

Algoritmuselmélet 4. előadás

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

2002 Február 25.

Bináris keresőfa

Java animáció: Bináris keresőfa

AVL-fák

G. M. Adelszon-Velszkij, E. M. Landisz, 1962 \implies kiegyensúlyozott bináris keresőfa

Jelölje $m(f)$ az f bináris fa magasságát (szintjeinek számát), ha x az f fa egy csúcsa; ekkor $m(x)$ jelöli az x -gyökerű részfa magasságát.

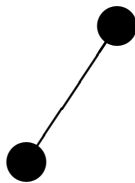
Definíció (AVL-tulajdonság). Egy bináris keresőfa AVL-fa, ha minden x csúcsára teljesül, hogy

$$|m(\text{bal}(x)) - m(\text{jobb}(x))| \leq 1.$$

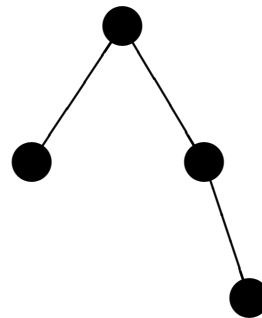
Mekkora a k szintű AVL-fa minimális csúcsszáma?



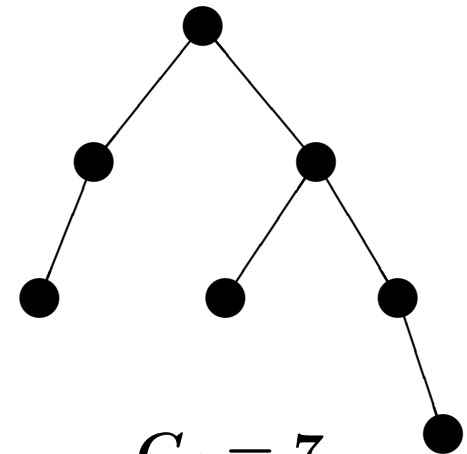
$$G_1 = 1$$



$$G_2 = 2$$

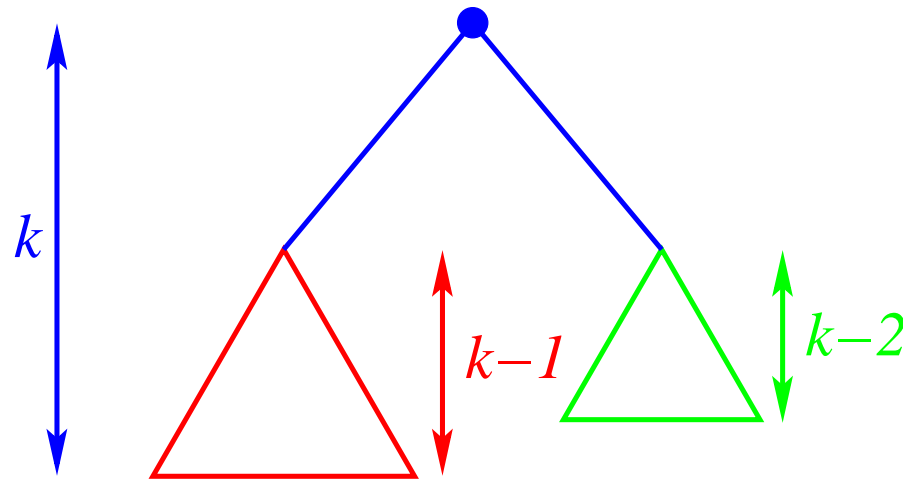


$$G_3 = 4$$



$$G_4 = 7$$

A k szintszámú minimális csúcsszámú AVL-fa gyökerének egyik részfája $k - 1$, a másik $k - 2$ szintű; az eredeti fa minimalitása miatt pedig mindkét részfa minimális csúcsszámú.



\implies rekurzió

$$G_k = 1 + G_{k-1} + G_{k-2}.$$

Tétel. $G_k = F_{k+2} - 1$ ha $k \geq 1$.

Bizonyítás: $k = 1, 2$ ✓ \implies indukció:

$$G_k = 1 + G_{k-1} + G_{k-2} = 1 + F_{k+1} - 1 + F_k - 1 = F_{k+2} - 1.$$

Tétel. Egy n -pontú AVL-fa szintjeinek k száma nem több mint $O(\log n)$, pontosabban $k \leq 1.44 \log_2(n + 1)$.

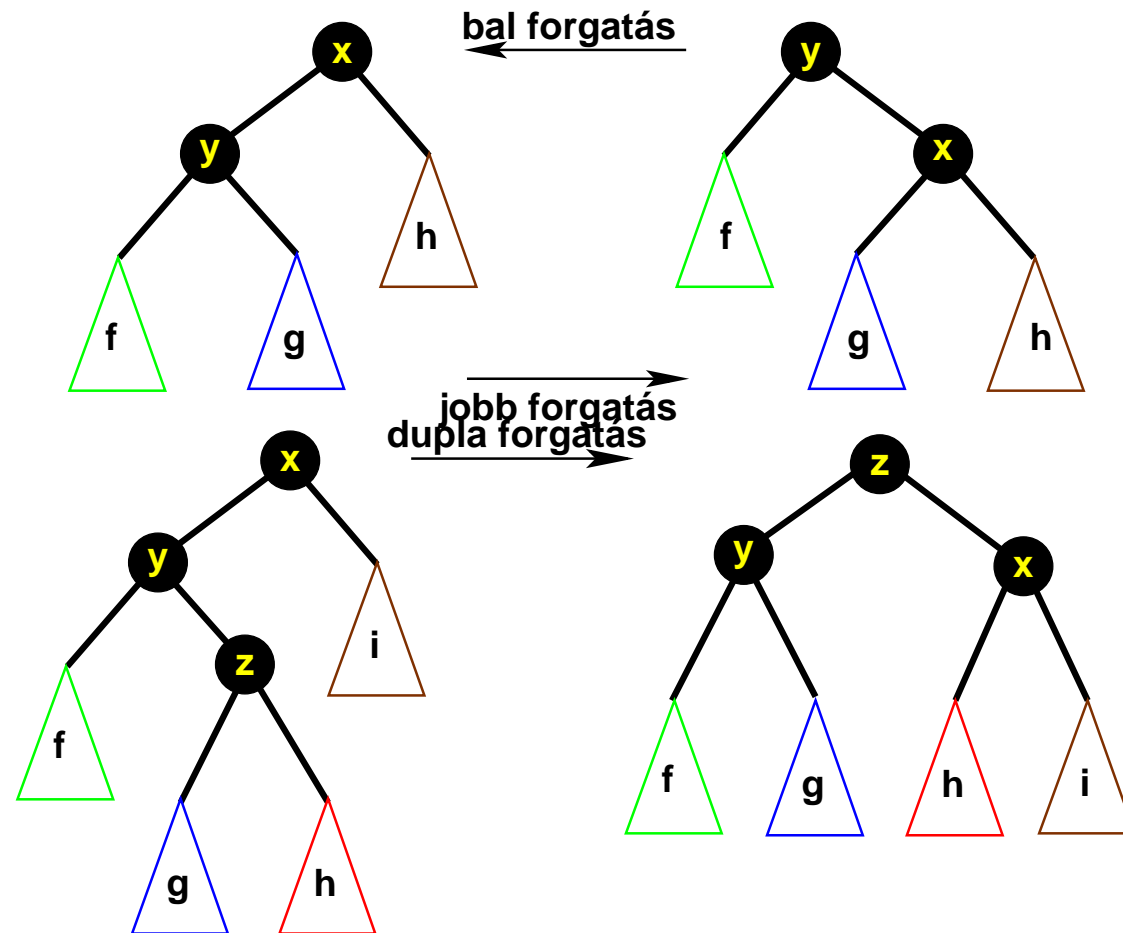
Bizonyítás: tétel $\implies n \geq F_{k+2} - 1$

Fibonacci-számokra vonatkozó alsó becslésből $n + 1 \geq \phi^k$

$\implies \log_\phi(n + 1) \geq k \implies k \leq 1.44 \log_2(n + 1)$

Az AVL-tulajdonság megőrzése

Hogyan lehet a BESZÚR és TÖRÖL eljárásokat úgy megvalósítani, hogy megtartsák az AVL-tulajdonságot? \implies forgatás



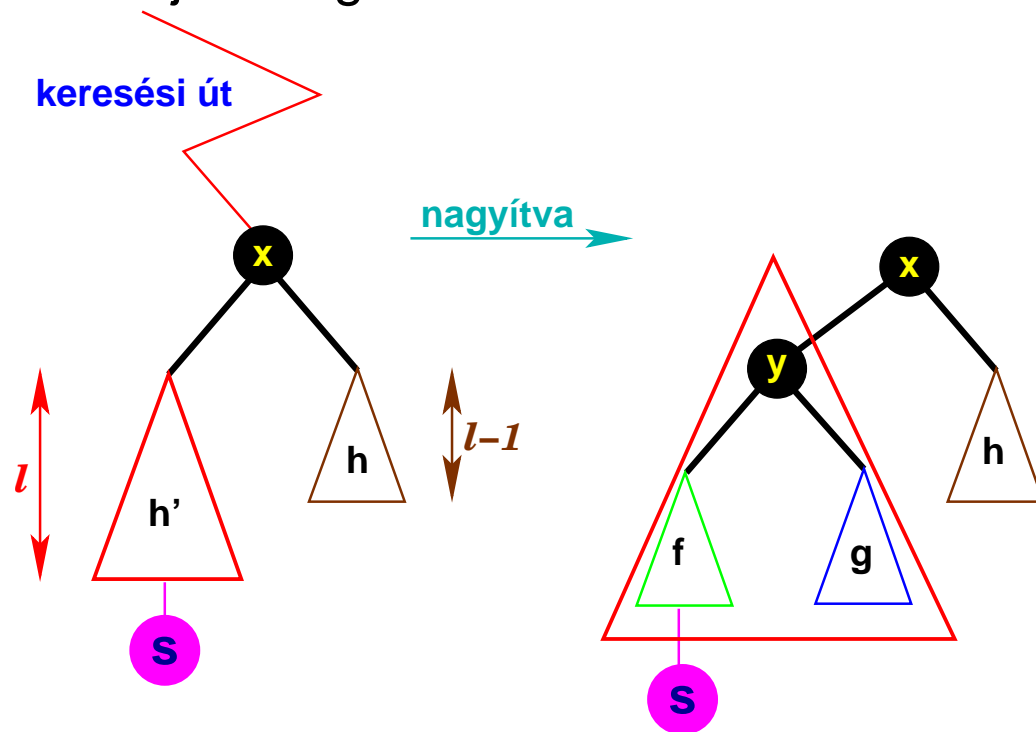
Tétel. Legyen S egy n csúcsból álló AVL-fa. $\text{BESZÚR}(s, S)$ után legfeljebb

egy (esetleg dupla) forgatással helyreállítható az AVL-tulajdonság. A beszúrás költsége ezzel együtt is $O(\log n)$.

Bizonyítás: Minden csúcsban tartjuk számon $m(f)$ -et, az f gyökerű részfa mélységét, ez könnyen karban tartható.

KERES(s, S) segítségével megkeressük s helyét.

A keresési utat végigjárva megkeressük a legalsó olyan csúcsot, ahol sérül az AVL-tulajdonság $\implies x$



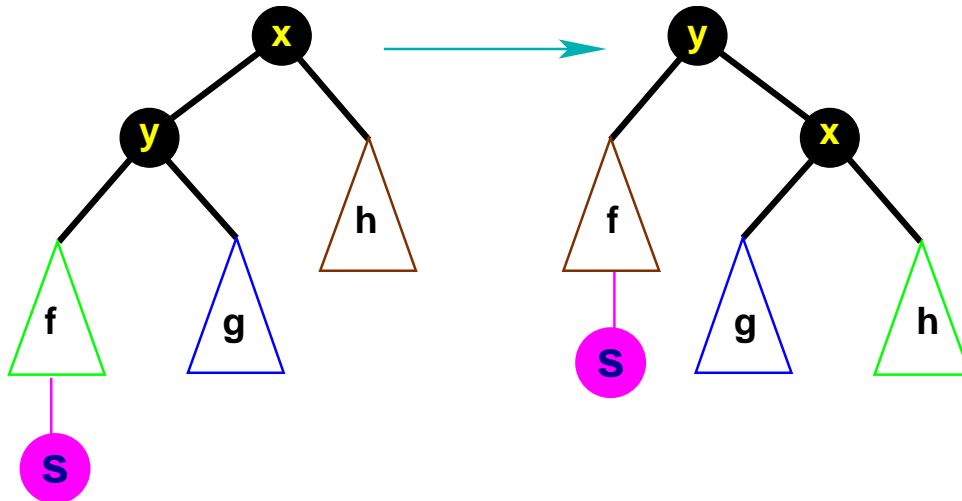
x definíciója $\implies m(h') \neq m(h)$

feltehetjük, hogy $m(h') = l, m(h) = l - 1$

Két eset van:

a) s az f részfába kerül

b) s a g részfába kerül



a) eset:

$m(f) < m(g) \implies x$ -ben nem sérül az AVL-tul.

$m(f) > m(g) \implies y$ -ben is sérül az AVL-tul.

\implies

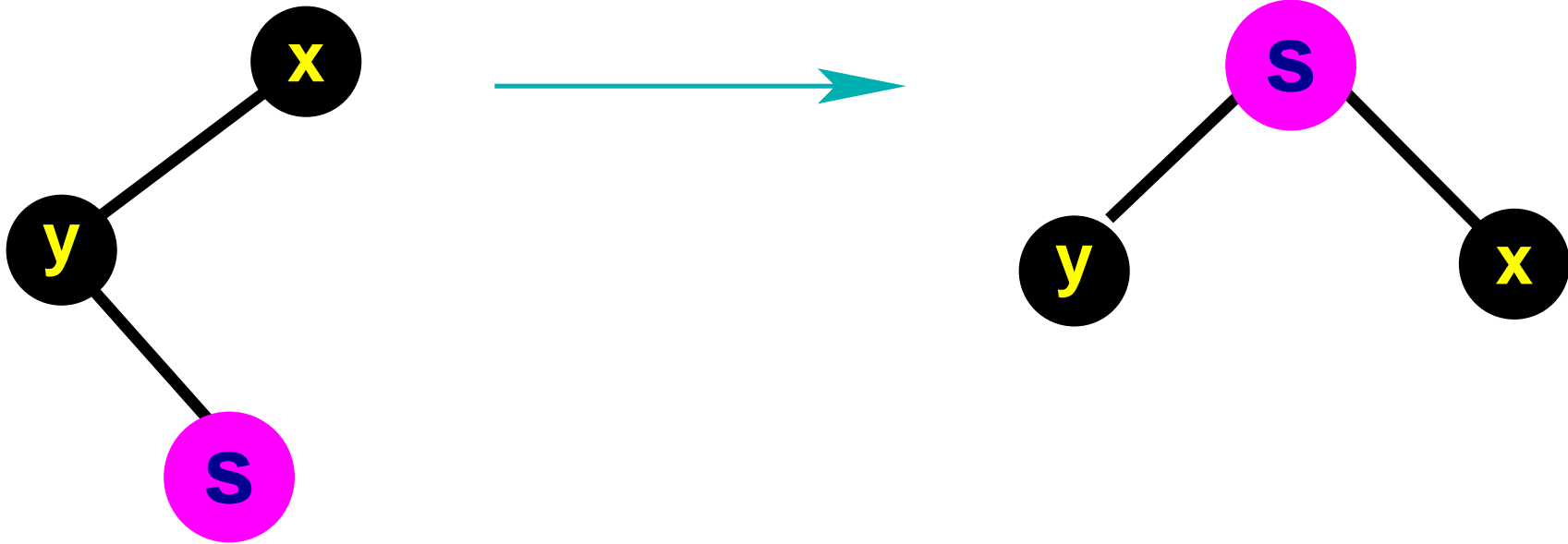
$m(f) = m(g) = m(h) =$
 $= m(y) - 1 = l - 1$

\implies jobb forgatás helyre állítja az AVL-tul.

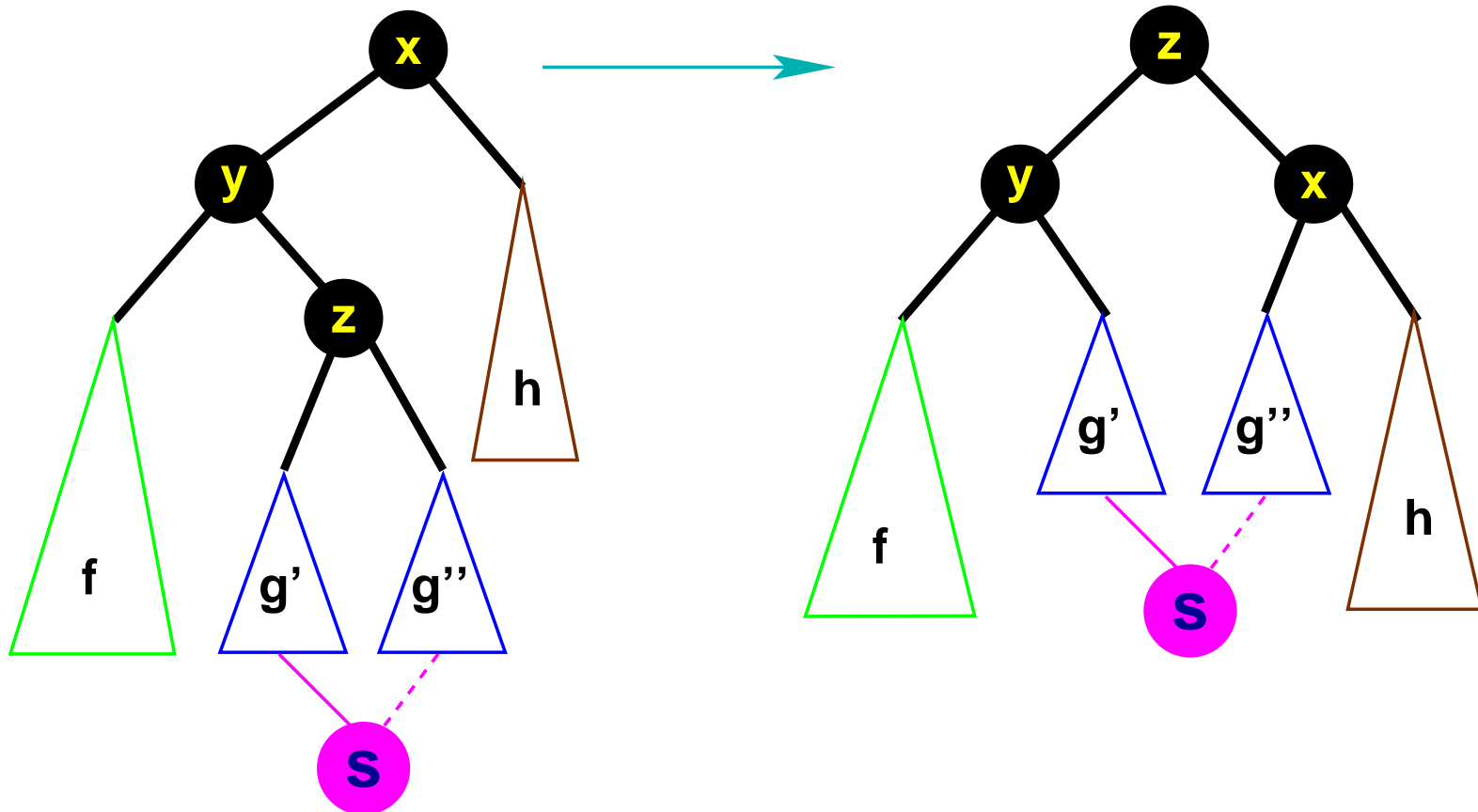
A forgatás után y mindkét részfájának a magassága l lesz, x új részfái g és h , mindkettő szintszáma $l - 1$.

y feletti csúcsok magassága nem változik, így az AVL-feltétel feljebb is megmarad a kereső út mentén.

b1) eset: s a g részfába került és s az y csúcs fia ($l = 1$)
 \implies dupla forgatás



b2) eset: s a g rész fába került és s nem az y csúcs fia ($l > 1$)



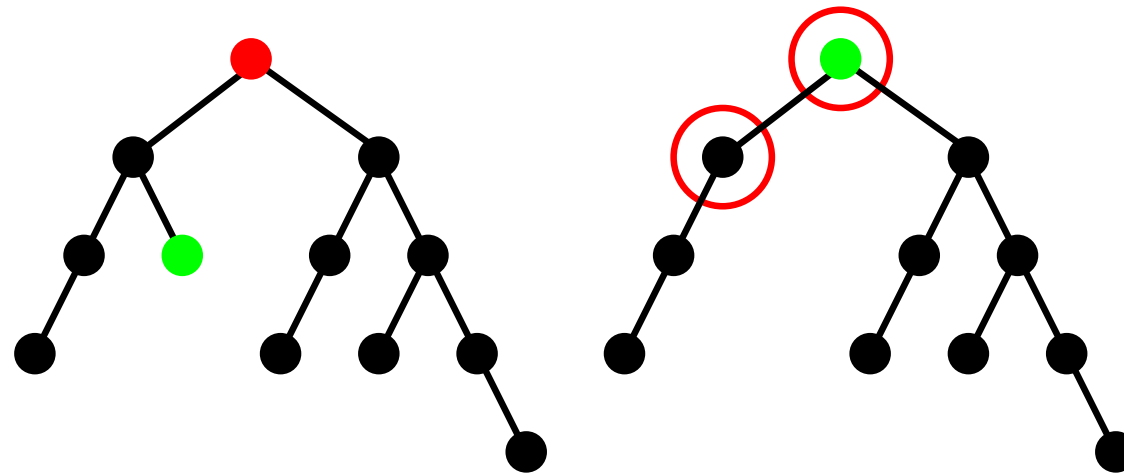
$\implies m(f) = l - 1$ (mert y -ban az AVL-tul. teljesül), és
 $m(g') = m(g'') = l - 2$ (mert z -ben sincs baj az AVL-tulajdonsággal)
 \implies dupla forgatás elég

Költség: $1.44 \log_2(n + 1) = O(\log n)$

Törlés AVL-fából

Tétel. Az n pontú AVL-fából való naiv törlés után legfeljebb $1.44 \log_2 n$ (szimpla vagy dupla) forgatás helyreállítja az AVL-tulajdonságot.

Nem bizonyítjuk, a módszer hasonló, de nem mindig elég egy forgatás. Pl.:



Java animáció: [AVL-fa](#)

További kiegyensúlyozott fák

Az AVL-tulajdonság csak egy a lehetséges kiegyensúlyozottsági feltételek közül. Általánosítása:

Definíció. **HB[k]-fák:** (C. C. Foster, 1973)

Legyen $k \geq 1$ egy egész szám. Egy bináris keresőfa $HB[k]$ -fa, ha minden x csúcsára teljesül, hogy

$$|m(\text{bal}(x)) - m(\text{jobb}(x))| \leq k.$$

$\implies HB[1]$ -fák \rightarrow AVL-fák

Súlyra kiegyensúlyozott fák

A részfák súlya legyen a csúcsszámuk: $s(f)$.

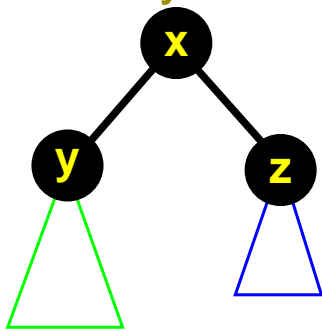
Definíció. Egy bináris keresőfát súlyra kiegyensúlyozott fának (röviden *SK-fának*) nevezünk, ha minden x belső csúcsára teljesül, hogy

$$\sqrt{2} - 1 < \frac{s(\text{bal}(x))}{s(\text{jobb}(x))} < \sqrt{2} + 1.$$

Feladat: Igazoljuk, hogy a leheletnyivel szigorúbb $1/2 < \frac{s(\text{bal}(x))}{s(\text{jobb}(x))} < 2$ korlátokat már csak az l szintből álló $2^l - 1$ pontú bináris fák a teljesítik.

Tétel. Egy n csúcsú *SK-fa* mélysége $\leq 2 \log_2 n + 1$.

Bizonyítás:



$$s(x) > s(y) + s(z) > s(y) + (\sqrt{2} - 1)s(y) = \sqrt{2}s(y)$$

Legyenek x_1, x_2, \dots, x_k egy k -hosszúságú gyökértől levélig menő út csúcsai.

$$n = s(x_1) > \sqrt{2}s(x_2) > (\sqrt{2})^2 s(x_3) > \dots > (\sqrt{2})^{k-1} s(x_k) = (\sqrt{2})^{k-1} = 2^{(k-1)/2}.$$

$\implies \checkmark$

S-fák

Splay-tree: D. D. Sleator és R. E. Tarjan, 1983.

Olyan bináris fa, ami **tanul**: pl. gyakran keresett elemet feljebb teszi.

⇒ Hosszú átlagos műveletsor alatt az egy lépésre eső költség optimális lesz.

Fő ötlet:

KIFORDÍT(x, f) átszervezi az f S-fát úgy, hogy x lesz az új gyökér, ha $x \in f$; különben f gyökere x valamelyik szomszédja lesz:

vagy $\max\{y \in f; y < x\}$ vagy $\min\{y \in f; y > x\}$.

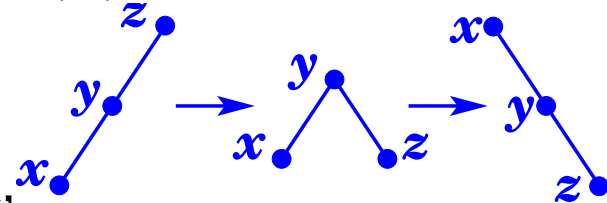
KIFORDÍT(x, f) implementációja: KERES(x, f) ⇒ ha $x \in f$, akkor majd x -et visszük fel, ha $x \notin f$, akkor a keresés x egyik szomszédjánál ($\max\{y \in f; y < x\}$ vagy $\min\{y \in f; y > x\}$) ér véget, vegyük ezt.
⇒ feltehetjük, hogy $x \in f$.

A következő eljárás x -et maximum két szinttel viszi feljebb, addig alkalmazzuk, amíg x felér.

(0) Ha x gyökér, akkor készen vagyunk.

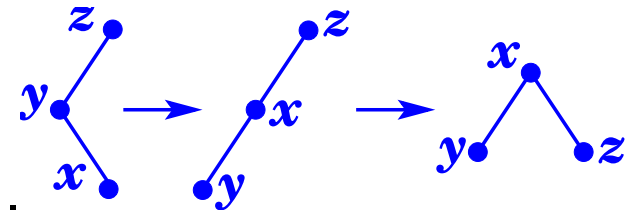
(* A továbbiakban jelölje y az x apját. *)

(1) Ha x -nek nincs nagyapja, akkor FORGAT(x), különben



(2) ha x és y is baloldali (jobboldali) gyerekek, akkor FORGAT(y), majd FORGAT(x), különben

(3) ha x és y különböző oldali gyerekek,



akkor FORGAT(x), majd ismét FORGAT(x).

Műveletek az S-fákban

A keresőfákra jellemző KERES(x, f), BESZÚR(x, f) és TÖRÖL(x, f) műveletek a szokásosak.

A RAGASZT(f, f') művelet az f és f' S-fákból egyetlen S-fát szervez, feltéve, hogy $x < y$ teljesül minden $x \in f$ és $y \in f'$ kulcsra.

A VÁG(x, f) művelet szétvágja f -et az f' és f'' S-fákra úgy, hogy $y \leq x \leq z$ teljesül minden $y \in f'$ és $z \in f''$ csúcsra.

Tétel. Az ismertetett műveletek mindegyike megvalósítható konstans számú KIFORDÍT és konstans számú elemi operáció (összehasonlítás, mutató beállítás, stb.) segítségével.

Bizonyítás: Csak két művelet, a RAGASZT és a TÖRÖL esetét nézzük meg közelebbről, többi trivi.

RAGASZT(f, f') \implies először KIFORDÍT($+\infty, f$) =: f^* \implies f^* gyökere x
 az új fa legnagyobb kulcsa
 \implies csatoljuk f' -t x jobb fiának

TÖRÖL(x, f) \implies KIFORDÍT(x, f) ha a kapott fa gyökere nem x , akkor
 $x \notin f \implies \checkmark$

Ha x az új fa gyökere, töröljük és a kapott két f_1 és f_2 részfára
RAGASZT(f_1, f_2)

Tétel. Egy üres S -fából induló olyan m műveletből álló sorozat költsége, melyben n beszúrás van, $O(m \log n)$.

Java animáció: S -fa