TP 09

Tri

Savoirs et compétences :

- ❒ *Th. 8 : Tris.*

Activité 1 – Tri pas comptage d'une liste

On suppose que la liste à trier `L` est constituée d'entiers de l'intervalle $[\![0;k[\![$. L'algorithme fonctionne suivant le principe suivant. On parcourt une fois la liste et on compte le nombre d'éléments de la liste égaux à 0, 1, ..., $k-1$. Pour ce faire on utilise une liste `C` de taille k. On peut alors facilement procéder à une réécriture de la liste initiale, de sorte qu'en sortie elle soit constituée des mêmes éléments, mais triés dans l'ordre croissant.

Ainsi, si on cherche à trier la liste `L=[2,1,4,1]`. Dans un premier temps on crée une liste `C` contenant cinq fois la valeur 0 : `C=[0,0,0,0,0]`. Une fois le comptage terminé on obtient la liste suivante : `C=[0,2,1,0,1]`. La liste triée sera donc constituée de 2 fois la valeur 1 puis 1 fois la valeur 2 puis une fois la valeur 4.

L'algorithme prend en entrée la liste L à trier, ainsi qu'un entier k tel que tous les éléments de la liste soient des entiers de l'intervalle $[\![0;k[\![$. On procède en deux étapes : d'abord compter les éléments de chaque type, ensuite réécrire la liste L.

Question 1 *Ecrire la fonction on pourra (au choix) utiliser l'une des signatures suivante :* `tri_comptage(L:list,k:int) -> None` *ou* `tri_comptage(L:list,k:int) -> list` *permettant de réaliser un tri par comptage (avec ou sans effet de bord).*

Question 2 *La fonction proposée agit-elle avec effet de bord ? Sans effet de bord ?*

Question 3 *La fonction proposée réalise-t-elle un tri stable ? un tri en place ?*

Activité 2 – Classement de l'étape 18 Embrun – Valloire – 208 km du tour de France

On donne la bibliothèque de tri `tris.py` dans laquelle différents tris ont été implémentés. On dispose ainsi des fonctions :

- `tri_insertion`;
- `tri_rapide`;
- `tri_fusion`.

Pour augmenter la limite de récursivité de Python, on utilisera les instructions suivantes :

```
import sys
sys.setrecursionlimit(100000).
```

Les coureurs du tour de France sont en train de terminer la dix huitième étape du Tour de France qui sépare Embrun et Valloire.

Le fichier `classement_general.txt` rassemble le classement général à l'issue de l'étape 17. Le fichier `etape_18.txt` contient le classement de l'étape 18 uniquement. Dans le fichier texte, les champs sont séparés par des tabulations.

Objectif L'objectif est de réaliser le classement général après la dix huitième étape.

Lecture des fichiers de résultat

Question 4 *Écrire la fonction* `chargeClassement(fichier:str)->list` *permettant de lire un fichier de classement et de renvoyer une liste de la forme* `[[Nom_1, Dossard_1, Temps_1], [Nom_2, Dossard_2, Temps_2], ...]`.

[['JULIAN ALAPHILIPPE ', '21', "69h 39' 16""]]

Question 5 *Écrire la fonction* `convertirTemps(temps:str)->int` *qui évalue le temps exprimé en heure, minutes et secondes en une valeur en seconde.*

convertirTemps("69h 39' 16"")=250756s

Question 6 *Écrire la fonction* `classement(fichier:str)->list` *permettant de lire un fichier de classement et de renvoyer une liste de la forme* `[[Nom_1, Dossard_1, Temps_1], [Nom_2, Dossard_2, Temps_2], ...]` *les temps exprimés en seconde. Vous devez utiliser les deux fonctions précédentes.*

LG[:1] affiche [['JULIAN ALAPHILIPPE ', '21', 250756]] et L18[:] affiche [['NAIRO QUINTANA ', '61', 20055]]

Classement en fin d'étape

Dans une première approche, on souhaite réaliser le classement général après la fin de l'étape 18.

R Pour faire une copie complète d'une liste de listes, il faut utiliser la fonction `deepcopy` du module `copy`. C'est une fonction récursive (ou profonde) qui construit un nouvel objet composé, puis récursivement, insère dans l'objet composé des copies des objets trouvés dans l'objet original. Attention les objets récursifs (objets composés qui, directement ou indirectement, contiennent une référence à eux-mêmes) peuvent causer une boucle récursive infinie.

Question 7 *Réaliser la fonction* `ajoutTemps(liste1:list,liste2:list)->list` *qui à partir des deux listes du classement de l'étape 18 et du classement général renvoie la liste de la forme* `[[Nom_1, Dossard_1, Temps_1], [Nom_2, Dossard_2, Temps_2], ...]` *dont les temps sont la somme des temps des deux listes pour chaque coureur.*

['NAIRO QUINTANA ', '61', 271381]

Question 8 *Quelle méthode de tri vous semble la mieux adaptée au tri du classement général ?*

Question 9 *Modifier les algorithmes de tris pour pouvoir trier la liste obtenue en sortie de la fonction* `ajoutTemps(liste1:l` *suivant le temps de course d'un coureur. Le classement général a-t-il changé à l'issue de la dix huitième étape ?*

[['JULIAN ALAPHILIPPE ', '21', 271129], ['EGAN BERNAL ', '2', 271219], ['GERAINT THOMAS ', '1', 271224]]

R Travaillant sur une liste de listes, la méthode `sort` n'est plus adaptée. On peut donc utiliser la méthode `sorted` en utilisant une clef de tri (la clef correspondant à la colonne sur laquelle on souhaite trier la liste), tri de la liste `Liste` sur la colonne `i` :

```
sorted(Liste, key=lambda colonnes: colonnes[i])
```

Classement en cours d'étape – Implémentation d'une file

On cherche à reconstituer le classement général au fur et à mesure que les coureurs arrivent. Dans cette partie le classement de l'étape (liste de listes) sera vu comme une **file** FIFO (First In First Out) où le premier élément est le premier coureur arrivé et le dernier élément est le dernier coureur à avoir passé la ligne d'arrivée.

Question 10 *Implémenter les fonctions élémentaires liées à la gestion des files :* `enfiler`, `defiler`, `est_vide`. *À l'intérieur de ces fonctions, on s'autorise les méthodes liées aux listes (*`append`, `pop`, *...).*

Question 11 *Implémenter la fonction* `ajout` *ayant pour but d'ajouter le temps de l'étape d'un coureur dans le classement général et de mettre à jour ce classement. La gestion du classement de l'étape devra être réalisé grâce à une liste.*

Question 12 *Quelle pourrait être l'utilité de la fonction* `enfiler` *dans un tel contexte ?*

ANNEXE : opérations et fonctions Python disponibles

Pour la copie de liste de listes, le module copy avec la fonction deepcopy sont efficaces.
Ci-dessous un exemple avec la fonction copy du module copy et avec la fonction deepcopy du module copy.

```
from copy import copy, deepcopy

L = [['JULIAN ALAPHILIPPE ', '21', 250756], ['GERAINT THOMAS ', '1', 250851],['STEVEN ↙
    KRUIJSWIJK ', '81', 250863]]

L_copy = copy(L) # copie superficielle
L_deepcopy = deepcopy(L) # copie profonde

L[1][0] = 5

copy: [['JULIAN ALAPHILIPPE ', '21', 250756], [5, '1', 250851],
['STEVEN KRUIJSWIJK ', '81', 250863]]
deepcopy: [['JULIAN ALAPHILIPPE ', '21', 250756], ['GERAINT THOMAS ', '1', 250851],
['STEVEN KRUIJSWIJK ', '81', 250863]]
```