

# Tris

Christophe Viroulaud

Première - NSI

**Algo 04**

## Algorithmes de tris

Recherche

Tri par sélection

Tri par insertion

## Implémentation

Rappel : Passer un tableau  
à une fonction

Implémentations des tris

## Études des implémentations

Terminaison

Correction

Complexité



FIGURE 1 – Trier un jeu de cartes est un problème informatique.

## Algorithmes de tris

- Recherche
- Tri par sélection
- Tri par insertion

## Implémentation

- Rappel : Passer un tableau à une fonction
- Implémentations des tris

## Études des implémentations

- Terminaison
- Correction
- Complexité

## Algorithmes de tris

Recherche

Tri par sélection

Tri par insertion

## Implémentation

Rappel : Passer un tableau  
à une fonction

Implémentations des tris

## Études des implémentations

Terminaison

Correction

Complexité

Déterminer plusieurs méthodes de tris de données.

## 1. Algorithmes de tris

### 1.1 Recherche

### 1.2 Tri par sélection

### 1.3 Tri par insertion

## 2. Implémentation

## 3. Études des implémentations

### Algorithmes de tris

Recherche

Tri par sélection

Tri par insertion

### Implémentation

Rappel : Passer un tableau  
à une fonction

Implémentations des tris

### Études des implémentations

Terminaison

Correction

Complexité



## Algorithmes de tris

### Recherche

Tri par sélection

Tri par insertion

### Implémentation

Rappel : Passer un tableau  
à une fonction

Implémentations des tris

### Études des implémentations

Terminaison

Correction

Complexité

## Activité 1 :

1. Prendre le paquet de cartes mélangées et les étaler sur la table.
2. Trier les cartes.
3. Formaliser la méthode utilisée sous forme d'un algorithme.

## 1. Algorithmes de tris

### 1.1 Recherche

### 1.2 Tri par sélection

### 1.3 Tri par insertion

## 2. Implémentation

## 3. Études des implémentations

### Algorithmes de tris

Recherche

**Tri par sélection**

Tri par insertion

### Implémentation

Rappel : Passer un tableau  
à une fonction

Implémentations des tris

### Études des implémentations

Terminaison

Correction

Complexité

## Algorithmes de tris

Recherche

**Tri par sélection**

Tri par insertion

## Implémentation

Rappel : Passer un tableau  
à une fonction

Implémentations des tris

## Études des implémentations

Terminaison

Correction

Complexité

- Pour chaque carte du tas :
  - Trouver la plus petite carte dans la partie non triée.
  - Échanger cette carte avec la première de la partie non triée.

## Algorithmes de tris

Recherche

**Tri par sélection**

Tri par insertion

## Implémentation

Rappel : Passer un tableau  
à une fonction

Implémentations des tris

## Études des implémentations

Terminaison

Correction

Complexité



|   |   |    |    |   |   |    |    |   |   |   |   |   |
|---|---|----|----|---|---|----|----|---|---|---|---|---|
| 6 | 5 | 10 | 12 | 4 | 2 | 11 | 13 | 1 | 3 | 7 | 9 | 8 |
|---|---|----|----|---|---|----|----|---|---|---|---|---|

FIGURE 2 – Modélisation



## Algorithmes de tris

Recherche

**Tri par sélection**

Tri par insertion

## Implémentation

Rappel : Passer un tableau  
à une fonction

Implémentations des tris

Études des  
implémentations

Terminaison

Correction

Complexité

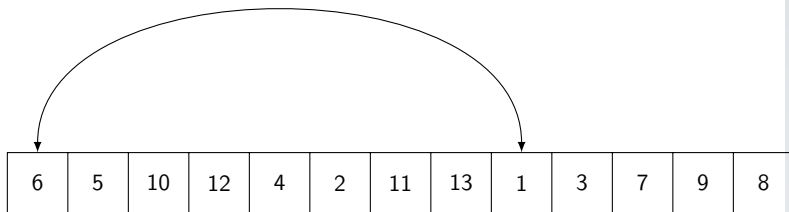


FIGURE 3 – Sélection du plus petit élément.

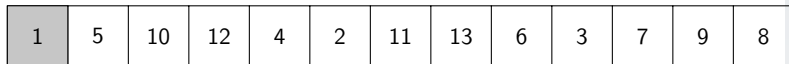


FIGURE 4 – La partie triée est à gauche.

## Algorithmes de tris

Recherche

**Tri par sélection**

Tri par insertion

## Implémentation

Rappel : Passer un tableau  
à une fonction

Implémentations des tris

Études des  
implémentations

Terminaison

Correction

Complexité

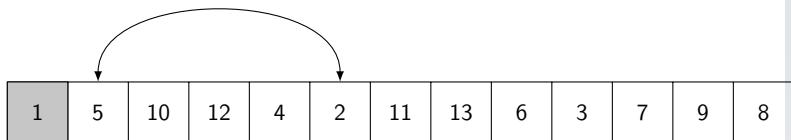


FIGURE 5 – Sélection du plus petit élément.

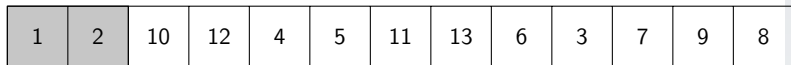


FIGURE 6 – La partie triée est à gauche.

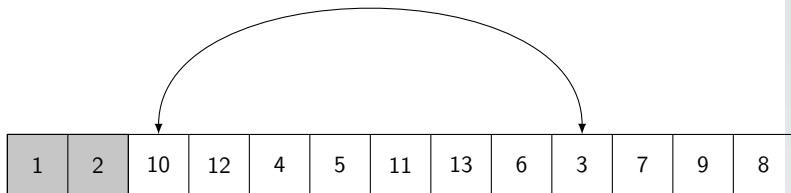


FIGURE 7 – Sélection du plus petit élément.

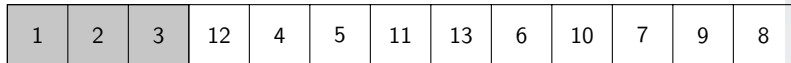


FIGURE 8 – La partie triée est à gauche.

## 1. Algorithmes de tris

### 1.1 Recherche

### 1.2 Tri par sélection

### 1.3 Tri par insertion

## 2. Implémentation

## 3. Études des implémentations

### Algorithmes de tris

Recherche

Tri par sélection

**Tri par insertion**

### Implémentation

Rappel : Passer un tableau  
à une fonction

Implémentations des tris

### Études des implémentations

Terminaison

Correction

Complexité

## Algorithmes de tris

Recherche

Tri par sélection

**Tri par insertion**

## Implémentation

Rappel : Passer un tableau  
à une fonction

Implémentations des tris

## Études des implémentations

Terminaison

Correction

Complexité

- ▶ Pour chaque carte du tas :
  - ▶ Tant que la carte précédente est plus petite
  - ▶ Échanger cette carte avec la carte en cours.



|   |   |    |    |   |   |    |    |   |   |   |   |   |
|---|---|----|----|---|---|----|----|---|---|---|---|---|
| 6 | 5 | 10 | 12 | 4 | 2 | 11 | 13 | 1 | 3 | 7 | 9 | 8 |
|---|---|----|----|---|---|----|----|---|---|---|---|---|

FIGURE 9 – Modélisation

## Algorithmes de tris

Recherche

Tri par sélection

Tri par insertion

## Implémentation

Rappel : Passer un tableau  
à une fonction

Implémentations des tris

## Études des implémentations

Terminaison

Correction

Complexité

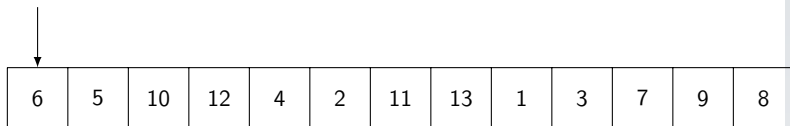


FIGURE 10 – Carte en cours

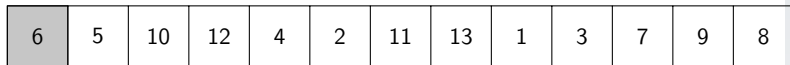


FIGURE 11 – La partie triée est à gauche.

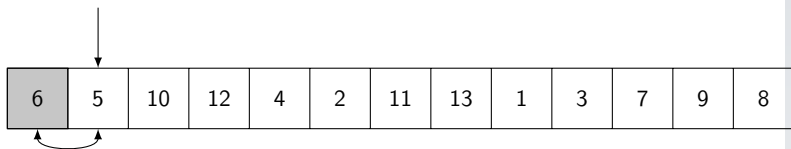


FIGURE 12 – Carte en cours

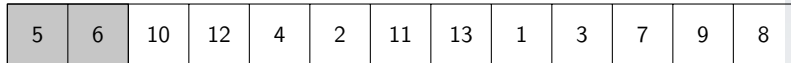


FIGURE 13 – La partie triée est à gauche.



## Algorithmes de tris

Recherche

Tri par sélection

**Tri par insertion**

## Implémentation

Rappel : Passer un tableau  
à une fonction

Implémentations des tris

Études des  
implémentations

Terminaison

Correction

Complexité

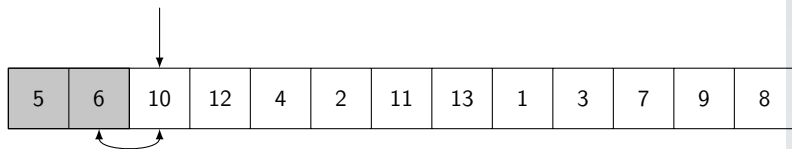


FIGURE 14 – Carte en cours

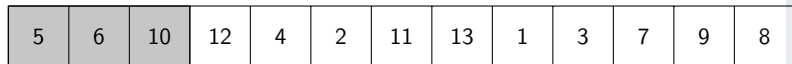


FIGURE 15 – La partie triée est à gauche.

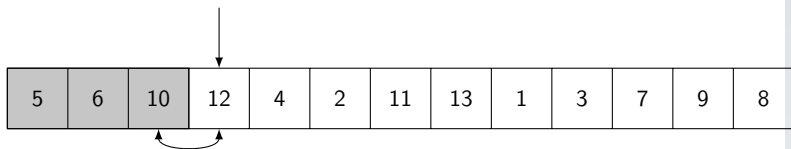


FIGURE 16 – Carte en cours

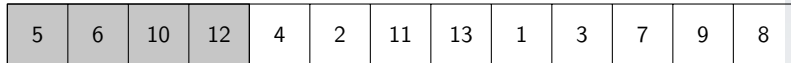


FIGURE 17 – La partie triée est à gauche.

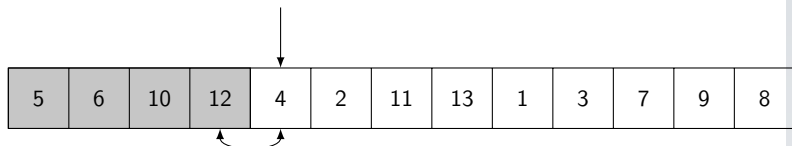


FIGURE 18 – Carte en cours

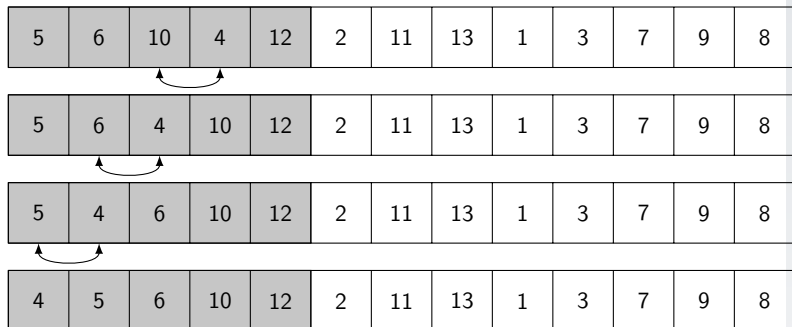


FIGURE 19 – La partie triée est à gauche.

Algorithmes de tris

Recherche

Tri par sélection

Tri par insertion

Implémentation

Rappel : Passer un tableau  
à une fonction

Implémentations des tris

Études des  
implémentations

Terminaison

Correction

Complexité

## 1. Algorithmes de tris

## 2. Implémentation

### 2.1 Rappel : Passer un tableau à une fonction

### 2.2 Implémentations des tris

## 3. Études des implémentations

### Algorithmes de tris

Recherche

Tri par sélection

Tri par insertion

### Implémentation

Rappel : Passer un tableau  
à une fonction

Implémentations des tris

### Études des implémentations

Terminaison

Correction

Complexité

## Algorithmes de tris

Recherche

Tri par sélection

Tri par insertion

## Implémentation

Rappel : Passer un tableau  
à une fonction

Implémentations des tris

Études des  
implémentations

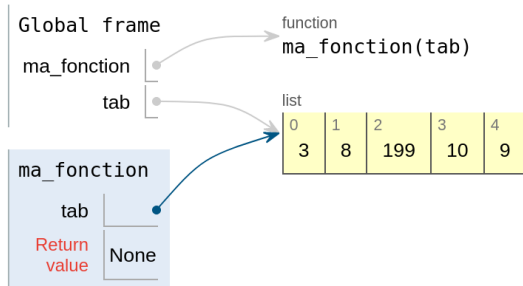
Terminaison

Correction

Complexité

```
1 def ma_fonction(tab: list) -> None:
2     tab[2] = 199
3
4     tab = [3, 8, 1, 10, 9]
5     ma_fonction(tab)
```

Code 1 – Quand on passe un tableau en argument à une fonction, on passe en réalité **une référence** au tableau original.



## 1. Algorithmes de tris

## 2. Implémentation

### 2.1 Rappel : Passer un tableau à une fonction

### 2.2 Implémentations des tris

## 3. Études des implémentations

### Algorithmes de tris

Recherche

Tri par sélection

Tri par insertion

### Implémentation

Rappel : Passer un tableau  
à une fonction

**Implémentations des tris**

### Études des implémentations

Terminaison

Correction

Complexité

# Implémentation des tris - Tri par sélection

- Pour chaque carte du tas :
  - **Trouver la plus petite** carte dans la partie non triée.
  - **Échanger** cette carte avec la première de la partie non triée.

## Activité 2 :

1. Écrire la fonction `indice_mini(tab: list, dep: int) → int` qui renvoie l'indice de la valeur minimale de `tab`, entre l'élément d'indice `deb` et la fin du tableau.
2. Écrire la fonction `echanger(tab: list, i: int, j: int) → None` qui échange les éléments d'indices `i` et `j`.
3. Écrire alors la fonction `tri_insertion(tab: list) → None`.

Algorithmes de tris

Recherche

Tri par sélection

Tri par insertion

Implémentation

Rappel : Passer un tableau  
à une fonction

Implémentations des tris

Études des  
implémentations

Terminaison

Correction

Complexité

## Algorithmes de tris

Recherche

Tri par sélection

Tri par insertion

## Implémentation

Rappel : Passer un tableau  
à une fonction

Implémentations des tris

## Études des implémentations

Terminaison

Correction

Complexité

```
1 def indice_mini(tab: list, dep: int) -> int:
2     i_mini = dep
3     mini = tab[dep]
4     # parcours de la partie du tableau
5     for i in range(dep, len(tab)):
6         if tab[i] < mini:
7             i_mini = i
8             mini = tab[i]
9     return i_mini
```



## Algorithmes de tris

Recherche

Tri par sélection

Tri par insertion

## Implémentation

Rappel : Passer un tableau  
à une fonction

**Implémentations des tris**

## Études des implémentations

Terminaison

Correction

Complexité

```
1 def echanger(tab: list, i: int, j: int) -> None:
2     temp = tab[i]
3     tab[i] = tab[j]
4     tab[j] = temp
```

## Algorithmes de tris

Recherche

Tri par sélection

Tri par insertion

## Implémentation

Rappel : Passer un tableau  
à une fonction

**Implémentations des tris**

## Études des implémentations

Terminaison

Correction

Complexité

```
1 def tri_selection(tab: list) -> None:
2     for i in range(len(tab)):
3         i_mini = indice_mini(tab, i)
4         echanger(tab, i, i_mini)
```

## Algorithmes de tris

Recherche

Tri par sélection

Tri par insertion

## Implémentation

Rappel : Passer un tableau  
à une fonction

Implémentations des tris

Études des  
implémentations

Terminaison

Correction

Complexité

**Activité 3 :**

1. Construire par compréhension un tableau de 10 éléments aléatoires compris entre 0 et 100.
2. Tester alors la fonction de tri.

## Algorithmes de tris

Recherche

Tri par sélection

Tri par insertion

## Implémentation

Rappel : Passer un tableau  
à une fonction

**Implémentations des tris**

## Études des implémentations

Terminaison

Correction

Complexité

```
1 t = [randint(0, 100) for _ in range(10)]
2 print(t)
3 tri_selection(t)
4 print(t)
```

- Pour chaque carte du tas :
  - Tant que la carte précédente est plus petite
  - **Échanger** cette carte avec la carte en cours.

## Activité 4 :

1. Écrire la fonction `insérer(tab: list, j: int)`  
→ `None` qui insère l'élément de rang `j` dans la partie déjà triée.
2. Écrire alors la fonction `tri_insertion(tab: list)` → `None`.

Algorithmes de tris

Recherche

Tri par sélection

Tri par insertion

Implémentation

Rappel : Passer un tableau  
à une fonction

Implémentations des tris

Études des  
implémentations

Terminaison

Correction

Complexité

## Algorithmes de tris

Recherche

Tri par sélection

Tri par insertion

## Implémentation

Rappel : Passer un tableau  
à une fonction

Implémentations des tris

## Études des implémentations

Terminaison

Correction

Complexité

```
1 def inserer(tab: list, j: int) -> None:
2     while j-1 >= 0 and tab[j-1] > tab[j]:
3         echanger(tab, j-1, j)
4         j = j-1
```

## Remarque

La condition  $j-1 \geq 0$  évite de *sortir* du tableau.

## Algorithmes de tris

Recherche

Tri par sélection

Tri par insertion

## Implémentation

Rappel : Passer un tableau  
à une fonction

Implémentations des tris

Études des  
implémentations

Terminaison

Correction

Complexité

```
1 def tri_insertion(tab: list) -> None:
2     for i in range(len(tab)):
3         inserer(tab, i)
```

## 1. Algorithmes de tris

## 2. Implémentation

## 3. Études des implémentations

### 3.1 Terminaison

### 3.2 Correction

### 3.3 Complexité

#### Algorithmes de tris

Recherche

Tri par sélection

Tri par insertion

#### Implémentation

Rappel : Passer un tableau  
à une fonction

Implémentations des tris

#### Études des implémentations

Terminaison

Correction

Complexité



## Algorithmes de tris

Recherche

Tri par sélection

Tri par insertion

## Implémentation

Rappel : Passer un tableau  
à une fonction

Implémentations des tris

## Études des implémentations

### Terminaison

Correction

Complexité

## À retenir

Pour montrer que l'algorithme termine (ne part pas dans une boucle sans fin), on utilise **un variant de boucle**.

# Tri par sélection

Pour le tri par sélection, on utilise deux boucles **bornées**. La terminaison est dans ce cas évidente : les deux boucles s'arrêteront obligatoirement.

## Algorithmes de tris

Recherche

Tri par sélection

Tri par insertion

## Implémentation

Rappel : Passer un tableau  
à une fonction

Implémentations des tris

## Études des implémentations

Terminaison

Correction

Complexité

# Tri par insertion

Pour le tri par insertion :

```
1 def inserer(tab: list, j: int) -> None:
2     while j-1 >= 0 and tab[j-1] > tab[j]:
3         echanger(tab, j-1, j)
4         j = j-1
```

Code 2 – La fonction `inserer` contient une boucle non bornée.

- La variable `j` est un variant de boucle.
- ligne 4 : À chaque tour, `j` est diminué de 1.
- ligne 1 : L'instruction `j-1 >= 0` assure que la boucle se termine.

Algorithmes de tris

Recherche

Tri par sélection

Tri par insertion

Implémentation

Rappel : Passer un tableau  
à une fonction

Implémentations des tris

Études des  
implémentations

Terminaison

Correction

Complexité

## 1. Algorithmes de tris

## 2. Implémentation

## 3. Études des implémentations

### 3.1 Terminaison

### 3.2 Correction

### 3.3 Complexité

#### Algorithmes de tris

Recherche

Tri par sélection

Tri par insertion

#### Implémentation

Rappel : Passer un tableau  
à une fonction

Implémentations des tris

#### Études des implémentations

Terminaison

**Correction**

Complexité

## Algorithmes de tris

Recherche

Tri par sélection

Tri par insertion

## Implémentation

Rappel : Passer un tableau  
à une fonction

Implémentations des tris

## Études des implémentations

Terminaison

Correction

Complexité

### À retenir

Pour montrer que l'algorithme est correct, on utilise **un invariant de boucle**. Un invariant est une expression qui reste vraie à chaque itération de boucle.

Pour le tri par sélection, l'invariant de boucle est :

Avant chaque itération de la boucle externe, la partie gauche du tableau est triée.

|   |   |    |    |   |   |    |    |   |   |   |   |   |
|---|---|----|----|---|---|----|----|---|---|---|---|---|
| 6 | 5 | 10 | 12 | 4 | 2 | 11 | 13 | 1 | 3 | 7 | 9 | 8 |
|---|---|----|----|---|---|----|----|---|---|---|---|---|

FIGURE 20 – Vraie avant la première itération.

## Algorithmes de tris

Recherche

Tri par sélection

Tri par insertion

## Implémentation

Rappel : Passer un tableau  
à une fonction

Implémentations des tris

Études des  
implémentations

Terminaison

Correction

Complexité

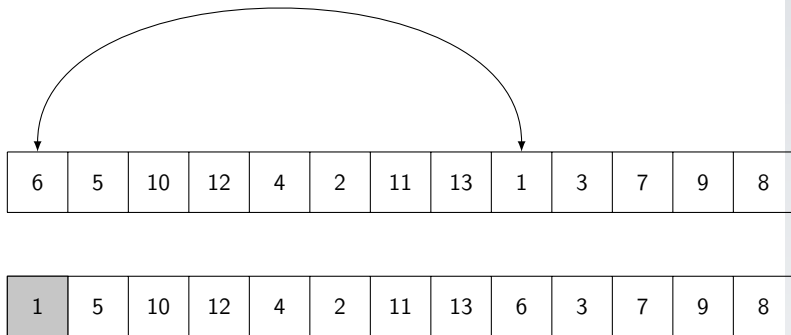


FIGURE 21 – Vraie avant la deuxième itération.

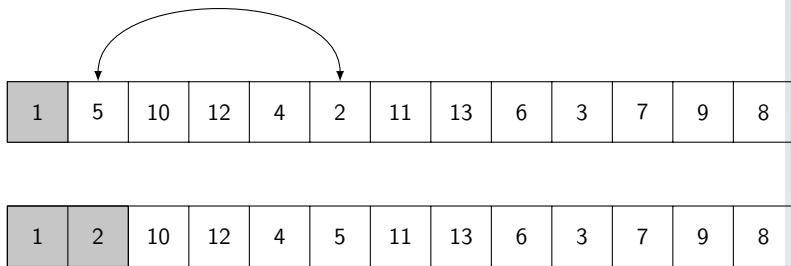


FIGURE 22 – Vraie avant la troisième itération.

## À retenir

On peut démontrer que la relation est vraie pour n'importe quelle itération.



Pour le tri par insertion, l'invariant de boucle est le même :

Avant chaque itération de la boucle externe, la partie gauche du tableau est triée.

|   |   |    |    |   |   |    |    |   |   |   |   |   |
|---|---|----|----|---|---|----|----|---|---|---|---|---|
| 6 | 5 | 10 | 12 | 4 | 2 | 11 | 13 | 1 | 3 | 7 | 9 | 8 |
|---|---|----|----|---|---|----|----|---|---|---|---|---|

FIGURE 23 – Vraie avant la première itération.

## Algorithmes de tris

Recherche

Tri par sélection

Tri par insertion

## Implémentation

Rappel : Passer un tableau  
à une fonction

Implémentations des tris

Études des  
implémentations

Terminaison

Correction

Complexité

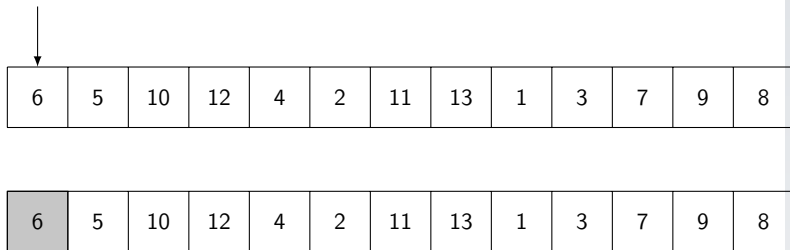


FIGURE 24 – Vraie avant la deuxième itération.

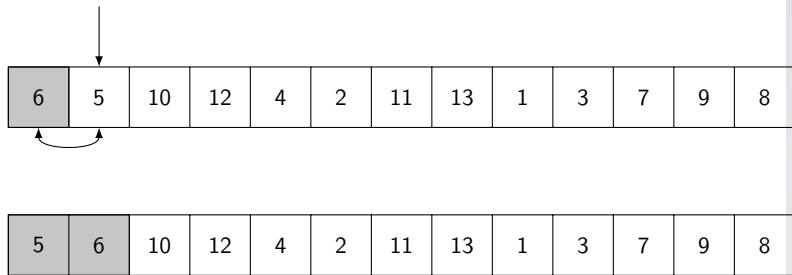


FIGURE 25 – Vraie avant la troisième itération.

## À retenir

On peut démontrer que la relation est vraie pour n'importe quelle itération.

Algorithmes de tris

Recherche

Tri par sélection

Tri par insertion

Implémentation

Rappel : Passer un tableau  
à une fonction

Implémentations des tris

Études des  
implémentations

Terminaison

Correction

Complexité

## 1. Algorithmes de tris

## 2. Implémentation

## 3. Études des implémentations

### 3.1 Terminaison

### 3.2 Correction

### 3.3 Complexité

#### Algorithmes de tris

Recherche

Tri par sélection

Tri par insertion

#### Implémentation

Rappel : Passer un tableau  
à une fonction

Implémentations des tris

#### Études des implémentations

Terminaison

Correction

**Complexité**

## Algorithmes de tris

Recherche

Tri par sélection

Tri par insertion

## Implémentation

Rappel : Passer un tableau  
à une fonction

Implémentations des tris

## Études des implémentations

Terminaison

Correction

Complexité

### À retenir

- ▶ La complexité étudie les performances d'un algorithme.
- ▶ Elles sont indépendantes de la puissance de la machine.
- ▶ On étudie le nombre d'opérations que doit effectuer l'algorithme.

```
1 def tri_selection(tab: list) -> None:
2     for i in range(len(tab)):
3         i_mini = indice_mini(tab, i)
4         echanger(tab, i, i_mini)
```

## Observation

On note  $n$  la taille du tableau.

La boucle (ligne 2) effectue  $n$  itérations.

## Algorithmes de tris

Recherche

Tri par sélection

Tri par insertion

## Implémentation

Rappel : Passer un tableau  
à une fonction

Implémentations des tris

Études des  
implémentations

Terminaison

Correction

Complexité

```
1 def tri_selection(tab: list) -> None:
2     for i in range(len(tab)):
3         i_mini = indice_mini(tab, i)
4         echanger(tab, i, i_mini)
```

```
1 def indice_mini(tab: list, dep: int) -> int:
2     i_mini = dep
3     mini = tab[dep]
4     for i in range(dep, len(tab)):
5         if tab[i] < mini:
6             i_mini = i
7             mini = tab[i]
8     return i_mini
```

## Observation

À chaque itération, la fonction de tri appelle la fonction `indice_mini`. Cette dernière effectue  $n - \text{dep}$  itérations.

## Algorithmes de tris

Recherche

Tri par sélection

Tri par insertion

## Implémentation

Rappel : Passer un tableau  
à une fonction

Implémentations des tris

Études des  
implémentations

Terminaison

Correction

Complexité

- ▶ à la première itération de  $i$ , la boucle de `indice_mini` effectue  $n-1$  itérations.

|   |   |   |   |   |
|---|---|---|---|---|
| 1 | 3 | 7 | 8 | 5 |
|---|---|---|---|---|

- ▶ à la deuxième itération de  $i$ , la boucle de `indice_mini` effectue  $n-2$  itérations.

|   |   |   |   |   |
|---|---|---|---|---|
| 1 | 3 | 7 | 8 | 5 |
|---|---|---|---|---|

- ▶ ...



## Algorithmes de tris

Recherche

Tri par sélection

Tri par insertion

## Implémentation

Rappel : Passer un tableau  
à une fonction

Implémentations des tris

Études des  
implémentations

Terminaison

Correction

Complexité

$$\sum_{k=1}^{n-1} k = (n-1) + (n-2) + \cdots + 2 + 1 = \frac{n \cdot (n-1)}{2}$$

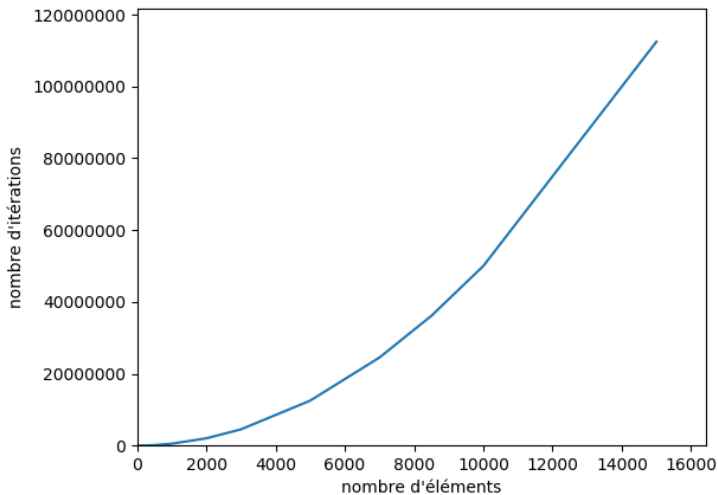
**À retenir**

Le tri par sélection effectue  $\frac{n \cdot (n-1)}{2}$  opérations pour ordonner le tableau.

Le nombre d'opérations dépend de  $n^2$ . On dit que la complexité est **quadratique**.

# Évolution du nombre d'itérations

Tris



## Algorithmes de tris

Recherche

Tri par sélection

Tri par insertion

## Implémentation

Rappel : Passer un tableau  
à une fonction

Implémentations des tris

## Études des implémentations

Terminaison

Correction

**Complexité**

## Algorithmes de tris

Recherche

Tri par sélection

Tri par insertion

## Implémentation

Rappel : Passer un tableau  
à une fonction

Implémentations des tris

Études des  
implémentations

Terminaison

Correction

Complexité

```
1 def tri_insertion(tab: list) -> None:
2     for i in range(len(tab)):
3         inserer(tab, i)
```

## Observation

On note  $n$  la taille du tableau.

La boucle (ligne 2) effectue  $n$  itérations.

## Algorithmes de tris

Recherche

Tri par sélection

Tri par insertion

## Implémentation

Rappel : Passer un tableau  
à une fonction

Implémentations des tris

Études des  
implémentations

Terminaison

Correction

Complexité

```
1 def tri_insertion(tab: list) -> None:
2     for i in range(len(tab)):
3         inserer(tab, i)
```

```
1 def inserer(tab: list, j: int) -> None:
2     while j-1 >= 0 and tab[j-1] > tab[j]:
3         echanger(tab, j-1, j)
4         j = j-1
```

## Observation

À chaque itération, la fonction de tri appelle la fonction **inserer**. Le nombre d'itérations de la boucle **while** peut varier.

```
1 def inserer(tab: list, j: int) -> None:  
2     while j-1 >= 0 and tab[j-1] > tab[j]:  
3         echanger(tab, j-1, j)  
4         j = j-1
```



FIGURE 26 – 3 itérations pour placer 2



FIGURE 27 – 0 itération pour placer 9

## Algorithmes de tris

Recherche

Tri par sélection

Tri par insertion

## Implémentation

Rappel : Passer un tableau  
à une fonction

Implémentations des tris

Études des  
implémentations

Terminaison

Correction

Complexité

## Algorithmes de tris

Recherche

Tri par sélection

Tri par insertion

## Implémentation

Rappel : Passer un tableau  
à une fonction

Implémentations des tris

Études des  
implémentations

Terminaison

Correction

Complexité

**Activité 5 :**

1. Compter le nombre d'itérations de la boucle **while** si le tableau est déjà trié.
2. Compter le nombre d'itérations de la boucle **while** si le tableau est trié dans l'ordre décroissant.

## Algorithmes de tris

Recherche

Tri par sélection

Tri par insertion

## Implémentation

Rappel : Passer un tableau  
à une fonction

Implémentations des tris

Études des  
implémentations

Terminaison

Correction

Complexité

```
1 def inserer(tab: list, j: int) -> None:
2     while j-1 >= 0 and tab[j-1] > tab[j]:
3         echanger(tab, j-1, j)
4         j = j-1
```

|   |   |   |   |   |
|---|---|---|---|---|
| 1 | 4 | 5 | 7 | 8 |
|---|---|---|---|---|

Code 3 – Le tableau est déjà trié. La boucle `while` n'effectue aucune itération.

## Algorithmes de tris

Recherche

Tri par sélection

Tri par insertion

## Implémentation

Rappel : Passer un tableau  
à une fonction

Implémentations des tris

Études des  
implémentations

Terminaison

Correction

Complexité

```
1 def inserer(tab: list, j: int) -> None:
2     while j-1 >= 0 and tab[j-1] > tab[j]:
3         echanger(tab, j-1, j)
4         j = j-1
```

|   |   |   |   |   |
|---|---|---|---|---|
| 8 | 7 | 5 | 4 | 1 |
|---|---|---|---|---|

Code 4 – Le tableau est inversé. La boucle `while` effectue `i` itérations.



## À retenir

- ▶ Dans le meilleur des cas (tableau déjà trié), la boucle imbriquée **while** effectue 0 itération. Dans ce cas particulier, le tri par insertion est en temps **linéaire**.

### Algorithmes de tris

Recherche

Tri par sélection

Tri par insertion

### Implémentation

Rappel : Passer un tableau à une fonction

Implémentations des tris

### Études des implémentations

Terminaison

Correction

Complexité

## À retenir

- ▶ Dans le meilleur des cas (tableau déjà trié), la boucle imbriquée **while** effectue 0 itération. Dans ce cas particulier, le tri par insertion est en temps **linéaire**.
- ▶ Dans le pire des cas (tableau inversé), la boucle imbriquée **while** effectue **n** itérations. Le tri par insertion est en temps **quadratique** ( $n^2$ ).

### Algorithmes de tris

Recherche

Tri par sélection

Tri par insertion

### Implémentation

Rappel : Passer un tableau à une fonction

Implémentations des tris

### Études des implémentations

Terminaison

Correction

Complexité

## À retenir

- ▶ Dans le meilleur des cas (tableau déjà trié), la boucle imbriquée **while** effectue 0 itération. Dans ce cas particulier, le tri par insertion est en temps **linéaire**.
- ▶ Dans le pire des cas (tableau inversé), la boucle imbriquée **while** effectue **n** itérations. Le tri par insertion est en temps **quadratique** ( $n^2$ ).
- ▶ En moyenne (tableau quelconque), la boucle imbriquée effectue **n** itérations. Le tri par insertion est en temps **quadratique** ( $n^2$ ).

### Algorithmes de tris

Recherche

Tri par sélection

Tri par insertion

### Implémentation

Rappel : Passer un tableau à une fonction

Implémentations des tris

### Études des implémentations

Terminaison

Correction

Complexité