《夹娃娃》命题报告

广州市第六中学 刘丹清

1 试题

1.1 题目描述

小 L 和小 J 去夹娃娃。街上一共有 n 家娃娃店，第 i 家娃娃店有 a_i 台娃娃机。一台娃娃机里只有一种娃娃，所有娃娃机里的娃娃都互不相同。在开始夹娃娃之前，小 L 和小 J 已经预估了每一台娃娃机的难度，对于第 i 家店里的第 j 台娃娃机，需要花**恰好** b_{ij} 元夹到一个娃娃，并且一共只有 c_{ij} 个娃娃。

小 L 和小 J 一共去了 q 次。第 i 次去的时候，一共带了 m_i 块钱。小 J 突然很喜欢某些娃娃店，想要在这些店里**各**花至少 k_i 块钱。小 L 想知道一共有多少种夹娃娃的方式（不一定要把钱全部花完），对 p 取模，其中 p 为 998244353 或 10^9+7。

两种方式不同当且仅当至少一种娃娃在两个方式中被夹到的数量不同。询问之间相互独立，即每次去夹娃娃时数量都会恢复初始状态。

1.2 输入格式

第一行三个整数 n, q, p 表示娃娃店数量，询问次数和模数。

接下来 n 行，每行第一个整数 a_i，然后 a_i 对整数 b_{ij} 和 c_{ij} 描述每台娃娃机。

接下来 q 行，每行一个 01 串和两个整数 m_i, k_i，串的第 j 个字符为 1 表示小 L 突然很喜欢第 j 家娃娃店。

1.3 输出格式

一共 q 行，每行一个整数表示答案对 p 取模的结果。

1.4 样例输入 1

```
4 6 998244353
1 2 1
2 3 4 5 6
2 2 2 1 3
2 9 9 13 14
0101 12 1
1010 4 1
1111 100 10
0000 5 500
0110 10 2
0101 200 40
```

1.5 样例输出 1

```
1
3
0
22
40
1950
```

1.6 数据范围

对于所有数据，$1 \le n \le 15$，$1 \le m_i, k_i, a_i, b_{ij}, c_{ij} \le 520$，$1 \le q \le 52099$。

子任务	$n \le$	$p =$	特殊限制	分值
1	2	998244353	无	3
2	4			13
3	8			14
4	15		$a_i = 1$	10
5			$m \le 250$	25
6			无	15
7		$10^9 + 7$		20

1.7 时空限制

时间限制：3s

空间限制：1GB

2 题目大意 & 初步分析

记号约定：以下的 m 均指 520，m_i 为题目给定的钱数。S 为小 J 喜欢的商店表示的二进制数。M 表示点值表示法的长度。

2.1 题目大意

给出 n 组多重背包，q 次询问舍弃掉其中一部分背包的前 k_i 个位置后进行卷积的结果在 m_i 处的前缀和。

2.2 初步分析

除了子任务 4，需要对于每个娃娃店预处理出多重背包。

令当前背包 A_i 表示花恰好 i 块钱的方案数。加入一组价格为 b，数量为 c 的物品时，令 A'_i 表示新的背包，则 $A'_i = A_i + A_{i-b} + A_{i-b*2} + \cdots + A_{i-b*c}$。使用模 b 意义下的前缀和即可 $O(m)$ 计算出 A'。实现时只需正反两次循环，无需使用辅助数组。

对于 $p = 998244353$ 的子任务，直接卷积可以做到 $O(m \log mnq)$ 的复杂度，无法获得分数。

3 基于折半的做法

注意到我们只需要求结果在 m_i 处的前缀和，而无需知道结果背包的每一位，可以省去一次卷积。

3.1 算法一：$n \leq 2$

对于两个背包的卷积，可以对第一个背包做前缀和后枚举第二个背包的下标，以 $O(m)$ 的复杂度求出结果在 m_i 处的前缀和。期望得分 3 分。

3.2 算法二：$n \leq 8$

考虑折半，对于左右两边的娃娃店，分别预处理出每一种情况 (S, k) 对应的背包。

令 $f_{i,j,k,l}$ 表示前 i 位的 S 为 j，至少花的钱数为 k 时的背包。直接进行递推，预处理半边的复杂度为 $O(2^{\frac{n}{2}+1} m \times m \log m)$，由于常数很大，期望得分 16 分。

对于子任务 3，考虑卷积时压缩位数。令 $ppc = \text{popcount}(S)$ 为小 J 喜欢的商店数量，则对于 $k \times ppc > m$ 的询问，答案显然为 0；否则，可以以 $m - k \times ppc$ 为位数进行卷积。使用该性质剪枝后可以通过子任务 3，期望得分 30 分。

4 基于生成函数的做法

对于子任务 4，询问中第 i 家娃娃店的背包的形式幂级数形如 $\dfrac{x^{A_i} - x^{B_i}}{1 - x^{C_i}}$，并且分母不变。

4.1 算法三：

考虑预处理分母的积 $\prod\limits_i \dfrac{1}{1 - x^{C_i}}$，询问时求出分子的积然后使用算法一快速求出答案。直接将 n 个式子每次以 $O(m)$ 的复杂度乘起来，总复杂度为 $O(qnm)$，有可能通过子任务 4。

优化：暴力求出前 10 个分子一共产生的 2^{10} 个单项式，再对最后 5 个分子 $O(m)$ 乘，可以通过子任务 4，期望得分 10 分。（实际上并没有写这个部分分）

前 4 个子任务思路较为常规，一共可以获得 40 分。

5 基于点值表示法的做法

根据前面的观察，我们即无法预处理折半后的背包，又无法承受每次询问做一次卷积的复杂度。这启发我们思考卷积的核心思想：将两个背包放在点值表示法下线性完成乘法。

5.1 算法四：$p = 998244353$

我们发现，如果我们能够求出结果背包 DFT 后的点值表示法，同样可以线性求出答案。

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_n^1 & \omega_n^2 & \omega_n^3 & \cdots & \omega_n^{n-1} \\ 1 & \omega_n^2 & \omega_n^4 & \omega_n^6 & \cdots & \omega_n^{2(n-1)} \\ 1 & \omega_n^3 & \omega_n^6 & \omega_n^9 & \cdots & \omega_n^{3(n-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_n^{n-1} & \omega_n^{2(n-1)} & \omega_n^{3(n-1)} & \cdots & \omega_n^{(n-1)^2} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_{n-1} \end{bmatrix}$$

[1]

IDFT 的作用相当于乘以图中矩阵的逆，等于 ω 的逆的范德蒙得矩阵乘上 M 的逆，只需要预处理出矩阵系数并对每行做前缀和，即可算出点值表示法下每一位对答案的贡献。

先考虑容斥。枚举 S 的子集 T，强制 T 以内的娃娃店花钱均 $< k$，乘上容斥系数 $(-1)^{|T|}$，加起来得到答案。令 $g_{S,k,l}$ 表示只考虑 S 内部的娃娃店花钱均 $< k$ 时的背包，结合 $k \times ppc \le m$ 的性质，这些背包的长度均不超过 m，我们可以只存储它们的点值表示法，并且将两个背包卷积的复杂度从 $O(m \log m)$ 降为 $O(M)$。这部分预处理的复杂度为 $O(2^n mM)$，可以将 $k \times ppc > m$ 的部分舍弃减小复杂度，具体为 $M \sum_{c=1}^{n} \binom{n}{c} \frac{m}{c}$。

另外，求出只考虑 S 内部的集合且没有额外限制时的背包 $f_{S,l}$，同样在点值表示法下存储。这部分预处理的复杂度为 $O(2^n m \log m)$。

结合 f 与 g，我们可以得到每个被容斥的子集 T 的背包。由于询问量太大，不能枚举子集，所以这部分需要用高维前缀和预处理。于是，我们得到了一个 $O(n2^n mM)$ 的做法来预处理出所有情况的点值表示法。

接下来，考虑分块。令块长为 L，块数为 B，为了保证点值表示法不溢出，需要将 M 开到最大可能出现的位数。所有被容斥的位置的次数为 $(k-1) \times ppc$，没有被容斥的次数为 $B \times m$，所以总共的位数为

$$B \times m + (k-1) \times ppc + 1 \le B \times (m+1)$$

总复杂度为 $O(B * (L2^L mM + qM))$，其中左式带小常数。取 $L = 5, B = 3, M = 1024$ 即可轻松通过子任务 5。

子任务 6 可能需要注意一些常数。取 $M = 4096$ 时，有大量空闲的位数，但是取 $M = 2048$ 又会刚好不够。实际上只有当 $k \times ppc$ 非常接近 m 时才会出现位数不够用的情况，这时我们以 $m - k \times ppc$ 为长度跑 n 次卷积即可。

期望得分 80 分。

5.2 算法五：$p = 10^9 + 7$

实际上，我们不一定要用原根处的点值表示法，可以改为存储 $0, 1, \cdots, M-1$ 处的点值，并且复杂度瓶颈不在于卷积。求答案时，仍然是一个线性变换，其系数可以使用拉格朗日插值 $O(M^2)$ 或 $O(Mm)$ 预处理。

取 $L = 5, B = 3, M = 2080$，期望得分 100 分。

接下来是卡常环节。首先，对所有询问按 k 离线以减小空间复杂度。求答案时需要计算 M 次四个数乘积的和，使用 `___uint128_t` 存储答案，每隔 $2^8 = 256$ 位取一次模。同样的，在进行暴力卷积等计算时也可以使用该优化。在老爷机上跑了 1.2s，预计不特意优化也可以通过（但是如果询问时不优化取模次数似乎不能过）。

代码中，为了让编译器预处理取模优化，可以使用 `template<int mod>` 来让编译器自动优化而无需复制一份代码使用不同的模数 [2]。当然，也可以自己实现 [3]。

参考文献

[1] FFT，NTT 基础详解 https://zhuanlan.zhihu.com/p/40505277

[2] 感谢宋佳兴同学的分享

[3] Barrett reduction 取模优化 http://platelet.top/misc/