# Tris 1

**Objectifs**

— Comprendre le principe, la mise en œuvre et la complexité du tri sélection.
— Comprendre le principe, la mise en œuvre et la complexité du tri à bulles.

## Introduction

### Qu'est-ce que trier ?

**Définition**

Le problème du tri en informatique consiste à réorganiser un ensemble de données en vue de faciliter leur traitement. Bien qu'en pratique les données à trier puissent être d'une grande variété, il peut se résumer au problème suivant :

| | |
|---|---|
| *Problème* | Tri |
| *Entrée* | Tableau d'entiers $T$ de taille $n$ |
| *Sortie* | Une permutation des éléments de $T$ telle que $T[1] \leq T[2] \leq \cdots \leq T[n]$ |

Cela nécessite que les éléments puissent être comparés entre eux.

### Pourquoi trier ?

C'est historiquement le problème le plus étudié en informatique et de nombreuses idées à la base des algorithmes de tri sont réutilisées dans de nombreux contextes. En particulier, trier des données est à la base de nombreux algorithmes :

— recherche d'élément
— plus proche paire
— unicité

— enveloppe convexe

### Propriétés des tris étudiés

L'objectif de ce chapitre est de présenter des algorithmes dits *standards* qui, malgré leur relative inefficacité, permettent de mettre en œuvre de nombreuses méthodes et techniques fondamentales en algorithmique.

> **Propriétés**
>
> Ces tris partagent 3 grandes caractéristiques :
> — Ce sont des tris **comparatifs**, c'est-à-dire qu'ils reposent exclusivement sur l'opération de comparaison des éléments
> — Ce sont des tris **in situ** (en place) car seul un nombre constant d'éléments est stocké hors du tableau lors du tri
> — Ils opèrent en échangeant les valeurs de deux cases du tableau

Voici l'algorithme `Echanger` de complexité $\Theta(1)$ chargé d'effectuer de tels échanges :

```
ALGORITHME Echanger(T, i, j):
DONNEES
  T : tableau d'entiers
  i, j : entiers
VARIABLES:
  aux : entier
DEBUT
  aux ← T[i]
  T[i] ← T[j]
  T[j] ← aux
FIN
```

Nous verrons dans le prochain chapitre un autre tri de ce genre.

## Tri sélection

### Principe

— On cherche le plus petit élément du tableau
— On l'échange avec la première position non triée

Ainsi, le début du tableau est trié, et on recommence jusqu'à avoir trié tout le tableau.

**Exemple visuel**

Soit le tableau initial : `[4, 6, 8, 1, 7, 2, 5, 3]`

1. **Étape 1** : Chercher le minimum (1) et l'échanger avec la 1ère position
   — Résultat : `[1, 6, 8, 4, 7, 2, 5, 3]`
2. **Étape 2** : Chercher le minimum du reste (2) et l'échanger avec la 2ème position
   — Résultat : `[1, 2, 8, 4, 7, 6, 5, 3]`
3. **Étape 3** : Chercher le minimum du reste (3) et l'échanger avec la 3ème position
   — Résultat : `[1, 2, 3, 4, 7, 6, 5, 8]`
4. … et ainsi de suite jusqu'à ce que tout soit trié
5. **Final** : `[1, 2, 3, 4, 5, 6, 7, 8]`

*Note : Les  représentent les éléments déjà triés*

**Pseudo-code**

```
ALGORITHME TriSelection(T):
DONNEES
  T : tableau d'entiers de taille n
VARIABLES:
  i, imin, debut : entiers
DEBUT
  debut ← 1
  TQ debut < n FAIRE
    i ← debut + 1
    imin ← debut
    TQ i   n FAIRE
      SI T[i] < T[imin] ALORS
        imin ← i
      FSI
      i ← i+1
    FTQ
    Echanger(T, debut, imin)
    debut ← debut + 1
  FTQ
FIN
```

### Arrêt

Dans la boucle intérieure, $i$ prend les valeurs d'une suite arithmétique strictement croissante donc $i$ dépassera $n$, ce qui prouve l'arrêt de cette boucle. De même pour la boucle principale qui dépend de la variable *debut*.

Ainsi, l'algorithme s'arrête.

### Validité

Posons la propriété $\mathcal{P}(k)$ suivante pour l'entier $k \in [1, n-1]$ :

$$\mathcal{P}(k) : \text{"À la fin de l'itération } k, \text{ le tableau de 1 à } k \text{ est trié dans l'ordre croissant et } T[k] \text{ est inférieur au minimum du tableau de } k+1 \text{ à } n\text{"}$$

— **Initialisation :** Lors du premier passage dans la boucle principale, on parcourt le tableau en recherchant le minimum et on l'échange avec la première place. Ainsi, $T[1]$ est trié (il n'a qu'un seul élément) et $T[1]$ est inférieur ou égal à tous les autres éléments car c'est le minimum du tableau.

— **Hérédité :** On suppose $\mathcal{P}(k)$ vraie pour un $k \in [1, n-1]$. Au prochain passage dans la boucle, la variable `debut` vaut $k+1$. On parcourt le tableau de *debut* à $n$ en recherchant le minimum, et on l'échange avec $T[debut]$. Ainsi, $T[k+1]$ contient le plus petit élément du tableau entre $k+1$ et $n$.

Comme $\mathcal{P}(k)$ est vraie, le tableau de 1 à $k$ est trié et $T[k]$ était inférieur au minimum du tableau de $k+1$ à $n$. Donc $T[k] \leq T[k+1]$ et le tableau de 1 à $k+1$ est trié. $\mathcal{P}(k+1)$ est donc vraie.

— **Conclusion :** La propriété étant vraie pour $k = 1$, héréditaire pour $k \in [1, n-1]$, elle est donc vraie pour tout $k \in [1, n-1]$. Ainsi, $\mathcal{P}(n-1)$ est vraie et le tableau est trié de 1 à $n-1$. Mais comme $T[n-1] \leq T[n]$, il est aussi trié de 1 à $n$.

### Complexité

La procédure `Echanger` est de complexité $\Theta(1)$.

**Boucle intérieure :** $i$ prend les valeurs de la suite récurrente $(u_k)$ définie par :

$$\begin{cases} u_0 = debut + 1 \\ u_{k+1} = u_k + 1 \text{ pour } k \in \mathbb{N} \end{cases}$$

Son écriture fonctionnelle est $u_k = (debut + 1) + k$. La condition d'arrêt étant $u_k > n$, cela implique que $k > n - debut - 1$. Ainsi il y aura $n - debut$ passages dans la boucle intérieure.

Le coût de la boucle intérieure est en $\Theta(1)$, donc la complexité de la boucle intérieure est :

$$C_{int}(n, debut) = \sum_{i=debut+1}^{n} \Theta(1) = (n - debut)\Theta(1) = \Theta(n - debut)$$

**Boucle principale :** *debut* prend les valeurs de 1 à $n-1$, il y aura donc $n-1$ passages dans la boucle. Ainsi la complexité de l'algorithme sera :

$$C(n) = \sum_{debut=1}^{n-1} 3 + \Theta(1) + \Theta(n - debut) = 3(n-1) + \Theta(n-1) + \Theta\left(\sum_{debut=1}^{n-1} (n - debut)\right)$$

Or $\sum_{debut=1}^{n-1} (n - debut) = \sum_{j=1}^{n-1} j = \frac{n(n-1)}{2}$ donc :

$$C(n) = 3(n-1) + \Theta(n-1) + \Theta(n^2) = \Theta(n^2)$$

## Tri à bulles

### Principe

On parcourt le tableau à plusieurs reprises, et à chaque passage, on fait remonter par échanges successifs l'élément le plus grand.

> **i** Exemple visuel
>
> Soit le tableau initial : `[6, 4, 8, 1, 7, 2, 5, 3]`
> **Premier passage (faire remonter le 8) :**
>
> 1. Comparer 6 et 4 : pas d'échange → `[6, 4, 8, 1, 7, 2, 5, 3]`
> 2. Comparer 6 et 8 : pas d'échange → `[6, 4, 8, 1, 7, 2, 5, 3]`
> 3. Comparer 8 et 1 : échange → `[6, 4, 1, 8, 7, 2, 5, 3]`
> 4. Comparer 8 et 7 : échange → `[6, 4, 1, 7, 8, 2, 5, 3]`
> 5. Comparer 8 et 2 : échange → `[6, 4, 1, 7, 2, 8, 5, 3]`
> 6. Comparer 8 et 5 : échange → `[6, 4, 1, 7, 2, 5, 8, 3]`
> 7. Comparer 8 et 3 : échange → `[6, 4, 1, 7, 2, 5, 3, 8]`
>
> Le plus grand élément (8) est maintenant à sa place finale !

Pseudo-code

```
ALGORITHME TriABulles(T):
DONNEES
  T : tableau d'entiers de taille n
VARIABLES
  i, fin : entiers
DEBUT
  fin ← n
  TQ fin > 1 FAIRE
    i ← 1
    TQ i < fin FAIRE
      SI T[i] > T[i+1] ALORS
        Echanger(T, i, i+1)
      FSI
      i ← i+1
    FTQ
    fin ← fin - 1
  FTQ
FIN
```

### Arrêt

De la même façon que pour le tri sélection, la boucle intérieure est basée sur une variable $i$ qui est incrémentée de 1 à chaque tour, donc qui dépassera nécessairement toute valeur `fin` fixée.

Et pour la boucle principale, elle est basée sur la variable `fin` qui est décrémentée de 1 donc passera en dessous de la valeur 1 à un moment. Les deux boucles s'arrêtent donc.

### Validité

Nous allons procéder en deux étapes.

### Première étape : Boucle intérieure

Posons la propriété $\mathcal{P}'(k)$ suivante pour l'entier $k \in [1, fin - 1]$ :

$\mathcal{P}'(k)$ : "À la fin de l'itération $k, T[k{+}1]$ est supérieur ou égal à tous les éléments du tableau entre 1 et $k$"

— **Initialisation :** Lors du premier passage dans la boucle intérieure, si $T[1] > T[2]$ alors on les échange. Donc à la fin de l'itération 1, $T[2] \geq T[1]$ donc $\mathcal{P}'(1)$ est vraie.

— **Hérédité :** On suppose $\mathcal{P}'(k)$ vraie pour un $k \in [1, fin-1]$. Au prochain passage dans la boucle, la variable `i` vaut $k+1$.

Deux possibilités :

— Soit $T[k+1] \leq T[k+2]$, dans ce cas, d'après l'hypothèse de récurrence, $T[k+2]$ est aussi supérieur ou égal à tous les éléments du tableau entre 1 et $k$, donc entre 1 et $k+1$, ce qui vérifie $\mathcal{P}'(k+1)$.
— Soit $T[k+1] > T[k+2]$, dans ce cas, ils sont échangés et ainsi on se retrouve dans le premier cas. $\mathcal{P}'(k+1)$ est donc vraie.

— **Conclusion :** La propriété étant vraie pour $k=1$, héréditaire pour $k \in [1, fin-1]$, elle est donc vraie pour tout $k \in [1, fin-1]$.

### Deuxième étape : Boucle principale

Posons la propriété $\mathcal{P}(k)$ suivante pour l'entier $k \in [1, n-1]$ :

$\mathcal{P}(k)$ : "À la fin de l'itération $k$, le tableau de $n-k+1$ à $n$ est trié dans l'ordre croissant et

$T[n-k+1]$ est supérieur à tous les éléments du tableau de 1 à $n-k$"

— **Initialisation :** Lors du premier passage dans la boucle intérieure, la variable $fin = n$ et le passage dans la boucle intérieure valide $\mathcal{P}'(n-1)$, ce qui veut dire que $T[n]$ est le plus grand élément du tableau $T$ à la sortie de la boucle intérieure. Donc $\mathcal{P}(1)$ est vraie.
— **Hérédité :** On suppose $\mathcal{P}(k)$ vraie pour un $k \in [1, n-1]$. Au prochain passage dans la boucle, la variable `fin` vaut $n-k$. Ainsi, le passage dans la boucle intérieure valide $\mathcal{P}'(n-k-1)$, donc $T[n-k]$ est supérieur à tous les éléments du tableau entre 1 et $n-k-1$, de plus avec l'hypothèse de récurrence, le tableau $T$ est trié entre $n-k+1$ et $n$, il est donc aussi trié entre $n-k$ et $n$. $\mathcal{P}(k+1)$ est donc vraie.
— **Conclusion :** Comme $\mathcal{P}(n-1)$ est vraie, alors le tableau $T$ est rangé dans l'ordre croissant de 2 à $n$ et $T[2] \geq T[1]$ donc $T$ est totalement ordonné.

### Complexité

La procédure `Echanger` est de complexité $\Theta(1)$.

**Boucle intérieure :** $i$ prend toutes les valeurs entre 1 et $fin-1$. Ainsi il y aura $fin-1$ passages dans la boucle intérieure.

Le coût de la boucle intérieure est en $\Theta(1)$, donc la complexité de la boucle intérieure est :

$$C_{int}(fin) = \sum_{i=1}^{fin-1} \Theta(1) = (fin-1)\Theta(1) = \Theta(fin)$$

**Boucle principale :** $fin$ prend les valeurs de $n$ à 2, il y aura donc $n-1$ passages dans la boucle. Ainsi la complexité de l'algorithme sera :

$$C(n) = \sum_{fin=2}^{n} (2 + \Theta(fin)) = 2(n-1) + \Theta\left(\sum_{fin=2}^{n} fin\right)$$

Or $\sum_{fin=2}^{n} fin = \frac{n(n+1)}{2} - 1$ donc :

$$C(n) = 2(n-1) + \Theta(n^2) = \Theta(n^2)$$

> **Exercice : Amélioration**
>
> Améliorer l'algorithme en remarquant que si aucun échange n'est effectué, alors le tableau est trié.
> Calculer la complexité dans le meilleur des cas.

### Comparaison des tris

| Algorithme | Complexité pire cas | Complexité meilleur cas | En place | Stable |
|---|---|---|---|---|
| Tri sélection | $\Theta(n^2)$ | $\Theta(n^2)$ | Oui | Non |
| Tri à bulles | $\Theta(n^2)$ | $\Theta(n^2)$ | Oui | Oui |
| Tri à bulles optimisé | $\Theta(n^2)$ | $\Theta(n)$ | Oui | Oui |