Lecture notes on the FFT, 15-451, Fall 2000

Daniel Sleator

These notes describe my way of deriving the FFT algorithm. Read these notes in conjunction with chapter 9.6 of Manber.

Let N be the length of the vectors we're working with Let ω_N be a principle Nth root of unity. (If we omit the N and write ω, then we mean ω_N.)

Let P be a polynomial of degree $N-1$:

$$P(x) = a_0 + a_1 x + a_2 x_2 + \cdots + a_{N-1} x^{N-1}$$

Define the DFT (Discrete Fourier Transform) of the vector $(a_0, \ldots, a_{N-1})$ to be another vector of N numbers as follows:

$$F_N(a_0, a_1, \ldots, a_{N-1}) = (P(\omega^0), P(\omega^1), \ldots, P(\omega^{N-1}))$$

(This can be expressed as a matrix multiplication.)

Now we can derive the fast algorithm for coumputing the DFT. (This is called the FFT algorithm.) Let A be the vector $(a_0, a_1, \ldots, a_{N-1})$. Let $F_N(A)_j$ denote the jth component of the DFT of the vector A.

$$\begin{aligned}
F_N(A)_j &= P(\omega^j) \\
&= \sum_{i=0}^{N-1} a_i \omega^{ij} \\
&= \sum_{0 \le i \le N-1,\, i \text{ even}} a_i \omega^{ij} + \sum_{0 \le i \le N-1,\, i \text{ odd}} a_i \omega^{ij} \\
&= \sum_{i=0}^{\frac{N}{2}-1} a_{2i} \omega^{2ij} + \sum_{i=0}^{\frac{N}{2}-1} a_{2i+1} \omega^{(2i+1)j} \\
&= \sum_{i=0}^{\frac{N}{2}-1} a_{2i} (\omega^2)^{ij} + \omega^j \sum_{i=0}^{\frac{N}{2}-1} a_{2i+1} (\omega^2)^{ij} \\
&= \sum_{i=0}^{\frac{N}{2}-1} a_{2i} (\omega_{N/2})^{i(j \bmod N/2)} + \omega_N^j \sum_{i=0}^{\frac{N}{2}-1} a_{2i+1} (\omega_{N/2})^{i(j \bmod N/2)}
\end{aligned}$$

In the last step we've simply used the fact that $\omega_N^2 = \omega_{N/2}$, and the observation that $\omega_{N/2}^{ij} = (\omega_{N/2}^j)^i = (\omega_{N/2}^{j \bmod N/2})^i$ because $\omega_{N/2}^j$ is a periodic funciton of j with period $N/2$.

Now the key point is that these two summations are in fact just DFTs half the size. Let's let A_{even} denote the vector of as with even subsubscripts (of length $N/2$) and A_{odd} be the odd ones. We can write the final equation above using these vectors as follows:

$$F_N(A)_j = F_{N/2}(A_{\text{even}})_{j \bmod N/2} + \omega_N^j F_{N/2}(A_{\text{odd}})_{j \bmod N/2}$$

This immediately gives an $O(N \log N)$ algorithm (for N a power of 2). To compute the vector $F_N(A)$ we extract the even subarray of A, and the odd subarray of A, and compute the DFT of each of these recursively. Then we put the two parts together in $O(N)$ time. The recurrence is then:

$$T(N) = 2T(N/2) + N$$

which solves to $O(N \log N)$.