FFT – szybka transformacja Fouriera

Problem: mnożenie wielomianów

Dane: $\{a_i\}, \{b_i\}$ t. że $\begin{cases} A(x) = \sum a_i x^i \\ B(x) = \sum b_i x^i \end{cases}$

wyniki: $\{c_i\}$ t. że $C(x) = \sum c_i x^i = A(x) \cdot B(x)$

Naiwnie: $O(n^2)$

- Może jakieś dziel i zwyciężaj?

Reprezentacje wielomianów (n-tego stopnia):

- $n+1$ współczynników
- wartości w $n+1$ punktach

operacja	wsp.	punkty
+	$\Theta(n)$	$\Theta(n)$
*	$\Theta(n^2)$	$\Theta(n)$

punkty ← muszą być w tych samych punktach

$\Theta(n)$ ← potrzebujemy więcej punktów (2n = stopień wynikowy)

Chcemy wsp. $\xrightarrow{?}$ punkty $\xrightarrow[\text{mnożenie}]{\Theta(n)}$ punkty $\xrightarrow{?}$ wsp.

Rozpatrzmy:

$$\begin{array}{ccc} \text{wsp} & & \text{wsp} \\ \Big\downarrow \Theta(n^2) & & \Big\uparrow \Theta(n^2) \\ \text{PUNKTY} & \longrightarrow & \text{PUNKTY} \end{array}$$

n razy schemat Hornera → $\Theta(n^2)$

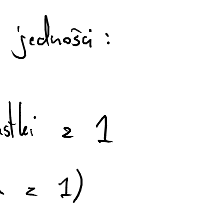
$\Theta(n^2)$ – interpolacja Lagrange'a

- te algorytmy działają dla dowolnych punktów

Może dla konkretnych możnaby szybciej?

↓

n-te pierwiastki z jedności



(A nawet 2^k-te pierwiastki z jedności.)

Niech $n = 2^k$

Def.: n-ty <u>pierwotny</u> pierwiastek z jedności:

↓

- generuje wszystkie n-te pierwiastki z 1

(• i jest n-tym pierwiastkiem z 1)

Def.: $\omega_n = e^{\frac{2\pi i}{n}}$

- jest n-tym pierwotnym pierw. z 1
- $\forall n d k.\ \omega_{dn}^{dk} = \omega_n^k$
- $\forall k.\ \{ (\omega_{2k}^i)^2 \mid i \in \underline{2k} \} = \{ \omega_k^i \mid i \in \underline{k} \}$

$\underline{n} = \{0, 1, \ldots, n-1\}$

Mamy: $A(x) = \sum_{i=0}^{2k-1} a_i x^i$

Chcemy policzyć $A(\omega_{2k}^i)$ $i \in \underline{2k}$

Niech:

$A_0(x) = \sum_{i=0}^{k-1} a_{2i} x^i$

$A_1(x) = \sum_{i=0}^{k-1} a_{2i+1} x^i$

Wtedy:

$A(x) = A_0(x^2) + x A_1(x^2)$

A ma stopień 2n; A_0, A_1 mają n

Co więcej (wedle obserwacji wyżej):

$\{ A((\omega_{2n}^i)^2) \mid i \in \underline{2n} \} = \{ A(\omega_n^i) \mid i \in \underline{n} \}$

Mamy wsp. → punkty

Jak policzyć punkty → wartości?

$a_i \to A(\omega_n^i)$

$A(\omega_n^0) = a_0 + a_1 \omega_n^0 + a_2 (\omega_n^0)^2 + \ldots$

$A(\omega_n^1) = a_0 + a_1 \omega_n^1 + a_2 (\omega_n^1)^2 + \ldots$

$\bar{a} = a_0, \ldots, a_n$

$$\underbrace{\begin{bmatrix} (\omega_1^0)^0 & (\omega_1^0)^1 & \cdots & (\omega_1^0)^{n-1} \\ (\omega_1^1)^0 & (\omega_1^1)^1 & & \\ \vdots & & \ddots & \\ (\omega_n^{n-1})^0 & & & (\omega_n^{n-1})^{n-1} \end{bmatrix}}_{V} \underbrace{\begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{bmatrix}}_{\bar{a}} = \underbrace{\begin{bmatrix} A(\omega_n^0) \\ A(\omega_n^1) \\ \vdots \\ A(\omega_n^{n-1}) \end{bmatrix}}_{\bar{y}}$$

FFT tak naprawdę oblicza $y = f(\bar{a}) = V \cdot \bar{a}$ w $\Theta(n \log n)$

Zobaczmy, że $\bar{a} = V^{-1} y$

$$V_n = \begin{bmatrix} (\omega_n^0)^0 & (\omega_n^0)^1 & \cdots & (\omega_n^0)^{n-1} \\ (\omega_n^1)^0 & (\omega_n^1)^1 & & \\ \vdots & & \ddots & \\ (\omega_n^{n-1})^0 & & & (\omega_n^{n-1})^{n-1} \end{bmatrix}$$

V_n jest macierzą Vandermonde'a

↳ jest nieosobliwa

Można pokazać, że $(V_n^{-1})_{ij} = \frac{1}{n} \omega_n^{-ij} = \frac{1}{n} (\omega_n^{-1})^{ij}$

$[(V_n)_{ij} = \omega_n^{ij}]$

ω_n^{-1} ↑ też jest pierwiastkiem pierwotnym

$n \cdot V_n = [(\omega_n^{-1})^{ij}]_{ij}$

$V_n \cdot \bar{y} = [(\omega_n^{-1})^{ij}]_{ij} \cdot \frac{1}{n} \bar{y}$

Można nie używać liczb zespolonych (ω_n)

- zamiast tego np. $\mathbb{Z}_p$

(chodzi o to, żeby mieć dużo pierwiastków)

Fakt

Niech n i w – potęgi liczby 2; $n, w \neq 1$

$m = w^{\frac{n}{2}} + 1$

Wówczas

n, w są odwracalne w $\mathbb{Z}_m$

oraz

w jest n-tym pierwotnym pierwiastkiem z jedności.

→ mnożenie liczb

(robiliśmy w $O(n^{1+\varepsilon})$, teraz możemy spróbować lepiej FFT)

$L = a_{n-1} a_{n-2} \ldots a_0$

↓

$A(x) = \sum a_i x^i$

→ Algorytm Schönhage-Strassena

(mnożenie liczb n-bitowych w czasie $O(n \log n \cdot \log \log n)$)