数学提高

王彦淞

天津大学

2023 年 8 月 5 日

目录

目录

引理 1

$\forall a, b, c \in \mathbb{Z}^+, \lfloor \frac{\lfloor \frac{a}{b} \rfloor}{c} \rfloor = \lfloor \frac{a}{bc} \rfloor$

引理 1

$\forall a, b, c \in \mathbb{Z}^+, \lfloor \frac{\lfloor \frac{a}{b} \rfloor}{c} \rfloor = \lfloor \frac{a}{bc} \rfloor$

引理 2

$\forall n \in \mathbb{N}^+, \|\{\lfloor \frac{n}{d} \rfloor | d \in \mathbb{Z}^+, d \leq n\}\| \leq \lfloor 2\sqrt{n} \rfloor$

整除分块

求 $\Sigma_{i=1}^{n} f(i) * g(\lfloor \frac{n}{i} \rfloor)$, 多组询问 T 1e4,n 1e6
f 和 g 均可 O(1) 求出.

整除分块

求 $\Sigma_{i=1}^{n} f(i) * g(\lfloor \frac{n}{i} \rfloor)$, 多组询问 T 1e4,n 1e6
f 和 g 均可 O(1) 求出.

```
int ans=0;
for(int l = 1;l <= n;l = r + 1) {
    r = n / (n / l);
    ans+=(sum_f(r)-sum_f(l-1))*(r-l+1)*g(n/l);
}
```

目录

莫比乌斯函数

$$\mu(n) = \begin{cases} 1 & n = 1 \\ (-1)^k & n = p_1 p_2 \cdots p_k \\ 0 & \text{其他} \end{cases}$$

莫比乌斯函数

$$\mu(n) = \begin{cases} 1 & n = 1 \\ (-1)^k & n = p_1 p_2 \cdots p_k \\ 0 & \text{其他} \end{cases}$$

性质

$$\sum_{d|n} \mu(d) = \lfloor \frac{1}{n} \rfloor = \begin{cases} 1 & n = 1 \\ 0 & n > 1 \end{cases}$$

莫比乌斯反演

$f(n)$ 和 $g(n)$ 是定义在正整数集合上的两个函数，若

$$f(n) = \sum_{d|n} g(d)$$

则

$$g(n) = \sum_{d|n} \mu(d) f(\frac{n}{d})$$

反之亦然.

Problem b

求 $\Sigma_{i=a}^{b}\Sigma_{j=c}^{d}[gcd(i,j)=k]$，多组询问，数据范围均为 5e4

约数个数和

求$\Sigma_{i=1}^{n}\Sigma_{j=1}^{m}d(ij)$**，**

d表示约数个数，多组询问，数据范围均为 $5e4$

约数个数和

求$\Sigma_{i=1}^{n}\Sigma_{j=1}^{m}d(ij)$，

d表示约数个数，多组询问，数据范围均为 $5e4$

$$d(ij)=\sum_{x|i}\sum_{y|j}[gcd(x,y)=1]$$

约数个数和

求$\Sigma_{i=1}^{n}\Sigma_{j=1}^{m}d(ij)$，

d表示约数个数，多组询问，数据范围均为 $5e4$

$$d(ij)=\sum_{x|i}\sum_{y|j}[gcd(x,y)=1]$$

$$\sum_{d=1}^{n}\mu(d)(\sum_{x=1}^{\lfloor\frac{n}{d}\rfloor}\lfloor\frac{n}{xd}\rfloor)(\sum_{y=1}^{\lfloor\frac{m}{d}\rfloor}\lfloor\frac{m}{yd}\rfloor)$$

中间一坨前缀和预处理，整除分块即可

目录

常见积性函数

μ: 莫比乌斯函数
ϕ: 欧拉函数
ϵ: 单位元函数
1: 恒等函数
id: 幂函数
σ: 约数和函数
d: 约数个数函数（除数函数）

线性筛积性函数

积性函数都是可以线性筛出来的，具体筛法只需要打开线性筛模板，找到三种情况分类讨论即可。

狄利克雷卷积

$$(f * g)(n) := \sum_{d|n} f(d)g\left(\frac{n}{d}\right)$$

狄利克雷卷积

$$(f * g)(n) := \sum_{d|n} f(d)g\left(\frac{n}{d}\right)$$

常见卷积

$$Id_k * 1 = \sigma_k$$

$$\phi * 1 = Id$$

$$1 * 1 = d$$

$$\sigma = Id * 1 = \phi * 1 * 1 = \phi * d$$

$$\mu * 1 = \epsilon$$

$$\mu * Id = \phi$$

$$\mu * d = 1$$

狄利克雷卷积

$$(f*g)(n) := \sum_{d|n} f(d)g\left(\frac{n}{d}\right)$$

常见卷积

$$Id_k * 1 = \sigma_k$$

$$\phi * 1 = Id$$

$$1 * 1 = d$$

$$\sigma = Id * 1 = \phi * 1 * 1 = \phi * d$$

$$\mu * 1 = \epsilon$$

$$\mu * Id = \phi$$

$$\mu * d = 1$$

积性函数的狄利克雷卷积也是积性函数

杜教筛

求积性函数前缀和$\Sigma_{i=1}^{n} f(i)$，$n \leq 10^{10}$

杜教筛

求积性函数前缀和$\Sigma_{i=1}^{n} f(i)$，$n \leq 10^{10}$

构造一个前缀和容易求的积性函数 g，让 $h = f * g$，如果 h 也容易求前缀和，就能够在 $O(n^{\frac{3}{4}})$ 的亚线性时间得到 $\Sigma_{i=1}^{n} f(i)$ 的结果。如果预处理出前 $n^{\frac{2}{3}}$ 项前缀和，可以让时间复杂度变为 $O(n^{\frac{2}{3}})$

$$\begin{aligned}
\sum_{i=1}^{n} h(i) &= \sum_{i=1}^{n} \sum_{d|i} g(d) f(\frac{i}{d}) \\
&= \sum_{d=1}^{n} g(d) \sum_{i=1}^{\lfloor \frac{n}{d} \rfloor} f(i) \\
&= \sum_{d=1}^{n} g(d) S(\lfloor \frac{n}{d} \rfloor) \\
&= S(n) + \sum_{d=2}^{n} g(d) S(\lfloor \frac{n}{d} \rfloor)
\end{aligned}$$

例题

求$\Sigma_{i=1}^{n}\phi(i), n \leq 10^{10}$

例题

求$\Sigma_{i=1}^{n}\phi(i), n \leq 10^{10}$

$$\begin{aligned}
\sum_{i=1}^{n} h(i) &= \sum_{i=1}^{n}\sum_{d|i} g(d) f(\frac{i}{d}) \\
&= \sum_{d=1}^{n} g(d) \sum_{i=1}^{\lfloor \frac{n}{d} \rfloor} f(i) \\
&= \sum_{d=1}^{n} g(d) S(\lfloor \frac{n}{d} \rfloor) \\
&= S(n) + \sum_{d=2}^{n} g(d) S(\lfloor \frac{n}{d} \rfloor)
\end{aligned}$$

例题

求$\Sigma_{i=1}^{n}\phi(i), n \le 10^{10}$

$$\begin{aligned}\sum_{i=1}^{n} h(i) &= \sum_{i=1}^{n}\sum_{d|i} g(d) f(\frac{i}{d}) \\ &= \sum_{d=1}^{n} g(d) \sum_{i=1}^{\lfloor \frac{n}{d} \rfloor} f(i) \\ &= \sum_{d=1}^{n} g(d) S(\lfloor \frac{n}{d} \rfloor) \\ &= S(n) + \sum_{d=2}^{n} g(d) S(\lfloor \frac{n}{d} \rfloor)\end{aligned}$$

$$\sum_{i=1}^{n}\phi(i) = \frac{n\cdot(n+1)}{2} - \sum_{j=2}^{n} I(j) \sum_{i=1}^{n/j}\phi(i)$$

$$\sum_{i=1}^{n}\phi(i)=\frac{n\cdot(n+1)}{2}-\sum_{j=2}^{n}I(j)\sum_{i=1}^{n/j}\phi(i)$$

```
LL dfsphi(int n) {
    if (n <= MAXN)
        return prephi[n];
    if (mpphi.count(n))//如果之前求过，就把值拿来用
        return mpphi[n];
    LL sum = 0;
    for (int i = 2; i <= n; i++) {
        int last = i - 1;
        i = n / (n / i);
        sum = sum + dfsphi(n / i) * (i - last);
        if (i == n)//因为i加1后有可能会爆int
            break;
    }
    return mpphi[n] = n * (n + 1ll) / 2 - sum;
}
```

目录

多项式

多项式英文是 polynomial 而不是 polygen， polygen 是多边形的意思。

卷积

假设有两个多项式 f 和 g，则其卷积 h 定义为 $h[k] = \sum_{i+j=k} f[i]g[j]$。
这种满足 $i+j=k$ 的卷积形式被称为加法卷积。
所以还能有其他形式的卷积，比如满足 $i \oplus j = k$ 的与卷积等等。

卷积的实现

我们知道多项式有两种表示：系数表示与点值表示。系数表示就是最常见的表示法，就是拿一个向量来表示多项式的各项系数。而点值表示则是拿几个点 (x_i, y_i) 来表示。把系数表示的多项式转成点值的过程叫做求值，反之叫插值。

卷积的实现

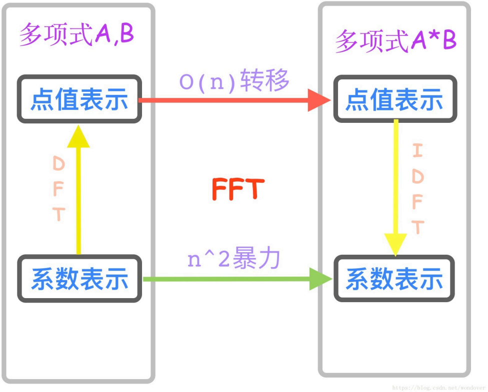
我们知道多项式有两种表示：系数表示与点值表示。系数表示就是最常见的表示法，就是拿一个向量来表示多项式的各项系数。而点值表示则是拿几个点 (x_i, y_i) 来表示。把系数表示的多项式转成点值的过程叫做求值，反之叫插值。

对于多项式加法，点值与系数表示都是 $O(n+m)$（n，m 分别是原多项式的次数），但是点值的话必须选的 x 都一样。

对多项式乘法，系数是 $O(nm)$ 的，而点值还是 $O(n+m)$ 的（只需要把对应点乘上去）。

卷积的实现

由此可见，如果在算乘法时，表示法是点值的，那么就可以很快算出。而如果用最朴素的一个点一个点算，还是 $O(nm)$ 的。如果能快速的将多项式转成点值表示（以及反过来），则可以快速的实现卷积。常用方法是 FFT，其时间复杂度为 $O(n\log n)$ 的。其中，将系数换成点值的过程叫做 DFT，反之叫做 IDFT。



DFT 的实现

单位根及其性质

设 $w_n^k = e^{k\frac{2\pi i}{n}}$，即 n 次方程的 k 次单位根。我们有如下几个极其重要的性质：

（1）相消引理

对于任何整数 $n \geq 0, k \geq 0, d > 0$，有 $w_{dn}^{dk} = w_n^k$

DFT 的实现

单位根及其性质

设 $w_n^k = e^{k\frac{2\pi i}{n}}$，即 n 次方程的 k 次单位根。我们有如下几个极其重要的性质：

（1）相消引理

对于任何整数 $n \geq 0, k \geq 0, d > 0$，有 $w_{dn}^{dk} = w_n^k$

（2）折半引理

如果 $n > 0$ 且为偶数，则 $(w_n^{k+\frac{n}{2}})^2 = (w_n^k)^2$

DFT 的实现

单位根及其性质

设 $w_n^k = e^{k\frac{2\pi i}{n}}$，即 n 次方程的 k 次单位根。我们有如下几个极其重要的性质：

（1）相消引理

对于任何整数 $n \geq 0, k \geq 0, d > 0$，有 $w_{dn}^{dk} = w_n^k$

（2）折半引理

如果 $n > 0$ 且为偶数，则 $(w_n^{k+\frac{n}{2}})^2 = (w_n^k)^2$

（3）求和引理

任意整数 $n \geq 1$ 和不能被 n 整除的非零整数 k，有 $\sum_{i=0}^{n-1}(w_n^k)^i = 0$（整除的则是 n）

基本 FFT

将原来的多项式按下标奇偶分类得到两个新的多项式 $A0$ 与 $A1$。其中 $A0 = a_0x^0 + a_2x^2 + \ldots$ 存偶数位置，$A1 = a_1x^1 + a_3x^3 + \ldots$ 存奇数位置的。

基本 FFT

将原来的多项式按下标奇偶分类得到两个新的多项式 $A0$ 与 $A1$。其中 $A0 = a_0x^0 + a_2x^2 + \ldots$ 存偶数位置，$A1 = a_1x^1 + a_3x^3 + \ldots$ 存奇数位置的。

从而有 $A(w_n^k) = A_0((w_n^k)^2) + (w_n^k)A_1((w_n^k)^2)$

对于 $n = 2^q$，若有 $i < \frac{n}{2}$, 则

$A(w_n^i) = A_0((w_n^i)^2) + w_n^i A_1((w_n^i)^2)$

$A(w_n^{i+\frac{n}{2}}) = A_0((w_n^{i+\frac{n}{2}})^2) - w_n^i A_1((w_n^{i+\frac{n}{2}})^2)$

基本 FFT

从而我们可以通过两个较短的多项式得到一个更大的多项式，按此方法进行分治即可。

基本 FFT

从而我们可以通过两个较短的多项式得到一个更大的多项式，按此方法进行分治即可。
复杂度为 $T(n) = 2 * T(n/2) + O(n)$
由主定理可知为 $O(n \log n)$

优化

按照刚才的方法已经可以写出一个递归版的 FFT 了，但是递归版 FFT 常数较大，通常我们采用非递归版即迭代版 FFT 来实现。
迭代版采用了蝴蝶操作的方式来得到每个数最后分治时会到哪一层，具体做法建议直接粘板。

DFT 的本质

DFT 实际上就是 f 的列向量右乘如下矩阵：

$$\begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & w^1 & w^2 & \dots & w^{(n-1)} \\ 1 & w^2 & w^4 & \dots & w^{2(n-1)} \\ . & . & . & \dots & . \\ 1 & w^{n-1} & w^{2(n-1)} & \dots & w^{(n-1)(n-1)} \end{pmatrix}$$

不难证明其逆矩阵为：

$$\frac{1}{n}\begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & w^{-1} & w^{-2} & \dots & w^{-(n-1)} \\ 1 & w^{-2} & w^{-4} & \dots & w^{-2(n-1)} \\ . & . & . & \dots & . \\ 1 & w^{-(n-1)} & w^{-2(n-1)} & \dots & w^{-(n-1)(n-1)} \end{pmatrix}$$

故 IDFT 也可以直接得到。

NTT

将单位根换成质数原根即可。
通常是 998244353，其原根是 3。

多项式工业

常用的有：多项式求逆，多项式除法，多项式求 Ln，多项式求 Exp，多项式开根等等。
定义很简单：
多项式求逆，对于 x^n，求一个 g 使得 $f * g = 1 \pmod{x^n}$
多项式除法：正常的带余数除法
多项式求 Exp：将 $e^f(x)$ 按照泰勒公式展开得到的式子。
多项式求 Ln 和开根：字面意思。
由于现在是 ACM 不是 OI，所以不会具体讲解这些东西的具体实现，直接粘板即可。

目录

生成函数

形式幂级数

基本可以看成不考虑敛散性的幂级数，即我们只关注其形式而不关注其敛散性。
普通生成函数（OGF）：设有一个无穷（有限）数列 $\{a_n\}$，则其对应的普通生成函数即 $\sum_i a_i x^i$

生成函数

例子

求 Fibonacci 数列的生成函数 F。

生成函数

例子

求 Fibonacci 数列的生成函数 F。

高数入门题。注意到 Fibonacci 数列的通向公式是 $a_n = a_{n-1} + a_{n-2}(n \geq 3)$。所以有

$$F = xF + x^2F + 1$$

移项解得:

$$F = \frac{1}{1 - x - x^2}$$

生成函数

背包计数

物品体积和 $<= 1e5$ 求对于每一种体积的方案数 $\mod 998244353$

生成函数

背包计数

物品体积和 $<=$ 1e5 求对于每一种体积的方案数 $\mod 998244353$

入门生成函数，每个物品的生成函数是 $1+x^{a_i}$，把所有物品的生成函数乘起来即可。注意要采用分治 NTT 的形式进行处理。复杂度 $O(n\log^2 n)$。

生成函数

骨牌问题

有若干种颜色互不相同的骨牌，其中长度为 i 的骨牌有 A[i] 种。
每种颜色的骨牌都可以无限次使用，求不重叠地铺满 n 个长度的方案数。

一个骨牌的生成函数显然可以写出，现在考虑枚举骨牌的数量，则方案数为 $\sum_{k=0} A(x)^k = \frac{1}{1-A(x)}$
多项式求逆即可。

floj 103 不可知圆环

给定一棵树，求包含 1 节点的大小为 $i = 1 - n$ 的连通块的数量。
$n \le 50000$

树上 dp 是 n^2 的，这里就不再赘述。

floj 103 不可知圆环

给定一棵树，求包含 1 节点的大小为 $i = 1 - n$ 的连通块的数量。
$n \leq 50000$

树上 dp 是 n^2 的，这里就不再赘述。
考虑树上 dp 的生成函数，设 f(x) 表示 x 这个点的 OGF，且 x 这个点必须得选（这样是方便之后写成多项式乘积形式）。
显然对于 x 的儿子 y，答案的合并是 $f(x) = f(x) * (g(x) + 1)$ 这里之所以要加 1 是因为可以不选 y 这颗子树。
如果这样直接多项式相乘，复杂度最大可达 $O(n^2 \log n)$ 甚至不如暴力。

floj 103 不可知圆环

给定一棵树，求包含 1 节点的大小为 $i = 1 - n$ 的连通块的数量。
$n \leq 50000$

考虑优化，注意到 f 和子树 size 有关，所以考虑重链剖分优化。
对于两个重链之间，暴力合并，这样一个点最多被暴力更新 $O(\log n)$ 次，这部分的复杂度为 $O(n \log^2 n)$

floj 103 不可知圆环

给定一棵树，求包含 1 节点的大小为 $i = 1 - n$ 的连通块的数量。
$n \leq 50000$

考虑优化，注意到 f 和子树 size 有关，所以考虑重链剖分优化。
对于两个重链之间，暴力合并，这样一个点最多被暴力更新 $O(\log n)$ 次，这部分的复杂度为 $O(n\log^2 n)$
考虑重链中如何处理，假设重链从浅到深的多项式分别是 $f_1, f_2 \ldots f_k$。
那么最终的多项式 $g_1, g_2 \ldots g_k$ 应当是

$$g_{k-1} = f_{k-1} * (f_k + 1), g_{k-2} = f_{k-2} * (g_{k-1} + 1) = f_{k-2} * (f_{k-1} * (f_k + 1) + 1)$$

注意到这可以通过分治 NTT 解决。此部分中，每个点最多有 log 次贡献，再加上分治 NTT 的复杂度，最后为 $O(n\log^3 n)$

生成函数

指数生成函数

指数生成函数（EGF）：设有一个无穷（有限）数列 $\{a_n\}$，则其对应的普通生成函数即 $\sum_i a_i \frac{x^i}{i!}$

注意指数生成函数实际上是带标号的计数，而 OGF 则是简单的拼接。何为带标号？

生成函数

指数生成函数

指数生成函数（EGF）：设有一个无穷（有限）数列 $\{a_n\}$，则其对应的普通生成函数即 $\sum_i a_i \frac{x^i}{i!}$

注意指数生成函数实际上是带标号的计数，而 OGF 则是简单的拼接。
何为带标号？
考虑两个 EGF 的卷积：$f * g = h$

$$h[k] = \sum_{i+j=k} f[i]/i! * g[j]/j! = \sum_{i+j=k} f[i] * g[j] * (k!)/(i!j!)/k!$$

而 $k!/(i!j!) = C(k, i)$，所以此步相当于是进行了标号分配。

生成函数

染色问题

用红蓝绿三种颜色，涂一个长度为 n 的纸条，使得红色和蓝色的个数是偶数，求方案数。

生成函数

染色问题

用红蓝绿三种颜色，涂一个长度为 n 的纸条，使得红色和蓝色的个数是偶数，求方案数。

此题为什么不能直接用 OGF？因为 OGF 只是简单拼接，而这里染色还与位置相关！所以应该是其 EGF 相卷积。

生成函数

EGF 的 Exp

注意到由于 EGF 的卷积相当于是标号，同时又由于 $e^{f(x)} = \sum_i \frac{f(x)^i}{i!}$。这也是一个 EGF 的形式，并且显然可以看出这是任取多个 f 组合所得到的物品，所以 EGF 的 Exp 实质上是若干个单个物品的组合。

生成函数

集训队作业 2013 城市规划

求 n 个点的连通图个数。

生成函数

集训队作业 2013 城市规划

求 n 个点的连通图个数。

注意到一个一般图是若干个联通图组合而成。设一般图的 EGF 为 F，联通图为 G，所以 $e^G = F$，$G = \ln F$。
多项式求 Ln 即可。

二进制卷积

二进制卷积可以分为或（与）卷积，异或卷积，子集卷积等。

FMT

FMT

FMT 即快速莫比乌斯变化，定义也挺简单的。

$$f(S) = \sum_{T \subset S} g(T)$$

实际上就是子集和，同理定义快速莫比乌斯反演，即将快速莫比乌斯变化反演即可

$$\hat{f}(S) = \sum_{T \subset S} (-1)^{|S|-|T|} g(T)$$

或卷积就明显可以通过此来优化。
同理就是与卷积，其变化定义是超集，反之亦然。

子集卷积

不同于一般的或卷积，子集卷积还需要满足 $i\&j=0$，所以多加一维表示大小即可。

k 进制 FWT

k 进制下的异或任然是不进位加法，假设符号仍然是 $\oplus$，并且是数组 A * B = C。
从原理上讲，正变换是我们要做的事情是找到这样一个矩阵 w，使得原数组左乘上这个矩阵后还满足 A * B = C。
逆变换是左乘这个矩阵的逆矩阵。那么考虑是什么意思

$$\sum_{i \oplus j = k} A_i * B_j * w(x, i) * w(x, j) = C_k * w(x, k)$$

对比系数可知

$$w(x, i) * w(x, j) = w(x, k) \ (i \oplus j = k)$$

那么我们的目的即使找到这样的一个矩阵。

FWT

那么这样的性质让我们想到单位根：$w_k^i * w_k^j = w_k^{i \oplus j}$。从而联想到范德蒙德矩阵

$$\begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & w_k^1 & w_k^2 & \dots & w_k^{k-1} \\ 1 & w_k^2 & w_k^4 & \dots & w_k^{2(k-1)} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & w_k^{k-1} & w_k^{2(k-1)} & \dots & w_k^{(k-1)(k-1)} \end{bmatrix}$$

然后我们也知道其逆矩阵

$$\frac{1}{k}\begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & w_k^{-1} & w_k^{-2} & \dots & w_k^{-(k-1)} \\ 1 & w_k^{-2} & w_k^{-4} & \dots & w_k^{-2(k-1)} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & w_k^{-(k-1)} & w_k^{-2(k-1)} & \dots & w_k^{-(k-1)(k-1)} \end{bmatrix}$$

FWT

故我们可以得到 k 进制 FWT 的变化，注意到这个变化和 DFT 是很像的，所以我们可以按照 DFT 的方式来处理 k 进制 FWT。

目录

Thank you! Any Question?