ADATSZERKEZETEK ÉS ALGORITMUSOK

Gyorsrendező

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

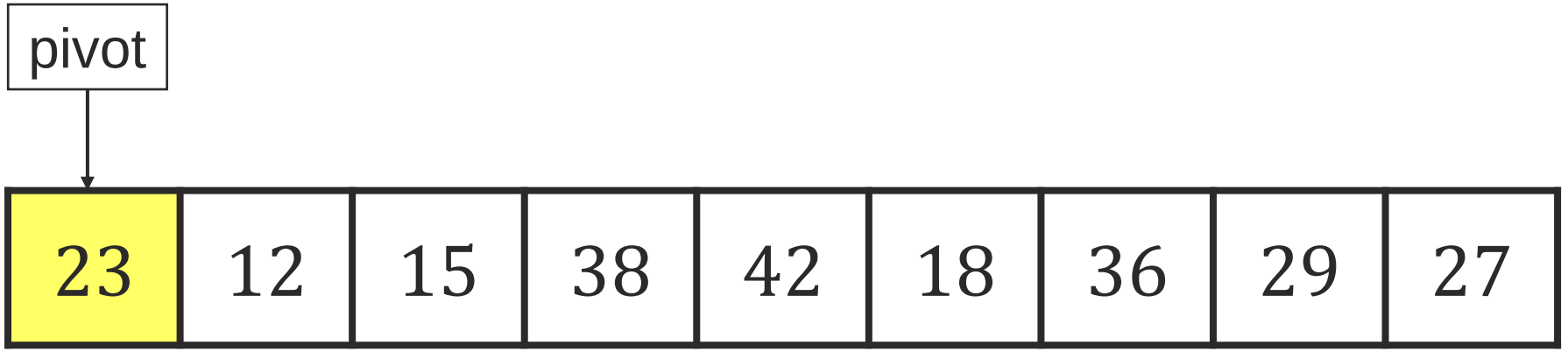
```
quicksort( void *a, int also, int felso )
{
  int pivot;
  /* Terminálási feltétel! */
  if ( felso > also )
  {
    pivot = feloszt( a, also, felso );   // Oszd meg!
    quicksort( a, also, pivot-1 );       // Uralkodj!
    quicksort( a, pivot+1, felso );      // Uralkodj!
  }
}
```

Quicksort – Megosztás

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

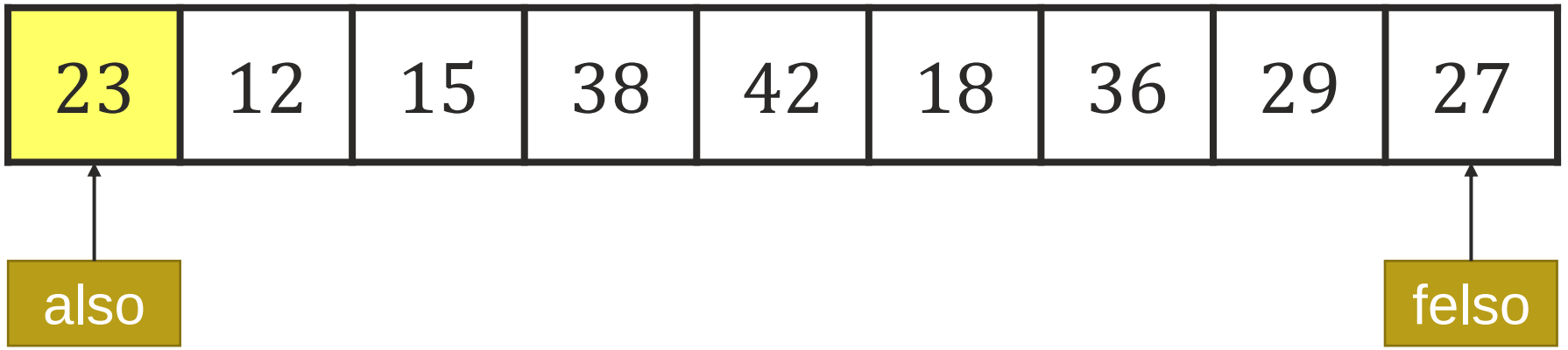
Quicksort – Megosztás

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



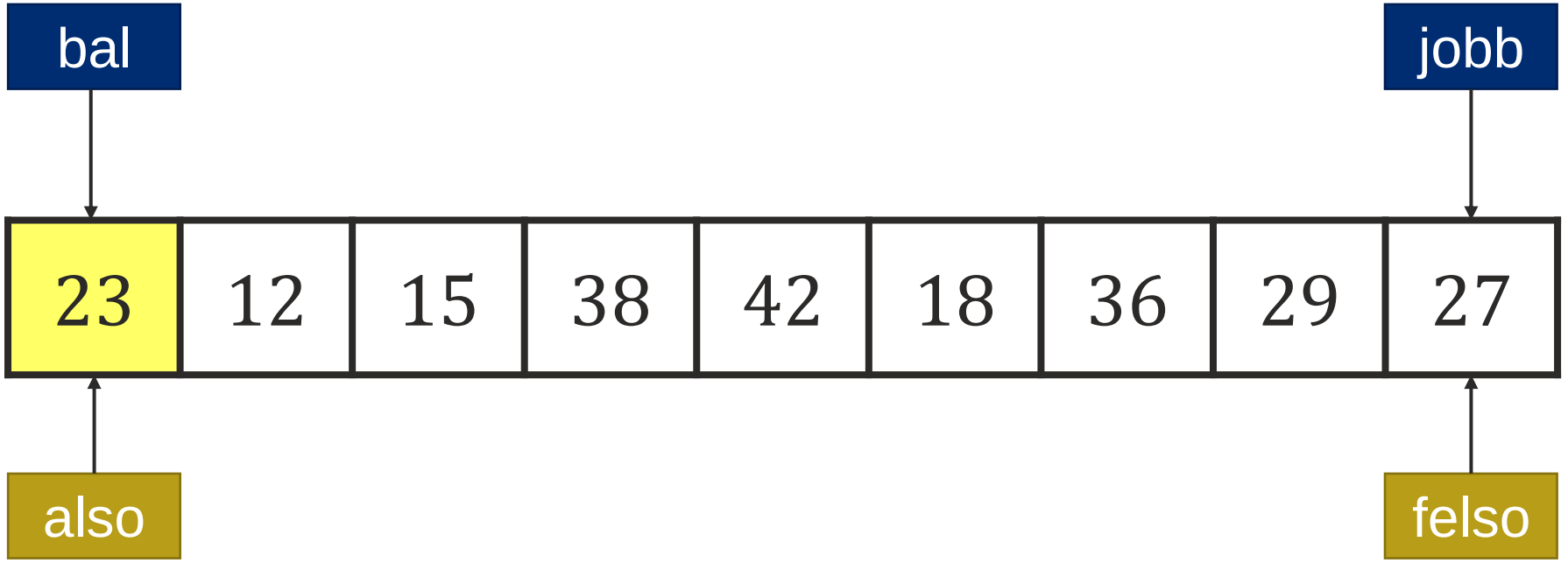
Quicksort – Megosztás

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



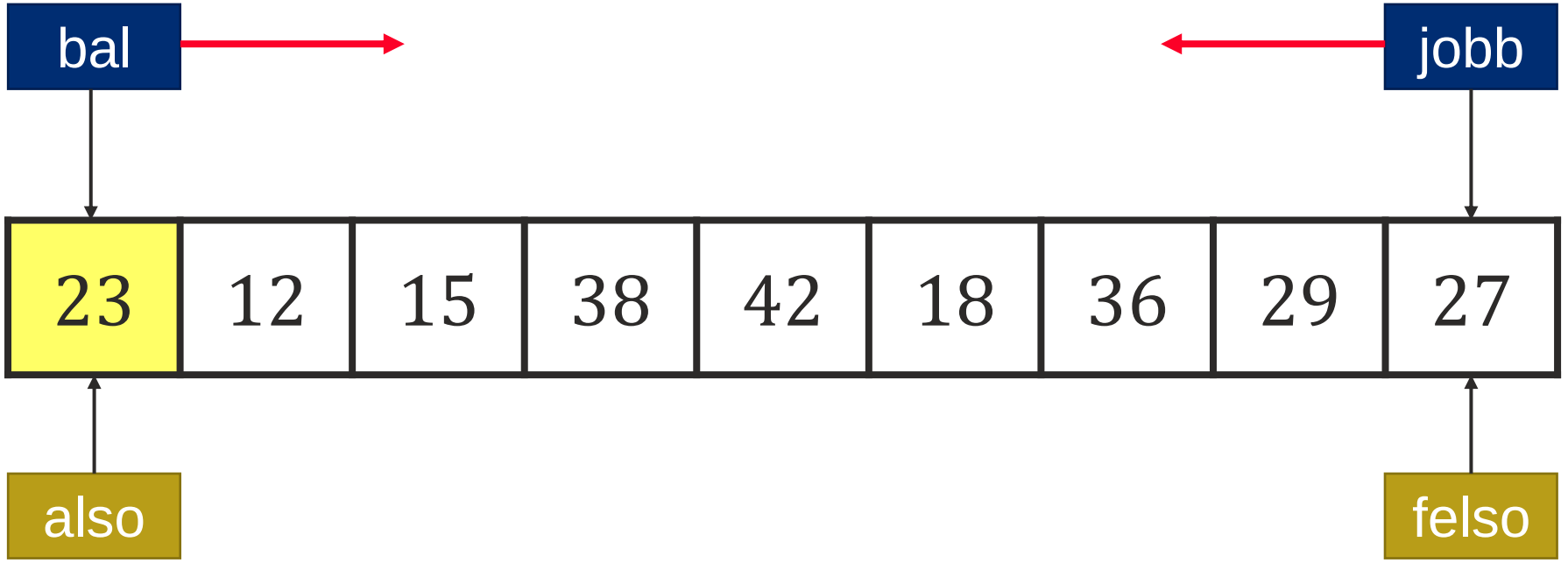
Quicksort – Megosztás

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



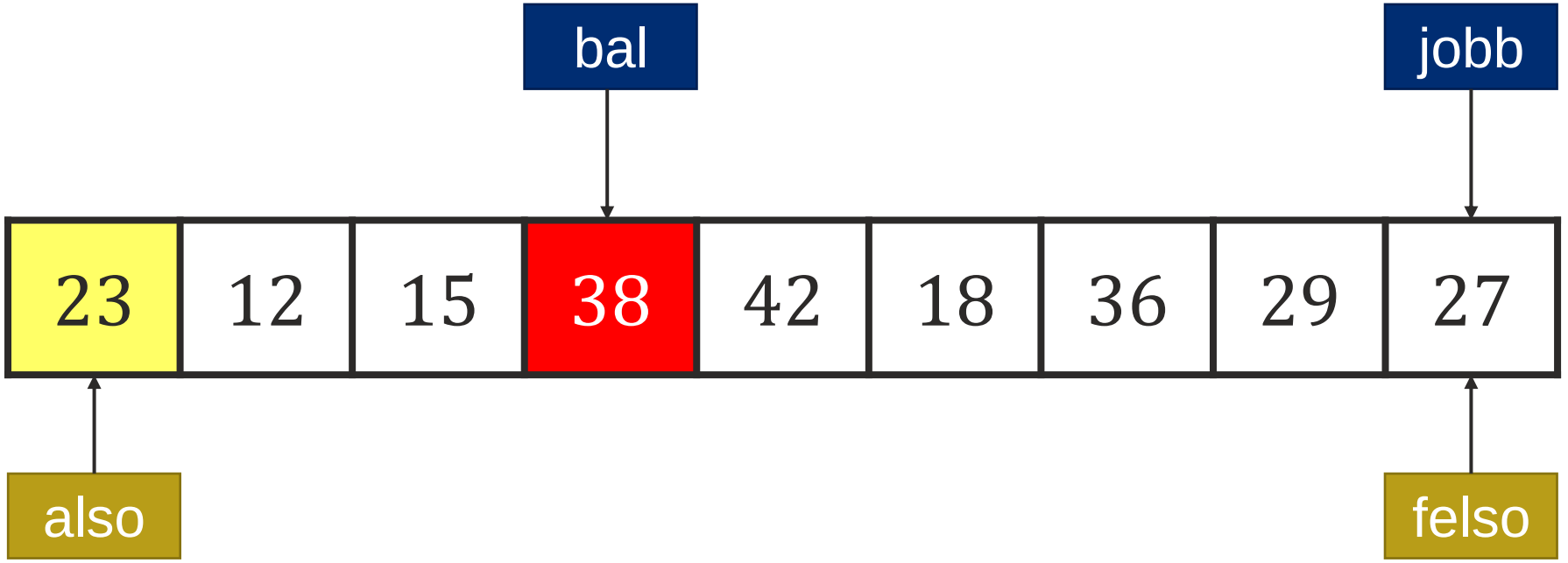
Quicksort – Megosztás

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



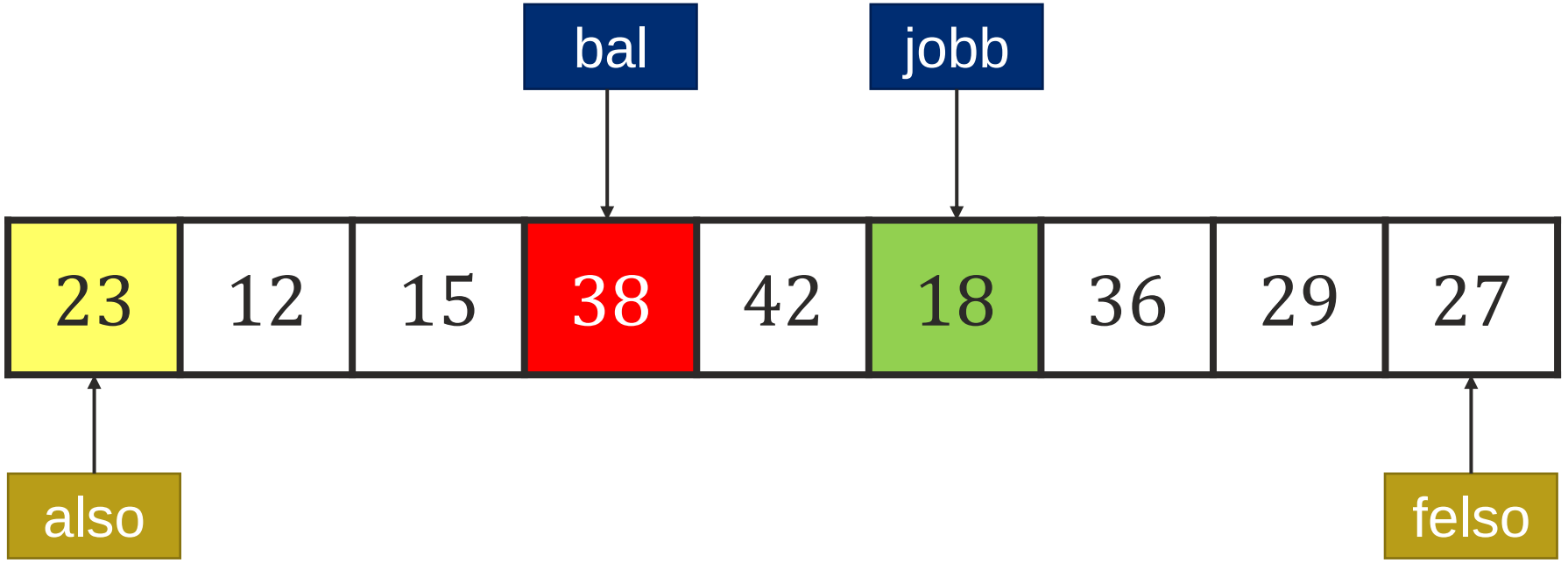
Quicksort – Megosztás

- A két indexet egymással szemben mozgatjuk
  - A bal jelzőt addig visszük jobbra, amíg nem igaz, hogy bal ≤ strázsa



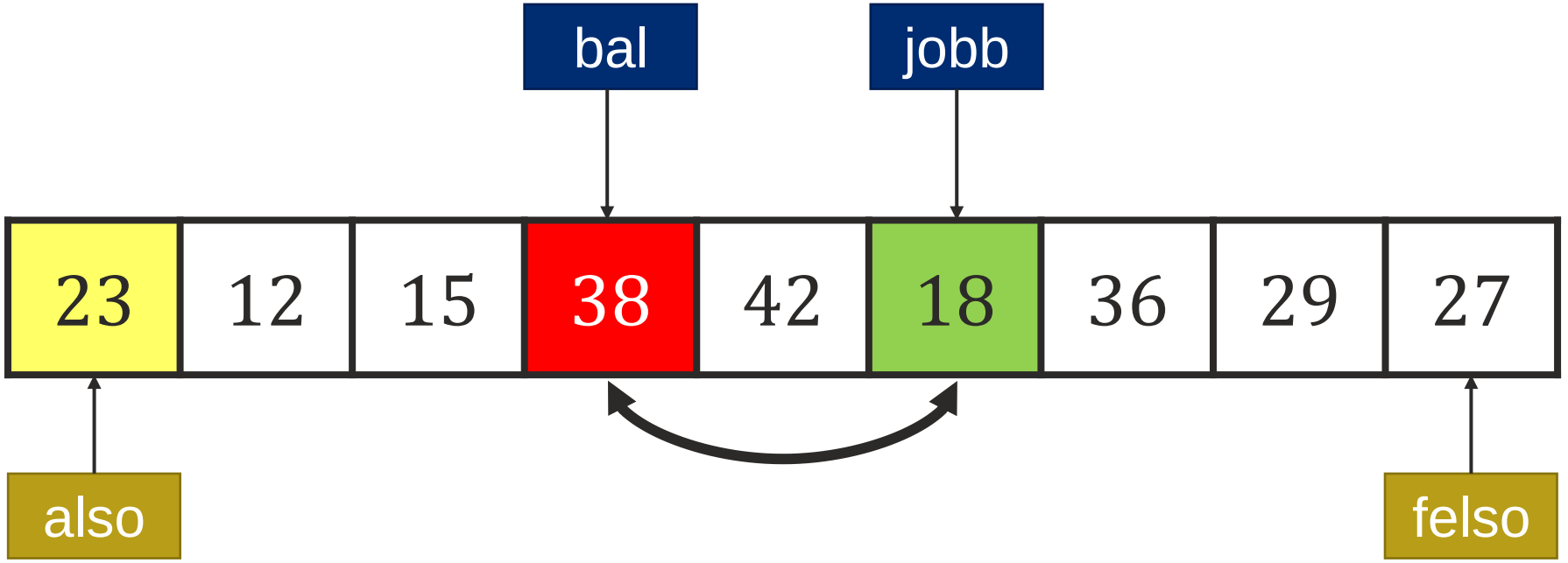
Quicksort – Megosztás

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



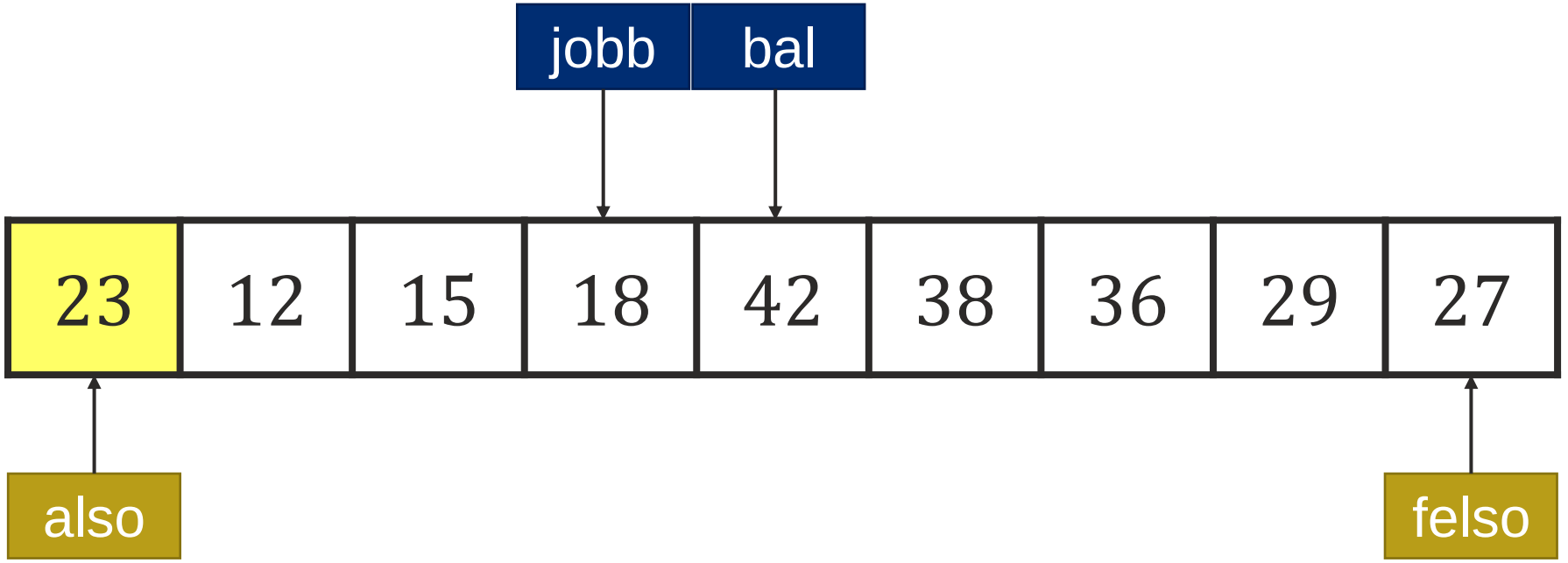
Quicksort – Megosztás

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



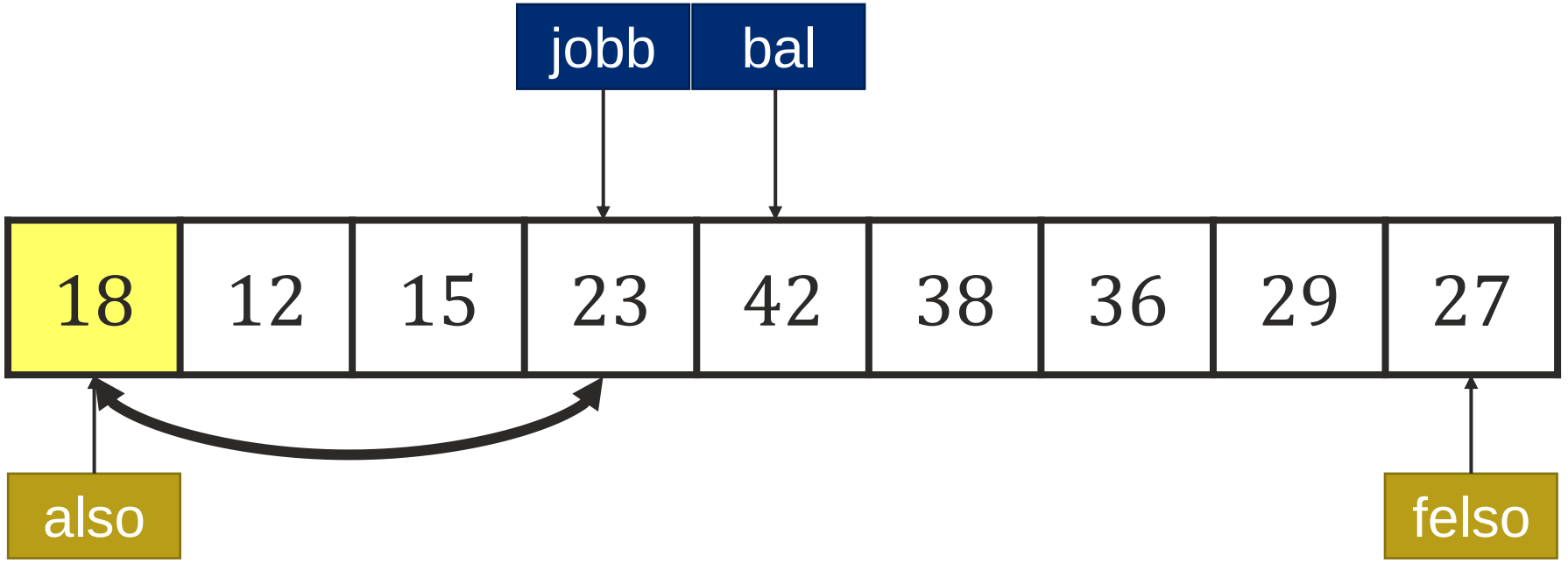
Quicksort – Megosztás

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



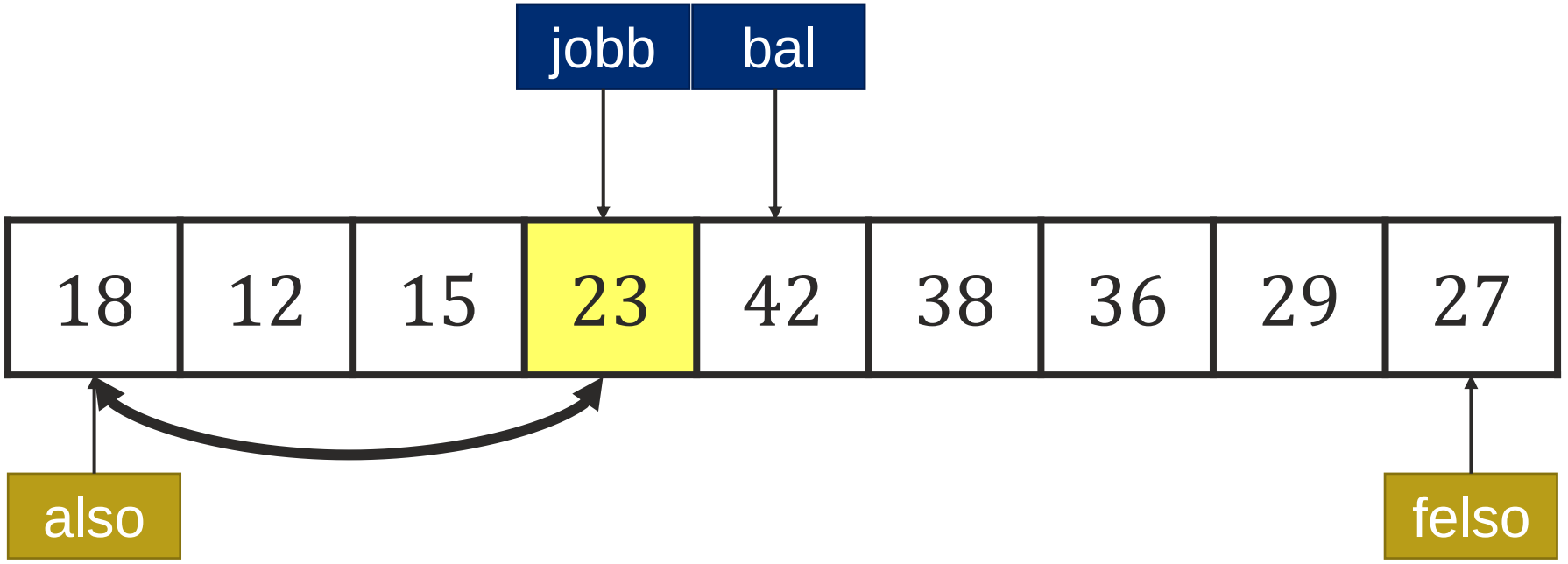
Quicksort – Megosztás

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



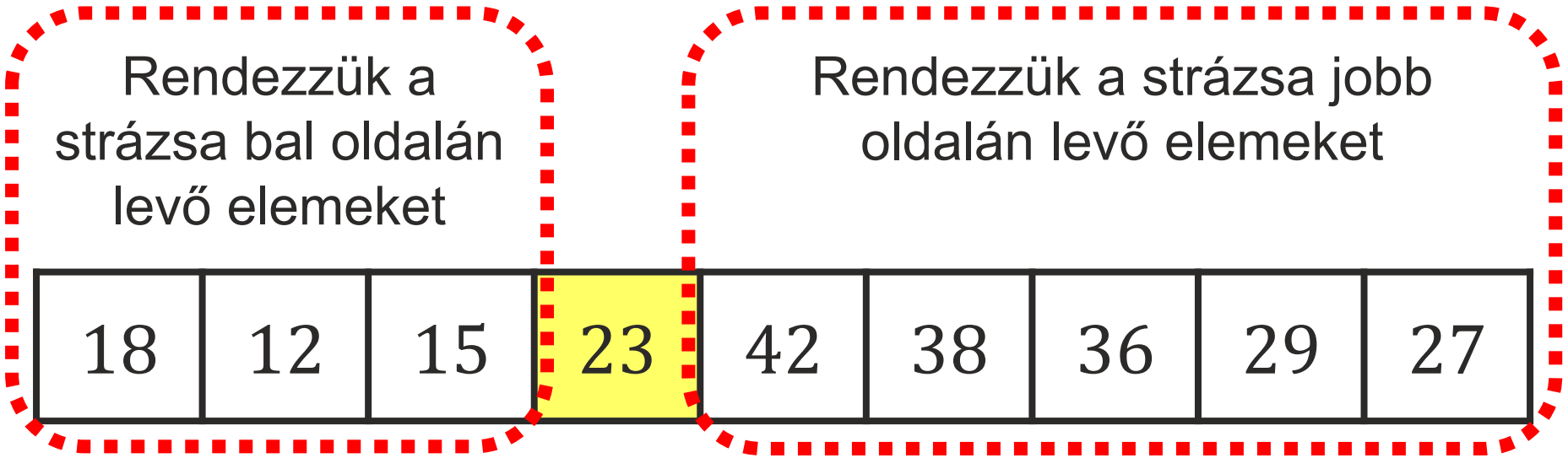
Quicksort – Megosztás

- Folytatjuk a bal és jobb léptetését
  - …
  - A felosztó eljárás visszatér a strázsa indexével (ami a jobb)



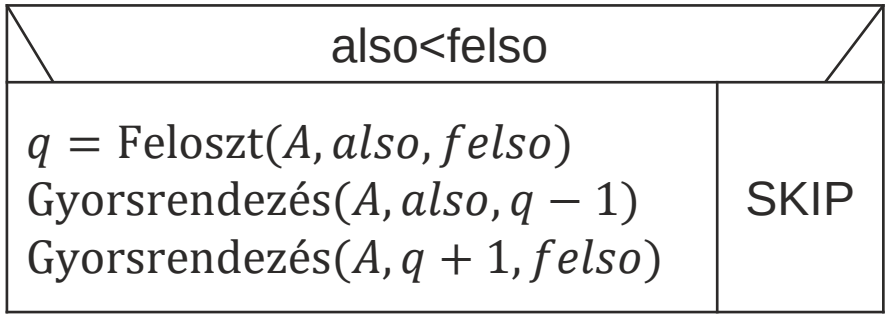
Quicksort – Uralkodj

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

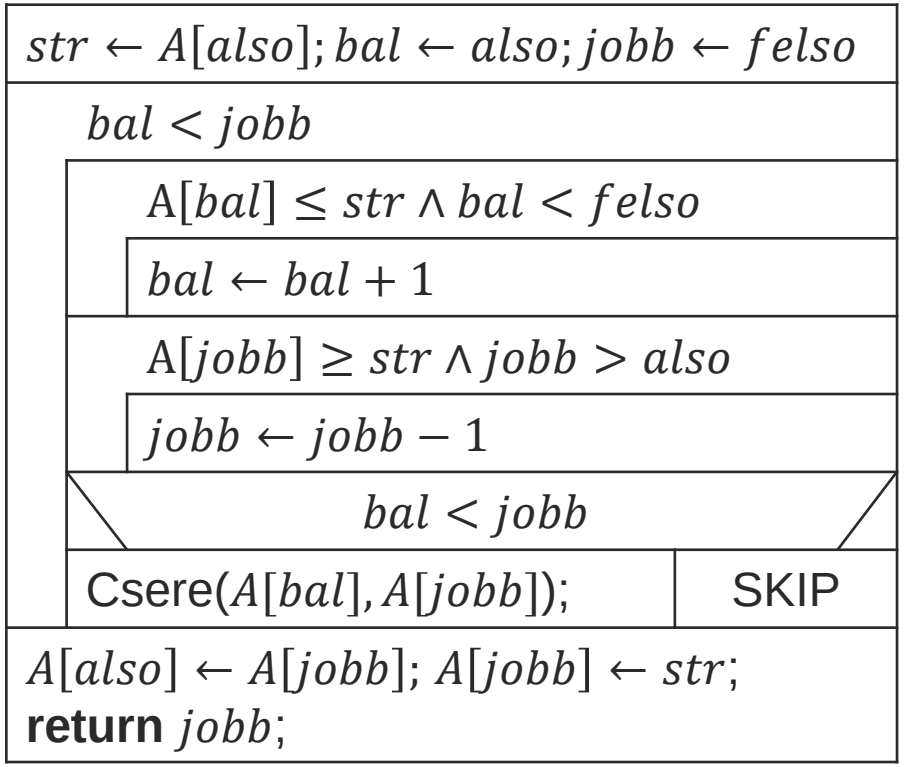


Gyorsrendezés algoritmusa

`Gyorsrendezés (A, also, felso)`



`Feloszt(A, also, felso)`



Quicksort - Analízis

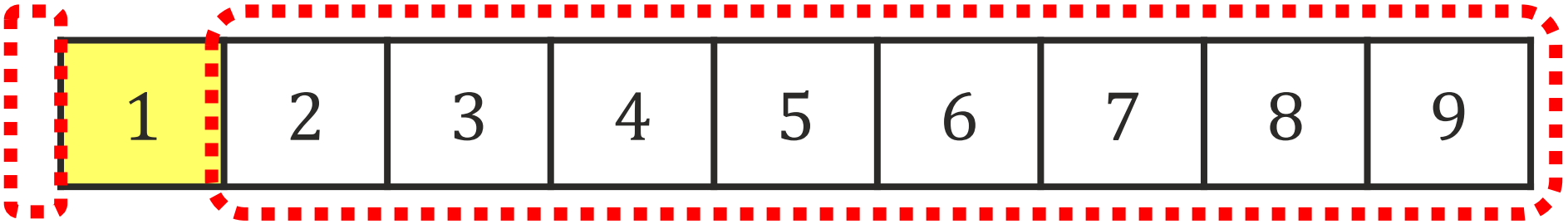
- Felosztás
  - vizsgálj meg minden elemet egyszer $\mathcal{O}(n)$
- Uralkodás
  - az adatok kétfelé osztása $\mathcal{O}(\log_2 n)$
- összesen
  - szorzat $\mathcal{O}(n \log n)$
- De van egy gond …

Quicksort – az igazság!

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

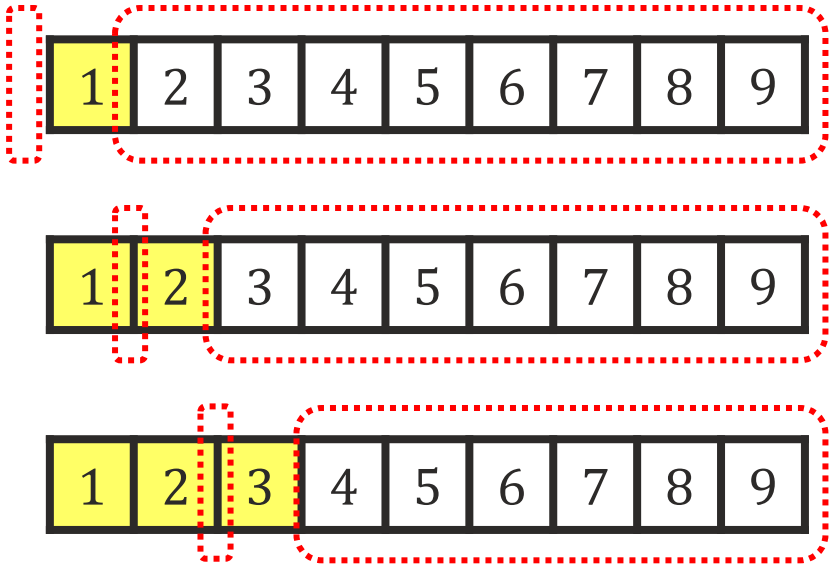
Quicksort – az igazság!

- 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



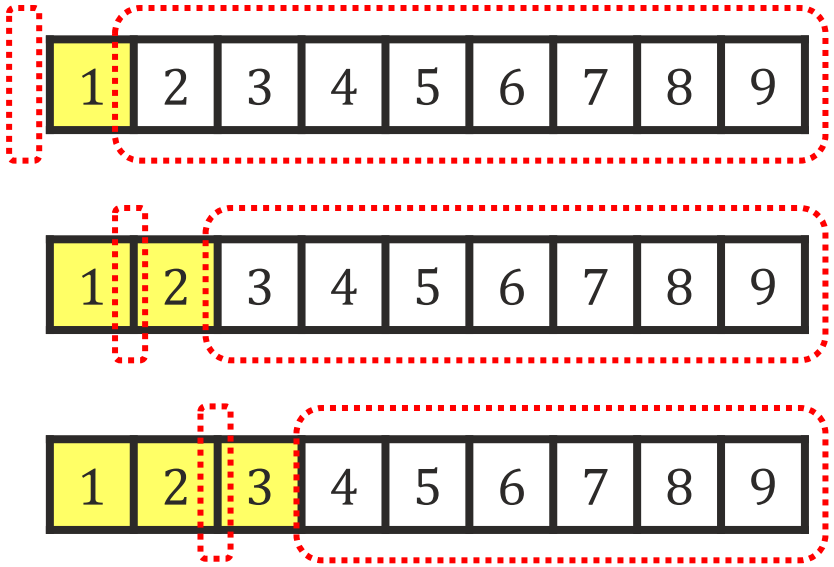
Quicksort – az igazság!

- 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!?

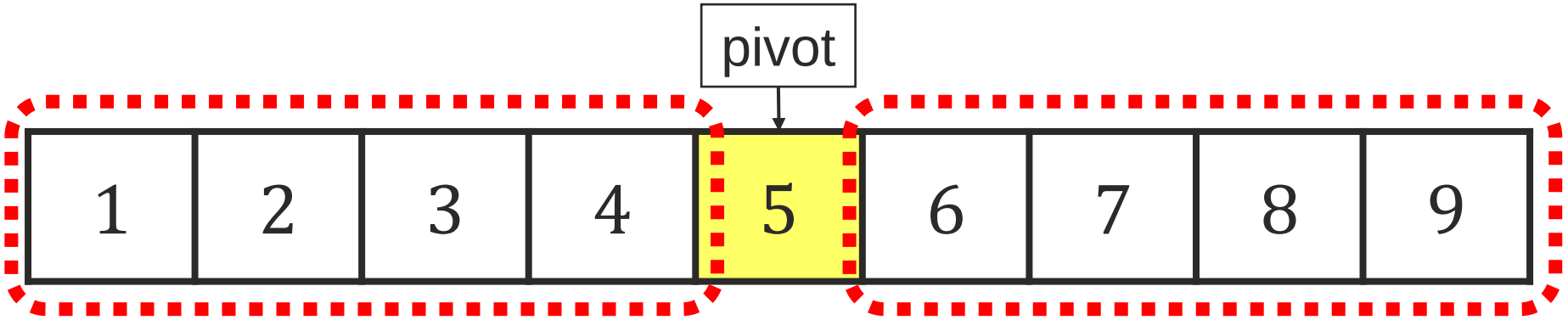


Quicksort – az igazság!

- 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 sem mindig 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

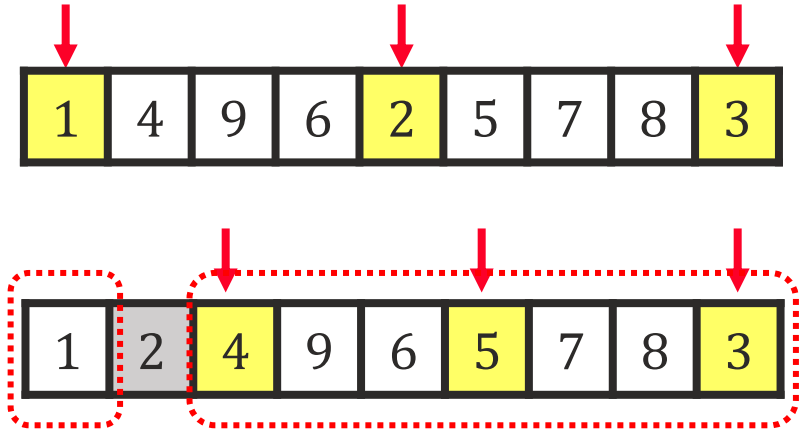
  - Az értékek közül a középső az 5
  - Ez a rendezett adatok esetén jó választás!
  - $\mathcal{O}(n * \log n)$ idő
- Mivel a rendezett (vagy majdnem rendezett) adatok elég gyakoriak, ez egy jó stratégia
  - Különösen, ha azt várjuk, hogy az adataink rendezettek lesznek!

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

Rendezők implementálása

Következő téma