– Algorithmique – TD – Série N°3 – LES TABLEAUX

Partie I : Tableaux Unidimensionnels (Tableaux à une dimension)

Soit ***TAB*** un tableau de ***N*** entiers.

Exercice 1 : Recherche

Q1) Ecrire un algorithme permettant de saisir (lire) les éléments du tableau ***TAB*** puis de les afficher.

*A partir de la question 2, on suppose que les éléments du tableau **TAB** ont été déjà saisis (lus).*

Q2) Ecrire un algorithme qui permet de vérifier si une valeur entière ***Val*** existe dans le tableau et dans ce cas d'afficher sa position (indice, rang).

Q3) Modifier l'algorithme précédent pour afficher toutes les positions de la valeur entière ***Val*** et de compter combien de fois elle existe (≡ son nombre d'occurrences).

Q4) Ecrire un algorithme qui permet de chercher le minimum dans le tableau.

Q5) (Optionnel) Modifier l'algorithme précédent pour qu'il cherche le maximum.

Exercice 2 : Tri par sélection (tri par extraction)

Le principe du tri par sélection est le suivant :

On commence par parcourir le tableau pour trouver le minimum. On le place à l'indice 1.

Ensuite, on recommence à parcourir le tableau à partir de l'indice 2 pour trouver le minimum que l'on place à l'indice 2. Et ainsi de suite pour l'indice 3, 4 jusqu'à N - 1. A la fin de l'algorithme, le tableau devient trié : le minimum se trouve au début du tableau (la position 1) et le maximum se trouve à la fin du tableau (la position N).

Question) Ecrire un algorithme permettant d'effectuer le tri par sélection.

Exercice 3 : Recherche dichotomique (Recherche Binaire)

La recherche dichotomique d'une valeur ***Val*** dans un tableau ***TAB*** déjà trié par ordre croissant, consiste à comparer la valeur ***Val*** avec l'élément ***TAB[m]*** du milieu du tableau.

- Si ***Val = TAB[m]***, on a trouvé la valeur ***Val*** et la recherche s'arrête.
- Si ***Val < TAB[m]***, la valeur ***Val*** ne peut se trouver que dans la moitié gauche du tableau ***TAB*** puisque celui-ci est trié par ordre croissant, on poursuit alors la recherche de la même manière uniquement dans la moitié gauche du tableau.

Si ***Val > TAB[m]***, ***Val*** ne peut se trouver que dans la moitié droite du tableau ***TAB***, on poursuit alors la recherche uniquement dans la moitié droite du tableau.

Ecrire un algorithme permettant la recherche d'une valeur ***Val*** à base de cette méthode.

Exercice 4 : Fusion de deux tableaux triés

Ecrire un algorithme qui permet de construire un tableau ***F*** trié par ordre croissant à partir des éléments de deux tableaux ***A*** et ***B*** triés par ordre croissant. Pour deux tableaux de tailles ***n*** et ***m*** respectivement, le tableau ***F*** sera de taille ***n+m***.

<u>Partie II</u> : Matrices (Tableaux à deux dimensions)

<u>Exercice 5</u> : Somme de deux matrices

La fonction qui calcule les éléments d'une matrice ***C***, somme de deux matrices ***A*** et ***B*** de même dimension ***(n, m)***, est donnée par la relation suivante : $c_{ij} = a_{ij} + b_{ij}$.

Ecrire un algorithme permettant de : ①- Lire les deux matrices ***A*** et ***B,*** ②- construire la matrice ***C*** à partir des deux matrices ***A*** et ***B,*** ③- Afficher la matrice C.

<u>Exercice 6</u> : Matrice transposée / Matrice Symétrique

Q1) Ecrire un algorithme qui donne la matrice transposée ***TR*** d'une matrice carrée ***Mat***. La matrice transposée ***TR*** est obtenue en échangeant les lignes et les colonnes de la matrice ***Mat***.

Q2) Modifier l'algorithme pour avoir la matrice transposée directement dans la matrice ***Mat***.

Q3) Ecrire un algorithme qui vérifie si une matrice carrée ***Mat*** est symétrique ou non. Une matrice est symétrique si chaque ligne ***i*** est égale à la colonne ***i***. En d'autres termes, chaque élément $\mathbf{mat_{ij} = mat_{ji}}$.

La matrice ci-dessous est symétrique :

15	3	7	2
3	11	4	37
7	4	23	18
2	37	18	12

Q4 - Optionelle) Ci-dessous des exemples de matrices particulières. Pour chaque cas, écrire un algorithme qui permet de vérifier la propriété indiquée.

0	*-3*	*-7*	*8*
3	***0***	*5*	*-37*
7	-5	***0***	*12*
-8	37	-12	***0***

Matrice Antisymétrique

15	***0***	***0***	***0***
3	11	***0***	***0***
7	4	23	***0***
2	37	0	12

Triangulaire inférieure

15	7	13	8
0	11	0	34
0	***0***	23	37
0	***0***	***0***	12

Triangulaire supérieure

15	***0***	***0***	***0***
0	**11**	***0***	***0***
0	***0***	**23**	***0***
0	***0***	***0***	**12**

Diagonale

<u>Exercice 7</u> : Triangle de PASCAL

Le triangle de Pascal est représenté par une matrice triangulaire ***P*** dont les éléments p_{ij} sont définis comme suit :

✓ $p_{ij} = 1$ ***si j=1*** (la première colonne est toujours à ***1***).

✓ $p_{ij} = 1$ ***si i =j*** (la diagonale est toujours à ***1***).

✓ $p_{ij} = p_{i-1\ j-1} + p_{i-1\ j}$ ***si j > 1 et j < i*** (l'élément de la ligne ***i*** et colonne ***j*** s'obtient en ajoutant les valeurs des éléments de la ligne ***i - 1*** et colonne ***j - 1*** et de la ligne ***i - 1*** et colonne ***j***).

✓ $p_{ij} = 0$ ***si j > i*** (partie supérieure de la matrice est nulle).

Voici un triangle de Pascal de dimension $N = 6$.

Ecrire un algorithme permettant de générer automatiquement le triangle de Pascal pour une taille **N** donnée.

P	1	2	3	4	5	N=6
1	1	0	0	0	0	0
2	1	1	0	0	0	0
3	1	2	1	0	0	0
4	1	3	3	1	0	0
5	1	4	6	4	1	0
N=6	1	5	10	10	5	1

<u>Partie III</u> : Exercices supplémentaires

<u>Exercice 1</u> : Soit ***TAB*** un tableau de ***N*** nombres entiers.

Q1) Ecrire un algorithme permettant de compter le nombre des éléments nuls, pairs (non nuls) ou impairs.

Q2) Ecrire un algorithme qui vérifie si les éléments du tableau sont triés dans l'ordre croissant.

Exemple : Les valeurs suivantes sont triées : 7 13 24 84 125 127

Q3) Ecrire un algorithme qui vérifie si les éléments du tableau sont tous consécutifs.

Exemple : Les valeurs suivantes sont consécutives : 3 4 5 6 7 8 9

Q4) Ecrire un algorithme qui permet de décaler les éléments du tableau circulairement vers la droite :
Le premier élément est déplacé à la position 2, le deuxième élément est déplacé à la position 3 et ainsi de suite.
L'élément qui se trouve à la fin du tableau est déplacé au début du tableau (à la position 1).

Q5) Modifier l'algorithme précédent pour faire un décalage circulaire vers la gauche.

<u>Exercice 2</u> : Tri à bulles (tri par propagation)

Soit ***TAB*** un tableau de ***N*** nombres entiers à trier.

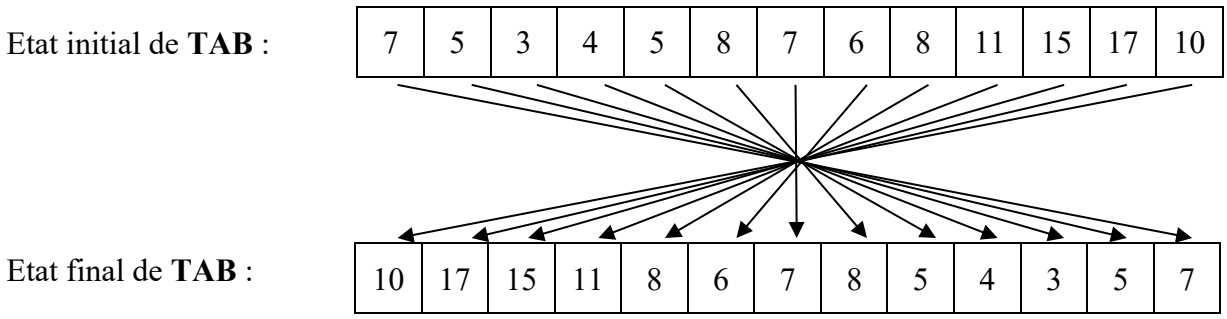
Le principe du **tri à bulles** consiste à parcourir le tableau de la fin jusqu'au début, et comparer les couples d'éléments successifs. Lorsque deux éléments successifs ne sont pas dans l'ordre croissant, ils sont échangés.
Après chaque parcours complet du tableau, on recommence l'opération.
Lorsque aucun échange n'a lieu pendant un parcours, cela signifie que le tableau est trié. On s'arrête alors.

Question) Ecrire un algorithme permettant d'effectuer le tri du tableau par cette méthode.

<u>Exercice 3</u> : Inversion / Tableau Symétrique

Q1) Ecrire un algorithme qui permet d'inverser les éléments d'un tableau **TAB**. C'est-à-dire, le premier élément devient dernier, et le dernier devient premier. Le deuxième devient l'avant dernier et l'avant dernier devient deuxième, et ainsi de suite. Voir le schéma ci-dessous.

<u>Exemple</u> : **Si N = 13**



<u>Remarque</u> : Il est interdit d'utiliser un deuxième tableau. Tous les traitements doivent être effectués sur le tableau **TAB**.

Q2) Ecrire un algorithme qui vérifie si un tableau est **<u>symétrique</u>** : Le premier élément est égal au dernier, le deuxième est égal à l'avant dernier et ainsi de suite. En d'autres termes si on inverse le tableau, on obtient le tableau initial.

<u>Exercice 4</u> : Quelques opérations élémentaires sur les matrices

Soit **MAT** une matrice d'entiers de taille ***(n, m)***. Ecrire un algorithme qui permet de :

Q1) Rechercher une valeur **Val**.

Q2) Rechercher le minimum (et/ou le maximum).

Q3) Calculer la somme des éléments d'une ligne I (puis déduire la moyenne de la ligne).

Q4) Calculer la somme des éléments d'une colonne J (puis déduire la moyenne de la colonne).

Q5) Calculer la somme des éléments de la diagonale principale. Cette somme est appelée **trace** de la matrice.

Q6) Calculer la somme des éléments de la diagonale secondaire (deuxième diagonale).

Q7) Permuter les éléments de la ligne I1 avec les éléments de la ligne I2.

Q8) Permuter les éléments de la colonne J1 avec les éléments de la colonne J2.

<u>Exercice 4</u> : Matrices Particulières

Soit ***M*** une matrice carrée. Ecrire un algorithme pour vérifier les propriétés suivantes :

Q1) ***M*** est **antisymétrique** si chaque élément $\mathbf{m_{ij} = - m_{ji}}$. Les éléments de la diagonale principale sont obligatoirement nuls.

Q2) ***M*** est **triangulaire inférieure** si tous les coefficients au-dessus de la diagonale principale sont nuls. Les coefficients de la diagonale et sous la diagonale peuvent être ou ne pas être nuls (Voir Exemples).

Q3) ***M*** est **triangulaire supérieure** si tous les coefficients en dessous de la diagonale principale sont nuls. Les coefficients de la diagonale et au-dessus de la diagonale peuvent être ou ne pas être nuls.

Q4) ***M*** est **diagonale** si tous les coefficients en dehors de la diagonale principale sont nuls. Les coefficients de la diagonale peuvent être ou ne pas être nuls.

Ecrire un algorithme qui vérifie si une matrice carrée ***M*** est diagonale ou non.

Q5) Une matrice est **scalaire** si elle est diagonale et tous les coefficients diagonaux sont égaux.

Q6) Une matrice **identité** est une matrice scalaire où chaque coefficient diagonal est égal à un (1).

Ci-dessous des exemples de matrices particulières :

0	*-3*	*-7*	*8*
3	***0***	*5*	*-37*
7	-5	***0***	*12*
-8	37	-12	***0***

Matrice Antisymétrique

15	***0***	***0***	***0***
3	11	***0***	***0***
7	4	23	***0***
2	37	0	12

Triangulaire inférieure

15	7	13	8
0	11	0	34
0	***0***	23	37
0	***0***	***0***	12

Triangulaire supérieure

15	***0***	***0***	***0***
0	**11**	***0***	***0***
0	***0***	**23**	***0***
0	***0***	***0***	**12**

Diagonale