Informatique

8

Tris

TD 8-7

Scm et Alpha-tri

Exercice 1: Scm et α-tri

Cet algorithme de tri est d'autant plus efficace que la liste initiale est partiellement triée. Il s'agit d'une version simplifiée du tri TimSort utilisé par Python que l'on nommera α-tri.

Tri par fusion de scm

Soit s une liste d'entiers de longueur $n \geq 1$. Son partitionnement en séquences croissantes maximales d'éléments consécutifs (scm) est l'unique séquence de longueur $k \geq 1$ de couples d'entiers $(d_0, f_0), (d_1, f_1), \dots, (d_{k-1}, f_{k-1})$ telle que :

- $d_0 = 0$
- $f_{k-1} = n - 1$
- $\forall i \in \{0, \dots, k-1\}, d_i \leq f_i$
- $\forall i \in \{0, \dots, k-2\}, d_{i+1} = f_i + 1$
- $\forall i \in \{0, \dots, k-1\}, s(d_i) \leq s(d_i + 1) \leq \cdots \leq s(f_i)$
- $\forall i \in \{0, \dots, k-2\}, s(f_i) > s(d_{i+1})$

Exemple :

$$s = [3,4,8,11,1,5,2,7,9,0,10,0]$$

On obtient la décomposition scm suivante avec $k = 4$:

$$\underset{(d_0,f_0)=(0,3)}{3 \leq 4 \leq 8 \leq 11} > \underset{(d_1,f_1)=(4,5)}{1 \leq 5} > \underset{(d_2,f_2)=(6,8)}{2 \leq 7 \leq 9} > \underset{(d_3,f_3)=(9,10)}{0 \leq 10} > \underset{(d_4,f_4)=(11,11)}{0}$$

Question 1: Créer une fonction scm(s) qui renvoie la liste ordonnée de couples d'indices de la liste L.

Remarque : vous risquez fort d'oublier la dernière 😊

Vérifier que :

```
>>> scm([3,4,8,11,1,5,2,7,9,0,10,0])
[[0, 3], [4, 5], [6, 8], [9, 10], [11, 11]]
```

Juste pour être sûrs que vous ne faites pas comme Ozgur (PSI 17/18), essayez :

```
>>> scm([0, 0, 1, 2, 3, 4, 5, 7, 8, 9, 10, 11])
[[0, 11]]
```

On souhaite maintenant créer une fonction qui va fusionner deux scm consécutives.

Question 2: Créer une fonction tri_local(s,scm1,scm2) qui prend en paramètre une liste s et deux scm consécutives encodées par leurs couples d'indices de début et de fin et les fusionne DANS S de manière ordonnée

Remarques :

- Cette fonction ne renvoie pas de nouvelle liste mais modifie L
- On pourra utiliser la fonction f_fusion_ordonnee du tri fusion avec listes auxiliaires puis modifier localement S
- Mieux encore, on pourra directement utiliser la fonction fusion_ordonnee_ep 😊

Vérifier que :

```
>>> s = [3,4,8,11,1,5,2,7,9,0,10,0]

>>> scm(s)
[[0, 3], [4, 5], [6, 8], [9, 10], [11, 11]]

>>> tri_local(s,[4,5],[6,8])

>>> s
[3, 4, 8, 11, 1, 2, 5, 7, 9, 0, 10, 0]
```

Soit l'algorithme suivant appliqué à une liste s :

- On crée la liste Liste_scm des scm de s
- On fusionne les deux dernières scm de s et on met à jour la liste des scm
- On recommence jusqu'à ce que Liste_scm n'ait qu'une scm

Question 3: Créer une fonction Fusion_2_dernieres_scm(s,Liste_scm) qui fusionne les deux dernières scm de Liste_scm en place et met à jour la liste Liste_scm

Remarque : on supposera qu'il y a au moins 2 scm dans la liste Liste_scm

Vérifier que :

```
>>> s = [3,4,8,11,1,5,2,7,9,0,10,0]

>>> SCM = scm(s)

>>> SCM
[[0, 3], [4, 5], [6, 8], [9, 10], [11, 11]]

>>> Fusion_2_dernieres_scm(s,SCM)

>>> s
[3, 4, 8, 11, 1, 5, 2, 7, 9, 0, 0, 10]

>>> SCM
[[0, 3], [4, 5], [6, 8], [9, 11]]
```

Question 4: Créer une fonction tri_scm(s) qui trie s selon l'algorithme proposé

Vérifier que :

```
>>> s = [3,4,8,11,1,5,2,7,9,0,10,0]

>>> tri_scm(s)

>>> s
[0, 0, 1, 2, 3, 4, 5, 7, 8, 9, 10, 11]
```

Algorithme α-tri

On appelle toujours s la liste à trier.
Pour toute scm de la forme $x = [d, f]$, on note $|x|$ sa longueur :

$$|x| = f - d + 1$$

Nous allons manipuler une pile notée P, pour laquelle vous n'utiliserez que P.pop(), p.append() et len(P).

Dans l'algorithme α-tri, les fusions des scm sont réalisées comme suit :

- Au départ, la pile P est vide
- On ajoute à P les scm une par une, dans l'ordre (de gauche à droite), et :
    - A chaque ajout d'une scm dans P, on compare la longueur de la scm ajoutée $|z|$ à celle ajoutée à l'itération précédente $|y|$, si elle existe. Si $|y| < 2|z|$, on retire y et z, on les fusionne dans la liste s et on ajoute la scm fusionnée à P. On réitère cette opération (on réduit P) jusqu'à ce que, soit :
        - P ne possède plus qu'un élément
        - $|y| \geq 2|z|$
- Lorsque toutes les scm initiales ont été ajoutées à P, on fusionne itérativement les deux dernières scm jusqu'à ce que P ne contienne plus qu'un élément.

Question 5: Créer une fonction yz(P) qui renvoie les valeurs de y et z sans que P ne soit modifiée après exécution en utilisant les fonctions permises sur P uniquement

Vérifier que :

```
>>> P=[[1,5],[6,7]]

>>> yz(P)
(5, 2)

>>> P
[[1, 5], [6, 7]]
```

Question 6: Créer une fonction Reduction(s,P) qui réalise la réduction de P comme précisé ci-dessus

Vérifier que :

```
>>> s = [3,4,8,11,1,5,2,7,9,0,10,0]

>>> P=[[0,3],[4,5]]

>>> Reduction(s,P)

>>> s
[3, 4, 8, 11, 1, 5, 2, 7, 9, 0, 10, 0]

>>> P
[[0, 3], [4, 5]]

>>> P=[[0,3],[4,5],[6,8]]

>>> Reduction(s,P)

>>> s
[1, 2, 3, 4, 5, 7, 8, 9, 11, 0, 10, 0]

>>> P
[[0, 8]]
```

Question 7: Créer une fonction alpha_tri(s) qui trie s en place avec l'algorithme α-tri

Vérifier que :

```
>>> s = [3,4,8,11,1,5,2,7,9,0,10,0]

>>> alpha_tri(s)

>>> s
[0, 0, 1, 2, 3, 4, 5, 7, 8, 9, 10, 11]
```

Vérifiez les étapes successives suivantes, car la liste peut être triée par Tri_scm sans que les étapes intermédiaires ne soient correctes. Voici le détail des étapes afin de vous aider à vérifier votre code :

```
Liste à trier       :  [3, 4, 8, 11, 1, 5, 2, 7, 9, 0, 10, 0]
Etat avant réduction:  [[0, 3]] [3, 4, 8, 11, 1, 5, 2, 7, 9, 0, 10, 0]
Etat après réduction:  [[0, 3]] [3, 4, 8, 11, 1, 5, 2, 7, 9, 0, 10, 0]
Etat avant réduction:  [[0, 3], [4, 5]] [3, 4, 8, 11, 1, 5, 2, 7, 9, 0, 10, 0]
Etat après réduction:  [[0, 3], [4, 5]] [3, 4, 8, 11, 1, 5, 2, 7, 9, 0, 10, 0]
Etat avant réduction:  [[0, 3], [4, 5], [6, 8]] [3, 4, 8, 11, 1, 5, 2, 7, 9, 0, 10, 0]
Fusion              :  [[0, 3], [4, 5], [6, 8]] [3, 4, 8, 11, 1, 5, 2, 7, 9, 0, 10, 0]
Résultat            :  [[0, 3], [4, 8]] [3, 4, 8, 11, 1, 2, 5, 7, 9, 0, 10, 0]
Fusion              :  [[0, 3], [4, 8]] [3, 4, 8, 11, 1, 2, 5, 7, 9, 0, 10, 0]
Résultat            :  [[0, 8]] [1, 2, 3, 4, 5, 7, 8, 9, 11, 0, 10, 0]
Etat après réduction:  [[0, 8]] [1, 2, 3, 4, 5, 7, 8, 9, 11, 0, 10, 0]
Etat avant réduction:  [[0, 8], [9, 10]] [1, 2, 3, 4, 5, 7, 8, 9, 11, 0, 10, 0]
Etat après réduction:  [[0, 8], [9, 10]] [1, 2, 3, 4, 5, 7, 8, 9, 11, 0, 10, 0]
Etat avant réduction:  [[0, 8], [9, 10], [11, 11]] [1, 2, 3, 4, 5, 7, 8, 9, 11, 0, 10, 0]
Etat après réduction:  [[0, 8], [9, 10], [11, 11]] [1, 2, 3, 4, 5, 7, 8, 9, 11, 0, 10, 0]
Tri scm             :  [1, 2, 3, 4, 5, 7, 8, 9, 11, 0, 10, 0]
Liste triée         :  [0, 0, 1, 2, 3, 4, 5, 7, 8, 9, 10, 11]
```

Question 8: Comparer finalement sur une même courbe la complexité des tris scm et α-tri aux tris réalisés dans les exercices précédents