

## 1 Problématique

Trier un jeu de cartes est une opération qui peut paraître triviale, mais qui ouvre des enjeux essentiels dans le contexte des données. En effet, on demande régulièrement à une machine de manipuler et donc trier des quantités importantes d'informations et il convient alors d'élaborer des algorithmes rigoureux.

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

## 2 Trier des cartes manuellement



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.

On distingue deux algorithmes :

- 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 1 – Tri par sélection (en place)

- 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 2 – Tri par insertion (en place)

## 3 Transposer au tri de données

### 3.1 Tri par sélection

#### 3.1.1 Implémentation

**Activité 2 :**

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

**3.1.2 Terminaison**

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

**3.1.3 Correction**

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

**3.1.4 Complexité**

Il convient de se demander si cette méthode de tri est efficace. Remarquons que le nombre d'opérations effectuées dépend du nombre *n* d'éléments dans le tableau :

- 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.
- ...

**À retenir**

Le tri par sélection effectue  $\frac{n.(n - 1)}{2}$  opérations pour ordonner le tableau. Le nombre d'opérations dépend de  $n^2$ .

**3.2 Tri par insertion****3.2.1 Implémentation****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.

### 3.2.2 Terminaison

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

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

### 3.2.3 Correction

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

### 3.2.4 Complexité

La boucle externe effectue  $n$  itérations dans tous les cas. Cependant, le nombre d'itérations de la boucle interne peut varier.

**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.

### À retenir

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