整数乘法算法

林宪正

算法分析

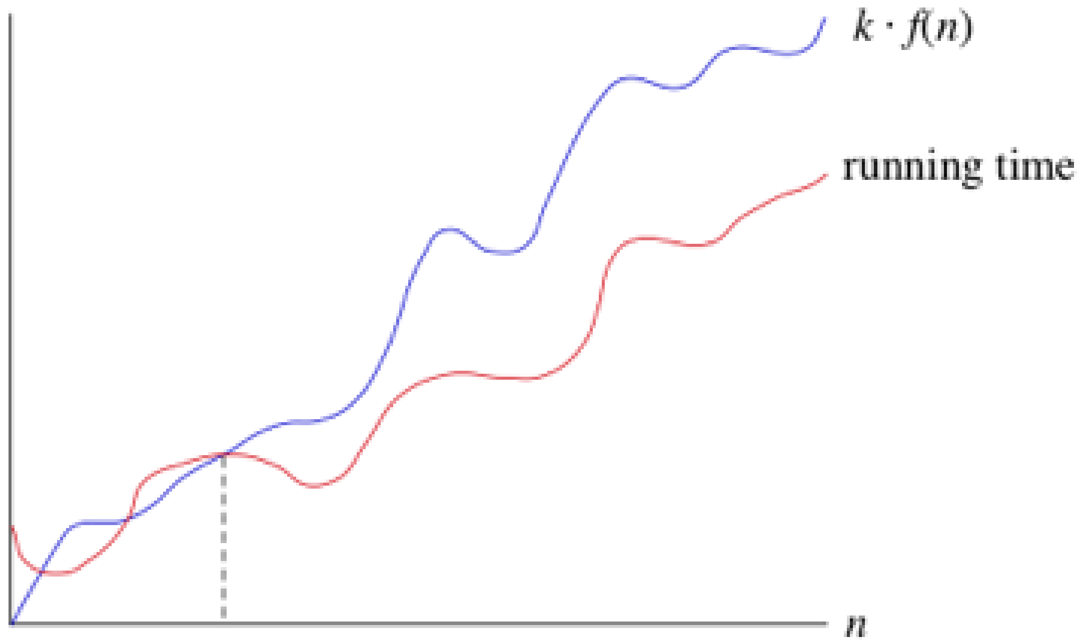
- 算法（Algorithm）：对一定规范的输入，在有限时间内获得所要求的输出。
- 一个算法的优劣可以用空间复杂度与时间复杂度来衡量。
- 搜索问题：给定一个数s，和长度为n的数组A。任务是当s在A中时，输出s的位置，而s不在A中时，输出"未找到"。
- 算法：依次读取A中的每个数，并与s进行比较。
- 假设s在A中且每个位置机率相同，则算法平均需要$(1+2+...+n)/n=(n+1)/2$的比较。如果s不在A中，那么需要n次比较。
- 假设A是已排序的，那么有二分查找算法只需约 $\log_2 n$次比较。

大O符号(Big O notation)

- 目前已知最少计算量的快速傅立叶变换(Tangent FFT)需要 $T(n)=(34/9)n\log_2 n-(124/27)n-2\log_2 n+\ldots$ 乘法与加法 。
- 当n增大时，$n\log_2 n$项将占主导地位，而其他各项可以被忽略。
- $T(n)\in O(n\log_2 n)$或$T(n)=O(n\log_2 n)$

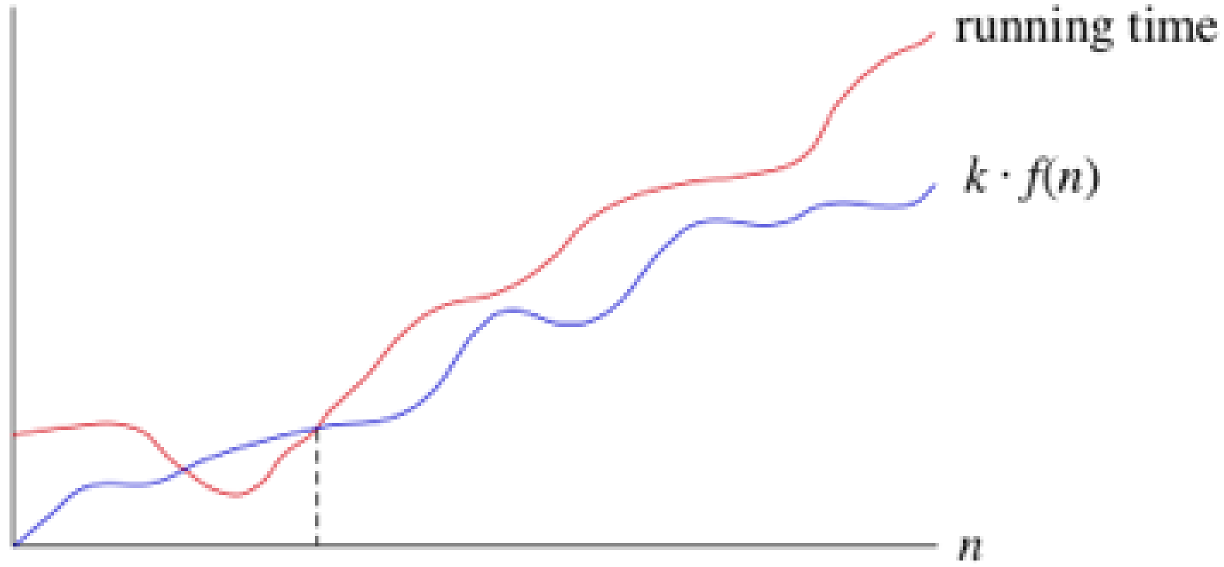
大O符号(Big O notation)

- 給定兩正值函數 R 和 f,定義：$R(n) = O(f(n))$,條件為：存在正實數c和k，使得對於所有的 $n \geq c$，有 $|R(n)| \leq |k \cdot f(n)|$ 。



大Ω符號(Big Omega notation)

- 給定兩正值函數 R 和 f,定義：
  $R(n) = \Omega(f(n))$,條件為：存在正實數c和k，使得對於所有的 $n \geq c$，有 $|R(n)| \geq |k \cdot f(n)|$。



大Θ符號(Big Theta notation)

- 給定兩正值函數 R 和 f,定義：$R(n) = \Theta(f(n))$,條件為： $R(n) = O(f(n))$ 且$R(n) = \Omega(f(n))$ 。

主定理(Master Theorem)

- 假设有递推关系
$$T(n) = aT\left(\frac{n}{b}\right) + f(n),$$
其中$a \geq 1, b > 1$。
- Case 1：如果存在$\epsilon > 0$，有$f(n) = O\left(n^{\log_b(a)-\epsilon}\right)$，则$T(n) = \Theta(n^{\log_b(a)})$。
- Case 2：如果存在$k \geq 0$，有$f(n) = \Theta\left(n^{\log_b(a)} \log^k n\right)$，则$T(n) = O(n^{\log_b(a)} \log^{k+1} n)$。
- Case 3：如果存在$\epsilon > 0$，有$f(n) = \Omega\left(n^{\log_b(a)+\epsilon}\right)$，则$T(n) = \Theta(f(n))$。

长整数乘法

```
      1234567
×        1234
------------------
      4938268
     3703701
    2469134
   1234567
------------------
   1523455678
```

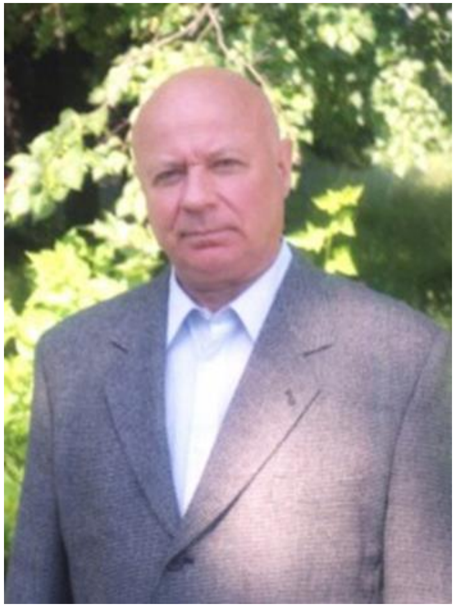
```
      1100110
×        1010
------------------
      0000000
     1100110
    0000000
   1100110
------------------
   1111111100
```

整数乘法算法

- 长整数乘法 $O(n^2)$
- Karatsuba乘法 $O(n^{\log_2 3}) \approx O(n^{1.585})$
- Toom–3乘法 $O\left(n^{\log_3 5}\right) \approx O(n^{1.465})$
- Schönhage–Strassen乘法 $O(n \log n \log \log n)$
- Fürer's algorithm $O(n \log n \, 2^{O(\log^* n)})$
- Harvey et al. algorithm $O(n \log n \, 8^{\log^* n})$

Karatsuba乘法

Andrej Kolmogorov
安德雷·柯爾莫哥洛夫



Anatolii Karatsuba

A. Karatsuba and Y. Ofman, “Multiplication of Many-Digital Numbers by Automatic Computers,” Doklady Akad. Nauk SSSR, vol. 145, No.2, pp. 293-294, July1962.

Karatsuba乘法

- 给予2个n位二进制表示整数A，B，目标为输出$A \times B$的值。
- A与B可表示为

$$A = A_0 + A_1 2^{n/2},$$
$$B = B_0 + B_1 2^{n/2}.$$

- 令

$$A(X) = A_0 + A_1 X,$$
$$B(X) = B_0 + B_1 X.$$

则$A = A(2^{n/2})$，$B = B\left(2^{n/2}\right)$。

- $AB = A(X)B(X)|_{X=2^{n/2}}$

Karatsuba乘法（续）

- $A(X)B(X) = (A_0 + A_1X)(B_0 + B_1X)$
  $= A_0B_0 + X(A_0B_1 + A_1B_0) + X^2A_1B_1$。
- 若直接计算，其计算复杂度为

$$T(n) = 4T\left(\frac{n}{2}\right) + O(n),$$

$T(n) = O(n^2)$。

Karatsuba乘法（续）

- $A(X)B(X) = (A_0 + A_1X)(B_0 + B_1X)$
  $= A_0B_0 + X(A_0B_1 + A_1B_0) + X^2A_1B_1$。
- 其值可用下列方式计算：
  - 计算$Z_0 = A_0B_0$，$Z_\infty = A_1B_1$，$Z_1 = (A_0 + A_1)(B_0 + B_1)$。
  - 则 $A(X)B(X) = Z_0 + X(Z_1 - Z_0 - Z_\infty) + X^2Z_\infty$ 。
- 其计算复杂度为$T(n) = 3T\left(\frac{n}{2}\right) + O(n)$,

$$T(n) = \mathrm{O}(n^{\log_2 3}) \approx \mathrm{O}(n^{1.585})$$

Karatsuba乘法（续）
－多项式求值／内插观点

- 多项式求值：求*A*(*X*)与*B*(*X*)在0, ∞, 1的值：

$A(0) = A_0$，$A(\infty) = A_1$，$A(1) = A_0 + A_1$，
$B(0) = B_0$，$B(\infty) = B_1$，$B(1) = B_0 + B_1$。

- 成对相乘：

$Z_0 = A(0)B(0)$ ，$Z_\infty = A(\infty)B(\infty)$，$Z_1 = A(1)B(1)$。

- 多项式内插：

$$A(X)B(X) = -(X-1)Z_0 + XZ_1 + X(X-1)Z_\infty。$$

Karatsuba乘法（续）-偽代碼

```
mul (A, B){
        if(n< const){
                return A*B;
        }
        Z_0 = mul (A_0, B_0);
        Z_1 = mul (A_0+A_1, B_0+B_1);
        Z_∞ = mul (A_1, B_1);
        return Z_0 + (Z_1–Z_0–Z_∞)2^(n/2)+ Z_∞2^n;
}
```

$$Z_0 = \mathrm{mul}(A_0, B_0);$$
$$Z_1 = \mathrm{mul}(A_0+A_1, B_0+B_1);$$
$$Z_\infty = \mathrm{mul}(A_1, B_1);$$
$$\text{return } Z_0 + (Z_1 - Z_0 - Z_\infty)2^{n/2} + Z_\infty 2^n;$$

Karatsuba乘法实现

- 输入：两个整数(十进制)
- 输出：相乘结果(十进制)
- 实现Karatsuba乘法(十进制)
- 需实现大数加法与大数减法

補充問題

- Unbalanced multiplication

TOOM–3乘法

Toom–3乘法

- 将A与B切成3段
$$A = A_0 + A_1 2^{n/3} + A_2 2^{2\times n/3},$$
$$B = B_0 + B_1 2^{n/3} + B_2 2^{2\times n/3}.$$
- 令
$$A(X) = A_0 + A_1 X + A_2 X^2,$$
$$B(X) = B_0 + B_1 X + B_2 X^2.$$

则$A = A(2^{n/3})$，$B = B\left(2^{n/3}\right)$。
- $AB = A(X)B(X)|_{X=2^{n/3}}$

Toom–3乘法（续）

- $\deg\big(A(X)B(X)\big) < 5$。
- 多项式求值：求*A*(*X*)与*B*(*X*)在0, 1, -1, -2, ∞的值。
- 成对相乘：

$Z_0 = A(0)B(0)$，$Z_1 = A(1)B(1)$，$Z_{-1} = A(-1)B(-1)$，$Z_{-2} = A(-2)B(-2)$，$Z_\infty = A(\infty)B(\infty)$。

- 多项式内插：

$$A(X)B(X) = -\frac{1}{2} \times (X-1)(X+1)(X+2)Z_0 + \frac{1}{6} \times X(X+1)(X+2)Z_1 + \frac{1}{2} \times X(X-1)(X+2)Z_{-1} - 1/6 \times X(X-1)(X+1)Z_{-2} + X(X-1)(X+1)(X+2)Z_\infty$$。

- 其计算复杂度为$T(n) = 5T\left(\frac{n}{3}\right) + O(n)$,

$$T(n) = O\left(n^{\log_3 5}\right) \approx O(n^{1.465})$$

Toom–4乘法

- deg(A(X)B(X))<7
- 多项式求值：求A(X)与B(X)于0, ∞, ±1, ± 2, ½的值并成对相乘。
- $A(X^{-1})=A_0+A_1X^{-1}+A_2X^{-2}+A_3X^{-3}$
  $=X^{-3}(A_3+A_2X^1+A_1X^2+A_0X^3)$
  $=X^{-3}A'(X)$
- $A(2^{-1})\ B(2^{-1})=\ 2^{-6}A'(2)B'(2)$

Schönhage-Strassen乘法

- 多项式求值→快速傅立叶变换(FFT)。
- FFT可快速计算$F(X)$于$\{w^i | i=0,\ldots,n-1\}$的值，其w为n次单位根(n-th root of unity)。
- 典型FFT为复数域上的算法，能否用於处理整数计算？
- 将FFT应用於整数环模N(ring of integers modulo N)。
- 当$N=2^n+1$，存在$2n$次单位根2。
  $2^{2n}=1 \pmod{2^n+1}$
- 对任意整数m，$m \times 2^j \pmod{2^n+1}$可用bit shift快速算出。

Schönhage-Strassen乘法

- 给定A, B, 算法计算AB mod $2^N + 1$。
- 将*A, B*表示成$A(X)$, $B(X)$，且deg $A(X) < n$，deg $B(X) < n$。
- 用FFT计算A(X), B(X)于1, 2, 2^2,…,2^{2n-1}的值。
- 两两相乘。如数字仍太大则采用递回。
- 用IFFT反算$A(X) \times B(X)$，转换成AB并输出结果。

程序库

- The GNU Multiple Precision Arithmetic Library (GMP)
- Fast Library for Number Theory (Flint)
- Number Theory Library (NTL)