Lassú rendezések és gyorsrendezés

8. GYAKORLAT

Buborék rendezés

BUBBLE-SORT

Buborék rendezés

Analógia: mint ahogy a buborékok szállnak felfelé az ásványvízben.

Az első elemet hasonlítsuk össze a másodikkal, és ha nagyobb annál, akkor cseréljük meg a kettőt. Ismételjük meg a műveletet a második és harmadik elemre, stb. Mire végigérünk a listán, a legnagyobb elem felkúszik a sor végére, majd a második kör után a második legnagyobb az utolsó előtti helyre, és így tovább…

Összehasonlítások és cserék számát tekintve is $O(n^2)$ komplexitású.

Buborék rendezés

Írjátok meg a sorts.hpp-ben a bubbleSort függvényt!

Maximum kiválasztásos rendezés

(MAXIMUM) SELECTION-SORT

Maximum kiválasztásos rendezés

Ötlet: keressük meg a legnagyobb elemet, majd cseréljük fel az utolsó elemmel. Ha megvan, keressük meg a második legnagyobbat (1 és n-1 között keresünk) és tegyük az utolsó előtti pozícióba.

Az algoritmus egyből a végleges pozíciójába mozgat minden elemet.

Cserékre nézve O(n) komplexitású, összehasonlításokra $O(n^2)$.

Figyelem! Hiába használ kevesebb cserét, az összehasonlítások száma miatt a maximum kiválasztásos rendezés műveletigénye még mindig $O(n^2)$ komplexitású.

Maximum kiválasztásos rendezés

Írjátok meg a maximum kiválasztásos rendezést a sorts.hpp-ben!

Beszúró rendezés

INSERTION-SORT

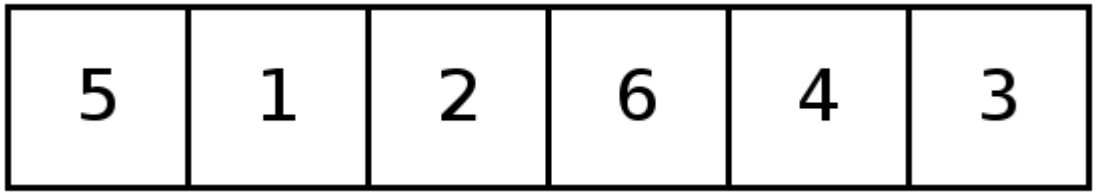
Beszúró rendezés

Ha van egy már rendezett sorozatunk, akkor egy új elemet a megfelelő helyre beszúrva nem rontjuk el a rendezettséget.

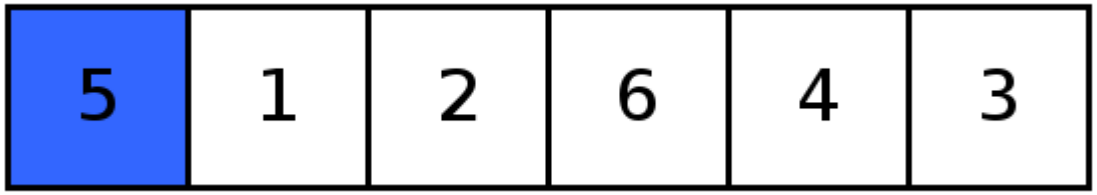
Ötlet: fogjuk fel a rendezendő sorozatot egy rendezett és egy rendezetlen lista egymásutánjaként. A rendezett lista kezdetben egyelemű. A rendezetlen lista elemeit egyesével átrakjuk a rendezettbe, mindig a neki megfelelő pozícióba.

Összehasonlításokra $O(n^2)$ komplexitású. Ha tömbben rendezünk, akkor másolások számára is, ha láncolt listában, akkor ez az érték átláncolás alkalmazásával O(n)-re csökkenthető.

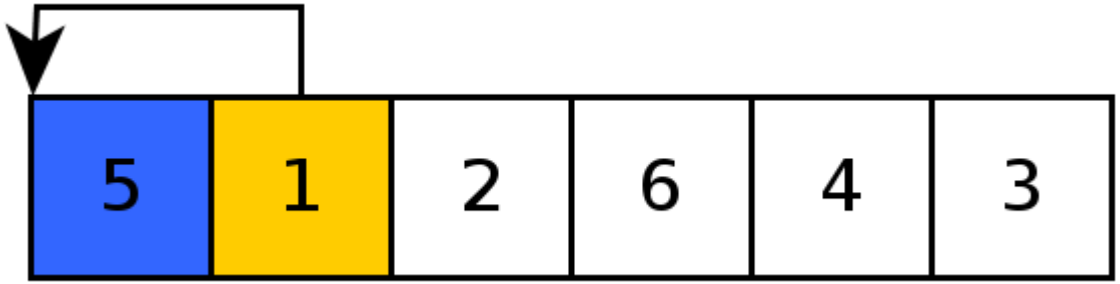
Beszúró rendezés



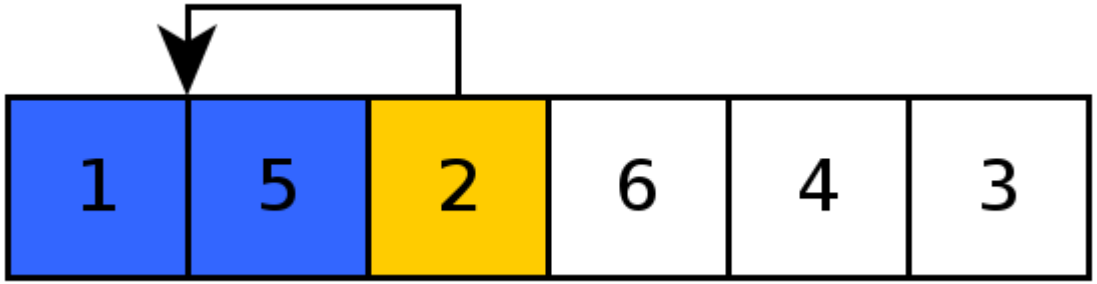
Beszúró rendezés



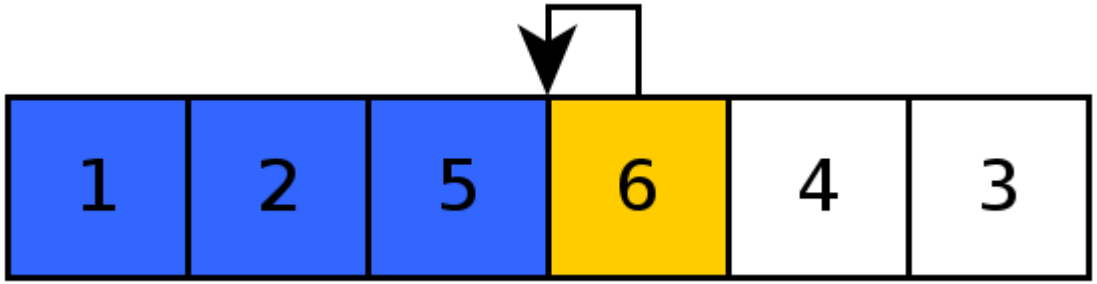
Beszúró rendezés



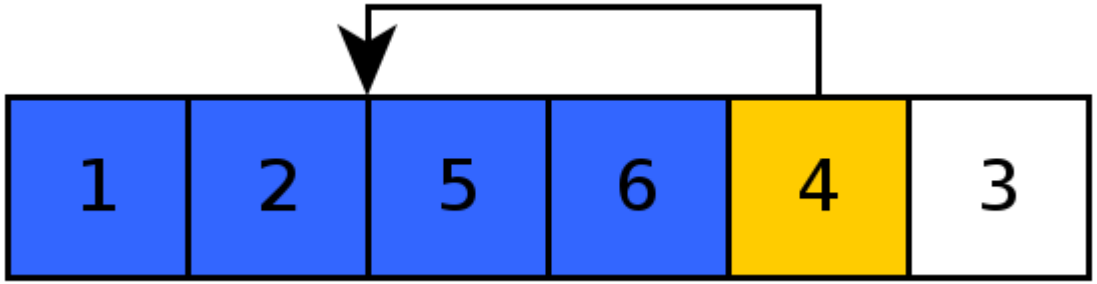
Beszúró rendezés



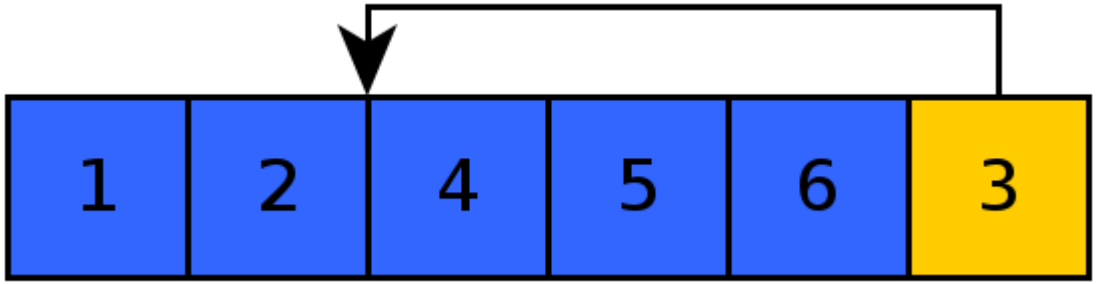
Beszúró rendezés



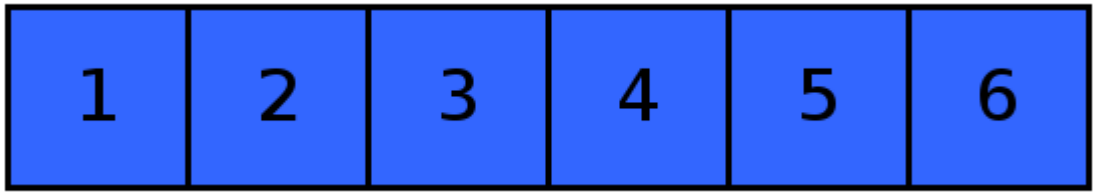
Beszúró rendezés



Beszúró rendezés



Beszúró rendezés



Beszúró rendezés

Írjátok meg a beszúró rendezést a sorts.hpp-ben!

Gyorsrendezés

QUICKSORT

A gyorsrendezés lényege

Az algoritmus két alaplépése:

- Elsőként egy előre kiválasztott elemhez (pivot) képest rendezzük a tömb elemeit (nagyobbak a jobb oldalon, kisebbek a bal oldalon),
- Majd újrakezdjük a rendezést az így kapott két oldalon.

Rendezés általában $O(n \cdot \log_2(n))$ lépés alatt.

Miért jó?

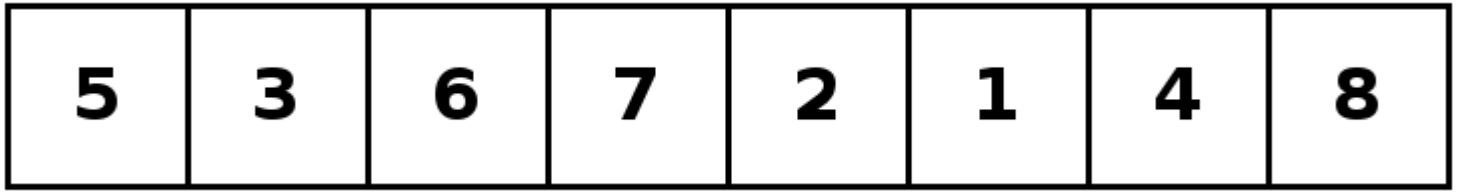
Nem ”túl” bonyolult?

Igaz, hogy legrosszabb esetben a futásidő $O(n^2)$, de átlagos esetben $O(n \cdot \log_2(n))$.

A legrosszabb eset nagyban függ a pivot választástól, így például random pivot választással a legrosszabb eset előfordulási valószínűsége csökkenthető.

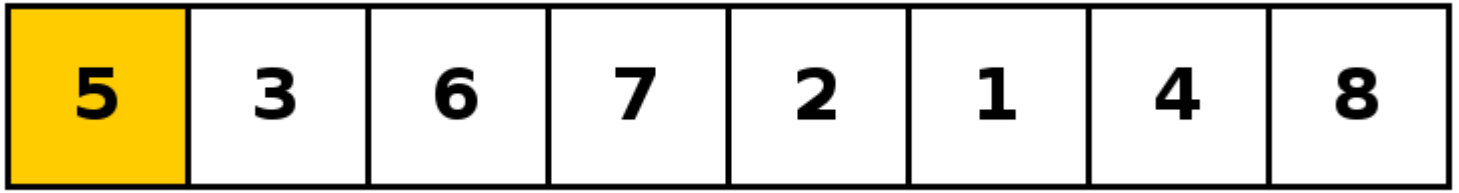
Könnyen párhuzamosítható, mert a kezdeti szétválasztás után a jobbra illetve balra került elemek rendezését végezhetjük egyszerre.

Gyorsrendezés



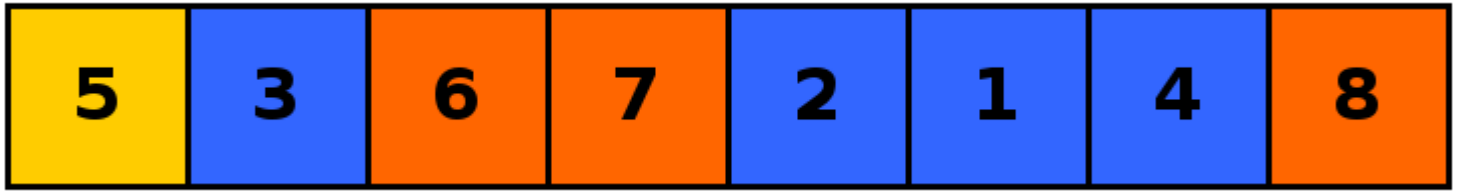
Gyorsrendezés

Sárgával jelöljük a pivotot

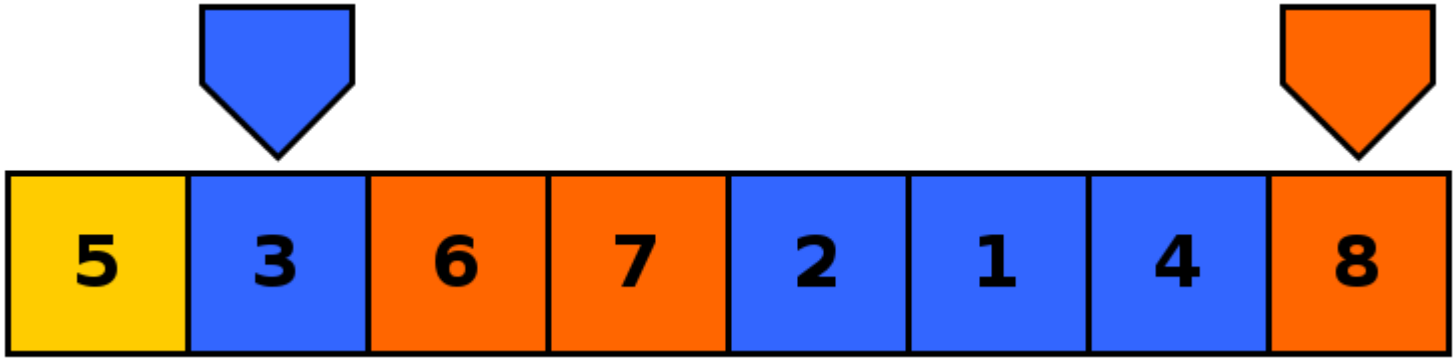


Gyorsrendezés

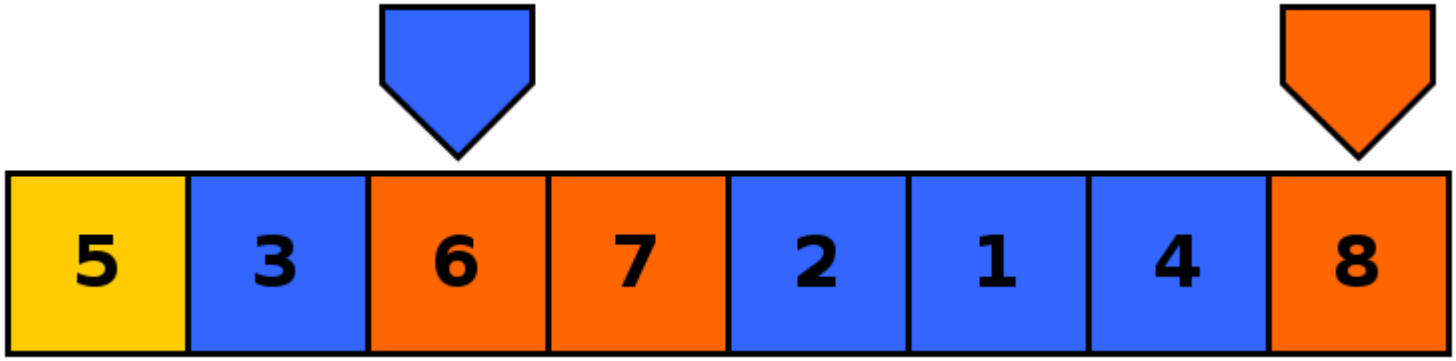
Kékkel jelöljük a pivotnál kisebb, narancssárgával a nagyobb elemeket.



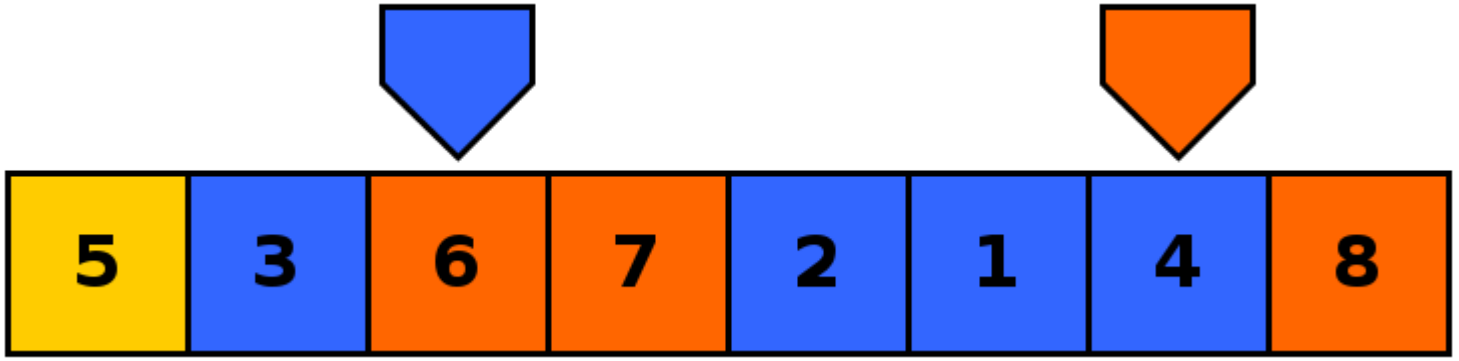
Gyorsrendezés



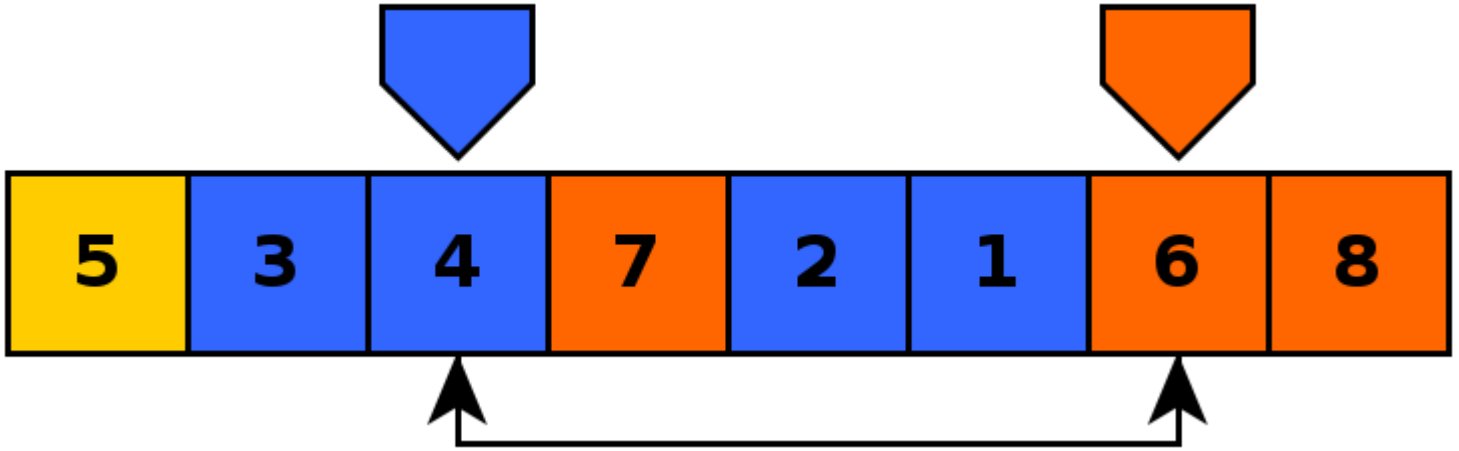
Gyorsrendezés



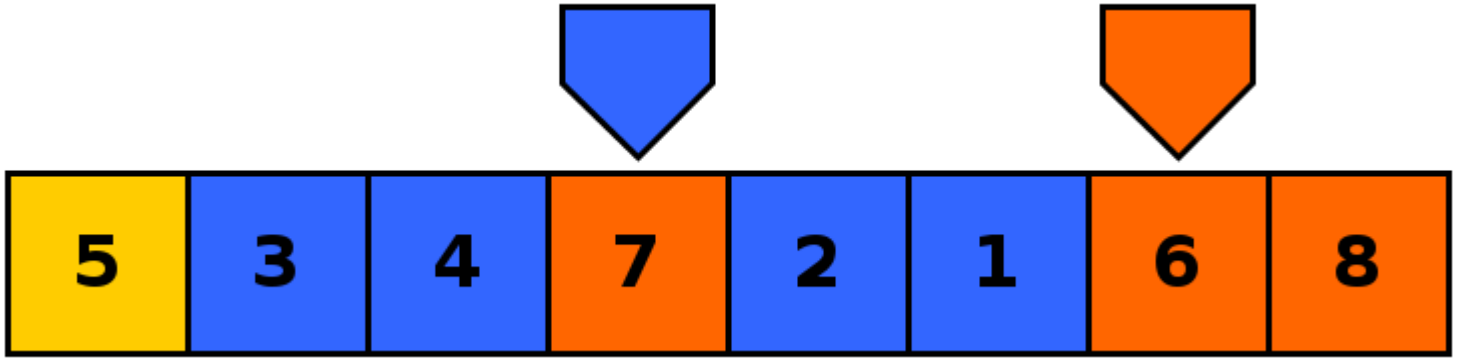
Gyorsrendezés



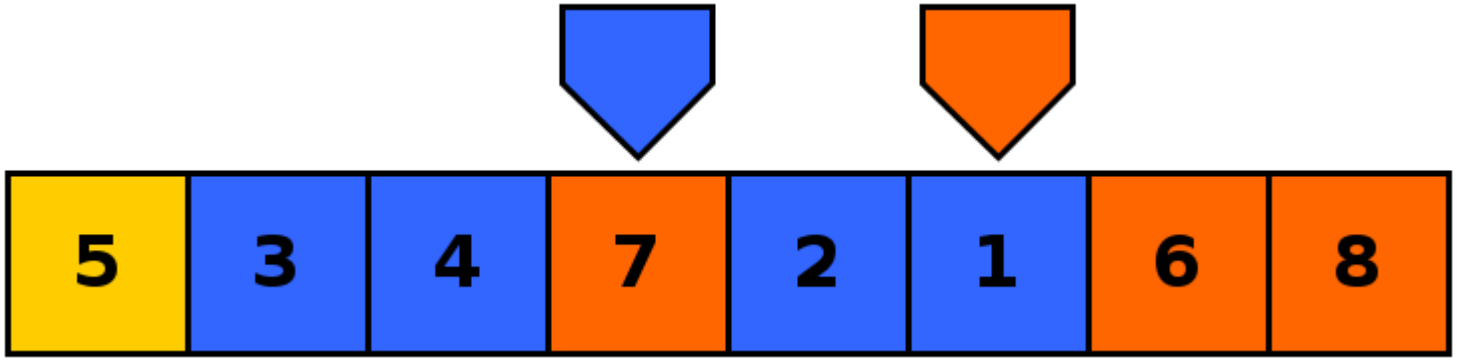
Gyorsrendezés



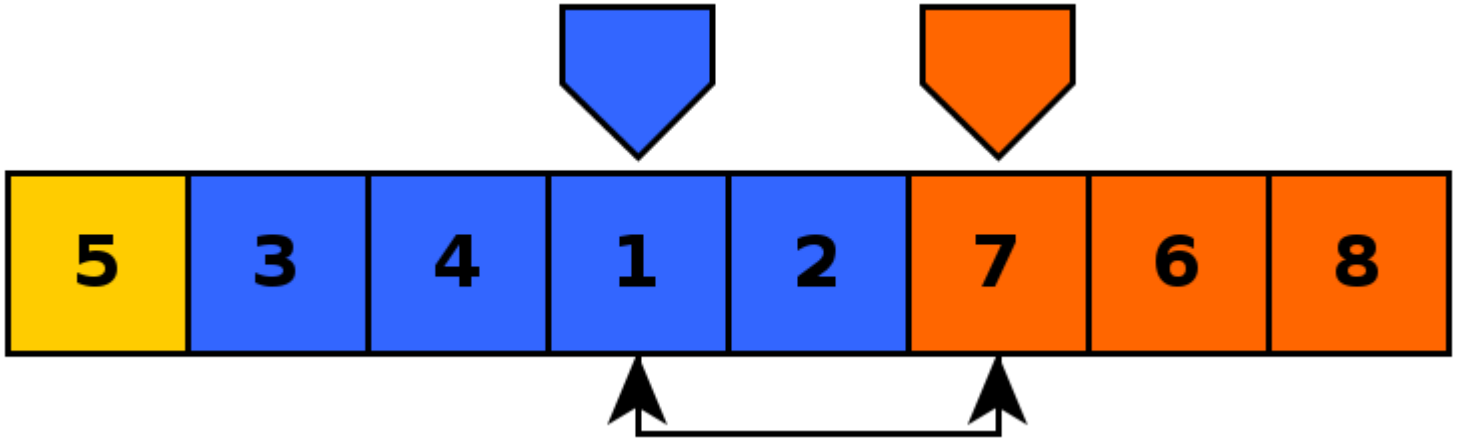
Gyorsrendezés



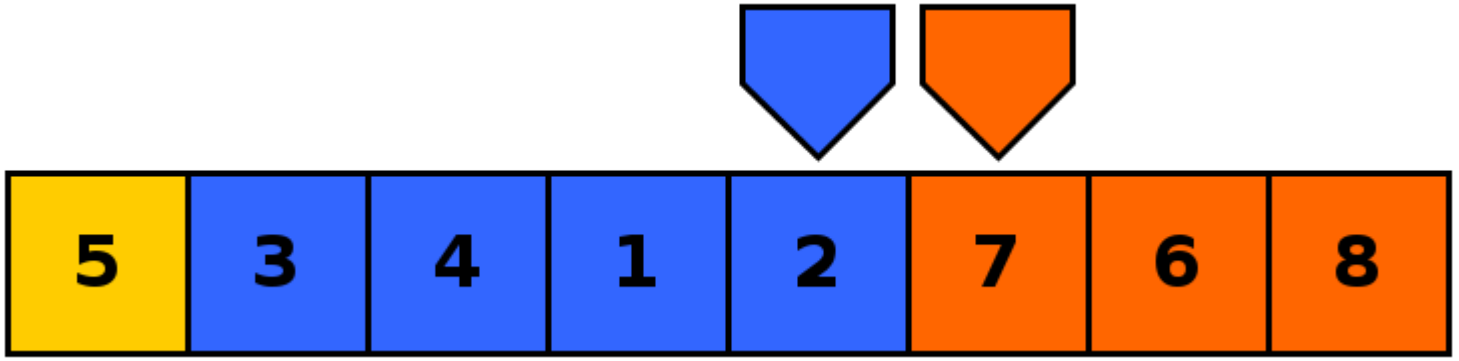
Gyorsrendezés



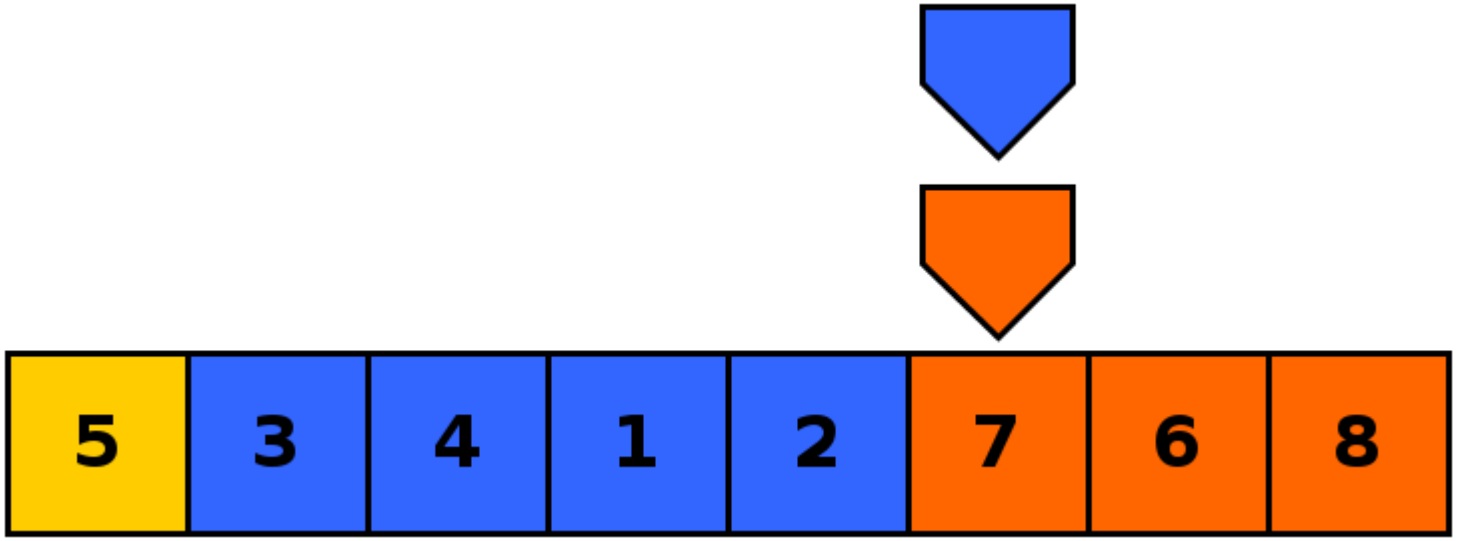
Gyorsrendezés



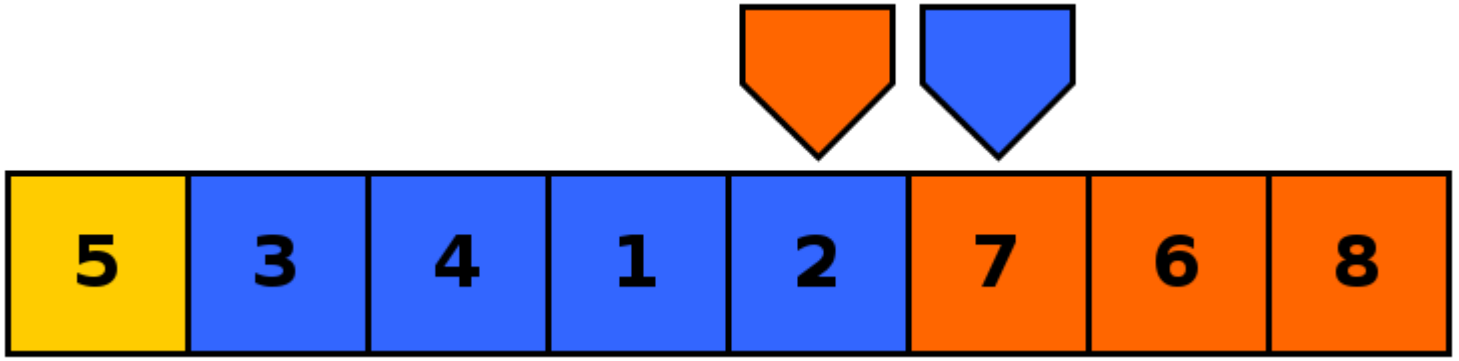
Gyorsrendezés



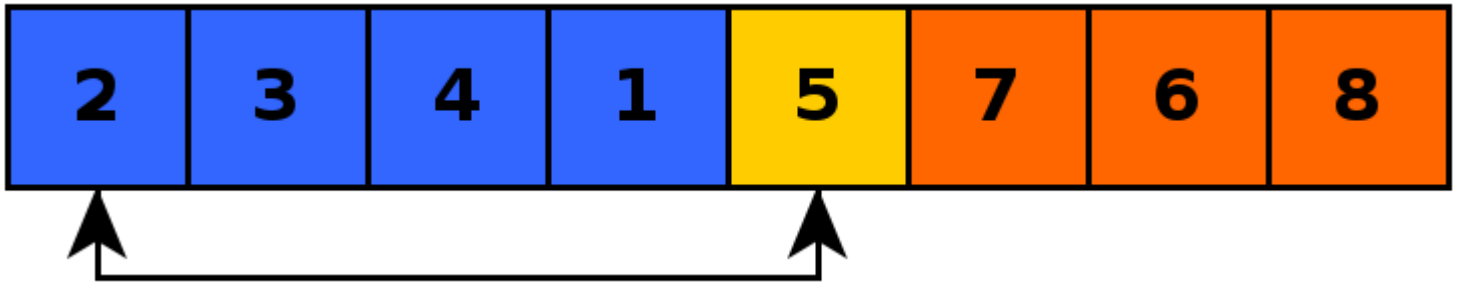
Gyorsrendezés



Gyorsrendezés



Gyorsrendezés



Gyorsrendezés

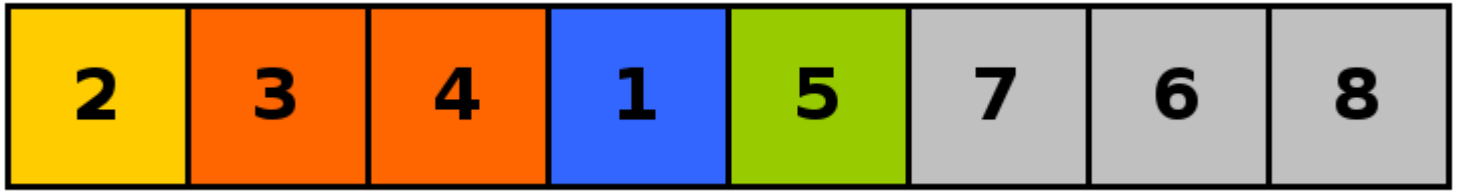
Zölddel jelöljük azokat az elemeket, amik már biztos jó helyen vannak

Gyorsrendezés

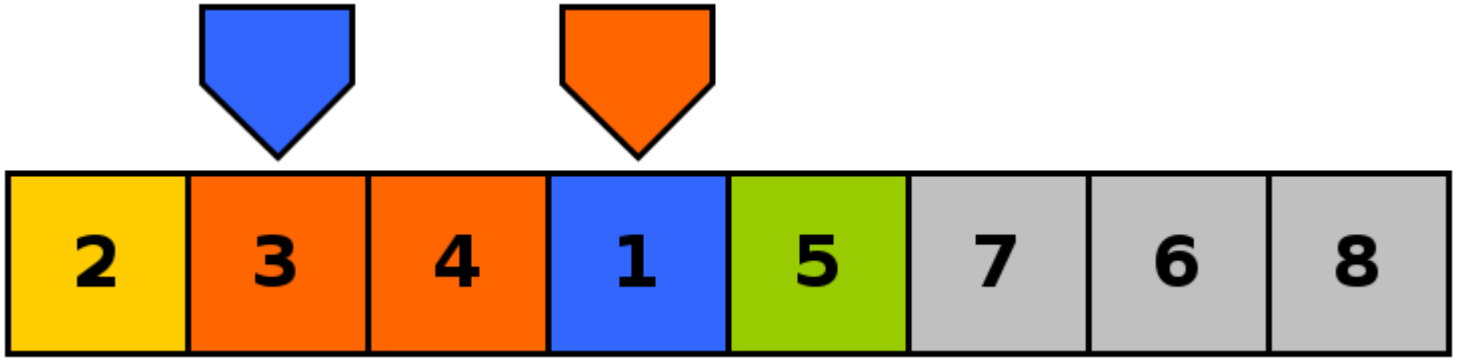
Szürkével jelöljük azokat az elemeket, amikkel éppen nem foglalkozunk

Gyorsrendezés

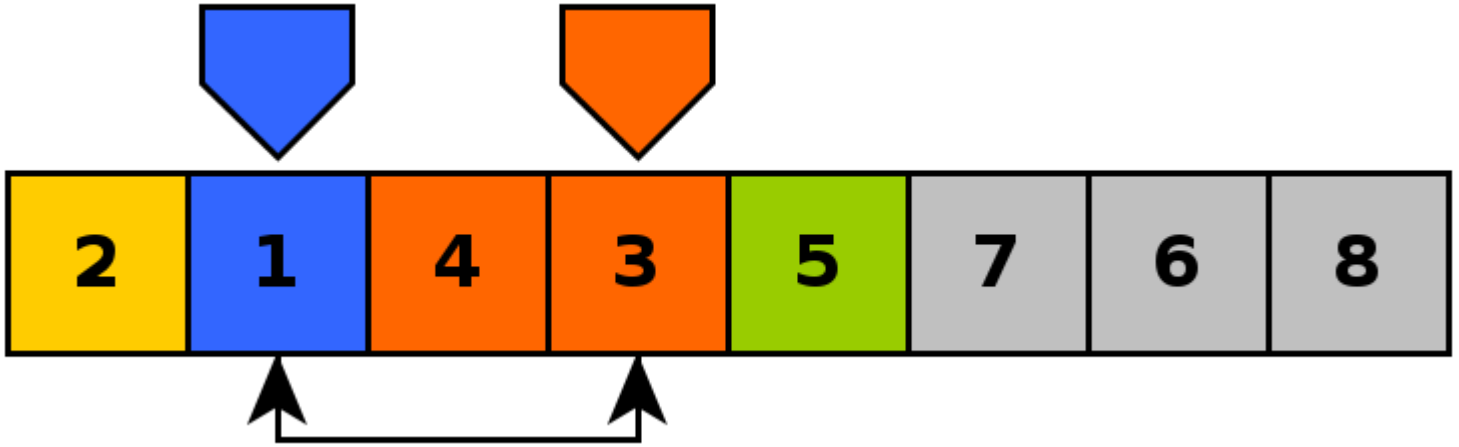
Gyorsrendezés



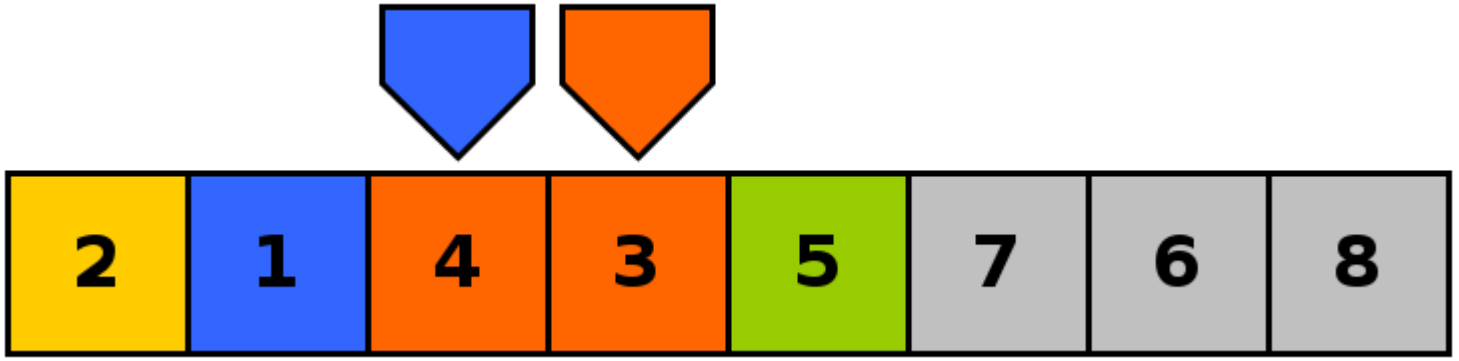
Gyorsrendezés



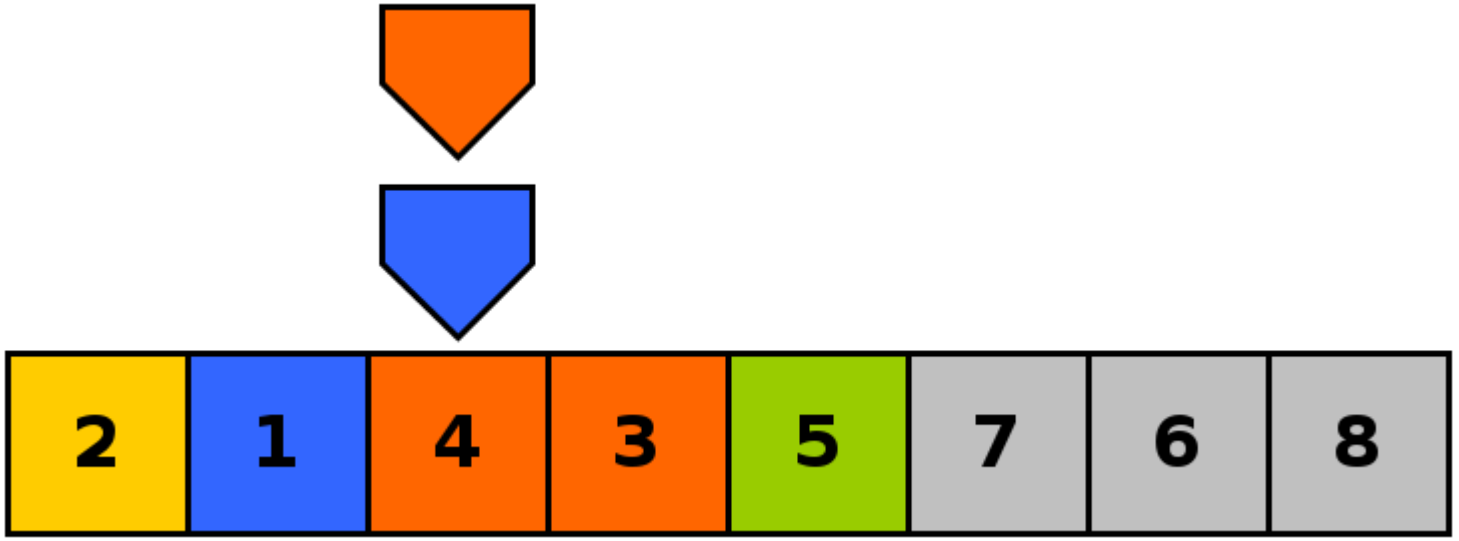
Gyorsrendezés



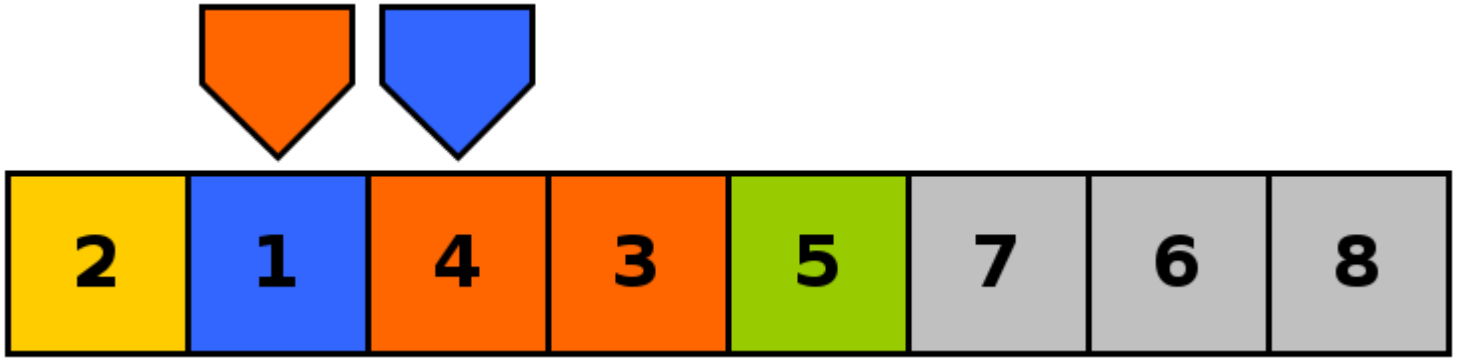
Gyorsrendezés



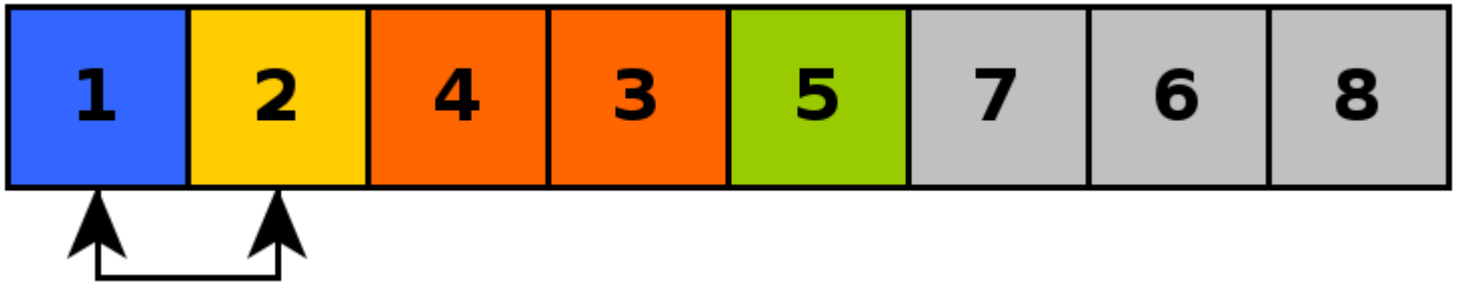
Gyorsrendezés



Gyorsrendezés

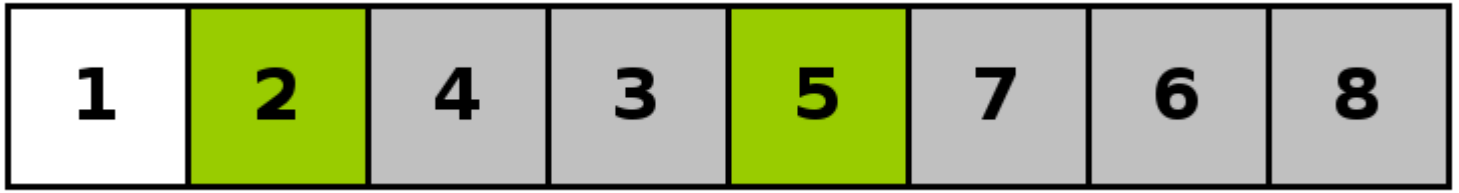


Gyorsrendezés

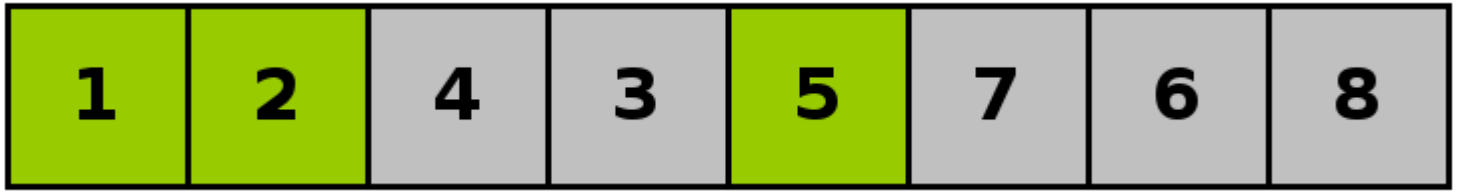


Gyorsrendezés

Gyorsrendezés



Gyorsrendezés

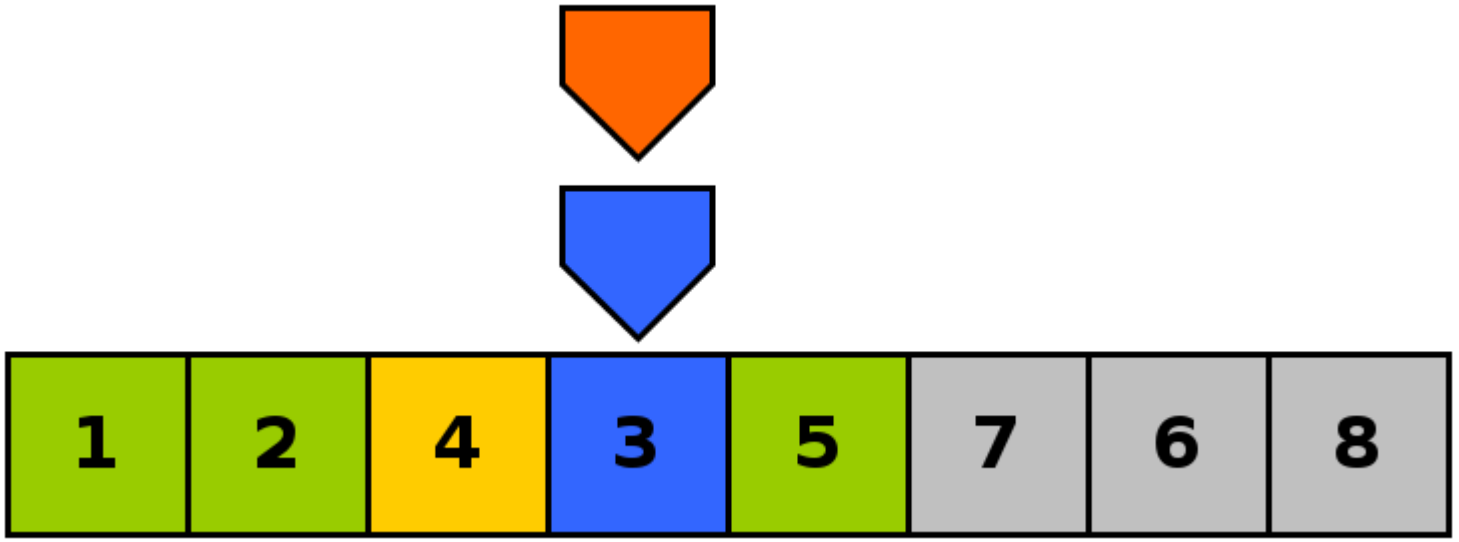


Gyorsrendezés

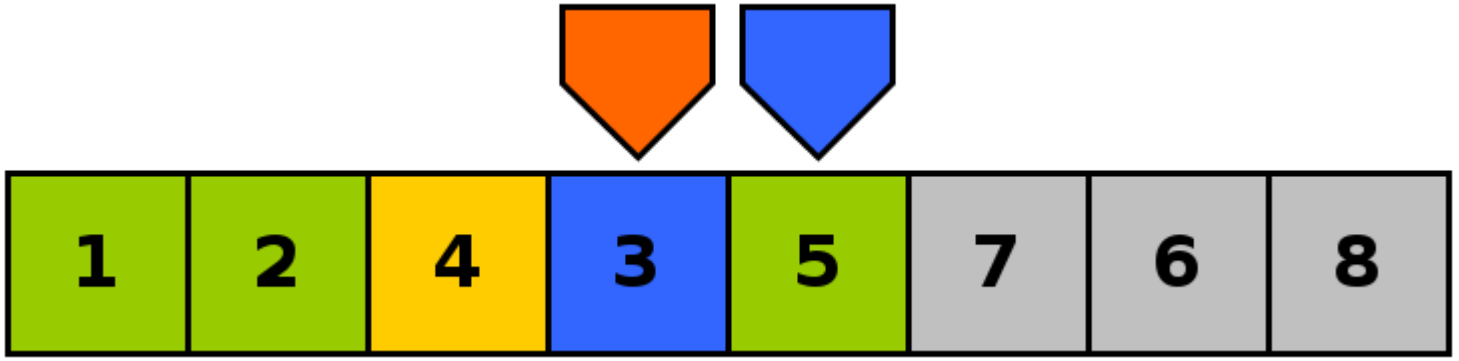
Gyorsrendezés

Gyorsrendezés

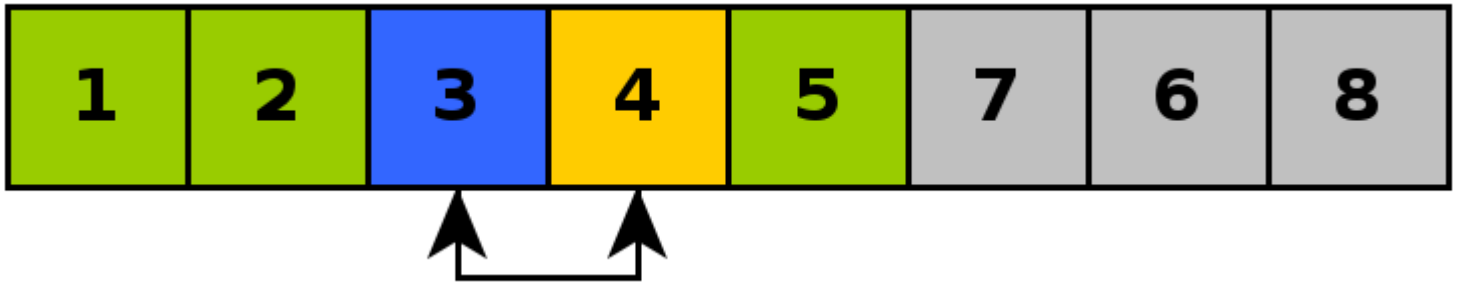
Gyorsrendezés



Gyorsrendezés



Gyorsrendezés



Gyorsrendezés

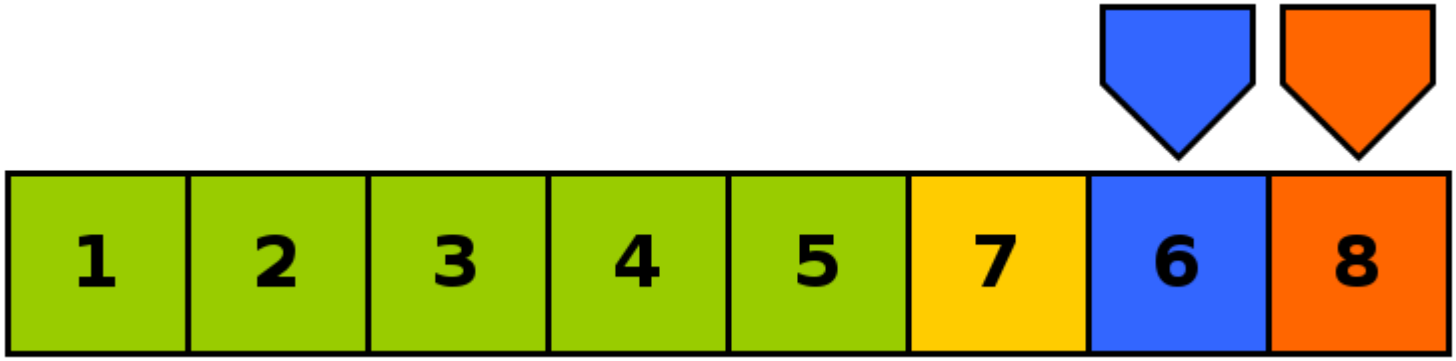
Gyorsrendezés

Gyorsrendezés

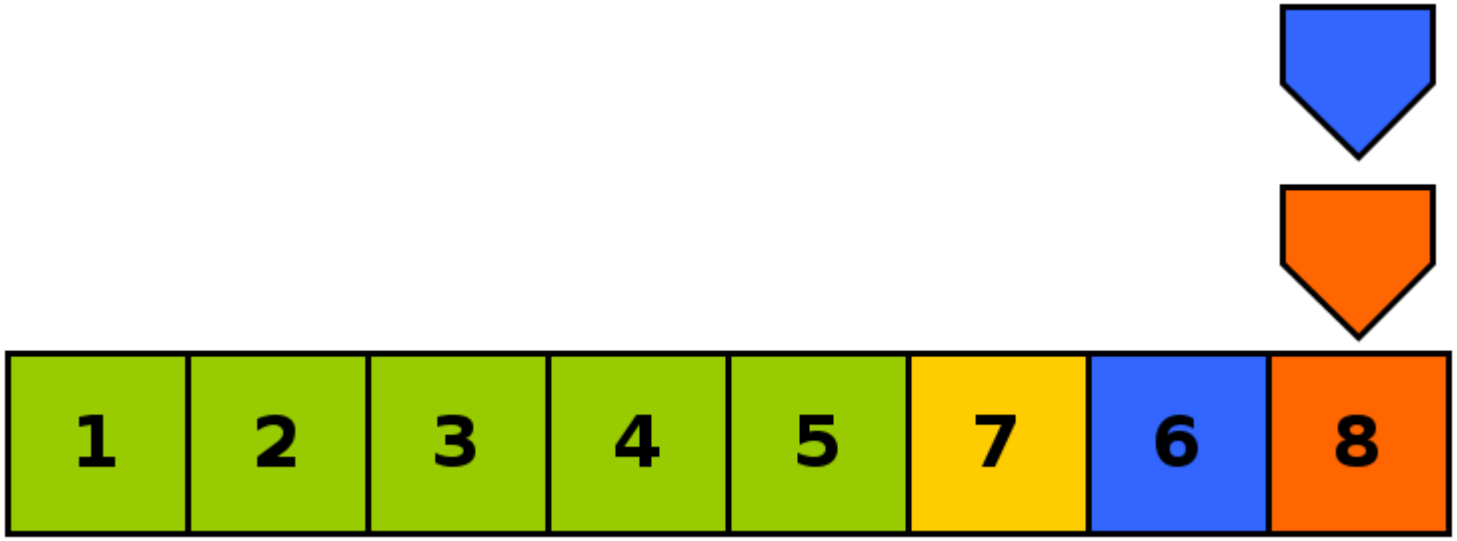
Gyorsrendezés

Gyorsrendezés

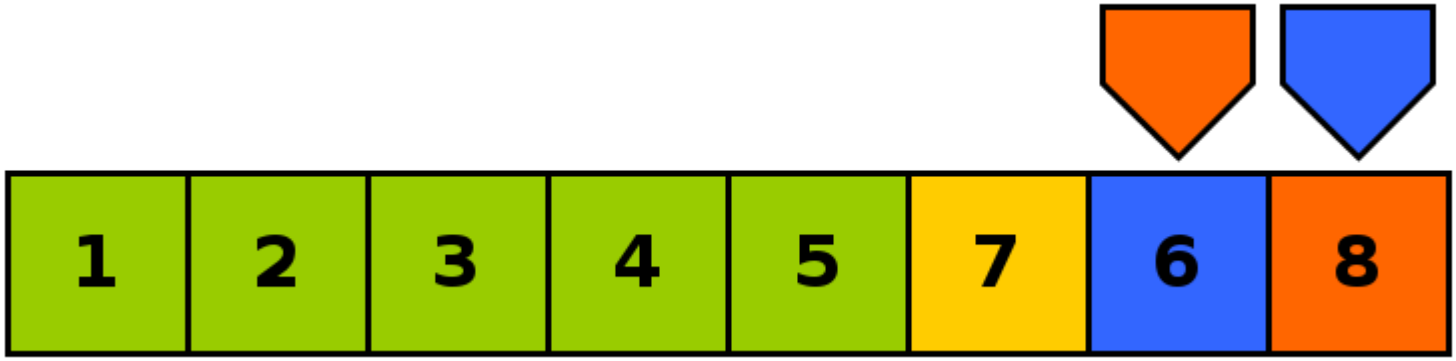
Gyorsrendezés



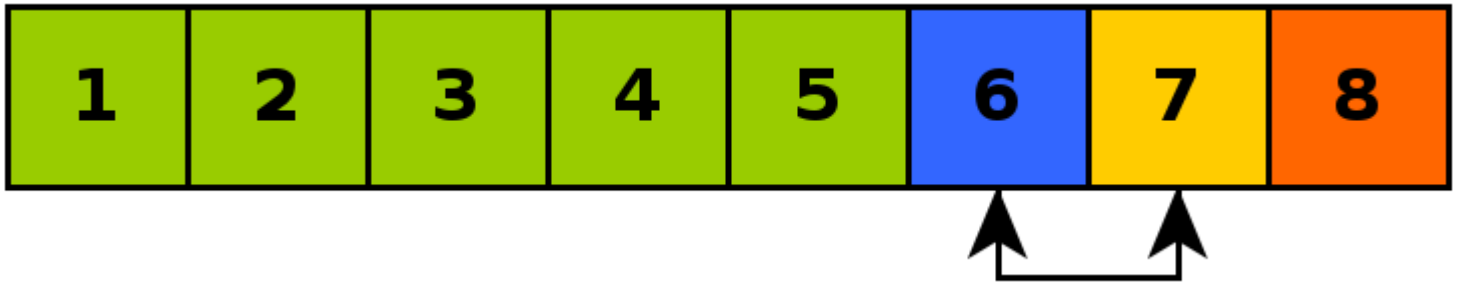
Gyorsrendezés



Gyorsrendezés



Gyorsrendezés



Gyorsrendezés

Gyorsrendezés

Gyorsrendezés

Gyorsrendezés - kódolás

Írjuk meg a hiányzó függvényeket a quicksort.hpp-ben!

Egy kis segítség

Gyorsrendezés (A, also, felso)

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

Feloszt(A, also, felso)

$str \leftarrow A[also]; bal \leftarrow also; jobb \leftarrow felso$

$bal < jobb$

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

$A[also] \leftarrow A[jobb]; A[jobb] \leftarrow str;$
return $jobb;$

Házi feladat

A házi feladat egy olyan függvény implementálása, ami megkeresi egy n elemű tömb k-adik legkisebb elemét úgy, hogy közben esetleg átrendezi az elemeket.

A feladatra **két** megoldást kell készíteni. Az elsőnek O(k·n) időn belül kell megoldást adnia (ezt érdemes a kiválasztásos vagy a buborékrendezésre alapozva elkészíteni), a másodiknak átlagosan O(n) idő alatt (ehhez a gyorsrendezést kell átírni).

Mérd le és vesd össze a két algoritmus átlagos futásidejét különböző méretű tömbök esetén. **Egy tömbméretre nem elég egyetlen mérést végezni.**