Parcours imbriqué [tb07] – Exercice résolu

Karine Zampieri, Stéphane Rivière

Unisciel algoprog Université Haute-Alsace

Version 19 mai 2018

Table des matières

Python - Parcours imbriqué (Solution)

Mots-Clés Tableau unidimensionnel ■
Utilise Définitions et notations, Tableaux et paramètres, Parcours de tableaux ■
Difficulté • ○ ○

1 Parcours imbriqué / pgimbrique

Objectif

Cet exercice détermine le nombre de successeurs supérieurs pour chaque case d'un tableau d'entiers.

Définitions Python

```
TMAX = ...
```

1.1 Procédure calculerNSuperieurs (nombre de supérieurs)

Ce problème calcule, pour chaque case d'un `ITableau`, le nombre de cases suivantes qui contiennent un élément strictement supérieur. Exemple :

```
[45, 54, 1, -56, 22, 134, 49, 12, 90, -27]
==> [ 4 2 5 6 3 0 1 1 0 0 ]
```

Écrivez une fonction `nsuperieursTab(t,k,n,valeur)` qui calcule et renvoie le nombre d'éléments de `t[k..n]` supérieur à `valeur` (entier), où `t` est un `ITableau`.

Déduisez une procédure `calculerNSuperieurs(t,n,trs)` qui, dans un `ITableau trs`, calcule le nombre de valeurs successeurs supérieures à `t[ix]` pour chaque case `ix` des `n` éléments d'un `ITableau t`.

Validez votre fonction et votre procédure avec la solution.

Solution Python @[pgimbrique.py]

```
def nsuperieursTab(t, k, n, valeur):
    """  Nombre de valeurs d'un ITableau supérieurs à une valeur

     :param t: un ITableau
     :param k: indice de début de t
     :param n: nombre de valeurs de t
     :param valeur: valeur
     :return: le nombre de valeurs de t[k..n] supérieurs à valeur
    """
    rs = 0
    for j in range(k, n):
        if t[j] > valeur:
            rs += 1
    return rs
```

```
def calculerNSuperieurs(t, n, trs):
    """  Nombre de successeurs supérieurs d'un ITableau

     :param t: un ITableau
     :param n: nombre de valeurs
     :param trs: un ITableau résultat
    """
    for j in range(0, n):
        trs[j] = nsuperieursTab(t, j + 1, n, t[j])
```

1.2 Procédure afficherTab (affichage d'un tableau)

Écrivez le **profil** d'une procédure `afficherTab(t,n)` qui affiche les `n` premières valeurs d'un `ITableau t`.

Affichage des valeurs
L'affichage de `t` est un parcours complet des `n` valeurs.
Il est donc piloté par une boucle `Pour`.

Écrivez le corps de la procédure. Affichez les valeurs à la queue-leu-leu séparés par un espace, le tout entre crochet. Exemple :

```
[45 54... -27]
```

Validez votre procédure avec la solution.

Solution Python @[UtilsTB.py]

```
def afficherTab(t, n):
    """  Affichage d'un ITableau

     :param t: un ITableau
     :param n: nombre de valeurs dans [0..TMAX]
    """
    print("[", end="")
    for j in range(0, n):
        print(t[j], " ", sep="", end="")
    print("]")
```

1.3 Programme de test

Téléchargez le fichier suivant et mettez-le dans votre dossier.
Python @[UtilsTB.py]

Copiez/collez ensuite les lignes suivantes :

Python Au début de votre programme :

```
import UtilsTB
```

Soit la fonction `saisirTab(t)` qui effectue la saisie contrôlée du nombre de valeurs (entier compris entre 1 et `TMAX`), saisit les valeurs entières dans un `ITableau t` puis renvoie l'entier du nombre de valeurs saisies.

Python @[saisirTab] (dans UtilsTB)

Écrivez un script qui calcule, pour chaque case d'un `ITableau` de `nelems` valeurs, le nombre de cases suivantes qui contiennent un élément strictement supérieur. Les résultats seront placés dans un autre `ITableau` puis ce dernier sera affiché.

Testez. Exemple d'exécution :

```
Nombre d'éléments dans [1..50]? 10
t[0]? 45
t[1]? 54
t[2]? 1
t[3]? -56
t[4]? 22
1t[5]? 134
t[6]? 49
t[7]? 12
t[8]? 90
t[9]? -27
[ 4 2 5 6 3 0 1 1 0 0 ]
```

Validez votre script avec la solution.

Solution Python @[pgimbrique.py]

```
def PGImbrique():
    tab = [0 for x in range(TMAX)]
    nelems = UtilsTB.saisirTab(tab)
    trs = [0 for x in range(TMAX)]
    calculerNSuperieurs(tab, nelems, trs)
    UtilsTB.afficherTab(tab, nelems)
```

1.4 Fonction ninferieursTab (nombre d'inférieurs)

En partant de la fonction et procédure écrites ci-dessus,

Écrivez une fonction `ninferieursTab(t,k,n,valeur)` qui calcule et renvoie le nombre de valeurs inférieures à une valeur `valeur` (entier) dans les `t[k..n]` valeurs d'un `ITableau t`.

Déduisez une procédure `calculerNInferieurs(t,n,trs)` qui calcule dans un `ITableau trs` le nombre de valeurs successeurs inférieures à `t[ix]` pour chaque case `ix` des `n` éléments d'un `ITableau t` d'entiers.

Validez votre fonction et votre procédure avec la solution.

Solution Python @[pgimbrique.py]

```
def ninferieursTab(t, k, n, valeur):
    """  Nombre de valeurs d'un ITableau inférieurs à une valeur

     :param t: un ITableau
     :param k: indice de début de t
     :param n: nombre de valeurs de t
     :param valeur: valeur
     :return: le nombre de valeurs de t[k..n] inférieurs à valeur
    """
    rs = 0
    for j in range(k, n):
        if t[j] < valeur:
            rs += 1
    return rs
```

```
def calculerNInferieurs(t, n, trs):
    """  Nombre de successeurs inférieurs d'un ITableau

     :param t: un ITableau
     :param n: nombre de valeurs de t
     :param trs: un ITableau résultat
    """
    for j in range(0, n):
        trs[j] = ninferieursTab(t, j + 1, n, t[j])
```

2 Références générales

Comprend ■