Informatique

8

Tris

TD 8-6

Tri radix

Exercice 1: Tri radix

Quelques outils

Le tri radix (ou tri par base) est un tri lexicographique qui permet de trier des éléments en comparant, position après position, tous les sous éléments de ceux-ci. Par exemple, pour trier des mots selon l'ordre alphabétique afin de créer un dictionnaire…

Soit un entier n dans l'intervalle $[0, 10^{m-1} - 1]$. Il s'écrit donc forcément à l'aide de m chiffres entiers de 0 à 9 en base 10. Par exemple, si $m = 4$, n dans $[0,9999]$. Alors, $832 = 0832$ et $1 = 0001$.

Question 1: Créer une fonction Convertir_Liste(n,m) qui transforme un entier n en une liste de ses chiffres

Vérifier :

```
>>> Convertir_Liste(832,3)
[8, 3, 2]

>>> Convertir_Liste(832,5)
[0, 0, 8, 3, 2]
```

Question 2: Créer une fonction Convertir_Entier(L) réalisant l'opération inverse

Vérifier :

```
>>> Convertir_Entier([0,0,8,3,2])
832
```

Question 3: Créer une fonction Calcul_m(L) qui détermine et renvoie m tel que tous les entiers dans L soient dans $[0, 10^m - 1]$

Remarques :

- Interdiction d'utiliser la fonction max de Python
- Essayez log(1000,10) après import de log dans le module math… Vous réaliserez une fonction qui n'utilise pas cette fonction afin d'éviter les problèmes d'erreurs d'arrondis

Vérifier :

```
>>> Calcul_m([999])
3

>>> Calcul_m([1000])
4

>>> Calcul_m([1001])
4
```

Mise en place du tri radix

Le principe du tri radix appliqué aux entiers est le suivant :

- Convertir les entiers en listes de leurs chiffres
- Trier avec un tri « stable » les différents entiers, dans l'ordre :
  - Selon leur chiffre des unités 10^0
  - Selon leur chiffre des dizaines 10^1
  - …
- Reconvertir les listes de chiffres en entiers (triés)

Un tri « stable » garde l'ordonnancement des termes considérés comme égaux. Si le tri utilisé n'était pas stable, en triant les dizaines, il ne permettrait pas de garder les unités dans l'ordre du tri précédent, etc.

On propose de réaliser un tri stable selon la technique du dénombrement vue plus haut avec des entiers. Nous allons l'adapter au cas traité où l'on doit trier des listes selon l'un de leurs éléments.

Soit L une liste de listes l de mêmes dimensions, du type : $L = [[0,8,3,2],[0,0,1,0],[1,0,0,0]]$.
On définit un alphabet (variable globale) $A = [0,1,2,3,4,5,6,7,8,9]$ qui définit l'ordre dans lequel les éléments à trier doivent être considérés.
On souhaite mettre en place une fonction Etape(L,i,A) capable de renvoyer une liste Lt des sous listes l de L triées selon leur élément d'indice i ($e = l[i]$). La méthode est la suivante :

- Créer une liste D (dénombrement) de listes vides de même dimension que A - Attention à bien créer une liste de listes vides différentes (si vous faites par exemple $[[]] * len(A)$, vous verrez apparaître des problèmes, car c'est la même liste répétée, et si on en change une de ces listes, on les change toutes ☹)
- Pour chaque sous liste l de L :
  - Déterminer l'indice ind de $e = l[i]$ dans A (on pourra utiliser A.index(Elem) pour s'adapter à tous types d'éléments dans A, pas uniquement des entiers de 0 à 9)
  - Ajouter l à la sous liste $D[ind]$
- Reconstruire la liste Lt triée en prenant dans l'ordre, les éléments de D

Question 4: Créer une fonction Etape(L,i,A) qui trie les sous listes de L selon leurs éléments L[i] d'indice i en utilisant le tri par dénombrement avec A une liste Alphabet qui définit l'ordre dans lequel les éléments doivent être triés et créée en variable globale

Vérifiez :

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

>>> A = [0,1,2,3,4,5,6,7,8,9]

>>> Etape(L,0,A)
[[1, 9, 2], [4, 5, 6], [9, 2, 1]]

>>> Etape(L,1,A)
[[9, 2, 1], [4, 5, 6], [1, 9, 2]]

>>> Etape(L,2,A)
[[9, 2, 1], [1, 9, 2], [4, 5, 6]]
```

```
>>> A.reverse()

>>> A
[9, 8, 7, 6, 5, 4, 3, 2, 1, 0]

>>> Etape(L,0,A)
[[9, 2, 1], [4, 5, 6], [1, 9, 2]]

>>> Etape(L,1,A)
[[1, 9, 2], [4, 5, 6], [9, 2, 1]]

>>> Etape(L,2,A)
[[4, 5, 6], [1, 9, 2], [9, 2, 1]]
```

Normalement, votre code devrait aussi être capable de trier des mots, sous forme de listes de lettres c'est sûr, mais probablement aussi sous forme de chaines de caractères :

```
>>> A = ['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z']

>>> L = [['d','e','n','i','s'],['c','o','u','c','o','u'],['i','p','t']]

>>> Etape(L,0,A)
[['c', 'o', 'u', 'c', 'o', 'u'], ['d', 'e', 'n', 'i', 's'], ['i', 'p', 't']]

>>> Etape(L,1,A)
[['d', 'e', 'n', 'i', 's'], ['c', 'o', 'u', 'c', 'o', 'u'], ['i', 'p', 't']]

>>> Etape(L,2,A)
[['d', 'e', 'n', 'i', 's'], ['i', 'p', 't'], ['c', 'o', 'u', 'c', 'o', 'u']]
```

```
>>> A = 'abcdefghijklmnopqrstuvwxyz'

>>> L = ['denis','coucou','ipt']

>>> Etape(L,0,A)
['coucou', 'denis', 'ipt']

>>> Etape(L,1,A)
['denis', 'coucou', 'ipt']

>>> Etape(L,2,A)
['denis', 'ipt', 'coucou']
```

Il faut juste veiller à ne pas dépasser l'indice max du mot le plus court.

Question 5: Déterminer la complexité en temps de la fonction Etape
Question 6: Créer une fonction tri_radix(L) qui réalise ce tri
Question 7: Déterminer la complexité de ce tri

Je n'irai pas plus loin, mais on pourrait adapter la fonction tri_radix à des mots de même taille m, puis un tri qui s'adapterai à des mots de tailles différentes pour les trier par ordre alphabétique par exemple.