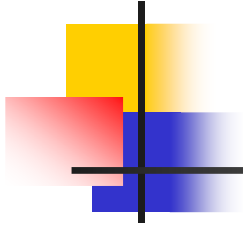


Cours 6 :

Algorithmes de tri rapides

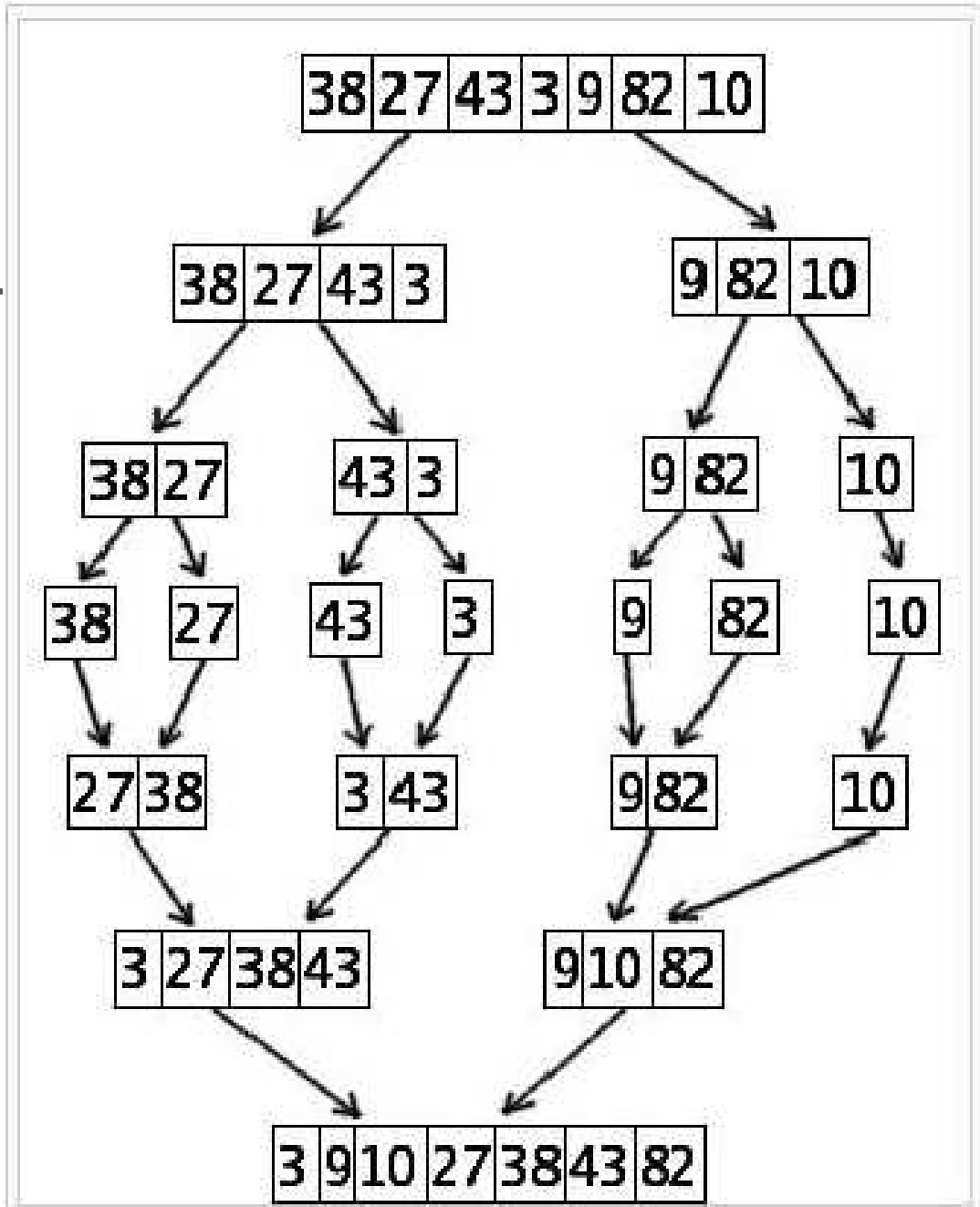
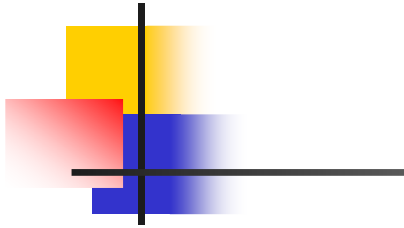


- Tri fusion (merge sort)
- Tri rapide (quick sort)



Tri fusion

- Principe
 - Découper en deux parties à peu près égales les données à trier
 - Trier les données de chaque partie
 - Fusionner les deux parties





Procédure **TriFusion**(ES $A[n]$: Tableau de T, début, fin : Entier)

Variables milieu : Entier

Début

Si (début < fin) Alors

// au moins deux cases

milieu \leftarrow (début + fin)/2

TriFusion(A, début, milieu)

TriFusion(A, milieu+1, fin)

Fusionner(A,debut,milieu,fin)

FinSi

Fin

- Appel: **TriFusion**(A,0,n-1)



Procédure Fusionner(ES A[n] : Tableau de T, début, milieu, fin : Entier)

Variables i, i1, i2 : Entier, Tmp[n] : Tableau de T

Début

i ← 0

i1 ← début

i2 ← milieu + 1

Tantque (i1 ≤ milieu) **et** (i2 ≤ fin)

Si A[i1] < A[i2] **Alors**

 Tmp[i] ← A[i1]

 i1 ← i1 + 1

Sinon

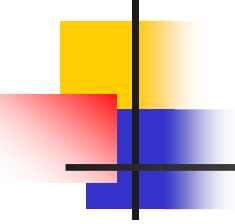
 Tmp[i] ← A[i2]

 i2 ← i2 + 1

FinSi

 i ← i + 1

FinTantque



```
Tantque i1 <= milieu  
  Tmp[i] ← A[i1]  
  i ← i + 1  
  i1 ← i1 + 1
```

```
FinTantque
```

```
Tantque i2 <= fin  
  Tmp[i] ← A[i2]  
  i ← i + 1  
  i2 ← i2 + 1
```

```
FinTantque
```

```
A ← Tmp
```

```
Fin
```



Tri fusion : complexité

- L'équation de récurrence (affectations Tmp):

$$C(1) = 0$$

$$C(n) = n + 2 * C(n/2)$$

=> complexité $O(n * \log n)$



Tri rapide

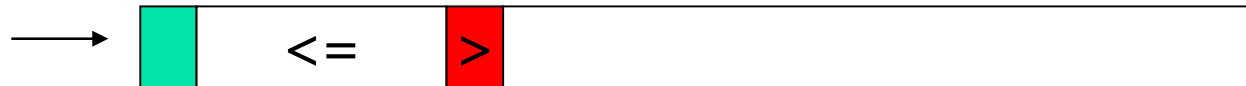
Principe

- Diviser le tableau en deux parties séparées par un **pivot** de telle manière que
 - les éléments de la partie de gauche soient tous inférieurs ou égaux à ce pivot et
 - ceux de la partie de droite soient tous supérieurs à ce pivot
- Itérer d'une manière récursive ce procédé sur les deux parties ainsi créées



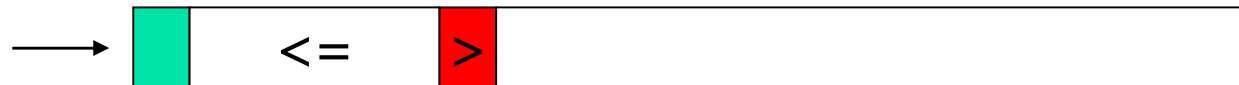
Partitionnement (choix de pivot)

1. Parcourir de gauche à droite jusqu'à rencontrer un élément supérieur au pivot



Partitionnement (choix de pivot)

1. Parcourir de gauche à droite jusqu'à rencontrer un élément supérieur au pivot

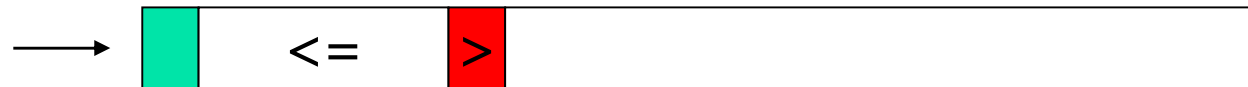


2. Parcourir de droite à gauche jusqu'à rencontrer un élément inférieur au pivot



Partitionnement (choix de pivot)

1. Parcourir de gauche à droite jusqu'à rencontrer un élément supérieur au pivot



2. Parcourir de droite à gauche jusqu'à rencontrer un élément inférieur au pivot

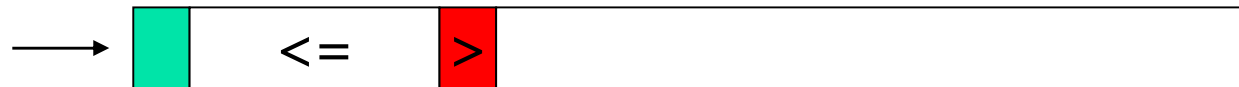


3. Echanger ces deux éléments



Partitionnement (choix de pivot)

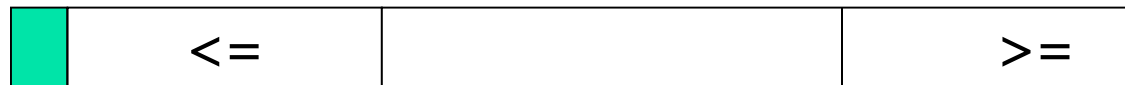
1. Parcourir de gauche à droite jusqu'à rencontrer un élément supérieur au pivot



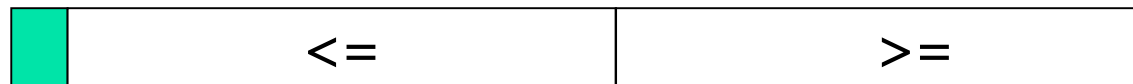
2. Parcourir de droite à gauche jusqu'à rencontrer un élément inférieur au pivot



3. Echanger ces deux éléments

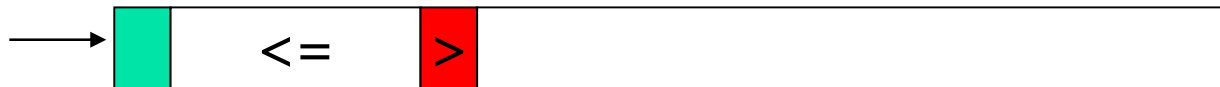


4. Recommencer les parcours gauche-droite et droite-gauche jusqu'à avoir



Partitionnement (choix de pivot)

1. Parcourir de gauche à droite jusqu'à rencontrer un élément supérieur au pivot



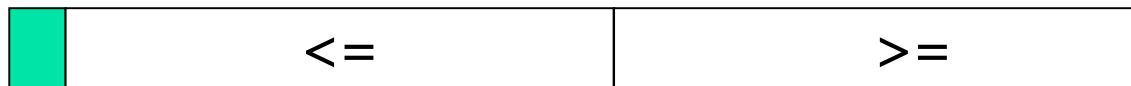
2. Parcourir de droite à gauche jusqu'à rencontrer un élément inférieur au pivot



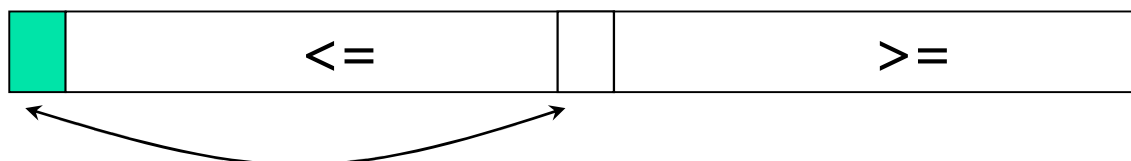
3. Echanger ces deux éléments



4. Recommencer les parcours gauche-droite et droite-gauche jusqu'à avoir



5. Mettre le pivot à la frontière (par un échange)





Exemple

27		63		1		72		64		58		14		9
----	--	----	--	---	--	----	--	----	--	----	--	----	--	---



Exemple

27 | 63 | 1 | 72 | 64 | 58 | 14 | 9

27 | 9 | 1 | 72 | 64 | 58 | 14 | 63



Exemple

27 | 63 | 1 | 72 | 64 | 58 | 14 | 9

27 | 9 | 1 | 72 | 64 | 58 | 14 | 63

27 | 9 | 1 | 14 | 64 | 58 | 72 | 63



Exemple

27 | 63 | 1 | 72 | 64 | 58 | 14 | 9

27 | 9 | 1 | 72 | 64 | 58 | 14 | 63

27 | 9 | 1 | 14 | 64 | 58 | 72 | 63

27 | 9 | 1 | 14 | 64 | 58 | 72 | 63



Exemple

27 | 63 | 1 | 72 | 64 | 58 | 14 | 9

27 | 9 | 1 | 72 | 64 | 58 | 14 | 63

27 | 9 | 1 | 14 | 64 | 58 | 72 | 63

27 | 9 | 1 | 14 | 64 | 58 | 72 | 63

14 | 9 | 1 | 27 | 64 | 58 | 72 | 63



Exemple

27 | 63 | 1 | 72 | 64 | 58 | 14 | 9

27 | 9 | 1 | 72 | 64 | 58 | 14 | 63

27 | 9 | 1 | 14 | 64 | 58 | 72 | 63

27 | 9 | 1 | 14 | 64 | 58 | 72 | 63

14 | 9 | 1 | 27 | 64 | 58 | 72 | 63

14 | 9 | 1 | 27 | 64 | 58 | 72 | 63

...



Tri rapide

Procédure `TriRapide(ES A[n] : Tableau de T, début, fin : Entier)`

Variables `indexPivot : Entier`

Début

Si (`début < fin`) **Alors**

`// au moins deux cases`

`indexPivot ← Partitionner(A, début, fin)`

`TriRapide(A, début, indexPivot - 1)`

`TriRapide(A, indexPivot + 1, fin)`

FinSi

Fin

- Appel: `TriRapide(A,0,n-1)`



Partition

Fonction Partitionner(ES $A[n]$: Tableau de T, début, fin : Entier) : Entier

Variables i, j : Entier, pivot : T

Début

 pivot $\leftarrow A[\text{début}]$, $i \leftarrow \text{début}+1$, $j \leftarrow \text{fin}$

TantQue $i < j$

TantQue $(i < j)$ et $(A[i] \leq \text{pivot})$

$i \leftarrow i+1$

FinTantQue

TantQue $(i < j)$ et $(A[j] \geq \text{pivot})$

$j \leftarrow j-1$

FinTantQue

Si $(i < j)$ **Alors** $\text{echanger}(A[n], i, j)$

FinSi

FinTantQue $\{ i = j \}$

Si $A[i] > \text{pivot}$ **Alors** $i \leftarrow i - 1$

$A[\text{début}] \leftarrow A[i]$, $A[i] \leftarrow \text{pivot}$

Retourner i

Fin



Tri rapide : complexité

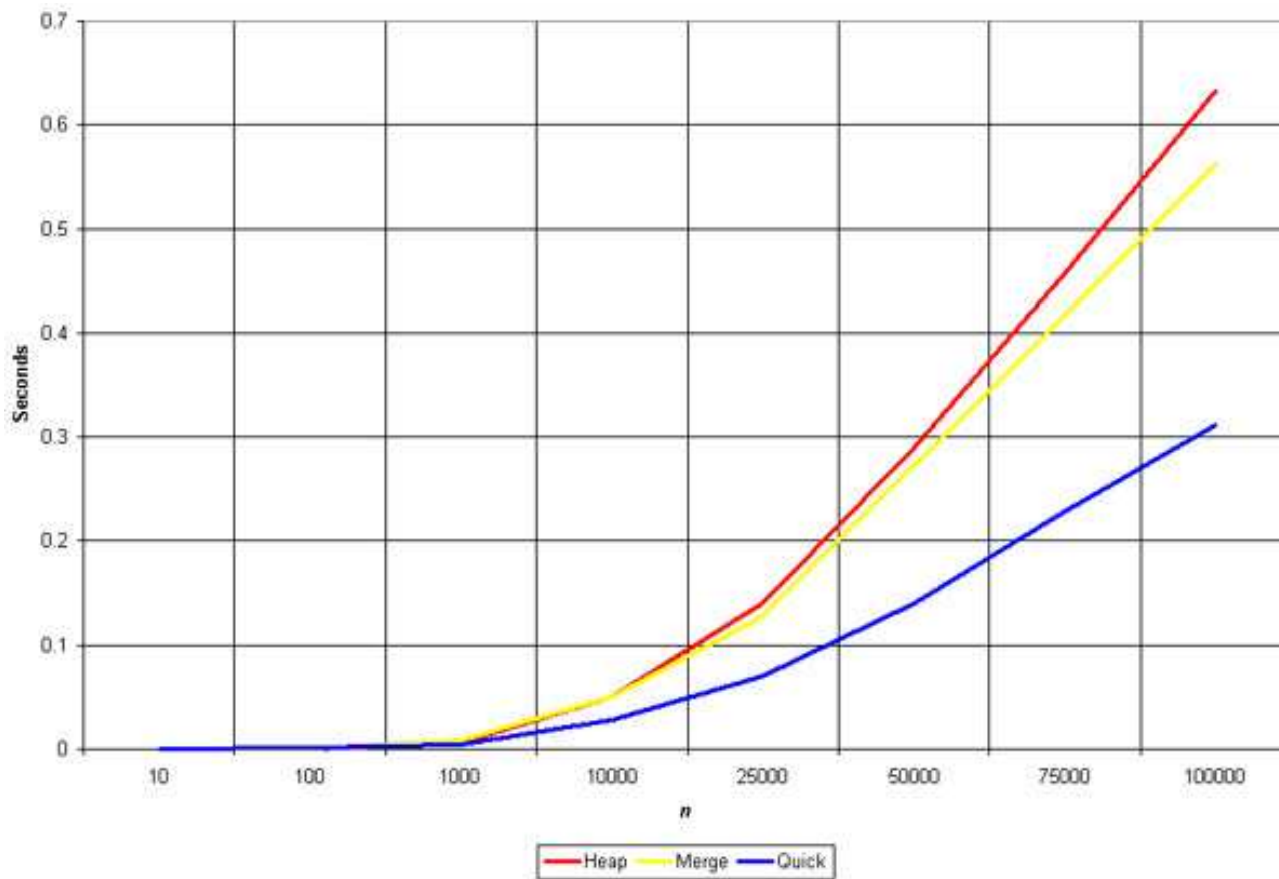
- L'équation de récurrence (**comparaisons**):

$$C(1) = 0$$

$$C(n) = (n-1) + 2 * C(n/2)$$

=> complexité $O(n * \log n)$

Comparaisons des algorithmes de tri rapides





Exercices

- Soit un tableau T avec $T[i] \in \{0,1\}$. Ecrire une fonction qui retourne la position j dans le tableau telle que $T[j]$ est le début de la plus longue suite consécutive de zéros. Si T ne contient aucun 0, la fonction retourne -1.
- Un tableau carré à 2 dimensions contient des lettres à raison d'un caractère par case du tableau. Ecrire une fonction qui détermine si un mot donné est présent dans le tableau *en ligne ou en colonne*. Ce mot est stocké dans un tableau à 1 dimension à raison d'un caractère par case.