《欧伊昔》解题报告

福建师范大学附属中学 魏忠浩

目录

1 题目大意 **2**

2 数据范围 **2**

3 解题过程 **2**

3.1 算法 1 3
3.2 算法 2 3
3.3 算法 3 3
3.4 算法 4 5
3.5 算法 5 5
3.6 算法 6 5
3.7 算法 7 6
3.8 算法 8 6

4 致谢 **7**

5 参考资料 **8**

1 题目大意

定义一个 3×3 的随机运算表 op，数组下标和值域为 $[0, 2]$ 内的整数。
随机方式是在满足数据限制的所有可能方式内随机选择一种。

定义 $i \operatorname{op} j$ 的值为三进制下按位做 op 操作的结果。
即答案的三进制第 k 位（$0 \le k < n$）是 op_{i_k, j_k}，i_k, j_k 这里指其三进制第 k 位。

给出 A, B，需要求 C:

$$C_x = \sum_{i \operatorname{op} j = x} A_i B_j$$

A_i, B_i 在 $[0, 9]$ 的整数内选取。

2 数据范围

`Subtask 1 (5 pts)`: $\mathrm{op}_{i,j} = i + j \bmod 3$;
`Subtask 2 (5 pts)`: $\mathrm{op}_{i,j} = \mathrm{mex}(i, j)$，$\mathrm{mex}(i, j)$ 表示最小的不等于 i 或 j 的非负整数;
`Subtask 3 (20 pts)`: $\mathrm{op}_{i,j} \in \{0, 1\}$, 且任意选择两行，要么这两行相同，要么这两行每一位都不同;
`Subtask 4 (30 pts)`: $\mathrm{op}_{i,j} \in \{0, 1\}$;
`Subtask 5 (10 pts)`: $n \le 9$;
`Subtask 6 (10 pts)`: $n = 10$，依赖 `Subtask 5`;
`Subtask 7 (20 pts)`: $n = 11$，依赖 `Subtask 6`。

对于 100% 的数据，保证 $1 \le n \le 11$，$\mathrm{op}_{i,j}$ 在子任务要求下均匀随机地从所有可能方案中选择一种。
保证 $0 \le A_i, B_i \le 10$ 且为整数、除最后一组外每组子任务恰有 5 组测试数据，最后一组子任务有 10 组。

时间限制 3s，空间限制 1024MB。

3 解题过程

这里记 a, b 是题目所述的输入序列 A, B，c 是输出的答案序列 C。

3.1 算法 1

我会 FWT!

分析快速沃尔什变换（FWT），二进制下的异或卷积实际上使用了长为二的序列 DFT 和 IDFT。
三进制下就是长度为三的序列 DFT 和 IDFT。用复数存储中间结果可以做到 $O(3^n n)$，亦可选取恰当的大模数。

3.2 算法 2

我会暴力!

如果直接枚举两个数复杂度 $O(9^n n)$，不能得分。
如果按照某个顺序同时枚举两个数的某一位就可以做到 $O(9^n)$，实现不差可以通过 Subtask 5。

3.3 算法 3

考虑如何解决 $\mathrm{op}_{i,j} = \mathrm{mex}(i,j)$。
尝试使用算法 1 的修改版本。

由于高维“卷积”每一维间独立，所以**不妨考虑 $n = 1$ 的时候如何做**。
接下来所有的具体分析都**基于 $n = 1$**，但不破坏算法的结构使得其可以向高维拓展。

为什么不能使用对于每一维暴力处理?
暴力如果套到这类算法上，$a_{0,1,2}$ 和 $b_{0,1,2}$ 分成了 9 份 $a_i b_i$，最后重新构建成 $c_{0,1,2}$。
与之相似的还有 Karatsuba 算法和直接高精度乘法、Strassen 算法和直接矩阵乘法之间的区别。

这启发我们需要尽可能减少中间量（变换后的结果和逆变换的输入）的个数，这里记最终逆变换时的输入为 d_i，个数为 r
观察上述算法，变量间的乘法只在中间量 d_i 产生，变换和逆变换的过程只是线性变换。
则需要构造 $r \times 3$ 的矩阵 A, B, C，使得 $d_i = (\sum A_{i,j} a_j)(\sum B_{i,j} b_j)$，$c_i = \sum C_{j,i} d_j$。

题目要求的:

$$\begin{aligned} c_0 &= a_1 b_2 + a_2 b_1 + a_1 b_1 + a_2 b_2 \\ c_1 &= a_0 b_0 + a_0 b_2 + a_2 b_0 \\ c_2 &= a_0 b_1 + a_1 b_0 \end{aligned}$$

对于 $r=3$ 的构造非常困难，可以认为基本不可能。
但可以找出 $r=4$ 的构造：

$$\begin{aligned} d_0 &= (a_0+a_1+a_2)(b_0+b_1+b_2) \\ d_1 &= (a_1+a_2)(b_1+b_2) \\ d_2 &= (a_0+a_2)(b_0+b_2) \\ d_3 &= a_2b_2 \end{aligned}$$

最后

$$\begin{aligned} c_0 &= d_1 \\ c_1 &= d_2-d_3 \\ c_2 &= d_0-c_0-c_1 \\ &= d_0-d_1-d_2+d_3 \end{aligned}$$

复杂度 $O(4^n)$，按照 Karatsuba 分治算法的写法会比普通的 FWT 写法好写。

特别的，使用分治乘法写法在 n 较小的时候（比如 $n\le 1$）可以暴力，可以降低常数。且空间复杂度能够做到 $O(3^n)$。

算法 3.5（另一种“更好”的构造思路）

算法 3 的约束 $d_i=(\sum A_{i,j}a_j)(\sum B_{i,j}b_j)$ 较为抽象，尝试找出较为形象的解释。
这相当于是一个**能被写成两个向量 A_i,B_i^{T} 相乘的 3×3 矩阵（秩一矩阵）** D_i。
那么若对于所有可能的 a,b 输入均能运行，就能通过矩阵 D_i 线性组合出 G_i，其中 $G_{i,j,k}=[\mathrm{op}_{j,k}=i]$。

在 mex 的子任务中，需要表示出：

$$\begin{bmatrix}0&0&0\\0&1&1\\0&1&1\end{bmatrix}\begin{bmatrix}1&0&1\\0&0&0\\1&0&0\end{bmatrix}\begin{bmatrix}0&1&0\\1&0&0\\0&0&0\end{bmatrix}$$

第一个矩阵是秩一矩阵。
第二个矩阵可以用两个秩一矩阵相减。
第三个用全一矩阵减去前两个。

具体来说，令：

$$D_0=\begin{bmatrix}0&0&0\\0&1&1\\0&1&1\end{bmatrix},D_1=\begin{bmatrix}1&0&1\\0&0&0\\1&0&1\end{bmatrix},D_2=\begin{bmatrix}0&0&0\\0&0&0\\0&0&1\end{bmatrix},D_3=\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

上述的 D_i 可以组合出 G_i，总共使用四个秩一矩阵。
也就是 $r=4$，且这里对应的 A,B,C 和前文所述一致。
此解法会有助于理解一些构造且易于说明，所以后文大部分解法都使用这类方法描述。

3.4 算法 4

发现二进制异或的运算表也满足 Subtask 3 性质，可以考虑拓展。
可以构造出：

$$\begin{aligned} A_{0,i} &= B_{0,i} = 1 \\ A_{1,i} &= 2[\mathrm{op}_{i,0} = \mathrm{op}_{0,0}] - 1 \\ B_{1,i} &= 2[\mathrm{op}_{0,i} = \mathrm{op}_{0,0}] - 1 \end{aligned}$$

其满足

$$(A_1 B_1^{\mathrm{T}})_{i,j} = 2[\mathrm{op}_{i,j} = \mathrm{op}_{0,0}] - 1$$

可以通过 d_0, d_1 相加或相减后除二得出 c_0 和 c_1，受 $\mathrm{op}_{0,0}$ 取值影响，C 略有不同。
用算法 3.5 来讲，分别构造一个全 1 矩阵，一个是用 -1 和 1 区分 0 和 1 的矩阵。所需要的 A,B,C 容易得出，两个矩阵相加和相减就能表示 G_0, G_1。

复杂度 $O(3^n)$。

亦可发现两个输入的 3^n 其实可以分别被压缩成 2^n，具体来说当 $n=1$ 时把 $\{i\}$ 视作 0，$\{1,2,3\}-\{i\}$ 视作 1。后面做法和二进制异或差不多。其本质和上述解法一致。

3.5 算法 5

我会 $\mathrm{op}_{i,j} < 2$!
可以先构造一个全一矩阵，然后对于每一列单独用一个中间变量 d_i 表示，复杂度 $O(4^n)$。

举例如果运算表为 $\begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}$，构造

$$D_0 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} D_1 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} D_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix} D_3 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

3.6 算法 6

我会乱搞!

可以先构造一个全一矩阵 D_0。
选择一个 o，对于 $\mathrm{op}_{i,j} \neq o$ 的每个单点 (i,j) 单独开一个 d_x 存储。
而 o 的部分用全一矩阵减去其余的 D_x 就能得到。

既然可以随意选择，那选择出现次数最大的 o，可以做到 $r = 7$。
复杂度 $O(7^n)$，因为上界出现概率较大，只能过 $n \leq 10$。（也过不去 $i + j \bmod 3$ 的点）
并且这时候的 A, B, C 较为稀疏且特殊，若实现较差需对一些矩阵乘法部分进行特化以减小常数。

3.7 算法 7

我会再乱搞!

对于每一个 (i,j) 都开一个 d_x 实在太浪费了。
只需要再少掉一个 d_x，就可以做到 $O(6^n)$。

最简单的方法是找到两个在一行或一列的、op 相同且不等于 o 的拼在一起。

这种算法应该在随机情况下非常容易做到 $r = 6$。
因为，如果每个答案出现 3 次、相同答案不在同一行或者同一列，这是个拉丁方。
拉丁方方案数不到 20，出现的概率小于千分之一。
在最后五组数据不出现的概率大于 99%，对运算表无限制的数据里不出现的概率为 95% 以上，因此几乎不可能出现。此时复杂度是 $O(6^n)$ 可以通过几乎所有数据。（除了 $i + j \bmod 3$ 的点）

对于 Subtask 1 的情况使用特殊的算法，就能通过此题。
对于三阶拉丁方的 corner case，可以通过对输入和答案的每一位进行长度为 3 的置换来归纳到 Subtask 1。
于是解决 corner case 后此算法可以通过这题的所有数据，且不依赖随机生成。

如果直接采用对于每个 $\mathrm{op} = x$ 的区域直接一行一行覆盖、或者一列一列覆盖（同时去掉空列空行）也能通过非 corner case 部分。

可能存在一些通过枚举可能性较大的 D_i，然后判断是否存在 C_i，做到更优的 r 的算法，因为直至目前 A_i, B_i 取值范围为 $\{1, 0, -1\}$（除了 $i + j \bmod 3$，但可以找到一组 r 稍大的解也满足此性质）。

3.8 算法 8

这就结束了吗?
进一步分析算法流程，不妨只考虑一组 (i,j,k) 的贡献，也就是 $a_i, b_j \to c_k$ 的贡献。
此时有 $\forall i,j,k, \sum A_{x,i}B_{x,j}C_{x,k} = [\mathrm{op}_{i,j} = k]$，等价于如下三阶张量的 CP 分解：

$$X_{i,j,k} = [\mathrm{op}_{i,j} = k]$$

$3 \times 3 \times 3$ 张量 CP 分解的定义是找出 $r \times 3$ 的矩阵 A, B, C，使得 $X_{i,j,k} = \sum_t A_{t,i} B_{t,j} C_{t,k}$。其中最小的存在分解的 r 叫做 X 的张量秩。

但三阶张量的 CP 分解（分解成若干一秩张量的和）目前是 NP-hard。尽管如此，其近似解仍有许多在实际大量数据里表现优秀的做法。容易想出来的启发式做法诸如根据误差每次向较为正确的方向调整，不断执行等等。这可能会出现类梯度下降法的算法，重复跑几轮就会发现当 r 取 4 或 5 时误差大部分会迅速向 0 收敛。

实际上，梯度下降法及其各种类似写法都可以在几轮 3×10^4 次的迭代后获得非常优秀的误差（误差矩阵范数在 10^{-15} 以内）。这类稀疏 01 张量采用交替最小二乘法可能更好，但没有尝试。所以迭代过程正常情况下不会是此题的瓶颈。

r 对这题的复杂度影响甚大，那么 r 在大部分情况能取多少？虽然本题可以贴着时间取 $r = 5$ 甚至 6，但不一定最优。
资料 3 的 Table 1 说明 $3 \times 3 \times 3$ 张量的典型秩（出现概率非 0）在实数域下为 5。
在这题数据生成的范围内，通过蒙特卡罗方法或者穷举，可以得出本题题目限制下 $r = 4$ 几乎可以分解成功。但若采用 $r = 5$ 仍然可以通过。

如果想要不依赖于生成数据的随机性，需要考虑实数域的张量最大秩。
在参考资料里有相应的文章，但在这题并不重要。

假设分解完后的向量组为 x_i, y_i, z_i。
令 $A_{i,j} = (x_i)_j, B_{i,j} = (y_i)_j, C_{i,j} = (z_j)_i$ 即可。
由于使用近似算法，所以会有误差，假设误差为 δ，在经过变换和逆变换后，误差会增大至 $3^{2n+C}\delta$，C 是一个不太大的常数。
所以为了让最后的误差不超过 10^{-3}，可以让 δ 和误差矩阵范数控制在 10^{-14} 以内。
因此需要用 `double` 或 `long double` 进行运算。除去生成 A, B, C 外的复杂度为 $O(4^n)$。

一份标准程序的 CP 分解部分使用了梯度下降法，但调高了学习率（变化幅度），并且若收敛速度过慢或不收敛（出现 `inf` 或 `nan`），重新运行直至找到一组解。这在远低于题目限制的时间内就可以获得一组精度足够的解。在 OJ 上最大时间约为 300 ms。

4 致谢

感谢中国计算机学会提供本次交流的平台。
特别感谢任舍予学长和林圣涵学长给予我的启发。
感谢黄佳旭同学和我一起讨论此问题。

5 参考资料

位运算卷积 (FWT) & 集合幂级数 - 洛谷专栏
快速沃尔什变换 - OI Wiki
Tensor Decompositions and Applications
Simplicity and Typical Rank Results for Three-Way Arrays | Psychometrika
On Kruskal’s theorem that every 3 Œ 3 Œ 3 array has rank at most 5 - ScienceDirect