Notes on Recursive FFT (Fast Fourier Transform) algorithm
Fall 2010, COSC 511
Susan Haynes

(1) The Fourier Transform transforms a \(|a| = n\) vector in spatial or time domain to a vector in frequency domain.

(2) The Fourier Transform is invertible.

(3) Convolution (e.g., polynomial multiplication) is \(O(n^2)\).

(4) Convolution in spatial domain is the same as pair-wise multiplication in the frequency domain.

(5) Fourier transform of a vector \(a\) can be performed by a matrix - vector multiply, where the matrix encodes the Fourier transform and the vector is \(a\). Matrix - vector multiply is \(O(n^2)\).

(6) The FFT uses a particular matrix, \(F\), where each element is one of the \(n\)-th roots of 1 (unity). The element of the matrix at row \(i\), column \(j\) (starting at \((0,0)\)) is \(\omega_{n}^{ij}\). To compute \(FFT(a)\) using the matrix \(F\) and vector \(a\) is \(O(n^2)\) (see comment 5). \(FFT^{-1}\) element at \((i, j)\) is \((1/n) \omega_{n}^{-ij}\)

(7) \(\omega_{n}^{i}\) has some nice properties:
   - \(\omega_{n}^{i} * \omega_{n}^{j}\) computes to one of the \(n\) complex \(n\)-th roots.
   - \(\omega_{n}^{i+n} = \omega_{n}^{i}\) adding any multiple of \(n\) to the exponent, gives the original \(\omega_{n}^{i}\).
   - \(\omega_{n}^{ik} = \omega_{n}^{k}\) (e.g., \(\omega_{4}^{6} = \omega_{4}^{3}\)) AKA cancellation property
   - If \(n = 0\) is even, then the squares of the \(n\)th-roots of unity are the \(n/2\) complex \((n/2)\)-th roots of unity. E.g., Consider the 4 fourth roots of unity: \(\omega_{4}^{1}, \omega_{4}^{2}, \omega_{4}^{3}, \omega_{4}^{4}\). Square them: \(\omega_{4}^{2}, \omega_{4}^{4}, \omega_{4}^{2}, \omega_{4}^{4}\). The previous cancellation property. The squared roots are \(\omega_{2}^{1}, \omega_{2}^{0}, \omega_{2}^{1}, \omega_{2}^{0}\). By the mod property, those squared roots are \(\omega_{2}^{1}, \omega_{2}^{0}, \omega_{2}^{1}, \omega_{2}^{0}\) which are exactly the two \(2^{2}\) roots of unity.

(8) The FFT can be implemented as a divide-and-conquer (hence a recursive) algorithm, giving \(O(n \log n)\).
The analysis is similar to the analysis for merge-sort.

(9) Convolution \(p1 \otimes p2 = q\), using FFT is \(O(n \log n)\) (rather than \(O(n^2)\), see comment 3) because:
   \[ P1 = FFT(p1) \text{ is } O(n \log n) \]
   \[ P2 = FFT(p2) \text{ is } O(n \log n) \]
   \[ Q = P1 * P2 \text{ (pairwise) is } O(n) \]
   \[ q = FFT^{-1}(Q) \text{ is } O(n \log n) \]
(10) The pseudocode for FFT (recursive version) from Cormen, Leisserson, et al, is:

```plaintext
FFT (a)   { // 1D array a = (a0, a1, ..., an-1)
1    n = length (a) // n is a power of 2
2.  if (n == 1)
3.      return a
4.  \omega_n = e^{2\pi i/n} // \omega^n_1
5.  \omega = 1
6.  a[0] = (a0, a2, a4, ... a_{n-2} )
7.  a[1] = (a1, a3, a5, ... a_{n-1} )
8.  y[0] = FFT(a[0])
9.  y[1] = FFT(a[1])
10.  for (k=0; k <= n/2-1; k++) {
11.      y_k = y_k[0] + \omega y_k[1]
12.      y_{k+(n/2)} = y_k[0] - \omega y_k[1]
13.      \omega = \omega \omega_n
14.  }
15.  return y // y is vector, |y| = n
}
```

(11) Unit circle with roots, FFT, and FFT^{-1} are given on the following pages.
\( n = 1 \)

\[ \omega_i = \omega_i' = e^{i \frac{2\pi}{2}} = 1 \]

**FFT:**
\[
\begin{bmatrix}
\omega^0 & \omega^1
\end{bmatrix}
\begin{bmatrix}
\omega^0
\end{bmatrix}
= \begin{bmatrix}1\end{bmatrix}
\]

**FFT\(^{-1}\):**
\[
\begin{bmatrix}
\omega^0 & \omega^{-1}
\end{bmatrix}
\begin{bmatrix}
\omega^0
\end{bmatrix}
= \begin{bmatrix}1\end{bmatrix}
\]

\( n = 2 \)

\[ \omega_i = \omega_i' = e^{i \frac{2\pi}{4}} \]

\[ \omega^0 = +1 \]

\[ \omega^1 = -1 \]

**FFT:**
\[
\begin{bmatrix}
\omega^0 & \omega^1 & \omega^0 & \omega^1
\end{bmatrix}
\begin{bmatrix}
\omega^0 & \omega^1
\end{bmatrix}
= \begin{bmatrix}1 & 1
1 & -1
\end{bmatrix}
\]

**FFT\(^{-1}\):**
\[
\begin{bmatrix}
\omega^0 & \omega^1 & \omega^0 & \omega^1
\end{bmatrix}
\begin{bmatrix}
\omega^0 & \omega^1
\end{bmatrix}
= \begin{bmatrix}1 & 1
1 & -1
\end{bmatrix}
\]
\( n = 4 \)

\[
\begin{align*}
\omega_1 &= e^{\frac{i \pi}{2}} = i \\
\omega_2 &= e^{i \pi} = -1 \\
\omega_3 &= e^{\frac{3i \pi}{2}} = -i \\
\omega_4 &= \omega^3 = 1
\end{align*}
\]

\[
\text{FFT} = \begin{bmatrix}
\omega^0 & \omega^1 & \omega^2 & \omega^3 \\
\omega^1 & \omega^0 & \omega^3 & \omega^2 \\
\omega^2 & \omega^3 & \omega^0 & \omega^1 \\
\omega^3 & \omega^2 & \omega^1 & \omega^0
\end{bmatrix}
\]

\[
= \begin{bmatrix}
\omega^0 & \omega^0 & \omega^0 & \omega^0 \\
\omega^0 & \omega^1 & \omega^2 & \omega^3 \\
\omega^0 & \omega^2 & \omega^3 & \omega^1 \\
\omega^0 & \omega^3 & \omega^1 & \omega^2
\end{bmatrix}
\]

\[
\text{FFT}^{-1} = \frac{1}{4} \begin{bmatrix}
\omega^0 & \omega^0 & \omega^0 & \omega^0 \\
\omega^0 & \omega^{-1} & \omega^{-2} & \omega^{-3} \\
\omega^0 & \omega^{-2} & \omega^{-3} & \omega^{-1} \\
\omega^0 & \omega^{-3} & \omega^{-1} & \omega^{-2}
\end{bmatrix}
\]
\[ n = 8 \]

\[ w^1 = e^{\frac{2\pi}{8}} = \frac{1}{\sqrt{2}} + \frac{i}{\sqrt{2}} = \frac{1+i}{\sqrt{2}} \]
\[ w^2 = e^{\frac{4\pi}{8}} = i \]
\[ w^3 = e^{\frac{6\pi}{8}} = -\frac{1}{\sqrt{2}} + \frac{i}{\sqrt{2}} = -\frac{1+i}{\sqrt{2}} \]
\[ w^4 = e^{\frac{8\pi}{8}} = -1 \]
\[ w^5 = e^{\frac{10\pi}{8}} = -\frac{1}{\sqrt{2}} - \frac{i}{\sqrt{2}} = -\frac{1-i}{\sqrt{2}} \]
\[ w^6 = e^{\frac{12\pi}{8}} = -i \]
\[ w^7 = e^{\frac{14\pi}{8}} = \frac{1}{\sqrt{2}} - \frac{i}{\sqrt{2}} = \frac{1-i}{\sqrt{2}} \]
\[ w^8 = e^{\frac{16\pi}{8}} = 1 \]

*Calculate the rectangular representation for \( w^1 \), then use symmetry for \( w^3, w^5, w^7 \)*

**Pythagorean theorem says**
\[ r^2 = x^2 + y^2 \]

For \( w^1 \)
\[ \frac{1}{r} = x^2 + y^2 \]
\[ x = \sqrt{\frac{1}{2}} \]
\[ y = \sqrt{\frac{1}{2}} \]