Product 解题报告

浙江省镇海中学 杜瑜皓

1 试题来源

2015年集训队互测

2 试题大意

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。

3 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

4 算法介绍

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$M \nmid x_i$，所以对于所有$i$，存在$j$使得$s_{i,j} > b_j$。

那么子问题1等价于求集合$\{s_i\}$的个数，子问题3等价于求可重集$\{s_i\}$的个数。

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。

期望得分10分。

4.2 算法2

第二个点中$k \leq 5$。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$s_{i,j+1} \leq s_{i+1,j+1}$。

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$O(ne^3k4^k)$。

期望得分20分。

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$\sum_{i=1}^{n}t_i=a_1$，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$\delta_{k-i+1}=x_i-x_{i-1}$，令$x_0=0$，那么$\sum_{i=1}^{k}x_i=\delta_i*i=a_1$。

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$i\ne 1$，$c_i=0$。

所以解的个数为幂级数$\prod_{i=1}^{n}\sum_{j\ge c_i}x^{ji}$中第$a_1$项的系数。

$$\prod_{i=1}^{n}\sum_{j\ge c_i}x^{ji}=\prod_{i=1}^{n}\frac{x^{c_ij}}{1-x^i}=\frac{x^{\sum_{i=1}^{n}c_ij}}{\prod_{i=1}^{n}(1-x^i)}$$

也就是$\frac{1}{\prod_{i=1}^{n}(1-x^i)}$中第$a_1-\sum_{i=1}^{n}c_ij$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。通过倍增可以做到$O(k^4\log e)$。

期望得分10分。

4.4 算法4

接下来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。

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$\sum_{i=1}^{k}s_{i,j}=a_j$。

4.4.1 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_l。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$\sum_{i=1}^{k} a_i * i = n$，那么对应关系就相当于把$n$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。

4.4.2 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$\sum_{i=1}^{k} a_i * i = n$，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$S \le T$。

令$f(S)$表示所有等价类T满足$S \le T$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，$f(S) = \sum_{S \le T} g(T) * w(S,T)$，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$\sum f(S) * w(S)$。

我们只要预处理出$w(S)$将其打表即可。这样也能解决题目中的限制$k = 25$。

4.4.3 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$\sum_{i=1}^{k} a_i * i = n$，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$\frac{k!}{\prod_{i=1}^{k} i^{a_i} a_i!}$。

4.5 算法5

上面的算法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$O((nk+e)P_k)$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$O((nk+e)\sum_{i=1}^{k} P_i)$。

期望得分10分。

4.6 算法6

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$O(P_k 2^k (ke+n))$。

期望得分30分。

4.7 算法7

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。

接着考虑当e大时候的情况。首先考虑没有限制的情况，假设当前等价类的大小为$p_1,p_2,\cdots,p_m$，对于一个b就是求$\sum_{i=1}^{m}p_ix_i=a$的非负整数解。参考算法3使用生成函数和倍增能在$O(k^2\log e)$时间内解决。

为了进一步优化复杂度，首先证明一个引理。

引理： $\sum_{i=1}^{m}p_ix_i=a$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$(1-\omega_j^ix)$的乘积。然后一定可以拆成若干个部分分式，也就是$\frac{p(x)}{(1-\omega_j^ix)^k}$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$\frac{1}{(1-\omega x)^k}=\sum_{i\geq 0}\binom{i+k-1}{k-1}\omega^ix^i$，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。

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$O(nk+k^2)$。

所以总时间复杂度为$O(R_k\sum_{i=1}^{k}P_i+nk^2P_k+(nk+k^2)Q_k)$，期望得分100分。

5 总结

本题是一道难题，主要考察选手的计数水平，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。虽然这些知识都很基础，但是同时出现在一个题目中并且灵活运用这些手段去解决问题还是困难的。

预计集训队互测中有0 ~ 2名选手得到50分以上的分数，选手可以根据投入的时间得到其中相应的部分分。