
RETOUR SUR LE TRI PAR INSERTION

Objectifs

- Implémenter une version récursive du tri par insertion,
- Mesurer la complexité temporelle d'un algorithme récursif.

Problématique On rappelle ci-dessous le code permettant de trier en place un tableau $t[0\dots n-1]$, de taille n , par insertion.

```

1 def tri_insertion(t):
2     for i in range(len(t)-1):
3         # t[0...i] est trié
4         j = i + 1
5         while j > 0 and t[j] < t[j-1]:
6             t[j-1], t[j] = t[j], t[j-1]
7             j -= 1
8         # t[0...i+1] est trié

```

On souhaite construire une version récursive de cette fonction, en remplaçant les boucles imbriquées par des appels récursifs de fonctions. Pour cela, nous introduirons les deux fonctions récursives suivantes :

- **tri_insertion_rec** : c'est la fonction principale qui se chargera de parcourir tous les éléments, dernier exclu, et qui appellera la *fonction auxiliaire* **insérer** pour insérer l'élément $t[i+1]$ dans la partie du tableau $t[0\dots i]$ déjà trié.
- **insérer** : fonction qui se charge d'insérer l'élément $t[j]$ parmi les j éléments précédents, déjà triés.

Une fois ces fonctions implémentées, on analysera la complexité temporelle de l'algorithme sous-jacent, principalement en comptant le nombre d'appels récursifs effectués.

1. Implémente la fonction **insérer**(t , j) prenant en argument un tableau t et l'indice j de l'élément à insérer dans $t[0 : j-1]$ puis valide-la comme sur l'exemple suivant :

```

1 t = [1, 2, 3, 0, 7, 6]
2 insérer(t, 3)
3 print(t)
4 >>> [0, 1, 2, 3, 7, 6]

```

2. Implémente la fonction **tri_insertion_rec** comme spécifiée en introduction.
3. Instrumente ton programme pour compter le nombre d'appels de chacune des fonctions récursives, en introduisant deux variables globales **nb_appels_tri** et **nb_appels_insérer** qui seront incrémentés à chaque appel.
 - (a) Dans le cas du tri du tableau $[7, 6, 5, 4, 3, 2, 1, 0]$, combien d'appels à la fonction **insérer** sont effectués ? Et à la fonction **tri_insertion_rec** ?
 - (b) De même question dans le pire des cas, pour un tableau de taille n . Comparer le nombre d'appels de chaque fonction.
 - (c) Génère des tableaux de nombres correspondant au pire cas, d'une taille variant de 1 à 100, puis stocke le nombre d'appels à la fonction **insérer** obtenu pour chacun d'entre eux dans un tableau **t_insérer**. Détermine ensuite la fonction mathématique qui permet de représenter la relation entre la taille n du tableau et nombre d'appels. On pourra utiliser la bibliothèque **matplotlib** pour représenter graphiquement cette fonction et la comparer aux fonctions de référence, comme dans l'exemple suivant :

```

1 import matplotlib.pyplot as plt
2 plt.plot(t_n, t_insérer)
3 plt.plot(t_n, [x**2 for x in t_n])
4 plt.plot(t_n, [2**x for x in t_n])
5 plt.plot(t_n, t_n)
6 plt.ylim([0, 10000])
7 plt.show()

```

où **t_n** est un tableau contenant toutes les tailles de tableaux générés.

Spé. Maths : En modélisant le nombre d'appels à **insérer** par une suite arithmétique, calcule la somme de ses termes correspondant au nombre total d'appels.