Tris de tableaux

1. Tri par insertion

Le tri par insertion est le plus naturel : il consiste à parcourir le tableau de la gauche vers la droite, et pour chaque élément, le classer dans la partie du tableau située à sa gauche.

Algorithme

Voici un exemple d'algorithme de tri par insertion :

```
1. def tri_insertion(tab):
2.     for i in range(1,len(tab)):
3.         cle=tab[i]
4.         j=i-1
5.         while j >= 0 and tab[j] >cle:
6.             tab[j+1] = tab[j]
7.             j = j - 1
8.         tab[j+1]=cle
```

On se propose d'effectuer un tri croissant par insertion sur le tableau tab = [9,8,5,4,7,6] .
On peut résumer dans un tableau ce que fait l'algorithme pour chaque itération de la boucle for :

Valeur de i	Tableau avant la boucle	Valeur de la clé	Tableau en fin de boucle
1	[9, 8, 5, 4, 7, 6]	8	[8, 9, 5, 4, 7, 6]
2	[8, 9, 5, 4, 7, 6]	5	[5, 8, 9, 4, 7, 6]
3	[5, 8, 9, 4, 7, 6]	4	[4, 5, 8, 9, 7, 6]
4	[4, 5, 8, 9, 7, 6]	7	[4, 5, 7, 8, 9, 6]
5	[4, 5, 7, 8, 9, 6]	6	[4, 5, 6, 7, 8, 9]

Coût

Dans le pire des cas, (tableau initial trié par ordre décroissant), le coût est de l'ordre de n^2 (noté $O(n^2)$). On dit que le coût est **quadratique** dans le pire des cas.
Au meilleur des cas (liste déjà triée) le coût n'est que **linéaire** (noté O(n)), car dans ce cas la boucle while n'est jamais exécutée.

Preuve de correction

Le tableau est trié jusqu'à la case n° i est ce qu'on appelle un **invariant de boucle.**
Cette propriété est vraie avant et après chaque itération.
Elle prouve que l'algorithme est correct.

2.Tri par sélection

Le tri par sélection consiste à :

- ➔ rechercher le plus petit élément du tableau et l'échanger avec celui d'indice 0,
- ➔ rechercher le plus petit élément du tableau restant et l'échanger avec celui d'indice 1,
- ➔ continuer ainsi pour tous les éléments

Algorithme

Voici un algorithme de tri par sélection écrit en Python :

```
1. def tri_selection(tab):
2.     for i in range(0,len(tab)-1):
3.         mini = i
4.         for j in range(i+1, len(tab)):
5.             if tab[j] < tab[mini]:
6.                 mini = j
7.         temp = tab[i]
8.         tab[i] = tab[mini]
9.         tab[mini] = temp
```

Le tableau suivant résume ce que fait l'algorithme à chaque itération sur le tableau tab = [9,8,5,4,7,6] :

Valeur de i	Tableau avant la boucle	Tableau en fin de boucle	Index des valeurs échangées
1	[9, 8, 5, 4, 7, 6]	[4, 8, 5, 9, 7, 6]	0 et 3
2	[4, 8, 5, 9, 7, 6]	[4, 5, 8, 9, 7, 6]	1 et 2
3	[4, 5, 8, 9, 7, 6]	[4, 5, 6, 9, 7, 8]	2 et 5
4	[4, 5, 6, 9, 7, 8]	[4, 5, 6, 7, 9, 8]	3 et 4
5	[4, 5, 6, 7, 9, 8]	[4, 5, 6, 7, 8, 9]	4 et 5

Coût

La première boucle s'effectue (n – 2) fois et la seconde (n – 1 – i) fois. Le coût est donc proportionnel à n^2, donc noté $O(n^2)$, soit un coût **quadratique**.

Preuve de correction

Comme pour le tri par insertion, la propriété « Le tableau est trié jusqu'à la case n° i – 1 » est un **invariant de boucle**. Cette propriété est vraie avant et après chaque itération. Elle prouve que l'algorithme est correct.