数学常识

dc

May 29, 2014

...

卷积

- 给定 A, B，求

$$C_i = \sum_{j+k=i} A_j B_k$$

。

Binomial

- 给定 n, p，对 $[0, p)$ 中的所有 i，求出满足

$$\binom{n}{k} \equiv i (\text{mod } p)$$

  的 k 的数量。
- $n \leq p^{10}, p = 51061$（p 是质数）。

Binomial

- Lucas 定理：
$$\binom{n}{k} \equiv \prod \binom{n_i}{k_i} \pmod p$$
。

Binomial

- Lucas 定理：
$$\binom{n}{k} \equiv \prod \binom{n_i}{k_i} \pmod p$$
。
- $f(i,j)$ 表示 k 的前 i 位已固定，当前组合数之积为 j 的方案数。
$$f(i+1,j') = \sum_{jl=j'} f(i,j)b(i,l)$$
，其中 $b(i,l)$ 表示 $\binom{n_i}{k_i} = l$ 的 k_i 数量。

Binomial

- 求离散对数可得

$$f'(i+1,j') = \sum_{j+l=j'} f'(i,j)b'(i,l)$$

。

- FFT 完成每次转移，$O(P\log N)$。

城市规划

- 统计有 N 个点的带标号联通无向图的数量。
- $N \le 10^3, 10^5$。

城市规划

- n 个点的带标号无向图共有 $2^{\binom{n}{2}}$ 种。
- 设 F_n 为 n 个点的联通无向图数量，枚举与 1 相连的点数，可得

$$\begin{aligned} F_n &= 2^{\binom{n}{2}} - \sum_{i=1}^{n-1} \binom{n-1}{i-1} F_i 2^{\binom{n-i}{2}} \\ &= C_1(n) - \sum_{i=1}^{n-1} G(i) C_2(n-i) \end{aligned}$$

，其中 $G(i)$ 与 F_i 有关。

城市规划

- 无法直接用卷积计算 F。
- 对时间分治：计算 $F_1 \mathinner{\ldotp\ldotp} F_{n/2}$ 对 $F_{n/2+1} \mathinner{\ldotp\ldotp} F_n$ 的贡献。这可以用 FFT 计算。
- $O(N \log^2 N)$。

Nim

- 定义一种类似卷积的新运算 $\oplus$：

$$(X \oplus Y)_i = \sum_{j \oplus k = i} X_j Y_k$$

。

- 给定 X、k，求 $X \oplus X \oplus ... X$（k 个 X）。
- $n = len(X) \leq 10^5, k \leq 10^9$。

Nim

- 利用迭代快速幂可以把问题转化为求 $\lg k$ 次 “卷积”，所以我们考虑优化单次运算。
- 注意到

$$(a,b)\oplus(c,d)=(ac+bd,ad+bc)=((a+b)(c+d)-(ad+bc),ad+bc)$$

，我们可以将计算中的四次乘法减少到三次；
这个式子可以推广到 a,b,c,d 是 2^k 维向量的情况，由此得到一个类似 Karatsuba 的分治算法，时间复杂度为 $O(N^{\log 3})$。

- 设 $A=(b+a)(d+c), B=(b-a)(d-c)$，于是

$$(ac+bd, ad+bc)=((A+B)/2, (A-B)/2)$$

。得到 $O(N\log N\log K)$ 算法。

Nim

- 设 $A = (b+a)(d+c), B = (b-a)(d-c)$，于是

$$(ac+bd, ad+bc) = ((A+B)/2, (A-B)/2)$$

  。得到 $O(N \log N \log K)$ 算法。

- 我们希望找到一种可逆变换，使得变换前序列的“卷积”运算对应于将变换后的序列按位相乘。

Nim

- 考虑变换 $T[(a,b)] = (b-a, b+a)$。
  注意到 $T[(ac+bd, ad+bc)] = T[(a,b)]T[(c,d)]$[1]，这是一个合法的变换。
- 所以我们只要求出 $T[X]$，并对每个元素快速幂就可以了。
- 复杂度 $O(N(\log N + \log K))$。

[1]我们用 ab 表示两个向量按位相乘得到的序列。

Nim

- 在卷积的定义中，将异或改成其它位运算也是可以的。

Optimize

- 定义
$$(A \times B)_i = \sum_{j+k \equiv i \pmod{N}} A_j B_k$$
。
- 给定 A, B, n, c，求 $A \times B \times B \times B..$（$c$ 个 B）。
- 输出每一项模 $n+1$ 的值。
- $n \leq 10^5$，且 $n+1$ 为质数；$n = 2^k$ / n 只包含 $2, 3, 5, 7$ 作为约数。

- 将 A, B 和 $A \times B$ 看作 n 阶多项式，考虑某个 n 次单位根 ω 在这三个多项式上的值。

$$
\begin{aligned}
(A \times B)(\omega) &= \sum_{i=0}^{n-1} (A \times B)_i \omega^i \\
&= \sum_{i=0}^{n-1} \Big(\sum_{j+k=i} A_j B_k + \sum_{j+k=i+n} A_j B_k \Big) \omega^i \\
&= \sum_{i=0}^{n-1} \Big(\sum_{j+k=i} A_j B_k \Big) \omega^i + \sum_{i=n}^{2n-2} \Big(\sum_{j+k=i} A_j B_k \Big) \omega^{i+n} \\
&= (A * B)(\omega) = A(\omega) B(\omega)
\end{aligned}
$$

[2]

[2] 由于 $\omega^n = 1$

Optimize

- 换言之，
$$DFT_n(A \times B) = DFT_n(A)DFT_n(B)$$
。于是我们可以将 A, B 作 DFT 之后相乘。
- 在 n 不是 2 的幂次时，DFT 和 IDFT 需要做简单的修改。

线性递推式

- $$f_i = \sum_{j=1}^{k} c_j f_{i-j}$$

  。给定 n, P，求 $f_n \bmod P$。

线性递推式

- $$f_i = \sum_{j=1}^{k} c_j f_{i-j}$$
  。给定 n, P，求 $f_n \bmod P$。
- 矩阵快速幂；
- 有时 f 的特征根恰好都在 $\mathbb{Z}/P\mathbb{Z}$ 中，于是我们可以求出特征根，消元求一下通项系数；

线性递推式

- 设转移矩阵为 A，于是 $A^k = \sum_{j=1}^{k} c_j A^{k-j}$。
- 那么 $A^n = P(A^k - \sum_{j=1}^{k} c_j A^{k-j}) + A^n$。
- 设 R 为 x^n 除 $x^k - \sum_{j=1}^{k} c_j x^{k-j}$ 的余式，于是 $A^n = R(A)$。
- 取决于多项式除法的实现，我们可以得到 $O(K^2 \log N)$ 或 $O(K \log K \log N)$ 的复杂度。

……

一些数论函数

- $\phi(n)$：不大于 n 且与 n 互质的数的个数。
- $\mu(n)$：如果 $n = kd^2\,(d > 1)$，$\mu(n) = 0$；否则，$\mu(n) = (-1)^k$，其中 k 是 n 的素因子数。
- 积性函数：$f(x)$ 被称为积性函数，如果 $f(xy) = f(x)f(y)$ 对 $(x, y) = 1$ 的所有 (x, y) 成立。
  - 完全积性函数：去掉 $(x, y) = 1$ 的约束；
  - 积性函数可以用 Euler 筛法预处理。

Mobius 反演

- Dirichlet 卷积：$(f*g)(n)=\sum_{d|n} f(d)g(\frac{n}{d})$。
  - 如果 f 和 g 是积性的，则 $f*g$ 是积性的。
- Mobius 反演：如果 f,g 是积性函数，那么
  - $g(n)=\sum_{d|n} f(d) \Leftrightarrow f(n)=\sum_{d|n} \mu(d)g(\frac{n}{d})$
  - $g(n)=\sum_{n|d} f(d) \Leftrightarrow f(n)=\sum_{n|d} \mu(d)g(\frac{d}{n})$
- 常见谓词的替换
  - $[n=1]=\sum_{d|n} \mu(d)$
  - $[a=b]=[\frac{a}{b}=1]=[\frac{b}{a}=1]$

简单应用

- 求 $\sum_{i=1}^{n}\sum_{j=1}^{m}[(i,j)=1]$，$i,j \le 10^6$，$T \le 10^3$。

简单应用

- 求 $\sum_{i=1}^{n}\sum_{j=1}^{m}[(i,j)=1]$，$i,j\le 10^6$，$T\le 10^3$。

$$\sum_{i=1}^{n}\sum_{j=1}^{m}[(i,j)=1]=\sum_{i,j}\sum_{d|(i,j)}\mu(d)=\sum_{d}\mu(d)\lfloor\frac{n}{d}\rfloor\lfloor\frac{m}{d}\rfloor$$

- 因为 $\lfloor n/d\rfloor$ 只有 $O(\sqrt{n})$ 个值，预处理后单次询问的复杂度是 $O(\sqrt{n}+\sqrt{m})$ 的。

简单应用'

- 求 $\sum_{i=1}^{n}\sum_{j=1}^{m}[i,j]$，$i,j \leq 10^6$，$T \leq 10^3$。

简单应用'

$$\sum_{i=1}^{n}\sum_{j=1}^{m}[i,j]=\sum_{d}\sum_{d|i}\sum_{d|j}[(i,j)=d]\frac{ij}{d}$$

$$=\sum_{d}d\sum_{i'd\le n}\sum_{j'd\le m}[(i',j')=1]i'j'=\sum_{d}d\sum_{i',j'}i'j'\sum_{e|i',e|j'}\mu(e)$$

$$=\sum_{d}d\sum_{e}e^2\mu(e)\times\frac{1}{4}\lfloor\frac{n}{de}\rfloor(\lfloor\frac{n}{de}\rfloor+1)\lfloor\frac{m}{de}\rfloor(\lfloor\frac{n}{de}\rfloor+1)$$

$$=\sum_{D}(\mathrm{id}(\mathrm{id}\mu*1))(D)\times\frac{1}{4}\lfloor\frac{n}{D}\rfloor(\lfloor\frac{n}{D}\rfloor+1)\lfloor\frac{m}{D}\rfloor(\lfloor\frac{n}{D}\rfloor+1)$$

- $O(N+T\sqrt{N})$。

Mertens Function

- 计算 $M(n) = \sum_{i=1}^{n} \mu(i)$。

- 计算 $M(n)=\sum_{i=1}^{n}\mu(i)$。
- 注意到

$$M(n)=\sum_{i=1}^{n}\mu(i)=1-\sum_{i=2}^{n}\sum_{d|i,d\neq i}\mu(d)$$
$$=1-\sum_{i'=2}^{n}\sum_{d=1}^{\lfloor n/i\rfloor}\mu(d)=1-\sum_{i=2}^{n}M(\lfloor\frac{n}{i}\rfloor)$$

。

Mertens Function

- 可以发现如果记忆化搜索的话，总共只会访问到 $O(\sqrt{n})$ 个不同的 $M(i)$。可以算出复杂度是 $O(n^{3/4})$ 的。

Mertens Function

- 可以发现如果记忆化搜索的话，总共只会访问到 $O(\sqrt{n})$ 个不同的 $M(i)$。可以算出复杂度是 $O(n^{3/4})$ 的。
- 如果预处理出 $i \le n^{2/3}$ 时的 $M(i)$，复杂度是

$$O(n^{2/3}) + \sum_{i=2}^{n^{1/3}} O(\sqrt{\frac{n}{i}}) = O(n^{2/3} + \int_2^{n^{1/3}} \sqrt{\frac{n}{x}} \mathrm{d}x) = O(n^{2/3})$$

。

Mertens Function

- 对于其它函数的前缀和也是适用的。
- 设待求函数为 f，前缀和为 S，设 $g(n) = \sum_{d|n} f(d)$，于是

$$S(n) = \sum_{i=1}^{n} f(i) = \sum_{i=1}^{n}(g(i) - \sum_{d|i,d\neq i} f(d))$$
$$= \sum_{i=1}^{n} g(i) - \sum_{i=2}^{n} S(\lfloor \frac{n}{i} \rfloor)$$

。

Mertens Function

- 对于其它函数的前缀和也是适用的。
- 设待求函数为 f，前缀和为 S，设 $g(n)=\sum_{d|n} f(d)$，于是

$$S(n)=\sum_{i=1}^{n} f(i)=\sum_{i=1}^{n}(g(i)-\sum_{d|i,d\neq i} f(d))$$
$$=\sum_{i=1}^{n} g(i)-\sum_{i=2}^{n} S(\lfloor\frac{n}{i}\rfloor)$$

。

- 只要整个算法中只询问了形如 $S(\lfloor\frac{N}{d}\rfloor)$（$N$ 固定）的数，之前的复杂度分析都成立。所以也可以优化一些反演题。

Dzy Loves Math 4

- 给定 n, m，计算
$$\sum_{i=1}^{n}\sum_{j=1}^{m}\phi(ij)$$
。
- $n \le 10^5, m \le 10^9$。

Dzy Loves Math 4

- $\mu(n) \neq 0$ 时，有
$$\phi(nk) = \phi(k) \sum_{d|(n,k)} \phi(\frac{n}{d})$$
。

- $\mu(n) \neq 0$ 时，有
$$\phi(nk) = \phi(k) \sum_{d|(n,k)} \phi(\frac{n}{d})$$
。

- 这是因为
$$\phi(nk) = \phi(k)\phi(n) \prod_{\text{prime } g_i|(n,k)} \frac{g_i}{g_i - 1}$$
。展开乘式和 $\phi(n)$ 即可。

- 设 $S(n,m)=\sum_{j=1}^{m}\phi(nj)$，于是 $\mu(n)\neq 0$ 时，

$$
\begin{aligned}
S(n,m) &= \sum_{j=1}^{m}\phi(j)\sum_{d|(n,j)}\phi(\frac{n}{d}) = \sum_{d|n}\sum_{d|j}\phi(j)\phi(\frac{n}{d}) \\
&= \sum_{d|n}\phi(\frac{n}{d})\sum_{j'd\leq m}\phi(j'd) = \sum_{d|n}\phi(\frac{n}{d})S(d,\lfloor\frac{m}{d}\rfloor)
\end{aligned}
$$

Dzy Loves Math 4

- $\mu(n) = 0$ 时，设 k 为 n 最大的无平方因子，于是 $\phi(n) = \frac{n}{k}\phi(k)$，$S(n,m) = \frac{n}{k}S(k,m)$；
- $n = 1$ 时，可以类似 Mertens Function 推导 $S(1,n)$ 的递归式。
- 可以算出 $O(M^{7/8})$ 的复杂度。