Product 命题报告

宁波市镇海中学 杜瑜皓

1 试题

1.1 题目描述

令$N = \prod_{i=1}^{n} p_i^{a_i}, M = \prod_{i=1}^{n} p_i^{b_i}$，其中$p_i$是两两不同的素数。

求将N表示成k个数的乘积的方案数，也就是$N = \prod_{i=1}^{k} x_i$解的个数，答案对$10^9 + 21$ 取模。

对于子问题1，要求对于所有整数i满足$1 \le i < k$，都有$x_i < x_{i+1}$。

对于子问题2，要求对于所有整数i满足$1 \le i < k$，都有$x_i \le x_{i+1}$。

对两个子问题都要求对于所有整数i满足$1 \le i \le k$，都有$x_i \nmid M$。

只有子问题1答案正确将获得3分，只有子问题2答案正确将获得6分，两问都正确将获得10 分。

1.2 输入格式

第一行两个正整数n, k。

接下来一行n个非负整数，第i个表示a_i。

接下来一行n个非负整数，第i个表示b_i。

1.3 输出格式

一行两个整数，表示子问题1和2的答案。

1.4 数据范围

数据编号	n	a_i	b_i	k
1	≤ 5	≤ 5	≤ 5	≤ 3
2	≤ 10	≤ 20	≤ 20	≤ 5
3	$= 1$	$\leq 10^{18}$	$\leq 10^{18}$	≤ 25
4	≤ 50	$\leq 10^3$	$= 0$	≤ 20
5	≤ 50	$\leq 10^{18}$	$= 0$	≤ 25
6	≤ 50	$\leq 10^3$	$\leq 10^3$	≤ 10
7	≤ 50	$\leq 10^{18}$	$\leq 10^{18}$	≤ 10
8	≤ 50	$\leq 10^{18}$	$\leq 10^{18}$	≤ 15
9	≤ 50	$\leq 10^{18}$	$\leq 10^{18}$	≤ 20
10	≤ 50	$\leq 10^{18}$	$\leq 10^{18}$	≤ 25

时限为6秒，内存256MB。

2 一个计数的方法

考虑一个一般化的问题，对k个计数对象计数，这k个计数对象之间是等价的，同时这些计数对象总的有一个限制，求这样k个对象的集合或者可重集的个数。这里的顺序问题处理起来十分困难，所以我们要通过一些手段转化成求着k个对象的序列，或者转化成只有等价关系没有大小比较的序关系。

2.1 关于集合的做法

首先考虑集合的计数问题，由于所有数字都不同，所以只要求出两两不同的方案然后除掉$k!$即可。

所有数字两两不同，等价于$\frac{k(k-1)}{2}$个限制，$s_i \neq s_j$，于是可以考虑容斥。

也就是没有限制的方案数减去违反了某个限制的方案数加上违反了某两个限制的方案数等等的结果。

对于某个限制$s_i \neq s_j$，如果违反了也就代表着$s_i = s_j$。

如果容斥中枚举了方案的子集，也就是子集中每个限制对应的两个数必须相同，也就是所有数字划分成了若干个等价类。

如果暴力枚举这些限制是$O(2^{\frac{k(k-1)}{2}})$的。

注意只要关注每个等价类的大小就行了，具体的顺序和对应哪些数并不影响答案。因为这k个计数对象是不可区分的。

上述做法中枚举限制关系是瓶颈。我们可以换一个思路，枚举等价类的划分，所以也就是k的拆分数，记作P_k。然后对每一类拆分计算它在答案中会被统计到几次。

将每个数当成一个点，等价关系当成边，那么一个等价类相当于一个连通块。首先枚举连通块和点的对应关系，也就是点的编号。

假设这个拆分中有a_i个i，其中有$\sum_{i=1}^{k} a_i * i = k$，那么对应关系就相当于把$k$个数划分成若干个无序集合的方案数。

首先假设集合有标号，那么由多项式系数得方案为$\frac{k!}{\prod_{i=1}^{k}(i!)^{a_i}}$，然后大小相同的集合之间又是无序的，所以还要除掉$a_i!$。

所以方案数为$\frac{k!}{\prod_{i=1}^{k}(i!)^{a_i}a_i!}$。

点的对应方案确定，接下来要算边的系数。

令S_n表示n个点的连通图集合，$e(G)$表示G的边数。

假设这个划分为$p_1, p_2, \cdots, p_m$，那么系数为$\sum_{G_1 \in S_{p_1}} \cdots \sum_{G_m \in S_{p_m}} (-1)^{\sum_{i=1}^{m} e(G_i)}$。

因为每个求和之间独立，也就是$\prod_{i=1}^{m} \sum_{G_i \in S_{p_i}} (-1)^{e(G_i)}$。

记$g_i = \sum_{G \in S_i} (-1)^{e(G)}$，所以系数也就是$\prod_{i=1}^{m} g_{p_i}$。

接下来考虑如何求g_n，也就是对所有n个点的连通图求和，对于图G，权值为$(-1)^{e(G)}$。

类比求连通图的做法，使用容斥，也就是首先考虑全集的权值，也就是n个点图的集合，减去不连通图的权值，也就是枚举点1所在的连通块。

首先假设有x条边，如果$x = 0$，那么全集的权值和为1，否则由二项式定理$\sum_{i=0}^{x} (-1)^i \binom{x}{i} = 0$。

对于不连通图，考虑1所在的连通块，假设有k个点，那么就有$\binom{n-1}{k-1}$个集合，然后对剩下$n-k$个点的所有图权值求和。

如果$n - k > 1$，那么和为0，所以当且仅当$k = n - 1$时，对g_n有贡献，也就是$g_n = -(n-1)g_{n-1}, g_1 = 1$，所以有$g_i = (-1)^{i-1}(i-1)!$。

结合这两部分，就能知道每个拆分对答案的贡献。

这样做是以$O(P_k)$和额外的限制若干个等价类转化到序列问题，也就是没有了序关系。等价关系比序关系好处理得多。

2.1.1 关于g_i取值的另一个证明

关于g_i的取值有一个不基于数学归纳法的证明，在这里补充。

对一个集合S，令g_S表示这个集合内点形成连通图的权值和。如果$|S| = 1$，那么$g_S = 1$。

否则考虑编号最小的两个点v_1, v_2。如果一个连通图不存在(v_1, v_2)这条边，那么将这条边加入到图中，还是个连通图，并且这两个图边数差1，所以恰好配对，权值和为0。考虑不能配对的连通图，也就是多出来的，这样的图中一定存在(v_1, v_2)这条边，并且去掉这条边整个图不连通，分成两个子集。

接着递归考虑这两个子集，考虑两个子集编号最小两个点连成的边。在不能配对的情况中，相当于去掉这条边这个子集不连通，接着分成两个更小的子集。

这样最后形成了一棵树的结构，对于每个点$v_i(2 \le i \le n)$，向$v_j(1 \le j < i)$连边，表示这条边是某个子集割边。同时也表示只有这样的图无法配对。

所以$g_i = (-1)^{i-1}(i-1)!$

2.2 关于可重集的容斥原理做法

类比上面的算法，可以这么想，首先可以枚举每种等价类的拆分S，假设大小分别为$p_1, p_2, \cdots, p_m$，然后计算出恰好有p_1 个数相同，p_2个数相同等等，然后每类数之间两两不同的方案数。每种拆分的重复计算的次数都是相同的。

首先考虑每个等价类重复计算的次数，然后将其除去得出有序的方案。对于每个等价类，求出的方案相当于按某个顺序排序了，只有若干个大小相同的等价类不可区分。如果一个拆分有a_i个大小为i的等价类，其中有$\sum_{i=1}^{k} a_i * i = k$，那么这$a_i$个是不可区分的，所以被重复计算了$a_i!$次。所以这个方案被重复计算了$\prod_{i=1}^{k} a_i!$，在总方案中除掉即可。

接下来要计算的等价类恰为某种划分的方案数。集合也就是两两不同为这个问题的一个特例。

如果使用容斥，对于一个划分，限制为每两个等价类之间的数字两两不同。所以可以暴力枚举这些限制，然后容斥。

这样做的代价为$O(2^{\frac{k(k-1)}{2}} P_k)$。

剩下要做的也就是快速计算对于拆分之间的容斥系数。

拆分之间构成了偏序集，如果拆分S通过合并其中某些等价类能得到T，那么称为$S \leq T$。

令$f(S)$表示所有等价类T满足$S \leq T$的划分的方案数的和，$g(S)$表示等价类恰为S的集合。

那么$f(S)$就是对于每个划分dp出来的方案，$f(S) = \sum_{S \leq T} g(T) * w(S,T)$，其中$w(S,T)$表示重复的方案数，也就是$T$这个划分可能在$S$中被计数多次。

而要计算的是$g(S) = \sum_{S \leq T} f(T) * \mu(S,T)$，也就是要计算$\mu(S,T)$。

考虑状压dp，首先枚举集合S，然后每次选出一个子集，将其合并成为一个数字。类似前文的论证可得将i个等价类合并的权值为$(-1)^{i-1}(i-1)!$，而一个方案最后的权值为所有等价类合并权值的积。

为了防止重复计数，可以对子集进行编号，也就是让每次选出的子集编号递增，所有子集个数为$\sum_{i=1}^{k} P_i$。

那么状态可以表示为$dp_{S_1,S_2,p}$，即现在已经选完编号在$1 \leq p$之间的子集，之前选出的子集合并后的数集合为S_1，剩下没合并的数集合为S_2的权值总和。

转移的时候假设当前考虑合并子集T，那么转移到$S_1 \cap \{\sum T\}, S_2 \backslash T$这个状态，其中$\sum T$表示$T$中数的和，同时乘上将$T$这个集合内等价类合并的权值和在$S_2$中选出$T$这个子集的方案。如果$T$这个子集被删去了$k$次，那么要在最后除掉$k!$这个系数，因为这$k$个子集不可区分。

这样总状态数为$O((\sum_{i=1}^{k} P_i)^3)$，然而对某个$S$的集合做的时候，$S_1$一定是$S$的子集能合并出来的状态，$S_2$一定是$S$的子集，$p$也只要对$S$的子集编号即可，所以远远达不到上面的复杂度。

考虑最后的答案，将$g(S)$全部表示成$f(S)$的线性组合，那么答案一定能被表示成$f(S)$的线性组合，也就是$\sum f(S) * w(S)$。

那么这样只要枚举将S划分成多少个子集，然后统计对应的系数。只要爆搜每个数字的划分方案，然后记忆化即可，这样做更加方便。

2.3 关于可重集的Burnside引理做法

换个角度考虑问题，那么可重集相当于就是问在在所有k的置换也就是对称群S_k作用下不同构的元素个数，即在对称群S_k下轨道的个数。所以可以用Burnside引理解决。

对于某个置换P，将其拆成若干个轮换，如果一个方案在这个置换下不变就

相当于在同一个轮换中的元素相同。

所以相当于一个置换将一些元素划分成了若干个等价类。枚举等价类的划分，也就是k的拆分，然后计算多少个置换对应的等价类是这样的。

和前面相同，首先考虑将元素划分到若干个等价类的方案。假设这个拆分中有a_i个i，其中有$\sum_{i=1}^{k} a_i * i = k$，方案数为$\frac{k!}{\prod_{i=1}^{k}(i!)^{a_i}a_i!}$。

然后对于一个轮换长度为i，那么对应着$(i-1)!$个圆排列，所以方案为$\frac{k!}{\prod_{i=1}^{k} i^{a_i}a_i!}$。

2.4 从群论的角度考虑集合的计数

我们注意到一个这样的事实，对于一个等价类的划分，计算可重集时Burnside引理得到的系数和计算集合时从容斥原理得到的系数除了符号全部相同，更一般的有假设有m个等价类，那么容斥系数为$(-1)^{k-m}$倍。

如果一个置换σ由m个轮换[1]组成，那么$(-1)^{k-m} = \mathrm{sgn}(\sigma)$。

类似Burnside引理，有对每个σ属于S_k令X^σ表示X中在σ作用下的不动元素。由这个容斥得到的式子为：

$$\frac{1}{k!}\sum_{\sigma\in S_k}|X^\sigma|\mathrm{sgn}(\sigma)$$

如果一个序列中的元素两两不同，那么它只在恒等置换作用下不动。由于这个序列的所有置换都被算入，最后又除掉了$k!$，所以这样的序列恰好被算了一次。

假设一个序列中等价类的大小为$p_1, p_2, \cdots, p_m$，那么在一个置换作用下这个序列不动，必有这个置换中的每个轮换的元素属于同一个等价类。

如果存在一个等价类大小超过1，记作t，因为交错群$|A_t| = t!/2$，所以奇置换和偶置换个数相同，也就是对应的和为0，所以如果有两个元素相同，那么在上面的式子中被算入0次。

不过对于一般的置换群，关于只在恒等置换作用下不动的元素计数，并没有上面这样简单的式子，并且我也没有找到相关的资料。

关于这个问题，我也在思考中，也欢迎读者来交流。

[1]这里一个置换的轮换个数默认为完全轮换分解中轮换的个数，下同。

2.5 第一类Stirling数

对于这些计数对象，如果加的限制为只与等价类的大小有关，那么可以有多项式的做法。

我们只要知道对称群S_k中轮换个数为m的置换个数，也就是将k个元素的分作m个圆排列的方法数，也就是$(-1)^{m+k}s(k,m)$，其中$s(k,m)$为第一类Stirling数。

其中$\prod_{i=0}^{k-1}(x-i)=\sum_{i=1}^{k}s(k,i)x^i$。

如果是集合的计数，那么对应的系数恰为$s(k,m)$。使用这个结论，可以很方便的解决一些问题。

2.5.1 例题 CountTables[2]

求$n*m$的数表个数，要求每个数在$1\sim c$之间，并且要求任意两行不能完全相同，任意两列不能完全相同。

分析一下这个题的计数对象是大小为n的向量，并且每一维在$1\sim c$之间，额外的限制为不存在某两维完全相同。

假设有i个等价类，那么每一维有c^i个取值，两两不同的方案数为$\binom{c^i}{n}*n!$。

于是答案为$\sum_{i=1}^{m}s(m,i)\binom{c^i}{n}*n!$。

这个题通过直接dp和Stirling反演能得到同样的式子，但是通过这个方法就能很轻松地得到这个问题显示的解。

3 一个不定方程的非负解数

接下来再考虑一个方程$\sum_{i=1}^{m}p_ix_i=a$解数的算法。

3.1 基于动态规划的做法

可以看成一个简单的完全背包计数。那么可以在$O(ma)$的代价用背包解决。

令$dp_{i,s}$表示选了前i个数，总和为s，那么$dp_{i,s}=dp_{i,s-p_i}+dp_{i-1,s}$。

[2]来源: TCO 2014 Wildcard 750分题

3.2 基于循环节的做法

引理： $\sum_{i=1}^{m} p_i x_i = a$的非负整数解个数是一个以$a$模$\mathrm{lcm}_{i=1}^{m} p_i$为周期的关于$a$的次数不超过$m-1$的多项式。

简单的证明，这相当于$\frac{1}{\prod_{i=1}^{m}(1-x^{p_i})}$的第$a$项，可以将分母的所有根求出来，表示成若干个$(1-\omega_j^i x)$的乘积。然后一定可以拆成若干个部分分式，也就是$\frac{p(x)}{(1-\omega_j^i x)^k}$的和，其中$p(x)$为一个关于$x$的次数不超过$k-1$的多项式。

对于每项，分母可以用二项式定理展开$\frac{1}{(1-\omega x)^k} = \sum_{i\geq 0}\binom{i+k-1}{k-1}\omega^i x^i$，其中$\binom{i+k-1}{k-1}$是一个次数不超过$k-1$的多项式，假设$\omega$是$j$次单位根，那么这一部分是模$j$为周期的多项式。易证将这个幂级数乘上$p(x)$还有同样的性质。

所以若干个加起来就有引理的结论了，并且次数不超过重数最多的根也就是1的重数。

如果知道了前$m * \mathrm{lcm}_{i=1}^{m} p_i$项的取值，那么可以在$O(m)$的时间复杂度内求出某项的值。

首先可以得到模周期的$m+1$个值$f(0), f(1), \cdots, f(m)$，假设要求的是$f(n)$，那么由拉格朗日插值得$f(n) = \sum_{j=0}^{m}(-1)^{m-j} f(j)\frac{\prod_{i=0}^{m}(n-i)}{(m-j)!\,j!\,(n-j)}$。这个可以在$O(m)$内算出。

3.2.1 引理的一个简单证明

令$M = \mathrm{lcm}_{i=1}^{m} p_i$，令$x_i = \frac{M}{p_i} t_i + r_i (0 \leq r_i < \frac{M}{p_i})$。如果枚举了$r_1, r_2, \cdots, r_m$，那么方程变成$M\sum_{i=1}^{m} t_i + \sum_{i=1}^{m} p_i * r_i = a$。令$c = \sum_{i=1}^{m} p_i * r_i$，如果$a \equiv c \pmod{M}$，那么就变成$\sum_{i=1}^{m} t_i = \frac{a-c}{M}$。因为$c \leq mM$，所以$\frac{a-c}{M} \geq -m+1$，那么解数为$\binom{\frac{a-c}{M}+m-1}{m-1}$。

对于模M同余的某个数，都是有限个次数不超过$m-1$的多项式相加，所以也是个多项式。

3.3 基于生成函数的做法

所以解的个数为幂级数$\prod_{i=1}^{m}\sum_{j\geq 0} x^{jp_i}$中第$a$项的系数。

$$\prod_{i=1}^{m}\sum_{j\geq 0} x^{jp_i} = \prod_{i=1}^{m}\frac{1}{1-x^{p_i}} = \frac{1}{\prod_{i=1}^{m}(1-x^{p_i})}$$

也就是$\frac{1}{\prod_{i=1}^{m}(1-x^{p_i})}$中第$a$项前的系数。

将其看成一个数列的生成函数，这是一个常系数线性递推数列。将分母展开就能得到得到递推的系数，而通过动态规划能得出这个数列初始值。

令$s = \sum_{i=1}^{m} p_i$，通过矩阵快速幂可以做到$O(s^3 \log a)$，通过倍增可以做到$O(s^2 \log a)$，可以使用FFT加速做到$O(s \log s \log a)$。

3.4 基于数位dp的做法

令$x_i = \sum_{j\geq 0} 2^j x_{i,j}, (0 \leq x_{i,j} \leq 1)$，于是相当于物品为$p_i * 2^j$的01背包。令$dp_{i,j}$表示只考虑$2^j$倍数的物品，剩余的体积为$i * 2^j$的方案数。剩余的物品体积不会超过$s * 2^j$，所以当$i \geq s$时不可能再填满。

所以状态数不会超过$s \log a$，然后对于每个物品使用01背包转移，时间复杂度为$O(ms \log a)$。

3.5 基于部分分式的做法

类似引理的第一个证明方法，可以将它拆成若干个部分分式，然后分别计算。当p_i两两互质时，可以使用单位根的技巧做到$O(ms)$的复杂度。不互质时也可以暴力拆成部分分式。然后在$O(m)$的复杂度解决。

由于过于复杂，并且与这个题关系不大，所以略去。有兴趣的读者可以阅读参考文献[2]。

4 关于前三个点的算法

在这个问题里面计数对象为存在一个j满足$s_{i,j} > b_j$的n维向量，限制为对于某一维j满足$\sum_{i=1}^{k} s_{i,j} = a_j$。

令$x_i = \prod_{j=1}^{n} p_j^{s_{i,j}}$，也就是每个数的因子分解。

满足$\sum_{i=1}^{k} s_{i,j} = a_j$。因为$x_i \nmid M$，所以对于所有$i$，存在$j$使得$s_{i,j} > b_j$。

记e为a_i, b_i中的最大值。

4.1 算法1

爆搜出所有可行方案，然后验证是否合法。

4.2 算法2

第二个点中$k \le 5$。考虑使用动态规划一个一个因子确定，同时保证s_i按字典序不降。

当两个前缀$s_{i,1..j}, s_{i+1,1..j}$确定时，如果$s_{i,1..j} < s_{i+1,1..j}$，那么$s_{i,j+1}$和$s_{i+1,j+1}$之间没有限制。如果$s_{i,1..j} = s_{i+1,1..j}$，那么要有$s_{i,j+1} \le s_{i+1,j+1}$。

这个时候要用一个状态S表示相邻两个前缀相同的集合和对于每个前缀i是否已近存在j满足$s_{i,j} > b_j$。令$f_{i,S}$表示处理完前i个因子状态为S的方案数。同组内转移记$g_{S',i,j,k}$表示当前集合为S'，选了i个数，总和为j，上一个数为k，枚举每个数的取值，然后更新当前状态。

同组内状态数为$O(e^2k4^k)$，转移复杂度为$O(e)$，共有$O(n)$组。

时间复杂度为$O(ne^3k4^k)$。

4.3 算法3

第三个点中$n = 1$，令$t_i = s_{i,1}$，所以问题等价于求$\sum_{i=1}^{k} t_i = a_1$，然后$M \nmid x_i$，也就是$t_i > b_1$。

因为$t_1 \le t_2 \le \cdots \le t_n$，所以就是$t_1 > b_1$。

记$\delta_i = t_{k+1-i} - t_{k-i}$，令$t_0 = 0$，那么$\sum_{i=1}^{k} t_i = \sum_{i=1}^{k} \delta_i * i = a_1$。

求这个方程的非负整数解，并且对于δ_i有$\delta_i \ge c_i$这样的限制。对于子问题1，有$c_1 = b_1 + 1$，对于所有$i \ne 1$，$c_i = 1$。对于子问题2，有$c_1 = b_1 + 1$，对于所有$i \ne 1$，$c_i = 0$。

就相当于$\sum_{i=1}^{k} x_i * i = a_1 - \sum_{i=1}^{k} c_i * i$的非负整数解。

套用前面的生成函数或者数位dp做法就可以通过。

5 后面几个点的做法

5.1 算法1

上面的结论还不能直接解决这个问题，因为它只是转化到了有若干个等价类的方案数。对于这个问题中特殊的计数对象还要特别考虑。

首先考虑一个简单的问题即没有限制的情况。在上一步转化下就变成某些变量是相等的，然后每一维加起来和要等于这一维对应的值。

那么可知每一维都是独立的，也就是算出每一维分解成若干个等价类，然后将所有答案乘起来的方案数。

假设等价类的大小为$p_1, p_2, \cdots, p_m$，这一维总和为a，那么就是将$\sum_{i=1}^{m} p_i x_i = a$的非负整数解的个数。

那么可以在$O(ma)$的代价用背包解决，$dp_{i,s}$表示做了前i个等价类，总和为s，那么$dp_{i,s} = dp_{i,s-p_i} + dp_{i-1,s}$。

所以这个问题可以在$O((n+ke)P_k)$内解决。

记$f(N,k)$表示将N分解成k个严格递增的数的方案数，$g(N,k)$表示将N分解成k个不降的数的方案数。

回到原问题，对于$b_i = 0$的情况就是$x_i \neq 1$。

对于子问题2，答案就是$g(N,k) - g(N,k-1)$，也就是去掉$x_1 = 1$的方案数。

对于子问题1，令$h(N,k)$表示将N分解成k个严格递增的数且第一个数超过1的方案数，那么$h(N,k) = f(N,k) - h(N,k-1)$。也就是如果$x_1 = 1$，那么一定有$x_2 \geq 2$。所以最后的答案为$\sum_{i=0}^{k} f(N,i)(-1)^{k-i}$。

时间复杂度为$O((n+ke)\sum_{i=1}^{k} P_i)$。

5.2 算法2

接下来考虑如何处理存在一个j满足$s_{i,j} > b_j$的n维向量的计数问题。

考虑使用容斥，如果不存在j满足$s_{i,j} > b_j$，也就是$s_{i,j} \leq b_j$。

对于每个等价类，枚举对应的变量是否违反限制也就是M的约数。如果一个变量违反了限制，也就是每一维不能超过b_j。

假设等价类的大小为$p_1, p_2, \cdots, p_m$，令$dp_{i,s}$表示做了前i个等价类，总和为s，那么$dp_{i,s} = dp_{i,s-p_i} + dp_{i-1,s} - dp_{i,s-(b_j+1)*p_i}$。即减去这一维超过$b_j$的方案。

时间复杂度$O(P_k 2^k (ke+n))$。

5.3 算法3

首先考虑加快容斥的情况，如果有a_i个大小为i的等价类，那么这a_i个等价类是不可区分的，也就是如果其中x个违反了限制，对应的情况数为$\binom{a_i}{x}(-1)^x$。所以对于一个划分，只要做$\prod_{i=1}^{k}(a_i+1)$种情况就好了。

当$k = 25$时所有划分的情况总和为129512，为了方便描述，将这个记为k的一个函数Q_k。

有前面的做法可知，如果知道了前$m*\mathrm{lcm}_{i=1}^{m}p_i$项的取值，那么可以在$O(m)$的时间复杂度内求出某项的值。

为了方便描述，令所有拆分中对应要知道取值的项记为R_k。当$k=25$时，R_k[3]为5040。

在枚举这些拆分的时候可以用dp顺便算出这R_k个初值。

接着考虑这些限制。对于第j维，如果一个大小为p_i的等价类内数只能取$0\cdots b_j$，那么对应的生成函数为$\frac{1-x^{(b_j+1)p_i}}{1-x^{p_i}}$。所以分母为$\prod_{i=1}^{m}(1-x^{p_i})$，分子中因为指数都为$(b_j+1)$的倍数，所以可以当成$x^{b_j+1}$的$m$次的多项式，在$O(m^2)$ 复杂度内展开，并且对于n维形式都相同。

此时分子至多有m项，那么可以对每项分别算对答案的贡献。假设第i项为wx^c，那么对应的答案为$\frac{1}{\prod_{i=1}^{m}(1-x^{p_i})}$的第$a-c$ 项乘上w，所以可以在$O(nk^2)$的时间复杂度内算出答案。

注意对于每个拆分，每一维要求的值为$a_j, a_j-(b_j+1),\cdots,a_j-m*(b_j+1)$，这些项可以在$O(nk^2)$内时间预处理出来，可以将上面每个容斥的复杂度降到$O(nk+k^2)$。

其中$Q_k=\sum_{i=0}^{k}P_iP_{k-i}$，因为$Q_k$表示将数字分成两部分，一部分选入，一部分不选入，两边分别划分的方案总和。

所以总时间复杂度为$O(R_k\sum_{i=1}^{k}P_i+nk^2P_k+(nk+k^2)(\sum_{i=0}^{k}P_iP_{k-i}))$。

6 总结

这个题的本意是对于一个特殊的计数对象求它的集合和可重集个数。在第二节中叙述了怎么将这个问题转化，通过了容斥原理和Burnside引理，最后引入了群的作用的观点，直观地统一地解释了这两个方法。这个做法对一般的计数对象都是可行的，并且给出了它方便地解决算法竞赛题的一个例子。

对于可重集的计数使用容斥原理，是一个不太成功的尝试，因为它打破了问题的对称性，所以失去了很多优美的性质。但是它能得到一些更微观具体的结果，比如某个特定结构的方案数。

由于本题的特殊性，第三节介绍了求一类不定方程解数的一些算法，要结合实际实际情况，选择最简便并且效率可以接受的做法。第四第五节介绍了这

[3]这个可以近似看成S_k中阶最大的元素的阶乘上k，关于这个的渐进复杂度可以查阅参考文献[1], [3]

个问题几个特殊点的做法和一般的做法。

本题是一道难题，主要考察选手的计数水平和数学功底，包括Burnside引理，容斥原理，生成函数，多项式等方面的知识和灵活的运用。

对于一般置换群轨道的计数已经由Burnside引理可以解决，关于只在恒等置换作用下不动的元素计数，或者某种特定结构的元素的计数还没有什么好的方法。

虽然这并不是一个简单的问题，不过也希望大家包括我自己能提出一些更有效的算法。

致谢

感谢中国计算机学会提供学习和交流的平台。

感谢李建老师的关心和指导。

感谢上海交通大学郭晓旭同学和杭州学军中学徐寅展同学的交流和帮助。

参考文献

[1] E. Landau, “Uber die Maximalordnung der Permutationen gegebenen Grades”

[2] 黄志翱, “Codechef CHANGE解题报告”

[3] Jean-Pierre Massias, Jean-Louis Nicolas, and Guy Robin, “Effective Bounds for the Maximal Order of an Element in the Symmetric Group”

[4] Hansraj Gupta, “On the Partition of J-partite Numbers”