Schönhage–Strassen algorithm

Vanshika Jain

March 15, 2016

1 Multiplying

1.1 The Old Fashioned Way

When we multiply numbers normally, we have to do m^2 multiplications, which is inefficient. For example, suppose $p = 4358$ and $q = 6734$

$$\begin{array}{cccccccc}
 & & & & 4 & 3 & 5 & 8 \\
 & & & \times & 6 & 7 & 3 & 4 \\ \hline
 & & & 1 & 7 & 4 & 3 & 2 \\
 & & 1 & 3 & 0 & 7 & 4 & \\
 & 3 & 0 & 5 & 0 & 6 & & \\
2 & 6 & 1 & 4 & 8 & & & \\ \hline
2 & 9 & 3 & 4 & 6 & 7 & 7 & 2
\end{array}$$

1.2 Multiplying with Polynomials

In order to make this more efficient, we are going to look at multiplying numbers like multiplying polynomials. We can rewrite the multiplication we did before as

$$p * q = (4 * 10^3 + 3 * 10^2 + 5 * 10^1 + 8) * (6 * 10^3 + 7 * 10^2 + 3 * 10^1 + 4)$$

This looks awfully similar to a polynomial. $P(x) = 8 + 5x^1 + 3x^2 + 4x^3$ and $Q(x) = 4 + 3x^1 + 7x^2 + 6x^3$ and if we evaluate the two polynomials at 10 then we get the numbers we began with. If we multiply the two polynomials, we would get $R(x) = r_0 + r_1x^1 + r_2x^2 + r_3x^3 + r_4x^4 + r_5x^5 + r_6x^6 + r_7x^7$, and if we evalutated this polynomial at 10 then we would get the product. If we found the coefficients to this polynomial we would save no steps.

Now we must begin our search for a faster way to determine the coefficients: the coefficients of a degree $m - 1$ polynomial are determined by the polynomial's values at m points. We will assume that r has m digits and that p and q have $m/2$. Thus the degree of $R(x)$ is $m - 1$

2 Schönhage–Strassen algorithm

2.1 Outline

1. First, we are going to find the values of P and Q at m points, $x_0, x_1, x_2, ..., x_{m-1}$
2. Then, we will evaluate R at m points by multiplying $P(x_i) * Q(x_i)$.
3. Finally, use the values of $R(x_i)$ to find the coefficients of R

Notice that step two requires only m multiplications. Therefore, we have to figure out how to do steps one and three faster.

2.2 The Fourier Transform

We have the polynomial $A(x)$. We want to evaluate A at m points. Let $\omega = e^{\frac{2*\pi*i}{m}}$, which is the m^{th} root of unity. We can evaluate A at the points $x_s = \omega^s$. This will still require m^2 multiplications. However, there is a fast way to find these coefficients, called the Fourier coefficients, given to us by the Fast Fourier Transform.

FFT provides an efficient means of computing the Fourier coefficients. Lets call the Fourier coefficient $\overline{a_s}$. We are going to break the sum that defines the Fourier coefficient into smaller sums of even and odd values of the the power.

For example,

$$\overline{a_s} = \sum_{t=0}^{m-1} a_t e^{\frac{2\pi i s t}{m}} = \sum_{t=0}^{m/2-1} a_{2t} e^{\frac{2\pi i s(2t)}{m}} + \sum_{t=0}^{m/2-1} a_{2t+1} e^{\frac{2\pi i s(2t+1)}{m}} = \sum_{t=0}^{m/2-1} a_{2t} e^{\frac{2\pi i s t}{m/2}} + \omega^s \sum_{t=0}^{m/2-1} a_{2t+1} e^{\frac{2\pi i s t}{m/2}}$$

In this we can replace s with s' which is the remainder of $\frac{s}{m/2}$ because ω is a root of unity. Instead of having one hard problem, we have made two easier problems. We can continue to do this (remainder 0, 1, 2, 3 when divided by 4 and so on).

That's the main idea, and I'm going to black box the rest. But what we end up getting is that to find the Fourier coefficients is $m\log(m)$

2.3 Convolution

The Fourier coefficients of a polynomial $R(x) = P(x) * Q(x)$ can be obtained by multiplying the Fourier coefficients of $P(x)$ and $Q(x)$. That means that $\overline{r_s} = \overline{p_s} * \overline{q_s}$. We only need m multiplications for this step.

2.4 Finding coefficients of $R(x)$

We find the coefficients of $R(x)$ by using the inverse Fourier transform. If we have $\overline{A}(x) = \overline{a}_0 + \overline{a}_1 x + ... + \overline{a}_{m-1} x^{m-1}$ then we could find a_t by evaluating $\overline{A}(x)$ at $\overline{\omega}^t$ which is the conjugate of ω^t. We can find this quickly by using FFT.

3 Summary

Here we have looked at a simplified version of the algorithm that takes $mln(m)$ time. However, the actual algorithm does better because it takes $mlog(m)log(log(m))$ time. If you would like to see why, there is a wikipedia article that explains it.