TD d'Algo n° 3

EPITA ING1 2013

1 On pratique le "théorème général"...

Rappel du théorème vu en cours :

Soit à résoudre $T(n) = aT(n/b) + f(n)$ avec $a \geq 1,\ b > 1$.

— Si $f(n) = \mathrm{O}(n^{\log_b a - \varepsilon})$ pour un $\varepsilon > 0$, alors $T(n) = \Theta(n^{\log_b a})$.
— Si $f(n) = \Theta(n^{\log_b a})$, alors $T(n) = \Theta(n^{\log_b a} \log n)$.
— Si $f(n) = \Omega(n^{\log_b a + \varepsilon})$ pour un $\varepsilon > 0$, **et** si $af(n/b) \leq cf(n)$ pour un $c < 1$ et tous les n suffisamment grands, alors $T(n) = \Theta(f(n))$.
— Dans les autres cas, le théorème ne s'applique pas.

Cela vaut aussi pour des équations de la forme $T(n) = aT(\lfloor n/b \rfloor) + f(n)$ ou $T(n) = aT(\lceil n/b \rceil) + f(n)$.

Utilisez ce théorème pour résoudre les équations de complexité suivantes :

1. $T(n) = 3T(n/3) + \Theta(1)$
2. $U(n) = 2U(n/3) + \Theta(1)$
3. $V(n) = V(n/2) + n + 2$
4. $W(n) = 2W(n/2) + \Theta(\log n)$
5. $X(n) = 2X(n/2) + \Theta(n \log n)$

2 On l'applique sur de vrais algos

2.1 Addition de matrices

1. Écrivez un algorithme itératif, $\textsc{MatAdd}(A, B, n)$, qui calcule la somme de deux matrices de taille $n \times n$. Ça ne doit pas vous prendre plus que deux minutes.
2. Quelle est la complexité de votre algorithme ? (Le théorème ne sert pas ici.)
3. Justifiez qu'il n'est pas possible de faire mieux.

2.2 Multiplication de deux matrices par blocs, façon naïve

Entrée : deux matrices A et B de taille $n \times n$ (une puissance de 2).
Sortie : une matrice $C = AB$ de taille $n \times n$.

$\textsc{MatBMul}(A, B, n)$
0 if $n = 1$ then
1 $\quad C[0][0] \leftarrow A[0][0] \times B[0][0]$
2 $\quad$ return C
3 $\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \leftarrow A$ *// découpe A en 4 sous-matrices de taille $\frac{n}{2} \times \frac{n}{2}$*
4 $\begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} \leftarrow B$
5 $C_{11} \leftarrow \textsc{MatAdd}\big(\textsc{MatBMul}(A_{11}, B_{11}, \frac{n}{2}), \textsc{MatBMul}(A_{12}, B_{21}, \frac{n}{2}),\ \frac{n}{2}\big)$
6 $C_{12} \leftarrow \textsc{MatAdd}\big(\textsc{MatBMul}(A_{11}, B_{12}, \frac{n}{2}), \textsc{MatBMul}(A_{12}, B_{22}, \frac{n}{2}),\ \frac{n}{2}\big)$
7 $C_{21} \leftarrow \textsc{MatAdd}\big(\textsc{MatBMul}(A_{21}, B_{11}, \frac{n}{2}), \textsc{MatBMul}(A_{22}, B_{21}, \frac{n}{2}),\ \frac{n}{2}\big)$
8 $C_{22} \leftarrow \textsc{MatAdd}\big(\textsc{MatBMul}(A_{21}, B_{12}, \frac{n}{2}), \textsc{MatBMul}(A_{22}, B_{22}, \frac{n}{2}),\ \frac{n}{2}\big)$
9 return $\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$

1. Donnez une équation récursive de la complexité de cet algorithme en fonction de n. (Supposez que les opérations de découpage et de regroupement de sous-matrices se font par copie ; donc qu'elles ne sont pas en temps constant.)
2. Appliquez le théorème général pour calculer cette complexité.
3. Que pensez-vous de cet algorithme par rapport à la multiplication naïve (avec une triple boucle).

2.3 Multiplication de deux matrices par blocs, à la Strassen

Mêmes questions pour l'algorithme suivant, où MATSUB fait la soustraction de deux matrices.

MATSMUL(A, B, n)
0 if $n = 1$ then
1 $\quad C[0][0] \leftarrow A[0][0] \times B[0][0]$
2 $\quad$ return C
3 $\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \leftarrow A$
4 $\begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} \leftarrow B$
5 $M_1 \leftarrow \text{MATSMUL}(\text{MATADD}(A_{11}, A_{22}, \frac{n}{2}), \text{MATADD}(B_{11}, B_{22}, \frac{n}{2}), \frac{n}{2})$
6 $M_2 \leftarrow \text{MATSMUL}(\text{MATADD}(A_{21}, A_{22}, \frac{n}{2}), B_{11}, \frac{n}{2})$
7 $M_3 \leftarrow \text{MATSMUL}(A_{11}, \text{MATSUB}(B_{12}, B_{22}, \frac{n}{2}), \frac{n}{2})$
8 $M_4 \leftarrow \text{MATSMUL}(A_{22}, \text{MATSUB}(B_{21}, B_{11}, \frac{n}{2}), \frac{n}{2})$
9 $M_5 \leftarrow \text{MATSMUL}(\text{MATADD}(A_{11}, A_{12}, \frac{n}{2}), B_{22}, \frac{n}{2})$
10 $M_6 \leftarrow \text{MATSMUL}(\text{MATSUB}(A_{21}, A_{11}, \frac{n}{2}), \text{MATADD}(B_{11}, B_{12}, \frac{n}{2}), \frac{n}{2})$
11 $M_7 \leftarrow \text{MATSMUL}(\text{MATSUB}(A_{12}, A_{22}, \frac{n}{2}), \text{MATADD}(B_{21}, B_{22}, \frac{n}{2}), \frac{n}{2})$
12 $C_{11} \leftarrow \text{MATSUB}(\text{MATADD}(M_1, M_4, \frac{n}{2}), \text{MATADD}(M_5, M_7, \frac{n}{2}), \frac{n}{2})$
13 $C_{12} \leftarrow \text{MATADD}(M_3, M_5, \frac{n}{2})$
14 $C_{21} \leftarrow \text{MATADD}(M_2, M_4, \frac{n}{2})$
15 $C_{22} \leftarrow \text{MATADD}(\text{MATSUB}(M_1, M_2, \frac{n}{2}), \text{MATADD}(M_3, M_6, \frac{n}{2}), \frac{n}{2})$
16 return $\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$

Notez qu'il n'est pas nécessaire de comprendre complètement l'algorithme pour en calculer la complexité.

2.4 Exponentiation rapide

Calcule x^p pour $p \in \mathbb{N}$
FASTPOWER(x, p)
1 if $p = 0$ then
2 $\quad$ return 1
3 if $odd(p)$ then
4 $\quad$ return $x \times$FASTPOWER$(x \times x, \lfloor p/2 \rfloor)$
5 else
6 $\quad$ return FASTPOWER$(x \times x, p/2)$

1. En supposant que la multiplication utilisée ici se fait en temps constant, calculez (en fonction de p) la complexité de cet algorithme dans le pire cas et dans le meilleur cas. Déduisez-en la complexité dans le cas général.
2. Cet algorithme peut se décliner pour tout type qui dispose d'une opération associative (ici $\times$) et d'un élément neutre (ici 1) ; c'est-à-dire un type qui possède une structure de monoïde. Proposez une variante de FASTPOWER pour la puissance de matrice en utilisant MATSMUL, et calculez sa complexité en fonction de p (la puissance) et n la taille de la matrice.
3. Comment détourner cet algorithme pour calculer un produit de deux entiers sur une machine dont l'opération de multiplication est hors service ?