正睿OI 浙江省选模拟题3 Solution

安徽师范大学附属中学 罗哲正

2017 年 3 月 4 日

题目名称	Alchemy	Algebra	Anarchy
源文件名称	alchemy	algebra	anarchy
输入文件名	alchemy.in	algebra.in	anarchy.in
输出文件名	alchemy.out	algebra.out	anarchy.out
每个测试点时限	1s	4s	7s
测试点数目	10	25	20
每个测试点分值	10	4	5
内存限制	233MB	233MB	233MB
是否有部分分	否	否	否
题目类型	传统型	传统型	传统型
是否有SPJ	无	无	无
编译优化	-O2	-O2	-O2

1 Alchemy

我们直接模拟移动的过程，例如将盘子x从a借用b移到c，那么我们的策略是：

1.把x上面的盘子移到b

2.把x移到c

3.把刚刚移到b的再移到c

我们直接通过x的位置就可以知道当且正在进行哪一步分移动，若在第一部分可以直接考虑$x-1$的移动，若在第三部分那么直接吧第一部分加第二部分的2^{x-1}的步数加到答案，然后直接跳到第三部分开始，再接着考虑$x-1$的移动，这样考虑到最后一个盘子把加起来的步数输出就可以了。

详见代码。

2 Algebra

先思考一个问题，我们知道n, k, s，求大于等于n的第一个满足$f(x, k) = s$的整数，这个可以简单的使用贪心来完成，具体做法如下：

从高位到低位贪心，每一位尽可能的少放，如何尽可能的少放呢？可以从0开始依次尝试，尝试完之后看看用剩余的数字和填剩下的位能不能保证大于等于n，更精细的实现可以看std，省去每一位从零开始依次尝试的常数10。

那么我们实现了$next(n, k, s)$之后怎么做呢，我们令g_s表示大于等于n的满足$f(x, a) = f(x, b) = s$的整数不小于g_s（s的取值不会超过400）。

显然初始的时候所有的g_s都为n，接着，我们每次找到最小的$g_s = t$，令$w = max(next(t, a, s), next(t, b, s))$，若$w = t$则就找到了答案直接输出即可，否则令$g_s = w$，开始新的一轮的寻找。

为什么这样做是可以的呢？

我们考虑一个区间里$f(x, a) = s$和$f(x, b) = s$的数，考虑他们交替的次数，例如$a, b, a, a, a, b, b, a, a$等价于$a, b, a, b, a$交替了5。可以发现求$next$的次数就是交替次数，而交替次数的级别是不会低于任何一种数的出现次数的，由于进位的存在这两种数的出现较为随机，而根据生日悖论，碰撞出现的位置一般不超过出现次数的平方，所以寻找次数的上界是$O(\sqrt{n})$的，于是复杂度是$O(400 * \sqrt{n})$。但是注意到某些比较极端的s会导致交替次数很小，实际上这种算法速度是非常快的。

时间复杂度$O(400\sqrt{n})$，实现详见代码。

3 Anarchy

预备知识：快速傅里叶变换FFT，快速沃尔什变换FWT。

首先扔掉一堆吓人的公式，我们考虑求每个点的电势，其实就是求这么个东西：

$$\Phi(k) = \frac{1}{2m}\sum_{i\bigoplus j=k}\rho(i)*dist(j)^2$$

其中i,j,k都是用题目描述的方式代表坐标，$dist$是距离函数可以暴力求，ρ是输入的，$\bigoplus$运算就是满足$\mathbf{x}+\mathbf{y}=\mathbf{x}\Leftrightarrow x_i+y_i\equiv z_i \pmod{s}_i$的运算。

我们知道FWT的异或运算构造是：

$$F(a_0,a_1)=(F(a_0)+F(a_1),F(a_0)-F(a_1))$$

$$F^{-1}(a_0,a_1)=(\frac{F^{-1}(a_0)+F^{-1}(a_1)}{2},\frac{F^{-1}(a_0)-F^{-1}(a_1)}{2})$$

而实际上，异或运算每一位的本质就是$x+y\equiv z \pmod 2$，其实就是本题在$s_i=2$的情况。

那么对于s_i不等于2的情况应该怎么处理呢？

我们重新思考一下$F(a_0,a_1)$的构造，会发现我们只要满足当前这一位满足FWT的要求即可，即当a_0,a_1是数时成立就可保证当a_0,a_1为等长序列时成立。

接着我们可以发现$F(a_0,a_1)=(F(a_0)+F(a_1),F(a_0)-F(a_1))$本质上就是要满足一个长度为2的循环卷积的变换，那么只要使用离散傅里叶变换来构造就好了（接着还会发现异或卷积的变换式就是$n=2$的DFT的形式），我直接给出构造。

$$F(a_0,a_1,\cdots,a_{n-1})=DFT(a)$$

$$F^{-1}(a_0,a_1,\cdots,a_{n-1})=DFT^{-1}(a)$$

于是我们只要直接算DFT就行了，直接暴力DFT复杂度是$O(n*\sum s_i)$，能通过s_i不大的点。

那么如何优化DFT的计算呢？

我们可以使用Bluestein算法，令$\omega=e^{-\frac{2\pi}{n}}$：

$$\begin{aligned}A_m &= \sum_{k=0}^{n-1}\omega^{mk}a_k\\ &= \sum_{k=0}^{n-1}\omega^{\frac{m^2+k^2-(m-k)^2}{2}}a_k\\ &= \omega^{\frac{m^2}{2}}\sum_{k=0}^{n-1}\omega^{-\frac{(m-k)^2}{2}}*\omega^{\frac{k^2}{2}}a_k\end{aligned}\tag{1}$$

令$C_i = \omega^{\frac{i^2}{2}} a_i$，$B_i = \omega^{-\frac{(i-n)^2}{2}}$，且$C_i$只有在$0 \le i \le n-1$时非零。

$$\begin{aligned} A_m &= \omega^{\frac{m^2}{2}} \sum_{k=0}^{n-1} \omega^{\frac{(m-k)^2}{2}} * \omega^{\frac{k^2}{2}} a_k \\ &= \omega^{\frac{m^2}{2}} \sum_{k=0}^{n-1} B_{n+m-k} * C_k \end{aligned} \tag{2}$$

不妨设：

$$A_m = \omega^{\frac{m^2}{2}} A'_{n+m} \tag{3}$$

C_i只有在$0 \le i \le n-1$时非零，则有：

$$A'_{n+m} = \sum_{k=0}^{n+m} B_{n+m-k} * C_k \tag{4}$$

这就是一个标准的卷积形式了，使用FFT计算即可，于是时间复杂度是$O(n \log^2 n)$，由于FFT常数较大，可以当DFT的序列大小不超过50时使用暴力，就可以通过全部数据了。

FFT和DFT以及相关技巧更加详细的介绍可参考毛啸IOI2016国家集训队论文《再探快速傅里叶变换》。

扫码关注正睿教育