CSC721 Algorithms Fall 2017

Recursive FFT

Consider the powers of a principal n^{th} root of unity, ω:

$\mathbf{x}$	ω^0	ω^1	ω^2	ω^3	$\cdots$	$\omega^{n/2-1}$	$\omega^{n/2}$	$\omega^{n/2+1}$	$\omega^{n/2+2}$	$\cdots$	ω^{n-1}
$\mathbf{x}^2$	$(\omega^0)^2$	$(\omega^1)^2$	$(\omega^2)^2$	$(\omega^3)^2$	$\cdots$	$(\omega^{n/2-1})^2$	$(\omega^{n/2})^2$	$(\omega^{n/2+1})^2$	$(\omega^{n/2+2})^2$	$\cdots$	$(\omega^{n-1})^2$

The table above shows the values of the powers of the principal root of unity ω, and the values squared. If we simplify the values squared, we get:

$\mathbf{x}^2$	ω^0	ω^2	ω^4	ω^6	$\cdots$	ω^{n-2}	ω^n	ω^{n+2}	ω^{n+4}	$\cdots$	ω^{2n-2}

But remember that $\omega^n = 1$, so whenever I have something of the form ω^{n+k}, I can simplify. I.e.,

$$\omega^{n+k} \;=\; \omega^n\omega^k \;=\; 1\omega^k \;=\; \omega^k$$

We can re-write the table above as:

$\mathbf{x}^2$	ω^0	ω^2	ω^4	ω^6	$\cdots$	ω^{n-2}	ω^0	ω^2	ω^4	$\cdots$	ω^{n-2}

If we look at the values in the table above, we notice that the numbers $\omega^0, \omega^2, ..., \omega^{n-2}$ occur twice. Observe that the numbers $\omega^0, \omega^2, ..., \omega^{n-2}$ can be interpreted as the $n/2$ distinct powers of a **principal $(n/2)^{\text{th}}$ root of unity**.

When using the formula

$$p(x) = p_{\text{even}}(x^2) + xp_{\text{odd}}(x^2)$$

We only have $n/2$ distinct values of x^2 at which to evaluate the polynomials p_{even} and p_{odd}.
So suppose I am trying to compute $p(\omega^j)$ where $0 \le j < n/2$.

$$p(\omega^j) = p_{\text{even}}(\omega^{2j}) + \omega^j p_{\text{odd}}(\omega^{2j}) \tag{1}$$

Thus we have the psudo-code:

```
for ( j = 0 ; j < n/2 ; j++ ) {
     xpodd = omega[ j ] * odds[ j ] ;
     b[j] = evens[j] + xpodd ;
}
```

Our goal is to construct a recursive algorithm. At the top level of the recursion, we use the powers of an n^{th} principal root of unity, i.e., $\omega = e^{-2\pi\imath/n}$. We pre-compute all the powers $\omega^0, \omega^1, ..., \omega^{n-1}$ and store them in an array (conveniently) named `omega[]`.

At the next lower recursive level the size of our sub-problem is half as large and we use powers of an $(n/2)^{\text{th}}$ principal root of unity. Instead of re-computing the powers of the $(n/2)^{\text{th}}$ principal

roots of unity, we can simply index the even positions in the array `omega[]`, i.e, count by two as we scan through the array.

The pattern of counting by two, then by four, then by eight, etc., continues as we descend each level of recursion. We use a book-keeping variable `m` to keep track of the stride.

Thus, our up-dated psudo-code is:

```
for ( j = 0 ; j < n/2 ; j++ ) {
     xpodd = omega[ m * j ] * odds[ j ] ;
     b[j] = evens[j] + xpodd ;
}
```

Let us consider evaluating $p(x)$ for the rest of the powers of the principal root of unity. I.e., for

$$x \in \{\ \omega^{n/2}, \omega^{n/2+1}, \omega^{n/2+2}, ..., \omega^{n-1}\ \}$$

I can write each of these values of x in the form: $\omega^{j+n/2}$ where $0 \leq j < n/2$.

Following equation (1), we have:

$$\begin{aligned} p(\omega^{j+n/2}) &= p_{\text{even}}(\omega^{2(j+n/2)}) + \omega^{j+n/2} p_{\text{odd}}(\omega^{2(j+n/2)}) && (2)\\ &= p_{\text{even}}(\omega^{2j+n}) + \omega^{n/2}\omega^{j} p_{\text{odd}}(\omega^{2j+n}) && (3)\\ &= p_{\text{even}}(\omega^{2j}) + \omega^{n/2}\omega^{j} p_{\text{odd}}(\omega^{2j}) && (4)\\ &= p_{\text{even}}(\omega^{2j}) - \omega^{j} p_{\text{odd}}(\omega^{2j}) && (5) \end{aligned}$$

Equation (5) justifies the pseudo-code:

```
for ( j = 0 ; j < n/2 ; j++ ) {
     xpodd = omega[ m * j ] * odds[ j ] ;
     b[j] = evens[j] + xpodd ;
     b[j+n/2] = evens[j] - xpodd ;
}
```

Finally, our algorithm is;

```
fft( a, n, m, omega ; b )
{
    if ( n == 1 )  b[0] = a[0]  ;
    else {
        fft( [ a_0, a_2, a_4, ... a_{n-2} ], n/2, 2*m, omega ; evens ) ;
        fft( [ a_1, a_3, a_5, ... a_{n-1} ], n/2, 2*m, omega ; odds ) ;
        for ( j = 0 ; j < n/2 ; j++ ) {
            xpodd = omega[ m * j ] * odds[ j ] ;
            b[j] = evens[j] + xpodd ;
            b[j+n/2] = evens[j] - xpodd ;
        }  // end for
    }  // end else
} // end fft
```