

# Algoritmusok és adatszerkezetek II.

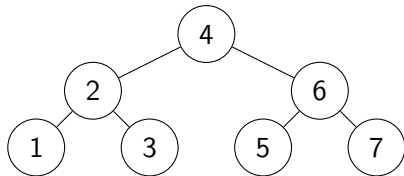
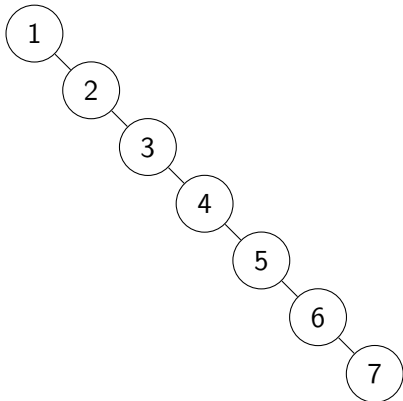
## Kiegyensúlyozott keresőfák

Szegedi Tudományegyetem

# Mit értünk kiegyensúlyozott keresőfa alatt?

## Emlékeztető

Az eddig tárgyalt műveletek  $h$  magas fákra  $O(h)$  idejűek voltak  
A továbbiakban szeretnénk, ha  $h = \Theta(\log n)$  is teljesülne



# Véletlen építésű bináris keresőfák (CLRS 12.4)

- $n$  elemű bináris fa **legrosszabb esetben**  $\Theta(n)$  magas is lehet
- $n$  növekedésével a legrosszabb eset bekövetkezése azonban egyre valószínűtlenebb
- Ha egy bináris keresőfa előállítása során csak beszúrás műveleteket alkalmazunk, úgy igazolható a következő

## Tétel

*Egy  $n$  különböző kulcsot tartalmazó véletlen építésű bináris keresőfa várható magassága  $O(\log n)$ .*



# AVL fák (Adelson-Velsky, Landis, 1962)

- **Tetszőleges** műveletsorozat végrehajtása után **legrosszabb esetben is**  $\Theta(\log n)$  magasságú kiegyensúlyozott keresőfa
- Garancia: semelyik csúcs egyensúlyi faktorának abszolút értéke nem lehet nagyobb 1-nél

## Definíció

Egy  $p$  csúcs **egyensúlyi faktora** fiai magasságának különbsége.

## Definíció

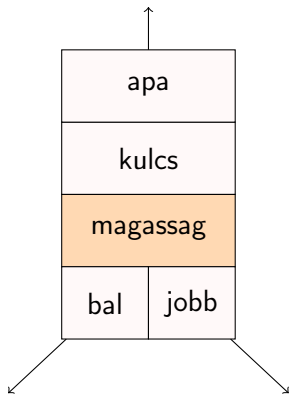
Üres fa magassága:  $h(\text{nil}) = 0$

$p$  gyökerű fa magassága:  $h(p) = \max(h(p.\text{bal}), h(p.\text{jobb})) + 1$



## AVL fa implementációja

```
class Node {  
    Object kulcs;  
    int magassag;  
    Node *apa;  
    Node *bal;  
    Node *jobb;  
}
```

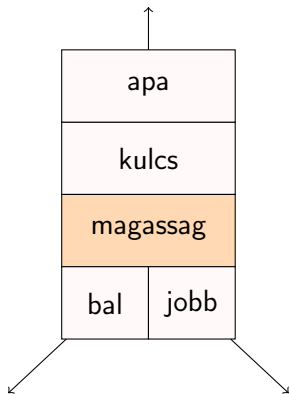


# AVL fa implementációja

```
class Node {  
    Object kulcs;  
    int magassag;  
    Node *apa;  
    Node *bal;  
    Node *jobb;  
}
```

## Megjegyzés

Találkozni olyan implementációval is, ahol a magasság helyett az egyensúlyi faktort tárolják



Legalább hány kulcsból áll egy  $h$  magas AVL fa?

magasság	$m$
1	1
2	2
3	$1+2+1=4$
4	$2+4+1=7$
5	$4+7+1=12$
6	$7+12+1=20$
7	$12+20+1=33$
8	$20+33+1=54$
$\vdots$	$\vdots$



Legalább hány kulcsból áll egy  $h$  magas AVL fa?

magasság	$m$
1	1
2	2
3	$1+2+1=4$
4	$2+4+1=7$
5	$4+7+1=12$
6	$7+12+1=20$
7	$12+20+1=33$
8	$20+33+1=54$
$\vdots$	$\vdots$
$h$	$m_{h-2} + m_{h-1} + 1$





# Miért igaz, hogy az AVL fák $O(\log n)$ magasak?

- Jelölje  $m_h$  a  $h$  magas AVL fában lévő minimálisan található kulcsok számát ( $m_1 = 1, m_2 = 2$ )
- Általánosságban ( $h > 2$  esetén):  $m_h = m_{h-2} + m_{h-1} + 1$
- $m_{h-2} < m_{h-1}$  természetesen teljesül, ahonnan

$$m_h > 2m_{h-2}$$



# Miért igaz, hogy az AVL fák $O(\log n)$ magasak?

- Jelölje  $m_h$  a  $h$  magas AVL fában lévő minimálisan található kulcsok számát ( $m_1 = 1, m_2 = 2$ )
- Általánosságban ( $h > 2$  esetén):  $m_h = m_{h-2} + m_{h-1} + 1$
- $m_{h-2} < m_{h-1}$  természetesen teljesül, ahonnan

$$m_h > 2m_{h-2} > 2 * 2m_{h-4} > \dots > 2^i m_{h-2i}$$



# Miért igaz, hogy az AVL fák $O(\log n)$ magasak?

- Jelölje  $m_h$  a  $h$  magas AVL fában lévő minimálisan található kulcsok számát ( $m_1 = 1, m_2 = 2$ )
- Általánosságban ( $h > 2$  esetén):  $m_h = m_{h-2} + m_{h-1} + 1$
- $m_{h-2} < m_{h-1}$  természetesen teljesül, ahonnan

$$m_h > 2m_{h-2} > 2 * 2m_{h-4} > \dots > 2^i m_{h-2i}$$

- $m_h > 2^i m_{h-2i}$  összefüggést  $m_1$ -ig kijátszva  $m_h > 2^{h/2}$



# Miért igaz, hogy az AVL fák $O(\log n)$ magasak?

- Jelölje  $m_h$  a  $h$  magas AVL fában lévő minimálisan található kulcsok számát ( $m_1 = 1, m_2 = 2$ )
- Általánosságban ( $h > 2$  esetén):  $m_h = m_{h-2} + m_{h-1} + 1$
- $m_{h-2} < m_{h-1}$  természetesen teljesül, ahonnan

$$m_h > 2m_{h-2} > 2 * 2m_{h-4} > \dots > 2^i m_{h-2i}$$

- $m_h > 2^i m_{h-2i}$  összefüggést  $m_1$ -ig kijátszva  $m_h > 2^{h/2}$
- Tegyük fel, hogy egy  $h$  magas AVL fa  $n \geq m_h$  csúcsból áll, azaz  $n > 2^{h/2}$ , vagyis  $h < 2 \log_2(n)$



# Miért igaz, hogy az AVL fák $O(\log n)$ magasak?

- Jelölje  $m_h$  a  $h$  magas AVL fában lévő minimálisan található kulcsok számát ( $m_1 = 1, m_2 = 2$ )
- Általánosságban ( $h > 2$  esetén):  $m_h = m_{h-2} + m_{h-1} + 1$
- $m_{h-2} < m_{h-1}$  természetesen teljesül, ahonnan

$$m_h > 2m_{h-2} > 2 * 2m_{h-4} > \dots > 2^i m_{h-2i}$$

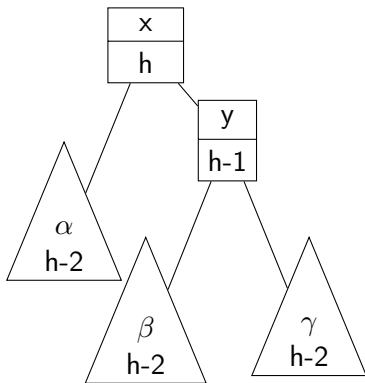
- $m_h > 2^i m_{h-2i}$  összefüggést  $m_1$ -ig kijátszva  $m_h > 2^{h/2}$
- Tegyük fel, hogy egy  $h$  magas AVL fa  $n \geq m_h$  csúcsból áll, azaz  $n > 2^{h/2}$ , vagyis  $h < 2 \log_2(n)$

## Megjegyzés

Az élesebb  $h < 1.44 \log_2(n)$  korlát is bizonyítható.



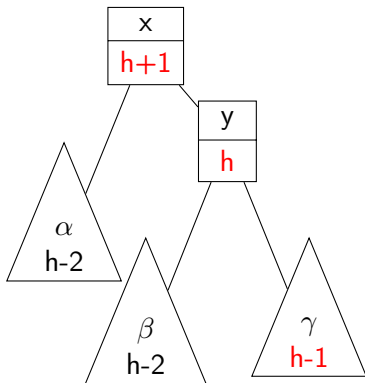
## AVL fák kiegyensúlyozottságának fenntartása forgatásokkal



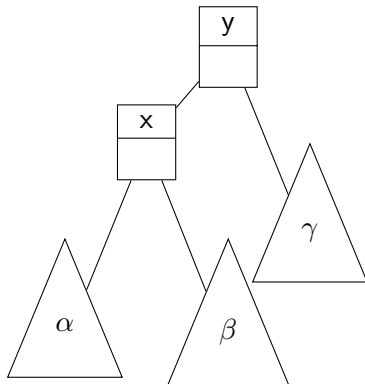
(a) beszúrás előtt



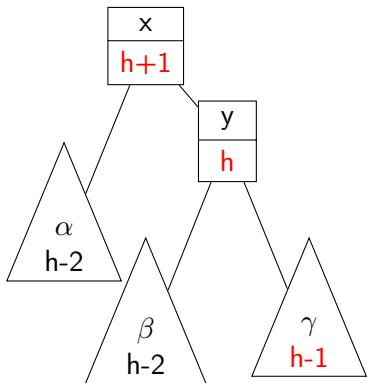
## AVL fák kiegyensúlyozottságának fenntartása forgatásokkal



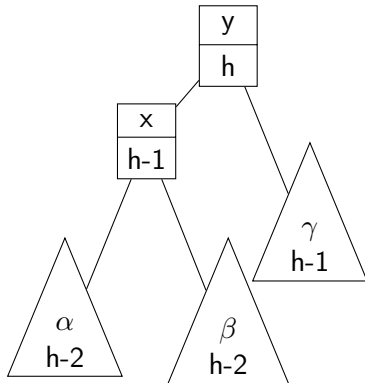
(a) beszúrás után

(b)  $x$  körüli balra forgatva

## AVL fák kiegyensúlyozottságának fenntartása forgatásokkal

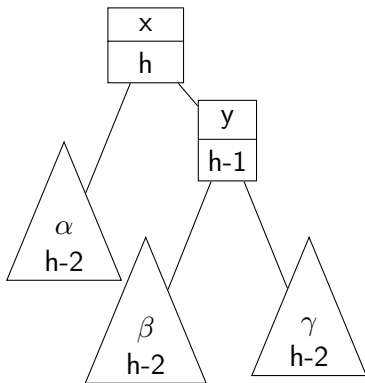


(a) beszúrás után

(b)  $x$  körüli balra forgatva



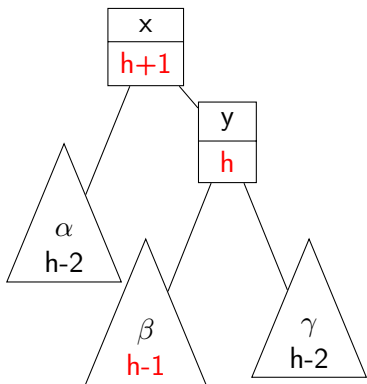
## Amikor egy forgatás nem elég



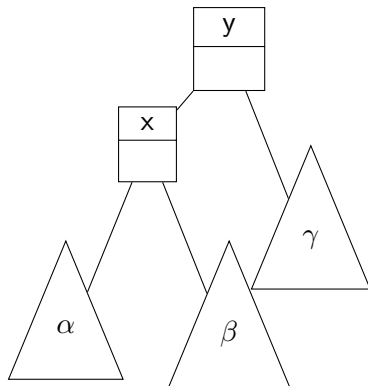
(a) beszúrás előtt



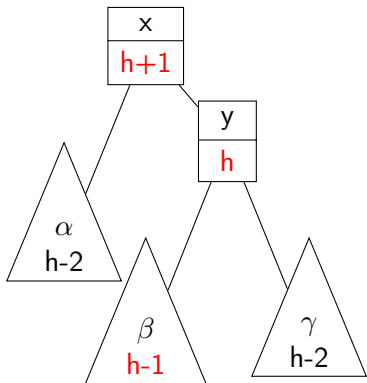
## Amikor egy forgatás nem elég



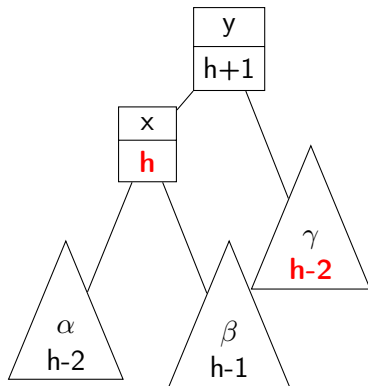
(a) beszúrás után

(b)  $x$  körüli balra forgatva

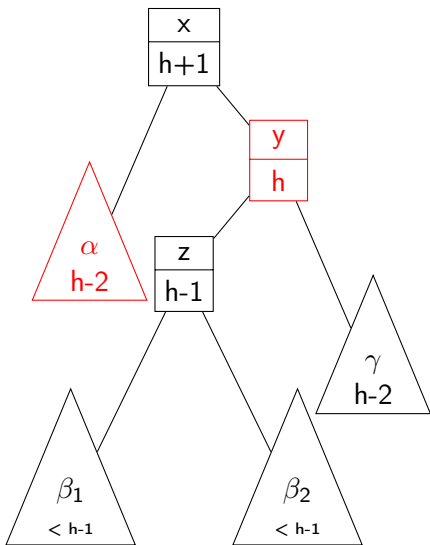
## Amikor egy forgatás nem elég



(a) beszúrás után

(b)  $x$  körüli balra forgatva

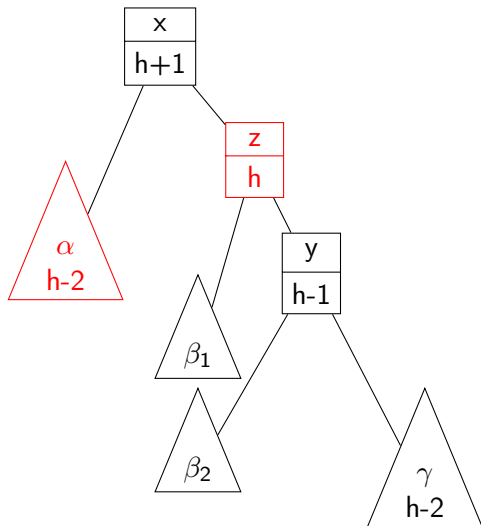
## Amikor egy forgatás nem elég – segédforgatás



## Megoldás

$y$  körül jobbra forgatunk, majd  $x$  körül balra

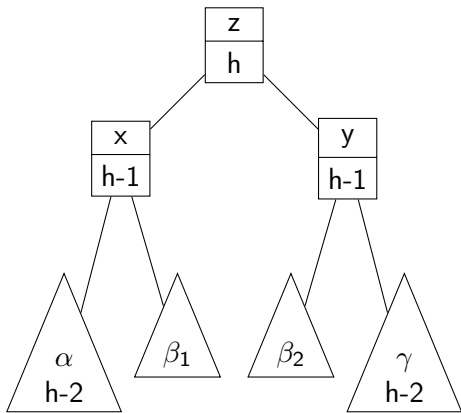
## Amikor egy forgatás nem elég – segédforgatás



Észrevétel

 $\beta_2$  nem lehetett magasabb  $\gamma$ -nál

## Amikor egy forgatás nem elég – segédforgatás



Miért kellett kettőt forgassunk?

Mert (kiinduláskor)  $x$  **jobboldali** részfája volt magasabb  
És mert ennek a részfának már a **baloldali** részfája volt magasabb  
(zikk-zakk)



# Megjegyzések a helyreállításokhoz

- A tárgyalt esetek tükörképei is előfordulhatnak
- A törlés hatására elromló AVL-fát azonos módon állítjuk helyre

# Általános keresőfák

- Az általános keresőfát az különbözteti meg a bináris keresőfától, hogy egy csúcs több kulcsot is tartalmazhat
- Keresőfa tulajdonság kiterjesztése
  - A csúcsban található kulcsok  $<$  szerint rendezettek
  - A tárolt kulcsok értékei meghatározzák a kulcsértékeknek azon tartományait, amelyekbe a részfák kulcsai eshetnek

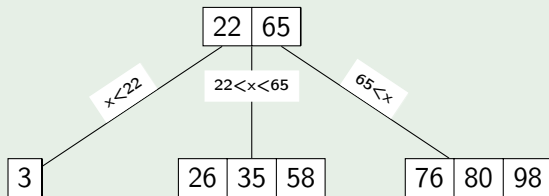




# Általános keresőfák

- Az általános keresőfát az különbözteti meg a bináris keresőfától, hogy egy csúcs több kulcsot is tartalmazhat
- Keresőfa tulajdonság kiterjesztése
  - A csúcsban található kulcsok  $<$  szerint rendezettek
  - A tárolt kulcsok értékei meghatározzák a kulcsértékeknek azon tartományait, amelyekbe a részfák kulcsai eshetnek

## Példa



# B-fa definíciója

## Definíció

A  $t$ -rendű B-fa olyan általános keresőfa, amelyre teljesül, hogy:

- Minden gyökértől különböző  $p$  csúcsára  $t \leq Rang(p) \leq 2t$   
(rang alatt a fapontban tárolt kulcsok számát értjük)
- $r$  gyökerének rangjára pedig  $1 \leq Rang(r) \leq 2t$
- Minden nemlevél  $p$  csúcsra és  $1 \leq i \leq Rang(p) + 1$  esetén  $p.gyerekek[i] \neq Nil$
- Minden levél azonos mélységű (ez az érték a fa  $h$  magassága).



# B-fa implementációja

```
class Node {  
    Object[] kulcsok;  
    int meret;  
    Node *apa;  
    Node *gyerekek [meret+1];  
    boolean level;  
}
```



# B-fa implementációja

```
class Node {  
    Object [] kulcsok ;  
    int meret ;  
    Node *apa ;  
    Node *gyerekek [meret + 1] ;  
    boolean level ;  
}
```

## Fontos!

Az eddigiektől eltérően egy csúcsban több kulcs is található.  
A csúcson belüli kulcsokra érvényesül a  $<$  rendezés.  
A csúcsookról eltároljuk, hogy levelek-e (level változó)



## B-fában keresés

```
B-FÁBANKERES(x, k) {  
    i=0  
    while i < meret és k > x.kulcsok[i] {  
        i = i+1  
    }  
  
    if (i < meret és k = x.kulcsok[i]) {  
        return (x,i) // az x csúcs i-edik kulcsát kerestük  
    }  
    if (x.level) {  
        return nil // a B-fa nem tartalmazza k-t  
    } else {  
        // a megfelelő ágban keresünk tovább  
        return B-FÁBANKERES(x.gyerekek[i], k)  
    }  
}
```



# B-fák kiegyensúlyozottsága

- B fákat olyan esetekben szokás használni, amikor az adatunk nem fér be a főmemóriába, azt a háttértáron tároljuk
- A B-fák kiegyensúlyozottsága abból fakad, hogy minden levél azonos mélységen található, illetve, hogy minden (nemgyökér) csúcs legalább  $t + 1$  elágazással rendelkezik
  - $t$ -rendű B-fa  $i$ -edik ( $i > 1$ ) szintjén legalább  $2(t + 1)^{i-2}$  csúcs és  $2t(t + 1)^{i-2}$  érték található
  - $t$  értéke a valóságban nagy, mi ezen a kurzuson 2-nek vesszük (hacsak mást nem mondunk)
- Az AVL-fánál "kiegyensúlyozottabb", magassága  $O(\log_t n)$ 
  - Aszimptotikusan nincs jelentősége a logaritmus alapjának, viszont ha másodlagos memóriából olvasunk, számíthat



# B-fák tulajdonságainak fenntartása – méret túlcserélődése

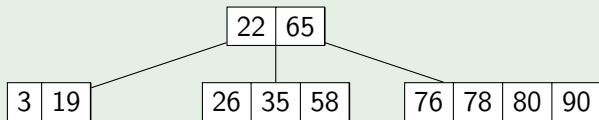
- B-fákba is levélként szúrunk be
- $2t$  méretű csúcsba beszúrva,  $2t + 1$  méretű csúcsot kapunk



# B-fák tulajdonságainak fenntartása – méret túlcserélődése

- B-fákba is levélként szúrunk be
- $2t$  méretű csúcsba beszúrva,  $2t + 1$  méretű csúcsot kapunk a "középső" elem mentén kettévágva éppen  $2t$  méretű csúcsunk lesz (a középső elemet küldjük föl az ősbé)
- Szükség szerint ismételjük az előző lépést

## Példa



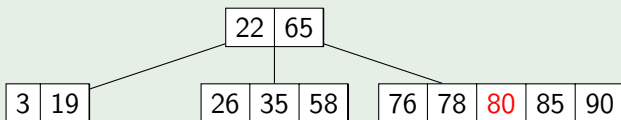


## B-fák tulajdonságainak fenntartása – méret túlcsoordulása

- B-fákba is levélként szúrunk be
- $2t$  méretű csúcsba beszúrva,  $2t + 1$  méretű csúcsot kapunk a "középső" elem mentén kettévágva éppen  $2t$  méretű csúcsunk lesz (a középső elemet küldjük föl az ősbé)
- Szükség szerint ismételjük az előző lépést

## Példa

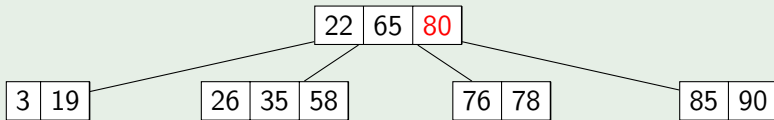
Beszúr(85)



# B-fák tulajdonságainak fenntartása – méret túlcserélődése

- B-fákba is levélként szúrunk be
- $2t$  méretű csúcsba beszúrva,  $2t + 1$  méretű csúcsot kapunk a "középső" elem mentén kettévágva éppen  $2t$  méretű csúcsunk lesz (a középső elemet küldjük föl az ősbé)
- Szükség szerint ismételjük az előző lépést

## Példa



# B-fák tulajdonságainak fenntartása – méret alulcsordulása

- Kulcs törlése  $t$  méretű csúcsból  $t - 1$  méretűvé teheti azt
- Itt is megelőzővel helyettesítünk
- Két eset lehetséges
  - 1 Szomszédától kölcsönzünk, ha annak van feleslege ( $> t$  méretű)
  - 2 Összeolvasztjuk a kritikusan kicsi csúcsot a  $t$  méretű szomszédjával  $\Rightarrow t + (t - 1) + 1 = 2t$  méretű csúcs jön létre
- A kölcsönvételt preferáljuk, hiszen az  $O(1)$  időben elvégezhető



# B fa variánsok

- Egyes implementációk/források (pl. CLRS könyv is)  $t$  értéket a csúcsok fokszámkorlátjára használja: ilyenkor a csúcsokban lévő kulcsok száma  $\in [t - 1, 2t - 1]$



# B fa variánsok

- Egyes implementációk/források (pl. CLRS könyv is)  $t$  értéket a csúcsok fokszámkorlátjára használja: ilyenkor a csúcsokban lévő kulcsok száma  $\in [t - 1, 2t - 1]$
- A telített csúcsok kaszkád vágását elkerülendő, bizonyos implementációk proaktívan viselkednek: már leszálló ágban (top-down) elvégzik a vágásokat (preemptive split)
  - Így garantált, hogy amennyiben egy csúcsot szétvágunk, akkor az ő őse már nem szorul további szétvágásra (tail rekurzió)



# B fa variánsok

- Egyes implementációk/források (pl. CLRS könyv is)  $t$  értéket a csúcsok fokszámkorlátjára használja: ilyenkor a csúcsokban lévő kulcsok száma  $\in [t - 1, 2t - 1]$
- A telített csúcsok kaszkád vágását elkerülendő, bizonyos implementációk proaktívan viselkednek: már leszálló ágban (top-down) elvégzik a vágásokat (preemptive split)
  - Így garantált, hogy amennyiben egy csúcsot szétvágunk, akkor az ő őse már nem szorul további szétvágásra (tail rekurzió)
- B+ fa: a kulcsokat kísérő (esetleges) adattagok csak a levelekben legyenek tárolva  $\rightarrow$  a belső pontokban csak pointerok és indexelt kulcsok  $\rightarrow$  nagyobb elágazási faktort tudunk alkalmazni  $\rightarrow$  tovább csökkenthető a fa magassága



# Tail rekurzió

- Olyan rekurzív függvény, amely legutolsó műveletként hajtja végre a rekurzív hívást
- Hatékonyabb a nem tail rekurzív megoldásnál

```
FIBOT(n, a, b) {  
    if (n==0) return a  
    if (n==1) return b  
    return Fibo(n-1, b, a+b)  
}
```

```
FIBO(n) {  
    if (n<2) return n  
    return Fibo(n-1) + Fibo(n-2)  
}
```

```
In [33]: %timeit fibo(35)  
4.82 s ± 60.7 ms per loop  
  
In [34]: %timeit fiboT(35)  
9.63 μs ± 33.7 ns per loop
```



# Összegzés

- A bináris keresőfák műveletei  $O(h)$  idejűek
- Legrosszabb esetben azonban  $n$  is lehet a fák magassága ( $\Theta(\log n)$  helyett)
- Kiegyensúlyozott keresőfák használatával garantálható, hogy a keresőfa kiegyensúlyozottsága sose romoljon el "túlságosan"

