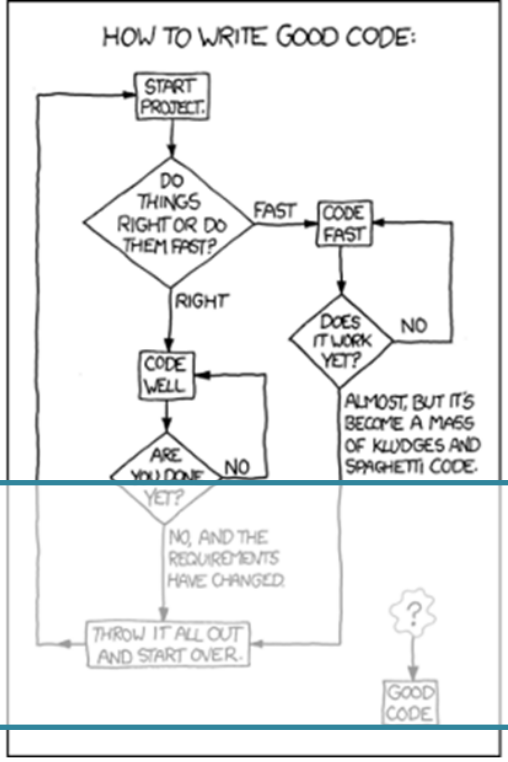

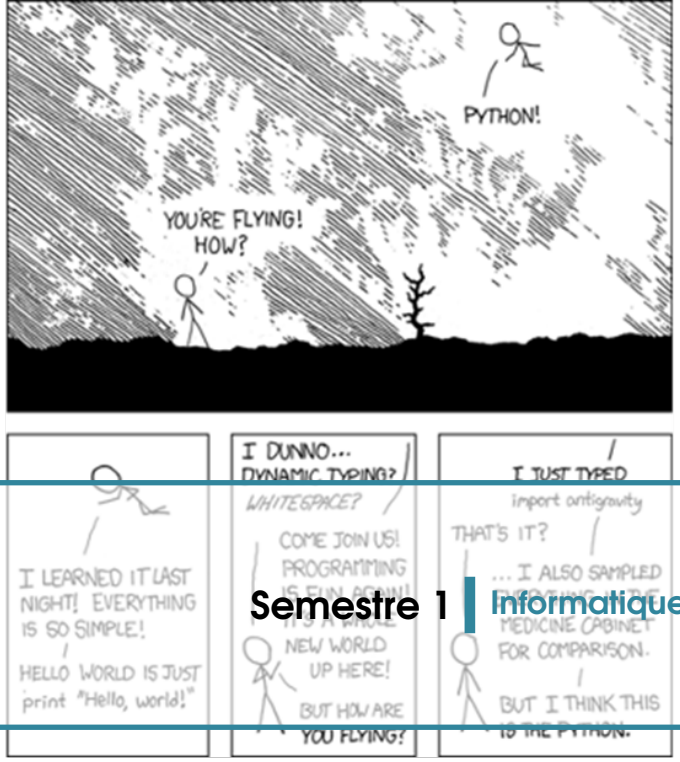

Semestre 1 | Informatique

Thèmes d'étude

J.-P. Berne – E. Durif – X. Pessoles

MPSI

Thème : Tris.
Commentaires :
- algorithmique quadratique : tri par insertion, par sélection ;
- tri par partition-fusion ;
- tri par comptage.

On fait observer différentes caractéristiques (par exemple stable ou non, en place ou non, comparatif ou non ...).

1 Présentation

Le tri de données ou de valeurs est omniprésent en informatique. Pour cela, beaucoup d'algorithmes ont été développés afin de réaliser des tris rapidement, notamment lorsque le nombre de données est important.

Définition Stabilité

Définition Tri en place
Un tri est effectué en place lorsque la liste à trier est modifiée jusqu'à devenir triée. Dans le cas contraire, la fonction de tri pourra renvoyer une novelle liste contenent les mêmes éléments, mais triés.

Définition Tri comparatif
Un tri est dit comparatif lorsqu'il s'appuie uniqument sur la comparaison deux à deux des éléménts de la liste et pas sur la valeur ce ces éléments.

2 Tris

Définition Tri par insertion

À partir d'une sous-liste triée, le tri par insertion consiste à parcourir les éléments non triés et de les insérer successivement dans la sous-liste déjà triée.

```
def insere(t, j):
    k, a = j, t[j]
    while k > 0 and a < t[k-1]:
        t[k] = t[k-1]
        k = k-1
    t[k] = a

def insertionSort(t):
    for j in range(1, len(t)):
        insere(t, j)
```

Définition Tri rapide

Soit une liste L non triée. Soit p un terme appelé pivot. Le tri rapide consiste à répartir les éléments strictement inférieur au pivot avant ce dernier et les termes plus grand après le pivot (segemntation). Le pivot est à ce stade trié correctement par rapport aux autres valeurs de la liste. Ce principe est alors appliqué récursivement aux deus-sous listes séparées par le pivot.

```
def segmente(t, i, j):
    p = t[j-1] # On prend comme pivot le dernier élément de la sous liste.
    a = i
    for b in range(i, j-1):
        if t[b] < p:
            t[a], t[b] = t[b], t[a]
            a += 1
    t[a], t[j-1] = t[j-1], t[a] # On positionne le pivot "à sa place".
    return a # On retourne l'index du pivot. Le tableau a été modifié en place.
```

```
def quickSort(t, i, j):
    if i + 1 < j:
        a = segmente(t, i, j)
        quickSort(t, i, a)
        quickSort(t, a + 1, j)
```

```
# Instruction pour trier une liste
quickSort(t, 0, len(t))
```

Définition Tri fusion
Il s'agit d'un tri s'appuyant sur la stratégie diviser pour régner.
L'algorithme est le suivant :
- on divise la liste en deux listes de tailles quasi-identiques ;
- on trie récursivement ces deux listes ;
- on fusionne les deux listes triées.

```
def placer(L :list, p :int, x) :
    """Place un élément x àsa place dans une liste L triée àpartir de l'indice p
    Entrée :
        L : une liste, p : un entier, x : un élément
    Sorties :
        La liste est modifiée mais n'est pas renvoyée. k la valeur de l'indice de la liste où l'
            élément a été placé"""
    k = p
    while ( k < len(L) and x > L[k]) :
        k = k+1
    L.insert(k, x)
    return k

def fusion(a:list, b:list) :
    """Fusionne les deux listes
    Entrée : deux listes a et b triées.
    Sortie : La liste b modifiée"""
    p = 0
    for x in a :
        p = placer(b, p, x)+1
    return b

def tri_fusion(t : list) :
    """Trie la liste t
    Entrée : une liste.
    Sortie : la liste est modifiée."""
    if len(t) < 2 :
        return (t)
    else :
        m = len(t) // 2
        return (fusion (tri_fusion(t[:m]) , tri_fusion(t[m:]) ))
```

Proposition **Complexité des algorithmes** On note $T(n)$ le nombre de comaraisons nécessaire pour trier une liste de longueur n. On montre que dans le pire des cas, les complexités sont les suivantes :
- tri par insertion : $T_{\text{Max}}(n) = \mathcal{O}\left(n^2\right)$;
- tru rapide : $T_{\text{Max}}(n) = \mathcal{O}\left(n^2\right)$;
- tru partition-fusion : $T_{\text{Max}}(n) = \mathcal{O}\left(n \log n\right)$.

3 Acticité préparatoire

Pour réaliser l'activité associée à ce cours, suivre le lien suivant :
- Sujet : `https://bit.ly/3iAc5do`
- Corrige : `https://bit.ly/3eFuHrt`

4 QCM

Question 1 *On dispose d'une liste de triplets : t = [(1,12,250),(1,12,251), (2,12,250),(2,13,250),(2,11,250),(2,12,249)]. On trie cette liste par ordre croissant des valeurs du second élément des triplets. En cas d'égalité, on trie par ordre croissant du troisième champ. Si les champs 2 et 3 sont égaux, on trie par ordre croissant du premier champ. Après ce tri, quel est le contenu de la liste ?*

1. `[(1,12,249),(1,12,250),(1,12,251),(2,11,250),(2,12,250),(2,13,250)]`.
2. `[(2,11,250),(1,12,249),(1,12,250),(2,12,250),(1,12,251),(2,13,250)]`.
3. `[(2,11,250),(1,12,249),(1,12,250),(1,12,251),(2,12,250),(2,13,250)]`.
4. `[(1,12,249),(2,11,250),(1,12,250),(2,12,250),(2,13,250),(1,12,251)]`.

Question 2 *Quelle valeur retourne la fonction "mystere" suivante ?*

```
def mystere(liste):
    valeur_de_retour = True
    indice = 0
    while indice < len(liste) - 1 :
        if liste[indice] > liste[indice + 1]:
            valeur_de_retour = False
        indice = indice + 1
    return valeur_de_retour
```

1. Une valeur booléenne indiquant si la liste liste passée en paramètre est triée.
2. La valeur du plus grand élément de la liste passée en paramètre.
3. La valeur du plus petit élément de la liste passée en paramètre.
4. Une valeur booléenne indiquant si la liste passée en paramètre contient plusieurs fois le même élément.

Question 3 *Combien d'échanges effectue la fonction Python suivante pour trier un tableau de 10 éléments au pire des cas ?*

```
def tri(tab) :
    for i in range (1, len(tab)) :
        for j in range (len(tab) - i) :
            if tab[j] > tab[j+1] :
                tab[j], tab[j+1] = tab[j+1], tab[j]
```

1. 45.
2. 100.
3. 10.
4. 55.

Question 4 *Que vaut l'expression* `f([7, 3, 1, 8, 19, 9, 3, 5], 0)` *?*

```
def f(t,i) :
    im = i
    m = t[i]
    for k in range(i+1, len(t)) :
        if t[k] < m :
            im, m = k, t[k]
    return im
```

1. 1.
2. 2.
3. 3.
4. 4.

Question 5 *Laquelle de ces listes de chaînes de caractères est triée en ordre croissant ?*

1. `['Chat' , 'Cheval' , 'Chien' , 'Cochon']`.
2. `['Cochon' , 'Chat' , 'Cheval' , 'Chien']`.
3. `['Cheval' , 'Chien‘ , 'Chat' , 'Cochon']`.
4. `['Chat' , 'Cochon' , 'Cheval' , 'Chien']`.

Question 6 *Laquelle de ces listes de chaînes de caractères est triée en ordre croissant ?*

1. ['12','142','21','8'].
2. ['8','12','142','21'].
3. ['8','12','21','142'].
4. ['12','21','8','142'].

Question 7 *Quelle est la valeur de la variable* `table` *après exécution du programme Python suivant ?*

```
table = [12, 43, 6, 22, 37]
for i in range(len(table) - 1):
    if table [i] > table [i+1] :
        table [i] ,table [i+1] = table [i+1] ,table [i]
```

1. [12, 6, 22, 37, 43].
2. [6, 12, 22, 37, 43].
3. [43, 12, 22, 37, 6].
4. [43, 37, 22, 12, 6].

Question 8 *Un algorithme cherche la valeur maximale d'une liste non triée de taille n. Combien de temps mettra cet algorithme sur une liste de taille $2n$?*

1. Le même temps que sur la liste de taille n si le maximum est dans la première moitié de la liste.
2. On a ajouté n valeurs, l'algorithme mettra donc n fois plus de temps que sur la liste de taille n.
3. Le temps sera simplement doublé par rapport au temps mis sur la liste de taille n.
4. On ne peut pas savoir, tout dépend de l'endroit où est le maximum.

Question 9 *Quel est le coût en temps dans le pire des cas du tri par insertion ?*

1. $\mathcal{O}(n)$.
2. $\mathcal{O}\left(n^2\right)$.
3. $\mathcal{O}\left(2^n\right)$.
4. $\mathcal{O}(\log n)$.

Question 10 *On souhaite écrire une fonction* `tri_selection(t)`*, qui trie le tableau* `t` *dans l'ordre croissant : parmi les 4 programmes suivants, lequel est correct ?*

```
def tri_selection(t) :
    for i in range (len(t)-1) :
        min = i
        for j in range(i+1,len(t)):
            if t[j] < t[min]:
                min = j
        tmp = t[i]
        t[i] = t[min]
        t[min] = tmp
def tri_selection(t) :
    for i in range (len(t)-1) :
        min = i
        for j in range(i+1,len(t)-1):
            if t[j] < t[min]:
                min = j
        tmp = t[i]
        t[i] = t[min]
        t[min] = tmp
def tri_selection(t) :
    for i in range (len(t)-1) :
        min = i
        for j in range(i+1,len(t)):
            if t[j] < min:
                min = j
        tmp = t[i]
        t[i] = t[min]
        t[min] = tmp
def tri_selection(t) :
    for i in range (len(t)-1) :
        min = i
        for j in range(i+1,len(t)):
            if t[j] < t[min]:
```

```
                    min = j
            tmp = t[i]
            t[min] = t[i]
            t[i] = tmp
```

1. Fonction 1.
2. Fonction 2.
3. Fonction 3.
4. Fonction 4.

Question 11 *De quel type de tri s'agit-il ?*

```
def tri(lst):
    for i in range(1,len(lst)):
        valeur = lst[i]
        j = i
        while j>0 and lst[j-1]>valeur:
            lst[j]=lst[j-1]
            j = j-1
        lst[j]=valeur
```

1. Tri par insertion.
2. Tri fusion.
3. Tri par sélection.
4. Tri à bulles.

Question 12 *De quel type de tri s'agit-il ?*

```
def tri(lst):
    nb = len(lst)
    for i in range(0,nb):
        ind_plus_petit = i
        for j in range(i+1,nb) :
            if lst[j] < lst[ind_plus_petit] :
                ind_plus_petit = j
        if ind_plus_petit is not i :
            temp = lst[i]
            lst[i] = lst[ind_plus_petit]
            lst[ind_plus_petit] = temp
```

1. Tri par insertion.
2. Tri fusion.
3. Tri par sélection.
4. Tri à bulles.

Question 13 *Un algorithme est en complexité quadratique. Codé en python, son exécution pour des données de taille 100 prend 12 millisecondes. Si l'on fournit des données de taille 200 au programme, on peut s'attendre à un temps d'exécution d'environ :*

1. 48 millisecondes.
2. 24 millisecondes.
3. 12 millisecondes.
4. 96 millisecondes.

Question 14 *À quel type tri correspond l'invariant de boucle ci-dessous :*

- tous les éléments d'indices 0 à $i-1$ sont déjà triés,
- tous les éléments d'indices i à n sont de valeurs supérieures à ceux de la partie triée.

1. Tri par insertion.
2. Tri fusion.
3. Tri par sélection.
4. Tri à bulles.

Question 15 *Quel est l'invariant de boucle qui correspond précisément à cet algorithme ?*

On considère un algorithme de tri par sélection, dans lequel la fonction `echanger(tab[i], tab[j])` effectue l'échange des ième et jième valeurs du tableau `tab`.

```
nom: tri_sélection

paramètre: tab, tableau de n entiers, n>=2

Traitement:
pour i allant de 1 àn-1:
    pour j allant de i+1 àn:
        si tab[j] < tab[i]:
            echanger(tab[i], tab[j])
renvoyer tab
```

1. Tous les éléments d'indice supérieur ou égal à i sont triés par ordre croissante.
2. Tous les éléments d'indice compris entre 0 et i sont triés et les éléments d'indice supérieurs ou égal à i leurs sont tous supérieurs.
3. Tous les éléments d'indice supérieur ou égal à i sont non triés.
4. Tous les éléments d'indice compris entre 0 et i sont triés, on ne peut rien dire sur les éléments d'indice supérieur ou égal à i.

Question 16 *Quel est le type de tri qui correspond à cet algorithme?*

```
nom: tri_mystere

paramètre: tab, tableau de n entiers, non trié, non vide

Traitement:
pour i allant de 1 àn-1:
    pour j allant de i+1 àn:
        si tab[j] < tab[i]:
            echanger(tab[i], tab[j])
renvoyer tab
```

1. Tri par insertion.
2. Tri fusion.
3. Tri par sélection.
4. Tri rapide.

Question 17 *Quel est l'invariant de boucle qui correspond précisément à cet algorithme?*

```
nom: tri_insertion

paramètre: tab, tableau de n entiers, n >= 2

Traitement:
pour i allant de 2 àn:
    j = i
    tant que j > 1 et tab[j-1] > tab[j]:
        echanger(tab[j-1], tab[j])
        j = j-1
renvoyer tab
```

1. Tous les éléments d'indice compris entre 0 et i sont triés et les éléments d'indice supérieurs ou égal à i leurs sont tous supérieurs.
2. Tous les éléments d'indice supérieur ou égal à i sont triés par ordre croissant.
3. Tous les éléments d'indice compris entre 0 et i sont triés, on ne peut rien dire sur les éléments d'indice supérieur ou égal à i.
4. Tous les éléments d'indice supérieur ou égal à i sont non triés par ordre croissants.

Question 18 *Parmi les propositions suivantes, quelle est celle qui ne correspond pas à une méthode de tri?*

1. Par sélection.
2. Par insertion.
3. Par rotation.
4. Par fusion.

5 TP

Exercice 1 – Tris d'une base de données des films de cinéma

Objectif Réaliser un tri numérique et un tri alphabétique à partir d'une base de données

On donne le fichier `films_martiniere_2018.csv` dans lequel un peu plus de 2000 films sont référencés avec le titre, l'année de création, le réalisateur et le box office.

Une proposition de lecture du fichier csv et de création de la liste de films est donnée ci-dessous et dans le fichier `lecture_fichier_csv.py`:

```
f=open('films_martiniere.csv','r')
ligne=f.readline()
fichier=f.readlines()
f.close()
L=[]
for ligne in fichier:
    ligne=ligne.replace('"','')
    ligne=ligne.split(';')
    ligne[-1]=ligne[-1].rstrip('\n')
    ligne[-1]=int(ligne[-1])
    ligne[1]=int(ligne[1])
    L.append(ligne)
```

Avec le logiciel Pyzo, vous devez avoir votre dossier visible dans la fenêtre `file browser` ou alors exécuter le fichier en cliquant droit sur l'onglet de votre nom de fichier et sélectionner *Exécuter en tant que script*. Votre fichier python et le fichier `films_martiniere.csv` doivent être dans le même dossier.

Question 1 *Commenter chaque ligne du fichier* `lecture_fichier_csv.py`. *Copier vos tests dans le script python.*

Question 2 *Choisir l'algorithme de tri le plus efficace pour trier le fichier de 2000 films (autre que* `sort`*). Copier l'algorithme choisi dans votre script et modifier-le afin qu'il puisse trier une liste de listes.*

Question 3 *Trier les films en fonction du box office. Quel est le film qui a été le plus vu au cinéma ?*

Question 4 *Définir la fonction* `comparer(L:list)` *qui a pour argument une liste* `L` *de deux mots* `mot1:str` *et* `mot2:str` *et qui renvoie cette liste triée par ordre alphabétique. Les mots de la liste seront écrits en lettres majuscules. Les titres de films peuvent comporter des chiffres.*

Question 5 *Implémenter un algorithme de tri alphabétique adapté au fichier* `films_martiniere_2018.csv`. *Quel est le titre du premier film de la liste ?*

Exercice 2 – Tris d'une base de données des films de cinéma

Objectif Réaliser un tri numérique et un tri alphabétique à partir d'une base de données

On donne le fichier `films_martiniere_2018.csv` dans lequel un peu plus de 2000 films sont référencés avec le titre, l'année de création, le réalisateur et le box office.

Une proposition de lecture du fichier csv et de création de la liste de films est donnée ci-dessous et dans le fichier `lecture_fichier_csv.py`:

```
f=open('films_martiniere.csv','r')
ligne=f.readline()
fichier=f.readlines()
f.close()
L=[]
for ligne in fichier:
    ligne=ligne.replace('"','')
    ligne=ligne.split(';')
    ligne[-1]=ligne[-1].rstrip('\n')
    ligne[-1]=int(ligne[-1])
    ligne[1]=int(ligne[1])
    L.append(ligne)
```

Avec le logiciel Pyzo, vous devez avoir votre dossier visible dans la fenêtre `file browser` ou alors exécuter le fichier en cliquant droit sur l'onglet de votre nom de fichier et sélectionner *Exécuter en tant que script*. Votre fichier python et le fichier `films_martiniere.csv` doivent être dans le même dossier.

Question 1 *Commenter chaque ligne du fichier* `lecture_fichier_csv.py`. *Copier vos tests dans le script python.*

Commentaires dans le code ci-dessous :

```
def lectureCSV(nom) :
    f=open(nom,'r') # ouverture du fichier texte
    ligne1=f.readline() # on sauvegarde dans ligne1 la premiere ligne
    fichier=f.readlines() # on stocke tout le contenu du fichier texte sans la premiere ligne
        dans fichier
    f.close() #ne pas oublier de fermer l'acces a l'objet

    Lfilm=[]
    for ligne in fichier:
        ligne=ligne.replace('"','')# enlever les guillemets
        ligne=ligne.split(';')# creation d'une liste en coupant au niveau des ;
        ligne[-1]=ligne[-1].rstrip('\n')
        ligne[-1]=int(ligne[-1]) # remplacer la chaine de caracteres par un entier
        ligne[1]=int(ligne[1]) # remplacer la chaine de caracteres par un entier
        Lfilm.append(ligne)
    return(Lfilm)
```

Question 2 *Choisir l'algorithme de tri le plus efficace pour trier le fichier de 2000 films (autre que* `sort`*). Copier l'algorithme choisi dans votre script et modifier-le afin qu'il puisse trier une liste de listes.*

Le tri rapide propose un bon compromis élégance, complexité mémoire et temporelle. Il est donc nécessaire de modifier notre fonction de tri rapide afin de lui passer en argument sur quel critère le tri va être réalisé. La difficulté principale réside dans la re-construction de liste de liste (ne pas oublier les crochets).

Question 3 *Trier les films en fonction du box office. Quel est le film qui a été le plus vu au cinéma ?*

```
def tri_rapide_modif(liste,index):
    # uniquement utilisable avec une liste de listes
    if liste==[]:
        return([])
    grand=[]
    petit=[]
    pivot=liste[0][index] #modification pour acceder au bon element
    for i in range(len(liste[1:])):#on change la maniere d'iterer (obligatoire)
        elmt=liste[i+1][index] #on cherche l'element en cours
        if elmt>pivot:
            grand+=[liste[i+1]] # Ne pas oublier les crochets pour generer une liste
        else:
            petit+=[liste[i+1]] # Idem
    return(tri_rapide_modif(petit,index)+[liste[0]]+tri_rapide_modif(grand,index)) # idem
```

```
Lfilm=tri_rapide_modif(Lfilm,3)
print(Lfilm[-1])
Lfilm=tri_rapide_modif(Lfilm,0)
print(Lfilm[0])
```

La réponse à cette question est donc le film "INTOUCHABLES".

Remarque : si vous êtes bien attentifs notre fonction renvoie comme réalisateur 'ERIC TOLEDANO'. Cela ne sera pas le réalisateur renvoyé par les tris modifiés précédemment.

Question 4 *Définir la fonction* `comparer(L:list)` *qui a pour argument une liste* `L` *de deux mots* `mot1:str` *et* `mot2:str` *et qui renvoie cette liste triée par ordre alphabétique. Les mots de la liste seront écrits en lettres majuscules. Les titres de films peuvent comporter des chiffres.*

En fait cette question ne présente aucune difficulté puisque notre code vu en cours gère très bien l'ordre alphabétique, et même plus que ça, puisqu'est inclus un ordre alphanumérique grâce au codage ASCII ! Donc tout caractère pourra être géré. La seule ligne de commande `print(tri_rapide_modif(Lfilm,0)[0])` nous donne l'information recherchée : "71".

Question 5 *Implémenter un algorithme de tri alphabétique adapté au fichier* `films_martiniere_2018.csv`. *Quel est le titre du premier film de la liste ?*

Voici le code pour le tri par insertion :

```
def tri_insertion_modif(liste,index):
    for inc in range(1,len(liste)):
        encours=inc-1
        temp=liste[inc]
        tempindex=liste[inc][index] #sotckage de la valeur liee a l'index
        while encours>=0 and liste[encours][index]>tempindex: #ici il faut modifier
            liste[encours+1]=liste[encours]
            encours-=1
        liste[encours+1]=temp
```

Et celui pour le tri par fusion

```
def fusion_modif(gauche,droite,index):
    resultat = []
    index_gauche, index_droite = 0, 0
    while index_gauche < len(gauche) and index_droite < len(droite):
        if gauche[index_gauche][index] <= droite[index_droite][index]: # seule ligne a modifier
            resultat.append(gauche[index_gauche])
            index_gauche += 1
        else:
            resultat.append(droite[index_droite])
            index_droite += 1
    if gauche:
        resultat.extend(gauche[index_gauche:])
    if droite:
        resultat.extend(droite[index_droite:])
    return resultat

def tri_fusion_modif(liste,index):
    # modification uniquement sur les arguments des appels de fonctions
    if len(liste) <= 1:
        return liste
    milieu = len(liste) // 2
    gauche = liste[:milieu]
    droite = liste[milieu:]
    gauche = tri_fusion_modif(gauche,index)
    droite = tri_fusion_modif(droite,index)
    return list(fusion_modif(gauche, droite,index))
```

Et pour déterminer le film à succès ainsi que le premier de la liste :

```
tri_insertion_modif(Lfilm,3)
print(Lfilm[-1])
tri_insertion_modif(Lfilm,0)
print(Lfilm[0])

Lfilm=tri_fusion_modif(Lfilm,3)
print(Lfilm[-1])
Lfilm=tri_fusion_modif(Lfilm,0)
print(Lfilm[0])
```

Remarque : ici le réalisateur retourné est ici 'OLIVIER NAKACHE'...mince pourquoi cette différence? Tout simplement à cause de la gestion différente des doublons selon les fonctions proposées. On pourrait améliorer notre code en passant en argument une liste comme index...

Exercice 3 – Tri à bulles Pour trier une liste selon la méthode du tri à bulles, on réalise des balayages successifs : à chaque balayage, on compare les éléments du tableau 2 à 2 et on les réordonne.

Question 1 *En utilisant la liste* `[10, 3, 7, 5, 9, 7, 8, 0, 8]`*, écrire la séquence d'échanges permettant d'arriver à la liste triée.*

Question 2 *Donner un algorithme naïf permettant de trier un algorithme selon la méthode du tri à bulles.*

Question 3 *Dans quel cas est-on dans le meilleur des cas? Quelle alors la complexité de l'algorithme?*

Question 4 *Dans quel cas est-on dans le pire des cas? Quelle alors la complexité de l'algorithme?*

Question 5 *En remarquant qu'à l'étape i, les i derniers éléments sont triés, proposer un nouvel algorithme du tri à bulles.*

Question 6 *En remarquant qu'à l'itération i, il est possible d'arrêter le tri là où la dernière inversion a eu lieu à l'étape $i-1$, proposer un nouvel algorithme du tri à bulles.*

Exercice 4 – Tri à bulles Pour trier une liste selon la méthode du tri à bulles, on réalise des balayages successifs : à chaque balayage, on compare les éléments du tableau 2 à 2 et on les réordonne.

Question 1 *En utilisant la liste* `[10, 3, 7, 5, 9, 7, 8, 0, 8]`*, écrire la séquence d'échanges permettant d'arriver à la liste triée.*

Première série de comparaison sur la liste faisant monter 10 : [10, 3, 7, 5, 9, 7, 8, 0, 8], [3, 10, 7, 5, 9, 7, 8, 0, 8], [3, 7, 10, 5, 9, 7, 8, 0, 8], [3, 7, 5, 10, 9, 7, 8, 0, 8], [3, 7, 5, 9, 10, 7, 8, 0, 8], [3, 7, 5, 9, 7, 10, 8, 0, 8], [3, 7, 5, 9, 7, 8, 10, 0, 8], [3, 7, 5, 9, 7, 8, 0, 10, 8], [3, 7, 5, 9, 7, 8, 0, 8, 10]
Deuxième série de comparaison à partir des deux premiers éléments : [3, 7, 5, 9, 7, 8, 0, 8, 10], [3, 5, 7, 9, 7, 8, 0, 8, 10], [3, 5, 7, 7, 9, 8, 0, 8, 10], [3, 5, 7, 7, 8, 9, 0, 8, 10], [3, 5, 7, 7, 8, 0, 9, 8, 10], [3, 5, 7, 7, 8, 0, 8, 9, 10]
Troisième série de comparaison à partir des deux premiers éléments : [3, 5, 7, 7, 8, 0, 8, 9, 10], [3, 5, 7, 7, 0, 8, 8, 9, 10]
Quatrième série de comparaison à partir des deux premiers éléments : [3, 5, 7, 7, 0, 8, 8, 9, 10], [3, 5, 7, 0, 7, 8, 8, 9, 10]
Cinquième série de comparaison à partir des deux premiers éléments : [3, 5, 7, 0, 7, 8, 8, 9, 10], [3, 5, 0, 7, 7, 8, 8, 9, 10]
Sixième série de comparaison à partir des deux premiers éléments : [3, 5, 0, 7, 7, 8, 8, 9, 10], [3, 0, 5, 7, 7, 8, 8, 9, 10]
Septième série de comparaison à partir des deux premiers éléments : [3, 0, 5, 7, 7, 8, 8, 9, 10], [0, 3, 5, 7, 7, 8, 8, 9, 10]
Huitième série de comparaison à partir des deux premiers éléments : [0, 3, 5, 7, 7, 8, 8, 9, 10]

Question 2 *Donner un algorithme naïf permettant de trier un algorithme selon la méthode du tri à bulles.*

```
def tri_bulles_naif(l):
    for i in range(len(l)):
        for j in range(len(l)-1): #on ne peut pas comparer le dernier élément avec un suivant
            if l[j]>l[j+1]:
                l[j],l[j+1]=l[j+1],l[j]
```

Question 3 *Dans quel cas est-on dans le meilleur des cas ? Quelle alors la complexité de l'algorithme ?*

Quand la liste est triée, il n'y a pas d'échange d'élément. On a qu'en même les deux boucles `for` imbriquées.

Question 4 *Dans quel cas est-on dans le pire des cas ? Quelle alors la complexité de l'algorithme ?*

Quand la liste est triée dans le sens inverse, il y a échange de tous les éléments. On a qu'en même les deux boucles `for` imbriquées.

Question 5 *En remarquant qu'à l'étape i, les i derniers éléments sont triés, proposer un nouvel algorithme du tri à bulles.*

```
def tri_bulles(l):
    for i in range(0,len(l)-1):
        for j in range(0,len(l)-i-1):
            if l[j]>l[j+1]:
                l[j],l[j+1]=l[j+1],l[j]
```

Question 6 *En remarquant qu'à l'itération i, il est possible d'arrêter le tri là où la dernière inversion a eu lieu à l'étape $i-1$, proposer un nouvel algorithme du tri à bulles.*

```
def tri_bulles_optimise(tableau):
    permutation = True
    passage = 0
    while permutation == True:
        permutation = False
        passage = passage + 1
        for en_cours in range(0, len(tableau) - passage):
            if tableau[en_cours] > tableau[en_cours + 1]:
                permutation = True
                tableau[en_cours], tableau[en_cours + 1] = tableau[en_cours + 1],tableau[en_cours
                    ]
```

Exercice 5 – Classement de l'étape 18 Embrun – Valloire – 208 km

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 1 *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 2 *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 3 *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 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 les algorithmes de tris pour pouvoir trier la liste obtenue en sortie de la fonction* `ajoutTemps(liste1:list` *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 7 *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 8 *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 9 *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]]
```

Exercice 6 – Classement de l'étape 18 Embrun – Valloire – 208 km

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 1 *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""]]

```
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
```

Question 2 *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

```
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
```

Question 3 *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]]

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

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 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]

```
def ajoutTemps(L1,L2):
    '''L1 est la classement de l'etape et L2 le classement general certains ont ete
    disqualifies ou ont abandonne'''
    assert len(L1)<=len(L2)
    Lnew=deepcopy(L1)
    n=len(Lnew)
    for i in range(n):
        d=Lnew[i][1]
        j=0
        while d!=L2[j][1]: #attention certains n'ont pas fait l'etape completement
        # et sont disqualifies
            j+=1
        Lnew[i][-1]=Lnew[i][-1]+L2[j][-1]
    return Lnew
```

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

Question 6 *Modifier les algorithmes de tris pour pouvoir trier la liste obtenue en sortie de la fonction* `ajoutTemps(liste1:list` *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]]

```
def tri_insertion_modifie(tab):
    '''Trie la liste t
```

```
    Entree :
        Une liste
    Sortie :
        La liste est modifiee mais n est pas renvoyee'''
    for i in range(1,len(t)) :
        element=t[i]
        x=t[i][-1]
        k=0
        while (k<i and x>t[k][-1]) :
            k=k+1
        for j in range(i,k,-1) :
            t[j]=t[j-1]
        t[k]=element
```

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 7 *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`, *...).*

```
def enfiler(file, element):
    return file.append(element)

def defiler(file):
    return file.pop(0)

def est_vide(file):
    return len(file)==0
```

Question 8 *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.*

```
def ajout(L_etape_triee,LG):
    ''' ajout au classement general le temps de la nouvelle etape et refait le classement'''
    new_classement=[]
    while est_vide(L_etape_triee)==False: #tant que la file n'est pas vide
        cycliste=defiler(L_etape_triee)# on prend le premier element et on l'enleve
        i=0
        new_classement.append(cycliste)# on place le nouvel arrivant a la queue de la liste
                                       # triee du classement general
        while cycliste[1]!=LG[i][1]: #on cherche le meme dossard de cycliste
            i=i+1
        new_classement[-1][-1]=new_classement[-1][-1]+LG[i][-1] #on additionne les temps du
                                            # classement general et du classement d'etape
        tri_insertion_modifie(new_classement) #on trie le nouveau classement
    return new_classement
```

Question 9 *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]]
```