

99

34

原

# 十分简明易懂的FFT（快速傅里叶变换）

2018年08月07日 11:38:56 路人黑的纸巾 阅读量 66464

版权声明：欢迎dalao指出错误暴踩本SD，蒟蒻写的博客转载也要吱一声 [https://blog.csdn.net/enjoy\\_pascal/article/details/81478582](https://blog.csdn.net/enjoy_pascal/article/details/81478582)

## FFT前言

**快速傅里叶变换 (fast Fourier transform)**,即利用计算机计算离散傅里叶变换（DFT）的高效、快速计算方法的统称，简称FFT。快速傅里叶变换是1965年由J.W.库利和出的。采用这种算法能使计算机计算离散傅里叶变换所需要的乘法次数大为减少，特别是被变换的抽样点数N越多，FFT算法计算量的减少就越显著。  
**FFT (Fast Fourier Transformation)** 是离散傅氏变换（DFT）的快速算法。即为快速傅氏变换。它是根据离散傅氏变换的奇、偶、实、虚等特性，对离散傅立叶变换改进获得的。  
——百度百科

**FFT (Fast Fourier Transformation)**，中文名**快速傅里叶变换**，用来**加速多项式乘法**  
朴素高精度乘法时间 $O(n^2)$ ，但**FFT**能 $O(n \log_2 n)$ 的时间解决  
**FFT**名字逼格高，也难怪，其他教程写得让人看不太懂，于是自己随便写一下

- 建议对**复数**、**三角函数**相关知识有所耳闻（不会也无所谓）
- 下面难懂的点我会从网上盗

## 多项式的系数表示法和点值表示法

- **FFT**其实是一个用 $O(n \log_2 n)$ 的时间将一个用**系数表示**的多项式转换成它的**点值表示**的算法
- 多项式的系数表示和点值表示可以**互相转换**

### 系数表示法

一个 **$n-1$ 次 $n$ 项**多项式 **$f(x)$** 可以表示为 **$f(x) = \sum_{i=0}^{n-1} a_i x^i$**   
也可以用**每一项的系数**来表示 **$f(x)$** ，即

$$f(x) = \{a_0, a_1, a_2, \dots, a_{n-1}\}$$

这就是**系数表示法**，也就是平时数学课上用的方法

### 点值表示法

- 把多项式放到平面直角坐标系里面，看成一个**函数**
- 把 **$n$ 个不同的 $x$** 代入，会得出 **$n$ 个不同的 $y$** ，在坐标系内就是 **$n$ 个不同的点**
- 那么这 **$n$ 个点唯一确定**该多项式，也就是**有且仅有一个**多项式满足 **$\forall k, f(x_k) = y_k$**
- 理由很简单，把 **$n$ 条式子联立起来**成为一个**有 **$n$** 条方程的 **$n$** 元方程组**，每一项的系数都可以解出来

那么 **$f(x)$** 还可以用 **$f(x) = \{(x_0, f(x_0)), (x_1, f(x_1)), (x_2, f(x_2)), \dots, (x_{n-1}, f(x_{n-1}))\}$** 来表示  
这就是**点值表示法**

## 高精度乘法下两种多项式表示法的区别

对于两个用**系数表示**的多项式，我们把它们相乘  
设两个多项式分别为 $A(x), B(x)$   
我们要枚举 $A$ 每一位的系数与 $B$ 每一位的系数相乘  
那么系数表示法做多项式乘法**时间复杂度** $O(n^2)$

但两个用**点值表示**的多项式相乘，只需要 $O(n)$ 的时间

什么意思呢？

设两个点值多项式分别为

$$f(x) = \{(x_0, f(x_0)), (x_1, f(x_1)), (x_2, f(x_2)), \dots, (x_{n-1}, f(x_{n-1}))\}$$
$$g(x) = \{(x_0, g(x_0)), (x_1, g(x_1)), (x_2, g(x_2)), \dots, (x_{n-1}, g(x_{n-1}))\}$$

设它们的乘积是 $h(x)$ ，那么

$$h(x) = \{(x_0, f(x_0) \cdot g(x_0)), (x_1, f(x_1) \cdot g(x_1)), \dots, (x_{n-1}, f(x_{n-1}) \cdot g(x_{n-1}))\}$$

所以这里的时间复杂度只有一个枚举的 $O(n)$

- 突然感觉高精度乘法能 $O(n)$ 暴\*\*一堆题？
- 但是**朴素**的系数表示法转点值表示法的算法还是 $O(n^2)$ 的，逆操作类似
- 朴素**系数转点值的算法叫**DFT（离散傅里叶变换）**，点值转系数叫**IDFT（离散傅里叶逆变换）**
- 难道高精度乘法只能 $O(n^2)$ 了吗？

## DFT前置知识&技能

复数

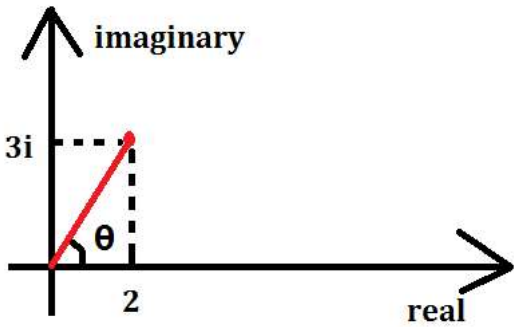
毕竟高中有所以不多说

我们把形如 $a+bi$  ( $a, b$ 均为实数) 的数称为**复数**，其中 $a$ 称为实部， $b$ 称为虚部， $i$ 称为虚数单位。当虚部等于零时，这个复数可以视为实数；当 $z$ 的虚部不等于零时，实部常称 $z$ 为纯虚数。复数域是实数域的代数闭包，也即任何复系数多项式在复数域中总有根。复数是由意大利米兰学者卡当在十六世纪首次引入，经过达朗贝尔、棣莫弗、等人的工作，此概念逐渐为数学家所接受。  
——百度百科

初中数学老师会告诉你没有 $\sqrt{-1}$ ，但仅限 $R$   
扩展至复数集 $C$ ，定义 $i^2 = -1$ ，一个复数 $z$ 可以表示为 $z = a + bi(a, b \in R)$   
其中 $a$ 称为**实部**， $b$ 称为**虚部**， $i$ 称为**虚数单位**

- 在复数集中就可以用 $i$ 表示负数的平方根，如 $\sqrt{-7} = \sqrt{7}i$

还可以把复数看成复平面直角坐标系上的一个点，比如下面



[https://blog.csdn.net/enjoy\\_pascal](https://blog.csdn.net/enjoy_pascal)

$x$ 轴就是实数集中的坐标轴， $y$ 轴就是虚数单位 $i$ 轴

99

34









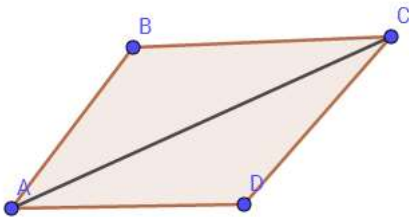
这个点 $(2, 3)$ 表示的复数就是 $2 + 3i$ ，或者想象它代表的向量为 $(2, 3)$   
其实我们还可以自己想象（其实没有这种表达方式）它可以表示为 $(\sqrt{13}, \theta)$   
一个复数 $z$ 的模定义为它到原点的距离，记为 $|z| = \sqrt{a^2 + b^2}$   
一个复数 $z = a + bi$ 的共轭复数为 $a - bi$ （虚部取反），记为 $\bar{z} = a - bi$

### 复数的运算

复数不像点或向量，它和实数一样可以进行四则运算  
设两个复数分别为 $z_1 = a + bi, z_2 = c + di$ ，那么

$$z_1 + z_2 = (a + c) + (b + d)i$$
$$z_1 z_2 = (ac - bd) + (ad + bc)i$$

复数相加也满足平行四边形法则



这张是从网上盗的

即 $AB + AD = AC$

复数相乘还有一个值得注意的小性质

$$(a_1, \theta_1) * (a_2, \theta_2) = (a_1 a_2, \theta_1 + \theta_2)$$

即模长相乘，极角相加

### DFT（离散傅里叶变换）

- 一定注意从这里开始所有的 $n$ 都默认为2的整数次幂

VIP







👍  
99

💬  
34

📖

🔖

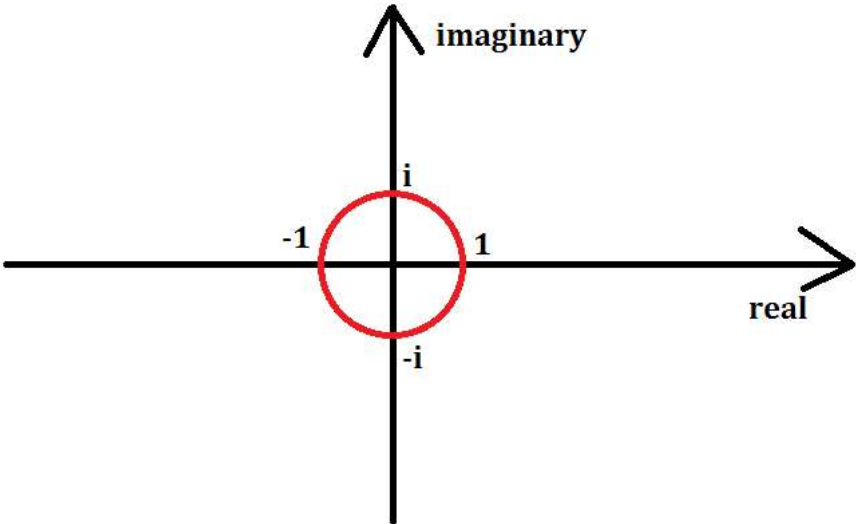
📱

...

对于任意系数多项式转点值，当然可以随便取任意 $n$ 个 $x$ 值代入计算  
但是暴力计算 $x_k^0, x_k^1, \dots, x_k^{n-1} (k \in [0, n))$ 当然是 $O(n^2)$ 的时间  
其实可以代入一组神奇的 $x$ ，代入以后不用做那么多的次方运算  
这些 $x$ 当然不是乱取的，而且取这些 $x$ 值应该就是 傅里叶 的主意了

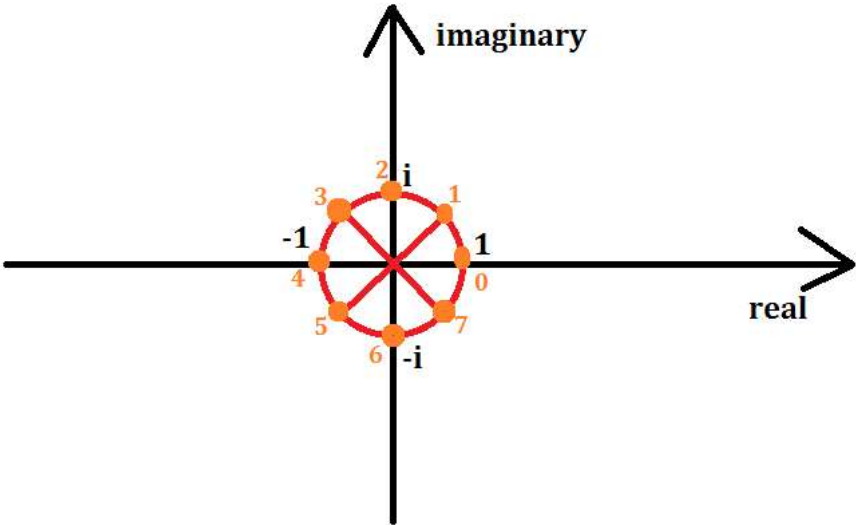
考虑一下，如果我们代入一些 $x$ ，使每个 $x$ 的若干次方等于1，我们就不用做全部的次方运算了  
 $\pm 1$ 是可以的，考虑虚数的话 $\pm i$ 也可以，但只有这四个数远远不够

- 傅里叶说：这个圆圈上面的点都可以做到



[https://blog.csdn.net/enjoy\\_pascal](https://blog.csdn.net/enjoy_pascal)

以原点为圆心，画一个半径为1的单位圆  
那么单位圆上所有的点都可以经过若干次方得到1  
傅里叶说还要把它给 $n$ 等分了，比如此时 $n = 8$



[https://blog.csdn.net/enjoy\\_pascal](https://blog.csdn.net/enjoy_pascal)

Vip

📖

🔖

📱

👍  
99

💬  
34

📖

🔖

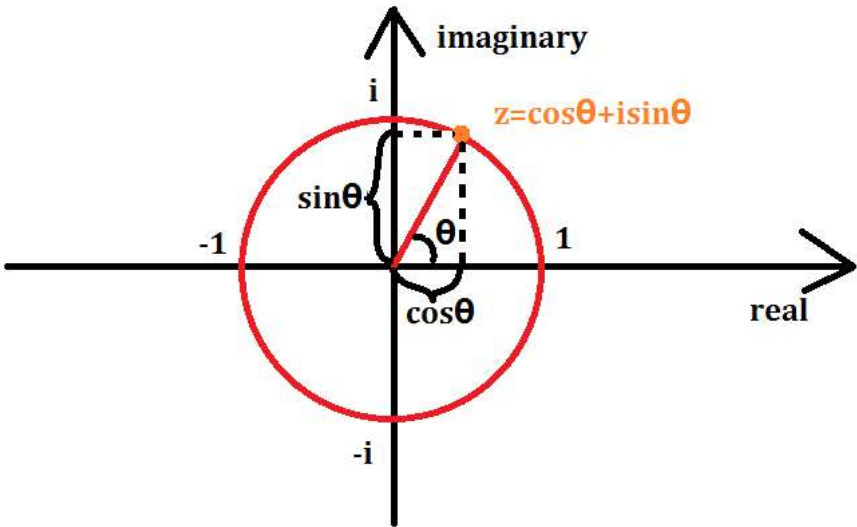
📱

...

橙色点即为 $n = 8$ 时要取的点，从 $(1, 0)$ 点开始，逆时针从0号开始标号，标到7号  
记编号为 $k$ 的点代表的复数值为 $\omega_n^k$ ，那么由模长相乘，极角相加可知 $(\omega_n^1)^k = \omega_n^k$   
其中 $\omega_n^1$ 称为 $n$ 次单位根，而且每一个 $\omega$ 都可以求出一（我三角函数不好）

$$\omega_n^k = \cos \frac{k}{n}2\pi + i \sin \frac{k}{n}2\pi$$

或者说也可以这样解释这条式子



[https://blog.csdn.net/enjoy\\_pascal](https://blog.csdn.net/enjoy_pascal)

注意 $\sin^2\theta + \cos^2\theta = 1$ 什么的，就容易理解了

那么 $\omega_n^0, \omega_n^1, \dots, \omega_n^{n-1}$ 即为我们要代入的 $x_0, x_1, \dots, x_{n-1}$

## 单位根的一些性质

推FFT的过程中需要用到 $\omega$ 的一些性质

$$\omega_n^k = \omega_{2n}^{2k}$$

- 它们表示的点（或向量）表示的复数是相同的
- 证明
- $\omega_n^k = \cos \frac{k}{n}2\pi + i \sin \frac{k}{n}2\pi = \cos \frac{2k}{2n}2\pi + i \sin \frac{2k}{2n}2\pi = \omega_{2n}^{2k}$

$$\omega_n^{k+\frac{n}{2}} = -\omega_n^k$$

- 它们表示的点关于原点对称，所表示的复数实部相反，所表示的向量等大反向
- 证明
- $\omega_n^{\frac{n}{2}} = \cos \frac{\frac{n}{2}}{n}2\pi + i \sin \frac{\frac{n}{2}}{n}2\pi = \cos \pi + i \sin \pi = -1$
- （这个东西和 $e^{ix} = \cos x + i \sin x$ 与 $e^{i\pi} + 1 = 0$ 有点关系，我不会就不讲子）

Vip

🔍

👤

🔔

$$\omega_n^0 = \omega_n^n$$

- 都等于1，或 $1 + 0i$

99

34









## FFT（快速傅里叶变换）

虽然DFT搞出来一堆很牛逼的 $\omega$ 作为代入多项式的 $x$ 值  
但只是代入这类特殊 $x$ 值法的变换叫做DFT而已，还是要代入单位根暴力计算

- DFT还是暴力 $O(n^2)$ ...

但DFT可以分治来做，于是FFT（快速傅里叶变换）就出来了  
首先设一个多项式 $A(x)$

$$A(x) = \sum_{i=0}^{n-1} a_i x^i = a_0 + a_1 x + a_2 x^2 + \dots + a_{n-1} x^{n-1}$$

按 $A(x)$ 下标的奇偶性把 $A(x)$ 分成两半，右边再提一个 $x$

$$\begin{aligned} A(x) &= (a_0 + a_2 x^2 + \dots + a_{n-2} x^{n-2}) + (a_1 x + a_3 x^3 + \dots + a_{n-1} x^{n-1}) \\ &= (a_0 + a_2 x^2 + \dots + a_{n-2} x^{n-2}) + x(a_1 + a_3 x^2 + \dots + a_{n-1} x^{n-2}) \end{aligned}$$

两边好像非常相似，于是再设两个多项式 $A_1(x), A_2(x)$ ，令

$$\begin{aligned} A_1(x) &= a_0 + a_2 x + a_4 x^2 + \dots + a_{n-2} x^{\frac{n}{2}-1} \\ A_2(x) &= a_1 + a_3 x + a_5 x^2 + \dots + a_{n-1} x^{\frac{n}{2}-1} \end{aligned}$$

比较明显得出

$$A(x) = A_1(x^2) + x A_2(x^2)$$

再设 $k < \frac{n}{2}$ ，把 $\omega_n^k$ 作为 $x$ 代入 $A(x)$ （接下来几步变换要多想想单位根的性质）

$$\begin{aligned} A(\omega_n^k) &= A_1((\omega_n^k)^2) + \omega_n^k A_2((\omega_n^k)^2) \\ &= A_1(\omega_n^{2k}) + \omega_n^k A_2(\omega_n^{2k}) = A_1(\omega_{\frac{n}{2}}^k) + \omega_n^k A_2(\omega_{\frac{n}{2}}^k) \end{aligned}$$

那么对于 $A(\omega_n^{k+\frac{n}{2}})$ 的话，代入 $\omega_n^{k+\frac{n}{2}}$

$$\begin{aligned} A(\omega_n^{k+\frac{n}{2}}) &= A_1(\omega_n^{2k+n}) + \omega_n^{k+\frac{n}{2}} A_2(\omega_n^{2k+n}) \\ &= A_1(\omega_n^{2k} \omega_n^n) - \omega_n^k A_2(\omega_n^{2k} \omega_n^n) \\ &= A_1(\omega_n^{2k}) - \omega_n^k A_2(\omega_n^{2k}) = A_1(\omega_{\frac{n}{2}}^k) - \omega_n^k A_2(\omega_{\frac{n}{2}}^k) \end{aligned}$$

- 发现了什么？

$A(\omega_n^k)$ 和 $A(\omega_n^{k+\frac{n}{2}})$ 两个多项式后面一坨东西只有符号不同  
就是说，如果已知 $A_1(\omega_{\frac{n}{2}}^k)$ 和 $A_2(\omega_{\frac{n}{2}}^k)$ 的值，我们就可以同时知道 $A(\omega_n^k)$ 和 $A(\omega_n^{k+\frac{n}{2}})$ 的值  
现在我们可以递归分治来搞FFT了

每一次回溯时只扫当前前面一半的序列，即可得出后面一半序列的答案  
 $n == 1$ 时只有一个常数项，直接return  
时间复杂度 $O(n \log_2 n)$

## IFFT（快速傅里叶逆变换）

想一下，我们不仅要会FFT，还要会IFFT（快速傅里叶逆变换）  
我们把两个多项式相乘（也叫求卷积），做完两遍FFT也知道了积的多项式的点值表示  
可我们平时用系数表示的多项式，点值表示没有意义

- 怎么把点值表示的多项式快速转回系数表示法？

VIP







- $IDFT$ 暴力 $O(n^2)$ 做? 其实也可以用 $FFT$ 用 $O(n \log_2 n)$ 的时间搞

你有没有想过为什么傅里叶是把 $\omega_n^k$ 作为 $x$ 代入**而不是别的什么数**?  
原因是有的但是有我也看不懂  
由于我是沙雕所以只用记住**一个结论**

- 一个多项式在分治的过程中乘上单位根的共轭复数，分治完的每一项除以 $n$ 即为原多项式的每一项系数

意思就是说 $FFT$ 和 $IFFT$ 可以**一起搞**

## 朴素版FFT板子

`c++`有自带的复数模板`complex`库  
`a.real()`即表示复数 $a$ 的实部

```
1 | #include<complex>
2 | #define cp complex<double>
3 |
4 | void fft(cp *a,int n,int inv)//inv是取共轭复数的符号
5 | {
6 |     if (n==1)return;
7 |     int mid=n/2;
8 |     static cp b[MAXN];
9 |     fo(i,0,mid-1)b[i]=a[i*2],b[i+mid]=a[i*2+1];
10 |    fo(i,0,n-1)a[i]=b[i];
11 |    fft(a,mid,inv),fft(a+mid,mid,inv);//分治
12 |    fo(i,0,mid-1)
13 |    {
14 |        cp x(cos(2*pi*i/n),inv*sin(2*pi*i/n));//inv取决是否取共轭复数
15 |        b[i]=a[i]+x*a[i+mid],b[i+mid]=a[i]-x*a[i+mid];
16 |    }
17 |    fo(i,0,n-1)a[i]=b[i];
18 | }
```

两个多项式 $a, b$ 相乘再转系数多项式 $c$ ，通常只用打这么一小段

```
1 | cp a[MAXN],b[MAXN];
2 | int c[MAXN];
3 | fft(a,n,1),fft(b,n,1);//1系数转点值
4 | fo(i,0,n-1)a[i]*=b[i];
5 | fft(a,n,-1);//-1点值转系数
6 | fo(i,0,n-1)c[i]=(int)(a[i].real()/n+0.5);//注意精度
```

很明显， $FFT$ 只能处理 $n$ 为2的整数次幂的多项式  
所以在 $FFT$ 前一定要把 $n$ 调到2的次幂

这个板子看着**好像**很优美，但是

测试点信息

#1  
AC  
100ms/184984KB

#2  
AC  
100ms/184960KB

#3  
AC  
100ms/184960KB

#4  
AC  
128ms/185019KB

#5  
AC  
1264ms/184984KB

#6  
AC  
660ms/184941KB

#7  
AC  
2548ms/185015KB

#8  
TLE

#9  
TLE

[https://blog.csdn.net/enjoy\\_pascal](https://blog.csdn.net/enjoy_pascal)

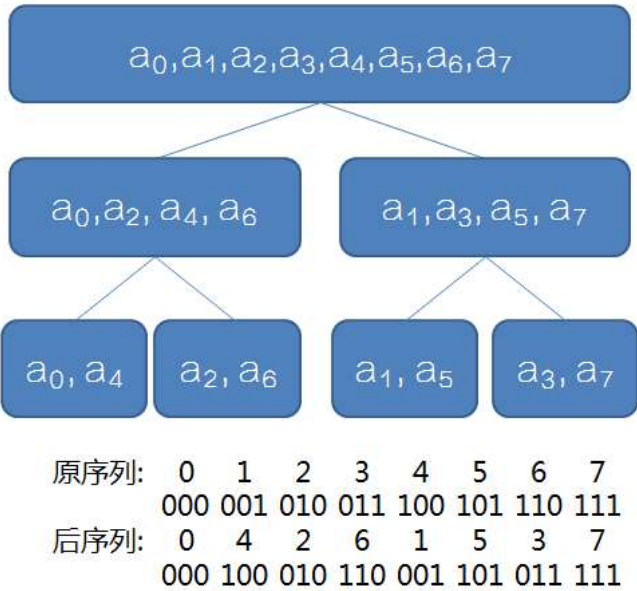
https://blog.csdn.net/enjoy\_pascal/article/details/81478582

7/15

递归常数太大，要考虑优化...

# FFTの优化——迭代版FFT

这个图也是盗的



这个很容易发现点什么吧？

- 每个位置分治后的最终位置为其二进制翻转后得到的位置

这样的话我们可以先把原序列变换好，把每个数放在最终的位置上，再一步一步向上合并  
一句话就可以 $O(n)$ 预处理第 $i$ 位最终的位置 $rev[i]$

```
1 | fo(i,0,n-1)rev[i]=(rev[i>>1]>>1)|((i&1)<<(bit-1));
```

至于蝴蝶变换它死了其实是我不会

## 真•FFT板子

```
1 void fft(cp *a,int n,int inv)
2 {
3     int bit=0;
4     while ((1<<bit)<n)bit++;
5     fo(i,0,n-1)
6     {
7         rev[i]=(rev[i>>1]>>1)|((i&1)<<(bit-1));
8         if (i<rev[i])swap(a[i],a[rev[i]]);//不加这条if会交换两次（就是没交换）
9     }
10    for (int mid=1;mid<n;mid*=2)//mid是准备合并序列的长度的二分之一
11    {
12        cp temp(cos(pi/mid),inv*sin(pi/mid));//单位根，pi的系数2已经约掉了
13        for (int i=0;i<n;i+=mid*2)//mid*2是准备合并序列的长度，i是合并到了哪一位
14        {
15            cp omega(1,0);
16            for (int j=0;j<mid;j++,omega*=temp)//只扫左半部分，得到右半部分的答案
17            {
18                cp x=a[i+j],y=omega*a[i+j+mid];
19                a[i+j]=x+y,a[i+j+mid]=x-y;//这个就是蝴蝶变换什么的
20            }
21        }
22    }
```



```
21 |      }
22 |    }
23 | }
```

99

34

这个板子好像不是那么好背  
至少这个板子已经很优美了

## FFT后记

本人版权意识薄弱.....

- 本博客部分知识学习于
- <https://www.cnblogs.com/RabbitHu/p/FFT.html>
- [https://www.cnblogs.com/zwfymqz/p/8244902.html?mType=Group#\\_label3](https://www.cnblogs.com/zwfymqz/p/8244902.html?mType=Group#_label3)
- [https://blog.csdn.net/ggn\\_2015/article/details/68922404](https://blog.csdn.net/ggn_2015/article/details/68922404)

NTT我来了

### 股神徐翔狱中曝出庄家洗盘规律，牢记这3点，看完恍然大悟！

股管家 · 顶新

想对作者说点什么

HOGWARTS333: a和b两个n项的相乘，c不应该最多是2n项的吗，你得出的c是n项的，怎么转化成系数 （1周前 #14楼） [查看回复\(7\)](#)

丁恩妃: 博主 你这段 朴素高精度乘法时间O(n2)，但FFT能O(nlog2n) FFT的复杂度为啥要写以2为底呀 不管多少都行吧 （1个月前 #13楼） [查看回复\(2\)](#)

QingchangHan: 初三学生。。博主真厉害。。 （2个月前 #12楼）

魔灵幽亭: 初中？我学白了！ （2个月前 #11楼）

[登录](#) [查看 34 条热评](#)

### FFT算法讲解——麻麻我终于会FFT了！

阅读数 4万+

FFT——快速傅里叶变换这块不写东西空荡荡的，我决定还是把FFT的定义给贴上吧FFT（FastFourierTransformatio... [博文](#) 来自: [谁能代表肯德基](#)

### 快速傅里叶变换(FFT)的原理及公式

阅读数 2349

为了庆祝我学会FFT转一篇blog快速傅里叶变换(FFT)的原理及公式 非周期性连续时间信号x(t)的傅里叶变换可以... [博文](#) 来自: [BlackJack](#)

### 我所理解的快速傅里叶变换（FFT）

阅读数 9万+

1.历史放在最前头首先FFT是离散傅立叶变换(DFT)的快速算法，那么说到FFT，我们自然要先讲清楚傅立叶变换。先... [博文](#) 来自: [沈春旭的博客](#)

### FFT详解

阅读数 6万+

一直想学FFT，当时由于数学基础太差，导致啥都学不懂。请教了机房里的几位学长大神，结果还是没太明白。因此... [博文](#) 来自: [总理同学的编程尝试](#)



### 40个漂亮的html5网站欣赏

### FFT（最详细最通俗的入门手册）

阅读数 3万+

声明首先，我需要声明，本文是在转载的基础上稍微修饰的，经过原创作者ZLH\_HHHH（佐理慧学姐）的许可方才... [博文](#) 来自: [逐梦者](#)

### FFT的详细解释，相信你看了就明白了

阅读数 2万+

转自: <http://www.ilovematlab.cn/thread-119939-1-1.html>FFT是离散傅立叶变换的快速算法，可以将一个信号变... [博文](#) 来自: [zhaopeizhaopeip...](#)

### Learning：多项式（一）（FFT）

阅读数 396

多项式的表示1 系数表示法设是一个关于的次多项式，则：2 点值表达法我们可以把次多项式看作一个函数，那么它... [博文](#) 来自: [一个蒟蒻的博客](#)