#### Trier des cartes

#### Problématique

rier des cartes

Transposer au tri de données

Tri par sélection Implémentation

reminaison

Correction

Tri par insertior

Implémentation

Preuve de terminaison

Complexité

## Trier des cartes

### Christophe Viroulaud

Première - NSI

### Sommaire

### Trier des cartes

### Problématique

manuellement

Transposer au tri de données

Tri par sélection

Implémentatio

Correction

Comployitá

ri par insertion

mplémentation

Preuve de terminaison

Preuve de corr Complovitá

### 1. Problématique

- 2. Trier des cartes manuellement
- 3. Transposer au tri de données

#### Trier des cartes

### Problématique

rier des cartes ianuellement

Transposer au tri de données

Iri par selection Implémentation

Correction

Complexité

I par insercion

Preuve de terminaison Preuve de correction

Trier un jeu de cartes est une opération qui trouve des applications en informatique.

Existe-t-il plusieurs méthodes pour trier des données?

### Sommaire

Trier des cartes

Problématique

Trier des cartes manuellement

Transposer au tri

Tri par sélection

Implémentatio

Correction

Complexité

ri par insertion

Implémentation

Preuve de terminaison

Complexité

1. Problématique

2. Trier des cartes manuellement

3. Transposer au tri de données

#### Trier des cartes

roblématique

## Trier des cartes manuellement

de données

Tri par sélection

Terminaison

Correction

Complexité ri nar insertion

Implémentation

Preuve de terminaison

Complexité



FIGURE 1 – Cartes mélangées

#### Trier des cartes

#### Trier des cartes manuellement

### 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.

### Différentes méthodes

#### Trier des cartes

Problématique

## Trier des cartes manuellement

Transposer au tri de données

Tri par sélection

Implémentation

. ..

Complexité

Tri par insertion

implementation

Preuve de terminaison

Complexité

- ► Tri par sélection en place
- ► Tri par sélection dans un nouveau tableau
- ► Tri par insertion en place
- Tri par insertion dans un nouveau tableau

L	Pour chaque carte du tas
2	Trouver la plus petite carte dans la partie non tri
	ée.
3	Échanger cette carte avec la première de la
	partie non triée.
1	

Code 1 – Tri par sélection (en place)

### Retour menu

#### Trier des cartes

#### Problématique

## Trier des cartes manuellement

Fransposer au tri de données

ri par sélection Implémentation

Correction

orrection

ri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

Complexité

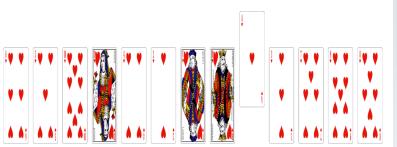


FIGURE 2 – Sélectionne la plus petite du tas non trié

### Retour menu

#### Problématique

## Trier des cartes manuellement

Transposer au tri de données

ri par sélection

mplémentation

Correction

Complexité Fri par insertion

Tri par insertion Implémentation

Preuve de terminaison Preuve de correction

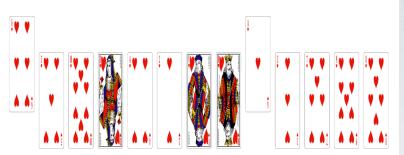


FIGURE 3 – Échange avec la première carte du tas non trié

### Retour menu

Problématique

Trier des cartes manuellement

Transposer au tri de données

ri par sélection Implémentation

Terminaison Correction

Complexité

Tri par insertion

Preuve de terminaison
Preuve de correction

omplexité

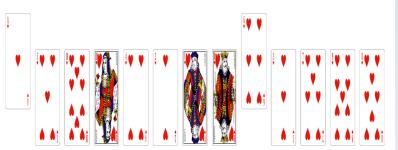


FIGURE 4 – Échange avec la première carte du tas non trié

### Retour menu

roblématique

Trier des cartes manuellement

Transposer au tri

ri par sélection

Terminaison

Correction

Tri par insertion

Preuve de terminaison
Preuve de correction



FIGURE 5 – La carte est à sa place

### Retour menu

#### roblématique

## Trier des cartes manuellement

Transposer au tri

Tri par sélection

Implémentation Terminairon

Terminaison Correction

Complexité

Tri par insertion

Preuve de terminaison

Preuve de correction

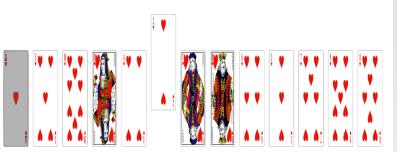


FIGURE 6 – Sélectionne la plus petite du tas non trié

### Retour menu

#### roblématique

## Trier des cartes manuellement

Transposer au tri de données

> ri par sélection mplémentation

> mplémentation Ferminaison

Correction

Complexité Fri par insertion

Implémentation

Preuve de terminaison
Preuve de correction
Complexitó

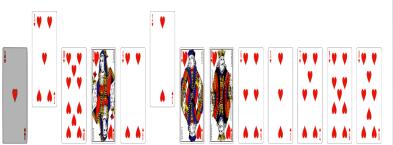


FIGURE 7 – Échange avec la première carte du tas non trié

### Retour menu

#### Problématique

#### Trier des cartes manuellement

Transposer au tri

ri par sélection Implémentation

Implémentation Terminaison

Comployitá

Tri par insertion

Implémentation
Preuve de terminaiso

Complexité



FIGURE 8 – Échange avec la première carte du tas non trié

### Retour menu

#### Problématique

## Trier des cartes manuellement

Transposer au tri

ri par sélection

Terminaison

Correction

Complexité Tri par insertion

Implémentation
Preuve de terminaison

Preuve de correct Complexité



FIGURE 9 – La carte est à sa place

### Retour menu

#### roblématique

## Trier des cartes manuellement

Transposer au tri

Tri par sélection

Implementation Terminaison

Correction

Complexité

Tri par insertion
Implémentation

Preuve de terminaison

Preuve de correct Complexité

## Tri par sélection - nouveau tableau

Trier des cartes

Transposer au tri de données

Trier des cartes

Tri par sélection Implémentation

Correction

Complexité

Implémentation

Preuve de terminaison

Complexité

Pour chaque carte du tas

Trouver la plus petite carte du tableau non trié.

La placer à la fin du tableau trié.

3

1

2

Code 2 - Tri par sélection dans un nouveau tableau

## Tri par sélection - nouveau tableau

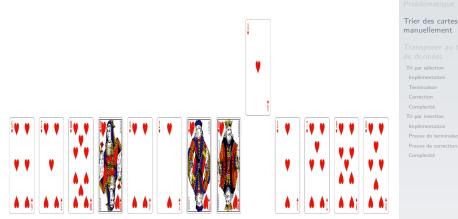


FIGURE 10 - Trouve la plus petite du tas non trié

### Retour menu

### 18 / 76

## Tri par sélection - nouveau tableau

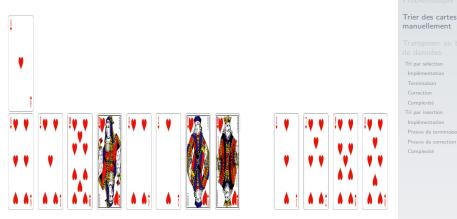


FIGURE 11 – La place à la fin du tableau trié

Trier des cartes manuellement

## Tri par sélection - nouveau tableau

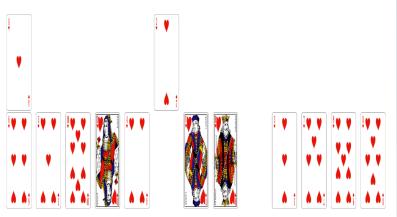


FIGURE 12 - Trouve la plus petite du tas non trié

Trier des cartes manuellement

## Tri par sélection - nouveau tableau

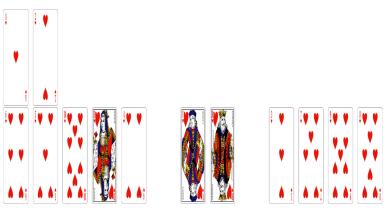


FIGURE 13 – La place à la fin du tableau trié

#### Problématique

## Trier des cartes manuellement

Transposer au tri de données

Iri par selection Implémentation

Terminaison

Correction

Tri par insertion

Implémentation

Preuve de terminaisor

Complexité

1 Pour chaque carte du tas

Mémoriser la carte en cours

Décaler vers la droite toutes les cartes précédentes, supérieures à la carte en cours.

Insérer la carte en cours dans l'espace vide.

4 5

3

Code 3 – Tri par insertion (en place)

Trier des cartes manuellement

## Tri par insertion en place

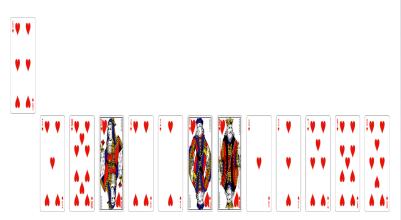


FIGURE 14 - Mémoriser la première carte dans le tas non trié

### Retour menu

### 23 / 76

Trier des cartes manuellement

## Tri par insertion en place

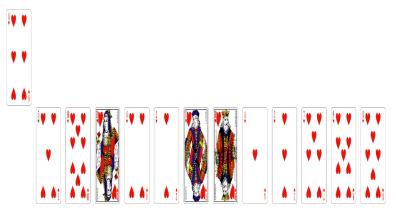


FIGURE 15 – Décaler les cartes supérieures déjà triées

### Retour menu

### 24 / 76



FIGURE 16 - Replacer la carte dans l'espace

### Retour menu

#### roblématique

## Trier des cartes manuellement

Transposer au tri

Tri par sélection

mplémentation Ferminaison

Correction

Complexité

Tri par insertion

Preuve de terminaison

Preuve de correction

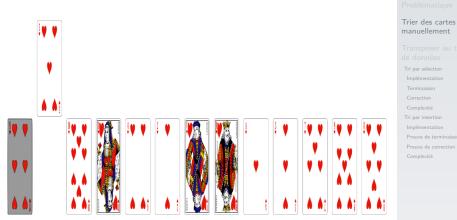


FIGURE 17 - Mémoriser la première carte dans le tas non trié

### Retour menu

### 26 / 76

Trier des cartes manuellement

## Tri par insertion en place

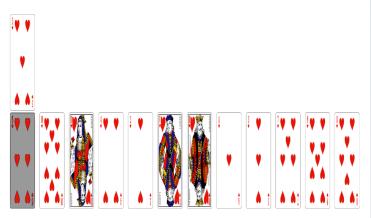


FIGURE 18 – Décaler les cartes supérieures déjà triées

### Retour menu

### 27 / 76



FIGURE 19 - Replacer la carte dans l'espace

### Retour menu

#### roblématique

## Trier des cartes manuellement

Transposer au tri

Tri par sélection

Implémentation Terminaison

Correction

Complexité

Tri par insertion

Preuve de terminaison

Complexité

#### Trier des cartes

roblématique

## Trier des cartes manuellement

Transposer au tri de données

ri par sélection mplémentation

Terminaison

Correction

Complexité

Iri par ins

Preuve de terminaison

Preuve de corre Complexité

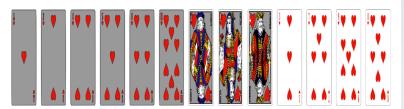


FIGURE 20 – Après plusieurs itérations

### Trier des cartes

Problématique

## Trier des cartes manuellement

Fransposer au tri le données

ri par sélection Implémentation

erminaison orrection omplexité

Tri par insert

Implémentation Preuve de terminai

Preuve de terminaison Preuve de correction Complexité

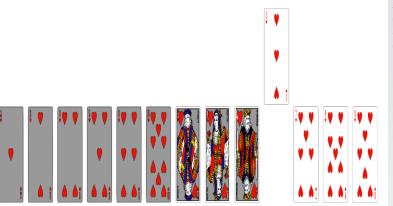


FIGURE 21 - Mémoriser la première carte dans le tas non trié

### Trier des cartes

Problématique

## Trier des cartes manuellement

ransposer au tri e données

ri par selection Implémentation Terminaison

Correction Complexité

Tri par insertion Implémentation

Preuve de terminaison
Preuve de correction

Preuve de corre Complexité

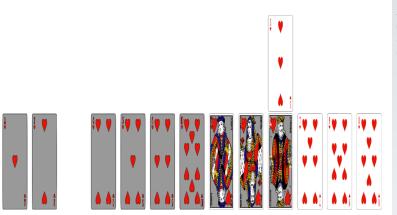


FIGURE 22 – Décaler les cartes supérieures déjà triées

#### Trier des cartes

oblématique

## Trier des cartes manuellement

Transposer au tri de données

ri par sélection

mplementation Ferminalson

Correction

Complexité

Iri par inse

Preuve de terminaison Preuve de correction

Preuve de Complexité

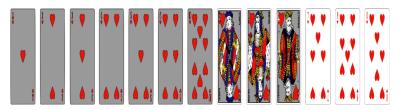


FIGURE 23 – Insérer la carte dans l'espace

#### Trier des cartes manuellement

Pour chaque carte du tas 1

Prendre la première carte du tableau non trié.

Dans le tableau trié, décaler vers la droite toutes les cartes plus grandes.

Insérer la carte dans le tableau trié.

4 5

3

Code 4 – Tri par insertion dans un nouveau tableau

# Tri par insertion - nouveau tableau

### Trier des cartes

Problématique

## Trier des cartes manuellement

Transposer au tri de données

ri par sélection Implémentation

Terminaison

Correction Complexité

Tri par inserti

Preuve de terminaison
Preuve de correction

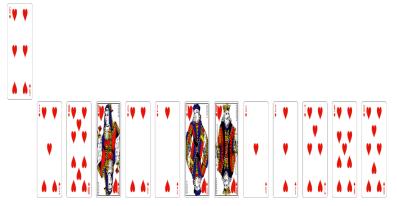


FIGURE 24 – Prendre la première carte non triée

manuellement

## Tri par insertion - nouveau tableau

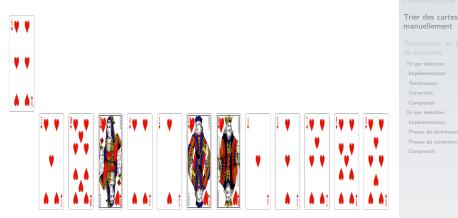


FIGURE 25 – Décaler les cartes supérieures du tableau trié

### Retour menu

### 35 / 76

## Tri par insertion - nouveau tableau



FIGURE 26 – Insérer la carte dans le tableau triée

### Retour menu

#### Trier des cartes

Problématique

## Trier des cartes manuellement

Transposer au tri

Tri par sélection

Implémentation

Correction

Complexité

mplémentation

Preuve de terminaison

# Tri par insertion - nouveau tableau



Trier des cartes



FIGURE 27 – Prendre la première carte non triée

Retour menu 37/76

# Tri par insertion - nouveau tableau



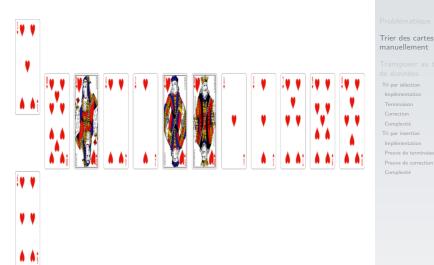


FIGURE 28 – Décaler les cartes supérieures du tableau trié

Retour menu 38/76

# Tri par insertion - nouveau tableau



FIGURE 29 – Insérer la carte dans le tableau triée

## Retour menu

### Trier des cartes

#### Problématique

# Trier des cartes manuellement

Transposer au tri

Tri par sélection Implémentation

Terminaison

Complexité

ri par insertion

# Sommaire

### Trier des cartes

Problématique

Trier des cartes manuellement

# Transposer au tri

Tri par sélection

Implementation

Correction

Complexité

Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

1. Problématique

2. Trier des cartes manuellement

- 3. Transposer au tri de données
- 3.1 Tri par sélection
- 3.2 Tri par insertion

```
def ma_fonction(tab: list)->None:
    tab[2] = 199

tab = [3, 8, 1, 10, 9]
ma_fonction(tab)
```

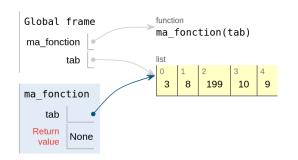


FIGURE 30 – La fonction modifie le tableau original

Trier des cartes

robiematique

rier des cartes nanuellement

Transposer au tri de données

par sélection

nplémentation

erminaison

orrection

omplexité

Implémentation

Preuve de corre Complexité

# Rappel de l'algorithme

Trier des cartes

rier des cart ianuellement

Fransposer au tri de données

Tri par sélection

#### Implémentation

Correction

Complexité

Tri par insertion

Implémentation

Preuve de terminaison

euve de correctio

Pour chaque carte du tas

1

2

Trouver la plus petite carte dans la partie non triée. Échanger cette carte avec la première de la partie non triée.

Code 5 – Tri par sélection (en place)

### Problematique

ier des cartes

Transposer au tri de données

Tri par sélection

#### Implémentation

Terminaison

Correction

Tri par incortion

Implémentation

Preuve de terminaison

# Activité 2 :

1. Écrire la fonction **trouver\_mini(tab : list)**  $\rightarrow$  **int** qui renvoie l'indice du plus petit élément de *tab*.

```
def trouver_mini(tab: list)->int:
1
       77 77 77
2
3
       Trouve l'indice du plus petit élément
4
       i mini = 0
5
       for j in range(1, len(tab)):
6
           if tab[j] < tab[i_mini] :</pre>
7
              i mini = i
8
       return i mini
9
```

#### Problématique

rier des cartes

Fransposer au tri le données

Tri par sélection

#### Implémentation

Terminaison

Correction

Tri nor incortion

Implémentation

Preuve de terminaison

Preuve de correctio

# Tri par sélection - Implémentation

## Activité 2:

- Adapter la fonction précédente pour renvoyer l'indice du plus petit élément de tab, compris entre l'indice i\_depart et la fin du tableau. La signature de la fonction deviendra trouver\_mini(tab : list, i\_depart : int) → int.
- Écrire la fonction echanger(tab : list, i : int, j : int) → None qui échange les éléments d'indice i et j du tableau tab.
- Écrire la fonction tri\_selection(tab : list) →
   None qui effectue un tri par sélection sur tab.

Problématique

rier des cartes anuellement

Transposer au tri de données

Tri par sélection Implémentation

Terminaison

Correction Complexité

Tri par insertion

Implémentation
Preuve de terminaison

```
def trouver_mini(tab: list, i_depart: int) -> int:
 1
        77 77 77
2
        renvoie l'indice du plus petit élément entre
3
        i_depart et la fin du tableau
4
 5
6
        i_mini = i_depart
        for j in range(i_depart+1, len(tab)):
7
           if tab[i] < tab[i_mini] :</pre>
8
               i mini = i
9
        return i_mini
10
```

#### Problématique

rier des cartes anuellement

Fransposer au tri de données

Tri par sélection

#### Implémentation

C-----

Complexité

ri par insertion

Preuve de terminaison

rreuve de terminaison

# Correction

1

3 4

5

### Trier des cartes

### Problématique

rier des cartes nanuellement

Fransposer au tri le données

Iri par selection

#### Implémentation

C-----

Complovitó

ri par insertion

Implementation

Prouve de correction

mplexité

```
def tri_selection(tab: list) -> None:
    """

tri le tableau dans l'ordre croissant

for i in range(len(tab)):
    position_du_mini = trouver_mini(tab, i)
    echanger(tab, i, position_du_mini)
```

#### Problématique

rier des cartes anuellement

Fransposer au tri le données

Tri par sélection

#### Implémentation

C----

Complexité

ri par insertion

Description

Preuve de terminaison

## Activité 2:

- 5. Construire par compréhension un tableau des entiers de 1 à 13.
- 6. Mélanger le tableau à l'aide de la méthode shuffle de la bibliothèque random.
- 7. Trier le tableau à l'aide de la fonction *tri selection*.

#### Implémentation

# Correction

```
Trier des cartes
```

```
cartes = [i for i in range(1, 14)]
shuffle(cartes)
tri_selection(cartes)
```

#### Problématique

rier des cartes anuellement

Transposer au tr de données

Iri par selection

#### Implémentation

-

Correction

T. . .

iri par insertion

Preuve de terminaison

# Preuve de terminaison : variant de boucle

Trier des cartes

La terminaison de la fonction est triviale. Le tri est composé de deux boucles bornées donc qui terminent.

Problematique

er des carte inuellement

Transposer au tri

Tri par sélection Implémentation

Correction

Correction

Tri par insertion

Iri par insertion

Preuve de terminaison

Preuve de terminaison

Complexité

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



Code 6 – Avant la première itération, la partie gauche est vide, donc triée.

# Preuve de correction : invariant de boucle



Code 7 – Avant la deuxième itération, la partie gauche est triée.

### Trier des cartes

#### Correction

# Preuve de correction : invariant de boucle



Code 8 – Avant la troisième itération, la partie gauche est triée.

#### Trier des cartes

Problématique

rier des cartes ianuellement

Transposer au tri de données

Tri par sélection Implémentation

Terminaison

Correction

Complexité
Tri par insertion

Tri par insertion

Preuve de terminaison

Preuve de correctio Complexité La boucle externe effectue **n itérations**.

à la première itération de *i*, la boucle de la fonction trouver\_mini effectue n-1 itérations.

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

 à la deuxième itération de *i*, la boucle de la fonction trouver\_mini effectue n-2 itérations.

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

**.**..

#### Problématiqu

rier des cartes

ransposer au tr e données

ri par sélection mplémentation

Correction

Complexité

Tri par insertion

Implémentation

Preuve de terminaison

#### Problématique

#### rier des cartes anuellement

Transposer au tri le données

Tri par sélection Implémentation

Correction

### Complexité

Tri par insertion

Preuve de terminaison

Complexité

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

# À retenir

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

Le nombre d'opérations dépend de  $n^2$ .

### B 117

rier des cartes

Fransposer au tr le données

ri par sélection

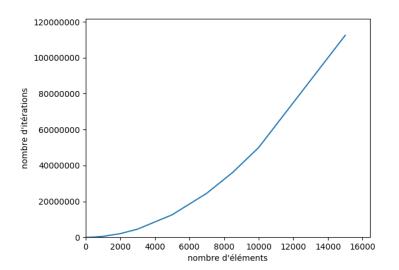
erminaison

Complexité

Tri par insertion

Preuve de terminaison

omplovitá



## Sommaire

Trier des cartes

- 1. Problématique
- 2. Trier des cartes manuellement
- 3. Transposer au tri de données
- 3.1 Tri par sélection
- 3.2 Tri par insertion
  Implémentation
  Preuve de terminaison
  Preuve de correction
  Complexité

#### Problématique

Trier des cartes nanuellement

Transposer au tri de données

Tri par sélection Implémentation

Terminaison

Complexité

Tri par insertion

Implémentation

Preuve de terminaison

## Activité 3:

- Écrire la fonction tri\_insertion(tab : list)
   →None en s'appuyant sur l'algorithme. Les
   indications suivantes permettront de construire les
   trois étapes :
  - Mémoriser : définir une variable en\_cours, élément en cours de placement et pos, position actuelle de cet élément.
  - <u>Décaler</u>: utiliser une boucle non bornée pour décaler les éléments vers la droite.
  - Insérer : placer l'élément en\_cours à la nouvelle position pos.
- 2. Tester la fonction sur un tableau.

### Problèmatique

manuellement

de données

Tri par sélection Implémentation

Correction

Complexité ri par insertion

#### Implémentation

# Rappel de l'algorithme

#### Trier des cartes

rier des carte anuellement

Fransposer au tri

Tri par sélection

Implementation

Correction

par insertion

Implémentation

Preuve de terminaison Preuve de correction

1 Pour chaque carte du tas

3

4

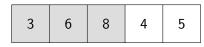
2 Mémoriser la carte en cours

Décaler vers la droite toutes les cartes précédentes, supérieures à la carte en cours.

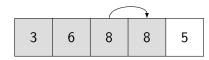
Insérer la carte en cours dans l'espace vide.

Code 9 – Tri par insertion (en place)

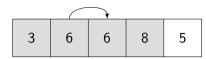
# Subtilité sur le tableau Python

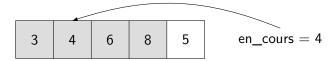


$$\mathsf{en}\_\mathsf{cours} = 4$$



$$\mathsf{en\_cours} = 4$$





Code 10 – Ce qu'il se passe réellement dans le tableau Python

Trier des cartes

Problématique

rier des cartes

. ransposer au tr e données

> par sélection plémentation

Correction

i par insertion

Implémentation

# Correction: boucle principale

```
def tri_insertion(tab: list) -> None:

tri le tableau dans l'ordre croissant
for i in range(len(tab)):
```

#### Trier des cartes

#### Problématique

#### rier des cartes anuellement

#### Fransposer au tri le données

Tri par sélection

Terminaison

Correction

mplexité

### Implémentation

#### Problématique

Frier des cartes nanuellement

Transposer au tri de données

Tri par sélection Implémentation

Terminaison

Complexité

par insertion

#### Implémentation

Transposer au tri de données

l'ri par sélection

erminaison

orrection

mplexité par insertion

#### Implémentation

```
\begin{array}{lll} & \text{while pos} > 0 \text{ and en\_cours} < \mathsf{tab[pos-1]}: \\ & \mathsf{tab[pos]} = \mathsf{tab[pos-1]} \\ & \mathsf{pos} = \mathsf{pos}{-1} \end{array}
```

Correction: insérer

1

 $tab[pos] = en\_cours$ 

#### Trier des cartes

Problématique

rier des cartes anuellement

Transposer au tri de données

l'ri par sélection

Terminaison

Correction

i nar insertion

#### Implémentation

# Correction: code complet

```
def tri insertion(tab: list) -> None:
 1
2
       tri le tableau dans l'ordre croissant
3
       77 77 77
 4
       for i in range(len(tab)):
5
           # mémoriser
6
           en cours = tab[i]
           pos = i
8
           # décaler
9
           while pos > 0 and en_cours < tab[pos-1]:
10
              tab[pos] = tab[pos-1]
11
12
              pos = pos - 1
           # insérer
13
           tab[pos] = en\_cours
14
```

#### Problématique

Trier des cartes

Transposer au tri de données

> ri par sélection Implémentation

Correction

mplexité par insertion

## Implémentation

## Correction: tester

```
Trier des cartes
```

## Trier des cartes

Transposer au tri

Tri par sélection

Implémentation Terminaison

Correction

omplexité

### Implémentation

```
cartes = [i for i in range(1, 14)]
shuffle(cartes)
tri_insertion(cartes)
```

## Preuve de terminaison

Trier des cartes

Il faut se focaliser sur la boucle interne, non bornée.

**Activité 4 :** Déterminer un variant de la boucle, qui prouve la terminaison.

#### Problématique

ier des carte

Fransposer au tri le données

Fri par sélection Implémentation

Correction

ri par insertion

Implementation

#### Preuve de terminaison

Preuve de correction

# Correction

```
while pos > 0 and en_cours < tab[pos-1] :
    tab[pos] = tab[pos-1]

pos = pos-1</pre>
```

pos est un variant de la boucle.

### Trier des cartes

#### Problématique

ier des cart anuellement

Fransposer au tri le données

mplémentation

erminaison

Correction

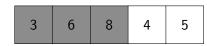
Tri par insertion

Implémentation

#### Preuve de terminaison

Preuve de correction

Comme pour le tri sélection, avant chaque itération de la boucle externe, la partie gauche du tableau est triée.



Code 11 - Insertion de l'élément 4

#### Problématique

rier des cartes

Transposer au tri

Tri par sélection Implémentation

> erminaison orrection

Complexité

Implémentation

Preuve de terminaison

### Preuve de correction

#### Trier des cartes

4



4





Code 12 - Insertion de l'élément 4

#### Problématique

# rier des cartes

#### Transposer au tri de données

Tri par sélection Implémentation

Correction

Complexité

Implémentation

## Preuve de terminaison Preuve de correction

La boucle externe effectue *n* itérations dans tous les cas.

for i in range(len(tab)):

Cependant, le nombre d'itérations de la boucle interne peut varier.

while pos > 0 and en\_cours < tab[pos-1]:

4



Code 13 - Insertion de l'élément 4

Trier des cartes

Problématiqu

rier des cartes

Fransposer au tri de données

ri par sélection mplémentation

orrection

Complexité

mplémentation

Preuve de terminaison

#### Trier des cartes

#### robiematique

ier des cartes anuellement

le données

Iri par selection Implémentation

Terminaison

Comployitá

Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

Complexité

## Activité 5:

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

while pos > 0 and en\_cours < tab[pos-1]:

1 4 5 7 8

Code 14 – Le tableau est déjà trié. La boucle interne n'effectue aucune itération.

#### Problématiqu

rier des cartes

ransposer au tr e données

ri par sélection Implémentation

Terminaison

Complexité

ri par insertion

Preuve de terminaison

```
for i in range(len(tab)):
    en_cours = tab[i]
    pos = i
    while pos > 0 and en_cours < tab[pos-1] :</pre>
```

```
8 7 5 4 1
```

Code 15 – Le tableau est inversé. La boucle interne effectue *i* itérations.

### Problématique

er des carte nuellement

ransposer au tri e données

ri par selection Implémentation

Correction

Complexité Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correct Complexité

# À retenir

Le tri par insertion effectue un nombre moyen d'opérations qui dépend de  $n^2$ .

#### Trier des cartes

#### Problématique

#### rier des cartes anuellement

#### Transposer au tri de données

#### Tri par sélection Implémentatio

### Terminaison

### Correction

### Complexite Tri par insertion

### Implémentation

### Preuve de terminaison