① **Tétel:** Fv. aszimp. viselk., műveletigények, beszúró és összefésülő rendezésre (Insert, Merge)

Rendezés fogalma:

Rendezésnek nevezünk egy algo.-t, ha az valamilyen szempont alapján sorba állítja elemek egy listáját.

Programok műveletigénye:

- $O(g)$ fv. halmaz def:

$$O(g)=\{f: \exists c>0, \exists n_0 \geq 0, \text{hogy } \forall n \geq n_0: f(n) \leq c \cdot g(n)\}$$

Ha $f \in O(g)$, akkor g aszimptot. felső korlátja f-nek.

- $o(g)$ fv. halmaz def:

$$o(g)=\{f: \forall c>0, \exists n_0 \geq 0, \text{hogy } \forall n \geq n_0: f(n) < c \cdot g(n)\}$$

- $\Omega(g)$ fv. halmaz def:

$$\Omega(g)=\{f: \exists c>0, \exists n_0 \geq 0, \text{hogy } \forall n \geq n_0: f(n) \geq c \cdot g(n)\}$$

Ha $f \in \Omega(g)$, akkor g aszimp. alsó korlátja f-nek.

- $\omega(g)$ fv. halmaz def:

$$\omega(g)=\{f: \forall c>0, \exists n_0 \geq 0, \text{hogy } \forall n \geq n_0: f(n) > c \cdot g(n)\}$$

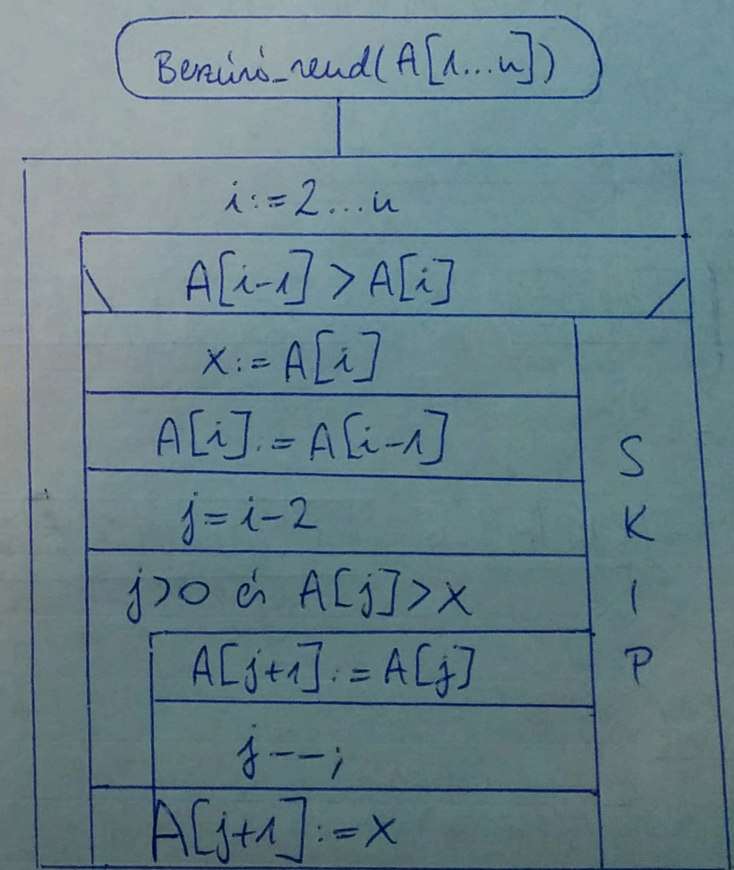
- $\Theta(g)$ fv. halmaz def:

$$\Theta(g)=\{f: \exists c_1, c_2>0, \exists n_0 \geq 0, \text{hogy } \forall n \geq n_0: c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)\}$$

Ha $f \in \Theta(g)$, akkor g aszimp. éles korlátja f-nek.

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

Beszúró rendezés (Insertation sort)



Ciklus invariáns: $2 \leq i \leq n+1$, $A[1..n]$ az input vektor egy permutáltja, mely az $(i-1)$-edik elemig mon. nő.-i rendezett

Beszúró rend műveletigénye:

$$MT(n) = n + \sum_{i=1}^{n-1} i = n + \frac{(n-1)(n-2)}{2} = \frac{n(n-1)}{2} = \Theta(n^2)$$

$mT(n) = 1 + (n-1) = \Theta(n)$ $\quad$ $AT(n) = \Theta(n^2)$, tár: $\Theta(1)$

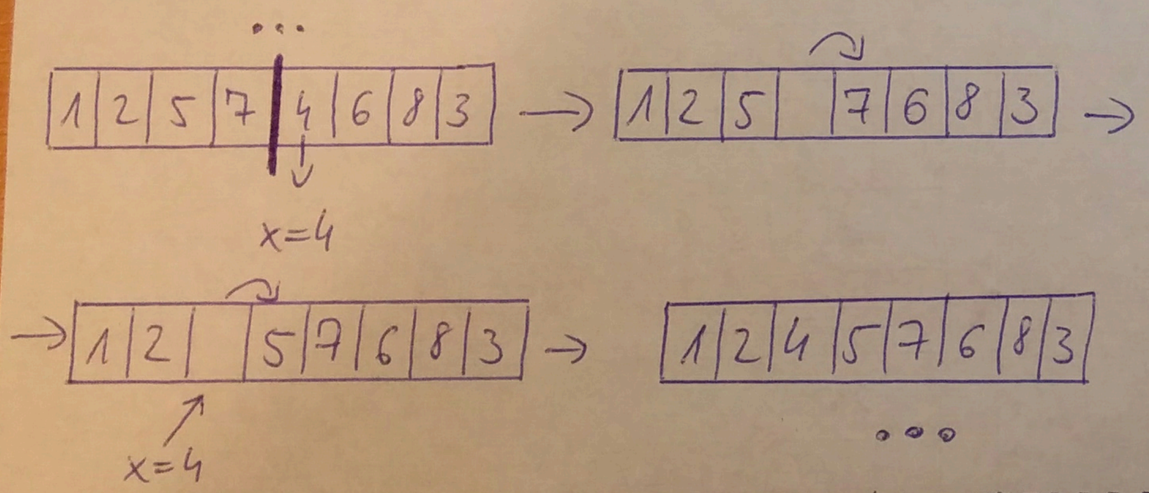
Beszúró rend. lépésrajza:

$\langle 7 | 4,1,4,3,8,9 \rangle \to \langle 4,7 | 1,4,3,8,9 \rangle \to \langle 1,4,7 | 4,3,8,9 \rangle \to$

$\to \langle 1,4,4,7 | 3,8,9 \rangle \to \langle 1,3,4,4,7 | 8,9 \rangle \to \langle 1,3,4,4,7,8 | 9 \rangle$

$\to \langle 1,3,4,4,7,8,9 \rangle$

Beszúró rend. tömbös megvalósítása:

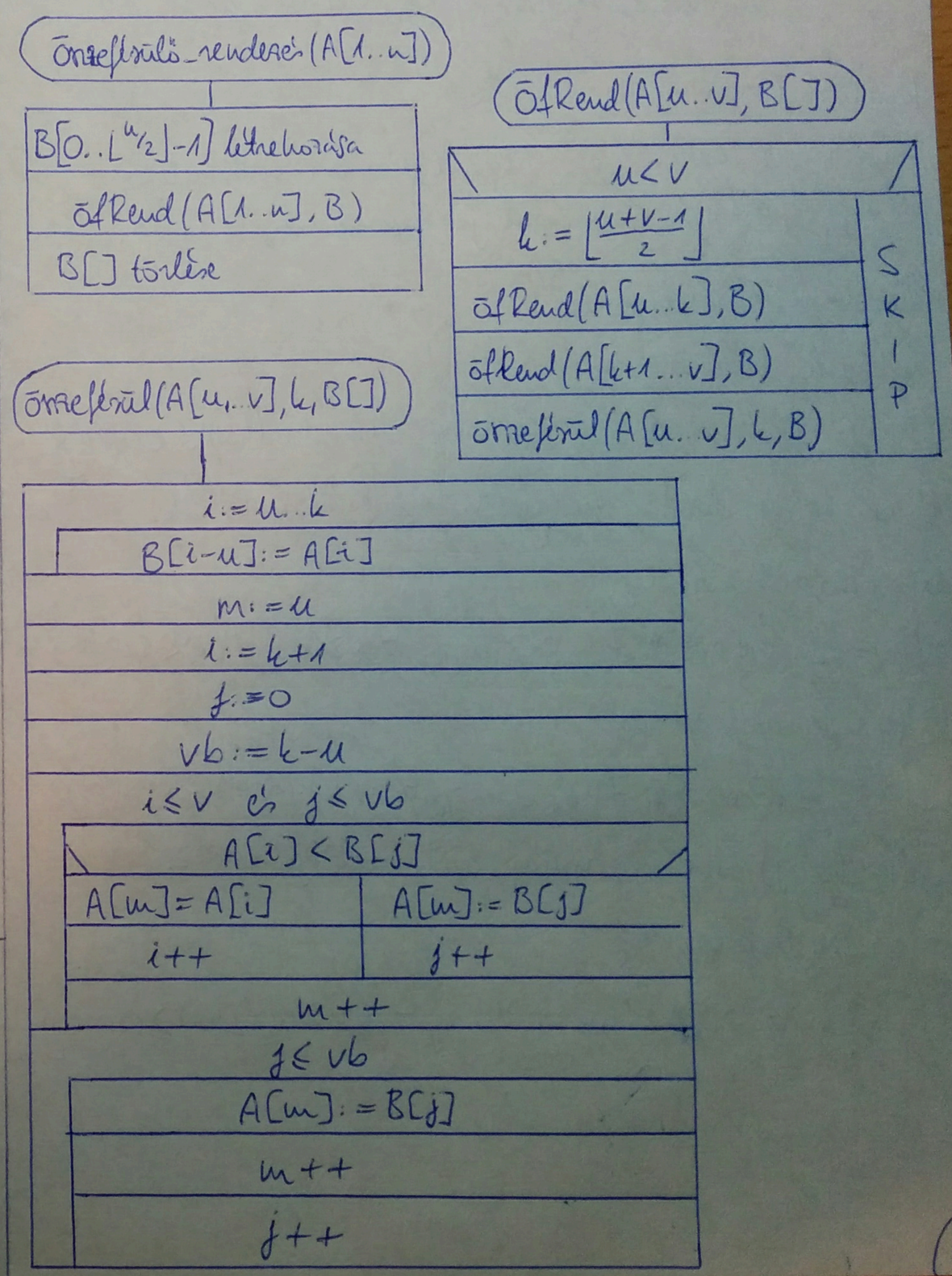


Rendezéses feladat:

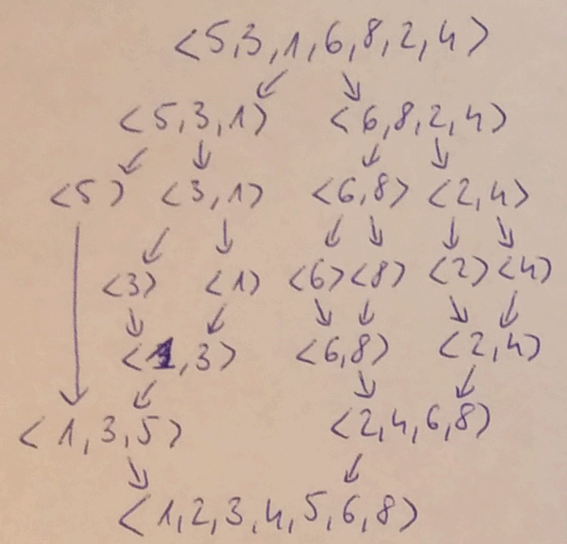
Bemenet: n db hulles sorozata

Kimenet: a bemenet egy monoton növekvő rendezett perm.-ja.

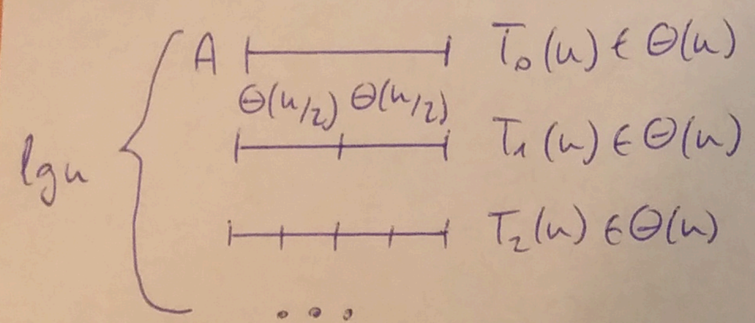
Összefésülő rendezés (MergeSort):



Összefésülő rend. lefutása:



Összefésülő rend. művelet igénye



Tehát $MT(n) \in \Theta(n \cdot \lg n)$

$AT(n) \in \Theta(n \lg n)$

$mT(n) \in \Theta(n \lg n)$

Tárigény:

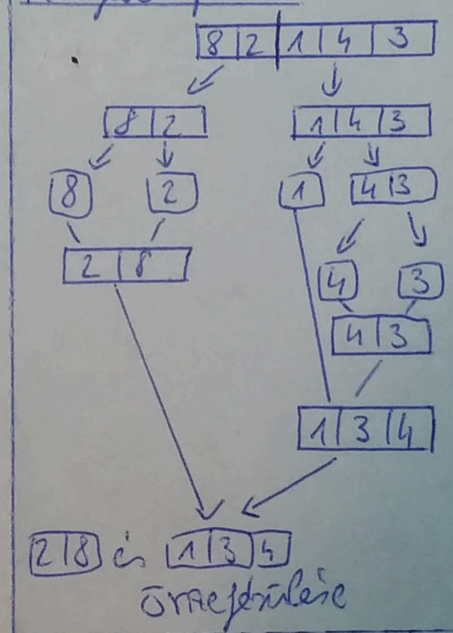
→ tömb: $\Theta(n)$

→ láncolt lista $\Theta(\lg n)$

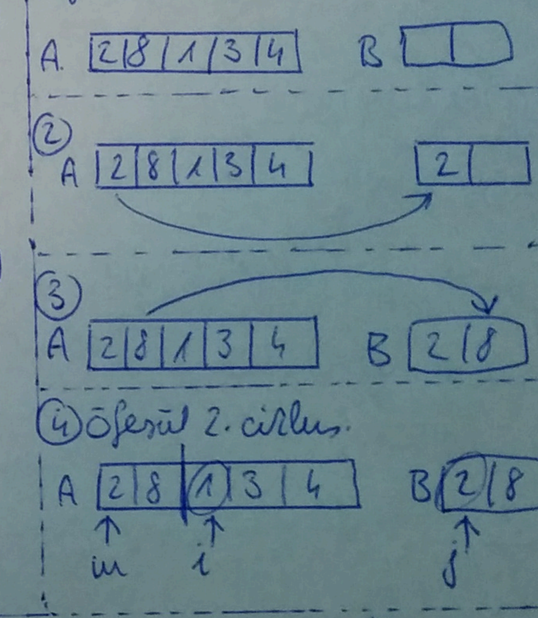
Érdemes-e egyesíteni Insert + Merge sortot? Hogyan?

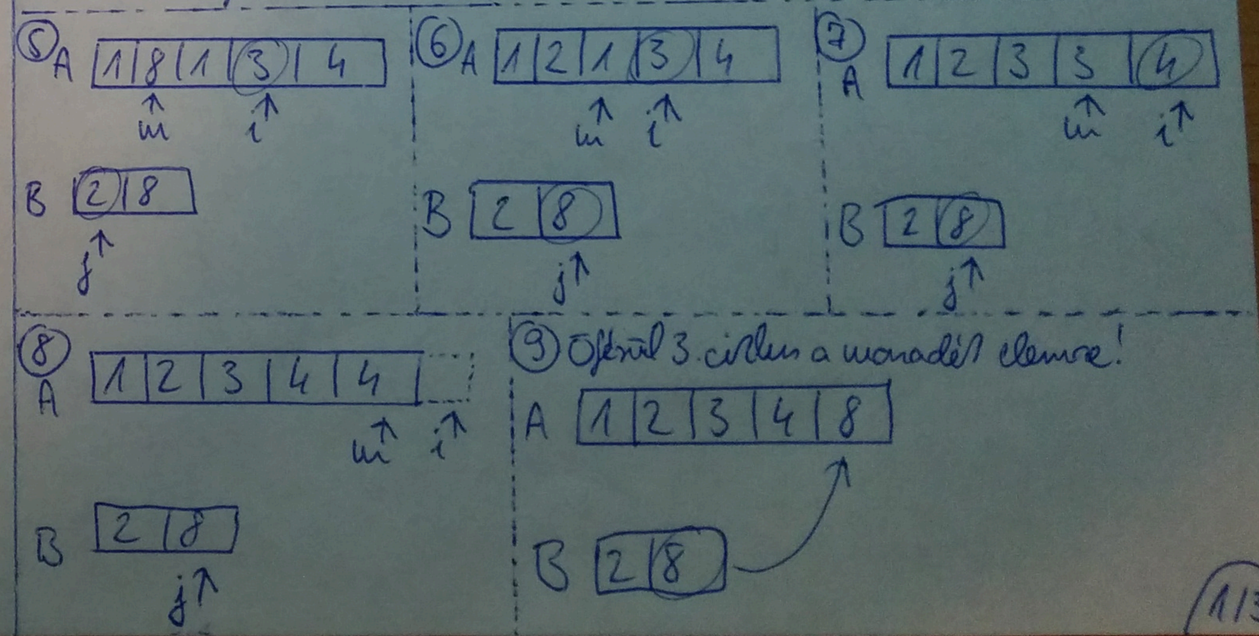
Az Insert sort. $c \cdot n^2$ idejű, az alacsony konstansa miatt jó az $\Theta(n \cdot \lg n)$ algorit.-k gyorsítására. A feladatot $\log n$ méretű részproblémákra osztunk és ezeket Insert s.-al rendezzük. A rendezés $\Theta(n \cdot \lg n)$ marad.

MergeSort példa:



Összefésülés 1. ciklus ①



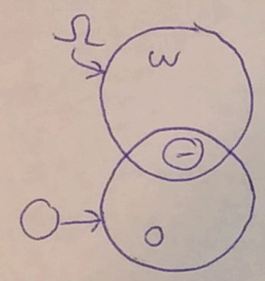


Fv.-ek asimptot. viselk.

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

$o(g) \subsetneq O(g) \setminus \Omega(g)$

$\omega(g) \subseteq \Omega(g) \setminus O(g)$



$1 < \lg n < n^{1/2} < n < n \cdot \lg n < n^{3/2} < n^2 < n^3 < 2^n < n!$

(D) f asimptotikusan nem negatív ha, elég nagy n-re $f(n) \geq 0$

(D) f asimptotikusan pozitív, ha elég nagy n-re $f(n) > 0$

$2^{n+1} \in \Theta(2^n)$? $\quad f(n) = 2^{n+1} \quad g(n) = 2^n$

$2^{n+1} \leq c \cdot 2^n$

$2^n \cdot 2^1 \leq c \cdot 2^n \quad / : 2^n$

$2 \leq c$ $\quad$ Igaz!

$2^{2n} \in O(2^n)$? $\quad f(n) = 2^{2n} \quad g(n) = 2^n$

$2^{2n} \leq c \cdot 2^n \quad / : 2^n$

$2^n \leq c$ $\quad$ Hamis.

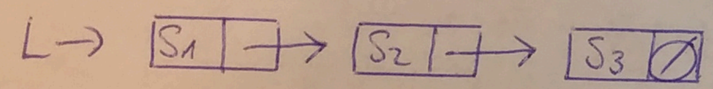
② Tétel: Tömbök, láncolt listák, típusai, listakezelés

lista tulajd.
- → fejelem → van / → nincs
- → láncolás iránya → 1 irányú / → 2 irányú
- → ciklikusság → ciklikus / → Nem ciklikus

(8 féle van)

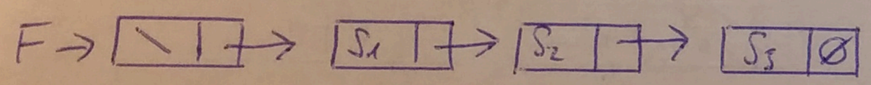
• egyszerű lista (EL)

∅ fejelem, 1 irányú, ∅ ciklikus



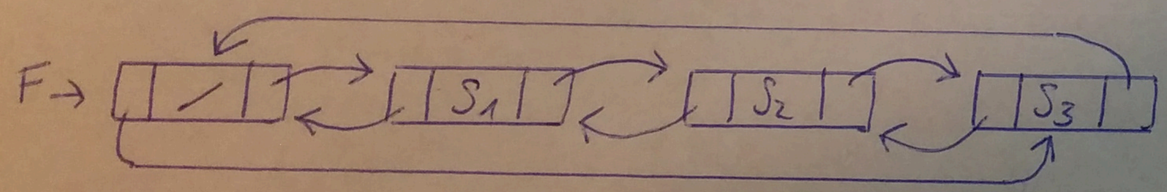
• fejelemes egyszerű lista

fejelemes, 1 irányú, ∅ ciklik.



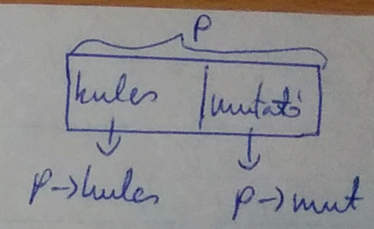
• FKCL

fejelemes, 2 irányú, ciklikus



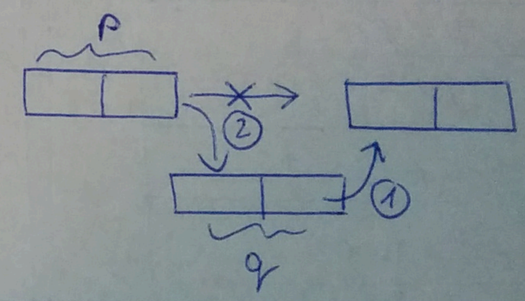
Általában van alt. pointer ami mutathat:
- lista elemére
- NIL-re
- ∅ mutathat fejelemre!

Egyszerű listák műveletei



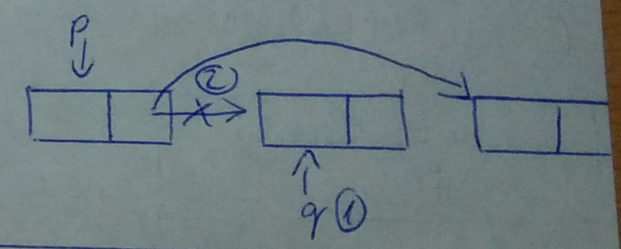
mögé(p,q) (beszúrni)

```
q→mut := p→mut
p→mut := q
```



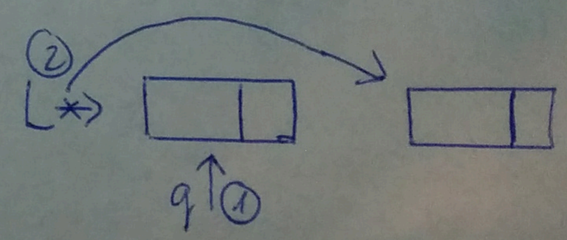
mögül(p,&q) (kiszúrni)

```
q := p→mut
p→mut := q→mut
```



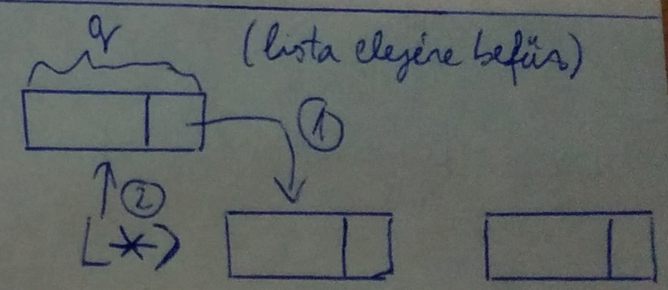
fu elejéről(&L) (lista elejéről kiszúr)

```
q := L
L := q→mut
return q
```



elejére(&L,q)

```
q→mut := L
L := q
```



```
fu EL_hossz(L)
    p := L
    i = 0
    p ≠ ∅
        i++
        p := p->mut
    return i
```

$\Theta(n)$

```
fu FL_hossz(L)
    return EL_hossz(F->mut)
```

Összefésülő (MergeSort) EL-re!

```
MergeSort(&L)
    n := EL_hossz(L)
    mSort(L, n)
```

```
mSort(&L, n)
    n > 1
    n1 := ⌊n/2⌋
    felez(L, n1, L2)
    mSort(L, n1)
    mSort(L2, n-n1)
    merge(L, L2)
```

S K I P

```
felez(L, n1, &L2)
    p := L
    n1 > 1
        p := p->mut
        n1--
    L2 := p->mut
    p->mut := ∅
```

```
merge(&L, L2)
    L->kulcs ≤ L2->kulcs
        p := L
        q = L->mut
        p := L2
        q = L
        L2 := L2->mut
        p->mut := q
        L := p
    q ≠ ∅ és L2 ≠ ∅
        q->kulcs ≤ L2->kulcs
            p := q
            q := q->mut
            r := L2
            L2 := L2->mut
            r->mut := q
            p->mut := r
            p := r
    L2 ≠ ∅
        p->mut := L2
        SKIP
```

$T_{merge}(n) \in \Theta(n)$

$T_{mSort}(n) \in \Theta(n \cdot \lg n)$,

(mivel: $n \cdot \lg n - n \leq T_{mSort} \leq 2 \lg n + 4n$)

$\Downarrow$

$T_{MergeSort}(n) \in \Theta(n \cdot \lg n)$

Fejelemes listák műveletei

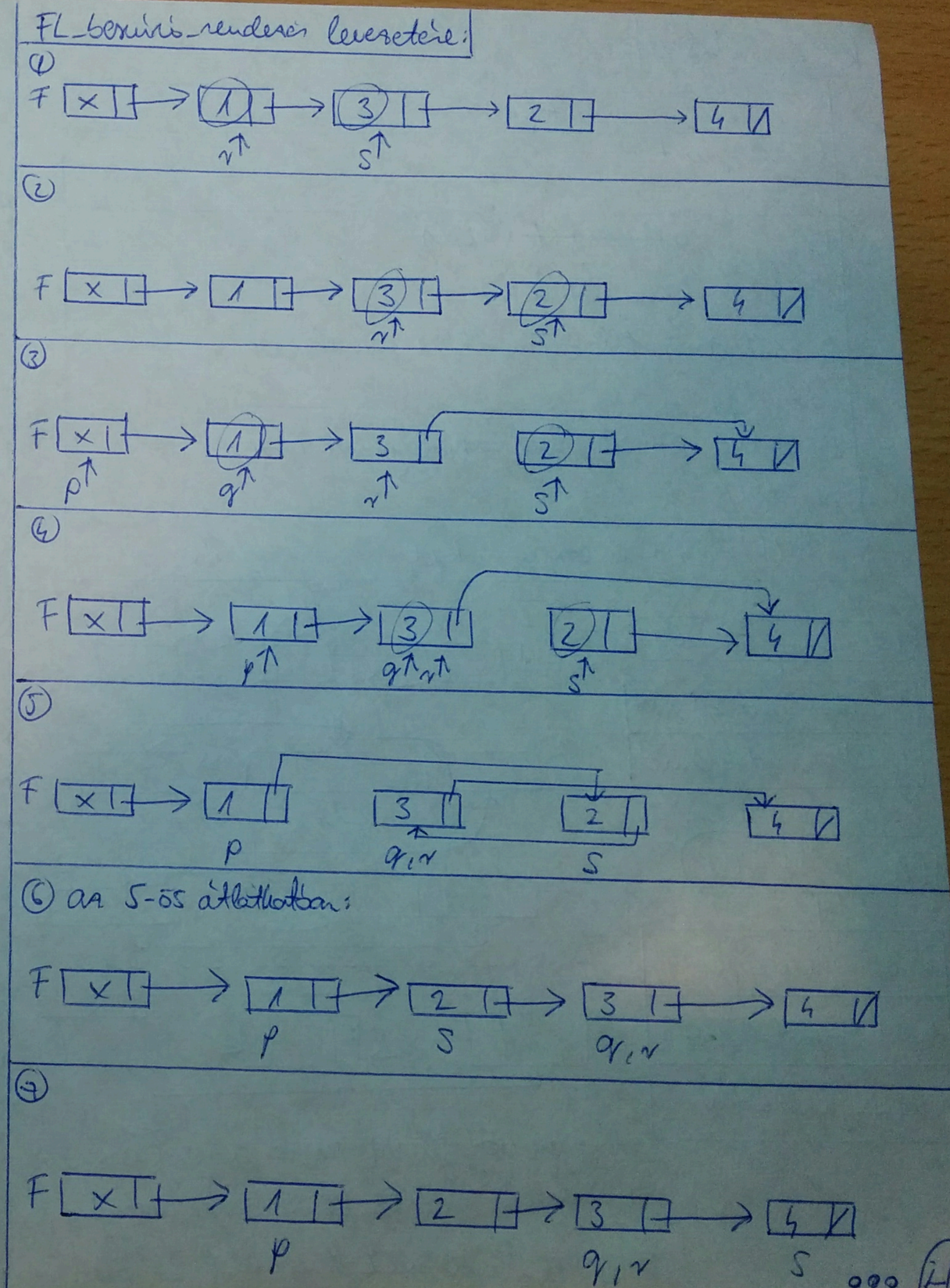
FL_beszúrás(&F)

```
F := V := new Elem
ciklus vége az input végéig
    U := V->mut := new Elem
    beolvas(V->kulcs)
V->mut := ∅
```

FL_beszúró_rendezés(F)

```
v := F->mut
v ≠ ∅
    S := v->mut
    S ≠ ∅
        v->kulcs ≤ S->kulcs
            v->mut := S->mut
            p := F
            q := F->mut
            q->kulcs ≤ S->kulcs
                p := q
                q := q->mut
            S->mut := q
            p->mut := S
        v := S
        S := v->mut
SKIP
```

FL_beszúró_rendezés lefutása:

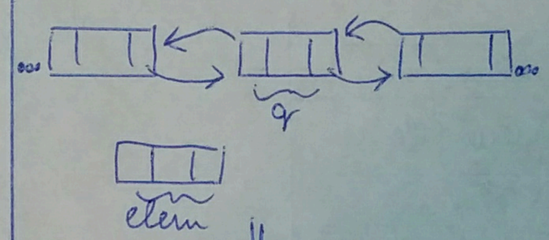


KCL osztály:

Elem2
bmut, jmut kulcs
Elem2 : egyelemű KCL-t hoz létre ~Elem2 : törlés előtt kifűzi balra(q) : q-tól balra átfűzi az elemet jobbra(q) fv bal() : az elem bal szomszédjának címe fv jobb() kifűz() : kifűzi és egyelemű KCL-t csinál belőle

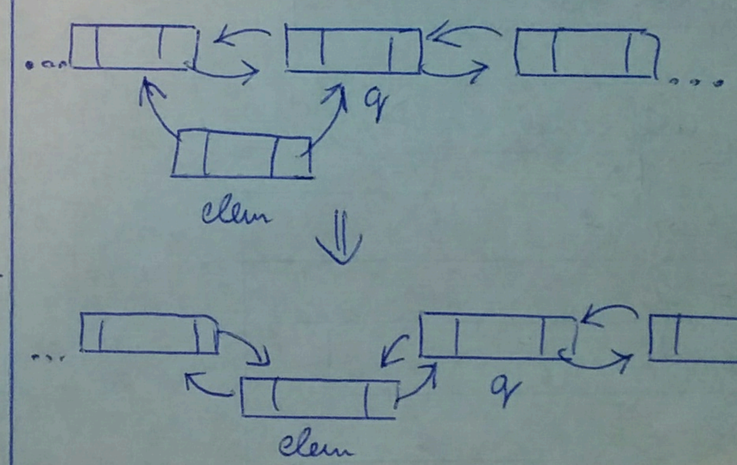
Elem2 balra(q):

-először kifűzzük, majd:



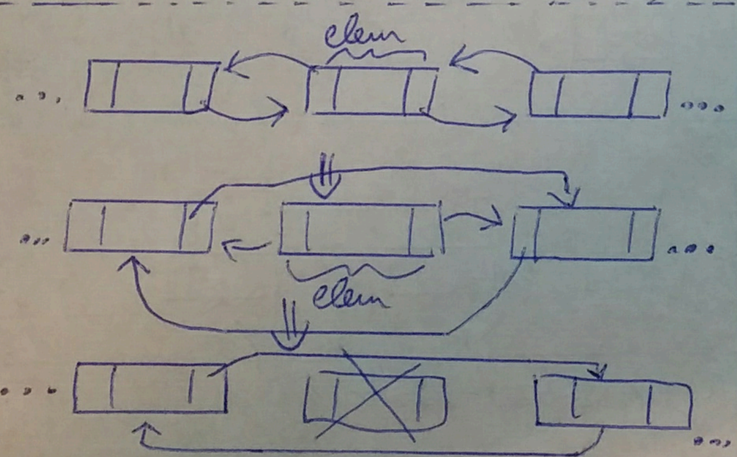
Elem2::balra(q)

kifűz()
p := q→bmut
bmut := p
jmut := q
p→jmut := q→bmut := this



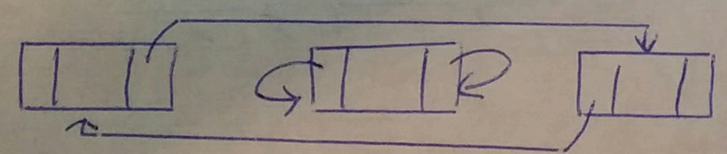
Elem2::~Elem2()

bmut→jmut := jmut
jmut→bmut := bmut



Elem2::kifűz()

bmut→jmut = jmut
jmut→bmut = bmut
bmut=jmut=this



Ugyan úgy kifűzi mint az ~Elem() művelet, csak ez nem törli hanem egy egyelemű KCL-t csinál belőle.

Elem2::jobbra(q)

kifűz()
p := q→jmut
jmut := p
bmut := q
p→bmut = q→jmut = this

FKCL műveletei:

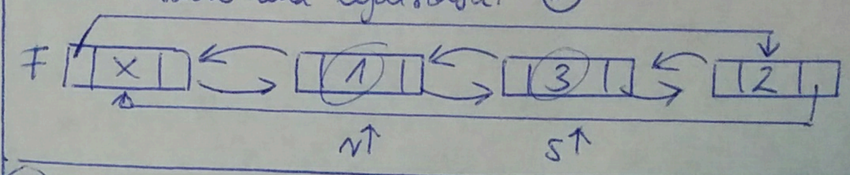
FKCL_beolvasása(&F)

```
F := new Elem2
van még beolvash. adat
    p := new Elem2
    beolvas(p->hules)
    p->balra(F)
```

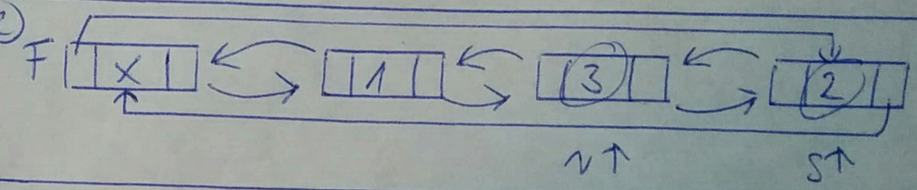
FKCL_beszúró_rendezés(F)

```
r := F->jobb() ; S := r->jobb()
S ≠ F
    r->hules ≤ S->hules
    r := S
        p := r->bal()
        p ≠ F és p->hules > S->hules
            p := p->bal()
        S->jobbra(p)
    S := r->jobb()
```

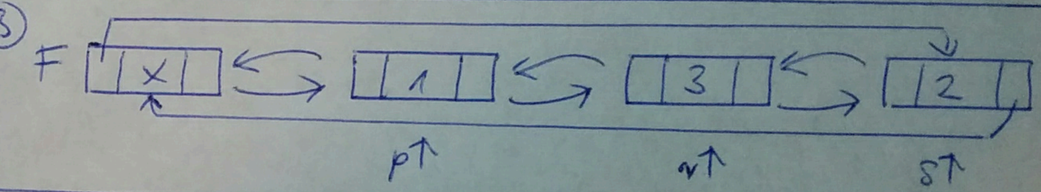
FKCL beszúró rend lépésrása: ①



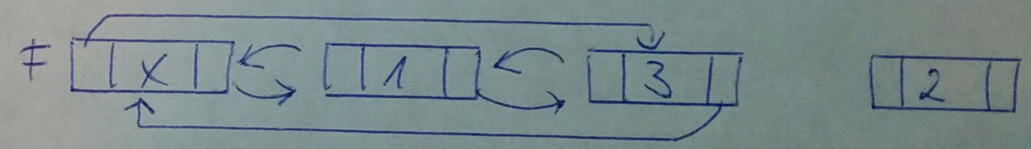
②



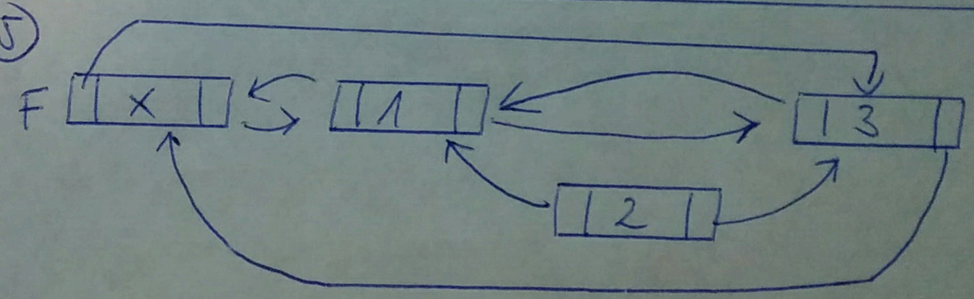
③



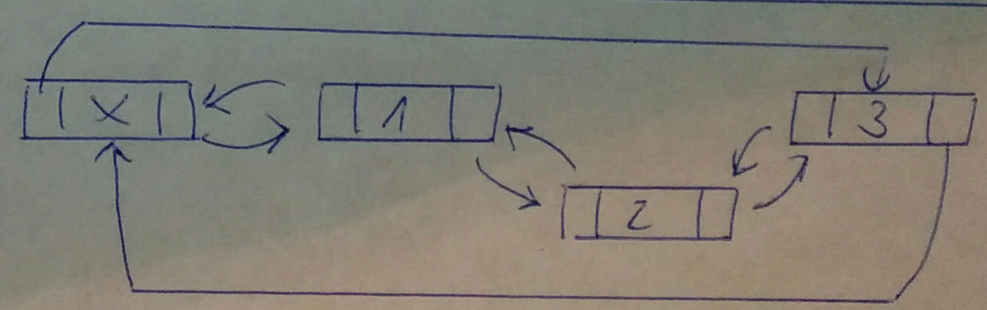
④ S->jobbra(p), ezáltal vagy az S lui is lett fűzve



⑤



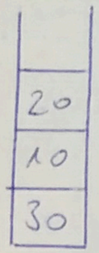
⑥

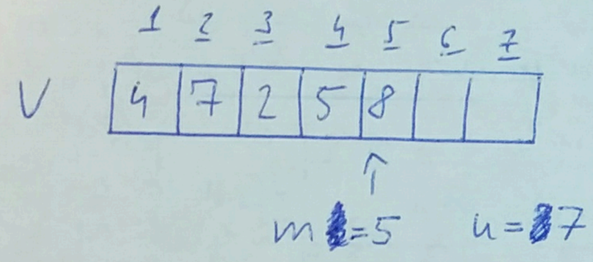


...

③ Tétel: Vermek, sorok, tömbös, láncolt represent., lengyel form.

Verem.

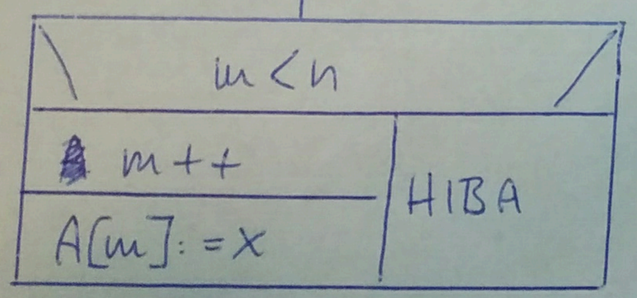




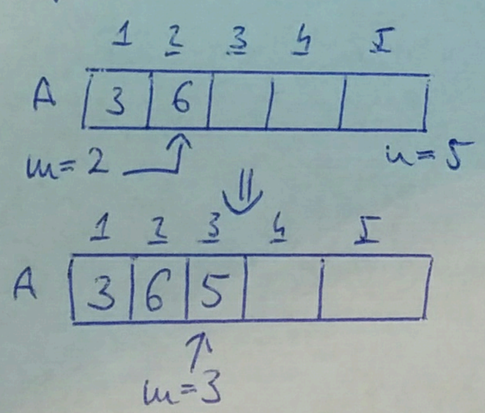
Verem
-A[1..n]: T -m: 0..n
+Verem() {m:=0} +Verembe(x:T) +Verembül():T +Tető():T +tele_e(): L{ret(m=n)} +üres_e(): L{ret(m=0)}

Verem műveletek (tömbös reprez.)

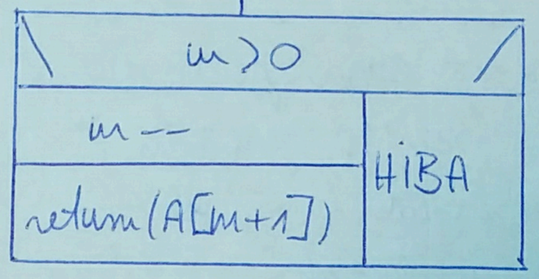
Verem::Verembe(x:T)

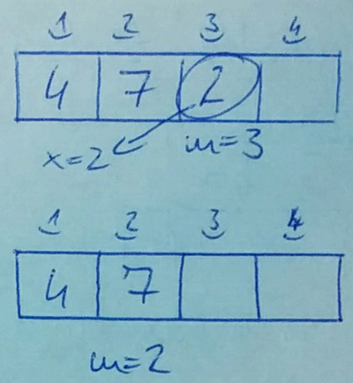


pl. x=5

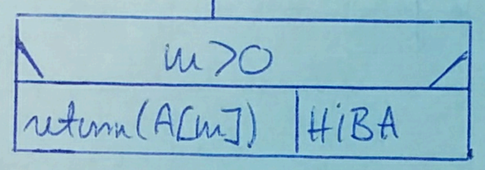


Verem::Verembül():T



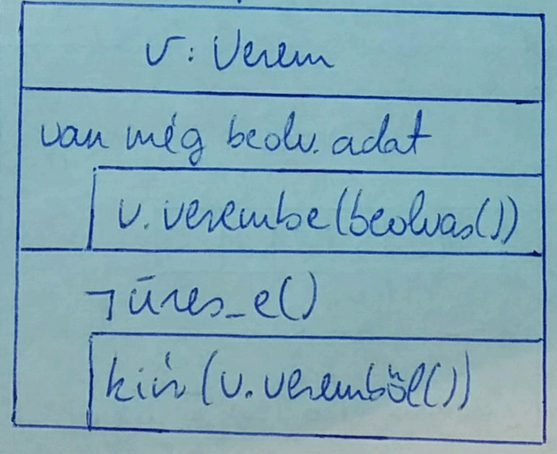


Verem::Tető():T



Visszaadja az elem értékét, de nem törli mint a Verembül művelet.

megfordít()

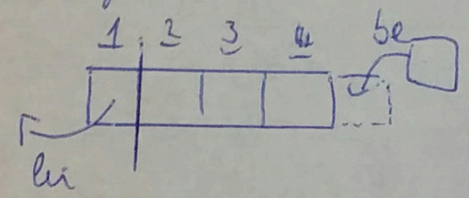


Ez a metódus a beolvasott input adatokat fordított sorrendben írja ki.

Sor (statikus tömbbel reprezentálva)

Sor
-A[0..n-1]:T -m:0..n //sor aktt. mérete -k:0..n-1 // a sor kezdő poz.-ja A-ban
+Sor() {m:=0; k:=0} +Sorba(x:T) +Sorból():T +Első():T //visszad, de nem veszi ki +hossz():$\mathbb{N}_0$ +üres_e():$\mathbb{L}$ +tele_e():$\mathbb{L}$

A sor egy végtelen szemre hasonlít; a legrégebben beleszórt elemet veszem ki legelőször.



Sor műveletei (statisztikus)

Sor::Sorba(x:T)

m<n	
A[(k+m) mod n]:=x	HIBA
m++	

x=6

	2	5	9		

m=3 k=2

A[(2+3) mod 6] → 5. hely

⇓ x=6

	2	5	9	6	

Sor::Sorból():T

m>0	
i:=k	HIBA
k:=(k+1) mod n	
return(A[i])	

Konkrétan kiveszi az elemet a sorból (kijövő) és visszaadja

1	2	3	4	5	6	7
		2	3	5	9	

k=3 m=4

Kiveszem

⟹

1	2	3	4	5	6	7
			3	5	9	

k:=(3+1) mod 7 = 4. elem lesz a sor új első eleme

Sor::Első():T

m>0	
return A[k]	HIBA

Visszaadja a sor első elemét, de nem veszi ki a sorból.

4. Tétel: Prior sorok, tömbbel, listával, kupaccal; Művelet igényük

PrSor tömbbel:

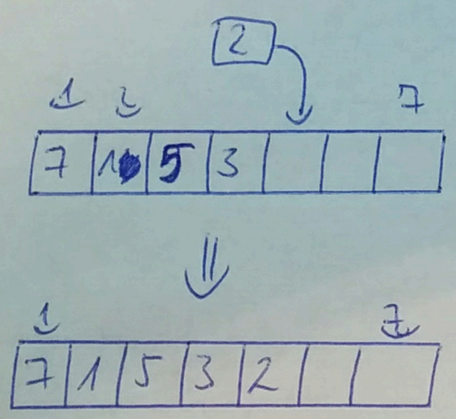
PrSor
-A[1..n]:T
-m:0..n // PrSor aktuál. mérete
+PrSor() {m:=0} // üres PrSor létreh.
+prSorba(x)
+fv maxKiven() // kiven és visszaad
+fv max() // megnézi és visszaadja
+fv tele_e() {return m=n}
+fv üres_e() {return m=0}

PrSor (rendezetlen tömb) műveletei

Tehát A[1..m] rendezetlen.

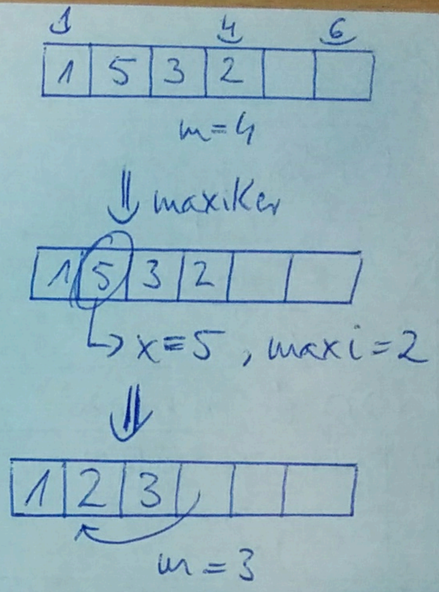
PrSor::prSorba(x)

m<n	
m++	HIBA (tele van)
A[m]:=x	



fv PrSor::maxKiven()

m>0	
maxi:=maxiKen(A[1..m])	HIBA
x:=A[maxi]	
A[maxi]:=A[m]	
m--	
return x	



fv maxiKen(A[1..m])

maxi=1	
i=2	
i≤m	
A[i] > A[maxi]	
maxi:=i	SKIP
i++	
return maxi	

Műveleti idő (rendezetlen A[1..m])

PrSor() = Θ(1)

PrSorba() = Θ(1)

maxKiven() = Θ(m)

max() = Θ(m) vagy Θ(1)

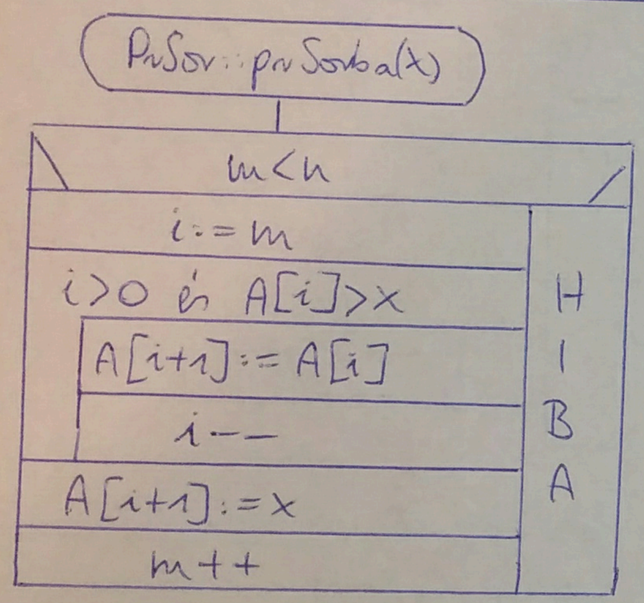
tele_e() = Θ(1)

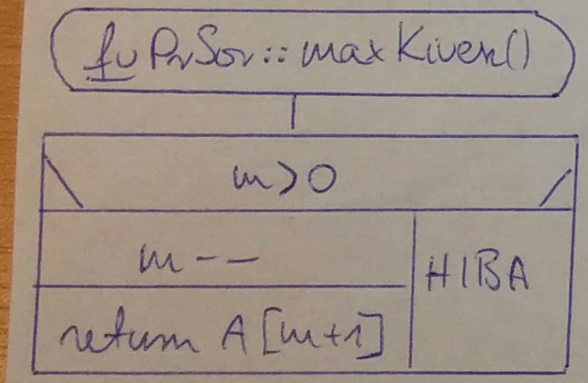
üres_e() = Θ(1)

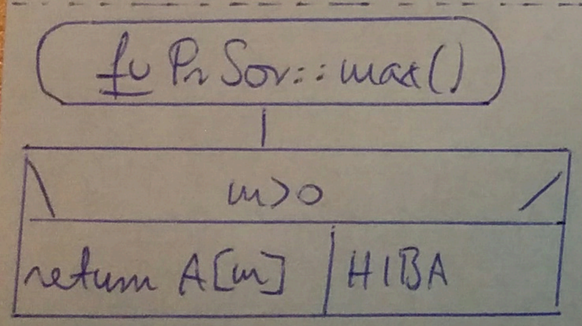
fv PrSor::max()

m>0	
maxi=maxiKen(A[1..m])	HIBA
return A[maxi]	

PrSor (mon, nov. rendezett tömb) műveletei







Műveleti idő (rendezett A[1..m])

$PrSor = \Theta(1)$

$PrSorba \rightarrow mT(n) = \Theta(1)$

$\rightarrow AT(n) = \Theta(m)$

$\rightarrow MT(n) = \Theta(m)$

$maxKiven = \Theta(1)$

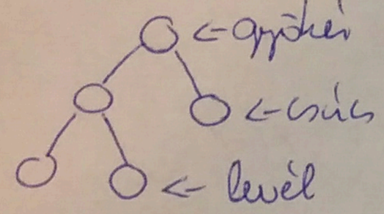
$max = \Theta(1)$

$tele_e = \Theta(1)$

$üres_e = \Theta(1)$

(5) Tétel: Fák, bináris fák, bejárások, láncolt reprezentációk

Bin. fák



t : fa

*t : gyökércsúcs

t->bal : t bal részfája

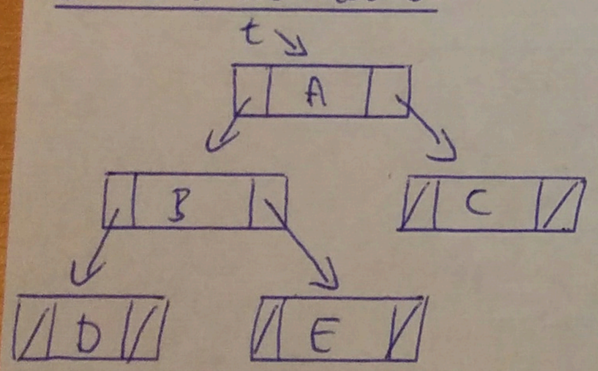
*t->bal : bal gyerek

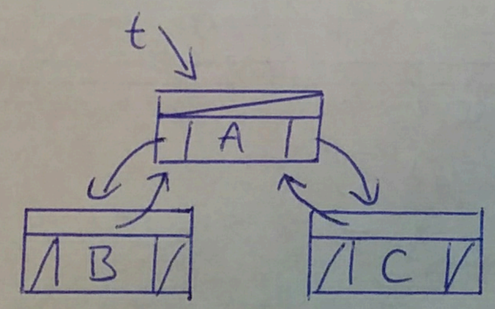
t->kulcs

Fa magassága:

- gyökér a 0. szint
- üres fa magassága −1
- ált. képlet: $h(t) = 1 + \max\{h(t\to bal), h(t\to jobb)\}$

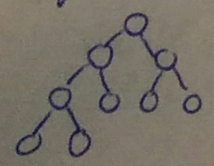
Láncolt ábrázolás:



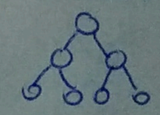


Bin. fák típusai:

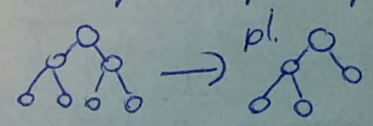
- **Szigorúan bin. fa:** Olyan bin. fa, melyben minden belső csúcsnak (azaz nem levél) két gyereke van.



- **Teljes bin. fa:** Olyan szigorú bin. fa, melynek minden levele azonos szinten van



- **Majdnem teljes bin. fa:** Ha egy teljes bin fa levélszintjéről 0, 1.. vagy több levelet elveszünk (de nem az összeset), akkor majdnem teljes bin fát kapunk



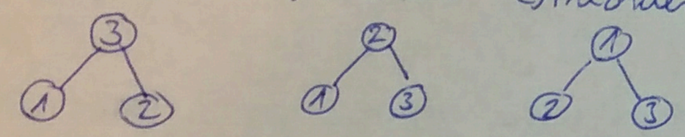
- **Méret szerint kiegyensúlyozott bin. fa:** Olyan bin fa melynek tetszőleges nemüres részfájának bal és jobb részfájának mérete max 1-el térhet el.

- **Magasság szerint kiegyensúlyozott bin fa:** Olyan bin fa melynek minden csúcsa kiegyens.-ít.

- **Balra tömörített bin. fa:** Olyan majdnem teljes bin fa, aminek az alsó szintjén egyetlen levéltől balra sem lehet új levelet beszúrni.

Fa bejárások:

a) Postorder b) Inorder c) Preorder d) Szintfolytonos



preorder(t)

$t \neq \emptyset$	
feldolgoz(t)	SKIP
preorder(t→bal)	
preorder(t→jobb)	

postorder(t)

$t \neq \emptyset$	
postorder(t→bal)	SKIP
postorder(t→jobb)	
feldolgoz(t)	

Szintfolytonos(t)

$t \neq \emptyset$	
S: Sor	SKIP
S.sorba(t)	
¬S.üres_e()	
f := S.sorból()	
feldolgoz(f)	
f→bal ≠ ∅	
S.sorba(f→bal) | SKIP	
f→jobb ≠ ∅	
S.sorba(f→jobb) | SKIP	

inorder(t)

$t \neq \emptyset$	
inorder(t→bal)	SKIP
feldolgoz(t)	
inorder(t→jobb)	

$T_{pre} \in \Theta(n)$

$T_{post} \in \Theta(n)$

$T_{in} \in \Theta(n)$

$T_{szintf.} \in \Theta(n)$

Fa magasságát meghatározó algoritmus

fv h(t)

$t \neq \emptyset$	
bm := h(t→bal)	return −1
bj = h(t→jobb)	
return 1+max(bm,bj)	

$T(n) \in \Theta(n)$

Bináris láncolt ábrázolás

Csúcs / Csúcs3

kulcs : T
bal, jobb : Csúcs* // pointer egyenlőre
(sz : Csúcs* // pointer szülőre)
Csúcs()
Csúcs(x:T)
Csúcs(x:T, b:Csúcs*, j:Csúcs*)
Csúcs(x:T, b:Csúcs*, j:Csúcs*, sz:Csúcs*)
...

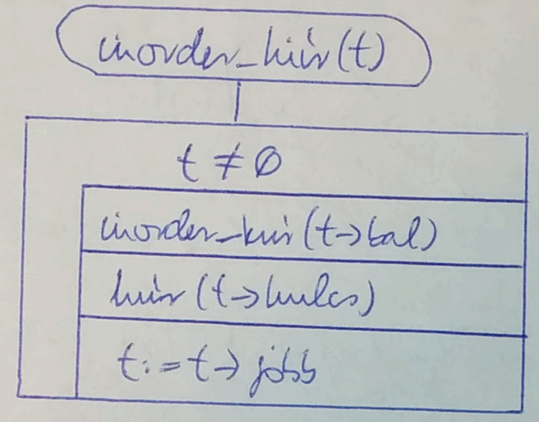
6. Tétel: Spec. bin. fák, bin. ker. fa, kupac, kupac műveletei, kupacrendezés

Bin. ker. fa def.: Egy t bin fa pontosan akkor bin. ker fa, ha t bármely x csúcsára igaz az, hogy ha y az x bal részfájának egy csúcsa, z pedig az x jobb részfájának a csúcsa, akkor: $kulcs(y) < kulcs(x) < kulcs(z)$

Bin ker fában inorder bejárással a kulcsokat rendezett sorozatként járjuk el.

Bin ker fa felírás pl. 40, 110, 30, 60, 90, 10

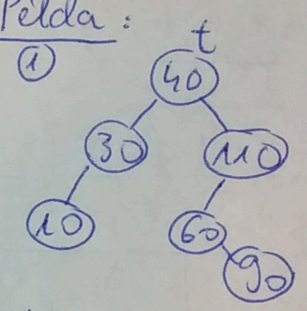
Bin. ker. fa műveletei:



$T(n) \in \Theta(n)$

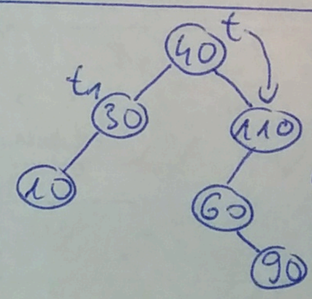
Példa:

1



Out:

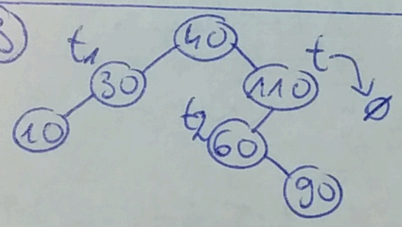
2



Out: 40

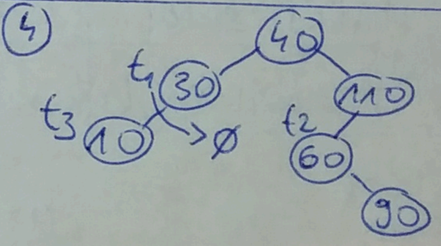
a) rekurzió
b) kiírás
c) lépés

3



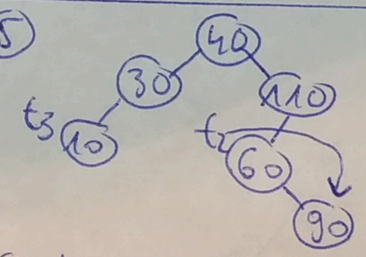
Out: 40,110

4



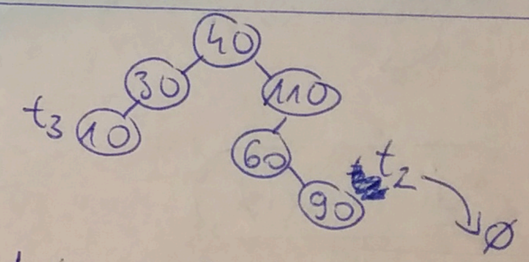
Out: 40,110,30

5



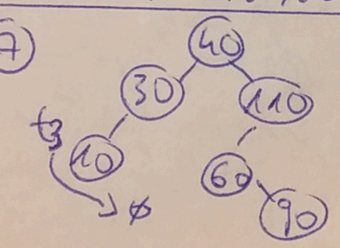
Out: 40,110,30,60

6



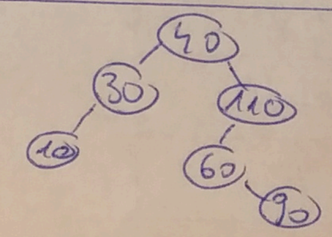
Out: 40,110,30,60,90

7



Out: 40,110,30,60,90,10

8



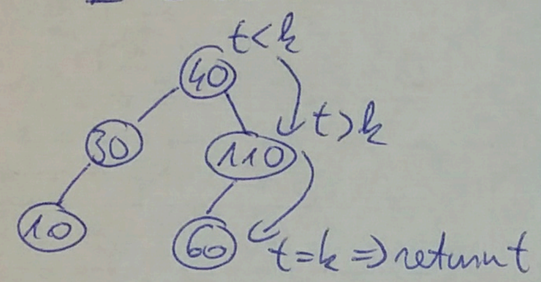
Out: 40,110,30,60,90,10 ✓

61

fv keres(t,k)

- $t \neq \emptyset$ és $t \to kulcs \neq k$
  - $k < t \to kulcs$
    - $t := t \to bal$
    - $t := t \to jobb$
  - return t

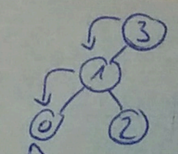
pl. k=60



$mT(h) \in \Theta(1)$, $MT(h) \in \Theta(h)$

minKiven(&t, &minp)

- $t \to bal = \emptyset$
  - $minp := t$
  - $t := minp \to jobb$
  - $minp \to jobb = \emptyset$
  - minKiven(t→bal, minp)

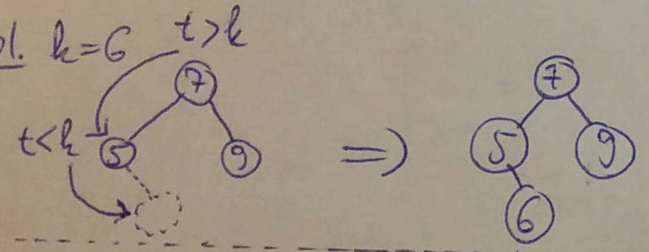


Mutain ∅ a t-nek semmi a bal részfájában csak ez lehet a min elem, tehát ráállítjuk minp-t. Ezután t-t kifűzzük

beszúr(&t, k)

- $t = \emptyset$
  - $t := new\ Csúcs(k)$
- $k < t \to kulcs$
  - beszúr(t→bal, k)
- $k > t \to kulcs$
  - beszúr(t→jobb, k)
- $k = t \to kulcs$
  - SKIP

pl. k=6



$MT(h) \in \Theta(h)$

$mT(h) \in \Theta(1)$

törül(&t, k)

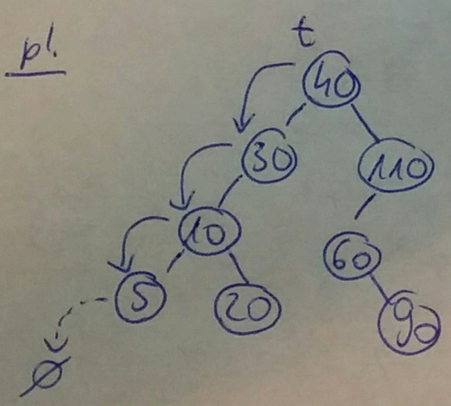
- $t \neq \emptyset$
  - $k < t \to kulcs$
    - törül(t→bal, k)
  - $k > t \to kulcs$
    - törül(t→jobb, k)
  - $k = t \to kulcs$
    - gyökerettörül(t)
  - SKIP

$MT(h) \in \Theta(h)$

fv min(t)

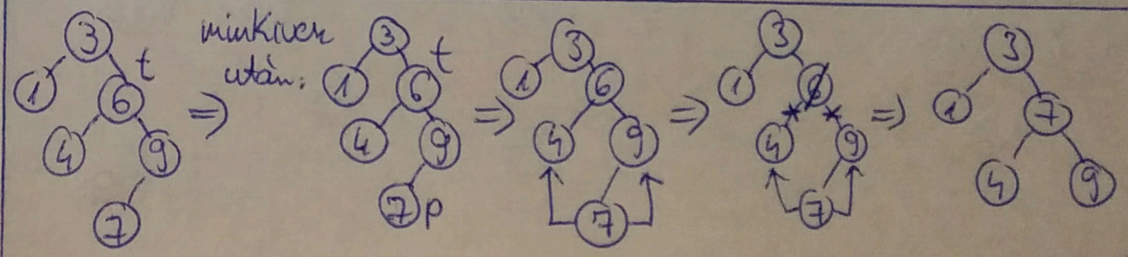
- $t \to bal \neq \emptyset$
  - $t := t \to bal$
- return t

pl.



gyökerettörül(&t)

$t \to bal = \emptyset$	$t \to jobb = \emptyset$	$t \to bal \neq \emptyset$ és $t \to jobb \neq \emptyset$
$p := t$	$p := t$	minKiven(t→jobb, p)
$t := p \to jobb$	$t := p \to bal$	$p \to bal = t \to bal$; $p \to jobb = t \to jobb$
delete p	delete p	$t \to bal = t \to jobb = \emptyset$; delete t; t=p



• Bináris fa összes eddigi művelete $MT(h) \in \Theta(h)$, a műveletek hat.-a h-tól (fa magasság) függ.

• n csúcsú fa magassága: $\lfloor \lg n \rfloor \ldots (n-1)$ között

• Ha a kulcsok sorrendje véletlenszerű akkor a fa magassága általában $\sim \Theta(\lg n)$

$\Downarrow$

tehát kedvezőbb mint tömbbel, listával!

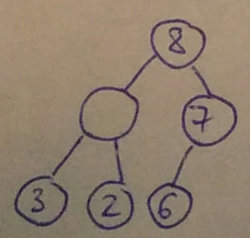
Kupac (heap)

• Max. kupac: bal-ra tömörített majdnem teljes bin. fa, aminek $\forall$ belső csúcsa nagyobb vagy egyenlő mint a gyerekeié

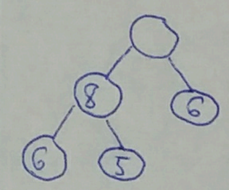
• Min. kupac: bal-ra tömör.-tt majdnem teljes bin fa, melynek $\forall$ belső csúcsa kisebb v. egyenlő mint a gyerekeié

Előadáson: kupac = max kupac

• Lyukas kupac: bal-ra t.-tt majd. telj. bin fa lyukas kupac ha a fa egyik csúcsa a lyuk és $\forall$ szülő-gyerek párosban a szülő kulcsa nagyobb mint a gyerek kulcsa, kivéve ha a páros egyik tagja a lyuk. Lyuk kulcsa nem definiált, de a lyuk szülőjének kulcsa nagyobb/egyenlő mint a gyerekei kulcsai.



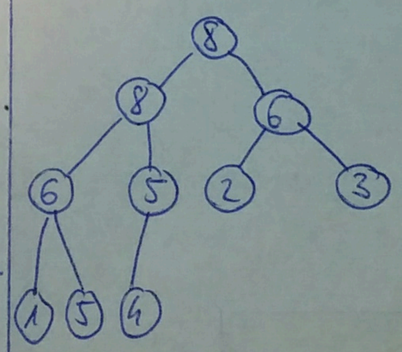
• Csonka kupac: Egy bal-ra tömör. majdn. telj. bin fa csonka kupac ha $\forall$ szülő-gyerek párosban a szülő kulcsa nagyobb/egyenlő mint a gyerek kulcsa, kivéve ha a szülő a gyökércsúcs. A gyökércsúcs kulcsa nem definiált (de lehet hogy kisebb mint a gyerek kulcsa)

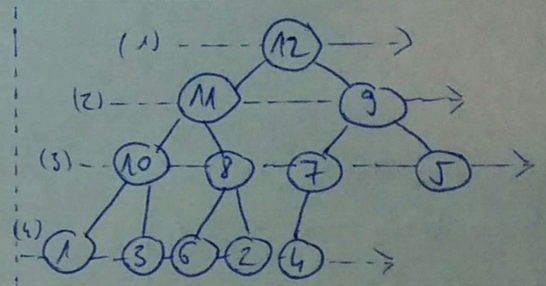


Vektor kupaccá alakítása és fordítva:

8	8	6	6	5	2	3	1	5	4

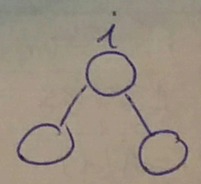
$\Downarrow$





12 | 11, 9 | 10, 8, 7, 5 | 1, 3, 6, 2, 4

• A kupacot egy $A[1..n]$ tömbben tároljuk amit n elemig használunk



$j \in \{2i \mid 2i+1\}$

i bal gyereke, i jobb gyereke

• Kupac max-a a gyökérben

Kupacok műveletei:

PrSor
-A[1..n]:T
-m:0..n // aktuális méret
PrSor() {m=0}
üresPrSor() {m:=0} // kitüntetett a PrSort
prSorba(x:T)
fv maxKiven():T // kiven és visszaad
fv max():T // megnézi, visszaadja
fv tele_e(): L
fv üres_e(): L

Ha A[1..m] rendezett tömb a kupac:

- prSorba → MT(n) ∈ Θ(lg m)
  → mT(n) ∈ Θ(1)
- maxKiven → MT(n) ∈ Θ(lg m)
  → mT(n) ∈ Θ(1)
- max → T(m) ∈ Θ(1)

PrSor::prSorba(x)

m<n	
m++	HIBA
j:=m	
i:= ⌊m/2⌋	
i>0 és A[i]<x	
A[j]:=A[i]	
j:=i	
i= ⌊i/2⌋	
A[j]:=x	

Algo lyátrása:

1. 12 | 11 | 9 | 10 | 8 | 7 |
   x=10 m=6

2. 12 | 11 | 9 | 10 | 8 | 7 |
   m=7

3. 12 | 11 | 9 | 10 | 8 | 7 |
   j=m
   $i=\lfloor \frac{m}{2} \rfloor = \lfloor \frac{7}{2} \rfloor = 3$

4. 12 | 11 | 9 | 10 | 8 | 7 |
   i j
   x=10

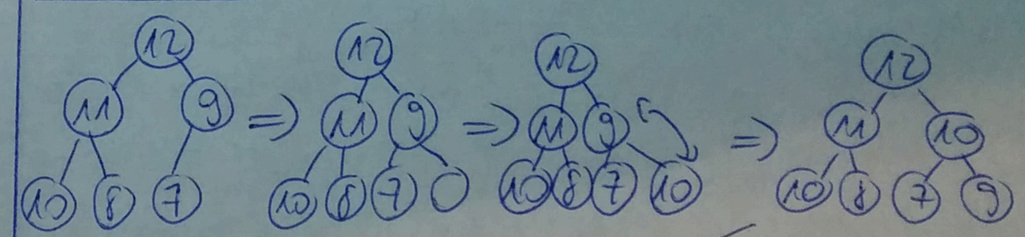
5. 12 | 11 | 9 | 10 | 8 | 7 | 9
   i j
   $\lfloor \frac{i}{2} \rfloor = 1$

6. 12 | 11 | 9 | 10 | 8 | 7 | 9
   i j
   x=10

7. 12 | 11 | 10 | 10 | 8 | 7 | 9
   ✓

Most a kendőcuktort nyisd fel bon fa-ként és ugy szúrgald be a 10-et!

10 beszúrása:



Input volta tömbre: 12,11,10,10,8,7,9

fv PrSor::max()

m>0	
return A[1]	HIBA

fv PrSor::maxKivesz()

m>0	
max:=A[1]	HIBA
m--	
lesüllyent(A[1..m],A[m+1])	
return max	

lesüllyent(A[k..m],x)

i jelzi a lyuk helyét, j a lyuk bal v. jobb gyerekére áll

)# A[j] a lyuk bal gyereke

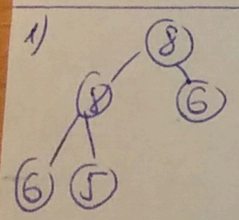
)# már A[j] a lyuk nagyobbik gyereke

i:=k	
j:=2k	
b:=igaz	
j≤m és b	
j<m és A[j+1]>A[j]	
j++	SKIP
A[j]>x	
A[i]:=A[j]	~~b:=hamis~~
i=j	
j=2j	
A[i]:=x	

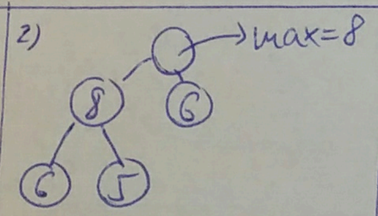
← megnézzük hogy a bal v. j. a nagyobb

maxKivesz:

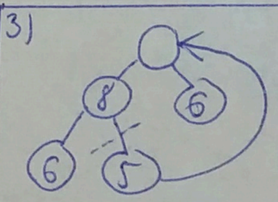
1)



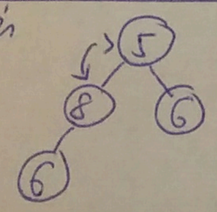
2) max=8



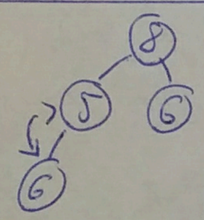
3)



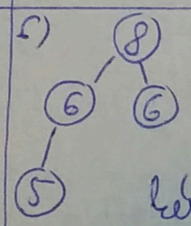
4) 5<8 és 8>6 ezért 5 és 8 cere



5) 5<6 ezért cere



6)



kész!

lesüllyent lépésrása:

0)

12	11	9	10	8	7

max (maxKivesz)

1)

12	11	9	10	8

i (lyuk) j (lyuk bal gy.) x=7 ← lesülly. elem b=↑

2) 11>7 ezért:

11	11	9	10	8

i (lyuk) j b=↑ x=7

3) 10>7 ezért:

11	10	9	10	8

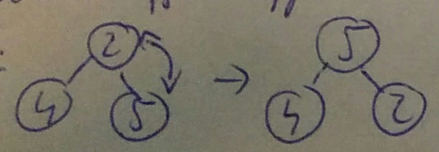
i j=8 (kifutott)

Mivel j=8 > m=5, vége a cél-node lyukba bemásolyuk x-et!

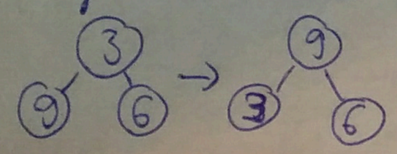
11	10	9	7	8

A következ. lesüllyent elj.-al A[i]-t addig süllyesztük le amíg x bele nem illik. A lyukat mindig az ő alá nagyobb gyerekkel cseréljük meg:

pl.



pl.2:

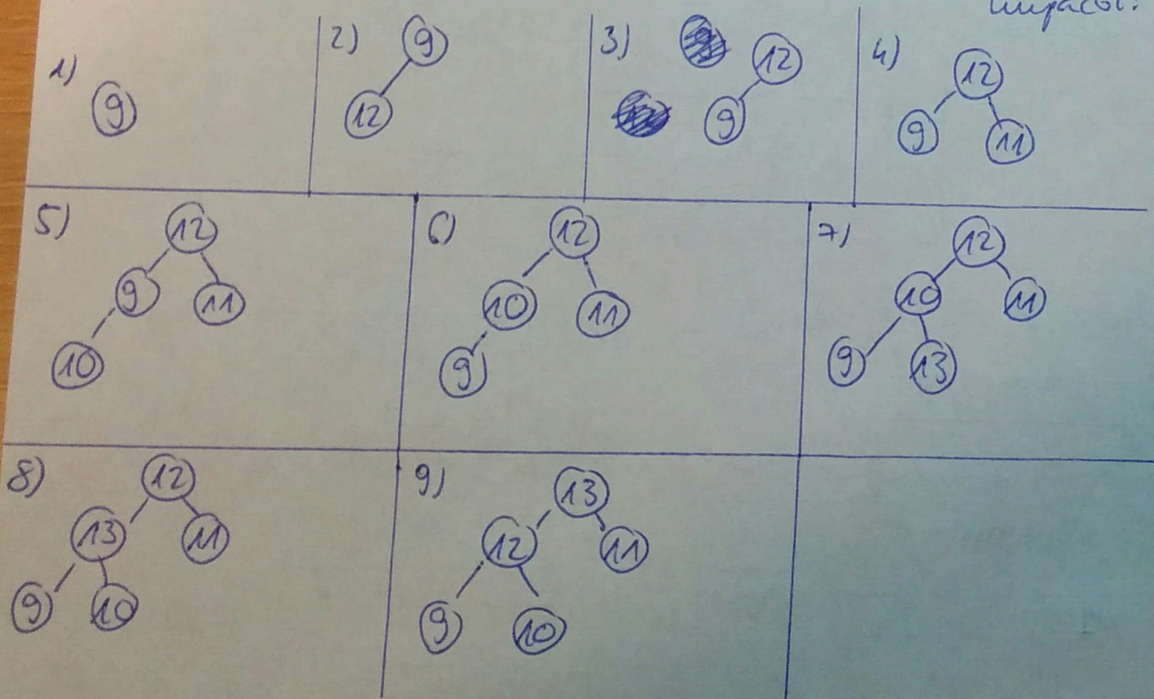


```
prSorralRendez(A[1..n])
    Q: PrSor
    i := 1..n
        Q.prSorba(A[i])
    i := n..1 (-1)
        A[i] := Q.maxKivesz()
```

$MT(n) \in \Theta(n \lg n)$

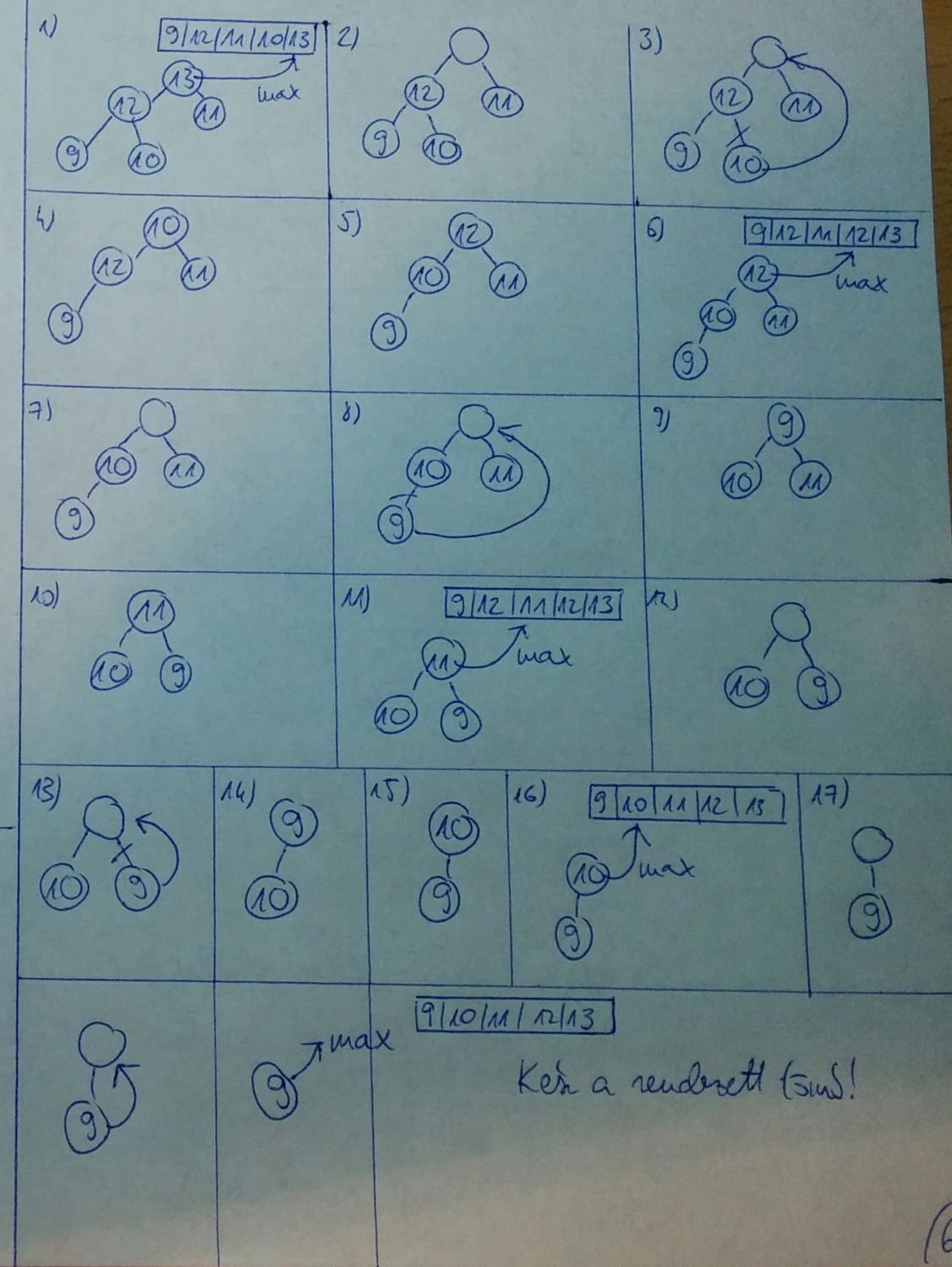
PrSorralRendez lejátszása:

pl. A [9|12|11|10|13], először felépítjük a prSorba() műv-el a kupacot!



Felépítve, ezután maxKivesszel hátulról feltöltjük az A tömböt.

maxKivesszel A rendezése:

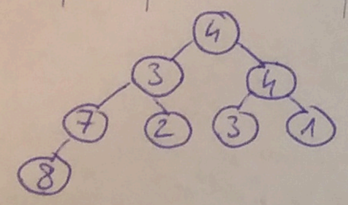


Kész a rendezett tömb!

Kupacrendezés (HeapSort)

A prioritásos sorok optimalizált változata.

4	3	4	7	2	3	1	8



kupaccá_alakít(A[1..n])

- $k := \lfloor \frac{n}{2} \rfloor \ldots 1 \ (-1)$
  - lesüllyent(A[k..n], A[k])

$mT(n), MT(n) \in \Theta(n)$

kupacrendezés(A[1..n])

- kupaccá_alakít(A[1...n]) } $\Theta(n)$
- $m := n$
- $m > 1$
  - $x := A[m]$
  - $A[m] := A[1]$
  - m --
  - lesüllyent(A[1..m], x) } $O \lg n$

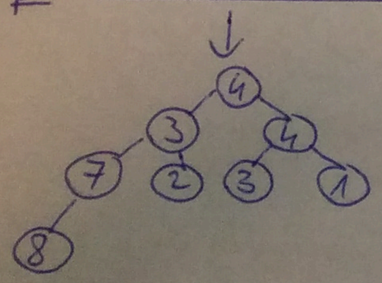
$MT(n)$

$AT(n) \in O(n \cdot \lg n)$

$mT(n)$

Kupacrendezés lépései:

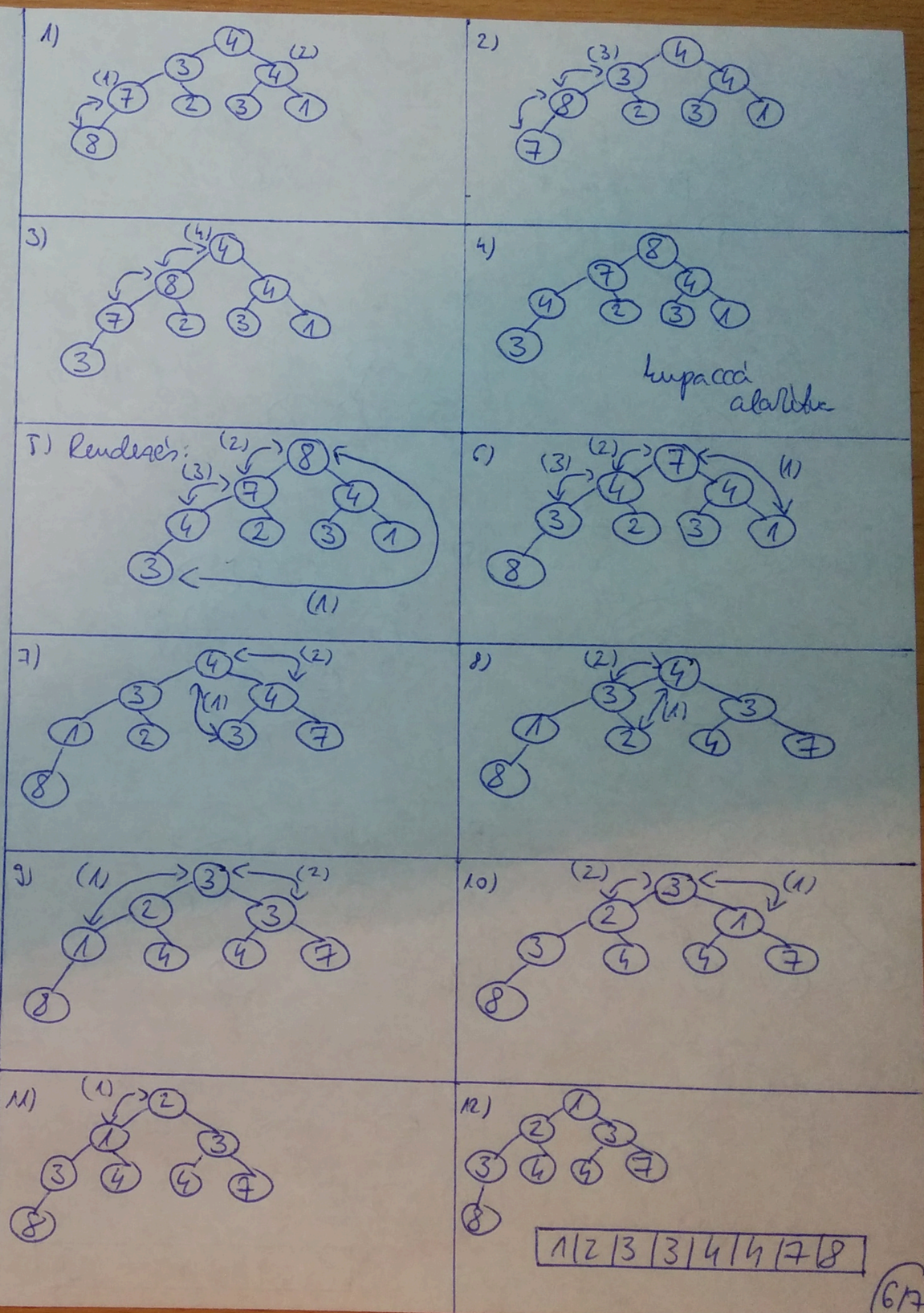
pl. | 4 | 3 | 4 | 7 | 2 | 3 | 1 | 8 |



Először kupaccá alakítás!

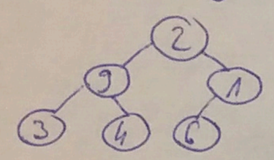
$k = \lfloor \frac{n}{2} \rfloor = \lfloor \frac{8}{2} \rfloor = 4$

$\Downarrow$



1	2	3	3	4	4	7	8

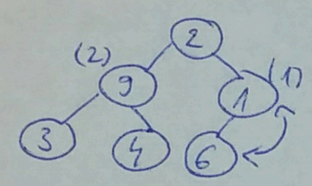
pl. 2.: | 2 | 9 | 1 | 3 | 4 | 6 |



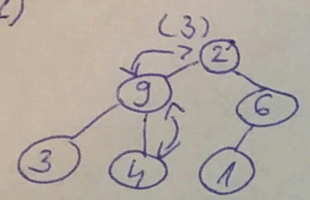
→ kupacá alakítás

$k = \lfloor \frac{n}{2} \rfloor = 3$

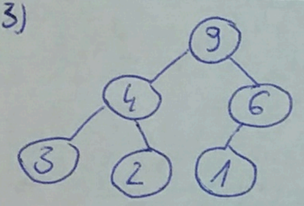
1)



2)

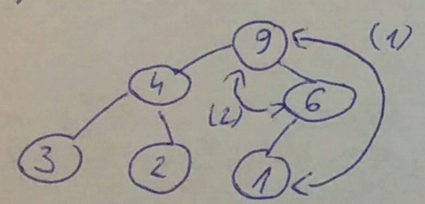


3)

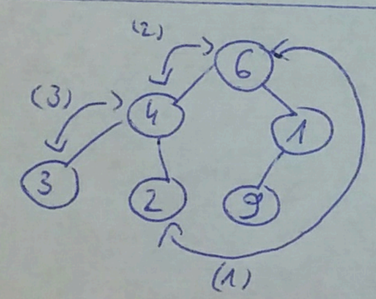


kupacca lett alakítva

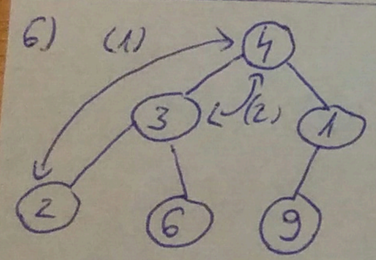
4) Rendezés:



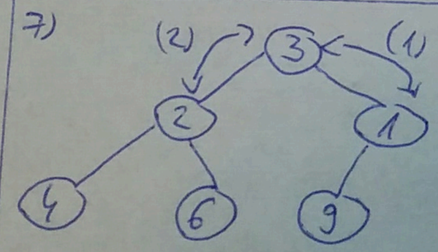
5)



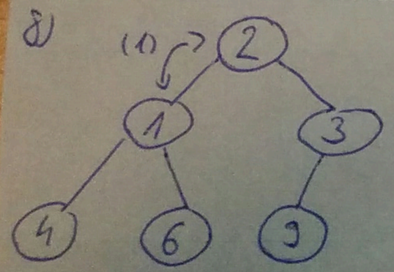
6)

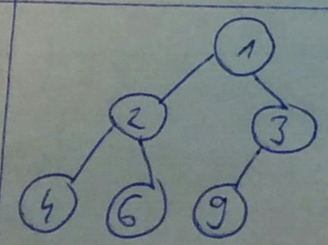


7)



8)





kész a rendezés

| 1 | 2 | 3 | 4 | 6 | 9 |

(8) Tétel: AVL fák, műveleteik, kiegyensúlyozás, magasság

AVL-fa: Magasság szerint kiegyensúlyozott bin. kereső fa.

Magasság szerint kiegy.: ha $\forall$ csúcsa kiegyensúlyozott

Kiegyensúlyozott csúcs: p csúcs kiegyensúlyozott $|p\to s| \leq 1$,

egyensúly számítás: $p\to s = h(t\to jobb) - h(t\to bal)$

AVL fákat láncoltan repr.-jük.

Csúcsokban s egyensúly attribútum: $s = -1, 0, 1$

Méret és magasság összefüggés:

$n>0$ a csúcsok száma, $h \geq 0$ a magasság

$$\lfloor \lg n \rfloor \leq h \leq 1.45 \cdot \lg n$$

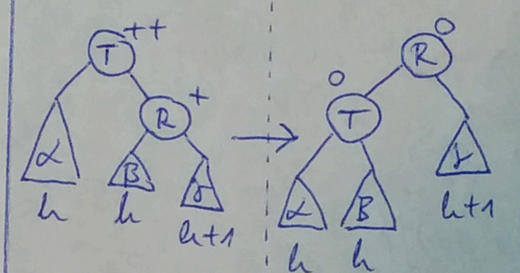
Mivel AVL-fák magassága $\Theta(\lg n)$ ezért: keres
min
beszúr $MTE_{(n)}$ $\Theta(\lg n)$
töröl
minKiver

AVL kiegyensúlyozások

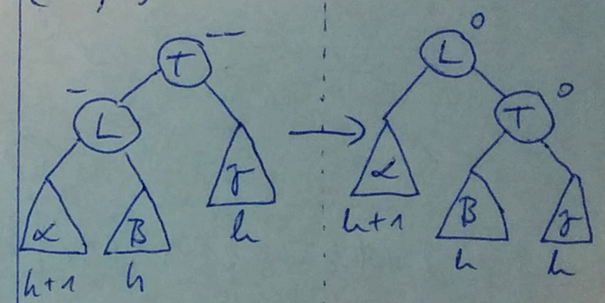
Beszúrás után csak: (++,+), (++,−), (−−,−), (−−,+)

beszúr + kiegy. visszaállítja $h(t)$-t

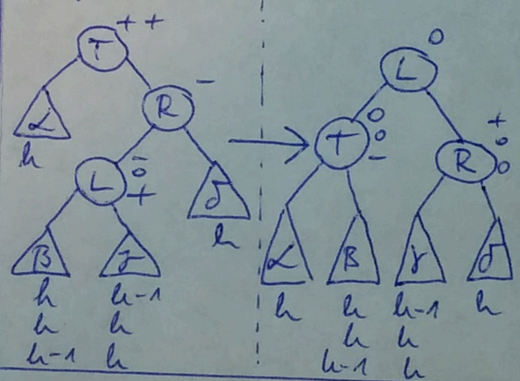
(++,+) séma



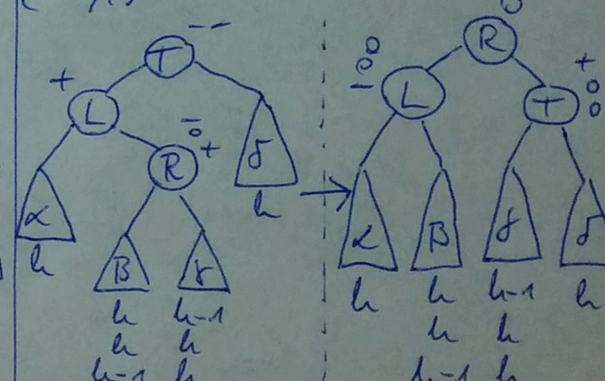
(−−,−)



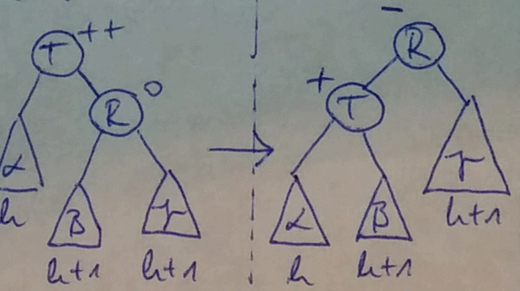
(++,−)



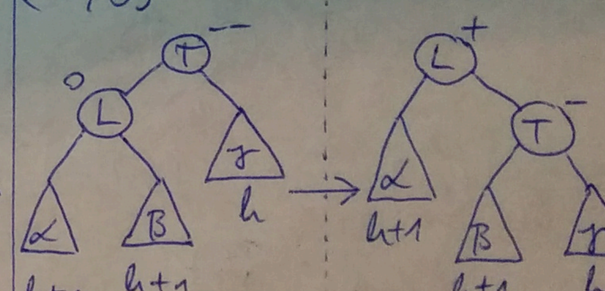
(−−,+)



(++,0)



(−−,0)



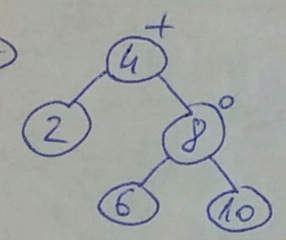
törlés + kiegy. (++,+), (++,−), (−−,−), (−−,+) csökkenti $h(t)$-t

törlés + kiegy. (++,0), (−−,0) megtartja $h(t)$-t

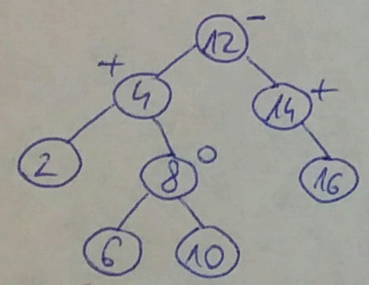
AVL-fa leírjolása:

1) ((2)4((6)8(10)))

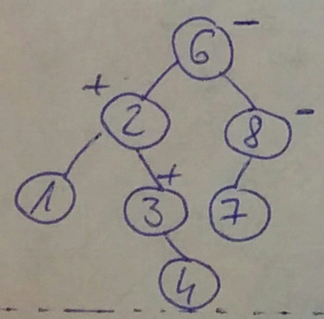
⟹



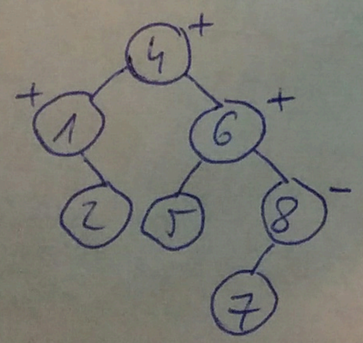
2) {[(2)4({6}8{10})]12[14(16)]}



3) {[(1)2(3{4})]6[(7)8]}

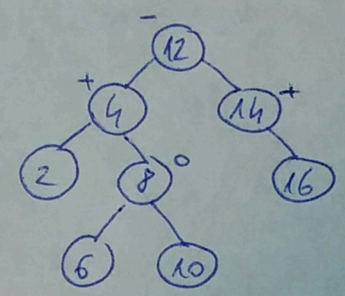


4) {[1(2)]4[(5)6({7}8)]}

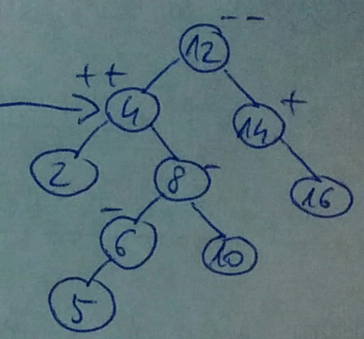


AVL beszúrás

1) Az 5 beszúrása:

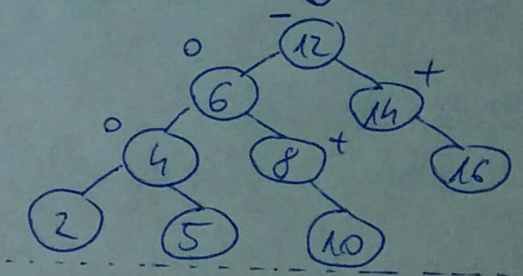


Beszúrás után ez a fa adódik

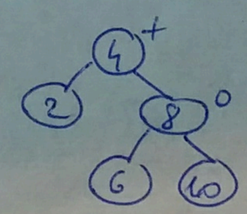


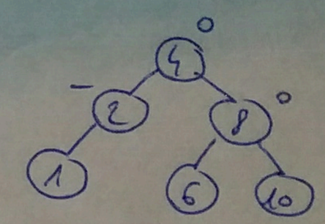
Itt kell egyensúlyozni (++, −) szerint

A (++,−) s. alk. után:



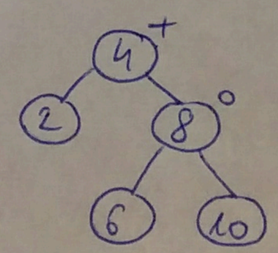
2) Az 1 beszúrása:

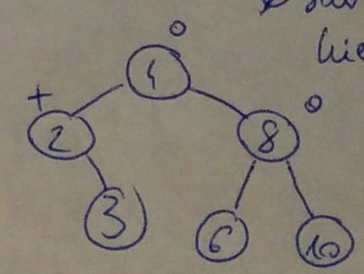




Ø szükség kiegyen.-re

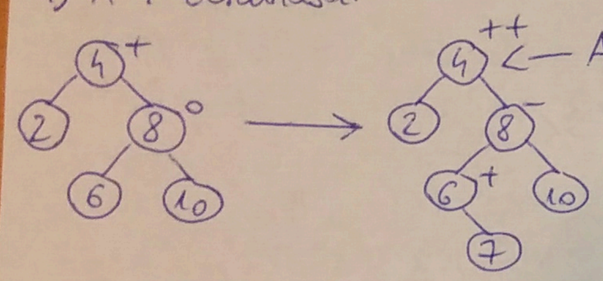
3) A 3 beszúrása:





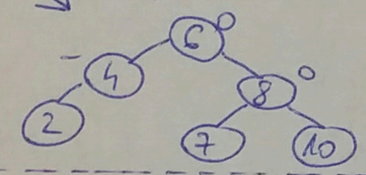
Ø szükség kiegyen.-ra

4) A 7 beszúrása:

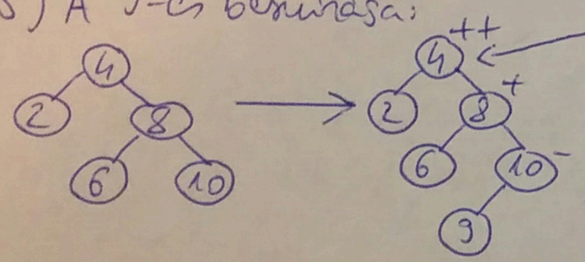


A 7-es beszúrása után, itt ki kell egyensúlyozni a (++,-) párral!

(++,-) után:

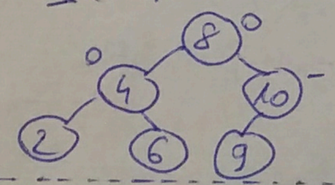


5) A 9-es beszúrása:

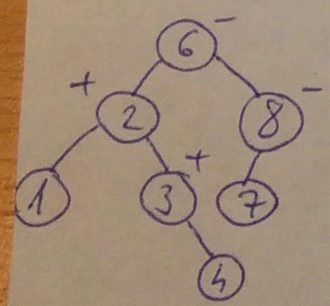


A 9 beszúrása után itt ki kell egyensúlyozni a (++,+) párral!

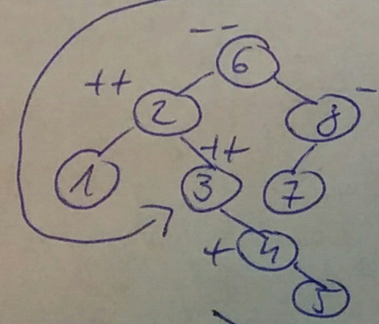
(++,+) után:



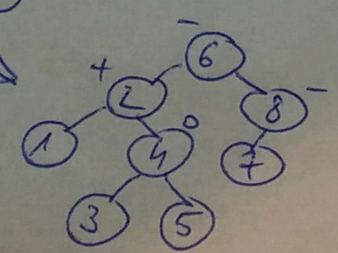
6) Az 5 beszúrása:



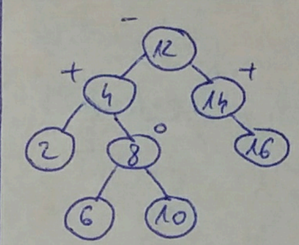
Itt beszúrás után ki kell egyens.-ni a (++,+) párral



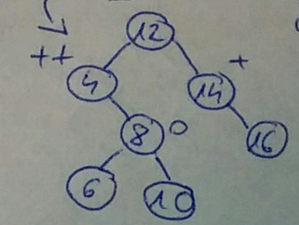
(++,+) után:



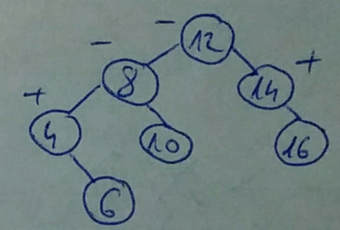
AVL minkiven



minkiven (2) után itt ki kell egyensúlyozni a (++,0) párral

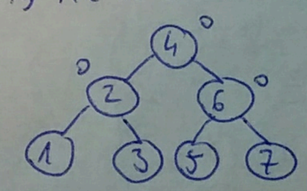


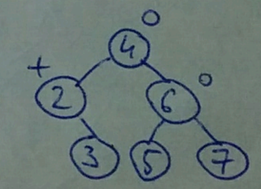
(++,0) után



AVL törlése

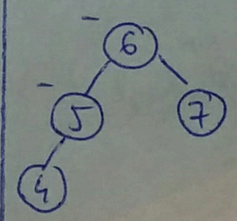
1) Az 1-es törlése

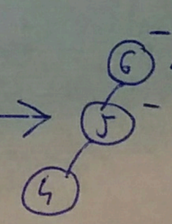




∅ kell kiegy.

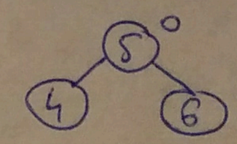
2) A 7-es törlése:



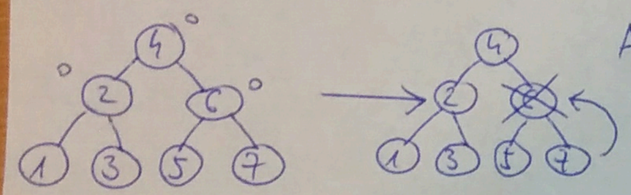


7-es törlése után ki kell egyensúlyozni (--,-) párral.

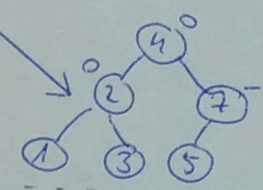
(--,-) után:



3) A 6-os törlése:

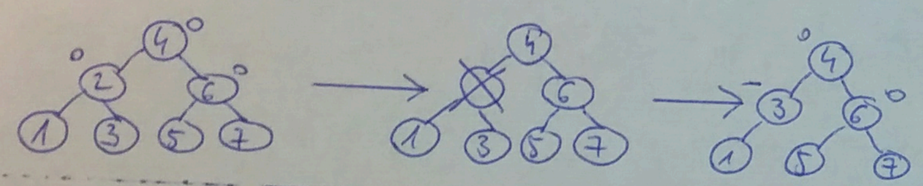


A törölt elem jobb oldali részfájának a legkisebb eleme kell felkerüljön.



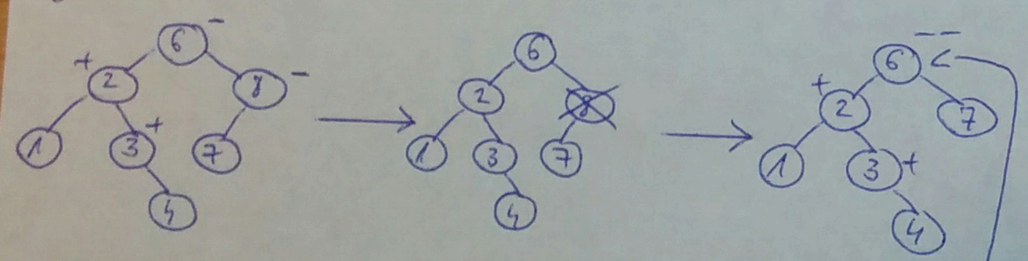
Ø kell kicserélni.

4) A 2-es törlése:



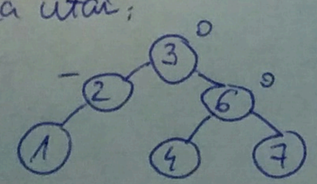
Ø kell kicserélni.

5) A 8-as törlése:

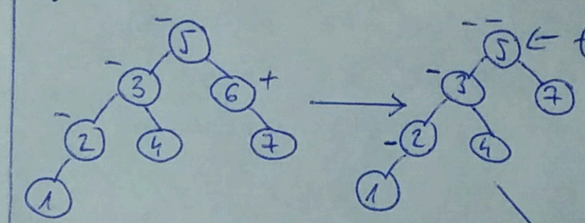


törlés után itt ki kell egyen.-ni (−−,+) sémával.

(−−,+) séma után:

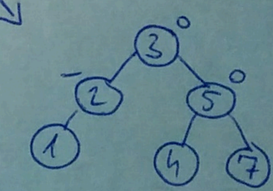


6) A 6-os törlése:

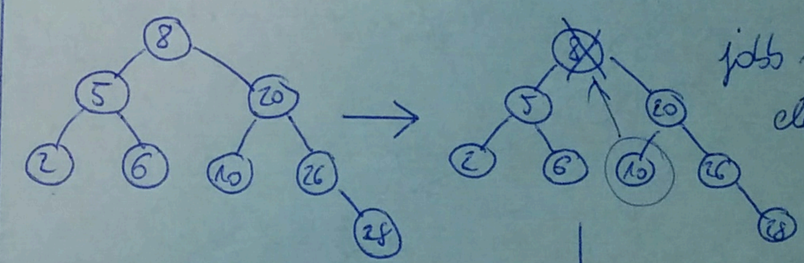


← törlés után ki kell egyensúlyozni a (−−,−) sémával.

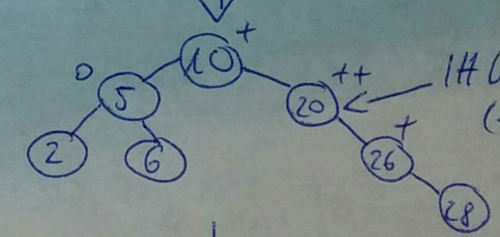
(−−,−) séma után:



7) A 8-as törlése:

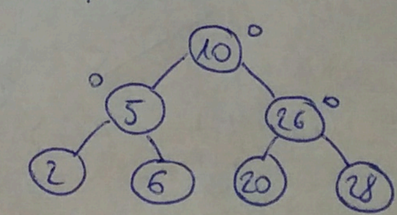


jobb részfa min eleme megy fel a 8-as helyére



itt ki kell egyen.-ni (++,+) sémával

(++,+) után:

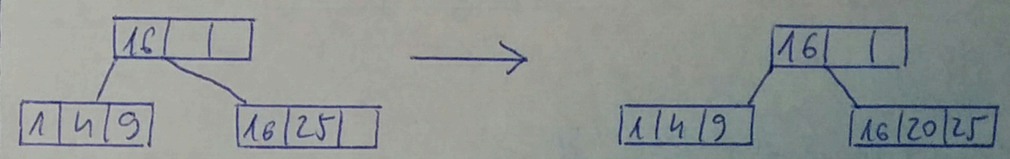


(10) Tétel B+ fa és műveletei

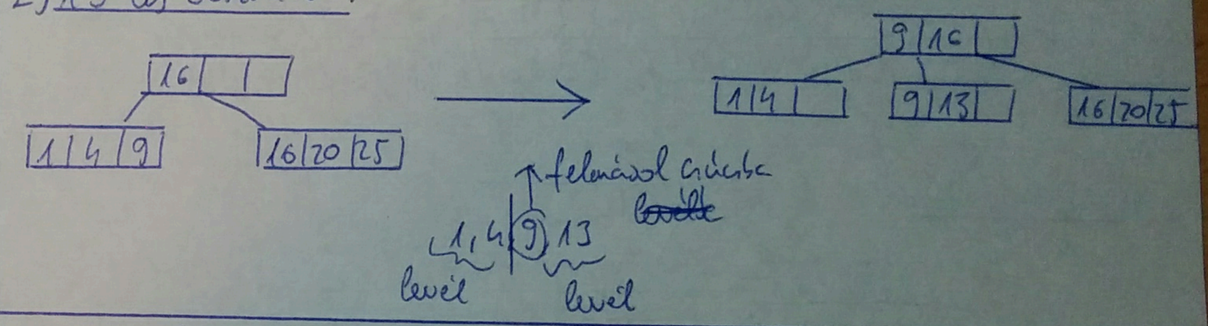
B+ fa:
- d a mutatók száma
- ∀ csúcsnak max d-1 kulcsa van
- gyökérnek legalább 2 gyereke van (kivéve ha egyetlen csúcs)
- ∀ belső csúcsnak legalább $\lfloor \frac{d}{2} \rfloor$ gyereke van
- ∀ levél legalább $\lfloor \frac{d}{2} \rfloor$ kulcsot tartalmaz
- kulcsok balról jobbra monot. növ.
- fa magassága $O(\lg n)$

Beszúrás:

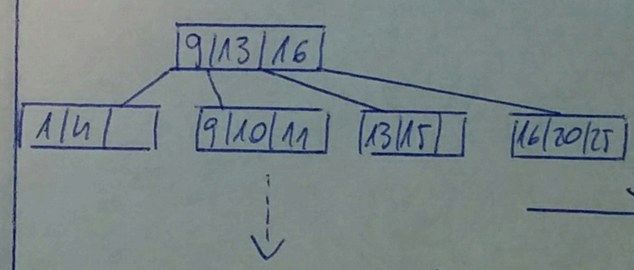
1) 20-as beszúrása:



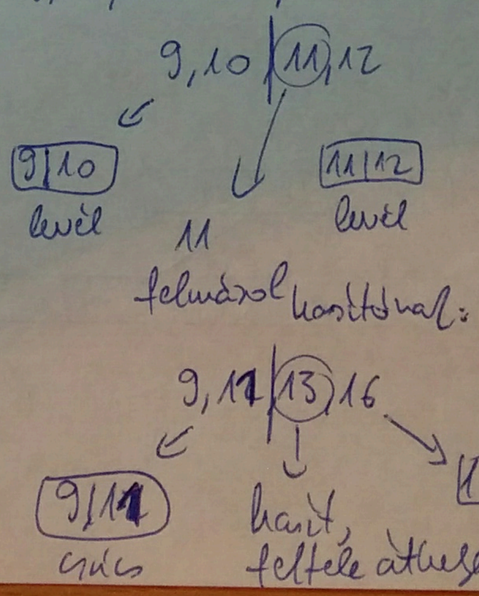
2) 13-as beszúrása:

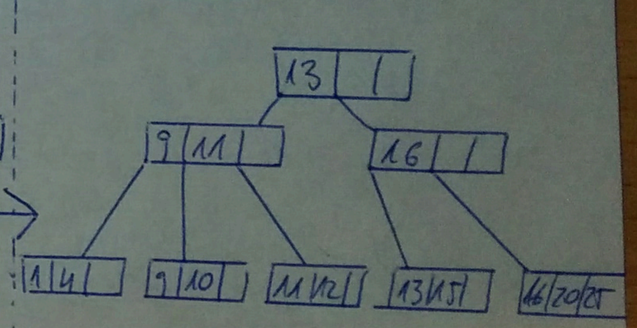


3) 12-es beszúr:

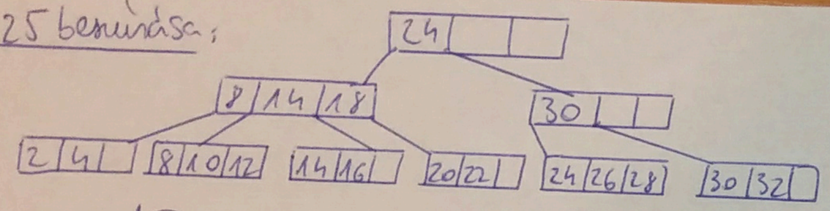


9,10,11 közé jön a 12-es:



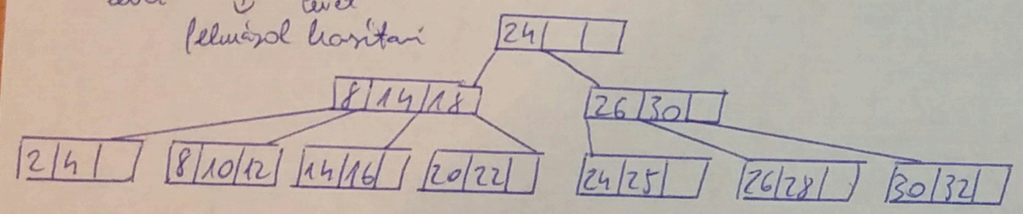


4) 25 beszúrása:

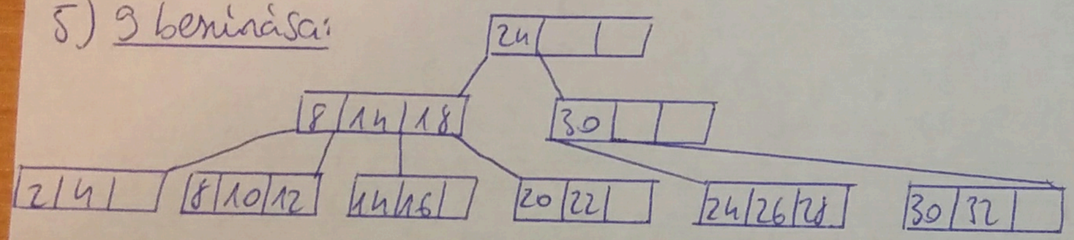


24, 25 | 26 | 28

level — felmászol kavitáni — level



5) 9 beszúrása:



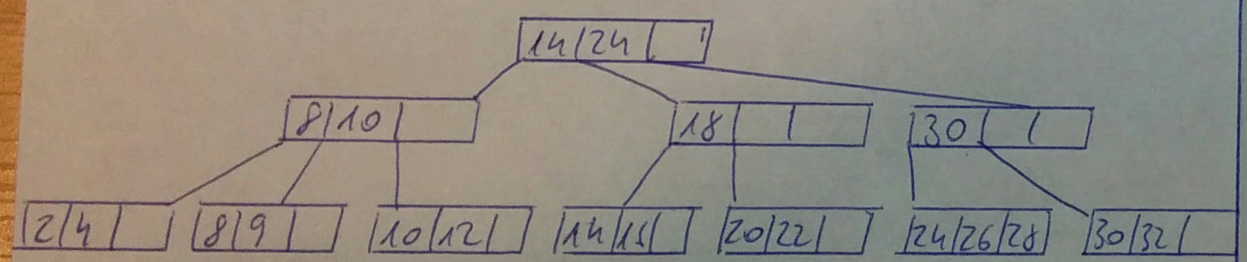
8, 9 | 10 | 12

level — level

kavit → 8, 10 | 14 | 18

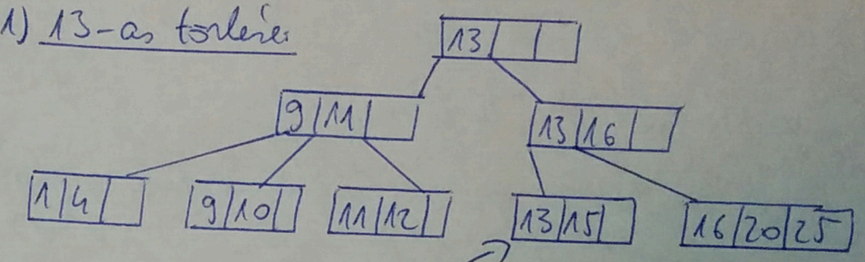
csúcs — csúcs

kavit

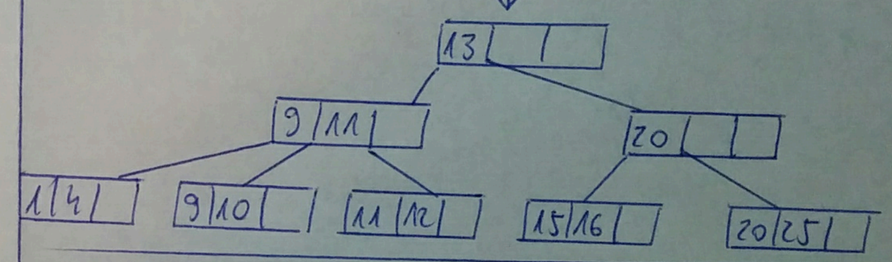


Törlés:

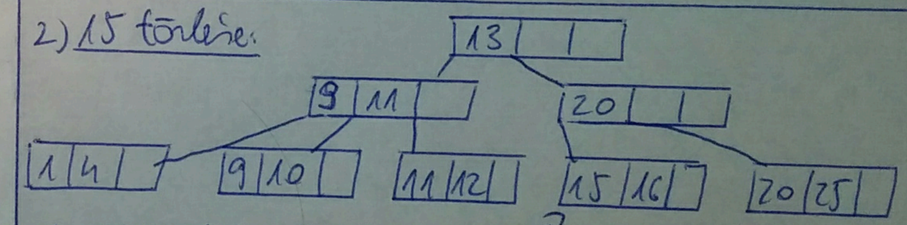
1) 13-as törlése:



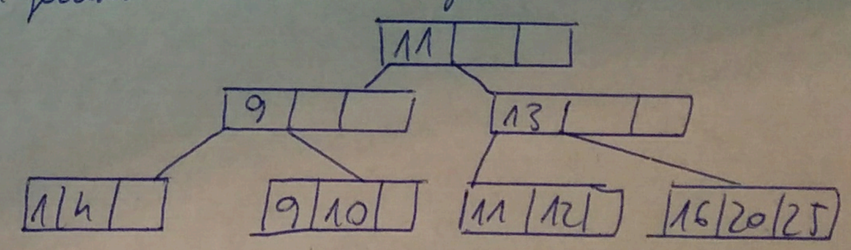
Ha a 13 kiesik akkor 15 egyedül marad, megkapja a testvérétől 16-ot, a 20-as felkerül kulcs-nak



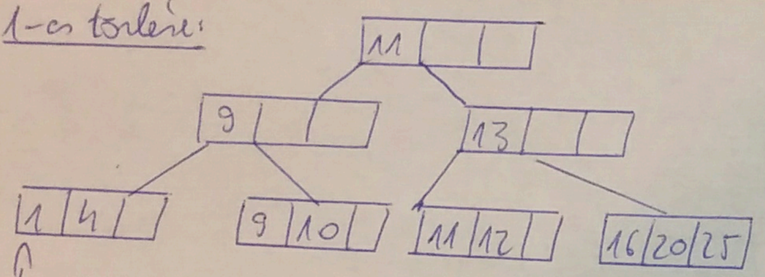
2) 15 törlése:



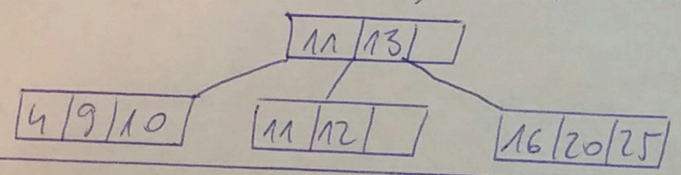
A 15 törlése után 16 egyedül marad, átrakunk a testvérnél és felette levő csúcsnak csak 1 eleme van, ezért az meg elkér egy elemet a testvérétől, tehát a 20-as kavitó nem jó, ezért 11 felmászik és 13 atúszik helyre.



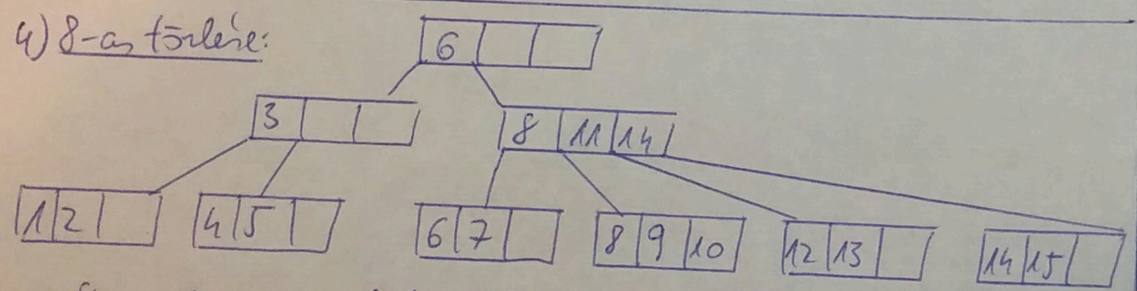
3) 1-es törlése:



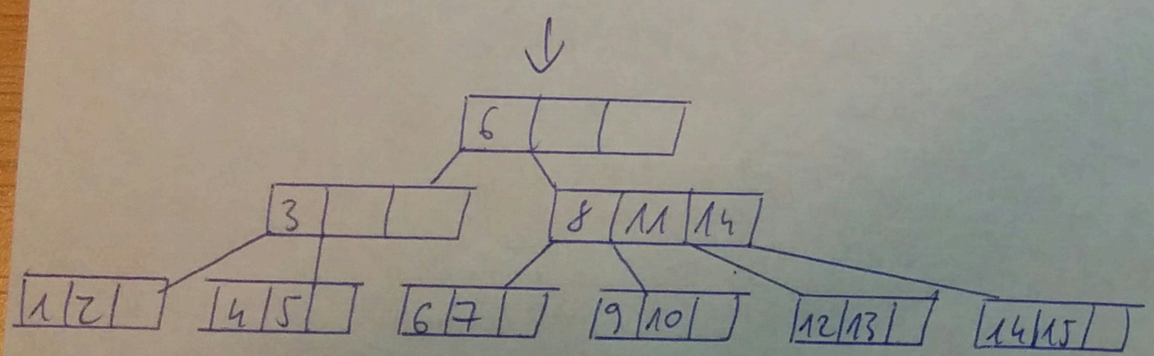
1-es törül, egyedül marad. (4) összevon (9,10)-el. Felette levő (9)-nek csak egy gyereke maradt, testvére tud lenni, ezért 9-es leverve, 11+13 összevon új gyökérnek.



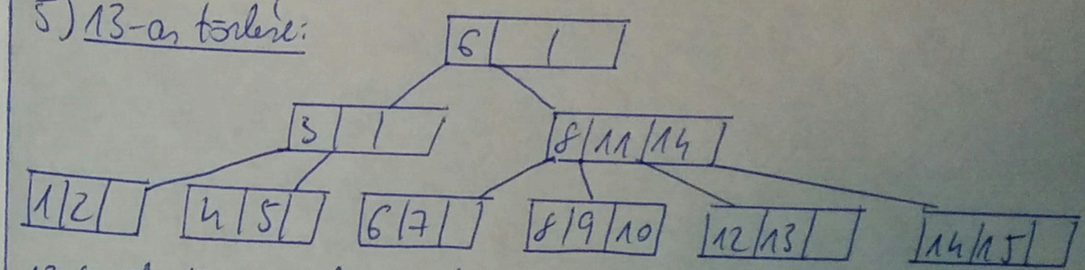
4) 8-as törlése:



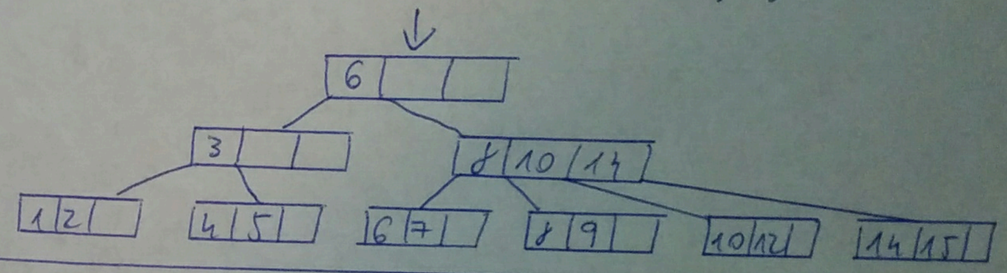
8-as törül, a levélben levő 9,10 baba marad



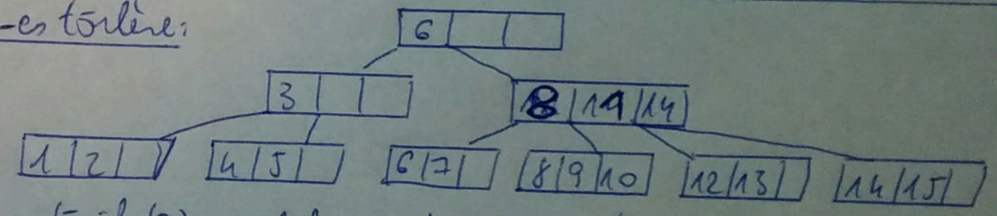
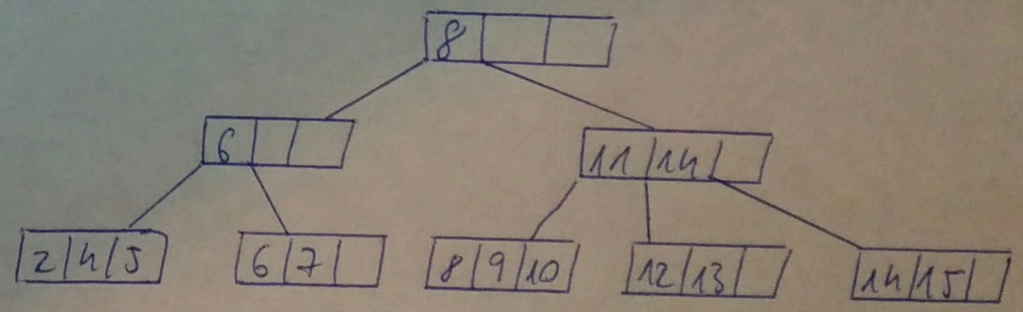
5) 13-as törlése:



13 törül, 12 egyedül marad, bal testvére 10 átmegy hozzá, emiatt 11-es kulcs nem jó, helyette 10-es lesz (8,10,14)



6) 1-es törlése:



1-es törül (2) egyedül marad, összevon testvével lesz (2,4,5) levél, felette levő (3)-asnak csak 1 gyereke maradt, testvére elkéri egyet, de a 3-as hasító szerv, ezért 6 lejön a gyökérből és helyére 8 megy