

Problématique

Trier des cartes
manuellement

Transposer au tri
de données

Tri par sélection

Implémentation

Terminaison

Correction

Complexité

Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

Complexité

Trier des cartes

Christophe Viroulaud

Première - NSI

1. Problématique

2. Trier des cartes manuellement

3. Transposer au tri de données

Problématique

Trier des cartes
manuellement

Transposer au tri
de données

Tri par sélection

Implémentation

Terminaison

Correction

Complexité

Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

Complexité

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

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

Problématique

Trier des cartes
manuellement

Transposer au tri
de données

Tri par sélection

Implémentation

Terminaison

Correction

Complexité

Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

Complexité

1. Problématique

2. Trier des cartes manuellement

3. Transposer au tri de données

Problématique

Trier des cartes manuellement

Transposer au tri de données

Tri par sélection

Implémentation

Terminaison

Correction

Complexité

Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

Complexité

Problématique

Trier des cartes
manuellement

Transposer au tri
de données

- Tri par sélection
- Implémentation
- Terminaison
- Correction
- Complexité
- Tri par insertion
- Implémentation
- Preuve de terminaison
- Preuve de correction
- Complexité



FIGURE 1 – Cartes mélangées

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.

Problématique

Trier des cartes
manuellement

Transposer au tri
de données

Tri par sélection

Implémentation

Terminaison

Correction

Complexité

Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

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

Tri par sélection en place

Problématique

Trier des cartes
manuellementTransposer au tri
de données

Tri par sélection

Implémentation

Terminaison

Correction

Complexité

Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

Complexité

- 1 Pour chaque carte du tas
- 2 Trouver la plus petite carte dans la partie non triée.
- 3 Échanger cette carte avec la première de la partie non triée.
- 4

Code 1 – Tri par sélection (en place)

[Retour menu](#)

Tri par sélection en place

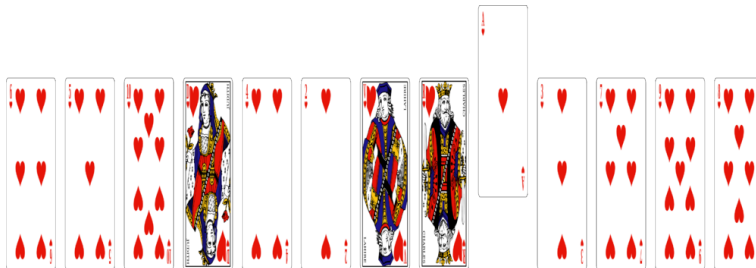


FIGURE 2 – Sélectionne la plus petite du tas non trié

[Retour menu](#)

Tri par sélection en place

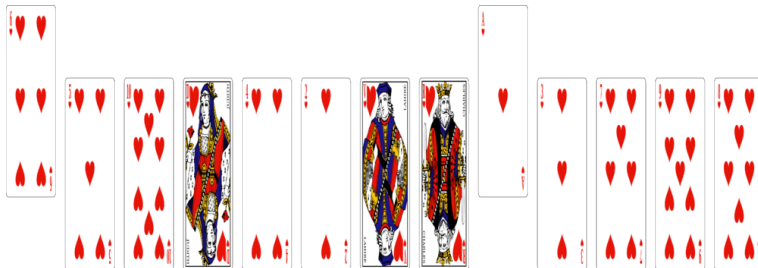


FIGURE 3 – Échange avec la première carte du tas non trié

[Retour menu](#)

Tri par sélection en place

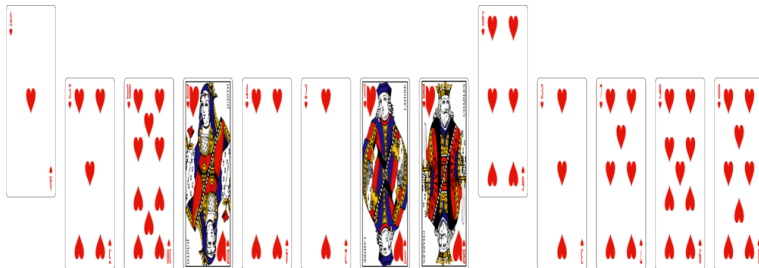


FIGURE 4 – Échange avec la première carte du tas non trié

[Retour menu](#)

Tri par sélection en place

Trier des cartes

Problématique

Trier des cartes
manuellement

Transposer au tri
de données

Tri par sélection

Implémentation

Terminaison

Correction

Complexité

Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

Complexité



FIGURE 5 – La carte est à sa place

[Retour menu](#)

Tri par sélection en place

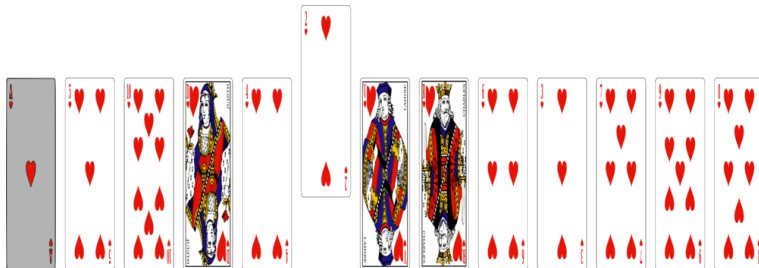


FIGURE 6 – Sélectionne la plus petite du tas non trié

[Retour menu](#)

Tri par sélection en place



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

[Retour menu](#)

Tri par sélection en place

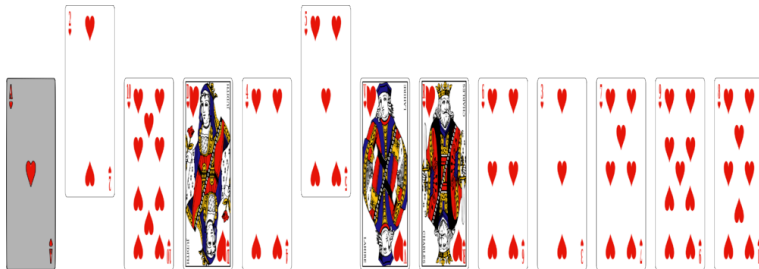


FIGURE 8 – Échange avec la première carte du tas non trié

[Retour menu](#)

Tri par sélection en place

Trier des cartes

Problématique

Trier des cartes
manuellement

Transposer au tri
de données

Tri par sélection

Implémentation

Terminaison

Correction

Complexité

Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

Complexité

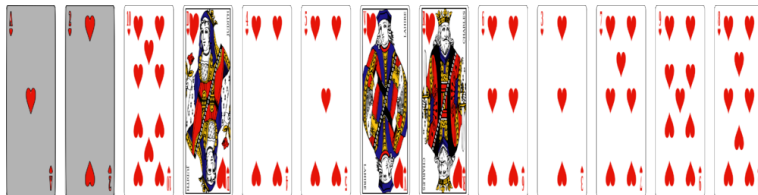


FIGURE 9 – La carte est à sa place

[Retour menu](#)

- 1 Pour chaque carte du tas
- 2 Trouver la plus petite carte du tableau non trié.
- 3 La placer à la fin du tableau trié.
- 4

Code 2 – Tri par sélection dans un nouveau tableau

[Retour menu](#)

Tri par sélection - nouveau tableau

Trier des cartes

Problématique

Trier des cartes
manuellement

Transposer au tri
de données

Tri par sélection

Implémentation

Terminaison

Correction

Complexité

Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

Complexité

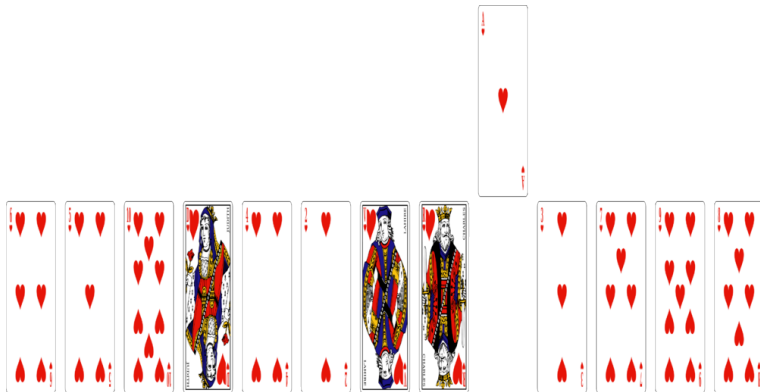


FIGURE 10 – Trouve la plus petite du tas non trié

[Retour menu](#)

Tri par sélection - nouveau tableau

Trier des cartes

Problématique

Trier des cartes
manuellement

Transposer au tri
de données

Tri par sélection

Implémentation

Terminaison

Correction

Complexité

Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

Complexité

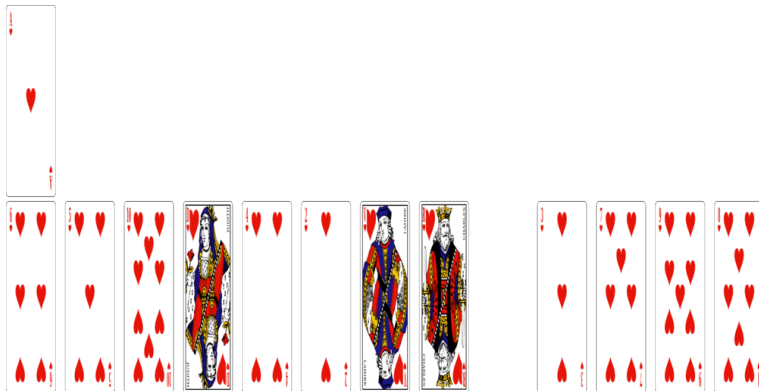


FIGURE 11 – La place à la fin du tableau trié

[Retour menu](#)

Tri par sélection - nouveau tableau

Trier des cartes

Problématique

Trier des cartes
manuellement

Transposer au tri
de données

Tri par sélection

Implémentation

Terminaison

Correction

Complexité

Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

Complexité

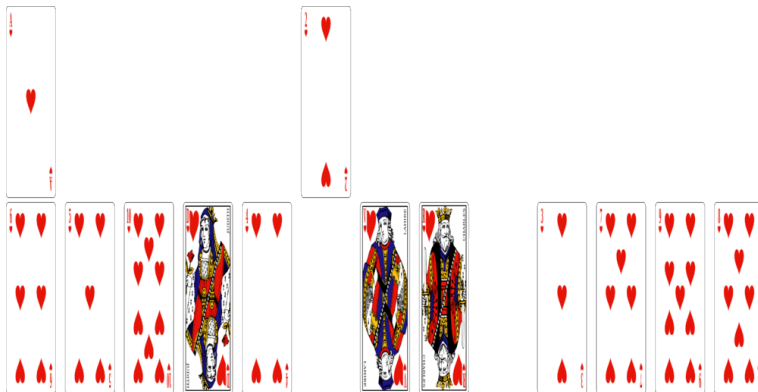
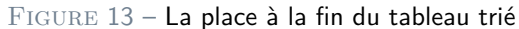


FIGURE 12 – Trouve la plus petite du tas non trié

[Retour menu](#)



21 / 76

Tri par insertion en place

Problématique

Trier des cartes
manuellementTransposer au tri
de données

Tri par sélection

Implémentation

Terminaison

Correction

Complexité

Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

Complexité

- 1 Pour chaque carte du tas
- 2 Mémoriser la carte en cours
- 3 Décaler vers la droite toutes les cartes précédentes,
 supérieures à la carte en cours.
- 4 Insérer la carte en cours dans l'espace vide.
- 5

Code 3 – Tri par insertion (en place)

[Retour menu](#)

Tri par insertion en place

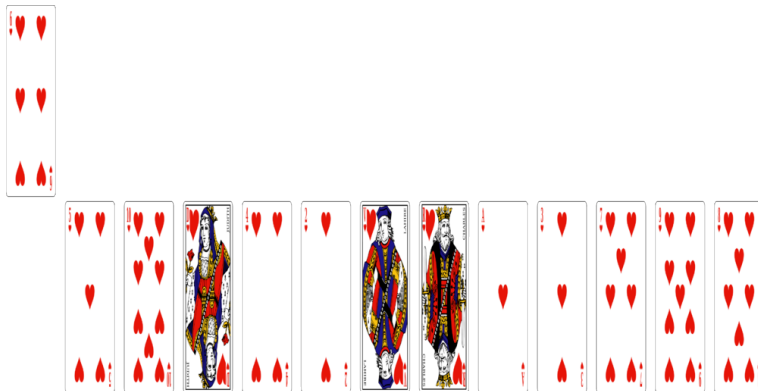


FIGURE 14 – Mémoriser la première carte dans le tas non trié

Tri par insertion en place

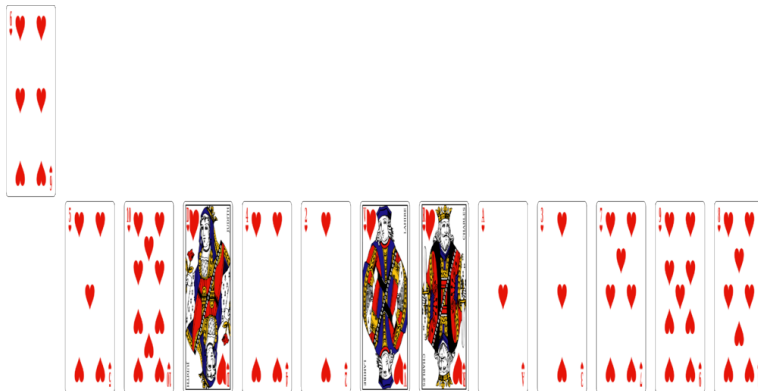


FIGURE 15 – Décaler les cartes supérieures déjà triées

Tri par insertion en place

Trier des cartes

Problématique

Trier des cartes
manuellement

Transposer au tri
de données

Tri par sélection

Implémentation

Terminaison

Correction

Complexité

Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

Complexité

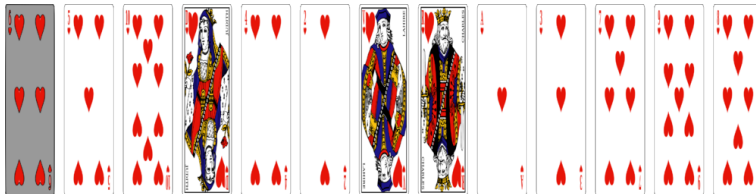


FIGURE 16 – Replacer la carte dans l'espace

[Retour menu](#)

Tri par insertion en place

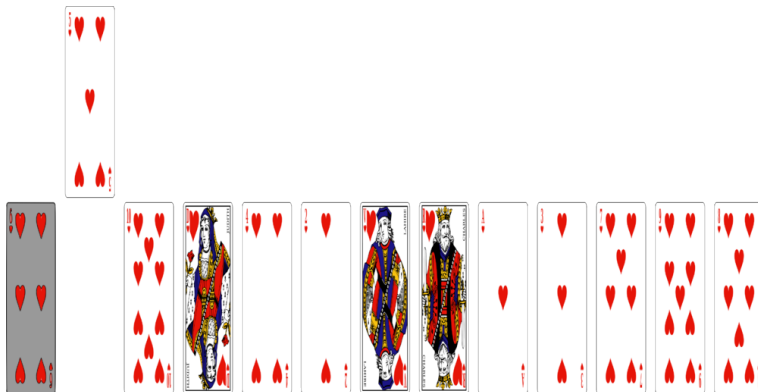


FIGURE 17 – Mémoriser la première carte dans le tas non trié

Tri par insertion en place

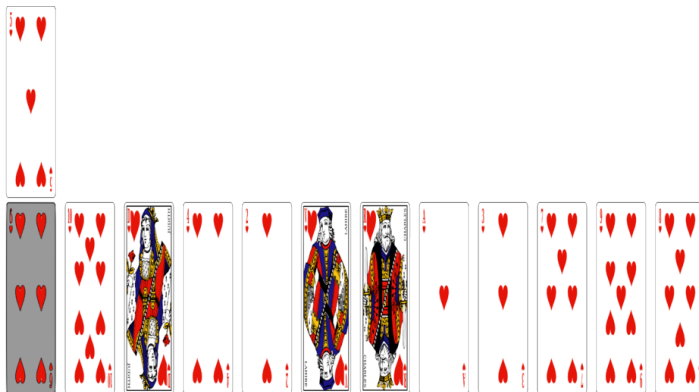


FIGURE 18 – Décaler les cartes supérieures déjà triées

Tri par insertion en place

Trier des cartes

Problématique

Trier des cartes
manuellement

Transposer au tri
de données

Tri par sélection

Implémentation

Terminaison

Correction

Complexité

Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

Complexité



FIGURE 19 – Replacer la carte dans l'espace

[Retour menu](#)

Tri par insertion en place

Trier des cartes

Problématique

Trier des cartes
manuellement

Transposer au tri
de données

Tri par sélection

Implémentation

Terminaison

Correction

Complexité

Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

Complexité

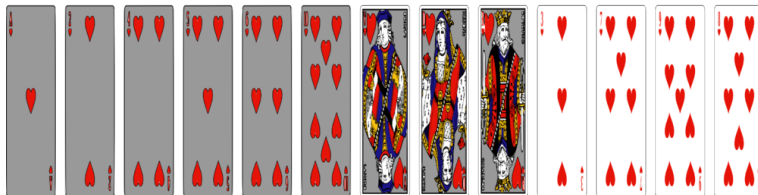


FIGURE 20 – Après plusieurs itérations

[Retour menu](#)

Tri par insertion en place

Trier des cartes

Problématique

Trier des cartes
manuellement

Transposer au tri
de données

Tri par sélection

Implémentation

Terminaison

Correction

Complexité

Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

Complexité

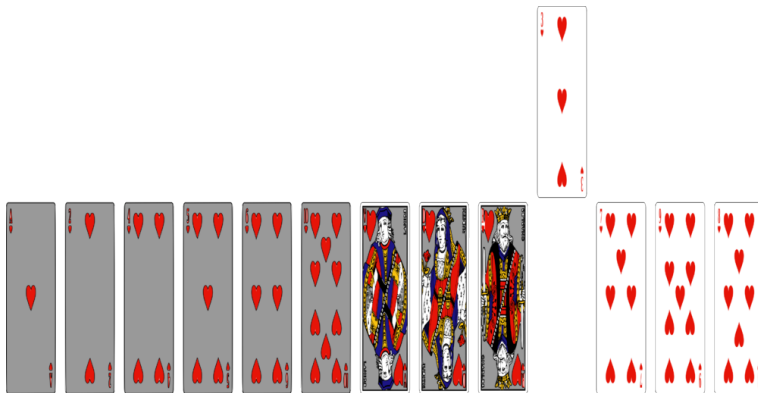


FIGURE 21 – Mémoriser la première carte dans le tas non trié

[Retour menu](#)

Tri par insertion en place

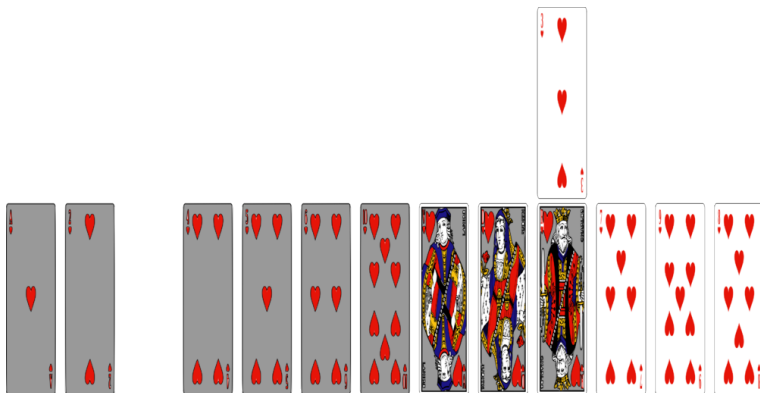


FIGURE 22 – Décaler les cartes supérieures déjà triées

Tri par insertion en place

Trier des cartes

Problématique

Trier des cartes
manuellement

Transposer au tri
de données

Tri par sélection

Implémentation

Terminaison

Correction

Complexité

Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

Complexité

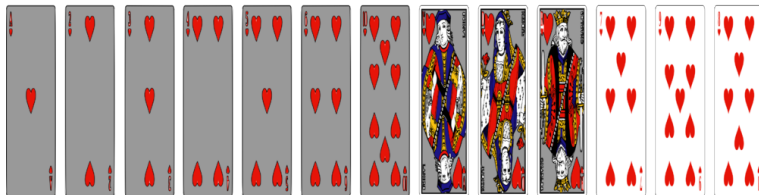


FIGURE 23 – Insérer la carte dans l'espace

[Retour menu](#)

- 1 Pour chaque carte du tas
- 2 Prendre la première carte du tableau non trié.
- 3 Dans le tableau trié, décaler vers la droite toutes les
 cartes plus grandes.
- 4 Insérer la carte dans le tableau trié.
- 5

Code 4 – Tri par insertion dans un nouveau tableau

[Retour menu](#)

Tri par insertion - nouveau tableau

Trier des cartes

Problématique

Trier des cartes
manuellement

Transposer au tri
de données

Tri par sélection

Implémentation

Terminaison

Correction

Complexité

Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

Complexité

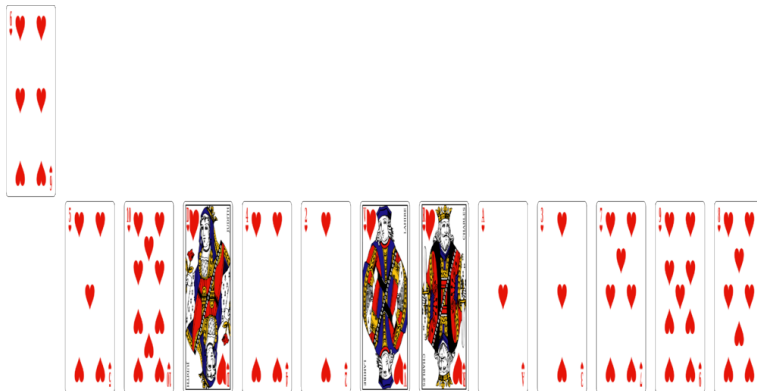


FIGURE 24 – Prendre la première carte non triée

[Retour menu](#)

Tri par insertion - nouveau tableau

Trier des cartes

Problématique

Trier des cartes
manuellement

Transposer au tri
de données

Tri par sélection

Implémentation

Terminaison

Correction

Complexité

Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

Complexité

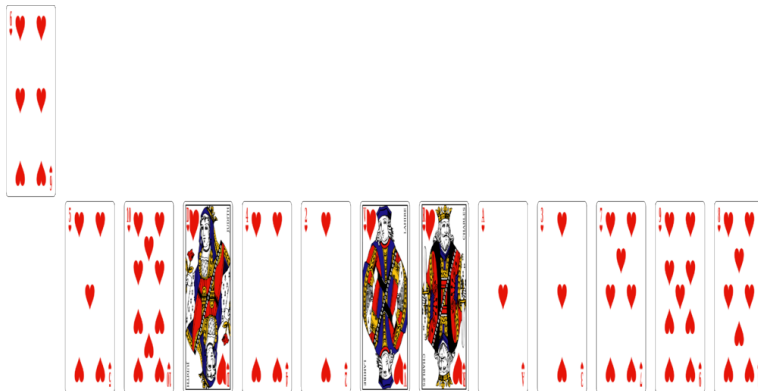


FIGURE 25 – Décaler les cartes supérieures du tableau trié

[Retour menu](#)

Tri par insertion - nouveau tableau

Trier des cartes

Problématique

Trier des cartes
manuellement

Transposer au tri
de données

Tri par sélection

Implémentation

Terminaison

Correction

Complexité

Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

Complexité

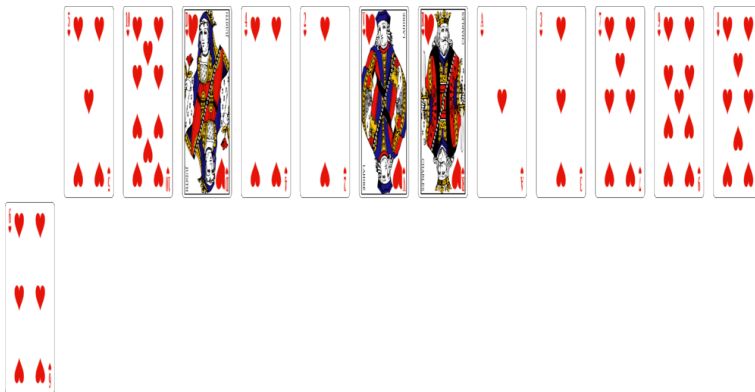


FIGURE 26 – Insérer la carte dans le tableau triée

[Retour menu](#)

Tri par insertion - nouveau tableau

Trier des cartes

Problématique

Trier des cartes
manuellement

Transposer au tri
de données

Tri par sélection

Implémentation

Terminaison

Correction

Complexité

Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

Complexité

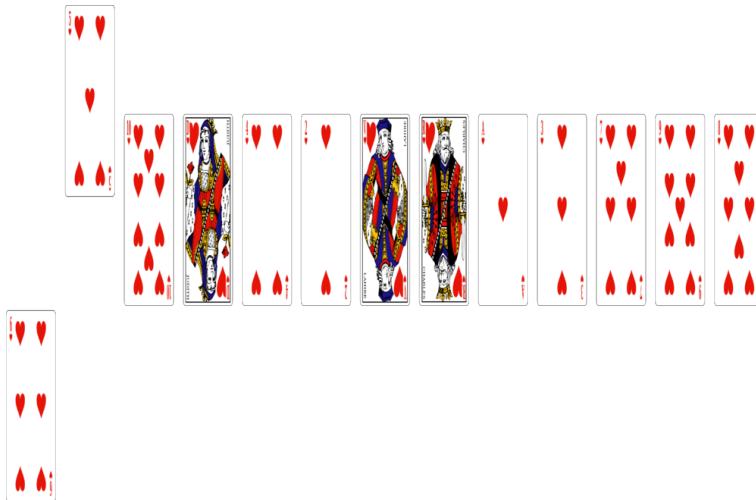


FIGURE 27 – Prendre la première carte non triée

Tri par insertion - nouveau tableau

Trier des cartes

Problématique

Trier des cartes
manuellement

Transposer au tri
de données

- Tri par sélection
- Implémentation
- Terminaison
- Correction
- Complexité
- Tri par insertion
- Implémentation
- Preuve de terminaison
- Preuve de correction
- Complexité

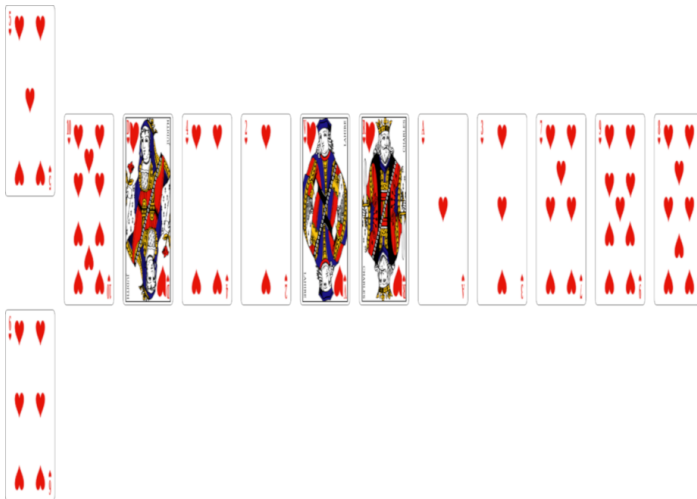


FIGURE 28 – Décaler les cartes supérieures du tableau trié

Tri par insertion - nouveau tableau

Trier des cartes

Problématique

Trier des cartes
manuellement

Transposer au tri
de données

Tri par sélection

Implémentation

Terminaison

Correction

Complexité

Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

Complexité

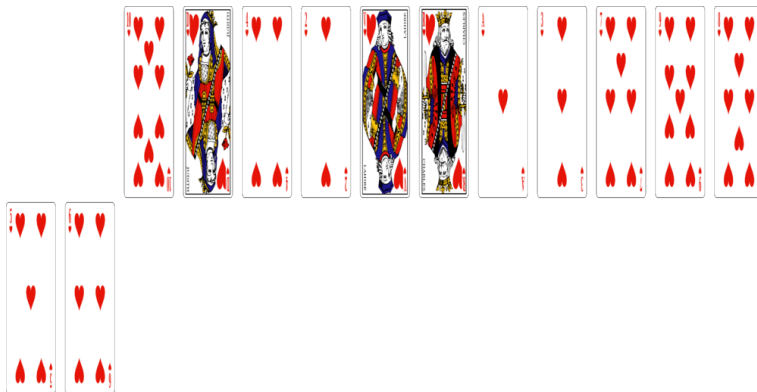


FIGURE 29 – Insérer la carte dans le tableau triée

[Retour menu](#)

1. Problématique

2. Trier des cartes manuellement

3. Transposer au tri de données

3.1 Tri par sélection

3.2 Tri par insertion

Problématique

Trier des cartes
manuellement

Transposer au tri
de données

Tri par sélection

Implémentation

Terminaison

Correction

Complexité

Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

Complexité

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

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

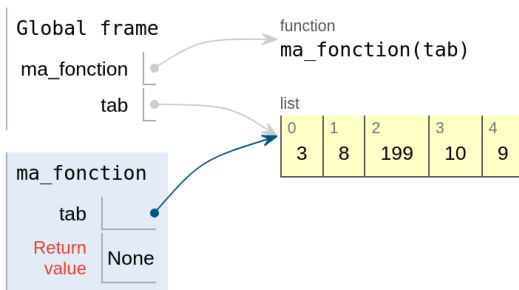


FIGURE 30 – La fonction modifie le tableau original

Problématique

Trier des cartes
manuellement

Transposer au tri
de données

Tri par sélection

Implémentation

Terminaison

Correction

Complexité

Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

Complexité

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

Code 5 – Tri par sélection (en place)

Problématique

Trier des cartes
manuellement

Transposer au tri
de données

Tri par sélection

Implémentation

Terminaison

Correction

Complexité

Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

Complexité

Activité 2 :

1. Écrire la fonction **trouver_mini(tab : list) → int** qui renvoie l'indice du plus petit élément de *tab*.

Problématique

Trier des cartes
manuellement

Transposer au tri
de données

Tri par sélection

Implémentation

Terminaison

Correction

Complexité

Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

Complexité

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

Activité 2 :

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**.
3. Écrire la fonction **echanger(tab : list, i : int, j : int) → None** qui échange les éléments d'indice *i* et *j* du tableau *tab*.
4. Écrire la fonction **tri_selection(tab : list) → None** qui effectue un tri par sélection sur *tab*.

Problématique

Trier des cartes
manuellement

Transposer au tri
de données

Tri par sélection

Implémentation

Terminaison

Correction

Complexité

Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

Complexité

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

Problématique

Trier des cartes
manuellement

Transposer au tri
de données

Tri par sélection

Implémentation

Terminaison

Correction

Complexité

Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

Complexité

```
1 def echanger(tab: list, i: int, j: int) -> None:
2     """
3     inverse les élément d'indices i et j du tableau
4     """
5     tab[i], tab[j] = tab[j], tab[i]
```

Problématique

Trier des cartes
manuellement

Transposer au tri
de données

Tri par sélection

Implémentation

Terminaison

Correction

Complexité

Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

Complexité

```
1 def tri_selection(tab: list) -> None:
2     """
3     tri le tableau dans l'ordre croissant
4     """
5     for i in range(len(tab)):
6         position_du_mini = trouver_mini(tab, i)
7         echanger(tab, i, position_du_mini)
```


Problématique

Trier des cartes
manuellement

Transposer au tri
de données

Tri par sélection

Implémentation

Terminaison

Correction

Complexité

Tri par 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*.

Problématique

Trier des cartes
manuellement

Transposer au tri
de données

Tri par sélection

Implémentation

Terminaison

Correction

Complexité

Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

Complexité

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

Problématique

Trier des cartes
manuellement

Transposer au tri
de données

Tri par sélection

Implémentation

Terminaison

Correction

Complexité

Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

Complexité

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

Preuve de correction : invariant de boucle

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

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

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

Preuve de correction : invariant de boucle

Trier des cartes

Problématique

Trier des cartes
manuellement

Transposer au tri
de données

Tri par sélection

Implémentation

Terminaison

Correction

Complexité

Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

Complexité

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

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

Preuve de correction : invariant de boucle

Trier des cartes

Problématique

Trier des cartes
manuellement

Transposer au tri
de données

Tri par sélection

Implémentation

Terminaison

Correction

Complexité

Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

Complexité

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

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

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
---	---	---	---	---

- ▶ ...

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

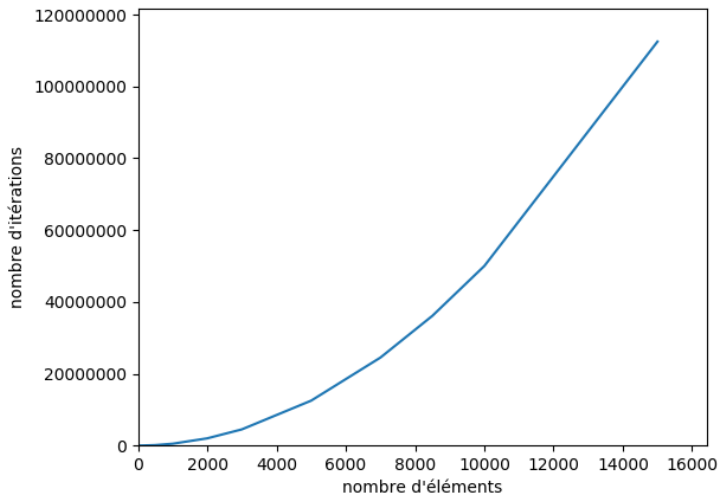
À retenir

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

Le nombre d'opérations dépend de n^2 .

Évolution du nombre d'itérations

Trier des cartes



Problématique

Trier des cartes
manuellement

Transposer au tri
de données

Tri par sélection

Implémentation

Terminaison

Correction

Complexité

Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

Complexité

- 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
manuellement

Transposer au tri
de données

Tri par sélection

Implémentation

Terminaison

Correction

Complexité

Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

Complexité

Activité 3 :

1. 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.
 - ▶ Décaler : 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

Implémentation

Terminaison

Correction

Complexité

Tri par insertion

Implémentation

Preuve de terminaison

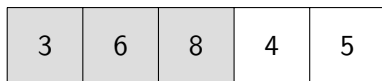
Preuve de correction

Complexité

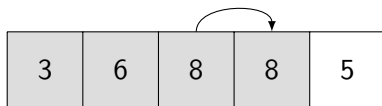
- 1 Pour chaque carte du tas
- 2 Mémoriser la carte en cours
- 3 Décaler vers la droite toutes les cartes précédentes,
 supérieures à la carte en cours.
- 4 Insérer la carte en cours dans l'espace vide.

Code 9 – Tri par insertion (en place)

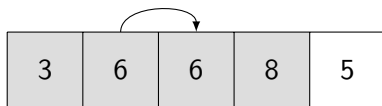
Subtilité sur le tableau Python



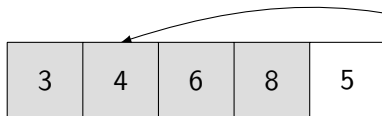
en_cours = 4



en_cours = 4



en_cours = 4



en_cours = 4

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

Problématique

Trier des cartes
manuellement

Transposer au tri
de données

Tri par sélection

Implémentation

Terminaison

Correction

Complexité

Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

Complexité

Correction : boucle principale

Problématique

Trier des cartes
manuellement

Transposer au tri
de données

Tri par sélection

Implémentation

Terminaison

Correction

Complexité

Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

Complexité

```
1 def tri_insertion(tab: list) -> None:
2     """
3     tri le tableau dans l'ordre croissant
4     """
5     for i in range(len(tab)):
```

Correction : mémoriser

Problématique

Trier des cartes
manuellement

Transposer au tri
de données

Tri par sélection

Implémentation

Terminaison

Correction

Complexité

Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

Complexité

```
1   en_cours = tab[i]
2   pos = i
```

Problématique

Trier des cartes
manuellement

Transposer au tri
de données

Tri par sélection

Implémentation

Terminaison

Correction

Complexité

Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

Complexité

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


Correction : insérer

Problématique

Trier des cartes
manuellement

Transposer au tri
de données

Tri par sélection

Implémentation

Terminaison

Correction

Complexité

Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

Complexité

1

```
tab[pos] = en_cours
```

Correction : code complet

Trier des cartes

Problématique

Trier des cartes
manuellement

Transposer au tri
de données

Tri par sélection

Implémentation

Terminaison

Correction

Complexité

Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

Complexité

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

Problématique

Trier des cartes
manuellement

Transposer au tri
de données

Tri par sélection

Implémentation

Terminaison

Correction

Complexité

Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

Complexité

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

Problématique

Trier des cartes
manuellement

Transposer au tri
de données

Tri par sélection

Implémentation

Terminaison

Correction

Complexité

Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

Complexité

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

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

Problématique

Trier des cartes
manuellement

Transposer au tri
de données

Tri par sélection

Implémentation

Terminaison

Correction

Complexité

Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

Complexité

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

pos est un variant de la boucle.

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

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

Code 11 – Insertion de l'élément 4

Problématique

Trier des cartes
manuellementTransposer au tri
de données

Tri par sélection

Implémentation

Terminaison

Correction

Complexité

Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

Complexité

4

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

4

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

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

Code 12 – Insertion de l'élément 4

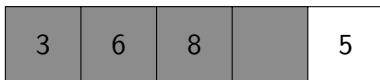
La boucle externe effectue n itérations dans tous les cas.

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

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

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

4



Code 13 – Insertion de l'élément 4

Problématique

Trier des cartes
manuellement

Transposer au tri
de données

Tri par sélection

Implémentation

Terminaison

Correction

Complexité

Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

Complexité

Problématique

Trier des cartes
manuellementTransposer au tri
de données

Tri par sélection

Implémentation

Terminaison

Correction

Complexité

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

```
1 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.

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

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

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

[Problématique](#)[Trier des cartes
manuellement](#)[Transposer au tri
de données](#)[Tri par sélection](#)[Implémentation](#)[Terminaison](#)[Correction](#)[Complexité](#)[Tri par insertion](#)[Implémentation](#)[Preuve de terminaison](#)[Preuve de correction](#)[Complexité](#)

Problématique

Trier des cartes
manuellementTransposer au tri
de données

Tri par sélection

Implémentation

Terminaison

Correction

Complexité

Tri par insertion

Implémentation

Preuve de terminaison

Preuve de correction

Complexité

À retenir

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