Schnelle diskrete Fouriertransformationen

Bernd Fiedler

Vorlesung

Hochschule für Telekommunikation Leipzig

Sommersemester 2007

Literatur:

Clausen, M., Baum, U. Fast Fourier Transforms. BI Wiss.-verl., 1993.

Zeitkosten für $r!^2$ Integer-Multiplikationen unter Mathematica

r	$t_{PC\ 486}$	$t_{Pentium\ III}$
2	0,0006199 sec	0,0000077 sec
3	0,0055787 sec	0,0000697 sec
4	0,0892584 sec	0,0011157 sec
5	2,2314608 sec	0,0278933 sec
6	80,3325893 sec	1,0041574 sec
7	65,6049479 min	0,8200618 min
8	2,9157754 day	0,0364472 day
9	236,1778125 day	2,9522227 day
10	64,706 year	0,808 year
11	7829,456 year	97,868 year
12	1127441,700 year	14093,021 year

1 Integermult. auf PC 486:
0,000154962556492504409171 0758 sec
$t_{Pentium\ III} = \frac{1}{80}\, t_{PC\ 486}$

Cooley-Tukey FFT

Input:	$n = 2^N$	length
	ω	n-th root of unity
	$(a_0, \ldots, a_{n-1})$	vector of complex coefficients
Output:	$(A_0, \ldots, A_{n-1})^T =$ $\mathsf{DFT}_n \cdot (a_0, \ldots, a_{n-1})^T$	

procedure FFT$(n, \omega, a_0, .., a_{n-1}; A_0, .., A_{n-1})$;
begin
  if $n = 1$ **then**
    $A_0 = a_0$;
  else
    FFT$(\frac{n}{2}, \omega^2, a_0, a_2, .., a_{n-2}; E_0, .., E_{\frac{n}{2}-1})$;
    FFT$(\frac{n}{2}, \omega^2, a_1, a_3, .., a_{n-1}; O_0, .., O_{\frac{n}{2}-1})$;
    for $k = 0$ **to** $n/2 - 1$ **do**
      $A_k := E_k + \omega^k O_k$;
      $A_{k+n/2} := E_k - \omega^k O_k$;
    end do
  end if
end;

Formeln für die schnelle Integer-Multiplikation

Zur Bitlänge N von a, b

$N := \nu \cdot 2^n$, wobei $n - 1 \leq \nu \leq 2n$, $(n \geq 4)$

Q-adisch: $a = \sum_{l=0}^{2^m-1} a_l \cdot Q^l$

Blockanzahl: 2^m mit $m := \lfloor n/2 \rfloor + 1$

$$Q := 2^{N/2^m}$$

Zur Bitlänge K von a_i, b_j, c_l

$K := \kappa \cdot 2^k$ mit $k := \lceil n/2 \rceil + 1$

$$\kappa := \lceil (\nu + 1)/2 \rceil$$

2^m-te primitive Einheitswurzel modulo $(2^K + 1)$

Wegen $k \geq m$ ist $K = \kappa \cdot 2^k$ Vielfaches von $L = 2^m$

$\Rightarrow \omega := 2^{2K/2^m}$ ist 2^m-te primitive Einheitswurzel modulo $(2^K + 1)$

Werte wichtiger Größen für die schnelle Integer-Multiplikation

n	4	5	6	7	8	9
2^n	16	32	64	128	256	512
ν	3	4	5	6	7	8
	8	10	12	14	16	18
N	48	128	320	768	1792	4096
	128	320	768	1792	4096	9216
m	3	3	4	4	5	5
2^m	8	8	16	16	32	32
k	3	4	4	5	5	6
2^k	8	16	16	32	32	64
κ	2	3	3	4	4	5
	5	6	7	8	9	10
K	16	48	48	128	128	320
	40	96	112	256	288	640

Integer-Multiplikation (Version 1)

Input: n, ν (mit $n-1 \leq \nu \leq 2n$), $N := \nu \cdot 2^n$
a, $b > 0$ Integer, (a, b, $a \cdot b$ in N Bit)
Output: $c = a \cdot b$

procedure IntMult(a, b, n, ν, N; c);
begin
  calc m, 2^m, Q, a_i, b_i;
  calc k, κ, K, ω, $\omega^{-1} \bmod(2^K+1)$,
    $(2^m)^{-1} \bmod(2^K+1)$;
  in $\mathbf{Z} \bmod(2^K+1)$ **do**
    FFT$(2^m, \omega, a_0, \ldots, a_{2^m-1}; A_0, \ldots, A_{2^m-1})$;
    FFT$(2^m, \omega, b_0, \ldots, b_{2^m-1}; B_0, \ldots, B_{2^m-1})$;
    for $l = 0$ **to** $2^m - 1$ **do**
      $C_l := A_l \cdot B_l \bmod(2^K+1)$;
    od
    FFT$(2^m, \omega^{-1}, C_0, .., C_{2^m-1}; X_0, .., X_{2^m-1})$;
    for $l = 0$ **to** $2^m - 1$ **do**
      $c_l := (2^m)^{-1} \cdot X_l \bmod(2^K+1)$;
    od
  od
  $c := \sum_{l=0}^{2^m-1} c_l \cdot Q^l$;
end

Selbstaufruf von IntMult möglich?

Input: n, ν (mit $n-1 \leq \nu \leq 2n$), $N := \nu \cdot 2^n$
a, $b > 0$ Integer, (a, b, $a \cdot b$ in N Bit)
Output: $c = a \cdot b \quad \Leftarrow \mathrm{mod}(2^N + 1)$

procedure IntMult$(a, b, n, \nu, N;\ c)$;
begin
calc m, 2^m, Q, a_i, b_i;
calc k, κ, K, ω, $\omega^{-1}\mathrm{mod}(2^K + 1)$,
$(2^m)^{-1}\mathrm{mod}(2^K + 1)$;
in $\mathbf{Z}\,\mathrm{mod}(2^K + 1)$ **do**
FFT$(2^m, \omega, a_0, \ldots, a_{2^m-1}; A_0, \ldots, A_{2^m-1})$;
FFT$(2^m, \omega, b_0, \ldots, b_{2^m-1}; B_0, \ldots, B_{2^m-1})$;
for $l = 0$ **to** $2^m - 1$ **do**
$C_l := A_l \cdot B_l \bmod(2^K + 1)$; $\Leftarrow$ IntMult
od
FFT$(2^m, \omega^{-1}, C_0, .., C_{2^m-1}; X_0, .., X_{2^m-1})$;
for $l = 0$ **to** $2^m - 1$ **do**
$c_l := (2^m)^{-1} \cdot X_l \bmod(2^K + 1)$;
od
od
$c := \sum_{l=0}^{2^m-1} c_l \cdot Q^l$; $\quad \Leftarrow \mathrm{mod}(2^N + 1)$
end

Integer-Multiplikation (Endversion)

Input: n, ν (mit $n-1 \leq \nu \leq 2n$), $N := \nu \cdot 2^n$
a, $b > 0$ Integer, (a, b, $a \cdot b$ in N Bit)
Output: $c = a \cdot b \bmod(2^N + 1)$

procedure IntMult(a, b, n, ν, N; c);
begin
if $n = 3$ **then** $c = a \cdot b \bmod(2^N + 1)$
else
calc m, 2^m, Q, a_i, b_i;
calc k, κ, K, τ, ω,
$\{\omega^{-1}, (2^m)^{-1}, \tau^{-1}\} \bmod(2^K + 1)$;
calc $\alpha_i := a_i \cdot \tau^i \bmod(2^K + 1)$,
$\beta_i := b_i \cdot \tau^i \bmod(2^K + 1)$;
in Z $\bmod(2^K + 1)$ **do**
FFT($2^m, \omega, \alpha_0, .., \alpha_{2^m-1}; A_0, .., A_{2^m-1}$);
FFT($2^m, \omega, \beta_0, .., \beta_{2^m-1}; B_0, .., B_{2^m-1}$);
for $l = 0$ **to** $2^m - 1$ **do**
IntMult(A_l, B_l, k, κ, K; C_l);
od
FFT($2^m, \omega^{-1}, C_0, .., C_{2^m-1}; X_0, ...$);
for $l = 0$ **to** $2^m - 1$ **do**
$c_l := \tau^{-l} \cdot (2^m)^{-1} \cdot X_l \bmod(2^K + 1)$;
od
od
$c := \sum_{l=0}^{2^m-1} c_l \cdot Q^l \bmod(2^N + 1)$;
fi
end