



peng-ym

从头再来，重新上线

[首页](#) [新随笔](#) [联系](#) [管理](#)

随笔 - 17 文章 - 0 评论 - 127 阅读 - 10

莫比乌斯反演

莫比乌斯反演

(PS：在评论区中众多dalao的催促下，我认真的写了三天三夜写完了这篇杜教筛，保证是精品！)

前言

(这大概是我第一次写学习笔记吧ovo)

可能每一个刚开始接触莫比乌斯反演的OIer，起初都会厌恶这个神奇的东西。我也一样233每一个人厌恶的原因有许多，可能是这个烦人的式子，也可能仅仅只是因为不理解 μ 函数而感到不爽。当然，莫比乌斯反演有一个小小的预备知识：**整除分块**

那么我们先从莫比乌斯反演中最基础的莫比乌斯函数 μ 开始说起：

莫比乌斯函数

- 首先，我们可以先明确一点，莫比乌斯函数并不是什么很高大上的东西，它其实只是一个由容斥系数所构成的函数。 $\mu(d)$ 的定义是：

- 当 $d = 1$ 时， $\mu(d) = 1$ ；
- 当 $d = \prod_{i=1}^k p_i$ 且 p_i 为互异素数时， $\mu(d) = (-1)^k$ 。（说直白点，就是 d 分解质因数后，没有幂次大于平方的质因子，此时函数值根据分解的个数决定）；
- 只要当 d 含有任何质因子的幂次大于等于2，则函数值为0.

- 当然，莫比乌斯函数也有很多有趣的性质：

- 对于任意正整数 n ， $\sum_{d|n} \mu(d) = [n = 1]$ 。 ($[n = 1]$ 表示只有当 $n = 1$ 成立时，返回值为1；否则，值为0；（这个就是用 μ 是容斥系数的性质可以证明）)(PS:这一条性质是莫比乌斯反演中最常用的)
- 对于任意正整数 n ， $\sum_{d|n} \frac{\mu(d)}{d} = \frac{\phi(n)}{n}$ 。（这个性质很奇妙，它把欧拉函数和莫比乌斯函数结合起来，或许我之后写杜教筛的学习笔记时会去证明吧）

- 程序实现并不难，我们可以在线性筛素数的程序上略作修改，便可以筛出 μ 函数。
- 那我还是给一段线筛的代码吧

```
void get_mu(int n)
{
    // code
}
```

公告



一言 (ヒトコト)

昵称： pengym
园龄： 3年10个月
粉丝： 124
关注： 0
+加关注

2021年5月						
日	一	二	三	四	五	六
25	26	27	28	29	30	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31	1	2	3	4	5

常用链接

[我的随笔](#)
[我的评论](#)
[我的参与](#)
[最新评论](#)
[我的标签](#)

我的标签

洛谷(8)
莫比乌斯反演(6)
学习笔记(4)
模拟退火(2)
HNOI(2)
杜教筛(1)
数据结构(1)

```

mu[1]=1;
for(int i=2;i<=n;i++)
{
    if(!vis[i]) {prim[++cnt]=i;mu[i]=-1;}
    for(int j=1;j<=cnt&&prim[j]*i<=n;j++)
    {
        vis[prim[j]*i]=1;
        if(i%prim[j]==0) break;
        else mu[i*prim[j]]=-mu[i];
    }
}
}

```

- 那么，莫比乌斯函数就这么告一段落了。

莫比乌斯反演

- 解决完莫比乌斯函数的问题后，我们便迎来了重头戏**莫比乌斯反演**
- 定理： $F(n)$ 和 $f(n)$ 是定义在非负整数集合上的两个函数，并且满足条件：

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

那么存在一个结论：

$$f(n) = \sum_{d|n} \mu(d) F\left(\left\lfloor \frac{n}{d} \right\rfloor\right)$$

这个定理就称作**莫比乌斯反演定理**。

- 莫比乌斯反演的证明主要有两种方式，其中一种就是通过定义来证明；另外一种，我则是会在**杜教筛**中提到（利用**狄利克雷卷积**）。那么我先来说一说第一种证明方法：

$$\begin{aligned} \sum_{d|n} \mu(d) F\left(\left\lfloor \frac{n}{d} \right\rfloor\right) &= \sum_{d|n} \mu(d) \sum_{i|\left\lfloor \frac{n}{d} \right\rfloor} f(i) \\ &= \sum_{i|n} f(i) \sum_{d|\left\lfloor \frac{n}{i} \right\rfloor} \mu(d) = f(n) \end{aligned}$$

(PS：如果不知道最后一步怎么来的，可以再去看性质一，至于和式的变换，就自己脑补一下吧)

- 当然，莫比乌斯反演有另外的一种形式，当 $F(n)$ 和 $f(n)$ 满足：

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

可以推出：

$$f(n) = \sum_{n|d} \mu\left(\frac{d}{n}\right) F(d)$$

- 感觉这个式子，可能在莫比乌斯反演中更加好用。

那么，莫比乌斯反演的基本内容就说完了。知道了这些内容，就已经可以解决一些有关的问题了。我做了一些关于莫比乌斯反演的题，具体题解可以看看我博客中的内容。

题目

YY的gcd

[POI2007]ZAP-Queries

[SDOI2015]约数个数和

[HAOI2011]Problem b

洛谷P1829 [国家集训队]Crash的数字表格

(未完，待更新)

标签： 莫比乌斯反演 , 学习笔记

可持久化并查集(1)
JSOI(1)
感受(1)
更多

积分与排名

积分 - 32926
排名 - 35506

随笔档案

2019年1月(1)
2018年8月(2)
2018年6月(3)
2018年4月(1)
2018年3月(10)

友情链接

zjp_shadow大佬的blog
CVH大佬的blog
Orange大佬的blog
蒟蒻自己在洛谷的blog
y1soi大佬的blog
Hany01大佬的blog
Shichengxiao大佬的blog
dyx大佬的blog
redbag大佬的blog
gaylunch大佬的blog
lstete大佬的blog
chinhhh大佬的blog

最新评论

1. Re:莫比乌斯反演
如果对和式变换看不懂的可以看看维基百科里面的讲解，很好懂
--LEEEEEEEEEEEEEE
2. Re:杜教筛
啧啧，这证明太香了，妙啊
--blanc
3. Re:[SDOI2015]约数个数和
orz
--ACwishes
4. Re:记OI退役
@思益益 bjtu...
--peng
5. Re:记OI退役
大佬现在在哪里
--思益

阅读排行榜

1. 莫比乌斯反演(32627)
2. 杜教筛(26310)
3. 整除分块(13053)
4. 洛谷【P2257】YY的GCD(7291)
5. 可持久化并查集(6127)
6. 模拟退火(5053)
7. [POI2007]ZAP-Queries(3396)
8. [SDOI2015]约数个数和(2915)
9. 记OI退役(1873)
10. 洛谷P1829 [国家集训队]Crash的数字表格(1806)

评论排行榜

1. 莫比乌斯反演(36)
2. 杜教筛(30)
3. 整除分块(14)
4. 记OI退役(9)
5. 洛谷P1829 [国家集训队]Crash的数字表格(9)

推荐排行榜

1. 莫比乌斯反演(37)
2. 杜教筛(36)
3. 整除分块(15)
4. 洛谷【P2257】YY的GCD(8)
5. 模拟退火(7)



37 0

« 上一篇： 洛谷【P2257】YY的GCD
» 下一篇： [POI2007]ZAP-Queries

posted @ 2018-03-26 19:53 pengym 阅读(32628) 评论(36) 编辑 收藏

[刷新评论](#) [刷新页面](#) [返回顶部](#)

登录后才能查看或发表评论，立即 [登录](#) 或者 [逛逛](#) 博客园首页

【推荐】阿里云爆品销量榜单出炉，精选爆款产品低至0.55折

【推荐】7大类400多种组件，HarmonyOS鸿蒙三方库来了，赶紧收藏！

【推荐】大型组态、工控、仿真、CAD\GIS 50万行VC++源码免费下载！

【推荐】限时秒杀！国云大数据魔镜，企业级云分析平台

园子动态：

- 致园友们的一封检讨书：都是我们的错
- 数据库实例 CPU 100% 引发全站故障
- 发起一个开源项目：博客引擎 fluss

最新新闻：

- 菜鸟：2021财年全年收入372.5亿元，同比增68%
- 阿里第四财季营收1874亿元，净亏损54.79亿元
- 小鹏汽车：Q1营收29.5亿元 净亏损7.866亿元
- B站发布Q1财报：总营收39亿元，同比增长68%
- HTC发布VIVE FOCUS 3等系列新品 虚拟代言人也来了
- » [更多新闻...](#)



[学习笔记] [数学] [数论] [线性筛] [莫比乌斯函数]
[莫比乌斯反演]

STATISTICS

在线用户: 1

累计访问: 93,045

2月 13, 2020

莫比乌斯反演

TEAMS

数论函数

NULL (2019)

定义域为正整数的函数称为数论函数。

One,Two,Three,AK
(2018)

积性函数

如果 $\forall a, b, (a, b) = 1, f(ab) = f(a)f(b)$, 这样的数论函数
称为积性函数。

TEMPLATE

常见的数论函数:

- 欧拉函数 (如果 $(a, b) = 1$, 则有 $\varphi(ab) = \varphi(a)\varphi(b)$)
- 莫比乌斯函数
- 除数函数 , 用 $d_k(n)$ 表示。其值等于所有 n 的因子的 k 次方之和。

Template

CATEGORIES

完全积性函数

如果 $\forall a, b, f(ab) = f(a)f(b)$ ，这样的数论函数称为完全积性函数。

常见的完全积性函数有：

- $f(x) = e^x$
- $f(x) = x$

Categories

选择分类目录

ARCHIVE

Dirichlet 卷积

两个数论函数 f, g 的 Dirichlet 卷积为：

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

其中 Dirichlet 卷积的单位元定义为 e ，且

$$e(n) = e(n) = \begin{cases} 1 & (n = 1) \\ 0 & (n \neq 1) \end{cases}$$

Archive

选择月份

SEARCH

【结论】如果 f, g 均为积性函数，则 $f * g$ 也为积性函数。

莫比乌斯函数

如果 n 含有平方因子，那么 $\mu(n) = 0$ ；否则 $\mu(n) = (-1)^k$ ，其中 k 为 n 的本质不同的质因子个数。

【性质】 $e(n) = \sum_{d|n} \mu(d)$ 。

COMMENTS

【证明】我们令 $n = \prod p_i^{q_i}, n' = \prod p_i$ 。

显然，枚举 n 的因子和枚举 n' 因子的差异就在于少枚举了含有平方因子的因子。

QAQ发表在《圆方树学习笔记》

则有 $\sum_{d|n} \mu(d) = \sum_{d|n'} \mu(d)$ 。

此时的 d ，我们就可以考虑在 p_1, p_2, \dots, p_k 中任意选取组成一个因子了，于是就有

$$\sum_{d|n'} \mu(d) = \sum_{i=0}^k \binom{k}{i} (-1)^i = \sum_{i=0}^k \binom{k}{i} (-1)^i \cdot 1^{(k-i)} = (-1+1)^k = 0^k$$

。

FRIENDS

Claris

也就是只有当 $k = 0$ 的时候 $\sum_{d|n} \mu(d) = 1$ ，其他时候 $\sum_{d|n} \mu(d) = 0$ 。

frank_c1

而显然当 $k = 0$ 的时候， $n = 1$ ，于是就证明了。

Awd

莫比乌斯反演

zerol

设 $f(n), g(n)$ 是两个数论函数，如果有 $f(n) = \sum_{d|n} g(d)$ ，那么有 $g(n) = \sum_{d|n} f(d)\mu\left(\frac{n}{d}\right)$ 。

【证明】因为我们有 $e = \mu * 1$ ，而 $f(n) = \sum_{d|n} g(d)$ 其实就是 $f = g * 1$ 。

于是 $\mu * f = g * 1 * \mu = g * (1 * \mu) = g * e = g$ 即 $g = f * \mu$ 也就是 $g(n) = \sum_{d|n} f(d)\mu\left(\frac{n}{d}\right)$ 。

不过一般情况下构造一个 $f(n) = \sum_{d|n} g(d)$ 形式的式子是比较难的，一般情况下我们会直接化成 $[gcd(i, j) = 1]$ 的形式，然后通过 $\sum_{d|gcd(i,j)} \mu(d)$ 来计算。

一种比较常见的问题是这样的：求

$$\sum_{i=1}^n \sum_{j=1}^m f(gcd(i, j)) (n \leq m).$$

我们考虑枚举 $gcd(i, j)$ 的结果，假设 $d = gcd(i, j)$ ，于是就有 $\sum_{i=1}^n \sum_{j=1}^m f(gcd(i, j))$

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

考虑将 $[gcd(i, j) = 1]$ 部分反演，则有

$$\begin{aligned} & \sum_{d=1}^n \sum_{i=1}^{\lfloor \frac{n}{d} \rfloor} \sum_{j=1}^{\lfloor \frac{m}{d} \rfloor} f(d) [gcd(i, j) = 1] \\ &= \sum_{d=1}^n \sum_{i=1}^{\lfloor \frac{n}{d} \rfloor} \sum_{j=1}^{\lfloor \frac{m}{d} \rfloor} f(d) \sum_{d'|gcd(i', j')} \mu(d') \\ &= \sum_{d=1}^n \sum_{d'=1}^{\lfloor \frac{n}{d} \rfloor} \mu(d') f(d) \left\lfloor \frac{n}{dd'} \right\rfloor \left\lfloor \frac{m}{dd'} \right\rfloor \end{aligned}$$

我们令 $g(T) = \sum_{d|T} f(d)\mu\left(\frac{T}{d}\right) = f * \mu$

则有 $\sum_{i=1}^n \sum_{j=1}^m f(gcd(i, j)) = \sum_{T=1}^n g(T) \lfloor \frac{n}{T} \rfloor \lfloor \frac{m}{T} \rfloor$

【例9】HAOI2011 Problem b

【HAOI2011】Problem b



题意 求 \\\
 $(\sum_{i=a}^b \sum_{j=c}^d [\gcd(i, j) = k])$ 。 分
 析 因为 ... 继续阅读



Xiejiadong's Blog

0

【例10】SPOJ5971 LCMSUM

cubercsl

cxhscst2

Manchery

oldjang

lkmcfj

jxtxzzw

godweiyang

zkx06111

billChen



【SPOJ】 LCM Sum

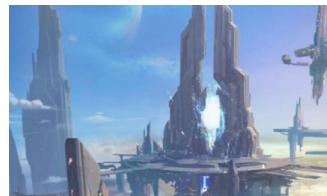
题意 求 $\sum_{i=1}^n \text{lcm}(i, n)$ 。 分析 我们并不太会直接求 lcm，于是考虑 ... 继续阅读



Xiejiadong's Blog

0

【例11】 hdu4944 FSF's game

题意 求 \sum

$\sum_{i=1}^n \sum_{j=i}^n \sum_{d|i,j} \frac{ij}{d}$... 继续
阅读

【例12】 BZOJ3601 一个人的数论

【湖北省队互测】一个人的数论



题意 求所有 $\leq n$

且与 n 互质的数

的 m 次幂的和。

分析 题目要求的就是 ... [继续阅读](#)



By Xiejiadong . No Comment

[f](#) [t](#) [g+](#) [vk](#)

XIEJIADONG Edit your profile or check this video to know

more

YOU MAY ALSO LIKE

“计算理论基础”学习

笔记

3月 18, 2020

“数据库”学习笔记

3月 12, 2020

“编译原理”学习笔记

3月 11, 2020

LEAVE A COMMENT

Your Message

发表评论前, 请滑动滚动条解锁

b

i

link

b-quote
del
ins
img
ul
ol
li
code
more
关闭标签

crayon

Your name *

Your email *

Your webiste

在此浏览器中保存我的姓名、电子邮件和站点地址。

发表评论

Copyrights © 2020 all rights reserved by Jiadong Xie

沪ICP备19039963号

牛顿迭代法

📅 2017-12-24 | 📅 2018-08-17 | ⏺ Studies | 💬 0 | 🕑 阅读次数: 308

🕸 5k | ⏱ 0:04

伽罗瓦云：五次及以上多项式方程没有根式解。但面对生活中求解各种复杂方程的真实需求，是否束手无策？非也，今用牛顿迭代法高效求解方程的近似根。本文介绍了牛顿迭代法的推导和其计算机实现，并使用牛顿迭代法实现了开方等运算。

牛顿迭代法

定义

我们已知：微分是曲线的线性逼近。

对于二阶连续可导函数 $y = f(x)$ ，设 r 是 $f(x) = 0$ 的根。对曲线上的任意一点 $A(x_0, y_0)$ ，作微分 $y' = f'(x_0)dx$ ，交 x 轴于一点 $(x_1, 0)$ ，如下图所示，称 x_1 为 r 的一次近似值。

注意到 x_1 与函数的零点仍有一段距离，

过 x_1 作 x 轴垂线，

交 $y = f(x)$ 于 $B(x_1, y_1)$ ，

继续作微分 $y' = f'(x_1)dx$ ，

交 x 轴于 $(x_2, 0)$ ，

称 x_2 为 r 的二次近似值。

如下图，此时 x_2 与 r 更加接近。

重复上述过程，经过50次迭代，得下图。此时第50次近似值已经十分接近 r 。

定义：

设二阶连续可导函数 $y = f(x)$ ， r 是 $f(x) = 0$ 的根，选取 x_0 作为 r 的初始近似值，过点 $(x_0, f(x_0))$ 做曲线 $y = f(x)$ 的微分 L ，求出 L 与 x 轴交点的横坐标 $x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$ ，称 x_1 为 r 的一次近似值。过点 $(x_1, f(x_1))$ 做曲线 $y = f(x)$ 的切线，并求该切线与 x 轴交点的横坐标 $x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$ ，称 x_2 为 r 的二次

近似值。重复以上过程，得 r 的近似值序列，其中 $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$ ，称为 r 的 $n + 1$ 次近似值，上式称为 **牛顿迭代序列**。

收敛条件

关于牛顿法的局部收敛性，我们有如下定理：

定理：设 x^* 是方程 $f(x) = 0$ 的根， f 在某个包含 x^* 为内点的区间内足够光滑，且 $f'(x) \neq 0$ 。那么存在 x^* 的一个邻域 $N(x^*) = [x^* - \delta, x^* + \delta]$ ，使得对于任意 $x_0 \in N(x^*)$ ，牛顿法产生的迭代序列以不低于二阶的收敛速度收敛于解 x^* 。

证明：

牛顿法是对应于函数 $\phi(x) = x - \frac{f(x)}{f'(x)}$ 的不动点迭代。我们有

$$\phi'(x) = \frac{f(x)f''(x)}{[f'(x)]^2}$$

若 $f'(x) \neq 0$ ，则有 $\phi'(x^*) = 0$ 。因此，牛顿迭代法是局部收敛的。

和不动点收敛类似，对于牛顿法迭代，我们有

$$\begin{aligned} x_{k+1} - x^* &= \phi(x_k) - \phi(x^*) = \phi(x^*) + \phi'(x^*)(x_k - x^*) + \frac{\phi''(\xi_k)}{2}(x_k - x^*)^2 - \phi(x^*) \\ &= \frac{\phi''(\xi_k)}{2}(x_k - x^*)^2 \end{aligned}$$

式中， ξ_k 位于 x_k 和 x^* 之间。因此

$$\lim_{k \rightarrow +\infty} \frac{|x_{k+1} - x^*|}{|x_k - x^*|^2} = \lim_{k \rightarrow +\infty} \frac{|\phi''(\xi_k)|}{2} = \frac{|\phi''(x^*)|}{2}$$

计算机实现

说明

首先我们的目标是近似求解函数的根。

由于牛顿迭代法迭代过程只限于一阶导数，

我们可以使用差分的方式来近似计算某点导数值：

$$f'(x) = \frac{f(x + \text{eps}) - f(x)}{\text{eps}}$$

其中的 eps 取决于我们对最终解精度的要求。

进而，我们由上一节中牛顿迭代公式有：

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

注意到 $\lim_{n \rightarrow +\infty} f(x_n) = 0$ ，

迭代最终收敛，则当我们迭代到 $|f(x_n)| < \text{eps}$ 时就可以退出迭代，

此时的 x_n 即为近似解。

注：在由于计算机内部对实数的存储是有误差的，所以对于 ϵ 的选取应格外小心。

Python 实现

```

1 def diff(f,x):
2     return (f(x+eps)-f(x))/eps
3 def getZero(f,x0):
4     n=0
5     s=x0
6     while True:
7         s=s-f(s)/diff(f,s)
8         n=n+1
9         if math.fabs(f(s))<eps:
10            return (s,n)

```

- $getZero(f, x_0)$ 函数接受一个函数 f ，和迭代起点 x_0
- ϵ 即 eps 为解的精度
- n 为迭代次数
- s 为最终的解值

使用计算机求平方根

我们可以使用上面的代码求解平方根。

首先求解平方根即求函数 $f(x) = x^2 - a$ 的零点。

调用以上函数即

```

1 f=lambda x: x**2-a
2 x, n=getZero(f,x0)
3 print(x,n,f(x))

```



我们取 $\epsilon = 10^{-15}$, $a = 2$, $x_0 = 10$ 时，程序输出

```
1 1.4142135623730951 18 4.440892098500626e-16
```

说明我们得到的近似解为 $\sqrt{2} \approx 1.4142135623730951$ ，迭代了 18 次，
此处的 $f(x_n) = 4.440892098500626 * 10^{-16}$ ，可见 $|f(x_n)| < 10^{-15} = \epsilon$ ，满足精度要求。

接下来我们使用系统自带的求平方根函数求值，其结果为

```
1 1.4142135623730951
```

由此可见我们使用牛顿迭代法求得的近似解已与系统提供的求根函数精度相等，能满足一般的需求。

较复杂函数的零点近似求解

接下来以一个较复杂函数

$$f(x) = \cos((2 - \sin(x))^{\arctan(x)})$$

为例演示函数零点的求解过程。

此函数在 $x = 0$ 附近的图像如下：

为求出函数在 $x = 2$ 附近的零点，

我们取 $\epsilon = 10^{-10}$, $x_0 = 2$ ，程序输出

```
1 2.567793875101797 5 -1.1263042511219229e-14
```

则经过 5 次迭代，我们得到近似解 2.567793875101797。

进一步的，我们可以通过其具体迭代过程发现，迭代的步长缩小得很快，这符合之前的预期：

$$x_0 = 2$$

$$x_1 = 3.048910600444615$$

$$x_2 = 2.5508408648839507$$

$$x_3 = 2.5679378839996416$$

$$x_4 = 2.567793884737295$$

$$x_5 = 2.567793875101797$$

总结

使用牛顿迭代法，我们可以快速近似计算满足收敛条件的函数的零点。

这种方法解决了很多非线性方程组的求根问题，简化了许多工程问题的计算。

参考资料：同济大学计算数学教研室《现代数值计算》（人民邮电出版社）

本文作者： Stardust D.L.

本文链接： <https://stardustdl.github.io/Blog/2017/12/24/牛顿迭代法/>

版权声明： 本博客所有文章除特别声明外，均采用 [CC BY-NC-SA 4.0](#) 许可协议。转载请注明出处！

CS # Math

◀ 线性筛法及扩展

iExpr 反馈收集 ➔

昵称

邮箱

网址(<http://>)

Just go go

Emoji | Preview

M+

回复

Code 1: The app is disabled for violation of user agreement.

Powered By Valine

v1.3.0



© 2017 – 2018 ❤ Stardust D.L. | 📺 27k | 🎵 0:24

👤 7892 🌐 12782

Menci's Blog

幻梦终醒，不悔华年

线性基学习笔记

⌚ 2017-02-25 | ☰ [OI](#)

线性基是竞赛中常用来解决子集异或一类题目的算法。

定义

异或和

设 S 为一无符号整数集（下文中除非特殊说明，**集合**均指「无符号整数集」），定义其**异或和** $\text{xor-sum}(S) = S_1 \text{ xor } S_2 \text{ xor } \dots \text{ xor } S_{|S|}$ 。

张成

设 $T \subseteq S$ ，所有这样的子集 T 的异或和组成的集合称为集合 S 的**张成**，记作 $\text{span}(S)$ 。即，在 S 中选出任意多个数，其异或和的所有可能的结果组成的集合。

线性相关

对于一个集合 S ，如果存在一个元素 S_j ，使得， S 在去除这个元素后得到的集合 S' 的张成 $\text{span}(S')$ 中包含 S_j ，即， $S_j \in \text{span}(S')$ ，则称集合 S **线性相关**。

更形象地，可以表示为，存在一个元素 S_j ，可以用其它若干个元素异或起来得到。

相对的，如果不存在这样的元素 S_j ，则称集合 S **线性无关**。

一个显然的结论是，对于这个线性相关的集合 S ，去除这个元素后，集合的张成不变。

线性基

我们称集合 B 是集合 S 的**线性基**，当且仅当：

1. $S \subseteq \text{span}(B)$ ，即 S 是 B 的张成的子集；
2. B 是线性无关的。

集合 B 中元素的个数，称为线性基的**长度**。

线性基有以下基本性质：

1. B 是极小的满足线性基性质的集合，它的任何真子集都不可能是线性基；
2. S 中的任意元素都可以唯一表示为 B 中若干个元素异或起来的结果。

构造与性质

这里讲解一种线性基的构造方法，及其特殊性质，之后，我们将讨论这种构造方法的正确性。下文中「线性基」均指这种方法构造出的线性基。

设集合 S 中最大的数在二进制意义下有 L 位，我们使用一个 $[0 \dots L]$ 的数组 a 来储存线性基。

这种线性基的构造方法保证了一个特殊性质，对于每一个 i , a_i 有以下两种可能：

1. $a_i = 0$, 并且
 - 只有满足 $j > i$ 的 a_j (即，位于 a_i 后面的所有 a_j) 的第 i 个二进制位可能为 1;
2. $a_i \neq 0$, 并且
 - 整个 a 数组中只有 a_i 的第 i 个二进制位为 1;
 - a_i 更高的二进制位 ($> i$ 的二进制位) 一定为 0;
 - a_i 更低的二进制位 ($< i$ 的二进制位) 可能为 1;

我们称第 i 位存在于线性基中，当且仅当 $a_i \neq 0$ 。

举个例子，我们的构造方法可能得到一个这样的数组 a (右侧为最低二进制位)，其中被标为**蓝色**的元素为上述的第 2 种情况，**绿色**的元素为上述的第 1 种情况。

a_0	0 0 0 0 1
a_1	0 0 0 0 0
a_2	0 0 1 1 0
a_3	0 0 0 0 0
a_4	1 1 0 1 0

举两个反例：

a_0	1 0 0 1	a_0	0 1 0 1
a_1	0 0 1 0	a_1	0 0 1 0
a_2	0 1 0 0	a_2	0 0 0 0
a_3	1 0 0 0	a_3	1 0 0 0

首先，线性基是动态构造的，我们只需要从空的 a 开始，每次考虑在一个已存在的线性基中插入一个数 t 即可。

从 t 最高位上的 1 开始考虑，设这是第 j 位，如果这一位已经存在于线性基中，则我们需要将 t 中的这一位消掉（将 t 异或上 a_j ），才可以继续插入（因为只有 a_j 的第 j 位可以为 1）。如果这一位不存在于线性基中，则可以将 t 插入到 a_j 的位置上，但插入时需要保证：

1. t 中比第 j 位更高的已经存在于线性基中的二进制位上必须为 0，而这时候 t 的最高位上的 1 一定是第 j 位（更高位上即使原本有，也被消掉了），所以无需考虑；
2. t 中比第 j 位更低的已经存在于线性基中的二进制位上必须为 0，对于这一点，我们可以枚举 t 中为 1 的这些二进制位 k ($k \in [0, j)$) 对应的元素 a_k ，并用类似上面的方式将 t 异或上 a_k ，以消掉这些位上的 1；
3. a 中必须只有 a_j 的第 j 位为 1：
 - a 中在 a_j 前面的元素的第 j 位必须为 0，这一点必定已经满足，假设存在一个 $k \in [0, j)$ 满足 a_k 的第 j 位为 1，则 a_k 在插入时就应该被插入到 a_j 的位置上，所以无需考虑；
 - a 中在 a_j 后面的元素的第 j 位必须为 0，对于这一点，我们可以枚举 a_j 后面的元素 a_k ($k \in (j, L]$)，将每个第 j 位为 1 的 a_k 异或上 t 。

流程

- 逆序枚举 t 所有为 1 的二进制位 $j = L \rightarrow 0$ ，对于每个 j
 - 如果 $a_j \neq 0$ ，则令 $t = t \text{ xor } a_j$
 - 如果 $a_j = 0$ ，则
 - 枚举 $k \in [0, j)$ ，如果 t 的第 k 位为 1，则令 $t = t \text{ xor } a_k$
 - 枚举 $k \in (j, L]$ ，如果 a_k 的第 j 位为 1，则令 $a_k = a_k \text{ xor } t$
 - 令 $a_j = t$ ，结束插入过程

代码

```
const int MAXL = 60;

struct LinearBasis
{
    long long a[MAXL + 1];

    LinearBasis()
    {
        std::fill(a, a + MAXL + 1, 0);
    }

    void insert(long long t)
    {
        // 逆序枚举二进制位
        for (int j = MAXL; j >= 0; j--)
        {
            // 如果 t 的第 j 位为 0，则跳过
            if ((t & (1LL << j)) == 0)
                continue;

            // 将 t 异或上 a_j
            t ^= a[j];
        }
    }
};
```

```

if (!(t & (1ll << j))) continue;

// 如果 a[j] != 0, 则用 a[j] 消去 t 的第 j 位上的 1
if (a[j]) t ^= a[j];
else
{
    // 找到可以插入 a[j] 的位置

    // 用 a[0...j - 1] 消去 t 的第 [0, j) 位上的 1
    // 如果某一个 a[k] = 0 也无须担心, 因为这时候第 k 位不存在于线性基中, 不需要保
    for (int k = 0; k < j; k++) if (t & (1ll << k)) t ^= a[k];

    // 用 t 消去 a[j + 1...L] 的第 j 位上的 1
    for (int k = j + 1; k <= MAXL; k++) if (a[k] & (1ll << j)) a[k] ^= t;

    // 插入到 a[j] 的位置上
    a[j] = t;

    // 不要忘记, 结束插入过程
    return;
}

// 此时 t 的第 j 位为 0, 继续寻找其最高位上的 1
}

// 如果没有插入到任何一个位置上, 则表明 t 可以由 a 中若干个元素的异或和表示出, 即 t 在
}
};


```

证明

首先, 根据归纳法, 可以得出其特殊性质是满足的。

我们枚举 S 中的所有元素作为 t , 分别插入, 并将数组 a 转化为一个集合 B (只需要去除重复的为 0 的元素即可, 显然不为 0 的元素不会重复), 此时 B 就是 S 的线性基:

首先, 在插入的过程中, 我们每次找到 t 最高位上的 1, 然后把它消去, 如果最终全部被消去, 则表示要插入的 t 已经可以由 a 中一些元素的异或和表示出, 此时不需要插入。这样保证了 B 是线性无关的。对于被实际插入到 a 中的元素, 它们在实际插入之前做了一些变换, 这些变换都是使它异或上 a 中已存在的元素, 或者使 a 中已存在的元素异或上它 -- 这些变换都是可逆的, 所以用 a 中一些元素的异或和可以表示出这些 (实际被插入的) 元素。

同时, 显然该算法的时间复杂度为 $O(\log t)$ 单次插入, 空间复杂度同样为 $O(\log t)$ 。

合并

两个集合的线性基可以在 $O(\log^2 t)$ 的时间内进行合并，合并后得到的线性基为两个集合的并的线性基。

合并只需要将一个线性基中的所有元素插入到另一个线性基中即可。

```
void mergeFrom(const LinearBasis &other) {
    for (int i = 0; i <= MAXL; i++) insert(other.a[i]);
}

static LinearBasis merge(const LinearBasis &a, const LinearBasis &b) {
    LinearBasis res = a;
    for (int i = 0; i <= MAXL; i++) res.insert(b.a[i]);
    return res;
}
```

应用

线性基最基本的应用，就是子集最大异或和问题：

给一个集合 S ，求它的一个子集 $T \subseteq S$ ，使得 $\text{xor-sum}(T)$ 最大，求出这个最大值。

首先，求出 S 的线性基 B ，问题转化为选择 B 的一个子集。从高到低考虑在线性基中的二进制位 j ，如果第 j 位存在于线性基中，则考虑到线性基中只有惟一一个元素的第 j 位为 1，所以之前加入到 T 中的元素的异或和的第 j 位一定为 0，即，将这个元素加入到 T 中一定会使答案更大——所以，求出线性基中所有元素的异或和，即为答案。

```
long long queryMax()
{
    long long res = 0;
    for (int i = 0; i <= MAXL; i++) res ^= a[i];
    return res;
}
```

- 子集 k 大异或和：[HDU 3949](#)
- 最大路径异或和：[WC2011 Xor](#)
- 求子集异或值排名：[BZOJ 2844](#)
- 树上两点间子集最大异或和：[SCOI2016 幸运数字](#)

模板

不显式构造出集合 B ，支持动态插入。

```

struct LinearBasis
{
    long long a[MAXL + 1];

    LinearBasis()
    {
        std::fill(a, a + MAXL + 1, 0);
    }

    LinearBasis(long long *x, int n)
    {
        build(x, n);
    }

    void insert(long long t)
    {
        for (int j = MAXL; j >= 0; j--)
        {
            if (!t) return;
            if (!(t & (1ll << j))) continue;

            if (a[j]) t ^= a[j];
            else
            {
                for (int k = 0; k < j; k++) if (t & (1ll << k)) t ^= a[k];
                for (int k = j + 1; k <= MAXL; k++) if (a[k] & (1ll << j)) a[k] ^= t;
                a[j] = t;
                break;
            }
        }
    }

    // 数组 x 表示集合 S, 下标范围 [1...n]
    void build(long long *x, int n)
    {
        std::fill(a, a + MAXL + 1, 0);

        for (int i = 1; i <= n; i++)
        {
            insert(x[i]);
        }
    }

    long long queryMax()
    {
        long long res = 0;
        for (int i = 0; i <= MAXL; i++) res ^= a[i];
        return res;
    }

    void mergeFrom(const LinearBasis &other)
}

```

```

{
    for (int i = 0; i <= MAXL; i++) insert(other.a[i]);
}

static LinearBasis merge(const LinearBasis &a, const LinearBasis &b)
{
    LinearBasis res = a;
    for (int i = 0; i <= MAXL; i++) res.insert(b.a[i]);
    return res;
}
};

```

二

显式构造出集合 B (代码中的 v)，不支持动态插入。

```

struct LinearBasis
{
    std::vector<long long> v;
    int n; // 原有集合 S 的大小

    // 数组 x 表示集合 S, 下标范围 [1...n]
    void build(long long *x, int n)
    {
        this->n = n;
        std::vector<long long> a(MAXL + 1);

        for (int i = 1; i <= n; i++)
        {
            long long t = x[i];

            for (int j = MAXL; j >= 0; j--)
            {
                if ((t & (1ll << j)) == 0) continue;

                if (a[j]) t ^= a[j];
                else
                {
                    for (int k = 0; k < j; k++) if (t & (1ll << k)) t ^= a[k];
                    for (int k = j + 1; k <= MAXL; k++) if (a[k] & (1ll << j)) a[k] ^= t

                    a[j] = t;
                    break;
                }
            }
        }

        v.clear();
        for (int i = 0; i <= MAXL; i++) if (a[i]) v.push_back(a[i]);
    }
};

```

```
long long queryMax()
{
    long long x = 0;
    for (size_t i = 0; i < v.size(); i++) x ^= v[i];
    return x;
};
```

数学 # 学习笔记 # 算法模板 # 线性基

◀ 「SCOI2016」美味 - 贪心 + 主席树

「HDU 3949」XOR - 线性基 ▶

[推荐 3](#)[推文](#)[分享](#)

评分最高

加入讨论...

通过以下方式登录

或注册一个 DISQUS 帐号 [?](#)

姓名



XG Zepto · 3 年前

Disqus 被墙了哇

换成基础版吧

1 ^ | v · 回复 · 分享 ,



mxr612 · 5 个月前

orz

^ | v · 回复 · 分享 ,



JiaYi Su · 9 个月前

Orz

^ | v · 回复 · 分享 ,



asd · 9 个月前

orz

^ | v · 回复 · 分享 ,



ce amtic · 10 个月前

Orz

^ | v · 回复 · 分享 ,

[✉ 订阅](#) [dí 在您的网站上使用 Disqus 添加 Disqus 添加](#)

© 2015 – 2021 ❤ Menci

运行于 [GigsCloud](#) 云平台 | 由 [Upyun](#) 提供 CDN 服务由 [Hexo](#) 强力驱动 | 主题 – [NexT.Pisces](#) v5.1.2

Menci's Blog

幻梦终醒，不悔华年

「BZOJ 2844」 albus 就是要第一个出场 - 线性基

2017-02-27 | [oi](#)

给定 n 个数 $\{a_i\}$, 以及一个数 x 。将 $\{a_i\}$ 的所有子集（可以为空）的异或值从小到大排序得到序列 $\{b_i\}$, 请问 x 在 $\{b_i\}$ 中第一次出现的下标是多少? 保证 x 在 $\{b_i\}$ 中出现。

链接

[BZOJ 2844](#)

题解

首先, 求出这 n 个数的线性基。

考虑线性基所控制的某个二进制位, 如果这一位为 1, 那么线性基中控制这一位的元素一定被选择, 这样可以求出 x 在去重后的 $\{b_i\}$ 中第一次出现的下标是多少。

之后, 计算每个重复的数字出现了多少次。设 a 中不在线性基中的数的集合为 S ($|S| = n - |B|$) , 考虑它的一个子集 S' (可以为空) , S' 的异或和一定可以唯一表示为 B 中若干个数的异或和, 将它们都异或起来, 我们可以的到 0, 这样, 我们就得到了 $2^{n-|B|}$ 种方案得到 0, 所以, 对于每一个 b_i , 它的出现次数至少为 $2^{n-|B|}$ 。接着证明它的上界, 假设在 S 中任意选, 最终都可以凑出这个数, 而选择 B 中的数的方案一定是唯一的, 即上界也为 $2^{n-|B|}$ 。

代码

```
#include <cstdio>
#include <vector>

const int MAXN = 100000;
const int MAXL = 30;
const int MOD = 10086;

struct LinearBasis {
    std::vector<int> bit;
```

```

void build(int *x, int n) {
    std::vector<int> a(MAXL + 1);

    for (int i = 1; i <= n; i++) {
        int t = x[i];

        for (int j = MAXL; j >= 0; j--) {
            if (!(t & (1 << j))) continue;

            if (a[j]) t ^= a[j];
            else {
                for (int k = 0; k < j; k++) if (t & (1 << k)) t ^= a[k];
                for (int k = j + 1; k <= MAXL; k++) if (a[k] & (1 << j)) a[k] ^= t;
                a[j] = t;
                break;
            }
        }
    }

    bit.clear();
    for (int i = 0; i <= MAXL; i++) if (a[i]) bit.push_back(i);
}

int size() {
    return bit.size();
}

int rank(int x) {
    int res = 0;

```

快速幂 # BZOJ # 线性基

◀ 「WC2011」 Xor - 线性基

「SCOI2016」 幸运数字 - 线性基 + 树链剖分 ➤

/ 倍增

2条评论

[Menci's Blog](#)[Disqus 隐私政策](#)[1 登录](#)[推荐](#)[推文](#)[分享](#)[评分最高](#)

加入讨论...

通过以下方式登录

或注册一个 DISQUS 帐号 [?](#)

姓名



Rapiz · 4 年前

我非常好奇这原题面是怎么能看懂的

12 ^ | v · 回复 · 分享 ,



Menci 管理员 → Rapiz · 4 年前

一脸智障.jpg

14 ^ | v · 回复 · 分享 ,

[✉ 订阅](#) [DISQUS API](#) 在您的网站上使用 Disqus 添加 Disqus 添加 [▲ 不要出售我的数据](#)

© 2015 – 2021 ❤ Menci

运行于 [GigsCloud](#) 云平台 | 由 [Upyun](#) 提供 CDN 服务由 [Hexo](#) 强力驱动 | 主题 – [NexT.Pisces v5.1.2](#)

Menci's Blog

幻梦终醒，不悔华年

「HDU 3949」 XOR - 线性基

⌚ 2017-02-26 | ☰ [OI](#)

给一组正整数，求其第 k 小的子集异或和。

链接

[HDU 3949 \(Virtual Judge\)](#)

题解

首先，求出这个集合的线性基 B ，选择线性基的一个非空子集共有 $2^{|B|} - 1$ 种方案。如果 $|B| < n$ ，则说明至少有一个没有被插入到线性基中的数可以被线性基中的数表示出来，选择线性基中的一些数与这个数，可以得到其异或和为 0，这样有 $2^{|B|}$ 种方案。

然后，考虑给出线性基，求选择若干（至少一个）数可以组成的第 k 小（ k 从 0 开始，第 0 小为 0）的数。

将 k 表示为一个长度为 $|B|$ 的二进制数（范围为 $[0, |B|]$ ）；如不足，可在高位补 0）。 k 的二进制排列符合以下性质：

1. 高位上的 1 比低位上的 1 更能使 k 更大；
2. 低位上的 1 一定会使 k 更大。

线性基的 $|B|$ 个元素控制了异或后结果的 $|B|$ 个二进制位，而二进制数的规律恰好与从线性基中选数的两条规律相对应：

1. 选择「控制较高位上的 1 的元素」更能使异或和更大；
2. 选择「控制较高位上的 1 的元素」后，再选择「控制更低位上 1 的元素」一定会使异或和更大。

解法就比较显然了——枚举 k 所有为 1 的二进制位，如果第 i 位为 1，则将线性基中控制的第 i 小的二进制位的元素异或到答案中。

代码

```

#include <cstdio>
#include <vector>

const int MAXN = 100000;
const int MAXL = 50;

struct LinearBasis {
    std::vector<long long> v;
    int n;

    void build(long long *x, int n) {
        this->n = n;

        std::vector<long long> a(MAXL + 1);

        for (int i = 1; i <= n; i++) {
            long long t = x[i];

            for (int j = MAXL; j >= 0; j--) {
                if ((t & (1ll << j)) == 0) continue;

                if (a[j]) t ^= a[j];
                else {
                    for (int k = 0; k < j; k++) if (t & (1ll << k)) t ^= a[k];
                    for (int k = j + 1; k <= MAXL; k++) if (a[k] & (1ll << j)) a[k] ^= t
                }

                a[j] = t;

                break;
            }
        }

        /*
        printf("insert(%d):\n", t);
        print();
        */
    }

    v.clear();
    for (int i = 0; i <= MAXL; i++) if (a[i]) v.push_back(a[i]);
}

long long query(long long k) {
    if (int(v.size()) != n) {
        // 可能是 0
    }
}

```

< 线性基学习笔记

「WC2011」Xor - 线性基 >

4条评论

Menci's Blog



Disqus 隐私政策

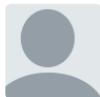
1 登录

推荐

推文

分享

评分最高



加入讨论...

通过以下方式登录

或注册一个 DISQUS 帐号

姓名



zzqDeco • 6 个月前

markdown挂了qaq

| · 回复 · 分享 ·



JiaYi Su • 9 个月前

markdown挂了qaq

| · 回复 · 分享 ·



ce amtic • 10 个月前

markdown挂了qaq

| · 回复 · 分享 ·



Dark Wong • 2 年前

markdown挂了qaq

| · 回复 · 分享 ·

订阅

在您的网站上使用 Disqus 添加 Disqus 添加

不要出售我的数据

© 2015 – 2021 Menci

运行于 GigsGigsCloud 云平台 | 由 Upyun 提供 CDN 服务由 Hexo 强力驱动 | 主题 – NexT.Pisces v5.1.2

Menci's Blog

幻梦终醒，不悔华年

「SCOI2016」幸运数字 - 线性基 + 树链剖分 / 倍增

⌚ 2017-02-27 | ☰ [OI](#)

A国共有 n 座城市，这些城市由 $n - 1$ 条道路相连，使得任意两座城市可以互达，且路径唯一。每座城市都有一个幸运数字，以纪念碑的形式矗立在这座城市的正中心，作为城市的象征。一些旅行者希望游览 A 国。旅行者计划乘飞机降落在 x 号城市，沿着 x 号城市到 y 号城市之间那条唯一的路径游览，最终从 y 城市起飞离开 A 国。

在经过每一座城市时，游览者就会有机会与这座城市的幸运数字拍照，从而将这份幸运保存到自己身上。然而，幸运是不能简单叠加的，这一点游览者也十分清楚。他们迷信着幸运数字是以异或的方式保留在自己身上的。例如，游览者拍了 3 张照片，幸运值分别是 5、7、11，那么最终保留在自己身上的幸运值就是 9 ($5 \text{ xor } 7 \text{ xor } 11$)。

有些聪明的游览者发现，只要选择性地进行拍照，便能获得更大的幸运值。例如在上述三个幸运值中，只选择 5 和 11，可以保留的幸运值为 14。现在，一些游览者找到了聪明的你，希望你帮他们计算出在他们的行程安排中可以保留的最大幸运值是多少。

链接

[BZOJ 4568](#)

题解

树链剖分或倍增维护链上的线性基，查询时合并线性基即可。

复杂度 $O(n \log^4 n)$ 或 $O(n \log^3 n)$ 。

代码

```
#include <cstdio>
#include <ctime>
#include <queue>
#include <algorithm>
```

```

#include <vector>
#include <stack>

const int MAXN = 20000;
const int MAXN_LOG = 14;
const int MAXL = 60;

struct Node {
    std::vector<Node *> adj;
    Node *fa, *ch, *top;
    bool vis;
    int dep, size, dfn;
    long long w;
} N[MAXN + 1];

inline void addEdge(int u, int v) {
    N[u].adj.push_back(&N[v]);
    N[v].adj.push_back(&N[u]);
}

struct LinearBasis {
    // std::vector<long long> v;
    long long a[MAXL + 1];

    LinearBasis() {
        std::fill(a, a + MAXL + 1, 0);
    }

    LinearBasis(long long *x, int n) {
        build(x, n);
    }

    void insert(long long t) {
        for (int j = MAXL; j >= 0; j--) {
            if (!t) return;
            if (!(t & (1ll << j))) continue;

            if (a[j]) t ^= a[j];
            else {
                for (int k = 0; k < j; k++) if (t & (1ll << k)) t ^= a[k];
                for (int k = j + 1; k <= MAXL; k++) if (a[k] & (1ll << j)) a[k] ^= t;
                a[j] = t;
                break;
            }
        }
    }

    void build(long long *x, int n) {
        std::fill(a, a + MAXL + 1, 0);

        for (int i = 1; i <= n; i++) {
            insert(x[i]);
        }
    }
}

```

```

        }

    }

    long long queryMax() {
        long long res = 0;
        for (int i = 0; i <= MAXL; i++) res ^= a[i];
        return res;
    }

    void mergeFrom(const LinearBasis &other) {
        for (int i = 0; i <= MAXL; i++) insert(other.a[i]);
    }

    static LinearBasis merge(const LinearBasis &a, const LinearBasis &b) {
        LinearBasis res = a;
        for (int i = 0; i <= MAXL; i++) res.insert(b.a[i]);
        return res;
    }
};

struct SegT {
    int l, r, mid;
    SegT *lc, *rc;
    LinearBasis lb;

    SegT(int l, int r, SegT *lc, SegT *rc) : l(l), r(r), mid(l + (r - 1) / 2), lc(lc), rc(rc) {}

    SegT(int l, int r, SegT *lc, SegT *rc, long long x) : l(l), r(r), mid(l + (r - 1) / 2), lc(lc), rc(rc) {
        lb.insert(x);
    }

    LinearBasis query(int l, int r) {
        if (l <= this->l && r >= this->r) return lb;
        else if (r <= mid) return lc->query(l, r);
        else if (l > mid) return rc->query(l, r);
        else return LinearBasis::merge(lc->query(l, r), rc->query(l, r));
    }

    static SegT *build(int l, int r, long long *a) {
        if (l == r) return new SegT(l, r, NULL, NULL, a[l]);
        else {
            int mid = l + (r - 1) / 2;
            return new SegT(l, r, build(l, mid, a), build(mid + 1, r, a));
        }
    }
} *seg;

int n, root;

inline void split() {
    std::stack<Node *> s;
    s.push(&N[root]);
    N[root].dep = 1;

    while (!s.empty()) {

```

```

Node *v = s.top();
if (!v->vis) {
    v->vis = true;
    for (Node **p = &v->adj.front(), *u = v->adj.front(); p <= &v->adj.back(); u
        if (!u->dep) {
            u->fa = v;
            u->dep = v->dep + 1;
            s.push(u);
        }
    }
} else {
    v->size = 1;
    for (Node **p = &v->adj.front(), *u = v->adj.front(); p <= &v->adj.back(); u
        v->size += u->size;
        if (!v->ch || v->ch->size < u->size) v->ch = u;
    }
    s.pop();
}
}

for (int i = 1; i <= n; i++) N[i].vis = false;

int ts = 0;
static Node *seq[MAXN + 1];
s.push(&N[root]);
while (!s.empty()) {
    Node *v = s.top();
    if (!v->vis) {
        v->vis = true;

        v->top = (!v->fa || v != v->fa->ch) ? v : v->fa->top;
        seq[v->dfn = ++ts] = v;
        for (Node **p = &v->adj.front(), *u = v->adj.front(); p <= &v->adj.back(); u
            if (u != v->fa && u != v->ch) {
                s.push(u);
            }
        }
        if (v->ch) s.push(v->ch);
    } else {
        s.pop();
    }
}

static long long x[MAXN + 1];
for (int i = 1; i <= n; i++) x[i] = seq[i]->w;

seg = SegT::build(1, n, x);
}

inline long long query(int u, int v) {
    Node *a = &N[u], *b = &N[v];
    LinearBasis lb;

```

```

while (a->top != b->top) {
    if (a->top->dep < b->top->dep) std::swap(a, b);
    lb.mergeFrom(seg->query(a->top->dfn, a->dfn));
    a = a->top->fa;
}
if (a->dep > b->dep) std::swap(a, b);
lb.mergeFrom(seg->query(a->dfn, b->dfn));
return lb.queryMax();
}

/*
int f[MAXN + 1][MAXN_LOG + 1], logn;
LinearBasis g[MAXN + 1][MAXN_LOG + 1];

inline void prepare() {
    std::queue<Node *> q;
    q.push(&N[root]);
    N[root].dep = 1;

    f[root][0] = root;
    g[root][0] = LinearBasis(&N[root].w - 1, 1);

    while (!q.empty()) {
        Node *v = q.front();
        q.pop();

        for (Node **p = &v->adj.front(), *u = v->adj.front(); p <= &v->adj.back(); u = *
            if (!u->dep) {
                u->dep = v->dep + 1;
                f[u - N][0] = v - N;

                long long a[2];
                a[0] = u->w, a[1] = v->w;
                g[u - N][0] = LinearBasis(a - 1, 2);

                q.push(u);
            }
        }
    }
}

while ((1 << (logn + 1)) <= n) logn++;

for (int j = 1; j <= logn; j++) {
    for (int i = 1; i <= n; i++) {
        f[i][j] = f[f[i][j - 1]][j - 1];
        g[i][j] = LinearBasis::merge(g[i][j - 1], g[f[i][j - 1]][j - 1]);
    }
}

inline long long query(int u, int v) {
    if (N[u].dep < N[v].dep) std::swap(u, v);
}

```

```

LinearBasis lb;

if (N[u].dep > N[v].dep) {
    for (int i = logn; i >= 0; i--) {
        if (N[f[u][i]].dep >= N[v].dep) {
            lb.mergeFrom(g[u][i]);
            u = f[u][i];
        }
    }
}

if (u != v) {
    for (int i = logn; i >= 0; i--) {
        if (f[u][i] != f[v][i]) {
            lb.mergeFrom(g[u][i]);
            lb.mergeFrom(g[v][i]);
            u = f[u][i];
            v = f[v][i];
        }
    }
}

lb.mergeFrom(g[u][0]);
lb.mergeFrom(g[v][0]);

u = f[u][0];
v = f[v][0];
}

lb.insert(N[u].w);
return lb.queryMax();
}
*/
}

int main() {
    int q;
    scanf("%d %d", &n, &q);

    // srand((n * q) ^ 20000528);
    // root = rand() % n + 1;
    root = 1;

    for (int i = 1; i <= n; i++) scanf("%lld", &N[i].w);

    for (int i = 1; i <= n - 1; i++) {
        int u, v;
        scanf("%d %d", &u, &v);
        addEdge(u, v);
    }

    // for (int i = 1; i <= n; i++) std::random_shuffle(N[i].adj.begin(), N[i].adj.end())
}

```

BZOJ # 最近公共祖先 # 线性基 # SCOI

◀ 「BZOJ 2844」 albus 就是要第一个出场 - 线性基

「BZOJ 2588」 Count on a tree - 主席树 ▶

0条评论

Menci's Blog

Disqus 隐私政策

1 登录

♥ 推荐

推文

f 分享

评分最高



开始讨论...

通过以下方式登录

或注册一个 DISQUS 帐号 [?](#)

姓名

来做第一个留言的人吧!

✉ 订阅 [D 在您的网站上使用 Disqus 添加 Disqus添加](#) [▲ 不要出售我的数据](#)

© 2015 – 2021 ♥ Menci

运行于 [GigsCloud](#) 云平台 | 由 [Upyun](#) 提供 CDN 服务

由 [Hexo](#) 强力驱动 | 主题 – [NexT.Pisces v5.1.2](#)

Menci's Blog

幻梦终醒，不悔华年

「WC2011」 Xor - 线性基

2017-02-26 | [OI](#)

给一个无向连通图，求一条从 1 到 n 的路径（可以不是简单路径），使经过的边权的异或和最大。

链接

[BZOJ 2115](#)

题解

首先，一个显然的结论是，图中所有简单环的异或和，都是可以直接获得的——因为我们可以从起点走到一个环，由 v 进入这个环，走一圈回到 v 然后原路返回起点，这样起点和 v 之间的边都经过了两次，相当于没有经过这些边，而环上的边权被留在了答案中。

我们可以 BFS 求出所有简单环的边权异或和，并求出它们的线性基。任意找一条从 1 到 n 的路径，然后从大到小考虑线性基中的每个元素加入到答案中会不会使答案更大即可。

证明：设当前答案为 s ，考虑到的线性基中的当前元素为 x ， x 控制着线性基中的第 k 位，那么分为三种情况：

1. s 中第 k 位为 1 ， x 中第 k 位为 1 ，此时不选 x ；
2. s 中第 k 位为 0 ， x 中第 k 位为 1 ，此时选 x ；
3. x 中第 k 位为 0 ，此时一定有 $x = 0$ ，不会影响。

根据归纳法，最终答案是正确的。

代码

```
#include <cstdio>
#include <vector>
#include <queue>

const int MAXN = 50000;
```

```

const int MAXL = 60;

struct Node {
    std::vector<struct Edge> e;
    Node *fa;
    long long dist;
    int dep;
} N[MAXN + 1];

struct Edge {
    Node *s, *t;
    long long w;
}

Edge(Node *s, Node *t, long long w) : s(s), t(t), w(w) {}

inline void addEdge(int s, int t, long long w) {
    N[s].e.push_back(Edge(&N[s], &N[t], w));
    N[t].e.push_back(Edge(&N[t], &N[s], w));
}

inline std::vector<long long> bfs() {
    std::queue<Node *> q;
    q.push(&N[1]);
    N[1].dep = 1;

    std::vector<long long> circles;
    while (!q.empty()) {
        Node *v = q.front();
        q.pop();

        for (Edge *e = &v->e.front(); e <= &v->e.back(); e++) {
            if (!e->t->dep) {
                e->t->dep = v->dep + 1;
                e->t->dist = v->dist ^ e->w;
                e->t->fa = v;
                q.push(e->t);
            } else if (e->t != v->fa) {
                circles.push_back(e->t->dist ^ v->dist ^ e->w);
            }
        }
    }

    return circles;
}

struct LinearBasis {
    std::vector<long long> v;

    void build(std::vector<long long> x) {
        std::vector<long long> a(MAXL + 1);

```

```

for (int i = 0; i < int(x.size()); i++) {
    long long t = x[i];

    for (int j = MAXL; j >= 0; j--) {
        if (!(t & (111 << j))) continue;

        if (a[j]) t ^= a[j];
        else {
            for (int k = 0; k < j; k++) if (t & (111 << k)) t ^= a[k];
            for (int k = j + 1; k <= MAXL; k++) if (a[k] & (111 << j)) a[k] ^= t;
            a[j] = t;
            break;
        }
    }
}

v.clear();

```

BZOJ # 线性基

< 「HDU 3949」XOR - 线性基

「BZOJ 2844」albus就是要第一个出场 - 线

性基

0条评论

Menci's Blog

🔒 Disqus 隐私政策

1 登录

♥ 推荐

推文

f 分享

评分最高



开始讨论...

通过以下方式登录

或注册一个 DISQUS 帐号 [?](#)

姓名

来做第一个留言的人吧！

[✉ 订阅](#) [Ⓓ 在您的网站上使用 Disqus 添加 Disqus 添加](#) [▲ 不要出售我的数据](#)

运行于 [GigsCloud](#) 云平台 | 由 [Upyun](#) 提供 CDN 服务

由 [Hexo](#) 强力驱动 | 主题 – [NexT.Pisces](#) v5.1.2



「笔记」Dirichlet卷积

莫比乌斯反演的前置知识

定义

设 f, g 是数论函数，考虑数论函数 h 满足

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

则称 h 为 f 和 g 的狄利克雷卷积，记作 $h = f * g$ ，这里的 $*$ 表示卷积。

比如 $h(6) = f(1) * g(6) + f(2) * g(3) + f(3) * g(2) + f(6) * g(1)$

性质

1. 单位函数 ϵ 是狄利克雷卷积的单位元，即对于任意函数 f ，有 $\epsilon * f = f * \epsilon = f$ 。
2. 狄利克雷卷积满足交换律和结合律。
3. 如果 f, g 都是积性函数，那么 $f * g$ 也是积性函数。

许多关系都可以用狄利克雷卷积来表示。

下面用 1 来表示取值恒为 1 的常函数，定义幂函数 $\text{Id}_k(n) = n^k$, $\text{Id} = \text{Id}_1$ 。

除数函数的定义可以写为：

$$\sigma_k = 1 * \text{Id}_k$$

欧拉函数的性质可以写为：

$$\text{Id} = \varphi * 1$$

计算狄利克雷卷积

设 f, g 是数论函数，计算 f 和 g 的狄利克雷卷积在 n 处的值需要枚举 n 的所有约数。

如果要计算 f 和 g 的狄利克雷卷积的前 n 项，可以枚举 1 到 n 中每个数的倍数，根据调和数的相关结论，这样做的复杂度是 $O(n \log n)$ 。

求函数的逆

狄利克雷卷积有一个性质：对每个 $f(1) \neq 0$ 的函数 f ，都存在一个函数 g 使得 $f * g = \epsilon$

那么我们如何求出一个函数的逆呢？

只需要定义：

$$g(n) = \frac{1}{f(1)} \left([n=1] - \sum_{i|n, i \neq 1} f(i)g\left(\frac{n}{i}\right) \right)$$

这样的话

$$\begin{aligned} & \sum_{i|n} f(i)g\left(\frac{n}{i}\right) \\ &= f(1)g(n) + \sum_{i|n, i \neq 1} f(i)g\left(\frac{n}{i}\right) \end{aligned}$$

作词 : Vincenzo/[CHUNG] HA/Wonhyuk Kim/Jeremy Ji/Dawn Elektra
作曲 : Vincenzo/[CHUNG] HA/Wonhyuk Kim/Jeremy Ji/Dawn Elektra



步直接把 $g(n)$ 的定义带进去就好

$$= f(1) * \frac{1}{f(1)} ([n = 1] - \sum_{i|n, i \neq 1} f(i)g(\frac{n}{i})) + \sum_{i|n, i \neq 1} f(i)g(\frac{n}{i})$$

例题

P2303 [SDOI2012]Longge的问题

给定正整数 n , 求

$$\sum_{i=1}^n \gcd(i, n), n \leq 2^{32}$$

枚举 \gcd :

$$\begin{aligned} \sum_{i=1}^n \gcd(i, n) &= \sum_{d|n} d \sum_{i=1}^n [\gcd(i, n) = d] \\ &= \sum_{d|n} d \sum_{i=1}^{\frac{n}{d}} [\gcd(i, \frac{n}{d}) = 1] \\ &= \sum_{d|n} d \varphi(\frac{n}{d}) \end{aligned}$$

枚举 n 的约数直接求。答案是积性的。

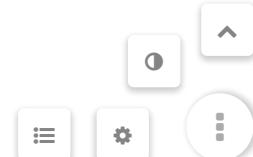
```
#include <cmath>
#include <cstdio>
#include <cstring>
#include <iostream>
#define int long long
using namespace std;

inline int read() {
    char c = getchar(); int x = 0, f = 1;
    for ( ; !isdigit(c); c = getchar()) if(c == '-') f = -1;
    for ( ; isdigit(c); c = getchar()) x = x * 10 + (c ^ 48);
    return x * f;
}

int n, ans;

int euler(int x) {
    int ans = x, rt = sqrt(x);
    for (int i = 2; i <= rt; i++) {
        if (x % i == 0) {
            ans = ans - ans / i;
            while (x % i == 0) x /= i;
        }
    }
    if (x > 1) ans = ans - ans / x;
    return ans;
}

signed main() {
    n = read();
    int x = sqrt(n);
    for (int i = 1; i <= x; i++)
        // 作词: Vincenzo/CHUNG HA/Wonhyuk Kim/Jeremy Ji/Dawn Elektra
        // 作曲: Vincenzo/CHUNG HA/Wonhyuk Kim/Jeremy Ji/Dawn Elektra
}
```



```

        if (n % i == 0) {
            ans += euler(n / i) * i;
            if (i * i != n) ans += euler(i) * (n / i);
        }
    }
cout << ans << '\n';
return 0;
}

```

作者: Loceaner

出处: <https://www.cnblogs.com/loceaner/p/12785524.html>

版权: 本作品采用「署名-非商业性使用-相同方式共享 4.0 国际」许可协议进行许可。

简介: 来自18线小县城的Oler一名, 觉得写的还行的话就关注我吧!

分类: ! 学习笔记 , 数学——莫比乌斯反演

标签: 狄利克雷卷积 , 数论

Buy me a cup of coffee ☕.



3

0

« 上一篇: 「笔记」积性函数

» 下一篇: 「笔记」高中生都能看懂的莫比乌斯反演

posted @ 2020-04-27 11:00 Loceaner 阅读(394) 评论(2) 编辑 收藏

登录后才能查看或发表评论, 立即 登录 或者 逛逛 博客园首页

园子动态:

- 致园友们的一封检讨书: 都是我们的错
- 数据库实例 CPU 100% 引发全站故障
- 发起一个开源项目: 博客引擎 fluss

最新新闻:

- 菜鸟: 2021财年全年收入372.5亿元, 同比增68%
 · 营收1874亿元, 净亏损54.79亿元
- 21营收29.5亿元 净亏损7.866亿元 作词: Vincenzo/CHUNG HA/Wonhyuk Kim/Jeremy Ji/Dawn Elektra
 · 报: 总营收39亿元, 同比增长68% 作曲: Vincenzo/CHUNG HA/Wonhyuk Kim/Jeremy Ji/Dawn Elektra

2021/5/14

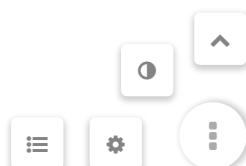
「笔记」Dirichlet卷积 - Loceaner - 博客园



历史上的今天：

2020-04-27 「笔记」积性函数

Copyright © 2021 Loceaner
Powered by .NET 5.0 on Kubernetes & Theme Silence v3.0.0





[数学] [数论] [线性筛] [莫比乌斯函数] [莫比乌斯反演]

STATISTICS

【HAOI2011】 PROBLEM B

在线用户: 1

2月 13, 2020

累计访问: 93,050

题意

TEAMS

求 $\sum_{i=a}^b \sum_{j=c}^d [\gcd(i, j) = k]$ 。

NULL (2019)

分析

One,Two,Three,AK
(2018)

因为 $\sum_{i=1}^n \sum_{j=1}^m f(\gcd(i, j)) = \sum_{T=1}^n g(T) \lfloor \frac{n}{T} \rfloor \lfloor \frac{m}{T} \rfloor$ ，
其中 $g(T) = \sum_{d|T} f(d) \mu(\frac{T}{d}) = f * \mu$ 。

我们令 $f(k) = 1, f(x) = 0(x \neq k)$ 并且假设 $n \leq m$ 。

于是就有 $\sum_{i=1}^n \sum_{j=1}^n [\gcd(i, j) = k] =$
 $= \sum_{T=1}^n \sum_{k|T} \mu(\frac{T}{k}) \lfloor \frac{n}{T} \rfloor \lfloor \frac{m}{T} \rfloor$
 $= \sum_{d=1}^{\lfloor \frac{n}{k} \rfloor} \mu(d) \left\lfloor \frac{n}{dk} \right\rfloor \left\lfloor \frac{m}{dk} \right\rfloor$

TEMPLATE

因为有多组询问，这样的时间复杂度仍然是不允许的。

Template

我们可以考虑数论分块，所谓数论分块，无非是 $\left\lfloor \frac{n}{dk} \right\rfloor \left\lfloor \frac{m}{dk} \right\rfloor$

这部分的取值是 \sqrt{n} 级别的，所以我们可以考虑把相同的一

CATEGORIES

起算。

```
C++
1 #include<bits/stdc++.h>
2 #define LL long long
3 using namespace std;
4
5 const LL p_max = 100010;
6 LL mu[p_max];
7 void get_mu() {
8     mu[1] = 1;
9     static bool vis[p_max];
10    static LL prime[p_max], p_sz, d;
11    for (int i=2;i<p_max;i++)
12    {
13        if (!vis[i]) {
14            prime[p_sz++] = i;
15            mu[i] = -1;
16        }
17        for (LL j = 0; j < p_sz && (d = i *
18            vis[d] = 1;
19            if (i % prime[j] == 0) {
20                mu[d] = 0;
21                break;
22            }
23            else mu[d] = -mu[i];
24        }
25    }
26 }
27
28 LL T,a,b,n,m,k,f[p_max];
29
30 LL calc(LL n,LL m)
31 {
32     if(n>m) swap(n,m);
33     LL ans=0;
34     for (LL i=1;i<=n/k;)
35     {
36         LL p=n/i,k,q=m/i/k;
37         p=min(n/k,n/p/k),q=min(n/k,m/q/k);
38         p=min(p,q);
39         ans+=(f[p]-f[i-1])*(n/i/k)*(m/i/k);
40         //cout<<"p,q="<<p<<" "<<q<<endl;
41         //cout<<i<<" "<<" "<<(f[p]-f[i-1])<<
42         i=p+1;
43     }
44     return ans;
45 }
46
47 void init()
48 {
49     f[0]=0;
50     for (int i=1;i<p_max;i++)
51         f[i]=f[i-1]+mu[i];
52 }
53
54 int main()
55 {
56     get_mu();init();
57     scanf("%lld",&T);
58     while(T--)
59     {
60         scanf("%lld%lld%lld%lld%lld",&a,&n,&
61             m);
62         printf("%lld\n",calc(n,m)-calc(a-1,m));
63     }
64 }
```

Categories

选择分类目录

ARCHIVE

Archive

选择月份

SEARCH

Search ...



COMMENTS

QAQ发表在《圆方树
学习笔记》

FRIENDS

Claris

frank_c1

Awd

zerol

By Xiejiadong . No Comment



cubercsl

cxhscst2

Manchery

oldjang

lkmcfj

jxtxzzw

godweiyang

zkx06111

billChen

“数论基础”课程学习

笔记

3月 4, 2020



CODEFORCES

ROUND #619

2月 15, 2020



【湖北省队互测】—

个人的数论

2月 14, 2020

LEAVE A COMMENT

Your Message

发表评论前, 请滑动滚动条解锁

b
i
link
b-quote
del
ins
img

ul
ol
li
code
more
关闭标签
crayon

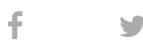
Your name *

Your email *

Your webiste

在此浏览器中保存我的姓名、电子邮件和站点地址。

[发表评论](#)



Copyrights © 2020 all rights reserved by Jiadong Xie

沪ICP备19039963号

[数学] [数论] [欧拉函数] [线性筛] [莫比乌斯函数]
 [莫比乌斯反演]

STATISTICS

在线用户: 1

【HDU4944】FSF'S GAME

累计访问: 93,052

2月 13, 2020

TEAMS

题意

NULL (2019)

求 $\sum_{i=1}^n \sum_{j=i}^n \sum_{d|i,j} \frac{ij}{\gcd(\frac{i}{d}, \frac{j}{d})}$ 。

One,Two,Three,AK

(2018)

分析

因为 $\gcd(\frac{i}{d}, \frac{j}{d}) = \frac{\gcd(i,j)}{d}$ 所以有

$$\begin{aligned} & \sum_{i=1}^n \sum_{j=i}^n \sum_{d|i,j} \frac{ij}{\gcd(\frac{i}{d}, \frac{j}{d})} \\ &= \sum_{i=1}^n \sum_{j=i}^n \sum_{d|\gcd(i,j)} \frac{ijd}{\gcd(i,j)} \\ &\text{显然 } \frac{ij}{\gcd(i,j)} = \text{lcm}(i, j), \text{ 于是} \\ & \sum_{i=1}^n \sum_{j=i}^n \sum_{d|\gcd(i,j)} \frac{ijd}{\gcd(i,j)} \\ &= \sum_{i=1}^n \sum_{j=i}^n \sum_{d|\gcd(i,j)} \text{lcm}(i, j)d \\ &= \sum_{d=1}^n \sum_{i=1}^{\lfloor \frac{n}{d} \rfloor} \sum_{j=i}^{\lfloor \frac{n}{d} \rfloor} \text{lcm}(id, jd)d \\ &= \sum_{d=1}^n d^2 \sum_{i=1}^{\lfloor \frac{n}{d} \rfloor} \sum_{j=i}^{\lfloor \frac{n}{d} \rfloor} \text{lcm}(i, j) \end{aligned}$$

TEMPLATE

Template

CATEGORIES

Categories

选择分类目录



ARCHIVE



Archive

选择月份



SEARCH

Search ...



【SPOJ】LCM Sum

题意求 $\sum_{i=1}^n \text{lcm}(i, n)$ 。分析我们并不太会直接求 lcm，于是考虑 ... 继续阅读



Xiejiadong's Blog

0

COMMENTS

QAQ发表在《圆方树学习笔记》

解决了 $\sum_{i=1}^n \text{lcm}(i, n)$ 这个问题，我们不妨设式子的后半部分为 $f\left(\left\lfloor \frac{n}{d} \right\rfloor\right) = \sum_{i=1}^{\left\lfloor \frac{n}{d} \right\rfloor} \sum_{j=i}^{\left\lfloor \frac{n}{d} \right\rfloor} \text{lcm}(i, j)$ 。这部分其实就是在此基础上再前缀和，有了上一题，可以很容易解决。

于是现在就是求 $\sum_{d=1}^n d^2 f\left(\left\lfloor \frac{n}{d} \right\rfloor\right)$ ，显然 $\left\lfloor \frac{n}{d} \right\rfloor$ 取值是 \sqrt{n} 级别的，可以数论分块，于是我们可以有 $O(n \log n + T \sqrt{n})$ 的做法。

Tle 了两发。过不去。只能考虑优化。

我们考虑枚举 d ，然后计算 d 的贡献。

假设当前 $\left\lfloor \frac{n}{d} \right\rfloor = i$ ，那么当前的 d 的贡献区间是 $[id, (i+1)d]$ ，且贡献为 $d^2 f(i)$ 。

可以考虑在前后打上标记，然后前缀和预处理计算。

此时复杂度变成了 $O(\sum_{d=1}^n \left\lfloor \frac{n}{d} \right\rfloor)$ 大概近似于 $O(n \log n)$ 就可以过了。

[Click To Expand Code](#)

FRIENDS

Claris

frank_c1

Awd

zerol

cubercsl

cxhscst2

Manchery

oldjang

lkmcfj

XIEJIADONG Edit your profile or check this video to know more

jtxzzw

godweiyang

zkx06111

billChen

YOU MAY ALSO LIKE

“数论基础”课程学习

笔记

3月 4, 2020



CODEFORCES
ROUND #619

2月 15, 2020



【湖北省队互测】—
个人的数论

2月 14, 2020

LEAVE A COMMENT

Your Message

发表评论前，请滑动滚动条解锁

b
i
link
b-quote
del
ins
img
ul
ol
li
code

[more](#)[关闭标签](#)[crayon](#)*Your name ***Your email ***Your webiste* 在此浏览器中保存我的姓名、电子邮件和站点地址。[发表评论](#)

Copyrights © 2020 all rights reserved by Jiadong Xie

[沪ICP备19039963号](#)

[数学] [数论] [欧拉函数] [线性筛]

STATISTICS

【SPOJ】LCM SUM

在线用户: 1

2月 13, 2020

累计访问: 93,051

题意

TEAMS

求 $\sum_{i=1}^n \text{lcm}(i, n)$ 。

NULL (2019)

分析

One,Two,Three,AK
(2018)

我们并不太会直接求 lcm，于是考虑转换成 gcd 来做。

$$\begin{aligned} & \sum_{i=1}^n \text{lcm}(i, n) \\ &= \sum_{i=1}^n \frac{in}{\text{gcd}(i, n)} \\ &= \frac{1}{2} \left(\sum_{i=1}^{n-1} \frac{in}{\text{gcd}(i, n)} + \sum_{i=1}^{n-1} \frac{in}{\text{gcd}(i, n)} \right) + n \end{aligned}$$

TEMPLATE

对于 gcd 而言，显然有 $\text{gcd}(a, b) = \text{gcd}(b - a, b)$ ，于是：

$$\begin{aligned} & \sum_{i=1}^n \text{lcm}(i, n) \\ &= \frac{1}{2} \left(\sum_{i=1}^{n-1} \frac{in}{\text{gcd}(i, n)} + \sum_{i=1}^{n-1} \frac{in}{\text{gcd}(n-i, n)} \right) + n \\ &= \frac{1}{2} \sum_{i=1}^{n-1} \frac{in + (n-i)n}{\text{gcd}(i, n)} + n \\ &= \frac{1}{2} \sum_{i=1}^{n-1} \frac{n^2}{\text{gcd}(i, n)} + n \end{aligned}$$

Template

考虑枚举 $\text{gcd}(i, n)$ 的值，如果 $\text{gcd}(i, n) = d$ ，显然有 $\text{gcd}(\frac{i}{d}, \frac{n}{d}) = 1$ ，那么显然这样的 i 有 $\varphi(\frac{n}{d})$ 个。

CATEGORIES

$$\begin{aligned} & \text{于是有 } \sum_{i=1}^n \text{lcm}(i, n) \\ &= \frac{1}{2} \sum_{d|n} \frac{n^2 \varphi(\frac{n}{d})}{d} + n \\ &= \frac{n}{2} \sum_{d|n} d \varphi(d) + n \end{aligned}$$

Categories

选择分类目录

令 $g(n) = \sum_{d|n} d \varphi(d)$ ，这部分显然可以通过线性筛预处理出来，于是每次询问就 $O(1)$ 了。

ARCHIVE

Click To Expand Code

By Xiejiadong . No Comment



Archive

选择月份

XIEJIADONG Edit your profile or check this video to know
more

SEARCH

Search ...



YOU MAY ALSO LIKE

“数论基础”课程学习

笔记

3月 4, 2020



CODEFORCES
ROUND #619

2月 15, 2020



【湖北省队互测】—
个人的数论

2月 14, 2020

COMMENTS

QAQ发表在《圆方树
学习笔记》

FRIENDS

Claris

frank_c1

Awd

zero1

cubercsl

cxhscst2

Manchery

oldjang

lkmcfj

LEAVE A COMMENT

Your Message

发表评论前, 请滑动滚动条解锁

b
i
link
b-quote
del
ins
img

ul	jxtxzzw
ol	godweiyang
li	zkx06111
code	
more	
关闭标签	
crayon	billChen

Your name *

Your email *

Your webiste

在此浏览器中保存我的姓名、电子邮件和站点地址。

发表评论



Copyrights © 2020 all rights reserved by Jiadong Xie

沪ICP备19039963号

[多项式] [拉格朗日插值法] [数学] [数论]

STATISTICS

[莫比乌斯函数] [莫比乌斯反演]

在线用户: 1

【湖北省队互测】一个人的数论

累计访问: 93,054

2月 14, 2020

TEAMS

题意

NULL (2019)

求所有 $\leq n$ 且与 n 互质的数的 m 次幂的和。

One,Two,Three,AK
(2018)

分析

题目要求的就是 $\sum_{i=1}^n i^m [\gcd(i, n) = 1]$ 。

TEMPLATE

用莫比乌斯反演 $\sum_{i=1}^n i^m \sum_{d|i, n} \mu(d)$ 。

Template

变换求和的顺序 $\sum_{d|n} \mu(d) d^m \sum_{i=1}^{\frac{n}{d}} i^m$ 。

显然 $\sum_{d|n} \mu(d) d^m$ 这部分是积性函数。于是考虑变换后一部分。

CATEGORIES

我们令 $f(n) = \sum_{i=1}^n i^m$ ，显然这是一个关于 n 的 $m+1$ 次多项式，于是可以用拉格朗日插值把系数都求出来，变成

$$f(n) = \sum_{i=0}^{m+1} x_i n^i.$$

于是原式就变成了 $\sum_{d|n} \mu(d) d^m \sum_{i=0}^{m+1} x_i (\frac{n}{d})^i$ 。

Categories

变换求和顺序变成 $\sum_{i=0}^{m+1} x_i \sum_{d|n} \mu(d) d^m (\frac{n}{d})^i$ 。

显然 $\sum_{d|n} \mu(d) d^m (\frac{n}{d})^i$ 是一个卷积的形式。

因为 $\mu(d) d^m$ 和 $(\frac{n}{d})^i$ 都是积性函数，那么 $\sum_{d|n} \mu(d) d^m (\frac{n}{d})^i$ 也是一个积性函数。

我们令 $g(n) = \sum_{d|n} \mu(d) d^m (\frac{n}{d})^i$ ，且有

$n = p_1^{a_1} \times p_2^{a_2} \times \cdots \times p_w^{a_w}$ 。那么有

$$g(n) = g(p_1^{a_1}) \times g(p_2^{a_2}) \times \cdots \times g(p_w^{a_w})$$

ARCHIVE

选择分类目录

考虑对于任意的 $g(p_i^{ai}) = \sum_{d|p_i^a} \mu(d) d^m (\frac{p_i^{ai}}{d})^i$ ，只有当 $d = 1$ 或者 $d = p_i$ 的时候 $\mu(d) \neq 0$ 于是只需要求这两项，即 $g(p_i^{ai}) = (p_i^{ai})^i - p_i^m (p_i^{ai-1})^i$ 。

[Click To Expand Code](#)

Archive

选择月份



SEARCH

By Xiejiadong . No Comment



Search ...



XIEJIADONG Edit your profile or check this video to know

more

COMMENTS

QAQ发表在《圆方树
学习笔记》

YOU MAY ALSO LIKE

“数论基础”课程学习
笔记

3月 4, 2020



CODEFORCES
ROUND #619

2月 15, 2020



【HDU4944】
FSF'S GAME

2月 13, 2020

FRIENDS

Claris

frank_c1

Awd

zero1

cubercsl

cxhscst2

Manchery

oldjang

lkmcfj

LEAVE A COMMENT

Your Message

发表评论前，请滑动滚动条解锁

b	jtxzzw
i	
link	
b-quote	godweiyang
del	
ins	
img	
ul	zkx06111
ol	
li	
code	
more	
关闭标签	

crayon

Your name *

Your email *

Your webiste

在此浏览器中保存我的姓名、电子邮件和站点地址。

[发表评论](#)



Copyrights © 2020 all rights reserved by Jiadong Xie

沪ICP备19039963号

望舒草

独行万里，只为曾允你一诺

管理

随笔 - 199 文章 - 0 评论 - 297 阅读 - 47680

min25筛学习笔记

话说我们现在要求一个函数 f 的前缀和。即求 $F(n) = \sum_{i=1}^n f(i)$ 。

min25筛这个算法的主要思想是把 $1 \dots n$ 这些数按质数和合数分类，然后分别考虑质数和合数的贡献。

STEP1 质数贡献

我们尝试先解决一个小问题：求 $G(m) = \sum_{i=1}^m [i \in prime] f(i)$ ，即 m 以下的质数的 f 值之和。其中 $m \in \{\lfloor \frac{n}{1} \rfloor, \lfloor \frac{n}{2} \rfloor, \dots, \lfloor \frac{n}{n} \rfloor\}$ ，共有 $O(\sqrt{n})$ 种取值。

这玩意很难直接求，于是考虑DP。设 $g(j, m) = \sum_{i=1}^m [i \in prime \text{ or } minp(i) > p[j]] f'(i)$ ，其中 $minp(i)$ 表示 i 的最小质因子。这个式子的含义是“ i 为质数或 i 的最小质因子 $> p[j]$ 的函数值 $f'(i)$ ”之和。其中 $f'(i)$ 并不一定是我们要求的 $f(i)$ ，它可以是我们构造出来的另一个函数，但它必须满足以下条件：

1. f' 在质数处的取值与 f 相同。即 $\forall p \in prime$ 有 $f'(p) = f(p)$ 。
2. f' 是完全积性函数。
3. f' 可以快速求前缀和。

显然DP的边界是 $g(0, m) = \sum_{i=2}^m f'(i)$ 。

考虑转移，求 $g(j, m)$ 。当 $p[j]^2 > m$ 时， m 以内不可能有任何最小质因子是 $p[j]$ 的合数，即 $p[j]$ 无法筛掉任何数，因此 $g(j, m) = g(j-1, m) \quad (p[j]^2 > m)$ 。

否则，我们只需要在 $g(j-1, m)$ 里“筛掉” $\leq m$ 的最小质因子为 $p[j]$ 的合数即可。而它们都可以表示为“ $p[j] \times$ 一个 $\lfloor \frac{m}{p[j]} \rfloor$ 以内的最小质因子大于等于 $p[j]$ 的数”。于是有：

$$g(j, m) = g(j-1, m) - f'(p[j]) \times \left(g(j-1, \lfloor \frac{m}{p[j]} \rfloor) - \sum_{i=1}^{j-1} f'(p[i]) \right) \quad (p[j]^2 \leq m)$$

这时 f' 作为一个完全积性函数的好处就体现出来了，它可以直接乘以后面的一坨东西而不用担心是否互质。请读者认真理解这个式子的含义。

令 $|P|$ 表示 $\leq m$ 的质数个数，则 $g(|P|, m)$ 就是我们前面要求的 $G(m)$ 的值了。显然，我们可以把DP的第一维滚掉。

在具体实现时，我们先预处理出 \sqrt{n} 以内的质数（线性筛），以及所有 m 的值（用数论分块）。暴力的做法我们可以用STL-map来记录所有 m 的值的编号，但这里可以更巧妙一些。我们开两个数组`id1`和`id2`，大小都是 \sqrt{n} ，巧妙地利用“对所有 $> \sqrt{n}$ 的数 x ， n/x 一定 $< \sqrt{n}$ ”，就可以记下所有 m 的编号了。

```
int n, val[N*2], id1[N], id2[N];
//主函数 (预处理)
int sqrt_n=sqrt(n), tot=0;
for(int i=1, j; i<=n; i=j+1) {
    j=n/(n/i);
    int w=n/i; val[++tot]=w;
    if(w<=sqrt_n) id1[w]=tot;
    else id2[n/w]=tot;
}
//查询某个m的编号
inline int get_id(int m) {
    if(m<=sqrt_n) return id1[m];
    else return id2[n/m];
}
```

这一部分的时间复杂度被证明是 $O(\frac{n^{\frac{3}{4}}}{\log n})$ 的。

公告

如果我博客里代码，在OJ提交TLE了，那是因为我写博客时自动去掉了快读。请您在这里复制上快读的板子，粘贴到代码前面即可！

如果觉得文章还不错，拜托各位点个推荐+关注，这对我非常重要！

昵称：dysyn1314
园龄：1年9个月
粉丝：87
关注：14
+加关注

搜索

随笔分类

算法-FFT(9)
算法-动态规划(61)
算法-二分(10)
算法-分块(8)
算法-分治(7)
算法-构造/结论/找规律(20)
算法-哈希(2)
算法-莫队(2)
算法-树/树剖/点分治(22)
算法-数据结构(41)
算法-数学-博弈论(3)
算法-数学-概率期望(11)
算法-数学-数论(17)
算法-数学-线性代数/矩阵乘法/高斯消元(5)
算法-数学-组合计数/生成函数/容斥原理(27)
更多

博主的沙雕朋友们

yzhang
hjmmmm姐姐
黄队(黄队稳了)
徐队(传说带妹子保送)
nfls学长 徐源xyleo
无敌myt本体
nfls神仙学弟ycx-akioi
nfls神仙学弟tzc
nfls学长 新加坡之王-泽远
wlzhouzuan转转转
nfls大给给ztr
罗马尼亚大师rainair
George1123

可能给你带来帮助的神仙们

STEP2 贡献加和

现在我们用刚刚搞出来的 $G(m) = \sum_{i=1}^m [i \in prime] f(i)$, 求出所有f的和。

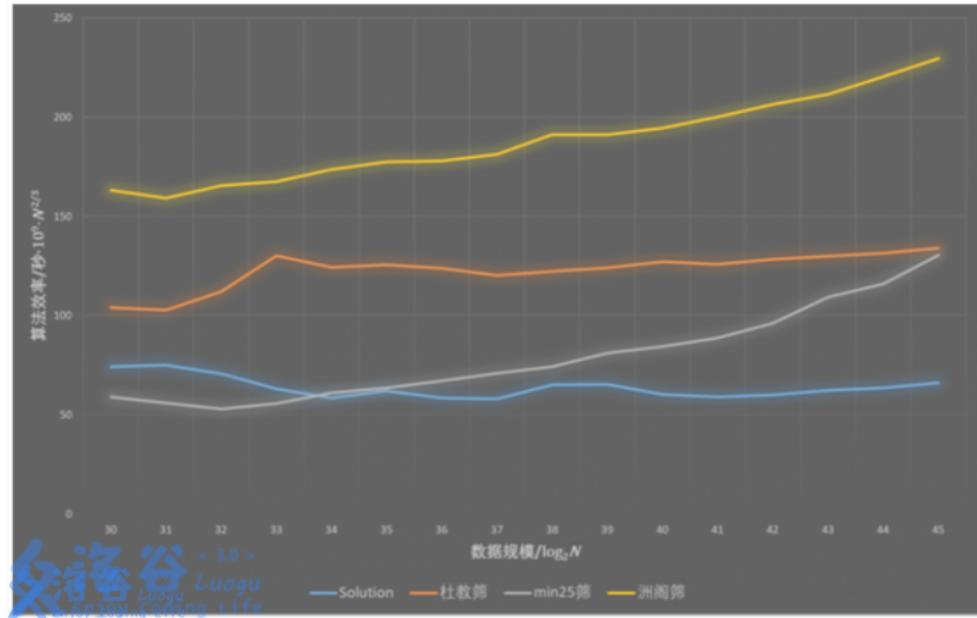
设 $S(n, j) = \sum_{i=1}^n [minp(i) \geq p[j]] f(i)$ 。则显然最终答案是 $f(1) + S(n, 1)$ 。

正如一开始所说的，我们分别考虑质数和合数对 $S(n, j)$ 的贡献。显然质数的贡献是 $G(n) - \sum_{i=1}^{j-1} f(p[i])$ 。对于合数的贡献，我们可以枚举这个合数的“最小质因子及其次数”，然后进行递推。容易列出递推式：

$$S(n, j) = G(n) - \sum_{i=1}^{j-1} f(p[i]) + \sum_{i=j}^{p[i]^2 \leq n} \sum_{e=1}^{p[i]^e \leq n} \left(f(p[i]^e) S(\lfloor \frac{n}{p[i]^e} \rfloor, i+1) + f(p[i]^{e+1}) \right)$$

式子里的二重循环就是在枚举合数的“最小质因子及其次数”，注意合数可以有多个不同的质因子（继续递归）或只有一个质因子的若干次方，这些都在式子里得以体现，请读者仔细理解。

在求 S 时我们直接递归，不用记忆化。这一部分的时间复杂度，被证明也是 $O(\frac{n^{\frac{3}{4}}}{\log n})$ 。反正你只要知道它能跑很大的数据（ 10^{10} 的数据大约1秒搞定）就可以了。



一些栗子

wqy大神的良心模板题

预处理 $f_1(i) = i^2$ 和 $f_2(i) = i$ 在质数处的前缀和，在需要用到时减一下即可。

参考代码：

```
//P5325
#include <bits/stdc++.h>
#define ll long long
#define pb push_back
#define mk make_pair
#define pii pair<int,int>
#define fst first
#define scd second
using namespace std;
inline ll read() {
    ll f=1,x=0;char ch=getchar();
    while(!isdigit(ch)){if(ch=='-')f=-1;ch=getchar();}
    while(isdigit(ch)){x=x*10+ch-'0';ch=getchar();}
    return x*f;
}
inline void write(ll x) {
    if(!x)putchar('0');if(x<0)x=-x,putchar('-');
    static int sta[20];register int tot=0;
    while(x)sta[tot++]=x%10,x/=10;
    while(tot)putchar(sta[--tot]+48);
}
const int MAXN=1e6+5;
const int MOD=1e9+7,INV6=16666666;
ll n,sqrtn,prm[MAXN],sump[MAXN],sum2p[MAXN],cnt,g1[MAXN],g2[MAXN];
bool v[MAXN];
```

租酥雨

cz_xuyixuan

4+7司公子

yyb

jiangly

command-block

zjc神仙

Soulst

PinkRabbit

xht

dqa

最新评论

1. Re:正睿1595 「20联赛集训day1」打字机（区间编辑距离查询问题）

@Flandre-Zhu 以改正，感谢提醒。...

--dysyn1314

2. Re:正睿1595 「20联赛集训day1」打字机（区间编辑距离查询问题）

感觉dys讲的很棒，完全明白了思路是怎么来的，比zr题解不知道高到哪里去了

--Flandre-Zhu

3. Re:正睿1595 「20联赛集训day1」打字机（区间编辑距离查询问题）

dp2的转移第三条里的条件是 $S_{i+1} = T_{j+1}$ 吧

（我想了一下是这样，您代码里也是这样写的）

--Flandre-Zhu

4. Re:LOJ3340 「NOI2020」命运

orz

--ducati♥OI

5. Re:拉格朗日插值学习笔记

Dlstdy!

--maoyiting

阅读排行榜

1. 2020省选联考翻车记(2270)

2. NOI2020游记(2245)

3. NOIP2020 游记(1842)

4. NOIP/CSP-S 考前注意事项(1618)

5. CSP-J/S2019试题选做(1270)

6. 「NOI Online 2021 #1」岛屿探险(1156)

7. 莫比乌斯反演 (入门级别) 学习笔记(1039)

8.

题解 CF1326E Bombs(852)

9. 构造题, 交互题选做(805)

10. 【2020杭电多校round1 1005】HDU6755 Fibonacci Sum (二项式定理) (721)

11. 快速读入、输出, 及其他模板(718)

12. 【2020杭电多校round1 1006】HDU6756 Finding a MEX(717)

13. 题解 CF1340 A,B,C Codeforces Round #637 (Div. 1)(652)

14. 题解 洛谷P6477 [NOI Online #2 提高组] 子序列问题(577)

15. 题解 LOJ3265 3266 3267 USACO 2020.2 Platinum(全)(560)

16. min25筛学习笔记(547)

17. 题解 CF1328 D,E,F Carousel, Tree Queries, Make k Equal(541)

18. 仓鼠的DP课 学习笔记(541)

19. CF1404C Fixed Point Removal(533)

20. 题解 洛谷P6478 [NOI Online #2 提高组] 游戏(518)

```

void sieve(int n) {
    cnt=0;
    for(int i=2;i<=n;++i) {
        if(!v[i]) {
            prm[++cnt]=i;
            sump[cnt]=(sump[cnt-1]+i)%MOD;
            sum2p[cnt]=(sum2p[cnt-1]+(ll)i*i%MOD)%MOD;
        }
        for(int j=1;j<=cnt && prm[j]*i<=n;++j) {
            v[prm[j]*i]=1;
            if(i%prm[j]==0) break;
        }
    }
    ll val[MAXN];
    int id1[MAXN],id2[MAXN];
    inline int get_id(ll x){return ((x<=sqrt(n))?id1[x]:id2[n/x]);}
    ll getS(ll x,ll y) {
        if(x<prm[y] || x<=1) return 0;
        int k=get_id(x);
        ll res=((g2[k]-g1[k]+MOD)%MOD-(sum2p[y-1]-sump[y-1]+MOD)%MOD+MOD)%MOD;
        for(int i=y;i<cnt && prm[i]*prm[i]<=x;++i) {
            ll t1=prm[i],t2=prm[i]*prm[i];
            for(int j=1;t2<=x;++j,t1=t2,t2*=prm[i]) {
                ll ttl=t1%MOD,tt2=t2%MOD;
                res=(res+getS(x/t1,i+1)*ttl%MOD*(tt1-1)%MOD+tt2*(tt2-1)%MOD)%MOD;
            }
        }
        return res%MOD;
    }
    int main() {
        ios::sync_with_stdio(0); /*syn加速*/
        n=read();
        sieve(sqrt(n));
        int tot=0; /*O(sqrt(n))级别*/
        for(ll i=1,j;i<=n;i=j+1) {
            j=n/(n/i);
            ll w=n/i;
            val[++tot]=w;
            if(w<=sqrt(n)) id1[w]=tot;
            else id2[n/w]=tot;

            w%=MOD;
            g1[tot]=w*(w+1)/2%MOD;
            g2[tot]=w*(w+1)%MOD*(2LL*w+1)%MOD*INV6%MOD;

            g1[tot]=(g1[tot]-1+MOD)%MOD;
            g2[tot]=(g2[tot]-1+MOD)%MOD;
        }
        for(int j=1;j<=cnt;++j) {
            for(int i=1;i<=tot && prm[j]*prm[j]<=val[i];++i) {
                int k=get_id(val[i]/prm[j]);
                g1[i]=(g1[i]-prm[j]*(g1[k]-sump[j-1]+MOD)%MOD+MOD)%MOD;
                g2[i]=(g2[i]-prm[j]*prm[j]%MOD*(g2[k]-sum2p[j-1]+MOD)%MOD+MOD)%MOD;
            }
        }
        //for(int i=1;i<=tot;++i) cout<<val[i]<<" "<<g1[i]<<" "<<g2[i]<<endl;
        write((getS(n,1)+1)%MOD); puts("");
        return 0;
    }
}

```

- 21. LOJ3277 「JOISC 2020 Day3」星座 3 (463)
- 22. CF1402C 「CEOI2020」 Star Trek(460)
- 23. 题解 CF1375E Inversion SwapSort (构造) (455)
- 24. 正睿2020提高组十连测 选做(433)
- 25. 【2020杭电多校round6】HDU6834 Yu kikaze and Smooth numbers(432)
- 26. 题解 CF1369 D,E,F Codeforces Round #652 (Div. 2)(426)
- 27. 题解 CF1335 E,F Three Blocks Palindrome, Robots on a Grid(414)
- 28. 题解 LOJ3278 「JOISC 2020 Day3」收获(396)
- 29. 题解 CF1332G No Monotone Triples(387)
- 30. 【2020省选Day1T1】LOJ3299 「联合省选 2020 A | B」冰火战士(384)

意外搞定了杜教筛的模板

先看 $\sum_{i=1}^n \varphi(i)$ 。首先 $\varphi(p) = p - 1$ ($p \in prime$)。于是我们预处理 $f_1(i) = i$ 和 $f_2(i) = 1$ 的前缀和，按与上一题相同的方法相减即可求出 $G(m)$ 。

另外， $\varphi(p^c) = p^{c-1}(p - 1)$ (自己yy可得)，这样在求 S 时，质数的整次幂的函数值也很好求。

然后是 $\mu(i)$ ，显然 $\mu(p) = -1$ ($p \in prime$)。于是我们只要对前面已经求好的 $f_2(i) = 1$ 求个相反数即可。

另外， $\mu(p^k) = 0$ ($k > 1$)，因此求 S 时不需要枚举最小质因子的次数。

参考代码：

```

//P4213 min_25
#include <bits/stdc++.h>
#define ll long long
#define pb push_back
#define mk make_pair
#define pii pair<int,int>
#define fst first

```

复制

```

#define SCD second
using namespace std;
/* ----- by:duyi ----- */
const int N=5e4;
const int MAXN=N*2+5;
int n,sn,cnt,tot,p[MAXN],sum[MAXN],id1[MAXN],id2[MAXN],val[MAXN];
ll g1[MAXN],g2[MAXN];
bool v[MAXN];
void sieve() {
    v[1]=1;
    for(int i=2;i<=N;++i) {
        if(!v[i]) p[++cnt]=i;
        for(int j=1;j<=cnt && (ll)i*p[j]<=N;++j) {
            v[i*p[j]]=1;
            if(i*p[j]==0) break;
        }
    }
    for(int i=1;i<=cnt;++i) sum[i]=sum[i-1]+p[i];
}
inline int get_id(int x) {
    if(x<=sn) return id1[x];
    else return id2[n/x];
}
ll S_phi(int x,int y) {
    if(x<=1 || p[y]>x) return 0;
    ll res=g1[get_id(x)]-g2[get_id(x)]-(sum[y-1]-(y-1));
    for(int i=y;i<=cnt && (ll)p[i]*p[i]<=x;++i) {
        ll pre=1,cur=p[i];
        for(int j=1;cur*p[i]<=x;++j) {
            res+=pre*(p[i]-1LL)*S_phi(x/cur,i+1)+cur*(p[i]-1LL);
            pre=cur;cur=cur*p[i];
        }
    }
    return res;
}
ll S_mu(int x,int y) {
    if(x<=1 || p[y]>x) return 0;
    ll res=-g2[get_id(x)]+y-1;
    for(int i=y;i<=cnt && (ll)p[i]*p[i]<=x;++i) {
        res+=(-S_mu(x/p[i],i+1));
    }
    return res;
}
void solve() {
    cin>>n;
    sn=sqrt(n);tot=0;
    for(int i=1,j;i<=n;i=j+1) {
        j=n/(n/i);int w=n/i;
        val[++tot]=w;
        if(w<=sn) id1[w]=tot;
        else id2[n/w]=tot;
        g1[tot]=(ll)w*(w+1LL)/2LL-1LL;
        g2[tot]=w-1;
    }
    for(int i=1;i<=cnt;++i) {
        for(int j=1;j<=tot && (ll)p[i]*p[i]<=val[j];++j) {
            int t=get_id(val[j]/p[i]);
            g1[j]-=(ll)p[i]*(g1[t]-sum[i-1]);
            g2[j]-=(g2[t]-(i-1));
        }
    }
    cout<<S_phi(n,1)+1LL<<" "<<S_mu(n,1)+1LL<<endl;
}
int main() {
    ios::sync_with_stdio(0); /*syn加速*/
    sieve();
    int T;cin>>T;while(T--)solve();
    return 0;
}

```

总结一下

用min_25做题主要是要想清楚两个问题：

1. 该函数在质数处的取值怎么求和？（找到合适的 f' ）
2. 质数的次幂处的取值 $f(p^c)$ 能否快速求？

本文写得比较粗浅，欢迎大家补充。如有纰漏欢迎指正。

分类: 算法-数学-数论 , 文章分类-学习笔记

[好文要顶](#)[关注我](#)[收藏该文](#)

dysyn1314

关注 - 14

粉丝 - 87

[+加关注](#)

1

0

[« 上一篇: 【2020杭电多校round5】HDU6818 Array Repairing](#)[» 下一篇: 【2020杭电多校round6】HDU6834 Yukikaze and Smooth numbers](#)

posted @ 2020-08-06 21:19 dysyn1314 阅读(547) 评论(3) 编辑 收藏

[刷新评论](#) [刷新页面](#) [返回顶部](#)登录后才能查看或发表评论, 立即[登录](#) 或者[逛逛](#) 博客园首页

Copyright © 2021 dysyn1314
Powered by .NET 5.0 on Kubernetes

[洛谷](#) / 题目列表 / 题目详情

P4213 【模板】杜教筛 (Sum)

提交 21.90k

通过 6.07k

时间限制 2.00s

内存限制 512.00MB

登录后才可提交

题目描述

给定一个正整数，求

$$ans_1 = \sum_{i=1}^n \varphi(i)$$

$$ans_2 = \sum_{i=1}^n \mu(i)$$

输入格式

本题单测测试点内有多组数据。

输入的第一行为一个整数，表示数据组数 T 。

接下来 T 行，每行一个整数 n ，表示一组询问。

输出格式

对于每组询问，输出一行两个整数，分别代表 ans_1 和 ans_2 。

输入输出样例

输入 #1

复制

```
6
1
2
8
13
30
2333
```

输出 #1**[复制]**

```
1 1
2 0
22 -2
58 -3
278 -3
1655470 2
```

说明/提示**数据规模与约定**

对于全部的测试点，保证 $1 \leq T \leq 10$, $1 \leq n < 2^{31}$ 。

[关于洛谷](#) | [帮助中心](#) | [用户协议](#) | [联系我们](#)

[小黑屋](#) | [陶片放逐](#) | [社区规则](#) | [招贤纳才](#)

Developed by the Luogu Dev Team

2013-2021, © 洛谷

增值电信业务经营许可证 沪B2-20200477

沪ICP备18008322号 All rights reserved.

[洛谷 / 题目列表 / 题目详情](#)

P5325 【模板】Min_25筛

提交 4.50k

通过 1.85k

时间限制 2.00s

内存限制 250.00MB

登录后才可提交

题目背景

模板题，无背景。

题目描述

定义积性函数 $f(x)$ ，且 $f(p^k) = p^k(p^k - 1)$ (p 是一个质数)，求

$$\sum_{i=1}^n f(i)$$

对 $10^9 + 7$ 取模。

输入格式

一行一个整数 n 。

输出格式

一个整数表示答案。

输入输出样例

输入 #1

复制

10

输出 #1

复制

263

输入 #2

复制

10000000000

输出 #2**复制**

710164413

说明/提示 $f(1) = 1, f(2) = 2, f(3) = 6, f(4) = 12, f(5) = 20$ $f(6) = 12, f(7) = 42, f(8) = 56, f(9) = 72, f(10) = 40$ 对于30%的数据，保证 $1 \leq n \leq 10^6$ 。对于100%的数据，保证 $1 \leq n \leq 10^{10}$ [关于洛谷](#) | [帮助中心](#) | [用户协议](#) | [联系我们](#)[小黑屋](#) | [陶片放逐](#) | [社区规则](#) | [招贤纳才](#)

Developed by the Luogu Dev Team

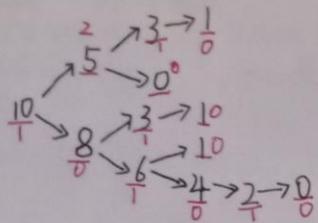
2013-2021, © 洛谷

增值电信业务经营许可证 沪B2-20200477

沪ICP备18008322号 All rights reserved.

SG函数：

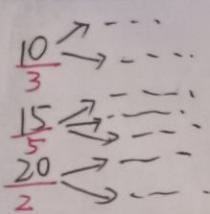
①设取石子的集合为 $\{2, 5\}$,且仅有一堆石子,石子数为10(终点的SG值为0)



当仅有一堆石子时,如果 $SG(10) \neq 0$,必胜, $SG(10)=0$,必败
原因:当 SG 值不为0时,因为 SG 值经过了 Max 运算,所以其连接的点必有一点为0,当走到 SG 值为0的这一点时,由于 Max 运算,该点所连接的点的值必定不为0,∴终点的 SG 值为0,所以只要先手的 SG 值不为0,便可以一直走到 SG 值为0的点,最终走向终点。反之同理。

②当有多堆石子的情况:

任意设每堆石子的 SG 值:



每堆石子的 SG 值都是与其连接点的 SG 值中最小的且与连接点的值不同的最小非负整数,所以选择时都可以选择 SG 值 $0-SG(x)$ 之间的任意数。

例:
 $\because SG(10)=3$, ∴ $SG(10)$ 可以选择连接点的 SG 值的范围是 $0-3$ ($0 \leq SG < 3$)
 $\therefore SG(15)=5$, $\therefore SG(15)$ 也可选择 $[0, 5]$, $SG(20)$ 可选择 $[0, 2]$

将每堆石子的 SG 值取出来,可以发现这些值拼接起来可以成为Nim游戏!

当 $SG(X_1) \wedge SG(X_2) \wedge SG(X_3) \neq 0$ 时,可以选择 SG 值最大的那个,如: $SG(X_2)$,使得

$$SG(X_1) \wedge SG(X_2) \wedge SG(X_3) = m \Rightarrow SG(X_1) \wedge SG(X_2) \wedge SG(X_3) = SG(X_2) = m \wedge m = 0$$

接下来的步骤可参考Nim游戏。

所以当所有石子堆的异或值不等于0的话,必胜,等于0,必败。

BSGS HASHMAP

```
// B^L == N (mod P)
// Code by KSkun, 2018/4
#include <cstdio>
#include <cmath>
#include <cstring>

#include <algorithm>

typedef long long LL;

inline LL fpow(LL n, LL k, LL p) {
    LL res = 1; n %= p;
    while(k) {
        if(k & 1) res = res * n % p;
        n = n * n % p;
        k >>= 1;
    }
    return res;
}

const int MO = 611977, MAXN = 1000005;

struct HashMap {
    int head[MO + 5], key[MAXN], value[MAXN], nxt[MAXN], tot;
    inline void clear() {
        tot = 0;
        memset(head, -1, sizeof(head));
    }
    HashMap() {
        clear();
    }
    inline void insert(int k, int v) {
        int idx = k % MO;
        for(int i = head[idx]; ~i; i = nxt[i]) {
            if(key[i] == k) {
                value[i] = v;
                return;
            }
        }
        key[tot] = k; value[tot] = v; nxt[tot] = head[idx]; head[idx] = tot++;
    }
}
```

```

inline int operator[](const int &k) const {
    int idx = k % MO;
    for(int i = head[idx]; ~i; i = nxt[i]) {
        if(key[i] == k) return value[i];
    }
    return -1;
}
} x;

inline LL bsgs(LL a, LL b, LL p) {
    a %= p; b %= p;
    if(a == 0) return b == 0 ? 1 : -1;
    if(b == 1) return 0;
    LL m = ceil(sqrt(p - 1)), inv = fpow(a, p - m - 1, p);
    x.clear();
    x.insert(1, 0);
    for(LL i = 1, e = 1; i < m; i++) {
        e = e * a % p;
        if(x[e] == -1) x.insert(e, i);
    }
    for(LL i = 0; i < m; i++) {
        if(x[b] != -1) {
            LL res = x[b];
            return i * m + res;
        }
        b = b * inv % p;
    }
    return -1;
}

LL p, b, n;

int main() {
    while(scanf("%lld%lld%lld", &p, &b, &n) != EOF) {
        LL res = bsgs(b, n, p);
        if(res != -1) printf("%lld\n", res); else puts("no solution");
    }
    return 0;
}

```

BSGS STLMAP

```
// B^L == N (mod P)
// Code by KSkun, 2018/4
#include <cstdio>
#include <cmath>

#include <map>

typedef long long LL;

inline LL fpow(LL n, LL k, LL p) {
    LL res = 1; n %= p;
    while(k) {
        if(k & 1) res = res * n % p;
        n = n * n % p;
        k >>= 1;
    }
    return res;
}

std::map<LL, LL> x;

inline LL bsgs(LL a, LL b, LL p) {
    a %= p; b %= p;
    if(a == 0) return b == 0 ? 1 : -1;
    if(b == 1) return 0;
    LL m = ceil(sqrt(p - 1)), inv = fpow(a, p - m - 1, p);
    x.clear();
    x[1] = m; // use m instead of 0
    for(LL i = 1, e = 1; i < m; i++) {
        e = e * a % p;
        if(!x[e]) x[e] = i;
    }
    for(LL i = 0; i < m; i++) {
        if(x[b]) {
            LL res = x[b];
            return i * m + (res == m ? 0 : res);
        }
        b = b * inv % p;
    }
    return -1;
}
```

```

LL p, b, n;

int main() {
    while(scanf("%lld%lld%lld", &p, &b, &n) != EOF) {
        LL res = bsgs(b, n, p);
        if(res != -1) printf("%lld\n", res); else puts("no solution");
    }
    return 0;
}

```

exBSGS HASHMAP

```

// B^L == N (mod P)
// Code by KSkun, 2018/4
#include <cstdio>
#include <cmath>
#include <cstring>

#include <algorithm>

typedef long long LL;

inline char fgc() {
    static char buf[100000], *p1 = buf, *p2 = buf;
    return p1 == p2 && (p2 = (p1 = buf) + fread(buf, 1, 100000, stdin), p1 == p2) ? EOF
        : *p1++;
}

inline LL readint() {
    register LL res = 0, neg = 1;
    char c = fgc();
    while(c < '0' || c > '9') {
        if(c == '-') neg = -1;
        c = fgc();
    }
    while(c >= '0' && c <= '9') {
        res = res * 10 + c - '0';
        c = fgc();
    }
}

```

```

        return res * neg;
    }

inline LL fpow(LL n, LL k, LL p) {
    LL res = 1; n %= p;
    while(k) {
        if(k & 1) res = res * n % p;
        n = n * n % p;
        k >>= 1;
    }
    return res;
}

inline LL exgcd(LL a, LL b, LL &x, LL &y) {
    if(!b) {
        x = 1; y = 0;
        return a;
    }
    LL res = exgcd(b, a % b, x, y);
    LL t = x; x = y; y = t - a / b * y;
    return res;
}

const int MO = 611977, MAXN = 1000005;

struct HashMap {
    LL head[MO + 5], key[MAXN], value[MAXN], nxt[MAXN], tot;
    inline void clear() {
        tot = 0;
        memset(head, -1, sizeof(head));
    }
    HashMap() {
        clear();
    }
    inline void insert(LL k, LL v) {
        int idx = k % MO;
        for(int i = head[idx]; ~i; i = nxt[i]) {
            if(key[i] == k) {
                value[i] = std::min(value[i], v);
                return;
            }
        }
        key[tot] = k; value[tot] = v; nxt[tot] = head[idx]; head[idx] = tot++;
    }
}

```

```

inline LL operator[](const LL &k) const {
    int idx = k % MO;
    for(int i = head[idx]; ~i; i = nxt[i]) {
        if(key[i] == k) return value[i];
    }
    return -1;
}
} x;

inline LL bsgs(LL a, LL b, LL p) {
    a %= p; b %= p;
    if(a == 0) return b == 0 ? 1 : -1;
    if(b == 1) return 0;
    LL m = ceil(sqrt(p - 1)), inv, y;
    exgcd(fpow(a, m, p), p, inv, y); inv = (inv % p + p) % p;
    x.clear();
    x.insert(1, 0);
    for(LL i = 1, e = 1; i < m; i++) {
        e = e * a % p;
        if(x[e] == -1) x.insert(e, i);
    }
    for(LL i = 0; i < m; i++) {
        if(x[b] != -1) {
            LL res = x[b];
            return i * m + res;
        }
        b = b * inv % p;
    }
    return -1;
}

inline LL gcd(LL a, LL b) {
    if(!b) return a;
    return gcd(b, a % b);
}

inline LL exbsgs(LL a, LL b, LL p) {
    if(b == 1) return 0;
    LL tb = b, tmp = 1, k = 0;
    for(int g = gcd(a, p); g != 1; g = gcd(a, p)) {
        if(tb % g) return -1;
        tb /= g; p /= g; tmp = tmp * a / g % p;
        k++;
        if(tmp == tb) return k;
    }
}

```

```

    }
    return bsgs(a, b, p);
}

LL a, b, p;

int main() {
    for(;;) {
        a = readint(); p = readint(); b = readint();
        if(!a && !b && !p) break;
        LL res = exbsgs(a, b, p);
        if(res != -1) printf("%lld\n", res); else puts("No Solution");
    }
    return 0;
}

```

高斯消元解异或线性方程组

```

#include <iostream>
#include <algorithm>

using namespace std;

const int N = 110;

int n;
int a[N][N];
int gauss()
{
    int c,r;
    for(c=0,r=0;c<n;c++)
    {
        // 找主元
        int t=-1;
        for(int i=r;i<n;i++)
            if(a[i][c])
        {
            t=i;
            break;
        }

```

```

        if(t==-1) continue;
        // 交换主元行
        for(int j=c;j<=n;j++) swap(a[r][j],a[t][j]);
        // 左下角消
        for(int i=r+1;i<n;i++)
            if(a[i][c])//漏了
                for(int j=n;j>=c;j--)//漏了
                    a[i][j] ^= a[r][j];
        r++;
    }
    // 判断
    if(r<n)
    {
        for(int i=r;i<n;i++)//i=r
            if(a[i][n])
                return 2;
        return 1;
    }
    // 消右上角
    for(int i=n-1;i>=0;i--)
        for(int j=i+1;j<n;j++)
            //如果是 0 就不用下面的 a[j][j] 来^a[i][j] 了
            //如果不是 0 才需要用第 j 行第 j 列 a[j][j] 来^第 i 行第 j 列 a[i][j]
            //进而进行整行 row[i]^row[j] 间接导致 a[i][n]^a[j][n]
            if(a[i][j])
                a[i][n]^=a[j][n];
    return 0;
}

int main()
{
    cin >> n;
    for(int i=0;i<n;i++)
        for(int j=0;j<=n;j++)
            cin >> a[i][j];
    int t = gauss();
    if(t==0)
    {
        for(int i=0;i<n;i++) cout << a[i][n] << endl;
    }
    else if(t==1) puts("Multiple sets of solutions");
    else puts("No solution");
    return 0;
}

```

模意义下的高斯消元

```
#include<cstdio>
#define maxn 110
#define r register
using namespace std;
typedef long long ll;
int n,p,maxi;
ll tmp,ans[maxn],a[maxn][maxn];
int read()
{
    r char ch=getchar();r int in=0;
    while(ch>'9' || ch<'0') ch=getchar();
    while(ch>='0'&&ch<='9') in=(in<<3)+(in<<1)+ch-'0',ch=getchar();
    return in;
}
ll ksm(r ll x,r int y)
{
    if(!y) return 1;
    r ll ret=ksm(x,y>>1);
    if(y&1) return ret*ret%p*x%p;
    return ret*ret%p;
}
int main()
{
    n=read(),p=read();
    for(r int i=1;i<=n;i++)
        for(r int j=1;j<=n+1;j++)
            a[i][j]=read();
    for(r int i=1;i<=n;i++)
    {
        if(!a[i][i])//主元不能为 0
        {
            maxi=0;
            for(r int j=i+1;j<=n&&!maxi;j++)
                if(a[j][i]) maxi=j;
            if(!maxi) continue;//如果一整列都为 0， 不需要消元
            for(r int j=i;j<=n+1;j++)
                tmp=a[maxi][j],a[maxi][j]=a[i][j],a[i][j]=tmp;
        }
        for(r int j=i+1;j<=n;j++)
        {
            tmp=a[j][i];
            a[j][i]=0;
            for(r int k=i+1;k<=n;k++)
                a[j][k]=(a[j][k]-tmp*a[i][k])%p;
        }
    }
}
```

```
if(!tmp) continue;//已经为 0, 不需要消元
for(r int k=i;k<=n+1;k++)
    a[j][k]=((a[j][k]*a[i][i]-a[i][k]*tmp)%p+p)%p;
}
}
for(r int i=n;i;i--)
{
    for(r int j=i+1;j<=n;j++)
        a[i][n+1]=((a[i][n+1]-ans[j]*a[i][j])%p+p)%p;
    ans[i]=a[i][n+1]*ksm(a[i][i],p-2)%p;
}
for(r int i=1;i<=n;i++) printf("%lld ",ans[i]);
return 0;
}
```

ACM常用数列 (斐波那契数列、卡特兰数、贝尔数、斯特灵数)

原创 九日王朝 2016-11-19 16:24:33 3356 收藏 3

版权

分类专栏: ACM 文章标签: ACM 数学数列 数论

斐波那契数列: 任意一个数是其前两位数只和, 即 $f(i)=f(i-1)+f(i-2)$, $f(1)=f(2)=1$

该数列也满足黄金分割比例, 所以又成为黄金分割数列

相关题目链接: Fibonacci Number

<http://acm.hdu.edu.cn/showproblem.php?pid=2070>

```

1 #include<stdio.h>
2 int main()
3 {
4     __int64 s[51]={0,1};
5     int i;
6     for(i=2;i<=50;i++)
7         s[i]=s[i-2]+s[i-1];
8     int n;
9     while(scanf("%d",&n)&&(n!=1))
10        printf("%I64d\n",s[n]);
11    return 0;
12 }
13 }
```

卡特兰数: 实际上就是出栈序列的种数, 记得有一年蓝桥杯考的卡特兰数, 当时还不知道, 所以写了32个for循环



令 $h(0)=1, h(1)=1$, catalan数满足递推式:

$$h(n)=h(0)*h(n-1)+h(1)*h(n-2) + \dots + h(n-1)*h(0) \quad (n \geq 2)$$

$$\text{例如: } h(2)=h(0)*h(1)+h(1)*h(0)=1*1+1*1=2$$

$$h(3)=h(0)*h(2)+h(1)*h(1)+h(2)*h(0)=1*2+1*1+2*1=5$$

另类递推式:

$$h(n)=h(n-1)*(4*n-2)/(n+1);$$

递推关系的解为:

$$h(n)=C(2n,n)/(n+1) \quad (n=0,1,2,\dots)$$

递推关系的另类解为:

$h(n) = c(2n, n) - c(2n, n-1)$ ($n=0, 1, 2, \dots$)

更详细说明: <http://baike.so.com/doc/6127416-6340576.html>

相关题目链接: 小兔的棋盘

<http://acm.hdu.edu.cn/showproblem.php?pid=2067>

```
1 #include<stdio.h>
2 int main()
3 {
4     __int64 a[36];
5     int i,j;
6     a[1]=1;
7     for(i=2;i<=35;i++)
8         a[i]=a[i-1]*1.0/(i+1)*(4*i-2);
9     int n;
10    int x=1;
11    while(scanf("%d",&n))
12    {
13        if(n===-1)
14            break;
15        printf("%d %d %I64d\n",x,n,2*a[n]);
16        x++;
17    }
18    return 0;
19 }
```

Bell数, 又称为贝尔数。

是以埃里克·坦普尔·贝尔(Eric Temple Bell)为名的。

$B(n)$ 是包含 n 个元素的集合的划分方法的数目。

$B(0) = 1, B(1) = 1, B(2) = 2, B(3) = 5,$

$B(4) = 15, B(5) = 52, B(6) = 203, \dots$

递推公式为,

$B(0) = 1,$

$B(n+1) = \sum_{k=0}^n C(n,k)B(k).$ $n = 1, 2, \dots$

其中, $\sum_{k=0}^n$ 表示对 k 从0到 n 求和, $C(n,k) = n!/[k!(n-k)!]$

Stirling数, 又称为斯特灵数。

在组合数学, Stirling数可指两类数, 都是由18世纪数学家James Stirling提出的。

第一类Stirling数是有正负的, 其绝对值是包含 n 个元素的集合分作 k 个环排列的方法数目。

递推公式为,

$S(n,0) = 0, S(1,1) = 1.$

$S(n+1,k) = S(n,k-1) + nS(n,k).$

第二类Stirling数是把包含 n 个元素的集合划分为正好 k 个非空子集的方法的数目。

递推公式为,

$S(n,n) = S(n,1) = 1,$

$S(n,k) = S(n-1,k-1) + kS(n-1,k).$

将 n 个有区别的球的球放入 k 个无标号的盒子中($n \geq k \geq 1$, 且盒子不允许为空)的方案数就是stirling

数.(即含 n 个元素的集合划分为 k 个集合的情况数)

递推公式:

$S(n,k) = 0$ ($k > n$)

$S(n,1) = 1$ ($k = 1$)

$S(n,k)=1$ ($n=k$)

$S(n,k) = S(n-1,k-1)+k*S(n-1,k)$ ($n \geq k \geq 2$)

分析: 设有 n 个不同的球, 分别用 b_1, b_2, \dots, b_n 表示。从中取出一个球 b_n , b_n 的放法有以下两种:

1. b_n 独占一个盒子, 那么剩下的球只能放在 $k-1$ 个盒子里, 方案数为 $S(n-1,k-1)$;

2. b_n 与别的球共占一个盒子, 那么可以将 b_1, b_2, \dots, b_{n-1} 这 $n-1$ 个球放入 k 个盒子里, 然后将 b_n 放

入其中一个盒子中, 方案数为 $k*S(n-1,k)$.

bell数和stirling数的关系为,

点赞3

评论2

分享

收藏3

打赏

举报

关注

一键三连

每个贝尔数都是"第二类Stirling数"的和。

B(n) = Sum(1,n) S(n,k).

此部分转载于 <http://www.cnblogs.com/xiaoxian1369/archive/2011/08/26/2154783.html> 含HDU两道例题

以下代码列出了斯特灵数

```
1 #include<stdio.h>
2 int main()//斯特灵数
3 {
4     char *w[100];
5     int n;
6     w[1]="1";
7     w[2]="3";
8     w[3]="13";
9     w[4]="75";
10    w[5]="541";
11    w[6]="4683";
12    w[7]="47293";
13    w[8]="545835";
14    w[9]="7087261";
15    w[10]="102247563";
16    w[11]="1622632573";
17    w[12]="28091567595";
18    w[13]="526858348381";
19    w[14]="10641342970443";
20    w[15]="230283190977853";
21    w[16]="5315654681981355";
22    w[17]="130370767029135901";
23    w[18]="3385534663256845323";
24    w[19]="92801587319328411133";
25    w[20]="2677687796244384203115";
26    w[21]="81124824998504073881821";
27    w[22]="2574844419803190384544203";
28    w[23]="85438451336745709294580413";
29    w[24]="2958279121074145472650648875";
30    w[25]="106697365438475775825583498141";
31    w[26]="4002225759844168492486127539083";
32    w[27]="155897763918621623249276226253693";
33    w[28]="6297562064950066033518373935334635";
34    w[29]="263478385263023690020893329044576861";
35    w[30]="11403568794011880483742464196184901963";
36    w[31]="510008036574269388430841024075918118973";
37    w[32]="23545154085734896649184490637144855476395";
38    w[33]="1120959742203056268267494209293006882589981";
39    w[34]="54984904077825684802426868390301049750104843";
40    w[35]="2776425695289206002630310219593685496163584253";
41    w[36]="144199280951655469628360978109406917583513090155";
42    w[37]="7697316738562185268347644943000493480404209089501";
43    w[38]="421985466101260424678587486718115935844245187819723";
44    w[39]="23743057231588741419119534567705900419786127935577533";
45    w[40]="1370159636942236704917645663312384364386256449136591915";
46    w[41]="81045623051154285047127402304207782853156976521592907421";
47    w[42]="491081297538957495431875959993938855544783946694910718603";
48    w[43]="304646637632091740261982544696657582136519552428876887346813";
49    w[44]="19338536506753895707368358095646384573117824953447578202397675";
50    w[45]="1255482482235481041484313695469155949742941807533901307975355741";
51    w[46]="83318804148028351409201335290659562069258599933450396080176273483";
52    w[47]="5649570401186486930330812460375430692673276472202704742218853260093";
53    w[48]="391229145645351175841837029639030040330277058716846008212321196523435";
54
55    w[49]="27656793065414932606012896651489726461435178241015434306518713649426461";
56    w[50]="1995015910118319790635433747742913123711612309013079035980385090523556363";
57    while(~scanf("%d",&n))58 | printf("%s\n",w[n]);
58
59    return 0;
60 }
```


/* 卡特兰数 1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, 208012, 742900, 2674440, 9694845, 3535...

BZOJ 1485: [HNOI2009]有趣的数列(catalan数)

weixin_30824599的博客 21

打个表找一下规律可以发现...就是卡特兰数...卡特兰数可以用组合数计算。对于这道题, ans(n) = C(n, 2n) / (n+1...

几种数学公式(环排列 母函数 唯一分解定理 卡特兰数 默慈金数 贝尔数 那... September_的博客 354

转自 <https://blog.csdn.net/ZCY19990813/article/details/81181505> —: 环排列 把一个m个元素的环在m个不同...

Codeforces 961G: 第二类斯特林数

辣鸡葫芦娃 424

题意: 给出一个数列 w_i , 对于一个集合 S , , 对于一个 w 的划分 R , , 求 w 的所有划分为 k 部分的 W 之和。题解: ...

斯特林数

wuxiaowu547的博客 3732

斯特林数分为第一类斯特林数和第二类斯特林数, 第一类斯特林数分为有符号斯特林数和无符号斯特林数; 1.什...

数的长度

sead+ 1368

<http://acm.nyist.net/JudgeOnline/problem.php?pid=69> 数的长度 时间限制: 3000 ms | 内存限制: 65535 KB 难...

杭电1018-Big Number

ecjtu_acm_菜鸟—鱼尾尾 782

Big Number Time Limit: 2000/1000 MS (Java/Others) Memory Limit: 65536/32768 K (Java/Others) Total Sub...

斯特林公式在 ACM 中的使用

ju136的专栏 3599

给两个斯特林公式的链接 http://episte.math.ntu.edu.tw/articles/mm/mm_17_2_05/page2.html [http://hi.baidu.c...](http://hi.baidu.com)

©2020 CSDN 皮肤主题: 编程工作室 设计师:CSDN官方博客 返回首页

关于我们 招贤纳士 广告服务 开发助手 ☎ 400-660-0108 📩 kefu@csdn.net 🌐 在线客服 工作时间 8:30-22:00

公安备案号11010502030143 京ICP备19004658号 京网文〔2020〕1039-165号 经营性网站备案信息 北京互联网违法和不良信息举报中心

网络110报警服务 中国互联网举报中心 家长监护 Chrome商店下载 ©1999-2021北京创新乐知网络技术有限公司 版权与免责声明 版权申诉

出版物许可证 营业执照

九日王朝 专家
码龄4年 企业员工

201 原创 9938 周排名 6214 总排名 71万+ 访问 等级

8397 积分 935 粉丝 490 获赞 200 评论 717 收藏

私信 关注

Q

九日王朝

热门文章

[斗地主AI算法——第一章の业务逻辑](#) 37885

[python——Tkinter图形化界面及threading多线程](#) 33428

[VS2017——50G超豪华IDE套餐酸爽体验!](#) 30952

[用代码证明自己闲的蛋疼 \(一\) ——cmd闪瞎狗眼](#) 22388

[python——wxpy模块实现微信尬聊 \(基于图灵机器人\)](#) 19047

分类专栏

xxl-job 1篇

系统性能 2篇

Elasticsearch 5篇



最新评论

python——Tkinter图形化界面及threadin...

Tisfy: 深得人心，正如古人云：宣父犹能畏后生，丈夫未可轻年少。

最短路径——Floyd算法及优化（蓝桥杯...
elecstar: 这还带“故事未完结，请听下回讲解的吗~！”

人工智能学习笔记——机器学习(14)mds...

qq_42757379: 想知道，新样本怎么映射到低维空间中

python——Tkinter图形化界面及threadin...

每天都很兴奋: 代码为python2代码 已经用不成了 思想可以借鉴

redis——redis主从复制

虚幻私塾: 太实用了，准备一试。

最新文章

分布式任务调度平台XXL-JOB搭建&使用

系统故障分析常用方法

profiling定位nodejs程序消耗情况

2021年 3篇

2020年 7篇

2019年 7篇

2018年 21篇

2017年 74篇

2016年 91篇

广告 | 关闭

有奖问卷调查

参与问卷赢取精美礼品



立即参与



努力努力再努力x

博客园 首页 新随笔 联系 订阅 管理

大数因式分解 Pollard_rho 算法详解

给你一个大数n，将它分解它的质因子的乘积的形式。

首先需要了解Miller_rabin判断一个数是否是素数

大数分解最简单的思想也是试除法，这里就不再展示代码了，就是从2到 \sqrt{n} ，一个一个的试验，直到除到1或者循环完，最后判断一下是否已经除到1了即可。

但是这样的做的复杂度是相当高的。一种很妙的思路是找到一个因子（不一定是质因子），然后再一路分解下去。这就是基于Miller_rabin的大数分解法Pollard_rho大数分解。

Pollard_rho算法的大致流程是先判断当前数是否是素数

（Miller_rabin）了，如果是则直接返回。如果不是素数的话，试图找到当前数的一个因子（可以不是质因子）。然后递归对该因子和约去这个因子的另一个因子进行分解。

那么自然的疑问就是，怎么找到当前数n的一个因子？当然不是一个一个慢慢试验，而是一种神奇的想法。其实这个找因子的过程我理解的不是非常透彻，感觉还是有一点儿试的意味，但不是盲目的枚举，而是一种随机化算法。我们假设要找的因子为p，他是随机取一个 x_1 ，由 x_1 构造 x_2 ，使得 {p可以整除 $x_1-x_2 \&& x_1-x_2$ 不能整除n} 则 $p=\gcd(x_1-x_2, n)$ ，结果可能是1也可能不是1。如果不是1就找寻成功了一个因子，返回因子；如果是1就寻找失败，那

公告



昵称：_努力努力再努力x

园龄：3年1个月

粉丝：63

关注：2

+加关注

2021年5月						
日	一	二	三	四	五	六
25	26	27	28	29	30	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31	1	2	3	4	5

搜索

找找看

谷歌搜索

随笔分类

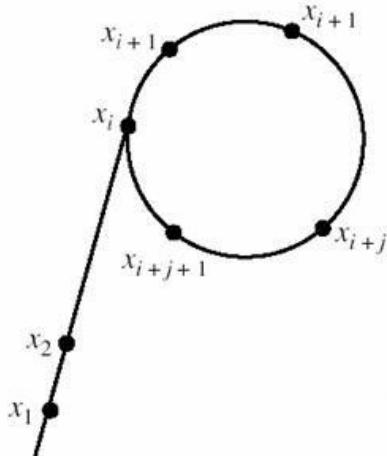
ACM基础篇(330)

ACM进阶篇(16)

ACM准备篇(4)

么我们就要不断调整 x_2 , 具体的办法通常是 $x_2=x_2*x_2+c$ (c 是自己定的) 直到出现 x_2 出现了循环 $=x_1$ 了表示 x_1 选取失败重新选取 x_1 重复上述过程。 (似乎还存在一个每次找寻范围*2的优化, 但是不太懂。。。)

因为 x_1 和 x_2 再调整时最终一定会出现循环, 形成一个类似希腊字母rho的形状, 故因此得名。



另外通过find函数来分解素数, 如果找到了一个素数因子则加入到因子map中, 否则如果用Pollard找到一个因子则递归去找素数因子。

```

1 #include<iostream>
2 #include<ctime>
3 #include<algorithm>
4 #include<map>
5 using namespace std;
6 typedef long long ll;
7 map<ll, int>m;
8 const int mod = 10000019;
9 const int times = 50; //测试50次
10 ll mul(ll a, ll b, ll m)
11 //求a*b%m
12 {
13     ll ans = 0;
14     a %= m;
15     while(b)
16     {
17         if(b & 1)ans = (ans + a) % m;
18         b /= 2;
19         a = (a + a) % m;
20     }
21     return ans;
22 }
23 ll pow(ll a, ll b, ll m)
24 //a^b % m

```

BZOJ(50)

Codeforces(3)

Contest_Self(5)

SGU刷题之路(8)

URAL刷题之路(2)

动态规划(43)

动态规划---背包DP(22)

动态规划---概率DP(3)

动态规划---基础DP(13)

动态规划---区间DP(3)

动态规划---树形DP(2)

动态规划---状态压缩DP(3)

更多

随笔档案

2019年4月(1)

2018年10月(5)

2018年9月(52)

2018年8月(6)

2018年7月(16)

2018年6月(2)

2018年5月(93)

2018年4月(239)

```

25 {
26     ll ans = 1;
27     a %= m;
28     while(b)
29     {
30         if(b & 1)ans = mul(a, ans, m);
31         b /= 2;
32         a = mul(a, a, m);
33     }
34     ans %= m;
35     return ans;
36 }
37 bool Miller_Rabin(ll n, int repeat)//n是测试的大数, repeat是测试重复次数
38 {
39     if(n == 2 || n == 3) return true;//特判
40     if(n % 2 == 0 || n == 1) return false;//偶数和1
41
42     //将n-1分解成2^s*d
43     ll d = n - 1;
44     int s = 0;
45     while(!(d & 1)) ++s, d >= 1;
46     // srand((unsigned)time(NULL));在最开始调用即可
47     for(int i = 0; i < repeat; i++)//重复repeat次
48     {
49         ll a = rand() % (n - 3) + 2;//取一个随机数,[2,n-1)
50         ll x = pow(a, d, n);
51         ll y = 0;
52         for(int j = 0; j < s; j++)
53         {
54             y = mul(x, x, n);
55             if(y == 1 && x != 1 && x != (n - 1)) return false;
56             x = y;
57         }
58         if(y != 1) return false;//费马小定理
59     }
60     return true;
61 }
62 ll gcd(ll a, ll b)
63 {
64     return b == 0 ? a : gcd(b, a % b);
65 }
66 ll pollard_rho(ll n, ll c)//找到n的一个因子
67 {
68     ll x = rand() % (n - 2) + 1;
69     ll y = x, i = 1, k = 2;
70     while(1)
71     {
72         i++;
73         x = (mul(x, x, n) + c) + n;//不断调整x
74         ll d = gcd(y - x, n);
75         if(1 < d && d < n)
76             return d;//找到因子
77         if(y == x)
78             return n;//找到循环, 返回n, 重新来
79         if(i == k)//一个优化
80         {
81             y = x;
82             k <= 1;
83         }
84     }
85 }
86 void Find(ll n, ll c)
87 {

```

2018年3月(25)

最新评论

1. Re:POJ-1700 Crossing River---过河问题 (贪心)

请问如何证明这个贪心策略的正确性呢?

--vcjmhg

2. Re:KM算法 (运用篇)

一次增广路中求出的slack值会更准确, 循环次数比全局变量更少 为啥啊?

--T_Orang

3. Re:组合数取模方法总结 (Lucas定理介绍)

%%%%%%%%%%%%%%

--Rain罗

4. Re:最小生成树之kruskal算法

%%,这个启发式合并必须赞

--咯咯的C

5. Re:组合数取模方法总结 (Lucas定理介绍)

orz

txdy! ! !

--huyinghao

阅读排行榜

1. 数论专题 (一) 数论基本概念(18414)

2. 最小生成树之prim算法(16300)

3. 大数因式分解 Pollard_rho 算法详解(13460)

4. 单源最短路径---Dijkstra算法(11144)

```

88     if (n == 1) return; //递归出口
89
90     if (Miller_Rabin(n, times)) //如果是素数, 就加入
91     {
92         m[n]++;
93         return;
94     }
95
96     ll p = n;
97     while (p >= n)
98         p = pollard_rho(p, c--); //不断找因子, 知道找到为止, 返回n说明没找到
99
100    Find(p, c);
101    Find(n / p, c);
102 }
103 int main()
104 {
105     ll n; srand((unsigned)time(NULL));
106     while (cin >> n)
107     {
108         m.clear();
109         Find(n, rand() % (n - 1) + 1); //这是自己设置的一个数
110         cout << n << " = ";
111         for (map<ll, int>::iterator it = m.begin(); it != m.end();)
112         {
113             cout << it->first << " ^ " << it->second;
114             if (++it) != m.end())
115                 cout << " * ";
116         }
117     }
118     cout << endl;
119 }
120     return 0;
121 }
```



NOIP普及组、提高组培训，有意可加微信fu19521308684

分类: ACM基础篇, 数学, 数学---数论

[好文要顶](#)
[关注我](#)
[收藏该文](#)


_努力努力再努力x
关注 - 2
粉丝 - 63

+加关注

« 上一篇: 大素数测试的Miller-Rabin算法
» 下一篇: hdu-2879 hehe---积性函数

posted @ 2018-05-16 19:28 _努力努力再努力x 阅读(13461) 评论(1) 编辑 收藏

5. KM算法 (运用篇) (10378)

评论排行榜

1. 组合数取模方法总结 (Lucas定理介绍)

(7)

2. SGU刷题之路开启(2)

3. KM算法 (运用篇) (2)

4. 最小生成树之prim算法(2)

5. 大数因式分解 Pollard_rho 算法详解(1)

推荐排行榜

1. 组合数取模方法总结 (Lucas定理介绍)

(5)

2. KM算法 (运用篇) (5)

3. 大数因式分解 Pollard_rho 算法详解(3)

4. 网络流 (二) 最大流的增广路算法(3)

5. EOJ-3300 奇数统计 (高维前缀和) (2)

[刷新评论](#) [刷新页面](#) [返回顶部](#)

登录后才能查看或发表评论，立即 [登录](#) 或者 [逛逛](#) 博客园首页

【推荐】阿里云爆品销量榜单出炉，精选爆款产品低至0.55折

【推荐】7大类400多种组件，HarmonyOS鸿蒙三方库来了，赶紧收藏！

【推荐】大型组态、工控、仿真、CAD\GIS 50万行VC++源码免费下载！

【推荐】限时秒杀！国云大数据魔镜，企业级云分析平台

园子动态：

- 致园友们的一封检讨书：都是我们的错
- 数据库实例 CPU 100% 引发全站故障
- 发起一个开源项目：博客引擎 fluss

最新新闻：

- 菜鸟：2021财年全年收入372.5亿元，同比增68%
- 阿里第四财季营收1874亿元，净亏损54.79亿元
- 小鹏汽车：Q1营收29.5亿元 净亏损7.866亿元
- B站发布Q1财报：总营收39亿元，同比增长68%
- HTC发布VIVE FOCUS 3等系列新品 虚拟代言人也来了

» 更多新闻...

Copyright © 2021 _努力努力再努力x

Powered by .NET 5.0 on Kubernetes

这是 Google 对 https://blog.csdn.net/qq_41603898/article/details/82722116 的缓存。这是该网页在 2021 年 5 月 12 日 18:08:19 GMT 的快照。当前页在此期间可能已经更改。[了解详情](#)

[完整版本](#) [纯文字版本](#) [查看源代码](#)

提示：要在此页面上快速找到您的搜索字词，请按 **Ctrl+F** 或者 **⌘+F** (Mac)，然后使用查找栏搜索。

杜教BM---解决线性递推

原创 wym_king 2018-09-16 11:57:23 698 收藏 2
分类专栏： [模板](#)

版权

无论是矩阵快速幂求第n项，还是给出输出前几项求规律的第n项，几乎就没有它做不到的

BM推线性递推式，最低阶的复杂度好像是 $n^2 \log(n)$ 。n是输入项数，比高斯消元算快很多。

对于k阶递推式至少要输入 $2k$ 项才能有足够大参数解出方程系数。

一阶是只这个递推数列只针对前一项有效，例如 $a(n+1)=2a(n)$ 。如果是针对前两项，则叫二阶，比如斐波那契数列。

又比如 $a(n) = 2*a(n-1) + a(n-2) - 2*a(n-3) - a(n-4)$, $n > 3$ 就是四阶

线性说白了就是一次函数关系。

```
#include<bits/stdc++.h>
using namespace std;
#define rep(i,a,n) for (int i=a;i<n;i++)
#define pb push_back
typedef long long ll;
#define SZ(x) ((ll)(x).size())
typedef vector<ll> VI;
typedef pair<ll,ll> PII;
const ll mod=998244353;
ll powmod(ll a,ll b) {
    ll res=1;
    a%=mod;
    assert(b>=0);
    for(; b; b>>=1) {
        if(b&1)res=res*a%mod;
        a=a*a%mod;
    }
    return res;
}
ll _n;
namespace linear_seq {
    const ll N=10010;
    ll res[N],base[N],_c[N],_md[N];

    vector<ll> Md;
    void mul(ll *a,ll *b,ll k) {
        rep(i,0,k+k) _c[i]=0;
        rep(i,0,k) if (a[i]) rep(j,0,k) _c[i+j]=(_c[i+j]+a[i]*b[j])%mod;
        for (ll i=k+k-1; i>=k; i--) if (_c[i])
            rep(j,0,SZ(Md)) _c[i-k+Md[j]]=(_c[i-k+Md[j]]-_c[i]*_md[Md[j]])%mod;
        rep(i,0,k) a[i]=_c[i];
    }
    ll solve(ll n,VI a,VI b) { // a 系数 b 初值 b[n+1]=a[0]*b[n]...
        ll ans=0,pnt=0;
        ll k=SZ(a);
        assert(SZ(a)==SZ(b));
        rep(i,0,k) _md[k-1-i]=-a[i];
        _md[k]=1;
        Md.clear();
        rep(i,0,k) if (_md[i]!=0) Md.push_back(i);
        rep(i,0,k) res[i]=base[i]=0;
        res[0]=1;
        while ((1ll<<pnt)<=n) pnt++;
        for (ll p=pnt; p>=0; p--) {
            mul(res,res,k);
            if ((n>>p)&1) {
                for (ll i=k-1; i>=0; i--) res[i+1]=res[i];
                res[0]=0;
                rep(j,0,SZ(Md)) res[Md[j]]=(res[Md[j]]-res[k]*_md[Md[j]])%mod;
            }
        }
        rep(i,0,k) ans=(ans+res[i]*b[i])%mod;
        if (ans<0) ans+=mod;
        return ans;
    }
    VI BM(VI s) {
}
```



举报

```

VI C(1,1),B(1,1);
11 L=0,m=1,b=1;
rep(n,0,SZ(s)) {
    11 d=0;
    rep(i,0,L+1) d=(d+(11)*C[i]*s[n-i])%mod;
    if (d==0) ++m;
    else if (2*L<=n) {
        VI T=C;
        11 c=mod-d*powmod(b,mod-2)%mod;
        while (SZ(C)<SZ(B)+m) C.pb(0);
        rep(i,0,SZ(B)) C[i+m]=(C[i+m]+c*B[i])%mod;
        L=n+1-L;
        B=T;
        b=d;
        m=1;
    } else {
        11 c=mod-d*powmod(b,mod-2)%mod;
        while (SZ(C)<SZ(B)+m) C.pb(0);
        rep(i,0,SZ(B)) C[i+m]=(C[i+m]+c*B[i])%mod;
        ++m;
    }
}
return C;
}
11 gao(VI a,11 n) {
    VI c=BM(a);
    c.erase(c.begin());
    rep(i,0,SZ(c)) c[i]=(mod-c[i])%mod;
    return solve(n,c,VI(a.begin(),a.begin()+SZ(c)));
}
};

int main() {
    while (~scanf("%lld",&n)) {
        vector<11>v;
        v.push_back(0); //前几项
        v.push_back(1);
        v.push_back(2);
        v.push_back(5);
        v.push_back(10);
        v.push_back(20);
        v.push_back(38);
        v.push_back(71);
        v.push_back(130);
        v.push_back(235);
        v.push_back(420);
        v.push_back(744);
        v.push_back(1308);
        //输入n ,输出第n项的值
        printf("%lld\n",linear_seq::gao(v,n-1));
    }
}
}

```



杜教BM模版(推规律)

Ivan_zcy的博客 473

无论是矩阵快速幂求第n项，还是给出输出前几项求规律的第n项。。。几乎没有它做不到的~至于原...

杜教板子 (BM) 线性递推式

小灰狼与大白兔的博客 434

据说这个模版可以解决任何线性递推式，听说是杜教的，只要我们手推递推式的前几项，然后扔进这个板...



优质评论可以帮助作者获得更高权重



评论

相关推荐

杜教BM(解决线性递推式的模版)

阿狸的博客 355

把递推式前几项扔进去就行了，至少扔个8项，然后mod根据题意改改，就能出结果了。太神奇了。#inclu...

BM求线性递推式(板子整理)

Code92007的博客 192

心得 杜老师的奇技淫巧，绝大多数不懂原理，但只要套板子就行了 板子整理 以2019牛客多校B题为例，求...

BM求线性递推式

蒟蒻 lxw的博客 704

只能解决常系数线性递推式 要求数域中每个的非0数存在乘法逆元 #include<bits/stdc++.h> using nam...

杜教BM (线性齐次递推式推演,无define)

平凡人Kalzn的博客 157

下面是模板代码: #include <cstdio> #include <cstring> #include <cmath> #include <algorithm> #include <v...



最短线性递推式

杜教bm -- 找规律大法好这是一个能够**线性递推**求规律的模板: #include <bits/stdc++.h> using namespace std; #define rep(i,a,...)**杜教(BM)**黑科技, 求**递推**公式 代码: #include <iostream> #include <cstring> #include <cmath> #include ...**杜教BM黑科技****杜教BM**黑科技 原理未知, 主要适用于**线性递推**式, 据说这个BUG级模板可以求**线性递推**式的第n项, 只要...**杜教BM模板** (仅适用于**线性递推**式)

Pandapan1997的博客 268

据说这个模板可以**解决任何线性递推**式, 听说是**杜教**的, 只要我们手推**递推**式的前几项, 然后扔进这个板...**杜教BM板子解决线性递推问题**

neuq_zsmj的博客 256

https://codeforces.com/contest/1117/problem/D 转自https://blog.csdn.net/qq_37632935/article/details/878...

杜教板子 (BM) 线性递推式 (板子)

weixin_38168590的博客 53

杜教板子 (BM) 线性递推式 解决传统线性递推**式神器 1 #include <cstdio> 2 #include <cstring> 3 #include ...**洛谷 P5487 【模板】**线性递推+BM算法 (BM+多项式取模)**

苟为蒟蒻又何妨 537

传送门 代码: #include<bits/stdc++.h> #define ri register int using namespace std; const int rlen=1<<18|1...

BM递推杜教版

qq_36876305的博客 2305

#include <iostream> using namespace std; #define rep(i,a,n) for (long long i=a;j<n;i++) #define ...

黑科技之**杜教bm**

weixin_30375427的博客 45

这个板子能够**解决任何线性递推**式, 只要你确定某个数列的某项只与前几项**线性**相关, 那么把它前若干项...**杜教BM模板**

Orz 197

据说这个BUG级模板可以求**线性递推**式的第n项, 只要手推**递推**式的前几项, 放入模板就能求出第n项...**杜教BM**

XFire的博客 320

#include <iostream> #include <cstring> #include <cmath> #include <algorithm> #include ...

杜教BM模板

Roar__的博客 107

矩阵快速幂+**线性递推** #include <bits/stdc++.h> using namespace std; #define rep(i,a,n) for (long long i=a;j<n;i++) #define ...**杜教BM递推**

cj1064789374的博客 160

杜教BM递推

©2020 CSDN 皮肤主题: 编程工作室 设计师:CSDN官方博客 返回首页

关于我们 招贤纳士 广告服务 开发助手 ☎ 400-660-0108 📩 kefu@csdn.net 💬 在线客服 工作时间 8:30-22:00

公安备案号11010502030143 京ICP备19004658号 京网文[2020]1039-165号 经营性网站备案信息

北京互联网违法和不良信息举报中心 网络110报警服务 中国互联网举报中心 家长监护 Chrome商店下载

©1999-2021北京创新乐知网络技术有限公司 版权与免责声明 版权申诉 出版物许可证 营业执照



wym_king
码龄3年  暂无认证

463	2万+	7056	48万+	
原创	周排名	总排名	访问	等级
8360	113	252	72	455
积分	粉丝	获赞	评论	收藏

[私信](#) [关注](#)

[搜博主文章](#) 

热门文章

opencv imwrite函数参数详解+例子 30660

结点和节点的区别 29090

ret, frame = cap.read() read函数返回值 26107

关于为什么要用 if cv2.waitKey(1) & 0xFF == ord('q'): break的解释 23042

严重性 代码 说明 项目 文件 行 禁止显示状态 错误 C3861 "time": 找不到标识符 WindowsProject1 d:\新建文件夹 \windowsproject1\windowspro 22098



分类专栏

	numpy	付费	2篇
	c++相关知识		15篇
	linux		25篇
	俄罗斯方块		10篇
	树状数组		15篇
	树		9篇

▼

最新评论

[Poetize6] IncDec Sequence 差分
why151: tql
opencv imwrite函数参数详解+例子
Tisfy: 他能使使人有三月不知肉味，使人有余
音穿梁，三日不绝的感受
attributeerror: 'nonetype' object has no at...
deepindeed: windows下图片所在路径中不
要有中文，否则找不到
Hash (散列) 冲突解决 线性探测再散列...
星辰诀: 抛开吸引人的标题，博主开发的精
神值得学习。
静态主席树 超详细！！！不看后悔一生
星辰诀: 很棒呀，学习啦，谢谢分享！

最新文章

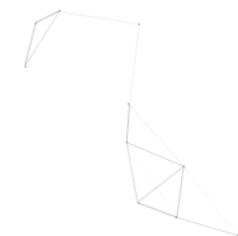
numpy 安装
P1198 [JSOI2008]最大数 线段树入门
P3119 [USACO15JAN]草鉴定Grass
Cownoisseur 缩点 topo或最长路

2021年 1篇 2019年 271篇
2018年 247篇

目录



举报



+ -
REVOLVERMAPS

peng-ym

从头再来，重新上线

首页

新随笔

联系

管理

随笔 - 17 文章 - 0 评论 - 127 阅读 - 10

杜教筛

杜教筛

(似乎有很多人在催我的杜教筛呢.....)

前言

- 话说，我是不是在自己的莫比乌斯反演中挖了许多杜教筛的坑啊.....
- 本文完整的总结介绍杜教筛，也算是将莫比乌斯反演中的坑全部填满吧！
- 真诚地希望来阅读这篇学习笔记的每一个人，仔仔细细的看完每一段。
- 我相信，只要认真的看完整篇文章并跟着一起思考的读者，一定能够有所收获！
- 如果您之前不会杜教筛，那么我希望这篇文章能够作为您学习杜教筛路上的有力援助，帮助您真正的了解与掌握杜教筛！
- 如果您之前早已熟知杜教筛或只有些模糊的印象，相信您一定也能有所收获！
- (PS：本文较长，请耐心阅读 ovo)

在OI中的意义

- 其实，对于一般的数论题，线性筛已经非常的优秀了。
- 但是就是有那些duliu出题人，硬是要把数据出到 $1e10$ 之类的，就看你会不会杜教筛，min_25筛，洲阁筛等各种神奇的筛法。(PS：后面这两个筛法我是真的不会QAQ)
- 要是不会，那就要少十分左右！
- 所以，专门用杜教筛来推式子的题目很少，一般都是用杜教筛优化线性筛，弄到最后的那些分。
- 不过，杜教筛的思想对于推式子是很有帮助的。（它那种递归求解的形式，以及复杂度 $O(n^{\frac{2}{3}})$ ）
- 因此，学会杜教筛也是一件挺好的事情！

前置技能

公告



一言 (ヒトコト)

昵称： pengym

园龄： 3年10个月

粉丝： 124

关注： 0

+加关注

2021年5月						
日	一	二	三	四	五	六
25	26	27	28	29	30	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31	1	2	3	4	5

常用链接

我的随笔

我的评论

我的参与

最新评论

我的标签

我的标签

洛谷(8)

莫比乌斯反演(6)

学习笔记(4)

模拟退火(2)

HNOI(2)

杜教筛(1)

数据结构(1)

- 杜教筛的前置技能挺多的.....

各种函数

概念

- 首先，我们需要知道有一个东西叫做**数论函数**
- 数论函数有很多种，但是我们身为Oier，并不需要知道它的具体的定义，具体的分类。
- 我们只需要知道，我们在OI中的数论中所用到的各种函数 μ, φ 等都是数论函数。（后面会将常见的都列举出来，当然不只这两种）。
- 当你了解**数论函数**后，你就需要知道有一类函数叫做**积性函数**。
- 还是同样的话语，我们平常所惯用的数论函数都是积性函数！
- 不过，对于积性函数的定义还是有必要了解一下。（毕竟有些函数看上去不常见，其实可能就是积性函数！）
- 积性函数定义：如果已知一个函数为数论函数，且 $f(1) = 1$ ，并且满足以下条件，若对于任意的两个互质的正整数 p, q 都满足 $f(p \cdot q) = f(p) \cdot f(q)$ ，那么则称这个函数为**积性函数**。
- 特殊的，如果当对于任意的正整数 p, q （即不一定互质），也满足以上这个式子，则称这个函数为**完全积性函数**。
- 而我们的**杜教筛**，则是用来筛积性函数前缀和的神奇筛法！！！
- 说了这么多概念性的东西，不如来点实质性的！

常见积性函数

1. $\mu(n)$ —莫比乌斯函数。关于这个函数，我在莫比乌斯反演中说的挺清楚的了(233)，(PS: 不过我将会在下文中，从另一种角度介绍它的性质。也算是把坑给填完吧)
2. $\varphi(n)$ —欧拉函数。表示不大于 n 且与 n 互质的正整数个数，十分常见的数论函数。用数学式子表示即：

$$\varphi(n) = \sum_{i=1}^n [(\text{gcd}(n, i) = 1)]$$
 (PS:(n, i)表示 $\text{gcd}(n, i)$)
3. $d(n)$ —约数个数。表示 n 的约数的个数。用式子表示为：

$$d(n) = \sum_{d|n} 1$$
，也可以写作：

$$d(n) = \sum_{d=1}^n [d|n]$$
 (其实没什么太大区别啦！)
4. $\sigma(n)$ —约数和函数。即 n 的各个约数之和。表示为：

$$\sigma(n) = \sum_{d|n} d = \sum_{d=1}^n [d|n] \cdot d$$

(PS: 接下来列举的是完全积性函数)

(PS: 代表字母可能会与他人的略有不同，似乎在数学中没有统一的字母)

1. $\epsilon(n)$ —元函数。似乎也有人把它叫作 $e(n)$? 其实无所谓啦~我们只需要知道 $\epsilon(n) = [n = 1]$ 。(看到这个是不是有种莫名的熟悉感呢？到了下文中，就会发现这种熟悉感是从哪来的啦！)
2. $I(n)$ —恒等函数。所谓恒等就是这个函数的值恒为1。
3. $id(n)$ —单位函数。 $id(n) = n$.

(当第一次看到这些完全积性函数的时候，是不是有人感觉这些完全积性函数毫无用处，都是一些简单的式子，只不过用符号表示了呢？在下一个前置技能—狄利克雷卷积中，你应该就会改变自己naive的想法啦~)

狄利克雷卷积 (*)

基本知识

- 听名字，似乎是一个很高深的东西。
- 其实，若是不理睬这个名字，只是把它当作一个新定义的符号，你应该就会发现，狄利克雷卷积也不是那样的难理解。
- 定义：两个数论函数 f 和 g 的卷积为 $(f * g)(n) = \sum_{d|n} f(d) \cdot g\left(\frac{n}{d}\right)$ 。前面的括号代表将 f 卷 g ，后面的括号代表范围。(PS: 后面的括号一般可以省略不写，默认为 n)
- 很显然，狄利克雷卷积满足以下运算规律：
 1. 交换律($f * g = g * f$);
 2. 结合律($(f * g) * h = f * (g * h)$);

可持久化并查集(1)

JSOI(1)

感受(1)

更多

积分与排名

积分 - 32926

排名 - 35506

随笔档案

2019年1月(1)

2018年8月(2)

2018年6月(3)

2018年4月(1)

2018年3月(10)

友情链接

zjp_shadow大佬的blog

CVH大佬的blog

Orange大佬的blog

蒟蒻自己在洛谷的blog

y1so1大佬的blog

Hany01大佬的blog

Shichengxiao大佬的blog

dyx大佬的blog

redbag大佬的blog

gaylunch大佬的blog

lstete大佬的blog

chinhhh大佬的blog

最新评论

1. Re:莫比乌斯反演

如果对和式变换看不懂的可以看看维基百科里面的讲解，很好懂

--LEEEEEEEEEEEEEE

2. Re:杜教筛

啧啧，这证明太香了，妙啊

--blanc

3. Re:[SDOI2015]约数个数和

orz

--ACwishes

4. Re:记OI退役

@思益 bjtu...

--peng

5. Re:记OI退役

大佬现在在哪里

--思益

阅读排行榜

1. 莫比乌斯反演(32627)

2. 杜教筛(26310)

3. 整除分块(13053)

4. 洛谷【P2257】YY的GCD(7291)

5. 可持久化并查集(6127)

6. 模拟退火(5053)

7. [POI2007]ZAP-Queries(3396)

8. [SDOI2015]约数个数和(2915)

9. 记OI退役(1873)

10. 洛谷P1829 [国家集训队]Crash的文字表格(1806)

评论排行榜

1. 莫比乌斯反演(36)

2. 杜教筛(30)

3. 整除分块(14)

4. 记OI退役(9)

5. 洛谷P1829 [国家集训队]Crash的文字表格(9)

推荐排行榜

1. 莫比乌斯反演(37)
2. 杜教筛(36)
3. 整除分块(15)
4. 洛谷【P2257】YY的GCD(8)
5. 模拟退火(7)

3. 分配律($(f + g) * h = f * h + g * h$);

- 在记忆方面，可以类比为乘法的运算法则，其实上面这几条运算规律是可以证明的！
- 举个例子，交换律。我们看狄利克雷卷积的式子，实质上就是 n 的每一个约数带入 f 中的值，乘上与之对应的约数在 g 中的值。
- 显然，当交换 f 和 g 时，仅仅时枚举约数的顺序发生了改变，而每一个约数对答案的贡献是不会改变的。因此存在交换律！
- 在大致了解了狄利克雷卷积的运算法则后，我们就需要提到上面所说的积性函数啦！
- 首先，元函数 ϵ 。所谓元函数，指的就是在狄利克雷卷积中充当单位元的作用，单位元即满足： $f * \epsilon = f$ 。不要小看这个元函数，当元函数配合上结合律时就可以用来证明一些结论啦~
- 除了元函数之外，我们最为常见的则是 μ, φ 之类的的函数，因此我们需要十分熟练它们与一些常见的完全积性函数的卷积，以及性质。

(PS：特别要记住一点：积性函数有一个特别重要的性质，那就是（积性函数*积性函数）仍然为积性函数！！！
这个性质可以用来判断能否被杜教筛！)

莫比乌斯函数 μ

- μ 。在莫比乌斯反演中，我们曾了解过一个与 μ 有关的性质： $\sum_{d|n} \mu(d) = [n = 1]$
- 我们将这个性质表示成狄利克雷卷积的形式即： $\mu * I = \epsilon$ 。这在狄利克雷卷积中是一个很常用的恒等式。当然，有了它，我们也能够证明出莫比乌斯反演啦！
- 开始填坑，证明莫比乌斯反演：

已知：

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

用狄利克雷卷积的形式表示这个式子即： $F = f * I$

利用狄利克雷卷积将 F 卷上 μ ，得到：

$$F * \mu = f * I * \mu$$

由于狄利克雷卷积具有结合律与交换律，因此原式可化为：

$$\rightarrow f * (I * \mu) = f * \epsilon = f$$

即： $f = F * \mu$ 。代入后即可证明莫比乌斯反演： $f(n) = \sum_{d|n} \mu(d) \cdot F\left(\frac{n}{d}\right)$

同理，自然也可以得到莫比乌斯反演的另一种形式： $f(n) = \sum_{n|d} \mu\left(\frac{d}{n}\right) \cdot F(d)$

(总算填完一个大坑……)

欧拉函数 φ

- φ 。欧拉函数有一个很著名的性质： $\sum_{d|n} \varphi(d) = n$ 。
- 与以上方法类似，我们将它表示成狄利克雷卷积的形式： $\varphi * I = id$ 。
- 这时候，看到这个式子我们会有一个大胆的想法，既然在这个欧拉函数与莫比乌斯函数的式子中都有 I ，那么我们不如将这个式子的两边同时卷上一个 μ 。
- 于是，我就可以开始填第二个坑了—欧拉函数与莫比乌斯函数的关系。

$$\begin{aligned} \varphi * I &= id \\ \rightarrow \varphi * I * \mu &= id * \mu \\ \rightarrow \varphi * \epsilon &= id * \mu \end{aligned}$$

即： $\varphi = id * \mu \rightarrow \varphi(n) = \sum_{d|n} \mu(d) \cdot \frac{n}{d}$

- 我们把这个式子的两边同时除以 n ，则可以推出这个巧妙的式子：

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

(至此，我终于把莫比乌斯反演中的坑填完啦~~23333)

(有关杜教筛的前置技能也说的差不多啦，终于可以步入正题啦！)



步入正题——杜教筛

- 说了这么久，终于可以开始讲杜教筛啦！（是不是有一种莫名的兴奋呢？）
- 首先，我们应该弄清楚一个问题：杜教筛到底是用来干什么的？
- 杜教筛是以低于线性的时间复杂度来计算积性函数的前缀和的神奇筛法！
- 即我们需要计算的式子为： $\sum_{i=1}^n f(i)$ ($f(i)$ 为积性函数)
- PS：接下来要讲解的是杜教筛的套路式，如果不懂为什么要这样做，也没有关系。只需要明白它是怎么推过来的就行了。实在看不懂就记个结论吧.....
- 推式子时间到！
- 为了解决这个问题，我们构造两个积性函数 h 和 g 。使得 $h = f * g$
- 现在我们开始求 $\sum_{i=1}^n h(i)$ 。
- 记 $S(n) = \sum_{i=1}^n f(i)$ 。

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

接着，我们将右边式子的第一项给提出来，可以得到：

$$\begin{aligned}\sum_{i=1}^n h(i) &= g(1) \cdot S(n) + \sum_{d=2}^n g(d) \cdot S\left(\lfloor \frac{n}{d} \rfloor\right) \\ \rightarrow g(1)S(n) &= \sum_{i=1}^n h(i) - \sum_{d=2}^n g(d) \cdot S\left(\lfloor \frac{n}{d} \rfloor\right)\end{aligned}$$

其中的 $h(i) = (f * g)(i)$ ；

- 这就是杜教筛的惯用套路式。经各种分析，只要当你的 $h(i)$ 的前缀和很好求，能在较短的时间内求出，那么当我们对后面的式子进行整除分块时，求 $S(n)$ 的复杂度为 $O(n^{\frac{2}{3}})$
- 当我们知道了这个套路式后，可能会思考，我们应该如何选择这个 g 与 h 呢？
- 对于这个疑问，我没有太好的回答。只能说，依靠平时对于狄利克雷卷积中的各种式子的熟悉，以及仔细观察式子的能力啦！（当然我下面也会介绍一种小方法啦~OVO）
- 知道了这个套路式总要练练手吧！

应用

(PS：以下例子中，假设线性筛均跑不过)

一：求 $S(n) = \sum_{i=1}^n \mu(i)$ ；

- 根据那个套路式： $g(1)S(n) = \sum_{i=1}^n (f * g)(i) - \sum_{d=2}^n g(d) \cdot S\left(\lfloor \frac{n}{d} \rfloor\right)$ ，我们只需要找一个积性函数 g 使得这个函数与 μ 的卷积的前缀和容易求。如果你认真的看了上文，应该就可以很轻松的想到一个积性函数 I 。
- 我们知道 $\mu * I = \epsilon$ ，很显然，单位元的前缀和非常好求，就是1，并且 I 十分方便整除分块。所以我们把这个积性函数带入上述式子中可以得到：

$$S(n) = 1 - \sum_{d=2}^n S\left(\lfloor \frac{n}{d} \rfloor\right)$$

因此，我们就学会了杜教筛莫比乌斯函数的前缀和啦！

二：求 $S(n) = \sum_{i=1}^n \varphi(i)$

- 与求莫比乌斯函数的思路类似。



- 我们在脑海中找到一个与欧拉函数有关的卷积式子: $\varphi * I = id$
- 我们可以发现, 在筛欧拉函数前缀和所选择的积性函数 g 同样也是 I 哟! 代入得:

$$S(n) = \sum_{i=1}^n i - \sum_{d=2}^n S(\lfloor \frac{n}{d} \rfloor)$$

前面那个式子可以利用等差数列求和公式 $O(1)$ 的计算出结果, 后面同样利用整除分块。
所以, 我们又学会了如何筛欧拉函数的前缀和啦!

三: 求 $S(n) = \sum_{i=1}^n i \cdot \varphi(i)$

- 这个式子是不是无法一眼看出需要配什么积性函数了呢?
- 我们考虑狄利克雷卷积的形式: $\sum_{d|n} (d \cdot \varphi(d)) \cdot g(\frac{n}{d})$
- 我们看前面这个 d 不太爽, 考虑后面配出一个积性函数使得这个 d 能够被约掉。因此, 我们尝试将 g 配成 id 。
这样就可以把 d 给弄没! 代入得:

$$\begin{aligned} \sum_{d|n} (d \cdot \varphi(d)) \cdot \frac{n}{d} &= \sum_{d|n} n \cdot \varphi(d) \\ &\rightarrow n \sum_{d|n} \varphi(d) = n^2 \end{aligned}$$

我们惊喜的发现, 似乎配对了!!!

得:

$$S(n) = \sum_{i=1}^n i^2 - \sum_{d=2}^n d \cdot S(\lfloor \frac{n}{d} \rfloor)$$

对于这个式子, 我们前面可以利用平方和的公式 $O(1)$ 算出结果, 后面的式子利用等差数列求和公式进行整除分块。

因此, 我们可以通过以上的思路求得这个看似无法筛的积性函数的前缀和!

代码实现

- 至于在信息学中的代码实现, 我给出一个大概的思路: 我们首先先线筛出数据范围根号左右的积性函数的前缀和。再递归的实现杜教筛。
- 特别要注意的是, 杜教筛筛出的前缀和一定要存下来!!!
- 如果你比较的勤劳, 那就去手写hash, 如果你想偷懒, 那就最好用stl中的unordered_map, 最好不要用map, 平白无故多个log的复杂度, 何必呢.....
- 还有一点, 一定要记得取模!!! 以及, 判断要不要开long long, 搞不好你TLE就是因为取模去多了, 或者long long开多啦!
- 有评论区的大佬提醒我, 说这份代码被卡了, 我调了一下前面线筛的范围, 有一定的加速, 最后发现, 果然是开long long的锅, 现在已经将代码改正, 是没有问题的啦!
- 在这里我就粘一下自己杜教筛 μ 和 φ 的板子吧。这种东西最好自己手打一遍, 不然你一没注意, 常数一大, 就很麻烦啦! (反正我写这个东西, 常数巨大)
- 因此, 代码仅供参考! [luoguP4213杜教筛模板](#)

```
#include<bits/stdc++.h>
#include<tr1/unordered_map>
#define N 6000010
using namespace std;
template<typename T> inline void read(T &x)
{
    x=0;
    static int p;p=1;
    static char c;c=getchar();
    while(!isdigit(c)){if(c=='-')p=-1;c=getchar();}
    while(isdigit(c)){x=(x<<1)+(x<<3)+(c-48);c=getchar();}
    x*=p;
}
```



```

bool vis[N];
int mu[N], sum1[N], phi[N];
long long sum2[N];
int cnt, prim[N];
tr1::unordered_map<long long, long long> w1;
tr1::unordered_map<int, int> w;
void get(int maxn)
{
    phi[1]=mu[1]=1;
    for(int i=2;i<=maxn;i++)
    {
        if(!vis[i])
        {
            prim[++cnt]=i;
            mu[i]=-1;phi[i]=i-1;
        }
        for(int j=1;j<=cnt&&prim[j]*i<=maxn;j++)
        {
            vis[i*prim[j]]=1;
            if(i%prim[j]==0)
            {
                phi[i*prim[j]]=phi[i]*prim[j];
                break;
            }
            else mu[i*prim[j]]=-mu[i],phi[i*prim[j]]=phi[i]*(prim[j]-1);
        }
    }
    for(int i=1;i<=maxn;i++)sum1[i]=sum1[i-1]+mu[i],sum2[i]=sum2[i-1]+phi[i];
}
int djsmu(int x)
{
    if(x<=6000000) return sum1[x];
    if(w[x]) return w[x];
    int ans=1;
    for(int l=2,r;l>=0&&l<=x;l=r+1)
    {
        r=x/(x/l);
        ans-=(r-l+1)*djsmu(x/l);
    }
    return w[x]=ans;
}
long long djspphi(long long x)
{
    if(x<=6000000) return sum2[x];
    if(w1[x]) return w1[x];
    long long ans=x*(x+1)/2;
    for(long long l=2,r;l<=x;l=r+1)
    {
        r=x/(x/l);
        ans-=(r-l+1)*djspphi(x/l);
    }
    return w1[x]=ans;
}
int main()
{
    int t,n;
    read(t);
    get(6000000);
    while(t--)
    {
        read(n);
        printf("%lld %d\n", djspphi(n), djsmu(n));
    }
    return 0;
}

```



总结

- 当然，杜教筛还能够筛许多东西，如： $\sum_{i=1}^n i^2 \cdot \mu(i)$ 之类的一系列积性函数，在这里就不一一列举啦。
- 其实，一般来说筛的就是那些常用的积性函数。
- 如果实在碰到类似与上面那个无法一眼看出结果的式子，我们就可以采用刚刚例三的思路。
- 先考虑将那些特殊性质不明显的数弄掉，再尝试猜积性函数。当然不一定一试就中，但是只要我们有足够的耐心与信念，相信这个题目所给的一定能筛，就一定能试出来233。
- 当然还有一种方法，从常见的完全积性函数开始试，如果都不行，在尝试一下高次的完全积性函数，之后尝试非完全积性函数（虽说一般都不是这个。。。），如果还是不行，那就算了吧，（反正就那一点分么。。。），试不出，技不如人，甘拜下风233。

题目

- 题目可以去51nod上找，那上面杜教筛的题目挺多的，我就不粘地址啦！
- 洛谷上也有模板题！
- 当然，洛谷上也有需要推式子的题目，我以后有时间再加吧！

标签： 杜教筛 , 学习笔记

[好文要顶](#)

[关注我](#)

[收藏该文](#)



pengym

关注 - 0

粉丝 - 124

[+加关注](#)

36

0

« 上一篇： [可持久化并查集](#)

» 下一篇： [记OI退役](#)

posted @ 2018-08-11 01:08 pengym 阅读(26311) 评论(30) 编辑 收藏

[刷新评论](#) [刷新页面](#) [返回顶部](#)

[登录后才能查看或发表评论，立即 登录 或者 逛逛 博客园首页](#)

【推荐】阿里云爆品销量榜单出炉，精选爆款产品低至0.55折

【推荐】7大类400多种组件，HarmonyOS鸿蒙三方库来了，赶紧收藏！

【推荐】大型组态、工控、仿真、CAD\GIS 50万行VC++源码免费下载！

【推荐】限时秒杀！国云大数据魔镜，企业级云分析平台

园子动态：

- 致园友们的一封检讨书：都是我们的错
- 数据库实例 CPU 100% 引发全站故障
- 发起一个开源项目：博客引擎 fluss

最新新闻：

- 菜鸟：2021财年全年收入372.5亿元，同比增68%
- 阿里第四财季营收1874亿元，净亏损54.79亿元
- 小鹏汽车：Q1营收29.5亿元 净亏损7.866亿元

- B站发布Q1财报：总营收39亿元，同比增长68%
- HTC发布VIVE FOCUS 3等系列新品 虚拟代言人也来了
- » 更多新闻...



Copyright © 2021 pengym
Powered by .NET 5.0 on Kubernetes



积性函数与Dirichlet卷积 学习小记

原创

Felix-Lee

2018-04-19 17:04:25

1223

收藏 1

版权

分类专栏:

狄利克雷卷积

模板算法

模板与算法

文章标签:

狄利克雷卷积

模板算法

你的浏览器目前处于缩放状态，页面可能会出现错位现象，建议100%大小显示。 ×

前言

- 首先感谢 XHM 大佬的悉心指导，我懂得了不少~。
- 链一下他关于这方面的见解、博客——XHM 的Dirichlet卷积 学习小记

一些定义

- 回归正题，这次我学习了一下狄利克雷卷积方面的知识。
- 先给一波定义：（这里也感谢 skywalkert大佬的精心讲解）

数论函数的定义

- 若 $f(n)$ 的定义域为 **正整数域**，值域为 **复数**，即 $f: \mathbb{Z}^+ \rightarrow \mathbb{C}$ ，则称 $f(n)$ 为 **数论函数**。

积性函数的定义

- 若 $f(n)$ 是 **数论函数**，且 $f(1) = 1$ ，对于 **互质** 的正整数 p, q ，有：

$$f(p \cdot q) = f(p) \cdot f(q)$$

- 则称 $f(n)$ 为 **积性函数**。

完全积性函数的定义

- 若 $f(n)$ 为 **积性函数**，且对于 **任意** 正整数 p, q 都有：

$$f(p \cdot q) = f(p) \cdot f(q)$$

- 则称 $f(n)$ 为 **完全积性函数**。

狄利克雷卷积的定义

- 定义 **数论函数** f 和 g 的 **狄利克雷卷积** 为 h ，则：

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

- 记作： $h = f * g$ 。

一些性质

- Dirichlet 卷积满足交换律、结合律**，对加法满足**分配律**（如： $(f + g) * h = f * h + g * h$ ）。
- 两个 **积性函数** 的 **狄利克雷卷积** 依旧为 **积性函数**。

一些常见的数论函数

- 常值函数： $n(i) = n$ （就是总是返回一个固定值的函数）。

一些常见的积性函数

- 单位元函数: $e(n) = [n = 1]$, 它卷上任意的数论函数仍为原数论函数, 既满足:

$$f * e = f = e * f$$

你的浏览器目前处于缩放状态, 页面可能会出现错位现象, 建议100%大小显示。 ×

- 幂函数: $id^k(n) = n^k$, 完全积性。

- 恒等函数: $I(n) = 1$ (也是常值函数, 只是比较特殊), 完全积性, 相当于 id^0 。

- 单位函数: $id(n) = n$, 完全积性, 相当于 id^1 。

- 莫比乌斯函数 $\mu(n)$, 在狄利克雷卷积的乘法中与 恒等函数 $I(n) = 1$ 互为 逆元, 即有:

$$\mu * I = e$$

- 且 $\mu(1) = 1$, 对于无平方因子数 $n = \prod_{i=1}^t p_i$ 有 $\mu(n) = (-1)^t$,

- 对于有平方因子数 n 有 $\mu(n) = 0$ 。

- 欧拉函数: $\varphi(n) = \sum_{i=1}^n [(n, i) = 1] \cdot 1$, 表示小于等于 n 与 n 互质的数的个数。

- 另外有: $\sum_{i=1}^n [(n, i) = 1] \cdot i = \frac{n \cdot \varphi(n) + [n=1]}{2}$, 且对于正整数 $n > 2$ 来说 $\varphi(n)$ 是偶数。

- 除数函数: $\sigma_k(n) = \sum_{d|n} d^k$, 表示 n 的约数的 k 次幂和, 注意 $\sigma_k(n)$ 与 $\sigma^k(n)$ 是不同的。

- 约数和函数: $\sigma(n) = \sigma_1(n) = \sum_{d|n} d$, 表示 n 的约数之和 (Σ)。

- 约数个数函数: $\tau(n) = \sigma_0(n) = \sum_{d|n} 1$, 表示 n 的约数个数, 一般也写为 $d(n)$ 。

一些常用的狄利克雷卷积

- 说在前面, 因为: $\sum_{d|n} \mu(d) = [n = 1]$, 有①: $I * \mu = e$ (即 I 和 μ 互为逆元)。

- 怎么证明呢? 考虑有值 (非零) 的 $\mu(n)$ 对答案的贡献。

- 当 $n = 1$ 时, 即 $\mu(1) = 1$, 此时 $(I * \mu)(1) = I(1) \cdot \mu(1) = 1 = e(1)$ 。

- 而当 $n > 1$ 时, 令 $n = \prod_{i=1}^k P_i$, 其中 P_i 为互不相同的质数。

- 此时:

$$\begin{aligned} \sum_{i=0}^k \mu(i) \cdot C_k^i &= \sum_{i=0}^k \mu(i) \cdot C_k^i \cdot I(k-i) \\ &= \sum_{i=0}^k (-1)^i \cdot C_k^i \cdot 1^{k-i} \\ &= (1 - 1)^k = 0 \text{ (二项式定理)} \end{aligned}$$

- 所以 $n > 1$ 后面的值之和为零, 则命题 $I * \mu = e$ 得证。

- ②: $\mu * id = \varphi$, 将欧拉函数的通式展开即可得到此式。

- ③: $I * id = \sigma$, 证明如下:

$$\begin{aligned} (I * id)(n) &= \sum_{d|n} id(d) * I\left(\frac{n}{d}\right) \\ &= \sum_{d|n} d \cdot 1 = \sigma(n) \end{aligned}$$

- ④: $I * I = \tau$, 证明跟上面同理。

- 以上①②③④都是非常常用的狄利克雷卷积。

莫比乌斯反演

- 而且我们也能够通过简单的狄利克雷卷积运算轻易地证出莫比乌斯反演。
- 设 $F(n) = \sum_{d|n} f(d)$ ，补上一项： $F(n) = \sum_{d|n} f(d) * I\left(\frac{n}{d}\right)$ (等价)
- 则有： $F = I * f$ ，两边同时卷上 μ ，即： $\mu * F = \mu * I * f$
- 由于①： $\mu * I = e$ ，单位元函数可以直接抹去，那就相当于： $f = \mu * F$
- 于是得证：

你的浏览器目前处于缩放状态，页面可能会出现错位现象，建议100%大小显示。 ×

$$f(d) = \sum_{d|n} \mu(d) \cdot F\left(\frac{n}{d}\right)$$

几个结论

- 结论①： $n = \sum_{d|n} \varphi(d)$ ，证明：

$$id = id * e = id * (\mu * I) = (id * \mu) * I = \varphi * I$$

- 则：

$$n = id(n) = \sum_{d|n} \varphi(d) * I\left(\frac{n}{d}\right) = \sum_{d|n} \varphi(d)$$

- 结论②： $\sigma(n) = \sum_{d|n} \tau(d) * \varphi\left(\frac{n}{d}\right)$ ，证明：

$$\sigma = \sigma * e = (I * id) * (I * \mu) = (I * I) * (id * \mu) = \tau * \varphi$$

二项式反演

- 由 $A_i = \sum_{j=0}^i C_i^j \cdot B_j$ ，反演得： $B_i = \sum_{j=0}^i (-1)^j \cdot C_i^j \cdot A_j$

总结

- 狄利克雷卷积让我们开阔了不少眼界，也让我们做题的思路打开了很多。
- 如结合杜教筛、拉格朗日插值法、莫比乌斯反演等算法可以更轻易地攻克更多题目。

 点赞3  评论  分享  收藏1  打赏  举报  关注 

相关推荐

©2020 CSDN 皮肤主题: 编程工作室 设计师:CSDN官方博客 返回首页

较完整多项式模板

依赖+多项式求导+多项式积分

```
1. #include <bits/stdc++.h>
2. #define ll long long
3. using namespace std;
4. const double PI = 3.14159265358979323846;
5. const int MAXN = 2e5 + 33;//数据范围
6. const int MOD = 998244353, wroot = 3; //NTT依赖数据
7.
8. //定义Complex类
9. class Complex
10. {
11. public:
12.     double r, i;
13.     Complex(double rr = 0, double ii = 0) { r = rr, i = ii; }
14.     Complex operator+(const Complex &op) const { return Complex(r + op.r, i + op.i); }
15.     Complex operator-(const Complex &op) const { return Complex(r - op.r, i - op.i); }
16.     Complex operator*(const Complex &op) const { return Complex(r * op.r - i * op.i, r * op.i + i * op.r); }
17. };
18.
19. //取整
20. int Round(double x)
21. {
22.     return int(x + 0.5);
23. }
24.
25. //快速幂
26. ll qn(ll x, ll v, ll MOD)
```

----- * * * * * -----

```
27. {  
28.     ll ret = 1;  
29.     while (y > 0)  
30.     {  
31.         if (y & 1)  
32.             ret = ret * x % MOD;  
33.         x = x * x % MOD;  
34.         y >>= 1;  
35.     }  
36.     return ret;  
37. }  
38.  
39. int numinv[MAXN << 1];  
40. //返回模MOD意义下x的逆元 MOD为质数  
41. int get_num_inv(int x)  
42. {  
43.     if (x < (MAXN << 1))  
44.     {  
45.         if (numinv[x])  
46.             return numinv[x];  
47.         return numinv[x] = qp(x, MOD - 2, MOD);  
48.     }  
49.     return qp(x, MOD - 2, MOD);  
50. }  
51.  
52. //对n次多项式p求导得模MOD意义下的多项式res  
53. void Poly_derivative(const int *p, int *res, const int n)  
54. {
```

```

55.     for (int i = 0; i <= n - 1; i++)
56.         res[i] = (ll)p[i + 1] * (i + 1) % MOD;
57.     res[n] = 0;
58. }
59. //对n次多项式p积分得模MOD意义下的多项式res 且res[0]=0;
60. void Poly_integral(const int *p, int *res, const int n)
61. {
62.     for (int i = n; i >= 1; i--)
63.         res[i] = (ll)p[i - 1] * (ll)get_num_inv(i) % MOD;
64.     res[0] = 0;
65. }

```

FFT+NTT

FFT

```

1. namespace FFT_template
2. {
3.     Complex temp1[MAXN << 2], temp2[MAXN << 2];
4.     void brc(Complex *p, const int N)
5.     {
6.         int i, j, k;
7.         for (i = 1, j = N >> 1; i < N - 2; i++)
8.         {
9.             if (i < j)
10.                 swap(p[i], p[j]);
11.             for (k = N >> 1; j >= k; k >>= 1)

```

```
12.         j -= k;
13.         if (j < k)
14.             j += k;
15.     }
16. }
17. void FFT(Complex *p, const int N, const int op) //op==1 为正变换， op== -1 为逆
18. {
19.     brc(p, N);
20.     double p0 = PI * op;
21.     for (int h = 2; h <= N; h <<= 1, p0 *= 0.5)
22.     {
23.         int hf = h >> 1;
24.         Complex unit(cos(p0), sin(p0));
25.         for (int i = 0; i < N; i += h)
26.         {
27.             Complex w(1.0, 0.0);
28.             for (int j = i; j < i + hf; j++)
29.             {
30.                 Complex u = p[j], v = w * p[j + hf];
31.                 p[j] = u + v;
32.                 p[j + hf] = u - v;
33.                 w = w * unit;
34.             }
35.         }
36.     }
37.     if (op == -1)
38.         for (int i = 0; i < N; i++)
39.             p[i].r /= N;
```

```
40. }
41. } // namespace FFT_template
```

NTT

```
1. namespace NTT_templates
2. {
3.     int wi[MAXN << 2];
4.     void brc(int *p, const int N)
5.     {
6.         int i, j, k;
7.         for (i = 1, j = N >> 1; i < N - 2; i++)
8.         {
9.             if (i < j)
10.                 swap(p[i], p[j]);
11.             for (k = N >> 1; j >= k; k >>= 1)
12.                 j -= k;
13.             if (j < k)
14.                 j += k;
15.         }
16.     }
17.     void NTT_init(const int N) //使用NTT之前调用,且N要保持一致,为2的幂
18.     {
19.         wi[0] = 1;
20.         wi[1] = qp(wroot, (MOD - 1) / N, MOD);
21.         for (int i = 2; i <= N; i++)
22.             wi[i] = (ll)wi[i - 1] * (ll)wi[1] % MOD;
```

```

23. }

24. void NTT(int *p, const int N, const int op)

25. {

26.     brc(p, N);

27.     for (int h = 2; h <= N; h <<= 1)

28.     {

29.         int unit = ((op == -1) ? (N - N / h) : (N / h));

30.         int hf = h >> 1;

31.         for (int i = 0; i < N; i += h)

32.         {

33.             int w = 0;

34.             for (int j = i; j < i + hf; j++)

35.             {

36.                 int u = p[j], v = (ll)wi[w] * (ll)p[j + hf] % MOD;

37.                 if ((p[j] = u + v) >= MOD)

38.                     p[j] -= MOD;

39.                 if ((p[j + hf] = u - v) < 0)

40.                     p[j + hf] += MOD;

41.                 if ((w += unit) >= N)

42.                     w -= N;

43.             }

44.         }

45.     }

46.     if (op == -1)

47.     {

48.         int inv = qp(N, MOD - 2, MOD);

49.         for (int i = 0; i < N; i++)

```

```
50.         p[i] = (ll)p[i] * (ll)inv % MOD;
51.     }
52. }
53. } // namespace NTT_templates
```

多项式乘法

```
1. namespace Polynomial_mul
2. {
3.     using namespace NTT_templates;
4.     using namespace FFT_template;
5.     int temp11[MAXN << 2], temp22[MAXN << 2];
6.     //对于给定n次多项式a和b, 多项式res=a*b (可能有精度损失)
7.     void Poly_mul_FFT(const int *a, const int *b, int *res, const int n) //n为最高次
8.     {
9.         int N = 2;
10.        while (N <= n + n)
11.            N <<= 1;
12.        Complex *A = temp1,
13.                *B = temp2;
14.        for (int i = 0; i < N; i++)
15.            A[i] = (i <= n ? Complex(a[i], 0.0) : Complex(0.0, 0.0));
16.        for (int i = 0; i < N; i++)
17.            B[i] = (i <= n ? Complex(b[i], 0.0) : Complex(0.0, 0.0));
18.        FFT(A, N, 1);
19.        FFT(B, N, 1);
20.        for (int i = 0; i < N; i++)
```

```
21.         A[i] = A[i] * B[i];  
22.         FFT(A, N, -1);  
23.         for (int i = 0; i < N; i++)  
24.             res[i] = Round(A[i].r);  
25.     }  
26. //对于给定n次多项式a和b,求出模MOD以及x^(2*n+1)下的多项式res=a*b  
27. void Poly_mul(const int *a, const int *b, int *res, const int n)  
28. {  
29.     int N = 2;  
30.     while (N <= n + n)  
31.         N <<= 1;  
32.     NTT_init(N);  
33.     int *A = temp11,  
34.         *B = temp22;  
35.     for (int i = 0; i < N; i++)  
36.         A[i] = (i <= n ? a[i] : 0);  
37.     for (int i = 0; i < N; i++)  
38.         B[i] = (i <= n ? b[i] : 0);  
39.     NTT(A, N, 1);  
40.     NTT(B, N, 1);  
41.     for (int i = 0; i < N; i++)  
42.         A[i] = (ll)A[i] * (ll)B[i] % MOD;  
43.     NTT(A, N, -1);  
44.     for (int i = 0; i <= n + n; i++)  
45.         res[i] = A[i];  
46. }  
47. } // namespace Polynomial_mul
```

多项式求逆

```
1. namespace Polynomial_inv
2. {
3.     using namespace NTT_templates;
4.     int tmp[MAXN << 2], tmp2[MAXN << 2], tmp3[MAXN << 2];
5.     void get_poly_inv(const int *p, int *res, const int N) //N是2的幂,一般不调用
6.     {
7.         if (N <= 1)
8.         {
9.             res[0] = qp(p[0], MOD - 2, MOD);
10.            return;
11.        }
12.        get_poly_inv(p, res, N >> 1);
13.        int K = N << 1;
14.        int *temp = tmp;
15.        for (int i = 0; i < N; i++)
16.        {
17.            temp[i] = p[i];
18.            for (int i = N; i < K; i++)
19.            {
20.                temp[i] = res[i] = 0;
21.                NTT_init(K);
22.                NTT(temp, K, 1);
23.                NTT(res, K, 1);
24.                for (int i = 0; i < K; i++)
25.                {
26.                    res[i] = (ll)res[i] * (2 - (ll)temp[i] * (ll)res[i] % MOD) % MOD;
```

```

25.         if (res[i] < 0)
26.             res[i] += MOD;
27.     }
28.     NTT(res, K, -1);
29.     for (int i = N; i < K; i++)
30.         res[i] = 0;
31. }
32. //对给定n次多项式p, 求出其模MOD以及x^(n+1)意义下的逆多项式res
33. void Poly_inv(const int *p, int *res, const int n) //n是最高次项次数 模x^(n+1)
34. {
35.     int N = 2;
36.     while (N <= n)
37.         N <<= 1;
38.     int dN = N << 1;
39.     int *temp_in = tmp3,
40.         *temp_out = tmp2;
41.     for (int i = 0; i < N; i++)
42.         temp_in[i] = (i <= n ? p[i] : 0);
43.     for (int i = 0; i < dN; i++)
44.         temp_out[i] = 0;
45.     get_poly_inv(temp_in, temp_out, N);
46.     for (int i = 0; i <= n; i++)
47.         res[i] = temp_out[i];
48. }
49. } // namespace Polynomial_inv

```

多项式求对数(ln)

```
1. namespace Polynomial_ln  
  
2. {  
  
3.     using namespace NTT_templates;  
  
4.     int tmp[MAXN << 2], tmp2[MAXN << 2], tmp3[MAXN << 2];  
  
5.     void get_poly_ln(const int *p, int *res, const int N) //N为2的幂 为项数,而非最高项次数  
6.     {  
  
7.         int *temp = tmp;  
  
8.         Polynomial_inv::get_poly_inv(p, temp, N);  
  
9.         int K = N << 1;  
  
10.        Poly_derivative(p, res, N - 1);  
  
11.        for (int i = N; i < K; i++)  
  
12.            res[i] = 0;  
  
13.        NTT_init(K);  
  
14.        NTT(res, K, 1);  
  
15.        NTT(temp, K, 1);  
  
16.        for (int i = 0; i < K; i++)  
  
17.            res[i] = (ll)res[i] * temp[i] % MOD;  
  
18.        NTT(res, K, -1);  
  
19.        Poly_integral(res, res, N - 1);  
  
20.    }  
  
21. //对给定n次多项式p (且p[0]==1) ,求出其模MOD以及x^(n+1)意义下的多项式res==ln  
  
22. void Poly_ln(const int *p, int *res, const int n) //n为最高项次数  
  
23. {  
  
24.     int N = 2;  
  
25.     while (N <= n)  
  
26.         N <<= 1;  
  
27.     int *temp_in = tmp3,
```

```

28.         *temp_out = tmp2;

29.     for (int i = 0; i < N; i++)
30.         temp_in[i] = (i <= n ? p[i] : 0);
31.     get_poly_ln(temp_in, temp_out, N);
32.     for (int i = 0; i <= n; i++)
33.         res[i] = temp_out[i];
34. }
35. } // namespace Polynomial_ln

```

多项式求指数(exp)

```

1. namespace Polynomial_exp
2. {
3.     using namespace NTT_templates;
4.     int tmp[MAXN << 2], tmp2[MAXN << 2], tmp3[MAXN << 2];
5.     void get_poly_exp(const int *p, int *res, const int N) //N为2的幂 为项数,而非
6.     {
7.         if (N <= 1)
8.             {
9.                 res[0] = 1;
10.            return;
11.        }
12.        get_poly_exp(p, res, N >> 1);
13.        int *temp = tmp;
14.        Polynomial_ln::get_poly_ln(res, temp, N);
15.        int K = N << 1;

```

```

16.     for (int i = 0; i < N; i++)
17.     {
18.         temp[i] = p[i] - temp[i];
19.         if (temp[i] < 0)
20.             temp[i] += MOD;
21.     }
22.     if ((++temp[0]) == MOD)
23.         temp[0] = 0;
24.     for (int i = N; i < K; i++)
25.         temp[i] = res[i] = 0;
26.     NTT_init(K);
27.     NTT(temp, K, 1);
28.     NTT(res, K, 1);
29.     for (int i = 0; i < K; i++)
30.         res[i] = (ll)res[i] * (ll)temp[i] % MOD;
31.     NTT(res, K, -1);
32.     for (int i = N; i < K; i++)
33.         res[i] = 0;
34. }
35. //对给定n次多项式p (且p[0]==0) ,求出其模MOD以及x^(n+1)意义下的多项式res==exp
36. void Poly_exp(const int *p, int *res, const int n)
37. {
38.     int N = 2;
39.     while (N <= n)
40.         N <<= 1;
41.     int *temp_out = tmp2,
42.         *temp_in = tmp3;

```

```

43.     for (int i = 0; i < N; i++)
44.         temp_in[i] = (i <= n ? p[i] : 0);
45.     get_poly_exp(temp_in, temp_out, N);
46.     for (int i = 0; i <= n; i++)
47.         res[i] = temp_out[i];
48. }
49. } // namespace Polynomial_exp

```

多项式除法 & 取模

```

1. namespace Polynomial_div
2. {
3.     int tmp[MAXN << 2], tmp2[MAXN << 2];
4.     //翻转n次多项式p的系数, 得到多项式res
5.     void Polynomial_flip(const int *p, int *res, const int n)
6.     {
7.         for (int i = 0; i <= n; i++)
8.         {
9.             res[i] = p[n - i];
10.        }
11.    }
12.    //对给定n次多项式p, m次多项式q , p=T*q+R, 求出多项式T和R 要求n>=m
13.    void Poly_div(const int *p, const int n, const int *q, const int m, int *re:
14.    {
15.        Polynomial_flip(q, tmp, m);
16.        Polynomial_inv::Poly_inv(tmp, tmp2, n - m);
17.        Polynomial_flip(p, tmp, n);

```

```
18.     Polynomial_mul::Poly_mul(tmp2, tmp, tmp2, n - m);
19.     Polynomial_flip(tmp2, tmp, n - m);
20.     for (int i = n - m + 1; i <= n; i++)
21.         tmp[i] = 0;
22.     Polynomial_mul::Poly_mul(q, tmp, tmp2, max(n - m, m));
23.     for (int i = 0; i <= n; i++)
24.     {
25.         resR[i] = (p[i] - tmp2[i]) % MOD + MOD;
26.         if (resR[i] >= MOD)
27.             resR[i] -= MOD;
28.     }
29.     for (int i = 0; i <= n - m; i++)
30.         resT[i] = tmp[i];
31.     for (int i = n - m + 1; i <= n; i++)
32.         resT[i] = 0;
33. }
34. } // namespace Polynomial_div
```

2020-01-01



Eserinc

Love life, love blank.





这是 Google 对 <https://blog.csdn.net/wbin233/article/details/72998375> 的缓存。这是该网页在 2021 年 5 月 13 日 07:28:04 GMT 的快照。当前页在此期间可能已经更改。[了解详情](#).

完整版本 纯文字版本 查看源代码

提示：要在此页面上快速找到您的搜索字词，请按 **Ctrl+F** 或者 **⌘+F** (Mac)，然后使用查找栏搜索。

康托展开和逆康托展开

原创 wbin233 2017-06-10 17:56:16 13402 收藏 39

版权

分类专栏： 算法 acm 文章标签： 算法 康托展开 逆康托展开 acm

文章目录

简述

原理

康托展开

逆康托展开

示例：

应用

简述

康托展开是一个全排列到一个自然数的双射，常用于构建hash表时的空间压缩。设有n个数（1, 2, 3, 4,...,n），可以有组成不同($n!$ 种)的排列组合，康托展开表示的就是在n个不同元素的全排列中，比当前排列组合小的个数，那么也可以表示当前排列组合在n个不同元素的全排列中的名次（当前的名次 = 比当前排列组合小的个数 + 1）。

原理

$X=a[n] * (n-1)! + a[n-1] * (n-2)! + \dots + a[i] * (i-1)! + \dots + a[1] * 0!$

其中， $a[i]$ 为整数，并且 $0 \leq a[i] \leq i$, $0 \leq i < n$, 表示当前未出现的元素中排第几个，这就是康托展开。

例如有3个数（1, 2, 3），则其排列组合及其相应的康托展开值如下：

排列组合	名次	康托展开	值
123	1	$0 * 2! + 0 * 1! + 0 * 0!$	0
132	2	$0 * 2! + 1 * 1! + 0 * 0!$	1
213	3	$1 * 2! + 0 * 1! + 0 * 0!$	2
231	4	$1 * 2! + 1 * 1! + 0 * 0!$	3
312	5	$2 * 2! + 0 * 1! + 0 * 0!$	4
321	6	$2 * 2! + 1 * 1! + 0 * 0!$	5

比如其中的 231：

- 想要计算排在它前面的排列组合数目（123, 132, 213），则可以转化为计算比首位小即小于2的所有排列「1 * 2！」，首位相等为2并且第二位小于3的所有排列「1 * 1！」，前两位相等为23并且第三位小于1的所有排列（0 * 0!）的和即可，康托展开为：1 * 2! + 1 * 1! + 0 * 0! = 3。
- 所以小于231的组合有3个，所以231的名次是4。

康托展开

再举个例子说明。

在（1, 2, 3, 4, 5）5个数的排列组合中，计算 34152 的康托展开值。

- 首位是3，则小于3的数有两个，为1和2， $a[5]=2$ ，则首位小于3的所有排列组合为 $a[5] * (5-1)!$
- 第二位是4，则小于4的数有两个，为1和2，注意这里3并不能算，因为3已经在第一位，所以其实计算的是在第二位之后小于4的个数。因此 $a[4]=2$
- 第三位是1，则在其之后小于1的数有0个，所以 $a[3]=0$
- 第四位是5，则在其之后小于5的数有1个，为2，所以 $a[2]=1$
- 最后一位就不用计算啦，因为在它之后已经没有数了，所以 $a[1]$ 固定为0



- 根据公式：

$$X = 2 * 4! + 2 * 3! + 0 * 2! + 1 * 1! + 0 * 0! = 2 * 24 + 2 * 6 + 1 = 61$$

所以比 34152 小的组合有61个，即34152是排第62。

具体代码实现如下：（假设排列数小于10个）

```
static const int FAC[] = {1, 1, 2, 6, 24, 120, 720, 5040, 40320, 362880}; // 阶乘
int cantor(int *a, int n)
{
    int x = 0;
    for (int i = 0; i < n; ++i) {
        int smaller = 0; // 在当前位之后小于其的个数
        for (int j = i + 1; j < n; ++j) {
            if (a[j] < a[i])
                smaller++;
        }
        x += FAC[n - i - 1] * smaller; // 康托展开累加
    }
    return x; // 康托展开值
}
```

tips: 这里主要为了讲解康托展开的思路，实现的算法复杂度为O(n^2)，实际当n很大时，内层循环计算在当前位之后小于当前位的个数可以用 线段树 来处理计算，而不用每次都遍历，这样复杂度可以降为O(nlogn)。

逆康托展开

一开始已经提过了，康托展开是一个全排列到一个自然数的 双射，因此是可逆的。即对于上述例子，在(1, 2, 3, 4, 5) 给出61可以算出起排列组合为 34152。由上述的计算过程可以容易的逆推回来，具体过程如下：

- 用 $61 / 4! = 2$ 余13，说明 $a[5]=2$, 说明比首位小的数有2个，所以首位为3。
- 用 $13 / 3! = 2$ 余1，说明 $a[4]=2$ ，说明在第二位之后小于第二位的数有2个，所以第二位为4。
- 用 $1 / 2! = 0$ 余1，说明 $a[3]=0$ ，说明在第三位之后没有小于第三位的数，所以第三位为1。
- 用 $1 / 1! = 1$ 余0，说明 $a[2]=1$ ，说明在第二位之后小于第四位的数有1个，所以第四位为5。
- 最后一位自然就是剩下的数2啦。
- 通过以上分析，所求排列组合为 34152。

具体代码实现如下：（假设排列数小于10个）

```
static const int FAC[] = {1, 1, 2, 6, 24, 120, 720, 5040, 40320, 362880}; // 阶乘

//康托展开逆运算
void decantor(int x, int n)
{
    vector<int> rest; // 存放当前可选数，保证有序
    for(int i=1;i<=n;i++)
        v.push_back(i);

    vector<int> ans; // 所求排列组合
    for(int i=n;i>=1;i--)
    {
        int r = x % FAC[i-1];
        int t = x / FAC[i-1];
        x = r;
        a.push_back(v[t]); // 剩余数里第t+1个数为当前位
        v.erase(v.begin() + t); // 移除选做当前位的数
    }
}
```

示例：



举报

60. 第k个排列

难度 中等 收藏 分享 切换为英文 关注 反馈

给出集合 $[1, 2, 3, \dots, n]$ ，其所有元素共有 $n!$ 种排列。按大小顺序列出所有排列情况，并一一标记，当 $n = 3$ 时，所有排列如下：

1. "123"
2. "132"
3. "213"
4. "231"
5. "312"
6. "321"

给定 n 和 k ，返回第 k 个排列。

说明：

- 给定 n 的范围是 $[1, 9]$ 。
- 给定 k 的范围是 $[1, n!]$ 。

示例 1：

输入: $n = 3, k = 3$
 输出: "213"

示例 2：

输入: $n = 4, k = 9$
 输出: "2314"

<https://blog.csdn.net/wbln233>直接套逆康托展开即可，需要处理的是求的是第 k 个排列，那么其对应的康托展开值应该减 1 ($k - 1$)。

```
class Solution {
public:
    // 逆康托展开
    string getPermutation(int n, int k) {
        static constexpr int FAC[] = {1, 1, 2, 6, 24, 120, 720, 5040, 40320, 362880}; // 阶乘

        vector<int> rest; // 存放当前可选数，有序
        for(int i = 1; i <= n; i++)
            rest.push_back(i);

        k--; // 要先 -1 才是其康托展开的值
        string ans = "";
        ans.reserve(n);

        for(int i = n; i >= 1; i--)
        {
            int r = k % FAC[i-1];
            int t = k / FAC[i-1];
            k = r;
            ans += to_string(rest[t]); // 剩余数里第t+1个数为当前位
            rest.erase(rest.begin() + t); // 移除选做当前位的数
        }

        return ans;
    }
};
```

应用

应用最多的场景也是上述讲的它的特性。

- 给定一个自然数集合组合一个全排列，所其中的一个排列组合在全排列中从小到大排第几位。
在上述例子中，在 $(1, 2, 3, 4, 5)$ 的全排列中，34152的排列组合排在第62位。
- 反过来，就是逆康托展开，求在一个全排列中，从小到大的第 n 个全排列是多少。
比如求在 $(1, 2, 3, 4, 5)$ 的全排列中，第62个排列组合是34152。[注意具体计算中，要先 -1 才是其康托展开的值。]
- 另外康托展开也是一个数组到一个数的映射，因此也是可用于hash，用于空间压缩。比如在保存一个序列，我们可能需要开一个数组，如果能够把它映射成一个自然数，则只需要保存一个整数，大大压缩空间。比如八数码问题。

点赞30

评论13

分享

收藏39

打赏

举报

关注

一键三连

都能看懂的康托展开

lrbless的博客 6726

康托展开简介：官方简介：康托展开是一个全排列到一个自然数的双射，常用于构建哈希表时的空间压缩。康托展...

康托展开

zhongkei的专栏 2万+





优质评论可以帮助作者获得更高权重



评论

相关推荐

康托展开和逆康托展开

Geek 1724

简述 康托展开是一个全排列到一个自然数的双射, 常用于构建hash表时的空间压缩。设有n个数 (1, 2, 3, 4,...,...)

算法基础：康托展开与逆康托展开

ajaxlt的博客 7388

康托展开与逆康托展开 序言: 本文记录康托展开与逆康托展开的原理以及其应用。1.概述 举例而言, 对于 1 ~ 4 的一...

康托展开与实例

qq_49662108的博客 828

康托展开与实例 康托展开是一个全排列到一个自然数的双射, 类似于哈希, 每一种排列都有一个唯一的对应字典序...

康托展开详解 -csdn博客

i-Curve的博客 217

康托展开详解 -csdn博客 定义: 康托展开是一个全排列到一个自然数的双射, 常用于构建哈希表时的空间压缩。康...

康托展开

obsorb_knowledge的博客 244

康托展开是什么? 定义: $X = a_n * (n-1)! + a_{n-1} * (n-2)! + \dots + a_1 * (1)!$ a_i 为整数, 并且 0 简单点说就是, 判...

康托展开与逆康托展开详解

zyz_bz的博客 387

文章目录康拓展开运用板子逆康拓展开板子 康拓展开 康托展开是一个全排列到一个自然数的双射, 常用于构建哈希...

康托展开的代码

10-08

实现康托展开hash的代码, pascal语言通过将数组转变为数字

【算法】康托展开和逆康托展开

立志如山, 行道如水 337

文章目录康拓展开逆康拓展开 康拓展开 康托展开是一个全排列到一个自然数的双射, 常用于构建hash表时的空间压...

康拓展开 (维基百科)

MrBlank的ACM记事本 514

via: <https://zh.wikipedia.org/wiki/康托展开> 康托展开是一个全排列到一个自然数的双射, 常用于构建哈希表时的空间...

康托展开及其逆运算 详解

继续激情, 继续奋斗。 1万+

康托展开 康托展开逆运算 详解

全排列计算 (康托展开)

52learn 1万+

题目描述 给出一个1~n的全排列中的某一个, 求它是按字典序排列的第几个。输入输出格式 输入格式: 第一行, 一...

详细讲解康托展开集齐基础例题+模板

塞满箱子的过程 264

单纯讲解怎么用怎么算, 说些人话。预备概念: 排列: 对于一个数字n的排列就是含有[1,n]所有数字的序列。也就...

康托展开及其应用

liulingbo918 1015

康托展开是一种特殊的哈西函数, 用阶乘的线性组合来表示一个数x, 即 $x = a[n] * n! + a[n-1] * (n-1)! + \dots + a[1] * 1!$, ...

CSDN开发者助手, 常用网站自动整合, 多种工具一键调用

CSDN开发者助手由CSDN官方开发, 集成一键呼出搜索、万能快捷工具、个性化新标签页和官方免广告四大功能。...

简介介绍康托展开

蚩尤煜的博客 2187

本篇文章并不会详细讲解康托展开的数学原理, 有兴趣的朋友可以另行寻找。康托展开 作用: 判断这个数在其各个...

关于康托展开的用途及写法

sluqy671的专栏 989

在处理八数码这一类需要用到全排列的问题的时候, 存储往往是一个难题, 因为明明只有 $n!$ 种情况, 数字的长度却...

c语言之康托展开

潘猪猪的世界 1044

/* $X = a[n] * (n-1)! + a[n-1] * (n-2)! + \dots + a[i] * (i-1)! + \dots + a[1] * 0!$ 其中, $a[i]$ 为整数, 并且 $X = a[n] * (n-1)! + a[n-1] * (n-2)! + \dots + a[i] * (i-1)!$...

康托展开及逆康托展开

落花时节又逢君 88

概念: 康托展开是一个全排列到一个自然数的双射, 常用于构建哈希表时的空间压缩。康托展开的实质是计算当前...

©2020 CSDN 皮肤主题: 大白 设计师:CSDN官方博客 返回首页

关于我们 招贤纳士 广告服务 开发助手 400-660-0108 kefu@csdn.net 在线客服 工作时间 8:30-22:00

公安备案号11010502030143 京ICP备19004658号 京网文〔2020〕1039-165号 经营性网站备案信息 北京互联网违法和不良信息举报中心
网络110报警服务 中国互联网举报中心 家长监护 Chrome商店下载 ©1999-2021北京创新乐知网络技术有限公司 版权与免责声明 版权申诉
出版物许可证 营业执照



wbin233
码龄5年  暂无认证

28	13万+	9万+	8万+	
原创	周排名	总排名	访问	等级

1085	34	82	48	101
积分	粉丝	获赞	评论	收藏

[!\[\]\(367221d8e2cd675350eeed60d5a8f593_img.jpg\)](#)
[!\[\]\(7b7524220c4548cce5f86613f2583c83_img.jpg\)](#)

[私信](#)
[关注](#)

举报

热门文章

- LIS算法: 最长上升子序列 ⚡ 15910
- 使用Flask-Mail和qq邮箱SMTP服务发送邮件 ⚡ 14473
- 康托展开和逆康托展开 ⚡ 13394
- Android 从相册中选择照片并返回 ⚡ 6853
- python 列表去重 (不可变类型和可变类型) ⚡ 4600

分类专栏

- android 3篇
- acm 6篇
- java 6篇
- python 4篇
- 杂七杂八
- 算法 4篇

**最新评论**

- java8 ArrayList源码阅读【2】 - 总结
Tisfy: 此贴甚妙!
- 分布式一致性协议: Basic Paxos原理及...
清清浅浅g: 写得真不错, 可以配合《从 Paxos 到 ZooKeeper》食用, 讲述的逻辑有 ...
- 康托展开和逆康托展开
独恒而后: 好的, 我知道了, 谢谢博主, 但是你的代码确实有点问题 (我觉得 (只 ...
- 康托展开和逆康托展开
wbin233: 0.0 只能说使用场景有局限 (对n而言) 啦。求第几个排列你用next_perm ...
- 康托展开和逆康托展开
独恒而后: 还有你发出的代码没有一份是对的.....

最新文章

- 分布式一致性协议: Basic Paxos原理及推导
- glusterfs IO缓存池
- gluster安装完全指南
- 2018年 3篇 2017年 16篇
- 2016年 6篇 2015年 3篇

目录

举报