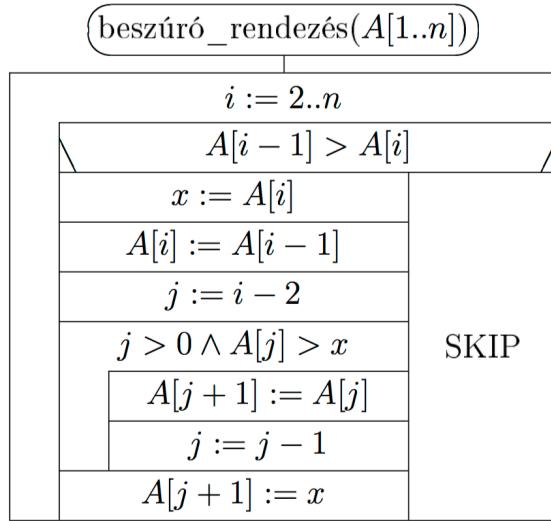


## Algo vizsga

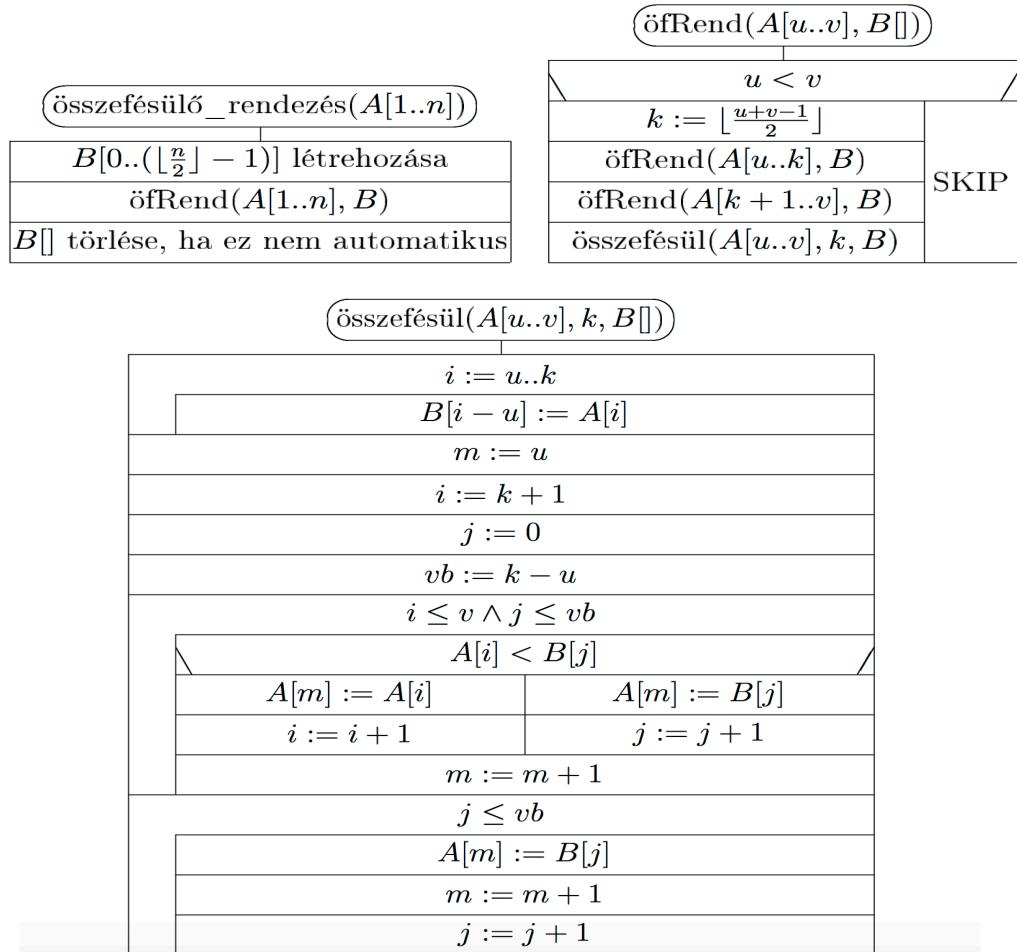
### Rendezések:

#### 1. Beszúró rendezés:

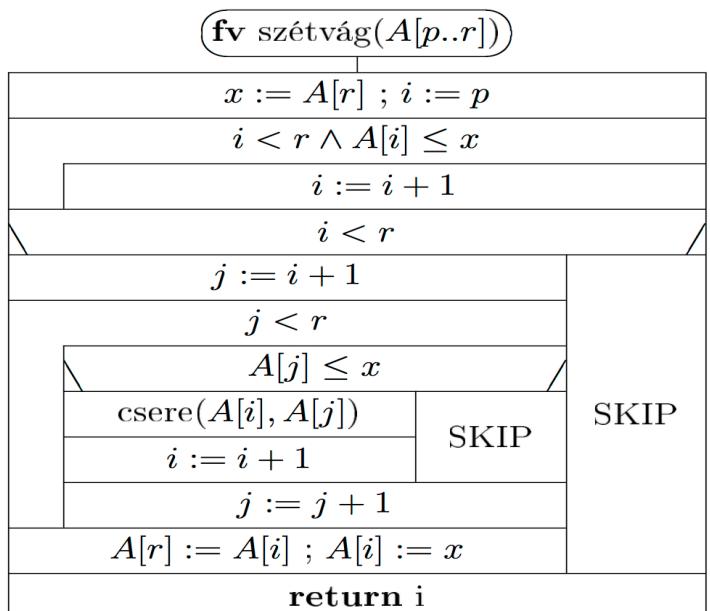
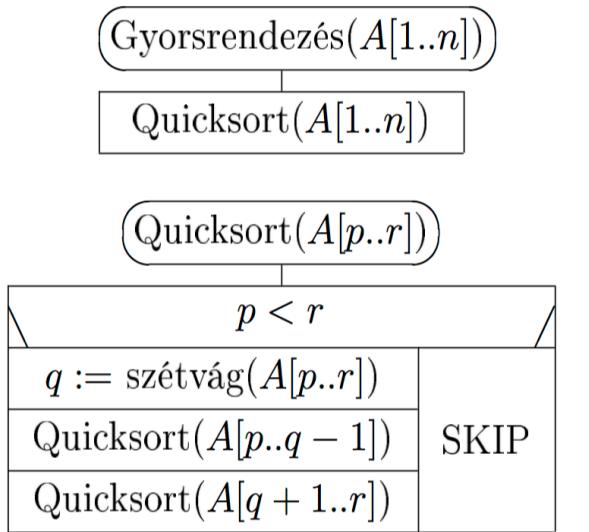


- Műveletigény: MTIS(n)  $\in \Theta(n^2)$
- Linkek: <https://www.youtube.com/watch?v=DFG-XuyPYUQ&t=205s>

#### 2. Összefésülő rendezés:

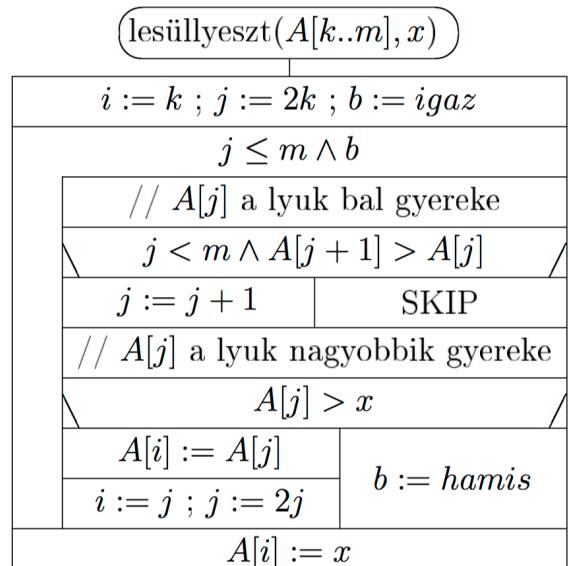
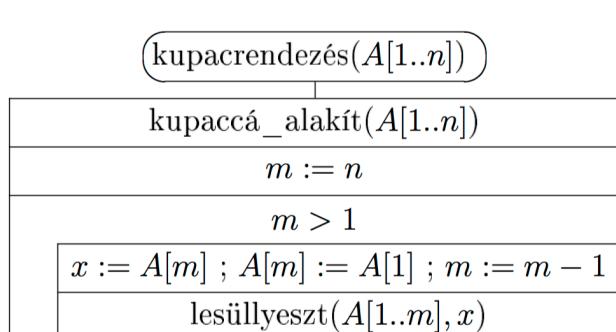


### 3. Gyorsrendezés:



- Műveletigény:  $mT(n) \in O(n \lg n)$
- Linkek: <https://www.youtube.com/watch?v=-7pzsM6gxgY>

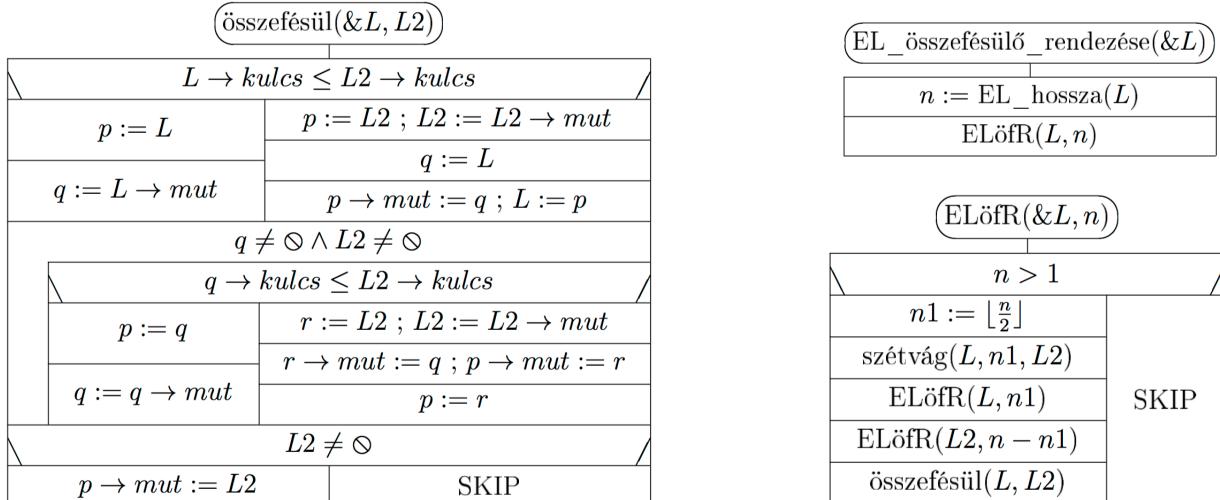
### 4. Kupacrendezés:



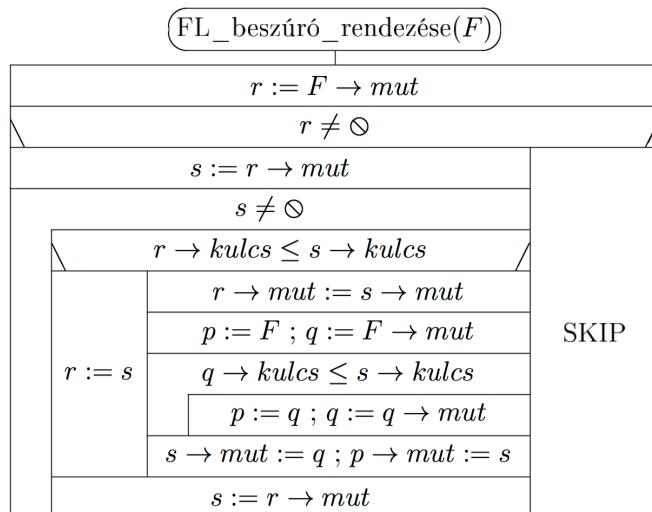
- Műveletigény:  $O(n \lg n)$

## Láncoolt Listák:

### 1. Összefűzés:



### 2. Beszúró rendezés:



## Bináris fák, Kupacok, Prior sorok:

### 1. Bináris fák:

- Szigorúan bináris fa:** Azokat a bináris fákat, amelyekben minden belső (azaz nem-levél) csúcsnak két gyereke van, szigorúan bináris fáknak nevezünk
- Teljes bináris fa:** Ha ez utóbbiaknak min- den levele azonos szinten van, teljes bináris fákról beszélünk
- Majdnem teljes:** Ha egy teljes bináris fa levélszintjéről nulla, egy vagy több levelet elveszünk, de nem az összeset, az eredményt majdnem teljes bináris fának nevezzük
- Méret szerint kiegyensúlyozott:** Egy bináris fa méret szerint kiegyensúlyozott, ha tetszőleges nemüres részfája bal és jobb részfájának mérete legfeljebb egyelőre elérhető
- Magasság szerint kiegyensúlyozott:** Egy bináris fa magasság szerint kiegyensúlyozott, ha minden csúcsa kiegyensúlyozott
- Balra tömörített:** Egy majdnem teljes bináris fa balra tömörített, ha az alsó szintjén egyetlen leveltől balra sem lehet új levelet beszúrni

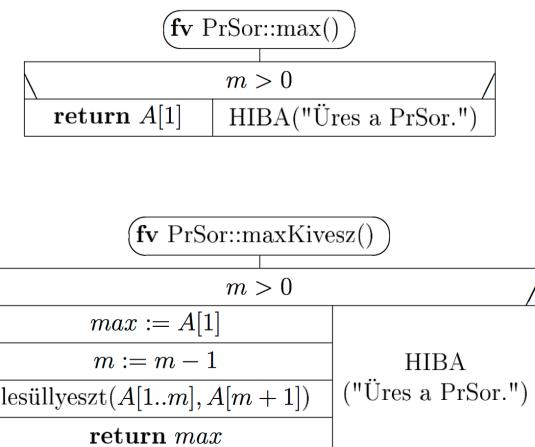
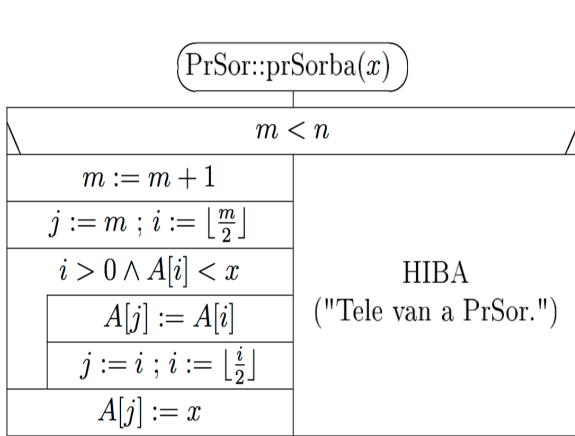
## 2. Kupacok:

- **Maximum-kupac:** Egy balra tömörített, majdnem teljes bináris fát maximum-kupacnak (heap) nevezünk, ha minden belső csúcs kulcsa nagyobb-egyenlő, mint a gyerekeié.
- **Minimum-kupac:** Ha minden belső csúcs kulcsa kisebb-egyenlő, mint a gyerekeié, minimum- kupacról beszélünk
- **Lyukas kupac:** Egy balra tömörített, majdnem teljes bináris fát lyukas kupacnak nevezünk, ha a fa egyik csúcsa a lyuk, és minden szülő-gyerek párosban a szülő kulcsa nagyobb-egyenlő, mint a gyereke kulcsa, kivéve, ha a páros egyik tagja maga a lyuk. A lyuk kulcsa nemde niált, de ha a lyuknak van szülője, akkor annak kulcsa  $\geq$  mint a lyuk leszármazottainak kulcsai
- **Csonka kupac:** Egy balra tömörített, majdnem teljes bináris fát csonka kupacnak nevezünk, ha minden szülő-gyerek párosban a szülő kulcsa nagyobb-egyenlő, mint a gyereke kulcsa, kivéve, ha a szülő a gyökérzsúcs. A gyökérzsúcs kulcsa is de niált, de lehet, hogy kisebb, mint a gyereke kulcsa.

## 3. Prior sorok:

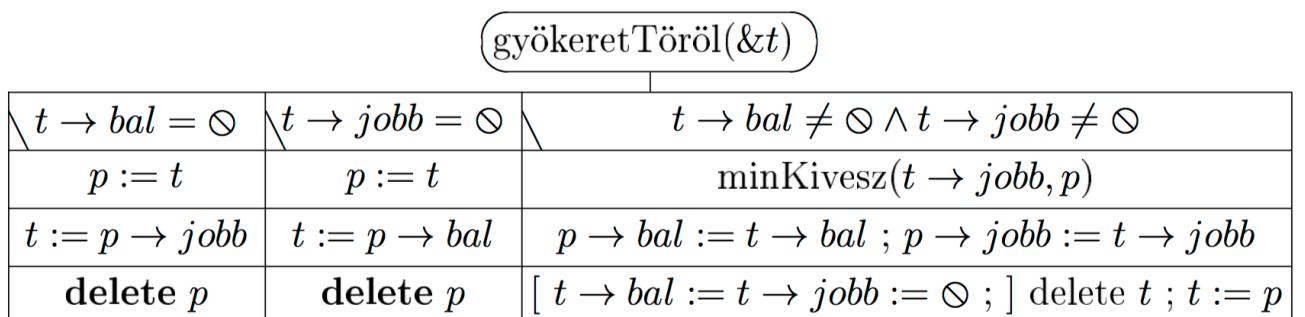
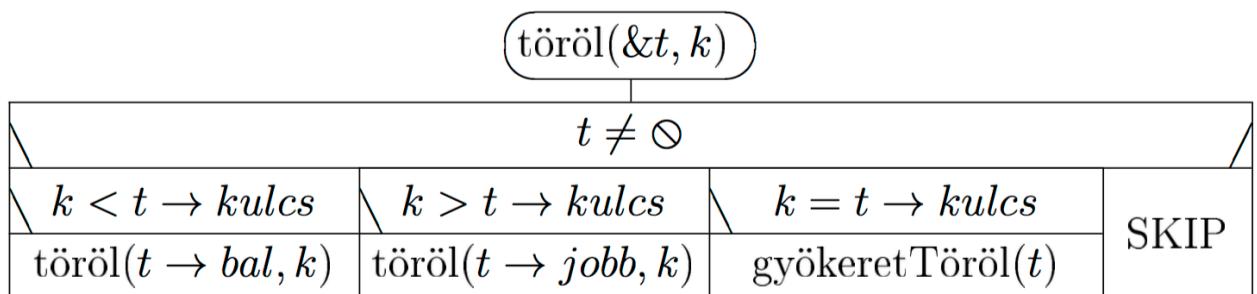
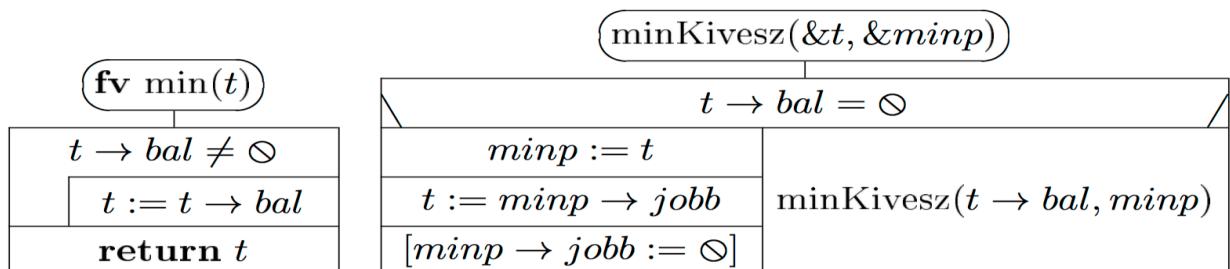
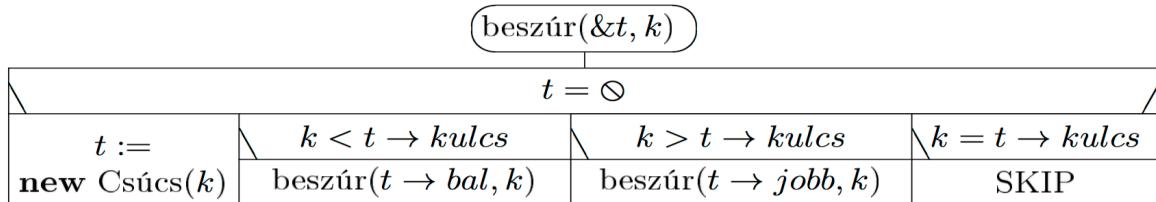
- **Műveletek:**

PrSor
- $A[1..n] : T // T$ ismert típus, $n$ globális konstans
- $m : 0..n // m$ a PrSor aktuális mérete
+ PrSor() $\{m := 0\} //$ üresre inicializálja a PrSort
// destruktort itt nem kell definiálni
+ prSorba( $x$ ) // beteszí $x$ -et a prSorba
+ fv maxKivesz() // kiveszi és visszaadja a PrSor maximális elemét
+ fv max() // megnézi és visszaadja a PrSor maximális elemét
+ fv tele_e() {return $m = n$ }
+ fv üres_e() {return $m = 0$ }



4. Bináris keresőfák:

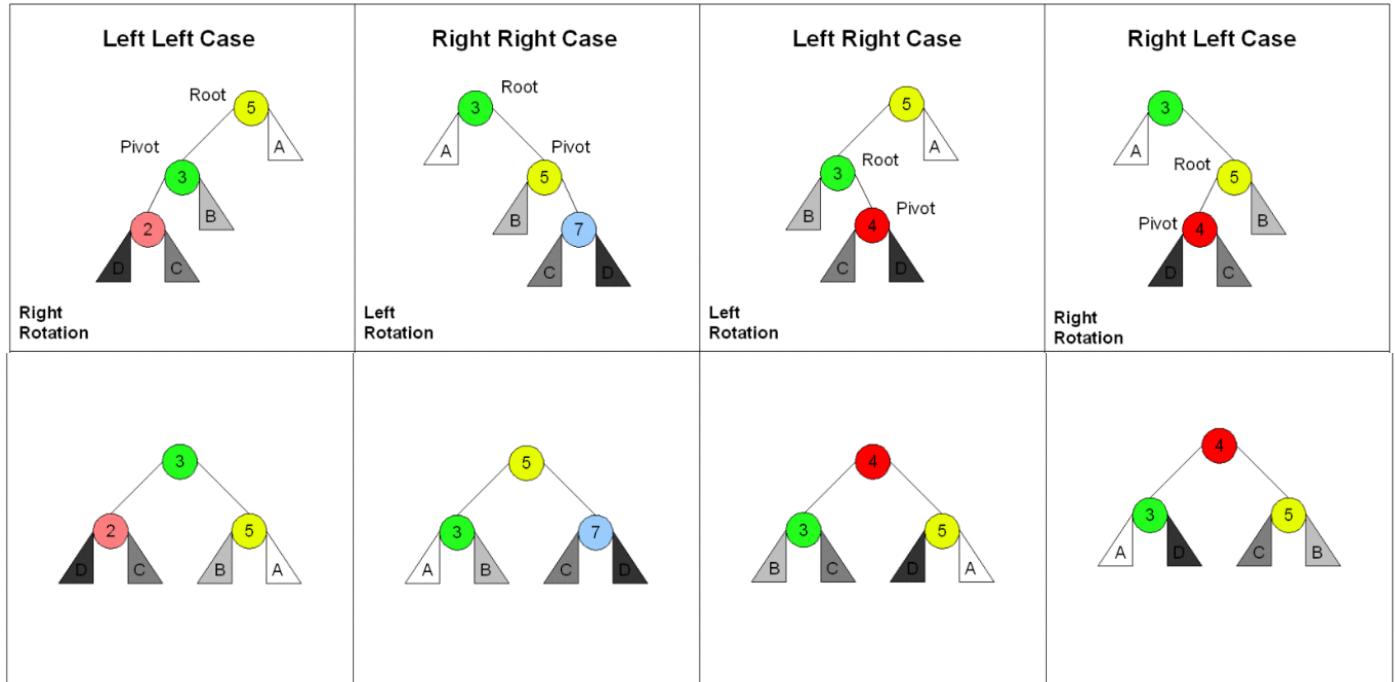
- **Fogalom:** Egy bináris fát keresőfának nevezünk, ha minden nemüres r részfájára és annak a gyökerében lévő y kulcsra igazak az alábbi követelmények:
  - Ha x egy tetszőleges csúcs kulcsa az r bal részfájából, akkor  $x < y$ .
  - Ha z egy tetszőleges csúcs kulcsa az r jobb részfájából, akkor  $z > y$ .
- **Műveletek:**



## AVL fák, B+ fák

### 1. AVL fák:

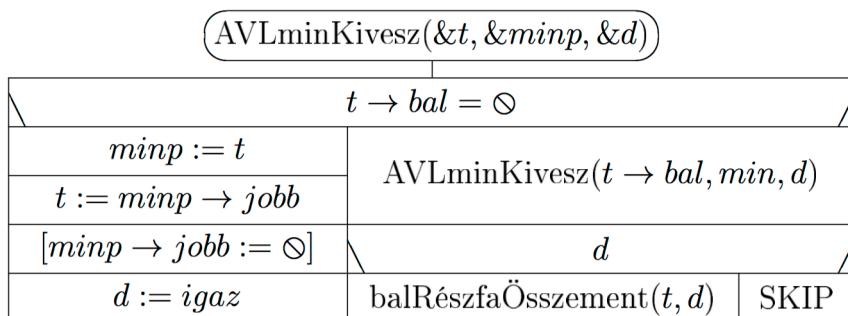
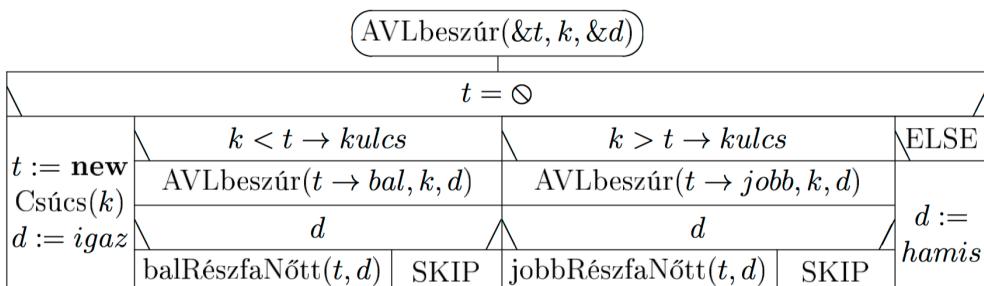
- Fogalom:** Az AVL fák magasság szerint kiegyensúlyozott bináris keresőfák.
- Magassága:**  $\lfloor \lg n \rfloor \leq h \leq 1,45 \lg n$ , azaz  $h \in \Theta(\lg n)$
- Forgatások:**



- Kiegyensúlyozás:**

- ha a bal oldali részfa magasabb 1-el
- + ha a jobb oldali magasabb 1-el
- ++ ha 2-vel magassabb a jobb
- ha 2-vel magasabb a bal
- = ha egyenlő

- Műveletek:**



$\text{AVLtöröl}(\&t, k, \&d)$			
$t \neq \infty$			
$k < t \rightarrow \text{kulcs}$	$k > t \rightarrow \text{kulcs}$	$k = t \rightarrow \text{kulcs}$	
$\text{AVLtöröl}(t \rightarrow \text{bal}, k, d)$	$\text{AVLtöröl}(t \rightarrow \text{jobb}, k, d)$		
$d$	$d$		
$\text{balRészfaÖsszement}(t, d)$	$\text{jobbRészfaÖsszement}(t, d)$	$\text{AVLgyTöröl}(t, d)$	$d := \text{hamis}$
Skip	Skip	Skip	

$\text{AVLgyTöröl}(\&t, \&d)$			
$t \rightarrow \text{bal} = \infty \wedge t \rightarrow \text{jobb} = \infty$			
$p := t$	$p := t$	$t \rightarrow \text{bal} \neq \infty \wedge t \rightarrow \text{jobb} \neq \infty$	
$t := p \rightarrow \text{jobb}$	$t := p \rightarrow \text{bal}$	$\text{jobbRészfaMinGyökérbe}(t, d)$	
$\text{delete } p$	$\text{delete } p$	$d$	
$d := \text{igaz}$	$d := \text{igaz}$	$\text{jobbRészfaÖsszement}(t, d)$	Skip

## 2. B+ fák:

- **Tulajdonságok:**

- minden levélben legfeljebb  $d-1$  kulcs, és ugyanennyi, a megfelelő (azaz ilyen kulcsú) adatrekordra hivatkozó mutató található.  
A gyökértől mindegyik levél ugyanolyan távol található. (Más szavakkal, minden levél azonos mélységben, a legalós szinten van.)
- minden belső csúcsban egyetlen több mutató van, mint kulcs, ahol  $d$  a felső határa a mutatók számára. minden Cs belső csúcsra, ahol  $k$  a Cs csúcsban a kulcsok száma: az első gyerekhez tartozó részfában minden kulcs kisebb, mint a Cs első kulcsa; az utolsó gyerekhez tartozó részfában minden kulcs nagyobb-egyenlő, mint a Cs utolsó kulcsa; és az  $i$ -edik gyerekhez tartozó részfában ( $2 \leq i \leq k$ ) lévő tetszőleges  $r$  kulcsra  $Cs.kulcs[i-1] \leq r < Cs.kulcs[i]$ .
- A gyökérCsúcsnak legalább két gyereke van (kivéve, ha ez a fa egyetlen csúcsa, következésképpen az egyetlen levele is).  
Minden, a gyökértől különböző belső csúcsnak legalább  $\text{floor}(d / 2)$  gyereke van. ( $\text{floor}(d / 2) = d/2$  alsó egész-rész.)
- minden levél legalább  $\text{floor}(d / 2)$  kulcsot tartalmaz (kivéve, ha a fának egyetlen csúcsa van).  
A B+ fa által reprezentált adathalmaz minden kulcsa megjelenik valamelyik levélben, balról jobbra szigorúan monoton növekvő sorrendben.

## Linkek:

- <http://people.inf.elte.hu/veanna/algo1/feladatgyujtemeny/>
- [http://people.inf.elte.hu/fekete/algoritmusok\\_jegyzet/](http://people.inf.elte.hu/fekete/algoritmusok_jegyzet/)
- <http://aszt.inf.elte.hu/~asvanyi/ad/>

## Bizonyítások:

### 1. AVL fa magassága:

**Tétel:** Tetszőleges  $n$  csúcsú nemüres AVL fa  $h$  magasságára:

$$\lfloor \lg n \rfloor \leq h \leq 1,45 \lg n, \quad \text{azaz} \quad h \in \Theta(\lg n)$$

**A bizonyítás vázlata:** Először a  $h$  magasságú, nemüres KBF-ek (kiegyen-súlyozott, bináris fák)  $n$  méretére adunk alsó és felső becslést. Az  $n < 2^{h+1}$  becslésből azonnal adódik  $\lfloor \lg n \rfloor \leq h$ . Másrészt meghatározzuk a  $h$  mélységű, legkisebb méretű KBF-ek csúcsainak  $f_h$  számát. Erre kapjuk, hogy  $f_0 = 1$ ,  $f_1 = 2$ ,  $f_h = 1 + f_{h-1} + f_{h-2}$  ( $h \geq 2$ ). Ezért az ilyen fákat *Fibonacci fáknak hívjuk*. Mivel tetszőleges  $h$  magasságú KBF  $n$  méretére  $n \geq f_h$ , némi matematikai ügyességgel kaphatjuk a  $h \leq 1,45 \lg n$  egyenlőtlenséget.

### 2. Összehasonlító műveletek alsókorlátja:

**Tétel.** Bármely  $R$  összehasonlításos rendező eljárás a legkedvezőtlenebb bemenő adata rendezése során nagyságrendben legalább  $(n \log n)$  összehasonlítást végez, azaz

$$MÖ_R(n) = \Omega(n \log n)$$

*Bizonyítás.* A lemmához fűzött megjegyzés – a példa alapján – eljutott a következő általános érvényű összefüggésig. Egy  $R$  összehasonlító rendező algoritmus által végzett összehasonlítások maximális száma (legrosszabb eset)  $n$  méretű input esetén megegyezik az  $R$ -hez és  $n$ -hez tartozó döntési fa magasságával. Gondoljuk meg még egyszer, hogy a legtöbb összehasonlítás egy leghosszabb végrehajtási úton történik, amelynek hossza meghatározza a fa magasságát. Formálisan is kifejezve:  $MÖ_R(n) = h(t_R(n))$ . Összevetve ezt a lemmával:

$$MÖ_R(n) \geq \log_2(n!)$$

Alakítsuk át a jobb oldalon álló kifejezést:

$$\log_2(n!) = \log_2(n(n-1)\dots 1) = \log_2 n + \log_2(n-1) + \dots + \log_2 1 = \sum_{i=1}^n \log_2 i$$

Tekintsük ezt az összeget, mint kívül írt téglalapok területének összegét, a logaritmus függvény integrál közelítő összegének és becsüljük alulról a határozott integrál értékével (lásd: 19.2. ábra).

$$\begin{aligned} \sum_{i=1}^n \log_2 i &\geq \int_1^n \log_2 x \, dx = \log_2 e \int_1^n \ln x \, dx = \\ &= \log_2 e \cdot [x \ln x - x]_1^n = \log_2 e \cdot n \ln n - \log_2 e \cdot n + \log_2 e = \\ &= n \log_2 n - \log_2 e \cdot n + \log_2 e = \Omega(n \log n) \end{aligned}$$

A következtetési láncolat elején és végét egybevetve, a bizonyítandó állítást kapjuk. ■

### 3. Programok műveletigénye:

**7.5 Definíció** Az  $O(g)$  függvényhalmaz olyan  $f$  függvényekből áll, amiket elég nagy  $n$  helyettesítési értékekre, megfelelő pozitív konstans szorzóval jól becsül felülről a  $g$  függvény:

$$O(g) = \{f : \exists d \in \mathbb{P}, \text{ hogy elég nagy } n \text{-ekre } d * g(n) \geq f(n).\}$$

**7.6 Definíció** Az  $\Omega(g)$  függvényhalmaz olyan  $f$  függvényekből áll, amiket elég nagy  $n$  helyettesítési értékekre, megfelelő pozitív konstans szorzóval jól becsül alulról a  $g$  függvény:

$$\Omega(g) = \{f : \exists c \in \mathbb{P}, \text{ hogy elég nagy } n \text{-ekre } c * g(n) \leq f(n).\}$$

**7.7 Definíció** A  $\Theta(g)$  függvényhalmaz olyan  $f$  függvényekből áll, amiket elég nagy  $n$  helyettesítési értékekre, megfelelő pozitív konstans szorzókkal alulról és felülről is jól becsül a  $g$  függvény:

$$\Theta(g) = \{f : \exists c, d \in \mathbb{P}, \text{ hogy elég nagy } n \text{-ekre} \\ . \quad c * g(n) \leq f(n) \leq d * g(n).\}$$

### 7.8 Tulajdonság

$$\Theta(g) = O(g) \cap \Omega(g)$$