Schönhage–Strassen 演算法

R11921128 陳昭瑋

National Taiwan University

基本介紹

- 由 Arnold Schönhage 和 Volker Strassen 於 1971 年發表。
- 一個漸近快速的大整數乘法算法。
- 遞迴地應用數論變換來實現。
- 對於兩個 n 位元整數，時間複雜度為 $O(n \log n \log \log n)$。
- 實際上，用於十進制數字 10,000～40,000 位以上的乘法。
- 應用：梅森質數搜索、計算 π、Lenstra 橢圓曲線分解。

大數乘法演算法整理

- 小學直式乘法：$\mathrm{O}(n^2)$
- 1960 Karatsuba 演算法：$\mathrm{O}\left(n^{\log_2 3}\right)$
- 1963 Toom-Cook 演算法： $\mathrm{O}\left(c(k)n^e\right)$，其中 $e = \log_k(2k-1)$。
- 1971 Schönhage–Strassen 演算法：$\mathrm{O}(n \log n \log\log n)$。
- 2007 Fürer 演算法：$\mathrm{O}\left(2^{2\log^* n} n \log n\right)$，其中 $\log^* n$ 為迭代對數。
- 2021 Harvey & van der Hoeven 演算法：$\mathrm{O}(n \log n)$。

  需要 1729 維的 Fourier transform。

  用在 $10^{10^{38}}$ 位以上十進制數字乘法才能體現出優勢。

Harvey & van der Hoeven $O(n \log n)$ 演算法

Integer multiplication in time $O(n \log n)$

Pages 563-617 from Volume 193 (2021), Issue 2 *by David Harvey, Joris van der Hoeven*

Abstract

We present an algorithm that computes the product of two n-bit integers in $O(n \log n)$ bit operations, thus confirming a conjecture of Schönhage and Strassen from 1971. Our complexity analysis takes place in the multitape Turing machine model, with integers encoded in the usual binary representation. Central to the new algorithm is a novel "Gaussian resampling" technique that enables us to reduce the integer multiplication problem to a collection of multidimensional discrete Fourier transforms over the complex numbers, whose dimensions are all powers of two. These transforms may then be evaluated rapidly by means of Nussbaumer's fast polynomial transforms.

Fourier 轉換 versus 數論轉換

離散 Fourier 轉換（DFT）

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi kn/N} \qquad k = 0, 1, 2, \ldots, N-1$$

離散 Fourier 逆轉換（IDFT）

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j2\pi kn/N} \qquad n = 0, 1, 2, \ldots, N-1$$

數論轉換（NTT）
（Number Theoretic Transform）

$$F[k] = \sum_{n=0}^{N-1} f[n] \omega^{nk} \pmod{M} \qquad k = 0, 1, 2, \ldots, N-1$$

數論逆轉換（INTT）

$$f[n] = N^{-1} \sum_{k=0}^{N-1} F[k] \omega^{-nk} \pmod{M} \qquad n = 0, 1, 2, \ldots, N-1$$

數論轉換

$$F[k] = \sum_{n=0}^{N-1} f[n]\omega^{nk} \pmod{M} \qquad k = 0, 1, 2, \dots, N-1$$

可以改寫成矩陣乘法的形式

$$\begin{bmatrix} F[0] \\ F[1] \\ F[2] \\ \cdots \\ F[N-1] \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega^{1\cdot 1} & \omega^{2\cdot 1} & \cdots & \omega^{(N-1)\cdot 1} \\ 1 & \omega^{1\cdot 2} & \omega^{2\cdot 2} & \cdots & \omega^{(N-1)\cdot 2} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & \omega^{1\cdot (N-1)} & \omega^{2(N-1)} & \cdots & \omega^{(N-1)(N-1)} \end{bmatrix} \begin{bmatrix} f[0] \\ f[1] \\ f[2] \\ \cdots \\ f[N-1] \end{bmatrix} \pmod{M}$$

不用考慮複係數、正餘弦值的浮點數，只用於整數訊號

原理

欲計算整數 A 與 B 的乘積，令

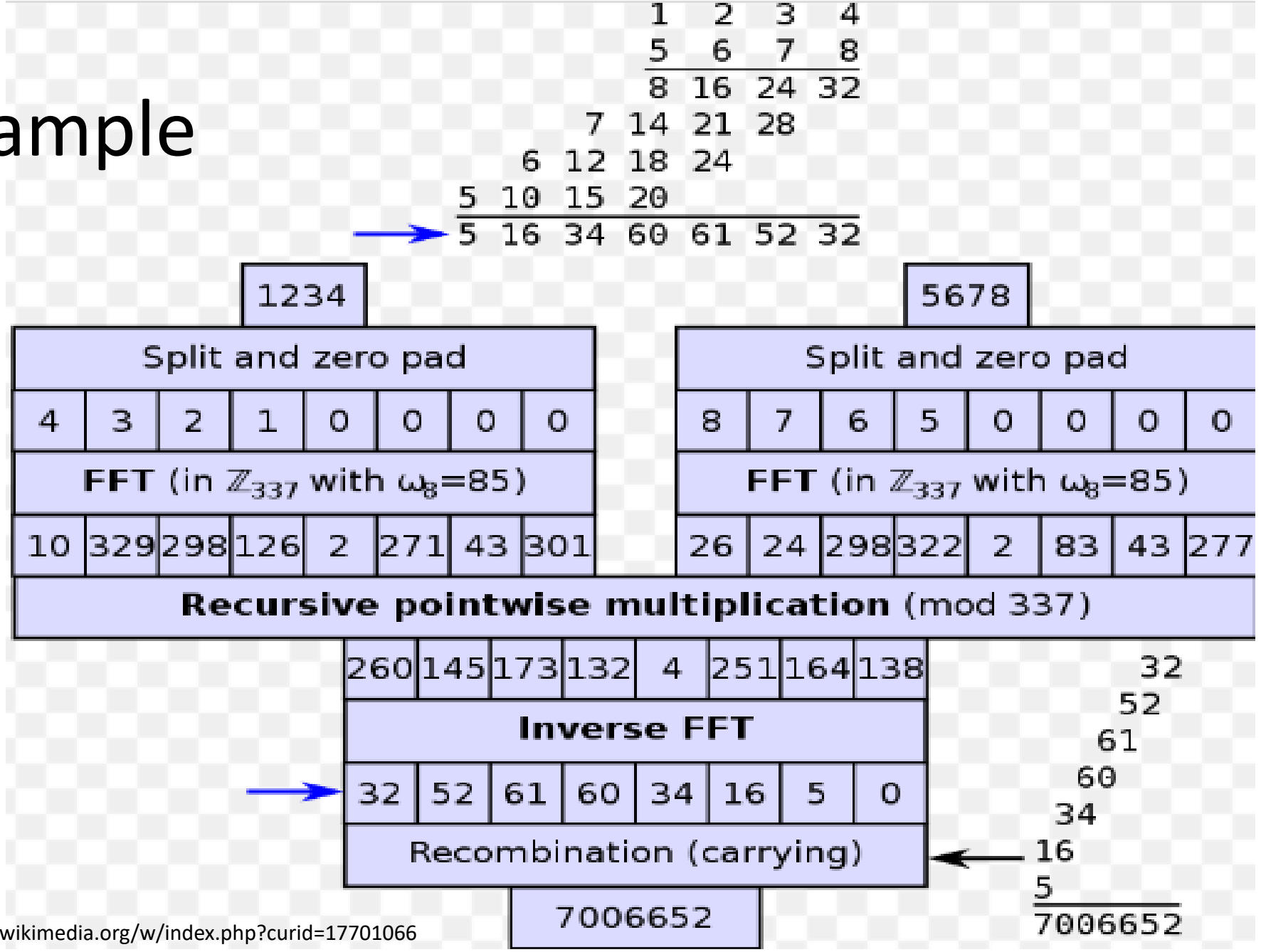
$$A = (a_0 a_1 \dots a_{n-1})_2, B = (b_0 b_1 \dots b_{n-1})_2$$

則 AB 中 2^k 的係數為 $a_0 b_k + a_1 b_{k-1} + \cdots + a_k b_0 = (a * b)[k]$

假設 x, y 皆為長度為 B 的整數訊號

1. 計算 $X := NTT(x)$ 與 $Y := NTT(y)$。
2. 計算 $X[k] \cdot Y[k]$ 得 $NTT(x * y)[k]$。
3. 計算 $INTT\big(NTT(x * y)\big)$，得 $x * y$。

Example



https://commons.wikimedia.org/w/index.php?curid=17701066

Example (cont.) 左半邊 NTT 轉換

取 $M = 337$，$\omega = 85$，

使得 $N = 8$ 為最小的正整數滿足 $\omega^N = 1 \pmod{M}$

$$\begin{bmatrix} 10 \\ 329 \\ 298 \\ 126 \\ 2 \\ 271 \\ 43 \\ 301 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \omega^4 & \omega^5 & \omega^6 & \omega^7 \\ 1 & \omega^2 & \omega^4 & \omega^6 & \omega^8 & \omega^{10} & \omega^{12} & \omega^{14} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \omega^{12} & \omega^{15} & \omega^{18} & \omega^{21} \\ 1 & \omega^4 & \omega^8 & \omega^{12} & \omega^{16} & \omega^{20} & \omega^{24} & \omega^{28} \\ 1 & \omega^5 & \omega^{10} & \omega^{15} & \omega^{20} & \omega^{25} & \omega^{30} & \omega^{35} \\ 1 & \omega^6 & \omega^{12} & \omega^{18} & \omega^{24} & \omega^{30} & \omega^{36} & \omega^{42} \\ 1 & \omega^7 & \omega^{14} & \omega^{21} & \omega^{28} & \omega^{35} & \omega^{42} & \omega^{49} \end{bmatrix} \begin{bmatrix} 4 \\ 3 \\ 2 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \pmod{M}$$

關於 M 的選擇

假設將 n 位元的整數拆成 B 塊，每塊長為 L。（$n = BL$）

注意到

$$INTT\big(NTT(x) \cdot NTT(y)\big) \equiv (x * y) \pmod{M}$$

為避免模數溢出，M 在選擇上應大於 $B(2^L - 1)$，這是因為

$$(x * y)[k] = \sum_{m=0}^{B-1} x[m]y[k-m] \leq \sum_{m=0}^{B-1} (2^L - 1)^2 = B(2^L - 1)$$

Why $O(n \log n \log \log n)$?

令 $\mathcal{M}(n)$ 為兩個 n 位元整數相乘所需的花費，則

$$\mathcal{M}(n) = B\mathcal{M}(L) + O(B \log B)\mathcal{M}(L) + O(n \log B)$$

計算逐點乘積 $X \cdot Y$ 的運行時間

計算 *NTT* 與 *INTT* 中乘法的運行時間

計算 *NTT* 與 *INTT* 中加減法的運行時間

$$O(B \log B)O(L) = O(BL \log B) = O(n \log B)$$

Why $O(n \log n \log \log n)$?

若取 $B = n^{\alpha}$、$L = n^{1-\alpha}$，得

$$\mathcal{M}(n) = n^{\alpha}\mathcal{M}(2n^{1-\alpha}) + O(n \log n)$$

解此遞迴後得結論：

- 當 $\alpha < 1/2$ 時，$\mathcal{M}(n) = O(n \log^2 n)$
- 當 $\alpha = 1/2$ 時，$\mathcal{M}(n) = O(n \log n \log \log n)$
- 當 $\alpha > 1/2$ 時，$\mathcal{M}(n) = O(n \log n)$

$\alpha > 1/2$ 不可能發生，故最佳解為 $O(n \log n \log \log n)$

Schönhage–Strassen 演算法

目標：兩個 n 位元整數相乘
Step 1. 將 n 填充數個 0 使得 $n = 2^k$。
Step 2. 將 n 分割成 $B = \sqrt{n}$ 塊，每塊長為 $L = \sqrt{n}$。
Step 3. 對每一塊做 NTT，其中取 $\omega = 2, M = 2^{2\sqrt{n}} + 1$。
Step 4. 遞迴地計算兩個 NTTs 的逐點乘積。
Step 5. 使用相同的 ω 與 M 計算 INTT。

Note. 當 k 為奇數時，$\sqrt{n}$ 的計算會造成一些困擾
此時可以取 $B = \sqrt{2n},\ L = \sqrt{n/2}$ 或 $B = \sqrt{n/2},\ L = \sqrt{2n}$
可證明這樣的改變不影響最終時間複雜度。

Reference

- https://psun.me/post/fft2/?fbclid=IwAR3zyVLgHcO6CdaO7sA7-8Sv6LfczGdiZ2-LPbopNKkiIEGljFib_VW_i30

  有時間複雜度為 $O(n \log n \log \log n)$ 的推導

- https://en.wikipedia.org/wiki/Sch%C3%B6nhage%E2%80%93Strassen_algorithm

  Schönhage–Strassen 演算法的維基百科頁面