Paradigme diviser pour régner [dr]
Algorithmique

Karine Zampieri, Stéphane Rivière, Béatrice Amerein-Soltner

Unisciel algoprog Version 21 mai 2018

Table des matières

Paradigme diviser pour régner

Mots-Clés Techniques de conception, Algorithmes diviser pour régner ■
Requis Axiomatique impérative, Récursivité des actions, Complexité des algorithmes ■
Difficulté ● ● ○

Objectif
Ce module présente le paradigme « diviser pour régner », donne le Master-Théorème des algorithmes « diviser pour régner » puis l'applique sur la multiplication de matrices.

1 Paradigme diviser pour régner

1.1 Principe

Nombres d'algorithmes ont une structure récursive : pour résoudre un problème donné, ils s'appellent eux-mêmes récursivement une ou plusieurs fois sur des problèmes très similaires, mais de tailles moindres, résolvent les sous-problèmes de manière récursive puis combinent les résultats pour trouver une solution au problème initial.

Paradigme « diviser pour régner »
Il donne lieu à trois étapes à chaque niveau de récursivité :

1. **Diviser** le problème en un certain nombre de sous-problèmes.
2. **Régner** sur les sous-problèmes en les résolvant récursivement ou, si la taille d'un sous-problème est assez réduite, le résoudre directement.
3. **Combiner** les solutions des sous-problèmes en une solution complète du problème initial.

Exemple
L'algorithme récursif des Tours de Hanoï (module @[Récursivité des actions]) en est un exemple d'illustration au même titre que celui de la recherche dichotomique (module @[Algorithmes de recherche]) ou encore celui du tri rapide (module @[Algorithmes de tri interne]).

1.2 Analyse des algorithmes

Lorsqu'un algorithme contient un appel récursif à lui-même, son temps d'exécution peut souvent être décrit par une équation de récurrence qui décrit le temps d'exécution global pour un problème de taille n en fonction du temps d'exécution pour des entrées de taille moindre.

La récurrence définissant le temps d'exécution d'un algorithme « diviser pour régner » se décompose suivant les trois étapes du paradigme de base :

1. Si la taille du problème est suffisamment réduite, $n \leq c$ pour une certaine constante c, la résolution est directe et consomme un temps constant $\Theta(1)$.
2. Sinon, on divise le problème en a sous-problèmes chacun de taille $1/b$ de la taille du problème initial. Le temps d'exécution total se décompose alors en trois parties :
   (a) $D(n)$: le temps nécessaire à la division du problème en sous-problèmes.
   (b) $aT(n/b)$: le temps de résolution des a sous-problèmes.
   (c) $C(n)$: le temps nécessaire pour construire la solution finale à partir des solutions aux sous-problèmes.

Relation de récurrence
Elle prend alors la forme :

$$T(n) = \begin{cases} \Theta(1) & \text{si } n \leq c \\ aT(n/b) + D(n) + C(n) & \text{sinon} \end{cases}$$

où l'on interprète n/b soit comme $\lfloor n/b \rfloor$, soit comme $\lceil n/b \rceil$.

2 Master-Théorème

2.1 Master-Théorème

Théorème : Résolution des récurrences « diviser pour régner »
Soient $a \geq 1$ et $b > 1$ deux constantes, soit $f(n)$ une fonction et soit $T(n)$ une fonction définie pour les entiers positifs par la récurrence :

$$T(n) = aT(n/b) + f(n)$$

où l'on interprète n/b soit comme $\lfloor n/b \rfloor$, soit comme $\lceil n/b \rceil$.

Alors $T(n)$ peut être bornée asymptotiquement comme suit :

1. Si $f(n) = O(n^{\log_b a - \varepsilon})$ pour une certaine constante $\varepsilon > 0$, alors $T(n) = \Theta(n^{\log_b a})$.
2. Si $f(n) = \Theta(n^{\log_b a})$ alors $T(n) = \Theta(n^{\log_b a} \log n)$.
3. Si $f(n) = \Omega(n^{\log_b a - \varepsilon})$ pour une certaine constante $\varepsilon > 0$ et si $af(n/b) \leq cf(n)$ pour une certaine constante $c < 1$ et n suffisamment grand, alors $T(n) = \Theta(f(n))$.

Démonstration
Voir [Cormen-A1]. ■

Remarque
Le remplacement des termes $T(n/b)$ par $T(\lfloor n/b \rfloor)$ ou $T(\lceil n/b \rceil)$ n'affecte pas le comportement asymptotique de la récurrence. On omettra donc en général les parties entières.

Ce théorème ne couvre pas toutes les possibilités pour $f(n)$
Par exemple, il y a un « trou » entre les cas 1 et 2 quand $f(n)$ est plus petite que $n^{\log_b a}$ mais pas polynomialement. Dans un tel cas, on ne peut tout simplement pas appliquer le Master-Théorème.

3 Exemple : Multiplication de matrices

Soient $A = (a_{ij})$ et $B = (b_{ij})$ deux matrices carrées de taille $n \times n$.

La multiplication de A et B est définie par

$$C = A \times B$$

avec :

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

3.1 Multiplication naïve de matrices

Algorithme produitMatN
(Version naïve de la multiplication)

```
Action produitMat ( DR A , B : Matrice ; m , n , p : Entier ; R C : Matrice )
Variable ix , jx , kx : Entier
Variable somme : Réel
Début
  |  Pour ix <- 1 à m Faire
  |     |  Pour jx <- 1 à p Faire
  |     |     |  somme <- 0.0
  |     |     |  Pour kx <- 1 à n Faire
  |     |     |     |  somme <- somme + A [ ix , kx ] * B [ kx , jx ]
  |     |     |  FinPour
  |     |     |  C [ ix , jx ] <- somme
  |     |  FinPour
  |  FinPour
Fin
```

Complexité
Cet algorithme effectue $n^2 + n^3$ affectations, n^3 multiplications et autant d'additions. La complexité est donc $n^2 + 2n^3$ opérations. Comme la complexité est indépendante du contenu des matrices (elle ne dépend que de leur taille), la complexité est en $\Theta(n^3)$.

3.2 Algorithme diviser pour régner naïf

Dans la suite nous supposerons que n est une puissance exacte de 2.

Décomposons les matrices A, B et C en sous-matrices de taille $n/2 \times n/2$. L'équation $C = A \times B$ peut alors se réécrire :

$$\begin{pmatrix} r & s \\ t & u \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} e & g \\ f & h \end{pmatrix}$$

En développant cette équation, nous obtenons :

$$\begin{cases} r = a\ e + b\ f \\ s = a\ g + b\ h \\ t = c\ e + d\ f \\ u = c\ g + d\ h \end{cases}$$

Chacune de ces quatre opérations correspond à deux multiplications de matrices carrés de taille $n/2$ et une addition de telles matrices.

L'addition des matrices carrés de taille $n/2$ étant en $\Theta(n^2)$, à partir de ces équations on peut aisément dériver un algorithme « diviser pour régner » dont la complexité est donnée par la récurrence :

$$T(n) = 8T(n/2) + \Theta(n^2)$$

Complexité

Utilisons le Master-Théorème pour calculer la complexité de l'algorithme diviser pour régner naïf. Ici, $a = 8$, $b = 2$ et $f(n) = \Theta(n^2)$.

Ainsi $\log_b a = 3$ et nous nous trouvons donc dans le cas 1 du Master-théorème (avec $\varepsilon = 1$). Par conséquent, l'algorithme a une complexité en $\Theta(n^3)$ et nous n'avons rien gagné.

3.3 Algorithme de Strassen

L'algorithme de Strassen est un algorithme « diviser pour régner » qui n'effectue que 7 multiplications de matrices (contrairement à 8 dans l'algorithme précédent) mais qui effectue plus d'additions et de soustractions de matrices : ceci est sans conséquence, une addition de matrices étant « gratuite » par rapport au coût d'une multiplication.

Complexité

La complexité de l'algorithme de Strassen est donnée par la récurrence suivante :

$$T(n) = 7T(n/2) + \Theta(n^2)$$

En utilisant le Master-théorème, nous obtenons comme complexité :

$$T(n) = \Theta(n^{\lg 7}) = O(n^{2.81})$$

Algorithme

L'algorithme se décompose en quatre étapes :

1. Diviser les matrices A et B en matrices carrés de taille $n/2$.
2. Au moyen de $\Theta(n^2)$ additions et soustractions scalaires, calculer 14 matrices (à préciser) A_1, ..., A_7, B_1, ...,B_7 carrés de taille $n/2$.

3. Calculer récursivement les 7 produits de matrices $P_i = A_i B_i$, $i \in [1..7]$.
4. Calculer les sous-matrices r, s, t et u en additionnant et/ou soustrayant les combinaisons idoines des matrices P_i *ad-hoc*, à l'aide de $\Theta(n^2)$ additions et soustractions scalaires.

Produits de sous-matrices

Supposons que chaque matrice P_i peut s'écrire sous la forme suivante :

$$\begin{aligned} P_i &= A_i B_i \\ &= (\alpha_{i,1}a + \alpha_{i,2}b + \alpha_{i,3}c + \alpha_{i,4}d)(\beta_{i,1}e + \beta_{i,2}f + \beta_{i,3}g + \beta_{i,4}h) \end{aligned}$$

où les coefficients $\alpha_{i,j}$ et $\beta_{i,j}$ sont tous pris dans l'ensemble $\{-1, 0, 1\}$. Nous supposons donc que chaque produit peut être obtenu :

- en additionnant et soustrayant certaines des sous-matrices de A,
- en additionnant et soustrayant certaines des sous-matrices de B,
- et en multipliant les deux matrices ainsi obtenues.

Récrivons l'équation définissant r :

$$r = a\,e + b\,f = (a\ b\ c\ d)\begin{pmatrix} +1 & 0 & 0 & 0 \\ 0 & +1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}\begin{pmatrix} e \\ f \\ g \\ h \end{pmatrix}$$

$$= \begin{array}{c} \\ a \\ b \\ c \\ d \end{array}\begin{array}{c} \begin{array}{cccc} e & f & g & h \end{array} \\ \begin{pmatrix} + & . & . & . \\ . & + & . & . \\ . & . & . & . \\ . & . & . & . \end{pmatrix} \end{array}$$

où « + » représente « +1 », « . » représente « 0 » et « - » représente « -1 ». Récrivons de même les équations définissant s, t et u :

$$s = a\,g + b\,h = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array}\begin{array}{c} \begin{array}{cccc} e & f & g & h \end{array} \\ \begin{pmatrix} . & . & + & . \\ . & . & . & + \\ . & . & . & . \\ . & . & . & . \end{pmatrix} \end{array}$$

$$t = c\,e + d\,f = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array}\begin{array}{c} \begin{array}{cccc} e & f & g & h \end{array} \\ \begin{pmatrix} . & . & . & . \\ . & . & . & . \\ + & . & . & . \\ . & + & . & . \end{pmatrix} \end{array}$$

$$u = c\,g + d\,h = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array}\begin{array}{c} \begin{array}{cccc} e & f & g & h \end{array} \\ \begin{pmatrix} . & . & . & . \\ . & . & . & . \\ . & . & + & . \\ . & . & . & + \end{pmatrix} \end{array}$$

On remarque que l'on peut calculer s par $s = P_1 + P_2$ où P_1 et P_2 sont calculées chacune au moyen d'une **unique** multiplication de matrice :

$$P_1 = A_1 B_1 = a(g-h) = ag - ah = \begin{pmatrix} . & . & + & - \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \end{pmatrix}$$

$$P_2 = A_2 B_2 = (a+b)h = ah + bh = \begin{pmatrix} . & . & . & + \\ . & . & . & + \\ . & . & . & . \\ . & . & . & . \end{pmatrix}$$

De même la matrice t peut être calculée par $t = P_3 + P_4$ avec :

$$P_3 = A_3 B_3 = (c+d)e = ce + de = \begin{pmatrix} . & . & . & . \\ . & . & . & . \\ + & . & . & . \\ + & . & . & . \end{pmatrix}$$

$$P_4 = A_4 B_4 = d(f-e) = df - de = \begin{pmatrix} . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ - & + & . & . \end{pmatrix}$$

Pour calculer r et u on introduit une matrice P_5 définie comme suit :

$$P_5 = A_5 B_5 = (a+d)(e+h) = \begin{pmatrix} + & . & . & + \\ . & . & . & . \\ . & . & . & . \\ + & . & . & + \end{pmatrix}$$

et on cherche à obtenir r à partir de P_5 :

$$\begin{aligned} r &= \begin{pmatrix} + & . & . & . \\ . & + & . & . \\ . & . & . & . \\ . & . & . & . \end{pmatrix} \\ &= \begin{pmatrix} + & . & . & + \\ . & . & . & . \\ . & . & . & . \\ + & . & . & + \end{pmatrix} + \begin{pmatrix} . & . & . & - \\ . & + & . & . \\ . & . & . & . \\ - & . & . & - \end{pmatrix} \\ &= \begin{pmatrix} + & . & . & + \\ . & . & . & . \\ . & . & . & . \\ + & . & . & + \end{pmatrix} + \begin{pmatrix} . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ - & + & . & . \end{pmatrix} + \begin{pmatrix} . & . & . & - \\ . & + & . & . \\ . & . & . & . \\ . & - & . & - \end{pmatrix} \\ &= \begin{pmatrix} + & . & . & + \\ . & . & . & . \\ . & . & . & . \\ + & . & . & + \end{pmatrix} + \begin{pmatrix} . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ - & + & . & . \end{pmatrix} + \begin{pmatrix} . & . & . & - \\ . & . & . & - \\ . & . & . & . \\ . & . & . & . \end{pmatrix} + \begin{pmatrix} . & . & . & . \\ . & + & . & + \\ . & . & . & . \\ . & - & . & - \end{pmatrix} \end{aligned}$$

En posant :

$$P_6 = A_6 B_6 = (b-d)(f+h) = \begin{pmatrix} . & . & . & . \\ . & + & . & + \\ . & . & . & . \\ . & - & . & - \end{pmatrix}$$

on obtient finalement :

$$r = P_5 + P_4 - P_2 + P_6$$

De même, on cherche à obtenir u à partir de P_5 :

$$\begin{aligned} u &= \begin{pmatrix} . & . & . & . \\ . & . & . & . \\ . & . & + & . \\ . & . & . & + \end{pmatrix} \\ &= \begin{pmatrix} + & . & . & + \\ . & . & . & . \\ . & . & . & . \\ + & . & . & + \end{pmatrix} + \begin{pmatrix} - & . & . & - \\ . & . & . & . \\ . & . & + & . \\ - & . & . & . \end{pmatrix} \\ &= \begin{pmatrix} + & . & . & + \\ . & . & . & . \\ . & . & . & . \\ + & . & . & + \end{pmatrix} + \begin{pmatrix} . & . & + & - \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \end{pmatrix} + \begin{pmatrix} - & . & - & . \\ . & . & . & . \\ . & . & + & . \\ - & . & . & . \end{pmatrix} \\ &= \begin{pmatrix} + & . & . & + \\ . & . & . & . \\ . & . & . & . \\ + & . & . & + \end{pmatrix} + \begin{pmatrix} . & . & + & - \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \end{pmatrix} + \begin{pmatrix} . & . & . & . \\ . & . & . & . \\ - & . & . & . \\ - & . & . & . \end{pmatrix} + \begin{pmatrix} - & . & - & . \\ . & . & . & . \\ + & . & + & . \\ . & . & . & . \end{pmatrix} \end{aligned}$$

En posant :

$$P_7 = A_7 B_7 = (a-c)(e+g) = \begin{pmatrix} + & . & + & . \\ . & . & . & . \\ - & . & - & . \\ . & . & . & . \end{pmatrix}$$

on obtient finalement :

$$u = P_5 + P_1 - P_3 - P_7$$

3.4 Discussion

L'algorithme de Strassen n'est intéressant en pratique que pour de grandes matrices ($n > 45$) denses (peu d'éléments non nuls).

La meilleure borne supérieure connue pour la multiplication de matrices carrés de taille n est environ en $O(n^{2,376})$. La meilleure borne inférieure connue est en $\Omega(n^2)$ (il faut générer n^2 valeurs). On ne connaît donc toujours pas le niveau de difficulté réel d'une multiplication de matrices.

4 Conclusion

Paradigme diviser-pour-régner
Technique très utilisée pour l'élaboration de programmes efficaces, elle consiste à diviser un problème de taille n en plusieurs sous-problèmes semblables plus petits de manière que la solution finale soit construite par combinaison des différentes solutions.

Découpage
La manière de décomposer le problème est importante : elle conditionne la performance de l'algorithme. En effet, la complexité totale est fonction du « coût du découpage » et du « coût de résolution des sous-problèmes ». En générale, la meilleure solution sera obtenue avec un découpage en sous-problèmes de tailles quasi-identiques.

5 Références générales

Comprend [Cormen-AL1] ■