TP n° 4 – Séquences mutables, recherche et premier tri

Rappels

— Enregistrez dès le début du TP votre travail avec un nom de la forme `VOTRENOM_TPX.py` (évitez les accents et caractères spéciaux) ;
— une séquence (liste, tuple ou chaîne de caractères) se parcourt soit

avec ses indices :

```
for i in range(len(L)):
    L[i]
```

par éléments :

```
for e in L:
    e
```

les deux simultanément :

```
for i,e in enumerate(L):
    e==L[i]
```

— une coupe des éléments de `L` d'indices `a` à `b` se note `L[a:b+1]` ;
— la méthode `append` permet d'ajouter un élément à une liste ;
— la méthode `pop` agit sur place, sur une liste, et renvoie la valeur supprimée ;
— pour les objets mutables, l'opération `=` recopie les adresses des pointeurs, pas les éléments ;
— il faut utiliser la fonction `deepcopy` de la bibliothèque `copy` pour créer une vraie copie d'une liste ;
— *Out Of Range* indique que l'indice demandé excède la dimension de la séquence.

Exercice 4.1 (Mutable *or not* ?)
Dans cet exercice, on va « jouer » avec les listes. Pour ne pas vous faire piéger, il est nécessaire de redéfinir un nouveau *shell* à chaque question.

1. Recopier le code suivant

```
L = list(range(11))
M = L
M[2] = 666
```

et prévoir ce que renvoie `L[2]`.

2. Recopier le code suivant

```
L = list(range(11))
M = [e for e in L]
M[2] = 666
```

et prévoir ce que renvoie `L[2]`.

3. Recopier le code suivant

```
A = list(range(11))
B = list(range(11,21))
C = list(range(21,31))
L = [A, B, C]
M = [e for e in L]
M[1].append(666)
```

et prévoir ce que renvoie `L` et `B`.

4. Recopier le code suivant

```
import copy
A = list(range(11))
B = list(range(11,21))
C = list(range(21,31))
L = [A, B, C]
M = copy.deepcopy(L)
M[1].append(666)
```

et prévoir ce que renvoient `L` et `B`.

5. Recopier le code suivant

```
L = list(range(11))
M = [L,L,L]
L[2] = 666
```

et prévoir ce que renvoie `[e[2] for e in M]`. Montrer que `[L,L,L]` et `3*[L]` sont égales.

6. Recopier le code suivant

```
L = [[]]*3
L[0].append(2)
```

et prévoir ce que renvoie `L`.

Exercice 4.2 (Mutation)
Dans cet exercice, on vous demande d'implémenter les fonctions suivantes sur des listes. On dit qu'une fonction agit « sur place » si elle modifie la liste donnée en argument et ne renvoie pas d'autre liste.

1. Écrire une fonction `Nettoyer` qui prend comme arguments une liste `L` et un objet `a`, puis renvoie la liste obtenue en retirant de `L` tous les termes égaux à `a`.
2. Écrire une fonction `Permuter` qui prend comme arguments une liste `L` et deux indices d'objets à permuter `i` et `j` et qui réalise « sur place » la permutation $L_i \leftrightarrow L_j$.
3. Écrire une fonction `Inverser` qui prend comme argument une liste `L` de la forme $[x_1, \dots, x_n]$ et qui inverse « sur place » ses éléments afin d'obtenir $[x_n, \dots, x_1]$.
4. Écrire une fonction `palindrome` qui prend comme argument une liste `L`, puis renvoie le booléen `True` si celle-ci est égale à son inverse, et `False` sinon.

Exercice 4.3 (Recherche par dichotomie dans une liste triée)
On dit qu'une liste est *triée* si ses termes sont rangés dans l'ordre croissant. Par exemple, la liste `[1.3,8.56,10.2]` est triée alors que `[-12.86,186.2,151,18.7]` ne l'est pas.

1. Écrire une fonction `Triee` qui prend comme argument une séquence `L` est qui renvoie le booléen `True` si elle est triée, `False` sinon.
2. Écrire une fonction `IndiceObjetListeTriee` qui prend en entrée une liste triée L et un objet x et qui renvoie la position de l'objet dans la liste avec le moins de calculs possibles.

Exercice 4.4 (Initiation au tri)
Pour trier une liste, on se propose d'implémenter le **tri à bulles**. Son fonctionnement s'appuie sur des permutations répétées d'éléments contigus qui ne sont pas dans le bon ordre. Le principe est parcourir la séquence, à partir de la fin, en comparant deux éléments et en les mettant éventuellement dans le bon ordre (voir figure 1). Le parcours est répété tant qu'il y a eu au moins une inversion dans le parcours précédent. L'absence d'inversion assure que chaque élément est inférieur ou égal au suivant, et donc que la séquence (de gauche) est triée.

1. Écrire une fonction `TriBulles` qui prend comme argument une liste `L` et qui trie la liste « sur place » avec l'algorithme du tri à bulles.
2. Donner l'ordre de grandeur du nombre d'opération réalisées pour une liste de longueur n dans le meilleur et le pire des cas.

* *
*

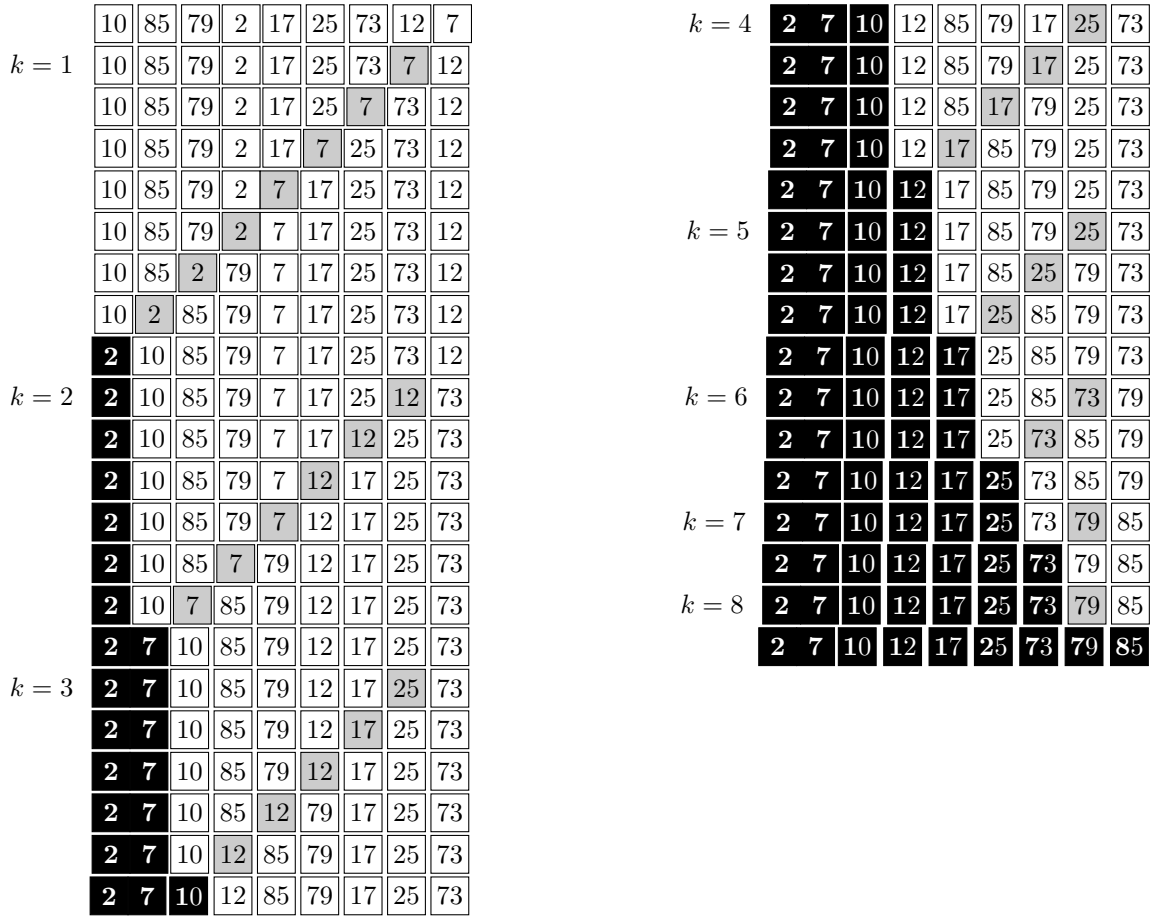

FIGURE 1 – Fonctionnement du tri à bulles avec la séquence $\{10, 85, 79, 2, 17, 25, 73, 12, 7\}$. À chaque itération $k \in [\![1, 8]\!]$, la case grise représente le résultat de la comparaison des couples, en noir la portion de séquence déjà triée.