多项式选讲

宁波市镇海中学 梁晏成

2018 年 2 月 24 日

Overview

Before


中国移动
14:35
100%
王哲凡
手机在线 - 4G
? ? ?
log
我感觉你除了插值两个~~lg~~全都会啊
插值两个log是哪个
我可能也会
有点忘了
很黑科技吗?
蛤
GIF

多项式定义

这些式子都可以看作几个单项式的和. 例如，$v-2.5$ 可以看作单项式 v 与 -2.5 的和；$x^2+2x+18$ 可以看作单项式 x^2，$2x$ 与 18 的和.

像这样，几个单项式的和叫做**多项式**（polynomial）. 其中，每个单项式叫做多项式的**项**（term），不含字母的项叫做**常数项**（constant term）. 例如，多项式 $v-2.5$ 的项是 v 与 -2.5，其中 -2.5 是常数项；多项式 $x^2+2x+18$ 的项是 x^2，$2x$ 与 18，其中 18 是常数项.

多项式里，次数最高项的次数，叫做这个**多项式的次数**（degree of a polynomial）. 例如，多项式 $v-2.5$ 中次数最高项是一次项 v，这个多项式的次数是 1；多项式 $x^2+2x+18$ 中次数最高项是二次项 x^2，这个多项式的次数是 2.

> $v+2.5$，$3x+5y+2z$，$\frac{1}{2}ab-\pi r^2$ 的项分别是什么？次数分别是多少？

单项式与多项式统称**整式**（integral expression）. 例如，上面见到的单项式 $100t$，$0.8p$，mn，a^2h，$-n$，以及多项式 $v+2.5$，$v-2.5$，$3x+5y+2z$，$\frac{1}{2}ab-\pi r^2$，$x^2+2x+18$ 等都是整式.

多项式定义

这些式子都可以看作几个单项式的和. 例如，$v-2.5$ 可以看作单项式 v 与 -2.5 的和；$x^2+2x+18$ 可以看作单项式 x^2，$2x$ 与 18 的和.

像这样，几个单项式的和叫做**多项式**（polynomial）. 其中，每个单项式叫做多项式的**项**（term），不含字母的项叫做**常数项**（constant term）. 例如，多项式 $v-2.5$ 的项是 v 与 -2.5，其中 -2.5 是常数项；多项式 $x^2+2x+18$ 的项是 x^2，$2x$ 与 18，其中 18 是常数项.

多项式里，次数最高项的次数，叫做这个**多项式的次数**（degree of a polynomial）. 例如，多项式 $v-2.5$ 中次数最高项是一次项 v，这个多项式的次数是 1；多项式 $x^2+2x+18$ 中次数最高项是二次项 x^2，这个多项式的次数是 2.

$v+2.5$，$3x+5y+2z$，$\frac{1}{2}ab-\pi r^2$ 的项分别是什么？次数分别是多少？

单项式与多项式统称**整式**（integral expression）. 例如，上面见到的单项式 $100t$，$0.8p$，mn，a^2h，$-n$，以及多项式 $v+2.5$，$v-2.5$，$3x+5y+2z$，$\frac{1}{2}ab-\pi r^2$，$x^2+2x+18$ 等都是整式.

- 以上内容选自人教版七年级数学上册

多项式定义

这些式子都可以看作几个单项式的和．例如，$v-2.5$ 可以看作单项式 v 与 -2.5 的和；$x^2+2x+18$ 可以看作单项式 x^2，$2x$ 与 18 的和．

像这样，几个单项式的和叫做**多项式**（polynomial）．其中，每个单项式叫做多项式的**项**（term），不含字母的项叫做**常数项**（constant term）．例如，多项式 $v-2.5$ 的项是 v 与 -2.5，其中 -2.5 是常数项；多项式 $x^2+2x+18$ 的项是 x^2，$2x$ 与 18，其中 18 是常数项．

多项式里，次数最高项的次数，叫做这个**多项式的次数**（degree of a polynomial）．例如，多项式 $v-2.5$ 中次数最高项是一次项 v，这个多项式的次数是 1；多项式 $x^2+2x+18$ 中次数最高项是二次项 x^2，这个多项式的次数是 2．

$v+2.5$，$3x+5y+2z$，$\frac{1}{2}ab-\pi r^2$ 的项分别是什么？次数分别是多少？

单项式与多项式统称**整式**（integral expression）．例如，上面见到的单项式 $100t$，$0.8p$，mn，a^2h，$-n$，以及多项式 $v+2.5$，$v-2.5$，$3x+5y+2z$，$\frac{1}{2}ab-\pi r^2$，$x^2+2x+18$ 等都是整式．

- 以上内容选自人教版七年级数学上册
- 什么，你说你不知道单项式的定义？

单项式定义

这些式子都是数或字母的积，像这样的式子叫做**单项式**（monomial）．单独的一个数或一个字母也是单项式．

单项式中的数字因数叫做这个单项式的**系数**（coefficient）．例如，单项式 $100t$，a^2h，$-n$ 的系数分别是 100，1，-1．单项式表示数与字母相乘时，通常把数写在前面．

一个单项式中，所有字母的指数的和叫做这个**单项式的次数**（degree of a monomial）．例如，在单项式 $100t$ 中，字母 t 的指数是 1，$100t$ 的次数是 1；在单项式 a^2h 中，字母 a 与 h 的指数的和是 3，a^2h 的次数是 3．

对于单独一个非零的数，规定它的次数为 0．

单项式定义

这些式子都是数或字母的积，像这样的式子叫做**单项式**（monomial）．单独的一个数或一个字母也是单项式．

单项式中的数字因数叫做这个单项式的**系数**（coefficient）．例如，单项式 $100t$，a^2h，$-n$ 的系数分别是 100，1，-1．单项式表示数与字母相乘时，通常把数写在前面．

一个单项式中，所有字母的指数的和叫做这个**单项式的次数**（degree of a monomial）．例如，在单项式 $100t$ 中，字母 t 的指数是 1，$100t$ 的次数是 1；在单项式 a^2h 中，字母 a 与 h 的指数的和是 3，a^2h 的次数是 3.

对于单独一个非零的数，规定它的次数为 0.

- 以上内容选自人教版七年级数学上册

单项式定义

这些式子都是数或字母的积，像这样的式子叫做**单项式**（monomial）. 单独的一个数或一个字母也是单项式.

单项式中的数字因数叫做这个单项式的**系数**（coefficient）. 例如，单项式 $100t$，a^2h，$-n$ 的系数分别是 100，1，-1. 单项式表示数与字母相乘时，通常把数写在前面.

一个单项式中，所有字母的指数的和叫做这个**单项式的次数**（degree of a monomial）. 例如，在单项式 $100t$ 中，字母 t 的指数是 1，$100t$ 的次数是 1；在单项式 a^2h 中，字母 a 与 h 的指数的和是 3，a^2h 的次数是 3.

> 对于单独一个非零的数，规定它的次数为 0.

- 以上内容选自人教版七年级数学上册
- 什么，你说你不知道数和字母的定义？

单项式定义

这些式子都是数或字母的积，像这样的式子叫做**单项式**（monomial）。单独的一个数或一个字母也是单项式。

单项式中的数字因数叫做这个单项式的**系数**（coefficient）。例如，单项式 $100t$，a^2h，$-n$ 的系数分别是 100，1，-1。单项式表示数与字母相乘时，通常把数写在前面。

一个单项式中，所有字母的指数的和叫做这个**单项式的次数**（degree of a monomial）。例如，在单项式 $100t$ 中，字母 t 的指数是 1，$100t$ 的次数是 1；在单项式 a^2h 中，字母 a 与 h 的指数的和是 3，a^2h 的次数是 3。

对于单独一个非零的数，规定它的次数为 0。

- 以上内容选自人教版七年级数学上册
- 什么，你说你不知道数和字母的定义？
- [素质三连.jpg]

生成函数

假设存在一个数列$a_0, a_1, \ldots, a_n$，我们将a_i作为i次项前的系数，得到一个函数：

$f(x) = a_0 + a_1x + \ldots + a_nx^n$，称为生成函数

如果这个数列有无数项，对应的生成函数可被称为形式幂级数

生成函数

假设存在一个数列$a_0, a_1, \ldots, a_n$，我们将a_i作为i次项前的系数，得到一个函数：

$f(x) = a_0 + a_1 x + \ldots + a_n x^n$，称为生成函数

如果这个数列有无数项，对应的生成函数可被称为形式幂级数

以上内容

生成函数

假设存在一个数列$a_0, a_1, ..., a_n$，我们将a_i作为i次项前的系数，得到一个函数：

$f(x) = a_0 + a_1x + ... + a_nx^n$，称为生成函数

如果这个数列有无数项，对应的生成函数可被称为形式幂级数

以上内容纯属xbb

多项式加法和减法

多项式乘法

一种特殊的卷积

形式化的定义:

- 如果$c = a * b$:
- $c_i = \sum_{j=0}^{i} a_j * b_{i-j}$

多项式乘法

一种特殊的卷积

形式化的定义：

- 如果$c = a * b$:
- $c_i = \sum_{j=0}^{i} a_j * b_{i-j}$

直接用定义计算，枚举i,j
时间复杂度$O(N^2)$

多项式乘法

一种特殊的卷积

形式化的定义:

- 如果$c = a * b$:
- $c_i = \sum_{j=0}^{i} a_j * b_{i-j}$

直接用定义计算，枚举i,j

时间复杂度$O(N^2)$

~~可以通过$N = 266666$的数据范围~~

分治乘

接下来多项式的度数N默认为2的幂，多项式的度数定义为多项式的次数$+1$

分治乘

接下来多项式的度数N默认为2的幂，多项式的度数定义为多项式的次数+1

考虑一些更快的做法，例如分治。

假设有两个度数为N的多项式f, g，我们要计算$f * g$:

- 首先令$M = N/2$

分治乘

接下来多项式的度数N默认为2的幂，多项式的度数定义为多项式的次数+1

考虑一些更快的做法，例如分治。

假设有两个度数为N的多项式f,g，我们要计算$f*g$:

- 首先令$M=N/2$
- 我们把f分成两个多项式f_0,f_1，满足$f=f_0+f_1*x^M$
- 同理令$g=g_0+g_1*x^M$

分治乘

接下来多项式的度数N默认为2的幂，多项式的度数定义为多项式的次数+1

考虑一些更快的做法，例如分治。

假设有两个度数为N的多项式f,g，我们要计算$f*g$:

- 首先令$M=N/2$
- 我们把f分成两个多项式f_0,f_1，满足$f=f_0+f_1*x^M$
- 同理令$g=g_0+g_1*x^M$
- 那么$f*g=(f_0+f_1*x^M)*(g_0+g_1*x^M)$
- $=f_0*g_0+f_1*g_1*x^N+(f_0*g_1+f_1*g_0)*x^M$

分治乘

假设有两个度数为N的多项式f,g，我们要计算$f*g$:

- 首先令$M=N/2$
- 我们把f分成两个多项式f_0,f_1，满足$f=f_0+f_1*x^M$
- 同理令$g=g_0+g_1*x^M$
- 那么$f*g=(f_0+f_1*x^M)*(g_0+g_1*x^M)$
- $=f_0*g_0+f_1*g_1*x^N+(f_0*g_1+f_1*g_0)*x^M$

分治乘

假设有两个度数为N的多项式f,g，我们要计算$f*g$:

- 首先令$M=N/2$
- 我们把f分成两个多项式f_0,f_1，满足$f=f_0+f_1*x^M$
- 同理令$g=g_0+g_1*x^M$
- 那么$f*g=(f_0+f_1*x^M)*(g_0+g_1*x^M)$
- $=f_0*g_0+f_1*g_1*x^N+(f_0*g_1+f_1*g_0)*x^M$
- 由于$f_0*g_1+f_1*g_0=(f_0+f_1)*(g_0+g_1)-f_0*g_0-f_1*g_1$
- 只需要分治计算$f_0*g_0,f_1*g_1,(f_0+f_1)*(g_0+g_1)$即可

时间复杂度

简单分析可以发现分治乘的复杂度$T(N)$可以表示为:

$T(N) = O(N) + 3T(N/2)$

时间复杂度

简单分析可以发现分治乘的复杂度$T(N)$可以表示为:

$T(N) = O(N) + 3T(N/2)$

那么$T(N) = 3^{logN} \approx N^{1.59}$

时间复杂度

简单分析可以发现分治乘的复杂度$T(N)$可以表示为:

$T(N) = O(N) + 3T(N/2)$

那么$T(N) = 3^{logN} \approx N^{1.59}$

~~并不能通过$N = 266666$的数据范围~~

51nod马拉松29F

空间统计学

m维空间中存在n个点，满足每一维的坐标属于集合$\{0,1,2,3\}$，对任意整数$x \in [0,3m]$，求出距离为x的点对数。

$n \le 200000, m \le 9$

51nod马拉松29F
空间统计学

定义运算$a \oplus b$为将a,b每一维做差得到的点，那么假设存在两个多项式f,g，求$f \oplus g$

51nod马拉松29F
空间统计学

定义运算$a \oplus b$为将a, b每一维做差得到的点，那么假设存在两个多项式f, g，求$f \oplus g$

运用分治乘的思想，我们把f, g分成$f_0, f_1, f_2, f_3, g_0, g_1, g_2, g_3$，求出

- $(f_0 + f_1 + f_2 + f_3) \oplus (g_0 + g_1 + g_2 + g_3)$
- $(f_0 - f_1 + f_2 - f_3) \oplus (g_0 - g_1 + g_2 - g_3)$
- $(f_0 + f_3) \oplus (g_0 + g_3)$
- $(f_0 - f_3) \oplus (g_0 - g_3)$
- $f_1 \oplus g_1$，$f_2 \oplus g_2$

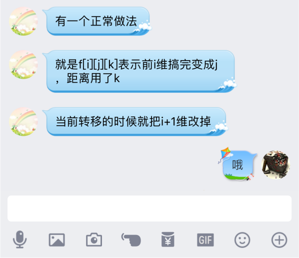
然后加加减减即可，复杂度$O(N^{\log_4 6})$

51nod马拉松29F 空间统计学

等等，这题似乎不需要这样?

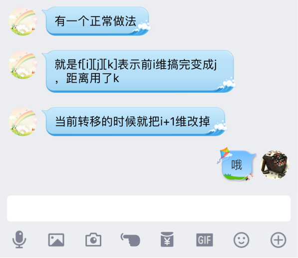
51nod马拉松29F
空间统计学

等等，这题似乎不需要这样?



51nod马拉松29F
空间统计学

等等，这题似乎不需要这样?



新的题目

不如试试 海蜇？ 海蜇！ 这道题？

新的题目

不如试试 海蜇？ 海蜇！ 这道题？

题面见simple OJ

多项式乘法

如果只需要用多项式乘法，常用离散傅里叶变换

多项式乘法

如果只需要用多项式乘法，常用离散傅里叶变换

具体可以参考《算法导论》或者电音之王逸松的WC课件

时间复杂度$O(N\log N)$

常数优化

如果我们要求两个度为N的多项式A, B相乘

一般的FFT需要3次长度为$2N$的离散傅里叶变化

常数优化

如果我们要求两个度为N的多项式A, B相乘

一般的FFT需要3次长度为$2N$的离散傅里叶变化

注意到DFT的时候虚部是零，考虑利用这一部分来搞事情

多项式乘法

令$P = A + B * i, Q = A - B * i$，可以发现$P_i, Q_i$共轭。

令ω为1的2N单位根，$\omega_i = \omega^i$，$dft(f)_i = f(\omega_i)$

另一方面，w_i, w_{2N-i}共轭

由于$\overline{a + b} = \overline{a} + \overline{b}$，$\overline{a * b} = \overline{a} * \overline{b}$

那么我们可以证明$dft(P)_i, dft(Q)_{2N-i}$共轭（逃

多项式乘法

令$P = A + B * i, Q = A - B * i$，可以发现$P_i, Q_i$共轭。

令ω为1的2N单位根，$\omega_i = \omega^i$，$dft(f)_i = f(\omega_i)$

另一方面，w_i, w_{2N-i}共轭

由于$\overline{a+b} = \overline{a} + \overline{b}$，$\overline{a*b} = \overline{a} * \overline{b}$

那么我们可以证明$dft(P)_i, dft(Q)_{2N-i}$共轭（逃

具体推导可以见$Xiao\ Mao$的国家集训队论文（手动滑稽

多项式乘法

令$P = A + B * i, Q = A - B * i$，可以发现$P_i, Q_i$共轭。

令ω为1的2N单位根，$\omega_i = \omega^i$，$dft(f)_i = f(\omega_i)$

另一方面，w_i, w_{2N-i}共轭

由于$\overline{a+b} = \overline{a} + \overline{b}$，$\overline{a*b} = \overline{a} * \overline{b}$

那么我们可以证明$dft(P)_i, dft(Q)_{2N-i}$共轭（逃

具体推导可以见*Xiao Mao*的国家集训队论文（手动滑稽

那么我们只需要计算出$dft(P)$即可，然后:

$dft(A)_i = \frac{dft(P)_i + dft(Q)_i}{2}$，$dft(B)_i = i\frac{dft(P)_i - dft(Q)_i}{2}$

注意区别单位根i和下标i

多项式乘法

上述过程给我们提供了一个将两个多项式合并在一起求dft的做法

这样我们可以把两个dft一起做，这样只需要2次长度为$2N$的dft

能不能更优呢

多项式乘法

上述过程给我们提供了一个将两个多项式合并在一起求dft的做法

这样我们可以把两个dft一起做，这样只需要2次长度为$2N$的dft

能不能更优呢

如果我们能将IDFT的过程变成两个长度为N的IDFT，那么就可以做到原来一半的常数

考虑将奇偶分开，

$$A_0 = a_0 + a_2 x + \dots + a_{N-2} x^{\frac{N}{2}-1}, A_1 = a_1 + a_3 x + \dots + a_{N-1} x^{\frac{N}{2}-1}$$

同理将B展开。

多项式乘法

$$A_0 = a_0 + a_2 x + \ldots + a_{N-2} x^{\frac{N}{2}-1}, A_1 = a_1 + a_3 x + \ldots + a_{N-1} x^{\frac{N}{2}-1}$$
$$B_0 = b_0 + b_2 x + \ldots + b_{N-2} x^{\frac{N}{2}-1}, B_1 = b_1 + b_3 x + \ldots + b_{N-1} x^{\frac{N}{2}-1}$$

多项式乘法

$$A_0 = a_0 + a_2x + \ldots + a_{N-2}x^{\frac{N}{2}-1}, A_1 = a_1 + a_3x + \ldots + a_{N-1}x^{\frac{N}{2}-1}$$
$$B_0 = b_0 + b_2x + \ldots + b_{N-2}x^{\frac{N}{2}-1}, B_1 = b_1 + b_3x + \ldots + b_{N-1}x^{\frac{N}{2}-1}$$

如果令$C = A * B$，那么可以发现

$C_0 = A_0 * B_0 + A_1 * B_1 * x$，$C_1 = A_0 * B_1 + A_1 * B_0$

所以只需要求出A_0, A_1, B_0, B_1的DFT，得出C_0, C_1的DFT然后变换回去即可。

需要4次DFT和2次IDFT，运用之前的方法只需要2次长度为N的DFT和1次长度为N的IDFT即可。常数降为原来的一半。

特殊规则的乘法

*xor*规则下的乘法

特殊规则的乘法

xor规则下的乘法

and, or规则下的乘法

codechef 2017Oct XORTREEH

定义chef期望值$C(Y)$为：

$C(Y)=\sum_{i=1}^{a}y_i^{2y_i}*p_i^{3y_i}$，$y_i$为$Y$所有可能取值，$p_i$为每种取值的概率。

给定长度为N的非负整数序列A，求下面这个值的chef期望值：

从序列A总任选X个可相同的子序列，每个序列的mex值的$K(K\le 10)$进制异或和。K进制异或即K进制不进位加法

对$330301441=2520*2^{18}+1$取模的值

codechef 2017Oct
Chef and Horcrux

可以发现题目的本质就是求K进制fwt

codechef 2017Oct
Chef and Horcrux

可以发现题目的本质就是求K进制fwt

二进制fwt的变换：$(a, b) \Rightarrow (a+b, a-b)$

三进制fwt的变换：

$$(a, b, c) \Rightarrow (a+b+c, a+b\omega+c\omega^2, a+b\omega^2+c\omega)$$

codechef 2017Oct
Chef and Horcrux

可以发现题目的本质就是求K进制fwt

二进制fwt的变换：$(a, b) \Rightarrow (a+b, a-b)$

三进制fwt的变换：

$$(a, b, c) \Rightarrow (a+b+c, a+b\omega+c\omega^2, a+b\omega^2+c\omega)$$

可以发现fwt变换的本质就是对每一维进行一次长度为K的DFT。然后我们发现2520恰好整除330301440，因此仿照二进制fwt的变换对每一维进行一次暴力DFT即可。

时间复杂度$O(NKlog_K N)$

多项式求逆

给定一个度为N的多项式$f(x)$，

求$g(x)$，满足$f(x) * g(x) = 1 \pmod{x^N}$

多项式求逆

给定一个度为N的多项式$f(x)$,

求$g(x)$，满足$f(x) * g(x) = 1 \ (\ mod\ x^N\ )$

考虑倍增，令$M = N/2$，$f_0(x)$为$f(x)$的前M位,

假设现在已知$g0(x)$满足$f_0(x) * g_0(x) = 1 \ (\ mod\ x^M\ )$，那么

- $(f_0(x) * g_0(x))^2 = 1 \ (\ mod\ x^N\ )$
- $(f(x) - f_0(x))^2 = 0 \ (\ mod\ x^N\ )$

多项式求逆

给定一个度为N的多项式$f(x)$，

求$g(x)$，满足$f(x) * g(x) = 1 \ (\ mod \ x^N \ )$

考虑倍增，令$M = N/2$，$f_0(x)$为$f(x)$的前M位，

假设现在已知$g0(x)$满足$f_0(x) * g_0(x) = 1 \ (\ mod \ x^M \ )$，那么

- $(f_0(x) * g_0(x))^2 = 1 \ (\ mod \ x^N \ )$
- $(f(x) - f_0(x))^2 = 0 \ (\ mod \ x^N \ )$
- 展开得$f^2(x) - 2f(x)f_0(x) + f_0^2(x) = 0 \ (\ mod \ x^N \ )$

多项式求逆

给定一个度为N的多项式$f(x)$，
求$g(x)$，满足$f(x)*g(x)=1\ (\ mod\ x^N\ )$

考虑倍增，令$M=N/2$，$f_0(x)$为$f(x)$的前M位，
假设现在已知$g0(x)$满足$f_0(x)*g_0(x)=1\ (\ mod\ x^M\ )$，那么

- $(f_0(x)*g_0(x))^2=1\ (\ mod\ x^N\ )$
- $(f(x)-f_0(x))^2=0\ (\ mod\ x^N\ )$
- 展开得$f^2(x)-2f(x)f_0(x)+f_0^2(x)=0\ (\ mod\ x^N\ )$
- 同乘$g(x)$得到，$f(x)-2f_0(x)+f_0^2(x)g(x)=0\ (\ mod\ x^N\ )$

多项式求逆

给定一个度为N的多项式$f(x)$，
求$g(x)$，满足$f(x) * g(x) = 1 \ (\ mod\ x^N\ )$

考虑倍增，令$M = N/2$，$f_0(x)$为$f(x)$的前M位，
假设现在已知$g0(x)$满足$f_0(x) * g_0(x) = 1 \ (\ mod\ x^M\ )$，那么

- $(f_0(x) * g_0(x))^2 = 1 \ (\ mod\ x^N\ )$
- $(f(x) - f_0(x))^2 = 0 \ (\ mod\ x^N\ )$
- 展开得$f^2(x) - 2f(x)f_0(x) + f_0^2(x) = 0 \ (\ mod\ x^N\ )$
- 同乘$g(x)$得到，$f(x) - 2f_0(x) + f_0^2(x)g(x) = 0 \ (\ mod\ x^N\ )$
- 乘上$g_0^2(x)$，移项可得：
- $g(x) = (2 - f(x) * g_0(x)) * g_0(x)$

多项式求逆

给定一个度为N的多项式$f(x)$，
求$g(x)$，满足$f(x) * g(x) = 1 \ (\ mod\ x^N\ )$

考虑倍增，令$M = N/2$，$f_0(x)$为$f(x)$的前M位，
假设现在已知$g0(x)$满足$f_0(x) * g_0(x) = 1 \ (\ mod\ x^M\ )$，那么

- $(f_0(x) * g_0(x))^2 = 1 \ (\ mod\ x^N\ )$
- $(f(x) - f_0(x))^2 = 0 \ (\ mod\ x^N\ )$
- 展开得$f^2(x) - 2f(x)f_0(x) + f_0^2(x) = 0 \ (\ mod\ x^N\ )$
- 同乘$g(x)$得到，$f(x) - 2f_0(x) + f_0^2(x)g(x) = 0 \ (\ mod\ x^N\ )$
- 乘上$g_0^2(x)$，移项可得:
- $g(x) = (2 - f(x) * g_0(x)) * g_0(x)$
- 时间复杂度$O(NlogN)$

多项式除法

给定两个度数为n, m的多项式$P(x), Q(x)$,

求$P(x) = Q(x) * H(x) + R(x)$满足$R(x)$的度$\leq m$

多项式除法

给定两个度数为n, m的多项式$P(x), Q(x)$，
求$P(x) = Q(x) * H(x) + R(x)$满足$R(x)$的度$\le m$

为了方便，我们假设$R(x)$的度为m。
首先定义$f^R(x)$为将$f(x)$系数反转得到的多项式

那么我们可以发现$P^R(x) = Q^R(x) * H^R(x) + R^R(x) * x^{n-m+1}$
也就是说$P^R(x) = Q^R(x) * H^R(x) \pmod{x^{n-m+1}}$

多项式除法

给定两个度数为n, m的多项式$P(x), Q(x)$，
求$P(x) = Q(x) * H(x) + R(x)$满足$R(x)$的度$\le m$

为了方便，我们假设$R(x)$的度为m。
首先定义$f^R(x)$为将$f(x)$系数反转得到的多项式

那么我们可以发现$P^R(x) = Q^R(x) * H^R(x) + R^R(x) * x^{n-m+1}$
也就是说$P^R(x) = Q^R(x) * H^R(x) \pmod{x^{n-m+1}}$

多项式求逆得出$H(x)$后，$R(x)$也随之得到了
时间复杂度$O(NlogN)$

多项式求以e为底的对数

求$g(x) = ln(f(x))$

多项式求以e为底的对数

求$g(x) = ln(f(x))$

由于$ln(f(x))' = \frac{f'(x)}{f(x)}$

多项式求以e为底的对数

求$g(x) = ln(f(x))$

由于$ln(f(x))' = \frac{f'(x)}{f(x)}$

只需要一次多项式求逆即可

多项式求以e为底的对数

求$g(x) = ln(f(x))$

由于$ln(f(x))' = \frac{f'(x)}{f(x)}$

只需要一次多项式求逆即可

为了满足某些人的好奇心，给出一个简单的证明（默认$\lim_{\Delta x \to 0}$）：

$$ln(f(x))' = \frac{ln(f(x+\Delta x)) - ln(f(x))}{(x+\Delta x) - x} = \frac{ln(\frac{f(x+\Delta x)}{f(x)})}{\Delta x} = \frac{ln(\frac{f'(x)\Delta x + f(x)}{f(x)})}{\Delta x} = \frac{ln(\frac{f'(x)\Delta x}{f(x)} + 1)}{\Delta x} = \frac{f'(x)}{f(x)} * \frac{f(x)}{f'(x)\Delta x} ln(\frac{f'(x)\Delta x}{f(x)} + 1) = \frac{f'(x)}{f(x)} * ln((\frac{f'(x)\Delta x}{f(x)} + 1)^{\frac{f(x)}{f'(x)\Delta x}}) = \frac{f'(x)}{f(x)} * ln(e) = \frac{f'(x)}{f(x)}$$

多项式求以e为底的幂

求$g(x) = e^{f(x)}$

多项式求以e为底的幂

求$g(x)=e^{f(x)}$

$(e^{f(x)})'=f'(x)e^{f(x)}$，分治fft可以解决

时间复杂度$O(Nlog^2N)$

有了求对数和求幂，就能很方便地求一个多项式的若干次幂

多项式求以e为底的幂

求$g(x) = e^{f(x)}$

$(e^{f(x)})' = f'(x)e^{f(x)}$，分治fft可以解决

时间复杂度$O(Nlog^2N)$

有了求对数和求幂，就能很方便地求一个多项式的若干次幂

等等，为什么我学的以e为底的幂是时间复杂度$O(NlogN)$的呢?

多项式求以e为底的幂

使用牛顿迭代，可以做到$O(N\log N)$的时间复杂度。

多项式求以e为底的幂

使用牛顿迭代，可以做到$O(NlogN)$的时间复杂度。

同样假设我们求出$g_0(x) = e^{f_0(x)} \pmod{x^{N/2}}$

多项式求以e为底的幂

使用牛顿迭代，可以做到$O(NlogN)$的时间复杂度。

同样假设我们求出$g_0(x)=e^{f_0(x)} \pmod{x^{N/2}}$

对$e^{f(x)}$泰勒展开得到：

$$e^{f(x)}=\frac{e^{f_0(x)}}{0!}+\frac{e^{f_0(x)}(f(x)-f_0(x))}{1!}+\frac{e^{f_0(x)}(f(x)-f_0(x))^2}{2!}+\dots$$

显然从第三项开始，$(f(x)-f_0(x))^k$在模x^N意义下都为0，因此可以忽略

从而得到$g(x)=g_0(x)(1+f(x)-f_0(x))$

多项式求以e为底的幂

使用牛顿迭代，可以做到$O(NlogN)$的时间复杂度。

同样假设我们求出$g_0(x) = e^{f_0(x)} \pmod{x^{N/2}}$

对$e^{f(x)}$泰勒展开得到：

$$e^{f(x)} = \frac{e^{f_0(x)}}{0!} + \frac{e^{f_0(x)}(f(x)-f_0(x))}{1!} + \frac{e^{f_0(x)}(f(x)-f_0(x))^2}{2!} + \dots$$

显然从第三项开始，$(f(x) - f_0(x))^k$在模x^N意义下都为0，因此可以忽略

从而得到$g(x) = g_0(x)(1 + f(x) - f_0(x))$

时间复杂度$O(NlogN)$

多项式求以e为底的幂

听起来很有道理，可是为什么我wa了？

多项式求以e为底的幂

听起来很有道理，可是为什么我wa了？

$$e^{f(x)} = \frac{e^{f_0(x)}}{0!} + \frac{e^{f_0(x)}(f(x)-f_0(x))}{1!}$$

$$\Rightarrow g(x) = g_0(x)(1 + f(x) - f_0(x))$$

多项式求以e为底的幂

听起来很有道理，可是为什么我wa了？

$$e^{f(x)} = \frac{e^{f_0(x)}}{0!} + \frac{e^{f_0(x)}(f(x)-f_0(x))}{1!}$$

$$\Rightarrow g(x) = g_0(x)(1 + f(x) - f_0(x))$$

前半部分是没有问题的，但是需要注意在 $mod\ x^N$意义下，$g_0(x) = e^{f_0(x)}$并不成立

多项式求以e为底的幂

听起来很有道理，可是为什么我wa了?

$$e^{f(x)} = \frac{e^{f_0(x)}}{0!} + \frac{e^{f_0(x)}(f(x)-f_0(x))}{1!}$$

$$\Rightarrow g(x) = g_0(x)(1+f(x)-f_0(x))$$

前半部分是没有问题的，但是需要注意在 $mod\ x^N$ 意义下，$g_0(x) = e^{f_0(x)}$ 并不成立

注意到求$ln(f(x))$是十分容易的，而$ln(g_(x))$和$f_0(x)$在模$x^{\frac{N}{2}}$意义下显然相等。因此我们用$ln(g_0(x))$ 替换$f_0(x)$ 即可。

$$g(x) = g_0(x)(1+f(x)-ln(g_0(x)))$$

多项式开根

直接用求$ln(f(x))$和$e^{f(x)}$即可。

多项式开根

直接用求$ln(f(x))$和$e^{f(x)}$即可。

或者使用牛顿迭代，常数较小：

已知$f_0(x)=g_0^2(x) \pmod{x^{N/2}}$，求$f(x)=g^2(x) \pmod{x^N}$

同样，$f_1(x)=g_0^2(x) \pmod{x^N}$

多项式开根

直接用求$ln(f(x))$和$e^{f(x)}$即可。

或者使用牛顿迭代，常数较小：

已知$f_0(x)=g_0^2(x) \pmod{x^{N/2}}$，求$f(x)=g^2(x) \pmod{x^N}$

同样，$f_1(x)=g_0^2(x) \pmod{x^N}$

$$\sqrt{f_0(x)}=\sqrt{f_1(x)}+\frac{1}{2\sqrt{f_1(x)}}(f(x)-f_1(x))$$

$$g(x)=g_0(x)+\frac{f(x)-f_1(x)}{2g_0(x)}=g_0(x)+\frac{f(x)-g_0^2(x)}{2g_0(x)}=\frac{f(x)}{2g_0(x)}+\frac{g_0(x)}{2}$$

多项式多点求值

给定一个度为N的多项式$f(x)$和N个数$a_1, \ldots, a_n$，求$f(a_1), \ldots, f(a_n)$

多项式多点求值

给定一个度为N的多项式$f(x)$和N个数$a_1, ..., a_n$，求$f(a_1), ..., f(a_n)$

考虑分治，将$f(x)$写成$f(x) = g(x)\Pi_{i=1}^{N/2}(x - a_i) + r(x)$，那么我们递归对$a_1, ..., a_{N/2}$做在$r(x)$上的多点求值；对$a_{N/2+1}, ..., a_N$也类似。

只需要一次多项式除法

时间复杂度$T(N) = O(NlogN) + 2T(N/2) = O(Nlog^2N)$

多项式插值

给定N对数(a_i, b_i)表示$f(a_i) = b_i$，求一个度为N的多项式$f(x)$满足条件。

多项式插值

给定N对数(a_i, b_i)表示$f(a_i) = b_i$，求一个度为N的多项式$f(x)$满足条件。

我们模仿上面的过程，假设：

$f(x) = g(x)\Pi_{i=1}^{N/2}(x - a_i) + h(x)\Pi_{i=N/2+1}^{N}(x - a_i)$，

我们只需要对多项式$\Pi_{i=1}^{N/2}(x - a_i)$和$a_{N/2+1}, \dots, a_N$多点求值，然后更新对应的b数组，然后递归进行。递归完之后再合并

时间复杂度$T(N) = O(Nlog^2N) + 2T(N/2) = O(Nlog^3N)$

多项式插值

给定N对数(a_i,b_i)表示$f(a_i)=b_i$，求一个度为N的多项式$f(x)$满足条件。

我们模仿上面的过程，假设：

$f(x)=g(x)\Pi_{i=1}^{N/2}(x-a_i)+h(x)\Pi_{i=N/2+1}^{N}(x-a_i)$，

我们只需要对多项式$\Pi_{i=1}^{N/2}(x-a_i)$和$a_{N/2+1},\ldots,a_N$多点求值，然后更新对应的b数组，然后递归进行。递归完之后再合并

时间复杂度$T(N)=O(Nlog^2N)+2T(N/2)=O(Nlog^3N)$

什么？你说3个log太慢了？

多项式插值

给定N对数(a_i, b_i)表示$f(a_i) = b_i$，求一个度为N的多项式$f(x)$满足条件。

我们模仿上面的过程，假设：

$f(x) = g(x)\Pi_{i=1}^{N/2}(x - a_i) + h(x)\Pi_{i=N/2+1}^{N}(x - a_i)$，

我们只需要对多项式$\Pi_{i=1}^{N/2}(x - a_i)$和$a_{N/2+1}, \dots, a_N$多点求值，然后更新对应的b数组，然后递归进行。递归完之后再合并

时间复杂度$T(N) = O(Nlog^2N) + 2T(N/2) = O(Nlog^3N)$

什么？你说3个log太慢了？

别急还有操作（手动滑稽

多项式插值

给定N对数(a_i, b_i)表示$f(a_i) = b_i$，求一个度为N的多项式$f(x)$满足条件。

多项式插值

给定N对数(a_i, b_i)表示$f(a_i) = b_i$，求一个度为N的多项式$f(x)$满足条件。

仔细分析刚才的过程，对比拉格朗日插值：

多项式插值

给定N对数(a_i, b_i)表示$f(a_i) = b_i$，求一个度为N的多项式$f(x)$满足条件。

仔细分析刚才的过程，对比拉格朗日插值：

$$f(x) = \sum_{i=1}^{N} \frac{b_i}{\prod_{j=1,j\neq i}^{N}(a_i-a_j)} * \prod_{j=1,j\neq i}^{N}(x - a_j)$$

我们发现第一部分的递归就是求出所有的$c_i = \frac{b_i}{\prod_{j=1,j\neq i}^{N}(a_i-a_j)}$

第二部分就是通过归并将所有的式子合并

而第二部分的复杂度是$O(N\log^2 N)$的，因此考虑如何快速求出所有的c_i

多项式插值

$$c_i = \frac{b_i}{\prod_{j=1,j\neq i}^{N}(a_i - a_j)} \Rightarrow \frac{b_i}{c_i} = \prod_{j=1,j\neq i}^{N}(a_i - a_j)$$

多项式插值

$$c_i = \frac{b_i}{\prod_{j=1,j\neq i}^{N}(a_i-a_j)} \Rightarrow \frac{b_i}{c_i} = \prod_{j=1,j\neq i}^{N}(a_i - a_j)$$

洛必达法则（忽略一些条件）：

若$\lim_{x\to a} f(x) = 0$，$\lim_{x\to a} g(x) = 0$，那么
$\frac{\lim_{x\to a} f(x)}{\lim_{x\to a} g(x)} = \frac{\lim_{x\to a} f'(x)}{\lim_{x\to a} g'(x)}$

多项式插值

$$c_i = \frac{b_i}{\prod_{j=1,j\neq i}^{N}(a_i - a_j)} \Rightarrow \frac{b_i}{c_i} = \prod_{j=1,j\neq i}^{N}(a_i - a_j)$$

洛必达法则（忽略一些条件）：

若$\lim_{x\to a} f(x) = 0$，$\lim_{x\to a} g(x) = 0$，那么 $\frac{\lim_{x\to a} f(x)}{\lim_{x\to a} g(x)} = \frac{\lim_{x\to a} f'(x)}{\lim_{x\to a} g'(x)}$

令$h(x) = \prod_{i=1}^{N}(x - a_i)$，

考虑$\frac{b_i}{c_i} = \frac{h(a_i)}{x - a_i}$，根据洛必达法则我们可以得到$\frac{b_i}{c_i} = \frac{h'(a_i)}{(x-a_i)'} = h'(a_i)$，运用多点求值即可。

这个式子也可以通过将$h(x)$的导数展开得到。

这一部分和总的时间复杂度都是$O(N\log^2 N)$

与组合有关的卷积

考虑$a_i = \sum_{j=0}^{i} \binom{i}{j} b_j$

与组合有关的卷积

考虑$a_i = \sum_{j=0}^{i} \binom{i}{j} b_j$

我们把组合数展开移项得到：

$\frac{a_i}{i!} = \sum_{j=0}^{i} \frac{b_j}{j!} * \frac{1}{(i-j)!}$

转化为简单的卷积

关于自身的递推式

考虑$f_i = \sum_{j=0}^{i-1} f_j * a_{i-j}$，$f_0 = 1$

关于自身的递推式

考虑$f_i = \sum_{j=0}^{i-1} f_j * a_{i-j}$，$f_0 = 1$

将f, a作为形式幂级数，那么就有$f(x) = f(x) * a(x) + 1$

移项得到$f(x) = \frac{1}{1-a(x)}$，多项式求逆就能得到$f(x)$

卡特兰数

卡特兰数的一种组合意义是节点数为N的二叉树个数，那么我们可以列出如下递推式：

$f_i = \sum_{j=0}^{i-1} f_j * f_{i-1-j}$，$f_0 = 1$

卡特兰数

卡特兰数的一种组合意义是节点数为N的二叉树个数，那么我们可以列出如下递推式：

$f_i = \sum_{j=0}^{i-1} f_j * f_{i-1-j}$，$f_0 = 1$

同样我们可以列出一个方程式$f(x) = xf^2(x) + 1$

解方程得到$f(x) = \frac{1 \pm \sqrt{1-4x}}{2x}$

由于$f_0 = f(0) = 1$，我们直接在上面的方程将$x = 0$带入，就能发现$f(x) = \frac{1-\sqrt{1-4x}}{2x}$

运用多项式开根号和~~多项式求逆~~ ~~单项式求逆~~ 求啥求直接除即可得到$f(x)$

线性递推式

若$f_i = \sum_{j=1}^{k} f_{i-j} * a_j$，求$f_N$。

$N \le 10^{18}, k \le 10^5$

一道初中奥数好题

已知a是方程$6x^5 - 3x^2 + x - 4 = 0$的一个根，

求$a^{2333} + a^{233} + 1$的值

一道初中奥数好题

已知a是方程$6x^5-3x^2+x-4=0$的一个根，
求$a^{2333}+a^{233}+1$的值

- ~~方法零：观察发现$x=1$是一组解，带入可得答案~~

一道初中奥数好题

已知a是方程$6x^5-3x^2+x-4=0$的一个根，求$a^{2333}+a^{233}+1$的值

- ~~方法零：观察发现$x=1$是一组解，带入可得答案~~
- 方法一：移项得到$x^5=\frac{3x^2-x+4}{6}$，降幂

一道初中奥数好题

已知a是方程$6x^5-3x^2+x-4=0$的一个根，

求$a^{2333}+a^{233}+1$的值

- ~~方法零：观察发现$x=1$是一组解，带入可得答案~~
- 方法一：移项得到$x^5=\frac{3x^2-x+4}{6}$，降幂
- 方法二：直接大除法

线性递推式

若$f_i = \sum_{j=1}^{k} f_{i-j} * a_j$，求$f_N$。

$N \le 10^{18}, k \le 10^5$

线性递推式

若$f_i = \sum_{j=1}^{k} f_{i-j} * a_j$，求$f_N$。
$N \le 10^{18}, k \le 10^5$

我们发现线性递推式类似于降幂，而大除法和多项式除法也十分类似。

那么我们直接求x^N对$x^k - \sum_{j=1}^{k} a_j x^{k-j}$取模的结果即可。

由于N很大，我们需要用快速幂。

时间复杂度$O(Nlog^2N)$

求第二类斯特林数

第二类斯特林数s_n^k表示将n个不同的球放入k个相同盒子且盒子非空的方案数。

求第二类斯特林数

第二类斯特林数s_n^k表示将n个不同的球放入k个相同盒子且盒子非空的方案数。

对于s_n^k，我们可以用容斥和组合数进行表示。
枚举至多有x个非空的盒子，我们可以得到：
$s_n^k = \sum_{x=0}^k \binom{k}{x}(-1)^{k-x}x^n$

求第二类斯特林数

第二类斯特林数s_n^k表示将n个不同的球放入k个相同盒子且盒子非空的方案数。

对于s_n^k，我们可以用容斥和组合数进行表示。
枚举至多有x个非空的盒子，我们可以得到：
$s_n^k = \sum_{x=0}^{k} \binom{k}{x}(-1)^{k-x}x^n$

我们模仿求组合数的方法，展开得到：
$\frac{s_n^k}{k!} = \sum_{x=0}^{k} \frac{x^n}{x!} * \frac{(-1)^{k-x}}{(k-x)!}$
那么在n一定的情况下我们可以在$O(NlogN)$的时间内得到所有的s_n^k

一种特殊fmt

$$c_i = \sum_{j=0}^{N} \sum_{k=0}^{N} [j + k = i \ \&\& \ j | k = i] \, a_j * b_k$$

一种特殊fmt

$$c_i = \sum_{j=0}^{N} \sum_{k=0}^{N} [j + k = i \;\&\&\; j|k = i]\, a_j * b_k$$

定义$bit(x)$表示x的二进制表示中1的个数

构造新的多项式$A(x) = \sum_{i=0}^{N} (a_i p^{bit(i)}) x^i$，$B(x)$同理

然后对$A(x), B(x)$进行fmt，得到$C(x)$

那么$c_i = [p^{bit(i)}] C_i$

时间复杂度$O(N \log^2 N)$

bzoj3684

大朋友与多叉树

给出一个集合D，多叉树的每个非叶子结点需要满足孩子个数$\in D$

求叶子结点个数为s的满足条件的多叉树的个数，对一个ntt模数取模的值。

$1 \le max\{d_i\}, s \le 10^5$

bzoj3684

大朋友与多叉树

给出一个集合D，多叉树的每个非叶子结点需要满足孩子个数$\in D$

求叶子结点个数为s的满足条件的多叉树的个数，对一个ntt模数取模的值。

$1 \leq max\{d_i\}, s \leq 10^5$

前置技能 拉格朗日反演：

若$G(F(x)) = x$，那么$[x^n]F(x) = \frac{x^{n-1}(\frac{x}{G(x)})^n}{n}$

bzoj3684

大朋友与多叉树

直接写出生成函数$F(x) = x + \sum_{i \in D} F(x)^i$，

然后$\Rightarrow F(x) - \sum_{i \in D} F(x)^i = x$

bzoj3684
大朋友与多叉树

直接写出生成函数$F(x) = x + \sum_{i \in D} F(x)^i$，

然后$\Rightarrow F(x) - \sum_{i \in D} F(x)^i = x$

考虑函数$G(x) = x - \sum_{i \in D} x^i$，显然$G(F(x)) = x$

剩下就是一个多项式求幂了

bzoj4555
TJOI/HEOI2016 求和

给定$N \le 10^5$，求$\sum_{i=0}^{N}\sum_{j=0}^{i} S_i^j * 2^j * (j!)$

bzoj4555
TJOI/HEOI2016 求和

给定$N \le 10^5$，求$\sum_{i=0}^{N} \sum_{j=0}^{i} S_i^j * 2^j * (j!)$

考虑$S_i^j * 2^j * (j!)$的含义，相当于i个球放入j个不同的盒子，并且给每个盒子涂上红色或者蓝色中的一种的方案数

bzoj4555 TJOI/HEOI2016 求和

给定$N \le 10^5$，求$\sum_{i=0}^{N}\sum_{j=0}^{i} S_i^j * 2^j * (j!)$

考虑$S_i^j * 2^j * (j!)$的含义，相当于i个球放入j个不同的盒子，并且给每个盒子涂上红色或者蓝色中的一种的方案数

如果定义$F_i = \sum_{j=0}^{i} S_i^j * 2^j * (j!)$，考虑枚举第一个盒子所含球的个数和颜色：

$F_i = 2\sum_{j=1}^{i}\binom{i}{j}F_{i-j}$，运用之前组合数卷积和关于自身递推式的技巧就能得到$F(x)$

时间复杂度$O(NlogN)$

AGC019 E
Shuffle and Swap

给定两个长度为N的包含个数相同的1的01串A, B，进行下述操作：

- 令$a_1, \ldots, a_k$为1在A中的位置，$b_1, .., b_k$为1在B中的位置
- 随机打乱序列a, b
- 按顺序从1到k交换A_{a_i}, A_{b_i}

AGC019 E
Shuffle and Swap

给定两个长度为N的包含个数相同的1的01串A, B，进行下述操作：

- 令$a_1, \dots, a_k$为1在A中的位置，$b_1, .., b_k$为1在B中的位置
- 随机打乱序列a, b
- 按顺序从1到k交换A_{a_i}, A_{b_i}

令P为操作结束后A, B相同的概率，求$P * (k!)^2$的在模998244353意义下的期望。

$1 \leq N \leq 10^5$

AGC019 E
Shuffle and Swap

某一位置上A, B对应的值有$11, 10, 01$三种

我们把形如$(b, a) \to (c, b) \ldots \to (z, y)$的操作作为一个块（注意$z$可能和$a$ 相同）。

那么每一块的$a, \ldots, z$一定是$10, 11, 11, \ldots, 01 (z \neq a)$或者$11, \ldots, 11 (z = a)$

AGC019 E
Shuffle and Swap

某一位置上A,B对应的值有$11,10,01$三种

我们把形如$(b,a)\to(c,b)\ldots\to(z,y)$的操作作为一个块（注意$z$可能和$a$ 相同）。

那么每一块的$a,\ldots,z$一定是$10,11,11,\ldots,01(z\neq a)$或者$11,\ldots,11(z=a)$

考虑k个$z\neq a$的块，枚举这些块中11的个数x。那么01和10配对的方法有$k!$种，将x个11 分配的方法有k^x种，总的顺序有$\frac{(2k+x)!}{\prod_{i=1}^{k}(s_i+2)!}$种，其中$s_i$表示第$i$块内的$11$的个数，这一部分总方案数：

$$k!k^x\frac{(2k+x)!}{\prod_{i=1}^{k}(s_i+2)!}$$

AGC019 E
Shuffle and Swap

第一部分方案数：$k!k^x\frac{(2k+x)!}{\prod_{i=1}^{k}(s_i+2)!}$，其中$k$表示01的个数，$x$表示$z\neq a$ 块中11的个数。下面令m表示11的总个数，

显然$z=a$块中11可以随意排列，方案数为$((m-x)!)^2$。

AGC019 E

Shuffle and Swap

第一部分方案数：$k!k^x\frac{(2k+x)!}{\prod_{i=1}^{k}(s_i+2)!}$，其中$k$表示01的个数，$x$表示$z \neq a$ 块中11的个数。下面令m表示11的总个数，

显然$z = a$块中11可以随意排列，方案数为$((m-x)!)^2$。

将两个组合起来的方案数为$\frac{N!}{(2k+x)!(m-x)!}$

AGC019 E
Shuffle and Swap

第一部分方案数：$k!k^x\frac{(2k+x)!}{\prod_{i=1}^{k}(s_i+2)!}$，其中$k$表示01的个数，$x$表示$z \neq a$ 块中11的个数。下面令m表示11的总个数，

显然$z=a$块中11可以随意排列，方案数为$((m-x)!)^2$。

将两个组合起来的方案数为$\frac{N!}{(2k+x)!(m-x)!}$

那么总的方案数就是：

$$(k!)^2k^x\frac{(2k+x)!}{\prod_{i=1}^{k}(s_i+2)!}*((m-x)!)^2*\frac{N!}{(2k+x)!(m-x)!}$$
$$=\frac{N!k!k^x(m-x)!}{\prod_{i=1}^{k}(s_i+2)!}，\text{其中}\sum s_i=x$$

构造多项式$F(x)=\sum_{i=0}\frac{1}{(i+2)!}$，求出$F^k(x)$即可。

时间复杂度$O(NlogN)$，当然一般用$O(Nlog^2N)$的快速幂即可。

codechef 2017Nov Day Schedule

大厨一年有K个小时，第一个小时吃早饭，吃L个菜（L每次询问给出）。接下来每个小时，他可以选择吃简餐或者做事情。但是同一天内不能连着吃两顿饭。大厨共有A件事情可以做。Q次询问：

假设第一天餐馆有D种不同的菜肴供大厨选择，接下来每天都会推出一道新材。请你求出从第一天起的T天一共有多少种不同的安排，即：第一天方案数+第二天方案数+...+第T天的方案数，对质数P取模。

$$K \le 10^5, A \le 10^9, 10^8 + 7 \le P \le 10^9 + 7,$$
$$Q \le 500, L \le D \le D + T - 1 \le 10^7$$

codechef 2017Nov Day Schedule

首先我们用$F(D)$表示有D种不同的菜肴选择的方案数，那么答案就是$\sum_{i=D}^{D+T-1} F(i)$

而$F(D)$可以简单通过矩阵乘法得到，时间复杂度$O(Q(D+T)logK)$

codechef 2017Nov Day Schedule

首先我们用$F(D)$表示有D种不同的菜肴选择的方案数，那么答案就是$\sum_{i=D}^{D+T-1} F(i)$

而$F(D)$可以简单通过矩阵乘法得到，时间复杂度$O(Q(D+T)logK)$

我们发现$F(D)$是关于D的至多$\lfloor \frac{K}{2} \rfloor$次多项式，而且这个多项式是不会随着询问参数改变的。考虑枚举简餐个数：

$F(D) = \sum_i f_i * D^i = L * \sum_i \binom{K-2-i}{i} * A^{K-1-i} * D^i$

这样我们把f_i移到前面求和：

$\sum_{i=D}^{D+T-1} F(i) = \sum_j f_j \sum_{i=D}^{D+T-1} i^j$

然而$\sum_{i=0}^{n} i^k$并不太容易求和，我们考虑变换多项式的形式

codechef 2017Nov Day Schedule

我们发现$\sum_{i=0}^{n}(i+1)(i+2)...(i+k)$比较容易通过组合数上指标求和的方法快速求出。

$$\sum_{i=0}^{n}(i+1)...(i+k) = (k!) * \sum_{i=0}^{n}\binom{i+k}{k} = (k!) * \binom{i+n+1}{k+1}$$

那么假设我们能求出一个多项式形式如下：

$$P(D) = \sum_i p_i(D+1)...(D+i)$$

那么就可以比较容易地在$O(K)$时间复杂度求出答案

注意到这个多项式的点值可以在$O(logN)$的时间复杂度求出，我们考虑通过点值来得到这个多项式

codechef 2017Nov
Day Schedule

假设我们求出了这个多项式的前k项，我们现在考虑求p_k。

我们考虑将$D=-k-1$带入$P(D)$，那么从x^{k+1}开始都被消掉了，那么就有：

$P(-k-1)=(-1)^k*(k!)*p_k+\sum_{i=0}^{k-1}(-1)^i*\frac{k!}{(k-i)!}*p_i$

codechef 2017Nov Day Schedule

假设我们求出了这个多项式的前k项，我们现在考虑求p_k。

我们考虑将$D=-k-1$带入$P(D)$，那么从x^{k+1}开始都被消掉了，那么就有：

$$P(-k-1)=(-1)^k*(k!)*p_k+\sum_{i=0}^{k-1}(-1)^i*\frac{k!}{(k-i)!}*p_i$$

构造一个多项式$Q(x)$满足$q_k=\frac{P(-k-1)}{k!}$，那么就有：

$$q_k=(-1)^k*p_k+\sum_{i=0}^{k-1}(-1)^ip_i*\frac{1}{(k-i)!}$$

codechef 2017Nov Day Schedule

假设我们求出了这个多项式的前k项，我们现在考虑求p_k。

我们考虑将$D=-k-1$带入$P(D)$，那么从x^{k+1}开始都被消掉了，那么就有：

$$P(-k-1)=(-1)^k*(k!)*p_k+\sum_{i=0}^{k-1}(-1)^i*\frac{k!}{(k-i)!}*p_i$$

构造一个多项式$Q(x)$满足$q_k=\frac{P(-k-1)}{k!}$，那么就有：

$$q_k=(-1)^k*p_k+\sum_{i=0}^{k-1}(-1)^i p_i*\frac{1}{(k-i)!}$$

我们调整$P(D)$的系数：$p_i\Rightarrow(-1)^i p_i$

写成形式幂级数形式就有$Q(x)=P(x)+H(x)*P(x)$，多项式求逆即可。

由于模数不是ntt模数，需要多次ntt或者分治乘

一个优化

$$Q(x) = P(x) + H(x) * P(x) \Rightarrow P(x) = \frac{Q(x)}{1+P(x)}$$

一个优化

$$Q(x) = P(x) + H(x) * P(x) \Rightarrow P(x) = \frac{Q(x)}{1+P(x)}$$

$$1 + P(x) = \sum_{i=0} \frac{x^i}{i!} = e^x$$

一个优化

$$Q(x) = P(x) + H(x) * P(x) \Rightarrow P(x) = \frac{Q(x)}{1+P(x)}$$

$$1 + P(x) = \sum_{i=0} \frac{x^i}{i!} = e^x$$

从而$P(x) = Q(x) * e^{-x}$，
而$e^{-x} = \sum_{i=0} \frac{(-1)^i}{i!} x^i$
只需要一次多项式乘法

More

- uoj182 $a^{-1}+b$ problem 多项式多点求值
- uoj50 链式反应 牛顿迭代法/分治fft
- bzoj4589 Hard Nim fwt
- NOI2017 Day2T3 多项式取模求线性递推式
- codeforces# 250 小朋友与二叉树 多项式开根号
- 51nod算法马拉松16 F 不动点 牛顿迭代法/分治fft

感谢

感谢甡力的多项式课件

感谢毛(Xiao)爷(Mao)爷的集训队论文

感谢CC

感谢

感谢蔊力的多项式课件

感谢毛(Xiao)爷(Mao)爷的集训队论文

感谢CCF

感谢

感谢鞋力的多项式课件

感谢毛(Xiao)爷(Mao)爷的集训队论文

感谢CCF

The End