MAXimal

home
algo
bookz
forum
about

# Fast Fourier transform for O (N log N). Applicat multiplication of two polynomials or long numb

Here we consider an algorithm that allows us to multiply two polynomials of length $n$ duri $O(n \log n)$ , which is much better than time $O(n^2)$ , achieved by the trivial algorithm of multiplication. Obviously, multiplying two long numbers can be reduced to multiplying pol two long numbers can also be multiplied in time $O(n \log n)$ .

The invention of the Fast Fourier Transformation is attributed to Cooley and Tukey - 196 FFT has been repeatedly invented before, but its importance was not fully realized until t appearance of modern computers. Some researchers attribute the discovery of the FFT Konig in 1924. Finally, the discovery of this method is attributed to Gauss in 1805

## The discrete Fourier transform (DFT)

Let there be a polynomial $n$ degree:

$$A(x) = a_0 x^0 + a_1 x^1 + \ldots + a_{n-1} x^{n-1}.$$

Without loss of generality, we can assume that $n$ is a power of 2. If in fact $n$ is not a pow putting them equal to zero.

It is known from the theory of functions of a complex variable that the complex roots $n$ de by $w_{n,k}, k = 0 \ldots n-1$ , then it is known that $w_{n,k} = e^{i \frac{2\pi k}{n}}$ . In addition, one of thes root $n$ -th power of unity) is such that all other roots are its powers: $w_{n,k} = (w_n)^k$ .

Then the discrete Fourier transform (DFT) of the polynomial $A(x)$ (or, equivalently, the D are the values of this polynomial at the points $x = w_{n,k}$ i.e. this is a vector:

$$\mathrm{DFT}(a_0, a_1, \ldots, a_{n-1}) = (y_0, y_1, \ldots, y_{n-1}) = (A(w_{n,0}), A(w_{n,1}), \ldots, A(w_r$$
$$= (A(w_n^0), A(w_n^1), \ldots, A(w_n^{n-1})).$$

Similarly, the **inverse discrete Fourier transform** (InverseDFT) is defined. The inverse $(y_0, y_1, \ldots y_{n-1})$ Is the vector of the coefficients of the polynomial $(a_0, a_1, \ldots, a_{n-1})$

$$\mathrm{InverseDFT}(y_0, y_1, \ldots, y_{n-1}) = (a_0, a_1, \ldots, a_{n-1}).$$