

Trier des cartes

Christophe Viroulaud

Première - NSI

Algo 07



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

Coût



Figure 1 – Trier un jeu de cartes est un problème algorithmique.

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
- Coût

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

Coût

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

Coût



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.

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

Coût

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

Coût

Tri par sélection



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

Coût

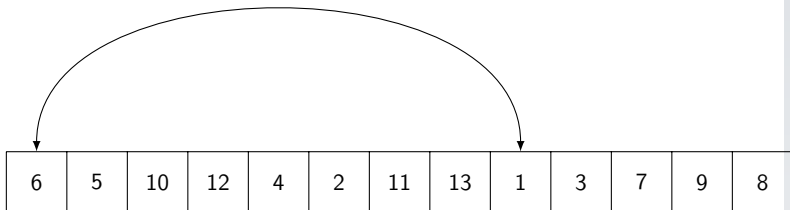


Figure 3 – Sélection du plus petit élément.

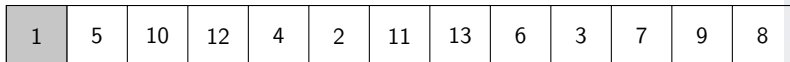


Figure 4 – La partie triée est à gauche.

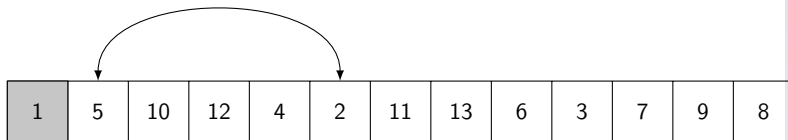


Figure 5 – Sélection du plus petit élément.

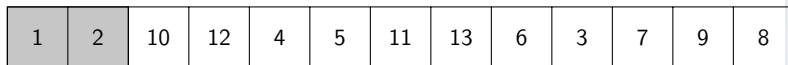


Figure 6 – La partie triée est à gauche.

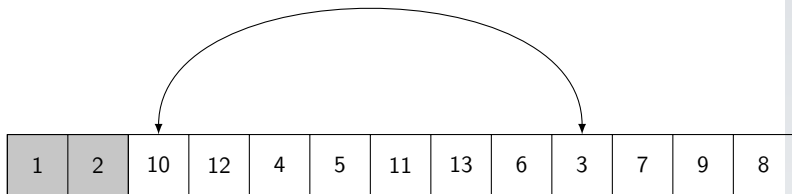


Figure 7 – Sélection du plus petit élément.

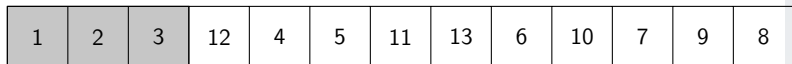


Figure 8 – La partie triée est à gauche.

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

Coût

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.

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

Coût

Tri par insertion

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

Coût



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

Figure 9 – Modélisation

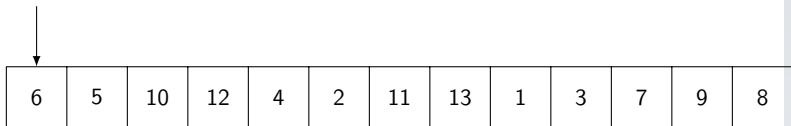


Figure 10 – Carte en cours

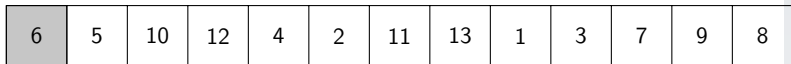


Figure 11 – La partie triée est à gauche.

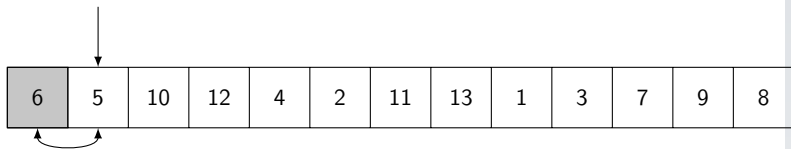


Figure 12 – Carte en cours

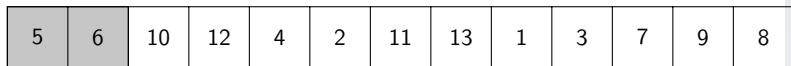


Figure 13 – La partie triée est à gauche.

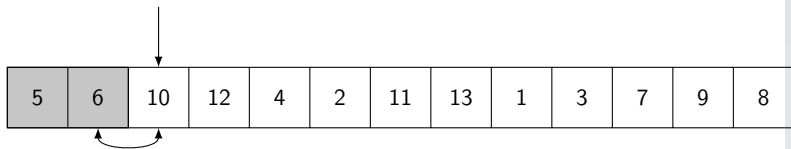


Figure 14 – Carte en cours

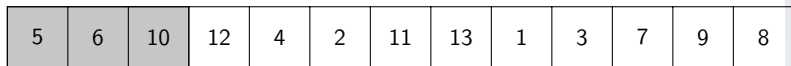


Figure 15 – La partie triée est à gauche.

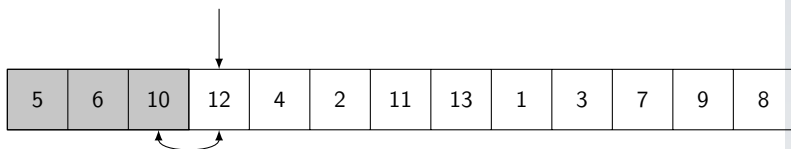


Figure 16 – Carte en cours

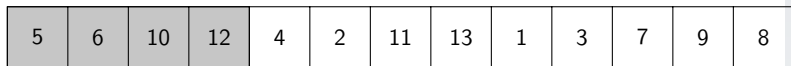


Figure 17 – La partie triée est à gauche.

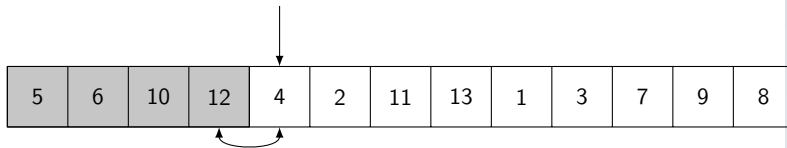


Figure 18 – Carte en cours

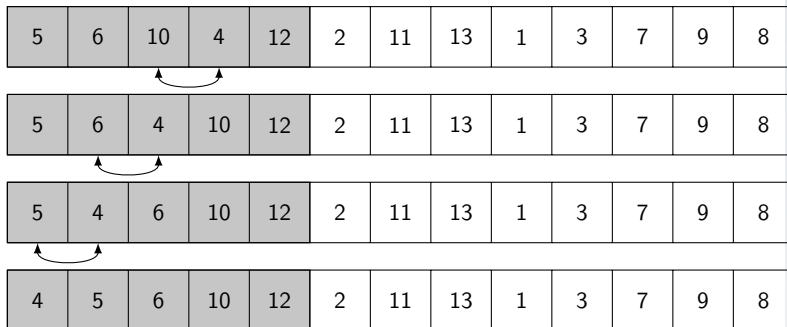


Figure 19 – La partie triée est à gauche.

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

Coût

Tri par sélection - version 1

- ▶ Pour chaque carte du tas :
 - ▶ **Insérer** la carte en cours dans la partie déjà 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

Coût

Tri par sélection - version 2

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

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

Coût

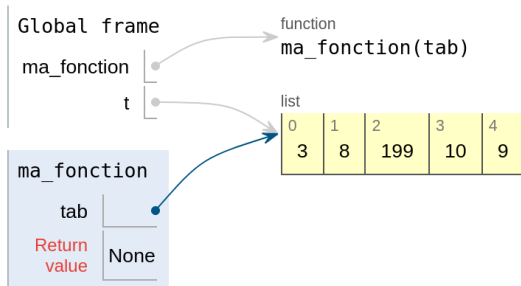
Implémentation - Rappel

```

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

```

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



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
- Coût

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

Coût

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 `dep` 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_selection(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

Coût

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

Coût

```
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

Coût

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

- ▶ 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.

```
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)
```

Code 2 – Tri par sélection

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

Coût

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

Coût

Activité 3 :

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

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

Code 3 – Programme principal

Remarque

La fonction `tri_selection` effectue un tri en place : le tableau initial est modifié.

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

Coût

Tri par insertion

- ▶ Pour chaque carte du tas :
 - ▶ **Insérer** la carte en cours dans la partie déjà triée.

Activité 4 :

1. Écrire la fonction `insérer(tab: list, j: int)`
→ `None` qui insère l'élément de rang `j` dans la partie déjà triée, en respectant l'algorithme :
 - ▶ Tant que la carte précédente est plus grande :
 - ▶ **Échanger** cette carte avec la carte en cours.
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

Coût

```
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

Coût

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

Coût

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

Code 4 – Tri par insertion

1. Algorithmes de tris

2. Implémentation

3. Études des implémentations

3.1 Terminaison

3.2 Correction

3.3 Coût

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

Coût

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

Coût

À retenir

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

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

Coût

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

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 5 – 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

Coût

1. Algorithmes de tris

2. Implémentation

3. Études des implémentations

3.1 Terminaison

3.2 Correction

3.3 Coût

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

Coût

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

Coût

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](#)[Coût](#)

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

Coût

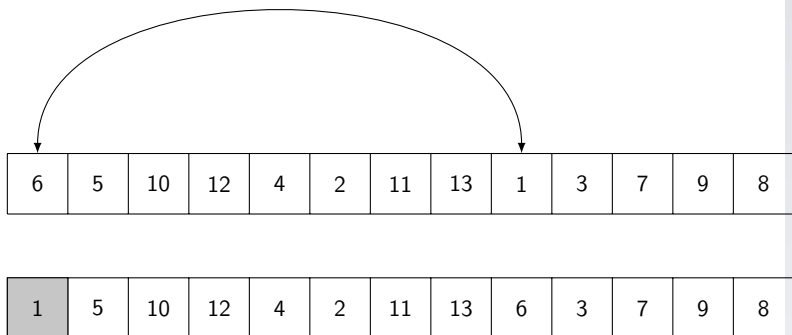


Figure 21 – Vraie avant la deuxième itération.

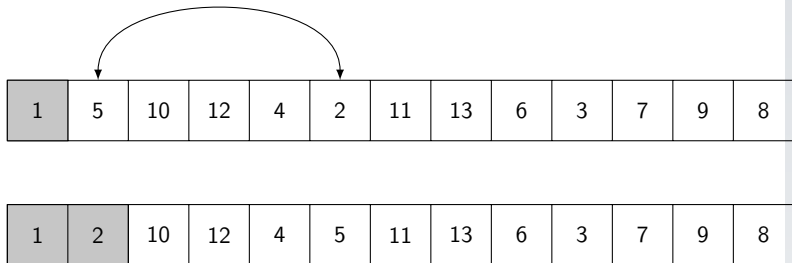


Figure 22 – Vraie avant la troisième itération.

À retenir

On peut démontrer mathématiquement 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

Coût

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

Coût

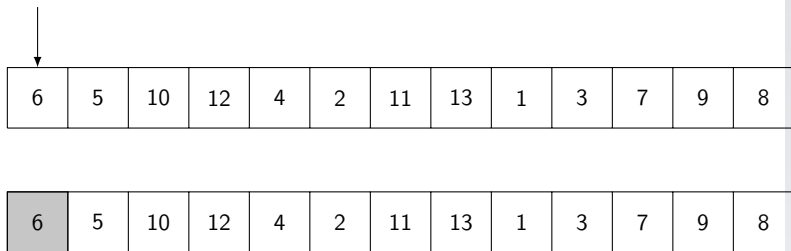


Figure 24 – Vraie avant la deuxième itération.

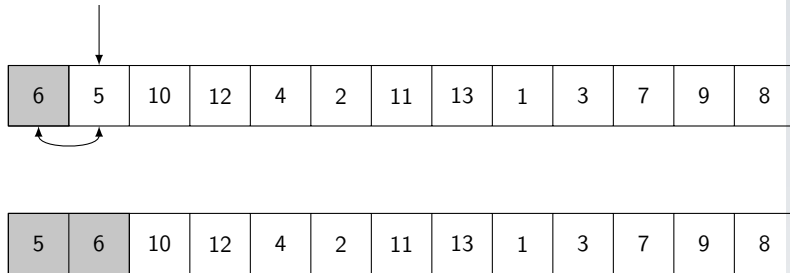


Figure 25 – Vraie avant la troisième itération.

À retenir

On peut démontrer mathématiquement 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

Coût

1. Algorithmes de tris

2. Implémentation

3. Études des implémentations

3.1 Terminaison

3.2 Correction

3.3 Coût

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

Coût

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

Coût

À retenir

- ▶ La coût 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.

Tri par sélection

```
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

Coût

```
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 $n - dep$ 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
- Coût

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
Coût

- ▶ 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
---	---	---	---	---

- ▶ ...

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

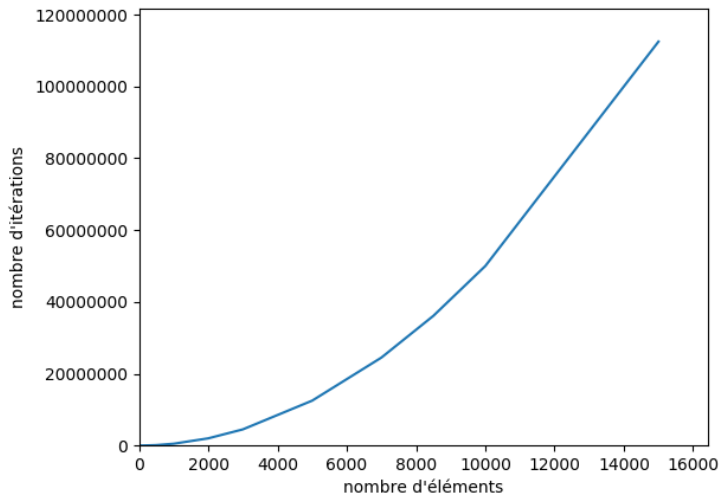
À retenir

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 le coût est **quadratique**.

Évolution du nombre d'itérations

[Trier des cartes](#)



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
Coût

Tri par insertion

```
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
- Coût

```
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.

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
- Coût

```
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
```

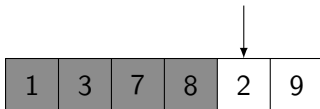


Figure 26 – 3 itérations pour placer 2



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
- Coût

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

Coût

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.

```
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 6 – 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](#)[Coût](#)

Correction

```
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 7 – Le tableau est inversé. La boucle `while` effectue `i` 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

Coût

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
Coût

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

Coût

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

Coût