
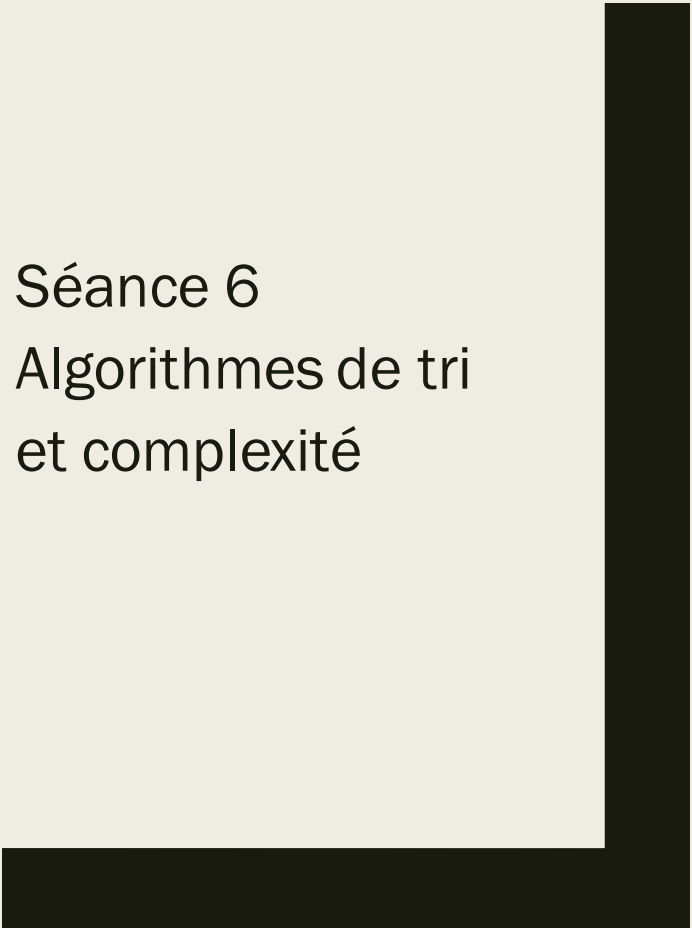




# ALGORITHMIQUE



Séance 6  
Algorithmes de tri  
et complexité



Camille SIMON – La Manu Le Havre

# Algorithmes de tris

- Permettent de trier des éléments dans un tableau ou une liste
- Grande diversité d'algorithmes
- Classés selon leur rapidité
- Deux exemples traités dans ce cours : le tri par insertion et le tri à bulles
  - A connaître et à savoir réécrire

# Tri par insertion

Sous-algorithme TriParInsertion

Paramètres d'entrée :

tableau[] : tableau d'entiers

Paramètres de sortie :

tableau[] : tableau d'entiers

Variables :

n, x, j : entiers

Instructions :

n ← Longueur(tableau)

Pour i de 1 à n - 1

    x ← tableau(i)

    j ← i

    Tant que j > 0 ET tableau(j - 1) > x

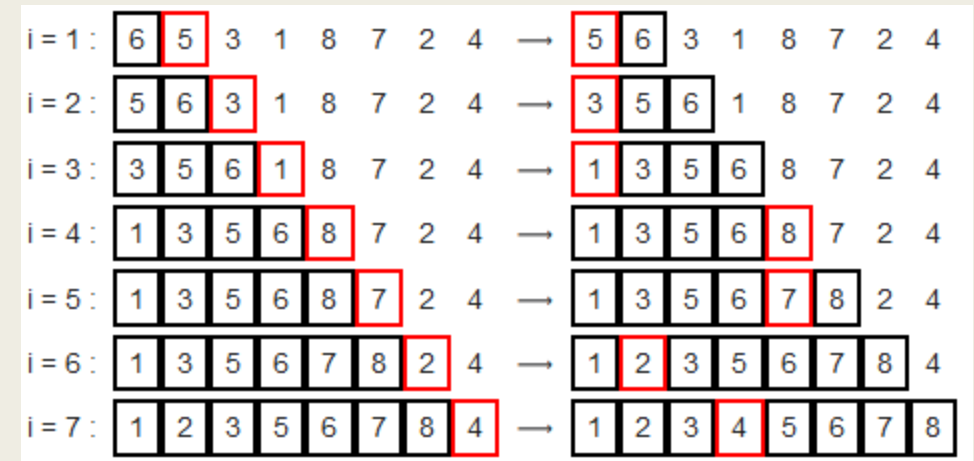
        tableau(j) ← tableau(j - 1)

        j ← j - 1

    FinTQ

    tableau(j) ← x

FinPour



# Tri à bulles

Sous-algorithme TriABulles

Paramètres d'entrée :

tableau[] : tableau d'entiers

Paramètres de sortie :

tableau[] : tableau d'entiers

Variables :

n, x : entiers

Instructions :

n ← Longueur(tableau)

Pour i de 0 à n - 1

Pour j de 0 à i - 1

Si tableau(j) > tableau(j + 1) alors

x ← tableau(j + 1)

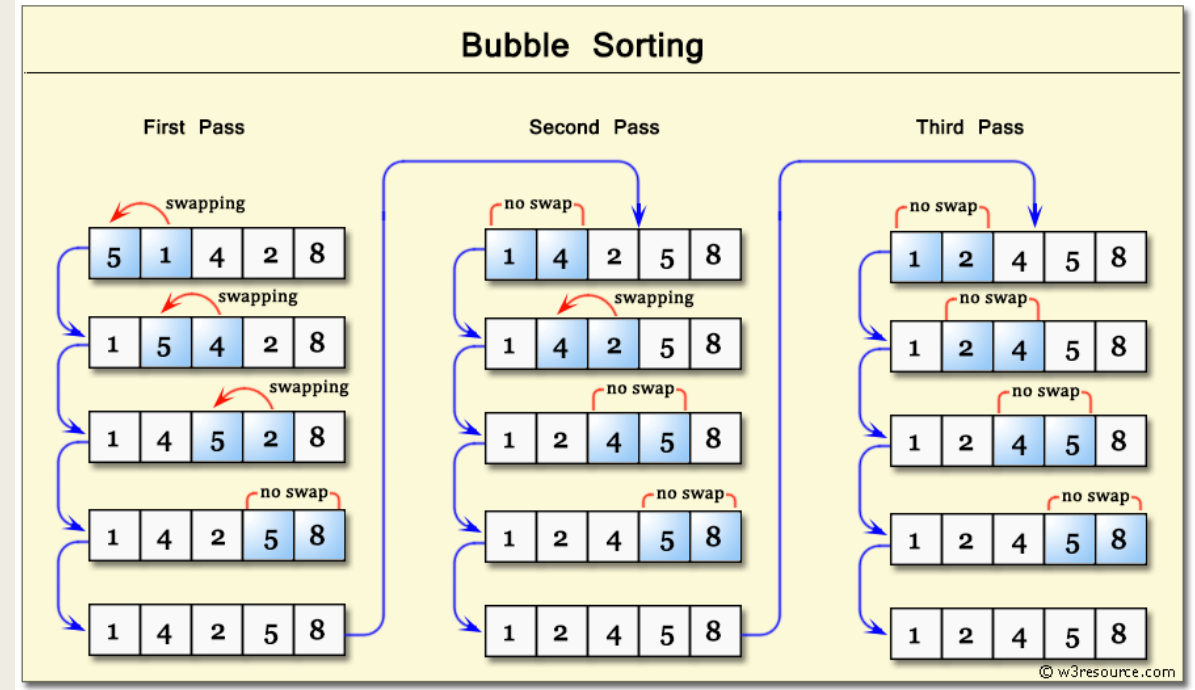
tableau(j + 1) ← tableau(j)

tableau(j) ← x

FinSi

FinPour

FinPour



# Exercice - Algorithmes de tris

## Exercice 1 :

Ecrire une version alternative du tri à bulles qui s'arrête lorsque le tableau est trié.

## Exercice 2 :

Ecrire un sous-algorithme qui prend en entrée deux tableaux **triés** et retourne un tableau trié qui contient les éléments des deux tableaux.



LEQUEL DES ALGORITHMES DE  
TRI EST LE PLUS RAPIDE ?

# Aparté processeur

- Lorsque l'on parle d'un processeur (*CPU*), on cite souvent sa cadence :

- Exemple : Intel Core i9 10850K 3,60 GHz

AMD Ryzen 9 3900X 3,80 Ghz

- A quoi correspond la cadence ?

# Complexité

- La complexité d'un algorithme correspond à l'**ordre de grandeur** de son nombre d'**opérations** dans le **pire des cas**.
- Une opération c'est :
  - Une affectation
  - Une lecture de donnée
  - Une comparaison
- Le nombre d'opération, donc la rapidité de l'algorithme, dépend de la taille du problème. C'est-à-dire de la quantité de données transmises à l'algorithme. On nomme cette quantité  $n$



# Complexité

Sous-algorithme TriABulles

Paramètres d'entrée :

tableau[] : tableau d'entiers

Paramètres de sortie :

tableau[] : tableau d'entiers

Variables :

n, x : entiers

Instructions :

n ← Longueur(tableau) ← 1 opération

Pour i de 0 à n - 1

Pour j de 0 à i - 1

Si tableau(j) > tableau(j + 1) alors ← 1 opération

x ← tableau(j + 1) ← 1 opération

tableau(j + 1) ← tableau(i) ← 1 opération

tableau(i) ← tableau(j + 1) ← 1 opération

FinSi

FinPour

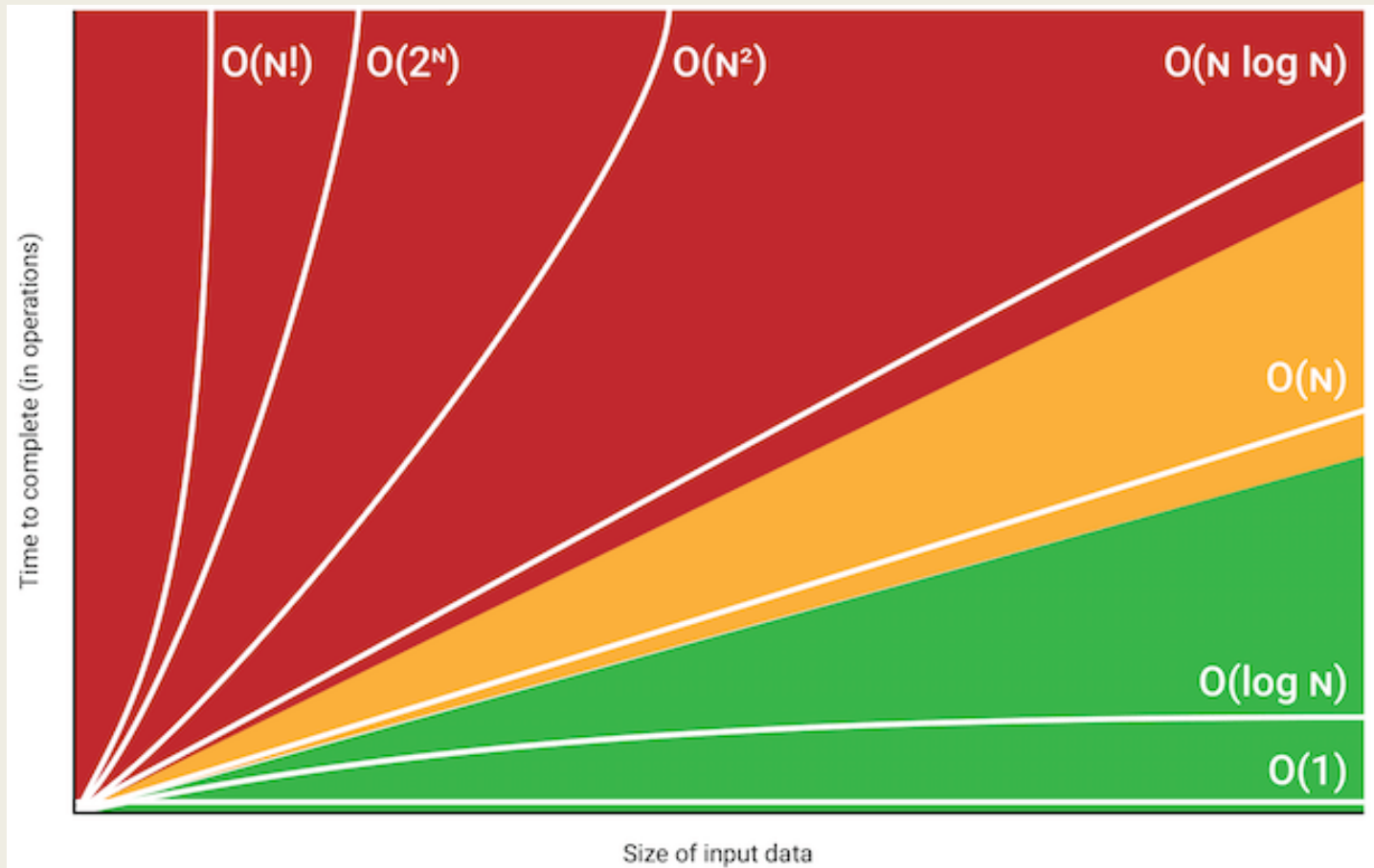
FinPour

n opérations

n opérations

Nombre d'opération :

$$1 + n * n * 4 = 4n^2 + 1$$



## Complexité

- Dans l'exemple précédent, on obtient  $4n^2 + 1$ , l'ordre de grandeur de cette expression est  $o(n^2)$ . Il s'agit de l'élément de plus hauts degrés de l'expression. On le note entre  $o()$  pour signifier qu'il s'agit de l'ordre de grandeur.
- Exemples :
  - $3n^4 + 8 \rightarrow o(n^4)$
  - $5 \rightarrow o(1)$
  - $3n \rightarrow o(n)$

# Exercice - Complexité

## Exercice 3 :

Prenez la correction des exercices sur les tableaux ([disponible sur Git](#)) et calculez leurs complexités.

## Exercice 4 :

Calculer la complexité des deux tris présentés sur les pages 3 et 4 ainsi que celui réalisé pour l'exercice 1.