I. Mise en jambe...

1 - Le tri du singe...

La fonction `shuffle` provient du module random. Elle permet de mélanger sur place une liste L. La fonction `randint(a,b)` permet de tirer un nombre aléatoire entre a et b.

```
>>>L = [1,2,3,4]
>>>shuffle(L)
>>>L
[4,2,3,1]
```

FIGURE 1 – Singes jouant aux cartes, Huile sur cuivre, David Teniers le Jeune (Anvers, 1610 - Bruxelles, 1690)

Un singe trie les cartes de la façon suivante : il prend les cartes, les jette en l'air, les ramasse puis regarde si elles sont triées. Si oui, il s'arrête, sinon il relance les cartes.

Ex. 1 :

Q 1 - Définir une fonction `gen_alea(N)` renvoyant une liste de N entiers entre 0 et $N-1$ mélangés

Q 2 - Définir une fonction `est_triee(L)` qui renvoie le booléen `True` si la liste est triée et `False` sinon.

Q 3 - Implémenter une fonction `tri_singe` et le tester sur la liste mélangée de 2, 3,4,5 éléments... éviter d'aller trop loin...

Q 4 - Quelle est la complexité d'un tel algorithme ?

2 - Le tri par comptage

Le tri comptage (counting sort en anglais), appelé aussi tri casier, est un algorithme de tri par dénombrement qui s'applique sur des valeurs **entières** uniquement.

FIGURE 2 – Casier de comptage

On suppose qu'on dispose d'une liste composée de 100 entiers entre 0 et 30 (bornes comprises). Le procédé du tri par comptage est le suivant : on compte le nombre de " 0 ", le nombre de " 1 "..., le nombre de " 30 " présents dans la liste, et on reconstruit une liste en y ajoutant les valeurs non nulles selon leur quantité croissante (on ne trie pas les valeurs mais le comptage de ces valeurs au sein du tableau).

Par exemple, le tableau de 5 entiers `[ 1, 27, 3, 1, 3]` contient 2 fois 1, 2 fois 3 et 1 fois 17, le tableau trié par la méthode du tri comptage est donc : `[ 1, 1, 3, 3, 17]`.

Ex. 2 :

Q 1 - Créer une liste `L` de 20 nombres entiers aléatoires entre 0 et 10 compris.

Q 2 - Définir une fonction `comptage(L)` de complexité linéaire qui renvoie une liste `C` de `11` éléments (soit `max(L) +1`) dénombrant les valeurs présentes dans `L`. Si `L` contient 4 fois la valeurs 9 alors `C[9] = 4`. Vérifier votre fonction avec la liste définiée en 1.

Q 3 - Proposer une fonction `tri_comptage(L)` qui renvoie une liste triée sous réserve que `L` soit à valeur entière.

Q 4 - Justifier que ce tri est de complexité linéaire.

Q 5 - Pourquoi ce tri pose problème si `L = [6,2,3,10**100]`

II. Tri par insertion

1 - Positionnement d'un élément

Le tri par insertion nécessite de connaître la position d'un élement dans une liste triée.

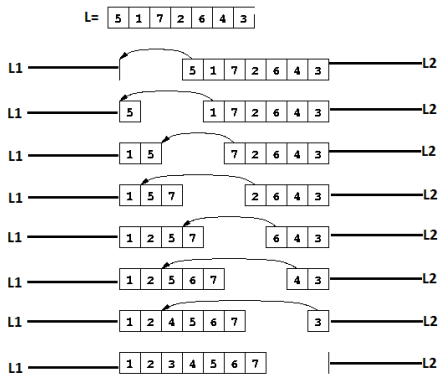

FIGURE 3 – Illustration du tri par insertion

Soit `Ltri=[2,3,4,6,9]` une liste triée par ordre croissante.

Ex. 3 :

Q 1 - À l'aide d'une boucle `for`, définir une fonction `position(x,Ltri)` qui renvoie la position k telle que : `Ltri[k-1]<x` $\leq$ `Ltri[k]` ou 0 si `x<=Ltri[0]`, `len(Ltri)` si `x>Ltri[-1]`. La fonction devra valider les tests suivants :

```
>>>position(1,Ltri)
0
>>>position(5,Ltri)
3
>>>position(10,Ltri)
5
```

Q 2 - En utilisant la dichotomie, proposer une fonction `position_dich(x,Ltri)` identique à la précédente mais de complexité logarithmique.

2 - Insertion d'un élément

Ex. 4 :

Soit *Ltri* une liste triée.

Q 1 - Définir une fonction `insertion(x,Ltri)` qui renvoie une liste où `x` a été inséré à la bonne place.

Q 2 - À partir des fonctions précédentes, en déduire une fonction `TriInsertion(L)` renvoyant une liste triée par insertion. La fonction devra valider les tests suivants :

```
>>>TriInsertion([0,5,2,3])
[0,2,3,5]
>>>TriInsertion([0,5,2,3,0])
[0,0,2,3,5]
>>>TriInsertion([0,5,2,3,9])
[0,2,3,5,9]
```

Le tri par insertion est possible en place. On considère un par un chaque élément du tableau et on l'insère à la bonne place parmi les éléments déjà triés. Ainsi, au moment où on considère un élément, les éléments qui le précèdent sont déjà triés, tandis que les éléments qui le suivent ne sont pas encore triés.

Ex. 5 :

En utilisant la méthode `L.insert(k,x)` qui place l'élément `x` à la position `k` de la liste ainsi que la méthode `L.remove(x)` qui supprime la valeur `x` de L, proposer une fonction `Tri_ins_enplace(L)` sans valeur de retour. La fonction devra valider les tests suivants :

```
>>>L = [9,0,5,2,3]
>>>Tri_ins_enplace(L)
>>>L
[0,2,3,5,9]
```