TP 09

Tri

Savoirs et compétences :

- ❒ *Th. 8 : Tris.*

Consignes

Attention : suivez précisément ces instructions.

Votre fichier portera un nom du type

`tp09_durif_berne.py`,

où les noms de vos enseignants sont à remplacer par ceux des membres du binôme. Le nom de ce fichier ne devra comporter ni espace, ni accent, ni apostrophe, ni majuscule. Dans ce fichier, vous respecterez les consignes suivantes.

- Écrivez d'abord en commentaires (ligne débutant par #), le titre du TP, les noms et prénoms des étudiants du groupe.
- Commencez chaque question par son numéro écrit en commentaires.
- Les questions demandant une réponse écrite seront rédigées en commentaires.
- Les questions demandant une réponse sous forme de fonction ou de script respecteront pointilleusement les noms de variables et de fonctions demandés.

Les éventuelles figures seront sauvegardées sous le nom `tp09_Qnum_nom1_nom2.png`, où Qnum est le numéro de la question, et nom1, nom2 les noms des membres du binôme.

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

On donne les fonctions suivante

- `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], ...]`.
- `convertirTemps(temps:str)->int` qui évalue le temps exprimé en heure, minutes et secondes en une valeur en seconde.

- `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.

```
def chargeClassement(fichier):
    f=open(fichier,'r')
    fichier=f.readlines()
    f.close() #ne pas oublier de fermer le fichier...
    L=[]
    for ligne in fichier:
        ligne=ligne.split('\t') # coupe aux tabulations
        L1=[]
        L1.append(ligne[1])
        L1.append(ligne[2])
        L1.append(ligne[4])
        L.append(L1)
    return L
```

```
def convertirTemps(temps:str):
    '''temps est un str de la forme "06h 09' 39''" '''
    t_course=temps.split('h') #on peut aussi couper a h
    heure=int(t_course[0])
    t_course2=t_course[1].split("'")
    duree=int(t_course2[1])+60*int(t_course2[0])+3600*heure
    return duree
```

```
def classement(fichier):
    L=chargeClassement(fichier)
    for element in L:
        element[2]=convertirTemps(element[2])
    return L
```

Classement en fin d'étape

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 4 *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 5 *Quelle méthode de tri vous semble la mieux adaptée au tri du classement général ?*

Question 6 *Modifier un algorithme de tri au choix pour pouvoir trier la liste obtenue en sortie de la fonction* `ajoutTemps(L1,L2)` *selon la dernière colonne. On l'appellera* `tri_modifie(liste)->list`.

Question 7 *Ecrire une fonction* `update_classement_general(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], ...]` *triée dans l'ordre du nouveau classement général.* *[['JULIAN ALAPHILIPPE ', '21', 271129], ['EGAN BERNAL ', '2', 271219], ['GERAINT THOMAS ', '1', 271224]]*

0.1 Comparaison des temps de traitement

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])`
- On peut utiliser le module `time` qui permet de mesurer le temps que met une fonction pour effectuer une tâche. Vous pourrez tester l'exemple suivant :

```
import time as t
tic=t.time()
for k in range(10):
       print(k)
toc=t.time()
print(toc-tic)
```

Question 8 *Utiliser le module* `time` *pour comparer les temps de traitement pour trier le classement à l'issu de la dernière étape selon votre algorithme de tri modifié et selon la fonction* `sorted`.

Question 9 *Comparer les différentes méthodes de tri en traçant en fonction de la taille du tableau à traiter le temps nécessaire au tri des données. On pourra pourra pour cela utiliser un tranchage. Vous pourrez renvoyer la figure tracée à vos enseignants.*

On donne les instructions suivantes pour une liste d'entiers aléatoires.

```
import random
def generer_liste(n,N):
    x = [random.randint(0, n) for p in range(0, N)]
    return(x)

N=int(1e3)

L=generer_liste(N,N)
```

Question 10 *Facultatif : comparer les différentes méthodes de tri vues en cours appliquée sur une liste aléatoire (comme donnée précédemment) en traçant en fonction de la taille du tableau à traiter le temps nécessaire au tri des données. On pourra pourra pour cela utiliser un tranchage. Vous pourrez renvoyer la figure tracée à vos enseignants.*