Informatique

8
Tris

TD 8-4
Tri rapide

Exercice 1: Tri rapide

Tri avec liste auxiliaire

Nous supposerons que le pivot est choisi naïvement comme la première valeur de la liste.

Question 1: Mettre en place une fonction $f\_combine(Listes)$ où $Listes$ est une liste de listes de taille quelconque, qui renvoie la liste combinée de gauche à droite des listes de $Listes$

Exemple :

```
>>> f_combine([[1,2,3],[4],[5,6],[5,4]])
[1, 2, 3, 4, 5, 6, 5, 4]
```

Question 2: Mettre en place une fonction $f\_divise(L)$ qui renvoie une liste de 3 listes de la forme $[L_1, [Pivot], L_2]$ où L_1 est la liste des éléments strictement inférieurs au pivot et L_2 la liste des éléments supérieurs ou égaux à celui-ci

Remarque : ne pas inclure le premier terme (pivot) dans L1…
Exemple :

```
>>> f_divise([5,2,9,4,7,3,6])
[[2, 4, 3], [5], [9, 7, 6]]
```

Question 3: Mettre en place une fonction $f\_tri\_rapide\_aux(L)$ qui renvoie une liste triée de L avec listes auxiliaires par récursivité en utilisant les deux fonctions précédentes

Erreur « concatenate » ? Combinez-vous bien 3 listes ?

Essayez :

```
>>> L = [1,2,3,1.0]

>>> f_tri_rapide_aux(L)
[1, 1.0, 2, 3]
```

Question 4: Si votre tris n'est pas stable, améliorer la fonction Divise afin d'obtenir la stabilité

La suite de ce TP ne sera réalisée que si vous avez de l'avance

Tri en place

Ce tri est un peu plus difficile à mettre en place (haha). Une petite remarque, j'ai décidé d'inclure j dans les fonctions suivantes, il arrive que l'on décide de ne pas l'inclure, il faudra savoir s'adapter.

Question 5: Mettre en place une fonction $f\_tri\_rapide\_loc(L,i,j)$ qui réalise l'opération de tri en place entre les indices i et j inclus de la liste L et qui renvoie l'indice q du pivot ainsi placé

Remarque : Il n'y a pas de cas de base à traiter ici, ce n'est pas récursif, cela viendra après ! Quand cette fonction sera appelée, c'est qu'il y aura du travail à réaliser.

Vérifiez l'exemple ci-contre :

```
>>> L=[5,1,6,9,3,2,4,7]

>>> f_tri_rapide_loc(L,0,7)
4

>>> L
[4, 1, 3, 2, 5, 9, 6, 7]
```

Le premier terme est le pivot, il vaut 5. Il est déplacé en position d'indice 4, les termes inférieurs sont avant, les termes supérieurs après…

Si besoin, voici ci-contre un descriptif des étapes intermédiaires :

5	1	6	9	3	2	4	7
5	1	6	9	3	2	4	7
5	1	6	9	3	2	4	7
5	1	6	9	3	2	4	7
5	1	3	9	6	2	4	7
5	1	3	2	6	9	4	7
5	1	3	2	4	9	6	7
5	1	3	2	4	9	6	7
4	1	3	2	5	9	6	7

Et.. juste au cas où, essayez :

```
>>> L = [5,1,6,9,3,2,4,7,5]

>>> f_tri_rapide_loc(L,0,8)
4

>>> L
[4, 1, 3, 2, 5, 9, 6, 7, 5]
```

```
>>> L = [3,2,1,4,7,6,5]

>>> f_tri_rapide_loc(L,4,6)
6

>>> L
[3, 2, 1, 4, 5, 6, 7]
```

Erreur possible : pour l'avoir vu plusieurs fois, il est possible que vous n'ayez pas incrémenté q (q+=1) au bon endroit !

Question 6: Mettre en place une fonction $f\_tri\_rapide\_en\_place\_rec(L,i,j)$ qui trie la liste L entre les indices i et j inclus par la méthode du cours sur le tri rapide en place par récursivité

Remarque : pas besoin de mettre de condition au début de cette fonction, si dans la question suivante, vous ne l'appelez que quand il le faut.

Question 7: Mettre en place une fonction $f\_tri\_rapide\_en\_place(L)$ qui trie la liste L par la méthode du cours sur le tri rapide en place par récursivité

Remarque : si votre code ne s'arrête pas, vous avez peut-être réalisé le test $i == j$ ou $i - j == 0$. Remarquez que si la portion de liste traitée ne possède que 2 termes (j=i+1), l'indice pivot sera soit i, soit j, et donc obligatoirement, en appelant [i,p-1] et [p+1,j], selon le cas, p-1<i ou p+1>j. Adaptez donc votre test de cas de base pour prendre en charge ce cas, ou bien limitez les indices à max(p-1,i) et min(p+1,j)…

Discussion sur la stabilité

Essayez :

```
>>> L = [1,2,3,1.0]

>>> f_tri_rapide_en_place(L)

>>> L
[1, 1.0, 2, 3]
```

Pour aller vers la stabilité de ce tri, et obtenir ce résultat, il faut déjà que la gestion des ex aequo avec le pivot soit bien faite dans la fonction f_tri_rapide_loc.

Essayez ensuite :

```
>>> L = [2,7,7.0,1]

>>> f_tri_rapide_en_place(L)

>>> L
[1, 2, 7.0, 7]
```

Vous remarquerez que les 7 sont inversés. En effet, comme vu en cours, le 1 est inversé avec le premier 7, le tri n'est donc pas stable.

On pourrait obtenir un tri stable en ne procédant pas à des échanges dans f_tri_rapide_loc, mais plutôt à la « descente » ou suppression/insertion de chaque terme à descendre à sa bonne place. Pour ne pas perdre de temps à créer une fonction « Descente », on pourra utiliser les fonctions :

- L.pop(i) qui retire et renvoie l'élément d'incide i de de L
- L.insert(i,t) qui insert t dans L au séparateur i

Mais attention, ces fonction ont le même coût en $O(n)$.

Question 8: Proposer une nouvelle fonction f_tri _loc_stable permettant la stabilité, ainsi que les fonction f_tri_rec_stable et f_tri_stable afin de rendre le tri en place stable

Remarque : on ne peut plus l'appeler « tri rapide ».

Question 9: Préciser la complexité dans le meilleur et dans le pire des cas

Voilà pourquoi le tri rapide en place reste sur le principe des échanges, et il n'est pas stable !

Exercice 2: Le tri rapide non récursif

Il est possible de réaliser le tri rapide en place de manière non récursive en utilisant une pile qui mémorise le travail à réaliser. Ainsi, en faisant appel à la fonction f_tri_rapide_loc, on empile les portions de liste à étudier, la première étant [0,len(L)-1], jusqu'à ce qu'il n'y en ait plus.

Question 1: Créer la fonction f_tri_rapide_non_rec(L) qui réalise ce tri en place

Exercice 3: Détermination de la médiane

La médiane d'une liste est la valeur telle qu'il y a autant d'éléments supérieurs et inférieurs à elle. On accepte un battement d'une position lorsque la liste contient un nombre impair de termes. Vous choisirez de prendre le terme à gauche, ou à droite, dans ce cas.

La première méthode pour trouver la médiane consiste à trier la liste, puis à prendre l'élément central.
Question 1: Mettre en place une fonction $f\_mediane\_normale(L)$ qui détermine la médiane d'une liste L par réalisation du tri rapide en place sur L puis par détermination de son élément central.

Il existe une méthode qui permet de ne trier que partiellement L et d'obtenir la médiane. Elle consiste à n'utiliser l'algorithme de tri rapide en place que partiellement.

Question 2: Après avoir réfléchi et trouver comment programmer cette méthode, proposer une fonction $f\_mediane\_rapide(L)$ permettant de le réaliser. On pourra définir une fonction intermédiaire si besoin !

Vous vérifierez évidemment que cela fonctionne :

```
>>> L = [4,1,7,2,6,3,5]

>>> f_mediane_normale(L)
4

>>> L
[1, 2, 3, 4, 5, 6, 7]
```

```
>>> L = [4,1,7,2,6,3,5]

>>> f_mediane_rapide(L)
4

>>> L
[3, 1, 2, 4, 6, 7, 5]
```