Karatsuba法

梅谷 武

作成：2001-02-14 更新：2005-12-31

多倍長整数の高速乗算法である Karatsuba 法についてわかりやすく解説する。
IMS:20010214001; NDC:418; keywords:Karatsuba 法, Landau の記号;

目　　次

1 Karatsuba法

1.1 ディジタル法

Knuth[S2] によれば、この算法がはじめて公表されたのは、1962 年に当時のソ連の Karatsuba が発表した論文によるそうです。Knuth 自身はこれをディジタル法と名付けています。

2 つの多倍長整数が基数 $P > 0$ によって次のように表現されているとします。

$$\begin{aligned} a &= a_1 P + a_0,\ 0 \leqq a_i < P\,(i = 0, 1) \\ b &= b_1 P + b_0,\ 0 \leqq b_i < P\,(i = 0, 1) \end{aligned}$$

この積を計算します。

$$\begin{aligned} c &= ab \\ &= (a_1 P + a_0)(b_1 P + b_0) \\ &= a_1 b_1 P^2 + (a_1 b_0 + a_0 b_1) P + a_0 b_0 \end{aligned}$$

この最後の式からこの計算には 4 回の積と 3 回の和と桁上げ処理が必要なことがわかります。ところがここで、

$$\begin{aligned} s_0 &= a_0 b_0 \\ s_2 &= a_1 b_1 \\ s_1 &= (a_1 - a_0)(b_0 - b_1) + s_0 + s_2 \end{aligned}$$

とおくと

$$c = s_2P^2 + s_1P + s_0$$

が成り立ちます。この場合は 3 回の積と 4 回の和と 2 回の差と桁上げ処理になります。つまり 1 回の積を 1 回の和と 2 回の差に置き換えたことになります。さらに、

$$\begin{aligned} t_0 &= a_0b_0 \\ t_2 &= a_1b_1 \\ t_1 &= (a_1 + a_0)(b_1 + b_0) - t_0 - t_2 \end{aligned}$$

とおくと

$$c = t_2P^2 + t_1P + t_0$$

が成り立ちます。この場合も 3 回の積と 4 回の和と 2 回の差と桁上げ処理になります。

前者は、例えば 2 進計算機が 32bit の演算器を持つときに、基数 $P = 2^{32}$ として倍精度の乗算に適用することができます。後者は、t_1 の計算の途中で負の値がでてくることはありませんので、多倍長整数計算で値を絶対値表現しているときに有効です。

1.2 計算量の評価

まず、計算量の評価を行なうために使われる道具を導入しておきましょう。

定義 1.1 (Landau の記号) 2 つの実関数 $g(x), h(x)$ について、x がある値 α に近づくときの大小関係を表わすために、

$$g(x) = O(h(x)), g(x) = o(h(x))$$

という記号を次のように定義する。α は $\pm\infty$ であってもよい。

i. 大文字の O は近づき方が同じオーダー *(*程度*)* であることを表わす。

$$g(x) = O(h(x)) \Leftrightarrow \exists A \in \mathbf{R} : \lim_{x\to\alpha} |\frac{g(x)}{h(x)}| < A$$

ii. 小文字の o は近づき方がずっと小さいことを表わす。

$$g(x) = o(h(x)) \Leftrightarrow \lim_{x\to\alpha} \frac{g(x)}{h(x)} = 0$$

n ビット長の多倍長整数の乗算の古典算法による計算時間 (これを計算量と呼ぶ) を $T_{CL}(n)$ とすると、$n \to \infty$ のとき

$$T_{CL}(n) = O(n^2)$$

となることがわかっています。Karatsuba 法の計算量 $T_{KO}(n)$ を評価してみましょう。加減算の計算量は $n \to \infty$ のとき $O(n)$ であることから、

$$T_{KO}(n) \leqq 3T_{KO}(n/2) + kn$$

となる定数 k が存在します。$T_{KO}(1)$ と k の大きい方を改めて k とおき、数列 $2^r | r = 0, 1, \cdots$ について、

$$\begin{aligned} S(2^0) &= k \\ S(2^r) &= 3S(2^{r-1}) + k2^r,\ r > 0 \end{aligned}$$

と定めると

$$S(2^r) = (3^{r+1} - 2^{r+1})k,\ r = 0, 1, \cdots$$

が成り立ちます。なぜならば、$r = 0$ のときは明らかで、$r = n$ で正しいと仮定すると、

$$\begin{aligned} S(2^{n+1}) &= 3S(2^n) + k2^{n+1} \\ &= 3(3^{n+1} - 2^{n+1})k + k2^{n+1} \\ &= (3^{n+2} - 2^{n+2})k \end{aligned}$$

となり、$r = n + 1$ でも成り立つからです (数学的帰納法)。このことから

$$\begin{aligned} T_{KO}(2^r) &\leqq S(2^r) = (3^{r+1} - 2^{r+1})k \\ &< 3^{r+1}k \end{aligned}$$

という不等式が得られます。n を任意に与えたときに、n 以上で最小の 2^r が存在します。このとき

$$\log_2 n \leqq r < \log_2 n + 1$$

となっています。上の不等式を使って、

$$\begin{aligned} T_{KO}(n) &\leqq T_{KO}(2^r) \\ &< 3^{r+1}k = 3k3^r \end{aligned}$$

ここで、

$$3^r < 3^{\log_2 n + 1}$$

と実数のべき乗の定義から導かれる等式

$$3^{\log_2 n} = n^{\log_2 3}$$

を使うと、

$$T_{KO}(n) < 9k \cdot n^{\log_2 3}$$

が導かれます。したがって次の定理が証明されました。

定理 1.2 (Karatsuba 法の計算量) n ビットの多倍長整数に対して *Karatsuba* 法による乗算の計算量を $T_{KO}(n)$ とおくと、$n \to \infty$ のとき、

$$T_{KO}(n) = O(n^{\log_2 3}) = O(n^{1.585})$$

が成り立つ。

参考文献

算法

[S1] 野下 浩平, 高岡 忠雄, 町田 元, “基本的算法”, 岩波書店, 1983

[S2] D. E. Knuth(中川 圭介訳), “準数値算法/算術演算”, サイエンス社, 1986