《密码系统》浅析

IOI 2010 中国国家集训队论文

杭州二中 赖陆航

本文提出了一道 NOI 难度的试题，笔者用快速转移的动态规划和容斥原理解决了它，并给出了一组有效的测试数据来评估选手的解法。同时，本文还总结了作者对于动态规划快速转移的一些想法。

目录

目录.....1
正文.....2
一、 试题.....2
【问题描述】.....2
【输入文件】.....2
【输出文件】.....2
【输入样例】.....2
【输出样例】.....2
【样例说明】.....3
【数据规模和约定】.....3
二、 解题思路.....4
三、 解题算法.....5
四、 出题思路.....5
五、 数据生成方法与分数分布.....6
六、 总结与扩展.....7
参考文献.....7
感谢.....8
附录.....8

正文

一、 试题

密码系统(crypto)

时间限制：2s

空间限制：128M

【问题描述】

Lambda 受任于某情报站，他的工作是获取敌人情报。一次他在破解密码系统时，得到了一个 *N* 位 *B* 进制数 φ，满足 $\varphi \equiv V \pmod{M}$。他发现组成 φ 的数字很奇特。为了验证 φ 的特殊性，他将所有模 *M* 为 *V* 的 *N* 位 *B* 进制数，按照各数位构成的集合分类，并想知道每一类数各有多少个。

【输入文件】

输入文件 crypto.in 共一行，包含四个整数 *N*, *B*, *M*, *V*。

【输出文件】

输出文件 crypto.out 共 2^B-1 行，每行包含一个集合 *S* 和整数 *Ans*[*S*]，以单个空格隔开。集合 *S* 用其所有元素的递减序列表示，如{2, 0, 1}表示为“210”。*Ans*[*S*]表示数位集合为 *S* 的满足以上性质的数的数目。

集合按照字典序输出，每个集合只输出一次。由于 *Ans*[*S*]可能很大，只需输出它除以 10007 的余数即可。

【输入样例】

```
3 3 4 1
```

【输出样例】

```
0 0
1 1
10 1
```

```
2 0
20 0
21 2
210 1
```

【样例说明】

在所有三位三进制数（100_3~222_3）中，模 4 为 1 的数为 100_3, 111_3, 122_3, 210_3, 221_3。数位集合为{1}的有 1 个（111_3），数位集合为{1, 0}的有 1 个（100_3），数位集合为{2, 1}的有 2 个（122_3, 221_3），数位集合为{2, 1, 0}的有 1 个（210_3）。

【数据规模和约定】

测试数据	*N*	*B*	*M*
1~2	≤10	≤3	≤12
3~6	$\leq 10^3$		
7~10	$\leq 10^9$		
11~14			≤120
15~18		≤10	≤8
19~20			≤40

对于 15%的测试数据，gcd(*B*, *M*)>1。

对于所有测试数据，2≤*N*, 2≤*B*, 0≤*V*<*M*。

二、 解题思路

本题 N 的范围比较大，考虑使用动态规划算法。容易想到朴素的动态规划：

设 $DP[n, s, m]$为长度为 n，数位集合为 s，模 M 为 m 的 B 进制数（首位可以为 0）的数目（模 10007）。则

$DP[n', s', m']=\text{sum}\{ DP[n, s, m] \mid d\in\{0, \dots, B-1\}, n+1=n', s\cup\{d\}=s', Bm+d\equiv m' \pmod M \}$

由于 N 最高可达 10^9，转移又是线性关系，尝试用矩阵快速幂加速。然而矩阵的阶 $Size=2^BM$，无法承受。注意到这个矩阵是循环矩阵，利用此性质可以在 $O(Size^2)$内进行矩阵乘法。

然而这个方法并不是很直观，能否直接对转移进行优化？在朴素的动态规划中，n 每次增加 1，难道不能提高吗？尝试将长度分别为 n_1 与 n_2 的两个 B 进制数合并，则可以得到以下转移方程：

$DP[n', s', m']=\text{sum}\{ DP[n_1, s_1, m_1]DP[n_2, s_2, m_2] \mid s_1\cup s_2=s', B^{n2}m_1+m_2\equiv m' \pmod M \}(n_1+n_2=n')$

如果把动态规划中一个阶段的决策和转移，看成定义在状态空间上的函数，那么这种方法相当于计算转移函数的复合函数，不妨将其定义为转移函数的积。（对于上文所提到的动态规划，其转移函数的定义较复杂，详见附录 1）由于其满足结合律，通过快速幂，它可以与矩阵乘法一样，达到 $O(Size^2\log N)$的复杂度。

显然 $Size=2^BM$ 时 $O(Size^2)$仍然无法承受，因为 2^B 实在很可观，必须把它从状态中去除。

假设数位集合只需要是某个 S 的子集，而不是恰好等于某个集合 S，那么可以删除这维状态并把转移中的 $d\in\{0, \dots, B-1\}$改成 $d\in S$，通过上面的方法就可以算出 $Ans'[S]=\text{sum}\{ Ans[T] \mid T$ 包含于 $S \}$。一旦知道所有 $Ans'[S]$，便可用容斥原理求出所有 $Ans[S]$。

三、 解题算法

标准解法主要分为两个阶段。

1. 枚举{0, 1, …, B-1}的所有子集 S。对于每个 S
   a) 设 $DP[n, m]$为长度为 n，数位集合为 S 的子集，模 M 为 m 的 B 进制数（首位可以为 0）的数目（模 10007）。
   b) 利用转移方程

      $DP[n', m']=\text{sum}\{ DP[n_1, m_1]DP[n_2, m_2] \mid B^{n2}m_1+m_2 \equiv m' (\text{mod } M) \}$ $(n_1+n_2=n')$

      （其中 $DP[1, m]=|(\{ d \in S \mid d \equiv m (\text{mod } M) \}|$）

      进行以转移函数快速幂优化的动态规划，求出所有 $DP[N-1, m]$。
   c) 枚举 m 和首位 $d \in S$ ($d \neq 0$)。若 $B^{N-1}d+m \equiv V (\text{mod } M)$，则将 $DP[N-1, m]$ 累加到 $Ans'[S]$中。
2. 再次枚举所有子集 S。对于每个 S，枚举 S 的所有真子集 T，将 $Ans[T]$从 $Ans'[S]$中减去，得到 $Ans[S]$。即

   $Ans[S]=Ans'[S]-\text{sum}\{ Ans[T] \mid T$ 真包含于 $S \}$

在第一阶段中，总共枚举了 2^B 次，每次复杂度为 $O(M^2 \log N)$。在第二阶段中，枚举所有集合的所有子集，复杂度为 $O(3^B)$。因此该算法的时间复杂度为 $O(2^B M^2 \log N+3^B)$。

四、 出题思路

本题主要考察选手对于动态规划加速和容斥原理的掌握程度。

笔者曾经做过一道有关倍增转移的动态规划的题目，并加以总结。为与更多的人分享这个巧妙的方法，笔者准备了此题，并通过容斥原理提高少许难度。

五、 数据生成方法与分数分布

为使分数分布均匀，笔者列出以下几种可能的解法，并分配期望得分。

编号	算法	复杂度	期望得分
1	搜索	$O(B^N)$	10
2	朴素动态规划	$O(2^B NM)$	30
3	以矩阵快速幂优化的状态压缩动态规划	$O(8^B M^3 \log N)$	50
4	以转移函数快速幂优化的状态压缩动态规划	$O(4^B M^2 \log N)$	70
5	以矩阵快速幂优化的动态规划+容斥原理	$O(2^B M^3 \log N+3^B)$	70
6	以转移函数快速幂优化的动态规划+容斥原理	$O(2^B M^2 \log N+3^B)$	100

针对不同算法，笔者为其生成了对应的测试数据，使之能够得到期望的分数。

测试数据	N	B	M	V	接受算法
1~2	随机整数 [2, 10]	3	11	随机整数 [0, M)	1, 2, 3, 4, 5, 6
3~6	随机整数 [2, 10^3]				2, 3, 4, 5, 6
7~10	随机整数 [2, 10^9]				3, 4, 5, 6
11			108		4, 6
12~14			110		
15		10	6		5, 6
16~18			7		
19			28		6
20			31		

注意在第 11、15、19 个测试数据中，gcd(*B*, *M*)>1，这是为了使某些特殊处理过的算法拿到更高的分数。因为一个 *N* 位的 *B* 进制数模 gcd(*B*, *M*)的值，只与它的末几位有关，那么在快速转移的动态规划的过程中，可以不考虑模 gcd(*B*, *M*)的状态，即令 *M*=*M*/gcd(*B*, *M*)。

六、 总结与扩展

本文提出了一道 NOI 难度的试题，笔者用快速转移的动态规划和容斥原理解决了它，并给出了一组有效的测试数据来评估选手的解法。

本文还提到了动态规划快速转移。动态规划通过提炼状态的特征值，减少重叠子问题的计算次数。而动态规划快速转移还要通过提炼转移的特征值，实现一次多步转移。矩阵快速幂作为它的一个特例，是以矩阵的形式来表示转移函数的。总而言之，动态规划快速转移通过表示与求出转移函数的积，并减少冗余计算，达到了加速转移的目的，为优化动态规划提供了一种新思路。

参考文献

1. Thomas H.Cormen, Charles E.Leiserson, Ronald L.Rivest, Clifford Stein《算法导论》第 2 版机械工业出版社
2. 刘汝佳，黄亮《算法艺术与信息学竞赛》清华大学出版社
3. 刘汝佳，周源，周戈林《〈算法艺术与信息学竞赛〉学习指导》

感谢

感谢余姚中学的冯一同学和杭州二中的徐天一同学的帮助。

附录

1. “解题思路”中转移函数和转移函数的积的定义

转移函数由二元组(n, T)决定，它将 DP 映射到 DP'，满足

$DP'[n_1+n, s', m']=\text{sum}\{ DP[n_1, s_1, m_1] T_{s2, m2} \mid s_1 \cup s_2=s', B^n m_1+m_2 \equiv m' (\text{mod } M) \}$。

两个转移函数 f, g（分别由(n_1, T_1)和(n_2, T_2)决定）的积 $f \times g$ 也是转移函数，它由二元组(n', T')决定，满足

$n'=n_1+n_2$, $T_{s', m'}=\text{sum}\{ T_{s1, m1} T_{s2, m2} \mid s_1 \cup s_2=s', B^{n2} m_1+m_2 \equiv m' (\text{mod } M) \}$。

2. 标准解法的 C++程序代码

std.cpp

3. 测试数据


data.rar