Lecture 3

Fast Fourier Transform I

Prof. Piotr Indyk
Fast Fourier Transform

- **Discrete Fourier Transform (DFT):**
  - Given: coefficients of a polynomial
    \[ A(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_{n-1} x^{n-1} \]
  - Goal: evaluate \( A(x_0), A(x_1), \ldots, A(x_{n-1}) \), where \( x_j = \omega_n^j, \omega_n = e^{2\pi i/n} = \cos(2\pi/n) + is\sin(2\pi/n) \)

- **Inverse Discrete Fourier Transform (IDFT):**
  - Given: values \( A(x_0), A(x_1), \ldots, A(x_{n-1}) \),
  - Goal: interpolate the coefficients of the polynomial \( A \)

- **FFT:**
  - Divide and conquer algorithm
  - Performs both tasks using \( O(n \log n) \) arithmetic operations
  - Amazing number of applications!

- **Plan:**
  - Today: intuitions, applications and a proof
  - Next week: the FFT algorithm
Fourier Transform Intuitions
(6.03 in 5 minutes)

• Fourier transform maps a vector \([a_0\ldots a_{n-1}]\) into \([y_0\ldots y_{n-1}]\), where

\[y_j = A(x_j) = \sum_k a_k x_j^k = \sum_k a_k e^{2\pi i j k/n} = \sum_k a_k \left[\cos(2\pi j k/n) + is\sin(2\pi j k/n)\right]\]

• Inverse Fourier transform does the opposite

• Physical interpretation:
  – \([a_0\ldots a_{n-1}]\): signal in time domain
  – \([y_0\ldots y_{n-1}]\): signal in frequency domain

• Demo applet: www.falstad.com/fourier
Polynomial multiplication via FFT

- Task: given two polynomials $A(x), B(x)$ of degree $n-1$, compute $C(x) = A(x) * B(x)$
- How can we do that fast using (inverse) DFT?
- Algorithm:
  1. Compute $A(x_0), ..., A(x_{2n-1})$ and $B(x_0), ..., B(x_{2n-1})$
  2. For $i=0 \ldots 2n-1$, compute $C(x_i) = A(x_i) * B(x_i)$
  3. Interpolate $C(x)$ from $C(x_0), ..., C(x_{2n-1})$

- Correctness: why it is sufficient to evaluate at $2n-1$ points?
  - $C(x)$ has degree at most $2n-2$
- The number of operations:
  - Steps 1, 3: $O(n \log n)$, thanks to FFT
  - Step 2: $O(n)$
  - Total: $O(n \log n)$
Discrete Fourier Transform

- Goal: evaluate $A(x_0), A(x_1), \ldots, A(x_{n-1})$, where $x_j = \omega^n_j$, $\omega_n = e^{2\pi i/n} = \cos(2\pi/n) + i\sin(2\pi/n)$
- A different interpretation:
  - Want to compute $y_j = A(x_j) = \sum_k a_k \omega_n^{jk}$
  - Can represent this as a matrix-vector product: $y = V*a$

$$
\begin{bmatrix}
y_0 \\
y_1 \\
\vdots \\
y_{n-1}
\end{bmatrix} =
\begin{bmatrix}
\omega_n^0 & \omega_n^0 & \cdots & \omega_n^0 \\
\omega_n^0 & \omega_n^1 & \cdots & \omega_n^{n-1} \\
\vdots & \vdots & \ddots & \vdots \\
\omega_n^0 & \omega_n^{n-1} & \cdots & \omega_n^{(n-1)(n-1)}
\end{bmatrix}
\begin{bmatrix}
a_0 \\
a_1 \\
\vdots \\
a_{n-1}
\end{bmatrix}
$$

- $V$ is called a Vandermonde or Fourier matrix
To be continued....