

9 Méthodes élémentaires de tri

On a un fichier d'éléments avec **clés comparables** — on veut les ranger selon l'ordre des clés. Clés comparables en Java :

```
public interface Comparable<T>
{
    int compareTo(T autre_objet);
}
```

`x.compareTo(y)` retourne une valeur négative si `x` précède `y`, ou positive si `x` suit `y`, selon l'ordre «naturel» des éléments.

Tri **externe** : fichier stocké partiellement ou entièrement en mémoire externe (disque) — accès à mémoire externe est coûteux...

Tri **interne** : tout le fichier est en mémoire (représenté par un tableau ou une liste chaînée) :

Méthode de tri interne	tableau	liste chaînée
tri par sélection (<i>selection sort</i>)	oui	oui
tri par insertion (<i>insertion sort</i>)	oui	oui
tri par fusion (<i>Mergesort</i>)	oui	oui
tri par tas (<i>Heapsort</i>)	oui	non
tri rapide (<i>Quicksort</i>)	oui	non

9.1 Tri par sélection

Boucler $i \leftarrow 0, \dots, n-2$; propriété invariante à la fin d'itération i :

$\mathcal{P}(i) = \langle A[0..i] \text{ contient les } i \text{ éléments les plus petits, dans l'ordre correct} \rangle$

Algo TRI-SELECTION($A[0 \dots n-1]$)

S1 **pour** $i \leftarrow 0, 1, \dots, n-2$ **faire**

S2 $\text{minidx} \leftarrow i$

S3 **pour** $j \leftarrow i+1, \dots, n-1$ **faire si** $A[j] < A[\text{minidx}]$ **alors** $\text{minidx} \leftarrow j$
 // maintenant $A[\text{minidx}] = \min\{A[i], A[i+1], \dots, A[n-1]\}$

S4 **si** $i \neq \text{minidx}$ **alors** échanger $A[i] \leftrightarrow A[\text{minidx}]$ // maintenant $\mathcal{P}(i)$ est vraie

Complexité de calcul : $\Theta(n^2)$ pour tout A .

★ comparaison d'éléments [ligne S3] : $(n-1) + (n-2) + \dots + 1 = \frac{n(n-1)}{2}$ fois ;

★ échange d'éléments [ligne S4] : $\leq (n-1)$ fois $\Rightarrow$ très efficace si l'échange est beaucoup plus coûteux que la comparaison !

9.2 Tri par insertion

Boucler $i \leftarrow 1, \dots, n-1$; propriété invariante à la fin d'itération i :

$\mathcal{Q}(i) = \langle \text{le sous-tableau } A[0..i] \text{ contient les éléments originaux du même préfixe, dans l'ordre trié.} \rangle$

```

Algo TRI-INSERTION( $A[0 \dots n-1]$ ) // première solution
I1  pour  $i \leftarrow 1, \dots, n-1$  faire
I2     $j \leftarrow i-1$ ; tandis que  $j \geq 0$  et  $A[j] > A[j+1]$  faire échanger  $A[j+1] \leftrightarrow A[j]$ ;  $j \leftarrow j-1$ 

```

Complexité — dépend de l'ordre des éléments au début :

meilleur cas (déjà trié) : $n-1$ comparaisons et aucun échange $\Rightarrow$ très utile si A est «presque trié» au début

pire cas (trié en ordre décroissant) : $\frac{n(n-1)}{2}$ comparaisons et échanges

moyen cas (permutation aléatoire) : $\Theta(n^2)$

Génie algorithmique :

- ★ placer le minimum en $A[0]$ — il sert comme sentinelle.
- ★ remplacer échange par décalage.

```

Algo TRI-INSERTION( $A[0 \dots n-1]$ ) // plus efficace
IM1  minidx  $\leftarrow 0$ ; pour  $i \leftarrow 1, \dots, n-1$  si  $A[i] < A[\text{minidx}]$  alors minidx  $\leftarrow i$ 
IM2  échanger  $A[0] \leftrightarrow A[\text{minidx}]$  // sentinelle : on ne devra pas vérifier  $j \geq 0$  en IM4
IM3  pour  $i \leftarrow 1, \dots, n-1$  faire
IM4     $j \leftarrow i-1$ ;  $a \leftarrow A[i]$ ; tandis que  $A[j] > a$  faire  $A[j+1] \leftarrow A[j]$ ;  $j \leftarrow j-1$ 
IM5     $A[j+1] \leftarrow a$ 

```

9.3 Tri de liste chaînée

On peut implanter la logique de ces tris élémentaires avec des listes chaînées aussi. Soit L une liste chaînée, avec une sentinelle (nœud factice) head à la tête, et null à la fin. Chaque nœud N possède la valeur $N.\text{val}$ et une référence à son successeur $N.\text{prochain}$. L'algorithme ci-dessous performe un tri par sélection.

```

Algo TRI-SELECTION-LISTE( $L$ )
SL1   $I \leftarrow \text{head}$ 
SL2  tandis que  $I.\text{prochain} \neq \text{null}$  faire
SL3    min  $\leftarrow I$ ;  $J \leftarrow I.\text{prochain}$ 
SL4    tandis que  $J.\text{prochain} \neq \text{null}$  faire
SL5      si  $J.\text{prochain}.\text{val} < \text{min}.\text{prochain}.\text{val}$  alors min  $\leftarrow J$ 
SL6    // maintenant min.prochain est le nœud avec val minimale dans la sous-liste après I
SL7    si min  $\neq I$  alors
SL8       $M \leftarrow \text{min}.\text{prochain}$ ; min.prochain  $\leftarrow M.\text{prochain}$  // suppression après min
SL9       $M.\text{prochain} \leftarrow I.\text{prochain}$ ;  $I.\text{prochain} \leftarrow M$  // insertion après I
SL10    $I \leftarrow I.\text{prochain}$ 

```