TP 2e année IPT : Tri rapide

Exercice 1 : Partition d'une liste selon un pivot

On dispose d'une liste Python `L` de longueur n. Le premier élément de `L`, l'entier `L[0]`, est choisi comme pivot. On souhaite réordonner les éléments de `L` de sorte que les éléments qui précèdent le pivot soient inférieurs (ou égal) au pivot.

1. Écrire une fonction `pivot(L)` qui réalise le travail demandé en créant une nouvelle liste `L2` de même taille que `L`, initialement remplie de zéros. On lit les éléments `L[k]` pour $k > 1$ et on modifie `L2` par la gauche ou par la droite selon que $L[k] \leqslant L[0]$ ou non. Le pivot est ajouté à la fin dans `L2` à la bonne place. Par exemple, pivot([3,1,6,8,4,9,2,0]) renvoie [1,2,0,3,9,4,8,6].

La complexité spatiale (coût mémoire) de la fonction `pivot` n'est pas bonne. On la remplace par une autre fonction qui agit en place :

2. Écrire une fonction `pivot_en_place(L,i,j)` qui prend comme pivot l'élément `L[i]` et modifie la sous-liste `L[i..j]` pour placer correctement le pivot. Cette fonction modifie `L` et renvoie la position finale du pivot.

Exercice 2 : Application au tri rapide

1. Utiliser la fonction `pivot_en_place(L,i,j)` pour écrire une procédure récursive `tri_rapide` qui modifie `L` de sorte qu'elle soit entièrement triée.

2. Tester `tri_rapide` sur de grandes listes aléatoires. On pourra utiliser la fonction `shuffle` du module `random`:

```
import random
L = [ i for i in range(10**6) ]
random.shuffle(L)
```

Au contraire, que se passe-t-il quand la liste est déjà triée (par ordre croissant ou décroissant)?

Exercice 3 : Tri rapide randomisé

Pour contrer le problème des listes partiellement triées, on va choisir aléatoirement le pivot à chaque étape.

1. Adapter la fonction `pivot_en_place(L,i,j)` en une fonction `pivot_en_place_random(L,i,j)` qui choisit le pivot aléatoirement entre i et j. En déduire une fonction `tri_rapide_random(L,i,j)`.

2. Tester cette fonction de tri sur des listes déjà triées. On pourra utiliser la fonction `clock` du module `time` pour évaluer le temps que met `tri_rapide_random` pour trier une liste triée dans l'ordre décroissant de taille 10^6.