Algorithmique 1 - Tri par insertion

Les objectifs : Etude du tri par sélection (principe, algorithme et fonction à programmer en python, rappels : trace/spécification/terminaison du programme)

Les acquis précédents : manipulation autour des tableaux, listes et autres conteneurs, fonctions.

Une présentation dynamique : https://www.tice-education.fr/index.php/tous-les-articles-er-ressources/codage/1129-les-methodes-de-tri (à télécharger)

Le principe dansé : https://youtu.be/ROalU379l3U

Le tri par insertion, dans lequel on part du deuxième élément de liste (E2) que l'on compare au précédent (E1), si E1 est plus petit, on échange leur place. Et on recommence ensuite le troisième élément que l'on compare à tous les éléments qui le précède, tant qu'on rencontre un élément plus grand, on échange leur place

Par exemple :

Liste de 6 éléments						**i**	**J de .. à ...**		**Nombre d'inversions réalisées /test**
2	**1**	**8**	**3**	**0**	**7**	**1**	**1**	**1**	**1 /1**
1	**2**	**8**	**3**	**0**	**7**	**2**	**2**	**2**	**0/1**
1	**2**	**8**	**3**	**0**	**7**	**3**	**3**	**2**	**1/2**
1	**2**	**3**	**8**	**0**	**7**	**4**	**4**	**1**	**4/4**
0	**1**	**2**	**3**	**8**	**7**	**5**	**5**	**4**	**1/2**

1. **Écrire la spécification d'un programme en python qui permettrait de réaliser ce tri.**

Spécification :

```
#Précondition : N >=0, T est une liste de N entiers
Action : Trier (T)
#Postcondition : Tout élément de T<=à son suivant dans T
```

L'algorithme :

```
Entrée : liste, liste de N entiers
Début
Pour k ←1à (nombre éléments de la liste) :
    temp←liste(k)
    j←k
    Tant que j>0 et temp<liste(j-1) :
        liste(j ←liste(j-1)
        j←j-1
        liste(j) ←temp
    Fin Tant que
Fin Pour
Afficher liste
Fin
```

2. **Ecrire une fonction de tri par insertion en python à partir de l'algorithme proposé. Voir tri_ins.py**

3. **Le tester avec les valeurs de l'exemple précédent puis faire d'autres jeux de tests dont une liste déjà triée dans l'ordre croissant et une autre dans l'ordre décroissant.**

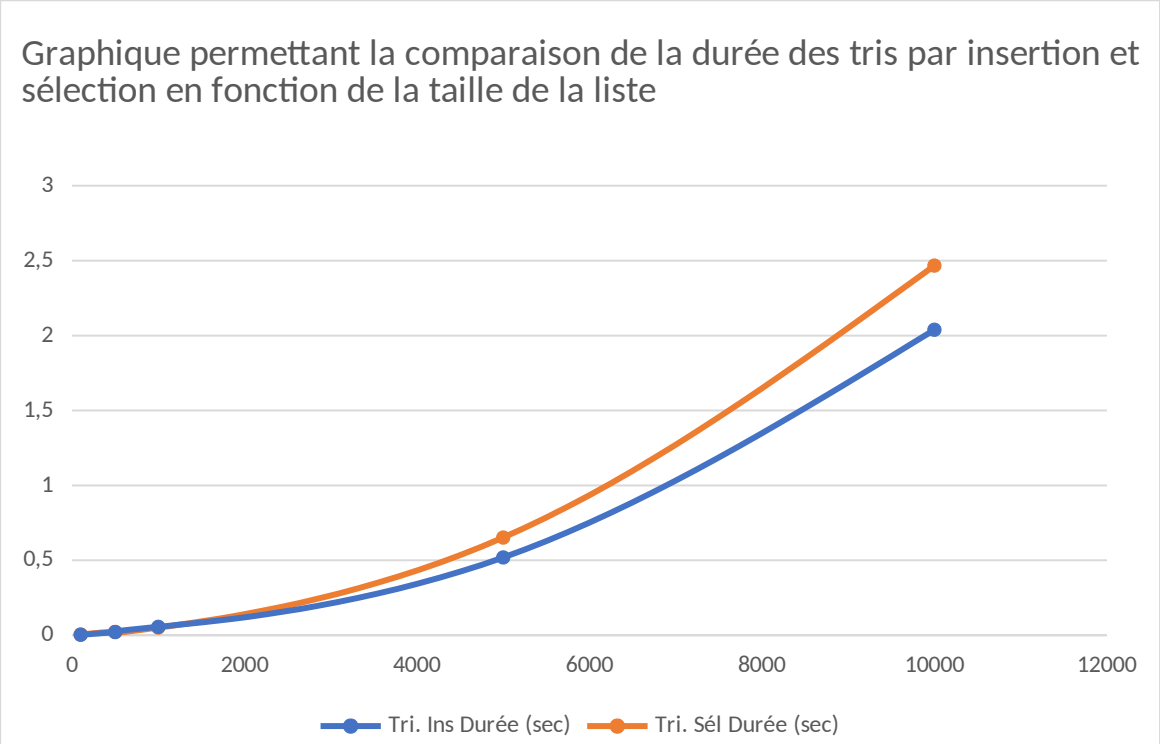
4. **Justifier la TERMINAISON du programme en identifiant un VARIANT.** Un variant diminue strictement à chaque itération d'une boucle. **Le variant qui diminue strictement à chaque tour est j dans la boucle while avec la condition j>0 donc si j<=0, on sort de la boucle**

5. **Ajouter une fonction time en utilisant time.time() pour évaluer la durée du tri.**
6. **Lancer le tri pour différentes tailles de liste. Pour cela, créer une liste par compréhension** (voir programme pour détails) et faire varier la taille de la liste

7. **Dans un tableur, indiquer pour les différentes tailles la durée d'exécution.**

	nombre éléments liste	100	500	1000	5000	10000
tri insertion	Durée (sec)	0,003	0,021	0,056	0,519	2,039

8. **Construire un graphique avec un tableur : temps d'exécution en fonction de la taille de la liste à trier.**



→ **Le tri par sélection sera étudié prochainement. Les 2 Tris de complexité équivalente**

9. **Quel est leur niveau de complexité ?** Type **N^2→ peu efficace pour de nombreuses valeurs**