



Rendezések

9. előadás

Rendezések

- A rendezési probléma:
 - Bemenet:
 - n számot tartalmazó $(a_1, a_2, \dots, a_n)$ sorozat
 - Kimenet:
 - a bemenő sorozat olyan $(a'_1, a'_2, \dots, a'_n)$ permutációja, hogy $a'_1 \leq a'_2 \leq \dots \leq a'_n$

Rendezési reláció

- Legyen U egy halmaz, és „ $<$ ” egy kétváltozós reláció U -n
- Ha $a, b \in U$ és $a < b$, akkor azt mondjuk, hogy „ a kisebb, mint b ”
- A „ $<$ ” reláció egy rendezés, ha teljesülnek a következők:
 1. $a \not< a: \forall a \in U$ elemre ($<$ irreflexív);
 2. Ha $a, b, c \in U$, $a < b$, és $b < c$, akkor $a < c$ ($<$ tranzitív);
 3. Tetszőleges $a \neq b \in U$ elemekre vagy $a < b$, vagy $b < a$ fennáll ($<$ teljes)
- Ha „ $<$ ” egy rendezés U -n, akkor az $(U; <)$ párt rendezett halmaznak
- nevezzük
- Példa:
 - $\mathbb{Z}$ az egész számok halmaza. A „ $<$ ” rendezés a nagyság szerinti rendezés
 - C a karakterek halmaza, a rendezést a karakterek kódja adja

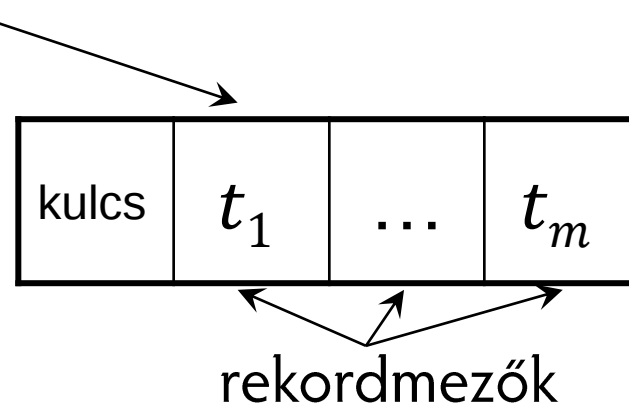
Rendezések

- Általánosabban:

- Legyen K egy teljesen rendezett halmaz, a kulcsok halmaza
- Legyenek T_i -k tetszőleges típusok $i \in [1, m]$

$$E = K \times \prod_{i=1}^m T_i$$

E egy eleme:



Rendezések

- A cél: $S \in E^*$ rendezése.
- Legyen $n = |S|$
- S rendezett $\Leftrightarrow \forall i \in [1, n - 1]: S_i.\text{kulcs} \leq S_{i+1}.\text{kulcs}$
- Előfeltétel: $S = S' \in E^*$
- Utófeltétel: S rendezett és $S \in \text{Perm}(S')$

Rendezések

- Például:
- Személy = Név × Magasság × Születési_év

Abigél	Janka	Zsuzsi	Dávid	Dorka
167	164	158	160	162
1996	1998	2001	2000	2002

- Akármelyiket választhatjuk kulcsnak – mindegyiken értelmezhető rendezés.

Rendezések

- Ha a név a kulcs:

Abigél	Dávid	Dorka	Janka	Zsuzsi
167	160	162	164	158
1996	2000	2002	1998	2001

Rendezések

- Ha a születési év a kulcs:

Abigél	Janka	Dávid	Zsuzsi	Dorka
167	164	160	158	162
1996	1998	2000	2001	2002

Rendezések

- Osztályozás:
 1. $m = 0$ – skalár rendezők
 $m \geq 1$ – rekord rendezők
 2. Belső rendezők:
központi memória + indexelés
Külső rendezők:
háttértárolón
 3. Összehasonlításos rendezők
(kulcsok értékét hasonlítjuk)
Edényrendezők
(kulcsok értéke szerint szétrakjuk)

Rendezések

4. Helyben rendezők
(segéd memória konstans)
Nem helyben rendezők
5. Stabil rendezők
(azonos kulcsú rekordok sorrendje nem változik)
Nem stabil rendezők
6. Előrendezéshez illeszkedő és nem illeszkedő rendezők
(kevesebbet dolgozik-e, ha a sorozat előrendezett)

Rendezések

7. Használt adatszerkezet szerint

- lineáris adatszerkezet
- fa

8. Módszer szerint:

Például összehasonlításos rendezőknél:

- Maximális elemet kiválasztó
- Csererendezők
- Egy elemet helyre vivők
- Összefuttatásos rendezők

Rendezések

- Milyen hatékony egy algoritmus?
 - Legtöbbször csak a lépésszám nagyságrendje érdekes.
 - Hogyan függ a lépésszám az input méretétől?
 - Az input méretét legtöbbször n -nel jelöljük.
 - A lépésszám ennek egy f függvénye, azaz ha n méretű az input, akkor az algoritmus $f(n)$ lépést végez.
 - Igazából az f függvény az érdekes.
 - $100n$ vagy $101n$, általában mindegy
 - n^2 vagy n^3 már sokszor nagy különbség, de néha mindegy
 - n^2 vagy $2n$ már mindig nagy különbség

Függvények nagyságrendje

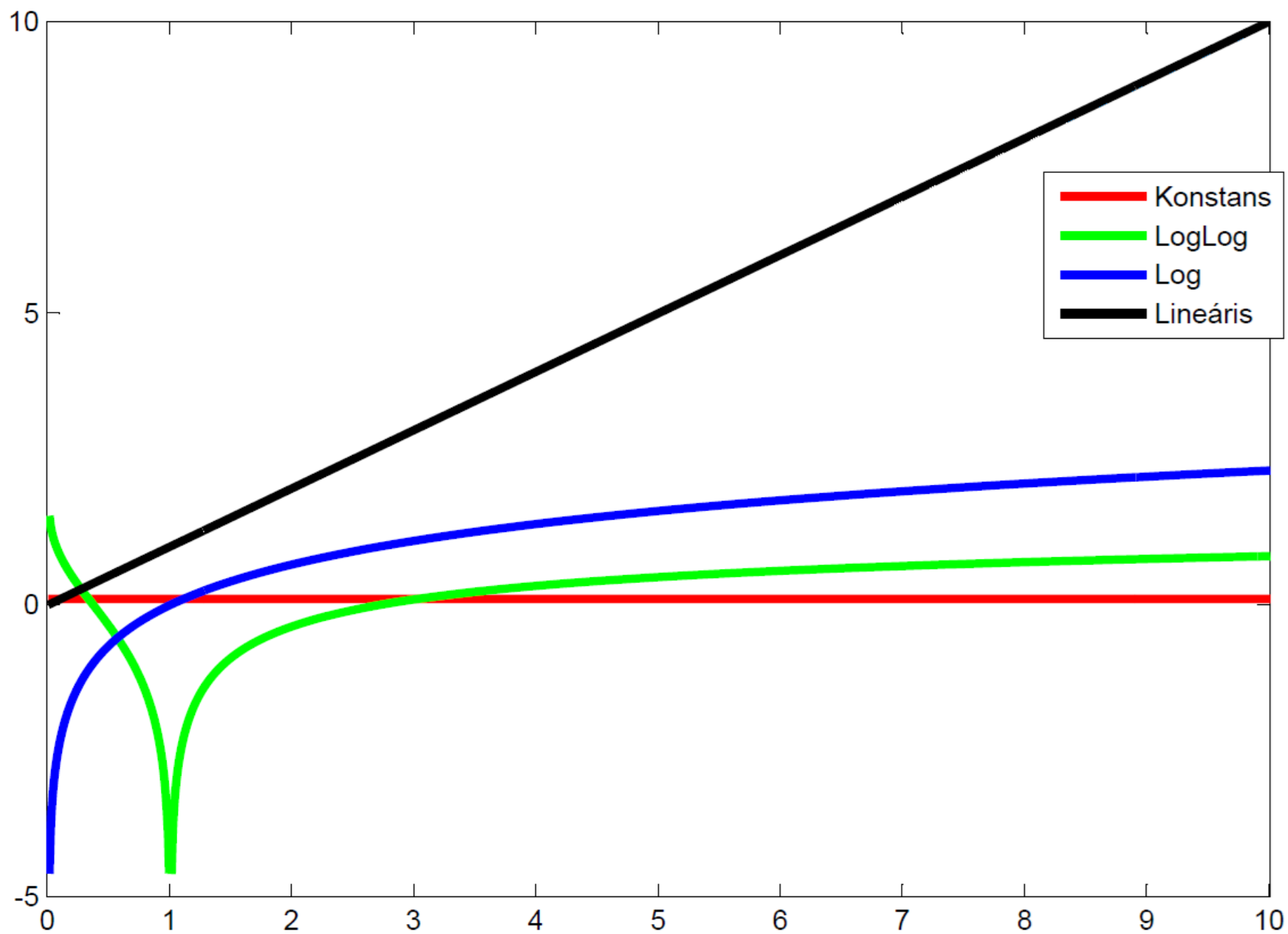
- Definíció – A g **aszimptotikus felső korlátja** f -nek
 - Ha $f(x)$ és $g(x)$ az R^+ egy részhalmazán értelmezett valós értékeket felvevő függvények, akkor $f = \mathcal{O}(g)$ jelöli azt a tényt, hogy vannak olyan $c, k > 0$ állandók, hogy $|f(x)| \leq c * |g(x)|$ teljesül, ha $x \geq k$.
- Például:
 - $100n + 300 = \mathcal{O}(n)$
 - hiszen $k = 300$; $c = 101$ -re teljesülnek a feltételek
 - $100n + 300 \leq 101n$, ha $n \geq 300$

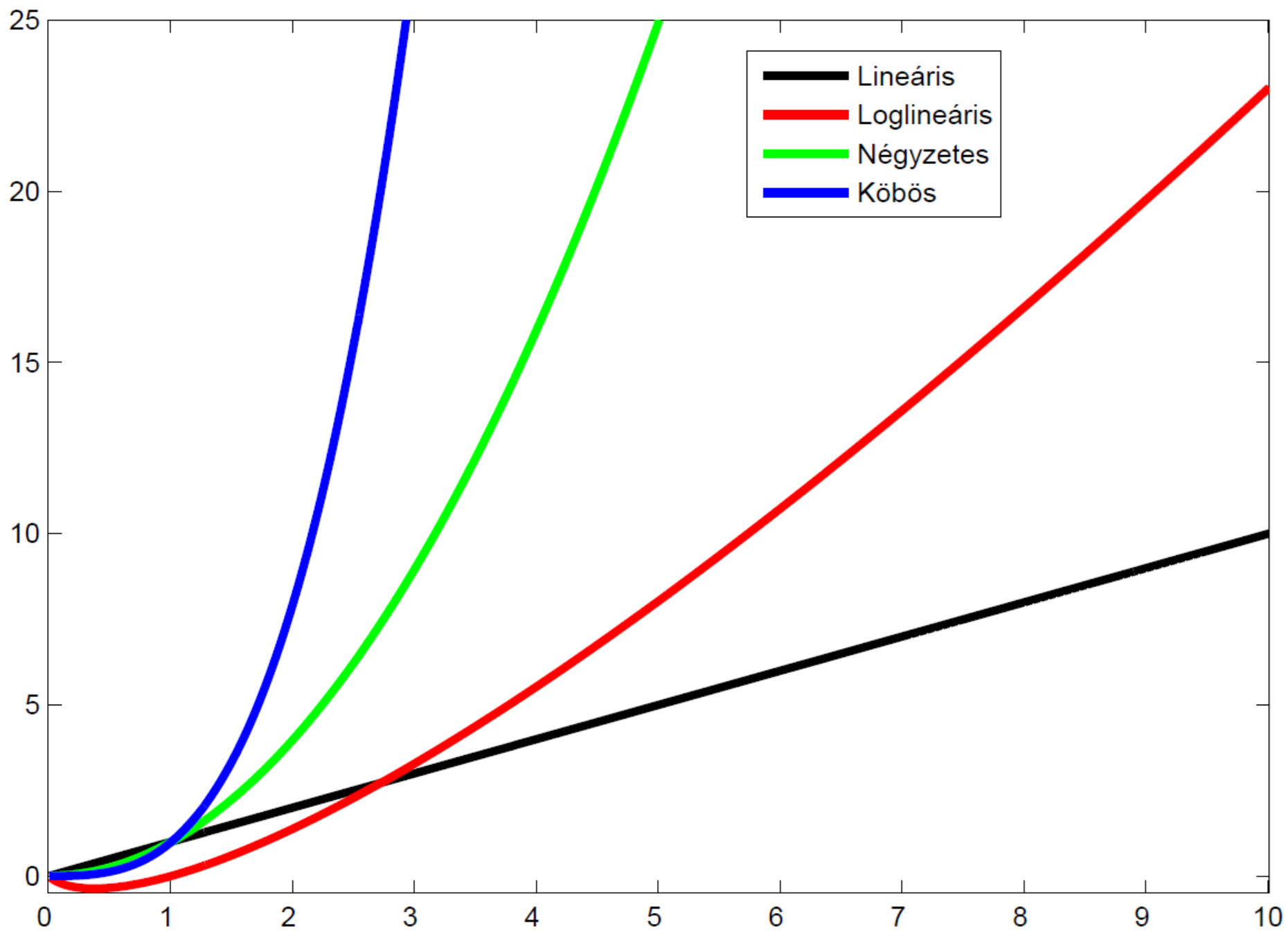
Függvények nagyságrendje

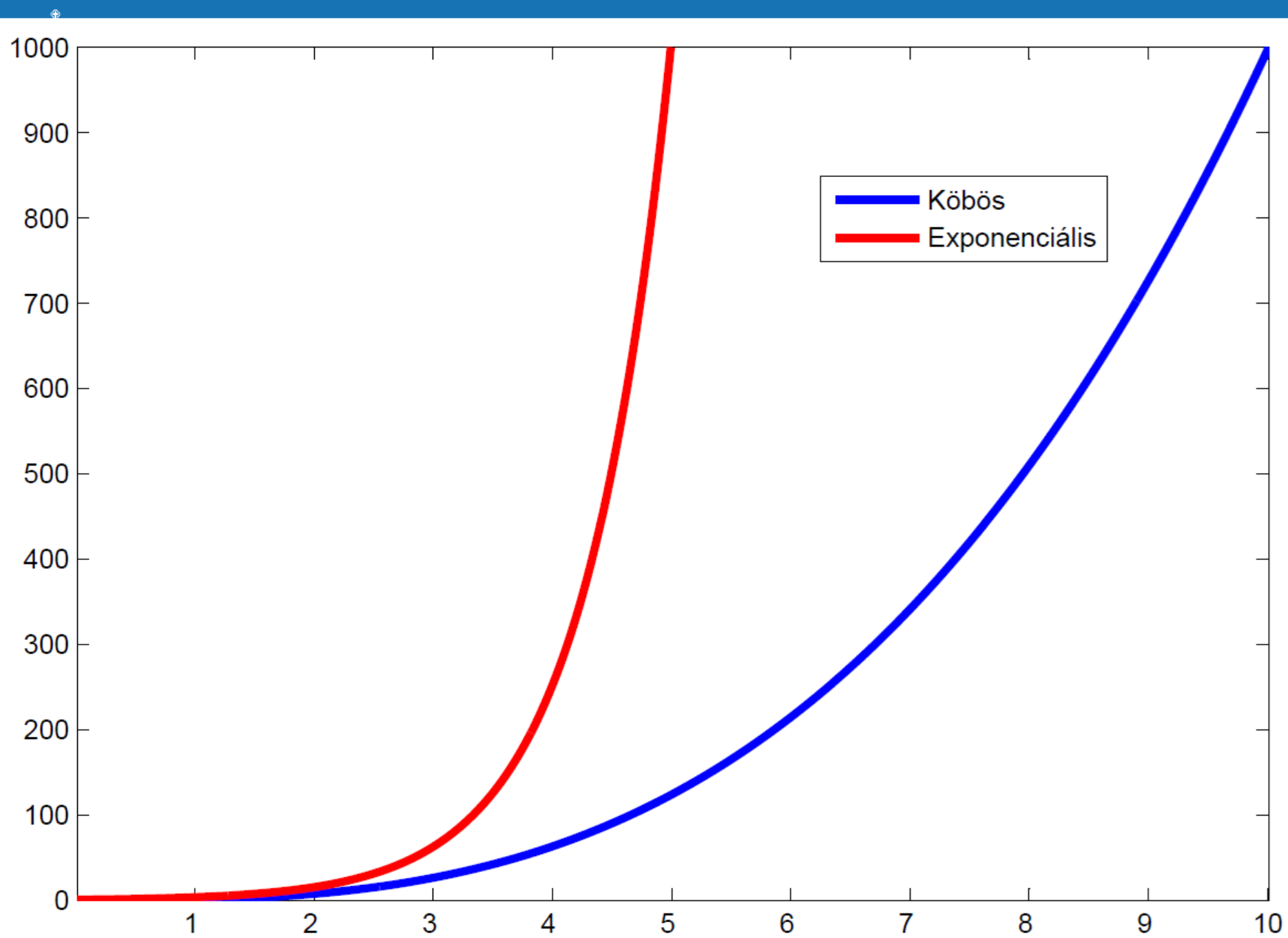
- Definíció – A g **aszimptotikus alsó korlátja** f -nek
 - Ha $f(x)$ és $g(x)$ az R^+ egy részhalmazán értelmezett valós értékeket felvevő függvények, akkor $f = \Omega(g)$ jelöli azt a tényt, hogy vannak olyan $c, k > 0$ állandók, hogy $|f(x)| \geq c * |g(x)|$ teljesül, ha $x \geq k$.
- Például:
 - $100n - 300 = \Omega(n)$
 - hiszen $k = 300$; $c = 99$ -re teljesülnek a feltételek

Függvények nagyságrendje

- Definíció – A g **aszimptotikus éles korlátja** f -nek
 - Ha $f = \Omega(g)$ és $f = \mathcal{O}(g)$ egyaránt teljesül, akkor $f = \Theta(g)$
- Például:
 - $100n - 300 = \Theta(n)$









Négyzetes rendezők

Lassú rendezők

Buborék rendezés

- Feladat: Rendezzük az $A[1 \dots n]$ vektort!
A vektor elemtípusa tetszőleges T típus, amire egy teljes rendezés értelmezhető
- Buborék rendezés alapötlete:
 - a vektor elejétől kezdve „felbuborékolatjuk” a legnagyobb elemet. Utána ugyanezt tesszük az eggyel rövidebb vektorra, stb. Végül, utoljára még az első két elemre is végrehajtjuk a „buborékolatást”

Buborék rendezés

- Egy sorozat rendezett $\Leftrightarrow$ nincs az elemek között inverzió
- Ez a rendezés az inverziók csökkentésével rendez

12	5	7	9	11	10
----	---	---	---	----	----

Buborék rendezés

12	5	7	9	11	10
----	---	---	---	----	----

Buborék rendezés



5	12	7	9	11	10
---	----	---	---	----	----

Buborék rendezés

5	12	7	9	11	10
---	----	---	---	----	----

Buborék rendezés



5	7	12	9	11	10
---	---	----	---	----	----

Buborék rendezés

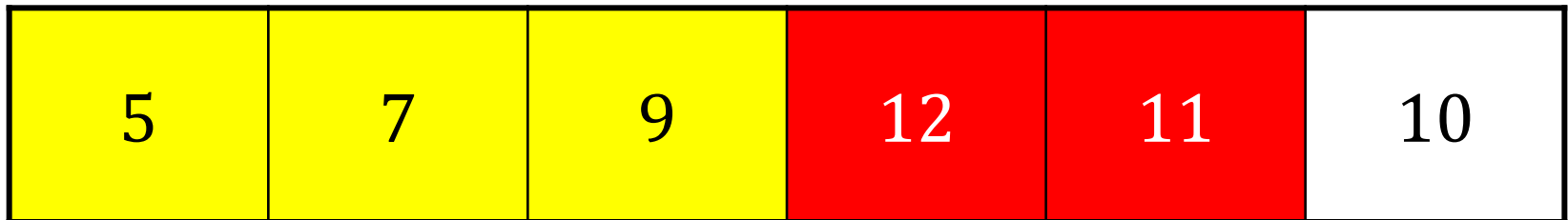
5	7	12	9	11	10
---	---	----	---	----	----

Buborék rendezés

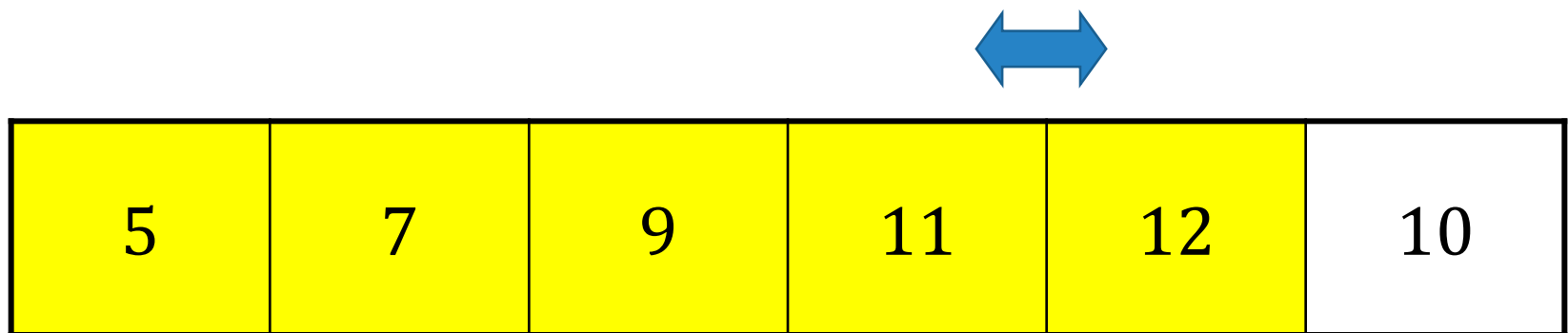


5	7	9	12	11	10
---	---	---	----	----	----

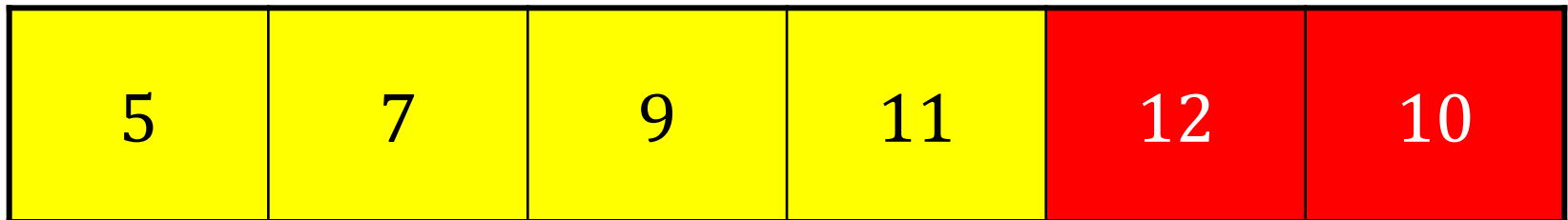
Buborék rendezés



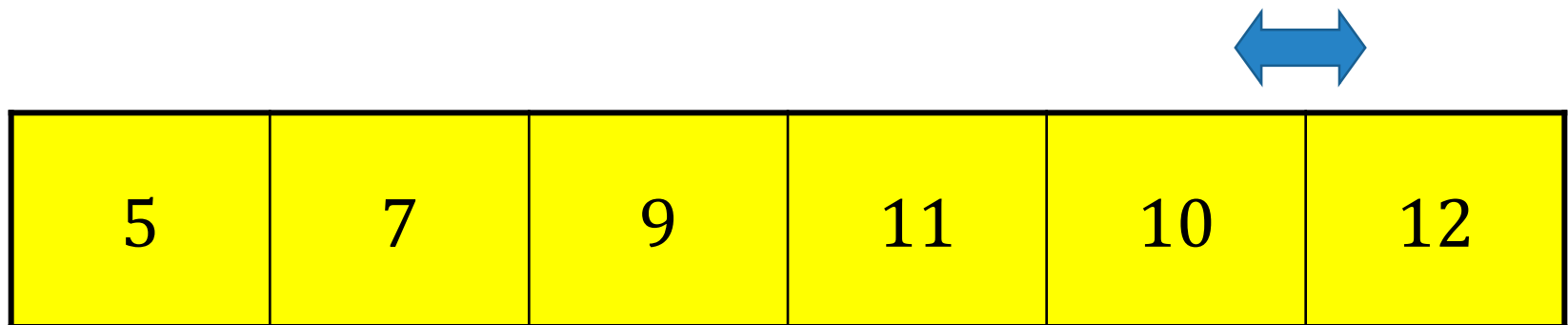
Buborék rendezés



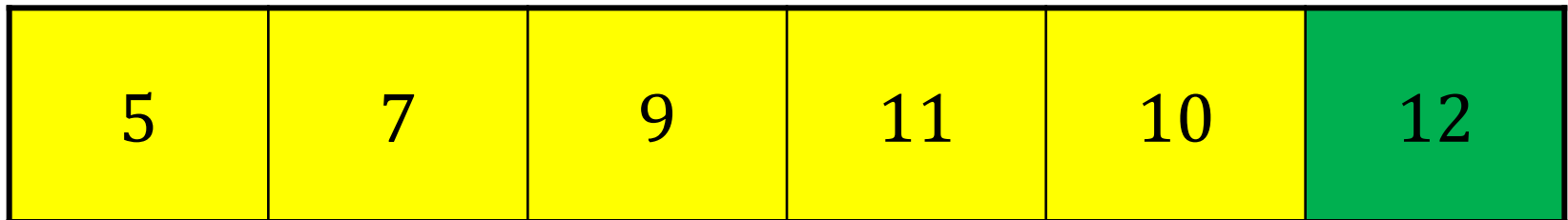
Buborék rendezés



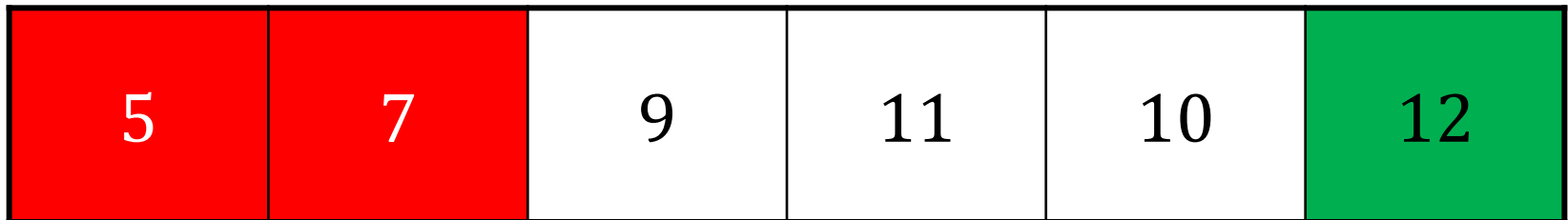
Buborék rendezés



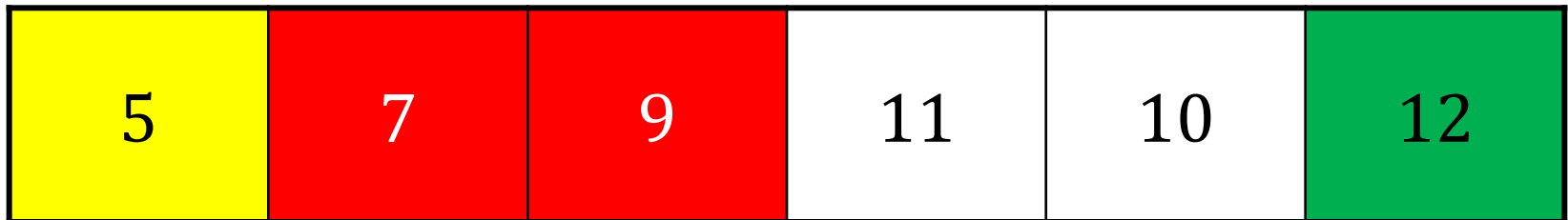
Buborék rendezés



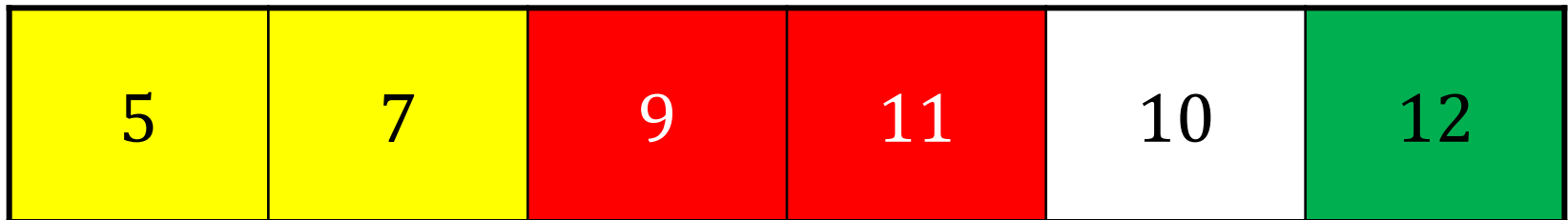
Buborék rendezés



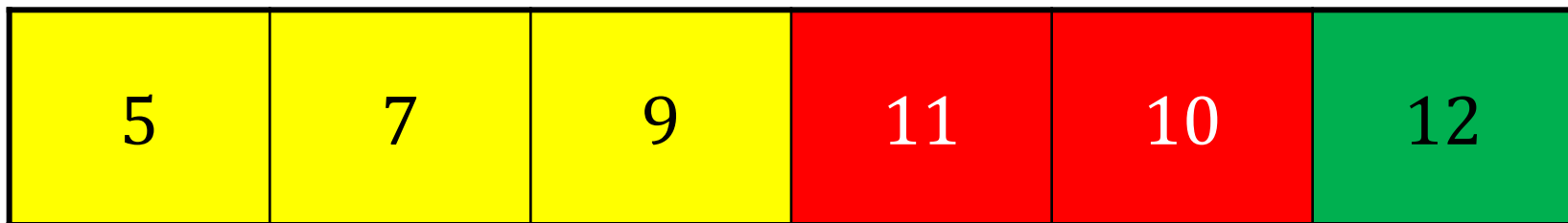
Buborék rendezés



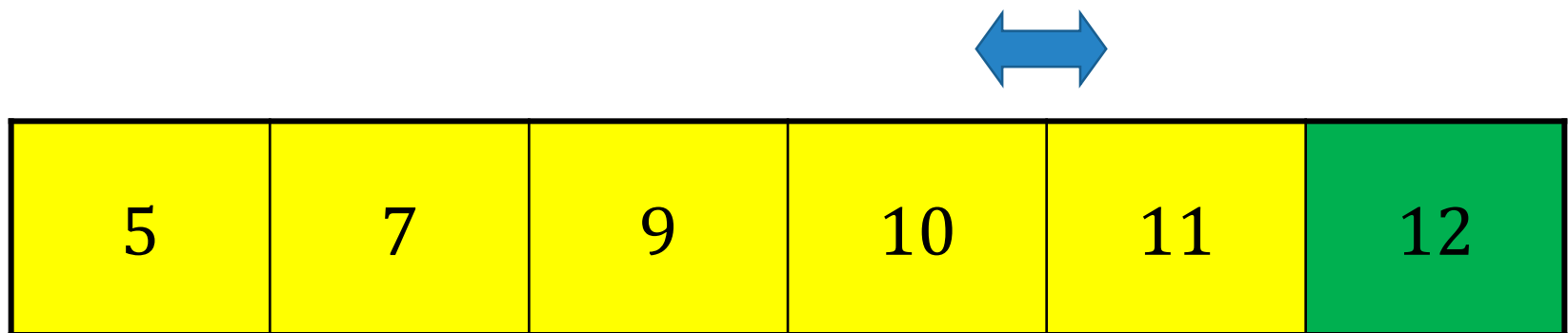
Buborék rendezés



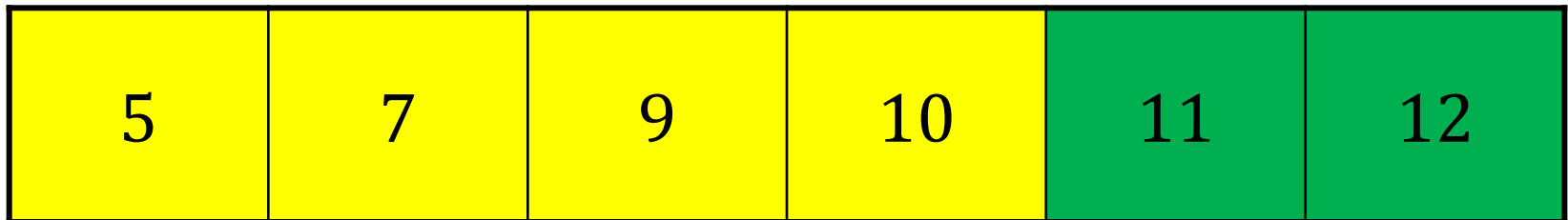
Buborék rendezés



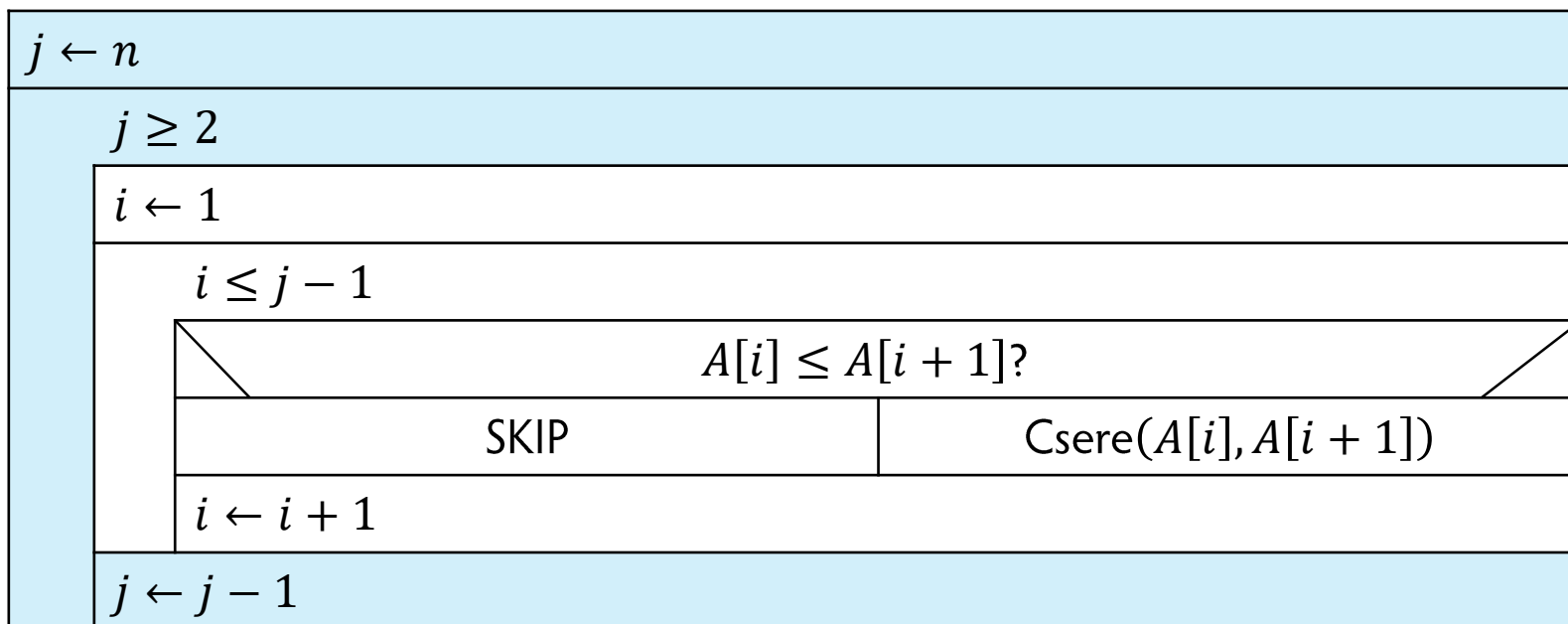
Buborék rendezés



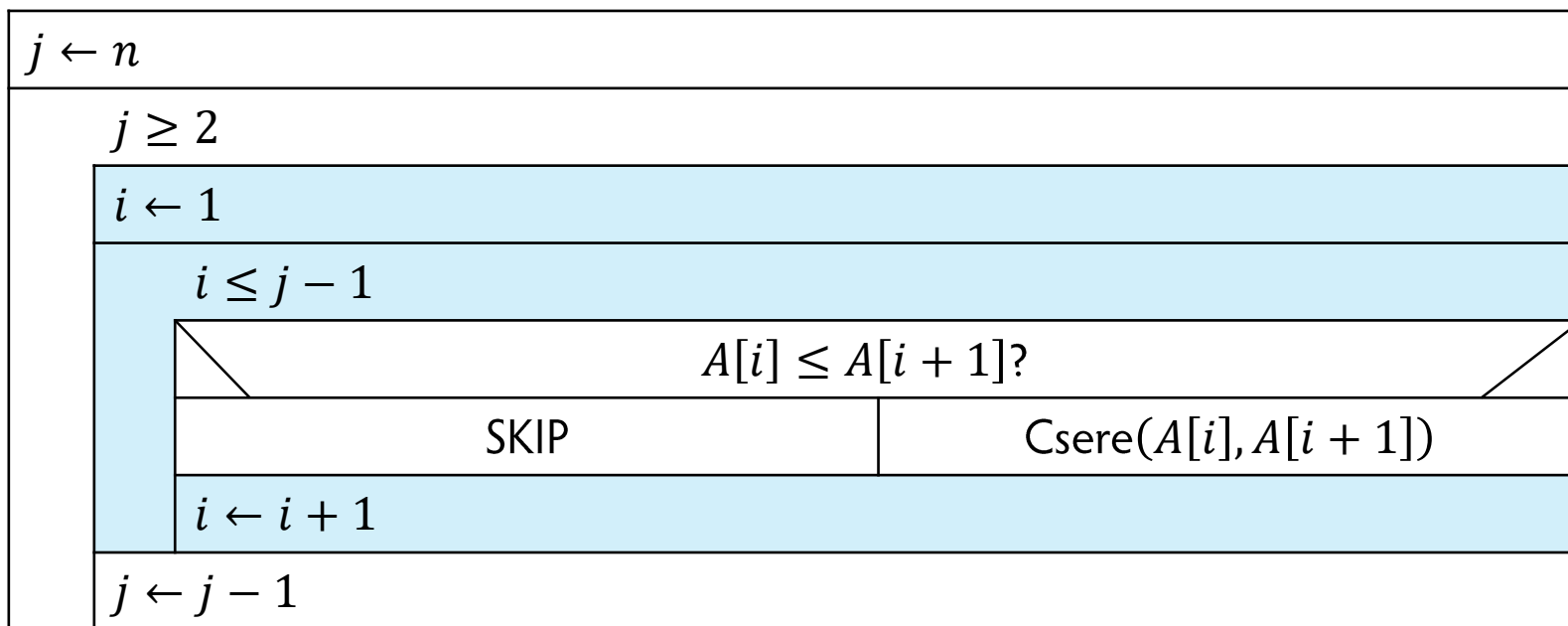
Buborék rendezés



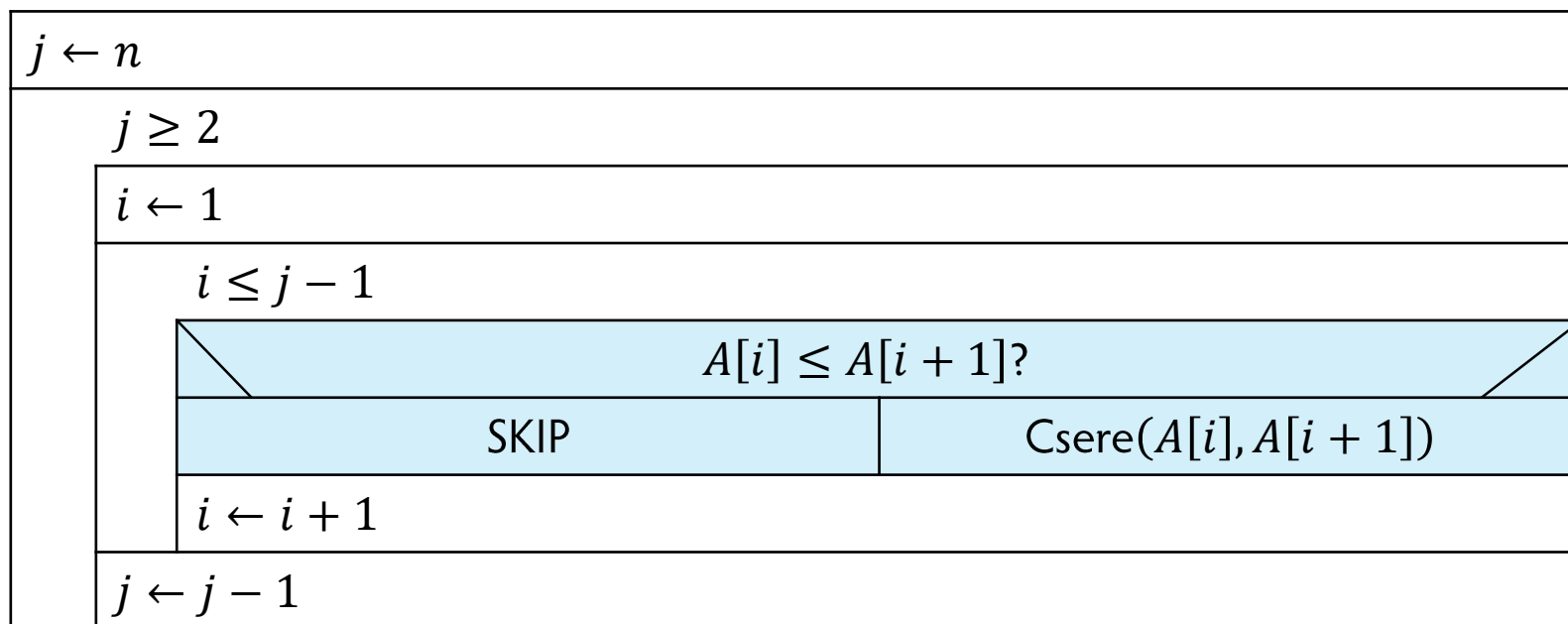
Buborék rendezés algoritmus



Buborék rendezés algoritmus



Buborék rendezés algoritmus



Buborék rendezés algoritmus

- Műveletek időigénye

	$j \leftarrow n$
C_1	$j \geq 2$
C_2	$i \leftarrow 1$
C_1	$i \leq j - 1$
C_2	$A[i] \leq A[i + 1]?$ SKIP Csere($A[i], A[i + 1]$)
C_3	
C_4	
C_1	$i \leftarrow i + 1$
C_1	$j \leftarrow j - 1$

Buborék rendezés – időráfordítás

- Külső ciklus
 - $n - 1$ –szer fut le, előtte kezdő értékadás
 - A feltételt n -szer ellenőrzi, j -t $n - 1$ -szer csökkenti
 - $c_1 + n * c_2 + (n - 1) * c_1$
- Belső ciklus lefutásainak száma: $n - 1, n - 2, \dots, 2, 1$
 - mindannyiszor 1 kezdőértékadás
 - a ciklusfeltétel eggyel többször ellenőrzi
 - i -t növeljük $(1 + \dots + n - 1)$ -szer
 - $(n - 1) * c_1 + (2 + \dots + n) * c_2 + (1 + \dots + n - 1) * c_1$
- $A[i]$ és $A[i + 1]$ összehasonlításainak száma:
 - $(1 + \dots + n - 1) * c_3$
- A cserék száma A -tól függ = A inverzióinak száma
 - Ez 0 és $\binom{n}{2}$ között van
 - $\text{inv}(A) * c_4$

Buborék rendezés – összegezve

- $$T(A) = c_1 + n * c_2 + (n - 1) * c_1 +$$

$$(n - 1) * c_1 + (2 + \dots + n) * c_2 + (1 + \dots + n - 1) * c_1 +$$

$$(1 + \dots + n - 1) * c_3 +$$

$$\text{inv}(A) * c_4 =$$
- $$c_1 + n * c_2 + n * c_1 + n * c_1 - c_1 +$$

$$(n + 2) * \frac{n - 1}{2} * c_2 + n * \frac{n - 1}{2} * c_1 +$$

$$n * \frac{n - 1}{2} * c_3 +$$

$$\text{inv}(A)^2 * c_4$$
- $$n^2 * \left(\frac{c_1}{2} + \frac{c_2}{2} + \frac{c_3}{2} \right) + n * \left(3 * \frac{c_1}{2} + 3 * \frac{c_2}{2} - \frac{c_3}{2} \right) - (c_1 - c_2) +$$

$$\text{inv}(A) * c_4$$

Buborék rendezés

- Néhány egyszerűsítő feltételezés

1. Feltételezés

$$c_1 \ll c_3 \text{ és } c_2 \ll c_4$$

vagyis az elemek összehasonlítása és cseréje adja a költség javát

- Ekkor

$$T(A) \approx n^2 * \frac{c_3}{2} - n * \frac{c_3}{2} + \text{inv}(A) * c_4 = n * \frac{n-1}{2} * c_3 + \text{inv}(A) * c_4$$

2. Feltételezés:

c_3 és c_4 az egyes gépekre jellemző állandók. Ha történetesen $c_3 \approx c_4$, akkor a végrehajtási időt jellemzi a domináns műveletek száma:

- $T(A) \approx n * \frac{n-1}{2} + \text{inv}(A)$

$T(A)$ helyett $\bar{T}(n)$ -t bevezetve, a szokásos írásmód:

$$\bar{T}(n) \approx n * \frac{n-1}{2} + \text{inv}(A)$$

Buborék rendezés

- Néhány egyszerűsítő feltételezés

- 3. Feltételezés:

- általában külön-külön kérdezzük az egyes műveletek számát:

- $\ddot{O}(n) \approx n * \frac{n-1}{2}$

- $Cs(n) \approx \text{inv}(A)$

- 4. Feltételezés:

- ismerjük a lehetséges bejövő input adatokat, kiszámíthatjuk

- a legrosszabb ($M T(n)$) és
 - a legjobb esetben ($m T(n)$)
 - átlagos esetben ($A T(n)$)

Buborék rendezés

- Az előzőekből:
 - Az összehasonlítások száma minden n hosszú vektorra ugyanannyi:
 - $M \ddot{O}(n) = m \ddot{O}(n) = A \ddot{O}(n) = n * \frac{n-1}{2}$
 - A cserék száma:
 - $M Cs(n) = n * \frac{n-1}{2}$
 - $m Cs(n) = 0$
 - $A Cs(n) = n * \frac{n-1}{4} = M \frac{Cs(n)}{2}$
 - Ez bizonyítható

Buborék rendezés

- Az előzőek pontos értékek, azonban gyakran a költség (műveletszám) nagyságrendje, aszimptotikus viselkedése a hasznos információ
 - $M \ddot{O}(n) = n * \frac{n-1}{2} = \Theta(n^2)$
 - $M Cs(n) = n * \frac{n-1}{2} = \Theta(n^2)$
 - $A Cs(n) = n * \frac{n-1}{4} = M \frac{Cs(n)}{2} = \Theta(n^2)$

Buborék rendezés

- Javított változat
 - Ha nincs már csere, leáll.
 - Cserék száma nem változik
 - $M \ddot{O}(n)$ nem változik
 - $m \ddot{O}(n) = n - 1$
 - $A \ddot{O}(n) = ?$ (sejtés: n^2)

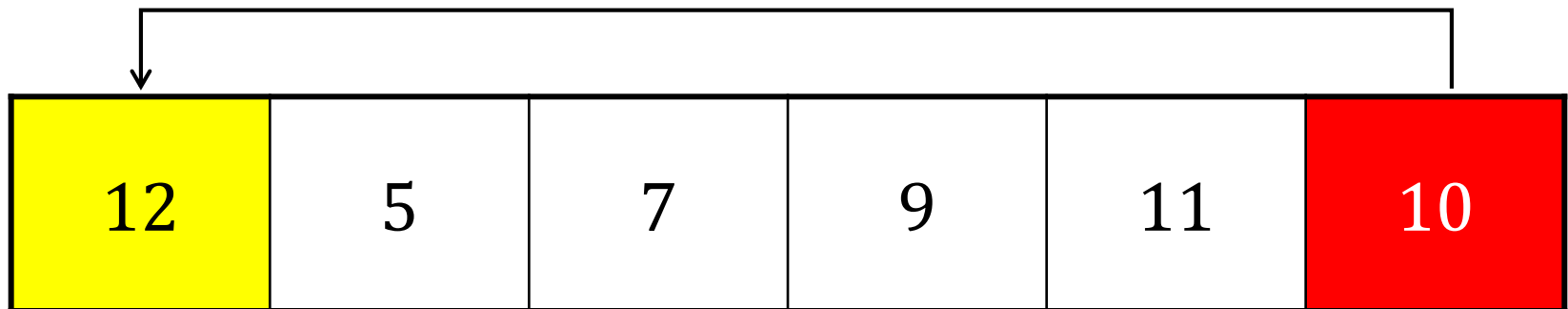
Maximum kiválasztásos rendezés

- Ötlet: feltételezzük, hogy az $A[1..n]$ tömb jobb szélé ($j + 1..n$) már rendezve van, minden lépésben kiválasztjuk az $A[1..j]$ résztömb maximális elemét, és kicseréljük a j . helyen lévővel, majd j -t csökkentjük
 - Kezdetben j értéke n .

12	5	7	9	11	10
----	---	---	---	----	----

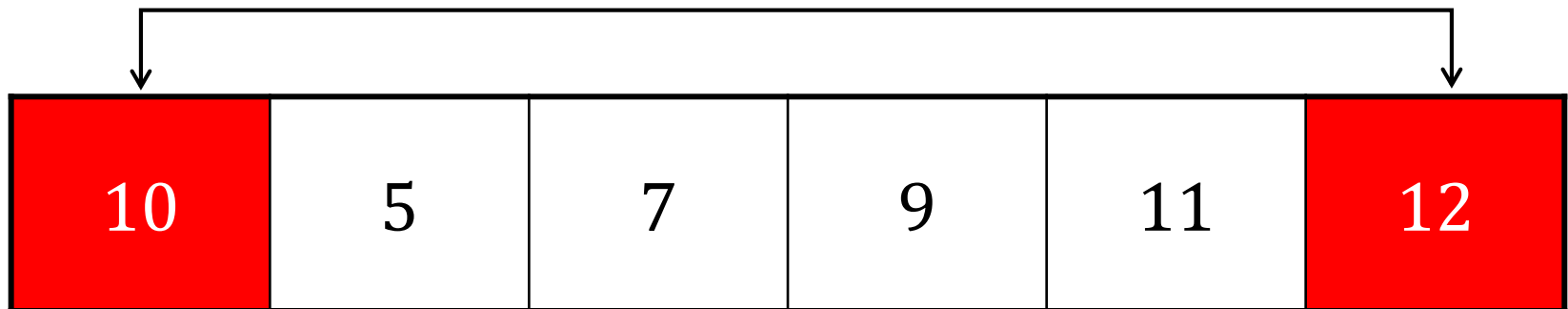
Maximum kiválasztásos rendezés

- Maximum 12
- Hozzá tartozó index 1



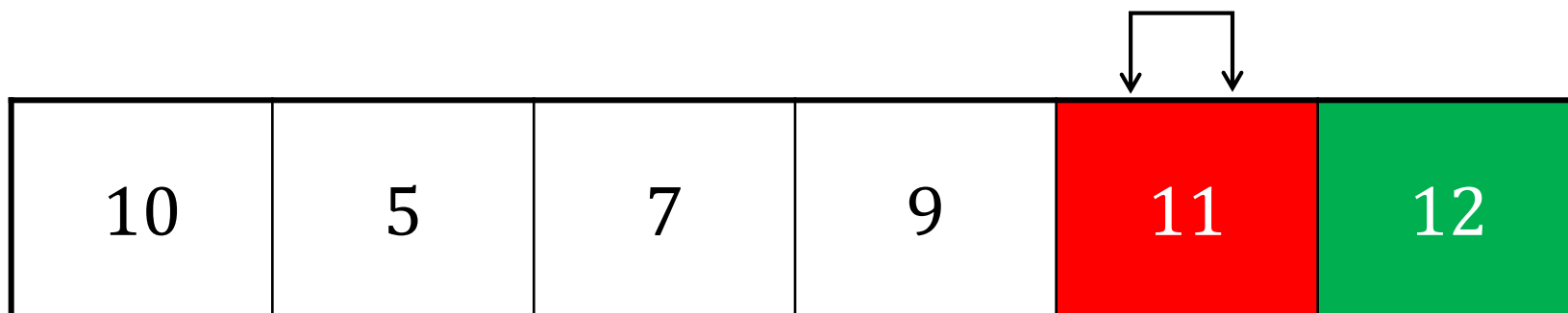
Maximum kiválasztásos rendezés

- Maximum 12
- Hozzá tartozó index 1



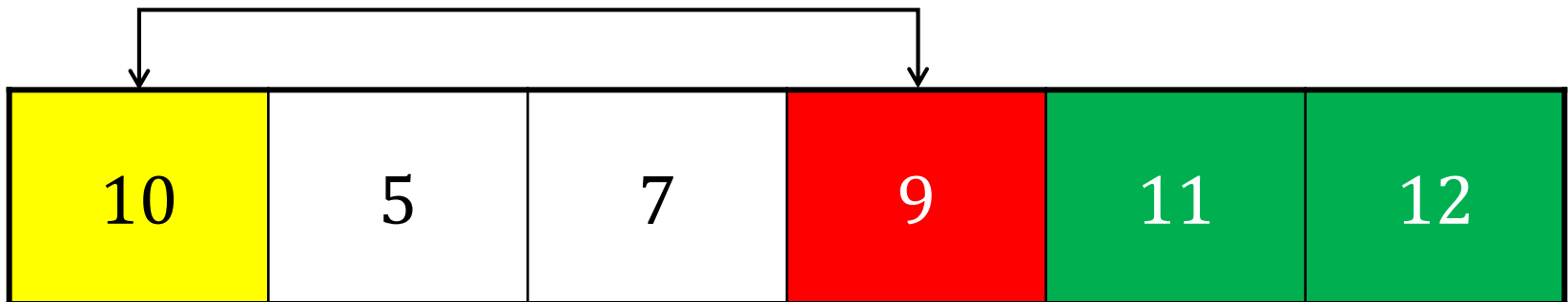
Maximum kiválasztásos rendezés

- Maximum 11
- Hozzá tartozó index 5



Maximum kiválasztásos rendezés

- Maximum 10
- Hozzá tartozó index 1



Maximum kiválasztásos rendezés

- Maximum 10
- Hozzá tartozó index 1

9	5	7	10	11	12
---	---	---	----	----	----

Maximum kiválasztásos rendezés

- A rendező algoritmus

$j \leftarrow n$
$j \geq 2$
MaximumKivalasztas($A[1 \dots j], ind$)
Csere($A[ind], A[j]$)
$j \leftarrow j - 1$

- Maximum kiválasztása

$j \leftarrow 1; ind \leftarrow 1; max \leftarrow A[1]$	
$i < j$	
$A[i + 1] > max$	
$ind \leftarrow i + 1$ $max \leftarrow A[i + 1]$	SKIP
$i \leftarrow i + 1$	

Maximum kiválasztásos rendezés

- A rendező algoritmus

$j \leftarrow n$
$j \geq 2$
MaximumKivalasztas($A[1 \dots j], ind$)
Csere($A[ind], A[j]$)
$j \leftarrow j - 1$

- Maximum kiválasztása

$j \leftarrow 1; ind \leftarrow 1; max \leftarrow A[1]$	
$i < j$	
$A[i + 1] > max$	
$ind \leftarrow i + 1$ $max \leftarrow A[i + 1]$	SKIP
$i \leftarrow i + 1$	

Maximum kiválasztásos rendezés

- A rendező algoritmus

$j \leftarrow n$
$j \geq 2$
MaximumKivalasztas($A[1 \dots j], ind$)
Csere($A[ind], A[j]$)
$j \leftarrow j - 1$

- Maximum kiválasztása

$j \leftarrow 1; ind \leftarrow 1; max \leftarrow A[1]$	
$i < j$	
$A[i + 1] > max$	
$ind \leftarrow i + 1$ $max \leftarrow A[i + 1]$	SKIP
$i \leftarrow i + 1$	

Maximum kiválasztásos rendezés

- A rendező algoritmus

- $M \ddot{O}(n) \geq (n-1) + (n-2) + \dots + 1 = \Theta(n^2)$

$j \leftarrow n$
$j \geq 2$
MaximumKivalasztas($A[1 \dots j], ind$)
Csere($A[ind], A[j]$)
$j \leftarrow j - 1$

- Maximum kiválasztása

- A maximum n elem közül legalább $n - 1$ összehasonlítással található meg

$j \leftarrow 1; ind \leftarrow 1; max \leftarrow A[1]$	
$i < j$	
$A[i + 1] > max$	
$ind \leftarrow i + 1$ $max \leftarrow A[i + 1]$	SKIP
$i \leftarrow i + 1$	

Beszűrő rendezés

- Egyszerű beillesztéssel egy – egy elemet a helyére viszünk – megkeressük, hogy hová való a már rendezett sorozatban
- Ha tömbben tároljuk, akkor a mozgatások száma is fontos

Beszűrő rendezés

- Példa



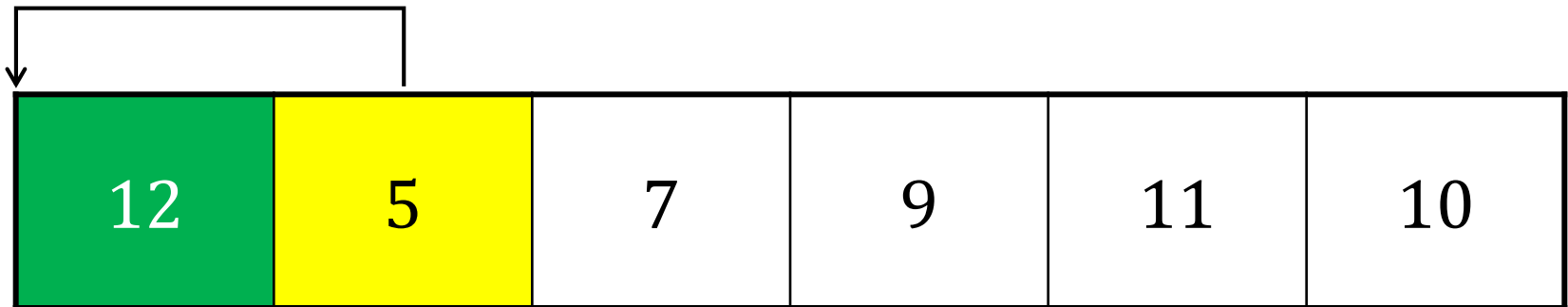
Beszúró rendezés

12	5	7	9	11	10
----	---	---	---	----	----

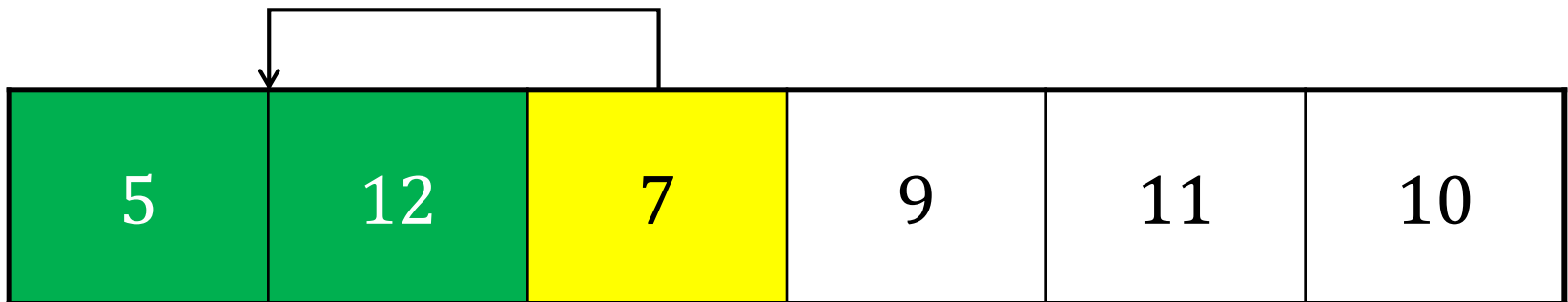
Beszúró rendezés

12	5	7	9	11	10
----	---	---	---	----	----

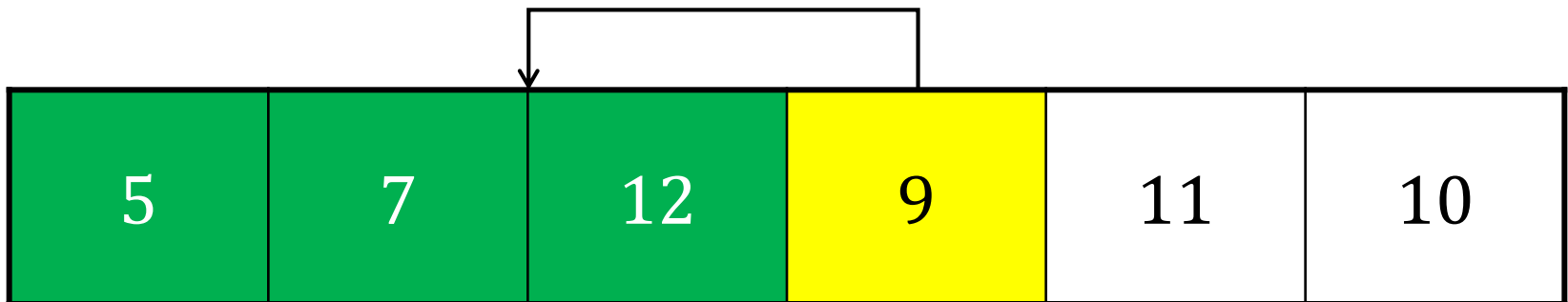
Beszűrő rendezés



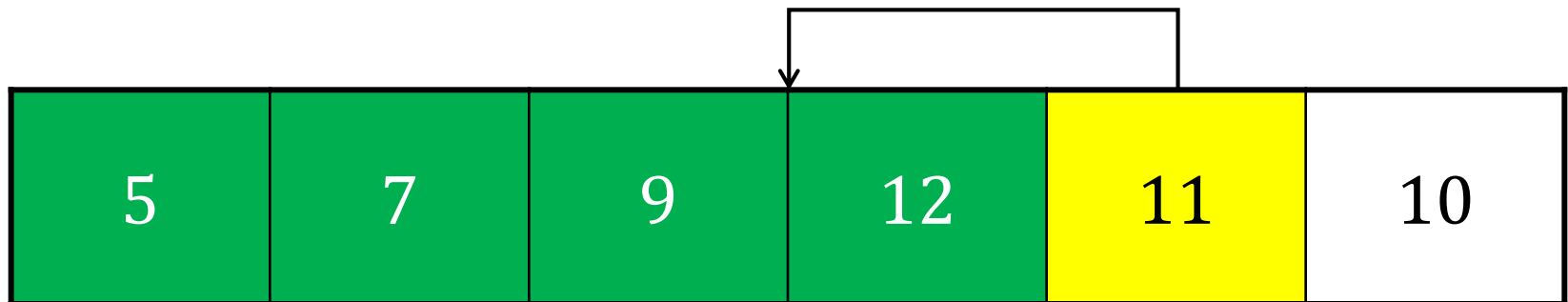
Beszűrő rendezés



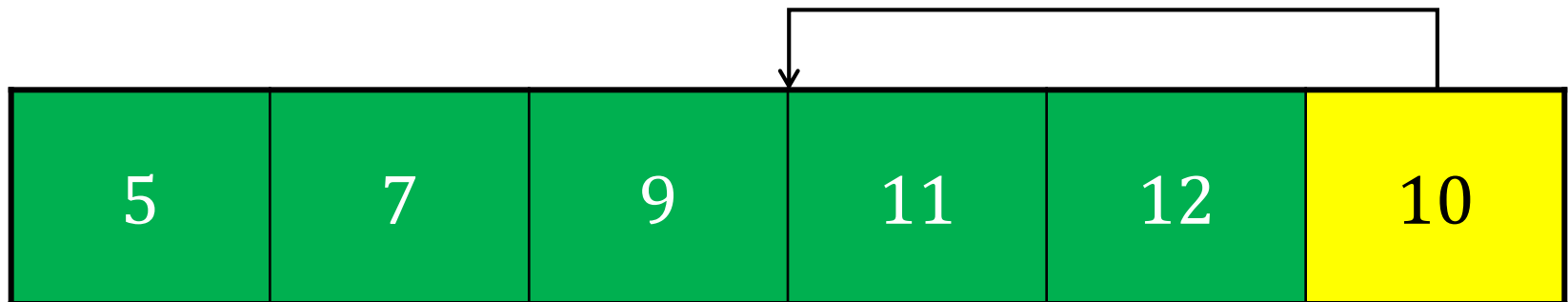
Beszűrő rendezés



Beszűrő rendezés



Beszűrő rendezés



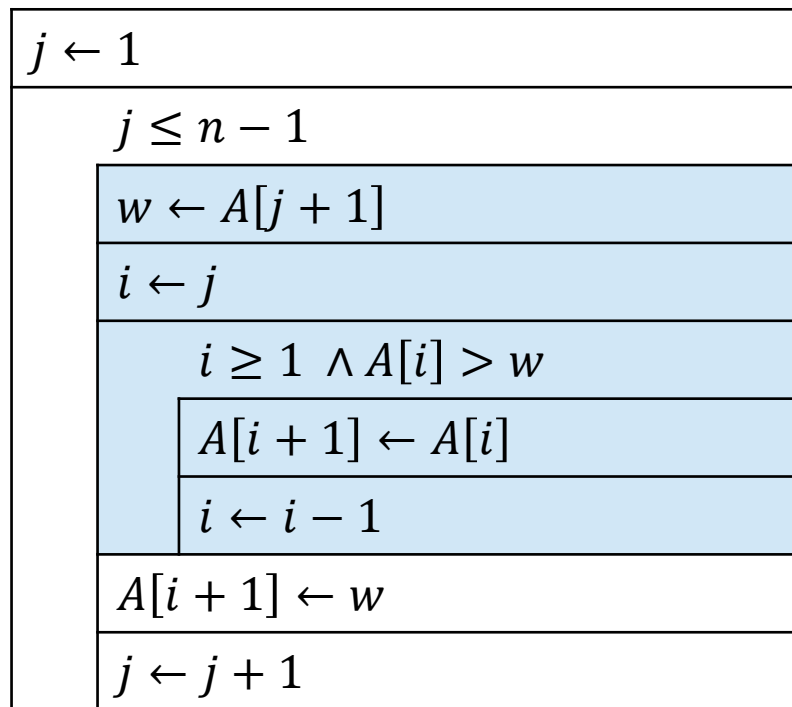
Beszűrő rendezés

5	7	9	10	11	12
---	---	---	----	----	----

Beszűrő rendezés algoritmus

$j \leftarrow 1$
$j \leq n - 1$
$w \leftarrow A[j + 1]$
$i \leftarrow j$
$i \geq 1 \wedge A[i] > w$
$A[i + 1] \leftarrow A[i]$
$i \leftarrow i - 1$
$A[i + 1] \leftarrow w$
$j \leftarrow j + 1$

Beszúró rendezés algoritmus



Beszűrő rendezés

- Műveletigény
 - Összehasonlítások
 - $M \ddot{O}(n) = n * \frac{n-1}{2} = \Theta(n^2)$
 - $A \ddot{O}(n) \approx M \frac{\ddot{O}(n)}{2} = \Theta(n^2)$
 - $m \ddot{O}(n) = n - 1$
 - M a mozgítás művelete:
 - $M M(n) = (n + 2) * \frac{n-1}{2} = \Theta(n^2)$
 - $A M(n) = \frac{n^2}{4} = \Theta(n^2)$
 - $m M(n) = 2 * (n - 1) = \Theta(n)$



Pázmány Péter Katolikus Egyetem
Információs Technológiai és Bionikai Kar

Quicksort

Gyorsabb rendező

Gyorsrendezés (Quicksort)

- Hatékony rendezési algoritmus
 - C. A. R. Hoare készítette, 1960.
- Az „Oszd meg és Uralkodj” algoritmus egy példája
- Két fázis
 1. **Partíciós fázis**
 - Oszd a munkát két részre!
 2. **Rendezési fázis**
 - Uralkodj a részeken!

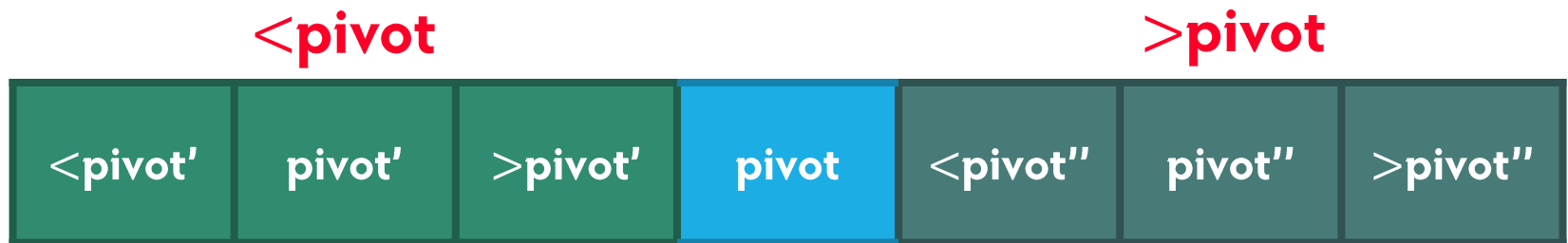
Gyorsrendezés (Quicksort)

- Partíció
 - Válassz egy „**strázsát**” (**pivot**)!
 - Válaszd a strázsa pozícióját olyanra, hogy
 - Minden elem tőle jobbra nagyobb legyen!
 - Minden elem tőle balra kisebb legyen!



Gyorsrendezés (Quicksort)

- Uralkodj
 - Alkalmazd ugyanezt az algoritmust mindkét félre!



Gyorsrendezés (Quicksort)

- Implementáció

```
quicksort( void *a, int also, int felso )  
{  
    int pivot;  
    /* Terminálási feltétel! */  
    if ( felso > also )  
    {  
        Oszd meg!   pivot = feloszt( a, also, felso );  
        Uralkodj!   quicksort( a, also, pivot-1 );  
        Uralkodj!   quicksort( a, pivot+1, felso );  
    }  
}
```

Quicksort – Megosztás

- A feladat megfelelő felosztás létrehozása
 - A példában egész értékek vannak, az egyszerűség kedvéért

23	12	15	38	42	18	36	29	27
----	----	----	----	----	----	----	----	----

Quicksort – Megosztás

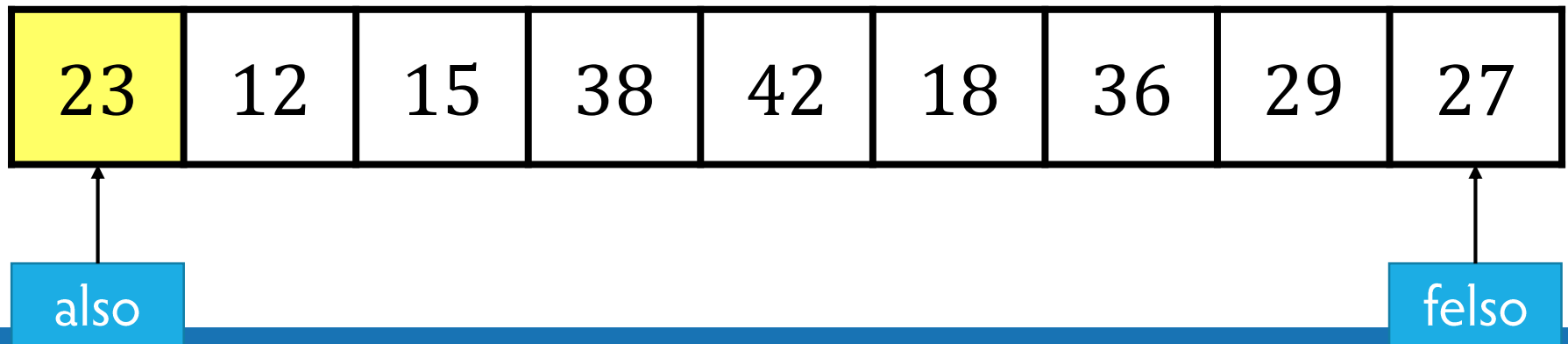
- A feladat a jó felosztás létrehozása
 - A példában egész értékek vannak, az egyszerűség kedvéért
- Bármelyik elem lehet strázsa
 - Legyen itt a bal szélső

pivot

23	12	15	38	42	18	36	29	27
----	----	----	----	----	----	----	----	----

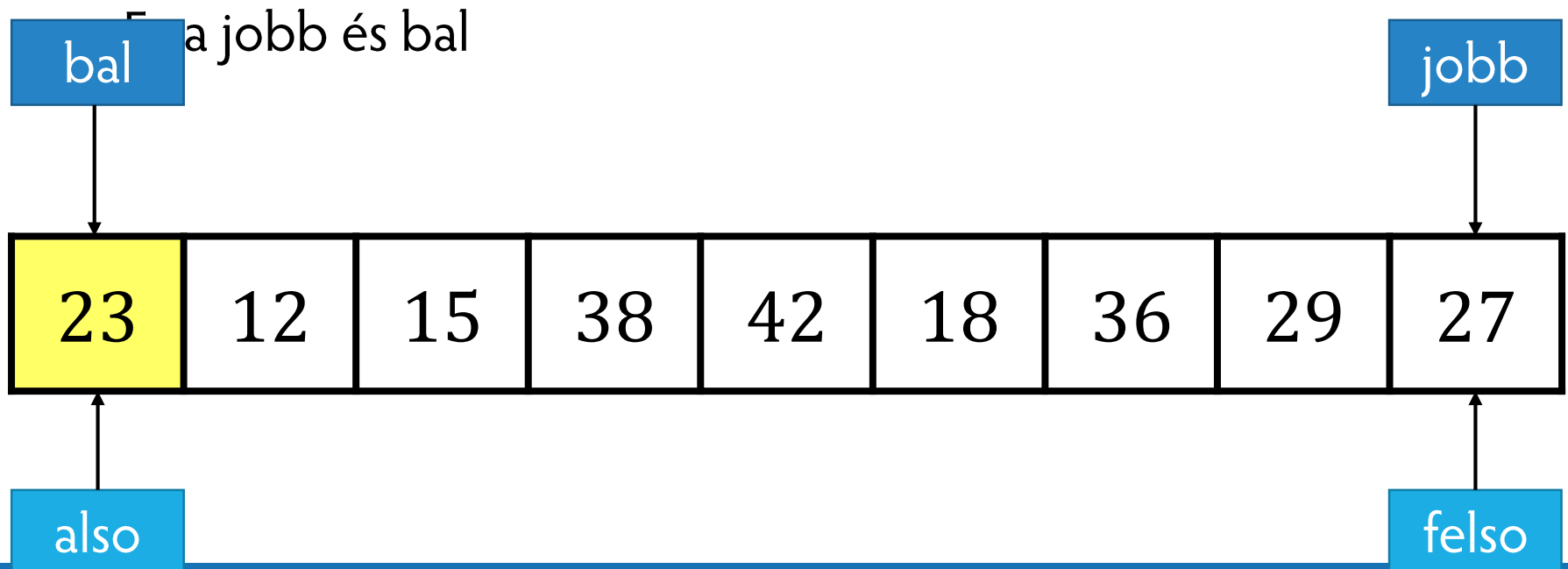
Quicksort – Megosztás

- A résztömb, amin dolgozunk az alsó és felső indexek között található
- A tömb két széléről indul szemben két indexelés
 - Ez a jobb és bal



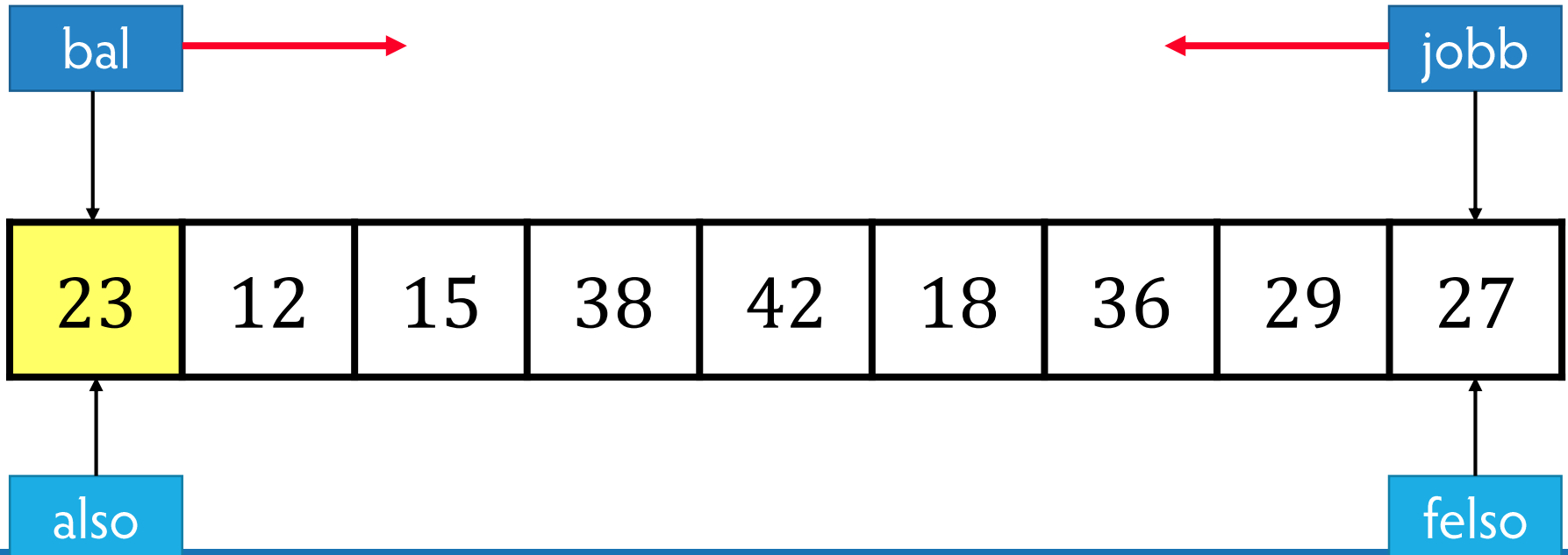
Quicksort – Megosztás

- A résztömb, amin dolgozunk az alsó és felső indexek között található
- A tömb két széléről indul szemben két indexelés



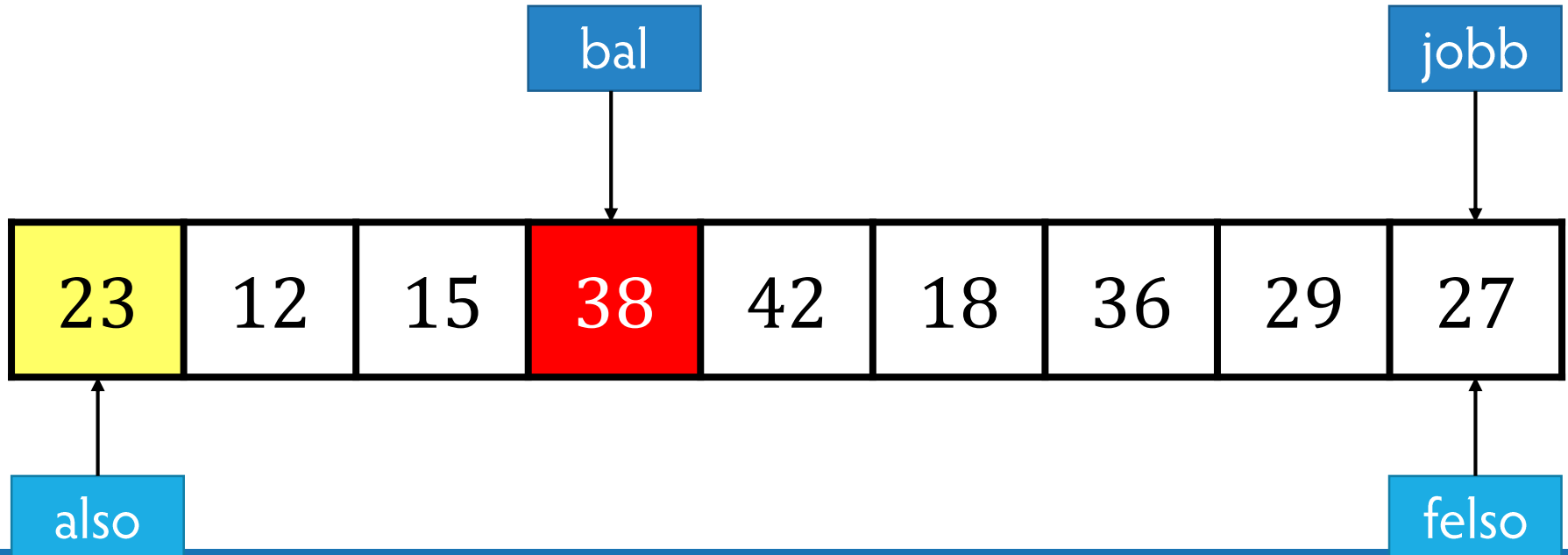
Quicksort – Megosztás

- A két indexet egymással szemben mozgatjuk
 - Addig, amíg találkoznak egymással
 - Ne feledjük, a strázsa értéke 23



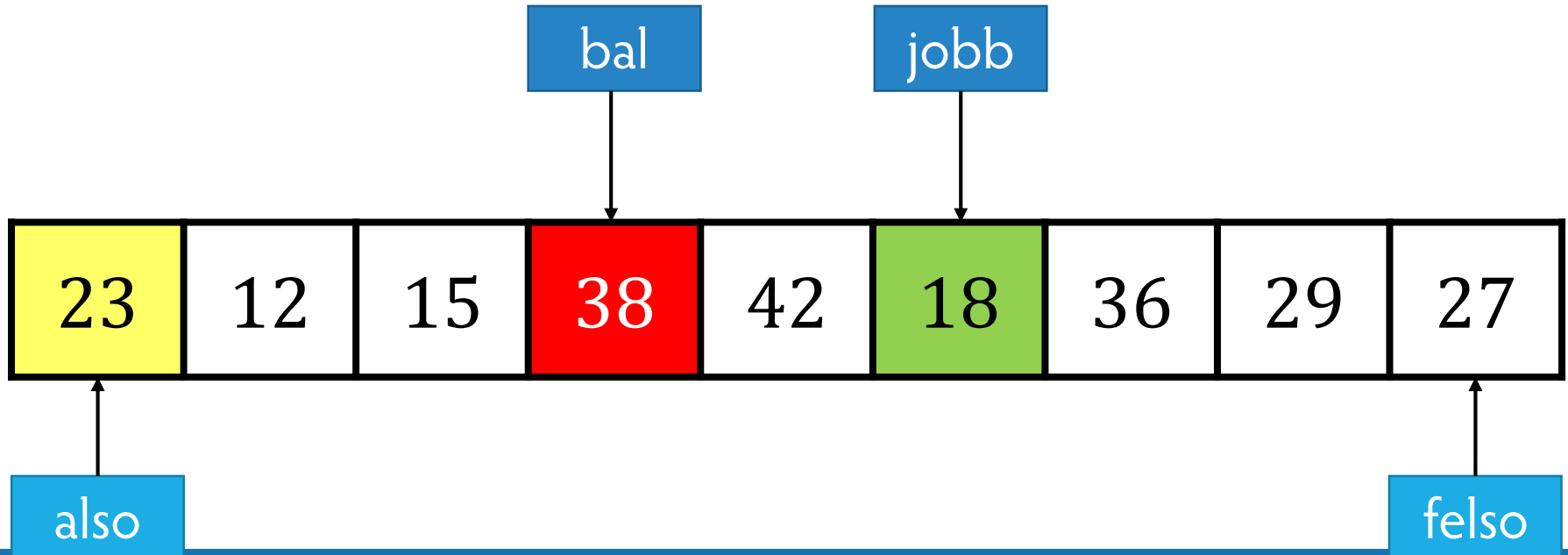
Quicksort – Megosztás

- A két indexet egymással szemben mozgatjuk
 - A bal jelzőt addig visszük jobbra, amíg nem igaz, hogy $\text{bal} \leq \text{strázs}$



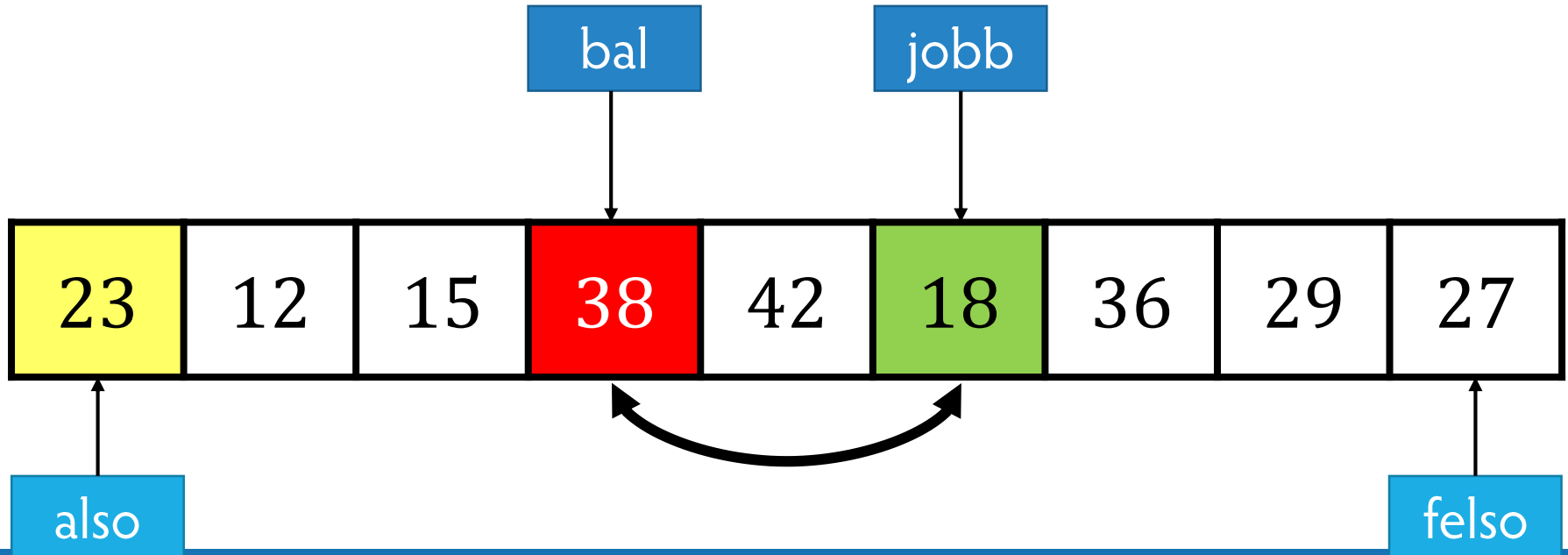
Quicksort – Megosztás

- A két indexet egymással szemben mozgatjuk
 - A jobb jelzőt addig visszük balra, amíg nem igaz, hogy $\text{jobb} \geq \text{strázs}$



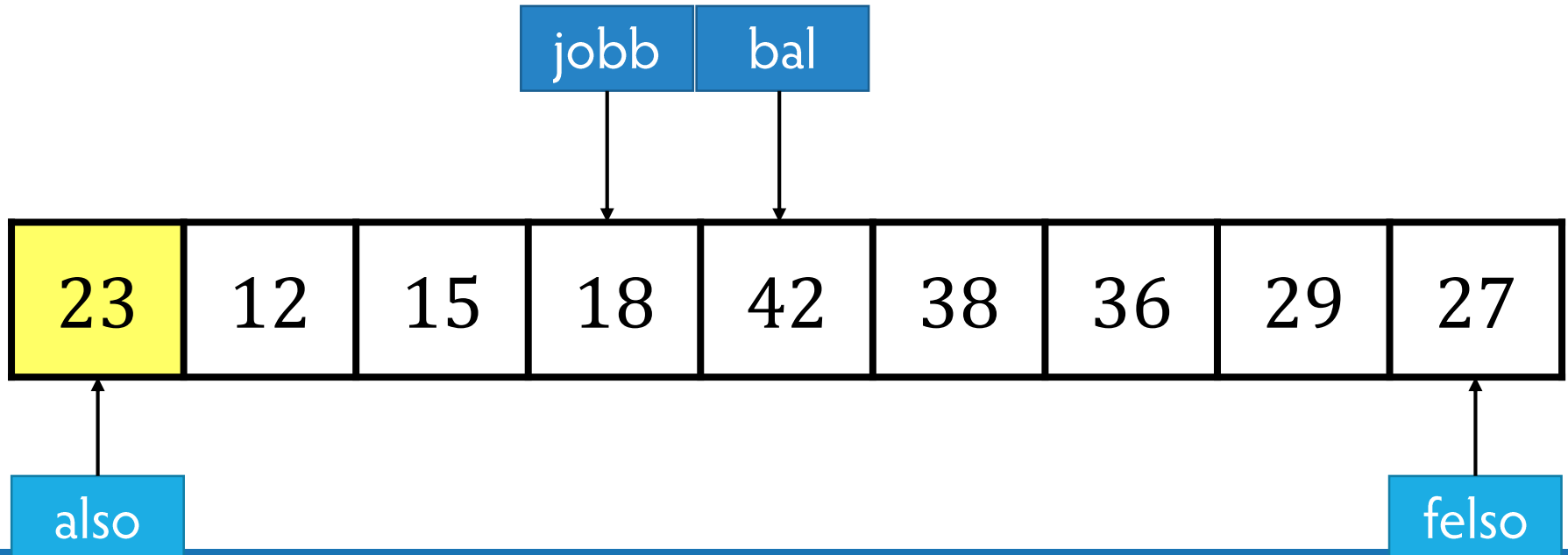
Quicksort – Megosztás

- Ha a két index nem találkozott, vagy nem kerültk ki egymást, akkor meg kell cserélni a két elemet
 - Mivel a strázsa rossz oldalain állnak



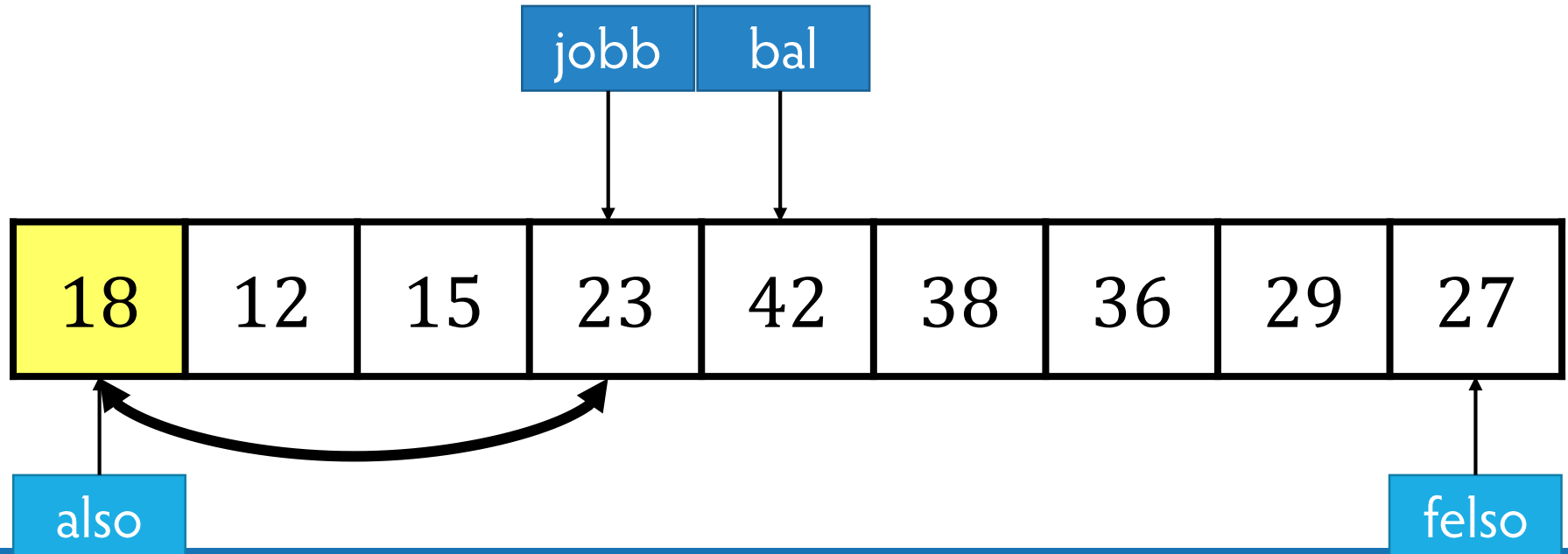
Quicksort – Megosztás

- Folytatjuk a bal és jobb léptetését
 - Szembetalálkoznak, így leáll a léptetés



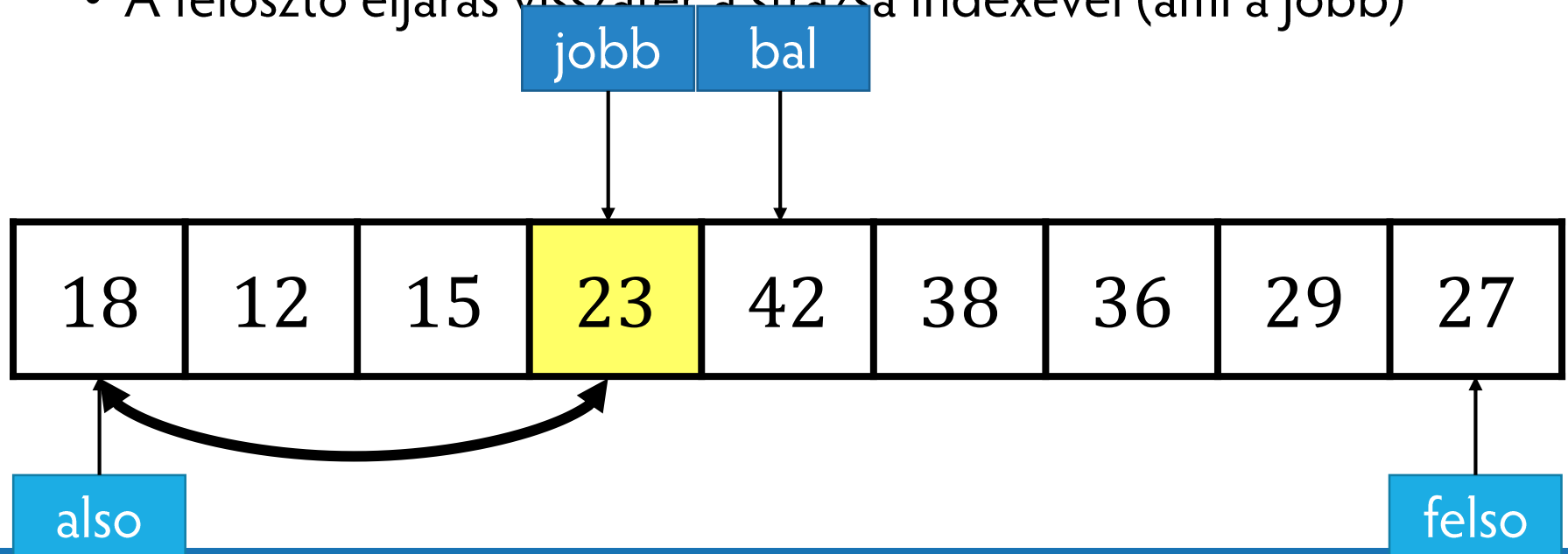
Quicksort – Megosztás

- Folytatjuk a bal és jobb léptetését
 - Szembetalálkoznak, így leáll a léptetés
 - Az utolsó lépés a strázsa és a jobb cseréje



Quicksort – Megosztás

- Folytatjuk a bal és jobb léptetését
 - Szembetalálkoznak, így leáll a léptetés
 - Az utolsó lépés a strázsa és a jobb cseréje
 - A felosztó eljárás visszatér a strázsa indexével (ami a jobb)



Quicksort – Uralkodj

- Ezután rekurzívan rendezzük a strázsa bal oldalán levő elemeket és a jobb oldalán levő elemeket
 - Meghívjuk az eredeti függvényt

Rendezzük a strázsa
bal oldalán levő
elemeket

Rendezzük a strázsa jobb oldalán
levő elemeket

18	12	15	23	42	38	36	29	27
----	----	----	----	----	----	----	----	----

Gyorsrendezés algoritmus

Gyorsrendezés (A , $also$, $felso$)

$also < felso$	
$q = \text{Feloszt}(A, also, felso)$ $\text{Gyorsrendezés}(A, also, q - 1)$ $\text{Gyorsrendezés}(A, q + 1, felso)$	SKIP

Feloszt(A , $also$, $felso$)

$str \leftarrow A[also]; bal \leftarrow also; jobb \leftarrow felso$	
$bal < jobb$	
$A[bal] \leq str \wedge bal < felso$	
$bal \leftarrow bal + 1$	
$A[jobb] \geq str \wedge jobb > also$	
$jobb \leftarrow jobb - 1$	
$bal < jobb$	
$\text{Csere}(A[bal], A[jobb]);$	SKIP
$A[also] \leftarrow A[jobb]; A[jobb] \leftarrow str;$ return $jobb;$	

Quicksort - Analízis

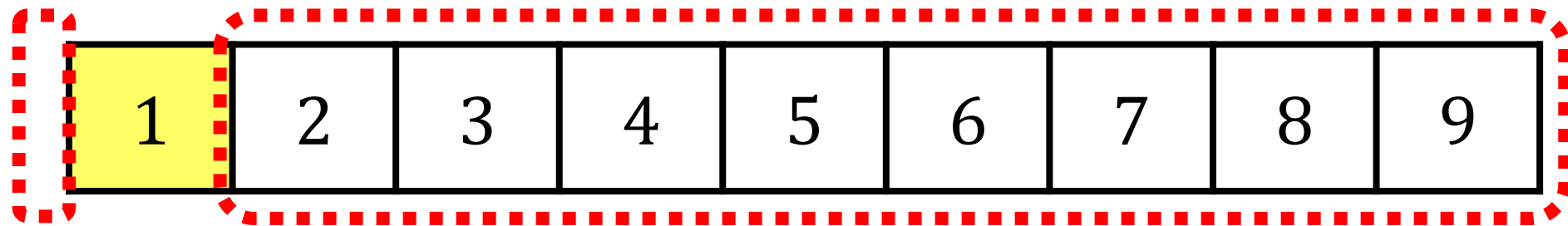
- Felosztás
 - vizsgálj meg minden elemet egyszer $\mathcal{O}(n)$
- Uralkodás
 - az adatok kétfelé osztása $\mathcal{O}(\log_2 n)$
- összesen
 - szorzat $\mathcal{O}(n \log n)$
- De van egy gond ...

Quicksort – az igazság!

- Mi történik, ha az adatok már rendezettek, vagy majdnem rendezettek?
 - Azt várnánk, hogy akkor gyorsabb lesz!

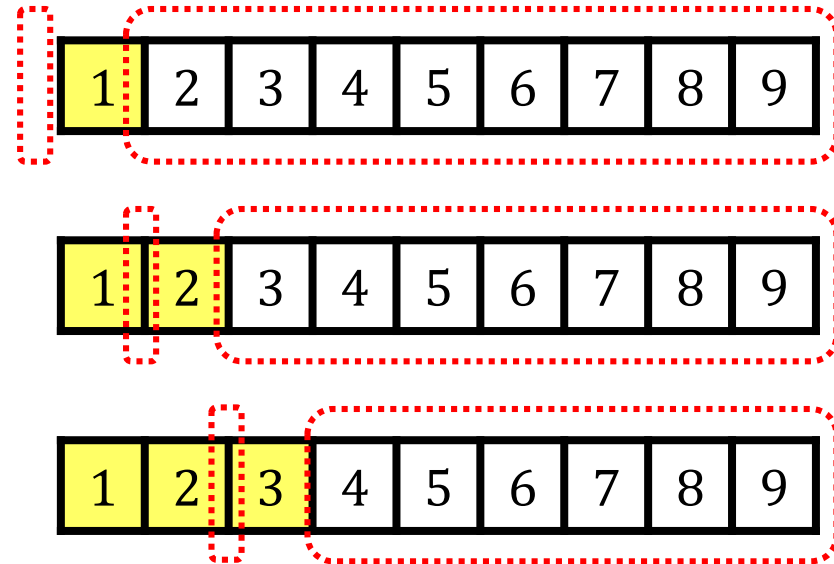
Quicksort – az igazság!

- Rendezett adatok esetén
 - A bal elem a strázsa
 - A felosztás végén a tömb bal szélén marad a strázsa
 - A tőle balra levő elemeket rendezzük – nincsenek ilyen elemek
 - A tőle jobbra levő elemeket rendezzük – a maradék tömb



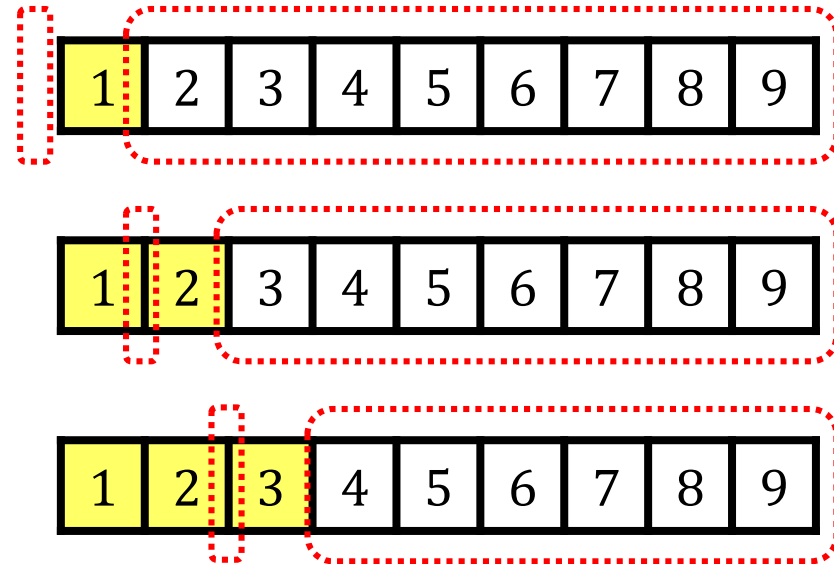
Quicksort – az igazság!

- Rendezett adatok esetén
- Minden partícionálás során létrejön egy
 - 0 méretű feladat
 - $n - 1$ méretű feladat
- A partíciók száma?



Quicksort – az igazság!

- Rendezett adatok esetén
- Minden partícionálás során létrejön egy
 - 0 méretű feladat
 - $n - 1$ méretű feladat
- A partíciók száma?
 - Minden n időigénye $\mathcal{O}(n)$
 - Összesen $n * \mathcal{O}(n)$
 - Ez pedig összesen $\mathcal{O}(n^2)$
 - Tehát a gyorsrendező olyan rossz, mint a buborék rendezés!?

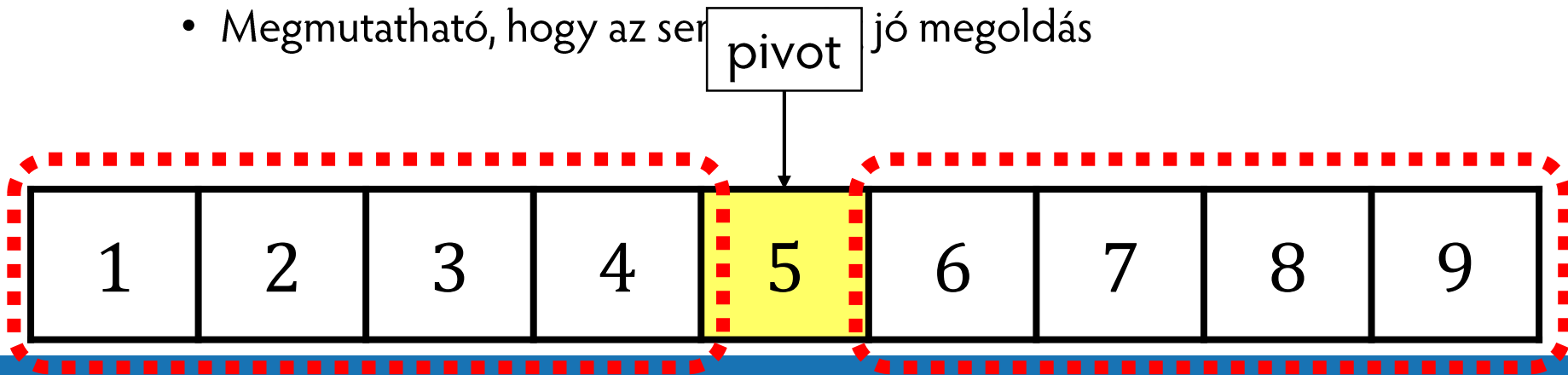


Quicksort – az igazság!

- A Quicksort $\mathcal{O}(n * \log n)$ viselkedése a majdnem egyforma partíciókon múlik
- Átlagosan is majdnem ez a helyzet
 - És a Quicksort általában $\mathcal{O}(n * \log n)$
- Mit lehet tenni?
- Általában semmit
 - De javíthatjuk az esélyeinket!

Quicksort – A pivot választása

- Bármelyik strázsa megteszi...
- Válasszunk egy másikat ...
 - Ha a partíciók egyformák akkor $\mathcal{O}(n * \log n)$ idő lesz a rendezés ideje
 - Legyen a középső
 - Rendezett elemek esetén jó választás
 - Megmutatható, hogy az sor pivot jó megoldás



Quicksort – 3-ból a középső pivot

- Válasszunk három strázsát, majd azok közül az érték szerinti középsőt használjuk
 - Például vegyük az indexek közül a két szélsőt és a középsőt

1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---
 - Az értékek közül a középső az 5
 - Ez a rendezett adatok esetén jó választás!
 - $\mathcal{O}(n * \log n)$ idő
- Mivel a rendezett (vagy majdnem rendezett) adatok elég gyakoriak, ez egy jó stratégia
 - Különösen, ha azt várjuk, hogy az adataink rendezettek lesznek!

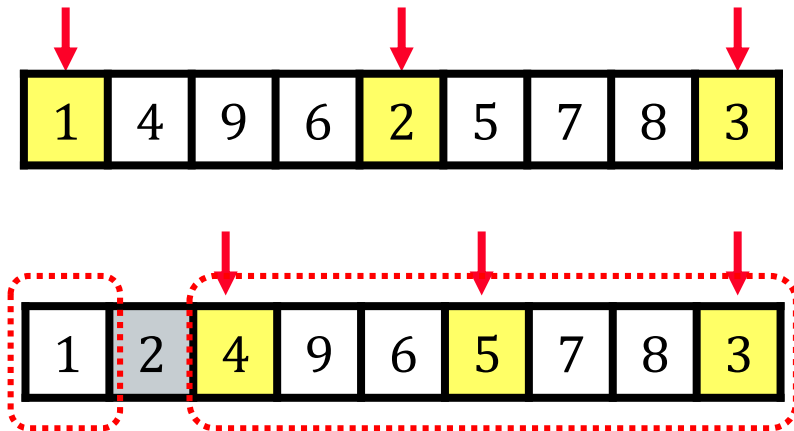
Quicksort – Véletlen pivot

- Válassz egy strázsát véletlenszerűen
 - Minden felosztásnál másik pozíciót
 - Átlagosan a rendezett adatokat jól osztja szét
 - $\mathcal{O}(n * \log n)$ idő
- Fontos követelmény
 - A pivot kiválasztása $\mathcal{O}(1)$ idő kell legyen

Quicksort – Garantált $\mathcal{O}(n * \log n)$?

- **Sohasem garantálható**

- A tetszőleges pivot választási stratégia vezethet $\mathcal{O}(n^2)$ időhöz
- Itt a 3-ból a középső kiválasztja a 2-t
 - egy partícióba 1 elem kerül,
 - a másikba 7
- következőnek kiválasztja a 4-t
 - egy partícióba 1 elem kerül,
 - a másikba 5



Rendezés

- Buborék, Beszúrásos, Maximumkiválasztásos
 - $\mathcal{O}(n^2)$ rendezések
 - Egyszerű kód
 - Kis n -re gyors lehet (rendszer függő)
- Quick Sort
 - Oszd meg és uralkodj
 - $\mathcal{O}(n * \log n)$

Quicksort

- $\mathcal{O}(n * \log n)$ de
- Lehet $\mathcal{O}(n^2)$ is
- A pivot kiválasztásától függ
 - 3-ból a középső
 - Véletlen pivot
 - Jobb eredmény, de **nem garantált**
 - **Népszerű algoritmus!**
- Felosztó algoritmus lehet többféle
 - Következő alkalommal részletesen



Nem helyben rendezés

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

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

4

	2	3	1	8	6	7	5
--	---	---	---	---	---	---	---

4

--	--	--	--	--	--	--	--

2	3	1
---	---	---

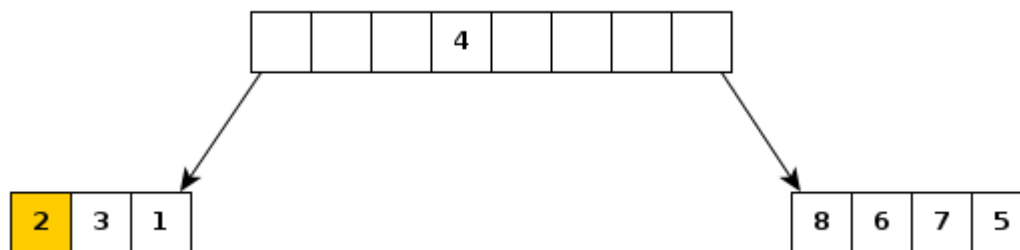
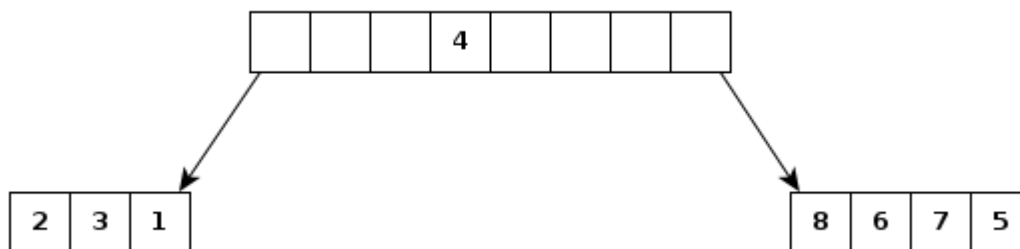
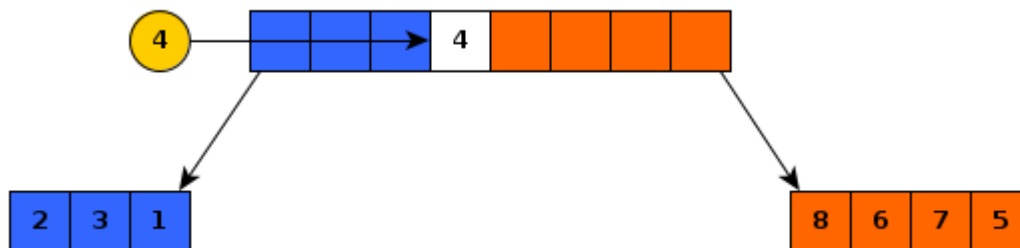
8	6	7	5
---	---	---	---

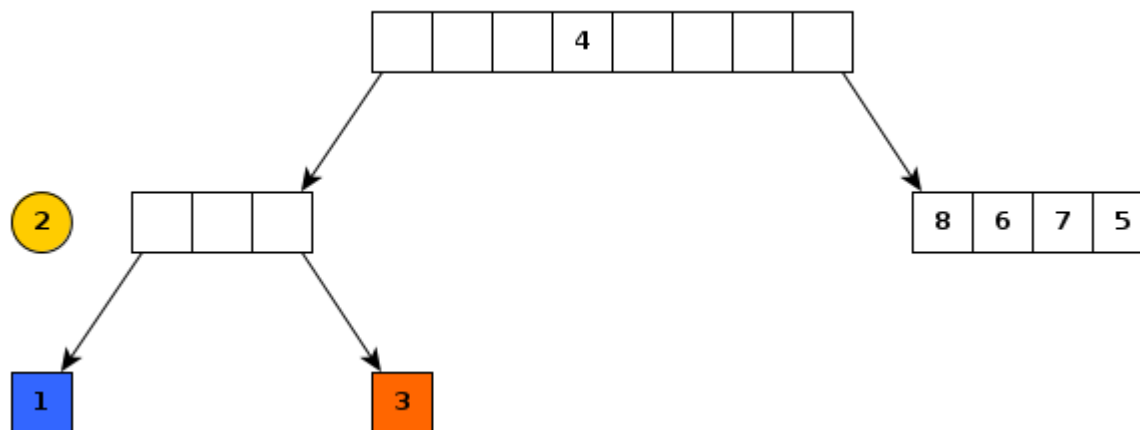
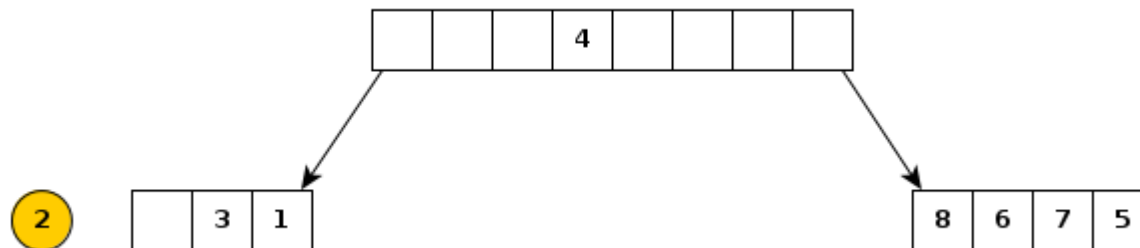
4

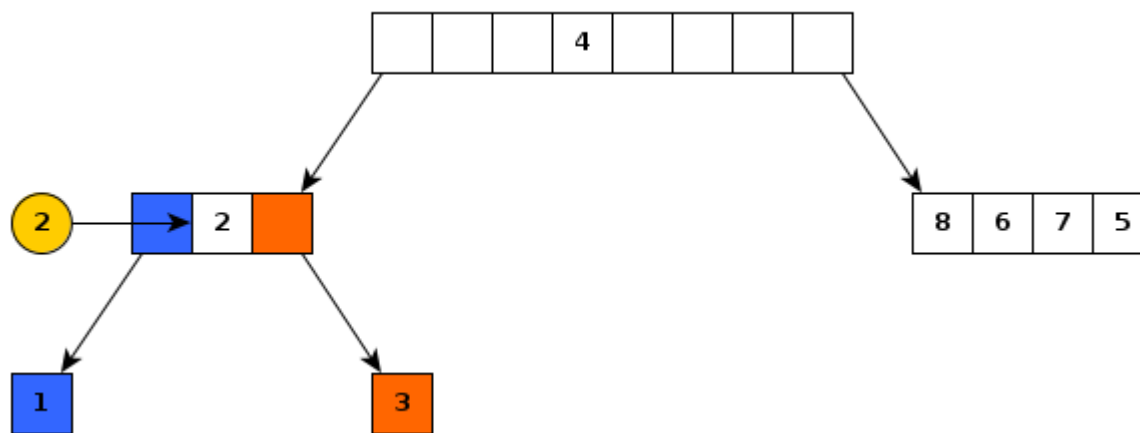
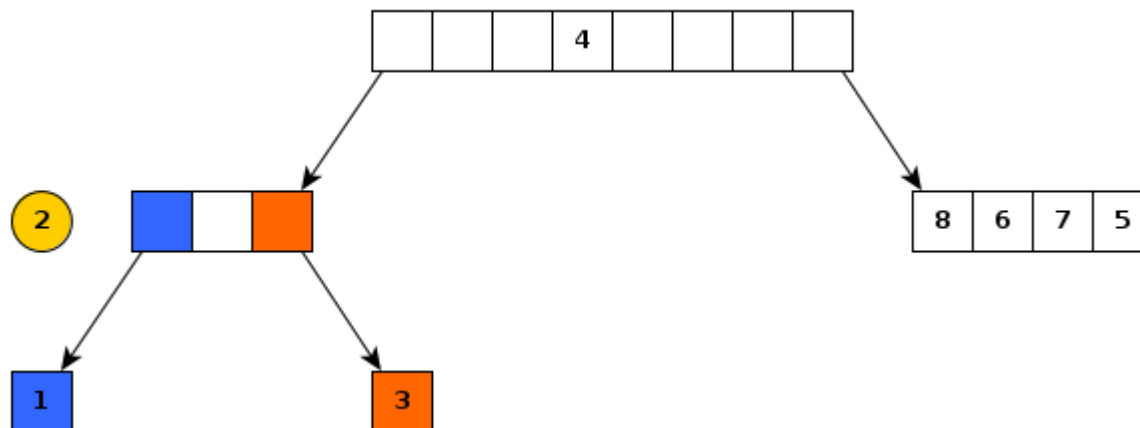
--	--	--	--	--	--	--	--

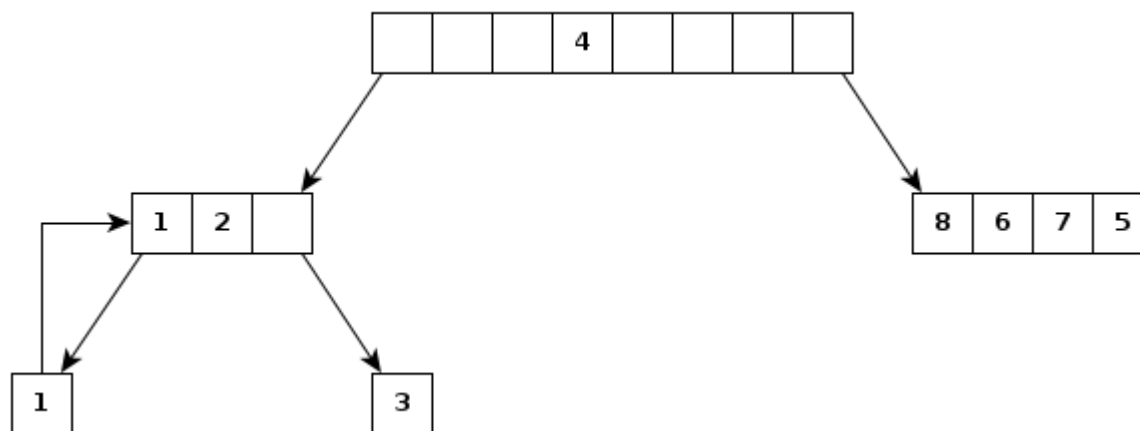
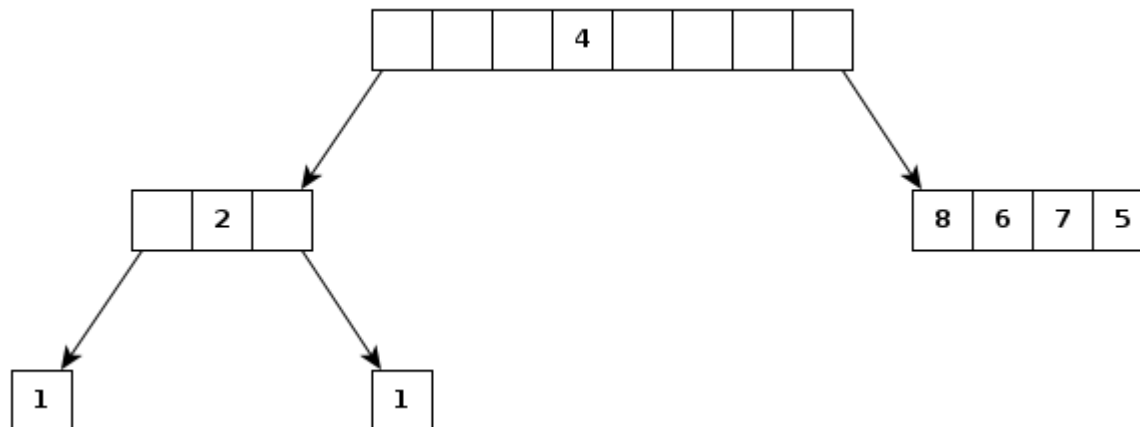
2	3	1
---	---	---

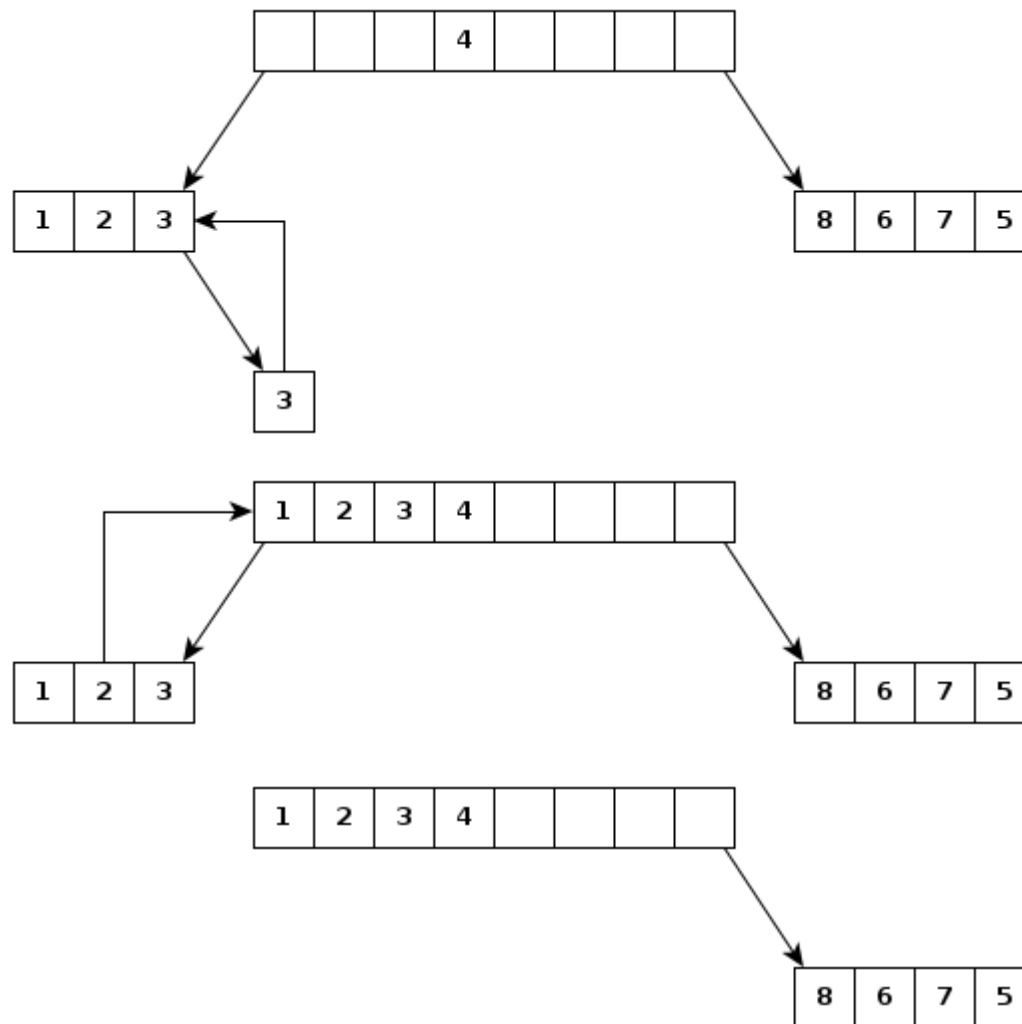
8	6	7	5
---	---	---	---

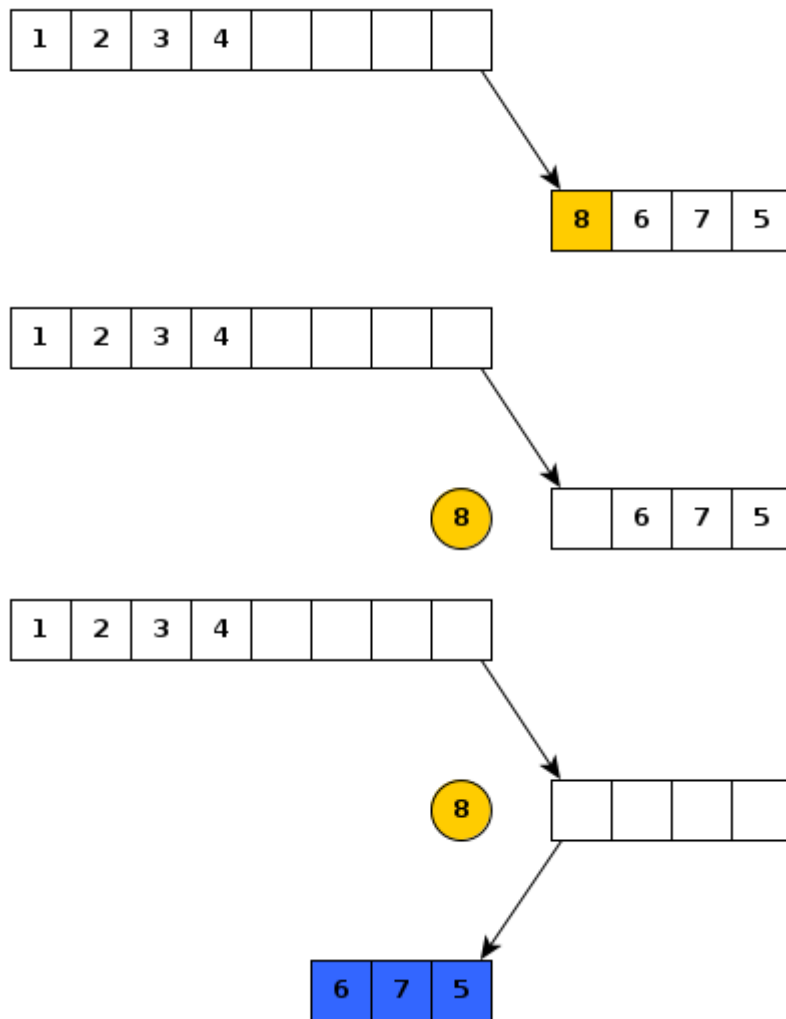


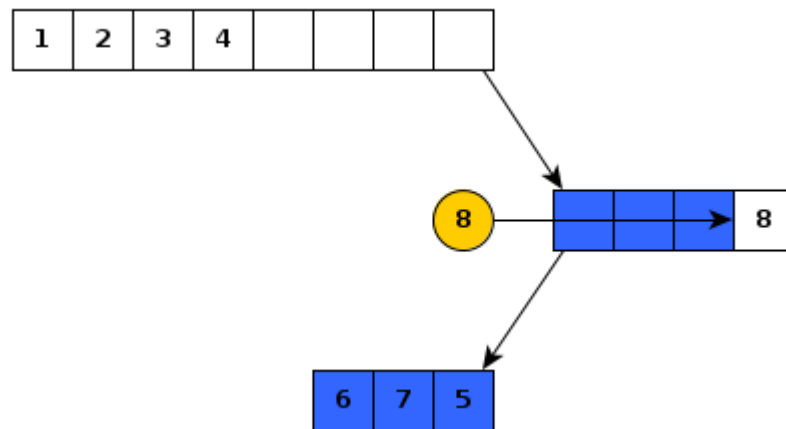
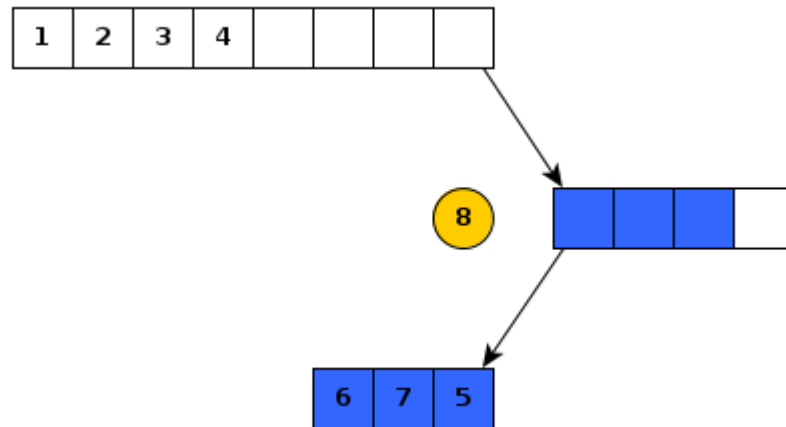


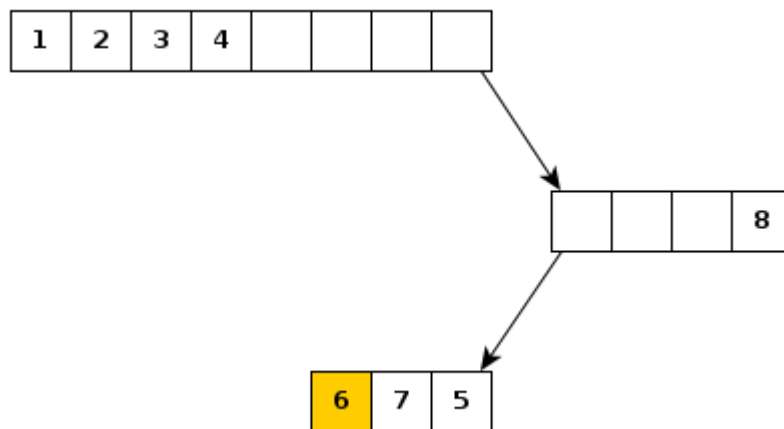
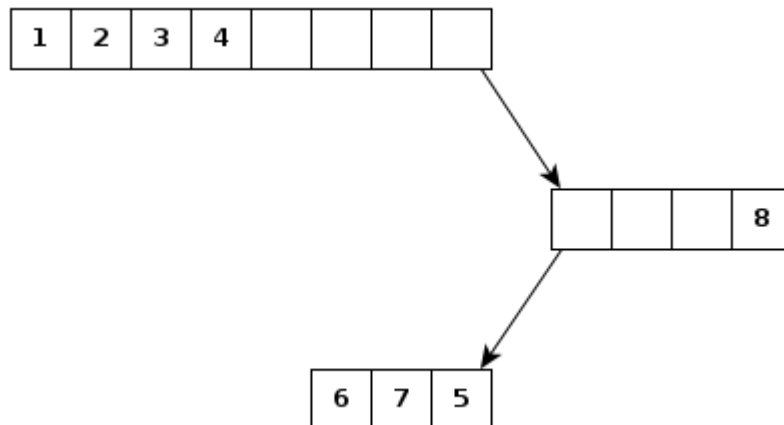


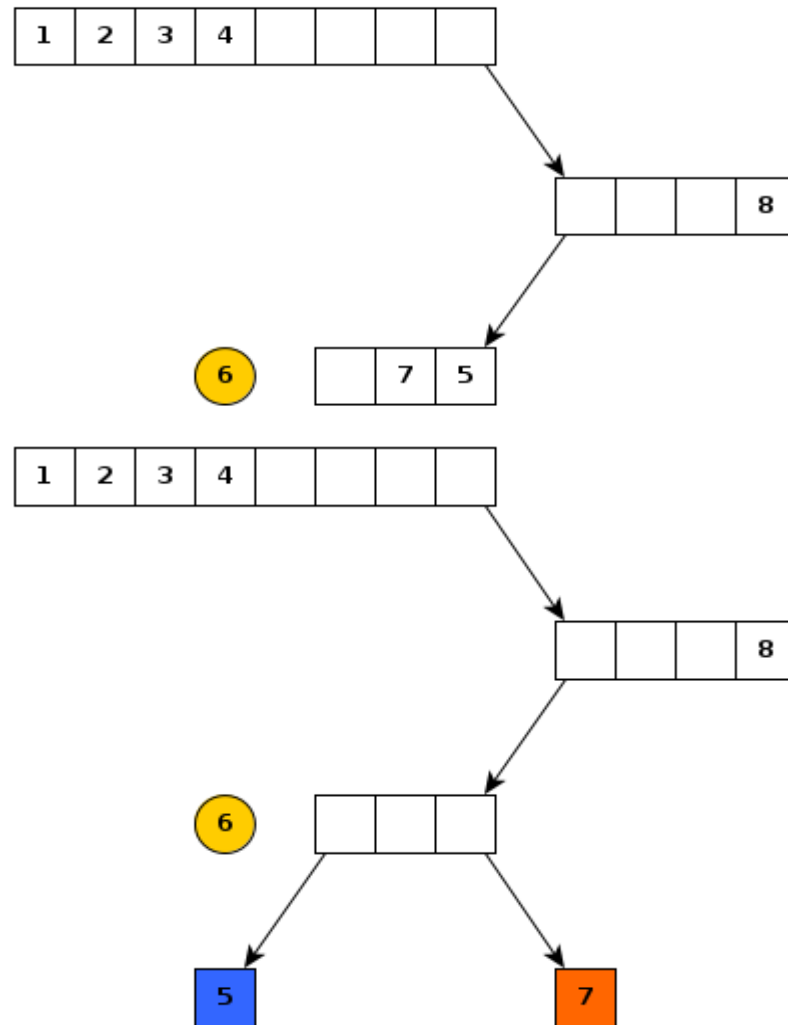


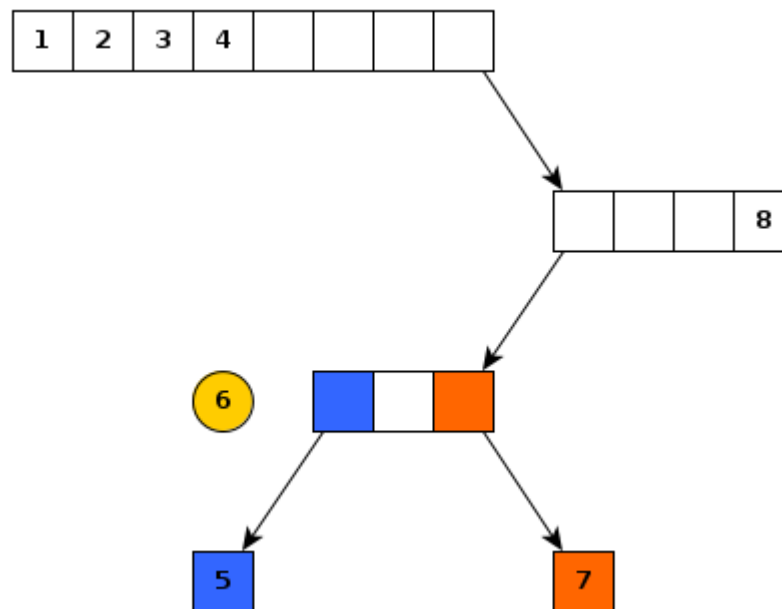


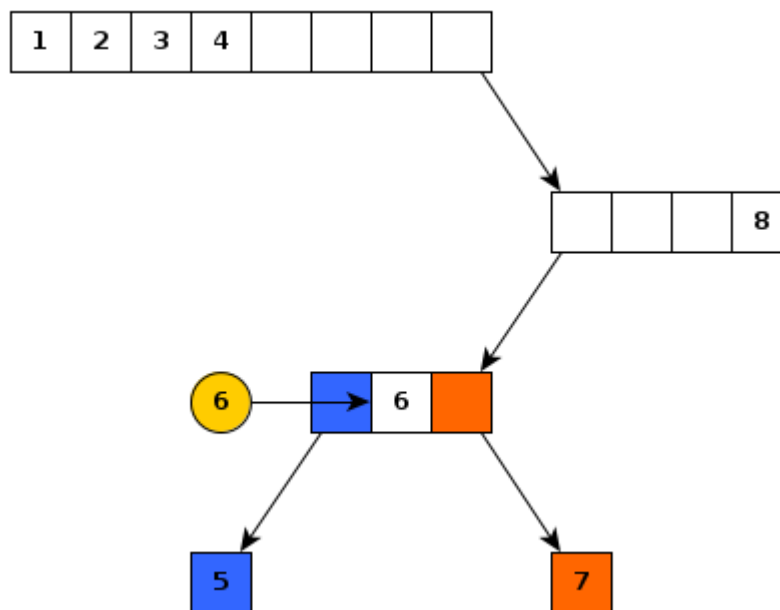


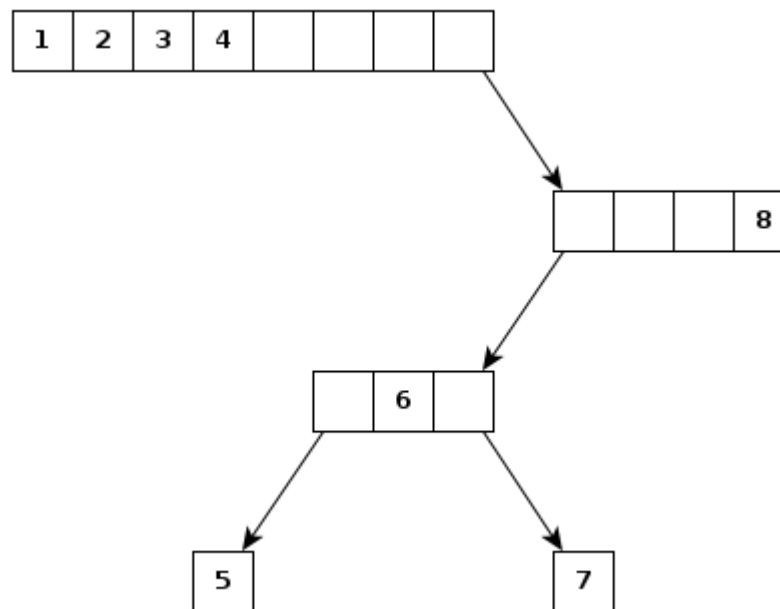


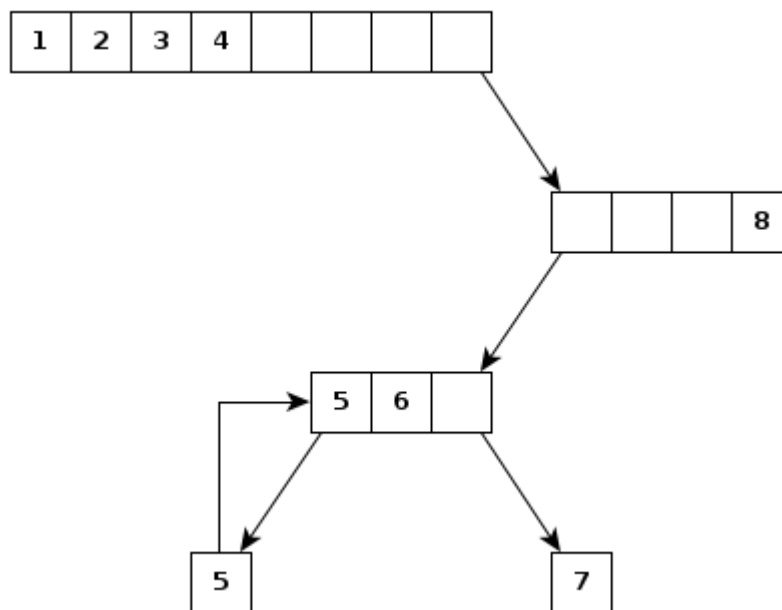


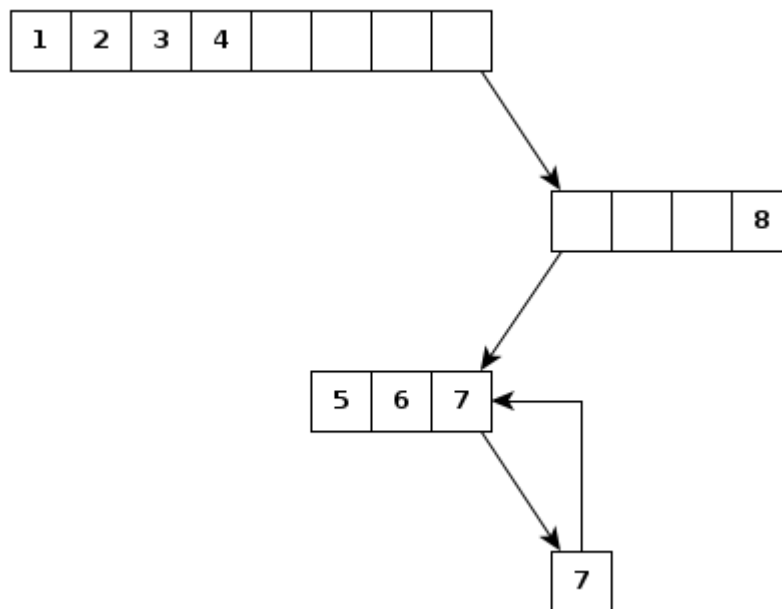


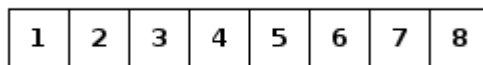
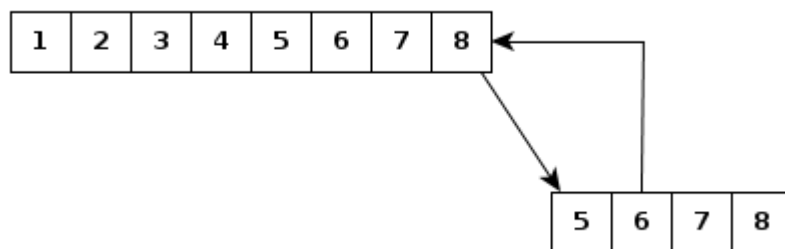
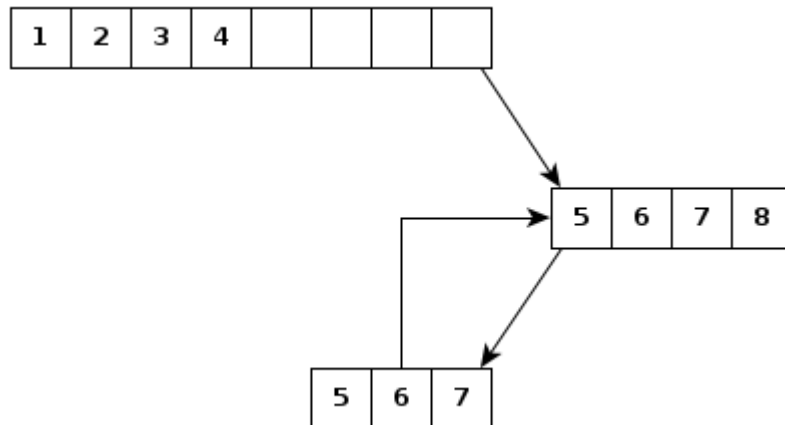














Rendező algoritmusok II.

Következő alkalommal