Feuille de TD N^o 6 : calcul matriciel et programmation dynamique

On considère un tableau à deux dimensions disposant de n lignes et p colonnes. Nous allons définir ici une multiplication sur les tableaux appelée multiplication matricielle.

Si on considère $tab1$ et $tab2$ deux tableaux alors on peut définir leur multiplication si et seulement si le nombre de colonnes de $tab1$ est égal au nombre de lignes de $tab2$. La cas échéant, si $tab1$ a $n1$ lignes et $k1$ colonnes et $tab2$ a $n2 = k1$ lignes et $k2$ colonnes alors on définit $c = tab1 \times tab2$ comme le tableau à $n1$ lignes et $k2$ colonnes tel que :

$$\forall 0 \leq i \leq n1-1, \forall 0 \leq j \leq k2-1, c[i][j] = \sum_{k=0}^{k1-1} tab1[i][k] \times tab2[k][j]$$

1 Parmi les tableaux suivants, lesquels peuvent être multipliés entre eux ? L'ordre de multiplication a-t-il une importance ? Calculer $t1 \times t2$ et $t2 \times t1$.

```
int t1[2][3]={{1,2,1},{2,1,1}} ;
int t2[3][2]={{1,0},{1,1},{2,1}};
int t3[2][2]={{1,2},{3,4}};
int t4[3][3]={{1,0,0},{0,1,0},{1,1,1}};
int t5[1][3]={{1,2,2}};
```

2 Ecrire une fonction qui prend en entrée deux tableaux et renvoie leur multiplication. On utilisera une assertion pour vérifier si les dimensions des tableaux passés en arguments sont compatibles.

Ecrire la fonction de signature `int** mult(int n1, int k1, int n2, int k2, int tab1[][k1],int tab2[][k2])`

3 Justifier que la complexité de votre fonction est $O(n1k1k2)$.

Nous allons maintenant étudier le problème du coût de la multiplication d'un ensemble de n tableaux $T_1, \ldots, T_n$. Nous noterons n_i le nombre de lignes du tableau T_i et c_i son nombre de colonnes.

4 Voici un premier exemple, considérons les tableaux suivants :
`int T1[2][3]; int T2=[3][5]; int T3=[5][2]; int T4[2][8]; int T5[8][1];`

Quel est le coût total de chacun des calculs suivants :

1. `T1T2`
2. `T2T3`
3. `(T3T4)T5`
4. `(T1T2)((T3T4)T5)`
5. `(T1(T2T3))(T4T5)`

5 Quel est le parenthésage optimal qui donnera le moins d'opérations pour le produit `T1T2T3` ?

On pose $m_{i,j}$ le coût minimal pour calculer le produit $T_i \ldots T_j$.

6 Quelles valeurs de (i, j) allons nous considérer ? Que vaut $m_{i,i}$?

7 Supposons que pour calculer optimalement le produit $T_i \ldots T_j$, on doit faire le parenthésage suivant : $(T_i \ldots T_{k-1})(T_k \ldots T_j)$. Dans ce cas que vaut $m_{i,j}$?

8 En déduire une formule de récurrence sur les $m_{i,j}$.

On veut enfin mettre en place une stratégie de programmation dynamique qui résout notre problème.

9 L'entrée du problème sera contenue dans un tableau `matrixdim` de taille $2n$ où n est le nombre de tableaux à multiplier tel que $\forall 0 \leq i \leq n-1, matrixdim[2i] =$ nb de lignes de T_{i+1} et $matrixdim[2i+1] =$ nb de colonnes de T_{i+1}. Donner le tableau `matrixdim` associé au problème à cinq tableaux donné ci-dessus.

10 Soit un couple d'indices (i, j) tel que $i \leq j$. Pour calculer $m_{i,j}$ quels sont les valeurs de $m_{?,?}$ qu'il faut déjà avoir calculées ? En déduire un ordre de remplissage d'un tableau à double entrée qui contiendra les coefficients $m_{i,j}$.

11 On a appliqué l'algorithme pour trouver la meilleure façon d'effectuer le produit de tableaux $A_1 * A_2 * A_3 * A_4 * A_5 * A_6$, sachant que les tableaux sont de dimensions respectives $10 * 5$, $5 * 20$, $20 * 4$, $4 * 100$, $100 * 10$, $10 * 5$. On obtient le tableau `m` suivant :

	5	20	4	100	10	5
	A_1	$A_1 * A_2$	$A_1 * A_2 * A_3$	$A_1 * A_2 * A_3 * A_4$	$A_1 * A_2 * A_3 * A_4 * A_5$	$A_1 * A_2 * A_3 * A_4 * A_5 * A_6$
10	0	1000	$400 + 200$ $1000 + 800$ $=> 600$	$2400 + 5000$ $9000 + 20000$ $600 + 4000$ $=> 4600$	$4600 + 500$ $5800 + 2000$ $4600 + 400$ $4600 + 10000$ $=> 5000$	$4700 + 250$ $5600 + 1000$ $4800 + 200$ $9600 + 5000$ $5000 + 500$ $=> 4950$
		A_2	$A_2 * A_3$	$A_2 * A_3 * A_4$	$A_2 * A_3 * A_4 * A_5$	$A_2 * A_3 * A_4 * A_5 * A_6$
5		0	400	$8000 + 10000$ $400 + 2000$ $=> 2400$	$4800 + 1000$ $4400 + 200$ $2400 + 5000$ $=> 4600$	$4600 + 500$ $4600 + 100$ $7400 + 2500$ $4600 + 250$ $=> 4700$
			A_3	$A_3 * A_4$	$A_3 * A_4 * A_5$	$A_3 * A_4 * A_5 * A_6$
20			0	8000	$4000 + 800$ $8000 + 20000$ $=> 4800$	$4200 + 400$ $13000 + 10000$ $4800 + 1000$ $=> 4600$
				A_4	$A_4 * A_5$	$A_4 * A_5 * A_6$
4				0	4000	$5000 + 2000$ $4000 + 200$ $=> 4200$
100					A_5 0	$A_5 * A_6$ 5000
10						A_6 0

Quelle est la meilleure façon de calculer le produit des tableaux ?

12 Ecrire une fonction `void solve (int n, int tab_matrix_dim[] , int m[n][n])` qui prend en entrée un entier n correspondant au nombre de tableaux dans le produit, un tableau `tab_matrix_dim` qui contient les dimensions des tableaux du produit et un tableau à deux dimensions `m` dont on peut supposer que toutes les cases sont initialisées à zéro. La fonction mettra à jour le tableau `m` de telle sorte que la case (i, j) contienne $m_{i+1,j+1}$.

13 Quelle est la complexité temporelle de la fonction `solve` ?

14 Après application de la fonction `solve`, quelle case de `m` du tableau contient le résultat cherché ?