

## T7.2 Tri par insertion

Après le tri par sélection, nous allons étudier un deuxième algorithme de tri: le tri par insertion. C'est le «tri du joueur de cartes».

Il consiste à choisir un élément et de l'insérer à la bonne position en faisant «remonter» les éléments plus grands que lui.

6    5    3    1    8    7    2    4

### Quelques remarques:

- on commence à l'indice 1;
- pour chaque indice de travail  $i$ , on appelle clé l'élément de la liste d'indice  $i$ ;
- on examine ensuite les éléments à gauche, c'est-à-dire les éléments d'indice  $j < i$ ;
- tant que l'élément d'indice  $j$  est supérieur à la clé, on le décale d'une position vers la droite;
- une fois que ce n'est plus possible, on insère la clé.

## Exercice 1

### Énoncé

Implémenter l'algorithme du tri par insertion sous la forme d'une fonction:

#### Script Python

```

1 def tri_insertion(tab: list) -> None:
2     """
3         Trie en place le tableau tab donné en paramètre
4     """

```

### Indication

#### Script Python

```

1 def tri_insertion(tab: list) -> None:
2     """
3         Trie en place le tableau tab donné en paramètre
4     """
5     for i in range():
6         cle =
7         j =
8         while and :
9             tab[ ] = tab[ ]
10            j =
11            tab[ ] =

```

### Correction

## Exercice 2

### Énoncé

Il s'agit d'étudier la complexité de cet algorithme de façon expérimentale.

Pour cela:

1. Reprendre les fonctions `chrono` et `pire_cas` de l'exercice 2 sur le tri par sélection et afficher le graphique des temps d'exécution.
2. Faire de même avec une liste déjà triée. Que remarquez-vous?

### Correction

 À voir

<https://www.toptal.com/developers/sorting-algorithms>