A Convolution-Oriented FFT Tutorial

for OI/XCPC participants and algebra enthusiasts

sun123zxy

BITACMCLUB

2023-08-01[1]

[1]最后更新于 2023-09-28.

目录

目录

卷积，但不止卷积 - FFT 漫谈

- 先有 FT，再有 DFT，才有 FFT
- 时频转换是最初的用途
- 发现单位根优秀性质，James Cooley, John Tukey 发明现代 FFT 加速 DFT，但此前相似的发现早已有之
- 后来将 DFT 与卷积定理联系，FFT 才被用于计算多项式乘法
- 复数运算精度误差推动了 NTT 的发展
- 应用：任何需要频率和卷积的地方．频谱、滤波器、音乐、雷达、图像处理……
- OI/XCPC 中主要关心卷积

推荐食用方法

- 初步要求
  - 知道 DFT、FFT 可用于快速计算多项式卷积
  - 掌握 FFT 加速 DFT 计算的原理和实现
  - 会应用结论改动 FFT 加速 NTT 计算
  - 见识一些卷积解决的基本问题，初步了解生成函数在组合计数中的应用
  - 题目可选择性完成，请多花时间消化原理和思想
- 学有余力 / 集训后继续消化
  - 系统学习生成函数
  - 实现多项式全家桶
  - 对原理感兴趣的同学可对数学部分做进一步研究．本讲内容是线性代数、抽象代数、数论等多领域的综合应用．欢迎讨论．
  - 学习集合幂级数相关知识点（FMT，FWT，……），体会其思想与 FFT 的同与异
  - 学习 FFT 在信号、频谱等非算法竞赛向实际问题中的应用
- Learn for fun :)

记号说明

- 记 $[n] = \{0, 1, \ldots, n-1\}$，此时可用 $k \in [n]$ 代替下标取值范围 $k = 0, 1, \ldots, n-1$ 的记法.
  - 集合论中已定义 $n = \{0, 1, \ldots, n-1\}$，这里的中括号是为了强调其集合含义.
- 使用 Iverson 括号约定：设 P 是一个命题，记

$$[P] := \begin{cases} 1 & P \text{ is true} \\ 0 & \text{otherwise} \end{cases}$$

- 多项式的规模定义为多项式的次数加一. 特别的，零多项式的规模为 0.
  - 以后会混用 $n-1$ 次多项式和规模为 n 的多项式的说法.

目录

目录

多项式卷积

给定两个至多 $n-1$ 次的多项式

$$A(x)=\sum_{k=0}^{n-1}a_kx^k,\quad B(x)=\sum_{k=0}^{n-1}b_kx^k$$

如何快速计算两者的卷积，即它们相乘得到的多项式的系数？

$$\begin{aligned}(A*B)(x)=A(x)B(x)&=\sum_{i=0}^{n-1}a_ix^i\sum_{j=0}^{n-1}b_jx^j\\&=\sum_{k=0}^{2n-2}x^k\sum_{i+j=k}a_ib_j\end{aligned}$$

上式给出了 $O(n^2)$ 的朴素做法.

系数 - 点值 - 系数

- 代入任意 x 可得到多项式在 x 处的点值
- 点值意义下的多项式乘法是 $O(n)$ 的

$$(A * B)(x) = A(x)B(x)$$

- n 点确定一个至多 $n-1$ 次的多项式[2]

若计算至多 $n-1$ 次的多项式的某 n 个点值存在快速算法，而通过多项式的某 n 个点值确定原多项式系数亦存在快速算法，就有通过 $O(n)$ 的点值乘法加速多项式乘法计算的可能.

[2]一种常见的证法是使用 Vandermonde 行列式证明矩阵可逆. 后面会介绍多项式环风格的证明.

系数 - 点值 - 系数 - 快速转换？

- 朴素计算任意指定 n 个位置点值需要 $O(n^2)$.
- Lagrange[1] 插值给出了 $O(n^2)$ 将任意位置 n 个点值还原为多项式系数的算法.
- 能否选取 n 个特殊的点值使系数 - 点值、点值 - 系数的变换支持快速计算？

Discrete Fourier Transform

离散傅里叶变换（Discrete Fourier Transform, DFT）接受一个至多 $n-1$ 次的多项式的 n 个系数，将复数域上的 n 个 n 次单位根代入系数表达式以得到给定多项式的 n 个点值.

$$a_0, a_1, \ldots, a_{n-1} \longrightarrow A(1), A(\omega_n), \ldots, A(\omega_n^{n-1})$$

得益于单位根的特殊运算性质，二者均有被称为快速傅里叶变换（Fast Fourier Transform, FFT）的快速算法.

复数域单位根

复数域上的 n 个 n 次单位根[3]

$$\omega_n^k := e^{\frac{2\pi k}{n}i} = \cos\frac{2\pi k}{n} + i\sin\frac{2\pi k}{n}, \quad k \in [n]$$

是复平面单位圆上的 n 等分点，易验证它们是复数域中唯一满足方程 $z^n = 1$ 的解.

所有单位根模长均为 1. 第 k 个单位根的辐角为 $\frac{2\pi k}{n}$. "复数乘法" 模长相乘，辐角相加" 的性质告诉我们 $\omega_n^i \omega_n^j = \omega_n^{i+j}$.

注记 (Euler's formula for nerds)

Euler 公式 $e^{it} = \cos t + i\sin t$ 的一种证明可用指数函数的另一定义 $\exp z = \lim_{n\to\infty}(1 + \frac{z}{n})^n$ 分析复数处极坐标的极限 [2]. 严格来讲，如果三角函数和指数函数都由级数定义 [3]，Euler 公式几乎是显然的. 总之，这里仅将 Euler 公式作为一种紧凑的记号使用，细节不做要求.

[3]有的文献定义 $\omega_n := e^{-\frac{2\pi}{n}i}$，或是因为信号处理领域常用 IDFT 将信号时域采样数据变为频域信息. 事实上，DFT/IDFT 的说法也常有反转，但这只是形式问题.

复数域单位根 - 三个重要性质[4]

定理 2.1 (消去引理)

$$\omega_{dn}^{dk} = \omega_n^k, \quad \forall d \in \mathbb{N}_+$$

定理 2.2 (折半引理)

$$\left\{\omega_{2n}^{2k} : k \in [2n]\right\} = \left\{\omega_n^k : k \in [n]\right\}$$

消去 / 折半引理将在 FFT 的推导中使用.

[4]这三个引理是《算法导论》[4] 引入的.

复数域单位根 - 三个重要性质

定理 2.3 (求和引理)

$$\frac{1}{n}\sum_{k=0}^{n-1}\omega_n^{ik} = [n \mid i]$$

求和引理的证明使用了等比数列求和公式. 将在 IDFT 的推导中用到.

考虑将至多 $2n-1$ 次的待变换多项式 $A(x)=\sum_{k=0}^{2n-1}a_kx^k$ 奇偶分项两个至多 $n-1$ 次的多项式 $A(x)=A_0(x^2)+xA_1(x^2)$，其中

$$A_0(x)=\sum_{k=0}^{n-1}a_{2k}x^k,\quad A_1(x)=\sum_{k=0}^{n-1}a_{2k+1}x^k$$

代入 $x=\omega_{2n}^k,\ k\in[2n]$，用单位根消去或折半引理（$\omega_{2n}^{2k}=\omega_n^k$）得

$$A(\omega_{2n}^k)=A_0(\omega_n^k)+\omega_{2n}^kA_1(\omega_n^k)$$

再用 $\omega_{2n}^n=-1$ 得

$$\begin{aligned}A(\omega_{2n}^k)&=A_0(\omega_n^k)+\omega_{2n}^kA_1(\omega_n^k)\\A(\omega_{2n}^{n+k})&=A_0(\omega_n^k)-\omega_{2n}^kA_1(\omega_n^k)\end{aligned},\quad k\in[n]$$

原多项式 $A(x)$ 规模为 $2n$ 的 DFT 转化为规模为 n 的 $A_0(x)$ 和 $A_1(x)$ 的 DFT. 递归计算就可得到 $O(n\log n)$ 的算法.

DFT 的矩阵表示

记

$$\mathbf{a} = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix}, \quad F = \left(\omega_n^{ij}\right)_{(i,j)\in n\times n} = \begin{pmatrix} 1 & 1 & \dots & 1 \\ 1 & \omega_n & \dots & \omega_n^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_n^{n-1} & \dots & \omega_n^{(n-1)(n-1)} \end{pmatrix}$$

则 DFT 的变换结果（给定系数 $a_0, a_1, \dots, a_{n-1}$ 的多项式在 n 个 n 次单位根处的点值）可表示为

$$\hat{\mathbf{a}} = \begin{pmatrix} A(1) \\ A(\omega_n) \\ \vdots \\ A(\omega_n^{n-1}) \end{pmatrix} = F\mathbf{a}$$

由单位根的消去引理可证，DFT 矩阵 F 的逆矩阵为[5]

$$F^{-1} = \frac{1}{n}\left(\omega_n^{-ij}\right)_{(i,j)\in n\times n} = \frac{1}{n}\begin{pmatrix} 1 & 1 & \dots & 1 \\ 1 & \omega_n^{-1} & \dots & \omega_n^{-(n-1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_n^{-(n-1)} & \dots & \omega_n^{-(n-1)(n-1)} \end{pmatrix}$$

于是 IDFT 的过程可表示为

$$\mathbf{a} = F^{-1}\hat{\mathbf{a}}$$

故快速计算 IDFT 的方法与 FFT 几乎一致，只需将计算 DFT 时使用的本原单位根 ω_n 替换为 ω_n^{-1} 并对最终结果除以 n 即可.

[5]这也表明适当归一化后的 DFT 矩阵是一个酉矩阵.

梳理

- 怎么计算卷积？
  - 把至多 $n-1$ 次的多项式 $A(x)$ 和至多 $m-1$ 次的多项式 $B(x)$ 写成至多 $n+m-2$ 次的多项式（高位补 0）. 为计算 FFT 方便，还要继续补 0 至一个大于其次数的 2 的幂.
  - 对 $A(x)$ 和 $B(x)$ 多点求值.
  - 把两个多项式的点值逐点相乘.
  - 多点插值还原 $(A*B)(x)$ 的系数.
- 怎么快速求值？
  - 选点选单位根就是 DFT.
  - 消去引理和折半引理使我们可以用 FFT 算法递归地计算 DFT.
  - 推导已经给出了递归的写法，之后还会介绍常数更优的迭代实现.
- 怎么快速插值？
  - 求和引理给出了 DFT 矩阵的逆矩阵.
  - 计算方法很相似，最后逐项除掉一个规模.

FFT 递归实现 - DFT 部分

方便起见，我们只处理 n 为 2 的幂的情形. 以下 C 风格的代码实现了递归的 DFT 和 IDFT.

```
#include<bits/stdc++.h>
#include<complex>
using namespace std;
typedef long long ll; typedef complex<double> CP;
const ll MXN=4E6+5; const double PI=3.14159265358979323846l;
CP tmp[MXN];
void _DFT(CP A[],ll n,ll typ){
    n/=2; if(n==0) return;
    for(ll k=0;k<n;k++) tmp[k]=A[2*k],tmp[n+k]=A[2*k+1];
    for(ll k=0;k<2*n;k++) A[k]=tmp[k];
    _DFT(A,n,typ); _DFT(A+n,n,typ);
    CP w(cos(2*PI/(2*n)),typ*sin(2*PI/(2*n))), wk=1;
    for(ll k=0;k<n;k++){
        tmp[  k]=A[k]+wk*A[n+k];
        tmp[n+k]=A[k]-wk*A[n+k];
        wk*=w;
    } for(ll k=0;k<2*n;k++) A[k]=tmp[k];
} void DFT(CP A[],ll n,ll typ){
    _DFT(A,n,typ); if(typ==-1) for(ll i=0;i<n;i++) A[i]*=1.0l/n;
}
```

FFT 递归实现 - 卷积部分

```
// alternatively, use std::__lg() in GCC
ll log2ceil(ll n){ll cnt=0; for(ll t=1;t<n;t<<=1) cnt++; return cnt;}
CP A[MXN],B[MXN],C[MXN]; ll outC[MXN];
ll* conv(ll inA[],ll aN,ll inB[],ll bN){
    ll n=1LL<<log2ceil(aN+bN-1);
    for(ll i=0;i<aN;i++) A[i]=inA[i];
    for(ll i=0;i<bN;i++) B[i]=inB[i];
    DFT(A,n,1); DFT(B,n,1);
    for(ll i=0;i<n;i++) C[i]=A[i]*B[i];
    DFT(C,n,-1); for(ll i=0;i<n;i++) outC[i]=round(C[i].real());
    return outC;
}
```

Drawbacks?

- 递归实现慢
- 临时数组丑
- 封装性为零

FFT 迭代

迭代地实现 FFT 不仅在常数上更加优秀，亦更便于使用 C++ 的容器进行封装. 这并不困难，只需自底向上模拟 FFT 递归过程即可.

唯一的问题——最底层的顺序？

来观察一轮 2^3-FFT 自顶向下的置换过程

2^3	$0/000$	$1/001$	$2/010$	$3/011$	$4/100$	$5/101$	$6/110$	$7/111$
2^2	$0/000$	$2/010$	$4/100$	$6/110$	$1/001$	$3/011$	$5/101$	$7/111$
2^1	$0/000$	$4/100$	$2/010$	$6/110$	$1/001$	$5/101$	$3/011$	$7/111$
2^0	$0/000$	$4/100$	$2/010$	$6/110$	$1/001$	$5/101$	$3/011$	$7/111$

你发现了什么？

蝶形运算

在计算点值前，2^n-FFT 事实上完成了一次 n-位逆序置换. 分解来看，规模为 2^k 的层的置换完成了对 2^{n-k} 对应二进制位的分类.

我们有 $O(n)$ 的递推方法获得这一置换.

```
void spawnrev(ll n){
    rev[0]=0;
    for(ll i=1;i<(1<<n);i++)
        rev[i]=(rev[i>>1]>>1)+((i&1)<<(n-1));
}
```

FFT 迭代实现

```
void DFT(CP A[],ll n,ll typ){ // rev[] should be spawned in advance
    for(ll i=0;i<n;i++) if(i<rev[i]) swap(A[i],A[rev[i]]); // a one-to-one permutat
    for(ll hf=1;hf<n;hf*=2){
        CP w(cos(2*PI/(2*n)),typ*sin(2*PI/(2*n))), wk=1;
        for(ll i=0;i<n;i+=hf*2){
            CP wk=1;
            for(ll k=0;k<hf;k++){
                CP x=A[i+k],y=wk*A[i+hf+k];
                A[i+k]=x+y; A[i+hf+k]=x-y;
                wk=wk*w;
            }
        }
    }
    if(typ==-1) for(ll i=0;i<n;i++) A[i]*=1.0l/n;
}
```

请自行实现更易用的容器封装版本.

目录

NTT 速成[7]

FFT 的缺点？浮点数带来的大常数与精度问题.

$998244353 = 2^{23} \times 7 \times 17 + 1$，同时是一个质数.

我们指出，在系数和点值模 $p = 998244353$ 的意义下，当规模 $n \le p$ 时[6]，至多 $n-1$ 次的多项式仍可由其 n 个点值唯一确定，故仍可使用系数-点值-系数的方法求得多项式卷积.

我们指出，对于满足 $n \mid p-1$ 的 n，依 $\omega_n := 3^{\frac{p-1}{n}}$ 定义的 ω_n 在模 p 意义下与复数域中定义的 ω_n 发挥相同的作用，仍可进行规模至多为 2^{23} 的 FFT 作为模 $p = 998244353$ 意义下 NTT 的快速算法.

只需修改单位根定义，把复数运算改为整数取模，就得到了能算 NTT 的 FFT 的实现.

```
const PR=3,MOD=998244353;
ll w=qpow(PR,(MOD-1)/(hf*2)); if(typ==-1) w=inv(w);
```

[6]否则只有 p 个不同元素的 $\mathbb{Z}_p$ 中根本取不到 n 个不同位置的点值. 后面会深入讨论.
[7]NTT 原理需较多笔墨，稍后介绍.

FFT/NTT in a nutshell - 小结：概念区分

- 关于 DFT
  - Discrete Fourier Transform, DFT, 离散傅里叶变换
  - Fast Fourier Transform, FFT, 快速傅里叶变换
  - FFT 是计算 DFT 的快速算法
- 关于 NTT
  - Number Theoretic Transform, NTT, 数论变换
  - FFT 在复数域上的多项式环 $\mathbb{C}[x]$ 中进行，而 NTT 在模 p 剩余类域 $\mathbb{Z}_p[x]$ 上进行
  - 快速计算 DFT / IDFT, NTT 都用 FFT，故一般不使用"FNTT" 的说法

目录

目录

基本应用

大整数乘法 十进制数可拆解为多项式表示，计算卷积后处理进位即可. 由于数字最大只是 9，合理数据范围下卷起来不会爆模数，直接用 NTT 实现即可.

背包计数 两个背包的合并就是多项式卷积.

滤波器 反转多项式的系数数组再做卷积，可以快速得到两个多项式滑动窗口式的内积.

位运算 有些位运算可以写成卷积的形式. 模 2 意义下”异或”是加法，“与”是乘法，“或”可以通过取反转化为”与”.

字符串 通过巧妙设定字符串距离函数，FFT 可解决更广泛的字符串匹配问题.

值域次数化 当值域较小时，将待计算的值放在多项式次数上统计贡献次数，可以绕开某些极难求解的数值问题. 例如 Vandermonde 行列式的快速计算.

基本应用 - 分治 FFT

对多个长度相同的多项式的卷积，分治地卷起来可降低时间复杂度. 长度不一时，挑小的先卷也可减小常数（用堆维护）.

另有一种 CDQ 风格的分治 FFT. CDQ 长于处理带偏序的二元点对贡献，在处理形如 $c_k = \sum_{i>j}[i+j=k]a_i b_j$ 的带偏序卷积时可以应用.

当卷积的前后项存在依赖关系时，也可使用此法保证处理顺序恰当. 然而此类依赖问题往往也可通过解生成函数方程的方法求得封闭形式.

时间复杂度均为

$$T(n) = 2\,T\left(\frac{n}{2}\right) + O(n\log n) = O(n\log^2 n)$$

目录

生成函数初步 - 导言

生成函数是一种对数列的操作技巧. 通过将数列表示为多项式或形式幂级数，数列间复杂的和式操作可用简单函数的乘法、复合等运算进行表示，从而大大降低了数列变换技巧的使用门槛.

生成函数在组合数学中应用广泛，且生成函数的部分操作在组合意义下也有较为直观的理解. 本节将带大家初窥其中的奥妙. 限于篇幅和主讲人能力，我们仅以题带点地讲解，期冀为大家建立构造生成函数的直觉. 请感兴趣的同学下来做进一步研究.

熟悉 Taylor 展开的同学或能较快上手此部分内容.

Ordinary Generating Function

序列 a_n 的普通生成函数（Ordinary Generating Function, OGF）定义为其对应的多项式（形式幂级数）$A(x) = \sum_{n=0}^{+\infty} a_n x^n$. 这样书写的目的是为了便于进行数列间的卷积操作. 特别的，Taylor 展开（或广义二项式定理）风格的 $\frac{1}{1-x} = 1 + x + x^2 + \dots$ 也是常用技巧之一[8].

习题 3.1

写出下列数列的 OGF. 下标从 0 开始.

- $1, 0, 0, 0 \dots$
- $1, 1, 1, 1, \dots$
- $1, 0, 1, 0, \dots$
- $1, -1, 1, -1, \dots$
- $1, 2, 3, 4, \dots$
- $1, 2, 4, 8, \dots$

[8] 多项式求逆等多项式进阶操作，我们后续讲解.

OGF 组合意义

OGF 相乘，是背包，是卷积，是两块无标号组合对象的有序拼接.

习题 3.2

写出下列计数问题的 OGF，均以 n 作为数列的下标.

- 在 m 个物品中选出 n 个的方案数.
- 容量为 n 的背包装下体积分别为 a_i 的 m 个物品的方案数.
- 同上，但每个物品有无限个.
- n 个无标号球放入 r 个有标号盒的方案数，要求盒非空.
- 同上，但盒可空.
- 将整数 n 分拆为若干正整数之和的方案数.

Exponential Generating Function

序列 a_n 的指数生成函数（Exponential Generating Function, EGF）定义为

$$A(x) = \sum_{n=0}^{+\infty} a_n \frac{x^n}{n!}$$

习题 3.3

写出下列序列的 EGF，下标从 0 开始.

- $1, 0, 0, 0, \ldots$
- $1, 1, 1, 1, \ldots$
- $1, 2, 4, 8, \ldots$
- $0, 1, 0, -1, 0, 1, 0, -1, \ldots$
- $1, 0, -1, 0, 1, 0, -1, 0, \ldots$
- $0, 1, -1, 2!, -3!, 4!, \ldots$

EGF 组合意义

观察两个 EGF 的乘积

$$
\begin{aligned}
A(x)B(x) &= \sum_{i=0}^{+\infty} a_i \frac{x^i}{i!} \sum_{j=0}^{+\infty} b_j \frac{x^j}{j!} \\
&= \sum_{i=0}^{+\infty} \sum_{j=0}^{+\infty} a_i b_j \frac{x^{i+j}}{i!j!} \\
&= \sum_{n=0}^{+\infty} \sum_{i=0}^{n} a_i b_{n-i} \frac{x^n}{i!(n-i)!} \\
&= \sum_{n=0}^{+\infty} \sum_{i=0}^{n} a_i b_{n-i} \binom{n}{i} \frac{x^n}{n!}
\end{aligned}
$$

EGF 的乘积，相当于两块有标号组合对象的有序拼接.

EGF 组合意义

$$A(x)B(x) = \sum_{i=0}^{+\infty} a_i \frac{x^i}{i!} \sum_{j=0}^{+\infty} b_j \frac{x^j}{j!} = \sum_{n=0}^{+\infty} \sum_{i=0}^{n} a_i b_{n-i} \binom{n}{i} \frac{x^n}{n!}$$

习题 3.4

写出下列计数问题的 EGF，均以 n 作为数列的下标.

- 长度为 n 的排列的构型数.
- 长度为 n 的圆排列的构型数.
- 将 n 个有标号球放入 r 个有标号盒的方案数，要求盒非空.
- 将 n 个有标号球放入 r 个无标号盒的方案数，要求盒非空.
- 将 n 元集合划分为 r 个等价类的方案数.
- 划分 n 元集合的方案数.

目录

目录

NTT 原理 - 导言

本节介绍 NTT 的原理.

FFT 加速卷积算法的核心，一是多项式的求值插值原理，二是单位根带来的分治快速算法. 我们将在本节中证明，模 p 剩余类域 $\mathbb{Z}_p$ 中，只要 $n \leq p$，仍可通过 n 个点值唯一确定至多 $n-1$ 次的 $\mathbb{Z}_p$ 上的多项式，这保证了系数-点值-系数方法的正确性；此外，基于原根构造出的 $\mathbb{Z}_p$ 上的本原单位根，使得使用与 FFT 相同的分治算法加速 NTT 的计算成为可能.

NTT 原理涉及原根等数论内容. 本讲的目标是建立 DFT 变换和 FFT 算法的通用数学框架，而非具体研究其某一特例. 故我们只讲解 NTT 所需的基础数论知识，无关的细节则略过处理. 对数论感兴趣的同学可前往 OI Wiki 学习.

模 p 剩余类域 $\mathbb{Z}_p$

模 p 剩余类域 $\mathbb{Z}_p$ 是刻画取模运算下整数加法、乘法运算规律的代数结构，其良定义性由

$$\begin{cases} a_1 \equiv a_2 \pmod{p} \\ b_1 \equiv b_2 \pmod{p} \end{cases} \Longrightarrow \begin{cases} a_1 + b_1 \equiv a_2 + b_2 \pmod{p} \\ a_1 b_1 \equiv a_2 b_2 \pmod{p} \end{cases}$$

保证.

模 p 剩余类域的最大特点是其中每个元素都存在（唯一）逆元. 这一点是数论中 Bézout 定理的直接应用.

模 p 剩余类域的另一优良性质是其满足消去律，即

$$ab \equiv 0 \pmod{p} \Longrightarrow a \equiv 0 \vee b \equiv 0 \pmod{p}$$

而模一般的数 m 形成的代数结构则不满足此性质.

$\mathbb{Z}_p[x]$ 上的多点插值

之后记系数均在 $\mathbb{Z}_p$ 上的多项式组成的集合为 $\mathbb{Z}_p[x]$.

引理 4.1 ($\mathbb{Z}_p[x]$ 上的多项式余式定理)

设 $A(x) \in \mathbb{Z}_p[x]$，用 $\mathbb{Z}_p[x]$ 中的一次多项式 $(x-a)$ 对 $A(x)$ 进行带余除法，得到的余式为至多零次的多项式 $A(a)$.

$\mathbb{Z}_p[x]$ 上的多点插值

定理 4.1 (Lagrange 定理)

设 p 是一个质数，设 $A(x) \in \mathbb{Z}_p[x]$. 同余方程 $A(x) \equiv 0 \pmod{p}$ 只有至多 $\deg A(x)$ 个模 p 意义下不同的整数解，除非这多项式的系数全为零.

推论 4.1

设 $A(x), B(x) \in \mathbb{Z}_p[x]$ 是至多 $n-1$ 次的两个多项式. 若同余方程 $A(x) \equiv B(x) \pmod{p}$ 有 n 个不同的模 p 意义下的整数解，则两多项式一定相等.

推论告诉我们，欲确定 $\mathbb{Z}_p$ 上的一个至多 $n-1$ 次的多项式，只需找到其 n 个模 p 意义下的点值即可. 因此，使用系数 - 点值 - 系数方法加速 $\mathbb{Z}_p[x]$ 上的卷积是可行的.

本原单位根

回顾 DFT 中复数域 $\mathbb{C}$ 上单位根的定义

$$\omega_n^k := e^{\frac{2\pi k}{n}i} = \cos\frac{2\pi k}{n} + i\sin\frac{2\pi k}{n}, \quad k \in [n]$$

其中 ω_n 是最重要的单位根，因为它生成了所有其它的单位根.

事实上所有的 $\omega_n^k : \gcd(n,k) = 1$ 也都是重要的，它们的 0 至 $n-1$ 次方也能生成所有的单位根. 这也是数论中 Bézout 定理的直接应用.

我们把这一类重要的单位根称为本原单位根. 抽象的来说，n 次本原单位根 ω_n 可定义如下

$$\omega_n^n = 1; \quad \omega_n^k \neq 1, \quad k = 1, 2, \ldots, n-1$$

$\mathbb{Z}_m$ 上的阶

如何定义 $\mathbb{Z}_m$ 上的“本原单位根”呢？

称在模 m 意义下使得 $a^n = 1$ 的最小正整数 n 为 a 的阶，记为 $\mathrm{ord}_m(a)$. 在不至混淆的情况下，也简记为 $\mathrm{ord}(a)$.

- 阶最高有多高？

定理 4.2 (Fermat 小定理)

若 p 是质数，a 是非零数，则 $a^{p-1} \equiv 1 \pmod{p}$.

定理 4.3 (Euler 定理)

若 $\gcd(a, m) = 1$，则 $a^{\varphi(m)} \equiv 1 \pmod{m}$，其中 $\varphi(m)$ 是数论中的 *Euler* 函数[a].

[a]对质数 p，$\varphi(p) = p - 1$. 故 Euler 定理是 Fermat 小定理的一个推广.

Euler 定理的证明 - 简化剩余系

$\mathbb{Z}_m^R := \{a \in \mathbb{Z}_m : \gcd(a, m) = 1\}$ 被称为 $\mathbb{Z}_m$ 的简化剩余系. 简化剩余系对模 m 乘法封闭. 简化剩余系中的每个元素都具有模 m 意义下的唯一逆元，且其也在 $\mathbb{Z}_m^R$ 中. 此二性质易由 $\gcd$ 的线性组合意义证明.

简化剩余系的大小即 $\varphi(m)$. 这是 Euler 函数的其中一种定义.

设 $a \in \mathbb{Z}_m^R$，其在 $\mathbb{Z}_m$ 中存在逆元，故我们有

$$\forall x, y \in \mathbb{Z}_m, \quad x \equiv y \pmod{m} \iff ax \equiv ay \pmod{m}$$

当限定 $x, y \in \mathbb{Z}_m^R$ 时，结合 $\mathbb{Z}_m^R$ 上述两个性质，就有

$$\mathbb{Z}_m^R = a\mathbb{Z}_m^R = \{ax \bmod m : x \in \mathbb{Z}_m^R\}$$

故

$$a^{\varphi(m)} \prod_{x \in \mathbb{Z}_m^R} x \equiv \prod_{x \in \mathbb{Z}_m^R} ax \equiv \prod_{x \in a\mathbb{Z}_m^R} x \equiv \prod_{x \in \mathbb{Z}_m^R} x \implies a^{\varphi(m)} \equiv 1 \pmod{m}$$

a^n 的取值 - $a,\ m$ 互质时

补充讨论 $a^n \bmod m,\ n \in \mathbb{Z}_+$ **可取得哪些** $\mathbb{Z}_m$ **中的元素.**

当 $\gcd(a, m) = 1$ **时，用** gcd **的线性组合含义，有**

$$\begin{aligned} \gcd(a,m) = 1 &\implies \exists x_0, \quad ax_0 = 1 \pmod m \\ &\implies \exists x_0, \quad a^n x_0^n = 1 \pmod m \\ &\implies \gcd(a^n, m) = 1 \end{aligned}$$

故此时 $a^n \bmod m$ **只可能取得** $\mathbb{Z}_m$ **的简化剩余系** $\mathbb{Z}_m^R$ **中的元素.**

a^n 的取值 - 一般情况[9]

当 $d_1 = \gcd(a, m) \neq 1$ 时，设 $a = d_1 a_1$，$m = d_1 m_1$，当 $n \geq 1$ 时，

$$a^n \bmod m = d_1 a_1 a^{n-1} \bmod d_1 m_1 = d_1 \left(a_1 a^{n-1} \bmod m_1\right)$$

此时 $\gcd(a_1, m_1) = 1$，根据线性同余方程理论，括号内表达式的取值集合仅与 $a^{n-1} \bmod m_1$ 有关. 此时若 $d_2 = \gcd(a, m_1)$ 仍不等于 1，则令 $a_1 = d_2 a_2$，$m_1 = d_2 m_2$. 类似地继续化归，最终有 $d_{k+1} = \gcd(a, m_k) = 1$，

$$a^n \bmod m = d_1 \left(a_1 d_2 \left(\ldots a_{k-1} d_k (a_k a^{n-k} \bmod m_k) \ldots\right) \bmod m_1\right)$$

故当 $n \geq k$ 时，$a^n \bmod m$ 的取值集合为

$$\left\{d_1 d_2 \ldots d_k (a^t \bmod m_k) : t \in \mathbb{Z}_+\right\} \subset d_1 d_2 \ldots d_k \mathbb{Z}_{m_k}^R$$

事实上，$\prod_{i=1}^{k} d_i$ 恰为 $\gcd(a, m)$ 的各素因子在 m 中全部成分的乘积.

[9] ExBSGS 求解离散对数 [5] 的推导与此相似.

原根[10]

需要注意的是，Euler 定理只给出了 $\mathbb{Z}_m$ 上元素阶的一个上界. 换言之，其说明

$$\forall a \in \mathbb{Z}_m^R \implies \mathrm{ord}_m(a) \mid \varphi(m)$$

而对于那些使得 $\gcd(a, m) \neq 1$ 的元素 a，用 gcd 的线性组合性质容易证明其永远不可能通过自乘变为 1. 我们设定这类元素的阶为 0.

称 $\mathbb{Z}_m$ 上那些确能达到阶数上界 $\varphi(m)$ 的元素是 $\mathbb{Z}_m$ 上的原根，用符号 g 表示.

定理 4.4 (原根存在定理)

$\mathbb{Z}_m$ 上存在原根，当且仅当 $m = 2, 4, p^\alpha, 2p^\alpha$ ，其中 p 是一奇质数.

[10] 部分证明稍复杂，我们略过处理. 感兴趣的同学请参考 OI Wiki [6].

求阶和原根

- 求阶，用定理 $\mathrm{ord}_m(a) \mid \varphi(m)$，求因子 + 快速幂即可 $O(\sqrt{m}\log m)$.
- 找原根，从小到大用上述求阶方法暴力即可. 最小原根一般不会太大.
- 更快的方法请参考洛谷求原根模板题题解.

下面再介绍原根的两个定理.

定理 4.5 (原根判定定理)

若 $\gcd(g, m) = 1$，则 g 是 $\mathbb{Z}_m$ 上原根的充要条件是，对 $\varphi(m)$ 的每个素因数 p，都有 $g^{\frac{\varphi(m)}{p}} \neq 1 \pmod m$.

必要性显然. 充分性，反证出所有 $\varphi(m)$ 的非平凡因子均不是 g 的阶即可.

定理 4.6 (原根个数定理)

若 $\mathbb{Z}_m$ 上存在原根，则它原根的个数为 $\varphi(\varphi(m))$.

$\mathbb{Z}_m$ 上的本原单位根构造

设 a 是 $\mathbb{Z}_m$ 上一元素，当 $n \mid \mathrm{ord}_m(a)$，令 $\omega_n = a^{\frac{\mathrm{ord}_m(a)}{n}}$，容易发现 ω_n 就是 $\mathbb{Z}_m$ 上的 n 次本原单位根.

使用数论中 Bézout 定理，ω_n^k 是 $\mathbb{Z}_m$ 上的 $\frac{n}{\gcd(n,k)}$ 次本原单位根. 这是 $\mathbb{Z}_m$ 上本原单位根的消去引理.

遗憾的是，一般的 $\mathbb{Z}_m$ 中，本原单位根不一定满足求和引理[11] $\frac{1}{n}\sum_{k=0}^{n-1}\omega_n^{ik} = [n \mid i]$. 这是因为，求和引理的证明中使用了等比数列求和和消去律，但一般的 $\mathbb{Z}_m$ 中消去律并不成立.

[11] 这是后文所述定理“整环上的本原单位根也是主要单位根”在一般环上的一个反例.

$\mathbb{Z}_m$ 上的 NTT - m 是质数

求和引理是保障 NTT 逆变换对应矩阵确为 $n^{-1}\left(\omega_n^{-ij}\right)_{(i,j)\in n\times n}$ 的关键性质. 如果我们还想保留这一点，就必须要求 m 是质数.

为保证 n 在 $\mathbb{Z}_m$ 中存在逆元，必须要求 $\gcd(n,m)=1$. 已经要求 m 是质数，故无需做出额外要求.

综上，逆变换矩阵确为 $n^{-1}\left(\omega_n^{-ij}\right)_{(i,j)\in n\times n}$ 的 n 点 NTT 要求 m 是质数，且 $n\mid \mathrm{ord}_m(a)$. 已经要求 m 是质数，故 $\mathbb{Z}_m$ 中必存在原根，不妨直接取 $\mathbb{Z}_m$ 的一个原根 g 构造 n 次本原单位根 $\omega_n=g^{\frac{\varphi(m)}{n}}=g^{\frac{m-1}{n}}$，这样的构造支持至多 $m-1$ 次单位根的存在，显然是最优的选择.

$\mathbb{Z}_m$ 上的 FFT 与卷积加速 - 998244353

为满足 FFT 对 2^m 次本原单位根的需求，只需选择 $p = k2^m + 1$ 型的奇质数 p，就可以在 $\mathbb{Z}_p$ 上支持规模至多为 2^m 的 NTT/FFT 及其逆变换运行.

$998244353 = 2^{23} \times 7 \times 17 + 1$，同时是一个质数，在 `int` 型中的单次加减操作不会溢出，是 OI/XCPC 计数题中不可多得的优秀模数[12].

结合前述关于 $\mathbb{Z}_p[x]$ 上多点插值的讨论，通过 NTT/FFT 加速卷积运算的正确性得到完整证明.

[12] 网传此模数由 UOJ 站长 vfleaking 提出并推广. 在所有需要取模的题目中使用该模数，可使选手无法通过模数判断题目的做法.

$\mathbb{Z}_m$ 上的 NTT - 对一般的 m

当 m 不一定是质数时，NTT 逆变换不能表示为 $n^{-1}\left(\omega_n^{-ij}\right)_{(i,j)\in n\times n}$ 的形式. 但这并不代表 NTT 不可逆. 因此，尽管不再实用，一般 $\mathbb{Z}_m$ 上的 NTT 变换仍有讨论价值，即研究 $\left(\omega_n^{ij}\right)_{(i,j)\in n\times n}$ 的可逆性.

我们找到一篇有关该问题的参考文献 [7, section 3 and appendix B]，但尚不确定其证明的正确性. 友情提示读者：$\mathbb{Z}_m$ 不是域，甚至不是整环，故线性空间中矩阵的性质不能直接应用于 $\mathbb{Z}_m$ 上的矩阵，讨论可逆性时还需小心谨慎.

从分析到代数

后续数学内容导读 NTT 原理虽已非常" 数学"，但也只是 DFT 在有限域上的一个实例. 本节往后，我们要尝试为多项式系数位于复数域 $\mathbb{C}$ 上的 DFT、位于模 p 剩余类域 $\mathbb{Z}_p$ 上的 NTT 及它们的快速算法 FFT 建立一个统一的数学框架. 这需要我们剖析求值插值的基本原理，提炼出 FFT 算法成立的根本要求.

后续数学内容不再要求掌握. 望同学们在纷繁的定理定义中抓住要旨，窥见抽象数学背后蕴藏的规律. 熟悉高等代数和抽象代数的同学或会对某些内容感到熟悉. 抽象地讨论 FFT 的资料并不多见，后续内容多为主讲人的新进探索，或有谬误，望不吝指正.

目录

求值与插值 - 导言

本节将重新审视已经熟知的多项式，把抽象的、代数的多项式和具体的、分析的多项式函数区分开来. 我们指出，多项式和多项式函数不同但关联紧密，形式幂级数与幂级数亦有此类联系. 这些抽象的讨论将帮助我们剖析多项式求值插值的基本原理.

除常见代数书目（如 [8]），也推荐参考 OI Wiki 的多项式基础 [9] 和 Wikipedia 的形式幂级数 [10].

多项式

设（无穷）序列 $\{a_n\}$ 是一个只有有限个非零项的序列，其元素均在环 R 上. 环 R 上的**多项式环**[13] $R[x]$ 是所有满足上述条件的序列构成的集合以及在它们之间定义的两种运算 $+, \times$ 的合称. 此语境下，我们也将构成 $R[x]$ 的序列称为 R 上的**多项式**. 序列中的元素被称为多项式的系数. **序列间定义的加法和乘法是普通序列升级为多项式的关键.**

多项式环上的加法、乘法的定义已经为大家所熟知. 系数所处的环保证了多项式加法和乘法的良定义，而在这两种运算下，R 上多项式的集合也构成一个环的结构.

注记（群、环、域）

群、环、域都是常见的代数结构，其中的元素在给定运算下封闭，并满足特定的运算性质. 简单来说，环[a]上定义了加法和可能不可逆的、不一定交换的乘法，域上定义了加减乘除所有四则运算. 交换环中的乘法满足交换律. 除环中的所有元素有乘法逆元. 域是交换除环.

[a] 本篇中环的定义包含乘法单位元，即幺环.

[13] 一般语境下讨论的多项式环是在域上的 $K[x]$，而非环上的 $R[x]$.

多项式

习惯上也会将多项式 A 写为 $A(x)=\sum_{k=0}^{n-1}a_kx^k$ 的形式. 其中未定元 x 只是一个符号，仅为方便加、乘法的理解而使用，并无任何含义.

多项式 A 的次数 $\deg A$ 定义为其最高非零项所处的位置下标. 特别的，定义零多项式的次数为 $-\infty$. 对任意多项式 $A,B\in R[x]$，$\deg(A+B)\le\max\{\deg A,\deg B\}$. 当 R 是整环[14]时，A,B 的首项乘积非零，故 $\deg(A*B)=\deg A+\deg B$.

下面额外为多项式定义一种新的运算. 多项式 A 和多项式 B 的复合 $A\circ B$ 定义为

$$A\circ B:=\sum_{k=0}^{+\infty}a_kB^k$$

其中 B^k 代表 k 个多项式 B 的乘积. 习惯上也会将多项式 A 和多项式 B 的复合写为 $A(B(x))=\sum_{k=0}^{+\infty}a_kB^k(x)$ 的形式. 由于多项式的次数有限，复合运算中仅包含了有限次多项式加法和乘法.

[14] 无零因子的交换幺环，稍后介绍.

形式幂级数

形式幂级数定义与多项式的唯一区别是其不要求 $\{a_n\}$ 只有有限项非零. 类似的，R 上的形式幂级数环记为 $R[[x]]$.

由于涉及无限次运算，形式幂级数的复合运算需考虑环 R 上的收敛问题. 为回避此问题，一般规定复合右侧函数的常数项须为零.

注记

DFT, NTT 与多项式环 DFT/FFT 加速的多项式卷积在复数域 $\mathbb{C}$ 上的多项式环 $\mathbb{C}[x]$ 中进行，NTT/FFT 加速的多项式卷积在模 p 剩余类域 $\mathbb{Z}_p$ 上的多项式环 $\mathbb{Z}_p[x]$ 中进行.

带余除法

整环是无零因子的交换幺环. 所谓无零因子，即环中任意元素 a, b 满足 $ab = 0 \implies a = 0 \vee b = 0$. 所有域都是整环. 当 R 是整环时，$R[x]$ 也是整环[15]. 整环上可以定义整除相关理论.

定义有带余除法的环被称为 Euclid 整环. 域上的多项式环都是 Euclid 整环 [8, 第 7 章第 2 节定理 3, p. 11]. 值得注意的是，R 的 Euclid 性不可传递至 $R[x]$[16].

如在带余除法中保证除式是首一多项式[17]，则带余除法的良定义和进行过程也均可在整环上实现.

[15] 可用前述多项式乘积次数公式证明.
[16] 试试在 $\mathbb{Z}[x]$ 上用 $2x + 1$ 去除 $x + 1$.
[17] 首项为 1 的多项式.

多项式函数 / 幂级数

刚刚强调，多项式 / 形式幂级数只是定义了加法和乘法的序列. 现在介绍多项式函数和幂级数. 它们不是序列，而是映射[18].

多项式 A 对应的环 R 上的多项式函数 $A(x)$ 定义为映射[19]

$$A : R \to R, \quad x \mapsto \sum_{k=0}^{+\infty} a_k x^k$$

其中 $\{a_n\}$ 只有有限项非零. 环 R 上的幂级数则无此限制[20].

与多项式 / 形式幂级数不同，这里的 $x \in R$ 代表函数的自变量，是会按环 R 上运算法则参与运算的有意义的变量.

多项式函数的加法和乘法定义为函数的加法和乘法，即

$$A + B : x \mapsto A(x) + B(x)$$

$$A \otimes B : x \mapsto A(x)B(x)$$

[18] 函数和映射几乎是等价名词. 有时函数特指值域包含于复数域 $\mathbb{C}$ 的映射.

[19] 为良定义 x^0，环 R 必须有单位元.

[20] 这里再次涉及环 R 上的收敛问题. 由于实践中只关心形式幂级数的前有限项，后续讨论系数-点值-系数法转换卷积时不需要用到幂级数理论，可以回避.

求值

多项式和多项式函数似乎在许多情况下有着平行的关系. 下面介绍一个较直观的结论.

若环 R 是交换环，则环 R 上每个多项式都唯一确定一个 R 上的多项式函数，且该映射保持加法和乘法运算[21]. 形式化地可以记为

$$(A+B)(x) = A(x) + B(x)$$
$$(A*B)(x) = A(x)B(x)$$

可用于证明上述结论的定理证明见 [8, 第 7 章第 1 节定理 4, p. 7][22]. 由于多项式的复合由多项式加法和乘法组成，而多项式函数的复合（定义为函数的复合）亦可在环上平行地展开，故上述映射也保持了复合运算.

多项式和多项式函数的这一关系为多项式在任意点的求值操作提供了理论基础. 在多项式函数的 t 处求值也被称为用 t 代入多项式 A，记为 $A(t)$.

[21] 这种保持结构不变的映射被称为同态（homomorphism）.

[22] 其证明了域上一元多项式环的通用性质. 仿照该证明应可证明环上的版本，从而证明这一同态关系.

插值

需要注意的是，前述结论的逆命题不一定成立，即环 R 上的一个多项式函数不一定与 R 上多项式一一对应. 一个经典反例是有限域上多项式环 $\mathbb{Z}_p[x]$ 上的 $(x+1)^p \equiv x^p + 1 \pmod{p}$. 我们指出其成立的一个充分条件是 R 是无限整环[23].

该逆命题的本质是通过多项式函数的所有函数值反过来确定多项式（系数）的过程. 而如果在确定时只使用一部分函数值，就是所谓的多点插值过程. 很多时候待求多项式的次数是已知的，这在相当程度上缩小了待定多项式的范围. 我们指出，只要 R 是整环，就可以通过多项式任意 n 个不同位置的点值确定一个至多 $n-1$ 次的多项式.

证明的关键是用带余除法[24]讨论多项式根与其一次因式的关系，即多项式余式定理或小 Bézout 定理 [8, 第 7 章第 6 节定理 6, p. 35].

需要强调，上述结论只能说明，只有那些确实可通过多项式 n 点求值得到的点值组，才能与多项式建立保持加法和乘法运算的双射. 对于任取的 R 中的一个 n 元组，则无法确定是否存在与之对应的多项式.

[23] 该定理是下方高亮定理的一个自然的推论.

[24] 由于一次因式均为首一多项式，可以在整环上对其使用带余除法.

求值与插值的线性表示

到这里，多项式的求值与插值的线性表示已经呼之欲出了. 若将交换环 R 上的至多 $n-1$ 次的多项式看做模[25] R^n 中的一个向量，则多项式多点求值的过程就是 R^n 上的一个线性变换（记为 V），其矩阵表示正是 Vandermonde 矩阵. 由先前对求值的讨论，我们还知道其保持模 R^n 上的加法运算，并将结果不超过 $n-1$ 次的卷积运算转化为 R^n 上的逐项乘法.

若将对 R 的要求加强到域，注意到 Vandermonde 行列式在参数互不相同时非零，故此时 V 可逆. 事实上，Vandermonde 矩阵求逆的过程就是 Lagrange 插值，从而给出了将任意 R^n 中向量还原为其对应多项式的构造方法[26]. 这与数论中中国剩余定理的构造有异曲同工之妙 [11, section: A perspective from linear algebra].

[25] 模是定义在环上的 “线性空间”.

[26] Lagrange 插值的构造用到了除法，且行列式非零推出矩阵可逆仅在域上的线性空间中适用，因此必须要求 R 是域.

求值与插值 - 小结

本节的核心是多项式和多项式函数的区别与联系，两个方向的”确定”分别给出了多项式多点求值和多点插值的理论基础.

由于多项式和多项式函数这种若即若离的关系，往往在记号上也有意无意地混淆了它们，某些情况下加大了区分的难度. 本篇使用的记号体系将尽量用单个字母 A 表示多项式，用 $A(x)$ 表示多项式 A 对应的多项式函数，用 $A(t)$ 表示用 t 代入多项式 A，同时在需要区分的符号前加上适当的名词.

目录

环上的 DFT - 导言

前面讨论了在多项式任意点处求值插值的基本原理，但 DFT/FFT 的运行只需在单位根处求值和插值. 本节将进一步放宽对多项式环的限制，介绍定义在有主要单位根的环上的一般的 DFT 及其快速算法 FFT.

本节内容主要参考了 [12, section 2, pp. 983–984] 和 [13].

环上的单位根 - 两个定义

定义环 R 上的 n 次主要单位根（principal n-th root of unity）ω_n 是满足如下条件的环 R 中的一个元素：

$$\omega_n^n = 1; \quad \sum_{k=0}^{n-1} \omega_n^{ik} = 0, \quad i = 1, 2, \ldots, n-1$$

由定义立得 $\sum_{k=0}^{n-1} \omega_n^{ik} = [n \mid i]n$，可见主要单位根的定义即求和引理本身.

定义环 R 上的 n 次本原单位根（primitive n-th root of unity）ω_n 是满足如下条件的环 R 中的一个元素：

$$\omega_n^n = 1; \quad \omega_n^k \neq 1, \quad k = 1, 2, \ldots, n-1$$

本原单位根在许多情况下与主要单位根等价，但亦非完全相同.

环上的单位根 - 区别与联系

命题 4.1

若环 R 是整环，则环上的本原单位根也是一个主要单位根.

证明

对任意正整数 $i \in [n]$，令 $S = \sum_{k=0}^{n} \omega_n^{ik}$，则 $(1-\omega_n^i)S = S - \omega_n^i S = 1 - \omega_n^n = 0$，由 R 是整环及 $\omega_n^i = 0$ 得 $S = 0$. □

命题 4.2

若环 R 的特征[a] $\operatorname{char} R$ 满足 $\operatorname{char} R \nmid n$，则环上的主要单位根也是一个本原单位根.

[a]使得 $m1 = \sum_{k=0}^{m-1} 1 = 0$ 的最小正整数 m. 不存在则记为 0. $\operatorname{char} \mathbb{C} = 0$, $\operatorname{char} \mathbb{Z}_p = p$. 可以证明域的特征一定是 0 或一质数 [8, 第 7 章第 11 节定理 3，p. 70].

证明

反证. 若存在一正整数 $i \in [n]$ 使得 n 次主要单位根 ω_n 的某一幂次 $\omega_n^i = 1$，则 $\sum_{k=0}^{n-1} \omega_n^{ik} = \sum_{k=0}^{n-1} 1^k = n1 \neq 0$，与主要单位根定义矛盾. □

环上的单位根 - 其它性质

分别根据定义和数论中的 Bézout 定理，容易证明主要单位根和本原单位根的消去引理：设 ω_n 是环上的 n 次主要（本原）单位根，则 ω_n^k 是环上的 $\frac{n}{\gcd(n,k)}$ 次主要（本原）单位根.

注意到 $\left(\omega_{2n}^n\right)^2 = 1 \implies (\omega_{2n}^n + 1)(\omega_{2n}^n - 1) = 0$，故整环上的 $2n$ 次主要单位根或本原单位根 ω_{2n} 均满足 $\omega_{2n}^n = -1$.

环上的 n 次主要单位根或本原单位根 ω_n 都存在逆元 $\omega_n^{-1} = \omega_n^{n-1}$.

环上的 DFT

设 ω_n 是环 R 上的一个 n 次主要单位根，其对应的 R^n 上的 n 点 DFT 定义为线性映射（或矩阵）

$$F = \left(\omega_n^{ij}\right)_{(i,j)\in n\times n} = \begin{pmatrix} 1 & 1 & \dots & 1 \\ 1 & \omega_n & \dots & \omega_n^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_n^{n-1} & \dots & \omega_n^{(n-1)(n-1)} \end{pmatrix}$$

若 $n1$ 在环 R 中存在乘法逆元，则 R^n 上的 DFT 可逆，其逆映射为[27]

$$F^{-1} = (n1)^{-1}\left(\omega_n^{-ij}\right)_{(i,j)\in n\times n} = (n1)^{-1}\begin{pmatrix} 1 & 1 & \dots & 1 \\ 1 & \omega_n^{-1} & \dots & \omega_n^{-(n-1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_n^{-(n-1)} & \dots & \omega_n^{-(n-1)(n-1)} \end{pmatrix}$$

[27] 证明使用主要单位根的定义（求和引理）即可.

环上的 FFT

除无法在任意环上使用 $\omega_{2n}^n = -1$，现有的抽象已足够让我们写出与之前类似的推导 FFT 的过程.

设 R 是任意环，ω_{2n} 是环 R 上的一个 $2n$ 次主要单位根. 由主要单位根的消去引理，ω_{2n}^2 是环 R 上的 $\frac{2n}{\gcd(2n,2)} = n$ 次主要单位根. 方便起见，记 $\omega_n = \omega_{2n}^2$.

考虑对模 R^{2n} 中向量 $\mathbf{a} = (a_0, a_1, \ldots, a_{2n-1})^T$ 做 ω_{2n} 对应的 $2n$ 点 DFT 变换，得到向量 $\hat{\mathbf{a}} = (\hat{a}_0, \hat{a}_1, \ldots, \hat{a}_{2n-1})^T$，其中 $\hat{a}_i = \sum_{k=0}^{2n-1} a_k \omega_{2n}^{ik}$. 将其奇偶分项为 $\hat{a}_i = A_0(i) + A_1(i)\omega_{2n}^i$，其中

$$A_0(i) := \sum_{k=0}^{n-1} a_{2k}\omega_n^{ik} = \sum_{k=0}^{n-1} a_{2k}\omega_{2n}^{2ik}$$

$$A_1(i) := \sum_{k=0}^{n-1} a_{2k+1}\omega_n^{ik} = \sum_{k=0}^{n-1} a_{2k+1}\omega_{2n}^{2ik}$$

环上的 FFT

注意到 $A_0(i) = A_0(n+i)$，$A_1(i) = A_1(n+i)$，故

$$\begin{aligned} \hat{a}_i &= A_0(i) + A_1(i)\omega_{2n}^i \\ \hat{a}_{n+i} &= A_0(i) + A_1(i)\omega_{2n}^{n+i}, \end{aligned} \qquad i \in [n]$$

而 $A_0(i),\ i \in [n]$ 的计算即计算向量 $\mathbf{a_0} = (a_0, a_2, \ldots, a_{2n-2})$ 对应于 ω_n 的 n 点 DFT 的过程；$A_1(i),\ i \in [n]$ 的计算即计算向量 $\mathbf{a_1} = (a_1, a_3, \ldots, a_{2n-1})$ 对应于 ω_n 的 n 点 DFT 的过程.

因此，令 $n = 2^m$，则只要环 R 上存在 2^m 次主要单位根，如上形式的 FFT 递归算法就可在 $O(n \log n)$ 的时间复杂度内快速计算 R^n 上的 n 点 DFT. 而若 $2^k 1,\ 0 \le k \le m$ 在 R 内均存在逆元，R^n 上的 n 点 IDFT 也可类似地快速计算.

目录

循环卷积与卷积定理 - 导言

上一节中，我们建立了在有主要单位根的环上的 DFT 及其快速算法 FFT 的相关理论，但由于放宽了对多项式环 $R[x]$ 的限制，多项式的求值插值理论无法在此直接得到应用，环上 DFT 加速多项式卷积的理论尚需重新构建.

在求值与插值部分已经介绍求值变换 V 在 R^n 上的线性表示，其保持 R^n 上的加法运算，并将规模不超过 n 的两向量的卷积转化为 R^n 上的逐项乘法. 本节中，我们来证明 DFT 变换 F 也满足同样的性质，且将这一结果推广到卷积规模超过 n 时的情况.

循环卷积

设 R 是交换环，我们记 $A \otimes B$ 代表模 R^n 上两个向量的逐项乘法.

设 F 是 R^n 上主要单位根 ω_n 对应的 n 点可逆 DFT 变换，A, B 是 R^n 中两个任意取定的向量. 显然 $F(A+B) = FA + FB$，故 DFT 变换保持 R^n 上的加法运算. 现在的主要问题是，求值与插值部分计算卷积的方法 $F^{-1}(FA \otimes FB)$ 在放宽环限制的 R^n 上进行时，将会得到什么结果？

令

$$\hat{\mathbf{a}} = F\mathbf{a}, \quad \hat{\boldsymbol{b}} = F\boldsymbol{b}$$
$$\hat{\boldsymbol{c}} = \hat{\mathbf{a}} \otimes \hat{\boldsymbol{b}}, \quad \boldsymbol{c} = F^{-1}\hat{\boldsymbol{c}}$$

则

$$\hat{a}_i = \sum_{j=0}^{n-1} a_j \omega_n^{ij}, \quad \hat{b}_i = \sum_{k=0}^{n-1} b_k \omega_n^{ik}$$

循环卷积

$$\hat{c}_i = \hat{a}_i \hat{b}_i = \sum_{j=0}^{n-1} a_j \omega_n^{ij} \sum_{k=0}^{n-1} b_k \omega_n^{ik} = \sum_{j=0}^{n-1} \sum_{k=0}^{n-1} a_j b_k \omega_n^{i(j+k)}$$

$$c_i = \sum_{t=0}^{n-1} \omega_n^{-it} \hat{c}_t = \sum_{t=0}^{n-1} \omega_n^{-it} \sum_{j=0}^{n-1} \sum_{k=0}^{n-1} a_j b_k \omega_n^{t(j+k)} = \sum_{j=0}^{n-1} \sum_{k=0}^{n-1} a_j b_k \sum_{t=0}^{n-1} \omega_n^{t(j+k-i)}$$

对最里侧的求和使用主要单位根的定义（求和引理），就有

$$c_i = \sum_{j=0}^{n-1} \sum_{k=0}^{n-1} a_j b_k [n \mid j + k - i] = \sum_{j+k \equiv i \bmod n} a_j b_k$$

这便是所谓的循环卷积. 以后将 R^n 中向量 $\boldsymbol{a}$ 和 $\boldsymbol{b}$ 的循环卷积记为 $\boldsymbol{a} * \boldsymbol{b}$. 形象地来看，循环卷积中，次数超过 $n-1$ 的卷积项被模意义地地叠加到了以次数为 0 为始的项上.

卷积定理

前述讨论已经证明[28]

$$F(A * B) = FA \otimes FB$$

这便是一般交换环上 DFT 变换的（循环）卷积定理.

我们指出，当要求所作变换可逆时，卷积定理反过来也要求所作变换是一类似 DFT 映射的变换[29].

习题 4.1

尝试将任意序列 DFT 两次，观察结果. 证明你的结论.

习题 4.2

求 DFT 矩阵的行列式. 尽可能缩小可行解范围.

[28] 虽然前面用到了 F 可逆的要求，但该定理在 F 不可逆时也成立. 只需类似地验证两边相等即可.

[29] 具体来说，该变换只能是 DFT 矩阵的某个行置换. 在 $\mathbb{C}^n$ 上的证明可参见 [14]，主讲人目前正在研究整环上的版本，欢迎讨论.

目录

多项式全家桶 - 序言

- 有哪些？
  - 求逆、开根、对数、指数、快速幂、复合等
- 咋推的？
  - 求解思路几乎都是倍增，时间复杂度几乎都是大常数 $O(n \log n)$
  - Newton 迭代法是推导全家桶迭代公式的通法
  - 严格化需要进一步的形式幂级数理论，主讲人不会
- 有啥用？
  - 当你一波操作化出生成函数发现不会求系数
- 怎么讲？
  - 受限于篇幅和主讲人能力，我们讲不完
  - 只讲求逆和对数，其余请左转 OI Wiki

多项式求逆

给定一多项式 $A(x)$，求解满足 $A(x)B(x) = 1 \pmod{x^n}$ 的多项式 $B(x)$.

多项式逆元存在的充分必要条件是其常数项非零（这是因为边界条件 $b_0 = \frac{1}{a_0}$），若存在则在模意义下一定唯一. 这结论可直接由下述求解方法得到. 不失一般性，只研究 $A(x)$ 的次数至多为奇数 $2n-1$ 的情况. 设

$$A(x) = A_0(x) + x^n A_1(x)$$
$$B(x) = B_0(x) + x^n B_1(x)$$

考虑递归地在已知

$$A(x)B_0(x) = A_0(x)B_0(x) = 1 \pmod{x^n}$$

的基础上求解 $A(x)$ 的逆元. 先指出该方法的时间复杂度为

$$T(n) = T(\frac{n}{2}) + O(n \log n) = O(n \log n)$$

多项式求逆

以下简记 $A(x)$ 为 A，其它多项式同理. 注意到

$$\begin{cases} AB_0 = 1 \pmod{x^n} \\ AB = 1 \pmod{x^n} \end{cases} \implies A(B - B_0) = 0 \pmod{x^n}$$

由于 $A \bmod x^n$ 非零，故

$$B - B_0 = 0 \pmod{x^n}$$

这也说明，多项式逆元在模不同 x^n 下的前缀保持一致.

两边平方得

$$B^2 - 2BB_0 + B_0^2 = 0 \pmod{x^{2n}}$$

两侧同乘 A 并移项得

$$B = 2B_0 - AB_0^2 \pmod{x^{2n}}$$

多项式求逆 - 实现

```
Poly inv(const Poly &A){
    ll n=A.len(); Poly B(1); B[0]=inv(A[0]);
    for(ll hf=1;hf<n;hf<<=1){
        B=B*2-B*B*A.subpoly(0,hf*2); B.resize(hf*2);
    } B.resize(n);
    return B;
}
```

- 常数巨大的写法，仅作演示，请勿学习.
- 正确的写法是在 DFT 后的点值上操作多项式，请小心实现封装.

多项式 ln

给定一多项式 $A(x)$，求解满足 $B(x) = \ln A(x) \pmod{x^n}$ 的多项式 $B(x)$.

次数为 $+\infty$ 的形式幂级数的 ln 存在的充分必要条件为其常数项非零（这是因为边界条件 $b_0 = \ln a_0$），同样一旦存在则唯一. 注意到仅整数 $a_0 = 1$ 时，$\ln a_0$ 可取得整数，故合理的 a_0 只能是 1.

推导是容易的. 方程两侧同时求导得

$$B'(x) = \frac{A'(x)}{A(x)} \pmod{x^{n-1}}$$

两侧再积分得

$$B(x) = \int \frac{A'(x)}{A(x)} \mathrm{d}x + C \pmod{x^n}$$

其中 $C = \ln a_0$. 多项式求逆、求导、积分即可. 时间复杂度 $O(n \log n)$.

多项式 ln - 实现

```
Poly drv(Poly A){ // derivative
    for(ll i=0;i<A.len();i++) A[i]=(i+1)*A[i+1]_;
    A.pop_back(); return A;
}
Poly itg(Poly A,ll c){ // integral
    A.push_back(0); for(ll i=A.len();i>=1;i--) A[i]=A[i-1]*inv(i)_;
    A[0]=c; return A;
}
Poly ln(const Poly &A){
    return itg((drv(A)*inv(A)).subpoly(0,A.len()-1),0/*log(A[0])*/);
}
```

Acknowledgements

- 感谢 keke_046 学长教授 FFT、集合幂级数与生成函数. 微言大义，博大精深，至今仍在消化.
- 感谢队友 ItzDesert 提供位运算典题一道并提供内容编排建议.

题单 I

主讲人练题少，仅供参考.

模板

- 洛谷 P3803 【模板】多项式乘法（FFT）
- 洛谷 P6091 【模板】原根
- 洛谷 P4238 【模板】多项式乘法逆
- 洛谷 P5205 【模板】多项式开根
- 洛谷 P4725 【模板】多项式对数函数（多项式 ln）
- 洛谷 P4726 【模板】多项式指数函数（多项式 exp）
- 洛谷 P5245 【模板】多项式快速幂

大整数乘法

- 洛谷 P1919 【模板】A*B Problem 升级版（FFT 快速傅里叶变换）

基础计数

- SPOJ-TSUM Triple Sums

题单 II

- BZOJ3513-MUTC2013 Idiots
  上面两道题都是 OGF 消序，较 EGF 消序困难. 一般的方法是使用 Polya 计数原理.

位运算

- ABC291G OR Sum
  也是滤波器的应用.

分治 FFT

- 百度之星 2023 初赛第二场 T8
  容斥后需要计算若干一次多项式乘积，分治 NTT 即可.
- 百度之星 2023 初赛第一场 T6
  求 $\sum_{i>j} \frac{a_i}{a_i+a_j}$. 值域次数化后 CDQ 处理偏序.
- 洛谷 P4721 【模板】分治 FFT
  CDQ 偏序化处理前后项依赖. 也可解生成函数方程再多项式求逆.

题单 III

字符串

- ABC196F Substring 2
- 洛谷 P4173 残缺的字符串
  带单字符通配符的字符串匹配. 体会设定字符串距离函数的方法.

全家桶

- 洛谷 P4721 【模板】分治 FFT
  体验生成函数简化复杂数列递推的威力.
- 洛谷 P4389 付公主的背包
  解决此问题的方法也可用于分拆数计算.
- 洛谷 P4841 【集训队作业 2013】城市规划
  EGF 消序划分典题. 体验生成函数风格的计数方法.
- LOJ6538 烷基计数 - 加强版 - 加强版
  对生成函数使用的 Polya 计数.
  Polya ex. 苯环碳接 $-\mathrm{C_4H_9}$ 同分异构体计数

题单 IV

其它

- QOJ5748-UCUP2023-Stage7-K Determinant, or...?
  看似人畜无害的 $(a_{i \operatorname{or} j})_{(i,j)\in n\times n}$ 行列式求值，解法的背后却潜藏 FMT 的思想. 此类行列式与各类反演、卷积有密切联系，我们给出一篇研究了其与偏序集上反演的关系的参考文献 [15]，欢迎讨论.

参考文献 I

[1] OI-Wiki, 拉格朗日插值, https://oi-wiki.org/math/numerical/lagrange/.

[2] 张筑生, "数学分析新讲（重排本）（第二册)," in 2nd ed. 北京: 北京大学出版社, 2021, ch. 微分方程初步, pp. 256–262, sec. 4.b and 4.c, ISBN: 978-7-301-32337-3.

[3] 杨树森, 三角函数的严格定义, https://zhuanlan.zhihu.com/p/58814328/, 2023.

[4] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, "算法导论（原书第三版)," in trans. by 殷建平 *et al.* 北京: 机械工业出版社, 2013, ch. 多项式与快速傅里叶变换, ISBN: 978-7-111-40701-0.

[5] OI-Wiki, 离散对数, https://oi-wiki.org/math/number-theory/discrete-logarithm/.

[6] OI-Wiki, 原根, https://oi-wiki.org/math/number-theory/primitive-root/.

参考文献 II

[7] R. Agarwal and C. Burrus, "Fast convolution using fermat number transforms with applications to digital filtering," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 22, no. 2, pp. 87–97, 1974. DOI: 10.1109/TASSP.1974.1162555.

[8] 丘维声, "高等代数 下册," in 3rd ed. 北京: 高等教育出版社, 2015, ch. 多项式环, ISBN: 978-7-04-042235-1.

[9] OI-Wiki, 多项式与生成函数简介, https://oi-wiki.org/math/poly/intro/.

[10] Wikipedia, *Formal power series*, https://en.wikipedia.org/wiki/Formal_power_series.

[11] Wikipedia, *Lagrange polynomial*, https://en.wikipedia.org/wiki/Lagrange_polynomial.

参考文献 III

[12] M. Fürer, "Faster integer multiplication," *SIAM Journal on Computing*, vol. 39, no. 3, pp. 979–1005, 2009. DOI: 10.1137/070711761. eprint: https://doi.org/10.1137/070711761.

[13] Wikipedia, *Discrete fourier transform over a ring*, https://en.wikipedia.org/wiki/Discrete_Fourier_transform_over_a_ring.

[14] I. Baraquin and N. Ratier, "Uniqueness of the discrete fourier transform," *Signal Processing*, vol. 209, p. 109 041, 2023.

[15] H. S. Wilf, "Hadamard determinants möbius functions, and the chromatic number of a graph,", 1968, Available: https://www.ams.org/journals/bull/1968-74-05/S0002-9904-1968-12104-4/.