Algoritmuselmélet 3. előadás

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

2002 Február 18.

Gyorsrendezés

[C. A. R. Hoare, 1960]

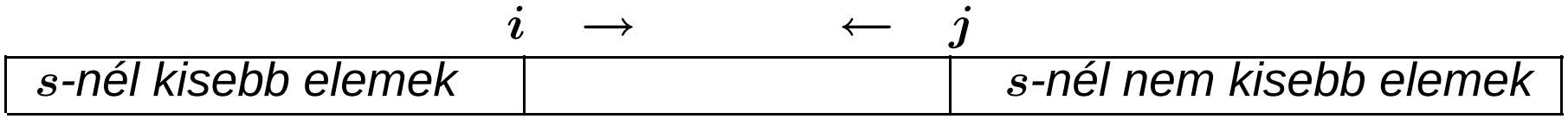
oszd meg és uralkodj: véletlen s elem a tömbből $\Longrightarrow$ PARTÍCIÓ(s) $\Longrightarrow$

s-nél kisebb elemek	s	$\ldots$	s	*s-nél nagyobb elemek*

GYORSREND($A[1:n]$)
1. Válasszunk egy véletlen s elemet az A tömbből.
2. PARTÍCIÓ(s); az eredmény legyen az $A[1:k]$, $A[k+1:l]$, $A[l+1:n]$ felbontás.
3. GYORSREND($A[1:k]$); GYORSREND($A[l+1:n]$).

A PARTÍCIÓ(s) működése

Legyen $i := 1, j := n$,
$\Longrightarrow$ i-t növeljük, amíg $A[i] < s$ teljesül
$\Longrightarrow$ j-t csökkentjük, amíg $A[j] \geq s$
$\Longrightarrow$



Ha mindkettő megáll (nem lehet továbblépni), és $i < j$, akkor $A[i] \geq s$ és $A[j] < s$ $\Longrightarrow$
Kicseréljük $A[i]$ és $A[j]$ tartalmát $\Longrightarrow$ $i := i + 1$ és $j := j - 1$. Ha a két mutató összeér (már nem teljesül $i < j$), akkor s előfordulásait a felső rész elejére mozgatjuk.
PARTÍCIÓ lépésszáma: $O(n)$
GYORSREND lépésszáma *legrosszabb esetben*: $O(n^2)$
GYORSREND lépésszáma *átlagos esetben*:
$1,39n \log_2 n + O(n) = O(n \log n)$
Java animáció: Gyorsrendezés

A k-adik elem kiválasztása

Első ötlet: válasszuk ki a legkisebbet, majd a maradékból a legkisebbet, stb.
$\Longrightarrow O(nk)$ lépés
Második ötlet: Rendezzük az elemeket, vegyük ki a k-adikat
$\Longrightarrow O(n \log_2 n)$ lépés
De van jobb: Tetszőleges k-ra meg lehet keresni $O(n)$ lépésben.

Kulcsmanipulációs rendezések

Nem csak összehasonlításokat használ.
Pl. ismerjük az elemek számát, belső szerkezetét.

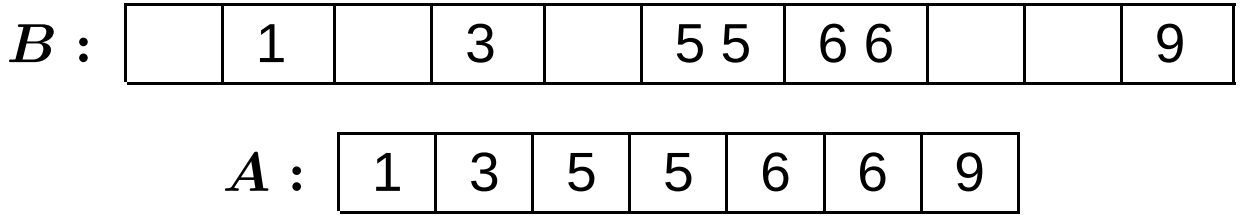
Ládarendezés (binsort)

Tudjuk, hogy $A[1:n]$ elemei egy m elemű U halmazból kerülnek ki, pl. $\in \{1, \ldots, m\}$
$\Longrightarrow$ Lefoglalunk egy U elemeivel indexelt B tömböt (m db ládát), először mind üres.
első fázis $\Longrightarrow$ végigolvassuk az A-t, és az $s = A[i]$ elemet a $B[s]$ lista végére fűzzük.
Példa: Tegyük fel, hogy a rendezendő $A[1:7]$ tömb elemei 0 és 9 közötti egészek:

A:	5	3	1	5	6	9	6

B:		1		3		5 5	6 6			9

második fázis $\Longrightarrow$ elejétől a végéig növő sorrendben végigmegyünk B-n, és a $B[i]$ listák tartalmát visszaírjuk A-ba.



Lépésszám: B létrehozása $O(m)$, első fázis $O(n)$, második fázis $O(n+m)$, összesen $O(n+m)$.

Ez gyorsabb, mint az általános alsó korlát, ha pl. $m \leq cn$.

Java animáció: Láda rendezés

Radix rendezés

A kulcsok összetettek, több komponensből állnak, $t_1 \dots t_k$ alakú szavak, ahol a t_i komponens az L_i rendezett típusból való, a rendezés lexikografikus.
Példa: Legyen $(U, <)$ a *huszadik századi dátumok* összessége az időrendnek megfelelő rendezéssel.

$$L_1 = \{1900, 1901, \dots, 1999\}, \quad s_1 = 100.$$

$$L_2 = \{\textit{január, február,..., december}\}, \quad s_2 = 12.$$

$$L_3 = \{1, 2, \dots, 31\}, \quad s_3 = 31.$$

A dátumok rendezése éppen az L_i típusokból származó lexikografikus rendezés lesz.

- rendezzük a sorozatot az utolsó, a k-adik komponensek szerint ládarendezéssel
- a kapottat rendezzük a $k-1$-edik komponensek szerint ládarendezéssel
- stb.

Fontos, hogy a ládarendezésnél, az elemeket a ládában mindig a lista végére tettük. Így ha két azonos kulcsú elem közül az egyik megelőzi a másikat, akkor a rendezés után sem változik a sorrendjük.
$\Longrightarrow$ konzervatív rendezés

Miért működik a radix jól?

Ha $X < Y$, az első $i-1$ tag megegyezik, de $x_i < y_i$, akkor az i-edik komponens rendezésekor X előre kerül.
konzervatív rendezés $\Longrightarrow$ később már nem változik a sorrendjük.
Példa:

	1969. jan. 18.	1969. jan. 1.	1955. dec. 18.	1955. jan. 18.	1918. dec. 18.
1.	1969. jan. 1.	1969. jan. 18.	1955. dec. 18.	1955. jan. 18.	1918. dec. 18.
2.	1969. jan. 1.	1969. jan. 18.	1955. jan. 18.	1955. dec. 18.	1918. dec. 18.
3.	1918. dec. 18.	1955. jan. 18.	1955. dec. 18.	1969. jan. 1.	1969. jan. 18.

Lépésszám: k ládarendezés összköltsége: $O(kn + \sum_{i=1}^{k} s_i)$

Ez lehet gyorsabb az általános korlátnál

- k = állandó és $s_i \leq cn \implies O(kn + \sum_{i=1}^{k} cn) = O(k(c+1)n) = O(n)$.
  pl. az $[1, n^k - 1]$ intervallumból való egészek rendezése

- $k = \log n$, $s_i = 2 \implies O(n \log n + 2 \log n) = O(n \log n)$.

Java animáció: Radix rendezés

A Batcher-féle páros-páratlan összefésülés

Párhuzamos algoritmus, az összefésüléses rendezés egy változata.
Az összefésülés lépése különböző, a többi rész ugyanaz.
$\mathcal{A} = a_1 < \ldots < a_l$ és $\mathcal{B} = b_1 < \ldots < b_m$ összefésülése
$\Longrightarrow \mathcal{C} = c_1 < \ldots < c_{l+m}$
1. brigád: $a_1 < a_3 < a_5 < \ldots$ és $b_2 < b_4 < b_6 < \ldots$
$\Longrightarrow u_1 < u_2 < u_3 < \ldots$
2. brigád: $a_2 < a_4 < a_6 < \ldots$ és $b_1 < b_3 < b_5 < \ldots$
$\Longrightarrow v_1 < v_2 < v_3 < \ldots$
Tétel. $c_{2i-1} = \min\{u_i, v_i\}$ *és* $c_{2i} = \max\{u_i, v_i\}$ $(1 \leq i \leq (l+m)/2)$.

Tétel. $c_{2i-1} = \min\{u_i, v_i\}$ *és* $c_{2i} = \max\{u_i, v_i\}$ $(1 \leq i \leq (l+m)/2)$.

Bizonyítás: Belátjuk, hogy
$\mathcal{C}_{2k} := \{c_1, \ldots, c_{2k}\} = \{u_1, \ldots, u_k\} \cup \{v_1, \ldots, v_k\}$.
Mivel $\mathcal{C}$ a növekvő $\mathcal{A}$ és $\mathcal{B}$ összefésülése
$\Longrightarrow \mathcal{C}_{2k} = \{a_1, \ldots, a_s\} \cup \{b_1, \ldots, b_{2k-s}\}$.
$|\{a_1, \ldots, a_s\} \cap \mathcal{U}| = \lceil \frac{s}{2} \rceil$ és $|\{b_1, \ldots, b_{2k-s}\} \cap \mathcal{U}| = \lfloor \frac{2k-s}{2} \rfloor$
$|\{a_1, \ldots, a_s\} \cap \mathcal{V}| = \lfloor \frac{s}{2} \rfloor$ és $|\{b_1, \ldots, b_{2k-s}\} \cap \mathcal{V}| = \lceil \frac{2k-s}{2} \rceil$
Mivel $\lceil s/2 \rceil + \lfloor (2k-s)/2 \rfloor = \lfloor s/2 \rfloor + \lceil (2k-s)/2 \rceil$ beláttuk az első állítást.

$\Longrightarrow \{c_{2i-1}, c_{2i}\} = \{u_i, v_i\}$ $\square$

A tétel miatt $\mathcal{U}$ és $\mathcal{V}$ már egy párhuzamos lépésben összefésülhető.

A rendezés:
MSORT$(A[1:n]) :=$
PM(MSORT$(A[1:\lceil n/2 \rceil])$, MSORT$(A[\lceil n/2 \rceil + 1 : n])$),
ahol PM a fenti párhuzamos összefésülés.
Párhuzamos lépésszám: $O(log^2 n)$

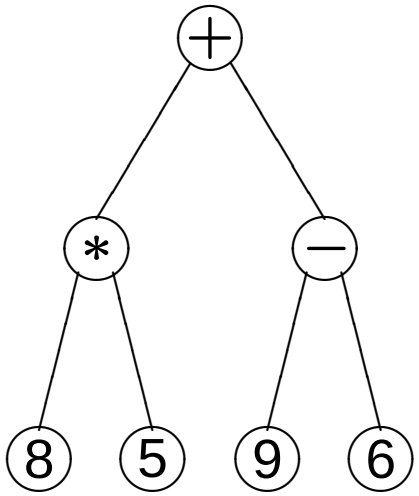
Keresőfák

Tároljuk az U rendezett halmaz elemeit, hogy BESZÚR, TÖRÖL, KERES, MIN, (MAX, TÓLIG) hatékonyak legyenek.

Bináris fa bejárása

teljes fa (új def.) $\Longrightarrow$ az alsó szint is tele van $\Longrightarrow$ l szintű, teljes fának $2^l - 1$ csúcsa van.

Fa csúcsai $\rightarrow$ $elem(x), bal(x), jobb(x)$ esetleg $apa(x)$ és $reszfa(x)$



ha x a gyökér, y pedig a 9-es csúcs, akkor

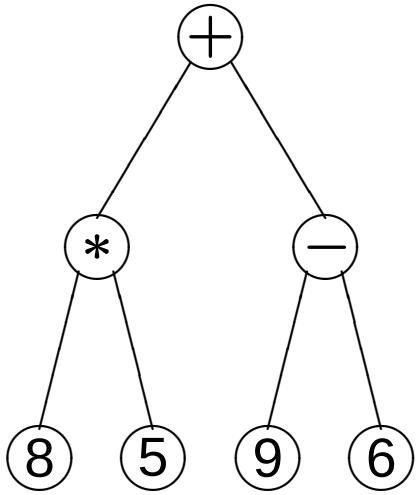
$$\begin{aligned} bal(jobb(x)) &= y, \\ apa(apa(y)) &= x, \\ elem(bal(x)) &= *, \\ reszfa(x) &= 7. \end{aligned}$$

PREORDER, INORDER, POSTORDER

pre(x)
begin
látogat(x);
pre($bal(x)$);
pre($jobb(x)$)
end

in(x)
begin
in($bal(x)$);
látogat(x);
in($jobb(x)$)
end

post(x)
begin
post($bal(x)$);
post($jobb(x)$);
látogat(x)
end



ha x a gyökér, y pedig a 9-es csúcs, akkor

PREORDER: $+ * 85 - 96$

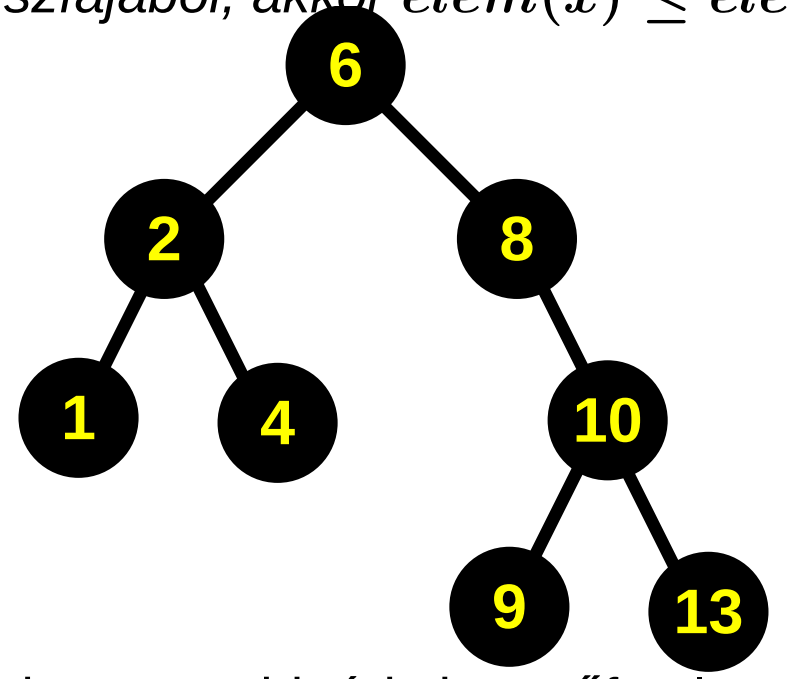
INORDER: $8 * 5 + 9 - 6$

POSTORDER: $85 * 96 - +$

Lépésszám: $O(n)$

Bináris keresőfa

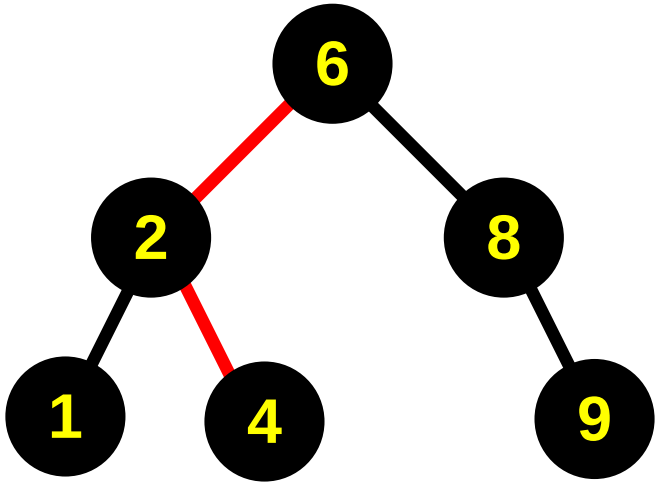
Definíció (Keresőfa-tulajdonság). *Tetszőleges x csúcsra és az x baloldali részfájában levő y csúcsra igaz, hogy $elem(y) \leq elem(x)$. Hasonlóan, ha z egy csúcs az x jobb részfájából, akkor $elem(x) \leq elem(z)$.*



Házi feladat: Igazoljuk, hogy egy bináris keresőfa elemeit a fa inorder bejárása *növekvő sorrendben* látogatja meg.

Egy kényelmes megállapodás: a továbbiakban feltesszük, hogy nincsenek ismétlődő elemek a keresőfában.

Naiv algoritmusok



KERES$(4, S)$

KERES(s,S): Összehasonlítjuk s-et S gyökerével s'-vel.

- Ha $s = s'$, akkor megtaláltuk.
- Ha $s < s'$, akkor balra megyünk tovább.
- Ha $s > s'$, akkor jobbra megyünk.

Ugyanezt az utat járjuk be a KERES$(5, S)$ kapcsán, de azt nem találjuk meg.

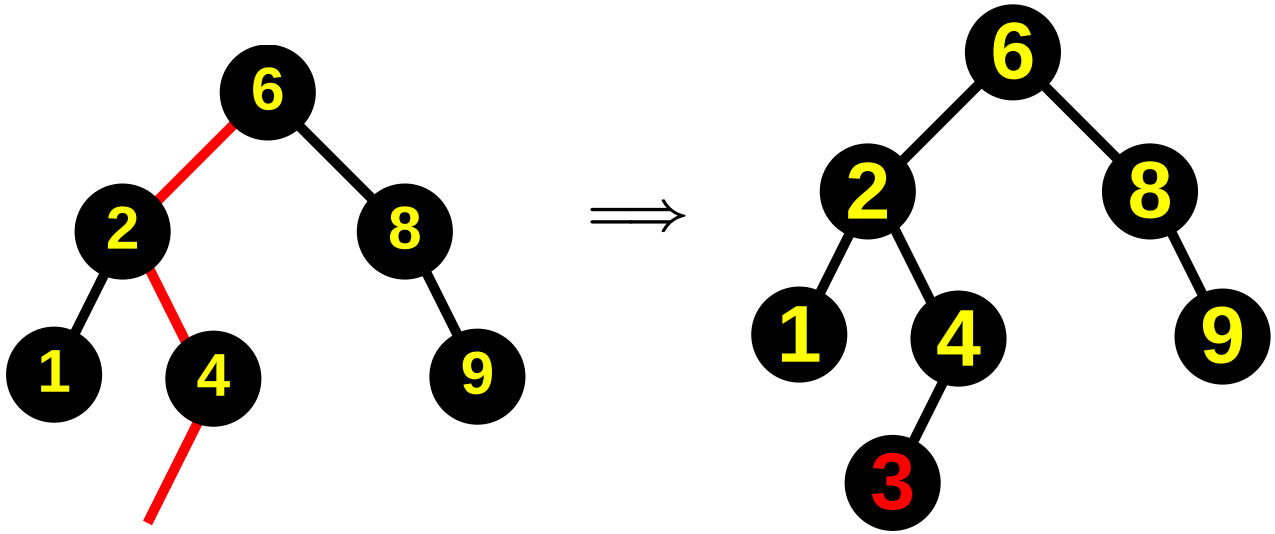
Lépésszám: $O(l)$, ahol l a fa mélysége

MIN: mindig balra lépünk, amig lehet
MAX: mindig jobbra lépünk, amig lehet Lépésszám: $O(l)$

TÓLIG(a, b, S): KERES$(a, S) \implies$ INORDER a-tól b-ig Lépésszám: $O(n)$

Naiv BESZÚR

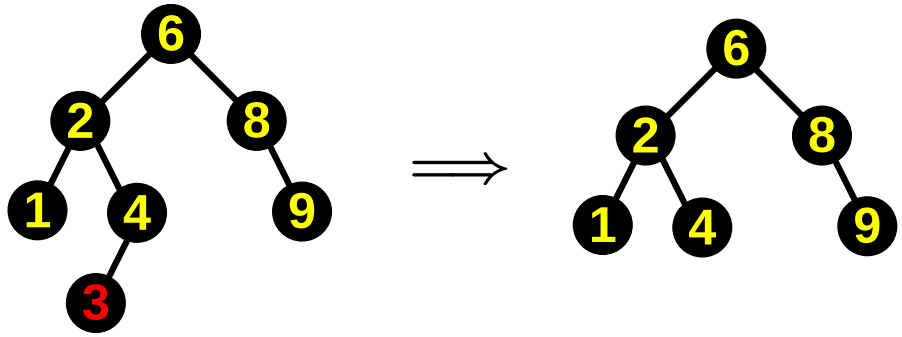
BESZÚR(s, S): KERES(s, S)-sel megkeressük hova kerülne és új levelet adunk hozzá, pl. BESZÚR$(3, S)$:



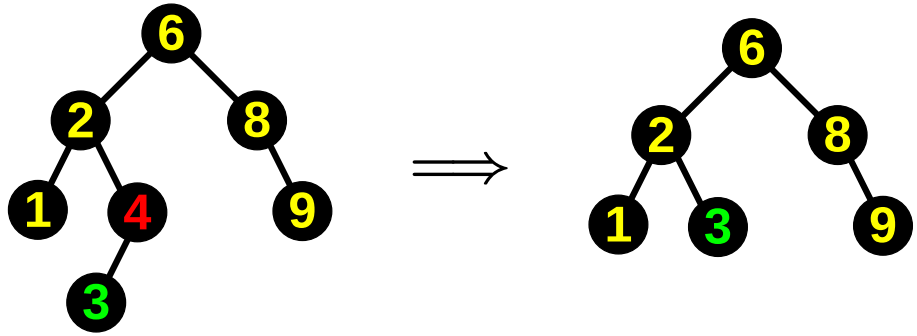
Lépésszám: $O(l)$

Naiv TÖRÖL

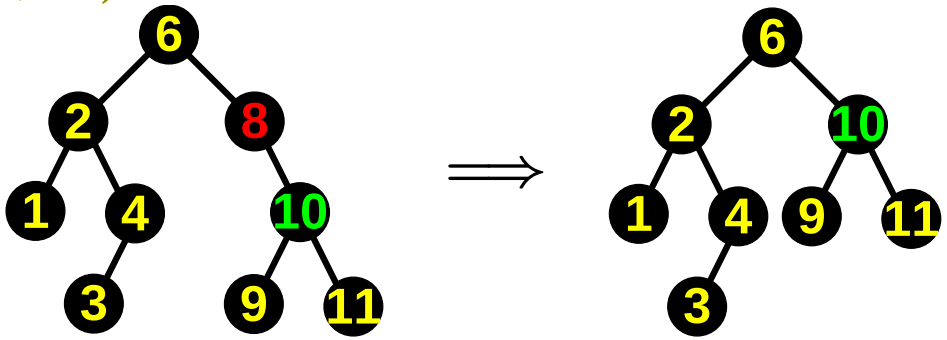
- TÖRÖL(s, S): Ha s levél, akkor trivi, pl. TÖRÖL$(3, S)$:



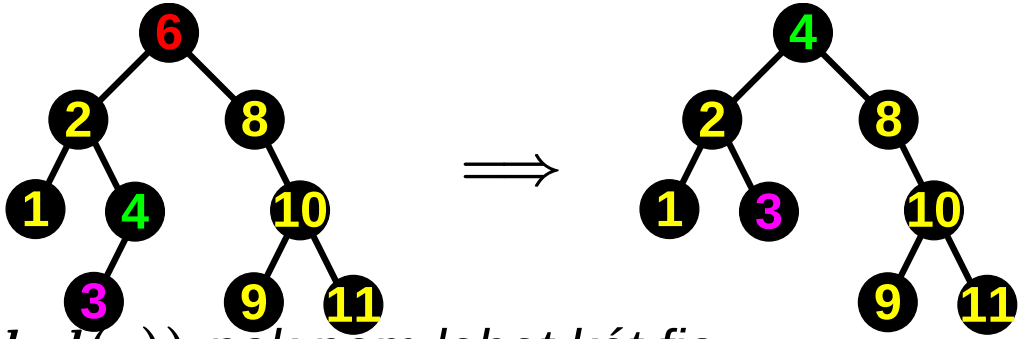
- TÖRÖL(s, S): Ha s-nek egy fia van, akkor: $s \leftarrow$ fiú(s), pl. TÖRÖL$(4, S)$:



- Vagy pl. TÖRÖL$(8, S')$:



- TÖRÖL(s, S): Ha s-nek két fia van, akkor visszavezetjük az előző esetre. s helyére tegyük $y := \mathrm{MAX}(bal(s))$-t és töröljük y-t. Pl. TÖRÖL$(6, S')$:



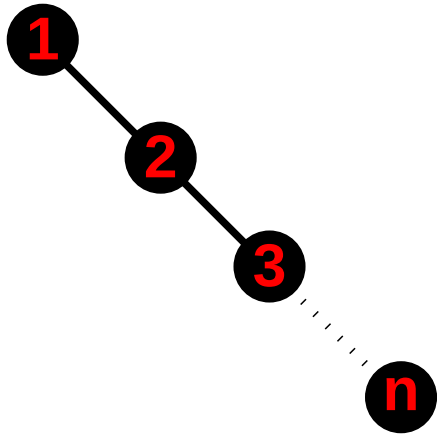
Állítás. $y := MAX(bal(s))$*-nak nem lehet két fia.*

Bizonyítás:
Ha lenne két fia, akkor lenne egy y' jobb fia is. De ekkor $y' > y$ ↯
Lépésszám: $O(l)$

Faépítés naiv beszúrásokkal

Ha pl. az $1, 2, \ldots, n$ sorozatból építünk fát így, akkor ezt kapjuk:

Az építés költsége: $2 + 3 + \ldots + (n-1) = O(n^2)$



Tétel. *Ha egy véletlen sorozatból építünk fát naiv beszúrásokkal, akkor az építés költsége átlagosan* $O(n \log_2 n)$*. A kapott fa mélysége átlagosan* $O(\log_2 n)$*.*