Problématique

Frier des cartes nanuellement

Transposer au tri de données

Tri par sélection Implémentation

rerminaisor

COLLECTION

Tri par insertio

Implémentation

Preuve de terminaison

Complexité

Trier des cartes

Christophe Viroulaud

Première NSI

Problématique

rier des cartes nanuellement

Transposer au tri de données

Irri par selection Implémentation

Correction

Complexité

Implémentation

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?

roblématique

Trier des cartes manuellement

de données

Tri par sélection

Terminaison

Correction

Complexité

i par insertion

Preuve de terminaison

Preuve de correction



FIGURE – Cartes mélangées

Problématique

Trier des cartes manuellement

de données

Implémentation

Correction

Complexité

Tri par insertion

Preuve de terminaison

Preuve de correction

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

Correction

Complexité

Implémentation

Drougo do torminaison

Prouve de cerminaison

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

Trier des cartes

Problématique

Trier des cartes manuellement

Transposer au tri de données

Tri par sélection Implémentation

Correction

Complexité

Implémentation

Preuve de terminaison

Preuve de correction

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.

Code 1 – Tri par sélection (en place)

Retour menu

1

3

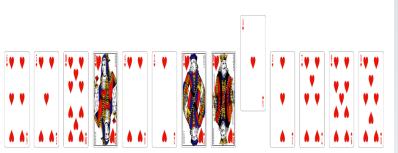


FIGURE - Sélectionne la plus petite du tas non trié

Retour menu

Problématique

Trier des cartes manuellement

Transposer au tri de données

> ri par sélection mplémentation

Terminaison

Correction

ri par insertion

Implémentation

Preuve de terminaisor

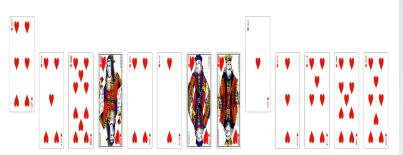


FIGURE - Échange avec la première carte du tas non trié

Retour menu

Problématique

Trier des cartes manuellement

Transposer au tri de données

> ri par sélection Implémentation

Terminaison Correction

Complexité Fri par insertion

Implémentation

Preuve de terminaison
Preuve de correction
Complexité

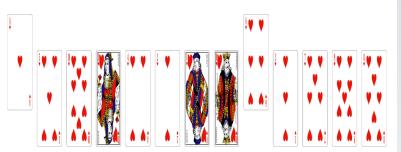


FIGURE - Échange avec la première carte du tas non trié

Retour menu

Problématique

Trier des cartes manuellement

Transposer au tri de données

> ri par sélection mplémentation

Terminaison Correction

Complexité Tri par insertion

Tri par insertion Implémentation

Preuve de terminaison
Preuve de correction



FIGURE – La carte est à sa place

Retour menu

roblématique

Trier des cartes manuellement

Transposer au tri

Tri par sélection

Implémentation Terminairon

Terminaison

Complexité

Tri par insertion

Preuve de terminaison

Preuve de terminaison
Preuve de correction

mplexité



FIGURE - Sélectionne la plus petite du tas non trié

Retour menu

roblématique

Trier des cartes manuellement

Transposer au tri de données

ri par sélection

mplémentation

Correction

Complexité

Tri par insertion

Preuve de terminaison
Preuve de correction

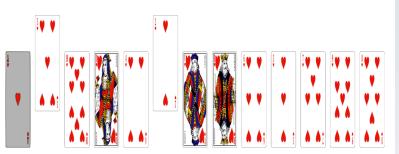


FIGURE - Échange avec la première carte du tas non trié

Retour menu

Problématique

Trier des cartes manuellement

Transposer au tri de données

> ri par sélection mplémentation

Terminaison

Correction Complexité

Complexité Tri par insertion

Implémentation
Preuve de terminaisor

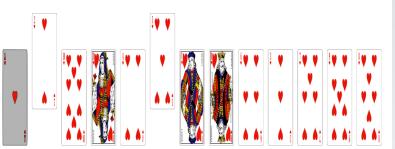


FIGURE - Échange avec la première carte du tas non trié

Retour menu

roblématique

Trier des cartes manuellement

Transposer au tri de données

> ri par sélection mplémentation

Terminaison

Correction Complexité

Tri par insertion

Preuve de terminaison



FIGURE – La carte est à sa place

Retour menu

roblématique

Trier des cartes manuellement

Transposer au tri

Tri par sélection

Implémentation Terminaison

Terminaison Correction

Complexité

Tri par insertion

Preuve de terminaison

Preuve de correction

Pour chaque carte du tas

3

Retour menu

Trouver la plus petite carte du tableau non trié. La placer à la fin du tableau trié.

Code 2 – Tri par sélection dans un nouveau tableau

'

Trier des cartes

Problématique |

Trier des cartes manuellement

Fransposer au tri le données

ri par sélection Implémentation

orrection

Complexité

Implémentation

Preuve de terminaisor

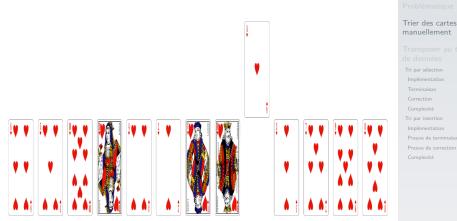


FIGURE - Trouve la plus petite du tas non trié

Retour menu

16 / 70

manuellement

Tri par sélection - nouveau tableau

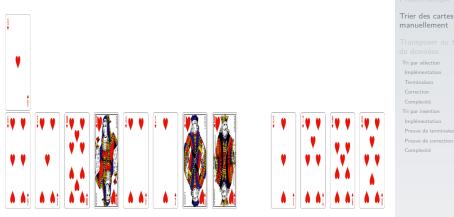
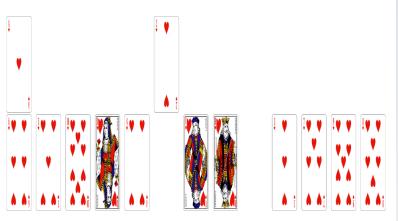


FIGURE - La place à la fin du tableau trié

Trier des cartes manuellement

Tri par sélection - nouveau tableau



 $\label{eq:figure} Figure - \text{Trouve la plus petite du tas non tri\'e}$

Trier des cartes

roblématique

Trier des cartes manuellement

Transposer au tri de données

Tri par sélection

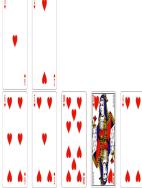
Terminaison

Correction

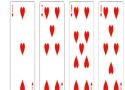
Complexité Fri par insertion

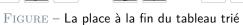
Implémentation
Preuve de terminaisor

Preuve de correction
Complexité









Problématique

Trier des cartes manuellement

Transposer au tri

Tri par sélection Implémentation

Terminaison

orrection omplexité

ri par insertion

Preuve de terminaison

Complexité

1	Pour	cha	que	carte	du	tas

2 Mémoriser la carte en cours

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

Insérer la carte en cours dans l'espace vide.

Code 3 – Tri par insertion (en place)

Retour menu

3

4

Trier des cartes manuellement

Tri par insertion en place

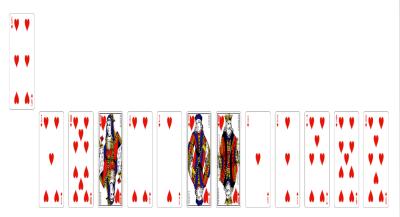


FIGURE - Mémoriser la première carte dans le tas non trié

Retour menu

21 / 70

Trier des cartes manuellement

Tri par insertion en place

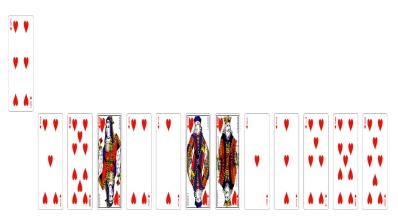


FIGURE - Décaler les cartes supérieures déjà triées

Retour menu

22 / 70



FIGURE - Replacer la carte dans l'espace

Retour menu

roblématique

Trier des cartes manuellement

Transposer au tri

Tri par sélection

Terminaison

Correction

Complexité

Tri par insertion Implémentation

Preuve de terminaison

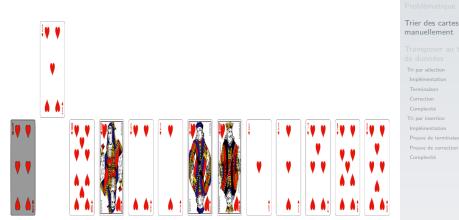


FIGURE - Mémoriser la première carte dans le tas non trié

Retour menu

24 / 70

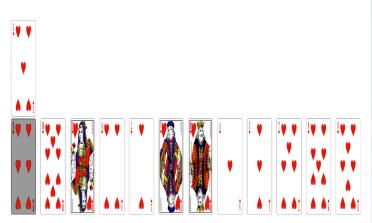


FIGURE - Décaler les cartes supérieures déjà triées

Retour menu

Trier des cartes manuellement

ransposer au tri

Fri par sélection
Implémentation

erminaison orrection omplexité

Implémentation
Preuve de terminaison
Preuve de correction

Preuve de terminaison
Preuve de correction
Complexité



FIGURE - 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 Implémentation

Preuve de terminaison

Trier des cartes

oblématique

Trier des cartes manuellement

Transposer au tri de données

ri par sélection Implémentation

mplementation Ferminaison

Correction

Complexité

Iri par ins

Preuve de terminaison

Preuve de correc Complexité

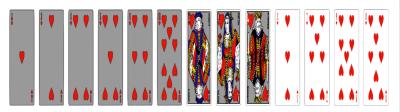


FIGURE – Après plusieurs itérations

Trier des cartes

Problématique

Trier des cartes manuellement

Fransposer au tri le données

ri par sélection Implémentation

l'erminaison Correction Complexité

Tri par inserti

Preuve de terminais

Preuve de correct Complexité

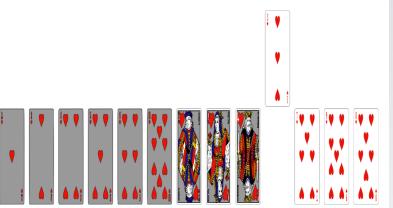


FIGURE - Mémoriser la première carte dans le tas non trié

Trier des cartes

Problématique

Trier des cartes manuellement

ransposer au tri e données

ri par sélection Implémentation

Terminaison Correction Complexité

Tri par inserti

Preuve de terminais

Preuve de corre Complexité

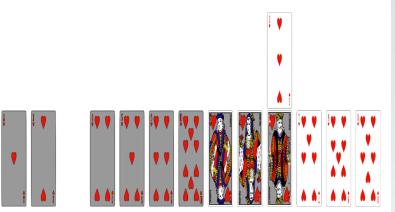


FIGURE - Décaler les cartes supérieures déjà triées

Trier des cartes

roblématique

Trier des cartes manuellement

Fransposer au tri de données

ri par sélection mplémentation

mplémentation Ferminaison

Correction

Complexité

Tri par ins

Preuve de terminaisor

Preuve de correcti Complexité

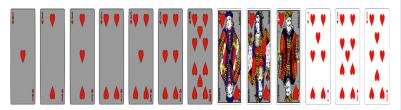


FIGURE – Insérer la carte dans l'espace

Problématique

Trier des cartes manuellement

Transposer au tri de données

Tri par sélection Implémentation

Terminaison Correction

Complexité

Implémentation

Preuve de terminaison Preuve de correction

Complexité

Pour chaque carte du tas

Prendre la première carte du tableau non trié.

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

Insérer la carte dans le tableau trié.

Code 4 - Tri par insertion dans un nouveau tableau

Retour menu

1

2

3

4

Trier des cartes

roblématique

Trier des cartes manuellement

Transposer au tri de données

ri par sélection Implémentation

Terminaison

Correction Complexité

Tri par insert

Implementation
Preuve de terminaisor

Preuve de c Complexité

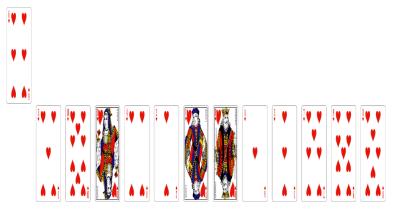


FIGURE - Prendre la première carte non triée

Trier des cartes

roblématique

Trier des cartes manuellement

Transposer au tri de données

Tri par sélection

Implémentation

Correction

Complexité

Implémentation

Preuve de terminaison
Preuve de correction
Complexité

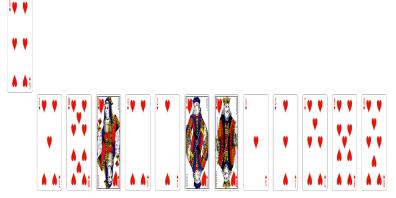


FIGURE - Décaler les cartes supérieures du tableau trié



FIGURE – 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

Correction

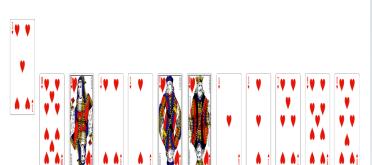
Complexité

Implémentation

Preuve de terminaisor

mplexité

! V V



Trier des cartes

Problématique

Trier des cartes manuellement

Transposer au tri

Tri par sélection

Terminaison

Correction

Complexité

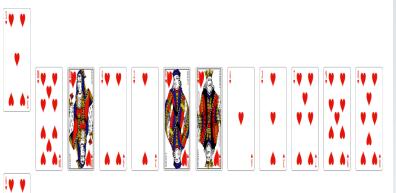
Implémentation

Preuve de terminaison

Complexité

FIGURE - Prendre la première carte non triée

Retour menu 35/70



Problematique

Trier des cartes manuellement

Transposer au tri de données

Tri par sélection

Terminaison

Correction

Complexité

Tri par insertion

Implémentation

Preuve de terminaison Preuve de correction

Complexité

FIGURE – Décaler les cartes supérieures du tableau trié

Tri par insertion - nouveau tableau





FIGURE – Insérer la carte dans le tableau triée

Retour menu

Trier des cartes

Problématique

Trier des cartes manuellement

Transposer au tri

Tri par sélection

Terminaison

orrection

ri par insertion

Implémentation Preuve de terminaisc

> euve de correction omplexité

Quand on passe un tableau en argument à une fonction, cette dernière manipule le tableau original.

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

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

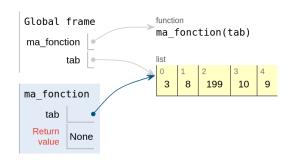


 Figure – La fonction modifie le tableau original

Trier des cartes

robiematique

Trier des cartes

Transposer au tri de données

par sélection
plémentation
rminaison
prection
pmplexité
par insertion
plémentation

reuve de correc

Problématique

ier des carte anuellement

Transposer au tr de données

Tri par sélection

Implémentation

Terminaison

Complexité

in par insertion

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
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

Complexité

Tri par insertion

D 1

Preuve de terminaison

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

Transposer au tri de données

Tri par sélection Implémentation

Implémentation

Correction

Complexité l'ri par insertion

Implémentation
Preuve de terminaison

Preuve de correct

```
def trouver_mini(tab: list , i_depart: int ) -> int:
 1
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
10
        return i_mini
```

Problématique

rier des cartes anuellement

ransposer au tri e données

Tri par sélection

Implémentation

*-----

Lorrection

ri par insertion

Implémentation

Preuve de terminaison

Correction

Trier des cartes

Problématique

rier des cartes nanuellement

Transposer au tri de données

Tri par sélection

Implémentation

°orraction

Correction

Tri par insertion

Implémentation

Preuve de terminaison

```
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]
```

Problématique

rier des cartes

Fransposer au tri le données

Iri par selection

Implémentation

Terminaison

Comployitá

Tri par insertion

Implémentation

Preuve de terminaison

mnlovitá

Froblematique

ier des carte anuellement

Transposer au tri de données

Tri par sélectio

Implémentation

Correction

Complexité

Fri nar insertion

Implémentation

Preuve de terminaison
Preuve de correction

Complexité

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.

Correction

```
Trier des cartes
```

```
Problématique
```

rier des cartes anuellement

Transposer au tri le données

```
Iri par selection
```

```
Implémentation
```

```
Terminaison
```

Correction

Tri nor insortion

Implémentation

Preuve de terminaison

euve de correction

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

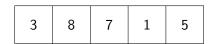
Preuve de terminaison : variant de boucle

Trier des cartes

Terminaison

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

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



Code 5 – Avant la première itération, la partie gauche est vide, donc triée.

Problématique

er des carte nuellement

Transposer au tri

ri par sélection Implémentation

Correction

Complexité
Tri par insertion

Iri par insertion

Preuve de terminaison

Preuve de correction

Preuve de correction : invariant de boucle



Code 6 – 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

Implémentation Terminaison

Correction

Complexité

Tri par insertion

Preuve de terminaison

Preuve de terminaison

Preuve de correction : invariant de boucle



Code 7 – Avant la troisième 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 incortion

Tri par insertion

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

mplémentation

Complexité

Lomplexite

Tri par insertio

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 .

Problématique

rier des cartes anuellement

Fransposer au tri le données

Implémentation

Complexité

Tri nor incortion

Tri par insertion

Preuve de terminaison

Problématiqu

ier des cartes

Transposer au tri de données

ri par sélection

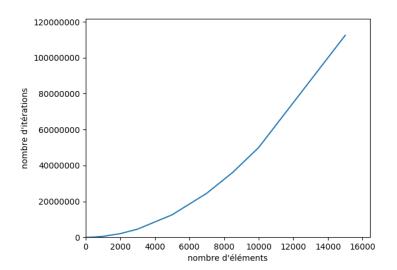
rminaison

Complexité

ri par insertion

Preuve de terminaison

reuve de co



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.

Problematique

manuellement

de données

Tri par sélection Implémentation

Correction

Complexité Tri par insertion

Implémentation

Rappel de l'algorithme

1

3

4

Trier des cartes

Implémentation

Pour chaque carte du tas						
Mémoriser la carte en cours						
Décaler vers la droite toutes les cartes précé						
dentes, supé rieures à la carte en cours.						
Insérer la carte en cours dans l'espace vide						

Code 8 - Tri par insertion (en place)

Correction: boucle principale

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

tri le tableau dans l'ordre croissant

r""

for i in range(len(tab)):
```

Trier des cartes

Problématique

Trier des cartes

Fransposer au tri de données

Tri par sélection Implémentation

Terminaison

Lorrection

ar insertion

Implémentation

Correction: mémoriser

Trier des cartes

Problématique

Frier des cartes nanuellement

Transposer au tri de données

Tri par sélection Implémentation

Terminaison

orrection omployité

par insertion

Implémentation

rier des cartes anuellement

Transposer au tri de données

ri par sélection Implémentation

erminaison

orrection

mplexité

Implémentation

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

Correction: insérer

1

Trier des cartes

Implémentation

 $tab[pos] = en_cours$

```
def tri insertion (tab: list ) -> None:
 1
2
        tri le tableau dans l'ordre croissant
3
 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

rier des cartes

Fransposer au tri de données

i par sélection mplémentation

orrection

par insertion

Implémentation

Transposer au tr de données

ri par sélection

Implémentation

Correction

omplexité

par insertion

Implémentation

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

Trier des cartes

Problématique

rier des cartes anuellement

Transposer au tri de données

Tri par sélection Implémentation

Terminaison

Comployitá

Tri par insertion

Preuve 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.

Correction

1

3

```
while pos > 0 and en_cours < tab[pos-1]:
      tab[pos] = tab[pos-1]
2
```

pos est un variant de la boucle.

pos = pos - 1

Trier des cartes

Preuve de terminaison

Trier des cartes

Problématique

rier des cartes anuellement

Fransposer au tri le données

Implémentation

°ormetion

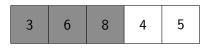
Complexité

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 9 - Insertion de l'élément 4

Trier des cartes

4



4





Code 10 - Insertion de l'élément 4

Problématique

rier des cartes

Transposer au tri de données

l'ri par sélection

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 11 - Insertion de l'élément 4

Trier des cartes

Problématique

rier des cartes

Fransposer au tri le données

> par sélection oplémentation

orrection

i par insertion

Preuve de terminaison

Trier des cartes

robiciliatique

ier des cartes anuellement

ransposer au tr e données

Implémentation

Correction

Complexité

Implémentation

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.

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

1 4 5 7 8

Code 12 – Le tableau est déjà trié. La boucle interne n'effectue aucune itération.

Problématique

ier des cart anuellement

ransposer au tr e données

ri par sélection Implémentation

Correction

Complexité

Implémentation

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

Problématique

ier des cart anuellement

ransposer au tri e données

Irri par selection Implémentation

orrection

Complexité Tri par insertion

Implémentation

Preuve de terminaison

Preuve de terminaison
Preuve de correction

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

Implémentation

Correction

Complexité

Tri par insertion

Preuve de terminaison

Preuve de correction