

Parcours de matrice

Table des matières

Introduction et rappels	1
Parcours simple	2
Principe	2
Algorithme	2
Complexité	3
Parcours en zigzag	3
Principe	3
Algorithmes	4
Version 1 : Test sur i	4
Version 2 : Variable de sens	4
Complexité	5
Parcours diagonal dans une matrice carrée	6
Définitions	6
Principe du parcours diagonal	6
Algorithme	7
Complexité	8
Récapitulatif des parcours	8

Objectifs

— Savoir parcourir une matrice de différentes façons

Introduction et rappels

Une matrice est un tableau à plusieurs dimensions. Dans ce chapitre, on se contentera de deux dimensions. On prend en règle générale l'indice i pour parcourir les lignes et l'indice j pour les colonnes.

Ainsi pour une matrice M , $M[i][j]$ sera l'élément situé à la ligne i et colonne j :

	1	2	...	j	...	n
1	...	...		...		...
2	...	...		...		...
$\vdots$				$\vdots$		
i	...	...	...	$M[i][j]$		...
$\vdots$						
n	...	...		...		...

La capacité à manipuler et parcourir des matrices est fondamentale en informatique, car elle intervient aussi bien dans le traitement d'images (sous forme de tableaux à deux dimensions) que dans le calcul matriciel utilisé en algèbre linéaire et en sciences des données.

Parcours simple

Principe

On va considérer qu'une matrice en deux dimensions est un **tableau de tableaux**. Ainsi, avec deux boucles imbriquées, une pour parcourir les lignes et une pour parcourir les colonnes, on accèdera à tous les éléments de la matrice.

Algorithme

Voici l'algorithme de parcours **ligne par ligne** d'une matrice M de taille $n \times m$:

```

ALGORITHME ParcoursMatrice(M) :
DONNEES
  M : matrice de taille n × m
VARIABLES
  i, j : entiers
DEBUT
  i ← 1
  TQ i ≤ n FAIRE
    j ← 1
    TQ j ≤ m FAIRE
      Affiche(M[i][j])
      j ← j + 1
    FTQ
    i ← i + 1
  FTQ
FIN

```

Exemple de parcours

Pour une matrice 3×4 :

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix}$$

Ordre de visite : 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12
(ligne 1 complète, puis ligne 2 complète, puis ligne 3 complète)

Complexité

La boucle intérieure : La variable j prend toutes les valeurs entre 1 et m , et le coût d'un passage est en $\Theta(1)$, donc sa complexité est de :

$$[_ \{j=1\}^{\{m\}} (1) = (m)]$$

La boucle principale : La variable i prend toutes les valeurs entre 1 et n , et le coût d'un passage est : $\Theta(m) + \Theta(1) = \Theta(m)$. Ainsi la complexité de l'algorithme sera :

$$[C(n,m) = _ \{i=1\}^{\{n\}} (m) = (n \times m)]$$

Question

Que faut-il changer dans cet algorithme pour parcourir la matrice selon les **colonnes** plutôt que les lignes ?

Réponse : Inverser les deux boucles (mettre la boucle sur j à l'extérieur et celle sur i à l'intérieur).

Parcours en zigzag

Principe

On parcourt une ligne dans un sens et la suivante dans le sens contraire, comme un serpent.

Exemple de parcours zigzag

Pour une matrice 8×8 :

```
→ 1   2   3   4   5   6   7   8
   17  16  15  14  13  12  11  10 ←
→ 19  20  21  22  23  24  25  26
```

```

    35  34  33  32  31  30  29  28 ←
→ 37  38  39  40  41  42  43  44
...

```

- **Ligne 1** (impaire) : parcours de gauche à droite ($1 \rightarrow 8$)
- **Ligne 2** (paire) : parcours de droite à gauche ($17 \rightarrow 10$)
- **Ligne 3** (impaire) : parcours de gauche à droite ($19 \rightarrow 26$)
- Et ainsi de suite...

Algorithmes

Version 1 : Test sur i

ALGORITHME ParcoursZigzag(M) :

DONNEES

M : matrice de taille $n \times m$

VARIABLES

i, j : entiers

DEBUT

i ← 1

TQ i ≤ n FAIRE

SI (i mod 2) = 1 ALORS

j ← 1

TQ j ≤ m FAIRE

Afficher(M[i][j])

j ← j + 1

FTQ

SINON

j ← m

TQ j ≥ 1 FAIRE

Afficher(M[i][j])

j ← j - 1

FTQ

FSI

i ← i + 1

FTQ

FIN

Version 2 : Variable de sens

ALGORITHME ParcoursZigzag2(M) :

```

DONNEES
  M : matrice de taille n × m
VARIABLES
  i, j, dir : entiers
DEBUT
  i ← 1
  dir ← 1
  TQ i ≤ n FAIRE
    j ← 1
    TQ 1 ≤ j ≤ m FAIRE
      Afficher(M[i][j])
      j ← j + dir
    FTQ
    i ← i + 1
    dir ← dir × (-1)
    j ← j + dir
  FTQ
FIN

```

Complexité

Dans les deux algorithmes, on parcourt chaque case **une seule fois**, la complexité est donc en $\Theta(n \times m)$.

💡 Exercice : Matrice triangulaire inférieure

À l'aide d'un parcours zigzag, écrire un algorithme qui renvoie **VRAI** si la matrice entrée en paramètre est **triangulaire inférieure** (c'est-à-dire que les cases **au-dessus** de sa diagonale principale ont toutes des valeurs nulles).

Exemple de matrice triangulaire inférieure :

$$\begin{pmatrix} a_{11} & 0 & 0 & 0 \\ a_{21} & a_{22} & 0 & 0 \\ a_{31} & a_{32} & a_{33} & 0 \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}$$

Tous les éléments au-dessus de la diagonale (où $j > i$) doivent être nuls.

Parcours diagonal dans une matrice carrée

Définitions

💡 Définition : Diagonale principale

La **diagonale principale** d'une matrice carrée de taille n est constituée des éléments $M[i][i]$ avec $i \in \{1, \dots, n\}$, soit ceux dont l'indice de ligne coïncide avec l'indice de colonne.

$$\begin{pmatrix} \boxed{a_{11}} & a_{12} & a_{13} & a_{14} \\ a_{21} & \boxed{a_{22}} & a_{23} & a_{24} \\ a_{31} & a_{32} & \boxed{a_{33}} & a_{34} \\ a_{41} & a_{42} & a_{43} & \boxed{a_{44}} \end{pmatrix}$$

i Question

Que vérifient les indices de la **diagonale secondaire** (antidiagonale) ?

Réponse : Pour une matrice $n \times n$, les éléments de l'antidiagonale vérifient : $i + j = n + 1$.

Exemple pour $n = 4$: - $M[1][4] : 1 + 4 = 5$ - $M[2][3] : 2 + 3 = 5$ - $M[3][2] : 3 + 2 = 5$
- $M[4][1] : 4 + 1 = 5$

Principe du parcours diagonal

On commence dans un coin, et on va suivre des diagonales parallèles à la diagonale principale pour parcourir toute la matrice.

i Exemple de parcours diagonal

Parcours en partant du coin **bas-gauche** vers le **haut-droite** :

Matrice 4×4 :

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Ordre de visite par diagonales parallèles :

- Diagonale 1 : 13
- Diagonale 2 : 9, 14
- Diagonale 3 : 5, 10, 15

- Diagonale 4 : 1, 6, 11, 16
- Diagonale 5 : 2, 7, 12
- Diagonale 6 : 3, 8
- Diagonale 7 : 4

! Propriété

Les indices i et j des cases qui appartiennent à des diagonales parallèles à la diagonale principale vérifient :

$[i-j=k, \text{ pour } k \in [1-n, n-1]]$

- Si $k < 0$: diagonale **au-dessus** de la diagonale principale
- Si $k = 0$: **diagonale principale**
- Si $k > 0$: diagonale **en-dessous** de la diagonale principale

Exemple pour $n = 4$: - $k = -3$: case (1, 4) - $k = -2$: cases (1, 3), (2, 4) - $k = -1$: cases (1, 2), (2, 3), (3, 4) - $k = 0$: cases (1, 1), (2, 2), (3, 3), (4, 4) (diagonale principale) - $k = 1$: cases (2, 1), (3, 2), (4, 3) - $k = 2$: cases (3, 1), (4, 2) - $k = 3$: case (4, 1)

💡 Question

Que vérifient les indices des cases qui appartiennent à une parallèle à l'**antidiagonale** ?

Réponse : $i + j = k$ pour $k \in [2, 2n]$.

Algorithme

ALGORITHME ParcourDiagonal(M) :

DONNEES

M : matrice de taille $n \times n$

VARIABLES

i, j, k : entiers

DEBUT

k ← 1-n

TQ k ≤ n-1 FAIRE

 i ← 1

 TQ i ≤ n FAIRE

 j ← i - k

 SI 1 ≤ j ≤ n ALORS

 Afficher(M[i][j])

 FSI

 i ← i + 1

 FTQ

```

    k ← k + 1
FTQ
FIN

```

Explication de l'algorithme

1. La variable k parcourt toutes les diagonales de $1 - n$ à $n - 1$
2. Pour chaque diagonale k , on parcourt toutes les lignes i de 1 à n
3. On calcule $j = i - k$ pour obtenir l'indice de colonne
4. On vérifie que j est valide (entre 1 et n) avant d'afficher

Exemple pour $n = 4$ et $k = 1$: - $i = 1$: $j = 1 - 1 = 0 \rightarrow$ non valide (ignoré) - $i = 2$: $j = 2 - 1 = 1 \rightarrow$ affiche $M[2][1]$ - $i = 3$: $j = 3 - 1 = 2 \rightarrow$ affiche $M[3][2]$ - $i = 4$: $j = 4 - 1 = 3 \rightarrow$ affiche $M[4][3]$

Complexité

Boucle intérieure : La variable i va prendre toutes les valeurs entre 1 et n , et le coût d'un passage est en $\Theta(1)$. Donc la complexité de la boucle intérieure est :

$$[\sum_{i=1}^n 1] = (n)$$

Boucle principale : La variable k prend toutes les valeurs entre $1 - n$ et $n - 1$ (soit $2n - 1$ valeurs). Le coût d'un passage est de $\Theta(n)$. Ainsi la complexité de l'algorithme est :

$$[C(n) = \sum_{k=1-n}^{n-1} \{n\} = ((2n-1) \times n) = (2n^2 - n) = \Theta(n^2)]$$

Remarque

Bien que cet algorithme visite chaque case exactement une fois (comme le parcours simple), sa complexité reste en $\Theta(n^2)$ car on effectue $2n - 1$ itérations de la boucle principale (une par diagonale) avec n itérations par boucle intérieure.

Récapitulatif des parcours

Type de parcours	Complexité	Caractéristique	Application
Simple (ligne par ligne)	$\Theta(n \times m)$	Parcours séquentiel classique	Traitement d'image ligne par ligne
Colonnes	$\Theta(n \times m)$	Parcours vertical	Opérations sur colonnes

Type de parcours	Complexité	Caractéristique	Application
Zigzag	$\Theta(n \times m)$	Alternance de sens	Compression d'image (JPEG)
Diagonal	$\Theta(n^2)$	Parcours par diagonales parallèles	Calculs sur matrices triangulaires

💡 Points clés

- Tous les parcours visitent chaque élément exactement une fois
- La complexité dépend du nombre total d'éléments ($n \times m$ ou n^2)
- Le choix du parcours dépend de l'application et de la structure des données
- Les parcours imbriqués utilisent toujours deux boucles pour les matrices 2D