计数2

卷积

- 卷积的形式一般为$c[n] = \sum a[i] * b[j] (i \text{ op } j = n)$
- op表示一种运算例如max,xor,+等
- 一般暴力计算的时间复杂度为n^2，但是我们可以通过某些变换把卷积变换成点积，这样计算就比较方便。

例题

给定 n 个 m 维向量，每一维的取值只能是 1 或 2 或 3 或 4。接下来有 q 个询问，每个询问都是这四种询问中的一种：

（1）给定一个向量 C，问有多少种方法从这 n 个向量中选出 2 个向量 A 和 B，使得对于每一维 i(1 ≤i ≤ m)，A i 和 B i 的最大值小于等于 C i 。

（2）给定一个向量 C，问有多少种方法从这 n 个向量中选出 2 个向量 A 和 B，使得对于每一维 i(1 ≤i ≤ m)，A i 和 B i 的最小值大于等于 C i 。

（3）给定一个向量 C，问有多少种方法从这 n 个向量中选出 2 个向量 A 和 B，使得对于每一维 i(1 ≤i ≤ m)，A i 和 B i 的最大值大于等于 C i 。

（4）给定一个向量 C，问有多少种方法从这 n 个向量中选出 2 个向量 A 和 B，使得对于每一维 i(1 ≤i ≤ m)，A i 和 B i 的最小值小于等于 C i 。

其中，C 也是一个 m 维向量，并且每一维的取值也只能是 1 或 2 或 3 或 4。A i 表示向量 A 的第 i 维的值，B i 和 C i 同理。

为了更加准确地说明“多少种方法”的含义，我们规定：令第 i 次读入的向量为第 i 个向量，那么一种方法就对应一个集合 {i,j}（i 不等于 j），两种方法不是同一种方法，当且仅当它们对应的集合 {i,j} 不相同。

n<=5e5,m<=10

把向量看成一个④进制数，相当于要我们对于每一个数求每一位的max/min等于这一个数的这一位的数对的方案数。
之后就是一个高维前缀和/后缀和就可以求答案了。

这里只讲求max的，求min的同理

记f[i]表示每一位都小于等于i的这一位的数的个数，那么从中选两个数，两个数每一位的max都会小于等于i的这一位。反过来，两个数的每一位都必须小于等于i的这一位，它们这一位的max才会小于等于i的这一位。
将原数组a做一遍高维前缀和就可以得到f，f做一遍高维差分就可以得到a
设g[i]=C(f[i],2)
g即为答案数组的高维前缀和，将g差分就是要求的方案数。

题

小月正沉迷某个网路游戏，该网游有 N 把剑可以选择，每把剑有可能拥有一些特殊的属性，例如；抗火、抗水、抗雷、可发光、可巨大化、很细等等之类的，总共恰有 20 种属性，把这些属性由 $1 \sim 20$ 编号，。每把剑都用一个长度 20 的字符串表示，第 i 个字符是 `'1'` 代表该把剑拥有第 i 种属性，若为 `'0'` 则没有。所有 2^{20} 不同的属性组合的剑都有可能存在。

现在小月要解 Q 个任务，每个任务都必须携带刚好 K 把不同的剑，任务一但开始后，所装备的 K 把剑就不能再更换。每个任务都可以用一个长度为 20 的字符串来表示，若第 i 个字符是 `'1'` 代表要通过该任务，所携带的剑至少要有一把拥有第 i 种属性，若第 i 个字符为 `'0'`，则代表就算没有任何携带的剑拥有第 i 个属性也有机会通过该任务。

现在对于每个任务，小月想知道，有多少种 K 把剑的组合，能让他有机会通过该任务，由于答案可能很大，你只要输出答案除以 10^9+7 的余数。

两种剑的组合视为不同当且仅当：存在某把剑，存在于其中恰一个组合。

n,k,q<=5e5

跟上一题一样，或运算同样能看做按位max

只是这题变成了取k个数
之前的C(f[i],2)变成了C(f[i],k)而已。

FWT

- fwt就是op为or,xor,and的卷积
- 从上一题可以知道or是做一遍高维后缀和，点积，再高维差分。
- 同理，and是高维后缀和。
- xor是一种奇怪的变换，有兴趣自己了解。

题

- 给一个n*n的矩阵。
- 要选n个格子，定义一种选择方案的价值为选的格子的权值在k进制下的不进位加法。
- 要求每一行每一列都要选恰好一个。
- 输出所有可能的价值。
- n<=50,k∈{2,3},0<=a[i][j]<k^7

- 提示：行列式

将每一个格子做一遍异或的fwt变换。
就是要求我们求这个矩阵的积和式的某一位的值是否为0。

由于积和式求不了，改为求行列式，为了避免正负抵消，给每一个位置一个随机权值。

3进制下的fwt
每一维都是模3意义下的循环卷积，我们可以类似fft求出在3次单位根中的点值。

最后逆变换回来，如果这个位置值不为0，那么这个价值一定存在，反之则大概率不存在。

太烦了不讲了（你们听不懂的
https://blog.csdn.net/qq_39972971/article/details/94439245
反正这个题很好，有兴趣自己看看吧。

多项式乘法

- 然而更多时候我们的卷积的op都是+，这个时候卷积的形式就跟多项式乘法一致了。
- 由于多项式乘法可以通过FFT(NTT)做到O(nlogn)，这就造成了万恶的多项式题目遍地开花。

- 而生成函数就是几乎所有多项式题目的必备前置技能。

题

- 给一棵树，有点权，定义一个连通块的权值为这个连通块内的点的权值之和模m，对于$i \in [0,m)$求权值为i的连通块数量，模$p=950009857$
- $n<=8000$，$m|p-1$，$m<=57984$，5s

循环卷积
首先有一个简单dp，f[i][j]表示在第i个点的子树中，权值为j的连通块的个数。
转移就是枚举子树的权值，类似背包的转移。O(nm^2)
当然用点分治然后做树形依赖背包可以做到O(nmlogn)，但是这个做法跟正解关系不大

观察转移的形式，发现这是一个循环卷积，即c[(i+j)mod m]=∑a[i]+b[j]
因为这是一个卷积的形式，当然是转换成点值计算。
循环卷积的点值参考fft
f[i]=∑a[j]*w^(ij) w是p的m次单位根
你会发现f是把a视为多项式之后带入第i个单位根的函数值，由于w^m=1=w^0所以溢出的系数会加到对m取模的那一项中。

由于转移的形式就是把子树的多项式相乘再乘以x^a[i]再加1。所以完全可以全程维护点值，最后再转成原来的值。复杂度O(nm)

生成函数

- 以下内容出自samjia2000《浅谈生成函数》

一般生成函数(Ordinary Generating Function)

- 我们定义

$$A(x)=\sum_{n\geq 0}A_n x^n$$

- 为A的一般生成函数(OGF)
- 当A为全体01串时，

$$A(x)=1+2x+4x^2+8x^3+\ldots$$

$$=\frac{1}{1-2x}$$

- 注意，在形式幂级数中，不需要考虑级数收敛

在形式幂级数中，x从来不指定一个数值，且对收敛和发散的问题不感兴趣，感兴趣的是系数序列(a(0),a(1),...,a(n),...) --百度百科

- 多项式后面的x^i可以理解成区分一个序列的每一项的工具。
- 你可以将其理解成一个dp数组，写成多项式的形式只是方便我们进行各种运算。

一般生成函数(Ordinary Generating Function)

- 以上只是对OGF的形式及运算的介绍，接下来放到实际问题中来理解生成函数：
- 有一个商店A，有许多商品，价格为i(i>0)的商品有i件
- 那么用生成函数A(x)表示在A商店买一件物品需要的价格所对应的方案数：

$$A(x) = x + 2x + 3x^3 + \dots$$

$$= \frac{x}{(1-x)^2}$$

- 其中A(x)[x^n]表示在A商店买一件价格为n的物品所对应的方案数

一般生成函数(Ordinary Generating Function)

- 接着，还有一个商店B，跟商店A一样，价格为i(i>0)的商品有i件
- 那么B(x)=A(x)
- 如果我们想要知道在A中买一件商品，B中买一件商品，对于每种可能的价格对应的方案数是多少
- 设这个的生成函数是D(x)
- 那么：

$$D(x) = A(x)B(x)$$

$$= \frac{x^2}{(1-x)^4}$$

- 其中$D(x)[x^n] = C(n+1,3)$

一般生成函数(Ordinary Generating Function)

- 在OGF的运算中，有一些很常用的式子：

$$\frac{1}{1-x}=\sum_{i\ge 0}x^i$$

$$\frac{1}{(1-ax)^m}=\sum_{n\ge 0}C(n+m-1,m-1)a^n x^n$$

$$e^x=\sum_{i\ge 0}\frac{x^i}{i!}$$

例题1

- 请求出$\{1,4,9,16,25\ldots n^2\}$的生成函数

例题1

$$F(x)=\sum_{i\geq 0} i^2 x^i$$

$$F(x)\cdot(1-x)=\sum_{i>0}(2i-1)x^i$$

$$=\frac{2x}{(1-x)^2}-\frac{x}{1-x}$$

$$=\frac{x(x+1)}{(1-x)^2}$$

$$F(x)=\frac{x(x+1)}{(1-x)^3}$$

例题2

- 给出下列数列的生成函数
- 1.斐波那契数列
- 2.卡特兰数

1.

由于f[i]=f[i-1]+f[i-2]

所以$F(x) = F(x)(x + x^2) + 1$

$F(x) = \frac{1}{1-x-x^2}$

2.

由于f[n]=∑f[i]*f[n-i-1](0<=i<n)

所以$F(x) = F^2(x)x + 1$

解得$F(x) = \frac{1-\sqrt{1-4x}}{2x}$

例题3

- 现在给你一有n个整数的序列a[]，有一个初始为0的值res，重复下面的过程k次：
- “随机选择一个[1,n]之间的下标x，res加上所有满足i≠x的a[i]的乘积，然后将a[x]减去1”
- 问最后res的期望值，对10^9+7取模
- n<=5000
- k<=10^9
- CF891E

例题3

- 设b[i]表示a[i]减少了多少次
- 答案其实就是$\prod_{i=1}^{n} a_i - \prod_{i=1}^{n}(a_i - b_i)$ 的期望值
- 那么设$F_i(x)$表示a[i]被减去b[i]之后对答案的贡献
- 那么

$$F_i(x) = \sum_{j\ge 0} \frac{(a_i - j)x^j}{j!}$$

$$= (a_i - x)e^x$$

- 那么 $F(x) = e^{nx}\prod_{i=1}^{n}(a_i - x)$
- 现在我们要求$F(x)[x^k]$，可以直接将后面的多项式展开，次数显然是n的，前面的展开很简单，暴力卷积即可，时间复杂度$O(n^2)$

指数型生成函数EGF

- 现在介绍另外一种生成函数：指数型生成函数(EGF)
- EGF的形式如下：

$$A(x)=\sum_{i\geq 0}\frac{a_i}{i!}x^i$$

- 在实际实现的时候，我们储存的值是 $\frac{a_i}{i!}$ 而不是 a_i，只有需要用的时候才会根据需要乘上i!来得到 a_i

指数型生成函数EGF

- 下面结合例子来进一步理解EGF
- 设n个节点带标号无向联通图个数为f_n
- 设n个节点的可以被分成两个无向联通图的无向图个数为g_n
- 那么

$$g_n = \frac{1}{2}\sum_{i=1}^{n-1} f_i f_{n-i}\binom{n}{i}$$

- 有

$$\frac{g_n}{n!} = \frac{1}{2}\sum_{i=1}^{n-1} \frac{f_i}{i!}\cdot\frac{f_j}{j!}$$

- 可以发现，与之前EGF的形式异曲同工，其实就是EGF卷积起来的形式

带标号对象的拼接

- 考虑一个带标号对象B是由若干个带标号对象A拼接而成的
- 那么 $B(x)=\sum_{i\geq 0}\frac{A(x)^i}{i!}=e^{A(x)}$
- 这个理解起来对于初学者有一定难度
- 举个例子，A中的三个带标号对象a,b,c，设其大小分别为|a|,|b|,|c|，将这三个对象拼接起来之后得到大小为|a|+|b|+|c|的B中的带标号对象，由于这样的组合总共算过了3!次，所以要除以3!
- 再举个更具体的例子，考虑长度为|a|+|b|+|c|的置换个数，对于其中一种情况，它可以由三个轮换a,b,c拼接起来，由于有组合数对节点标号进行分配，所以对于同一个置换是被计算了3!次的。

带标号对象的拼接

- 考虑长度为|a|+|b|+|c|的置换个数，对于其中一种情况，它可以由三个轮换a,b,c拼接起来，由于有组合数对节点标号进行分配，所以对于同一个置换是被计算了3!次的。
- 比如说a=(1,2),b=(1,3,2),c=(1)
- 那么会被这样计算3!次：
- a=(1,2)b=(4,6,5)c=(3)
- a=(1,2)c=(3)b=(4,6,5)
- b=(4,6,5)a=(1,2)c=(3)
- b=(4,6,5)c=(3)a=(1,2)
- c=(3)a=(1,2)b=(4,6,5)
- c=(3)b=(4,6,5)a=(1,2)

常见EGF

$$e^x = \sum_{i\geq 0} \frac{x^i}{i!}$$

$$\ln(1+x) = \sum_{i>0} \frac{(-1)^{i+1} x^i}{i}$$

$$-\ln(1-x) = \sum_{i>0} \frac{x^i}{i}$$

例题

- 求恰好包含n个节点的带标号无向联通图的个数。
- n<=100000
- 若干个带标号无向联通图拼接会得到带标号无向图。
- 设带标号无向联通图的生成函数是G(x)，带标号无向图的生成函数是F(x)
- 那么F(x)=e^G(x)，有G(x)=ln(F(x))
- 显然$F(x)=\sum_{i>0}\frac{2^{i(i-1)/2}x^i}{i!}$
- 那么可以先预处理出F(x)，通过对F(x)取ln可以得到G(x)

题

现在有 n 个点，每个点有个值 $f(i)$，表示 i 这个点走一步会到 $f(i)$ 这个点。一开始所有的 f 都没有确定。现在给定 k，要求对于每个 i，都满足 i 这个点，走不超过 k 步，能到一个点 c，满足 $f(c)=c$。

$n*k<=2000000, k<=3$

- 51nod 不动点

- 把 $f(i)$ 想象成是 i 指向祖先的边，那么就相当于要统计有多少种森林，满足每个点的深度都不超过 k。
- 令 $[x^n]g_k(x)$ 代表深度限制为 k 时，森林点数为 n 的方案数。当 $k=0$ 时，每个点都必须作为根，则有 $g_0(x)=e^x$，$k=1$ 时，我们考虑，一棵深度不超过 k 的树，相当于一个深度不超过 $k-1$ 的森林和一个根拼在一起，所以一棵深度不超过 k 的树的生成函数就是 $xg_{k-1}(x)$。然后这个新森林又是若干棵树的集合，所以 $g_k(x)=e^{xg_{k-1}(x)}$。

Ex:简单的多项式操作

- 你要是不会FFT就不用听了。

任意模数NTT

- 我们发现，当模数在1e9级别的时候，最后的答案会是约1e24级别的。
- 如果我们用3个不同的NTT模数，用CRT就可以还原出原来的答案，再对原模数取模。当然还可以用一大一小模数(29LL<<57|1)
- 用三模数NTT的都是sb（慢死了）
- 毛爷爷写了一篇论文解决这个东西。
- 把原数写成一个a*p+b的形式$p = \sqrt{Mod}$
- 那么答案是$f[i+j] = \sum(a[i]*p+b[i])(c[j]*p+d[j]) =$
- $\sum p^2 a[i]c[j] + p(a[i]d[j]+b[i]c[j]) + b[i]d[j]$
- 最终一共7次dft
- （如果把虚部也利用上就可以优化到4次dft）

求逆

$$D(x)B(x) = 1(mod\ x^n)$$

$$A(x)B(x) = 1(mod\ x^{2n})$$

$$B(x)(A(x) - D(x))^2 = 0(mod\ x^{2n})$$

$$A(x) = 2D(x) - D^2(x)B(x)$$

复杂度T(n)=O(nlogn)+T(n/2)=O(nlogn)

牛顿迭代

- 我们首先有多项式泰勒展开的式子

$$F(x) = \sum_{i \geqslant 0} \frac{F^{[i]}(x_0)(x-x_0)^i}{i!}$$

- F(x)在这里是一个自变量和因变量都是的函数。
- 设A是F(A)=0模x^2n的根，A0是模x^n的根，取前两项

$$F(A) = F(A_0) + F'(A_0)(A - A_0)$$

$$A = \frac{A_0 F'(A_0) - F(A_0)}{F'(A_0)}$$

- 由于后面每一项都包含(A-A0)^2，在模x^2n意义下为0
- 这样就可以将A的次数界提高一倍

- 练习：求逆和开根

$$F(A) = 1/A - B$$

$$A = A_0 - \frac{1/A_0 - B}{-1/A_0^2} = 2A_0 - BA_0^2$$

$$F(A) = A^2 - B$$

$$A = A_0 - \frac{A_0^2 - B}{2A_0} = \frac{B - A_0^2}{2A_0}$$

多项式ln

$$ln(A)' = \frac{A'}{A}$$

$$ln(A) = \int \frac{A'}{A}$$

复杂度T(n)=O(nlogn)+T(n/2)=O(nlogn)

多项式exp

$$F(A) = ln(A) - B$$

$$A = A_0 - \frac{ln(A_0) - B}{1/A_0} = A_0(1 + B - ln(A_0))$$

复杂度T(n)=O(nlogn)+T(n/2)=O(nlogn)