2025/10/17 01:57 1/14 Stirling数

Stirling数

总论

Stirling数是组合数学中两类很重要的数,分为第一类Stirling数和第二类Stirling数。

我先声明一下苏联式记法和美国式记法。对于组合数,苏联式记法是C_m^n□美国式记法是小括号里面上m下n□上下关系恰好与苏联式记法相反。虽然英美与西欧等大多通用美国式记法,但是俄罗斯以及东欧国家流行苏联式记法,并且我国的高中课本与高考也通用组合数的苏联式记法。

那么同样的,虽然Stirling数在西方有着类似的中括号与大括号的记法,我还是综合了国内的一些参考书, 给了以下定义:

第一类Stirling数,记作小写字母s□加下角标1。即:

\$\$s 1(n,k)\$\$

第二类Stirling数,记作大写字母S□加下角标2。即:

\$\$\$ 2(n,k)\$\$

这样无论是常见的大小写,还是下角标形式记录的Stirling数,全都没有歧义。

一些书上认为第一类Stirling数是全正的,但是绝大多数书上认为第一类Stirling数有正有负。为了方便使用,我们这里的第一类Stirling数是有正有负的。如果对于极少数书上全正的定义,在这里视作第一类Stirling数的绝对值,即 \mid s 1 \mid

第一类Stirling数

第一类Stirling数s 1(n,k)和置换相关。

置换:1到n的一个排列。例如:645231是长度为6的一个置换。

显然全体长为n的轮换有n!个。全体长为n的置换构成n元对称群。

置换的图:如果第i个位置上的数是i□就从顶点i到j画有向边。

最终置换的图由若干个圈组成,不动点是由顶点指向自身的边构成长为1的圈。

长为k的轮换:对于单位置换1 2 n□将其中的k位接连打乱顺序,对应图论中一个长为k的圈。

置换可以看成不相交轮换的乘积。不动点也看作一个轮换。

一个置换恰含k个轮换:恰好可以写成k个不相交轮换的乘积,即图中恰好有k个连通分支。

给出绝对值的定义[]s 1(n,k)]是全体置换中,恰含k个轮换的置换个数。

符号的定义如下:

 $s_1(n,k) = {(-1)}^{n-k}|s_1(n,k)|$

即当n与k的奇偶性相同为正,不同为负。

显然[n!是带绝对值的第n行第一类Stirling数的和。有一个定理是说,如果不带绝对值,有正有负,那么除了第一行第一类Stirling数的和是1以外,其余的行第一类Stirling数的和全是0。

第二类Stirling数

第二类Stirling数与集合的划分有关。

给出定义□S 2(n,k)是把集合1.....n分成k个非空子集的划分个数。

Bell数

Bell数与第二类Stirling数相关。第n个Bell数记作B_n[]

Bell数和Bernoli数没什么关系,极少同时出现。我的习惯是将Bell数记作大写,将Bernoli数记作小写,或者提前声明是哪一种数。

本文中的B n全部都是Bell数,与Bernoli数无关。

给出定义□B_n是把集合1.....n分成非空子集的划分个数。

因此B n是第n行第二类Stirling数的和。

自0开始的Bell数:

n	0	1	2	3	4	5	6	7	8
p n	1	1	2	5	15	52	203	877	4140

数表

研究组合数学,必须要有数表。

第一类Stirling数的数表如下:

下n右k	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8
-4	1	0	0	0	0	0	0	0	0	0	0	0	0
-3	-6	1	0	0	0	0	0	0	0	0	0	0	0
-2	7	-3	1	0	0	0	0	0	0	0	0	0	0
-1	-1	1	-1	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0
1	0	0	0	0	0	1	0	0	0	0	0	0	0
2	0	0	0	0	0	-1	1	0	0	0	0	0	0
3	0	0	0	0	0	2	-3	1	0	0	0	0	0
4	0	0	0	0	0	-6	11	-6	1	0	0	0	0

2025/10/17 01:57 3/14 Stirling数

5	0	0	0	0	0	24	-50	35	-10	1	0	0	0
6	0	0	0	0	0	-120	274	-225	85	-15	1	0	0
7	0	0	0	0	0	720	-1764	1624	-735	175	-21	1	0
8	0	0	0	0	0	-5040	13068	-13132	6769	-1960	322	-28	1

第二类Stirling数的数表如下:

下n右k	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8
-4	1	0	0	0	0	0	0	0	0	0	0	0	0
-3	6	1	0	0	0	0	0	0	0	0	0	0	0
-2	11	3	1	0	0	0	0	0	0	0	0	0	0
-1	6	2	1	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0
1	0	0	0	0	0	1	0	0	0	0	0	0	0
2	0	0	0	0	0	1	1	0	0	0	0	0	0
3	0	0	0	0	0	1	3	1	0	0	0	0	0
4	0	0	0	0	0	1	7	6	1	0	0	0	0
5	0	0	0	0	0	1	15	25	10	1	0	0	0
6	0	0	0	0	0	1	31	90	65	15	1	0	0
7	0	0	0	0	0	1	63	301	350	140	21	1	0
8	0	0	0	0	0	1	127	966	1701	1050	266	28	1

一些数感较好的人往往可以从数表中找到许多灵感。我们可以看到,第一类Stirling数的绝对值,增长速度比第二类Stirling数要快。

定义式

普通定义

首先限定下面几个式子的n的全体[n_1到n_k为正整数,并且:

\$\$n 1+n 2+.....+n k=n\$\$

以下的和式中的求和号,指对满足条件的n_1到n_k全体求和。那么,根据上文的定义,直接写出:

 $s_1(n,k)={(-1)}^n\frac{n!}{k!}\sum_{n_1,2,...,n_k}$

 $\sl _2(n,k)=\frac{n!}{k!}\sum_{n_1!n_2!....n_k!}$

我们根据定义式就能清楚看出,第一类Stirling数的绝对值增长速度,比第二类Stirling数增长速度快。因为相同求和条件下相似的式子,第二类分母中是阶乘,而第一类是简单的数本身。

对于第二类Stirling数,自然可以配成多项式定理中的系数,可以联系多项式定理,最后可以解出这个较为简单的式子:

 $$$S_2(n,k)=\frac{1}{k!}\sum_{j=0}^k C_k^j j^n{(-1)}^{k-j}$ \$

对于第一类Stirling数,没有解决办法。例如特例第二列,它恰好是调和级数部分和的分子(约分前)。记Hn是第n个调和级数部分和:

 $$H_n=\sum_{i=1}^n \frac{1}{i}$

就有:

 $$$s_1(n,2)={(-1)}^n (n-1)!H_{n-1}$$

负下标

如果利用下文的递推式,强行推广至负下标,会得到这样的结论:

 $s_1(n,k) = {(-1)}^{n+k}S_2(-k,-n)$

递推式

和组合数一样,在数表中Stirling数也由左上角的数和正上方的数求和得到。

但是[Stirling数都是"右倾"的,即从正上方继承的数前方都带有一个系数,而左上方没有。具体的写就是:

第一类Stirling数继承了上方数的行数,带绝对值的时候是简单的求和,不带时要加负号:

 $$$s_1(n,k)=s_1(n-1,k-1)-(n-1)s_1(n-1,k)$$$

第二类Stirling数继承了列数□

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

这两个递推式是最重要的公式。

虽然每一列的通项难以解出,但是斜线方向的通项总是多项式量级的,利用数学归纳法就可以解出来。最边上的斜线全是1,内侧的斜线是简单的自然数和(第二类为正,第一类为负)。用组合恒等式可以递推地算出再往内侧的斜线:

 $$$s 1(n,n-2)=3C n^4+2C n^3$$$

\$\$\$ 2(n,n-2)=3C n^4+C n^3\$\$

\$\$\$ 1(n,n-3)=-15C n^6-20C n^5-6C n^4\$\$

\$\$\$_2(n,n-3)=15C_n^6+10C_n^5+C_n^4\$\$

\$\$s_1(n,n-4)=105C_n^8+210C_n^7+130C_n^6+24C_n^5\$\$

\$\$\$ 2(n,n-4)=105C n^8+105C n^7+25C n^6+C n^5\$\$

可见,第t条斜线一定构成t-2次多项式,并且两类Stirling数的首项系数相同或相反(都是奇数的双阶乘)。

2025/10/17 01:57 5/14 Stirling数

生成函数

指数型生成函数

因为上面列出的数表都是下三角形矩阵,所以对于每一行,长度都是有限的,而对于每一列,长度都是无限的。如果配上两个变元x与y□就能写出二元的生成函数。

先看Bell数,Bell数是同行(固定n□对全体k求和)第二类Stirling数的和。对于Bell数,有指数型生成函数。 指数型生成函数就是每项下面要配一个阶乘,就像指数的展开式一样:

 $$B(x)=\sum_{n=0}^{\inf y} \frac{n!}{x^n=e^{e^x-1}}$

展开成第二类Stirling数,就有以下两个式子:

 $s=0^{\infty} \sum_{n=0}^{\infty} \sum_{k=0}^{\infty} \frac{s_2(n,k)}{n!}x^ny^k=e^{y(e^x-1)}$

 $s=0^{\infty} \$

对于第一类Stirling数,也有类似的指数型生成函数:

 $s=0}^{\left(1+x\right)^{s} \$

 $n=0^{\infty} \{n=0}^{\infty} \{n=0\}^{\infty} \{n,k\} \{n!\} x^n=\frac{1}{k!} {(\ln(1+x))}^k$

为了方便,引入一个记号"阶乘的比"表示一种自变量为x的一元多项式:

 $\frac{x!}{(x-n)!} = x(x-1)....(x-n+1)$

它是多项式,阶乘比只是一种形式上的记号,这里提前说明。

普通生成函数

对于第一类Stirling数,有按行的普通生成函数,直接配上系数,没有阶乘做分母:

 $s=0^{n} s 1(n,k)x^k=\frac{x!}{(x-n)!}$

通过同样手法就知道,它的绝对值的生成函数有:

\$\$\sum_{k=0}^{n} |s_1(n,k)|x^k=\frac{(x+n-1)!}{(x-1)!}\$\$

因为都包含了模p的全体零点,无论带不带绝对值都有:

 $s=0^{p} s_1(p,k)x^k=\frac{x!}{(x-p)!} equiv \sum_{k=0}^{p} |s_1(p,k)|x^k=\frac{(x+p-1)!}{(x-1)!} equiv x^p-x \mod p$

也就是说,除了两端的第一类Stirling数,第p行中间的第一类Stirling数都被p整除。

第二类没有按行的结论,但有按列的结论:

s=0^{\infty} S 2(n,k)x^n=\frac{x^k}{(1-x)(1-2x).....(1-kx)}\$\$

也有同余式:

 $s_{n=0}^{\infty} S_2(n,p)x^n=\frac{x^p}{(1-x)(1-2x).....(1-px)} \exp \sqrt{x^p}{1-x^{p-1}}\mod p$

矩阵的观点

接下来的阐述采用了矩阵"批量处理"的观点,来解释上一节的内容。

我们给出记法,数表构成的两个n阶方阵:

\$\${(s 1)} n\$\$

\$\${(S 2)} n\$\$

如果没有下角标,就默认为无穷阶阵。

上面这两个方阵是互逆的,乘在一起得到单位阵:

\${(s 1)} n{(S 2)} n=I n\$\$

因此下面的公式显然是成立的。

 $s_{x,k} = 1^{n} S_2(n,k) \frac{x!}{(x-k)!} = x^n$

 $s=0^{\infty}\$

 $s=0^{\infty} = 1_{xy}$

最后一个式子就是无穷大单位阵。

为什么互逆呢?这就要从矩阵的作用谈起。对于一元多项式F和G□如果它们之间存在着这样的关系:

 $\$\$G(x)=F(e^x-1)\$\$$

 $F(x)=G(\ln(1+x))$

那么,对于它们第n项的系数f n和g n[之间就应该满足这样的关系:

 $s_n=\sum_{k=0}^{\infty} S_2(n,k)f_k$

 $f = \sum_{k=0}^{\infty} s 1(n,k)g k$

即原来的多项式也可以直接写出:

 $sG(x)=\sum_{n=0}^{\left(n+0\right)^{\left(n+1\right)}} \sum_{n=0}^{\left(n+1\right)} S_2(n,k)f_kx^n$

 $f(x)=\sum_{n=0}^{\infty} \sum_{k=0}^{\infty} s 1(n,k)g kx^n$

我们把多项式的系数看作列向量f和g[]上面的等式就是熟知的矩阵左乘列向量。

\$g=(S 2)f\$\$

\$\$f=(s_1)g\$\$

2025/10/17 01:57 7/14 Stirling数

 $s=0^{\left(n=0\right^{\left(n,k\right)} \frac{1}{1-y(1+x)}}$

 $s=0^{\left(n=0\right^{\left(n+1\right)} \sum_{k=0}^{\left(n+1\right)} \frac{1-y(e^x-1)}}$

运用这个结论,几乎是显然的。我们还可以借此构造更多的实例,在此就不写了。

Lucas定理

生成函数

考虑Lucas定理的证明,见素数幂次与p进数问题第七节。

 $\c {m 1p+m 2}^{n 1p+n 2} \exp C {m 1}^{n 1}C {m 2}^{n 2} \mod p$$

只用到了:

 $$${(x+1)}^p\leq x^p+1 \mod p$$$

上文说,无论带不带绝对值都有:

 $s=0^{p} s 1(p,k)x^k \exp x^p-x \mod p$

除了两端的第一类Stirling数,第p行中间的第一类Stirling数都被p整除。

所以完全同样的可以写出第一类Stirling数与绝对值的Lucas定理:

也有第二类Stirling数按列的Lucas定理。

```
 $$\sum_{n=k_1p+k_2}^{\left(1-x^{p-1}\right)}^{k_1}}\sum_{n=k_2}^{\left(n,k_1p+k_2\right)x^n\leq x^{pk_1}}^{(1-x^{p-1})}^{k_1}}\sum_{n=k_2}^{\left(n,k_2\right)x^n=x^{pk_1}}\sum_{t=0}^{\left(t-t\right)t}\sum_{n=k_2}^{\left(t-t\right)t}} S_2(n,k_2)x^n\sum_{t=0}^{\left(t-t\right)t}\sum_{n=k_2}^{\left(t-t\right)t}} S_2(n,k_2)x^n\pmod{p$}
```

对于第二类Stirling数,这个式子可以直接展开。记带余除法:

 $$n-(k_1p+k_2)=n_1(p-1)+n_2$ \$

其中n 2取值为0到p-2□则有:

$$$$S_2(k_1p+k_2+n_1(p-1)+n_2,k_1p+k_2)=\sum_{t=0}^{n_1} C_{k_1+n_1-t-1}^{k_1-1}S_2(k_2+n_2+(p-1)t,k_2)$$
\$

这是一个较复杂的卷积式,并不是一一对应的化归,但是总之将情况化归到了0到p-1列的某一列中。

奇偶性

显然带不带绝对值奇偶性都是一样的,乘不乘-1奇偶性也是一样的。令n 2为0或者是1:

```
s_{k=0}^{2n_1} s_1(2n_1,k)x^k \leq x^{n_1}_{(x-1)}^{n_1}\sum_{k=0}^{0} s_1(0,k)x^k=x^{n_1}\sum_{t=0}^{n_1}C_{n_1}^tx^t \mod 2
```

 $\s = 0 ^{2n_1+1} s_1(2n_1+1,k)x^k \leq x^{n_1}_{(x-1)}^{n_1}\sum_{k=0}^{1} s_1(1,k)x^k = x^{n_1+1}\sum_{k=0}^{n_1}C_{n_1}^{x^k} \mod 2$

也就是说,每一行前面一半全是偶数,后面一半与它一半的那一行组合数奇偶性相同。

奇素数

接下来必须假设p不是2,即p是奇素数。由于只有第一类Stirling数有一一对应的化归,所以只研究第一类Stirling数。

二项式展开:

我们知道□n_2不超过p□即化归后到了前p-1行。原来一行的前n_1个数(0到n_1-1□都被p整除。

观察式中组合数部分。组合数部分每两项之间距离是p-1□但是后面k跑遍的部分是0到p-1□有p个数,会不会造成重叠?

不会。因为第一类Stirling数数表的形状:前p-1行中,只有第0行在第0列有数1,其余行在第0列都没有数 (为0);但是第0行,也只有在第0列有数1,在其余列都没有数。这就保证它们可以按照p-1的周期重复而不重叠。

为了方便讨论,将以上区分为第0行和1到p-1行两种情况,即整除与不整除两种情况。

当n 2等于0时:

 $s_{k=0}^{n_1p} s_1(n_1p,k)x^k \geq x^{n_1p} |s_1(n_1p,k)|x^k \geq x^{n_1}\sum_{t=0}^{n_1p} |s_1(n_1p,k)|x^k \geq x^{n_1}\sum_{t=0}^{n_1}(-1)}^{n_1+1}C {n_1}^{x^k} |s_1(n_1p,k)|x^k \geq x^{n_1}\sum_{t=0}^{n_1}(-1)}^{n_1+1}C {n_1}^{x^k} |s_1(n_1p,k)|x^k > x^{n_1}\sum_{t=0}^{n_1}(-1)}^{n_1+1}C {n_1}^{x^k} |s_1(n_1p,k)|x^k > x^{n_1}\sum_{t=0}^{n_1}(-1)}^{n_1}\sum_{t=0}^{n_1}(-1)^{x^k} |s_1(n_1p,k)|x^k > x^{n_1}\sum_{t=0}^{n_1}(-1)^{x^k} |s_1(n_1p,k)|x^k > x^{n_1}\sum_{t=0}^{n_1}($

2025/10/17 01:57 9/14 Stirling数

即p的n_1倍数行,前n_1个数(0到n_1-1 \square 被p整除,从第n_1个数开始后面每p-1个数中 \square p-1是偶数,因此奇偶性相同)才有一个不被p整除,并且与第n_1行对应的组合数(配上-1的系数)同余。即仅当k-n_1 是p-1倍数时不被p整除:

\$\$k-n_1=k_1(p-1)\$\$

则有:

 $s_1(n_1p,n_1+k_1(p-1)) = (-1)^{n_1-k_1}C_{n_1}^{k_1}\mod p$

当n_2不等于0时,前n_1个数(0到n_1-1 \square 仍旧被p整除。于是这里认为k至少为n_1 \square 将k-n_1 \square 原来k属于靠右的n-n_1列)做带余除法,只是除数换成p-1 \square 有:

\$\$k-n 1=k 1(p-1)+k 2\$\$

这里规定□k_2的取值范围是1到p-1□与通常的带余除法略有不同。由于k不超过n□而n有这个式子:

\$\$n-n 1=n 1(p-1)+n 2\$\$

这里n 2的范围是0到p-1□因此k 1不超过n 1□则有:

 $$5_1(n_1p+n_2,n_1+k_1(p-1)+k_2)\neq {(-1)}^{n_1-k_1}C_{n_1}^{k_1} s_1(n_2,k_2)\mod p$

 $|s_1(n_1p+n_2,n_1+k_1(p-1)+k_2)| = (-1)^{n_1+k_1}C_{n_1}^{k_1} |s_1(n_2,k_2)| = (-1)^{n_1+k_2}C_{n_1}^{k_1} |s_1(n_2,k_2)| = (-1)^{n_1+k_2}C_{n_1}^{k_1} |s_1(n_2,k_2)| = (-1)^{n_1+k_2}C_{n_2}^{k_1} |s_1(n_2,k_2)| = (-1)^{n_1+k_2}C_{n_2}^{k_2} |$

即从n_1开始,第k_1个连续p-1化归到了第一类Stirling数第n_2行的1到p-1位置,以及组合数的第n_1行的k_1位置。这样就完成了递归。绝对值可以去掉是因为保证了同样的奇偶性。

例题

第一类Stirling数绝对值同行部分和余数问题。求第一类斯特林数的绝对值n行l到r项和%p□

圆神现在正在解决一个有趣的任务,它给出四个整数n□□r和p□并询问:

 $s=1 ^r | 1(n,k) \mod p$

其中p是质数。似乎圆神在一分钟内就想出了主旨。你能快点吗?

输入唯一一行包含四个整数n□l□r和p□1≤n≤10^18□0≤l≤r≤n□2≤p≤10^6□p是质数)。

输出一个表示答案的整数。

题解

显然可以拆成两个从0开始的同行部分和的差。不妨记前m项为:

 $s=0^{m} | s 1(n 1p+n 2,k)|$

这一行给定了,仍旧可以设n比p大,于是可以化归到比p小的部分:

\$\$n=n 1p+n 2\$\$

这里n_1和n_2都相当于给定了。根据上文第一类Stirling数的Lucas定理,前n_1个数全都模p余0,即当m小于n 1的时候部分和全部为0,事实上只需要观察后半部分的部分和。

首先,右边两部分可以分开求和。类似于上文的k□设m也有相关的式子:

```
m-1=m 1(p-1)+m 2$
```

当n 2为0的时候,对m-n 1是简单的带余除法:

```
s_{k=0}^{m} |s_1(n_1p,k)| \leq k_1 C_{n_1}^{m_1} {(-1)}^{n_1-k_1} C_{n_1}^{k_1} \mod p
```

当n 2不为0的时候,对m-n 1是特殊的带余除法,余数m 2取值1到p-1□

```
 \begin{array}{l} $\scriptstyle k=0 ^{m} \\ |s_1(n_1p+n_2,k)| = \sum_{k=0}^{m_1-1}\sum_{k=2}^{p-1}|s_1(n_1p+n_2,n_1+k_1(p-1)+k_2)| \\ +\sum_{k=0}^{m_1-1}\sum_{k=2}^{p-1}|s_1(n_1p+n_2,n_1+k_2)| \end{array}
```

同一行第一类Stirling数绝对值部分和,恰好是行数的阶乘。即:

```
 \begin{array}{lll} $\scriptstyle k_0^{m} |s_1(n_1p+n_2,k)| & m_1-1 & (-1) ^{n_1-k_1} C_{n_1}^{k_1} + (-1) ^{n_1-m_1} C_{n_1}^{k_1} + (-1) ^{n_1-m_1} C_{n_1}^{k_2-1}^{k_2-1}^{k_2-1}^{k_2-1}^{k_2-1}^{k_2-1}^{k_2} & p$$ \end{array}
```

这里n 2总比p要小,因此阶乘不会被消掉。最后问题的关键在组合数的部分。

对k 2和n 2进行枚举,范围仍旧都是0到p-1□因此关键在左边n 1和k 1的部分。

```
\{(-1)\}^{n_1}\sum_{k_1=0}^{m_1-1} C_{n_1}^{k_1}\{(-1)\}^{k_1}
```

最后化归为简单的Lucas定理的问题,给定n 1□m 1-1□x□计算多项式取模意义下的值:

```
s_{k_1=0}^{m_1-1} C_{n_1}^{k_1}x^{k_1}
```

即二项式展开的部分和问题,见素数幂次与p进数问题第七节。

代码

```
#include<stdio.h>
int MOD;
long long powmod(long long x,int k)
{
    long long ans=1;
    while(k)
```

2025/10/17 01:57 11/14 Stirling数

```
{
        if(k&1)
            ans=ans*x%MOD;
        x=x*x%MOD;
        k >>= 1;
    return ans;
int getfact(int x,int *p)
    int t=x, sz=0;
    int i;
    for(i=2;i*i<=t;i++)
        if(x\%i==0)
             p[++sz]=i;
            while (x\%i==0)
                 x/=i;
    }
    if(x>1)
        p[++sz]=x;
    return sz;
}
long long facd[1000005], facv[1000005];
long long G,mi[1000005],inv[1000005];
void pre()
    facd[0]=1;
    int i;
    for(i=1;i<MOD;i++)</pre>
        facd[i]=facd[i-1]*i%MOD;
    facv[MOD-1]=facd[MOD-1];
    for(i=MOD-2;i>=0;i--)
        facv[i]=facv[i+1]*(i+1)%MOD;
    int prime[10];
    int sz=getfact(MOD-1,prime);
```

22:02

```
for(G=1;;G++)
        bool ok=1;
        for(i=1;i<=sz;i++)</pre>
             if (powmod(G, (MOD-1)/prime[i]) ==1)
             {
                  ok=0;
                  break;
             }
        if(ok)
             break;
    mi[0]=1;
    for(i=1;i<MOD-1;i++)</pre>
        mi[i]=mi[i-1]*G%MOD;
    inv[1]=1;
    for(i=2;i<MOD;i++)</pre>
        inv[i]=(MOD-MOD/i)*inv[MOD%i]%MOD;
}
long long C(long long n,long long m)
    return (n<m)?0:facd[n]*facv[m]%MOD*facv[n-m]%MOD;</pre>
}
long long calc(long long n,long long m)
    if(!m)
        return 1;
    if(n<m)</pre>
    {
        return 0;
    return C(n%MOD,m%MOD)*calc(n/MOD,m/MOD)%MOD;
}
long long query(long long n,long long m)
    m=(n < m) ?n : m;
```

2025/10/17 01:57 13/14 Stirling数

```
long long s=0;
                int i;
                for(i=0;i<MOD-1;i++)</pre>
                               int x=mi[i];
                               if(x+n>MOD)
                                               continue;
                               if(!i)
                                                s=(s+(m+1)*facd[x+n-1]%MOD*facv[x-1])%MOD;
                               else
                                                s = (s + (mi[(MOD - 1 - i*(m+1)%(MOD - 1))%(MOD - 1)] - 1LL)*inv[mi[MOD - 1 - i*(m+1)%(MOD - 1)] - 1LL)*inv[m
i]-1LL]%MOD*facd[x+n-1]%MOD*facv[x-1])%MOD;
               }
               s = (MOD - s) %MOD;
               if(n==MOD-1)
                               s=(s-1LL+MOD)%MOD;
               return s;
}
long long solve(long long n,long long m)
               long long u1=n/MOD, v1=n\%MOD;
               if(m<u1)
                {
                                return 0;
               m-=u1;
               long long u2=m/(MOD-1), v2=m\%(MOD-1);
               long long s=0;
               if(u2)
                               s = (s + (((u2-1)\&1)?MOD-1:1)*calc(u1-1,u2-1)%MOD*query(v1,MOD-2))%MOD;
                s=(s+((u2\&1)?MOD-1:1)*calc(u1,u2)%MOD*query(v1,v2))%MOD;
                if(v1==M0D-1\&\&u2)
                               s=(s+((u2-1)\&1)?MOD-1:1)*calc(u1-1,u2-1))*MOD;
                return s*((u1&1)?MOD-1:1)%MOD;
}
int main()
```

Last

 $2020\text{-}2021\text{:}teams:namespace:stirling} \\ \text{https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:namespace:stirling\%E6\%95\%B0\&rev=1596636141} \\ \text{https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:namespace:stirling\%E6\%95\%B0\&rev=15966361411 \\ \text{https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:namespace:stirling\%E6\%95\%B0\&rev=15966361411 \\ \text{https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:namespace:stirling\%E6\%95\%B0\&rev=15966361411 \\ \text{https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:namespace:stirling\%E6\%95\%B0\&rev=15966361411 \\ \text{https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:namespace:stirling\%E6\%95\%B0\&rev=15966361411 \\ \text{https://wiki.cvbbacm.com/doku.php.cvbbacm.com/doku.php.cvbbacm.com/doku.php.cvbbacm.com/doku.php.cvbbacm.com/doku.php.cvbbacm.com/doku.php.cvbbacm.com/doku.php.cvbbacm.com/doku.php.cvbbacm.com/doku.php.cvbbacm.com/doku.php.cvbbacm.com/doku.php.cvbbacm.com/doku.php.cvbbacm.com/doku$ update: 2020/08/05 22:02

```
long long n,l,r;
scanf("%lld%lld%d",&n,&l,&r,&MOD);
long long s1=solve(n,r);
long long s2=solve(n,l-1);
printf("%lld\n",(s1-s2+MOD)%MOD);
return 0;
```

https://wiki.cvbbacm.com/ - CVBB ACM Team

https://wiki.cvbbacm.com/doku.php?id=2020-2021: teams: namespace: stirling% E6% 95% B0& rev=1596636141. The property of the



