Fast Fourier Transform (FFT)

Applications:
- Discrete Fourier Transform (DFT)
  \[ \rightarrow \text{signal processing: speech, images, look for aliens (seti@home)} \]
- Integer Multiplication (c.f. Lec 2)
- Polynomial Multiplication

Algorithmic Technique: Divide & Conquer
+ complex number magic

Problem Statement?
Let's postpone that and start with an application

Polynomial Multiplication

\[
\begin{align*}
A(z) &= a_0 + a_1 z + \ldots + a_r z^r \\
B(z) &= b_0 + b_1 z + \ldots + b_s z^s \\
A(z) \cdot B(z) &= a_0 \cdot b_0 + (a_0 \cdot b_1 + a_1 \cdot b_0) z + \ldots + a_r \cdot b_s \cdot z^{r+s} \\
&= c_0 + c_1 z + \ldots + c_{r+s} z^{r+s} = C(z)
\end{align*}
\]
- k-th coefficient of the product?

\[ c_k = \sum_{j=0}^{k} a_j \cdot b_{k-j}, \text{ for all } k=0,1, \ldots, r+s \]

where for the purposes of this equation we assume that \( a_{r+1} = a_{r+2} = \ldots = a_{r+s} = 0 \) and \( b_{s+1} = b_{s+2} = \ldots = b_{r+s} = 0 \)

- Operation rings a bell?

\((c_0, c_1, \ldots, c_{r+s})\) is the convolution of \((a_0, \ldots, a_r)\) with \((b_0, \ldots, b_s)\)

\[ \text{notation: } (c_0, \ldots, c_{r+s}) = (a_0, \ldots, a_r) \ast (b_0, \ldots, b_s) \]

- How many additions & multiplications needed?

a.k.a. arithmetic operations

\((r+1) \cdot (s+1)\) multiplications (as every \(a_i\) will be multiplied w/ every \(b_j\))

\((r+1) \cdot (s+1) - (r+s+2)\) additions
- So if \( r = O(n), \ s = O(n), \) then \( O(n^2) \) arithmetic ops.

- A better (?) approach:
  - Recall that a polynomial of degree \( d \) is uniquely determined by its values at any \( d+1 \) points.
  - Strategy: evaluate \( A \) and \( B \) at some points \( x_0, x_1, \ldots, x_{r+s} \)
    - then \( C(x_k) = A(x_k) \cdot B(x_k) \) for all \( k = 0, \ldots, r+s \)
    - do polynomial interpolation to recover the coefficients \( c_0, \ldots, c_{r+s} \)
      from the values \( C(x_0), \ldots, C(x_{r+s}) \).

  - Naively evaluating a degree \( O(n) \) polynomial at \( O(n) \) points requires \( O(n^2) \) operations.
  - so vanilla evaluate + interpolate approach fails.

💡 Let's pick \( x_0, x_1, \ldots, x_{r+s} \) cleverly so that \( O(n) \) simultaneous evaluations are as expensive as a single evaluation!
- Complex number magic

(Preamble to clever choice of $x_0, \ldots, x_{N-1}$)

1. complex numbers: $x + iy$, where $i = \sqrt{-1}$

- in polar coordinates $r \cdot e^{i\theta}$, where $r = \sqrt{x^2 + y^2}$
  \[ \theta = \tan^{-1} \frac{y}{x} \]

(recall $e^{i\theta} = \cos \theta + i \sin \theta$)

- complex number multiplication
  \[ (r_1 e^{i\theta_1}) \cdot (r_2 e^{i\theta_2}) = r_1 r_2 \cdot e^{i(\theta_1 + \theta_2)} \]

2. roots of unity

$w$ is a $N$-th root of unity if $w^N = 1$

exactly $N$ of them $1, e^{i \frac{2\pi}{N}}, e^{i \frac{4\pi}{N}}, \ldots, e^{i \frac{2\pi(N-1)}{N}}$
e.g. $N=8$

- Back to evaluation/interpolation idea

  - first pick $N$ to be a sufficiently large power of 2 so that $r < N$

  - evaluate $A(.)$ at $1, w, w^2, \ldots, w^{N-1}$, where $w = e^{i \frac{2 \pi}{N}}$

  \[ A(w^j) = \sum_{k=0}^{N-1} a_k \cdot w^{k \cdot j} \]

  - note: $A(z) = \underbrace{A_{\text{even}}(z^2)}_{\text{even}} + \underbrace{z \cdot (A_{\text{odd}}(z^2))}_{\text{odd}}$

  where $A_{\text{even}}(z) = a_0 + a_2 z + a_4 z^2 + \ldots + a_{N-2} z^{\frac{N-2}{2}}$

  $A_{\text{odd}}(z) = a_1 + a_3 z + a_5 z^2 + \ldots + a_{N-1} z^{\frac{N-2}{2}}$

the eight $8$-th roots of unity are shown on the left
So $A(w^j) = A_{\text{even}}(w^j) + w^j A_{\text{odd}}(w^2j)$, for $j = 0, 1, \ldots, N-1$.

**Magic:** $\{w^{2j} \mid j \in \{0, 1, \ldots, N-1\}\} \equiv \{1, w^2, w^4, \ldots, w^\frac{N-1}{2}\}$ because $w = e^{\frac{2\pi i}{N}}$.

Hence if we have $A_{\text{even}}(1), A_{\text{even}}(w^2), \ldots, A_{\text{even}}(w^\frac{N-1}{2})$ and $A_{\text{odd}}(1), A_{\text{odd}}(w^2), \ldots, A_{\text{odd}}(w^\frac{N-1}{2})$, we immediately get $A(1), A(w), \ldots, A(w^{N-1})$.

To evaluate a degree $N-1$ polynomial on $1, w, w^2, \ldots, w^{N-1}$ (the $N$-th roots of unity), it suffices instead to evaluate 2 degree $\frac{N-1}{2}$ polynomials at $1, w^2, w^4, \ldots, w^{\frac{N-1}{2}}$ (the $\frac{N}{2}$-th roots of unity) and additional $N$ additions & $N$ multiplications.

Ready for D&C?

$T(N)$: # arithmetic operations to evaluate polynomial of degree $N-1$ at the $N$-th roots of unity.

$T(N) = 2T\left(\frac{N}{2}\right) + 2N$, $T(1) = 0$.

$\Rightarrow T(N) = O(N \cdot \log N)$. 
Done? A: partially - we're done w/ evaluation step

- can evaluate $A(z)$ and $B(z)$ on $1, w, w^2, \ldots, w^{N-1}$ in $O(N \cdot \log N)$ operations

$\Rightarrow$ we've got $C(z)$ on $1, w, \ldots, w^{N-1}$

w/ another $O(N)$ operations

- now need to interpolate, i.e. get the coefficients $c_0, \ldots, c_{N-1}$ of $C(z)$ from its values $C(1), C(w), \ldots, C(w^{N-1})$

- how are these related?

\[
\begin{array}{cccccc}
1 & 1 & 1 & \cdots & 1 & 1 \\
1 & w & w^2 & \cdots & w^{N-1} & w \\
1 & w^2 & w^4 & \cdots & w^{2(N-1)} & w^2 \\
1 & w^3 & w^6 & \cdots & w^{3(N-1)} & w^3 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
1 & w^{N-1} & w^{2(N-1)} & \cdots & w^{(N-1)^2} & w^{(N-1)^2} \\
\end{array}
\]

\[
\begin{array}{c}
c_0 \\
c_1 \\
c_2 \\
c_3 \\
\vdots \\
c_{N-1} \\
\end{array}
\]

\[
\begin{array}{c}
C(1) \\
C(w) \\
C(w^2) \\
C(w^3) \\
\vdots \\
C(w^{N-1}) \\
\end{array}
\]

\[\text{let's call this } M \]

\[\text{want this} \]

\[\text{know this} \]
\[ M = (w_{ij})_{i,j \in \{0, \ldots, N-1\}} \]

If \( M \) invertible then:

\[
\begin{pmatrix}
  c_0 \\
  c_1 \\
  \vdots \\
  c_{N-1}
\end{pmatrix} = M^{-1} \cdot \begin{pmatrix}
  C(1) \\
  C(w) \\
  \vdots \\
  C(w^{N-1})
\end{pmatrix}
\]  

(1)

Let us find \( M^{-1} \) (Magic 2)

guess \( M^{-1} = \frac{1}{N} (w^{-i:j}) \)

Claim: \((w^{i:j}) \cdot (w^{-i:j}) = \binom{N}{N} \binom{N}{0} \)

Proof: The \((i:j)\)-th entry of the product is

\[
\lambda_{ij} = \sum_{k=0}^{N-1} w^{ik} w^{-rj} = \sum_{k=0}^{N-1} (w^{-j})^k
\]

\[
\text{if } i=j, \quad \lambda_{ii} = N
\]
• If \( i \neq j \) : 
  \[ \lambda_{ij} = \begin{cases} 
    \frac{e^{-j}}{w^{i-j} - 1} & \text{if } i < j \\
    \frac{e^{j}}{w^{i-j} - 1} & \text{if } i > j 
  \end{cases} \]

Now that we have \( M^{-1} \), let’s go back to (1):

\[
\begin{pmatrix}
  c_0 \\
  c_1 \\
  \vdots \\
  c_{N-1}
\end{pmatrix} = \frac{1}{N} \begin{pmatrix}
  (w^{-i}) \\
  (w^{-i})^2 \\
  \vdots \\
  (w^{-i})^{N-1}
\end{pmatrix} \cdot \begin{pmatrix}
  c(w) \\
  c(w^2) \\
  \vdots \\
  c(w^{N-1})
\end{pmatrix}
\]

How long will this matrix-vector product take?

We have established that (using D&C) we can multiply a vector of length \( N \) with the \( N\times N \) matrix \((w^{-i,j})\) in \( O(N \log N) \) arithmetic operations.

The same D&C strategy finds the product of an arbitrary vector of length \( N \) with the \( N\times N \) matrix \( \frac{1}{N} (w^{-i,j}) \).
Summary:

1. **INPUT:** polynomials
   \[ A(z) = a_0 + a_1 z + \ldots + a_r z^r \]
   \[ B(z) = b_0 + b_1 z + \ldots + b_s z^s \]
   where \( r + s < N \), \( N \) is a power of 2

2. **evaluate** \( A(\cdot) \) on the \( N \)-th roots of unity
   \[ B(\cdot) \] on \(-1\)\(\rightarrow\)

\( O(N \cdot \log N) \) \[
\begin{bmatrix}
\text{amounts to multiplying matrix}
\end{bmatrix}
\begin{bmatrix}
w^{i:j} \end{bmatrix}
w/ \text{vectors}
\begin{bmatrix}
a_0, \ldots, a_r, 0, 0, \ldots, 0
\end{bmatrix}
\begin{bmatrix}
b_0, \ldots, b_s, 0, 0, \ldots, 0
\end{bmatrix}
\end{bmatrix}
\]

\( \rightarrow \)

\( O(N) \) \[ C(1) = A(1) \cdot B(1), C(w) = A(w) \cdot B(w), \ldots, C(w^{N-1}) = A(w^{N-1}) \cdot B(w^{N-1}) \]

\( \rightarrow \)

4. **Interpolate** to find coefficients of \( A(z) \cdot B(z) \)

\( O(N \cdot \log N) \) \[
\begin{bmatrix}
\text{amounts to multiplying matrix} \frac{1}{N} \cdot (w^{i:j})
\end{bmatrix}
\begin{bmatrix}
\text{with} \quad (C(1), C(w), \ldots, C(w^{N-1}))
\end{bmatrix}
\]
Discrete Fourier Transform (DFT) vs Fast Fourier Transform (FFT)

def: DFT is a transformation, in particular:
   For a vector $x$ of length $N$

   \[ \text{DFT}(x) = (w^{-i}) \cdot x \], where $w = e^{-i \frac{2\pi}{N}}$.

Note: this $i$ is an index, while that $i = \sqrt{-1}$

-def: FFT is an algorithm for computing DFT which takes $O(N \cdot \log N)$ operations, such as the one we derived today.
The meaning of DFT

gedanken experiment:

- Suppose \( F(t) \) is a real-valued function defined on integers \( t = 0, 1, 2, \ldots \) which has period \( N = 2^k \) for some \( k \).

- Find \( c_0, c_1, \ldots, c_{N-1} \) such that:

\[
F(t) = \sum_{j=0}^{N-1} c_j w^{jt}, \quad \text{where} \quad w = e^{\frac{i 2 \pi}{N}}
\]

for all \( t = 0, 1, \ldots, N-1 \).

How? Need:

\[
\begin{pmatrix}
F(0) \\
F(1) \\
\vdots \\
F(N-1)
\end{pmatrix} = \begin{pmatrix}
w^0 & \cdots & w^{N-1} \\
\vdots & \ddots & \vdots \\
w^{N-1} & \cdots & w^{2(N-1)}
\end{pmatrix}
\begin{pmatrix}
c_0 \\
c_1 \\
\vdots \\
c_{N-1}
\end{pmatrix}
\]

\[
\Rightarrow \begin{pmatrix}
c_0 \\
c_1 \\
\vdots \\
c_{N-1}
\end{pmatrix} = \frac{1}{N} \begin{pmatrix}
w^{-0} & \cdots & w^{-(N-1)} \\
\vdots & \ddots & \vdots \\
w^{-(N-1)} & \cdots & w^{-(2(N-1))}
\end{pmatrix}
\begin{pmatrix}
F(0) \\
F(1) \\
\vdots \\
F(N-1)
\end{pmatrix}
\]
Now take real parts on both sides of (2) and supposing that $c_j = r_j e^{i\theta_j}$

\[
F(t) = \sum_{j=0}^{N-1} \Re \left[ r_j e^{i\theta_j} e^{2\pi \frac{jt}{N}} \right]
\]

\[
= \sum_{j=0}^{N-1} \Re \left[ r_j e^{i \left( 2\pi \frac{jt}{N} + \theta_j \right)} \right]
\]

\[
= \sum_{j=0}^{N-1} r_j \cos \left( 2\pi \frac{t}{N} j + \theta_j \right), \quad \forall \ t=0,1,\ldots,N-1
\]

\[
\Rightarrow \quad \text{period} \quad \frac{N}{j}, \quad \text{phase} \quad \theta_j
\]

\[
\Rightarrow \quad \text{Every periodic real-valued function can be written as a superposition of cosines.}
\]

and in fact for all $t$ as $F(t) = F(t+N)$ for all $t$ and the same is true of all cosines used in superposition.
FFT History

- 1963 Cooley + Tukey
  - Princeton math
  - IBM programmer

  Tukey explains to IBM’s Dick Carwin
  Cooley implements

- Cooley wants to publish to avoid patent
- Tukey thinks it must be known + too easy

- Indeed:
  - 1800 Gauss on interpolation of orbits of celestial bodies