Adatszerkezet - műveletek

adatszerkezet létrehozása

adat felvétele

adat keresése

adat módosítása

adat törlése

elemszám visszaadása

minden adat törlése (üresít)

adatszerkezet felszámolása (megszüntet)

+ speciális (adatszerkezet-specifikus) műveletek:

pl.: adatszerkezetet kettévág,
2 adatszerkezetet egyesít...

Halmaz, függvény megvalósítása

AVL fák, B-fák, 2-3 fák, Piros-fekete fák, Hasítótáblázatok

Asszociatív tömb megvalósítása

Rendezettminta-fák (Piros-fekete rendezett-minta fa)

(Intervallum-halmaz megvalósítása)

Intervallum-fák (Piros-fekete intervallum fa)

Vágható, egyesíthető halmaz adattípus megvalósítása

Önszervező bináris keresőfák

Sor, Verem megvalósítása

Láncolt listák

Prioritási sor megvalósítása

Prioritási sor műveletek:

void s.sorba(T x)

T s.sorbol()

8 7 5 2 1

s.sorba(5)	s.sorba(1)
s.sorba(8)	s.sorbol() 8
s.sorba(2)	s.sorbol() 7
s.sorba(7)	s.sorbol() 5

Sor, Verem megvalósítása

Láncolt listák

Prioritási sor megvalósítása

Már ismert adatszerkezet: q -A

Egyesíthető prioritási sor megvalósítása

??????

Módosítható prioritási sor megvalósítása

Módosítható, egyesíthető prioritási sor megvalósítása

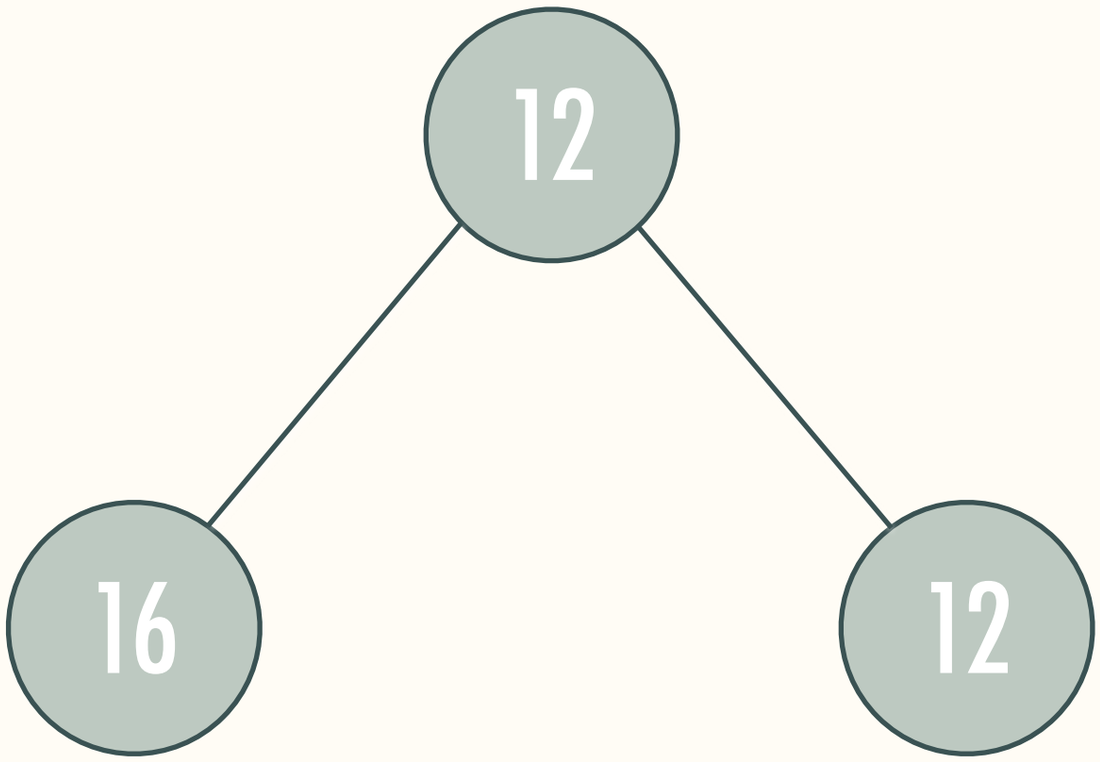
Bináris kupac

alkalmazása

prioritási sor

Kupac adatszerkezet tulajdonsága:

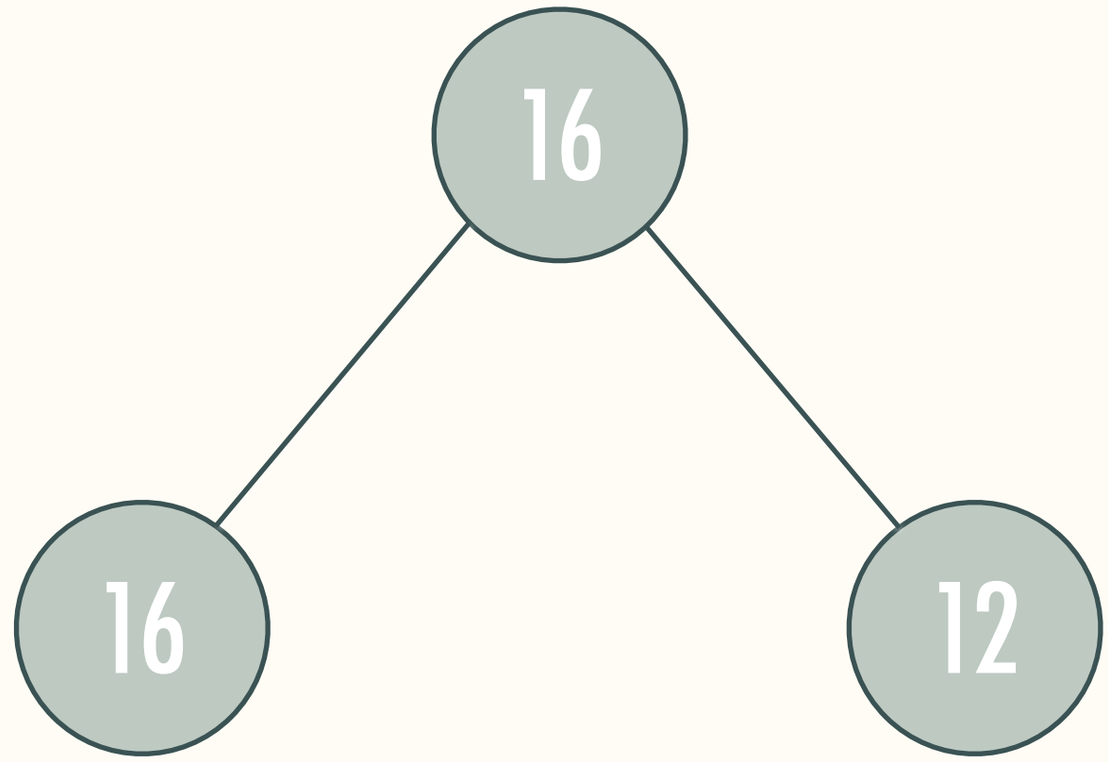
minimum kupac:



$$\forall p \in Q_{pac} : p_{qlcs} \leq \min(\langle p_{bal} \rangle_{qlcs}, \langle p_{jobb} \rangle_{qlcs})$$

Kupac adatszerkezet tulajdonsága:

maximum kupac:



$$\forall p \in Q_{pac} : p_{qlcs} \geq \max(\langle p_{bal} \rangle_{qlcs}, \langle p_{jobb} \rangle_{qlcs})$$

5, 8, 2, 7, 1

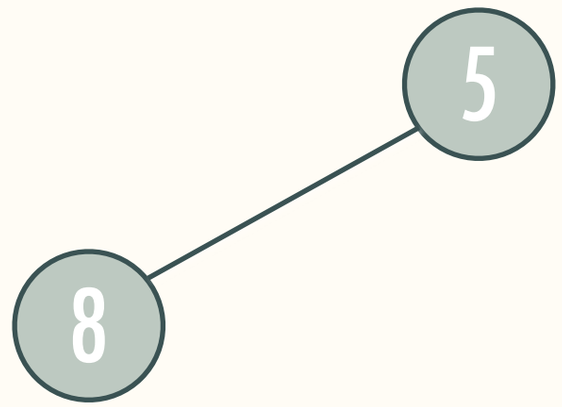
maximum kupac

void s.sorba(T x)

5, 8, 2, 7, 1

maximum kupac

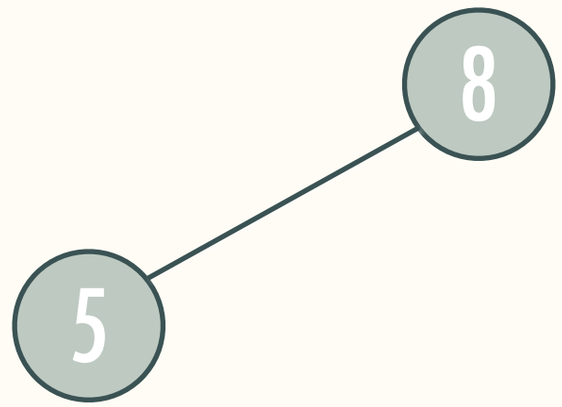
void s.sorba(T x)



5, 8, 2, 7, 1

maximum kupac

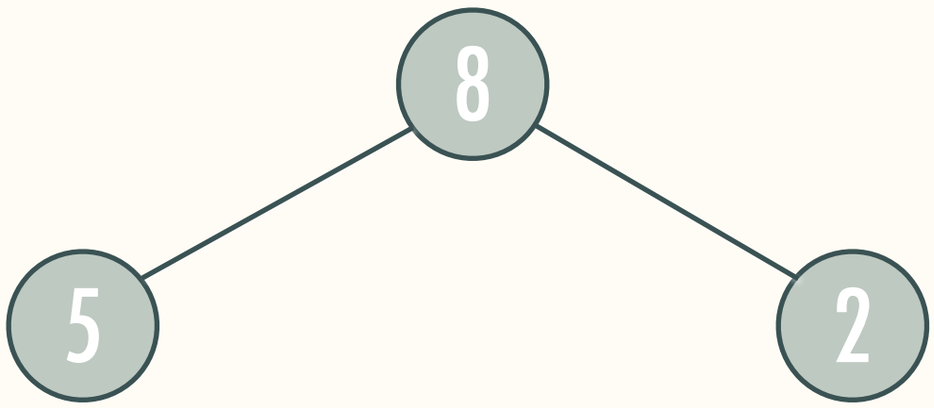
void s.sorba(T x)



5, 8, 2, 7, 1

maximum kupac

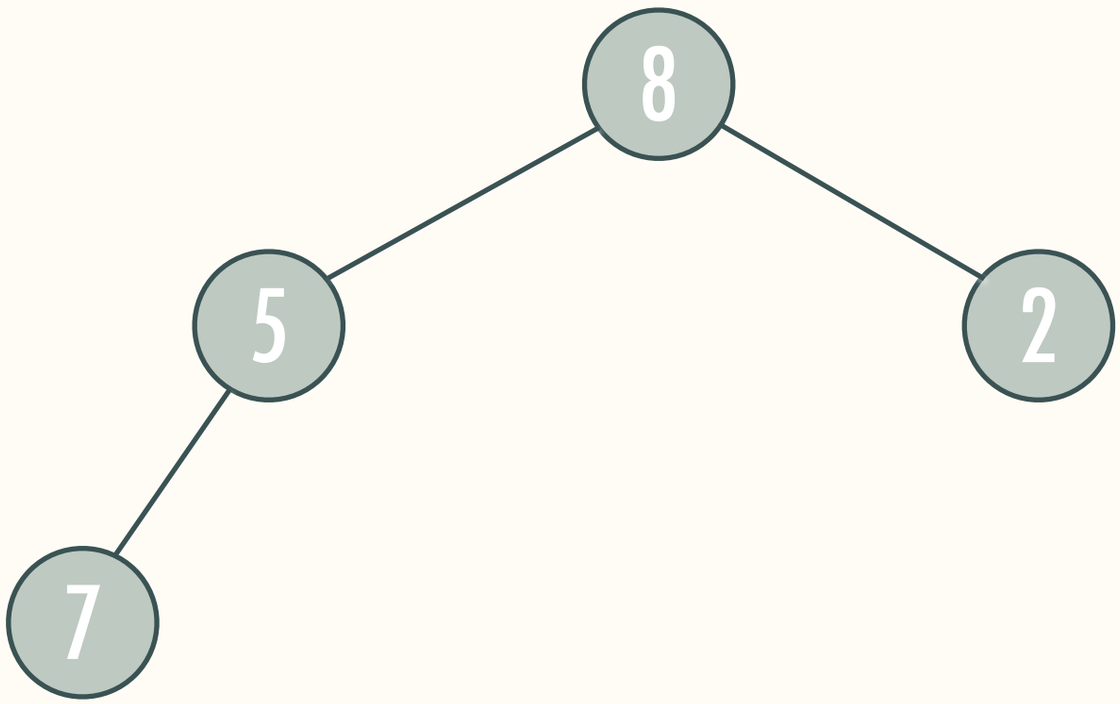
void s.sorba(T x)



5, 8, 2, 7, 1

maximum kupac

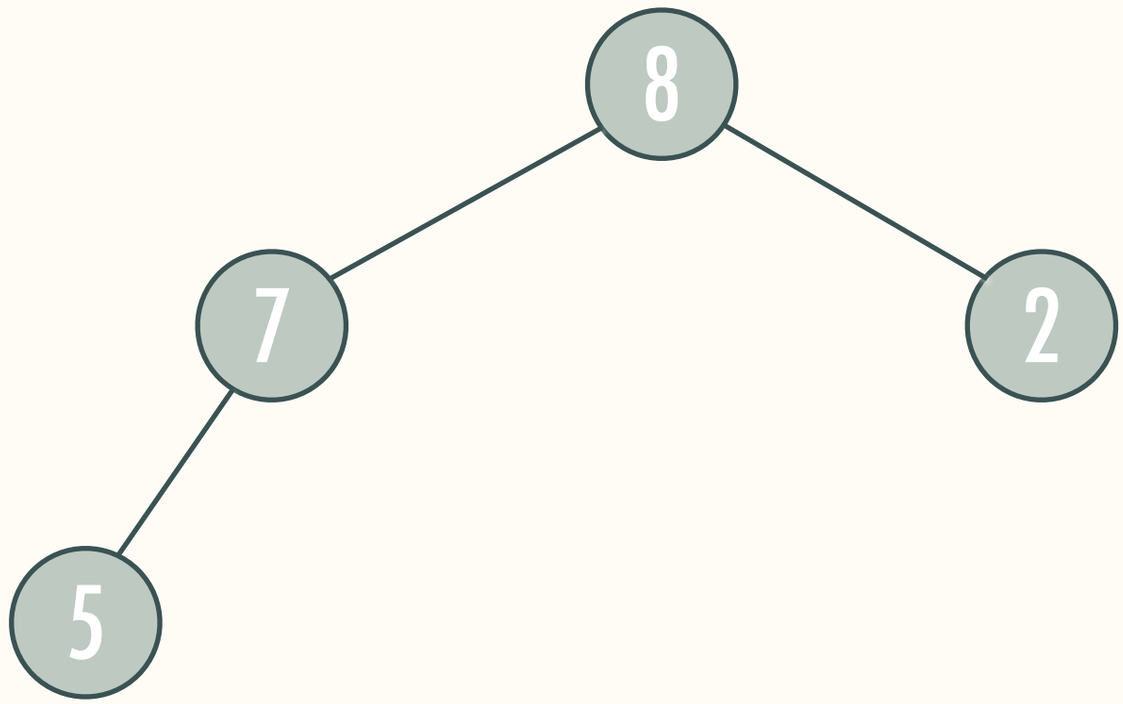
void s.sorba(T x)



5, 8, 2, 7, 1

maximum kupac

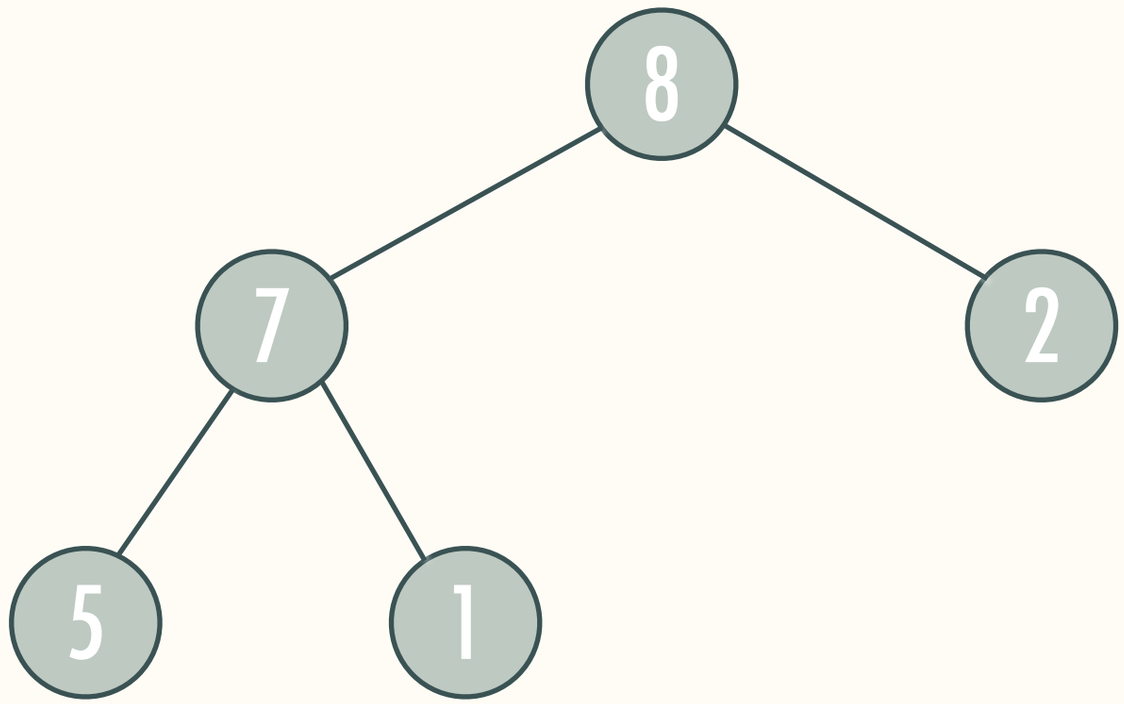
void s.sorba(T x)



5, 8, 2, 7, 1

maximum kupac

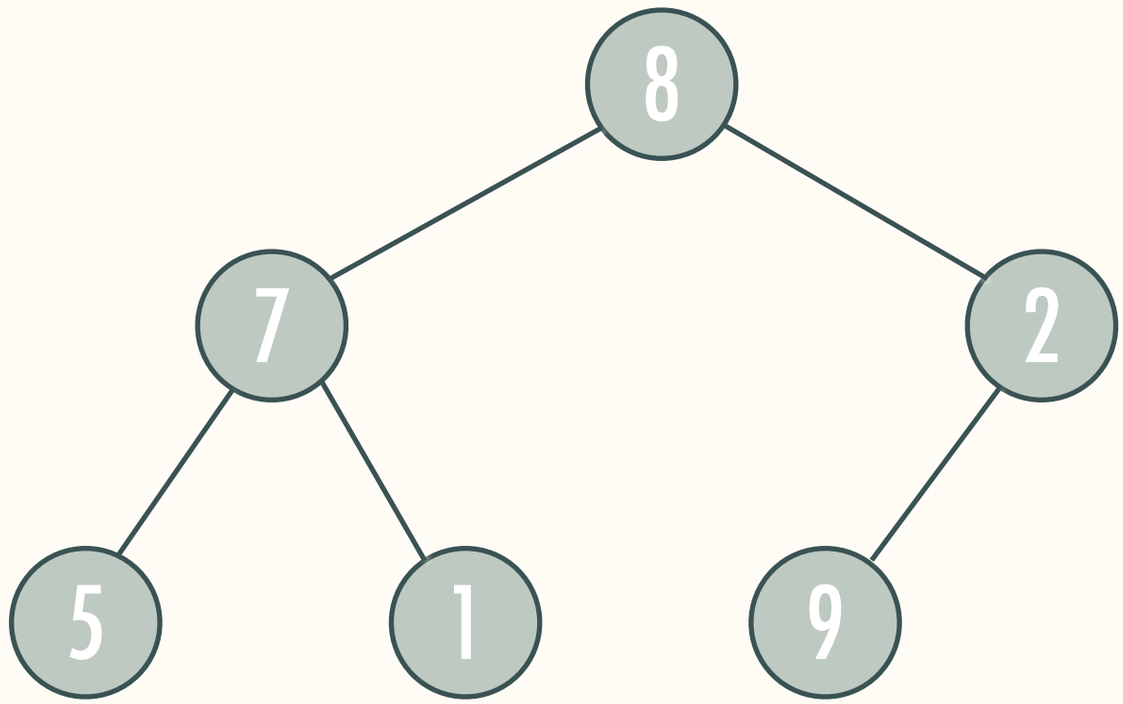
void s.sorba(T x)



5, 8, 2, 7, 1, 9

maximum kupac

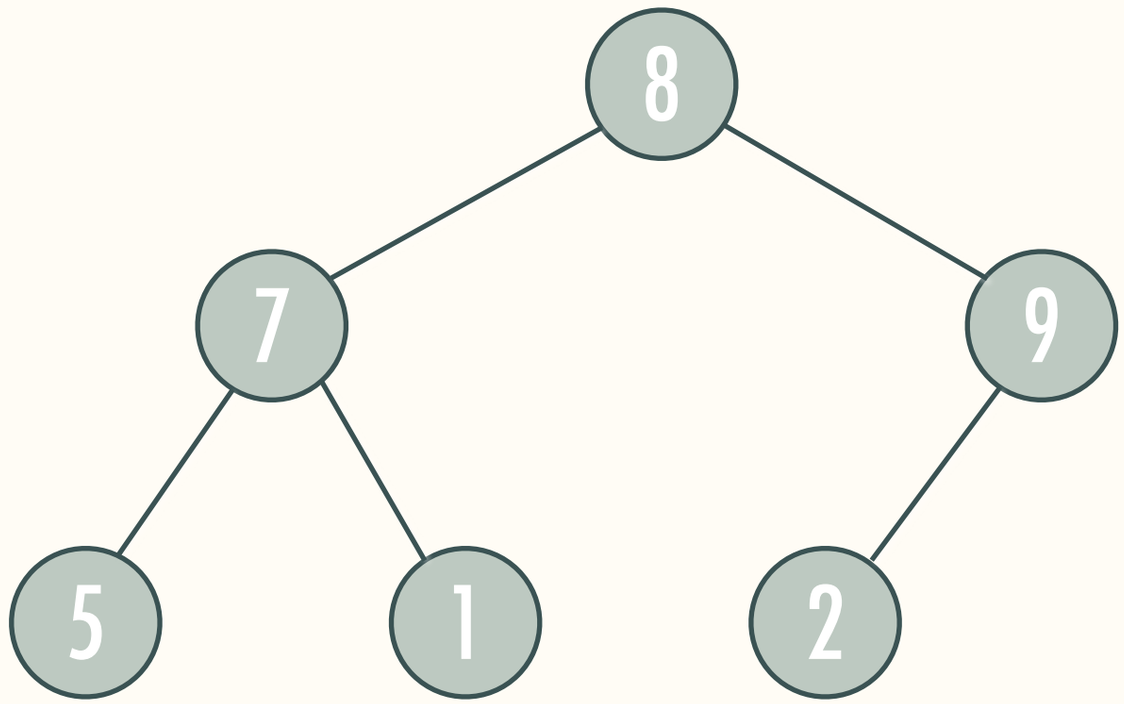
void s.sorba(T x)



5, 8, 2, 7, 1, 9

maximum kupac

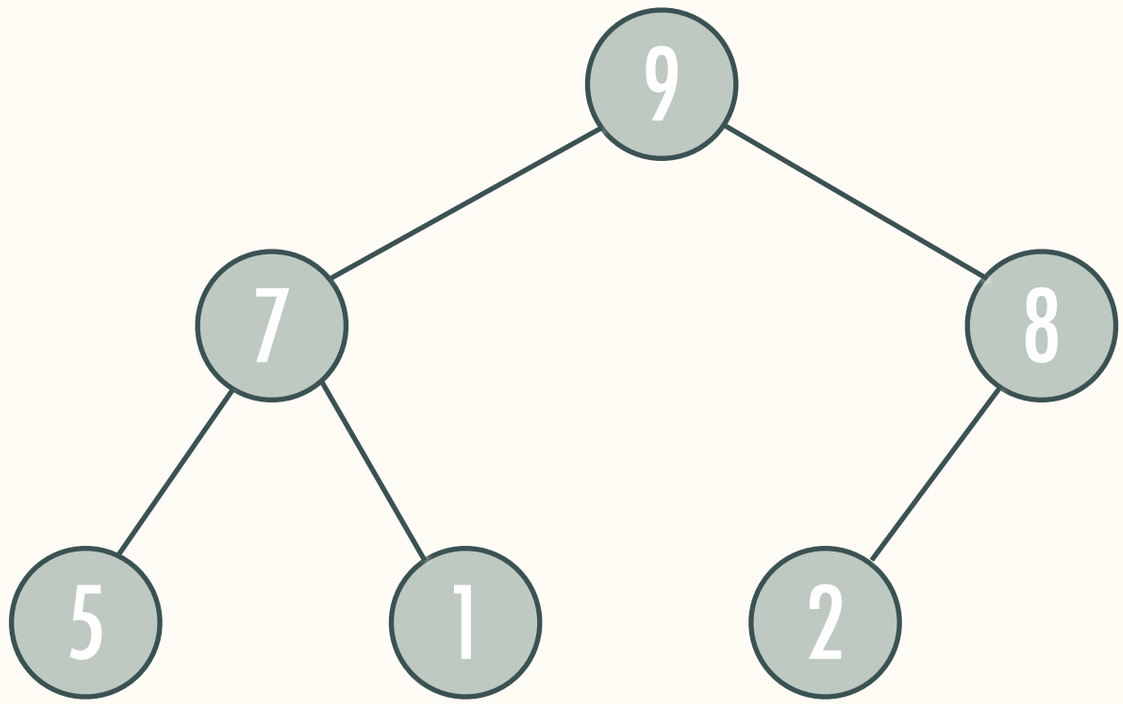
void s.sorba(T x)



5, 8, 2, 7, 1, 9

maximum kupac

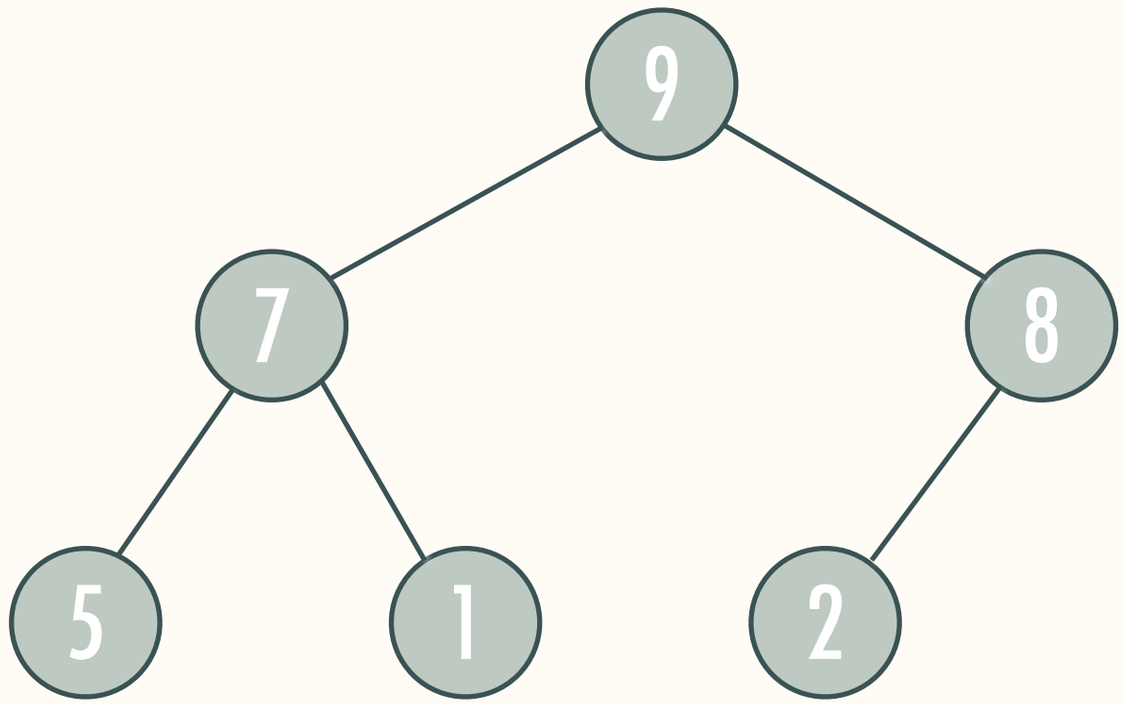
void s.sorba(T x)



sorba művelet időigénye O(log(n))

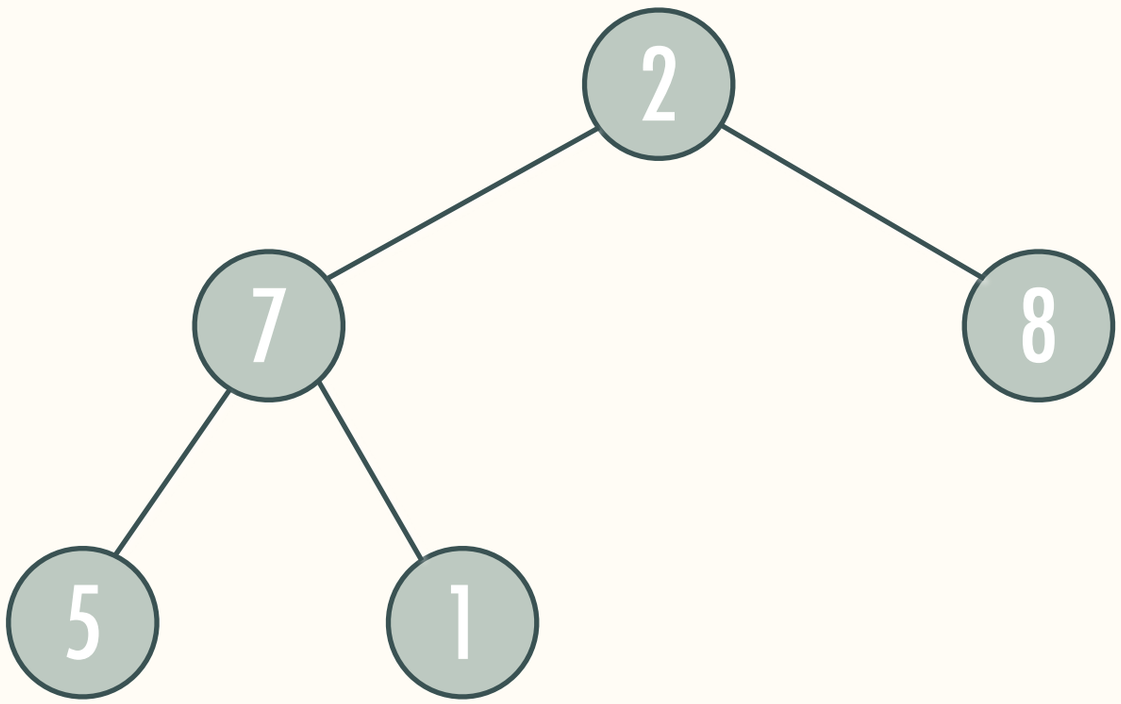
maximum kupac

T s.sorbol()



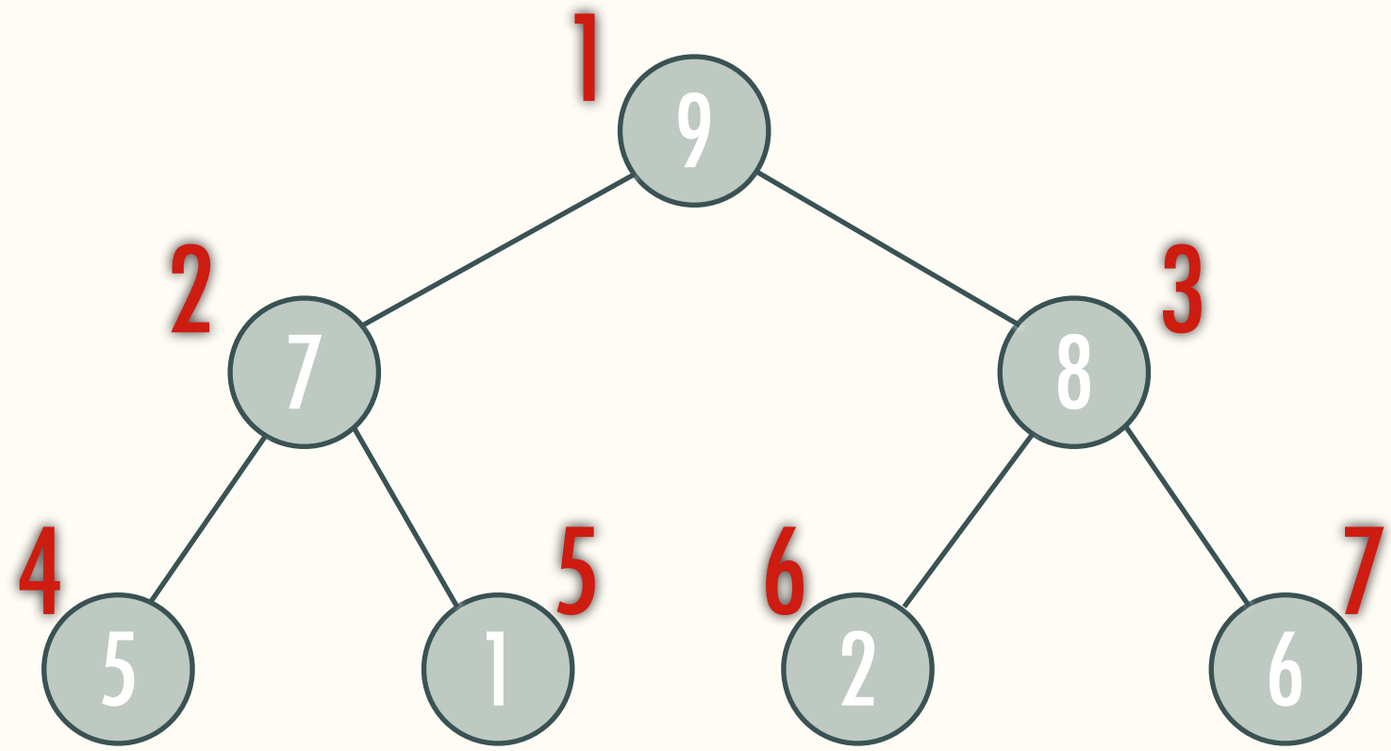
maximum kupac

T s.sorbol()



sorbol művelet időigénye O(log(n))

A kupac adatszerkezet már tanult megvalósítása

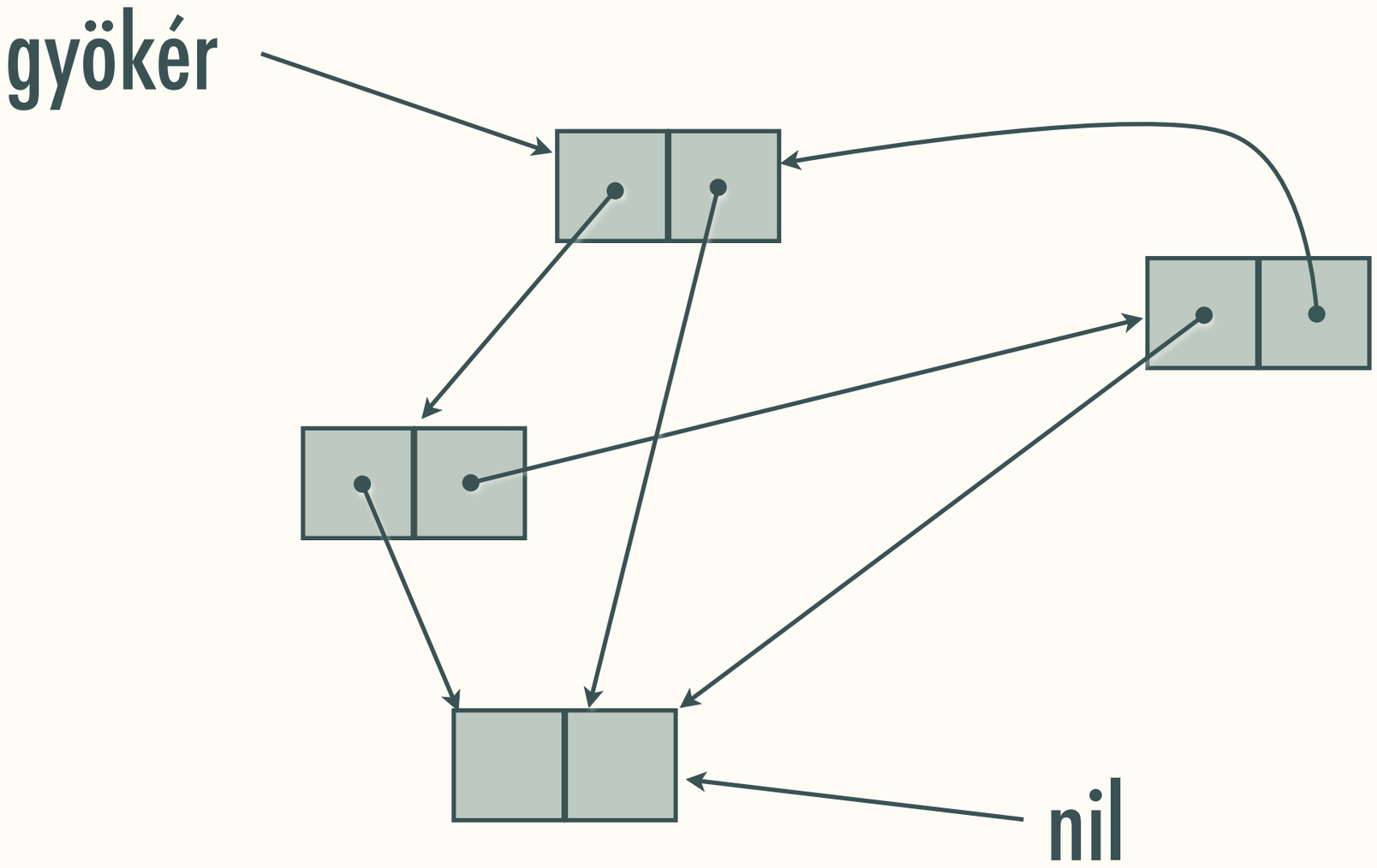


1	2	3	4	5	6	7
9	7	8	5	1	2	6

Nem dinamikus adatszerkezet

Dinamikus implementáció

Bináris fa 2 pointeres megvalósítása?



Egyesíthető prioritási sor

Egyesíthető prioritási sor műveletek:

Eprisor s1,s2 ;

void s.sorba(T x) ; O(log(n)) ?

T s.sorbol() ; O(log(n)) ?

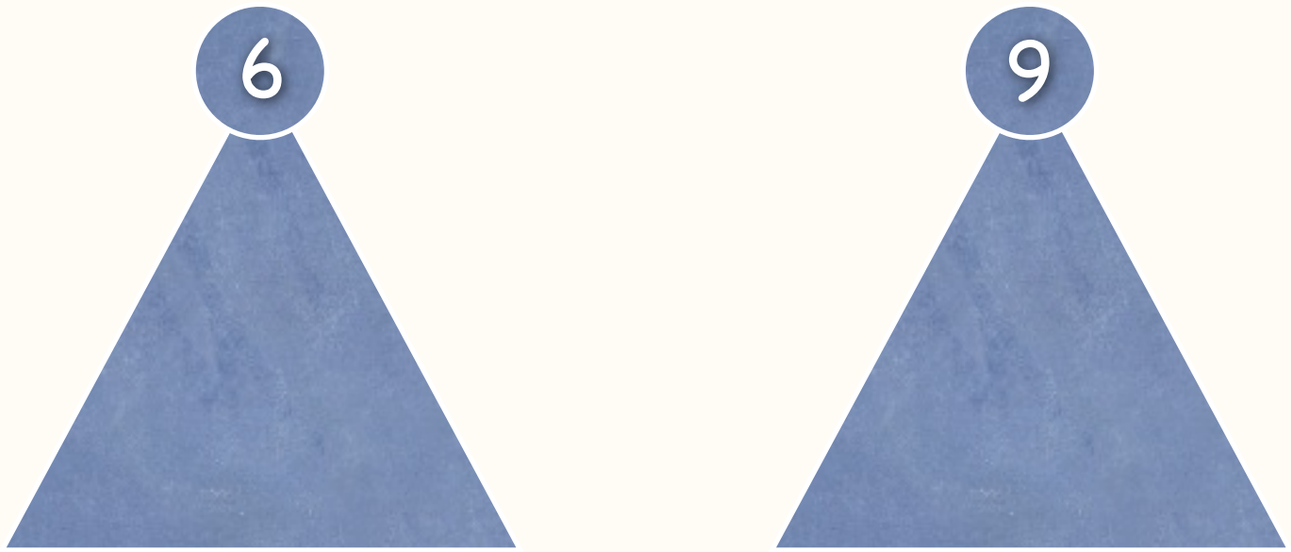
Eprisor s1.egyesit(Eprisor s2) ; ?

Binomiális fa

Prisor S_1, S_2 ;

maximum kupac

$S = S_1 \triangle S_2$

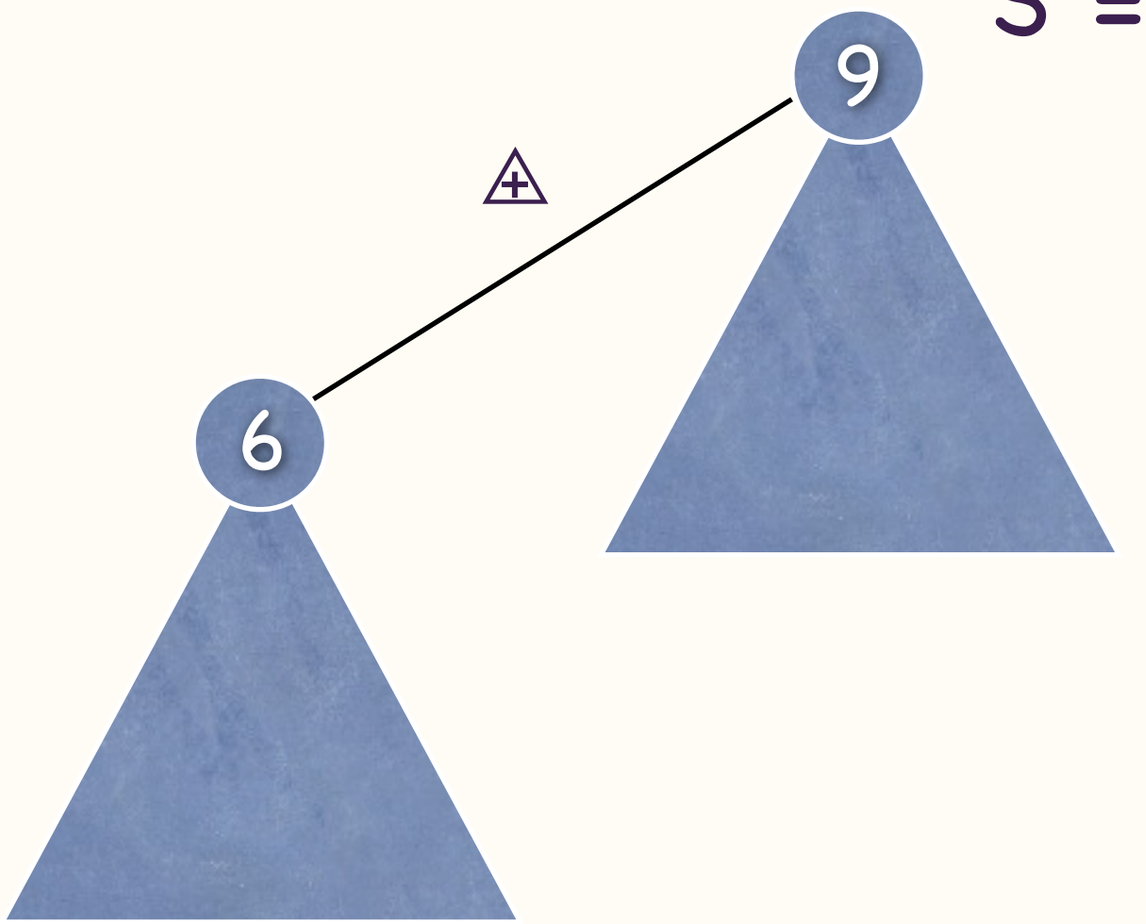


def.: $\triangle$ művelet a nagyobb kulcsot tartalmazó gyökérelemhez kapcsoljuk a kisebb kulcsot tartalmazó gyökérelemet

maximum kupac

Prisor S_1, S_2 ;

$S = S_1 \triangleq S_2$



def.: $\triangleq$ művelet a nagyobb kulcsot tartalmazó gyökérelemhez kapcsoljuk a kisebb kulcsot tartalmazó gyökérelemet

def.: legyen $\mathcal{B}(r)$ r-ed rangú binomiális fa

legyen $\mathcal{B}(0)$ egy egyetlen pontból

álló binomiáris fa

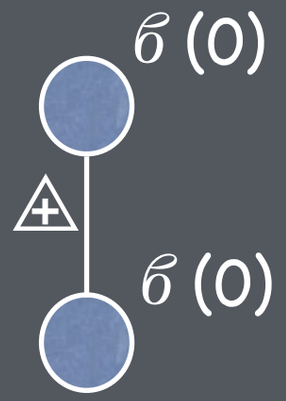
és $\mathcal{B}(r) = \mathcal{B}(r-1) \triangle \mathcal{B}(r-1)$

példa:

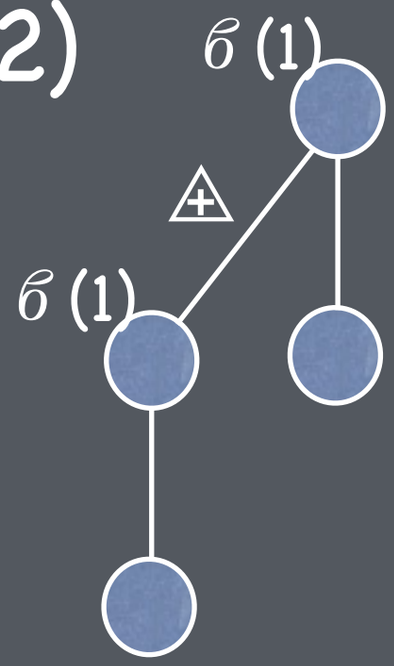
példa:

$\mathcal{B}(0)$

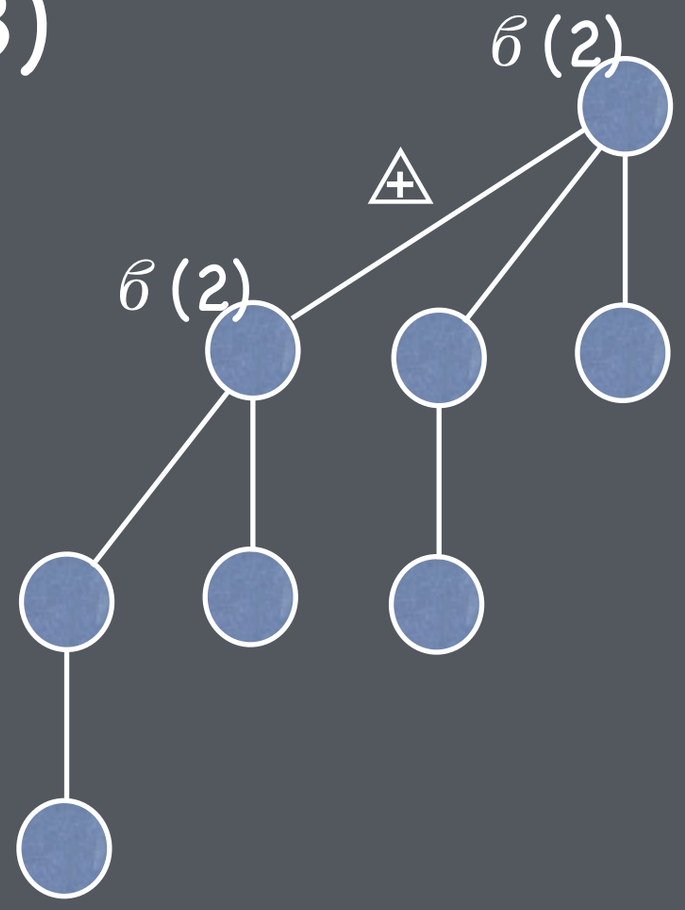
$\mathcal{B}(1)$



$\mathcal{B}(2)$



$\mathcal{B}(3)$



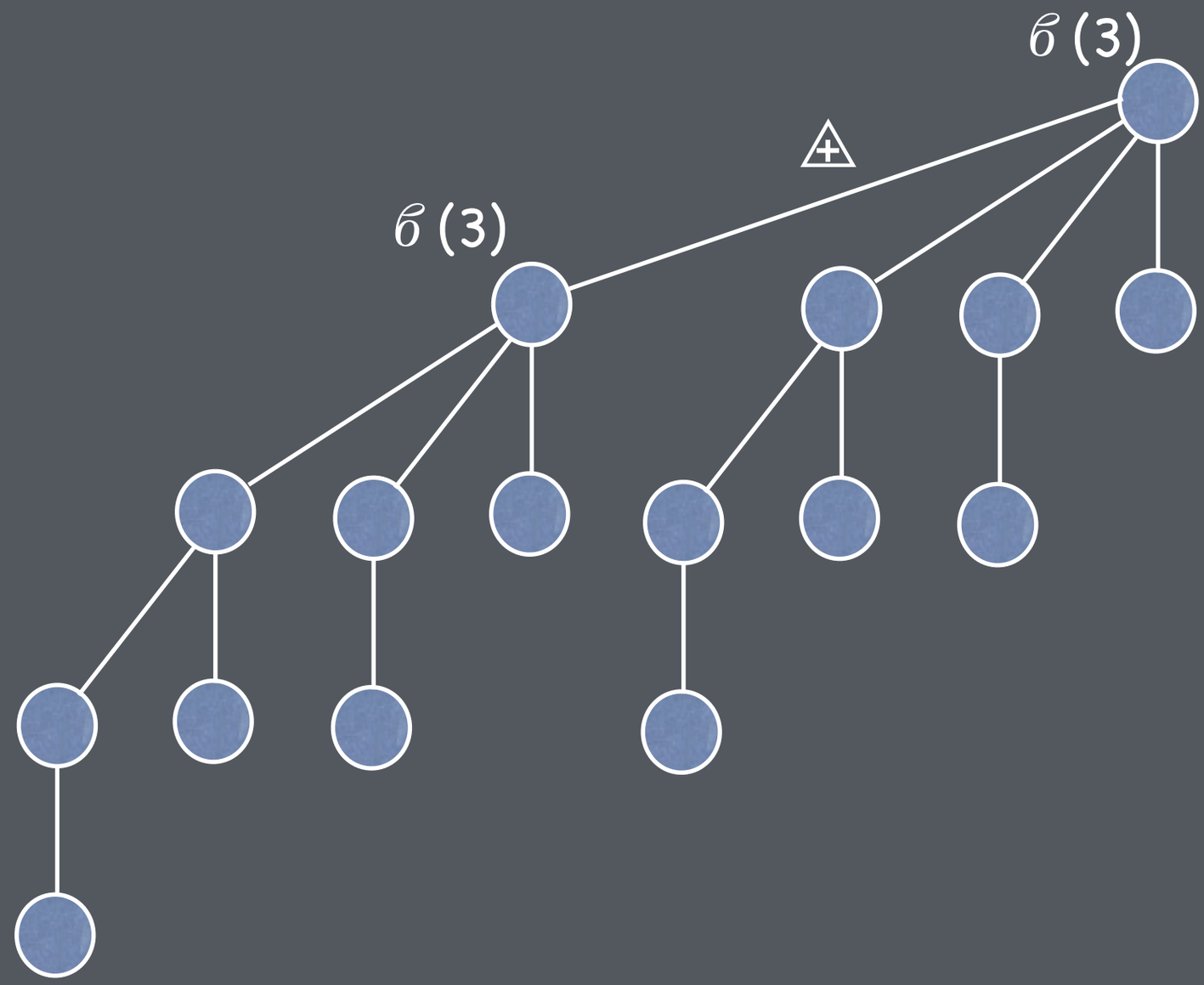
1

3

3

1

$B(4)$



1

4

6

4

1

tétel: $h=O(\log(n))$

biz.:

$h=h(\mathcal{G}(r))=r+1$

$n=N(\mathcal{G}(r))=2^r$

$\log_2(n)=\log_2(2^r)=r\cdot\log_2(2)=r=h-1$

$h-1=\log_2(n)$

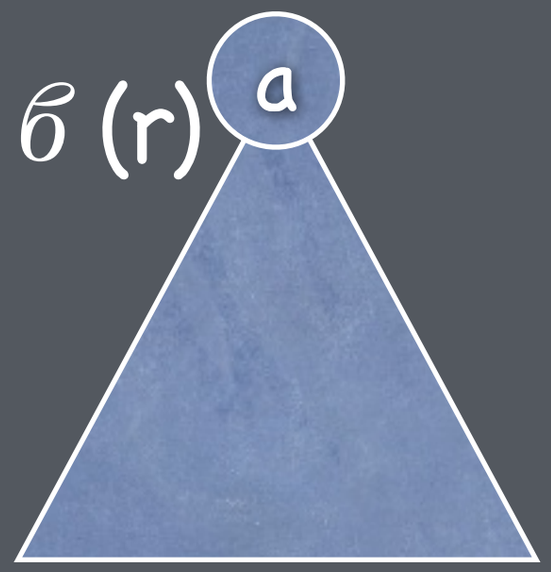
$h=\log_2(n)+1=O(\log(n))$

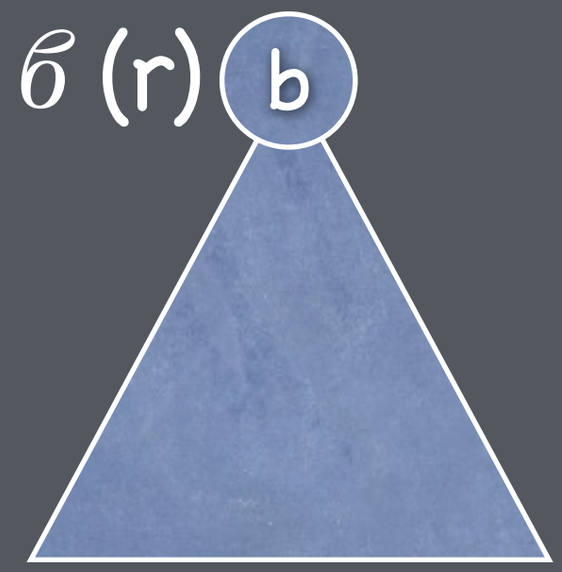
□

r	h	n
0	1	1
1	2	2
2	3	4
3	4	8

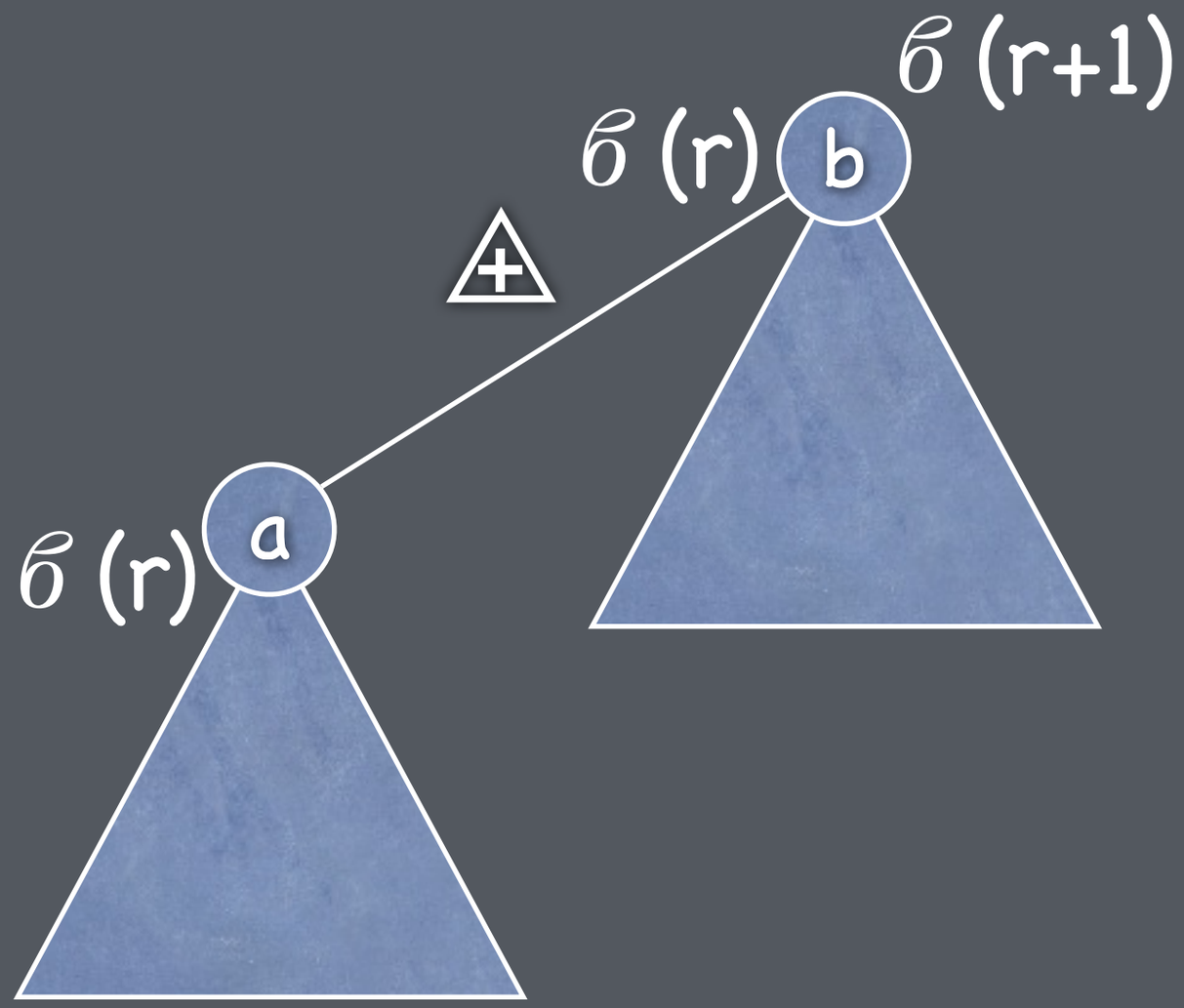
Binomiális kupac

$b \geq a$





$b \geq a$



Mennyi adatot kell tárolunk?

r	h	n
0	1	1
1	2	2
2	3	4
3	4	8

n=3

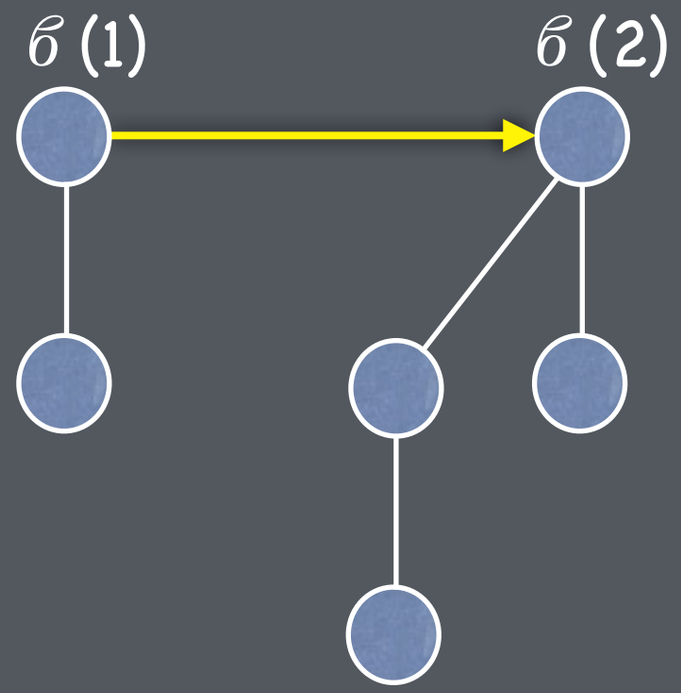
n=5 n=6 n=7

csak binomiáris fákat használjunk!

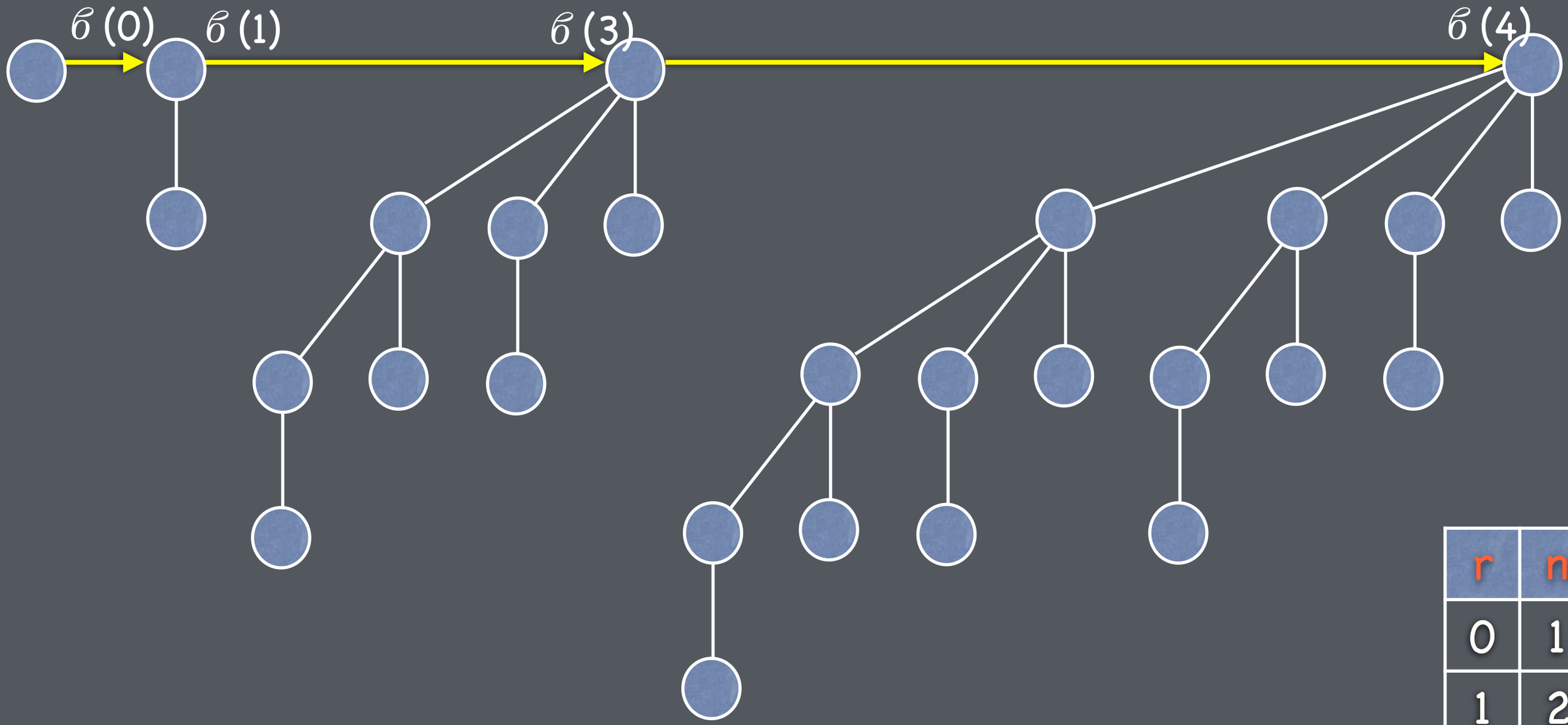
pl.:

n=6

r	h	n
0	1	1
1	2	2
2	3	4
3	4	8



pl.: n=27



r	n
0	1
1	2
2	4
3	8
4	16

n	n	l	bh(n)
1	*1*	1	1
2	*10*	1	2
3	*11*	2	
4	*100*	1	3
5	*101*	2	
6	*110*	2	
7	*111*	3	
8	*1000*	1	4
9	*1001*	2	
10	*1010*	2	
11	*1011*	3	
12	*1100*	2	
13	*1101*	3	
14	*1110*	3	
15	*1111*	5	
16	*10000*	1	5

tétel: $l \leq \log_2(n)$

biz: l egyenlő az n bináris alakjában szereplő 1-ek számával, ami kisebb, mint a bináris alak összes számjegyeinek száma □

Egyesíthető prioritási sor műveletek:

Eprisor s1,s2 ;

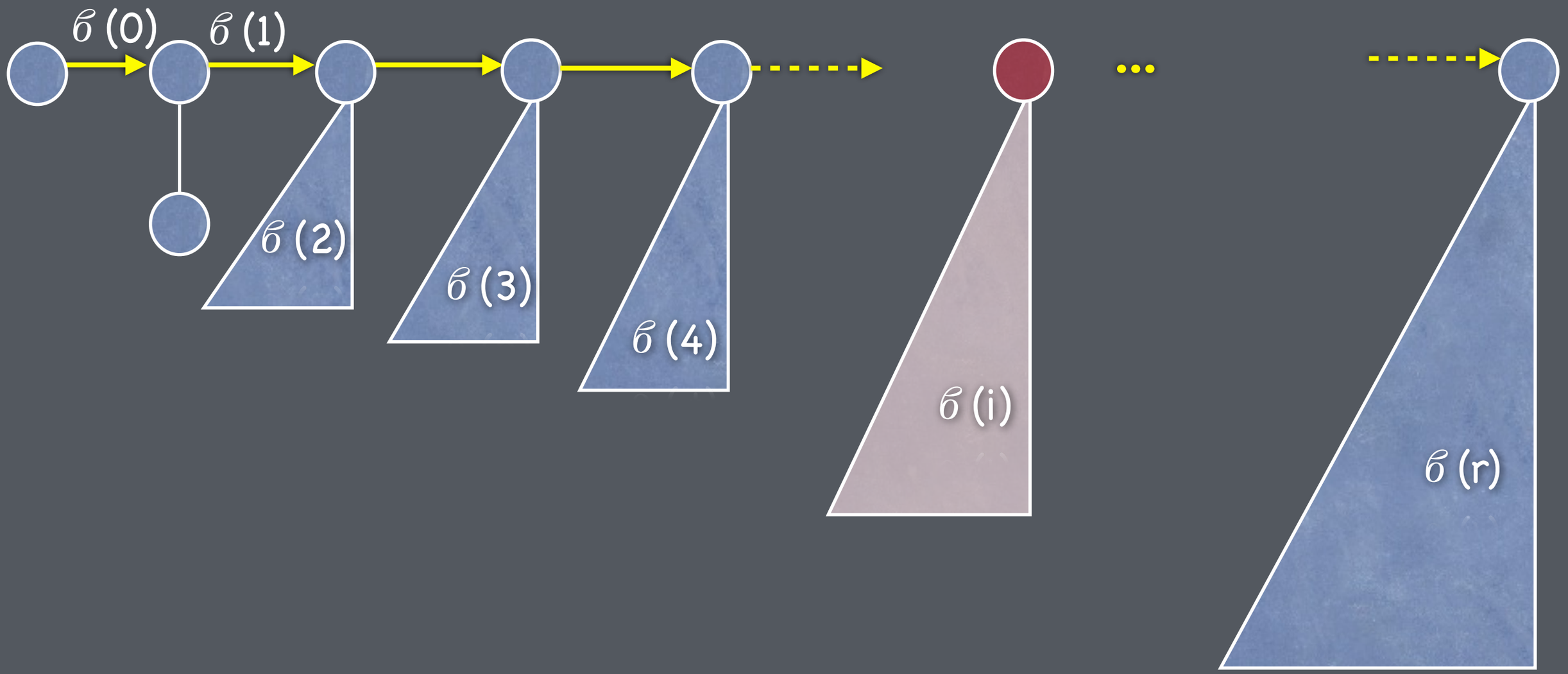
void s.sorba(T x) ;

T s.sorbol() ;

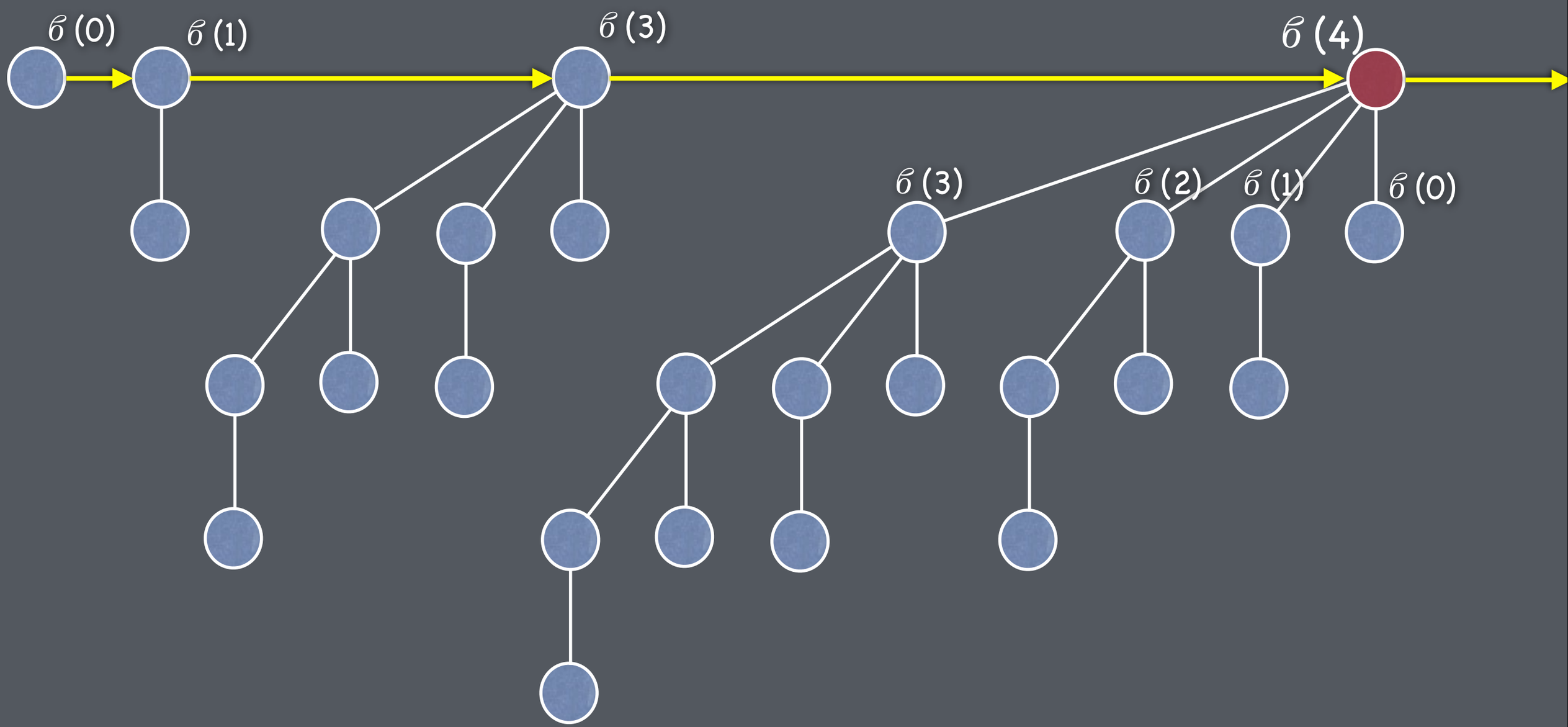
Eprisor s1.egyesit(Eprisor s2) ;

T s.sorbol()

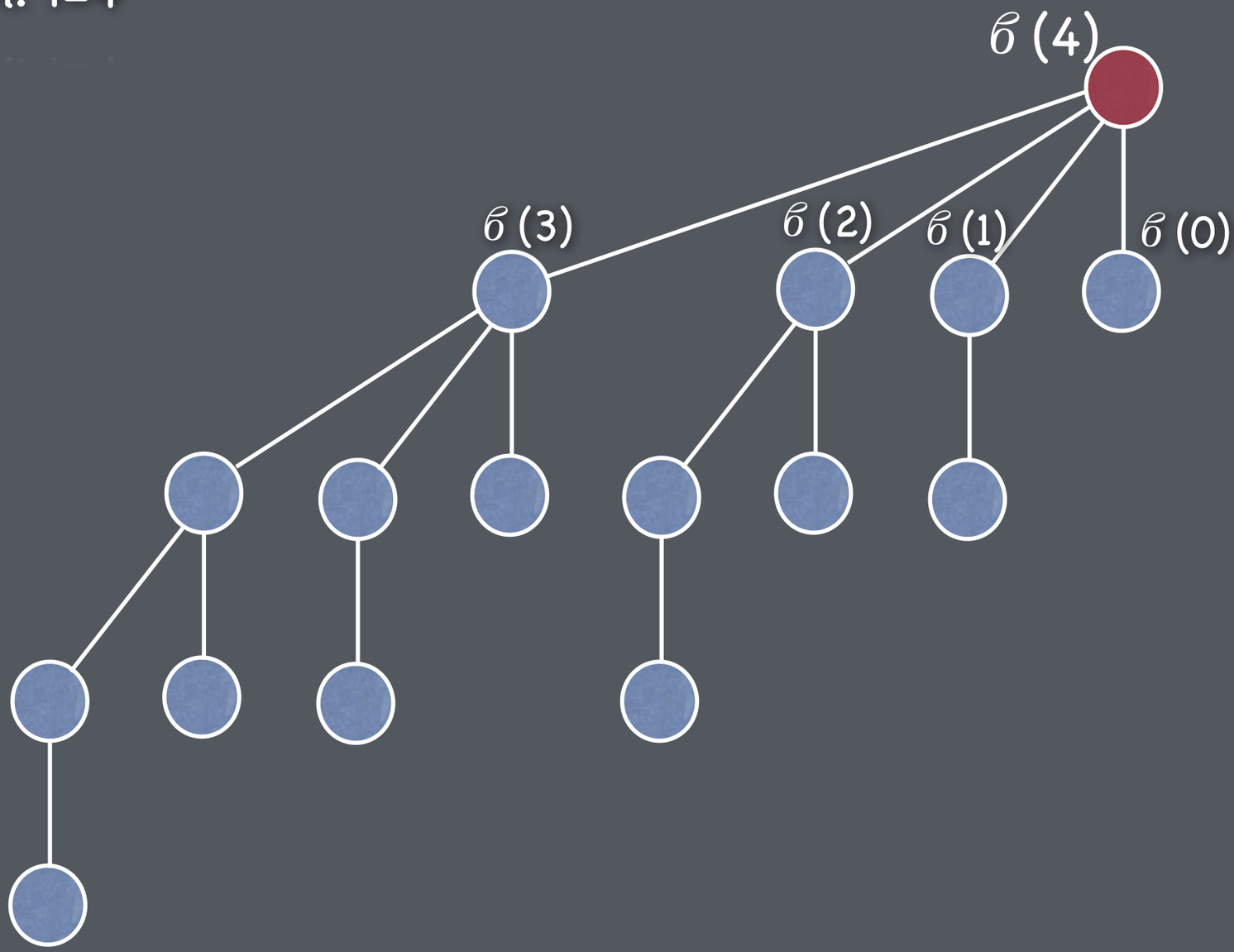
s



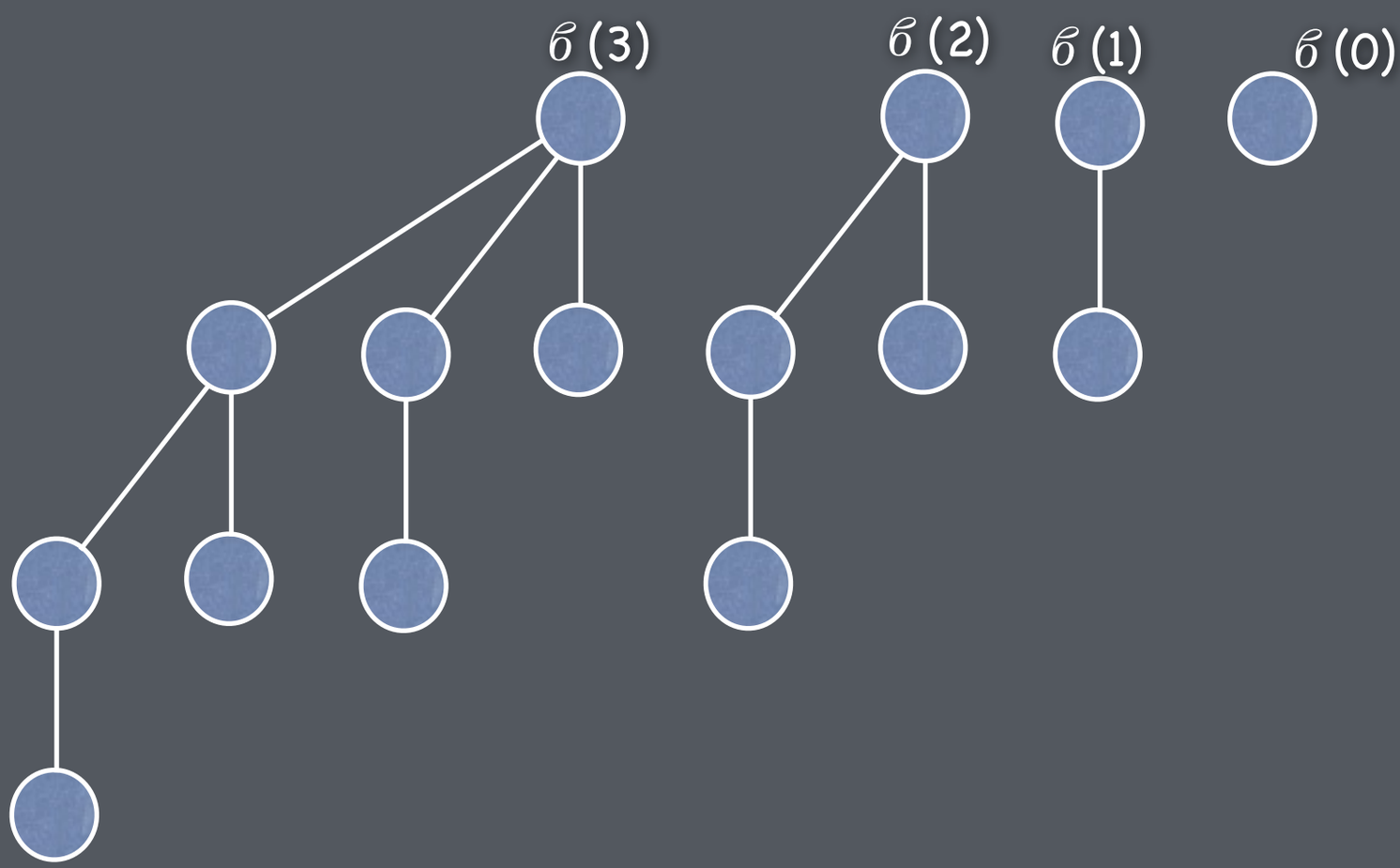
pl. i=4

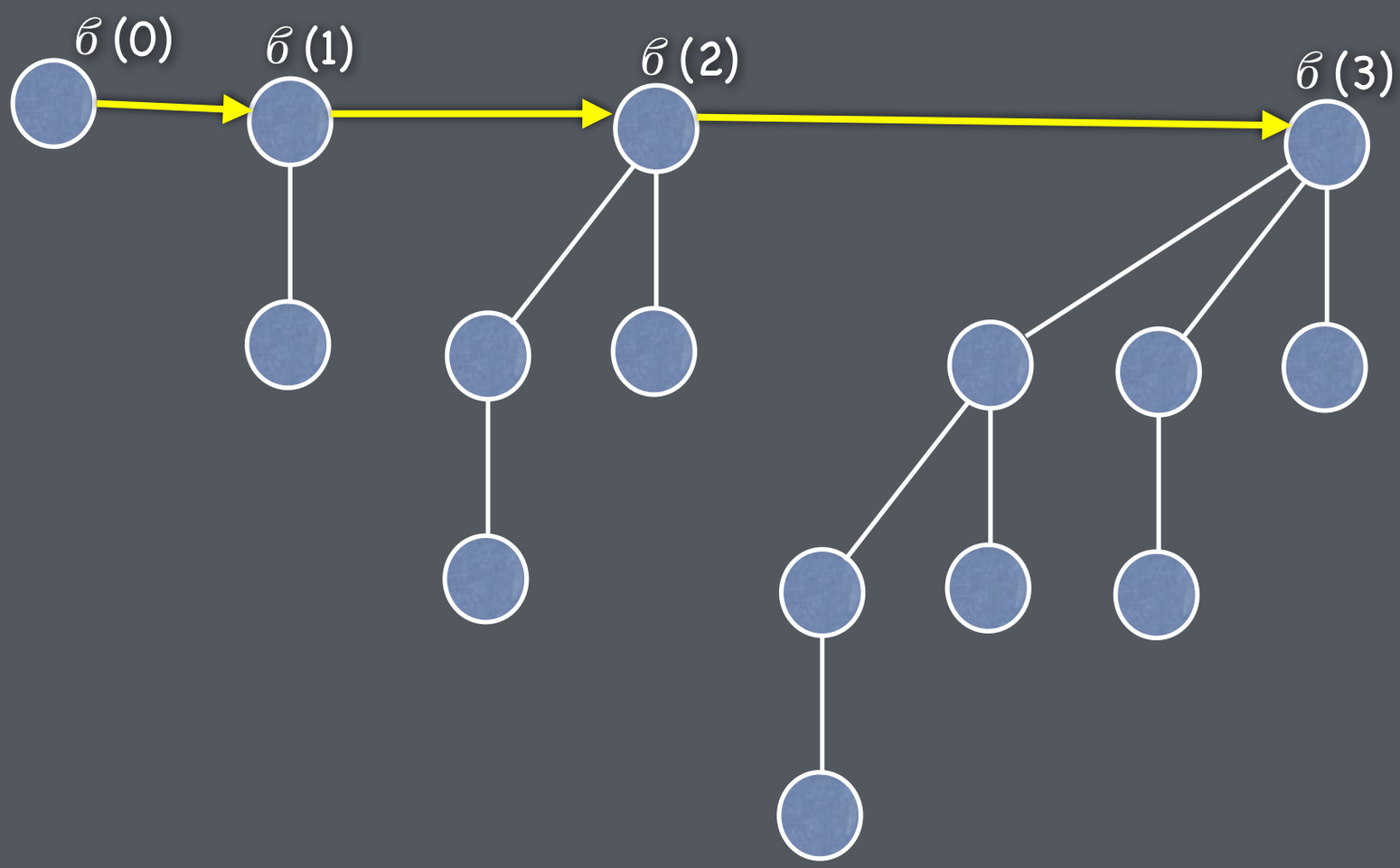


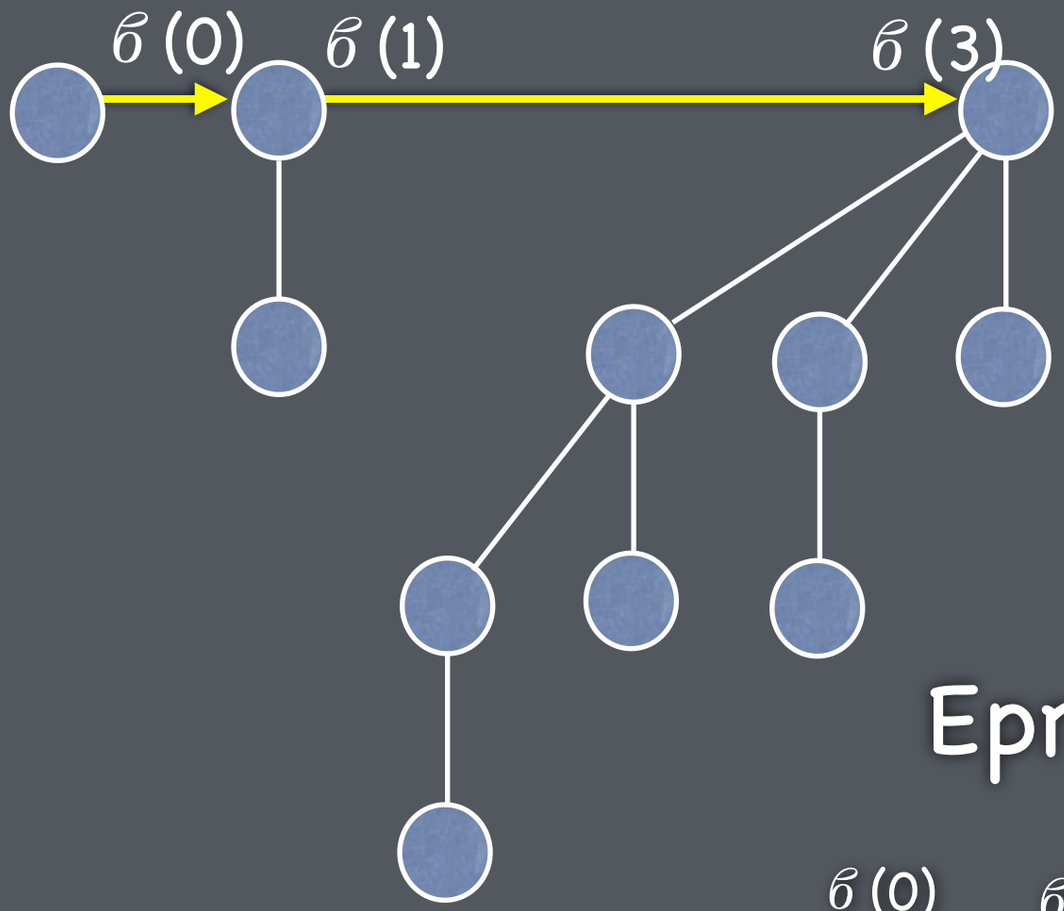
$\mathcal{B}$ (i) pl. i=4



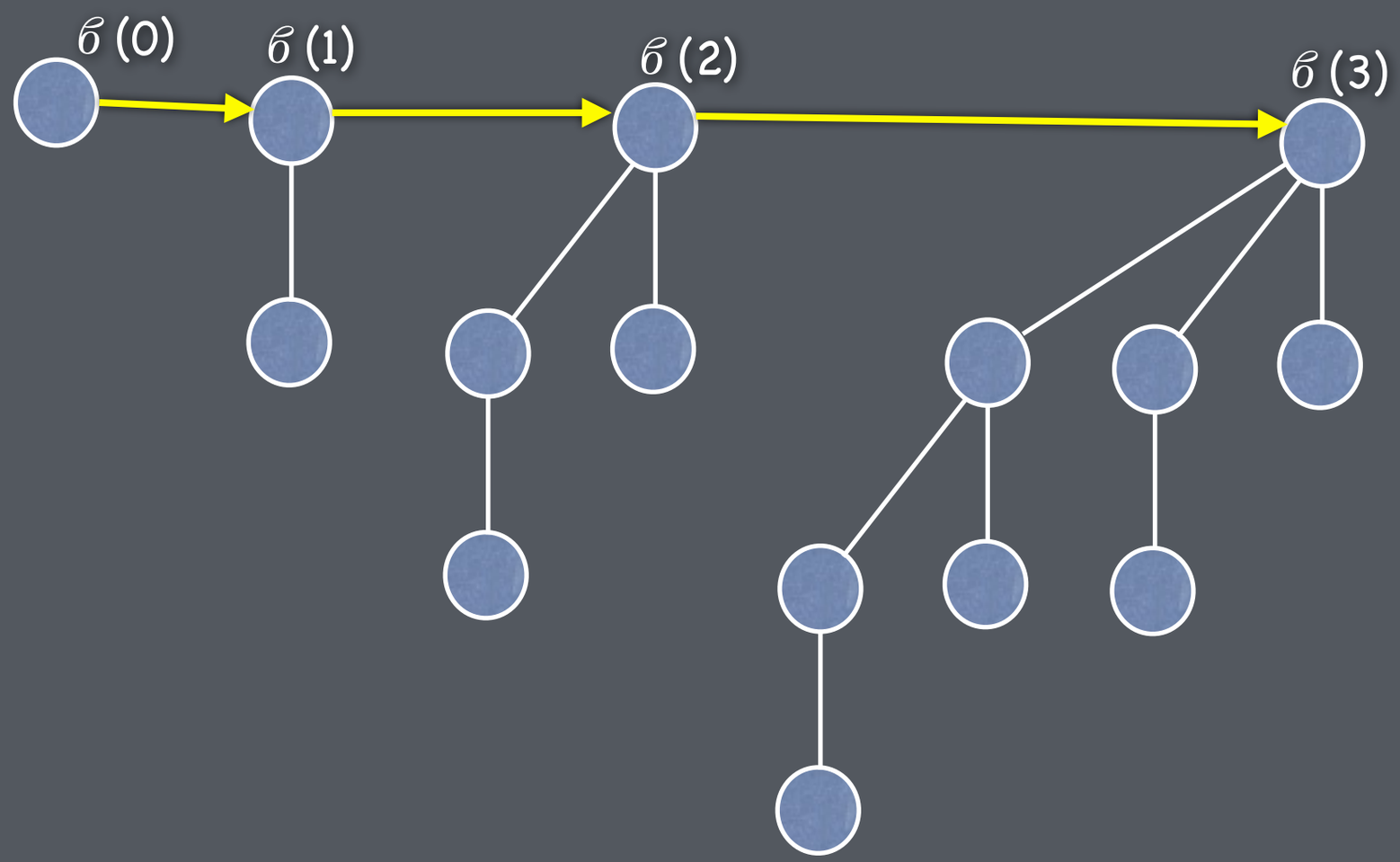
$\mathcal{B}$ (i) pl. i=4

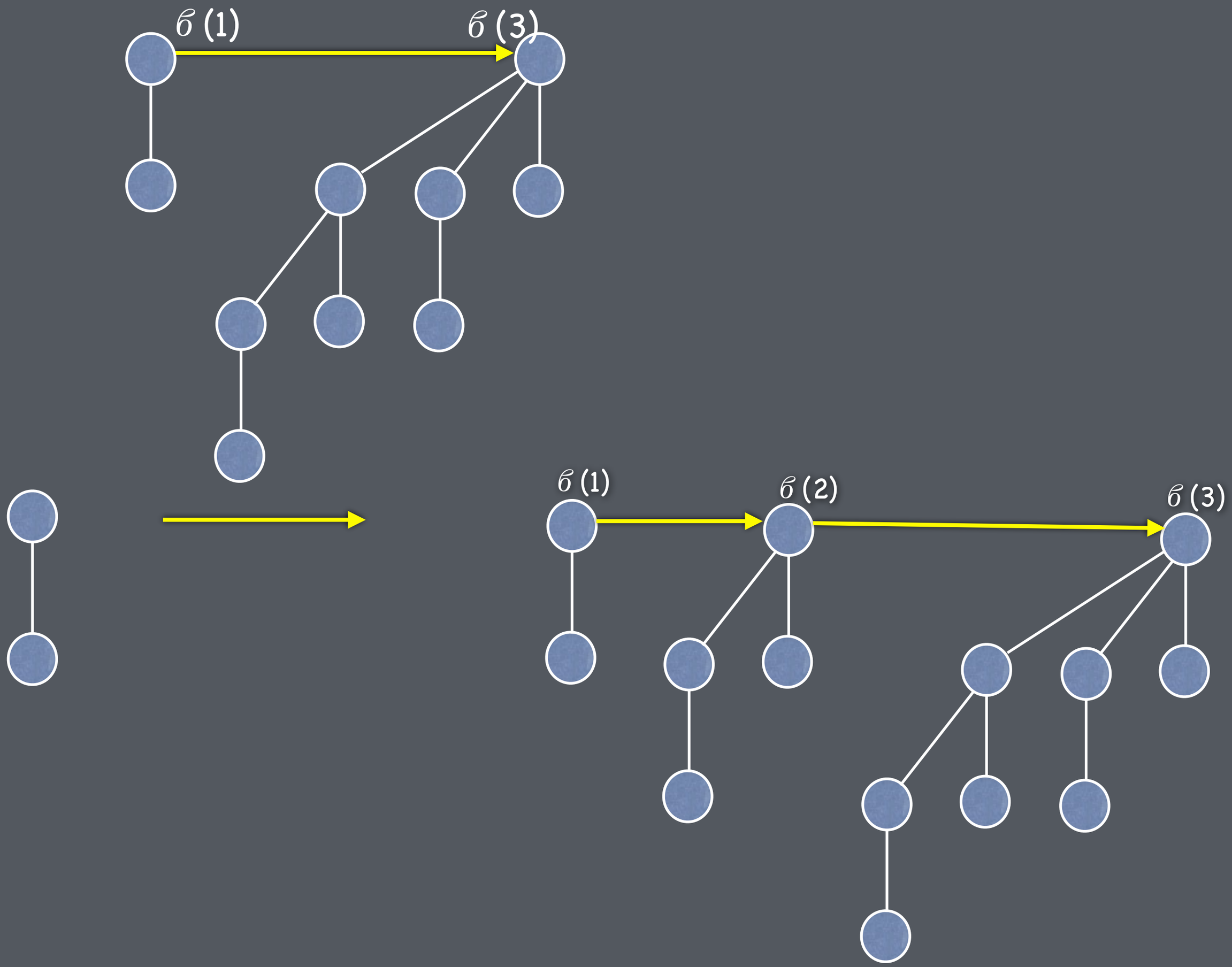


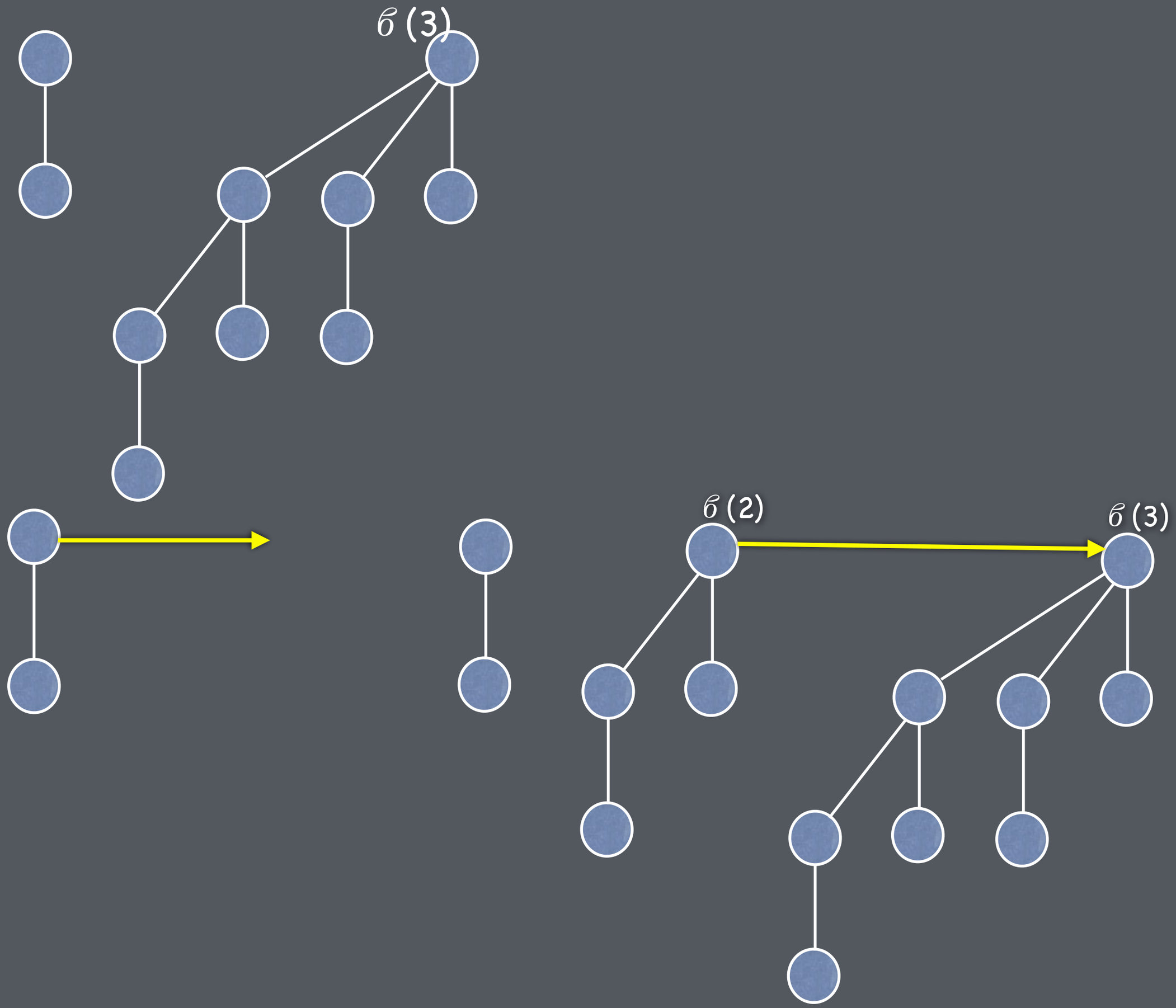

$\mathcal{B}(0)$
$\mathcal{B}(1)$
$\mathcal{B}(2)$
$\mathcal{B}(3)$

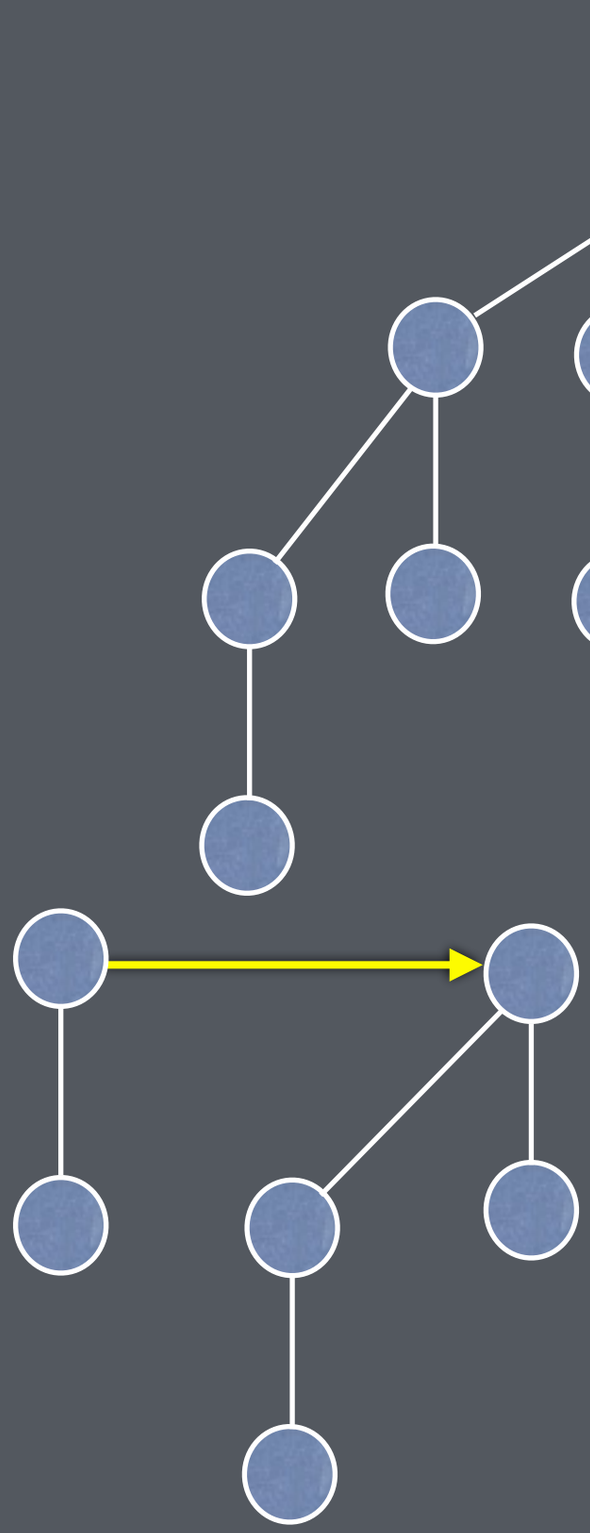


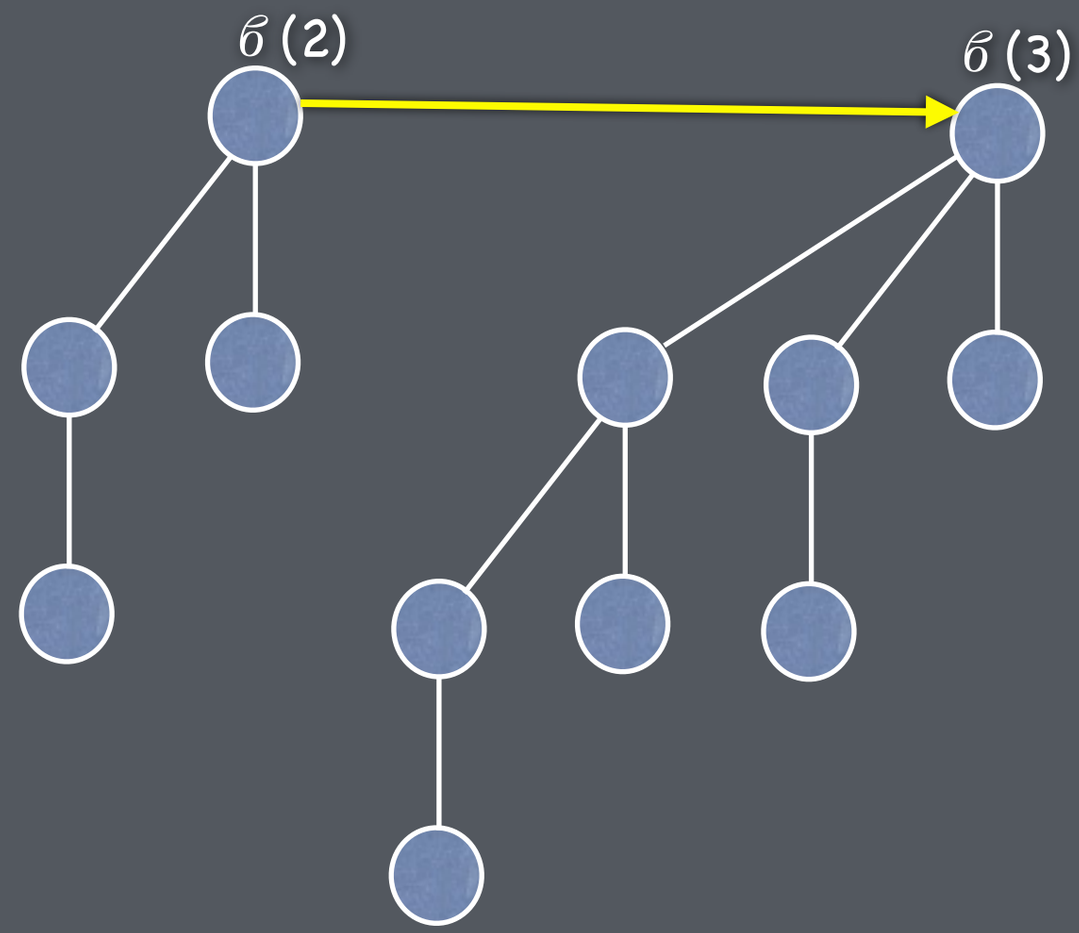
Eprisor s1.egyesit(Eprisor s2) ;

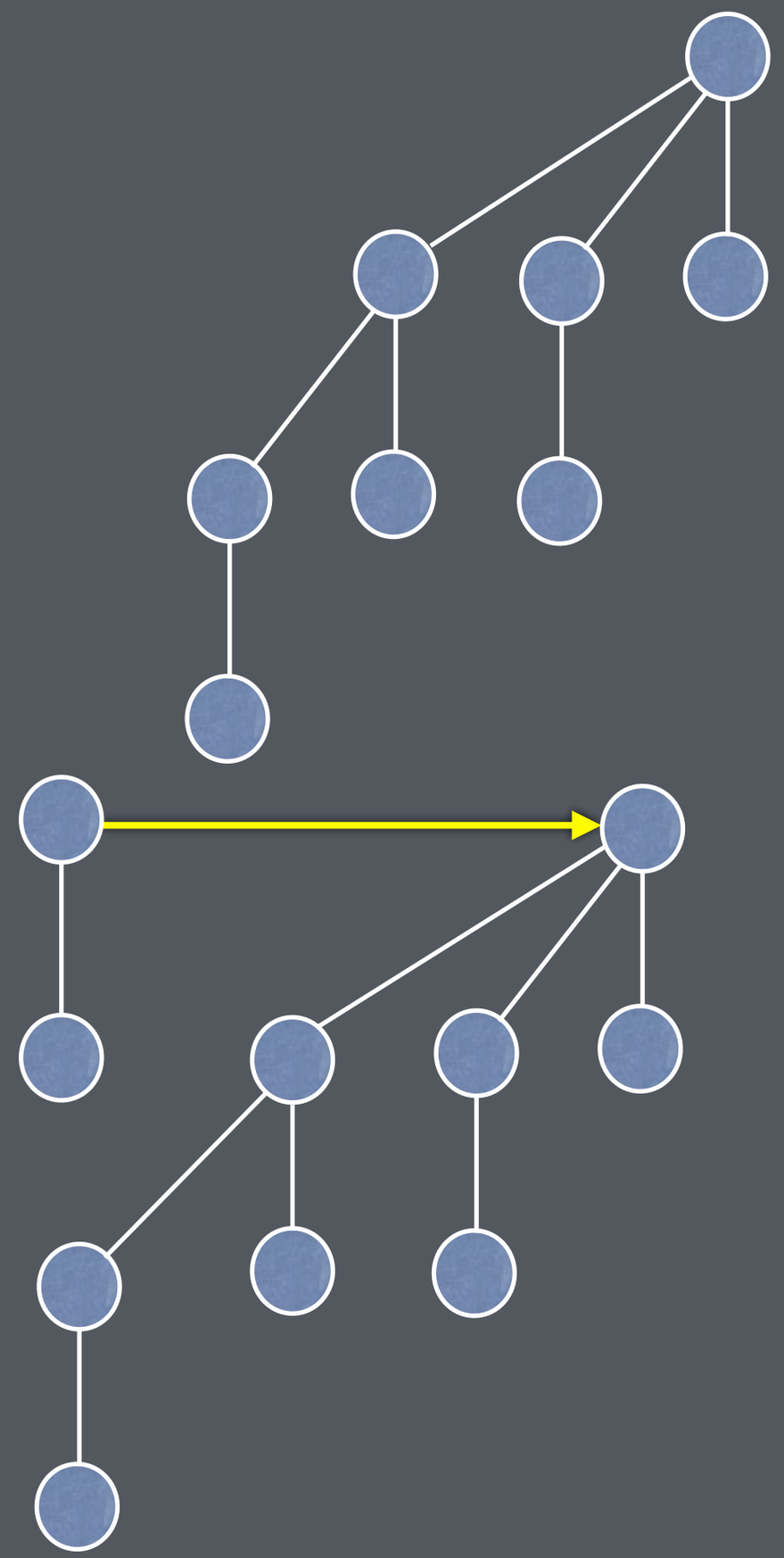


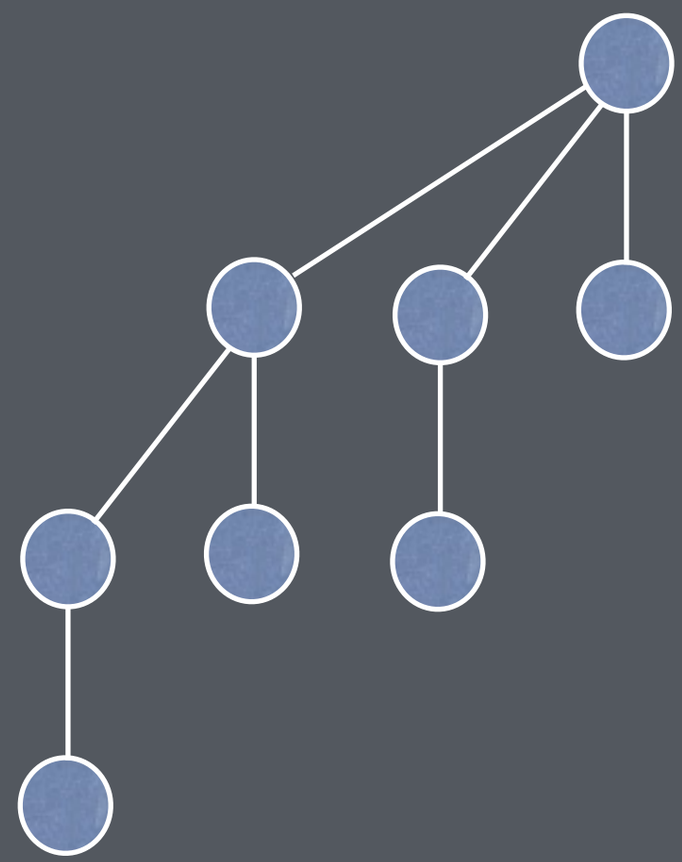

$\mathcal{b}$ (1)
$\mathcal{b}$ (3)
$\mathcal{b}$ (1)
$\mathcal{b}$ (2)
$\mathcal{b}$ (3)

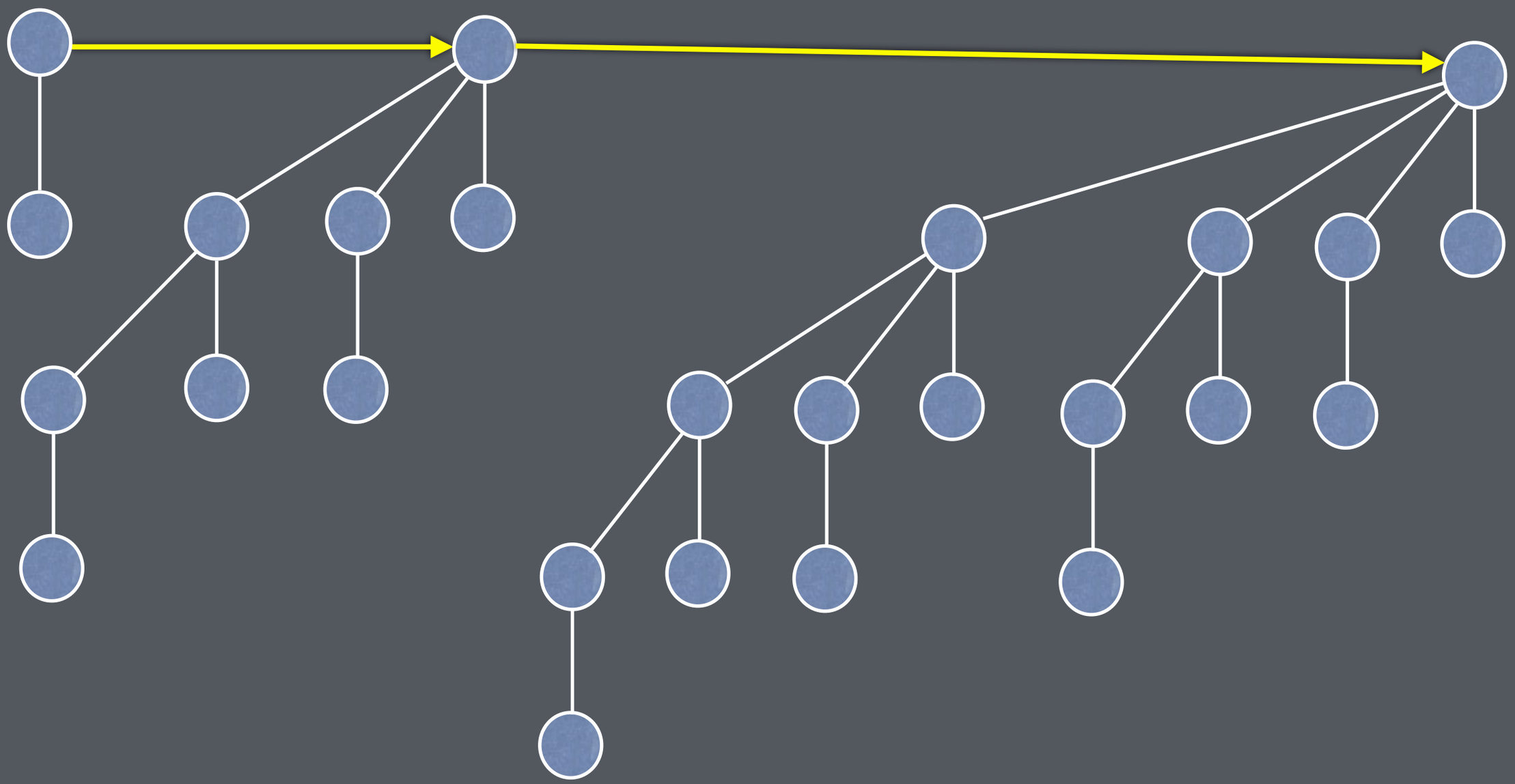

B (3)
B (2)
B (3)


$\mathcal{B}$ (3)

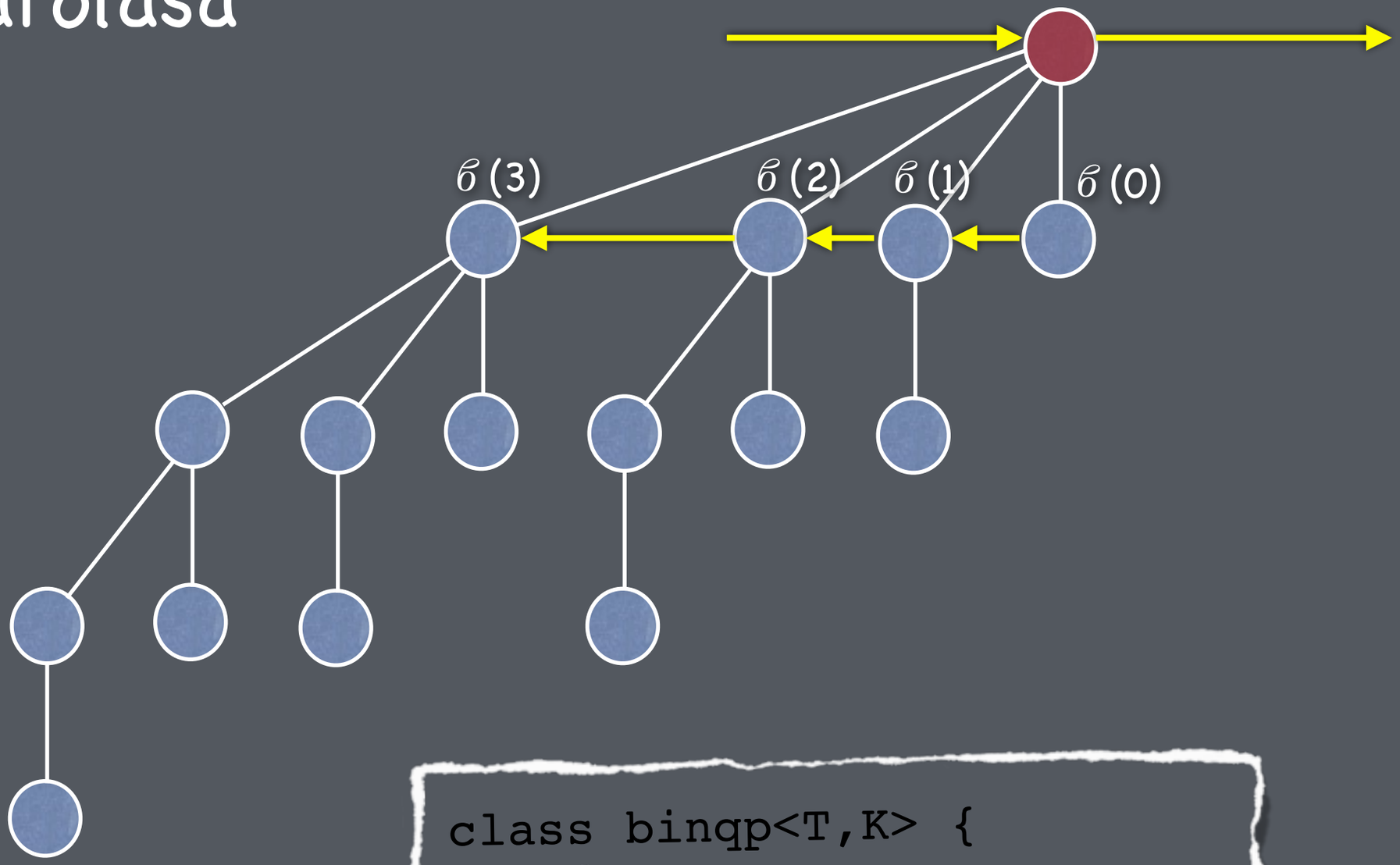

$\mathcal{B}$ (2)
$\mathcal{B}$ (3)





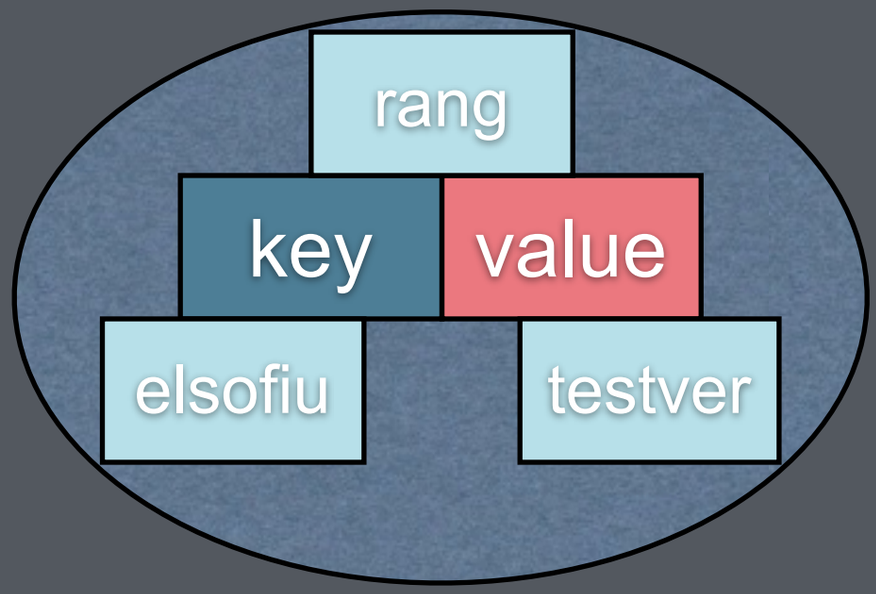


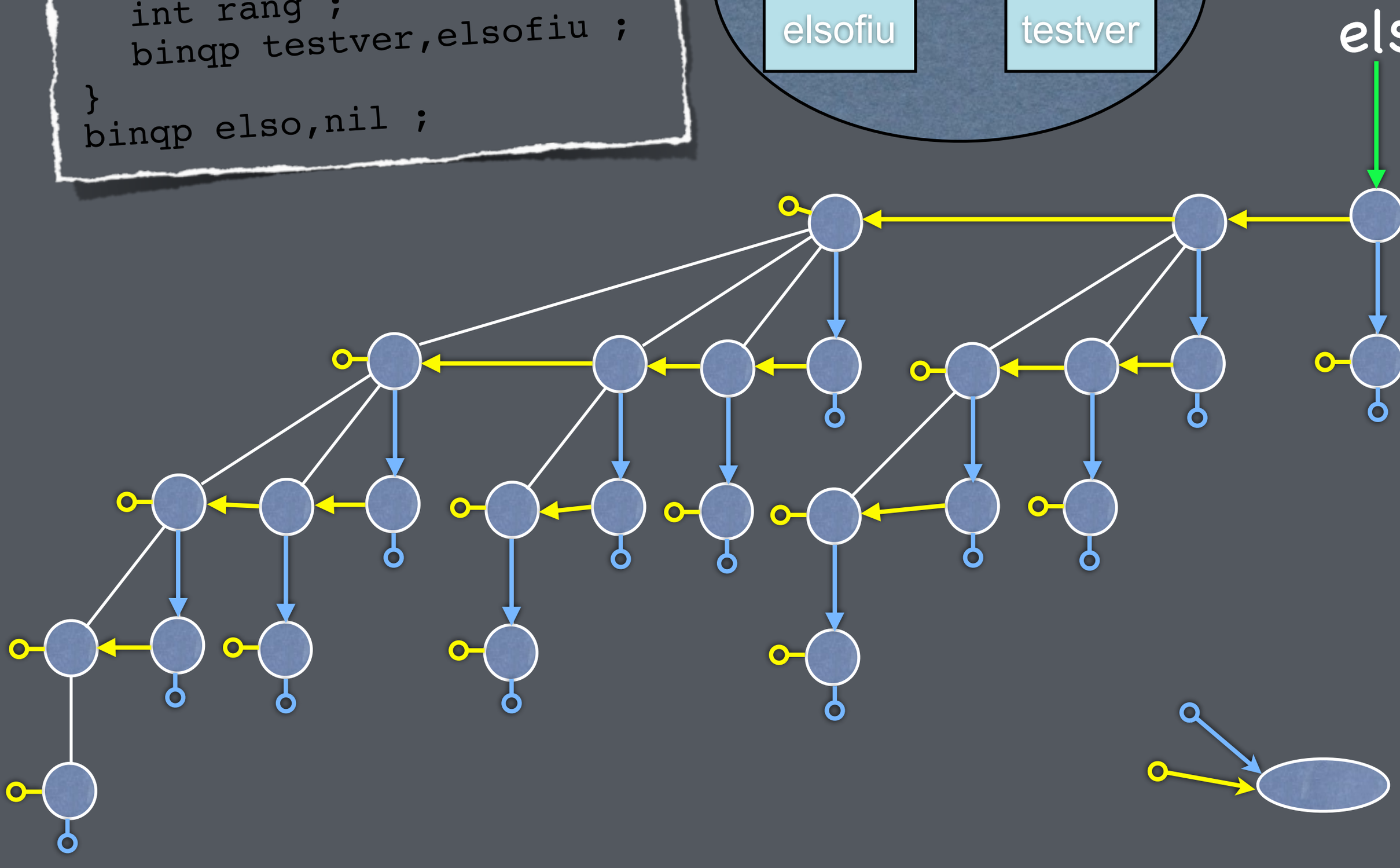
Pont tárolása




```
class binqp<T,K> {
   T key ;
   K value ;
   int rang ;
   binqp testver,elsofiu ;
}
```

```
class binqp<T,K> {
   T key ;
   K value ;
   int rang ;
   binqp testver,elsofiu ;
}
binqp elso,nil ;
```





Egyesíthető prioritási sor műveletek:

egyesít	O(logn)
sorbol	O(logn)
sorba	O(logn)

- létrehozunk egy 1 elemű kupacot
- egyesítjük az s kupaccal

létrehoz	O(1)
megszüntet	O(n)

Módosítható Prioritási sor Binomiális kupaccal

```
modosit(x,k) {
  if (k>x.kulcs) {
    x.kulcs=k ;
    y=x
    z=x.apa ;
    while(z!=nil and y.kulcs<z.kulcs) {
      csere(y kulcs,z.kulcs) ; // és érték
    y=z ;
    z=y.apa ;
    }
  } else ... (HF!)
}
```

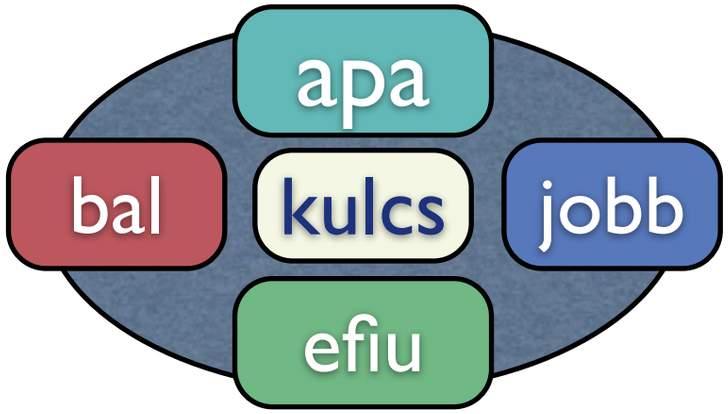
O(log(n))

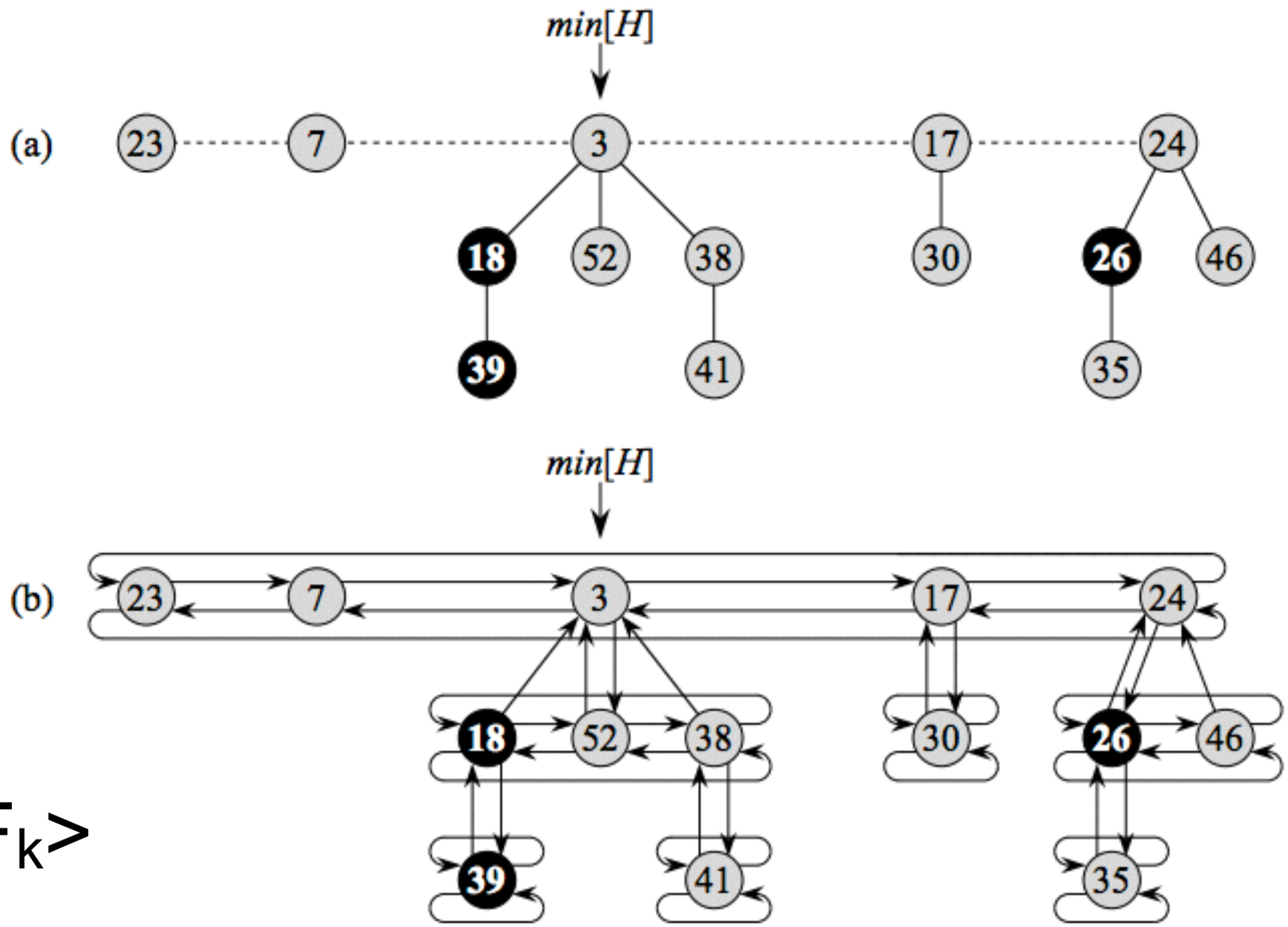
```
torol(x) {
  modosit(x,INF) ;
  sorbol() ;
}
```

O(log(n))

Fibonacci kupac

```
private class FibFaPont<E>{
        E kulcs;
        FibFaPont<E> bal, jobb, apa, efiu;
        byte fokszam;
        boolean megjelol=false;
        FibFaPont(E k){
                kulcs=k;
                this.bal=this;
                this.jobb=this;
        }
}
```





$H = \langle F_1;\ldots;F_k \rangle$

Jelölje t(H) a H sorozatbeli fák számát, m(H) pedig a megjelölt pontok számát!

$$\varphi(H) = t(H) + 2m(H)$$

Műveletek

H=egyesit(H_1,H_2);

A művelet egyszerűen megvalósítható két körlánc közönséges egyesítésével. Egyesítsük a ***H_1*** és ***H_2*** körláncot, majd ***H_1***:min és ***H_2***:min közül a kisebb kulcsot tartalmazó legyen az egyesített sorozat kijelölt minimum pontja.

$$\Delta\varphi:$$

$$\varphi(H)-(\varphi(H_1)+\varphi(H_2))=$$

$$=(t(H)+2m(H))-((t(H_1)+2m(H_1))+(t(H_2)+2m(H_2)))$$

$$=t(H)-((t(H_1)+t(H_2))+2m(H)-(2m(H_1)+2m(H_2))$$

$$=0$$

O(1)

$$\hat{c}_e = c_e + \Delta\varphi((H_1,H_2)\Rightarrow H) \qquad O(1)$$

sorba(x);

Képezzünk a beszúrandó x elemből egypontú fát, majd egyesítsük a H-t ábrázoló kupaccal.

$\Delta\varphi$: $((t(H)+1)+2m(H))-(t(H)+2m(H))=1$.

O(1)

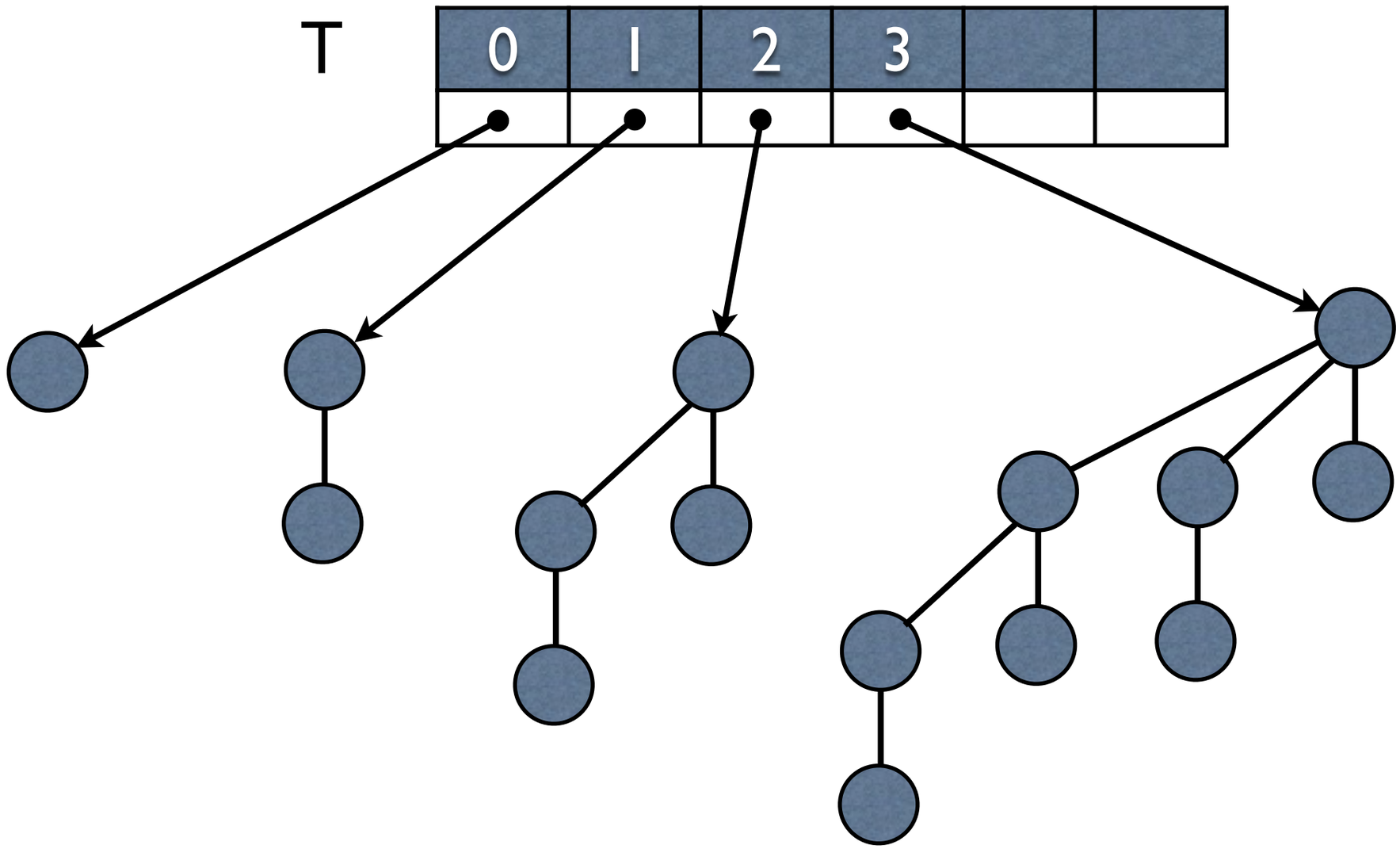
x=sorbol();

Az amortizált elemzés során feltételezzük, hogy adott egy D(n) felső korlát az n pontot tartalmazó Fibonacci kupacban szereplő csúcsok maximális fokszámára. Később az is igazolni fogjuk, hogy D(n) = O(log(n)).

Az algoritmus a minimum elem fiait átrakja a gyökérlistába, majd végrehajtja a ***kiegyenlit*** eljárást.

```
sorbol(x) {
   z=max ;
   if (z==nil) return z ;
   for (z minden x fiára) {
     x-et tegyük a gyökérlistába
     x.apa=nil ;
   }
   vegyük ki z-t a gyökérlistából
   if (z==z.jobb) max=nil ;
   else max=z.jobb ;
   kiegyenlit() ;
   elemszam-- ;
}
```

A ***kiegyenlit*** használja a ***kupszerk(y,x)*** eljárást, ami az y gyökerű fát berakja az x gyökerű fa alá. Továbbá felhasznál egy T tömböt, amelynek a mérete D(n) és amelyre T[i] az i fokszámú fát fogja tartalmazni.



```
kiegyenlit() {
   for (i=1;i<D(n);i++) A[i]=nil ;
   for (S minden w fájára) {
      x=w;
      d=x.fokszam;
      while (A[d]!=nil) {
         y=A[d] ;
         if (x.kulcs > y.kulcs) csere(x,y)
         kupszerk(y,x);
         A[d]=nil;
         d++;
      }
      A[d]=x;
   }
   max=nil ;
   for (i=1;i<D(n);i++)
      if (A[i]!=nil) {
         "tegyük A[i]-t a gyökérlistába"
         if (max==nil or A[i].kulcs>max.kulcs) min=A[i] ;
      }
}
```

```
kupszerk(y,x) {
   "vegyük ki y-t S-ből"
   "tegyük y-t x egy fiává"
   "növeljük x fokszámát"
   x.megjelolt=false;
}
```

A gyökérlista mérete legfeljebb D(n) +t(H)-1,
így az aktuális költség O(D(n) +t(H))

A potenciál a min. pont kivágása előtt: t(H) +2m(H)

A kivágás után D(n) +1+2m(H)

$$O(D(n)+t(H)) + ((D(n)+1)+2m(H))-(t(H)+2m(H))$$

$$= O(D(n)) + O(t(H))-t(H) = O(D(n))$$

`modosit(x,k)`

Az algoritmus kivágja az adott elemet ha sérül a kupactulajdonság, továbbá a KASZKÁD-VÁGÁS algoritmus segítségével gondoskodik arról, hogy ne legyen olyan pont, ami több fiát is elveszti.

```
modosit(x,k) {
  if (k>x.kulcs) {
    x.kulcs=k ;
    y=x.apa ;
    if (y!=Nil and x.kulcs<y.kulcs) {
      kivag(x) ;
      kaszkadvag(y) ;
    }
    if (x.kulcs<max.kulcs) min=x ;
  } else ...
}
```

```
kivag(x) {
   vegyük ki x-et x.apa fiainak listájából
   tegyük bele x-et a gyökérlistába
   x.apa=nil ;
   x.megjelolt=False ;
}


kaszkadvag(y) {
   z=y.apa ;
   if (z!=nil) {
      if (y.megjelolt==False) y.megjelolt=True ;
      else kivag(y) ;
      kaszkadvag(z) ;
   }
}
```

A ***modosit*** algoritmus futási idejének elemzése:

1. A tényleges költség kiszámítása.
   Tegyük fel, hogy ***modosit*** adott hívására c számú ***kaszkadvag*** hajtódik végre. Ekkor ***modosit*** tényleges futási ideje O(c).
2. A potenciál változás:
   A ***kaszkadvag*** minden hívása, az utolsó kivételével, kivesz egy fapontot a fából és beteszi azt H gyökérlistájába. Így a ***kaszkadvag***-ok végrehajtása után t(H)+c fa lesz a gyökérlistában. (t(H) volt eredetileg, az x pont, és még másik c-1 pont került be a vágások eredményeként). Legfeljebb m(H)-c+2 pont lesz megjelölve, mert c-1 pont jelöletlenné vált a ***kaszkadvag***-ok miatt, és esetleg egyet még jelöletlenné tesz.

A potenciál változása: ((t(H) +c) +2(m(H)-c+2))-(t(H) +2m(H)) = 4-c

Tehát a ***modosit*** eljárás amortizált futási ideje legfeljebb O(c) +4-c = O(1)

A gyökérlista mérete legfeljebb D(n).

A maximális fokszám felső korlátja:

Lemma.
Legyen x a H Fibonacci-kupac tetszőleges pontja, amelynek fokszáma k.
Jelölje az $y_1, y_2, \ldots, y_k$ sorozat x fiait abban az időrendi sorrendben, ahogyan x fiai lettek!
Ekkor
y_1.fokszam ≥ 0 és
y_i.fokszam ≥ i - 2 , ∀i = 2, ..., k-ra.

Bizonyítás.
Az y_1.fokszam ≥ 0 állítás nyilvánvaló.
i ≥ 2 esetén vegyük észre, hogy amikor y_i-t x-hez kapcsoltuk annak fiaként, akkor már az y_1, ..., y_{i-1} pontok x fiai voltak, így az x.fokszam = i -1 fennállt.
Az y_i pontot csak akkor kapcsoltuk x-hez, ha y_i.fokszam = x.fokszam, tehát y_i.fokszam = i - 1 is teljesült.
Azóta y_i legfeljebb egy fiát veszthette el, mert egyébként x-ből kivágtuk volna.

Tehát y_i.fokszam ≥ i - 2.

Tekintsük a következő Fibonacci számsorozatot:

$$F_k = \begin{cases} 0 & \text{ha } k = 0\,, \\ 1 & \text{ha } k = 1\,, \\ F_{k-1} + F_{k-2} & \text{ha } k \geq 2\,. \end{cases}$$

k	F_k	$\sum_{i=0}^{k} F_i$	$1+\sum_{i=0}^{k} F_i$
1	1	1	2
2	1	2	3
3	2	4	5
4	3	7	8
5	5	12	13
6	8	20	21
7	13	33	34
8	21	54	55

Lemma.

Minden k ≥ 0 egész számra:

$$F_{k+2} = 1 + \sum_{i=0}^{k} F_i\,.$$

Bizonyítás

k-szerinti indukcióval bizonyítunk.

k = 0-ra:

$$1+\sum_{i=0}^{0} F_i \;=\; 1+F_0 \;=\; 1+0 \;=\; 1 \;=\; F_2$$

Az indukciós feltétel szerint $F_{k+1} = 1+\sum_{i=0}^{k-1} F_i$

tehát:

$$F_{k+2} \;=\; F_k+F_{k+1} \;=\; F_k+\left(1+\sum_{i=0}^{k-1} F_i\right) \;=\; 1+\sum_{i=0}^{k} F_i$$

Lemma

Minden $k \geq 0$ egész számra $F_{k+2} \geq \Phi^k$.

Bizonyítás

Bizonyítás k-szerinti indukcióval.

$F_2 = 1 = \Phi^0$, $F_3 = 2 \geq (1+\sqrt{5})/2 = \Phi^1$.

$$F_{k+2} = F_k + F_{k+1} \geq \Phi^{k-2} + \Phi^{k-1} = \Phi^{k-2}(1+\Phi)$$

$$\Phi^2 = (1+\Phi)$$

$$F_{k+2} \geq \Phi^{k-2}\Phi^2 = \Phi^k$$

Lemma.
Legyen x a H Fibonacci-kupac tetszőleges pontja, amelynek fokszáma k. Ekkor $méret(x) \geq \Phi^k$

Bizonyítás

Jelölje s_k a legkevesebb pontot tartalmazó Fibonacci-kupac k fokszámú fájának pontjai számát!

$$s_0 = 1, s_1 = 2, s_2 = 3$$

s_k értéke k-szerint monoton nő

Jelölje az $y_1, y_2, ..., y_k$ sorozat x fiait abban az időrendi sorrendben, ahogyan x fiai lettek!

$$méret(x) \geq s_k = 2 + \sum_{i=2}^{k} s_{y_i.fokszam} \geq 2 + \sum_{i=2}^{k} s_{i-2} \geq 2 + \sum_{i=2}^{k} F_i = 1 + \sum_{i=0}^{k} F_i = F_{k+2}$$

Tehát $méret(x) \geq s_k \geq F_{k+2} \geq \Phi^k$.

következmény:

Egy n pontot tartalmazó Fibonacci-kupac tetszőleges pontjának maximális fokszáma: $D(n) = O(\lg n)$.

Bizonyítás:

$$n \geq \text{méret}(x) \geq \Phi^k$$

$$k \leq \log_\Phi n$$

Így bármely pont maximális fokszáma $D(n) = O(\lg n)$

```
torol(x) {
  modosit(x,INF) ;
  sorbol() ;
}
```

O(log(n))

BinQpac vs. FibQpac

	BinQpac	FibQpac
sorba	O(logn)	*O(1)
sorbol	O(logn)	*O(logn)
egyesit	O(logn)	*O(1)
modosit	O(logn)	*O(1)
torol	O(logn)	*O(logn)

[1] T. H. Cormen, C. E. Leiserson, R.L. Rivest: Algoritmusok
Műszaki Könyvkiadó, 2003.

[2] Imreh Csanád: Algoritmusok és Adatszerkezetek II. 2009
http://www.inf.u-szeged.hu/~cimreh/n10alg26el.pdf

[3] Horváth Gyula: Algoritmusok és Adatszerkezetek II. 2008

http://www.inf.u-szeged.hu/~tnemeth/a2anyagok/v10.pdf

http://www.inf.u-szeged.hu/~tnemeth/a2anyagok/v13.pdf