Algorithmique

Algorithmes opérants par boucles imbriquées

Anicet E. T. Ebou, ediman.ebou@inphb.ci

01

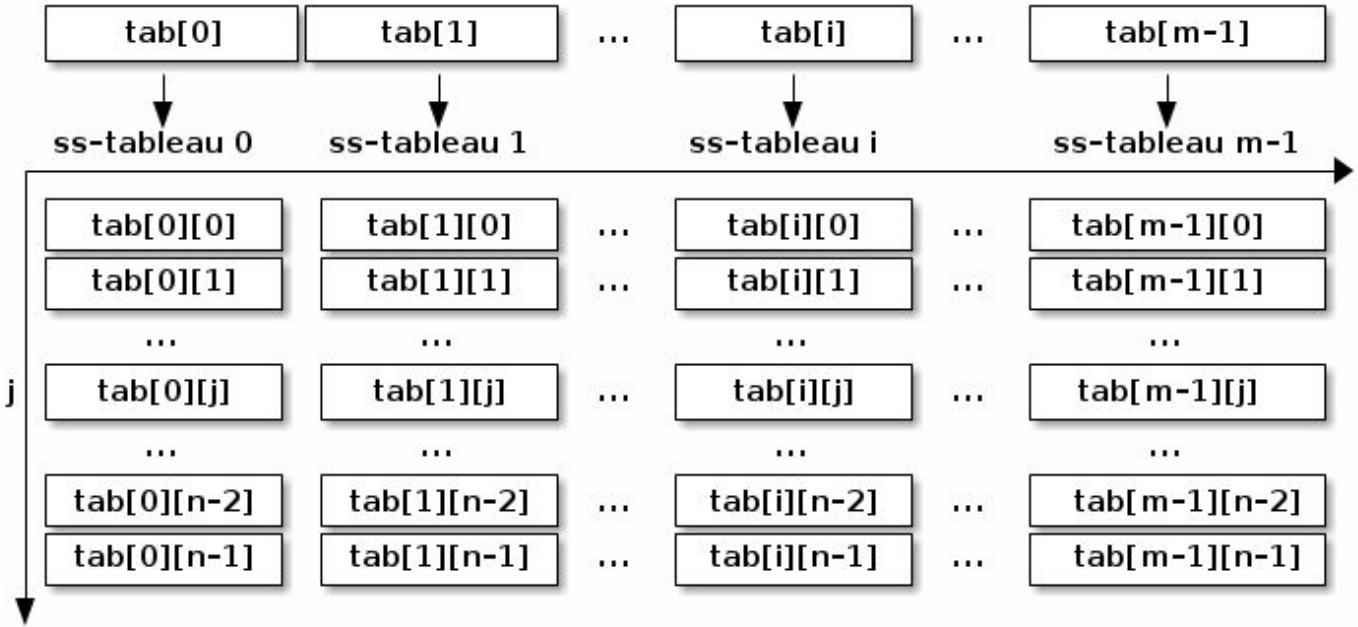
Tableaux à deux dimensions

Préambule motivant

Écrire un algorithme qui enregistre 10 notes entières et leurs coefficients et qui calcule la moyenne de ces 10 notes.

?

Format des tableaux à deux dimensions



Format des tableaux à deux dimensions

- Un tableau est une structure de données de dimension 2.
- `i` peut être considéré comme les abscisses, `j` comme les ordonnées.
- Une telle structure contient `m` × `n` éléments.
- Les autres noms des tableaux de dimension 2 sont : matrice, matrix…

Format des tableaux à deux dimensions

- Avec un tableau à deux dimensions, on peut ainsi modéliser les pixels d'un écran, mais surtout les matrices.
- Un tableau de longueur **`m`** contenant des tableaux de longueur **`n`** contenant à leur tour des tableaux de longueur **`p`** est une structure à trois dimensions contenant **`m×n×p`** éléments.

Syntaxe de déclaration en pseudo-code

```
Var:

    nom: TABLEAU[iMinL..iMaxL] DE TABLEAU [iMinC..iMaxC ] DE Type
```

- nom = nom de la matrice
- iMinL et iMaxL = indice min et max du vecteur « Père »
- iMinC et iMaxC = indice min et max du vecteur « Fils »
- Le nombre d'éléments N = (iMaxL-iMinL+1)*(iMalC-iMinC+1)

Syntaxe de déclaration en Python

```
# Matrice i x j

tab = [[0] * i] * j

# Matrice i x j avec la bibliothèque NumPy

import numpy as np

tab = np.zeros((j, i))
```

02

Manipulations élémentaires

Indexation d’une matrice

Entrer une valeur par indexation

```
# Ecriture pseudocode

tab[i][j] <- valeur

Lire(tab[i][j])

# Ecriture en Python

tab[i][j] = valeur

tab[i][j] = int(input())
```

Indexation d’une matrice

Extraction / lecture d’une valeur

```
# Extraction en pseudocode

valeur <- tab[i][j]

Ecrire(tab[i][j])

# Extraction en Python

valeur = tab[i][j]

print(tab[i][j])
```

Parcours d'une matrice

Le principe est de boucler sur les indices des lignes **m** et ensuite sur les indices des colonnes **n**.

```
Pour i ← 0 à m faire

    Pour j ← 0 à n faire

        Ecrire("Entrer l'élément ", i, j)

        Lire(t[i][j])

    FinPour

FinPour
```

Parcours d’une matrice

Le principe est de boucler sur les indices des lignes **m** et ensuite sur les indices des colonnes **n**.

```
for i in range(m):

    for j in range(n):

        print(“Entrez l’élément [“ + str(i) + “][” + str(j) + “]”)

        tab[i][j] = input()
```

Parcours d’une matrice

Le principe est de boucler sur les indices des lignes **m** et ensuite sur les indices des colonnes **n**.

```
for i in tab:

    for j in i:

        j = input()
```

Travaux Pratiques

Écrire un algorithme qui enregistre 10 notes entières et leurs coefficients et qui calcule la moyenne de ces 10 notes.

03

Opérations sur une matrice

Calcul de la somme d’une matrice: pseudo-code

somme

```
1  somme <- 0
2
3  Pour i <- 0 à m faire
4
4      Pour j <- 0 à n faire
5
5          somme <- somme + tab[i][j]
6
7  Ecrire(“La somme est: “, somme)
```

Calcul de la somme d’une matrice: Python

somme.py

```
1  somme = 0
2
3  for i in range(m):
4
5      for j in range(n):
6
7          somme = somme + tab[i][j] # ou encore: somme += tab[i][j]
8
9  print(“La somme est: “, somme)
```

Calcul de la somme d’une matrice: Python NumPy

```
import numpy as np

somme = np.sum(tab)

print(“La somme est: ”, somme)

# Somme suivant les axes

som1 = np.sum(tab, axis=1) # abscisse

som2 = np.sum(tab, axis=0) # ordonnée
```

Calcul du produit d’une matrice: pseudo-code

```
produit <- 0

Pour i <- 0 à m faire

    Pour j <- 0 à n faire

        produit <- produit * tab[i][j]

Ecrire(“Le produit est: “, produit)
```

Calcul du produit d’une matrice: Python

```
produit = 0

for i in range(m):

    for j in range(n):

        produit = produit * tab[i][j] # ou : produit *= tab[i][j]

print(“Le produit est: “, produit)
```

Calcul du produit d’une matrice: Python avec NumPy

```
import numpy as np

# Produit de la matrice

produit = np.prod(tab)

print(“Le produit est: ”, produit)

# Produit suivant les axes

prod1 = np.prod(tab, axis=1) # abscisse

prod2 = np.prod(tab, axis=0) # ordonnée
```

Calcul de la transposée d’une matrice: pseudo-code

```
Pour i <- 0 à n faire

    Pour j <- 0 à m faire

        tp_tab[i][j] <- tab[j][i]

Ecrire(“La transposée est: “, tp_tab)
```

Calcul de la transposée d’une matrice: Python

```
transpose = [[0] * n] * m

for i in range(n):

    for j in range(m):

        transpose[i][j] = tab[j][i]

print(“La transposée est: “, transpose)
```

Calcul de la transposée d’une matrice: Numpy

```
import numpy as np

transpose = np.transpose(tab)

print(“La transposée est: ”, transpose)
```