

Den hurtige Fourier-transformation



Jean Baptiste Joseph Fourier (1768-1830)

1

Polynomier



Polynomium:

$$p(x) = 5 + 2x + 8x^2 + 3x^3 + 4x^4$$

Generelt:

$$p(x) = \sum_{i=0}^{n-1} a_i x^i$$

eller

$$p(x) = a_0 + a_1 x + a_2 x^2 + \dots + a_{n-1} x^{n-1}$$

2

Evaluering



Horner's regel:

Givet koefficienter $(a_0, a_1, a_2, \dots, a_{n-1})$, der definerer polynomiet

$$p(x) = \sum_{i=0}^{n-1} a_i x^i$$

For givet x kan vi evaluere $p(x)$ i $O(n)$ tid ved hjælp af ligningen

$$p(x) = a_0 + x(a_1 + x(a_2 + \dots + x(a_{n-2} + xa_{n-1}) \dots))$$

Evaluate(A, x): [hvor $A=(a_0, a_1, a_2, \dots, a_{n-1})$]

if $n=1$ **then return** a_0

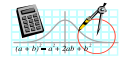
else

$A' \leftarrow (a_1, a_2, \dots, a_{n-1})$ [antag, at dette kan gøres i $O(1)$ tid]

return $a_0 + x * \text{Evaluate}(A', x)$

3

Multiplikation af polynomier



Givet koefficienter $(a_0, a_1, a_2, \dots, a_{n-1})$ og $(b_0, b_1, b_2, \dots, b_{n-1})$, som definerer to polynomier, $p(x)$ og $q(x)$. Bestem polynomiet $p(x)q(x)$, defineret ved

$$p(x)q(x) = \sum_{i=0}^{2n-2} c_i x^i$$

hvor

$$c_i = \sum_{j=0}^i a_j b_{i-j}$$

En lige ud ad landevejen evaluering tager $O(n^2)$ tid

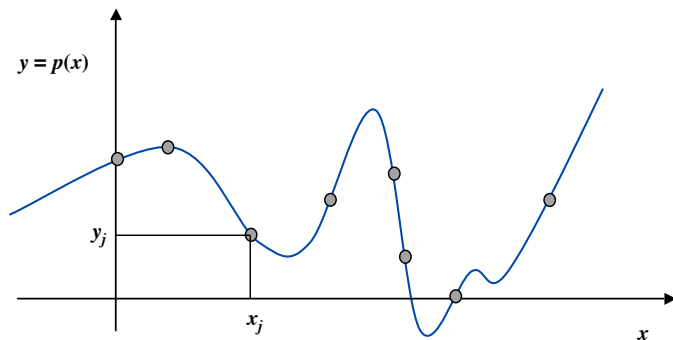
Den "magiske" FFT kan gøre det i $O(n \log n)$ tid

4



Interpolation

Lad der være givet n punkter i planet med forskellige x -koordinater. Da findes der **præcis ét** $(n-1)$ 'te-grad polynomium, der går igennem alle disse punkter



5



Interpolation og multiplikation

Alternativ metode til beregning af $p(x)q(x)$:

Beregn $p(x)$ for $2n$ værdier, $x_0, x_1, \dots, x_{2n-1}$

Beregn $q(x)$ for de samme $2n$ værdier

Find det $(2n-1)$ 'te-grad polynomium, $r(x)$, der går igennem punkterne $\{(x_0, p(x_0)q(x_0)), (x_1, p(x_1)q(x_1)), \dots, (x_{2n-1}, p(x_{2n-1})q(x_{2n-1}))\}$

Beregn værdien af $r(x)$

En lige ud af landevejen evaluering tager uheldigvis stadig $O(n^2)$ tid, da vi for hvert af de $2n$ punkter skal anvende Horner's regel (som tager $O(n)$ tid)

Den "magiske" FFT kan gøre det i $O(n \log n)$ tid ved at vælge $2n$ punkter, der er lette at evaluere...

6



Primitive enhedsrødder

Et tal ω er en **primitiv n 'te enhedsrod**, $n > 1$, hvis

- $\omega^n = 1$
- Tallene $1, \omega, \omega^2, \dots, \omega^{n-1}$ er indbyrdes forskellige

7



Eksempel 1 (Z_{11}^*)

x	x^2	x^3	x^4	x^5	x^6	x^7	x^8	x^9	x^10
1	1	1	1	1	1	1	1	1	1
2	4	8	5	10	9	7	3	6	1
3	9	5	4	1	3	9	5	4	1
4	5	9	3	1	4	5	9	3	1
5	3	4	9	1	5	3	4	9	1
6	3	7	9	10	5	8	4	2	1
7	5	2	3	10	4	6	9	8	1
8	9	6	4	10	3	2	5	7	1
9	4	3	5	1	9	4	3	5	1
10	1	10	1	10	1	10	1	10	1

2, 6, 7, 8 er 10'nde enhedsrødder i Z_{11}^*

$2^2=4, 6^2=3, 7^2=5, 8^2=9$ er 5'te enhedsrødder i Z_{11}^*

$2^{-1}=6, 3^{-1}=4, 4^{-1}=3, 5^{-1}=9, 6^{-1}=2, 7^{-1}=8, 8^{-1}=7, 9^{-1}=5$

8

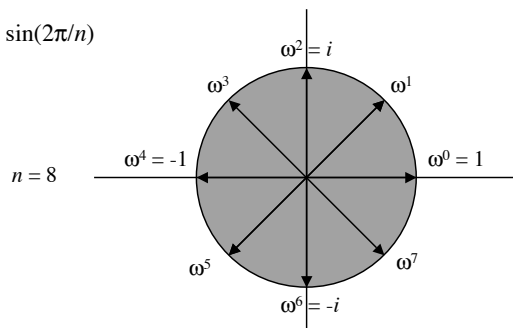
Eksempel 2



Det komplekse tal $e^{2\pi i/n}$ er en primitiv n 'te enhedsrod, hvor

$$i = \sqrt{-1}$$

$$e^{2\pi i/n} = \cos(2\pi/n) + i \sin(2\pi/n)$$



9

Egenskaber



Inversegenskaben: Hvis ω er en primitiv n 'te enhedsrod, så er $\omega^{-1} = \omega^{n-1}$

Bevis: $\omega\omega^{n-1} = \omega^n = 1$

Annuleringsegenskaben: For $-n < k < n$ og $k \neq 0$ er $\sum_{j=0}^{n-1} \omega^{kj} = 0$

Bevis:

$$\sum_{j=0}^{n-1} \omega^{kj} = \frac{(\omega^k)^n - 1}{\omega^k - 1} = \frac{(\omega^n)^k - 1}{\omega^k - 1} = \frac{(1)^k - 1}{\omega^k - 1} = \frac{1-1}{\omega^k - 1} = 0$$

10

Flere egenskaber



Reduktionsegenskaben: Hvis ω er en primitiv $2n$ 'te enhedsrod, så er ω^2 en primitiv n 'te enhedsrod

Bevis: Hvis $1, \omega, \omega^2, \dots, \omega^{2n-1}$ er indbyrdes forskellige, så er $1, \omega^2, (\omega^2)^2, \dots, (\omega^2)^{n-1}$ også indbyrdes forskellige

Refleksionsegenskaben: Hvis n er lige, er $\omega^{n/2} = -1$

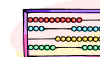
Bevis: Ved brug af annulleringsegenskaben for $k = n/2$ fås

$$0 = \sum_{j=0}^{n-1} \omega^{(n/2)j} = \omega^0 + \omega^{n/2} + \omega^0 + \omega^{n/2} + \dots + \omega^0 + \omega^{n/2} = (n/2)(1 + \omega^{n/2})$$

Følgesætning: For n lige gælder $\omega^{k+n/2} = -\omega^k$

11

Den diskrete Fourier-transformation



Givet koefficienterne $(a_0, a_1, a_2, \dots, a_{n-1})$ for et $(n-1)$ 'te grad polynomium $p(x)$

Den **diskrete Fourier-transformation** (DFT) evaluerer p for værdierne $1, \omega, \omega^2, \dots, \omega^{n-1}$

Vi beregner $(y_0, y_1, y_2, \dots, y_{n-1})$, hvor $y_j = p(\omega^j)$, d.v.s.

$$y_j = \sum_{i=0}^{n-1} a_i \omega^{ij}$$

På matrix-form: $y = Fa$, hvor $F[i,j] = \omega^{ij}$

Den **inverse diskrete Fourier-transformation** bestemmer koefficienterne for et $(n-1)$ 'te grad polynomium givet dets værdier for $1, \omega, \omega^2, \dots, \omega^{n-1}$

På matrix form: $a = F^{-1}y$, hvor $F^{-1}[i,j] = \omega^{-ij}/n$

12

Korrekthed af den inverse DFT



DFT og den inverse DFT er faktisk inverse operationer

Bevis: Lad $A = F^{-1}F$. Vi ønsker at vise, at $A = I$, hvor

$$A[i, j] = \frac{1}{n} \sum_{k=0}^{n-1} \omega^{-ki} \omega^{kj}$$

Hvis $i = j$, har vi

$$A[i, i] = \frac{1}{n} \sum_{k=0}^{n-1} \omega^{-ki} \omega^{ki} = \frac{1}{n} \sum_{k=0}^{n-1} \omega^0 = \frac{1}{n} n = 1$$

Hvis i og j er forskellige, har vi

$$A[i, j] = \frac{1}{n} \sum_{k=0}^{n-1} \omega^{(j-i)k} = 0 \quad (\text{ved brug af annulleringsegenskaben})$$

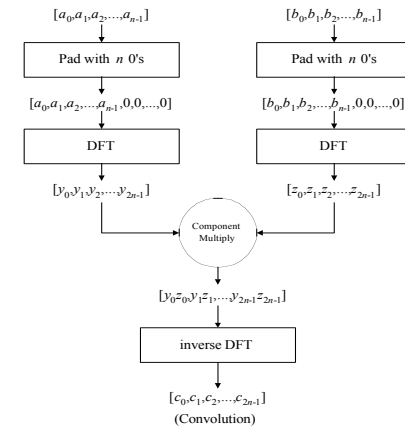
13

Foldning



DFT og den inverse DFT kan bruges til at multiplicere to polynomier

Derfor kan vi bestemme koefficienterne for produkt-polynomiet hurtigt, hvis vi kan bestemme DFT og dens inverse hurtigt



14

Den hurtige Fourier-transformation



Den **hurtige Fourier-transformation** (FFT) er en effektiv algoritme til beregning af DFT

FFT er baseret på paradigmet del-og-hersk:

Hvis n er lige, opdeles polynomiet

$$p(x) = a_0 + a_1x + a_2x^2 + \dots + a_{n-1}x^{n-1}$$

i polynomierne

$$p^{\text{even}}(x) = a_0 + a_2x + a_4x^2 + \dots + a_{n-2}x^{n/2-1}$$

$$p^{\text{odd}}(x) = a_1 + a_3x + a_5x^2 + \dots + a_{n-1}x^{n/2-1}$$

Vi har da, at

$$p(x) = p^{\text{even}}(x^2) + xp^{\text{odd}}(x^2).$$

15

FFT-algoritmen



Algorithm FFT(\mathbf{a}, ω):

Input: An n -length coefficient vector $\mathbf{a} = [a_0, a_1, \dots, a_{n-1}]$ and a primitive n th root of unity ω , where n is a power of 2

Output: A vector \mathbf{y} of values of the polynomial for \mathbf{a} at the n th roots of unity

if $n = 1$ **then**

return $\mathbf{y} = \mathbf{a}$.

$x \leftarrow \omega^0$ { x will store powers of ω , so initially $x = 1$.}

{Divide Step, which separates even and odd indices}

$\mathbf{a}^{\text{even}} \leftarrow [a_0, a_2, a_4, \dots, a_{n-2}]$

$\mathbf{a}^{\text{odd}} \leftarrow [a_1, a_3, a_5, \dots, a_{n-1}]$

{Recursive Calls, with ω^2 as $(n/2)$ th root of unity, by the reduction property}

$\mathbf{y}^{\text{even}} \leftarrow \text{FFT}(\mathbf{a}^{\text{even}}, \omega^2)$

$\mathbf{y}^{\text{odd}} \leftarrow \text{FFT}(\mathbf{a}^{\text{odd}}, \omega^2)$

{Combine Step, using $x = \omega^j$ }

for $i \leftarrow 0$ **to** $n/2 - 1$ **do**

$y_i \leftarrow y_i^{\text{even}} + x \cdot y_i^{\text{odd}}$

$y_{i+n/2} \leftarrow y_i^{\text{even}} - x \cdot y_i^{\text{odd}}$ {Uses reflective property}

$x \leftarrow x \cdot \omega$

return \mathbf{y}

Køretiden er $O(n \log n)$

16

Multiplikation af store heltal



Givet to N -bit heltal I og J . Beregn IJ

Antag, at vi kan multiplicere ord af længde $O(\log N)$ bit i konstant tid

Find et primtal $p = cn + 1$, der kan repræsenteres i et ord

Sæt $m = \lfloor (\log p) / 2 \rfloor$, således at I og J kan betragtes som vektorer af længde n bestående af m -bit ord

Bestem en primitiv enhedsrod:

Find en generator x i Z_p^* .

Så er $\omega = x^c$ en primitiv n 'te enhedsrod i Z_p^*

Foretag FFT og invers FFT for at beregne foldningen C af I med J

Beregn

$$K = \sum_{i=0}^{n-1} c_i 2^{mi}$$

K er værdien af IJ

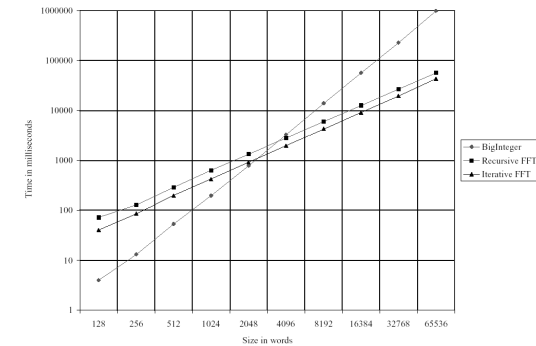
Beregningerne tager $O(n \log n)$ tid, eller $O(N)$, hvis n vælges, så n er $O(N / \log N)$

17

Eksperimentelle resultater



Log-log skala viser at sædvanlig multiplikation kører i kvadratisk tid, mens FFT-versioner kører i næsten lineær tid.



18