

## Exercice 5 (4 points)

Cet exercice traite de manipulation de tableaux, de récursivité et du paradigme « diviser pour régner ».

Dans un tableau Python d'entiers `tab`, on dit que le couple d'indices  $(i, j)$  forme une inversion lorsque  $i < j$  et `tab[i] > tab[j]`. On donne ci-dessous quelques exemples.

- Dans le tableau `[1, 5, 3, 7]`, le couple d'indices  $(1, 2)$  forme une inversion car  $5 > 3$ . Par contre, le couple  $(1, 3)$  ne forme pas d'inversion car  $5 < 7$ . Il n'y a qu'une inversion dans ce tableau.
- Il y a trois inversions dans le tableau `[1, 6, 2, 7, 3]`, à savoir les couples d'indices  $(1, 2)$ ,  $(1, 4)$  et  $(3, 4)$ .
- On peut compter six inversions dans le tableau `[7, 6, 5, 3]` : les couples d'indices  $(0, 1)$ ,  $(0, 2)$ ,  $(0, 3)$ ,  $(1, 2)$ ,  $(1, 3)$  et  $(2, 3)$ .

On se propose dans cet exercice de déterminer le nombre d'inversions dans un tableau quelconque.

### Questions préliminaires

1. Expliquer pourquoi le couple  $(1, 3)$  est une inversion dans le tableau `[4, 8, 3, 7]`.
2. Justifier que le couple  $(2, 3)$  n'en est pas une.

### Partie A : Méthode itérative

Le but de cette partie est d'écrire une fonction itérative `nombre_inversion` qui renvoie le nombre d'inversions dans un tableau. Pour cela, on commence par écrire une fonction `fonction1` qui sera ensuite utilisée pour écrire la fonction `nombre_inversion`.

1. On donne la fonction suivante.

```
def fonction1(tab, i):
    nb_elem = len(tab)
    cpt = 0
    for j in range(i+1, nb_elem):
        if tab[j] < tab[i]:
            cpt += 1
    return cpt
```

- a. Indiquer ce que renvoie la `fonction1(tab, i)` dans les cas suivants.

- Cas n°1 : `tab = [1, 5, 3, 7]` et `i = 0`.
- Cas n°2 : `tab = [1, 5, 3, 7]` et `i = 1`.
- Cas n°3 : `tab = [1, 5, 2, 6, 4]` et `i = 1`.

- b. Expliquer ce que permet de déterminer cette fonction.

2. En utilisant la fonction précédente, écrire une fonction `nombre_inversion(tab)` qui prend en argument un tableau et renvoie le nombre d'inversions dans ce tableau. On donne ci-dessous les résultats attendus pour certains appels.

```
>>> nombre_inversions([1, 5, 7])
0
>>> nombre_inversions([1, 6, 2, 7, 3])
3
>>> nombre_inversions([7, 6, 5, 3])
6
```

3. Quelle est l'ordre de grandeur de la complexité en temps de l'algorithme obtenu ? Aucune justification n'est attendue.

## Partie B : Méthode récursive

Le but de cette partie est de concevoir une version récursive de la fonction `nombre_inversion`.

On définit pour cela des fonctions auxiliaires.

1. Donner le nom d'un algorithme de tri ayant une complexité meilleure que quadratique.

Dans la suite de cet exercice, on suppose qu'on dispose d'une fonction `tri(tab)` qui prend en argument un tableau et renvoie un tableau contenant les mêmes éléments rangés dans l'ordre croissant.

2. Écrire une fonction `moitie_gauche(tab)` qui prend en argument un tableau `tab` et renvoie un nouveau tableau contenant la moitié gauche de `tab`. Si le nombre d'éléments de `tab` est impair, l'élément du centre se trouve dans cette partie gauche. On donne ci-dessous les résultats attendus pour certains appels.

```
>>> moitie_gauche([])
[]
>>> moitie_gauche([4, 8, 3])
[4, 8]
>>> moitie_gauche([4, 8, 3, 7])
[4, 8]
```

Dans la suite, on suppose qu'on dispose de la fonction `moitie_droite(tab)` qui renvoie la moitié droite sans l'élément du milieu.

3. On suppose qu'une fonction `nb_inv_tab(tab1, tab2)` a été écrite. Cette fonction renvoie le nombre d'inversions du tableau obtenu en mettant bout à bout les tableaux `tab1` et `tab2`, à condition que `tab1` et `tab2` soient triés dans l'ordre croissant. On donne ci-dessous deux exemples d'appel de cette fonction :

```
>>> nb_inv_tab([3, 7, 9], [2, 10])
3
>>> nb_inv_tab([7, 9, 13], [7, 10, 14])
3
```

En utilisant la fonction `nb_inv_tab` et les questions précédentes, écrire une fonction récursive `nb_inversions_rec(tab)` qui permet de calculer le nombre d'inversions dans un tableau. Cette fonction renverra le même nombre que `nombre_inversions(tab)` de la partie A. On procédera de la façon suivante :

- Séparer le tableau en deux tableaux de tailles égales (à une unité près).
- Appeler récursivement la fonction `nb_inversions_rec` pour compter le nombre d'inversions dans chacun des deux tableaux.
- Trier les deux tableaux (on rappelle qu'une fonction de tri est déjà définie).
- Ajouter au nombre d'inversions précédemment comptées le nombre renvoyé par la fonction `nb_inv_tab` avec pour arguments les deux tableaux triés.