类欧几里得算法

洪华敦

hhdskydec@sina.com

需要用的的一些数学符号

- $[a]$ 若表达式a为真，则$[a]=1$，否则$[a]=0$
- 例如$[1+1=2]=1$
- % 取模
- $S_d(n)=\sum_{x=1}^{n}x^d$
- $\lfloor \frac{a}{b} \rfloor$ $\frac{a}{b}$下取整

欧几里得算法

- 对于$b \neq 0$，有$gcd(a, b) = gcd(b, a\%b)$

复杂度证明

- 为了更好地讲后面的东西，我们来证明下欧几里得算法的复杂度
- 考虑最坏情况，$a \le 2 * b - 1$
- 那么有$gcd(a, b) = gcd(b, a - b)$
- 将a, b设为比较接近的斐波那契数，那么$a = F_n$，$b = F_{n-1}$，则有$gcd(F_n, F_{n-1}) = gcd(F_{n-1}, F_{n-2})$
- 显然n是$O(\log(F_n))$的
- 所以这个算法的复杂度是$O(\log(a))$的

【清华集训2014】Sum

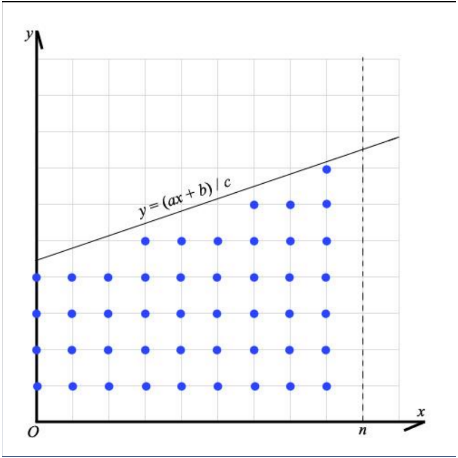
- 我们来看这样一个问题
- $\sum_{d=1}^{n}(-1)^{\lfloor\sqrt{d*r*d}\rfloor}$

做法

- 显然只要知道有多少$\left\lfloor \sqrt{d*r*d} \right\rfloor$是奇数就行了
- 于是就变成了求$\sum_{d=1}^{n} \left\lfloor \frac{d*\sqrt{r}}{2} \right\rfloor \%2$
- 考虑这个问题的一般形式$\sum_{i=0}^{n} \left\lfloor \frac{ax+b}{c} \right\rfloor$

传统做法

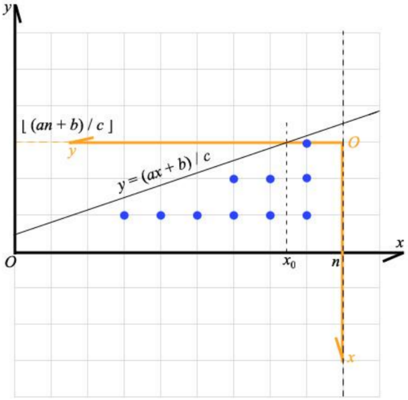
- 有一种传统的几何做法
- 我们要求的是这个



- (PS:图片选自其他课件)

传统做法

- 我们先把下面的整块矩形直接计算掉



- 我们以点$(n, \left\lfloor \frac{a*n+b}{c} \right\rfloor)$为原点重建坐标轴

传统做法

- 然后重新推导下这个直线的解析式
- 显然斜率是变成了倒数了的，所以a与c互换了
- 可以发现复杂度和计算gcd类似，都是$O(logn)$
- 这种做法虽然直观，但是有很大的局限性，下面介绍另一种做法

做法

- 首先，如果$a >= c$，设$d = \lfloor \frac{a}{c} \rfloor$则有：
- $\sum_{x=0}^{n} \lfloor \frac{ax+b}{c} \rfloor = \sum_{x=0}^{n} \left\lfloor \frac{(a\%c)x+b}{c} \right\rfloor + d * x$
- $= d * S_1(n) + \sum_{x=0}^{n} \left\lfloor \frac{(a\%c)x+b}{c} \right\rfloor$

做法

- 同理，如果有$b >= c$，设$d = \left\lfloor \frac{b}{c} \right\rfloor$则有：
- $\sum_{x=0}^{n} \left\lfloor \frac{ax+b}{c} \right\rfloor = \sum_{x=0}^{n} \left\lfloor \frac{ax+(b\%c)}{c} \right\rfloor + d$
- $= d * S_0(n) + \sum_{x=0}^{n} \left\lfloor \frac{ax+(b\%c)}{c} \right\rfloor$

做法

- 经过上面两个计算后，我们有$a < c$，$b < c$
- 于是我们的问题规模从(a, c, b, n)变成了$(a\%c, c, b\%c, n)$

做法

- 令$M = \left\lfloor \frac{an+b}{c} \right\rfloor$
- 原式$=\sum_{x=0}^{n}\sum_{y=0}^{M}[y < \left\lfloor \frac{ax+b}{c} \right\rfloor]$
- 我们来化简下这个式子$y < \left\lfloor \frac{ax+b}{c} \right\rfloor$
- $c*(y+1) \leq a*x+b$
- $c*y+c-b \leq a*x$
- $c*y+c-b-1 < a*x$
- $x > \left\lfloor \frac{c*y+c-b-1}{a} \right\rfloor$

做法

- 所以原式$=\sum_{y=0}^{M}\sum_{x=0}^{n}[x>\left\lfloor\frac{c*y+c-b-1}{a}\right\rfloor]$
- $=\sum_{y=0}^{M} n-\left\lfloor\frac{c*y+c-b-1}{a}\right\rfloor$
- $=n*(M+1)-\sum_{y=0}^{M}\left\lfloor\frac{c*y+c-b-1}{a}\right\rfloor$
- 于是我们的问题规模从$(a\%c, c, b\%c, n)$变成了$(c, a\%c, c-b\%c-1, M)$

复杂度

- 经过上面一番转化，我们的问题规模从(a, c, b, n)变成了$(c, a\%c, c - b\%c - 1, M)$
- 显然当a为0时，我们可以很轻松地计算答案
- 所以问题规模每次都从(a, c)变成$(c, a\%c)$
- 这和计算gcd的时间复杂度是一样的
- 所以时间复杂度是$O(log(a))$
- 看上去这种做法更加的麻烦，但是这种方法是可以扩展到高维情况的

WINDOW

- 简化题意：
- 给定N, A, B, K, L，计算:

$$\sum_{x=1}^{n} \binom{\lfloor \frac{Ax}{b} \rfloor}{K+1} \binom{n-x}{L}$$

- $N, A, B \le 10^{18}$，$K, L \le 10$
- 答案对一个大质数取模

做法

- 首先我们知道，$\binom{x}{k}$是一个关于x的k次多项式
- 设$\binom{\lfloor \frac{Ax}{b} \rfloor}{K+1} = \sum_{i=0}^{K+1} c_i * (\lfloor \frac{Ax}{b} \rfloor)^i$
- 设$\binom{n-x}{L} = \sum_{i=0}^{L} d_i * x^i$
- 那么原式$= \sum_{x=1}^{n} \sum_{i=0}^{L} d_i * x^i \sum_{j=0}^{K+1} c_j * (\lfloor \frac{Ax}{b} \rfloor)^j$
- 等于$\sum_{i=0}^{L} \sum_{j=0}^{K+1} d_i * c_j \sum_{x=1}^{n} x^i * (\lfloor \frac{Ax}{b} \rfloor)^j$
- d_i和c_i我们可以预处理用拉格朗日插值公式或者高斯消元算出来，那么问题就变成了求后面那些东西

做法

- 我们令$f(a,c,b,i,j)$表示$\sum_{x=0}^{n} x^i * (\lfloor \frac{a*x+b}{c} \rfloor)^j$
- 首先我们可以用二项式定理，让$a,b<c$
- 令$M = \lfloor \frac{a*n+b}{c} \rfloor$
- $$\sum_{x=0}^{n} x^U * (\left\lfloor \frac{a*x+b}{c} \right\rfloor)^V$$
- $$= \sum_{x=0}^{n} x^U * sum_{y=0}^{M} [y < \left\lfloor \frac{a*x+b}{c} \right\rfloor]((y+1)^V - y^V)$$
- $$= \sum_{y=0}^{M} ((y+1)^V - y^V) \sum_{x=0}^{n} x^U * [x > \left\lfloor \frac{c*y+c-b-1}{a} \right\rfloor]$$

做法

- 我们可以把$(y+1)^U-y^U$二项式展开
- 原式等于
- $=\sum_{i=0}^{V-1}\binom{V}{i}\sum_{y=0}^{M}y^i*\sum_{x=0}^{n}x^U*[x>\lfloor\frac{c*y+c-b-1}{a}\rfloor]$
- $=\sum_{i=0}^{V-1}\binom{V}{i}\sum_{y=0}^{M}y^i*(S_U(n)-S_U(\lfloor\frac{c*y+c-b-1}{a}\rfloor))$
- $=\sum_{i=0}^{V-1}\binom{V}{i}\sum_{y=0}^{M}y^i*S_U(n)-\sum_{i=0}^{V-1}\binom{V}{i}\sum_{y=0}^{M}y^i*S_U(\lfloor\frac{c*y+c-b-1}{a}\rfloor)$
- 前者等于$\sum_{i=0}^{V-1}\binom{V}{i}S_i(M)*S_U(n)$，于是问题就变成了求后者

做法

- 我们可以发现$S_d(x)$是一个关于x的$d+1$次多项式
- 设$S_d(x)=\sum_{i=0}^{d+1} a[d][i]*x^i$
- 数组a可以用高斯消元预处理，或者直接带伯努利数进去
- 原式等于$\sum_{i=0}^{V-1}\binom{V}{i}\sum_{y=0}^{M} y^i*\sum_{j=0}^{U+1} a[U][j]*(\lfloor\frac{c*y+c-b-1}{a}\rfloor)^j$
- 等于$\sum_{i=0}^{V-1}\sum_{j=0}^{U+1} a[U][j]*\binom{V}{i}\sum_{y=0}^{M} y^i*(\lfloor\frac{c*y+c-b-1}{a}\rfloor)^j$
- 于是我们再次完成了a,c的互换，这样递归下去只有$O(log(A))$层
- 我们可以对于每个(a,b,c,n)求出对于所有(i,j)的答案，具体方法就是递归下去，然后用返回的那些答案更新

小优化

- 然而裸着直接做好像是$O((K+L)^4 logn)$的，我们可以优化一下
- 对于每一个i，计算出$\sum_{j=0}^{U+1} a[U][j] \sum_{y=0}^{M} y^i * (\lfloor \frac{c*y+c-b-1}{a} \rfloor)^j$，这个是$(K+L)^3$的
- 然后对于每个V直接计算即可
- 时间复杂度$O((K+L)^3 logn)$
- 当然如果你用FFT的话复杂度会继续降下来