线性筛法和积性函数

visit_world

目录

欧拉筛

- 一种可以在线形复杂度内筛素数的筛法
- 马上可以看到它的功能不只是筛素数
- 线性复杂度的保证：每个合数只会被它最小的质因子筛去

欧拉筛

```
int prime[N], cnt = 0;
bool vis[N];

void prepare(int n)
{
    for (int i = 2; i <= n; ++i)
    {
        if (!vis[i])
            prime[++cnt] = i;
        for (int j = 1; j <= cnt; ++j)
        {
            int to = prime[j] * i;
            if (to > n)     break ;
            vis[to] = 1;
            if (i % prime[j] == 0)  break ;
        }
    }
}
```

欧拉筛

- 简单证明：
  - 设有一个数 m = p * q * s，其中 p 是最小的质因子，q 是另外一个质因子
  - 如果 m 会被 q 筛掉，那一定是在i = p * s 时筛掉的
  - 但是我们发现 i = p * s 时，j 枚举到 p 这个质数时就会 break 出来啦
- 这个特点不仅保证了欧拉筛的线性复杂度，还让我们能在线性时间内预处理出大部分积性函数在 [1, n] 内的值

欧拉筛

- 栗子：在 mod P 意义下[1, n] 中的数的逆元显然有：
  - m = p * q, 则inv[m] = inv[p] * inv[q];
  - 于是我们可以考虑，对每个质数使用 O(log n) 的算法暴力求出逆元，合数的逆元可以用 inv[prime[j]] 和 inv[i] 拼出来
  - n 以内的质数大概是 O(n / ln n) 的， 所以这种做法的复杂度是 O(n) 的
  - 于是你又有了一种线性时间递推逆元的方法辣 : -)

积性函数

- 若有函数 f(n) 的定义域为正整数，值域为复数，称为数论函数

- 进一步地，若数论函数 f(n) 满足：对于互质的 p、q，有 f(p * q) = f(p) * f(q)，称为积性函数，或者说函数满足积性

- 更进一步地，若数论函数 f(n) 满足，对于任意 p、q，有 f(p * q) = f(p) * f(q)，称为完全积性函数

积性函数

- 常见的积性函数：
  - 除数函数σk(n)=∑(d|n)d ^ k，表示n的约数的k次幂和
  - 约数个数函数τ(n)=σ0(n)=∑(d|n) 1，表示n的约数个数，一般也写为d(n)。
  - 约数和函数σ(n)=σ1(n)=∑(d|n)d，表示n的约数之和。
  - 欧拉函数 $\boldsymbol{\varphi}$(n)，不用介绍了~
  - 莫比乌斯函数μ(n)，n 有平方因子时值为 0， 否则值为 (-1) ^ (质因子个数)
  - 元函数 e(n) = [n = 1]，完全积性
  - 恒等函数 I(n) = 1，完全积性
  - 单位函数 id(n) = n，完全积性

关于欧拉函数的两个公式

- 1、$\sum_{(d|n)} \varphi(d) = n$
    - 证明：在[1, n] 中，有 $\varphi(d)$ 个数与 n 的最大公约数为 n / d。
- 2、小于 n 且与 n 互质的数的和为 $\varphi(n) * n / 2$
    - 证明：两两配对，具体来说，若 d 与 n 互质，我们可以得到 n - d 与 n 互质

狄利克雷卷积

- 对两个数论函数进行的运算
- 设我们有两个数论函数 $f(n)$ 和 $g(n)$
- 它们的狄利克雷卷积是一个新的函数 $(f * g)(n)$
- 设这个函数为 h
- 我们有 $h(n) = \sum_{k|n} f(k) * g(n / k)$

狄利克雷卷积

- 一些性质
- 积性函数的狄利克雷卷积仍然满足积性：
  - 证明：对互质的 p、q，有
  - $h(p) * h(q) = \sum(ab = p)\ f(a) * g(b) * \sum(cd = q)\ f(c) * g(d)$
  - $= \sum(ab = p, cd = q)\ f(ac) * g(bd)$
  - $=\sum(xy = pq)\ f(x) * g(y)$
  - $=h(p * q)$
- 注意：完全积性函数的狄利克雷卷积不一定满足完全积性

狄利克雷卷积

- 狄利克雷卷积满足结合律，即对于三个数论函数 f、g、h，有 $(f * g) * h = f * (g * h)$
  - 证明：
  - $(f * g) * h (n) = \sum(ab = n)\ h(b) * (\sum(xy = a)\ f(x) * g(y))$
  - $= \sum(xyb = n)\ f(x) * g(y) * h(b)$
  - $= \sum(xy = n)\ f(x) * (\sum(ab = y)\ g(a) * h(b))$
  - $= f * (g * h) (n)$
- 狄利克雷卷积满足交换律 = = 这个很显然，不证明
- 分配律： $f *(g + h) = f * g + f * h$，这个也不用证啦
- 单位元：$f * e = e * f = f$

狄利克雷卷积

- 常见的狄利克雷卷积：
  - $id = \varphi * 1$
  - $d = 1 * 1$
  - $\sigma = id * 1$
  - $e = 1 * \mu$ （反演式） ⋆
  - $\varphi = id * \mu$ ⋆
- 下面来看一道题吧！

HDU 5628

- 令 $g(n) = \sum_{(i1|n)} \sum_{(i2 | i1)} \sum_{(i3 | i2)} \ldots \sum_{(ik | ik-1)} f(ik)$
- 其中已经告诉你 f(i) 在 1 ~ n 的取值，没有特别规律
- 求 g(1) ~ g(n)，答案对 1e9 + 7 取模
- n、k <= 1e5

HDU 5628

- g = f * (1 ^ k)
- 卷积满足结合律，1 ^ k可以用快速幂的思路乘 log n 次算出来
- 这里有一个问题：如何求两个函数的狄利克雷卷积?
- 暴力啊少年 = = 复杂度其实是 n * log(n) 的
- 调和级数的那一套理论

模板

```
int mul(int * f, int * g, int * h)
{
    for (int i = 1; i <= n; ++i)
    {
        for (int j = i, k = 1; j <= n; j += i, ++k)
            h[j] += f[i] * g[k];
    }
}
```

积性函数的预处理举例

- 这里我们讲以下几个函数的预处理：
  - 莫比乌斯函数
  - 欧拉函数
  - 约数个数函数
  - 约数和函数

莫比乌斯函数

- μ(n) =
  - 0 (n 是有平方因子数)
  - (-1) ^ (质因子个数)
- 改造一下欧拉筛：
  - mu[1] = 1;
  - if (!vis[i]) mu[i] = -1;
  - if (i % prime[j] == 0) mu[to] = 0;
  - else mu[to] = -mu[i];

欧拉函数

- n = p1^ a1 * p2 ^ a2 * p3 ^ a3 * … pk ^ ak
- φ(pi ^ ai) = pi ^ (ai - 1) * (pi - 1)
- 对于 n 的每个质因子，我们需要在 φ(n) 上除以 p，乘以 (p - 1)
- 改造一下欧拉筛：
    - phi[1] = 1;
    - if (!vis[i]) phi[i] = i - 1;
    - if (i % prime[j] == 0) phi[to] = phi[i] * prime[j];
    - else phi[to] = phi[i] * (prime[j] - 1);

欧拉筛的特性

- 不难发现，欧拉筛有这样一个性质：对于任意一个数 n，它的相同的质因子是连续处理的
- 举例来说 : n = p1 ^ 2 * p2 * p3 ^ 3 (p1< p2 < p3)
    - n 被筛掉的过程 :
    - -> p1 * p2 * p3 ^ 3   * p1
    - -> p2 * p3 ^ 3   * p1
    - -> p3 ^ 3   * p2
    - -> p3 ^ 2   * p3
    - -> p3 ^ 1   * p3
    - -> p3   * p3
- 这个性质保证了用欧拉筛预处理积性函数的正确性
- 我们通过下面这个例子来进一步说明这一点

约数个数函数

- n = p1 ^ a1 * p2 ^ a2 * … * pk ^ ak
- 容易知道 d(n) = ∏(ai + 1)
- 改造欧拉筛：
  - 对每个数 n， 记录它最小的质因子的幂次为 num[i]
  - 然后分类讨论
  - 因为相同的质因子连续处理，保证了这样做的正确性
  - 这个给出完整的实现 = = 下一页

模板

```
void prepare(int n)
{
    d[1] = 1;
    for (int i = 2; i <= n; ++i)
    {
        if (!vis[i])
        {
            prime[++cnt] = i;
            d[i] = 2;
            num[i] = 1;
        }
        for (int j = 1; j <= cnt; ++j)
        {
            int to = prime[j] * i;
            if (to > n)     break ;
            vis[to] = 1;
            if (i % prime[j] == 0)
            {
                num[to] = num[i] + 1;
                d[to] = d[i] / (num[i] + 1) * (num[i] + 2);
                break ;
            }
            else
            {
                num[to] = 1;
                d[to] = d[i] * 2;
            }
        }
    }
}
```

约数和函数

- $n = p1 \wedge a1 * p2 \wedge a2 * \ldots * pk \wedge ak$
- $\sigma(p \wedge a) = (p \wedge (a + 1) - 1) / (p - 1)$
- 与上例一样的思路， 记录一个 g[n] 表示 n 的最小的质因子幂次 pi ^ ai 的相关信息
- 这个欧拉筛的改造留给你们作练习～

积性函数预处理总结

- 观察一个给出的函数 f(n)，归纳证明或小范围打表观察考察它的积性
- 对于积性函数，对于质数 p，讨论 f(p ^ a) 怎么算，得到一个关于 p 和 a 的计算式子
- 利用欧拉筛预处理，这个过程的本质是分别计算了 f(pi ^ ai) 的值再合并起来得到了 f(n)

BZOJ 2186

- 给出 n、m ，求 [1, n!] 中与 m! 互质的数的个数
- 保证 m <= n
- 有 T 组询问，答案模同一个数 R
- n, m <= 1e7，T <= 1e4，R <= 1e9 + 10

BZOJ 2186

- ans = $\varphi(m!) * (n! / m!)$
  - $= m! * \prod ((p_i - 1) / p_i) * n! / m!$
  - $= n! * \prod ((p_i - 1) / p_i)$
- 阶乘虽然很大，质因子都 <= 1e7
- 预处理出 [1, 1e7] 内的素数、阶乘、逆元等信息。
- 每个询问 O(1) 回答

BZOJ 2818

- 给定 n，求 1 <= x, y <= n 且 gcd(x, y) 为质数的有序对 (x, y) 的数量
- n <= 1e7

BZOJ 2818

- 枚举 n 以内的质数 p， 则gcd 为 p 的合法的方案有 $(2 * \sum(i <= n / p)\ \boldsymbol{\varphi}(i)) - 1$ 种
- 预处理欧拉函数的前缀和就好了

BZOJ 3629

- 给出 S， 求所有的 n 满足 $\sigma(n) = S$
- $S <= 2e9$

BZOJ 3629

- 对 n 质因数分解，则每个 p ^ a 的贡献为 (1 + p + … + p ^ a)，显然这个数是 S 的约数
- 预处理出[1, 1e5] 以内的质数，dfs 搜索可能的解
- dfs(S, step, now) 表示需要拼出来 S，目前考虑到第 step 个质数，已经用了的 p ^ a 的积为 now
- 搜索时只需要考虑小于 sqrt(S) 的素数和 (S - 1)
- 需要认真实现，常数太丑或者复杂度不对会 T 掉

习题

- BZOJ 1607 （热身）
- POJ 2478 （欧拉函数）
- BZOJ 2190 （欧拉函数）
- BZOJ 2721 （找规律，等式变换）
- POJ 1845 （约数和）
- HDU 4542 （搜索 + 打表）

下期预告

- 莫比乌斯反演
- 变换式子的技巧

Thanks

- email : 745350128@qq.com
- QQ : 745350128