傅里叶变换标准版

```
#define Maxn 1000005
using namespace std;
const double pi = acos(-1);
int n, m;

struct complex
{
    complex(double real = 0, double imag = 0) :real(real), imag(imag) {}
    double real, imag;
    complex operator + (complex const& B) const { return complex(real + B.real,
imag + B.imag); }
    complex operator - (complex const& B) const { return complex(real - B.real,
imag - B.imag); }
    complex operator * (complex const& B) const { return complex(real * B.real -
imag * B.imag, real * B.imag + imag * B.real); }
};
//注意多项式乘积最长会涨到原来二倍
complex poly1[Maxn << 1], poly2[Maxn << 1], temp[Maxn << 1];

//我很礼貌地没有动这一部分，只进行了一点缩进（因为不会）
void fft(complex* f, int len, bool flag)
{
    if (len == 1)   return;
    complex* fl = f, * fr = f + len / 2;
    for (int k = 0; k < len; k++)
        temp[k] = f[k];
    for (int k = 0; k < len / 2; k++)
    {
        fl[k] = temp[k << 1]; fr[k] = temp[k << 1 | 1];
    }

    fft(fl, len / 2, flag);
    fft(fr, len / 2, flag);

    complex tG(cos(2 * pi / len), sin(2 * pi / len)), buf(1, 0);
    if (!flag)tG.imag *= -1;
    for (int k = 0; k < len / 2; k++)
    {
        temp[k] = fl[k] + buf * fr[k];
        temp[k + len / 2] = fl[k] - buf * fr[k];
        buf = buf * tG;
    }
    for (int k = 0; k < len; k++)   f[k] = temp[k];
}

int main()
{
    cin >> n >> m;
```

```
    for (int i = 0; i <= n; i++)
        scanf("%lf", &poly1[i].real);
    for (int i = 0; i <= m; i++)
        scanf("%lf", &poly2[i].real);
    for (m += n, n = 1; n <= m; n <<= 1);//把长度补到2的幂
    //把多项式从频域模式换成时域模式
    fft(poly1, n, 1);
    fft(poly2, n, 1);
    for (int i = 0; i < n; ++i)
        poly1[i] = poly1[i] * poly2[i];

    fft(poly1, n, 0);
    for (int i = 0; i <= m; ++i)
        printf("%d ", (int)(poly1[i].real / n + 0.49));
}
```