TD d'Algo n° 2

EPITA ING1 2012; A. DURET-LUTZ

5 novembre 2009

1 Multiplication de matrices

1.1 Algorithme 1

Rappel : pour deux matrices A et B de taille $n \times n$, on calcule $C = A \times B$ avec

$$\forall i \in [\![1,n]\!], \forall j \in [\![1,n]\!], \quad c_{i,j} = \sum_{k=1}^{n} a_{i,k} \cdot b_{k,j}$$

où $a_{i,j}$, $b_{i,j}$ et $c_{i,j}$ désignent respectivement les coefficients des matrices A, B et C.

1. Écrivez un algorithme de multiplication de deux matrices, basé sur la définition ci-dessus.
2. Calculez sa complexité.

1.2 Algorithme 2

Pour cet algorithme et le suivant, on suppose que n est une puissance de 2 (c'est-à-dire qu'il existe m tel que $n = 2^m$) ; il est toujours possible de s'y ramener en complétant les matrices avec des lignes et des colonnes remplies de 0.

Découpons chacune de ces matrices en quatre blocs de taille $\frac{n}{2} \times \frac{n}{2}$:

$$A = \begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix}, \quad B = \begin{pmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{pmatrix}, \quad C = \begin{pmatrix} C_{1,1} & C_{1,2} \\ C_{2,1} & C_{2,2} \end{pmatrix}$$

On a alors :

$$\begin{aligned} C_{1,1} &= A_{1,1}B_{1,1} + A_{1,2}B_{2,1} \\ C_{1,2} &= A_{1,1}B_{1,2} + A_{1,2}B_{2,2} \\ C_{2,1} &= A_{2,1}B_{1,1} + A_{2,2}B_{2,1} \\ C_{2,2} &= A_{2,1}B_{1,2} + A_{2,2}B_{2,2} \end{aligned}$$

1. Écrivez un algorithme récursif pour le calcul des coefficients de C.
   Vous pouvez supposer qu'il existe des fonctions $(A11, A12, A21, A22) \leftarrow \text{Split2x2}(A)$ et $A \leftarrow \text{Join2x2}(A11, A12, A21, A22)$ pour découper et fusionner blocs de matrices.
2. Donnez une définition récursive de sa complexité.
3. Utilisez le théorème général pour exprimer la complexité de façon explicite.

1.3 Algorithme 3 (dit "Algorithme de Strassen")

Comme dans l'algorithme précédent, on suppose les matrices découpées en quatre blocs de taille $\frac{n}{2} \times \frac{n}{2}$.

Posons

$$\begin{aligned} M_1 &= (A_{1,1} + A_{2,2})(B_{1,1} + B_{2,2}) & M_2 &= (A_{2,1} + A_{2,2})B_{1,1} \\ M_3 &= A_{1,1}(B_{1,2} - B_{2,2}) & M_4 &= A_{2,2}(B_{2,1} - B_{1,1}) \\ M_5 &= (A_{1,1} + A_{1,2})B_{2,2} & M_6 &= (A_{2,1} - A_{1,1})(B_{1,1} + B_{1,2}) \\ M_7 &= (A_{1,2} - A_{2,2})(B_{2,1} + B_{2,2}) \end{aligned}$$

on a alors (faites moi confiance !)

$$\begin{aligned} C_{1,1} &= M_1 + M_4 - M_5 + M_7 & C_{1,2} &= M_3 + M_5 \\ C_{2,1} &= M_2 + M_4 & C_{2,2} &= M_1 - M_2 + M_3 + M_6 \end{aligned}$$

Ceci suggère un second algorithme récursif de calcul des coefficients de C. Quelle est sa complexité ?

2 Quick Sort

On considère l'algorithme Quick Sort implémenté avec un pivot qui est la médiane de trois valeurs :

```
QUICK-SORT(A, l, r)
  if l < r then
    p ← PARTITION(A, l, r)
    QUICK-SORT(A, l, p)
    QUICK-SORT(A, p + 1, r)
```

```
PARTITION(A, l, r)
  x ← median(A[l], A[⌊(l + r)/2⌋], A[r])
  i ← l − 1 ; j ← r + 1
  repeat forever
    do i ← i + 1 until A[i] ≥ x
    do j ← j − 1 until A[j] ≤ x
    if i < j then
      A[i] ↔ A[j]
    else
      return j
```

1. Déroulez l'algorithme sur le tableau $\boxed{5 \mid 4 \mid 2 \mid 8 \mid 1 \mid 2 \mid 3 \mid 7}$.
   Combien d'appel à partition on été effectués ? Quel est la profondeur maximale des appels récursifs (i.e. la hauteur de l'arbre des appels) ? Combien d'appels à median sont effectués ?
2. Quelle est la complexité de l'algorithme lorsqu'on l'applique à un tableau de n éléments triés par ordre décroissant ?
   Utilisez les notations $\Theta(...)$ ou $O(...)$ pour indiquer combien d'appels à median sont effectués dans ce cas.
3. Quelle est la complexité de l'algorithme lorsqu'on l'applique à un tableau dont tous les éléments sont égaux ? Combien d'appels à median sont effectués. ?