

# ALGORITHMES DE TRI

**OBJECTIF :** L'objectif de ce tp est de rendre l'élève capable :

- d'écrire en langage **Python** les algorithmes de tris vu en cours
- comparer les vitesses de convergence

## Table des matières

<b>1 Stratégies par parcours des données</b>	<b>1</b>
1.1 Tri par insertion . . . . .	1
1.1.1 Principe . . . . .	1
1.1.2 Algorithme . . . . .	2
1.2 Tri par propagation (tri bulle) . . . . .	2
1.2.1 Principe . . . . .	2
1.2.2 Algorithme . . . . .	2
1.3 Tri par sélection . . . . .	3
1.3.1 Principe . . . . .	3
1.3.2 Algorithme . . . . .	3
<b>2 Stratégies « diviser pour mieux régner »</b>	<b>3</b>
2.1 Tri rapide (Quick Sort) . . . . .	3
2.1.1 Principe . . . . .	3
2.1.2 Algorithme . . . . .	3
2.2 Tri fusion . . . . .	4
2.2.1 Principe . . . . .	4
2.2.2 Algorithmes . . . . .	4

## 1 Stratégies par parcours des données

### 1.1 Tri par insertion

#### 1.1.1 Principe

**PRINCIPE :**

*Étant donné un ensemble d'éléments ordonnables, le tri par insertion :*

- sélectionne le premier élément de l'ensemble et le place dans une nouvelle structure.
- sélectionne le deuxième élément, le compare au premier élément sélectionné puis le place avant ou après la structure.
- le  $i$ ème élément étant sélectionné, on parcourt la structure, triée, à  $(i-1)$  éléments, soit par ordre croissant, soit par ordre décroissant pour le ranger à sa place.

## 1.1.2 Algorithme

### Algorithm 1 Tri insertion

**entrée:** T un tableau de valeurs ordonnables  
**résultat:** T trié par ordre croissant  
tri\_insertion(T):

- 1: **pour** i de 0 à taille(T) - 1 **faire**
- 2:   x ← T[i]
- 3:   j ← i
- 4:   **tant que** j > 0 et T[j-1] > x **faire**
- 5:     T[j] ← T[j-1]
- 6:     j ← j - 1
- 7:   **fin tant que**
- 8:   T[j] ← x
- 9: **fin pour**

**renvoi:** T

## 1.2 Tri par propagation (tri bulle)

### 1.2.1 Principe

#### PRINCIPE :

Étant donné un tableau T de valeurs ordonnables, le tri bulle consiste :

- à parcourir le tableau dans le sens des indices croissants et pour chaque indice j, si la valeur d'indice j + 1 est plus petite que la valeur d'indice j, les deux valeurs sont permutées
- à parcourir le tableau dans le sens des indices décroissants et pour chaque indice j, si la valeur d'indice j-1 est plus grande que la valeur d'indice j, les deux valeurs sont permutées
- à limiter l'étude du tableau au tableau T[i-1 : -i] pour le ième parcours.
- à poursuivre jusqu'au statu quo ou quand la taille du tableau à étudier est inférieure à 2

### 1.2.2 Algorithme

#### Algorithm 2 Tri par propagation (tri bulle)

**entrée:** T un tableau de valeurs ordonnables  
**résultat:** T trié par ordre croissant  
tri\_bulle(T):

- 1: imin, imax ← 0, taille(T) - 1
- 2: **tant que** imin < imax **faire**
- 3:   **pour** i de imin à imax-1 **faire**
- 4:     **si** T[i] > T[i+1] **alors**
- 5:       T[i], T[i+1] ← T[i+1], T[i]
- 6:     **fin si**
- 7:   **fin pour**
- 8:   imax ← imax - 1
- 9:   **pour** i de imax à imin + 1 avec un pas de -1 **faire**
- 10:     **si** T[i] < T[i-1] **alors**
- 11:       T[i], T[i-1] ← T[i-1], T[i]
- 12:     **fin si**
- 13:   **fin pour**
- 14:   imin ← imin + 1
- 15: **fin tant que**
- 16: **renvoi:** T

## 1.3 Tri par sélection

### 1.3.2 Algorithme

#### 1.3.1 Principe

##### PRINCIPE :

Étant donné un tableau  $T$  de valeurs ordonnables, le tri par sélection consiste :

- à déterminer l'élément le plus petit du tableau et l'échanger de place avec le premier élément du tableau
- à recommencer à l'itération  $i+1$  en omettant les  $i$  premières valeurs du tableau

#### Algorithm 3 Tri sélection

**entrée:**  $T$  un tableau de valeurs ordonnables  
**résultat:**  $T$  trié par ordre croissant  
 $\text{tri\_selection}(T)$ :

```
1: deb, n ← 0, taille(T)
2: tant que deb < n-1 faire
3:   imin, vmin ← deb, T[deb]
4:   pour j de deb+1 à n-1 faire
5:     si vmin > T[j] alors
6:       imin, vmin ← j, T[j]
7:   fin si
8:   fin pour
9:   T[imin], T[deb] ← T[deb], T[imin]
10:  deb ← deb+1
11: fin tant que
12: renvoi: T
```

## 2 Stratégies « diviser pour mieux régner »

### 2.1 Tri rapide (Quick Sort)

#### 2.1.2 Algorithme

#### 2.1.1 Principe

##### PRINCIPE :

Étant donné un tableau  $T$  de valeurs ordonnables, le tri rapide consiste :

- à prendre au hasard un élément du tableau qu'on appelle pivot
- à créer une liste avec toutes les valeurs de  $T$  plus petites que le pivot et une autre liste avec toutes les valeurs de  $T$  plus grande que le pivot
- à assembler à gauche les valeurs plus petites que le pivot triées par tri rapide, le pivot au centre et les valeurs plus grandes que le pivot triées par tri rapide à droite
- à retourner le tableau si celui-ci est vide ou n'a qu'un élément

C'est l'algorithme de tri le plus utilisé et peut être même l'algorithme le plus utilisé dans le monde.

#### Algorithm 4 Tri rapide

**entrée:**  $T$  un tableau de valeurs ordonnables  
**résultat:**  $T$  trié par ordre croissant  
 $\text{tri\_rapide}(T)$ :

```
1: si taille(T) < 2 alors
2:   renvoi: T
3: fin si
4: petits, grands ← [], []
5: pivot ← T.pop()
6: pour i de 1 à taille(T) faire
7:   x ← T.pop()
8:   si x ≤ pivot alors
9:     petits.append(x)
10:  sinon
11:    grands.append(x)
12:  fin si
13: fin pour
14: tri_rapide(petits)
15: tri_rapide(grands)
16: T += petits + [pivot] + grands
17: renvoi: T
```

## 2.2 Tri fusion

### 2.2.1 Principe

#### PRINCIPE :

Étant donné un tableau  $T$  de valeurs ordonnables, le tri fusion consiste :

- à couper le tableau en deux parts égales
- à trier par tri fusion chacune des deux parties
- à fusionner les deux parties
- à retourner directement la partie si elle ne comprend qu'un élément

### 2.2.2 Algorithmes

---

#### Algorithm 5 Tri fusion

---

**entrée:**  $T$  un tableau de valeurs ordonnables  
**résultat:**  $T$  trié par ordre croissant  
 $\text{tri\_fusion}(T)$ :  
1: **si**  $\text{taille}(T) == 1$  **alors**  
2:     **renvoi:**  $T$   
3: **fin si**  
4:  $\text{mil} \leftarrow \text{taille}(T) // 2$   
5:  $\text{gauche} \leftarrow \text{tri\_fusion}(T[:\text{mil}])$   
6:  $\text{droite} \leftarrow \text{tri\_fusion}(T[\text{mil}:])$   
7: **renvoi:**  $\text{fusion}(\text{gauche}, \text{droite})$

---

---

#### Algorithm 6 Fusion

---

**entrée:**  $P$  et  $Q$  deux tableaux valeurs triés  
**résultat:**  $T$  la fusion triée des tableaux  $P$  et  $Q$   
 $\text{fusion}(P, Q)$ :  
1:  $T, \text{pi}, \text{qi} \leftarrow [], 0, 0$   
2:  $\text{np}, \text{nq} \leftarrow \text{taille}(P), \text{taille}(Q)$   
3: **tant que**  $\text{np} > \text{pi}$  et  $\text{nq} > \text{qi}$  **faire**  
4:     **si**  $P[\text{pi}] < Q[\text{qi}]$  **alors**  
5:          $T.\text{append}(P[\text{pi}])$   
6:          $\text{pi} \leftarrow \text{pi} + 1$   
7:     **sinon**  
8:          $T.\text{append}(Q[\text{qi}])$   
9:          $\text{qi} \leftarrow \text{qi} + 1$   
10:     **fin si**  
11: **fin tant que**  
12: **si**  $\text{np} > \text{pi}$  **alors**  
13:      $T \leftarrow T + P[\text{pi}:]$   
14: **sinon**  
15:      $T \leftarrow T + Q[\text{qi}:]$   
16: **fin si**  
17: **renvoi:**  $T$

---