Algel e.a

Pl. n nagyságú bemenetekre $f(n)$ műveletet hajt végre, 10^{10} művelet/mp.

	$f(n)=n$	$f(n)=n^2$	$f(n)=2^n$	$f(n)=n!$
$n=10$	10^{-9}	10^{-8}	$\approx 10^{-7}$	$3{,}6\cdot 10^{-4}$
$n=100$	10^{-8}	10^{-6}	$\frac{2^{100}}{10^{10}} \approx \frac{(2^{10})^{10}}{2^{10}} > \frac{10^{30}}{10^{10}} = 10^{20}$ mp $\Downarrow$ 10^{13} év	
$n=10^6$	10^{-4}	100 mp = 1 perc 40 mp		
$n=10^9$	10^{-1}	10^8 perc ≈ 3 év		

$f(n)=n!$ $\quad n=20 \Rightarrow >7$ év

$f(n)=2^n$ $\quad n=20 \Rightarrow 10^{-4}$ mp

$\quad n=40 \Rightarrow \sim 2$ perc

Pl.

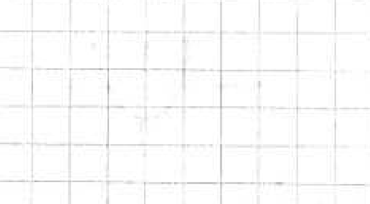
$n=20$ $\quad$ 6 mp $\quad$ $n=400$

$f(n) = c\cdot n$ $\quad 6\cdot 20 \approx 2$ perc

$= c\cdot n^2$ $\quad 6\cdot 20^2 \to 40$ perc

$= c\cdot \log n$ $\quad 2\cdot 6 = 12$ mp

$= c\cdot 2^n$ $\quad c\cdot 2^{400} = 6\cdot 2^{380} > 6\cdot (2^{10})^{38} > 6\cdot 10^{114} = \underbrace{6\cdot 10^{7}}_{2\text{ év}}\cdot\underbrace{10^{80}}_{\text{részecskék száma az univerzumban}}\cdot 10^{27}$



Nagyságrendek

$f, g : \mathbb{N} \to \mathbb{R}$

$f(n) = O(g(n))$ kiejtve $f(n)$ egyenlő nagy ordó $g(n)$, ha $\exists\, c>0$, $n_0>0$: $|f(n)| \le c\cdot|g(n)|$ $\quad \forall n \geq n_0$

$f(n) = \overset{\text{omega}}{\Omega}(g(n))$, ha $\exists\ c>0,\ n_0>0 : |f(n)| \geq c \cdot |g(n)| \quad \forall n \geq n_0$

$f(n) = \overset{\text{théta}}{\Theta}(g(n))$, ha $f(n) = O(g(n))$ és $f(n) = \Omega(g(n))$, azaz

$\exists\ c_1, c_2 > 0,\ n_0 > 0 : c_1|g(n)| \leq |f(n)| \leq c_2|g(n)| \quad \forall n \geq n_0$

Pl. $f(n) = 3n^2 - 100n + 6$

$f(n) \stackrel{?}{=} O(n^2)$ $\quad |3n^2 - 100n + 6| \leq c \cdot n^2$, ha $n \geq n_0$

$n > 100 \Rightarrow f(n) > 0$, az abs. é. elhagyható

$3n^2 - 100n + 6 \leq 3 \cdot n^2$ ✓

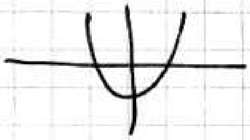
$c = 3,\ n_0 = 100$ jó

$f(n) \stackrel{?}{=} O(n^3)$ $\quad 0 < 3n^2 - 100n + 6 \leq 3n^2 < 3n^3$, ha $n \geq 100$

$c = 3,\ n_0 = 100$ jó

$f(n) \stackrel{?}{=} O(n)$ $\quad 0 < 3n^2 - 100n + 6 \leq c \cdot n \quad (n \geq 100)$

$3n^2 - (100 + c) \cdot n + 6 \leq 0$ egy adott küszöbtől $\forall$ n-re



de felfelé álló parabola $\Rightarrow$ valahol mindenképp átlép pozitívba

$f(n) \stackrel{?}{=} \Omega(n^2)$ $\quad 3n^2 - 100n + 6 \geq 2n^2 \quad n \geq 100 \quad c = 2$ jó

$f(n) \stackrel{?}{=} \Omega(n)$ $\quad$ HF

$f(n) = \Theta(n^2)$ $\quad$ a legnagyobb nagyságrendű tag határozza meg nagy n-ekre

$\log_a n = \Theta(\log_2 n)$ $\quad$ HF $\quad$ MOSTANTÓL $\forall$ LOGARITMUS KETTES ALAPÚ

Legyen $G(V, E)$ irányított gráf, nincs hurokél.

Forrás: $s \in V$ egy pontból se megy bele él.

Szuperforrás: $s \in V$ forrás és s-ből $\forall$ csúcsba megy él.

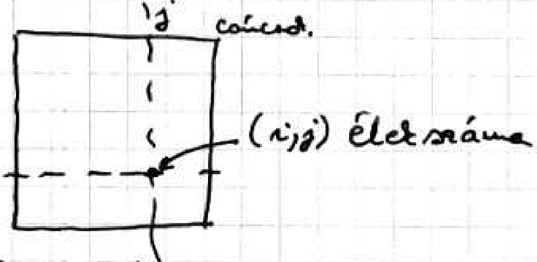
Algel

2009.02.11.
Friedl.

Feladat: találjunk szuperforrást (ha van)

Mj: legfeljebb 1 szuperforrás

megadás: G szomszédossági mtx



lépés: van-e i-ből j-be él $(A[i,j]=?)$

(1) alg. : $\forall$ pontra ellenőrizzük, hogy szuperforrás-e

1 ell: i. oszlopában 0 kell hogy legyen (mert nem megy bele él) } $\Rightarrow 2n-2$ lépés

i. sorában mindenütt kell elem legyen.

$\forall$ csúcsra: $n(2n-2) = 2n^2-2n = \Theta(n^2)$

(2) alg. $(i,j)\in E \Rightarrow j$ nem jó

$(i,j)\notin E \Rightarrow i$ nem jó

$i=1 \quad j=n \qquad V=\{1,2,\dots,n\}$

Ha $A(i,j)\neq 0 \Rightarrow j=j-1$

kül: $i=i+1$

Ismételjük, amíg $i\neq j$.

Ellenőrizzük, hogy i szuperforrás

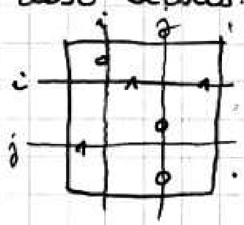
Lépésszám $\underbrace{(n-1)}_{\text{kiválasztás}} + \underbrace{2n-2}_{\text{ell}} = 3n-3 = \Theta(n)$

Minden algoritmus használ legalább $2n-2$ lépést (1 pont ellenőrzése)

$T(n)$: legjobb algoritmus lépésszáma $\rightarrow T(n)\geq 2n-2$

$T(n)\leq 3n-3$

Jobb alsó becslés: bármely algoritmus $n-2$ lépés után legalább 2 jelöltet hagy



A kettő közül valamelyikre $\leq \frac{n-2}{2}$ kérdés vonatkozott.

Ha ez a szuperforrás, akkor kell még $\geq 2n-2-\frac{n-2}{2}$.

Az össz lépésszám $\geq n-2+2n-2-\frac{n-2}{2} = 3n-4-\frac{n}{2}+1 = 2{,}5n-3 \Rightarrow T(n)\geq 2{,}5n-3$

Elágazás és korlátozás

ha ez már rossz, akkor ne folytassuk

(P1) $G = (V, E)$ egyszerű, irányítatlan gráf
FELADAT: max. fgtl ponthalmaz meghatározása

1. Algoritmus: kipróbálni ∀ ponthalmazt 2^n-féle lehetőség
↑ üreshalmaz

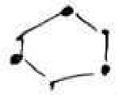
2. Algoritmus: MF (G) függvény a következőt csinálja
$v \in V$ csúcsot veszünk, legyen $\deg(v) \geq 3$
Ekkor a ponthalmaz v. tartalmazza v-t vagy nem.

Ha nem: $MF(G - v) = S_1$
Ha igen: $S_2 = MF(G - \{v,$ v szomszédai$\}) \cup \{v\}$

$S = \max\{S_1, S_2\}$

Ha $\deg(v) \geq 3$ nem teljesül, akkor kör vagy utak diszjunkt uniója

∀ ptl. csomópontot



∀ 2. pontot veszem

LÉPÉSSZÁM

$T(n)$ n csúcsú gráfnál max. hányszor hívódik meg MF

$$T(n) \le 1 + \underset{G-v}{T(n-1)} + \underset{G-v-\text{szomszédok}}{T(n-4)}$$

Állítás: $T(n) \le c^n$ (valamilyen $c>0$ konstans)

Tétel: Az algoritmus lépésszáma $O(1{,}381)^n$

Megjegyzés: $n=30$: $2^{30} \approx (2^{10})^3 \approx (10^3)^3 = 10^9$

$(1{,}381)^{30} < 20\,000$

Pl. $G=(V,E)$ egyszerű gráf, $|V|=n$

Feladat: kiszínezni a gráf csúcsait 3 színnel

1. algoritmus: $\forall$ csúcs 3-féle színt kaphat: 3^n lehetőség

$\frac{3^n}{3!}$ is jó, a színek permutációja miatt

2. algoritmus: Az egyik színosztály pontjait válasszuk ki, ezek legyenek mondjuk pirosak. 2^n lehetőség

Ezután a maradékot kell kiszínezni 2 színnel, ezt meg már tudjuk

Ha vmi gáz van, akkor nem színezhető ki 2 színnel

Látni fogjuk, h. a 2-színezés megy $O(n^2)$ lépésben.

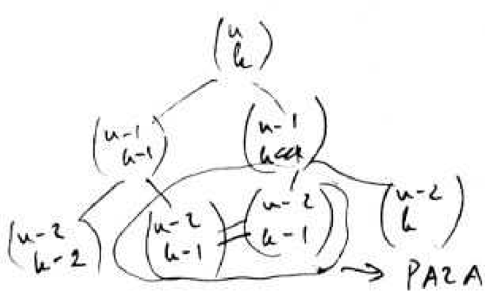
Lépésszám

$O(3^n \cdot n^2)$ ill. $O(2^n \cdot n^2)$

$n=30$-ra $2^n \cdot n^2 \approx 10^9 \cdot 3^2 \cdot 10^2 = 9 \cdot 10^{11}$

$3^n \cdot n^2 = 2 \cdot 10^{14} \cdot 3^2 \cdot 10^2 = 18 \cdot 10^{16}$

Pl. rekurzióra $\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$



→ PAZARLÁS, ha 2x hívjuk meg

Összesen: exponenciálisan sok hívás | 2003.02.14

Dinamikus programozás

Ha a megoldás vmilyen részfeladatok megoldásaiból álló: kiszámoljuk, de el is tároljuk az értékét.
Vmilyen táblázatkitöltéssel.

Pl. binom. együtthatóknál Pascal-Δ-et

$$\binom{0}{0}$$
$$\binom{1}{0} \quad \binom{1}{1}$$
$$\binom{2}{0}=1 \quad \binom{2}{1} \quad \binom{2}{2}=1$$

Hátizsák probléma

m db tárgy: $s_1, s_2, \ldots s_m$ súlyok
$v_1, v_2, \ldots v_m$ értékek

1 hátizsák: b súlykorlát

$s_i > 0 \ \forall i$-re $\quad s_i, v_i \in \mathbb{Z}^+$
$v_i > 0 \ \forall i$-re

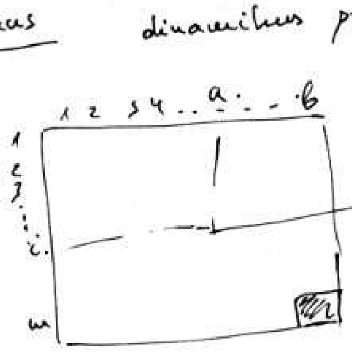
Feladat: kiválasztani a tárgyak egy részhalmazát, $I \subseteq \{1,2,\ldots m\}$:

$$\sum_{i\in I} s_i \le b$$
és $\sum_{i\in I} v_i$ maximális

Pl $m=3$ súlyok $2,2,3$ értékek $4,4,7$ súlykorlát: 4

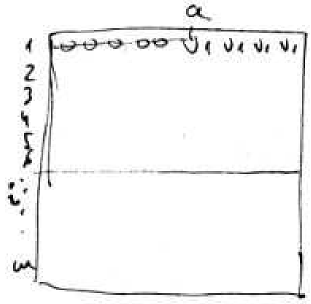
Mohó pakolás: $I=\{3\}$, érték $=7$
Van jobb: $I=\{1,2\}$, érték $=8$:)

1. Algoritmus: $\forall I$-t kipróbálunk, 2^m lehetőség
2. Algoritmus: dinamikus programozás



$T[i,a]$: $\{1,2,\ldots i\}$ tárgyak közül a súlykorlát $=a$, a max. érték

Ami kell: $T[m,b]$



$$T[1,a] = \begin{cases} 0, & \text{ha } s_1 > a \\ v_1, & \text{ha } s_1 < a \end{cases}$$

$T[i,a]$ = i. tárgyat elrakjuk v. nem $\overset{\text{nem}}{\underset{\text{igen}}{=}}$ a súlykorlát

$$T[i,a] = \begin{cases} T[i-1,a], & \text{ha } s_1 > a \\ \max\{\underbrace{T[i-1,a]}_{\text{nincs benne}}, \; v_i + T[i-1, \underbrace{a-s_i}_{\text{ennyi súlykorlát maradt}}]\} \end{cases}$$

Lépésszám

$m \times b$ elem

1 elem kitöltése: $O(1) \leftarrow$ vmi konstans

egész tábla: $mb \; O(1) = O(mb)$

Bemenet: $s_1, s_2, \dots s_m, v_1, \dots v_m, b$

Bemenet hossza: $\lceil \log(s_1+1) \rceil + \dots + \lceil \log(s_m+1) \rceil + \lceil \log(b+1) \rceil + \lceil \log(v_m+1) \rceil + \dots + \lceil v_1+1 \rceil$

$O(mb)$ $\lceil \log(b+1) \rceil$ - ben exponenciális

bemenet hossza $> m$

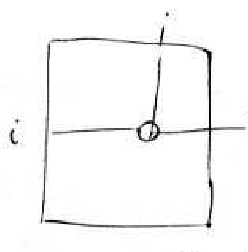
$\rightarrow$ nem polinomiális alg.

Gyors algoritmus, ha b kicsi, mondjuk $b = O(m^c)$

Gráfok megadása

$G = (V, E)$ $V = \{1, 2, \dots n\}$

szomszédossági mx



$T[i,j]$ — i-ből j-be menő élek száma

Egyszerű gráfra: 0, 1

Ha irányítatlan: főátlóra szimm.

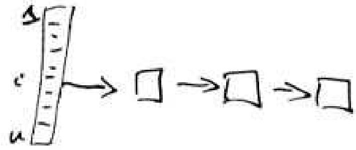
Súlyozott gráf: $\exists$ c súlyfv $c: E \to \mathbb{R}$

egyszerű, súlyozott gráf: súlyozott szomszédossági mx

$$T[i,j] = \begin{cases} c(ij), & \text{ha } ij \in E \\ 0, & \text{ha nem} \end{cases}$$

VAGY
$$C[ij] = \begin{cases} c(ij), & \text{ha } ij \in E \\ 0, & \text{ha } i = j \\ *, & \end{cases}$$

$\uparrow$ vagy ∞, programban jó vagy élista



i-ből induló élek (végpont, súly)

méret

mátrixnak: n^2 bit/szám
⇑
súlyozott

éllistának: $O(n+e)$ – irányított
az O azért kell, mert mutatót is tárolunk meg ilyenek

$O(n+2e)$ – irányítatlan
de mivel $n+2e < 2n+2e = 2(n+e)$
$O(n+2e) = O(n+e)$

(P) i-j-be megy-e él
mx: 1 lépés
éllista: $deg(i)$

i-t lerögzítem, kérek egy csúcsot, ahová megy él
mx: n
éllista: 1

$|E|$
mx: $\Theta(n^2)$ (összes elemet)
éllista: $\Theta(n+e)$

izolált pont
mx: n^2
éllista: n

$G = (V, E)$

Szélességi bejárás (Breadth First Search = BFS)

$v \in V$ kiindulópont

bejárva$[v]$ = igaz

$L = (v)$ (FIFO lista)

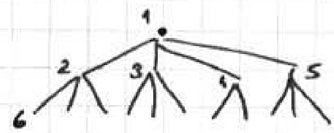
amíg L nem üres

x : L első eleme L-ből töröljük

$\forall (x, y) \in E$ ha bejárva$[y]$ = hamis

akkor bejárva$[y]$ = igaz

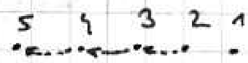
y-t L végére rakjuk

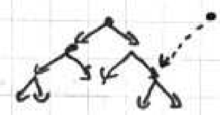


Ha nincs minden csúcs bejárva, akkor egy bejáratlanból folytatjuk

Pl G irányított út

de





az élek, amelyek bejáratlan pontba visznek = szélességi feszítő erdő

Lépésszám:

éllistás megadásnál: minden csúcs egyszer kerül a listába

minden élet egyszer nézünk meg (irányított gráfnál) $\Rightarrow O(n+e)$

kétszer (irányítatlan) $\rightarrow O(n+e)$

$n = |V|$
$e = |E|$

mátrixos megadás: x-ből kiinduló élek megtalálása $n \rightarrow O(n^2)$

a mátrix méretével arányos (lineáris)

Alkalmazások:

1: G - irányítatlan gráf: összefüggő-e? BFS egy tetszőleges v pontból, ha v-ből mindent elérünk, összefüggő, egyébként nem.

Lépésszám: $O(n+e)$ éllistával

2. irányítatlan gráf: összefüggő komponensek? BFS végig, szélességi feszítőerdő fái az öf. komponensek

Lépésszám: $O(n+e)$ éllist.

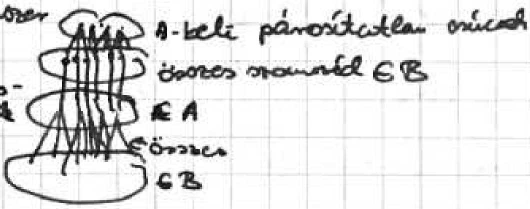
3. G irányítatlan: páros gráf-e? BFS

a szélességi bejárás mentén osztjuk 2 osztályba

tudjuk, hogy ha ellentmondás lesz, akkor nem páros gráf

4. G irányítatlan pos gráf: max. párosítás keresése

Algoritmus: magyar módszer



Ha egy B-beli pontból nincs folytatás → van javító út

1-gyel növeljük a párosítást

Lépésszám: 1 javító lépés $O(n+e)$

n pont ⇒ párosítás $\leq \frac{n}{2}$ élből áll

$\Rightarrow$ Össz lépésszám $\frac{n}{2} \cdot O(n+e) = O(n(n+e))$

Ha G öf, akkor $e \geq n-1$

$\Theta(n+e) = O(e)$

párosítás: lépésszám: $O(n \cdot e)$

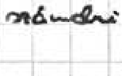
Mj: Lehet $O(\sqrt{n} \cdot e)$ lépésben is (az alternáló erdő ügyes kihasználásával)

(T) tetszőleges gráfra van $O(\sqrt{n} \cdot e)$ lépésszámú max. párosítást találó alg.

Max súlyú párosítás: $O(n \cdot e + n^2 \log n)$ lépésben található

5. G, v csúcsból min hány élű úttal érhető el egy $w \in V$

BFS v-ből



távolság: 0

1

2

⋮

az eljáráson belül lehet számolni

$O(n+e)$

$G = (V, E)$ $\quad c: E \to \mathbb{R}$

út súlya: élsúlyainak összege

Legrövidebb (legkisebb súlyú) út

legkisebb súlyú élsorozat

út: 11

1-szer körbe: 8

2-szer: 5

itt nincs legkisebb

ok: ∃ negatív súlyú kör

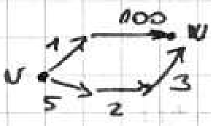
Feltétel: nincs negatív súlyú kör

Cél: legrövidebb út

Algel

2009.02.25.
Friedl

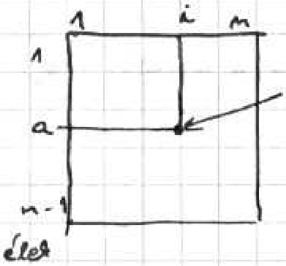
Min. élű út nem föltétlen jó



mohó nem jó

BELLMAN-FORD: rögzített csúcsból az összes többibe meghatározza a legrövidebb út hosszát

$V = \{1, 2, \ldots n\}$ 1-ből indulunk



$T[a,i]$ = 1-ből i-be vivő legfeljebb a db élet használó utak közül a legrövidebb hossza.

$T[1,i] = C[1,i]$ — (1, i) él súlya

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

$$T[a,i] = \min\{T[a-1,i],\ T[a-1,j] + C[j,i]\}$$

1-ből i-be legrövidebb út hossza $= T[n-1, i]$

Lépésszám: $n(n-1)\cdot \underbrace{(n-1)}_{\text{min keresés}} = O(n^3)$

Mj: 1. A táblázatból csak 2 sort kell tárolni

2. A legrövidebb út megtalálásához elég eltárolni mindig azt a j-t, amire a minimumot kapjuk → ez az utolsó előtti pontja az útnak.

HF: módosítás, amivel eldönthető, hogy van-e negatív kör

Bármely 2 pont közötti legrövidebb ~~...~~ út.

1. alg: Bellman-Ford minden csúcsból elindítva $\to O(n^4) = n\, O(n^3)$

2. Floyd-algoritmus

$F[i,j] = C[i,j]$ $i, j \in V$

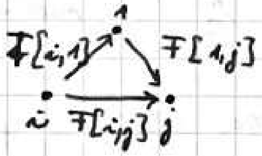
```
for k = 1 to n
  for i = 1 to n
    for j = 1 to n
        F[i,j] = min{F[i,j], F[i,k]+F[k,j]}
```

Áll: A végén $F[i,j]$ = i-ből j-be menő legrövidebb út hossza. $(\forall i, j \in V)$

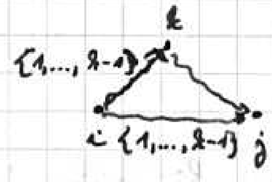
Lemma: $\forall$ k-ra $F[i,j]$ = i-ből j-be menő legrövidebb olyan út hossza, aminek belső pontjai $\in \{1, 2, \ldots, k\}$

Ebből az állítás $k=n$-re következik: minden út figyelembe van véve

(B) (lemma) $k=1$



k



Legjobb $\{1,\dots,k\}$ belső pontú élsorozat len

- van ilyen út is (nincs negatív kör)

Lépésszám: $O(n^3)$

Bellman-Ford, Floyd használható éllistával is — lépésszám: HF

$G=(V,E)$ irányított gráf, tranzitív lezártja $G'=(V,E')$, $(i,j)\in E'$

$\Updownarrow$

G-ben van i-ből j-be út

Alg. a tranzitív lezárt meghatározására

~Floyd $W[i,j] = W[i,j]$ vagy $(W[i,k]$ és $W[k,j])$ — WARSHALL-alg

Kezdetben $W[i,j] = \begin{cases} 1 & (i,j)\in E \\ 0 & (i,j)\notin E \end{cases}$ szomszédossági mtx

DIJKSTRA algoritmus bemutatása és bizonyítása

http://hu.wikipedia.org/wiki/Dijkstra-algoritmus

Adatszerkezetek = adatstruktúrák

adattípus, műveletek — megvalósítás

(Pl) Lista adatszerkezet : azonos típusú adatok

műveletek : első, következő, előző, utolsó

megvalósítás : tömb vagy láncolt lista (2 irányban)

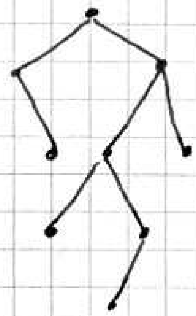
Bináris fa

gyökeres szintezett fa

minden csúcsnak a következő

szinten 2 szomszédja lehet

(bal fia, jobb fia)

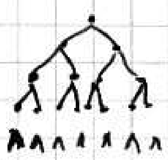


A gyökér kivételével $\forall$ csúcsnak van az előző

szinten egy szomszédja (apja)

levél : nincs fia

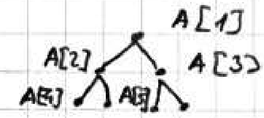
Teljes bináris fa



0, 1, ..., $l-1$. szinten $\forall$ csúcs megvan

l. szinten jobbról hiányozhatnak csúcsok

megvalósítás : • mutatókkal



• teljes bináris fát tömbbel : $A[1], A[2], \ldots A[n]$

$A[i]$-nek bal fia $A[2i]$

jobb fia $A[2i+1]$

apja $A\left[\lfloor \frac{i}{2} \rfloor\right]$

Kupac (heap)

műveletek: BESZÚR : új elem beillesztése

MINTÖR: minimális elem meghatározása, törlése

KUPACÉPÍTÉS: adott elemeket kupacba rendezi

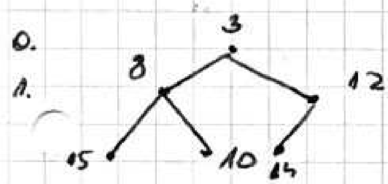
feltevés: csupa különböző elemet tárolunk

teljes bináris fában : elemek – csúcsok

kupac tulajdonság: $\forall$ x csúcsra : x-beli elem kisebb, mint a fiaiban tárolt elemek

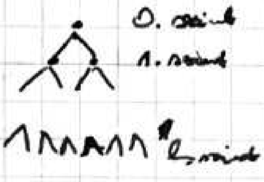
Algel

2009.03.04.
Friedl



Köv: a minimális elem a gyökérben van.

Szintek száma: i. szinten $(i<l)$ 2^i csúcs van



l szintű teljes bin fában:

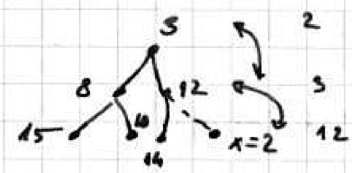
$$1+2+\ldots+2^{l-1}+1 \le n \le 1+2+\ldots+2^{l-1}+2^{l}$$

$$2^{l} \le n \le 2^{l+1}+2$$

$$l = \lfloor \log n \rfloor$$

Köv: Ha egy kupacban n elemet tárolunk akkor $\Theta(\log n)$ szintje van

BESZÚR: létrehozzuk az új elemet



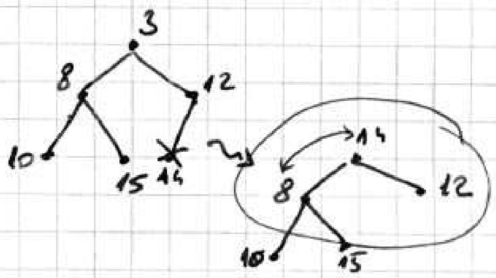
ide beírjuk az x elemet

ha az apja $> x$, akkor megcseréljük

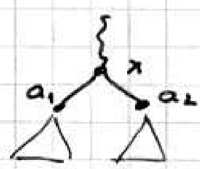
ismételjük, amíg jó lesz (apa $< x$ vagy x a gyökérben van)

Lépésszám: $O(\log n)$ n = csúcsok száma

MINTÖR: legkisebb elemet töröljük a gyökérből



az utolsó levelet töröljük, a benne levő elemet a gyökérbe rakjuk



a részfát kupacol

ha $x > \min\{a_1, a_2\} \Rightarrow$ az x-et felcseréljük a minimummal.

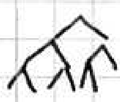
ezt folytatjuk, amíg lehet

Lépésszám: $O(\log n)$

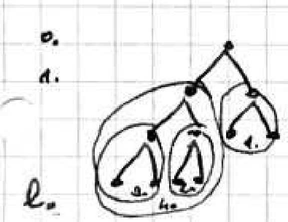
n elemből kupac készítése:

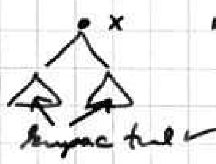
1. alg: sorozatos ~~...~~ BESZÚR-ral : $O(n \log n)$, van olyan eset, amikor $\geq c \cdot n \log n$ (HF találni)

2 alg: KUPACÉPÍTÉS



az elemek teljes bin fában alulról fel, jobbról balra a részfát kupaccá készítjük úgy, mint a MINTÖR-ben.





Lépésszám:

2 összehasonlítás $\rightarrow \leq 1$ csere

lehetséges cserék száma:

i. szint: 2^i csúcs, mindegyikre $\leq l-i$ csere.

összesen: $\displaystyle\sum_{i=0}^{l-1} 2^i (l-i) = \overbrace{1+1+\ldots+1}^{l} + \cancel{\overbrace{2+2+\ldots+2}^{l-1}}$

$2+2+\ldots+2 \quad \rightarrow l-1$ db

$4+4+\ldots+ \quad \rightarrow l-2$ db

$\vdots$

$2^{l-1} \quad \rightarrow 1$ db

Oszloponként összegezve

$(2^l-1)+(2^{l-1}-1)+\ldots+(4-1)+(2-1) =$

$= 2^{l+1} - l - \overset{?}{2} < 2^{l+1} = 2 \cdot 2^l \leq 2n$

Lépésszám: $O(n)$

További művelet: FOGYASZT(x, a): a kupacban az x helyen levő elemet a-ra cseréli ($a <$ eredeti elem)
eredmény KUPAC legyen HF

2009.03.11.

KUPACOK

//Folytatás//

FOGYASZT

1 konkrét elemet csökkent, utána a kupactulajdonságot visszaállítja

HOL LEHET BAJ?

– A szülőnél.

MO: cserélünk (míg a szülővel van baj ⇒ megoldható)

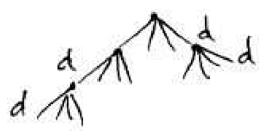
Lépésszám: ≤ szintszám

$O(\log n)$

KUPAC = egész bináris min. kupac

- egész ← ~~egy~~ teljes gráf
- bináris ← bin. fa ← ha nem, akkor legyen > 2, de konkrét szám
- min. kupac ← ez van fölül, ha nem, akkor nyilván maxtörlést kell majd megvalósítani

Pl.



∀ szülőnek d gyereke van, legalábbis az n-1. szintig, az n. szinten az utolsó levélnél a már nem biztosítható az elemek száma miatt

$\Theta(\log_d n)$

BESZÚR – $O(\log_d n)$ változatlan

MINTÖR – annyi változás, h. d db fiából kell kiv. a minimumot, akivel cserélni fog

lépésszámot viszont nem növeli nagyon:

$$O((d-1+1)\cdot\log_d n) = O(d\log_d n)$$

ÉPÍTÉS – n $O(\cancel{\log_d} n)$

FOGYASZT – n $O(\log_d n)$

Pl. Legyen $d=\sqrt{n}$

$\log_d n = 2$

BESZÚR: $O(1)$ = const

MINTÖR: $O(2\sqrt{n}) = O(\sqrt{n})$

ÉPÍTÉS: $O(n)$

FOGYASZT: const

ALKALMAZÁS DIJKSTRA - ALGORITMUSBAN

Az aktuális d értékek kupacban kéne tartani
$\Downarrow$
min, bináris

1. lépés $\Rightarrow$ kupacépítés
2. legkisebb elem ~~kivétel~~ iodlandbás $\Rightarrow$ MIN TÖR
3. frissítés $\Rightarrow$ FOGYASZT

$w \in V \setminus \{KÉSZ\}$? csak ezekkel érdemes
$(x, w) \in E$ foglalkozni

$\Uparrow$ éllistával elég ezeket megnézni

Ha a gráf éllistával adott

$$O(n) + (n-1)\,O(\log n) + \sum_x (x\text{-től kimenő élek száma})\,O(\log n) =$$

$\uparrow$ kezdeti értékek és kupacépítés

$$= O(n) + O(n \log n) + O(e \log n) =$$

elnyeli

$$= O(n \log n) + O(e \log n) = O((e+n)\log n)$$

Mi van, ha nem bináris a kupac?

$$O(n) + (n-1)\,O(d \log_d n) + e \cdot O(\log_d n) =$$

$$= O((nd + e) \log_d n)$$

Leggyen $d = \left\lceil \frac{e}{n} \right\rceil$ miért jó?
$O(e \log_d n)$

$\Uparrow$ átlagosan hány él indul ki 1 csúcsból

Értelmes feltevés, h. az élek száma leg. $n-1 \Rightarrow$ $\exists$ leggyen

Ha pl. $e \geq n\sqrt{n}$
$d \geq \sqrt{n}$
$\log_d n \leq 2 \Rightarrow O(e)$
Tehát lineáris algoritmust kaptunk :)

Van olyan algoritmus, amely $\forall$ gráfra élszámától függetlenül a lépésszáma $O(e + n \log n) \Rightarrow$ ehhez más adatszerkezet kell $\Rightarrow$ Fibonacci-kupac

KERESÉS

2009.03.11.

① $a_1, a_2, \dots a_n$
b elemet keresem az $a_1 \dots a_n$ elem között

Lépés: $(a_i == b)$
Végigpróbálja az összes elemet

ALGORITMUS

$i=1 \dots n$ $(a_i == b)$ $O(n)$ lépés

② $a_1, a_2 \dots a_n$ $\Rightarrow$ RENDEZETT LISTA
Lépés: $a_i \overset{?}{\substack{< \\ > \\ =}} b$

ALGORITMUS

Lineáris keresés

$i=1 \dots n$ VÉGE, ha $(a_i == b) \,||\, (b > a_i)$
$b < a_i$ ← Tan. szerint

$O(n)$ lépés

ALGORITMUS

Bináris keresés

~~$b > \lceil \frac{a_n}{2} \rceil$~~ $b \substack{< \\ > \\ =} a_{\lceil \frac{n}{2} \rceil}$

$=$ JUHÉ, megtaláltuk :)
$<$ $a_1, \dots a_{\lceil \frac{n}{2} \rceil - 1}$ között kell keresni ||:
$>$ $a_{\lceil \frac{n}{2} \rceil + 1}, \dots a_n$ között kell keresni ||:

VÉGE: ha beérjük a szomszédos elem közé, v. megtaláltuk

Lépésszám:
1 lépés után $\leq \frac{n}{2}$ elem marad
2. lépés után $\leq \frac{n}{4}$
⋮
l. lépés után $\leq \frac{n}{2^l}$

Addig megy, amíg csak 1 marad, azaz
$$\frac{n}{2^l} \leq 2$$
$$n \leq 2^{l+1}$$
$$\log n < l+1$$
$$\lfloor \log n \rfloor \leq l$$

+ a végén még 1 ellenőrzés

Állítás: A bin. keresés az adott feltételekkel optimális.

Bizonyítás: tekintsünk 1 tetszőleges algoritmust $b > a_i$ (i tetszőleges)
Úgy válaszoljunk, h. a másik algoritmus rosszabb legyen :)
Úgy legyen a válasz olyan, h. legalább $\frac{n}{2}$ elem versenyben maradjon 1. kérdés ⇒ a rosszre ont
2. kérdés ⇒ kettévág, megint legalább a fele marad
KELL $\lfloor \log n \rfloor$ lépés, h. csak 1 elem maradjon (ellenőrzés)

Egyetlen hibája a bin. keresésnek: nem túl könnyű megtalálni a középső elemet.

Pl. tömbnél → bináris

láncolt lista → lineáris

telefonkönyv → **INTERPOLÁCIÓS KERESÉS**

„Nagyjából" tudjuk, h. a b.

Néhány kellemes kis rendező algoritmus :)

Legyen $a_1, a_2, a_3, \dots a_n$ csupa különböző

BESZÚRÁSOS RENDEZÉS

Ha az első k elem már rendezve van, a k+1.-t beszúrja a helyére

k = 1   ✓ :)

k = 2   mögé v. elé

⋮

k = n   beszúrjuk a helyére

Mindig a következő elem helyét KERESEM, azt meg már tudunk → lin. → bin.

Lin. keresés esetén k összehasonlítás, azaz

$$1+2+3+\dots+n-2+n-1 = \frac{n\cdot(n-1)}{2}$$ összehasonlítás a teljes tömbre

Bin. keresés esetén

$$\sum_{k=1}^{n-1}\left(\lceil \log k\rceil+1\right) = n-1+\sum_{k=1}^{n-1}\log k =$$

$$= n-1+\log(1\cdot 2\cdot\ldots\cdot k) =$$

$$= n-1+\log(k!) =$$

$$= n\cdot\log n - nnzn = O(n\log n)$$ db összehasonlítás

További költség: elemcserék száma

k = 1   0

k = 2   1

k = n   max. n-1

$$1+2+\dots+n-1 = \frac{n(n-1)}{2}$$

Ha tömbként van megadva,

összehasonlítás: $O(n\log n)$:)

mozgatás: $O(n^2)$:(

Listával

összehasonlítás: $O(n^2)$:(

mozgatás: $O(1)$:)

} $O(n^2)$. . .

Feladat

Összefésülés $b_0 < b_1 < b_2 < \dots < b_r$ || => csupa kül. is $c_i \neq b_j$

$c_0 < c_1 < c_2 < \dots < c_s$

$d_0 < d_1 < d_2 < \dots < d_{s+r-1}$

$d_0 := \min\{b_0, c_0\}$

$i=0$
$j=0$

ha $(b_i < c_i)$ $\{ d_{i+j} = b_i,\ i{+}{+} \}$

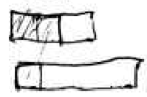
különben $d_{i+j} = c_j,\ j{+}{+}$

VÉGE, ha $i=r$, $j=s$

~~Lépésszám~~

Összehas. száma: $r+s-1$

Összefésüléses rendezés:



Lépésszám: 1 mint az összes összefésülés

Algel

2009.03.18
Friedl

Rendezések: $a_1, \dots a_n$ kül. elemek.

- Kupacos rendezés: ~~Kupacépítés~~ KUPACÉPÍTÉS, $\underbrace{\text{MINTÖR, MINTÖR, \dots}}_{n}$

$$O(n) \quad + n\,O(\log n) = O(n \log n)$$

- Buborék rendezés: $j = n-1, \dots, 1$
  $i = 1, \dots, j$
  ha $a_{i+1} < a_i$ akkor $a_{i+1} \leftrightarrow a_i$

Összehasonlítások száma: $(n-1) + (n-2) + \dots + 1 = \frac{n(n-1)}{2} = \Theta(n^2)$

cserék száma $\leq$ össz. száma (csökk. sorrend esetén lehet is)

__Áll__: a buborékrendezés rendez

B: $j = n-1$ végén a legnagyobb elem lesz a_n → ehhez többet nem nyúlunk
$j = n-2$ — " — ~~második~~ " — a_{n-1} — " —
⋮
$j = 1$ a végén az $n-1$. legnagyobb elem a_2 lesz
$\overset{\shortparallel}{a_1}$ csak a legkisebb lehet.

- Gyorsrendezés (quicksort): választunk egy $s = a_i$ véletlen elemet

partíció: $\underbrace{\qquad\quad}_{<s} \; s \; \underbrace{\qquad\quad}_{>s}$ $O(n)$ összehas.

rendezzük rendezzük

☺ a végén nem kell összefésülni

Pl: mindig a legkisebb elemet választjuk: $n-1$ összehas $s\underbrace{\qquad\qquad}_{>s}$

összehas. száma $n-1+n-2+\dots+1 = \frac{n(n-1)}{2} = \Theta(n^2)$

Pl: mindig a középső elemet választjuk: $n-1$ összehas → 2 fele akkora lista.

$1 \dots n$
$1 \dots \frac{n}{2}$ $\quad \frac{n}{2}+1 \dots n$

szintenként $< n$ összehas } $n \log n$
$\log n$ szint van

T A gyorsrendezésnél az összehasonlítások átlagos száma $= O(n \log n)$ ¬B
- ha biztosan gyors futás kell, akkor nem ez, de ha sokszor használ, akkor jó

(T) $a_1, \dots, a_n$ csupa különböző, algoritmus 1 lépésben 2 elemet tud összehasonlítani, 2-féle kimenet. $\left(a_i \, ? \, a_j \begin{matrix} < \\ > \end{matrix} \right)$ és minden lehetséges bemenetet rendez legfeljebb k összehasonlítással

$\Downarrow$

$$k \geq \log(n!)$$

$$n! = \left(\frac{n}{e}\right)^n \cdot \sqrt{2\pi n} \cdot \text{kicsi} \to \left.\begin{matrix} \log(n!) = \Omega(n \log n) \\ \log(n!) = O(n \log n) \end{matrix}\right\} \log(n!) = \Theta(n \log n)$$

(vagyis alsó becslés is $(n \log n)$-re)

(B) Úgy játszunk, hogy minél tovább tartson a játék.

Kezdetben $n!$ lehetséges sorrend van.

$\forall$ összehasonlítás 2 részre vágja a még lehetséges sorrendeket

Végén 1 lehetőség marad

$\forall$ lépésben megmaradhat a lehetőségeknek a fele $\Rightarrow$ kell $\geq \log(n!)$ lépés ∎

Kulcsmanipulációs rendezés

- láda rendezés (bin sort)

  A: alaphalmaz = lehetséges elemek (= kulcsok) halmaza

  Alg: $\forall x \in A$-hoz létrehoz egy $B[x]$ ládát (lista)

  $i = 1, \dots, n$ $\quad a_i$-t a $B[a_i]$ lista végére rakja

  a listák tartalmát sorban kiírja

  $$\text{Lépésszám}: \underbrace{O(|A|)}_{\text{lista létrehozása}} + \underbrace{O(n)}_{\text{bepakoljuk}} + O(\underbrace{n}_{\forall \text{ elem}} + \underbrace{|A|}_{\forall \text{ láda}}) = O(n + |A|)$$

  kiolvasás

  Pl: $|A| \leq c \cdot n \rightsquigarrow O(n)$

  Kell, hogy az alaphalmaz értelmes méretű legyen.

  (M) Ládarendezésnél az egyforma elemek sorrendje nem változik.

- $A \subseteq A_1 \times A_2 \times \dots \times A_k$ (azaz k koordinátájuk van)

  pl: dátum: év, hónap, nap

  szavak: $A_i :=$ abc

Ezekre ládarendezés $O(n + |A|) = O(n + |A_1| \cdot |A_2| \cdot \ldots \cdot |A_k|)$

Algel

2002. 03. 18.
Friedl

A_n rendezés = lexikografikus rendezés: $(s_1,\dots s_k) < (t_1,\dots,t_k)$, ha van olyan j: $s_i = t_i$ $i<j$, és $s_j < t_j$ (az első olyan számút, amiben különböznek)

Lassú, az utolsót sokszor használjuk

- Radix rendezés: ládarendezés k koord. szerint az összes elemet

$k-1$ ——— // ———

⋮

1\. ——— // ———

először az utolsó betű szerint

Pl:				
a_1: CCAB	DCBA	CCAB	ABAC	ABAC
a_2: DCBA	CCAB	ABAC	CCAB	ACAC
a_3: ABAC	ABAC	ACAC	ACAC	CCAB
a_4: ACAC	ACAC	DCBA	DCBA	DCBA

(T) A radix rendezés rendez.

(B) Elég megmutatni, hogy ha $(s_1,\dots,s_k) < (t_1,\dots,t_k)$, akkor az alg. végén az s megelőzi t-t.

$\exists j$: $s_j < t_j$, $s_i = t_i$, ha $i<j$

radix: k. koord, ..., $(j+1)$. koord. nem tudjuk a sorrendjüket

j. koord $\to$ s előbb lesz, mint t

$(j-1)$., ..., 1. koord $\to$ nem vált. a sorrend (mert a ládarendezés az egyforma elemek sorrendjét nem váltja)

Lépésszám: $O(n+|A_k|)+O(n+|A_{k-1}|)+\dots+O(n+|A_1|) = O(kn+|A_1|+|A_2|+\dots+|A_k|)$

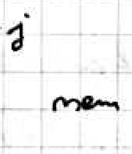
a ládarendezésben szorzat, itt összeg → sokkal szebb

Pl: $|A_i| \le cn$ $\quad i=1,\dots,k$ $\rightsquigarrow O(k\cdot n)$

Pl: $|A_i| \le c$ $\quad i=1,\dots,k$ $\quad k=\log n \rightsquigarrow O(n\log n)$

számok rendezése, számjegyek száma

Adatszerkezet

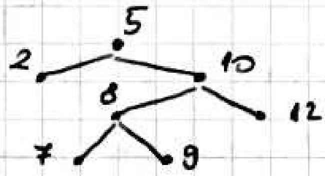


bináris fa (nem feltétlen teljes)

- bin. fa bejárás: preorder - csúcs, bal részfa, jobb részfa

  inorder - bal részfa, csúcs, jobb részfa

  postorder - bal részfa, jobb részfa, csúcs

Pl 

preorder : 5, 2, 10, 8, 7, 9, 12

inorder : 2, 5, 7, 8, 9, 10, 12

posztorder : 2, 7, 9, 8, 12, 10, 5

• bináris keresőfa

legkisebb elem megtalál

KERES, BESZÚR, TÖRÖL, MIN, MAX, TÓLIG

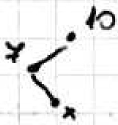
bináris fa

elemeket csúcsokban tároljuk (elem → csúcs kölcs. egyért. megf.)

(feltevés: csupa külö. elem van)

keresőfa tulajdonság:

~~??~~ bal részfa elemei < csúcs eleme < jobb részfa elemei teljesül ∀ csúcsra

Pl  $7 < x < 10$

KERES(x) : gyökér ? x
- = megtaláltuk
- < keresést a gyökér jobb részfájában folytatjuk
- > —"— bal —"—

ha nem tudunk lépni ⇒ nincs a fában

lépésszám = $O(l)$ l: szintszám, de lehet $l = n$

BESZÚR(x) : KERES(x) - ha talál, akkor nem rakjuk be

- ha nem talál, ahova lépni akart (de nem tudott), ott kell létrehozni egy új csúcsot x értékkel

lépésszám = $O(l)$

Milyen sorrendtől lesz cikcakkos a fa? 

2009.03.25.

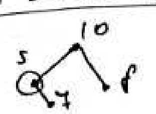
7

Bináris keresőfa
(Folytatás)

MIN() - visszaadja a legk. elemet

a gyökértől ℓ lépésben balra kell menni, ahol elakadunk, ott van a legkisebb.

NEM FELTÉTLENÜL LEVÉL!



Lépésszám: $O(\ell)$

↑ szintek száma

MAX() - leg> tárolt elem

a gyökértől ℓ jobbra

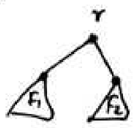
— " — "

Lépésszám: $O(\ell)$

Állítás:

Bináris keresőfa inorder bejárása a tárolt elemeket növekvő sorrendben adja.

Bizonyítás:



$\lfloor F_1 \rfloor \; r \; \lfloor F_2 \rfloor$

$\Downarrow$

Itt a megfelelő helyén van az r

$\Rightarrow$ ∀ csúcs a megfelelő helyén lesz

∀ki pont annyiadik helyen van, ahányadik a sorban

TÖRÖL (x)

- x-et törli a fából

→ keresés → KERES(x)

→ törlés

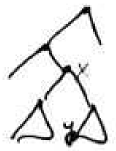
→ rendezés

KERES(x) → ha nem talál: KÉSZ ; ha megvan:

ha x_q levél → töröljük a csúcsot

ha x-nek 1 leszármazottja van → megszüntetjük, és a fiához tartozó részfát betoljuk

ha x-nek 2 fia van



→ vegyük x jobb oldali részfájának a minimális elemét mondjuk, azt tegyük oda be. Legyen ez a csúcs y. ~~y-nak akkor~~ y-nak biztosan csak 1 fia van → törlése egyszerű.

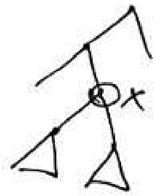
Lépésszám: $O(l)$

TÓL-IG (x,y)

Azon elemeket adja vissza, amelyre igaz:

$x \le a \le y$

Keressük meg az x helyét:



n-nél kevesebb lépést nem lehet, innen meg már megoldható

Minimalizálni kell!

KERES (x) elemből az inorder bejárást követve megyünk y-ig.

Gyors, ha a bináris keresőfa ∀ eleméből ∃ mutató az inorder bejárás szerinti köv. elemre.

(inorder fonál)

ha ilyen van, a TÓLIG lépésszáma:

$O(h+t)$

↓

t, a megfelelő elemek száma

Tétel:

Ha 1 kezdetben üres bin. fába n elemet beszúrunk, akkor az átlagos összlépésszám $O(n \log n)$

(mármint a különböző sorrendekre nézve)

A fa átlagos magassága $O(\log n)$

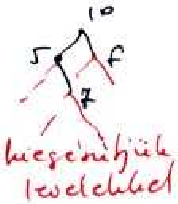
TB

Cél: KONKRÉT meghatározása a fáknak, nem átlagos ⇒ kiegyensúlyozás

1. AVL-fa (könyvben ha érdekel)
2. piros-fekete fák

PIROS-FEKETE FÁK

Bináris fa
nem levél csúcsoknak 2 fia van
|||
„belső csúcsok"
a belső csúcsokban tároljuk az elemeket
(csupa kül. elem)



• ∀ csúcs PIROS vagy FEKETE
gyökér, levelek
FEK

• PIROS csúcs ∀2 fia fekete

• ∀ csúcs: ∀ v-ből levélben lefelé vivő úton a FEKETE csúcsok száma ugyanannyi
$fm(v)$ // v nem számít bele!

$fm(10) = 2$
$fm(5) = 1$

$m(v)$ – a leghosszabb út, amellyel v-ből levélig mennénk
(vmiféle magasság)

$m(10) = 3$
$m(5) = 2$
$fm(18) = m(18) = 1$

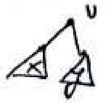
Állítás: $m(v) \geq fm(v) \geq \frac{m(v)}{2}$ ∀ v csúcsra

Bizonyítás: $m(v) \geq fm(v)$ fm-be csak a feketéket számoljuk
1 úton 2 piros nem jöhet egymás után ⇒
az úton a csúcsoknak $\underbrace{\text{leg. a}}_{\text{legalább}}$ fele fekete

Állítás: A v gyökerű fában tárolt elemek száma $\geq 2^{fm(v)} - 1$

Bizonyítás: $m(v)$ szerinti indukció
$m(v) = 0$ (a v levél $\Rightarrow fm(v) = 0$, $2^{fm(v)} - 1 = 2^0 - 1 = 0$
levélben nem tárolunk

$m(v) > 0 \Rightarrow$ van 2 fia



$\left.\begin{array}{l} m(x) \\ m(y) \end{array}\right\} \leq m(v) - 1 \Rightarrow$ alkalmazható az indukciós feltevés

$\Downarrow$

a tárolt elemek száma a v gyökerű fában $\geq \underset{\substack{\uparrow \\ \text{maga a } v}}{1} + \underbrace{2^{fm(x)} - 1 + 2^{fm(y)} - 1}_{\text{fiai}} =$

$= 2^{fm(x)} + 2^{fm(y)} - 1 \geq$

$fm(x) \leq fm(v)$

$fm(x) \geq fm(v) - 1$

$fm(y) \geq fm(v) - 1$

$\geq 2 \cdot 2^{fm(v)-1} - 1 = 2^{fm(v)} - 1$ ∎

Ha n elemet tárolunk, mit tudunk mondani a magasságáról?

Állítás: Ha n elemet tárolunk, akkor a gyökér mag-a $\leq 2 \log(n+1)$.

Bizonyítás: v − gyökér

$n \geq 2^{fm(v)} - 1 \geq 2^{\frac{m(v)}{2}} - 1$

$\log(n+1) \geq 2^{\frac{m(v)}{2}}$

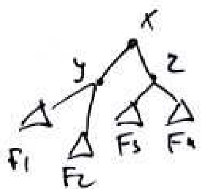
$m(v) \leq 2 \log(n+1)$ ∎

Következmény: KERES, MIN, MAX: $O(\log n)$

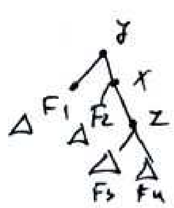
IH majd az eljárás a levelekben akadnak el
(bin. ker. fa − nem tudn lépni
piros-fekete fa − tudn, de levél van)

BESZÚR()
TÖRÖL() → kicsit más, mert el tudom rontani (változtatja a fa alakját)

A helyreállítás eszköze: FORGATÁS



FORGATÁS JOBBRA:



!! FONTOS: Keresőfából keresőfát csinál

BESZÚR()

Rakjuk be az új elemet az eredeti keresőfába

Az új belső csúcs legyen piros – z csúcs

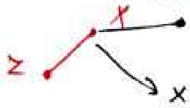
0. eset ⇒ z gyökér → ő az első elem :)

könnyű beszúrni

z ≠ gyökér ⇒ z-nek ∃ apja : x

Ha x-fekete, akkor KÉSZ

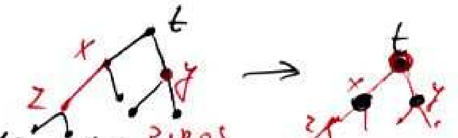
Ha x-PIROS,



→ piros, biztosan van apja

t ↑ ő pedig biztosan fekete

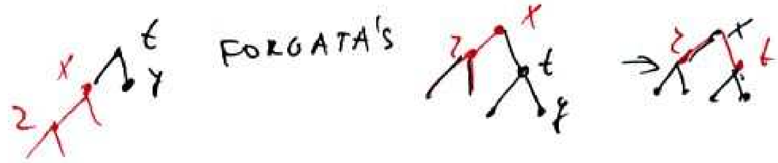
→ x-nek van testvére is : y (hisz z fia van)

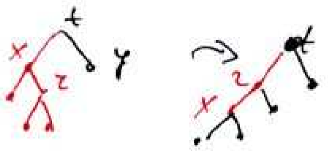


Ha y - PIROS

Baj akkor van, ha t apja piros

Ha y - FEKETE





Tétel:

BESZÚR() lépésszáma $O(\log n)$ és ≤ 2 forgatást használ

TÖRÖL() lépésszáma $O(\log n)$ és ≤ 3 forgatást használ

Emne **Algel** Dato 2009. ÁPRILIS ELSEJE!!!

piros-fekete fa:

- keresőfa
- levelek ← technikai:
  gyakorlati megoldás: 1 db csúcs helyettesíti az összes levelet
  (így már nem fa, de ez nem baj)

KERES, MIN, MAX, TÓLIG, BESZÚR, TÖRÖL – ugyanaz, mint a keresőfa, csak más adatszerkezet

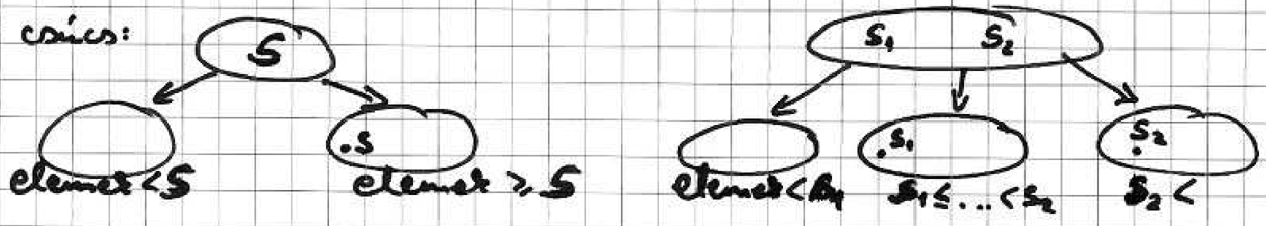
<u>2-3 fa</u>: gyökeres, színezett

levelek 1 szinten vannak

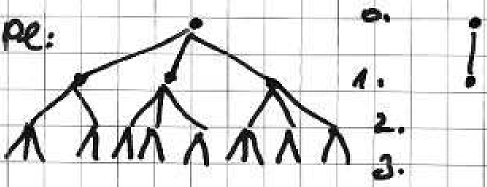
nem levelek (= belső csúcsok): 2 vagy 3 fia van (ha elemek száma > 1)

elemeket csak a levelekben tároljuk

belső csúcs:



Pl:



<u>Áll</u>: n elemet tároló 2-3 fa magassága $\Theta(\log n)$

<u>Biz</u>: k. szinten levő csúcsok száma $\geq 2^k$
$\leq 3^k$

n levél van: $2^k \leq n \leq 3^k$

$\log_3 n \leq k \leq \log n$

KERES: az útjelzők szerint lép a következő szintre $\Theta(\log n)$

elemek a levelekben növekvő sorrendben vannak

~~MIN~~: balra megyünk $\Theta(\log n)$

MAX: jobbra megyünk $\Theta(\log n)$

TÓLIG(a, b): ~~[illegible]~~ KERES(a), ha a levelek sorba vannak láncolva,
akkor ezen láncon végigmegyünk $\Theta(\log n + t)$ ← találatok száma

Emne **Algel** Dato 2009. ÁPRILIS ELSEJE Friedl

BESZÚR: KERES új levelet kell berakni

- 2 gyerekes csúcshoz → beraktjuk, 3 gyereke lesz
- 3 gyerekes: $\xrightarrow{\text{csúcsvágás}}$ 1 szinttel feljebb beszúrás

ha felmegy a gyökérig, ⇒ új gyökér, magasság nő

$\Theta(\log n)$

TÖRÖL: KERES

- ha a megtalált levél 3 gyerekes családban

~~töröljük~~ a levelet

útjelzőket frissítjük a gyökér felé vezető úton

- 2 gyerekes család → igazságosan elosztjuk a gyerekeket
  - ha van 3 gyereken szomszédos testvér
  - ha a szomszédos testvérnek 2 gyereke van $\xrightarrow{\text{csúcs összevonás}}$ 1 szinttel feljebb törlés

útjelzők állítása

$\Theta(\log n)$

B-fa: B_m fa

gyökeres, szintezett

levelek egy szinten, csak a levelekben tárolunk

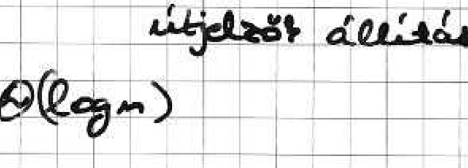
$\forall$ belső csúcsnak $\leq m$ fia van

$\forall$ belső csúcsnak, ami nem gyökér $\geq \left\lceil \frac{m-1}{2} \right\rceil$ fia van

gyökérnek ≥ 2 fia (ha 1-nél több elemet tárolunk)

$m=3 \Rightarrow$ 2-3 fa

belső csúcs:



KERES, MIN, MAX : mint a 2-3 fánál

$\Theta(l)$ l - szintszám

BESZÚR: m-nél kevesebb gyerek mellé ⇒ létrehozzuk az új gyereket

m gyerek mellé ⇒ csúcsvágás $\boxed{-2-}$ $\left\lfloor \frac{m+1}{2} \right\rfloor$ és $\left\lceil \frac{m+1}{2} \right\rceil$ fia

Emne: **Algel** Dato: 2005. április eleje, Friedl.

TÖRÖL: működik

Áll: n elemet tároló B_m fa magassága $\Theta\left(\frac{\log n}{\log m}\right)$

Biz: k szinten levő csúcsok száma $\le m^k$

$$\ge 2\cdot\left(\left\lfloor\frac{m+1}{2}\right\rfloor\right)^{k-1}$$

$$m^k \ge n \ge 2\left(\left\lfloor\frac{m+1}{2}\right\rfloor\right)^{k-1}$$

$$k \ge \frac{\log n}{\log m} \qquad \frac{n}{2} \ge \left\lfloor\frac{m+1}{2}\right\rfloor^{k-1}$$

$$\frac{\log\frac{n}{2}}{\log\left\lfloor\frac{m+1}{2}\right\rfloor}+1 \ge k$$

$$\log\frac{n}{2} \le \log n$$

$$\log\left\lfloor\frac{m+1}{2}\right\rfloor \ge \log\frac{m}{2} \underset{m\ge 4}{\ge} \frac{\log m}{2}$$

$$\Rightarrow \quad k \le \frac{\log n}{\frac{\log m}{2}}+1 = 2\frac{\log n}{\log m}+1 \le$$

$$\underset{n\ge m}{\le} \boxed{3\frac{\log n}{\log m}}$$

:-)

(H) • külső tár (lassú a beolvasás)

levél: nem 1 elem, hanem annyit tárol, amennyi egy fizikai (beolvasás egysége) lapra ráfér

m megválasztása: 1 belső csúcs = 1 fizikai lap

szintszám: lapbeolvasások száma

nem föltétlen csak a levelekben tároljuk elemet, hanem a belső csúcsokban is

ez a csúcsstruktúra ábrázolása 2D-ben

<u>Hash</u> (a hasis rövidített formája ; = aprít, gyúr, ...)

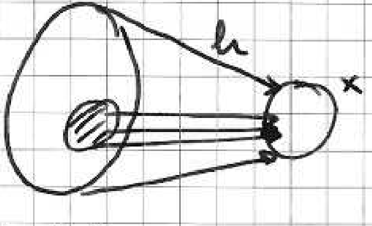
KERES, BESZÚR, TÖRÖL ~ Ládarendezés

Lehetséges elemek száma nagy

elemhalmaz: K = kulcsok halmaza

ténylegesen használt elemek kulcsa : lényegesen kisebb (nagy luxus $\forall$ kulcsnak 1 láda, mert a nagyrésze üres)

Emne **Algel** Dato 2009. ÁPRILIS ELSEJE Friedl



$|X| \ll |K|$

h - hash fv.

Pl: K - emberek, X: év napjai, h: születésnap

23 ember között ½-nél nagyobb valséggel van 2, akiknek ugyanarra a napra esik a születésnapjuk

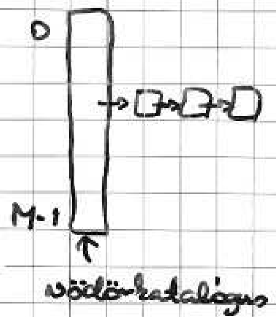
várható egybeeső párok száma 50 ember között 3
200 54 → elég magas, nem bíl jó fv.

K - marslakók $|X| = 669$

31 marslakó között van ≥ ½ valséggel azonos születésnap

tehát az ütközéseket nem tudjuk elkerülni => meg kell oldani

Vödrös hash: $h: K \to X = \{0, 1, 2, \ldots, M-1\}$



s elem helye: h(s) indexű lista

KERES(s): h(s) listában lineárisan keres

BESZÚR(s): h(s) listába berak

TÖRÖL(s): h(s) listából töröl

$O(n)$ lépés

ha h közel egyenletesen szórja szét az elemeket, a listák várható hossza $\frac{n}{M}$

(M) Külső táron használják (vödörkatalógus lehetőség szerint a memóriában)

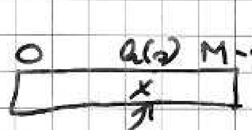
a listák elemei lapok, 1 lapon több rekord.

Ha L lap kell az összes elem tárolásához, akkor átlagos lapelérés-szám $L/M + 1$

Pl: $M \sim 1,2L$ => átlagos lapelérés < 2

Nyitott címzésű hash (memóriában)

tömbben tároljuk az elemeket T:



s helye: $T[h(s)]$

ugrások: $h_0(s) = 0, h_1(s), h_2(s), \ldots, h_{M-1}(s)$ a $0, 1, \ldots, M-1$ egy permutációja.

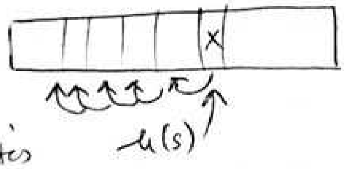
próbasorozat:
$h(s) + h_0(s) = h(s)$
$h(s) + h_1(s)$
$h(s) + h_2(s)$ mod M
⋮
$h(s) + h_{M-1}(s)$ ∀ helyet kipróbál

Nyitott címzésű hash (folyt.)

KERES: addig megy a próbasorozaton végig, amíg nem talál egy üreset (soha nem volt senki)
↕ vs.
BESZÚR: a próbasorozat első szabad helyére beírja (most épp nincs senki)

TÖRÖL: KERES, ha talált, törli + beállít „törölt" bitet (azért, hogy a keresés ne álljon meg rajta, de be lehessen szúrni)

Lineáris próbálás: $h_i(s) = -i \pmod{M}$



balra lépegetés

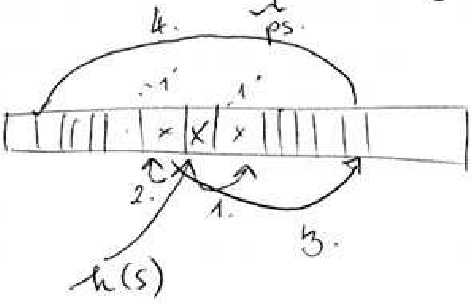
ha két próbasorozat találkozik, onnan együtt mennek tovább
→ elsődleges csomósodás

Kvadratikus próbálás (alternáló próba)

ugrás a kezdőponthoz képest:

$$\underbrace{h_{2j-1}}_{\text{ptlan}}(s) = j^2 \pmod{M} \qquad \left(1 \le j \le \frac{M}{2} \text{ között}\right)$$

$$\underbrace{h_{2j}}_{\text{ps.}}(s) = -j^2 \pmod{M}$$



→ kusza lesz ...

egy foglalt blokkon (=csomó) belül különböznek a próbasorozatok

De ha $h(s) = h(t) \Rightarrow$ próbasorozatuk is azonos.

~~Addig kell vizsgálni~~

Pl. ha $M=8$: $h_0 = 0$, $h_1 = 1$, $h_2 = 7$, $h_3 = 2^2 = 4$, $h_4 = -4 = 4$ (8)

$h_5 = 9 \equiv 1$ (tehát $h_3 = h_4$; $h_1 = h_5$:( )

$h_6 = 7$ $\quad h_7 = 16 \equiv 0 \ (8)$ $\quad \to$ EZ NEM PERMUTÁCIÓ!

$(= h_2)$ $\quad = h_0$ $\quad \to$ NEM JÓ.

pl. $M = 7$ $\quad h_0 = 0$; $h_1 = 1$; $h_2 = -1 \equiv 6 \ (7)$; $h_3 = 2 \stackrel{?}{=} 4$

$h_4 = -4 \equiv 3 \ (7)$; $h_5 = 3^2 = 9 \equiv 2 \ (7)$; $h_6 = -2 \equiv 5 \ (7)$

EZ JÓ ✓ $\to$ EZ PERMUTÁCIÓ ✓

Állítás: Ha $M = 4k+3$ és prím, akkor jó ($0, h_1 \dots h_{n-1}$ permutációja a $0, \dots n-1$-nek)

Bizonyítás:

1) $h_{2j-1} = h_{2i-1} \iff i^2 \equiv j^2 \ (M)$

$$\Updownarrow$$

$$M \mid j^2 - i^2 = (j-i)(j+i)$$

$$\underbrace{M \mid j-i}_{} \quad \vee \quad \underbrace{M \mid j+i}_{}$$

$(j, i \le \frac{M}{2})$ $\quad$ csak, ha $j = i$ $\qquad j+i < M \to$ nem lehet

$\to$ A páratlan indexűek (négyzetszámok) különbözőek.

2) $\to$ A ~~...~~ -1-szereseik is. (a páros indexűek is).

3) Lehet-e: ps és ptlan indexű $=$?

$h_{2j-1} = h_{2i} \iff j^2 \equiv -i^2 \ (M)$ $\to$ van-e ilyen j és i?

feltehetjük, hogy $i, j \neq 0$

j, i relatív prímek M-hez

létezik olyan t: $it \equiv 1 \ (M)$

$$i^2 t^2 \equiv 1 \ (M)$$

$$j^2 t^2 \equiv -i^2 t^2 \equiv -1 \ (M)$$

$$(j^2 t^2)^{2k+1} \equiv (-1)^{2k+1} \equiv -1 \ (\mathrm{mod}\ M)$$

$$(jt)^{2(2k+1)} = (jt)^{4k+2} = (jt)^{M-1} \equiv -1 \ (M)$$

→ ez ellentmond: Euler-Fermat-nak...

→ $(jt)^{M-1} \equiv 1\ (M)$ (M prím!)

→ ellentmondás →

Jó A MÓDSZER ✓

(megj. az egész azon múlik, hogy nincs olyan x:

$x^2 \equiv -1 \pmod M$ → −1 nem négyzetszám.)

Kvadr. hash (folyt):

→ másodlagos csomósodás

($h(s) = h(t)$ → próbasorozatuk)

kettős hash $h_i(s)$ függjön s-től

tipikusan: $h'(s)$ második hashfv.

$h_i(s) = -i \cdot h'(s) \pmod M$ feltétel: $(h'(s), M) = 1$

HASH-FÜGGVÉNYEK:

elvárjuk tőle:

- gyorsan számolható
- kevés ütközés (véletlenszerűen szórja az elemeket)

megoldások:

1) osztómódszer: $h(s) = s \bmod M$

⇒ nem jó, ha M 2-hatvány!

mert: ilyenkor csak az utolsó bitek levágása történik

nem az igazi, ha $M \approx$ 2-hatvány

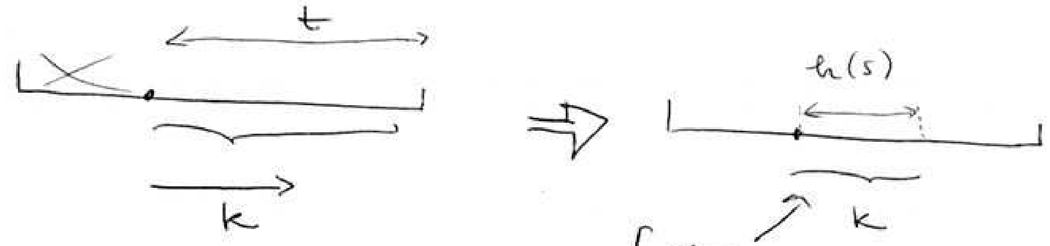
2) szorzómódszer: $h(s) = \lfloor \{\beta \cdot s\} \cdot M \rfloor$

törtrész   egészrész (lefelé kerekítve)

→ $h(s) \in \{0, \dots M-1\}$ ✓ $M = 2^k$

β eredetileg irracionális szám: $\beta = \frac{A}{2^t}$ A ptlan egészt

$\frac{A \cdot s}{2^t} \rightarrow$



[ez a néhány bit lesz a végeredmény]

HASH LÉPÉSSZÁMA

lehet lineáris: $O(n+M)$

átlagos lehet: $O(\log n)$

pl. átlagos lépésszámok:

1) ha $\frac{n}{M} = \frac{2}{3}$ (telítettség)

	sikeres	sikertelen
→ lineáris: ~~2, 3~~	2	3
→ kvadr:	1,8	~3
→ kettős h.:	1,5	~3

2) ha $\frac{n}{M} = \frac{4}{5}$

→ lineáris:	3	13
→ kvadr:	2,2	4,8
→ kettős h:	2,9	5

KERESÉS LÉPÉSSZÁMA

1) lineáris keresés rendezett listában

sikeres	sikertelen
$\frac{n+1}{2} = \frac{1+2+\dots+n}{n}$	$\dots = \frac{\frac{n(n+1)}{2}+n}{n+1} \simeq \frac{n}{2}+1$

$\propto \frac{n}{2}$ mind2

2) bináris keresés
   - sikertelen : log n
   - sikeres :

3) keresőfa $O(\log n)$

→ ehhez képest a hash-elés konstans lehet (jó esetben)
hirtelen lineáris is lehet
akkor jó a hash, ha elhúzódhat a keresés, de
átlagban gyors kell.

NEM EGYENLETES ELOSZLÁS, LINEÁRIS KERESÉS

ZIPF-eloszlás: - a szavak eloszlása egy ~~természetesen~~ szövegben
- az i. leggyakoribb valószínűsége : $\frac{c}{i}$
→ sikeres keresés: $\frac{n}{\log n}$ (átl.)

80-20 eloszlás : - esetek 80%-át megoldjuk az idő 20%-a alatt
a többi 20%-ot NEM 80% időben ~~kell~~ oldjuk meg.
→ hanem a maradék 20% 80%-át 20%-alatt megoldjuk
→ stb. rekurzívan
- az i. legnagyobb valószínűség
$$\frac{c}{i^{(1-\vartheta)}} \quad (\vartheta, \cancel{c} \text{ konstans})$$
→ sikeres keresés : $0{,}12 \cdot n$ (átl.)

GRÁFOK

MÉLYSÉGI BEJÁRÁS (Depth First Search - DFS)
- bátor felfedező
- addig megyünk előre, amíg van felfedezetlen út,
ha elértünk a végére, visszamegyünk & próbáljuk újra
másik irányban ...

mb(v)

(1) →

bejárva [v] := igaz;

∀ w : (v,w) ∈ E :

ha bejárva [w] = hamis, akkor mb(w);

(2) →

lépésszám: – éllistás megadásnál:

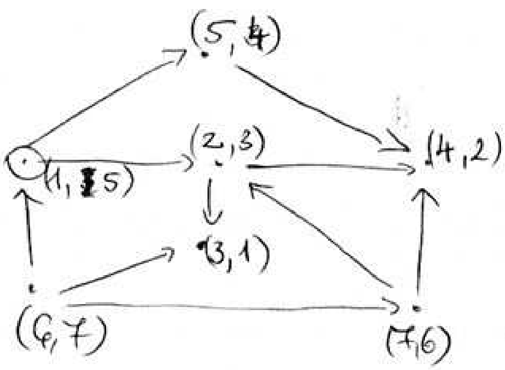
$O(n+e)$

minden csúcsot & élet 1x járunk be

mélységi szám: – csúcs jellemzője
– "hányadiknak értünk bele" msz[v]
– (1)-nél kap értéket ↑

befejezési szám: – csúcs jellemzője
– hányadiknak fejeztük be bsz[v]
– (2)-nél kap értéket ↑

Pl.



járjuk be!

⊙-ről indulunk

(3, 2)

↑ ↑

msz bsz

ha még nem voltunk mindenhol, tovább folytatjuk!

bal alsó sarok

Emne **Algel** Dato 2009.04.15. Friedl

Gráf

mélységi szám (msz) : elérési sorrend

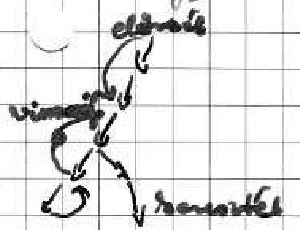
befejezési szám (bsz) : visszalépési sorrend

mélységi feszítő erdő: élek, amelyek bejáratlan pontba visznek.
↑ faélek

élek típusai: faélek, visszaél, keresztél, előreél.

Élek osztályozása bejáráskor:

$(x,y) \in E$ faél: még nincs msz[y]-nak értéke



előreél: msz[x] < msz[y]

visszaél: msz[x] > msz[y], még nincs bsz[y] érték

keresztél: msz[x] > msz[y], van bsz[y]

G irányított gráf, nincs irányított kör = dag (directed acyclic graph)

(T) G dag ⇔ mélységi bejárásban nincs visszaél.

(B) „⇒" megfelelő faélek + visszaél = irányított kör ⇒ ∄ visszaél, ha nincs kör.

„⇐" indirekt: tfh van kör G-ben. Legyen x a körnek az a pontja, aminek msz minimális

- ha a kör többi pontja az x gyökerű részfában van és y az x-et a körben megelőző pont, akkor (y, x) visszaél
- van a körnek a részfán kívül is pontja. Ekkor van a körnek a részfából kivezető éle. Ez az él nem fa- vagy előreél, nem visszaél (mert ilyen nincs) ⇒ ⇒ keresztél ⇒ ennek végpontját x előtt bejártuk, ami nem lehet, mert a körből x volt az első.

Köv: a G egy mélységi bejárásában nincs visszaél, akkor egyetlen mélységi bejárásban sincs

(B) nincs visszaél ⇒ dag ⇒ nincs visszaél

(M) a keresztélekre ez nem igaz. Pl: G = irányított út

v_1 → → → → v_n visszafelé bejárva v_n v_{n-1} v_{n-2} … v_1 ∀ él keresztél.

∀ él faél

Köv: adott G dag-e könnyen eldönthető, éllistás megadásnál $O(n+e)$ lépésben.

(B) Algo: ~~[illegible]~~ DFS dag $\Leftrightarrow \nexists$ visszaél.

(D) topologikus rendezés: a G csúcsainak egy olyan $v_1, v_2, \ldots v_n$ sorrendje, amire, ha $(v_i, v_j) \in E$, akkor $i < j$

$v_1 \quad v_2 \quad \ldots \quad v_n$ persze ha pl kör, akkor nem ilyen, visszafelé is mutató él

(T) G-nek van topologikus rendezése $\Leftrightarrow$ dag.

(B) „$\Rightarrow$" $(v_i, v_j) \in E \Rightarrow i < j$ nem lehet kör

„$\Leftarrow$" dag $\Rightarrow$ van benne nyelő (amiből nem megy ki él), mert tetsz. pontból elindulva egy kimenő él mentén nem juthatunk vissza (nincs kör) $\Rightarrow$ vége kell legyen egyszer $\Rightarrow$ nyelő

első pontja a körnek

v_n = nyelő, v_n-et hagyjuk el a gráfból $\Rightarrow$ dag marad $\Rightarrow \exists$ nyelő $= v_{n-1}, \ldots$

(lehet $O(n+e)$ lépésben csinálni) z, x-nek kisebb az száma

2. mo: DFS (mélységi bejárás)

top. rendezés: befejezési szám szerinti csökkenő sorrend.

ehhez csak azt kell belátni, hogy kisebb bsz élből nem megy nagyobba él, vagyis visszaél, de dag ✓

$O(n+e)$

(T) dag-ban s-ből a többi pontba vivő legrövidebb utak megtalálhatók $O(n+e)$ lépésben éllistás megadásnál.

Algo: DFS $\rightarrow$ topologikus rendezés (ez eddig $O(n+e)$, tehát belefér)

$v_1 \quad v_2 \quad \ldots \quad v_n$ $(v_i, v_j) \in E \Rightarrow i < j$ az élek mind jobbra mennek.

feltehető, hogy $s = v_1$, $\forall$ csúcsra kezdetben $d(s, v_j) = \infty$, $j > 1$; $d(s,s) = 0$

$i = 1, \ldots, n-1$

Ha $(v_i, v_j) \in E$, akkor $d(s, v_j) = \min\{d(s, v_j), d(s, v_i) + c(v_i, v_j)\}$

a végén $d(s, v_i)$-k a keresett értékek.

lépések: v_i kifoka, összesen $O(n+e)$

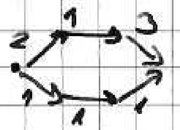
Emne **Algel** Dato 2009.04.15 Friedl

dag-ban s-ből a többiben menő leghosszabb utak $O(n+e)$

előző alg-ban min helyett max kell.

(M) általában gráfban leghosszabb út keresésére nem ismert polinom idejű algo.

Munka, több részfeladat, közöttük olyan megkötések, hogy A előbb végzendő, mint B → súlyozott gráf.



legrövidebb befejezési idő = leghosszabb út forrás és nyelő között

gráf: dag.

PERT (Program Evaluation and Review Technique)

2. algo = PERT módszer.

$$d(s, v_j) = \max_{(v_i, v_j)\in E} \{ d(s, v_i) + c(v_i, v_j) \}$$

$O(n+e)$, ha fordított éllista van (bemenő élekből)

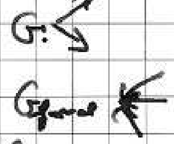
alkalmazható a legrövidebb útra is: max helyett min kell.

<u>Erősen összefüggő</u> a G irányított gráf, ha bármely 2 pontja között van irányított út.

Erősen összefüggőség eldöntése.

- 1. alg: $\forall$ v pontból DFS erősen öf $\Leftrightarrow$ mindegyik mélységi feszítőerdő fa.
  (v-ből fát kapunk = v-ből $\forall$ csúcs elérhető irányított úton)
  $n \cdot O(n+e) = O(n^2 + ne)$

- 2. alg: Tetszőleges v pontból DFS G-n.
  v pontból DFS G megfordítottján.



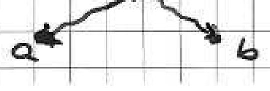
erősen öf G, ha mindkét mélységi feszítőerdő fa

1. bejárásnál fát kapunk = v-ből $\forall$ pont elérhető irányított úton G-ben
2. ——— G_{ford}-ban.

$\Updownarrow$

v minden pontból elérhető irányított úton G-ben.

a-ból b-be el lehet jutni v-n keresztül



gráf: erősen öf. komponensek.

két komponens között csak 1 irányba mehet él.

komponensek gráfja (csúcsok = komponensek) = dag.

Alg. erősen öf. komponensek megtalálására:

G-ben DFS

1 bejárásból

G_{transz}-ban DFS: ∀ fát a még bejáratlan pontok között a legnagyobb bef. számúval kezdünk.

erősen öf. komponensek = 2. bejárás fái → $O(n+e)$

Min ktg-ű feszítőfa:

$G(V,E)$ irányítatlan, egyszerű, összefüggő, súlyozott $c: E \to \mathbb{R}$

cél: $F \subseteq E$ feszítőfa súlya minimális.

Emne **Algel**

Dato 2009. 04. 22.
Friede

Zh: 04. 24.
14:15

A-J Ch Max

K-L St. Nagy
[illegible]

M-S Ka 51.

Sz-Zs E.T.B.

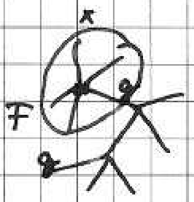
A vizsga formában a zh utáni anyagrész 50%-os súllyal fog szerepelni!!!

Emne | Dato

1 Lemma biz:

kezdetben $E = S$ takaros

$E = S \cup K \cup P$ takaros, $\exists F$ min feszfa $F \supseteq K$, $F \cap P = \emptyset$



- kék szabályt használjuk $X \subset V$ halmazra, $g \in E$ élet színezzük (kék-re)

ha $g \in F$ akkor F továbbra is jó, a színezés takaros

ha $g \notin F$ a fában út van g végpontjai között, ennek van olyan él ami X és $V-X$ között megy

legyen g' ilyen $\Rightarrow F' = F - \{g'\} \cup \{g'\}$ is feszítőfa

g' színe: nem piros mert $g' \in F$, nem kék a kék szabály miatt $\Rightarrow g' \in S$

$c(g') \geq c(g)$

$c(F) \geq c(F') \Rightarrow c(F) = c(F')$

$\Rightarrow F'$ minimális feszítőfa, ami mutatja, hogy az új színezés is takaros

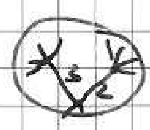
- piros szabályt használjuk a C körre, $g \in E$ élet színezzük (pirosra)

(M) ha már van $|V|-1$ kék él, akkor abbahagyhatjuk, mert ezek már egy feszítőfát adnak

Alkalmazás: (JARNIK) PRIM algoritmus

$s \in V$-ből növesztjük a fát

kék



* $X = \{s\}$ kék szabály

kék élek által elért pontok $= X$

⇓

ezekre kék szabály

piros-kék algo ⇒ ez jó

Emne **Algel** Dato 2009. 04. 22. Friedl

U: kék fa pontjai

$$\text{KÖZEL}[i] = \begin{cases} *, & \text{ha } i \in U \\ j, \; j \in U, \; c(i,j) \text{ min} & \text{ha } i \notin U \end{cases}$$

bevesszi U-ba

kék szabály alkalmazásakor szóba jövő élek: $\{i, \text{KÖZEL}[i]\}$ $i \notin U$

választás: ezek közül min.

utána KÖZEL frissítése.

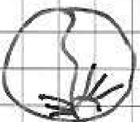
$x \notin U$ KÖZEL[x]: ha $c(i,x) < c(x, \text{KÖZEL}[x])$, akkor KÖZEL[x] = i

KÖZEL[x] = *

lépésszám: $\underbrace{(n-1)(n-1)}_{\text{min keres.}} + \underbrace{(n-1)\,O(n)}_{\text{közel frissít}} = O(n^2)$

az aktuális U és V-U között voltak, de már bekerültek U-ba.

<u>KUPACCAL</u>: U és V-U közötti élekből (+ korábbi maradékok)



MINTÖR által adott élről ellenőrizni kell, hogy

U és V-U között megy-e, ha nem, újabb MINTÖR

(vagyis nem akkor töröljük élet, amikor U-ba kerül, mert az nehézkes, hanem akkor amikor a kezünkbe kerül)

kupac előállítás megoldásánál: $O(e) + \underbrace{2e\,O(\log e)}_{\text{BESZÚR, MINTÖR}} = O(e \log e) = O(e \log n)$

(ha sűrű a gráf, nem bináris kupac, jó paraméterekkel még kevesebb lépés)

<u>BORŮVKA-alg</u> (párhuzamos alg)

kezdetben ∀ csúcs kék fa

∀ kék fából kivezető legkisebb élek kékre színezi (→ egy lépésben több komponens egy nagyobbá olvad össze)

(baj van ha azonos súlyú élek: lehet kör → egyszerűen javítható pl komponensek megszámozásával legkisebbet választ)

∀ menetben feleződik a fák száma (legrosszabb esetben) → log n menet max

⇓

$O(e \log n)$

Emne

Dato

Kruskal - alg ① növekvő sorrendben élek

② kékre színezzük, ha lehet (nem hoz létre ~~kék~~ kört)

kék vagy piros szabály.

① • rendezés : $O(e \log e) = O(e \log n)$

• kupac : ——— // ———

② keletkezik-e kör?

adatszerkezet: unió-holvan : ~~[illegible]~~

X alaphalmaz, $A_1, \dots, A_k \subseteq X$; $A_1 \cup \dots \cup A_k = X$; $A_i \cap A_j = \emptyset$ ha $i \neq j$

műveletek: UNIÓ(i,j) : A_i és A_j helyett $A_i \cup A_j$ lesz.

HOLVAN$(x) = i$ ha $x \in A_i$ $(x \in X)$

Kruskal - alg: rendezés + ~~[illegible]~~ élenként 2 db ~~[illegible]~~ HOLVAN + $n-1$ db UNIÓ

Unió-holvan megvalósítása:

1) tömbbel : $T[x] = i$ ha $x \in A_i$

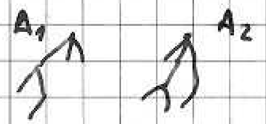
HOLVAN : $O(1)$

UNIÓ : $O(n)$, $n = |X|$

sűrű ↓ ritka él esetén dominál ↓

$\rightsquigarrow$ Kruskal: $O(e \log n) + e \cdot O(1) + (n-1)O(n) = O(e \log n + n^2)$

2) fákkal : $A_i \to$ gyökeres fa



x elem $\to$ csúcs

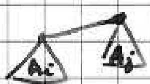
$i \to$ gyökér

(név, ezzel azonosítom a halmazt)

HOLVAN : csúcsból gyökérbe felmegyünk

~~[illegible]~~ lépésszám: O(a magasság)

UNIÓ :



ha $|A_j| \geq |A_i|$ (ilyenkor A_i az A_j fának egy részfává válik)

a kisebbik épül bele a nagyobbikba

lépésszám : $O(1)$

Kruskal : $O(e \log n) + e\,O(\log n) + (n-1)\,O(1) = O(e \log n)$

Állí: Ha kezdetben $\forall A_i$ egyelemű $\Rightarrow$ magasság $\leq$ mindig $O(\log n)$.

③ hányszor távolodik a gyökértől? $\Rightarrow$ akkor ha ~~[illegible]~~ lekerül egy legalább kétszer akkora halmazba

Legrosszabb halmaz méret min duplázódik $\Rightarrow \log n$ -szer lehet

mindig 1-gyel.

Emne **Algel** Date 2008.04.29. Friedl

pl.: lesznek feladatok...
A–E IB27
F–P ET B
R–Zs St Nagy
Szerdes?
A–Zs St Nagy

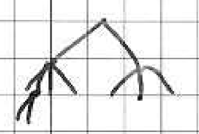
min. feszítőfa:

Kruskal lépésszáma az unió-holvan-on múlik

↓

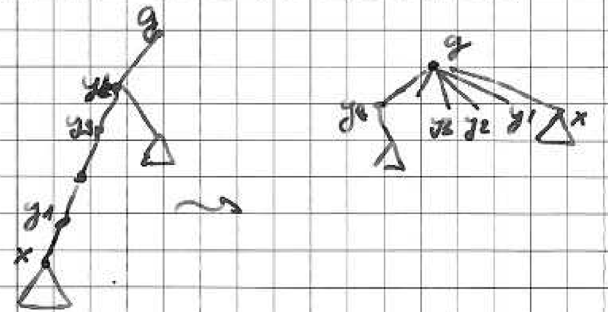
2 megvalósítás, egyikben az egyik gyorsabb, másikban a másik.

• fás megvalósítás: halmaz → fa (a fa mélysége a lényeges)



HOLVAN(x) során HOLVAN(y) kérdésre is megkapjuk a választ

• unió-holvan összenyomással



HOLVAN(x)-kor előny: $y_1, y_2, \dots, x$ vonatkozó HOLVAN gyors lesz.

Tétel: Ha 1 elemű halmazokkal kezdünk, $n-1$ UNIÓ és $m \geq n-1$ HOLVAN kérdés lépésszáma $O(m\,\alpha(n))$

n az elemek száma.
m: élek száma

$\alpha(n)$ jellemzői

• Ackerman fv inverz

• $\alpha(n)$ monoton nő

• $\alpha(n) \to \infty$ $n \to \infty$

• $\alpha(n) < 4$ ha $2^{2^{2^{2}}} = 2^{65536}$ (vagyis gyakorlatilag tekinthető konstansnak)

7B

Ebből Kruskal-alg. lépésszáma $O(e \log n) + O(\underbrace{2e \cdot \alpha(2e)}_{\text{HOLVAN}}) = O(e \log n) + O(e \cdot \alpha(2e))$

rendezés

itt van az a hiba:

Ⓜ • ha az élsúlyokat tudjuk lin. időben rendezni (pl. súly 1 és n között egész) ⇒

⇒ lépésszám $O(e) + O(e \cdot \alpha(2e)) = O(e \cdot \alpha(2e))$

• Nyitott kérdés, hogy van-e $O(e)$ lépésszámú algo? általános súlyok esetén?

• Tarjan: ellenőrizni, hogy min feszítőfa-e $O(e)$ lépés (de megtalálni még nem tudjuk)

véletlen mintavétel használatával minimális feszítőfa lépésszáma $= O(e)$

Bonyolultságelmélet

hatékony → polinom idejű (a bemenet hosszának fv-e polinom)

(Pl) • n elemű halmaz, cél az összes részhalmaz felsorolása.

↓

eredmény exponenciális hosszú (ha csak írunk és nem számolunk, akkor is)

⇓

nem lehet rá polinomiális alg.

• bemenet : $n \in \mathbb{N}$, cél 3^n

bemenet hossza : $\log(n+1)$

kimenet : $\log 3^n$ nem polinom hosszú

• $n \in \mathbb{N}$, $m \in \mathbb{N}$, cél $3^n \bmod m$

- $3 \cdot 3 \cdot 3 \cdot \ldots \cdot 3$ → $n-1$ szorzás nem polinom ($\log n$)-ben
- négyzetre emelések mod m

$$\left.\begin{array}{l} 3^2 \bmod m \\ 3^4 \bmod m \\ 3^8 \bmod m \end{array}\right\} \text{ezek közül a megfelelőket összeszorozni.}$$

⋮

⇓

$\log n$ db ilyen lesz ⇒ $O(\log n)$ szorzás, moduló.

Döntési problémák : kimenet igen/nem

pl. megállási probléma : bemenet (P, Q)

program, bemenet

eredmény : P a Q bemenettel indítva megáll-e. → nincs rá módszer, van róla egy verzió a tárgy honlapján.

(D) P = polinom időben megoldható döntési problémák.

(Pl) G gráf összefüggő? P-ben van.

NP : ~~Az~~ nemdeterminisztikus polinom időben megoldható döntési problémák

Ha az x bemenetre a válasz igen, akkor van olyan polinom hosszú y (bizonyítéktanú), hogy az (x, y) páros polinom időben ellenőrizhető, hogy y bizonyítja az „igent". (=Röviden bizonyítható)

Ha az x bemenetre a válasz nem, akkor ∄ ilyen y.

Emne

Dato

(Azaz $A \in NP$ ha van olyan $B \in P$, amikor x-re A-nál ~~igen~~ nem a válasz, akkor $\nexists y$ pol. hosszú ~~hogy~~, hogy (x,y)-ra B-nél igen;)

Pl. H: G gráfban $\exists$ Hamilton-kör

$H \in NP$, mert $x = G$ -hez y = Hamilton kör mentén a pontok felsorolása ⇒ y hossza $= |y|$ polinomiális $|x|$-ben.

Ellenőrzés: y-ban $\forall$ pont egyszer szerepel egymás után pontok és az első-utolsó pont között van él.

Ez jó, ha x-ben van Hamilton-kör, és nincs jó y ha x-ben nincs Hamilton-kör (y az a pici infó, ~~hogy~~ amitől el tudjuk dönteni, hogy x lehet-e jó)

Összetett: n szám összetett szám.

$\in NP$, mert $1 < y < n$ egész ellenőrzés $y \mid n$

Tul: $P \subseteq NP$

Nyitott: $P \stackrel{?}{=} NP$

Eldöntési probléma A: igen/nem.

komplementer problémája ($= \bar{A}$) $\bar{A}$: x bemenetre igen $\Leftrightarrow$ ha A-nál nem.

pl $\overline{ÖSSZETETT} = PRÍM$

D coNP a nem válaszra van polinomiális bizonyíték, azaz $A \in coNP \Leftrightarrow \bar{A} \in NP$

Tul: $P \subseteq coNP \Rightarrow P \subseteq NP \cap coNP$

Nyitott: $P \stackrel{?}{=} NP \cap coNP$

Pl: SÍK: G gráf síkba rajzolható

$\in NP$ Y: pontok koord. a síkban (racionális, élek = szakaszok)

Emne

Dato

Sík $\in$ coNP Y: Kuratowski gráf

$\Rightarrow$ Sík $\in$ NP $\cap$ coNP

ább, SÍK $\in$ P (TBiz)

D KARP-redukció (polinomiális visszavezetés)

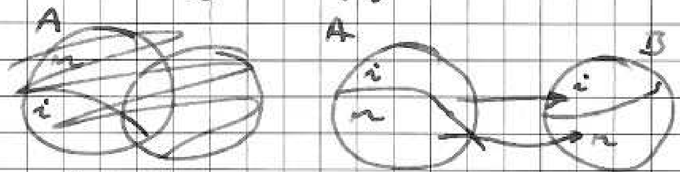
A Karp-redukciója B-re (A,B : eldöntési problémák)

Emne Algel

Dato 2005.05.06 Frédl

Fel: Ha $A < B$ akkor $\bar{A} < \bar{B}$

(B) f A-ból B-re Karp-redukció. Ugyanez az f jó $\bar{A}$-ból $\bar{B}$-re Karp redukciónak



Áll: Ha $A < B$ $B \in NP$ akkor $A < NP$

(B) f Karp-redukció

$\forall x$: $B(x) =$ igaz van polinom hosszú y tanúja, amit poli. időben ellenőrizhetünk.

z : bemenete A problémának

$f(z)$ tanúja egy jó tanú A-ra.

ellenőrzés: $|f(z), y|$-ra B-hez tartozó ellenőrzés

Kell: y hossza, ellenőrzés lépésszáma polinomja z lépésszámának.

Tudjuk ——— $f(z)$ hosszának

Mivel f poli időben számolható, ezért $f(z)$ hossza z hosszában polinomiális} polinom; valóban teljesül is

Áll: Ha $A < B$ és $B \in coNP$ akkor $A \in coNP$

(B): $B \in coNP \Leftrightarrow \bar{B} \in NP$

$A < B \Rightarrow \bar{A} < \bar{B} \Rightarrow \bar{A} \in NP \Rightarrow A \in coNP$ így kijött, hogy egy belátnunk kellett volna.

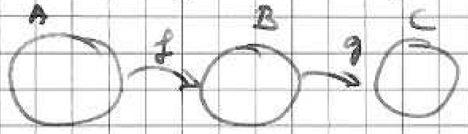
Emne Dato

Áll: Ha $A < B$ és $B < C$ akkor $A < C$ (tranzitivitás)
a, B-re vonatkozó programot nem használhatom, csak a végén és az eredményt el kell fogadnom

(B)

$f: A < B$

$g: B < C$



$g(f(x))$ Karp-redukció $A < C$

$A(x) = \text{igaz} \Leftrightarrow B(f(x)) = \text{igaz} \Leftrightarrow C(g(f(x))) = \text{igaz}$

és polidőben számolható: $x \to f(x) \to g(f(x))$

polidő $|x|$-ben; polidő $\underbrace{|f(x)|\text{-ben}}_{\text{polinom } x\text{-ben is.}}$

$\Rightarrow g(f(x))$ polidőben számolható ($|x|$-ben)

(D) NP-teljes az A probléma, ha $A \in NP$ és $\forall B \in NP$ $B < A$

(T) 3-SZÍN NP-teljes

bemenet G gráf
kérdés: $\chi(G) \leq 3$

Biz: 3-SZÍN $\in$ NP

tanú: $a_1, a_2, \dots a_n$ (egy feltételezett színezés)

ell: a_i: 1, 2, 3 valamelyike (tényleg 3 színnel színezve)

n csúcsa van a gráfnak

$\forall$ él 2 végpontja különböző színű

(itt akkor kell hogy polidőben megy-e az alg, mindegy, hogy $n \cdot k$-szal vagy éllistával van megadva, mert egyik a másikból polidőben előállítható)

NP-teljes: 7B

További NP-teljesség bizonyítások váza

A probléma: 1.) $A \in NP$

2.) veszünk egy tetszőleges NP-teljes problémát: B

elég megmutatni, hogy $B < A$ (mert akkor $\forall C \in NP$: $C < B < A$, tehát $C < A$)

Emne: Algel

Dato: 2006.05.06. B Frizell

Ha egy NP-teljes problémára találunk egy polinomiális alg-t, akkor P=NP

Mert A NP-teljes, $A \in P$

tetsz. $B \in NP$, $B \prec A \Rightarrow B \in P \Rightarrow NP \subseteq P$

De $P \subseteq NP$

$\Rightarrow P = NP$

(Pl) Bemenet: G gráf, $k \in \mathbb{N}$ poz. egész.

Van-e k fgt pontja? (Azaz $\alpha(G) \geq k$) → MAXFTL

MAXFTL ∈ NP (tanú k db pont

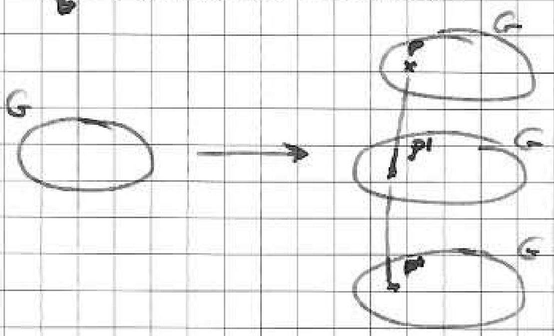
(T) MAXFTL NP-teljes

Biz: ∈ NP ✓

3-SZÍN ≺ MAXFTL

$f: G \to G', k$

G 3 színnel színezhető ⇔ G'-ben van k ftl pont



3 példány 1 pont példányait összekötjük ⇒ G'

k = G pontjainak száma

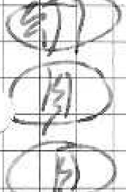
Ha $\chi(G) \leq 3$

színezés:

i. színű pont i. példánya kerül be a független halmazba G'-ben.

⇓

ez valóban fgt, mérete = k.



G'-ben fgt, ∀ G-beliének legfeljebb 1 példányát tartalmazza.

k db van ⇒ ∀ G-beliének 1 példánya

i. példány → i. szín G-ben

Emne | Dato

(Pl) MAXKLIKK : bemenet G gráf, k poz egész.

kérdés: van-e k pontú teljes részgráf G-ben. ($\omega(G) \geq k$)

(T) MAXKLIKK NP-teljes

(B) NP-beli: tanú k pont, ell teljes részgráfot határoz meg.

kell: MAXFTL $<$ MAXKLIKK

$$(G, k) \to (G', k')$$

$$\alpha(G) \geq k \Leftrightarrow \omega(G') \leq k'$$

$f: G' = \overline{G}$, $k' = k$ polinomiálisan számolható → jó

(Pl) H. bemenet G gráf

kérdés: van-e Hamilton-kör benne.

HÚT. bemenet G-gráf

kérdés: van-e H-út G-ben

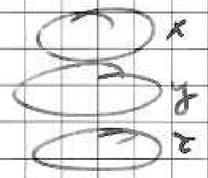
H,s,t,ÚT : bemenet G gráf, s, t csúcsok

kérdés: van-e olyan H-út, aminek végpontjai s és t.

(T) ezek irányított és irányítatlan esetben NP-teljesek „ez a kié ezzel a lánggal és ezzel a hangjával igen"

Biz.

3DH (3 dimenziós házasítási probléma)



$S \subseteq X, Y, Z$ bemenet

kérdés: van-e olyan $T \subseteq S$, hogy T minden pontot pontosan egyszer fed.

(T) 3DH NP-teljes

Biz

(M) 2DH: páros gráfban van-e teljes párosítás.

2DH $\in$ P

(Pl) X3C (exact 3 cover)

bemenet: X alaphalmaz, $F_1, F_2, \ldots F_m \subseteq X$, $|F_i| = 3$.

Emne: Algel — Dato: 2009. 05. 06. Friedl

Kérdés: van-e olyan $I \in \{1,\dots,k\}$: $\bigcup_{i\in I} F_i = X$

$F_i \cap F_j = \emptyset$, ha $i \neq j$, $i,j \in I$.

(T) X3C NP-teljes

(B) $X3H < X3C$ kf.

(M) X2C általános gráfban teljes párosítás $\in P$

(Pl) RH (részhalmazösszeg)

bemenet: $a_1, a_2, \dots, a_n \geq 0$ egészek, $b \geq 0$ egész.

kérdés: van-e $I \subseteq \{1, 2, \dots, n\}$ $\sum_{i\in I} a_i = b$

PARTICIÓ: bemenet $a_1, \dots, a_n \geq 0$ egész.

kérdés: van-e $I \subseteq \{1, 2, \dots, n\}$: $\sum_{i\in I} a_i = \sum_{j\notin I} a_j$

HÁTIZSÁK: bemenet: $a_1, a_2, \dots, a_n \geq 0$ egész, $v_1, v_2, \dots, v_n \geq 0$ egész.

$b \geq 0$ egész $k \geq 0$ egész.

kérdés: van-e $I \subseteq \{1, 2, \dots, n\}$ $\sum_{i\in I} a_i \leq b$, $\sum_{i\in I} v_i \geq k$

(T) RH, PARTICIÓ, HÁTIZSÁK NP-teljes.

(Pl) RÉSZGRÁFIZO

bemenet: G_1, G_2 gráf.

kérdés: van-e G_1-nek G_2-vel izomorf részgráfja.

NP-teljes $\Leftarrow$ H <

$\Leftarrow$ MAXKLIKK <

MAXFTL <

(Pl) GRÁFIZO:

bemenet: G_1, G_2 gráf.

kérdés: izomorfak-e.

$\in NP$ (tanú: izomorfizmus)

Nyitott, hogy $\overset{?}{\in} P$ vagy NP-teljes, vagy egyik sem.

Emne: Algel

Dato: 2005. 05. 23
Friedl.

P, NP   döntési probléma → nyelv: az igaz válaszokhoz tartozó bemenetek halmaza

L

$x \in L$

P, NP tekinthető nyelvek halmazaként (nyelvosztály)

Lineáris programozás

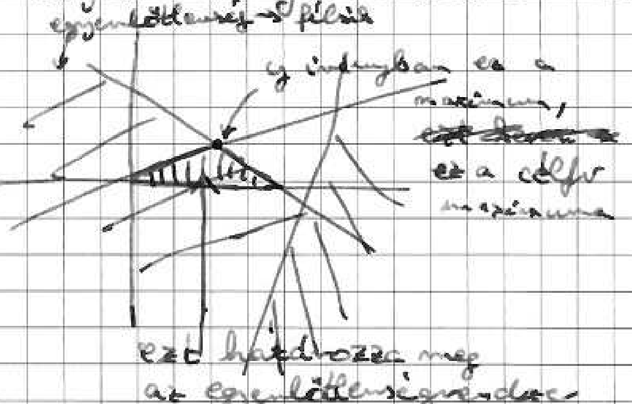
$\sum_{j=1}^{n} a_{ij} x_j \le b_i$ a_{ij}, b_i adott egészek, $x_1, \dots, x_n$ ismeretlen

$x_j \ge 0$

Cél: olyan c_j-ket találni, amelyre $\sum c_j x_j$ maximális, $c_1, \dots, c_n$ adott egészek

célfüggvény: irány

Mondjuk 2D-ben geometriai tartalom:



Algoritmus: szimplex módszer (Dantzig, 1947)

maximum csak a csúcson lehet, végigmegyünk ezeken.

exponenciális alg., de a gyakorlatban jól működik

- Ellipszoid módszer (Hacsijan, 1979)

  polinomiális algoritmus (csak túl nagy értékekkel)

← végülis egy döntési probléma, hogy van-e ilyen célfv, de sokáig nem tudták, hogy $\in P$ vagy NP-teljes.

- belső pont módszer (Karmarkar, 1984)

  gyakorlatban is használható (a továbbfejlesztése)

Emne Dato

Mi van, ha csak az egész megoldások érdekelnek, azaz $x_1,\dots,x_n \in \mathbb{Z}$

→ Egész(értékű) programozás (EP)

Tétel: Az EP döntési változata NP-teljes

NP-beliség sem triviális (ehhez az kell, hogy van rövid megoldás)

MAXFTL átfogalmazása EP feladatra

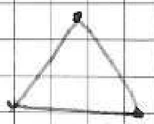
gráf csúcsai → x_i $\quad x_i \ge 0$, $x_i \le 1$ (vagyis a csúcs értéke 0 vagy 1)

$\{i,j\}$ él → $x_i + x_j \le 1$ (csak az egyik csúcs lehet benne a ftl ponthalmazban)

max ftl halmaz → $\max \sum_i x_i$ (a legtöbb csúcs legyen benne)

ftl halmaz $= \{ i : x_i = 1 \}$ (az lesz a ftl halmaz, ~~amely~~ akik a csúcsok értéke 1)

vagyis tényleg NP-teljes probléma ez is.

Pl. gráf  $0 \le x_1, x_2, x_3 \le 1$

$x_1 + x_2 \le 1$
$x_2 + x_3 \le 1$
$x_1 + x_3 \le 1$

$\max \sum_i x_i$

ha nem csak egész mo-kat keresünk
⇓
a 3 egyenlőtlenség összege:

$2x_1 + 2x_2 + 2x_3 \le 3$

$x_1 + x_2 + x_3 \le \frac{3}{2}$ maximalizálja $\sum x_i$-t

és $x_1 = x_2 = x_3 = \frac{1}{2}$-del megoldható

← de ezt nem lehet gráfra lefordítani

Nehéz problémák esetén:

- próbáljunk speciális esetet megoldani (pl: leghosszabb út: dag-ban polinom, általában nehéz, pl. #-ít)
- elágazás és korlát, din. pr. (lehet, hogy az alg. exponenciális, de [illegible] lefut)

Közelítő algoritmusok:

- maximalizálási feladat → ennek értéke OPT
- c-közelítő algoritmus egy olyan megoldást talál, aminek értéke $\ge c \cdot OPT$ $(c \le 1)$
  (1-hez közeli c a jó)

Emne: Algel   Dato: 2009.05.13. Friedl.

• minimalizálásnál

c-közelítő $\leq c \cdot OPT$ $(c \geq 1)$

(M) 1-közelítő alg. az optimumot találja meg.

(Pl) G gráf, max ftl él. (= max párosítás probléma)
– van polinomiális algo.

mohó algo.: fgt. éleket vesz, amíg tud

lineáris alg.: $O(n+e)$

lefogó pontok; max fgt. élek száma

megtalált fgt élek száma $\geq \frac{\tau(G)}{2} \geq \frac{\nu(G)}{2} = \frac{OPT}{2}$

$\frac{1}{2}$-közelítő alg.

<u>Utazó ügynök</u> Adott egy G irányítatlan gráf, az élein súlyok $s: E \to \mathbb{R}$

Feladat: minimális súlyú Hamilton-kör

Döntési változat: adott G, s, k

Kérdés: van-e legfeljebb k súlyú H-kör

Tétel: döntési változat NP-teljes

(B) $\in NP$: tanú a csúcsok felsorolása   polinom hosszú ✓

ellenőrzés: ∀ csúcs pontosan 1x szerepel, szomszédok között és a 2 vége között van él

összsúly $\leq k$

nehézségi reláció, H nem lehet nehezebb a másiknál

= visszavezetjük

$H \leq$ ~~ezt~~ erre a problémára

$G \to G^*$ $s \equiv 1$, $k = |V(G)|$

(M) sőt ha $G = K_n$, G-beli nem-élek nagy súlyt kapnak

(T) Ha valamilyen $c \geq 1$ konstansra van polinomiális c-közelítő alg. az utazó ügynök problémára, akkor $P = NP$

(B) Ilyen algoritmussal eldönthető lenne a H-probléma (lényegében Karp-val)

adott G kérdés: van-e benne Hamilton-kör

Emne | Dato

$n = |V(G)|$, K_n $\quad s(i,j) = \begin{cases} 1, & \text{ha } \{i,j\} \in E(G) \\ c\cdot n+1 & \text{ha } \notin \end{cases}$

Erre a problémára utazó ügynök

∀ él súlya = 1

Ha G-ben van H-kör ⇒ itt van n értékű mo.

mert belekerül ≥ 1 nagy súlyú él is.

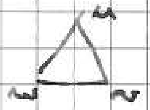
Ha G-ben nincs H-kör ⇒ ∀ megoldás értéke $\geq c\cdot n + n$

Ha a c-közelítő algo $\leq c\cdot n$ értékű mo-t ad, akkor G-ben van H-kör.

Ha $> c\cdot n$ értékűt ad, akkor nincs n értékű ⇒ ∄ H-kör G-ben.

Euklideszi utazóügynök: utazó ügynök, a súlyokra teljesül a Δ-egyenlőtlenség

$G = K_n$



$s(u,v) \leq s(u,w) + s(w,v)$

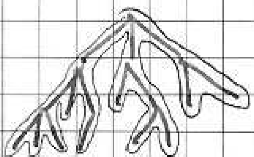
∀ u, v, w csúcsra

(T) Ennek a döntési változata is NP-teljes

7B

De: 2-közelítő algoritmus:

min súlyú feszítőfa = F (van rá polidejű alg.)



vagyis kiindulva, a csúcsok mentén

„körbe megyünk a fán", és levágjuk a ~~[illegible]~~ visszalépéseket.

= preorder sorrendben vesszük a pontokat = Hamilton-kör

Ennek súlya $\leq 2\, s(F)$

$\text{Hamilton-kör} = \underbrace{\text{H-út}}_{\substack{\text{feszítőfa} \\ \geq s(F)}} + \underbrace{\text{él}}_{\geq 0}$

<u>Ládapakolás</u>: 1 méretű ládák vannak

n db tárgy, méreteik $s_1, \dots, s_n$, $0 < s_i < 1$ $\quad s_i \in \mathbb{Q}$

cél: minél kevesebb ládába berakni ∀-t.

(T) döntési változat NP-teljes

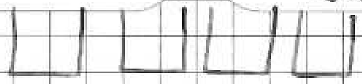
7B

Közelítő alg: First Fit: sorban ∀ tárgy az első olyan ládába kerül, ahova befér.

↓ polidejű.

pl: $\frac{1}{3}; \frac{1}{3}; \frac{1}{3}; \frac{2}{3}; \frac{2}{3}; \frac{2}{3}$

OPT = 3 láda $\quad$ FF = 4 nem opt.



Áll: Ez 2-közelítő alg.

Biz: - a ládák legfeljebb 1 kivétellel

kevesebb, mint félig tele

többen mint a félig tele vannak (különben a 2 ilyen egymásra került volna)

• két láda együtt > 1 méretű tárgyat tartalmaz.

Ha L láda van: $1 + (L-2)\frac{1}{2} < \sum_{i=1}^{n} s_i \le OPT$

kivételes valamelyik

(T) $\forall$ bemenetre $FF \le \lceil 1{,}7 \cdot OPT \rceil$

$\exists$ bemenet: $FF \geq 1{,}7(OPT-1)$

2. Alg. First Fit Decreasing (FFD)

csökkenő sorrendbe rendezi a tárgyakat, utána FF

(T) $\forall$ bemenetre $FFD \le \frac{11}{9} OPT + 4$

$\exists$ bemenet $FFD \geq \frac{11}{9} OPT$

(T) $\forall\, \varepsilon$ van $(1+\varepsilon)$-közelítő polinom idejű algoritmus.