THÈME 3: Algorithmes de tri (Partie 2)

Nous avons étudié un premier algorithme de tri la dernière séance. Cet algorithme est le tri par sélection. Toutefois, cet algorithme est très inefficace : il nous faut donc trouver un algorithme plus rapide.

Le but de ce TP est d'étudier un deuxième algorithme de tri classique appelé le tri par insertion.

1) Algorithme "avancé" : tri par insertion

L'algorithme de tri par insertion est plus avancé et permet d'obtenir de meilleures performances de tri. On vous donne ci-dessous l'algorithme de tri par insertion.

DEBUT

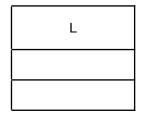
```
n ← longueur(L)
pour i allant de 1 à n-1:
   valeurTraitee ← L[i]
   j ← i-1
   tant que j>=0 et valeurTraitee < L[j]:
        L[j+1] ← L[j]
        j ← j-1
   L[j+1] ← valeurTraitee
renvoyer L
FIN</pre>
```

Pour comprendre cet algorithme, il est **CRITIQUE** de le dérouler à la main.

Question 1

À l'aide d'une feuille et d'un crayon et en recopiant le tableau algorithmique ci-dessous, appliquez cet algorithme sur la liste L = [10, 3, 7, 5, 6, 1]. En particulier, on suivra **l'évolution de L** à chaque passage dans les différentes boucles.

i	valeurTraitée	j	L[j]	j>= 0 et valeurTraitee < L[j]	L[j+1]



Question 2

En vous rappelant du principe du tri par sélection et en utilisant la question 1, expliquez le fonctionnement de l'algorithme de tri par insertion.

Question 3

- **a.** Montrez que la propriété "à l'étape k, les valeurs comprises entre 0 et k du tableau L sont triés dans l'ordre croissant" est un invariant de boucle.
- b. En déduire que cet algorithme réalise bien un tri d'un tableau dans l'ordre croissant.

Question 4

Reprenez le programme "theme3.py" déjà écrit au cours précédent et disponible sur mon site web (<u>bouillotvincent.github.io</u>) à l'emplacement *Chapitre 7, Fichier Python (Correction)* .

Traduisez l'algorithme en Python en complétant la fonction triInsertion.

Question 5

Tester votre code Python sur <u>PythonTutor.com</u> pour observer le fonctionnement du programme. On fera des tests pour les tableaux :

- **♦** L = [1, 5, 12, 8]
- **♦** L = [-3, -5, -12, -1]
- ♣ L = []

2) Comparaison d'algorithmes

La librairie timeit permet de faire des mesures de vitesse d'exécution (voir Partie 1 pour une explication plus détaillée).

Python possède une fonction de tri interne. Cette fonction s'appelle sorted et s'utilise ainsi :

sorted([10, 3, 7,5,6, 1])

renvoie

[1, 3, 5, 6, 7, 10]

Pour trier un tableau dans l'ordre inverse, on va utiliser L100inv = sorted(L100, reverse=True).

Question 1 — Identification du pire et du meilleur des cas

Indiquez le nombre d'étapes mis par PythonTutor pour trier (par insertion) les cas suivants :

- L = [4, 2, 8, 7, 9, 3, 11]
- **♦** L = [13, 11, 8, 6, 4, 2, 1]
- **♦** L = [1, 4, 5, 6, 9, 13,15]

Question 2 — Complexité

- a. Déduire de la guestion 1 le pire et le meilleur des cas pour cet algorithme.
- **b.** Afficher le temps mis par le tri par insertion pour :
 - trier un tableau de 100, 1000 et 10000 éléments déjà triés ;
 - trier un tableau de 100, 1000 et 10000 éléments triés en sens inverse ;
- c. En déduire la la complexité du tri par insertion dans ces deux cas.

Question 3

Affichez à présent le temps d'exécution des algorithmes de tri par insertion et par sélection dans les six cas précédents.

Question 4

Nous souhaitons généraliser nos résultats et obtenir une courbe.

On dispose d'un tableau avec les tailles de nos tableaux.

Modifiez votre programme de manière à **enregistrer** vos mesures de temps dans des **tableaux** appelés tpsSelTrie, tpsInsTrie, tpsSortTrie, tpsSelInv, tpsInsInv, TpsSortInv.

Exemple:

```
tailleTableau = [ 100, 300, 500, 700 ]

tpsSelTrie = [ 0,0001 , 0,001, 0,01, 0,2 ] # tri par sélection sur un tableau trié

tpsSortTrie = [ 0,0001 , 0,002, 0,008, 0,1 ] # tri sorted Python sur un tableau trié

Cela se lit : pour un tableau de 700 éléments trié croissant, le tri par sélection a pris 0,2 seconde.
```

Question 5

À l'aide de la bibliothèque graphique matplotlib, définissez x et y correctement puis représentez l'évolution du temps de calcul en fonction de la taille du tableau à trier. La complexité est-elle conforme à vos attentes ? On pourra une échelle logarithmique¹ sur l'axe des ordonnées en remplaçant plot par semilogy.

Question 6

Pensez-vous que le tri interne de Python soit le tri par sélection, par insertion ou un autre ? Pourquoi ?

¹ https://fr.wikipedia.org/wiki/Échelle_logarithmique