Algorithmique TD 2: Analyse d'algorithmes récursifs

Exercice 1. Master Théorème : quelques exemples

Pour chacun des problèmes suivants, décrire le principe d'un algorithme récursif pour le résoudre. Donner la formule de récurrence et calculer son coût en appliquant le Master Theorem.

a. Calcul du maximum d'un tableau d'entiers

b. Recherche d'un élément dans un tableau d'entiers triés

Exercice 2. Tri fusion

Le tri fusion est un algorithme de tri de type diviser pour régner. Il procède ainsi :

Division : partage le tableau en deux moitiés : $T = \begin{bmatrix} T_1 & T_2 \end{bmatrix}$.

Résolution : trie les deux moitiés T_1 et T_2 récursivement

Combinaison fusionne les deux moitiés triées

L'essentiel du travail algorithmique porte donc sur la phase de Combinaison.

a. Écrire l'algorithme `TriFusion`. Il fera appel à un algorithme `Fusion` qu'on détaillera plus tard. Attention, pour éviter de copier les tableaux, on passera en argument des indices de début et de fin permettant de délimiter la zone de travail d'un appel à l'algorithme.

b. Écrire l'algorihtme `Fusion`.

c. Analyser le coût de ces deux algorithmes (en supposant que n est une puissance de 2)

d. Montrer que l'on ne peut-on faire mieux, dans le modèle de coût "nombre de comparaisons". On pourra pour cela utiliser le fait que tout algorihtme de tri par comparaisons peut être représenté comme un arbre binaire dont chaque noeud interne représentent une comparaison entre deux éléments du tableau, et chaque feuille une permutation de $(1, 2, \ldots, n)$.

Exercice 3. Tri Rapide

Le tri rapide est un autre algorithme de tri de type diviser pour régner. Il procède ainsi sur un tableau T :

Division : On choisit l'élément $x = T[n-1]$ (appelé pivot) et retourne deux tableaux T_1 et T_2 formés des autres éléments $T[0], \ldots, T[n-2]$ tels que tous les éléments de T_1 sont inférieurs à x et tous les éléments de T_2 sont supérieurs à x.

Résolution : trie T_1 et T_2 récursivement.

Combinaison : combine le résultat pour en faire un tableau trié.

L'essentiel du travail algorithmique porte cette fois sur la phase de Division.

a. Écrire l'algorithme `Partitionne` qui effectue la phase de division. On cherchera à ce que l'algorithme travaille "en place", c'est à dire qu'il modifie le tableau d'entrée pour y représenter ses sorties. Il pourra être nécessaire de retourner un indice supplémentaire en sortie.

b. Écrire l'algorithm `TriRapide` faisant appel à l'algorithme `Partitionne`.

c. Analyser son coût en pire cas, en meilleur cas,

d. Cet algorithme peut être "randomisé" en choisissant aléatoirement la position du pivot (et en permutant cet élément avec $T[n-1]$). Analyser son coût en moyenne.

Exercice 4. Arithmétique des polynômes

On représente un polynôme $P = p_0 + p_1 X + \cdots + p_{n-1} X^{n-1}$ par le tableau des $n+1$ éléments $[p_0, p_1, \ldots, p_{n-1}]$.

a. Proposer un algorithme faisant l'addition de deux polynômes. Quelle est son coût ? Peut-on espérer l'améliorer en employant par exemple un algorithme diviser pour règner ?

b. Proposer un algorithme itératif faisant le produit de deux polynômes. Quelle est son coût ? Peut-on espérer l'améliorer ?

c. Proposer un algorithme *Diviser pour Règner* faisant la multiplication de deux polynômes (qu'on supposera de même degré).

Indication : on pourra utiliser l'identité :

$$(P_0 + XP_1)(Q_0 + XQ_1) = P_0Q_0 + X(P_0Q_1 + P_1Q_0) + X^2 P_1 Q_1$$

d. Analyser son coût (dans le cas ou le dégré des polynômes en entrée est égal à $2^k - 1$).

e. En 1960, Karatsuba proposa d'utiliser plutôt la formule suivante :

$$(P_0 + XP_1)(Q_0 + XQ_1) = P_0Q_0 + X((P_0 + P_1)(Q_0 + Q_1) - P_0Q_0 - P_1Q_1) + X^2 P_1 Q_1$$

Répéter les deux questions précédentes en vous basant sur cette identité. Conclure

On s'intéresse désormais à la décomposition en 3 des polynômes : $P = P_0 + XP_1 + X^2 P_2$ et $Q = Q_0 + XQ_1 + X^2 Q_2$. Toom a proposé une formule calculant $P \times Q$ à partir des cinq valeurs suivantes :

$$\begin{array}{rcl} M_0 & = & P_0 Q_0 \\ M_1 & = & (P_0 + P_1 + P_2)(Q_0 + Q_1 + Q_2) \\ M_2 & = & (P_0 - P_1 + P_2)(Q_0 - Q_1 + Q_2) \\ M_3 & = & (P_0 + 2P_1 + 4P_2)(Q_0 + 2Q_1 + 4Q_2) \\ M_4 & = & P_2 Q_2 \end{array}$$

Le produit $R = P \times Q = R_0 + R_1 X + R_2 X^2 + R_3 X^3 + R_4 X^4$ s'obtient alors ainsi :

$$\left\{ \begin{array}{rcl} R_0 & = & M_0 \\ R_1 & = & \frac{1}{6}(-3M_0 + 6M_1 - 2M_2 - M_3 + 12M_4) \\ R_2 & = & \frac{1}{2}(-2M_0 + M_1 + M_2 - 2M_4) \\ R_3 & = & \frac{1}{6}(3M_0 - 3M_1 - M_2 + M_3 - 12M_4) \\ R_4 & = & M_4 \end{array} \right. \tag{1}$$

f. Quelle est le coût de l'algorithme de multiplication de polynômes correspondant ?

g. Interpréter les formules calculant les M_i comment des évaluations. Préciser en quels points.

h. En déduire comment les coéfficients des formules (1) ont été trouvés.

Plus généralements les algorihtmes de Toom-Cook d'ordre k calculent le produit $P \times Q$ en $(2k - 1)$ multiplications de polynômes de taille k fois plus petite.

i. Quelle est leur coût ?

j. Retrouvez comment peut-on les construire

k. Conclure sur le coût du produit de polynômes.

Exercice 5. Produit de Matrices

a. Combien coûte le produit de deux matrices $n \times n$ en utilisant l'algorithme *classique*, de la définition du produit de matrices ?

b. Proposer un algorithme *Diviser Pour Règner* utilisant les découpages suivants sur A et B :

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \text{ et } B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}.$$

où chaque matrice A_{ij} et B_{ij} est de dimension $n/2 \times n/2$.

En suivant les pas de Karatsuba, Strassen a proposé, en 1969, un algorithme multipliant deux matrices de dimensions 2×2 en 7 multiplications. Les 22 opérations suivantes sont effectuées :

— 8 additions :

$$\begin{array}{ll} S_1 \leftarrow A_{21} + A_{22} & T_1 \leftarrow B_{12} - B_{11} \\ S_2 \leftarrow S_1 - A_{11} & T_2 \leftarrow B_{22} - T_1 \\ S_3 \leftarrow A_{11} - A_{21} & T_3 \leftarrow B_{22} - B_{12} \\ S_4 \leftarrow A_{12} - S_2 & T_4 \leftarrow T_2 - B_{21} \end{array}$$

— 7 multiplications :

$$\begin{array}{ll} P_1 \leftarrow A_{11} \times B_{11} & P_5 \leftarrow S_1 \times T_1 \\ P_2 \leftarrow A_{12} \times B_{21} & P_6 \leftarrow S_2 \times T_2 \\ P_3 \leftarrow S_4 \times B_{22} & P_7 \leftarrow S_3 \times T_3 \\ P_4 \leftarrow A_{22} \times T_4 & \end{array}$$

— 7 additions finales :

$$\begin{array}{ll} U_1 \leftarrow P_1 + P_2 & U_5 \leftarrow U_4 + P_3 \\ U_2 \leftarrow P_1 + P_6 & U_6 \leftarrow U_3 - P_4 \\ U_3 \leftarrow U_2 + P_7 & U_7 \leftarrow U_3 + P_5 \\ U_4 \leftarrow U_2 + P_5 & \end{array}$$

— Le résultat est le suivant : $C = \begin{bmatrix} U_1 & U_5 \\ U_6 & U_7 \end{bmatrix}$

c. Déduire un algorithme calculant le produit de deux matrices carrées de dimension $2^k \times 2^k$.

d. Quelle est le coût de cet algorithme ?

e. On a ensuite trouvé des formules (plus compliquées) calculant des produits de matrices 68×68 en 132464 multiplications. Puis 70×70 en 143640 multiplications, ou encore 72×72 en 155424 multiplication. Quelles sont les coûts de ces algorithmes ?

Exercice 6. Inverse de matrice triangulaire

a. Montrer que si $a \neq 0, c \neq 0$ alors $\begin{bmatrix} a & b \\ 0 & c \end{bmatrix}^{-1} = \begin{bmatrix} a^{-1} & -a^{-1}bc^{-1} \\ & c^{-1} \end{bmatrix}$

b. En déduire un algorithme de type diviser pour régner, calculant l'inverse d'une matrice triangulaire supérieure inversible.

c. Quelle est le coût de cet algorithme ? On supposera que le produit de deux matrices carrées $n \times n$ peut se calculer en $\Theta(n^\omega)$ avec $2 < \omega \leq 3$.

On suppose maintenant donné un algorithme `InvTri` qui inverse toute matrice triangulaire supérieure $n \times n$ inversible en $\Theta(n^\alpha)$.

d. Proposer un algorithme qui calcule le produit de deux matrices A et B en utilisant `InvTri`. Quelle est son coût ? Indication : on pourra considérer la matrice $U = \begin{bmatrix} I_n & A & \\ & I_n & B \\ & & I_n \end{bmatrix}$.

e. Que peut-on conclure sur les complexités des deux problèmes : produit de matrice et inversion de matrice triangulaire inversible ?