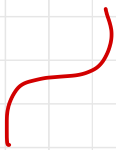
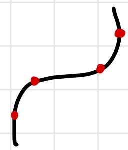
FFT

FFT

Szybka transformata Fouriera to algorytm zamieniający wielomian P z reprezentacji argumentowej na wartościową za pomocą techniki dziel i zwyciężaj.



$\Leftrightarrow$



$$\underbrace{2x^3 + x^2}_{n \text{ argumentów}} \qquad \underbrace{\langle 1,3\rangle, \langle -1,-3\rangle, \ldots}_{2n \text{ punktów}}$$

Proces zamiany z postaci wartościowej na argumentową nazywamy odwrotną transformatą Fouriera. Co ciekawe jest to ten sam algorytm z lekko zmodyfikowanymi danymi.

Zauważmy, że dla każdego wielomianu wystarczy obliczyć tylko $\frac{n}{2}$ wartości, ponieważ te same ujemne wartości będą symetrycznie odbite:

$$P_e(-x) = P_e(x) \qquad P_o(-x) = P_o(x)$$

„even" czyli parzysty wielomian np. x^2+2

„odd" czyli nieparzysty np. x^3+4x

Weźmy nasz wielomian P i wyciągnijmy z niego P_e oraz P_o.

$$P(x) = P_e(x^2) + x\,P_o(x^2)$$

<u>Przykład</u>

$$P(x) = 3x^5 + 2x^4 + x^3 + 7x^2 + 5x + 1$$

$$P_e(x^2) = 2x^4 + 7x^2 + 1$$
$$x \cdot P_o(x^2) = 3x^4 + x^2 + 5$$
$$\Downarrow$$
$$P_e(x) = 2x^2 + 7x + 1$$
$$x \cdot P_o(x) = 3x^2 + x + 5$$

P_e oraz P_o mają teraz stopień $\lfloor \frac{n}{2} \rfloor$

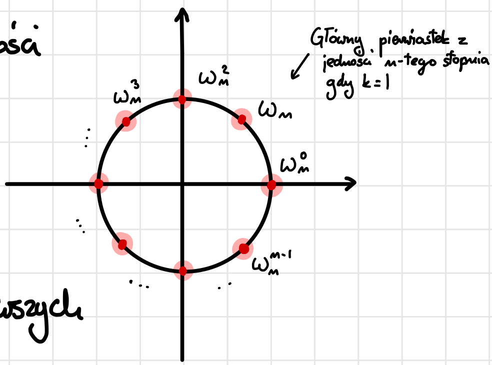
Następnie argumentami x, które są odpowiednie do podstawienia to są zespolone pierwiastki z jedności n-tego stopnia.

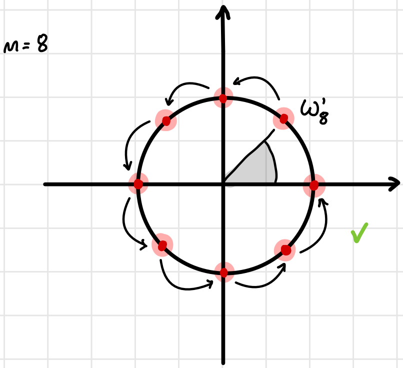
liczba zespolona

$$\omega_n^k = e^{\frac{2\pi i k}{n}}$$ dla $k = 0, 1, \ldots, n-1$ ← pierwiastek z jedności n-tego stopnia

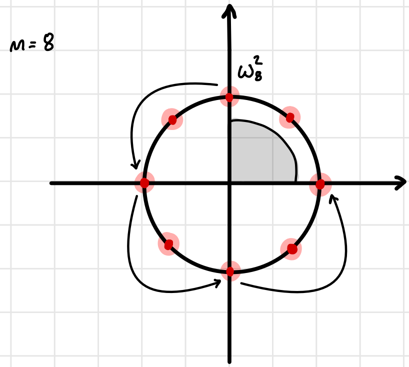
Istnieje n takich pierwiastków z jedności

ω_n dla n ma $\frac{n}{2}$ generujących pierwiastków z jedności, które przesuwane o swój kąt dadzą nam wszystkie n różnych pierwiastków n-tego stopnia.
Są to pierwiastki o potęgach względnie pierwszych z n.





Punkt ω_8^1 jest dobrym pierwiastkiem, ponieważ przesuwany o jego kąt (być może wielokrotnie) na okręgu jedności przejdzie przez każdy punkt.



Punkt ω_8^2 jest złym pierwiastkiem, ponieważ przesuwany o jego kąt (być może wielokrotnie) na okręgu jedności nie przejdzie przez każdy punkt.

Zatem algorytm wygląda tak:

1. Podziel wielomian P o długości n na P_e i P_o
2. $y_e, y_o = \mathrm{FFT}(P_e), \mathrm{FFT}(P_o)$
3. y = pusta tablica rozmiaru n
4. **for** j **from** 0 **to** $\frac{n}{2}$
   a) $y[j] = y_e[j] + y_o[j] \cdot \omega_n^j$ ← wartości dla argumentów dodatnich
   b) $y[j + \frac{n}{2}] = y_e[j] - y_o[j] \cdot \omega_n^j$ ← wartości dla argumentów ujemnych
5. **return** y

Inna reprezentacja FFT

$y = V \cdot a$ gdzie:

y - wynik FFT
a - argumenty
V_n - macierz, której (j,k)-ty wyraz równa się ω_n^{jk}

Przykład:

$$V_n = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_n & \omega_n^2 & \cdots & \omega_n^{n-1} \\ 1 & \omega_n^2 & \omega_n^4 & \cdots & \omega_n^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \\ 1 & \omega_n^{n-1} & \omega_n^{2(n-1)} & & \omega_n^{(n-1)(n-1)} \end{bmatrix}$$

IFFT

Inverse FFT przekształca wielomian z postaci wartościowej na postać argumentową.

Odwrócony algorytm FFT ma wszystko to samo oprócz macierzy przekształcenia V_n.

$$a = V_n^{-1} y$$

$$V_n^{-1} = \frac{1}{n} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_n^{-1} & \omega_n^{-2} & \cdots & \omega_n^{-(n-1)} \\ 1 & \omega_n^{-2} & \omega_n^{-4} & \cdots & \omega_n^{-2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \\ 1 & \omega_n^{-(n-1)} & \omega_n^{-2(n-1)} & & \omega_n^{-(n-1)(n-1)} \end{bmatrix}$$