Algorithmique Avancée
Diviser pour Régner

Denis TRYSTRAM
Exercice : Karatsuba

sept. 2023

Karatsuba

Multiplication des grands entiers.

Présentation

On considère la représentation en binaire de deux entiers (longs) A and B, de taille $n = 2^k$ pour un entier positif k.

$$A = (a_n a_{n-1} \dots a_1)_2 \quad \text{and} \quad B = (b_n b_{n-1} \dots b_1)_2$$

Méthode simple

L'objectif ici est de calculer la représentation binaire de $A \times B$.

On rappelle que l'algorithme simple (celui que l'on a appris à l'école primaire) est basé sur le calcul de n produits $a_n a_{n-1} \dots a_1$ par b_i pour $i = 1, \dots, n$.

Sa complexité est dans $\Theta(n^2)$.

Etant données les représentations binaires de A and B,

on demande d'écrire un algorithme diviser-pour-régner pour multiplier ces deux entiers.

Approche immédiate :

On découpe chaque entier A and B en deux parties de $n/2$ bits :

$$A = \underbrace{a_n \dots a_{n/2+1}}_{A_1} \cdot 2^{n/2} + \underbrace{a_{n/2} \dots a_1}_{A_2}$$

$$B = \underbrace{b_n \dots b_{n/2+1}}_{B_1} \cdot 2^{n/2} + \underbrace{b_{n/2} \dots b_1}_{B_2}$$

Etant données les représentations binaires de A and B,

on demande d'écrire un algorithme diviser-pour-régner pour multiplier ces deux entiers.

Approche immédiate :

On découpe chaque entier A and B en deux parties de $n/2$ bits :

$$A = \underbrace{a_n \dots a_{n/2+1}}_{A_1} \cdot 2^{n/2} + \underbrace{a_{n/2} \dots a_1}_{A_2}$$

$$B = \underbrace{b_n \dots b_{n/2+1}}_{B_1} \cdot 2^{n/2} + \underbrace{b_{n/2} \dots b_1}_{B_2}$$

Le produit de A par B s'écrit :

$$A \cdot B = A_1 \cdot B_1 \cdot 2^n + (A_1 \cdot B_2 + A_2 \cdot B_1) \cdot 2^{n/2} + A_2 \cdot B_2 \qquad (1)$$

Coût

- 4 multiplications d'entiers de $n/2$-bits
  $A_1 \cdot B_1$, $A_1 \cdot B_2$, $A_2 \cdot B_1$, $A_2 \cdot B_2$
- 3 additions entières avec au plus $2n$ bits
- 2 shifts (multiplications par des puissances de 2)

Coût (suite)

Comme les opérations 2 et 3 peuvent être faites en cn la complexité est donnée par l'équation suivante :

$$\begin{aligned} f(1) &= 1 \\ f(n) &= 4 \cdot f\left(n/2\right) + c \cdot n \end{aligned} \tag{2}$$

Comparaison

On cherche le coût asymptotique pour le comparer à l'algorithme classique.

Le *Master Theorem* donne une solution en $\Theta(n^2)$ car on est dominé par un grand nombre de découpes[1].
Ce n'est donc pas (asymptotiquement) une amélioration de la solution classique !

[1] Notons ici que $2 = log_2(4)$

Comparaison

On cherche le coût asymptotique pour le comparer à l'algorithme classique.

Le *Master Theorem* donne une solution en $\Theta(n^2)$ car on est dominé par un grand nombre de découpes[1].
Ce n'est donc pas (asymptotiquement) une amélioration de la solution classique !

Mais on peut quand même améliorer en diminuant le nombre de sous-problèmes, c'est-à-dire, le nombre d'appels à des multiplications d'entiers de $n/2$-bits.

[1] Notons ici que $2 = log_2(4)$

Version Diviser pour Régner

Montrer que :
$(A_1 - A_2) \cdot (B_2 - B_1) + A_1 B_1 + A_2 B_2 = A_1 B_2 + A_2 B_1$.

Version Diviser pour Régner

Montrer que :
$(A_1 - A_2) \cdot (B_2 - B_1) + A_1 B_1 + A_2 B_2 = A_1 B_2 + A_2 B_1$.

On en déduit l'algorithme diviser-pour-régner de Karatsuba.

$A \cdot B =$

$$A_1 B_1 \cdot 2^n + [A_1 B_1 + A_2 B_2 + (A_1 - A_2) \cdot (B_2 - B_1)] \cdot 2^{n/2} + A_2 B_2 \quad (3)$$

L'expression (3) parait plus compliquée que (1), mais sa résolution ne requiert que :

- 3 multiplications d'entiers de $n/2$-bit
  $A_1 \cdot B_1$
  $A_2 \cdot B_2$
  $(A_1 - A_2) \cdot (B_2 - B_1)$
- 4 additions et 2 soustractions d'entiers d'au plus $2n$ bits
- 2 shifts

Coût

L'équation est gouvernée par la récurrence suivante :

$$\begin{aligned} f(1) &= 1 \\ f(n) &= 3 \cdot f\left(n/2\right) + d \cdot n \end{aligned} \tag{4}$$

pour une certaine constante d.

On applique encore le *Master Theorem*... L'expression (4) est dans $O(n^{\log_2 3})$ où $\log_2 3 \approx 1.59$.

C'est toujours dominé par les découpes, mais le coût est meilleur.