Lycée Jean Moulin **Draguignan**
NSI 1re Année .....

Les tris

Le problème

Étant donné une liste d'entiers, il faut la trier dans l'ordre croissant.
On peut bien entendu l'étendre à autre chose que des entiers...
Bien que les listes aient une méthode prédéfinie pour trier (`ma_liste.sort()`), il est intéressant de faire soit même des algorithmes de tris.
Il existe plusieurs algorithme de tris :

- Le tri par insertion.
- Le tri par sélection.
- Le tri à bulles.
- Le tri à peigne.
- Le tri Shaker.
- Le tri Shell.
- Le tri Gnome.
- Le tri par tas.
- Le tri fusion.
- Le tri rapide.
- ...

On ne va pas tous les étudier...Mais l'étude de certains est un incontournable de l'algorithmique.

Le tri par insertion

Dans un premier temps regarder cette vidéo qui présente le tri par insertion :
lien vers la vidéo
Et si vous n'avez pas encore compris...Regarder cette simulation réalisée par vos professeurs...
lien vers la simulation

? Exercice 1 :

Compléter le texte.
Au début on compare le ..... et le ..... en les ..... si .....
Puis on compare le ..... avec le ..... en les échangeant si c'est nécessaire, si l'échange a eu lieu on compare le ..... avec le ..... en les échangeant si nécessaire
Et ainsi de suite.
Au bout d'un certain 'temps' on considère le $k^{\text{ème}}$ élément de la liste, il faut le comparer et l'échanger avec ses prédécesseurs tant qu'.....

Voici l'algorithme

```
fonction tri_insertion(L)
Données : L ← liste d'entiers
n ← taille de la liste
Pour k allant de 1 à n-1 faire
    j ← k
    Tant que j > 0 et L[j-1] > L[j] faire
        On échange L[j-1] et L[j]
        j ← j − 1
renvoyer L
```

? Question 1:

Pourquoi : Pour k allant de 1 à $n-1$?
.....
.....
.....

Terminaison

```
fonction tri_insertion(L)
Données : L ← liste d'entiers
n ← taille de la liste
Pour k allant de 1 à n-1 faire
    j ← k
    Tant que j > 0 et L[j-1] > L[j] faire
        On échange L[j-1] et L[j]
        j ← j − 1
renvoyer L
```

Question 2:

Étudier la terminaison de cet algorithme
Boucle bornée :

.....
.....

Boucle tant que : Variant de boucle :.....

.....
.....
.....
.....
.....

Correction

```
fonction tri_insertion(L)
Données : L ← liste d'entiers
n ← taille de la liste
Pour k allant de 1 à n-1 faire
    j ← k
    Tant que j > 0 et L[j-1] > L[j] faire
        On échange L[j-1] et L[j]
        j ← j − 1
renvoyer L
```

Question 3:

Étudier la correction de cet algorithme en considérant l'invariant de boucle : pour la $k^{\text{ème}}$ itération la sous liste des k premiers éléments est triée.

.....
.....
.....
.....
.....
.....

Complexité

```
fonction tri_insertion(L)
Données : L ← liste d'entiers
n ← taille de la liste
Pour k allant de 1 à n-1 faire
    j ← k
    Tant que j > 0 et L[j-1] > L[j] faire
        On échange L[j-1] et L[j]
        j ← j − 1
renvoyer L
```

Question 4:

Déterminer la complexité de cet algorithme dans le pire des cas.

.....
.....
.....
.....
.....
.....
.....
.....

Exercice 2 : *Voir le notebook*

Implémenter cet algorithme en Python. Vous écrirez une fonction `tri_insertion(L)`, qui prendra en paramètre une liste et qui retournera la liste triée dans l'ordre croissant, faites également une mesure du temps d'exécution.

Exercice 3 : *À faire dans le notebook*

Modifier le programme précédent, pour qu'il retourne la liste triée dans l'ordre décroissant.

Le tri par sélection

Exercice 4 : *Voir le notebook*

En examinant la simulation, écrire un algorithme de **tri par sélection** sur une liste d'entiers.

Exercice 5 :

Étudier sa terminaison

Exercice 6 :

Étudier sa correction

Exercice 7 :

Déterminer sa complexité

Le tri à bulles

Exercice 8 : *Voir le notebook*

Réaliser un programme de tri à bulles.