

zöld szó: a keresett minta megtalálható a szóban

s = 4 eltolás

két találat

s = 6 eltolás

Fa: T[1..n]:  $\Sigma$ ; P[1..m]:  $\Sigma$   $0 < m \leq n$ S := { s  $\in$  0..n-m | T[s+1..s+m] = P[1..m] } programozása

Egyszerű_mintaillesztő_alg(T[1..n], P[1..m], S)		
S := { }		
s := 0..n-m		
<table border="1"> <tr> <td>T[s+1..s+m] = P[1..m]</td></tr> <tr> <td>S := S <math>\cup</math> {s}</td></tr> </table>	T[s+1..s+m] = P[1..m]	S := S $\cup$ {s}
T[s+1..s+m] = P[1..m]		
S := S $\cup$ {s}		

T[s+1..s+m] = P[1..m]
j := 1
j $\leq$ m és T[s+j] = P[j]
j := j+1
return (j > m)

O(m)  
 MT(m)  $\in$   $\Theta(m)$   
 mT(m)  $\in$   $\Theta(1)$

Az egész eljárás műveletigénye:

MT(n,m)  $\in$   $\Theta((n-m+1)*m) = \Theta((n-m)*m)$ mT(n,m)  $\in$   $\Theta(n-m+1) = \Theta(n-m)$   $\xrightarrow{n > m \text{ esetén}}$ 

Megjegyzés:

Gyakran feltehető:  $n \gg m$ Ekkor: MT(n,m)  $\in$   $\Theta(n*m)$ mT(n)  $\in$   $\Theta(n)$ 

Elég jónak tűnhet, de nagy inputra hamar elszáll, kell ennél jobb algoritmus.

## QUICK SEARCH

Alapfilozófiája példán illusztrálva:

Pl:  $\Sigma = \{A, B, C, D\}$

[illegible]

P = CADA

**piros betű:** a keresett minta nem található

**zöld szó:** a keresett minta megtalálható a szóban

⊙

eltoljuk annyival, hogy a szövegnek a mintán túli első karaktere (jel.X) illeszkedjen a mintában az (X) karakter jobb szélső előfordulására.

ha mintákon nem szereplő input karakter jön, teljes hosszon átugorjuk, hiszen úgysem illeszkedne

ABCD

shift: 1 5 4 2    adott betű esetén annyit kell lépni, hogy a minta illeszkedhessen, ezek a mintához specifikusak

---

Init(shift[ $\Sigma$ ] : 1..n+1, P[1..m] :  $\Sigma$ )

$\forall \delta \in \Sigma$	}	$\Theta(d)$
shift[ $\delta$ ] := m + 1		
j = 1 .. m	}	$\Theta(m)$
shift[P[j]] := m + 1 - j		

először feltesszük, hogy  $\Sigma$  ábécé összes betűjét át kell ugrani.

j	shift[P[j]]
1	m
2	m-1
3	m-2
	...

⑥

hogy a  $P[1]$  kerüljön  $X=T[s+m+1]$  alá, a mintát a teljes hosszával el kell tolni, hogy  $P[2]$  kerüljön az  $X$  alá, csak egyet, kevesebbet kell tolni, stb...

QuickSearch(T[1..n]:  $\Sigma$ ; P[1..n]:  $\Sigma$ ; S)

Init(shift[Σ], P[1..m])	$\Theta(m + \overbrace{ \Sigma }^d) = \theta(m+d) \xrightarrow{\text{arrow}} \Theta(m)$
$s := 0$ $S := \{\}$	
$s + m \leq n$ <div> <div><math>T[s+1..s+m] = P[1..m]</math></div> <div> <math>S := S \cup \{s\}</math> </div> </div> $s+m < n$ <div> <math>s := s + \text{shift}[T[s+m+1]]</math> <math>s := s + 1</math> </div>	$mT(n, m) \in \Theta\left(\frac{n}{m+1}\right) \leftarrow \text{pl:}$ $MT(n, m) \in \Theta((n-m+1)*m) \leftarrow$

d konstans

## RABIN-KARP

$$\Sigma = \{\sigma_1, \dots, \sigma_d\}$$
$$\varphi: \Sigma \rightarrow 0..d-1$$
$$\varphi(\sigma_i) = i-1$$

$P[1..m] \sim \overbrace{\sum_{j=1}^m \varphi(P[j])d^{m-j}}^{P_0}$ : d számrendszerbeli szám, Horner elrendezéssel:  $\Theta(m)$  időben számítható.

$$T[s+1..s+m] \sim T_s = \sum_{i=1}^m \varphi(T[s+j])d^{m-j}: \text{Horner elrendezéssel: } \Theta(m) \text{ időben számítható.}$$



A keresett eltolásokat  $S := \{s \in 0..n-m \mid T_s = P_0\}$  alakban oldjuk meg, a  $d$ -számszerű számok egyenlőségét vizsgáljuk.

A számítás a Horner mellett tovább egyszerűsödik:

$$T_{s+1} = \sum_{j=1}^m \varphi(T[s+1+j])d^{m-1} = (T_s - \varphi(T[s+1])d^{m-1})d + \varphi(T[s+m+1]) : \Theta(1) \text{ időben,}$$

ha  $d^{m-1}$ -et előre kiszámoltuk.

$$T_s = \begin{bmatrix} 1 & 2 & 8 & 25 \end{bmatrix}$$

$$T_{s+1} = \begin{bmatrix} 2 & 8 & 25 & 14 \end{bmatrix}$$

$T_0$  és  $P_0$  számítása  $\Theta(m)$  és  $\Theta(m)$  idejű a rekurzió  $\Theta(1)$  idejű.

A teljes műveletigény így  $\Theta(m) + (n-m) * \Theta(1)$ , ami  $\Theta(n)$  igényű.

ha a rekurzió tényleg  $\Theta(1)$  és a  $T_i, P_0$  számokat a gép tudja hatékonyan ábrázolni.

Mi lesz a nagy mintákkal?

$p := P_0 \bmod q$  ( $q$  „nagy” prím)

$t_s := T_s \bmod q$

$h := d^{m-1} \bmod q$

$$t_{s+1} = (t_s - \varphi(T[s+1])h)d + \varphi(T[s+m+1]) \bmod q$$

a különbség átcsoportosíthat negatívba...

$$t_{s+1} = \left( \left( \frac{(t_s + dq - \varphi(T[s+1])h) \bmod q}{<(d+1)*q} \right) * d + \varphi(T[s+m+1]) \right) \bmod q$$

$$\Rightarrow t_s = \left( ((t_{s-1} + dq - \varphi(T[s])h) \bmod q) * d + \varphi(T[s+m]) \right) \bmod q =: \Delta$$

hogy jól működjön  $(d+1)q \leq$  legnagyobb ábrázolható szám kell legyen (`size_t`, `Integer'Last'`...)

Lehet hamis találat (fals pozitív), ekkor ellenőrizni kell, hogy valódi találat-e, eredeti karakterekkel.

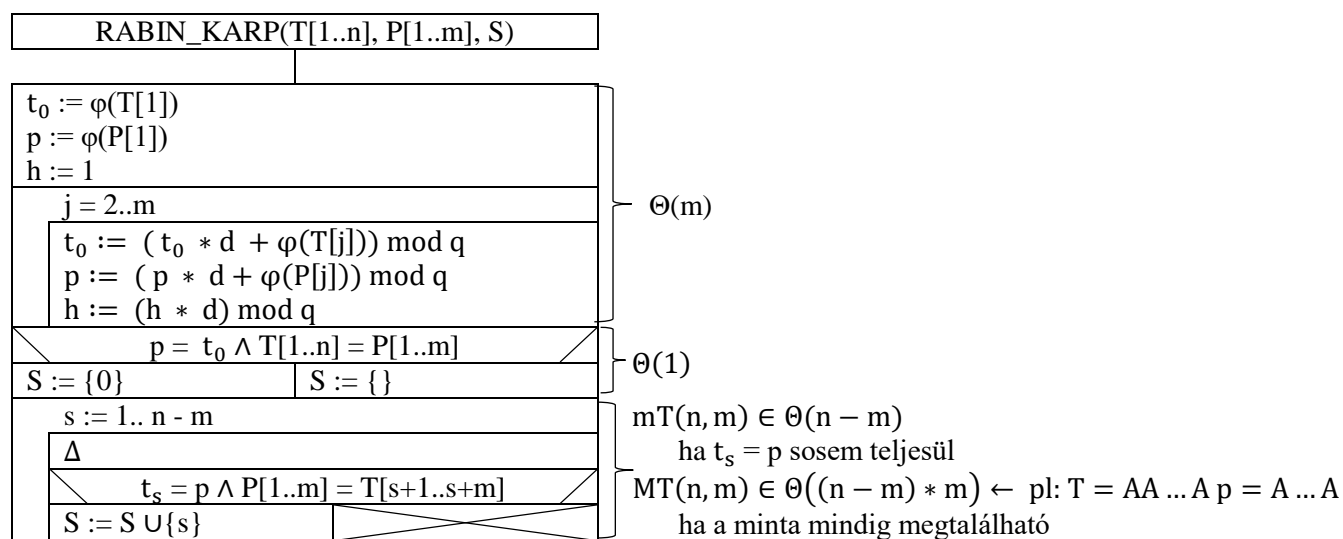
## RABIN-KARP folytatása

$q = \frac{\max N}{d+1}$  ← legnagyobb ábrázolható N szám, prímszám használatával a mod q számítások nem fognak túlszordulni

$$p \neq t_s \Rightarrow P_0 \neq T_s$$

$p = t_s \nRightarrow P_0 = T_s$  így potenciális találat esetén a valódi találatot ellenőrizni kell.

a modulo számítás miatt  
elveszik az információ



teljes algoritmus:

$$mT_{RK} \in \Theta(n)$$

$$MT_{RK} \in \Theta((n - m + 1) * m)$$

A gyakorlatban nincs túl sok valódi találat, nagy q esetén a hamis találatok száma se túl sok.

Így  $AT_{RK} \sim mT_{RK}$  (nem biz)

## Mintaillesztés lineáris időben

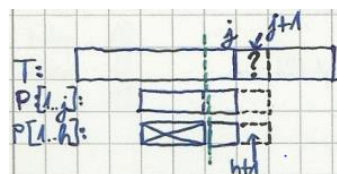
Példa:

$P[1..8] = BABABBAB$

$T[1..18] =$ 

A	B	A	B	A	B	A	B	A	B	A	B	A	B	A	B	A	B
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

eltoljuk a mintát annyival, hogy a minta illeszkedő része az eltolás után illeszkedjen részben a szövegre



Keressük a legnagyobb olyan h-t, hogy  $P[1..h]$  suffixe  $T[1..i]$  és  $h < j$

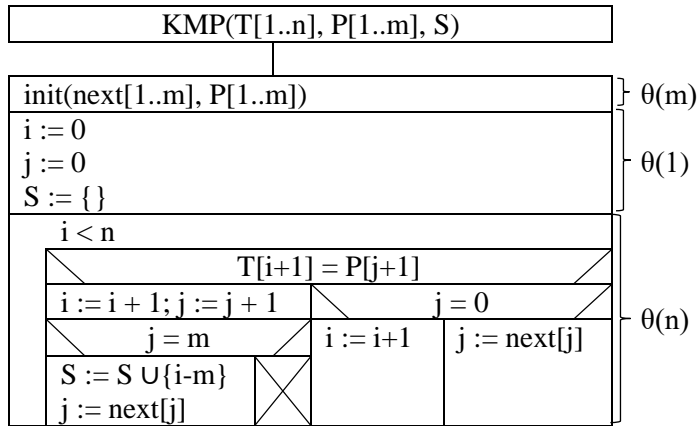
$T[1..i]_j$  hosszú suffixe =  $P[1..j] \Rightarrow T[1..i]_h$  hosszú suffixe =  $P[1..j]_h$  hosszú suffixe  $\Rightarrow P[1..h]$   
 ilyen feltételt kielégítő h-t keresünk.

Ez az információ csak a minta ismeretében megkapható.

Def:  $\text{next}(j) = \max\{h \in [0..j-1] \mid P[1..h] \supset P[1..j]\}$

$\text{next}[1..m] := \text{next}(1..m)$

Ezt kiszámolva úgy toljuk el a mintát, hogy  $p[1..\text{next}(j)]$  legyen a  $T[1..i]$  hosszú suffixe alatt, mivel korábban biztos nem lehet illeszkedés.



$$0 \leq i \leq n \wedge 0 \leq j < m \wedge i \geq j \wedge P[1..j] \supset T[1..i] \wedge S = \{s \in 0..i-m \mid P[1..m] = T[s+1..s+m]\}$$

$\text{next}^k(j) = P[1..j]$  k-adik leghosszabb illeszkedő prefixe

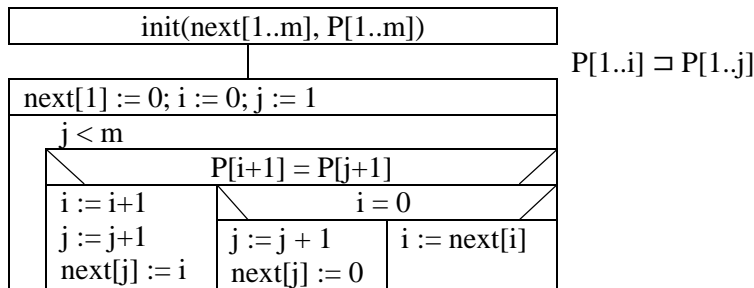
A fő ciklus  $T \in \Omega(n)$ , mivel  $i$  növekedni egyesével tud, és  $n$ -ig nő  $\Rightarrow$  biztos van  $n$  lefutás

$t(i,j) = 2i - j \in [0..2n]$  szig mon növény mind a négy ágon, de így a főciklus legfeljebb  $2n$ -szer fut le  $\Rightarrow T \in O(n)$

$$0 \leq j \leq i \leq n$$

$\Downarrow$

$T_{\text{fő}} \in \theta(n)$



most is a  $2j-i \in [0..2n]$ , az ágakon szigorúan monoton növény mindig így legfeljebb  $2m$ , legalább  $m$ -szer fut le  $\Rightarrow \Theta(m)$

## 1011<sub>2</sub> EA

Folytatás init stuki:

Def:  $\text{next}: [1..m] \rightarrow [0..m-1] \quad (m \in \mathbb{N}_+)$

$\text{next}(j) = \max\{h \in [0..j-1] \mid P[1..h] \supset P[1..j]\}$

$x \sqsubset y \Leftrightarrow xz = y \quad (\exists z)$

$x \supset y \Leftrightarrow zx = y \quad (\exists z)$

$x$  ps-párosa  $y$ -nak  $\Leftrightarrow x \neq y \wedge x \sqsubset y \wedge x \supset y$  /\*ps: prefix-suffix\*/

Tulajdonságok:

1.  $P[1..h]$  ps – párosa  $P[1..j]$  – nek  $\Leftrightarrow 0 \leq h < j \wedge P[1..h] \supset P[1..j]$
2.  $P[1..i+1] \supset P[1..j+1] \Leftrightarrow P[1..i] \supset P[1..j] \wedge P[i+1] = P[j+1]$
3.  $0 \leq \text{next}(j) < j$  Köv:  $j > \text{next}(j) > \text{next}^2(j) > \dots > \text{next}^k(j) = 0$   
↖ valamely  $k > 0$ -ra
4.  $0 \leq \text{next}(j+1) \leq \text{next}(j) + 1$
5.
  - a.  $\text{next}^k(j)$  értelmezve van  $\Leftrightarrow \exists$  a  $P[1..j]$ -nek a  $k$ . leghosszabb ps-párosa
  - b.  $\text{next}^k(j)$  értelmezve van  $\Rightarrow P[1..\text{next}^k(j)]$  a  $P[1..j]$   $k$ . leghosszabb ps-párosa

Q:  $m > 0$

/\*előfeltétel\*/

R:  $\text{next}[1..m] = \text{next}(1..m)$  /\*utófeltétel\*/

Inv:  $0 \leq i < j \leq m \wedge \text{next}[1..j] = \text{next}(1..j) \wedge P[1..i] \supset P[1..j] \wedge$  (a  $P[1..j]$  tetszőleges  $i$ -nél hosszabb  $P[1..i]$  ps-párosára:  $P[i+1] \neq P[j+1]$  ( $0 \leq i < j$ ))

$2j-i \in 2..2m$

2-ről indul, szig. mon. nő

}  $\Rightarrow$  a ciklus legfeljebb  $(2n-2)$ -szer hajtódik végre  
a ciklus legalább  $(n-1)$ -szer végrehajtódik

Példa:

j	1	2	3	4	5	6	7	8
P[j]	A	B	A	B	B	A	B	A
next[j]	0	0	1	2	0	1	2	3

Az algoritmus működése lépésről-lépésre:

			1	2	3	4	5	6	7	8
i	j	next[j]	A	B	A	B	B	A	B	A
0	1	0		A						
0	2	0			A					
1	3	1			A	B				
2	4	2			A	B	A			
0	4	üres mező					A			
0	5	0						A		
1	6	1						A	B	
2	7	2						A	B	A
3	8	3								

## Tömörítés

Naiv módszer:

$T[1..n]: \Sigma \quad \Sigma = \{\sigma_1, \dots, \sigma_d\}$

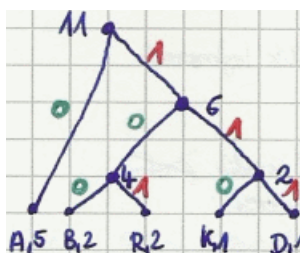
$\lceil \lg d \rceil$  bittel kódolható a karakter

$n * \lceil \lg d \rceil$  bittel kódolható a szöveg

Huffman kód: egyfajra prefix kód

a szöveg: ABRAKADABRA

jel	előfordulás	kód
A	5	0
B	2	100
R	2	101
K	1	110
D	1	111



Megjegyzés: A karakterenként kódoló tömörítések között a Huffman kód hossza minimális.

Tömörített kód:

Kitömörítés kódfa alapján

0 100 101 0 110 0 111 0 100 101 0  
 A B R A K A D A B R A

$$\Sigma = \{a, b, c\}$$

IN:	a	b	a	b	c	b	a	b	a	b	a	a	a	a	a
OUT	1	2	4	3	5	8	1	10	11	1					

szó	kód
a	1
b	2
c	3
ab	4
ba	5
abc	6
cb	7
bab	8
baba	9
aa	10
aaa	11
aaaa	12

szó	kód
a	1
b	2
c	3
ab	4
ba	5
abc	6
cb	7
bab	8
baba	9
aa	10
aaa	11
aaaa	12

### Rekonstruálás:

IN'	1	2	4	3	5	8	1	10	11	1
OUT'	a	b	ab	c	ba	bab	a	aa	aaa	a

8-as kód nincs a szótárban!

generáláskor a 8-as kód a ba... szövegből jött létre

$$\underbrace{ba}_{5} bab \rightarrow \underbrace{baba}_{8}..?$$

(hasonlóan járunk el a 10-es, 11-es esetben is)

az utolsó sor dekódoláskor  
legtöbbször felesleges már

LZW\_COMPRESS (In, Out,  $\Sigma$ )
$$\Sigma = \{\sigma_1, \dots, \sigma_d\} \text{ \u00e1b\u00e9c\u00e9}$$

s: string

c: char

D: szótár, „string: kód” alakú rendezett párosok halmaza

ŝc: az s string és a c char konkatenáltja

$\sigma_i$ : a  $\sigma_i$  betűből álló string (egybetűs string)

kód(D,s) = a D szótárban az s string kódja

$\text{eof}(\text{In}) \Leftrightarrow \text{az In inputról } \forall \text{ karaktert beolvastunk}$

„s: ” olyan rendezett páros, aminek az első komponense s

⊙ Az így jelzett utasítások csak akkor hajthatók végre, ha még kód  $\leq$  MAXKÓD, ahol MAXKÓD a kódok előre rögzített hosszától függ.

Pl: ha ez 12bit, akkor MAXKÓD =  $2^{12}-1 = 4095$ .

```
write(Out, kód(D, s))
```

LZW DECOMPRESS(In, Out,  $\Sigma$ )

s,t: string

k, kód: kódok

$t_1$ : a `t` string első betűje

string(D,k)= a D szótárban k kódnak megfelelő string

$\text{eof}(\text{In}) \Leftrightarrow$  az In inputról  $\forall$  kódot beolvastunk

„:k” olyan rendezett páros, aminek a 2. komponense k

⊗: Itt  $k = \text{kód teljesül}$

Megjegyzés: Balra, az elágazásban az „ $(\_k) \in D$ ” feltétel helyén „ $k < \text{kód}$ ” is állhatna (így egy kicsit gyorsabb lenne az algoritmus)

Felhasznált segédanyagok:

- Az előadáson készített jegyzetem (Koru),
- Whisperity előadás jegyzete (a jegyzetben található összes kép és néhány szöveges kiegészítés).
- <http://aszt.inf.elte.hu/~asvanyi/ad/sor%20alapu%20Bellman-Ford%20alg.pdf>
- <http://aszt.inf.elte.hu/~asvanyi/ad/Lempel-Ziv-Welch%20alg.pdf>

Külön köszönet Dr. Ásványi Tibor tanár úrnak a jegyzet alapos átolvasásáért, a talált hibák jelzéséért és az anyag megértését segítő megjegyzéseikért.

A jegyzetet átolvasták és a talált hibákat jelezték, amiért köszönet:

- Bán Róbert
- Fekete Anett
- László Tamás
- Nagy Vendel
- Szabó Gergő
- Szécsi Péter
- Tőkés Anna