TD n° 8 : Diviser pour régner

Exercice 1

Donner des algorithmes (et les implémenter en Caml) utilisant une stratégie « diviser pour régner » pour

1. Déterminer le minimum d'un tableau.
2. Calculer la sommes des éléments d'une liste d'entiers.
3. Calculer une exponentiation rapide en décomposant l'exposant modulo 3.

Est-on gagnant en terme de complexité par rapport à ce qu'on sait faire par ailleurs ?

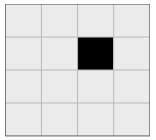
Exercice 2 *Pavage d'un échiquier*

On considère un échiquier carré de côté 2^k avec $k \geq 1$ que l'on souhaite paver (recouvrir) avec les quatre motifs suivants :

1. Pourquoi est-ce impossible ?

On suppose que l'on accepte de laisser une case vide, choisie arbitrairement, que l'on colorie en noir. On peut alors évidemment recouvrir un échiquier 2×2 à l'aide l'une des quatre pièces, en fonction de l'emplacement de la case noire à laisser libre.

2. Donner un exemple de recouvrement de l'échiquier suivant :



3. Montrer, en raisonnant par récurrence sur k et en utilisant une approche *diviser pour régner*, qu'un tel recouvrement — laissant une case libre arbitraire — est toujours possible.

Exercice 3

On s'intéresse à l'algorithme de Karatsuba de multiplication des polynômes. On représente les polynômes par des tableaux de leur cœfficients.

1. Donner la complexité de l'algorithme naïf permettant de calculer la somme de deux polynômes de degré n.
2. Donner la complexité d'un algorithme simple permettant de calculer le produit de deux polynômes de degré n.
3. On souhaite développer une stratégie « diviser pour régner ». On suppose, pour simplifier, que n s'écrit 2^p et on pose $m = \frac{n}{2} = 2^{p-1}$, $P = X^m P_1 + P_2$ et $Q = X^m Q_1 + Q_2$, le degré des autre polynômes P_1, P_2, Q_1 et Q_2 étant de degré au plus $m = 2^{p-1}$. Développer PQ. Quelle est la complexité de cet algorithme ? Commenter.
4. En remarquant que
$$P_1Q_2 + P_2Q_1 = (P_1 + P_2)(Q_1 + Q_2) - P_1Q_1 - P_2Q_2,$$
montrer qu'on peut trouver trois polynômes de degré au plus $\frac{n}{2}$ tels que $PQ = X^2 m R_1 + X^m(R_2 - R_1 - R_3) + R_3$.
Que devient la complexité ? Commenter.

La similitude entre la représentation binaire d'un entier et les polynômes permet d'adapter l'algorithme précédent aux grands entiers (et inversement), la seule différence se trouve dans la gestion de la retenue.

Dans les années 1950, Andreï Kolmogorov travaille sur la complexité des opérations arithmétiques et conjecture qu'une multiplication de deux nombres de n chiffres ne peut être réalisée en moins de $O(n^2)$ opérations. À l'époque, aucun algorithme plus rapide que la multiplication standard n'est connu. À l'automne 1960, Kolmogorov organise un séminaire dans lequel il parle de sa conjecture, auquel assiste Anatolii Alexevich Karatsuba. Une semaine plus tard, Karatsuba avait trouvé cet algorithme, qui prouvait que la conjecture était fausse.

L'algorithme Toom-Cook est un raffinement qui consiste à découper les polynômes en r blocs avec $r > 2$. La complexité peut alors passer en $O\left(n^{1+\varepsilon}\right)$ où ε est un réel strictement positif arbitraire. L'algorithme de Schönhage-Strassen, qui utilise la transformation de Fourier rapide, permet d'obtenir une complexité de $O(n \log n)$.

EXERCICE 4

Soit $n \in \mathbb{N}^*$. Intéressons-nous au produit matriciel.

1. (a) Si M_1 et M_2 sont deux matrices de tailles $n \times n$, combien d'additions scalaires nécessitent l'addition matricielle ?

   (b) Si M_1 et M_2 sont deux matrices de tailles $n \times n$, combien d'additions scalaires et de multiplications scalaires nécessitent le produit matriciel ?

2. On souhaite améliorer la complexité du produit en adoptant une méthode « diviser pour régner ». On suppose pour simplifier que $n = 2^p$ et on pose $m = \frac{n}{2} = 2^{p-1}$.

   Soit $(M, N) \in \mathcal{M}_n(\mathbb{K})^2$. On considère une disposition par blocs pour ces deux matrices.

   Il existe $(A_1, B_1, C_1, D_1, A_2, B_2, C_2, D_2) \in \mathcal{M}_m(\mathbb{K})^8$ telles que :

$$M_1 = \begin{pmatrix} A_1 & B_1 \\ C_1 & D_1 \end{pmatrix} \quad M_2 = \begin{pmatrix} A_2 & B_2 \\ C_2 & D_2 \end{pmatrix}$$

   (a) On effectue le produit par blocs :

$$M_1 M_2 = \begin{pmatrix} A_1A_2 + B_1C_2 & A_1B_2 + B_1D_2 \\ C_1A_2 + D_1C_2 & C_1B_2 + D_1D_2 \end{pmatrix}$$

   Combien cette méthode nécessite à priori de multiplications et d'additions sur des matrices $m \times m$?

   (b) On note c_n le coût pour effectuer la multiplication de deux matrices $n \times n$ en utilisant récursivement les formes blocs. On obtient une relation de la forme $c_n = ac_{\frac{n}{k}} + \Theta\left(n^d\right)$. Donner les valeurs de de a , k et d.

   (c) Déterminer le coût c_n. Commenter.

3. L'algorithme de Strassen utilise les formules suivantes :

$$\begin{array}{lll} S_1 = (B_1 - D_1)(C_2 + D_2) & S_5 = A_1(B_2 - D_2) & X = S_1 + S_2 - S_4 + S_6 \\ S_2 = (A_1 + D_1)(A_2 + D_2) & S_6 = D_1(C_2 - A_2) & Y = S_4 + S_5 \\ S_3 = (A_1 - C_1)(A_2 + B_2) & S_7 = (C_1 + D_1)A_2 & Z = S_6 + S_7 \\ S_4 = (A_1 + B_1)D_2 & & T = S_2 - S_3 + S_5 - S_7 \end{array}$$

   Et, dans ce cas, $MN = \begin{pmatrix} X & Y \\ Z & T \end{pmatrix}$.

   (a) On note d_n le coût pour effectuer la multiplication de deux matrices $n \times n$ en appliquant le principe précédent récursivement. On donnera autant d'importance aux additions qu'aux multiplications.

   On obtient une relation de la forme : $d_n = ad_{\frac{n}{k}} + \Theta\left(n^d\right)$. Donner les valeurs de de a , k et d.

   (b) Déterminer d_n. Commenter.

Le meilleur algorithme de calcul du produit de deux matrices de taille $n \times n$ a actuellement une complexité de l'ordre de $n^{2,79}$.