Trier des cartes

1 Toblematique

Trier des cartes

Transposer au tri de données

Tri par sélection Implémentation

reminaison

Correction

Tri par insertior

Implémentation

Preuve de terminaison

Preuve de correction

Trier des cartes

Christophe Viroulaud

Première - NSI

Sommaire

1. Problématique

2. Trier des cartes manuellement

3. Transposer au tri de données

Implémentation Terminaison Correction

Complexité

Implémentation

Preuve de terminaison

Preuve de correction

Complexité

Problématique

Trier des cartes manuellement

Transposer au tri de données

Tri par sélection

Terminaison

Correction

Complexité

Tri par insertion

Preuve de terminaison

Preuve de correction

Trier des cartes

Problématique

rier des cartes ianuellement

Transposer au tri de données

Tri par sélection Implémentation

C----

Complexité

Tri par insertion

Preuve de terminaison

euve de correction omplexité

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

1. Problématique

2. Trier des cartes manuellement

3. Transposer au tri de données

Implémentation Terminaison Correction Complexité

Implémentation

Preuve de terminaison Preuve de correction

Complexité

Problématique

Trier des cartes manuellement

Transposer au tri de données

Tri par sélection

implementation ---

Correction

Complexité

Implémentation

Preuve de terminaison
Preuve de correction

Problématique

Trier des cartes manuellement

de données

Tri par sélection

Terminaison

Correction

complexite

Implémentation

Preuve de terminaison



FIGURE 1 – Cartes mélangées

Trier des cartes

Problématique

Trier des cartes manuellement

Transposer au tri de données

Impar selection

Terminaison

Correction

Complexite

Implémentation

Preuve de terminaison

Preuve de correcti

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

reuve de correction

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

Pour chaque carte du tas

partie non triée.

Transposer au tri de données

Tri par sélection Implémentation

Correction

omplexité

ri par insertion

Preuve de terminaison

Complexité

Code 1 – Tri par sélection (en place)

Échanger cette carte avec la première de la

Trouver la plus petite carte dans la partie non tri

Retour menu

1

2

3

4

ée.

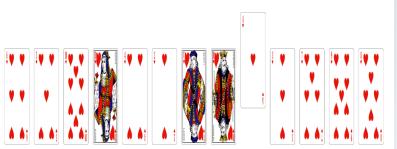


FIGURE 2 – Sélectionne la plus petite du tas non trié

Retour menu

Problématique

Trier des cartes manuellement

Transposer au tri de données

> i par sélection nplémentation

Terminaison

Comployitá

Complexite

Fri par insertion

Implémentation
Preuve de terminaison

Preuve de correction

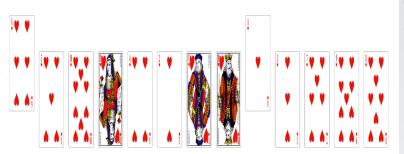


FIGURE 3 – Échange avec la première carte du tas non trié

Retour menu

Problématique

Trier des cartes manuellement

Transposer au tri de données

ri par sélection

Terminaison Correction

Complexité Tri par insertion

Implémentation
Preuve de terminaison

Preuve de correctio Complexité

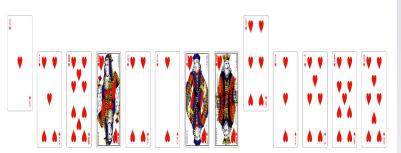


FIGURE 4 – Échange avec la première carte du tas non trié

Retour menu

roblématique

Trier des cartes manuellement

Transposer au tri

i par sélection nplémentation

Terminaison Correction

Complexité

Implémentation

Preuve de terminaison
Preuve de correction
Complexité



FIGURE 5 – La carte est à sa place

Retour menu

roblématique

Trier des cartes manuellement

Transposer au tri

Tri par sélection

Implémentation Terminaison

Correction

Complexité

Implémentation

Preuve de terminaison

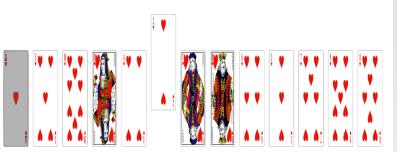


FIGURE 6 – Sélectionne la plus petite du tas non trié

Retour menu

roblématique

Trier des cartes manuellement

Transposer au tri de données

> ri par sélection mplémentation

Terminaison

Correction

Tri par insertion

Preuve de terminaison
Preuve de correction

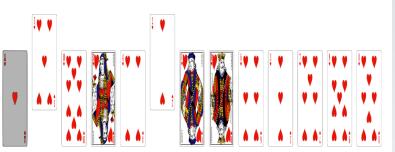


FIGURE 7 – Échange avec la première carte du tas non trié

Retour menu

roblématique

Trier des cartes manuellement

Transposer au tri

i par sélection nplémentation

Terminaison

Correction Complexité

Tri par insertion

Preuve de terminaison



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

Retour menu

Problématique

Trier des cartes manuellement

Transposer au tri

i par sélection

Terminaison

Correction

Tri par insertion

Implémentation
Preuve de terminaison



FIGURE 9 – La carte est à sa place

Retour menu

Problématique

Trier des cartes manuellement

Transposer au tri

Tri par sélection

Implémentation Terminaison

Correction

Complexité

Tri par insertion

Preuve de terminaison

Preuve de correction

Tri par sélection - nouveau tableau

Trier des cartes

Fransposer au tr

Trier des cartes

Tri par sélection

Terminaison

Correction

Complexite
Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

Complexité

Pour chaque carte du tas

Trouver la plus petite carte du tableau non trié.

La placer à la fin du tableau trié.

3 4

1

2

Code 2 - Tri par sélection dans un nouveau tableau

Tri par sélection - nouveau tableau

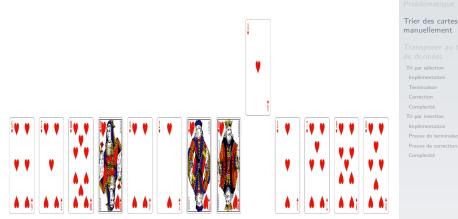


FIGURE 10 - Trouve la plus petite du tas non trié

manuellement

Tri par sélection - nouveau tableau

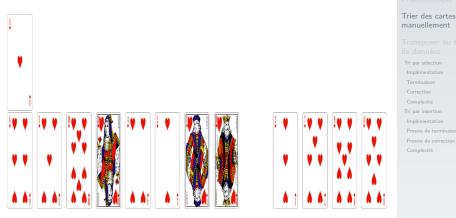


FIGURE 11 – La place à la fin du tableau trié

Trier des cartes manuellement

Tri par sélection - nouveau tableau

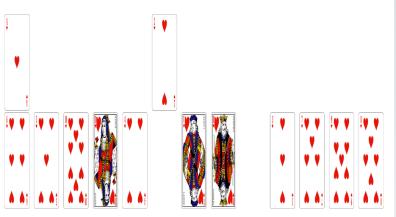


FIGURE 12 – Trouve la plus petite du tas non trié

Trier des cartes manuellement

Tri par sélection - nouveau tableau

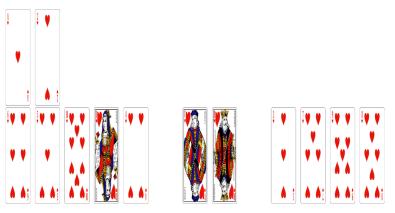


FIGURE 13 – La place à la fin du tableau trié

Problématique

Trier des cartes manuellement

Transposer au tri de données

Tri par sélection Implémentation

Terminaison

Correction
Comployité

Tri par insertion

Implémentation

Preuve de correction

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)

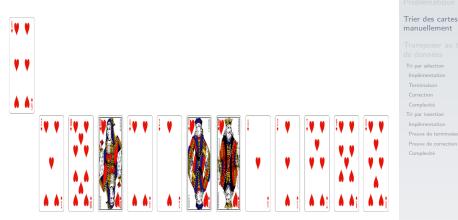


FIGURE 14 - Mémoriser la première carte dans le tas non trié

Trier des cartes manuellement

Tri par insertion en place

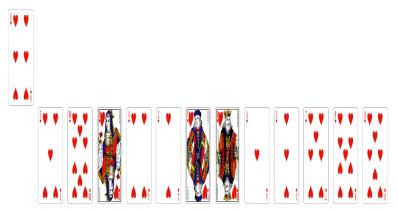


FIGURE 15 – Décaler les cartes supérieures déjà triées

Retour menu

24 / 75

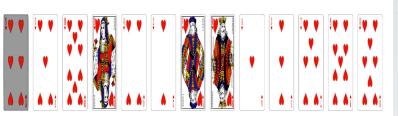


FIGURE 16 - Replacer la carte dans l'espace

Retour menu

roblématique

Trier des cartes manuellement

Transposer au tri

Tri par sélection

Implémentation Terminaison

Correction

Complexité

Fri par insertion

Implémentation

Preuve de terminaison
Preuve de correction

mplexité

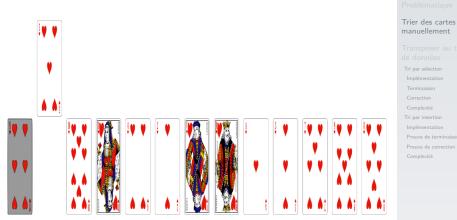


FIGURE 17 - Mémoriser la première carte dans le tas non trié

Retour menu

26 / 75

Trier des cartes manuellement

Tri par insertion en place

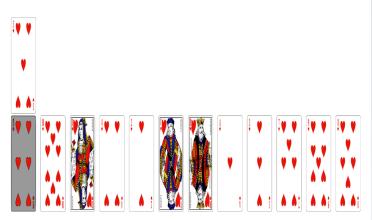


FIGURE 18 – Décaler les cartes supérieures déjà triées



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

Fri par insertion

Implémentation

Preuve de terminaison Preuve de correction

Trier des cartes

oblématique

Trier des cartes manuellement

Transposer au tri de données

ri par sélection mplémentation

mplémentation Ferminaison

Correction

Complexité

Iri par ins

Preuve de terminais

Preuve de correction Complexité

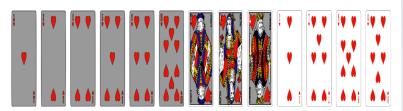


FIGURE 20 – Après plusieurs itérations

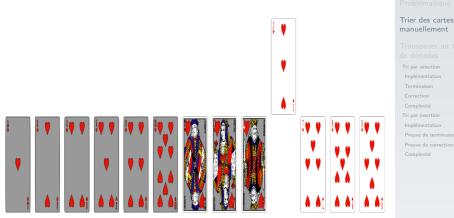


FIGURE 21 - Mémoriser la première carte dans le tas non trié

Retour menu

30 / 75

Trier des cartes



Trier des cartes manuellement

ransposer au tri e données

ri par sélection Implémentation Terminaison

Correction
Complexité

Implémentation

Preuve de termina

Preuve de terminaison Preuve de correction Complexité

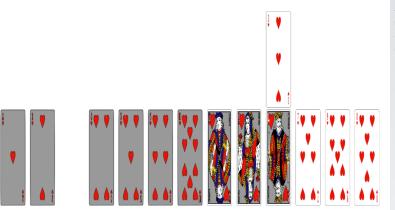


FIGURE 22 – Décaler les cartes supérieures déjà triées

Trier des cartes

roblématique

Trier des cartes manuellement

Fransposer au tri de données

ri par sélection

mplementation erminaison

Correction

Tri par insert

Implémen

Preuve de terminaisor

Preuve de corre Complexité

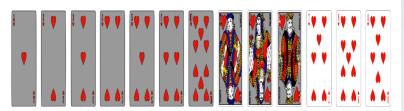


FIGURE 23 – Insérer la carte dans l'espace

Transposer au tri

Tri par sélection Implémentation

Terminaison

Correction

ri par insertion

Implémentation

Preuve de correction

Complexité

Pour chaque carte du tas

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

Trier des cartes manuellement

Tri par insertion - nouveau tableau

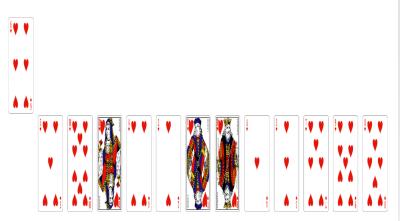


FIGURE 24 – Prendre la première carte non triée

Trier des cartes manuellement

Tri par insertion - nouveau tableau

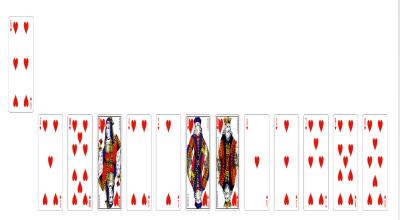


FIGURE 25 – Décaler les cartes supérieures du tableau trié

Tri par insertion - nouveau tableau





FIGURE 26 – Insérer la carte dans le tableau triée

Retour menu

Trier des cartes

roblématique

Trier des cartes manuellement

Transposer au tri

Tri par sélection Implémentation

Terminaison

Correction

ri par insertion

mplémentation

euve de correction mplexité

Tri par insertion - nouveau tableau

! V V



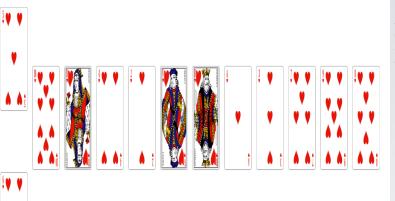
Trier des cartes

Trier des cartes manuellement

FIGURE 27 - Prendre la première carte non triée

Retour menu 37 / 75

Tri par insertion - nouveau tableau



Problematique

Trier des cartes manuellement

Fransposer au tri le données

In par selection Implémentation

Terminaison

Correction

Complexité

Tri par insertion

Preuve de terminaison

Complexité

FIGURE 28 – Décaler les cartes supérieures du tableau trié

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

Terminaison

Correction

ri par insertion

Implémentation

Sommaire

- 1. Problématique
- 2. Trier des cartes manuellement
- 3. Transposer au tri de données
- 3.1 Tri par sélection

Implémentation

Terminaison

Correction

Complexité

3.2 Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

Complexité

Problématique

Trier des cartes

Transposer au tri de données

Tri par sélection

Implémentation

Correction

Complexité

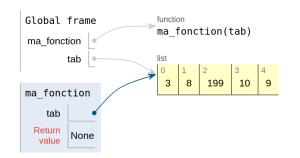
Tri par insertion

Prouve de terminairen

Preuve de correction

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

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



 ${
m Figure}$ 30 – La fonction modifie le tableau original

robiematique

rier des cartes nanuellement

Transposer au tri de données

par sélection
uplémentation
erminaison
orrection
omplexité
par insertion
uplémentation

reuve de termina reuve de correcti

Rappel de l'algorithme

Trier des cartes

Implémentation

Pour chaque carte du tas

1

2 3 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)

Problématique

ier des cartes

Transposer au tri de données

Tri par sélectio

Implémentation

Terminaison

Complexité

Iri par insertion

Preuve de terminaison

omplexité

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 anuellement

Fransposer au tri de données

Iri par selection

Implémentation

rerminaison

Completed

Tri par insertion

Implémentation

Preuve de terminaison

Complexité

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 carte anuellement

Transposer au tri de données

Tri par sélection Implémentation

Implémentation

Correction

Complexité ri par insertion

Implémentation

Preuve de correctio

```
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[j] < tab[i_mini] :</pre>
8
               i mini = i
9
        return i_mini
10
```

Problématique

rier des cartes anuellement

Fransposer au tri le données

Iri par selection

Implémentation

C----

"omnlexité

ri par insertion

Prouve de terminaison

Preuve de terminaison

omplexité

Correction

1

3 4

5

Trier des cartes

Problématique

rier des cartes nanuellement

Fransposer au tri le données

Iri par selection

Implémentation

Terminaison

Complexité

Tri par insertion

Preuve de terminaison

Prouve de correction

mplexité

```
def echanger(tab: list, i: int, j: int) -> None:
    """
    inverse les élément d'indices i et j du tableau
    """
    tab[i], tab[j] = tab[j], tab[i]
```

```
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

Iri par selection

Implémentation

C-----

Complexité

ri par insertion

D. I. . . .

Preuve de terminaison

mnlovitá

Tri par sélection - Tester

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.

Problématique

ier des carte anuellement

Transposer au tri de données

Iri par selection

Implémentation

Correction

Complexité

Tri par insertion

Preuve de terminaison

Complexité

Correction

```
Trier des cartes
```

Problématique

rier des cartes anuellement

Transposer au tr de données

Tri par sélection

Implémentation

reminaison

Correction

Tri nor insortion

Implémentation

Preuve de terminaison

euve de correction

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

- shuffle(cartes)
- 3 tri_selection(cartes)

Preuve de terminaison : variant de boucle

Trier des cartes

Problématique

er des carte nuellement

Transposer au tri de données

Tri par sélection

Implémentation Terminaison

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

Complexité

Implémentation

er des carte inuellement

Transposer au tri

Tri par sélection Implémentation

Correction

Correction

Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

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

Problématique

rier des cartes anuellement

Transposer au tri de données

Tri par sélection Implémentation

Terminaison

Correction

Complexité

Tri par insertion

Preuve de terminaison

reuve de correctior

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

Implementation

Preuve de terminaison

- à la première itération de *i*, la boucle de la fonction trouver_mini effectue n-1 itérations.
- à la deuxième itération de *i*, la boucle de la fonction trouver_mini effectue n-2 itérations.

▶ ..

```
for j in range(i_depart+1, len(tab)):
    if tab[j] < tab[i_mini]:
        i_mini = j</pre>
```

Problématique

nanuellement

Fransposer au tri le données

nplémentation

Complexité

Complexite Fri par insertion

Implémentation

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

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 .

Complexité



Fransposer au t le données

i par sélection

erminaison

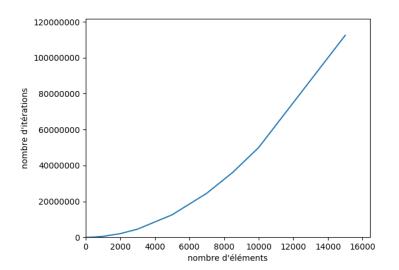
Complexité

ri par insertion

Implementation

Preuve de terminaison

reuve de co



Sommaire

- 3. Transposer au tri de données

Implémentation

Terminaison

Correction

Complexité

3.2 Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

Complexité

Tri par insertion

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.

Problématique

Trier des cartes manuellement

Transposer au tri de données

Tri par sélection

Terminaison Correction

Complexité ri par insertion

Implémentation

Preuve de terminaison
Preuve de correction
Complexité

Problématique

ier des carte anuellement

Transposer au tri de données

Fri par sélection

Terminaison

orrection

par insertion

Implémentation

Preuve de terminaison Preuve de correction

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)

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 de données

Tri par sélection Implémentation

Terminaison

Correction

nar insertion

Implémentation

Problématique

Trier des cartes

Transposer au tri de données

Tri par sélection Implémentation

Terminaison

Correction

par insertion

Implémentation

Transposer au tri de données

l'ri par sélection

erminaison

orrection

mplexité nar insertion

Implémentation

```
\begin{array}{ccc} \text{1} & \text{while pos} > 0 \text{ and en\_cours} < \mathsf{tab[pos-1]}: \\ \text{2} & \mathsf{tab[pos]} = \mathsf{tab[pos-1]} \\ \text{3} & \mathsf{pos} = \mathsf{pos-1} \end{array}
```

Correction: insérer

1

tab[pos] = en_cours

Trier des cartes

Problématique

rier des cartes lanuellement

Transposer au tri de données

ri par sélection

Terminaison

Lorrection

i par 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

par insertion

Implémentation

Preuve de terminaison
Preuve de correction
Complexité

Correction: tester

```
Trier des cartes
```

1 Toblematique

rier des cartes nanuellement

Transposer au tri de données

Tri par sélection Implémentation

Terminaison

Correction

omplexité i nar insertion

Implémentation

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

Problématique

rier des cartes anuellement

Transposer au tri de données

ri par sélection Implémentation

Terminaison

Correction

Tri par insertion

Preuve de terminaison

i reuve de terminaison

Preuve de correction

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

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

```
____
```

rier des carte anuellement

Transposer au tri de données

Fri par sélection Implémentation

Terminaison

Complexité

Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

```
 \begin{array}{c|c} \textbf{1} & \text{while pos} > 0 \text{ and en\_cours} < \mathsf{tab[pos-1]}: \\ \textbf{2} & \mathsf{tab[pos]} = \mathsf{tab[pos-1]} \\ \textbf{3} & \mathsf{pos} = \mathsf{pos}{-1} \\ \end{array}
```

pos est un variant de la boucle.

Problématique

rier des cartes anuellement

Fransposer au tri le données

Iri par selection Implémentation

erminaison

Complexité

Implémentation

Preuve de terminaison

Preuve de correction

69 / 75

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

3	6	8	4	5
---	---	---	---	---

Code 10 - Insertion de l'élément 4

4



4





Code 11 - Insertion de l'élément 4

Problématique

rier des cartes

Transposer au tr de données

l'ri par sélection Implémentation

C----ti--

Complexité

Implémentation

Preuve de terminaison Preuve de correction

Complexité

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 12 - Insertion de l'élément 4

Trier des cartes

Problématique

rier des cartes

Transposer au tri de données

ri par sélection mplémentation

rminaison

Complexité

mplémentation

Preuve de terminaison

Complexité

robiematique

rier des cartes anuellement

de données

Iri par selection Implémentation

Correction

Complexité

I I/ . . .

Preuve de terminaison

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.

Trier of

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

1 4 5 7 8

Code 13 – Le tableau est déjà trié. La boucle interne n'effectue aucune itération.

Problématique

rier des cartes anuellement

ransposer au tri e données

l'ri par sélection

Terminaison

Correction

Tri par insertion

Implémentation

Preuve de terminaison

Complexité

```
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 14 – Le tableau est inversé. La boucle interne effectue i itérations.

Problématique

er des carte nuellement

Transposer au tri de données

Fri par sélection Implémentation

orrection

Complexité
Tri par insertion

Implémentation

Preuve de correction

Complexité

Problèmatique

rier des cartes lanuellement

Transposer au tri de données

Tri par sélection Implémentation

Terminaison Correction

Complexité

Implémentation

Preuve de terminaison

Complexité

À retenir

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