

# DFT

作用：把一个多项式  $f(x)$  转化成点值表示  $\{(x_0, f(x_0)), (x_1, f(x_1)), \dots, (x_n, f(x_n))\}$  的形式。

推导过程：

记  $\omega_k^n = e^{\frac{2k\pi i}{n}}$

若  $f(x) = \sum_{i=0}^{n-1} a_i x^i$

将  $f(x)$  按奇偶次拆开

$f(x) = (a_0 + a_2 x^2 + \dots + a_{n-2} x^{n-2}) + x(a_1 + a_3 x^2 + \dots + a_{n-1} x^{n-2}) = f_1(x^2) + x f_2(x^2)$

其中

$f_1(x) = a_0 + a_2 x + a_4 x^4 + \dots + a_{n-2} x^{\frac{n}{2}-1}$ ,  $f_2(x) = a_1 + a_3 x + \dots + a_{n-1} x^{\frac{n}{2}-1}$

代入  $\omega_k^n$  得

$f(\omega_k^n) = f_1(\omega_k^n) + \omega_k^n f_2(\omega_k^n) = f_1(\omega_k^n) + \omega_k^{\frac{n}{2}} f_2(\omega_k^{\frac{n}{2}})$

同理  $f(\omega_k^n)^{k+\frac{n}{2}} = f_1(\omega_k^{\frac{n}{2}}) - \omega_k^{\frac{n}{2}} f_2(\omega_k^{\frac{n}{2}})$

到这里，可以对这个函数进行分治了，时间复杂度  $O(n \log n)$

代码：

```
#include<complex>
using namespace std;
const double Pi=acos(-1);
void dft(complex<double> *f,int len)
{
    if (len==1) return ;
    complex<double> *fl=f,*fr=f+len/2;
    for (int k=0;k<len;k++) sav[k]=f[k];
    for (int k=0;k<len/2;k++)
        {fl[k]=sav[k<<1];fr[k]=sav[k<<1|1];}
    dft(fl,len/2);
    dft(fr,len/2);
    complex<double> e(cos(2*Pi/len),sin(2*Pi/len));
    complex<double> i(1,0);
    for (int k=0;k<len/2;k++){
        sav[k]=fl[k]+i*fr[k];//(1)
        sav[k+len/2]=fl[k]-i*fr[k];//(2)
        i=e*i;
    }
    for (int k=0;k<len;k++) f[k]=sav[k];
}
```

From:  
<https://wiki.cvbbacm.com/> - CVBB ACM Team

Permanent link:  
[https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal\\_string:dft](https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal_string:dft) 

Last update: **2020/06/19 21:04**