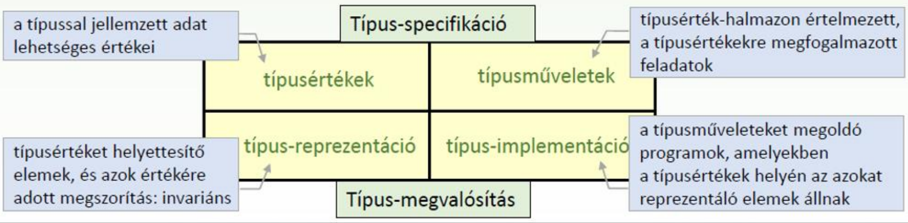
Adattípus fogalma (1. előadás)

- Típus-specifikáció
  - Típusértékek
  - Típusműveletek
- Típus megvalósítás
  - Típus-reprezentáció
  - Típus-implementáció

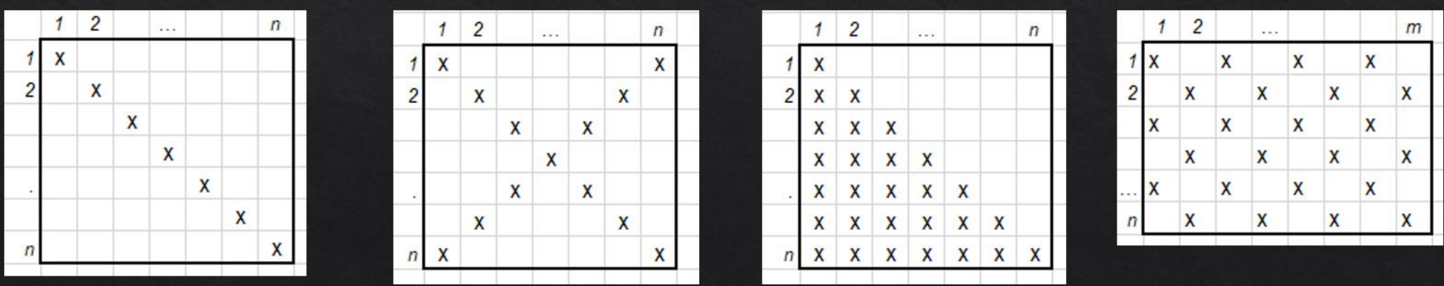
Adattípus fogalma

- Egy adat (változó) típusának definiálásához szükség van a típus specifikációjára és annak megvalósítására.
- A típus-specifikáció megadja:
  - az adat által felvehető értékek halmazát
  - a típusértékekkel végezhető műveleteket
- A típus-megvalósítás megmutatja:
  - hogyan ábrázoljuk (reprezentáljuk) a típus értékeit
  - milyen programok helyettesítsék (implementálják) a műveleteket



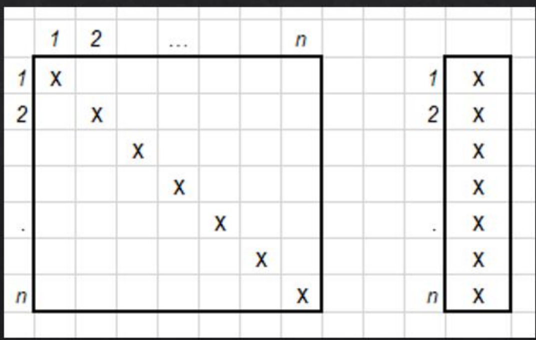
Ritka mátrixok helytakarékos ábrázolása

- Ritka mátrixok:
  - Bizonyos elemek mindig nulla értékűek.
  - A „nem nulla” elemek szabályos elhelyezkedésűek.
- Helytakarékos ábrázolás:
  - Nagy méret esetén érdemes csak azokat a mátrix elemeket tárolni, amelyek nullától különböző értéket (is) felvehetnek.



Diagonális mátrix

- Négyzetes mátrix
- Csak a főátlójában szerepelhetnek nullától különböző elemek:
  - $A[i,j] = 0, \quad \forall\ 1 \leq i,j \leq n,\ i \neq j$ esetén
  - $A[i,j] \neq 0, \quad \forall\ 1 \leq i,j \leq n,\ i = j$ esetén
- Tárolás: n méretű egydimenziós tömbben lehetséges:



$\text{index}(i,j) = i$

Diagonális mátrix

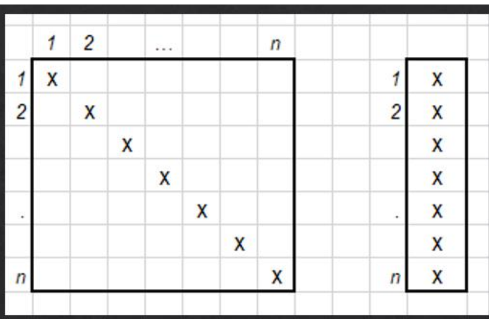
- Típus specifikáció (típus értékek, műveletek)

$Diag(\mathbb{R}^{n \times n})$

c := a+b	(a, b, c : $Diag(\mathbb{R}^{n \times n})$)	
c := a·b	(a, b, c : $Diag(\mathbb{R}^{n \times n})$)	
e := a[i,j]	(a : $Diag(\mathbb{R}^{n \times n})$, i,j : $\mathbb{N}$, e : $\mathbb{R}$)	
a[i,j] := e	(a : $Diag(\mathbb{R}^{n \times n})$, i,j : $\mathbb{N}$, e : $\mathbb{R}$)	//ha i=j

Lekérdezés: e := a[i,j] Értékadás: a[i,j] := e ha i=j

Diagonális mátrix

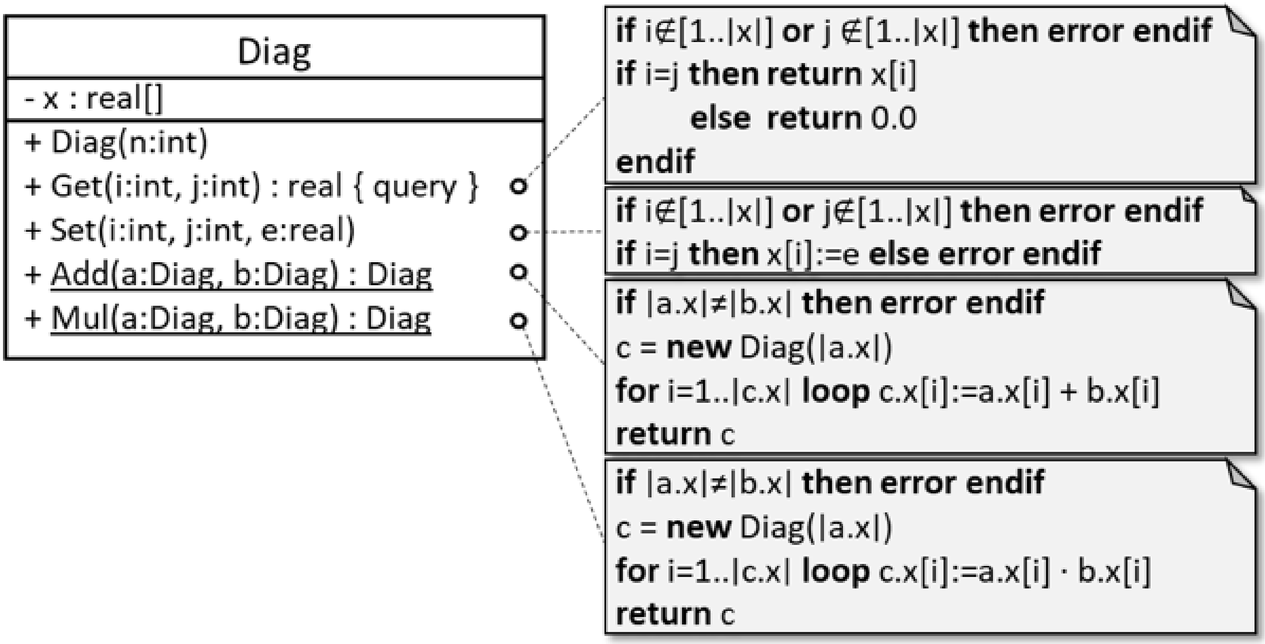


- Típus reprezentáció (típus értékek, műveletek)

Diagonális mátrix típus

$Diag(\mathbb{R}^{n\times n})$	c := a+b (a, b, c : $Diag(\mathbb{R}^{n\times n})$)
	c := a·b (a, b, c : $Diag(\mathbb{R}^{n\times n})$)
	e := a[i,j] (a : $Diag(\mathbb{R}^{n\times n})$, i,j : $\mathbb{N}$, e : $\mathbb{R}$)
	a[i,j] := e (a : $Diag(\mathbb{R}^{n\times n})$, i,j : $\mathbb{N}$, e : $\mathbb{R}$) //ha i=j
$x:\mathbb{R}^n$	$\forall i \in [1..n]$: c.x[i] := a.x[i]+b.x[i]
	$\forall i \in [1..n]$: c.x[i] := a.x[i]·b.x[i]
	ha i=j akkor e := a.x[i] különben e := 0.0
	ha i=j akkor a.x[i] := e

Osztály diagram:



Alsó háromszög mátrix

Alsó háromszög mátrix

- Valósítsuk meg az alsó háromszög mátrix típust (a mátrixok a főátlójuk felett csak nullát tartalmaznak)! Ilyenkor elegendő csak a főátló és az alatti elemeket reprezentálni egy sorozatban. Implementáljuk a mátrix i-edik sorának j-edik elemét visszaadó műveletet, valamint két mátrix összegét és szorzatát!

7 x 7 -es alsó háromszög mátrix

	1	2	3	4	5	6	7
1	1						
2	2	3					
3	4	5	6				
4	7	8	9	10			
5	11	12	13	14	15		
6	16	17	18	19	20	21	
7	22	23	24	25	26	27	28

A sorszámok mutatják, hogy hányadik lesz a mátrix elem a tömbben.

A mátrix elhelyezése sorfolytonosan egy egydimenziós tömbben
(a biztosan nulla értékű elemeket felesleges tárolni)

1	[1,1]	
2	[2,1]	
3	[2,2]	
4	[3,1]	
5	[3,2]	
6	[3,3]	
27	[7,6]	
28	[7,7]	
29	0	*a nulla értéket is tárolhatjuk egy példányban*

$A[i,j] \neq 0$, ha $1 \leq j \leq i \leq n$

$X[n(n+1)/2]$

Alsó háromszög mátrix

- Index függvény

7 x 7 -es alsó háromszög mátrix

	1	2	3	4	5	6	7
1	1						
2	2	3					
3	4	5	6				
4	7	8	9	10			
5	11	12	13	14	15		
6	16	17	18	19	20	21	
7	22	23	24	25	26	27	28

A sorszámok mutatják, hogy hányadik lesz a mátrix elem a tömbben.

A mátrix elhelyezése sorfolytonosan egy egydimenziós tömbben
(a biztosan nulla értékű elemeket felesleges tárolni)

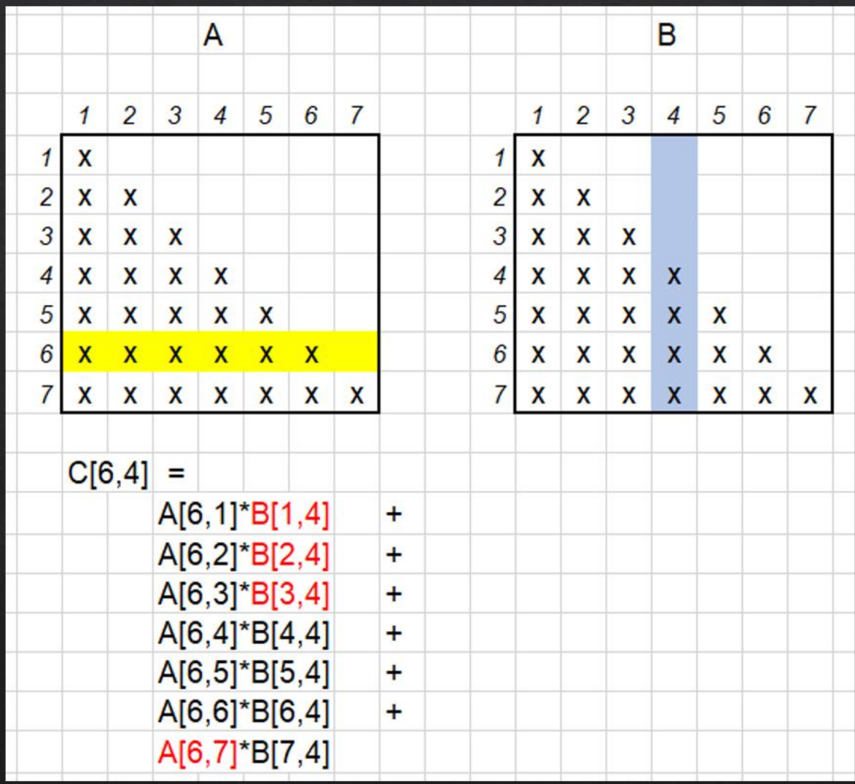
1	[1,1]	
2	[2,1]	
3	[2,2]	
4	[3,1]	
5	[3,2]	
6	[3,3]	
27	[7,6]	
28	[7,7]	
29	0	*a nulla értéket is tárolhatjuk egy példányban*

$$ind(i,j) = j + \sum_{k=1}^{i-1} k = j + \frac{i(i-1)}{2} \quad ,ha\ 1 \leq j \leq i \leq n$$

Próbák:
i=1, j=1 index(1,1)=1
i=4, j=2 index(4,2)=8
i=7, j=7 index(7,7)=28

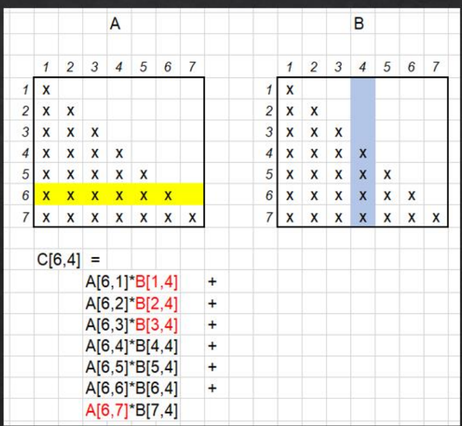
Alsó háromszög mátrix

- Szorzás művelet



Alsó háromszög mátrix

- Szorzás művelet



$\forall i,j \in [1..n]$: ha $i \geq j$ akkor

$$c.x\left[\frac{i(i-1)}{2}+j\right] := \sum_{k=j}^{i} a.x\left[\frac{i(i-1)}{2}+k\right] \cdot b.x\left[\frac{k(k-1)}{2}+j\right]$$

Alsó háromszög mátrix típus

AHM($\mathbb{R}^{n\times n}$)	c := a+b (a, b, c : AHM($\mathbb{R}^{n\times n}$)) // azonos n-re
	c := a*b (a, b, c : AHM($\mathbb{R}^{n\times n}$)) // azonos n-re
	e := a[i,j] (a : AHM($\mathbb{R}^{n\times n}$), i,j : $\mathbb{N}$, e : $\mathbb{R}$)
	a[i,j] := e (a : AHM($\mathbb{R}^{n\times n}$), i,j : $\mathbb{N}$, e : $\mathbb{R}$) //ha i ≥ j
x:$\mathbb{R}^{n(n+1)/2}$	$\forall$i$\in$[1..n(n+1)/2]: c.x[i] := a.x[i]+b.x[i]
	$\forall$i,j$\in$[1..n]: ha i≥j akkor $c.x\left[\frac{i(i-1)}{2}+j\right] := \sum_{k=j}^{i} a.x\left[\frac{i(i-1)}{2}+k\right]\cdot b.x\left[\frac{k(k-1)}{2}+j\right]$
	ha i≥j akkor e := $a.x\left[\frac{i(i-1)}{2}+j\right]$ különben e := 0.0
	ha i≥j akkor $a.x\left[\frac{i(i-1)}{2}+j\right]$:= e

Logaritmikus keresés

Logaritmikus keresés

- Rendezett sorozatban nagyon hatékony keresést biztosít.
- Elfelezve a keresés intervallumát, a középső elemmel összehasonlítja a keresett kulcsot, három eset lehetséges: [ah…fh] ind=(ah+fh) div 2 //egész osztás!
  - Keresett kulcs kisebb – bal oldali részben folytatja a keresést: [ah…ind-1]
  - Keresett kulcs nagyobb – jobb oldali részben folytatja a keresést [ind+1…fh]
  - Egyenlő – megtaláltuk, készen vagyunk: ind a keresett elem indexe
  - Ha leszűkítés után kapott intervallum üres, akkor a keresés véget ért, sikertelen volt

Logaritmikus keresés

- Keresett kulcs megtalálható:

Keressük meg: 8

1	2	3	4	5	6	7	8	9	10	11		ah	fh	ind
2	4	5	7	8	11	15	21	23	25	30		1	11	6
ah					ind					fh				

1	2	3	4	5	6	7	8	9	10	11		ah	fh	ind
2	4	5	7	8	11	15	21	23	25	30		1	5	3
ah		ind		fh										

1	2	3	4	5	6	7	8	9	10	11		ah	fh	ind
2	4	5	7	8	11	15	21	23	25	30		4	5	4
			ah	fh										
			ind											

1	2	3	4	5	6	7	8	9	10	11		ah	fh	ind
2	4	5	7	8	11	15	21	23	25	30		5	5	5
				ah,fh										
				ind										

Logaritmikus keresés

- Keresett kulcs nem található:

Keressük meg: 22

1	2	3	4	5	6	7	8	9	10	11		ah	fh	ind
2	4	5	7	8	11	15	21	23	25	30		1	11	6
ah					ind					fh				

1	2	3	4	5	6	7	8	9	10	11		ah	fh	ind
2	4	5	7	8	11	15	21	23	25	30		7	11	9
						ah		ind		fh				

1	2	3	4	5	6	7	8	9	10	11		ah	fh	ind
2	4	5	7	8	11	15	21	23	25	30		7	8	7
						ah	fh							

1	2	3	4	5	6	7	8	9	10	11		ah	fh	ind
2	4	5	7	8	11	15	21	23	25	30		8	8	8
							ah,fh							
							ind							

1	2	3	4	5	6	7	8	9	10	11		ah	fh	ind
2	4	5	7	8	11	15	21	23	25	30		9	8	

Logaritmikus keresés algoritmusa

Specifikáció:

$A = (m:\mathbb{Z},\ n:\mathbb{Z},\ h:H,\ l:\mathbb{L},\ ind:\mathbb{Z})$

$Ef = (\ m=m' \wedge n=n' \wedge h=h' \wedge \forall j\in[m..n\text{-}1]: f(j) \leq f(j+1)\ )$

$Uf = (\ Ef \wedge l=(\ \exists j\in[m..n]; f(j)=h\ ) \wedge l \rightarrow (\ ind\in[m..n] \wedge f(ind)=h\ )\ )$

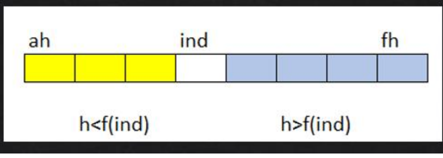
Algoritmus:

$ah, fh, l := m, n, hamis$ $\qquad ah, fh:\mathbb{Z}$

$\neg l \wedge ah \leq fh$

$ind := (ah+fh)\ div\ 2$

$f(ind)>h$	$f(ind)<h$	$f(ind)=h$
$fh := ind\text{-}1$	$ah := ind+1$	$l := igaz$



Zsák típus

Valósítsuk meg egy adott halmaz (E) elemeit tartalmazó zsák típusát úgy, hogy nincs felső korlát a zsákba bekerülő elemek számára. A szokásos (üres-e, betesz, kivesz, hányszor van benn egy szám) műveletek mellett szükségünk lesz a leggyakoribb elemet lekérdező műveletre is.

Bag azon zsákok halmaza, amelyek elemei (E) rendezhetőek	b := SetEmpty(b)	b : Bag	*// kiüríti a zsákot*
	l := Empty(b)	b : Bag, l : $\mathbb{L}$	*// üres-e zsák*
	c:= Multipl(b, e)	b : Bag, e : E, c : $\mathbb{N}$	*// elem multiplicitása*
	b := Insert(b, e)	b : Bag, e : E	*// elemet tesz be*
	b:= Remove(b, e)	b : Bag, e : E	*// elemet vesz ki*
	m:= Max(b)	b : Bag, m : E	*// leggyakoribb elem*

seq : Pair*

maxind : ℕ

ahol

Pair =
rec(data: E, count: ℕ)

Invariáns:

- a seq-ben az elemeket tartalmuk (data) szerint rendezetten tároljuk
- a maxind a nem üres seq sorozat legnagyobb count értékű elemének indexe

b := SetEmpty(b) b : Bag

seq := <>

l := Empty(b) b : Bag, l : 𝕃

l := |seq|=0

c := Multipl(b, e) b : Bag, e : E, c : ℕ

l, ind := logSearch(seq, e)
if l **then** c :=seq[ind].count

m := Max(b) b : Bag, m : E

|seq| > 0	
m := seq[maxind].data	hiba

b := Insert(b,e) b : Bag, e : E

<table>
<tr><td colspan="5">l, ind := logSearch(seq, e)</td></tr>
<tr><td colspan="5">l</td></tr>
<tr><td colspan="2">++seq[ind].count</td><td colspan="3">seq := seq[1..ind-1] ⊕ <(e,1)> ⊕ seq[ind..|seq|]</td></tr>
<tr><td colspan="2">seq[ind].count > seq[maxind].count</td><td>|seq|=1</td><td>|seq|>1 ∧ maxind≥ind</td><td>else</td></tr>
<tr><td>maxind := ind</td><td>–</td><td>maxind := 1</td><td>++maxind</td><td>–</td></tr>
</table>

⟐ Logaritmikus keresés, van-e zsákban már „e” érték?

⟐ Igen: count-ot növeljük eggyel, majd ellenőrizzük, hogy kell-e módosítani a leggyakoribb elem indexét.

⟐ Nem: beszúrjuk az elemet a rendezettség szerinti helyére. Ha maxind egy nagyobb indexű elemre mutat, akkor eggyel növelni kell

Elem szerinti keresés [átkerülhet a következő gyakorlat elejére]

$A = (seq : Pair^*,\ e : E,\ l : \mathbb{L},\ ind : \mathbb{N})$

$Ef = (seq=seq_0 \ \wedge\ e=e_0 \ \wedge\ \forall i\in[1\ ..\ |seq|-1] : seq[i].data < seq[i+1].data\ )$

$Uf = (Ef \ \wedge\ l = \ \exists i\in[1\ ..\ |seq|] : seq[i].data = e \wedge$
$\quad (\ \ l \ \rightarrow\ ind\in[1\ ..\ |seq|] \ \wedge\ seq[ind].data = e\ ) \wedge$
$\quad (\neg l \ \rightarrow\ \forall i\in [1..ind-1]: seq[i].data < e \ \wedge\ \forall i\in[ind..\ |seq|] : seq[i].data > e)\ \ )$

l, ind := logSearch (seq, e)

<table>
<tr><td colspan="4">l, ah, fh := hamis, 1, |seq|</td></tr>
<tr><td colspan="4">¬l ∧ ah ≤ fh</td></tr>
<tr><td rowspan="3"></td><td colspan="3">ind := ⌊(ah + fh) / 2⌋</td></tr>
<tr><td>seq[ind].key > key</td><td>seq[ind].key = key</td><td>seq[ind].key < key</td></tr>
<tr><td>fh := ind-1</td><td>l := igaz</td><td>ah := ind+1</td></tr>
<tr><td colspan="4">¬l</td></tr>
<tr><td colspan="2">ind := ah</td><td colspan="2">—</td></tr>
</table>

ah, fh : $\mathbb{N}$

b := Remove(b,e) b : Bag, e : E

l, ind := logSearch(seq, e) // data szerint

l

seq[ind].count > 1	seq[ind].count = 1	
seq[ind].count := seq[ind].count–1	seq := seq[1..ind-1] $\oplus$ seq[ind+1..|seq|]	–

|seq|>0

max, maxind := $MAX_{i=1..\|seq\|}$ (seq[i].count)	–

◈ Logaritmikus keresés, van-e zsákban „e” érték?

◈ Van, és több mint egy: count-ot csökkentjük eggyel.

◈ Van, de csak egy: az adott elmet kihagyjuk a sorozatból.

◈ Nincs: hatástalan.

◈ Ha volt törlés, a leggyakoribb elem megváltozhatott, frissíteni kell.

Osztály:

```
l, ind := LogSearch(e)
if l then  return seq[ind].count
else return 0
endif
```

```
if |seq|>0 then return seq[maxind].data endif
```

```
Bag
- seq : Element[]
- maxind : int
+ SetEmpty()                              seq := <>
+ Empty() : bool {query}                  return |seq|=0
+ Multipl(e : E) : int {query}
+ Max() : E {query}
+ Insert(e: E)
+ Remove(e: E)
- LogSearch(e:E) : (bool,int)  {query}
```

```
l, ind := LogSearch(e)
if l then
   ++seq[ind].count
   if seq[ind].count > seq[maxind].count then maxind := ind endif
else
   seq := seq[1..ind-1] ⊕ <(e,1)> ⊕ seq[ind+1..|seq|]
   if |seq|=1 then maxind := 1
   elsif maxind>ind then ++maxind
   elseif skip
   endif
endif
```

```
l, ind := LogSearch(e)
if l then
   if seq[ind].count > 1 then
      - - seq[ind].count
   elsif seq[ind].count  = 1 then
      seq := seq[1..ind-1] ⊕ seq[ind+1..|seq|]
   endif
   if |seq|>0 then max, maxind := MAX_i=1..|seq| (seq[i].count) endif
endif
```

```
l, ah, fh := false, 1, |seq|
while not l and ah ≤ fh loop
   ind := ⌊(ah + fh) / 2⌋
   if      seq[ind].data > e then  fh := ind-1
   elsif seq[ind].data = e  then     l := true
   elsif seq[ind].data < e then ah := ind+1
endloop
if not l then ind := ah endif
return (l, ind)
```