《具体数学》选讲

罗煜翔

宁波市镇海中学

Q&A&Q

- Q：怎么又是《具体数学》选讲？
- A：其实《具体数学》的内容只有一两页，剩下的大概都是多项式。
- Q：选的也太少了吧？
- A：本来就是来搞笑的。
- Q：你讲的都太水了，我都秒了该干什么？
- A：您或许可以考虑实现所有算法，或许足够您在这节课上使用时间的需求。
- Q：你讲的都太难了，我都听不懂该干什么？
- A：您或许可以改掉装弱的坏习惯。或者您可以看《具体数学》？
- Q：你讲的都很有道理，可我就是不想写该怎么办？
- A：不写也没关系，其实有些我也没写过。

一些说明

- 本课件中所有问题如无特殊说明模数均为 998244353，所有对数均以 2 为底。
- 默认大家都知道多项式运算（卷积的变形，任意模数乘法，求逆，求值，插值，对数，指数），高精度运算（加减法，乘法），二项式系数的性质与计算。
- 符号大概就是《具体数学》的符号，这里就不多说了。
- 由于我能力有限，一些算法并没有找到出处，一些问题也没有找到原题。

下降阶乘幂

下降阶乘幂

- 下降阶乘幂 $x^{\underline{m}} = x(x-1)\cdots(x-m+1) = m!\binom{x}{m} = \frac{x!}{(x-m)!} = \frac{\Gamma(x+1)}{\Gamma(x-m+1)}$ 。
- 上升阶乘幂 $x^{\overline{m}} = x(x+1)\cdots(x+m-1) = \frac{\Gamma(x+m)}{\Gamma(x)}$。
- 按照阶乘（伽马函数）比值的定理在 $\frac{\infty}{\infty}$ 时需要选择适当的极限值。
- 基本性质：
- $x^{\underline{m}} = (x-m+1)^{\overline{m}}$。
- $x^{\underline{m}} = (-1)^m(-x)^{\overline{m}}(m \in \mathbb{Z})$。
- $x^{\underline{n+m}} = x^{\underline{n}} \cdot (x-n)^{\underline{m}}$，与特殊情况 $1 = x^{\underline{n}} \cdot (x-n)^{\underline{-n}}$。

差分与和式

- 差分 $\Delta x^{\underline{m}} = (x+1)^{\underline{m}} - x^{\underline{m}} = m x^{\underline{m-1}}$。
- 不定和式 $\sum x^{\underline{m}} \delta x = \frac{x^{\underline{m+1}}}{m+1} + C$，其中 $m \neq -1$，C为有周期 1 的周期函数。
- 定和式 $\sum_a^b x^{\underline{m}} \delta x = \sum_{a \le x < b} x^{\underline{m}} = \frac{x^{\underline{m+1}}}{m+1}\Big|_a^b = \frac{b^{\underline{m+1}} - a^{\underline{m+1}}}{m+1}$，其中 $m \neq -1$。
- 当 $m = -1$ 时，和式变为了 $\sum \frac{1}{x+1} \delta x = H_x + C$，其中 H_x 为调和数。
- 差分简洁可以快速从 x 的情况变成 $x+1$ 的情况，求和简单可以在将形式转换为下降幂后快速求和。
- 下降幂与组合数的关系也可以带来一些相关的组合意义。
- 当然为了做到这一点我们需要先将信息转化为下降幂形式。

下降幂多项式与下降幂级数

- 我们定义下降幂多项式和（形式）下降幂级数：
- $f(x)=\sum_{i=0}^{n}a_i x^{\underline{i}}$ 或 $f(x)=\sum_{i=0}^{\infty}a_i x^{\underline{i}}$。
- 显然（形式）下降幂级数在全体自然数处收敛，在其他数基本都不收敛。
- 所以非常自然的，我们来观察下降幂级数在自然数处的点值。
- 可以发现点值大概是阶乘级别的，所以考虑点值的指数生成函数：
- $g(x)=\sum_{i=0}^{\infty}\frac{f(i)}{i!}x^i$。
- 对于下降幂单项式 $x^{\underline{n}}$，其对应点值指数生成函数为 $\sum_{i=n}^{\infty}\frac{i!}{(i-n)!i!}x^i=x^n e^x$。
- 如果设系数生成函数为 $h(x)=\sum_{i=0}^{\infty}a_i x^i$，那么 $g(x)=e^x h(x)$。

下降幂多项式与下降幂级数

- $g(x)=e^x h(x)$。
- 进行多项式求逆或者说二项式反演，显然有 $h(x)=e^{-x}g(x)$。
- 从而可以将 $0,1,2,\cdots,n$ 的点值和 $a_0,a_1,a_2,\cdots,a_n$ 这些系数通过一次多项式乘法进行转换。
- 从而也可以进行乘法，关于 $x^{\underline{n}}$ 求逆。
- 由于下降幂无法简单地进行复合，所以没有什么办法进行牛顿迭代。
- 可以通过和点值和多项式转换实现一般的多点求值和多点插值。
- 虽然后面这些东西通常并没有什么用，通常都是在点值和系数间进行转换。
- 现在对我们来说，$0,1,2,\cdots,n$ 的点值和下降幂形式就是等价的了。
- 而得到下降幂形式，也就是能做题啦！

下降幂多项式与下降幂级数

- 实际上，看到连续点值考虑求下降幂这样的操作，不仅是容易的，往往也是必要的。
- 因为实际上题目中出现连续点值表明出题人利用了连续点值的某些性质。而在实际上，这些性质大多数情况下可以用下降幂进行表达。
- 于是我们就用转下降幂完成了思考的过程，加速了思考。
- 从某种意义上来讲，就是减少了难度。
- 当然有些时候也有一些毒瘤故意给多项式形式却要求连续点值，这样就是要你写一个多点求值。实际上这也是有特殊方法比标准的做法快的。
- 而给你连续点值却要插值成多项式才能做的题，那我只能说出题人真是一个睿智的毒瘤！

【清华集训2016】如何优雅地求和

- 给定 m 次多项式 f 在 $0,1,2,\cdots,m$ 的值，整数 n,x，计算：
- $Q(f,n,x)=\sum_{k=0}^{n}f(k)\binom{n}{k}x^k(1-x)^{n-k}$。
- $1\le m\le 20000,1\le n\le 10^9,0\le x<998244353$。

- 按照套路看到了连续点值我们就先转成下降幂多项式来考虑。
- 于是我们只需要考虑 $f(x)=x^{\underline{i}}$ 的情况，此时：
- $Q\left(x^{\underline{i}},n,x\right)=\sum_{k=i}^{n}\frac{k!}{(k-i)!}\cdot\frac{n!}{(n-k)!k!}x^k(1-x)^{n-k}=x^i n^{\underline{i}}$。
- 于是可以 $O(m)$ 完成后续的计算。

一个基础期望DP练习题

- 有一个大小为 n 的数组，其第 i 个位置有 p_i 的概率为 1，$1-p_i$ 的概率为 0。
- 求这个数组中 1 的极长连续段长度的 k 次方和的期望。
- $n \leq 10^7, k \leq 10$。

- 无法直接DP答案，因为无法方便的从 x^k 的期望得到 $(x+1)^k$ 的期望。
- 如果DP所有的 $0 \sim k$ 次方，每次就可以 $O(k^2)$ 转移。
- 如果DP所有的 $0 \sim k$ 次下降幂，$(x+1)^{\underline{k}} = x^{\underline{k}} + kx^{\underline{k-1}}$，可以 $O(k)$ 转移。
- 在需要取模时可能改为二项式系数更好。
- 事实上有 $O(n \log n)$ 对任意给定点值多项式求期望的方法。

下降幂平移

- 给定一个 n 次多项式在 a 处的各阶差分，求它在 b 处的各阶差分。
- $n \le 10^6$。

- 显然 (a,b) 的答案与 $(0,b-a)$ 的答案是一样的。
- 对于 $a=0$，前者相当于给出了下降幂形式。我们考虑下降幂的高阶差分：
- $\Delta^m x^{\underline{n}} = n^{\underline{m}} x^{\underline{n-m}} = n! \cdot \frac{x^{\underline{n-m}}}{(n-m)!}$
- 直接进行一次多项式乘法即可，时间复杂度 $O(n \log n)$。

斯特林数

斯特林数

- 这里我们直接使用下降幂，普通幂和上升幂的关系定义斯特林数：
- $x^{\overline{n}} = \sum_k \begin{bmatrix} n \\ k \end{bmatrix} x^k$，$x^n = \sum_k \begin{Bmatrix} n \\ k \end{Bmatrix} x^{\underline{k}}$。
- 可以得到两个基本递推式：
- $\begin{bmatrix} n \\ k \end{bmatrix} = (n-1)\begin{bmatrix} n-1 \\ k \end{bmatrix} + \begin{bmatrix} n-1 \\ k-1 \end{bmatrix}$ 和 $\begin{Bmatrix} n \\ k \end{Bmatrix} = k\begin{Bmatrix} n-1 \\ k \end{Bmatrix} + \begin{Bmatrix} n-1 \\ k-1 \end{Bmatrix}$
- 这样可以在 $O(NK)$ 的时间内求出 $0 \le n \le N, 0 \le k \le K$ 的所有斯特林数。
- 但很多时候我们并不需要这么多项，也无法接受这样的复杂度。
- 通常我们有 6 种需求，即固定 n，固定 k 和固定 n, k。
- 下面对这六种情况分别考虑。

第二类斯特林数——行

- 先来考虑第二类斯特林求一行也就是固定 n 的情况，这也是最简单的一种。
- 事实上，对于 x^n 这个多项式，我们可以在 $O(n)$ 的时间复杂度中利用线性筛求出 $0 \sim n$ 的点值。
- 现在我们要求这个点值对应的下降幂多项式，利用之前的结果我们只需要将它对应的指数生成函数和 e^{-x} 做乘法就行了。
- 时间复杂度 $O(n \log n)$。

第一类斯特林数——行

- 下面是第一类斯特林数求一行。
- 相当于我们要将 $x^{\overline{n}}$ 展开成多项式。
- 直接分治复杂度是 $O(n \log^2 n)$ 的，不够优秀，我们考虑用倍增完成这个问题。
- 假设我们已经求出了 $x^{\overline{n}}$，现在我们需要由此得到 $x^{\overline{2n}}$ 或 $x^{\overline{n+1}}$。
- 后者直接做乘法即可，对于前者，相当于 $x^{\overline{n}} \cdot (x+n)^{\overline{n}}$。
- 于是我们需要根据 $x^{\overline{n}}$ 求出 $(x+n)^{\overline{n}}$ 或者说更一般的，给定多项式 $f(x)$ 求出 $f(x+c)$ 的系数。

多项式平移~~（见《算法导论》）~~

- 考虑将 $f(x)$ 在 c 处泰勒展开 $f(x)=\sum_{i=0}^{n}\frac{(x-c)^i f^{(i)}(c)}{i!}$。
- 于是有 $f(x+c)=\sum_{i=0}^{n}\frac{x^i f^{(i)}(c)}{i!}$，于是只需要求多项式 $f(x)$ 在 c 处的所有阶导数。
- 注意 $\left(\frac{\mathrm{d}}{\mathrm{d}x}\right)^m x^n=n^{\underline{m}}x^{n-m}=n!\cdot\frac{x^{n-m}}{(n-m)!}$，所以 $f^{(m)}(c)=\sum_{i=m}^{n}a_i i!\cdot\frac{x^{i-m}}{(i-m)!}$。
- 与前面讲过的下降幂平移类似，可以用一次多项式乘法解决。
- 于是可以在 $O(n\log n)$ 的时间内求出 $f(x+c)$ 的系数。
- 对于第一类斯特林数求一行，我们可以在 $O(n\log n)$ 的时间扩大一倍。
- $T(2n)=T(n)+O(n\log n)$，从而 $T(n)=O(n\log n)$。

第二类斯特林数——列

- 利用将 n 个元素分为 k 个非空子集的组合意义可以得到指数生成函数：
- $\left(\sum_{i\geq 1}\frac{x^i}{i!}\right)^m=(e^x-1)^m=m!\sum_{n\geq 0}\left\{{n\atop m}\right\}\cdot\frac{z^n}{n!}$，从而我们可以利用指数与对数在 $O(n\log n)$ 的时间求出第二类斯特林数的一列前 n 项。
- 但众所周知，指数与对数比较慢。
- 如果将斯特林数推广到负的，可以发现：$\left[{n\atop k}\right]=\left\{{-k\atop -n}\right\}$。
- 实际上固定 m 的第二类斯特林数相当于将 $x^{\overline{-m}}$ 在 ∞ 处展开，或 $(x^{-1})^{\overline{-m}}$ 在 0 处展开。
- 注意到 $x^{\overline{-m}}=\frac{1}{(x-1)^{\underline{m}}}$，实际上相当于一次上一部分的倍增和一次求逆。
- 时间复杂度还是 $O(n\log n)$ 的。

第一类斯特林数——列

- 同样利用将 n 个元素分成 k 个轮换的组合意义，可以得到指数生成函数：
- $\left(\sum_{i\geq 1}\frac{x^i}{i}\right)^m=\left(\ln\frac{1}{1-x}\right)^m=m!\sum_{n\geq 0}\begin{bmatrix}n\\m\end{bmatrix}\cdot\frac{x^n}{n!}$。
- 直接利用指数和对数得到 $O(n\log n)$ 的计算方法。
- 我并不是很清楚不用指数和对数的方法。

第一类斯特林数——单个

- 现在我们来考虑求单点的斯特林数的值。
- 显然直接套用刚才的算法就可以得到 $O(n \log n)$ 的算法，但我们显然不满足于此。
- 类比二项式系数的情况，我们希望能够对高精度级别的 n, m 来计算斯特林数的值。
- 类比二项式系数，我们需要 p 不太大，这里我们认为模一个 10^7 级别的数。
- 同时对于不超过 p 级别的大小，我们希望能得到一个比 $O(n \log n)$ 更快的算法。
- 由二项式系数的经验，我们猜测需要将数表示成 p 进制。

进制转换

- 我们需要快速将一个大整数 N 转换由 10 进制转换为 p 进制。
- 设 $N = A \cdot 10^k + B$，如果我们已经计算出了 $A, B, 10^k$ 在 p 进制下的表示就可以用一次高精度乘法和一次高精度加法完成。
- 如果同时计算 N 和 10^{2k}，就可以用两次 k 规模的递归完成 $A, B, 10^k$ 的计算。
- 一个 k 位 10 进制整数在 p 进制下有 $k \log_p 10$ 位。
- $T(2k) = 2T(k) + O\left(k \log_2 k \cdot \log_p 10\right)$。
- $T(n) = O\left(n \log^2 n \cdot \log_p 10\right) = O(n \log^2 n)$。
- 如果 p 是 10^7 级别的，那么这实际上是 $O(n \log n)$ 的，但 p 不一定很大。
- 事实上，可以先转换为 $p^k \geq \sqrt{n}$ 进制，则总有 $T(n) = O(n \log n)$。

第一类斯特林数——单个

- 注意到，上升幂 $x^{\overline{p}} \equiv x^p - x (\bmod p)$，于是对 $n = qp + r, 0 \le r < p$，我们有 $x^{\overline{n}} \equiv (x^p - x)^q \cdot x^{\overline{r}} \equiv x^q (x^{p-1} - 1)^q \cdot x^{\overline{r}} (\bmod p)$。
- 于是当 $m < q$ 时一定为 0，当 $m \ge q$ 时设 $k = m - q = q'(p-1) + r'$。
- 若 $r < p - 1$，则 $x^{\overline{r}}$ 次数小于 $p-1$，所以其中 r' 次项系数和 $(x^{p-1} - 1)^q$ 的 $q'(p-1)$ 次项系数的乘法就是答案。后者是 $\binom{q}{q'}(-1)^{q-q'}$，前者是 $\begin{bmatrix} r \\ r' \end{bmatrix}$。
- 若 $r = p - 1$，则 $x^{\overline{r}}$ 次数等于 $p-1$，但常数项为 0。若 $r' > 0$ 则与上面一样，否则是 $x^{\overline{r}}$ 中 $p-1$ 次项系数和 $(x^{p-1} - 1)^q$ 中的 $(q'-1)(p-1)$ 项系数的乘积，之后与上面类似。
- 组合数通过 p 进制表示计算，接下来需要考虑一个 $n, m < p$ 的 $\begin{bmatrix} n \\ m \end{bmatrix}$ 的计算。

第一类斯特林数——单个

- 我们需要求 $x^{\overline{n}}$ 这个多项式 x^m 项系数的值，但大多数的计算都需要计算出整个多项式的情况。
- 事实上，我们可以用 NTT 计算这一点。
- 假设 $m < p-1$。取模 p 的原根 g，可以得到：
- $\begin{bmatrix} n \\ m \end{bmatrix} \equiv \frac{1}{p-1} \Sigma_{i=0}^{p-2} \left(g^i\right)^{\overline{n}} \cdot g^{-im} \pmod p$。
- 原根的幂和一个数的上升幂都可以在 $O(p)$ 预处理后 $O(1)$ 直接计算。
- 于是计算一个值的时间复杂度是 $O(p)$ 的。

第二类斯特林数——单个

- 我们从 $\begin{bmatrix} -m \\ -n \end{bmatrix}$ 的角度考虑第二类斯特林数。
- 需要计算 $(x^{-1})^{\overline{-m}} = \frac{1}{(x^{-1}-1)^{\underline{m}}}$ 展开式中 x^n 系数。
- 设 $m = qp + r$，于是 $\frac{1}{(x^{-1}-1)^{\underline{m}}} \equiv \frac{x^{pq}}{(1-x^{p-1})^q (x^{-1}-1)^{\underline{r}}} \pmod{p}$。
- $\frac{x^{pq}}{(1-x^{p-1})^q} = x^{pq} \cdot \sum_{i=0}^{\infty} x^{i(p-1)} (-1)^i \binom{-q}{i} = x^{pq} \cdot \sum_{i=0}^{\infty} x^{i(p-1)} \binom{q+i-1}{i}$。
- 设 $n - m = k = q'(p-1) + r'$，于是有：
- $\begin{Bmatrix} n \\ m \end{Bmatrix} \equiv \sum_{i=0}^{q'} \begin{Bmatrix} i(p-1) + r' + r \\ r \end{Bmatrix} \cdot \binom{q + q' - i - 1}{q' - i} \pmod{p}$。
- 直接计算还是无法接受的。

第二类斯特林数——单个

- $\begin{Bmatrix} n \\ m \end{Bmatrix} \equiv \Sigma_{i=0}^{q'} \begin{Bmatrix} i(p-1)+r'+r \\ r \end{Bmatrix} \cdot \binom{q+q'-i-1}{q'-i} \pmod{p}$。
- 当 $r=0$ 时，只有 $i=r'=0$ 时项才非零。
- 当 $r \neq 0$ 时，考虑第二类斯特林数的计算公式：
- $\begin{Bmatrix} n \\ m \end{Bmatrix} = \frac{1}{m!} \Sigma_{i=0}^{m} \binom{m}{i} (-1)^{m-i} i^n$。
- 由费马小定理，对 $n>0$ 有 $i^{n+p-1} \equiv i^n \pmod{p}$，从而这些项的斯特林数部分在模意义下相同。
- 所以需要求二项式系数的和。显然有：
- $\binom{q'+q}{q'} = \Sigma_{i=0}^{q'} \binom{q+i-1}{i}$。

第二类斯特林数——单个

- 最后的问题是求单个的第二类斯特林数。
- 这是非常容易的，按照公式直接计算：
- $\left\{ {n \atop m} \right\} = \frac{1}{m!}\sum_{i=0}^{m} \binom{m}{i} (-1)^{m-i} i^n$。
- 预处理阶乘及逆元和数字的 n 次方即可。
- 这些都可以 $O(n)$ 完成计算。
- 由于 $n < p$，所以就在 $O(p)$ 的时间完成了单个的计算。

CCFEB18 Lucas Theorem

- 给定 n, p，其中 p 是质数，求满足 $\begin{bmatrix} n+1 \\ k \end{bmatrix}$ 不是 p 的倍数的 k 的个数。
- $n < 10^{501}, p \leq 10^5$。

- 利用求单个的结论，这里我们需要一个二项式系数和一个第一类斯特林数的乘积不是 p 的倍数。
- 相当于两者倍数 p 倍数的方案相乘。
- 对于第一类斯特林数非零，直接求出所有的计算即可。
- 对于二项式系数非零，这里上指标固定下指标变化，就是 p 进制下每一位不超过一个数，做乘法即可。
- 这里第一类斯特林数也可以用 NTT 的方法求，需要用卷积实现 NTT。

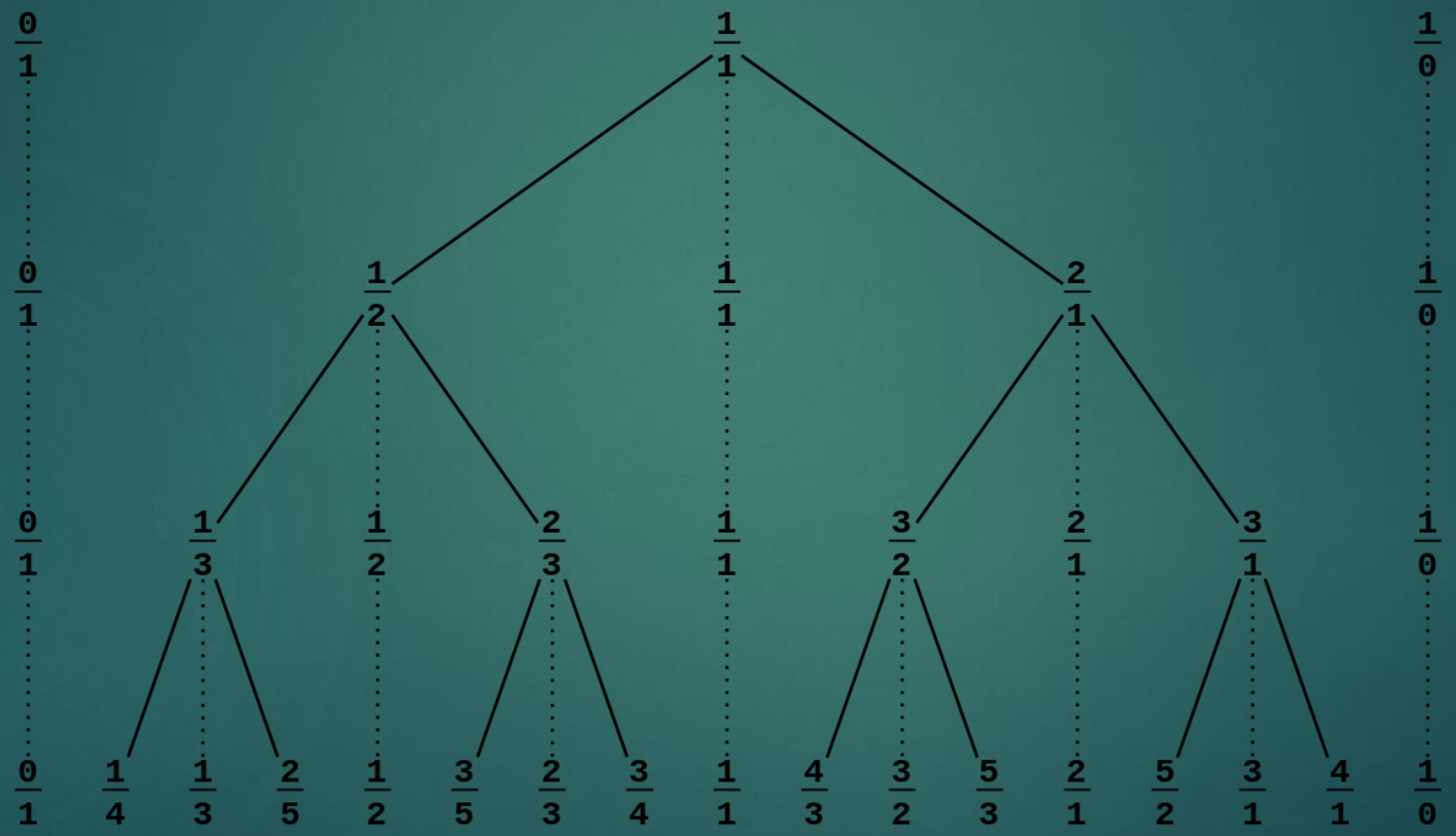
CCFEB18 Lucas Theorem——改

- 给定 N,k,p，其中 p 是质数，求满足 $\begin{Bmatrix} n \\ k \end{Bmatrix}$ 不是 p 的倍数且 $0 \leq n \leq N$的 n 的个数。
- $N,k < 10^{100000}, p \leq 10^5$。

- 这里还是一个二项式系数和一个斯特林数乘积非零。
- 但是这里 n 的范围是在一定的区间内变换的。
- 这里直接进行数位Dp就好了。

杂题选讲

Stern-Brocot 树



Stern-Brocot 树

- 这里只是描述一下如何在SB树上找一个数。
- 通常的做法是基于给定 $\frac{m}{n}$ 的情况下拐点是 $O(\log n)$ 的。
- 每次二分拐点是这个方向的第几个可以做到 $O(\log^2 n)$。
- 这个算法也非常容易卡满，斐波那契就行了。
- 实际上，每次连续走的长度给出了 $\frac{m}{n}$ 的连分数展开。
- 连分数展开中，除了最开始的一项都是正整数。
- 而这些正整数的乘积是不超过 n 的。
- 如果每次倍增拐点的位置就可以做到 $O(\log n)$ 的复杂度。

CTST2019R1T1D1P2

- 给定正整数 $n \geq 3$，构造两组长度为 n 的正整数等差数列，使得这 $2n$ 个数两两不同且两组数的乘积相同。
- $3 \leq n \leq 50$。
- ~~判断是否有无穷组本质不同的解。~~

- 观察小的 n 的最小解，发现 $n = 5,7,8$ 时两个公差中较小的都是 $2^n - 1$，较大的是 $2^{n+1} - 2$。$n = 6$ 时只是进行了约分。
- 进一步观察，发现 $a_i = n + (2^n - 1)i$，$b_i = 2n + (2^{n+1} - 2)(i - 1)$ 满足条件。
- 于是愉快地直接输出就好啦，是不是很水呢？

CTST2019R1T2D2P6

- 小凯手中有两种面值的金币，两种面值均为正整数且彼此互素。每种金币小凯都有无数个。在不找零的情况下，仅凭这两种金币，有些物品他是无法准确支付的。现在小凯想知道在无法准确支付的物品中，所有物品价值的 k 次方和是多少？
- $2 \leq a, b \leq 10^{18}, 1 \leq k \leq 100000$。
- ~~求证，存在只与 k 相关的常数 n，使得结果的 n 倍是 $(a-1)(b-1)$ 的倍数。~~

- 通过打表找规律发现答案是关于 a, b 的次数不超过 k 的多项式，并且含有 $(a-1)(b-1)$ 这个因式。
- 于是我们可以暴力出 a, b 分别为前 $k+1$ 个质数和接下来 $k+1$ 个质数的情况，插值出这个多项式。这样复杂度就是关于 k 的多项式级别的了。

CTST2019R1T2D2P6

- 然而这个算法并没有什么潜力，因为这个多项式的规模是 $O(k^2)$ 的已经无法接受了。
- 我们需要更加成熟的手段处理这个问题。
- 定义：$f(n)=[n\geq 0\wedge\forall i,j\in\mathbb{N},ai+bj\neq n]$，那么要求 $\sum_n f(n)n^k$。
- 由伯努利数的经验，我们转而求形式幂级数 $\sum_n f(n)e^{nx}$ 的 k 次项系数。
- 事实上一个布尔值是不太方便的，计数更接近我们平常的操作。
- 考虑 n 表示为 $ai+bj$ 的方案数 $g(n)$，当 $0\leq n<ab$ 时 $g(n)=1-f(n)$。
- 容易发现 $g(n)$ 满足 $g(n+a+b)=g(n+a)+g(n+b)-g(n)$。
- 于是对 $0\leq n<ab-a-b$，$f(n+a+b)=f(n+a)+f(n+b)-f(n)$。
- 当 $n=ab-a-b$ 时显然 $f(n+a+b)=f(n+a)+f(n+b)-f(n)+1$。

CTST2019R1T2D2P6

- 这表明，数列 $\{f(0), f(1), f(2), \cdots,\}$ 有递推式：
- $f(n+a+b) = f(n+a) + f(n+b) - f(n) + [n = ab - a - b]$
- 当 $-(a+b) \le n < 0$ 的时候，会有其他的特判。
- $\sum_n \big(f(n) - f(n-a) - f(n-b) + f(n-a-b)\big)x^n = x^{ab} + \sum_{0 \le n < a+b} t_n x^n$ 。
- 若生成函数为 $F(x)$，则：
- $\big(x^{a+b} - x^a - x^b + 1\big)F(x) = x^{ab} + R(x)$，其中 $R(x)$ 的次数小于 $a+b$。
  $R(x)$ 表示的就是其他的特判。
- 现在我们只需要求 $R(x)$ 了。

CTST2019R1T2D2P6

- 事实上 $R(x)=\left(1-x^a-x^b+x^{a+b}\right)F(x) \bmod x^{a+b}$。
- 考虑 $F(x) \bmod x^{a+b}$，就是当 $0 \le n < a+b$ 时 $f(n)$ 的值，显然：
- $f(n)=0 \Leftrightarrow a|n \vee b|n$。
- $F(x) \equiv \sum_{n\ge 0}\left(x^n-x^{an}-x^{bn}\right)+1 \equiv \frac{1}{1-x}-\frac{1}{1-x^a}-\frac{1}{1-x^b}+1\left(\bmod x^{a+b}\right)$。
- 从而 $R(x)=(1-x^a)\left(1-x^b\right)\left(\frac{1}{1-x}-\frac{1}{1-x^a}-\frac{1}{1-x^b}+1\right) \bmod x^{a+b}$。
- 由此可得：
- $R(x) \equiv \frac{(1-x^a)(1-x^b)}{1-x}-1\left(\bmod x^{a+b}\right)$
- 事实上，右边已经是 $a+b-1$ 次的多项式了，所以两端是相等的。

CTST2019R1T2D2P6

- 现在我们就得到了 $R(x)$，由此可得 $F(x)$：
- $F(x)=\frac{x^{ab}-1}{(x^a-1)(x^b-1)}-\frac{1}{x-1}$。
- 我们要求的是 $\sum_n f(n)e^{nx}=F(e^x)$，所以答案就是：
- $F(e^x)=\frac{e^{abx}-1}{(e^{ax}-1)(e^{bx}-1)}-\frac{1}{e^x-1}$。
- ~~由此容易证明存在只与 k 相关的常数 n，使得结果的 n 倍是 $(a-1)(b-1)$ 的倍数。~~
- 我们只需要多项式乘法和多项式求逆就可以解决这个问题。
- 具体实现时需要现在两端乘 x 保证可以分开计算，多组数据可以预处理伯努利数使得每次不需要求逆。

祝大家一起AK！