《路径计数》解题报告

长郡中学 周桓毅

2024 年 10 月 6 日

目录

1 题目描述

有一个 n 行 m 列的网格，网格上共有 $(n+1)\times(m+1)$ 个格点，其中第 x 行第 y 列的格点用一个二元组 (x,y) 表示 (格点的行与列均从 0 开始编号)。

初始时网格没有边，现在依次加入 $(3m+1)n$ 条有向边：

1. 对于 $0\le i\le n-1, 0\le j\le m-1$ 加入 A_j 条本质不同的从 (i,j) 到 $(i+1,j+1)$ 的有向边。

2. 对于 $0\le i\le n-1, 0\le j\le m$ 加入 B_i+C_j 条本质不同的从 (i,j) 到 $(i+1,j)$ 的有向边。

3. 对于 $0\le i\le n-1, 1\le j\le m$ 加入 D_j 条本质不同的从 (i,j) 到 $(i+1,j-1)$ 的有向边。

现在令对于满足 $0\le x\le n, 0\le y\le m$ 的整数 x,y，定义 $W(x,y)$ 表示 $(0,0)$ 到 (x,y) 有多少条本质不同的路径，不难证明路径的个数是有限的。现在你需要求出 $\sum_{i=0}^{n}\sum_{j=0}^{m}W(i,j)E_iF_j \bmod p$ 的结果。

2 输入格式

第一行输入三个正整数 c,n,m,p，第一个数表示子任务编号 (特别的，样例中的 c 表示该样例的满足的限制与第 c 个子任务所满足的限制相同)，第二个数与第三个数描述网格的大小，第四个数表示答案需要取模的模数。

第二行输入 m 个数，其中第 i 个数表示 A_{i-1} 的取值。

第三行输入 n 个数，其中第 i 个数表示 B_{i-1} 的取值。

第四行输入 $m+1$ 个数，其中第 i 个数表示 C_{i-1} 的取值。

第五行输入 m 个数，其中第 i 个数表示 D_i 的取值。

第六行输入 $n+1$ 个数，其中第 i 个数表示 E_{i-1} 的取值。

最后一行输入 $m+1$ 个数，其中第 i 个数表示 F_{i-1} 的取值。

3 输出格式

输出一行一个整数表示 $\sum_{i=0}^{n}\sum_{j=0}^{m}W(i,j)E_iF_j \bmod p$ 的结果。

4 数据范围

对于所有数据，保证 $1 \le n, m \le 2\times10^5, 1 \le p \le 10^9, 0 \le A_i, B_i, C_i, D_i, E_i, F_i < p$，不保证 p 为质数，但对于 $p \neq 998244353$ 的数据满足 $1 \le n, m \le 10^5$。

子任务编号	子任务分值	$n \le$	$m \le$	A_i	B_i	C_i	D_i	E_i	F_i	$p = 998244353$
1	3	5000	5000	-	-	-	-	-	-	是
2	5	2×10^5	2×10^5	-	$=0$	$=1$	$=0$	-	-	是
3	8	2×10^5	2×10^5	-	-	$=0$	$=0$	-	-	是
4	8	2×10^5	2×10^5	-	$=0$	-	$=0$	-	-	是
5	5	2×10^5	2×10^5	-	-	-	$=0$	-	-	是
6	15	2×10^5	2×10^5	-	-	-	-	-	$=[i=m]$	是
7	16	2×10^5	20000	-	-	-	-	-	-	是
8	16	2×10^5	2×10^5	-	-	-	-	-	有且仅有一个位置非 0	是
9	9	2×10^5	2×10^5	-	-	-	-	-	-	是
10	15	10^5	10^5	-	-	-	-	-	-	否

表格中的 - 表示无特殊性质。

时间限制：9s，空间限制：1024MB。

5 解题过程

5.1 算法一

令 $dp_{x,y}$ 表示 $(0,0)$ 到 (x,y) 的路径条数，直接按照题意 dp 即可。

时间复杂度为 $O(nm)$，可以通过子任务 1 得到 3 分。

在下文关于复杂度的描述中，若无特殊说明，我们将认为 n 与 m 同阶。

5.2 算法二

考虑 $B_i = D_i = 0, C_i = 1$ 的情况，此时可以考虑将 x 看作一个时间维，将其视作在一条横线上从 0 开始 x 步走到 y，由于路径并不会下降，路径一定是顺着从 0 走到 y 再添加 $x-y$ 个自环，不难得到 $W(x,y) = \prod_{i=0}^{y-1} A_i \binom{x-1}{y-1}$，令 $tA_i = \prod_{j=0}^{i-1} A_j$，则有 $\sum_{i=0}^{n}\sum_{j=0}^{\min(i,m)} W(i,j)E_iF_j = \sum_{i=0}^{n}(i-1)!\sum_{j=0}^{\min(i,m)} \frac{tA_jF_j}{(j-1)!}\frac{1}{(i-j)!}$，这是一个卷积的形式，可以使用 NTT 做到 $O(n\log n)$。

时间复杂度为 $O(n\log n)$，可以通过子任务 2，结合算法一可以获得 8 分。

5.3 算法三

考虑 $C_i = D_i = 0$ 的情况，此时走出的自环的代价与时间有关，考虑将其看作时间的舍去，舍去一个时间 i 会产生 B_i 的贡献，那么 $W(x,y) = K(x, x-y)\prod_{i=0}^{y-1} A_i$，其中 $K(x,y)$ 为在 B $[0, x-1]$ 的部分中选择 y 个元素的乘积和。考虑设计一个元 z，构造一个多项式 $P(z) = \sum_{i=0}^{m} F_i t A_i z^i$，如果要求 $\sum_{i=0}^{x} W(x,i)F_i$，实际上相当于对于 $[0, x-1]$ 的元素 i，不选择即为乘 z^{-1}，选择即为乘 B_i，最后求 $[z^0]P(z)$ 的值。

考虑对其建立一颗线段树，对于区间 $[l,r]$ 维护在处理完 $[0, l-1]$ 的操作指数在 $[0, r-l]$ 部分截断后得到的多项式 $p[l,r]$，并再之后从线段树上往下递归求出每一个节点所对应的多项式，具体的令 $mid = \lfloor \frac{l+r}{2} \rfloor$ 则有 $p[l, mid] = p[l,r], p[mid+1, r] = p[l,r]\prod_{i=l}^{mid}(z^{-1} + B_i)$，并在操作之后截断，对于 $\prod_{i=l}^{mid}(z^{-1} + B_i)$ 的部分需要我们一开始在线段树上对每一个节点 $[l,r]$ 求出 $w[l,r] = \prod_{i=l}^{r}(z^{-1} + B_i)$，可以自底而上合并得到。

线段树的每一层，若长度为 len，则需要进行一次长度为 len 的多项式乘法，根据主定理可以得到复杂度为 $O(n \log^2 n)$，可以通过子任务 3，结合算法一二可以获得 16 分。

5.4 算法四

考虑 $B_i = D_i = 0$ 的情况，如果将式子写为 $\sum_{i=0}^{n} E_i \sum_{j=0}^{\min(i,m)} t A_j [x^{i-j}] \frac{1}{\prod_{j=0}^{i}(1-C_j x)} F_j$，实际上我们仅需求出 $\sum_{i=0}^{m} t A_i F_i \frac{x^i}{\prod_{j=0}^{i}(1-C_j x)}$ 即可，如果一开始乘上 $\prod_{i=0}^{m}(1 - C_i x)$ 那么即为 $\sum_{i=0}^{m} t A_i F_i x^i \prod_{j=i+1}^{m}(1 - C_j x)$。仍然考虑建立线段树，每一个节点 $[l,r]$ 维护 $p[l,r] = \sum_{i=l}^{r} t A_i F_i x^{i-l} \prod_{j=i+1}^{r}(1 - C_j x)$，多项式的次数不超过 $r-l$，转移即 $p[l,r] = p[l, mid]\prod_{i=mid+1}^{r}(1 - C_i x) + p[mid+1, r] x^{mid-l+1}$。由于要求如 $w[l,r] = \prod_{i=l}^{r}(1 - C_i x)$ 的形式，在线段树上同样可以预处理好，最终乘上 $\prod_{i=0}^{m}(1 - C_i x)$ 的逆元即可得到答案。

复杂度为 $O(n \log^2 n)$，可以通过子任务 2, 4，结合算法一三可以得到 24 分，如果将得到的多项式代入算法三的思路可以得到一个仅需满足 $D_i = 0$ 的做法，可以通过子任务 2, 3, 4, 5，结合算法一可以得到 29 分。

5.5 算法五

考虑 $F_i = [i = m]$ 的情况，由于加入了 i 向 $i-1$ 的边，路径形态会十分复杂，不便分析。首先 B_i 可以根据算法三的想法将其去除，此时如果换个角度考虑将原图看作一个矩阵，仍将 x 看作时间维，那么相当于有一个 $(m+1)\times(m+1)$ 的矩阵 A 和若干个 k 要求 $(A^k)_{0,m}$，注意到当 $k<m$ 时 $(A^k)_{0,m}=0$，$(A^m)_{0,m}=tA_m$，对于矩阵大小规模的幂我们都可以快速求出，这启发我们利用特征多项式的性质：对于 A 的特征多项式 F 有 $F(A)=0$。实际上这个结论是非常有效的，由于特征多项式的次数为 m，对于 $k>m$ 的 k 我们总可以将 A^k 替换为 $-\sum_{i=0}^{m-1} f_{m-i}A^{k-i}$ ，于是我们可以得到答案的递推式 $ans_x = tA_x - \sum_{i=0}^{m-1} f_{m-i}ans_{x-i}$。

现在的问题在于如何快速求解特征多项式，如果求出了特征多项式，直接求逆即可做到 $O(n\log n)$ 求 ans。注意到该矩阵实际上是一个上海森堡矩阵，可以直接利用行列式的定义求解，相当于一个 n 个点的图，可以用经过 i 的自环覆盖，有 $z-C_i$ 的贡献，也可以用 $i,i+1$ 间的二元环覆盖，有 $-A_iD_{i+1}$ 的贡献，要求最终覆盖掉所有点，考虑将其看作一个对 1×2 的多项式向量的线性变换，每次相当于乘上 2×2 的矩阵 $\begin{bmatrix}0 & -A_{i-1}D_i\\ 1 & z-C_i\end{bmatrix}$，使用分治来优化求乘积即可做到 $O(n\log^2 n)$。

可以通过子任务 6，结合算法一四可以得到 44 分。

5.6 算法六

考虑 $m\le 20000$ 的情况，先采用 $O(m^2)$ 的 dp 求出 ans 在 $[0,m]$ 处的值，再求逆即可得到 ans 在 $[0,n]$ 位置的值。

实际上如果注意到 ans 具有 $O(m)$ 的线性递推式，可以使用 BM 算法 $O(m^2)$ 求出递推式，同样可以通过该档子任务。

可以通过子任务 7，结合算法一四五可以得到 60 分。

5.7 算法七

考虑 F_i 有且仅有一个位置非零的情况，由于终点不一定是 m 了，这也就意味着对于终点 t，在 $[t+1,m]$ 的部分不一定为 0，仍要能快速算出，

这样就并没有使得问题得到简化。但实际上可以发现由于路径要从 0 之后要再回到 t，由于路径是连续的，中间有很多位置是不会走到的，如要 k 步从 0 走到 t，令途中 h，则有 $h+h-t\le k$，那么实际上 $h\le\lfloor\frac{t+k}{2}\rfloor$，这也就意味着如果要求 k 步的答案，仅保留 $[0,\lfloor\frac{t+k}{2}\rfloor]$ 的部分的特征多项式是足够的！此时可以通过建立递推式将 k 变为若干个 $[0,\lfloor\frac{t+k}{2}\rfloor]$ 的子问题，每一次迭代都会使 $k-t$ 减半，$O(\log)$ 轮后 k 就变成了 t，这样就好办了。

但是每一次 k 迭代为 $[0,\lfloor\frac{t+k}{2}\rfloor]$ 时变成了太多的子问题，不可以对每一个子问题都做，实际上可以考虑用倍增尽量充分的保留信息，初始时令 $k=t$，每次通过 $[k,t]$ 部分的结果，来推出 $[k,2t-k+1]$ 的结果，特征多项式仅需用到 $\lfloor\frac{2t-k+1+k}{2}\rfloor=t$ 的部分，可以求逆得到 $[t+1,2t-k+1]$ 部分的结果。由于每次需要 $O(n\log^2 n)$ 求特征多项式，而倍增时长度和为 $n+\frac{n}{2}+\frac{n}{4}+...=O(n)$，复杂度为 $O(n\log^2 n)$。

可以通过子任务 $6,8$，结合算法一四六可以得到 76 分。

5.8 算法八

当模数为质数的时候，由于求逆元是良定义的，存在一种简单的维护消元的做法。考虑设一些下标为位置指数为步数的多项式组 G，有 $[x^j]G_i$ 表示从 i 出发走 j 步所有方案走到的位置的 F 之和，则有 $G_i=A_ixG_{i+1}+D_ixG_{i-1}+C_ixG_i+F_i$，通过移项可以得到 $G_{i+1}=\frac{(1-C_ix)G_i-D_ixG_{i-1}-F_i}{A_ix}$。

令 $H_i=G_ix^i$ 则可以得到 $H_{i+1}=\frac{(1-C_ix)H_i-D_ix^2H_{i-1}-F_i}{A_i}$，如果令 $H_i=a_iH_0+b_i$，则有 $a_{i+1}=\frac{(1-C_ix)a_i-D_ix^2a_{i-1}}{A_i},b_{i+1}=\frac{(1-C_ix)b_i-D_ix^2b_{i-1}-F_i}{A_i}$，$a,b$ 都可以表示为一个多项式的线性变换的形式，可以统一在一个 3×3 的矩阵中，通过分治维护矩阵乘法可以 $O(n\log^2 n)$ 求解，由于有 $H_{m+1}=a_{m+1}H_0+b_{m+1}=0$，则 $H_0=-\frac{b_{m+1}}{a_{m+1}}$，使用求逆可以快速求出 $G_0=H_0$。

但是可能存在 $A_i=0$ 的情况，注意到令第一个满足 $A_i=0$ 的位置为 d，则 $\ge d$ 的部分都不会被走到，可以直接去除，由于此时所有的 A 都具有逆元，求逆是可以定义的。

时间复杂度为 $O(n\log^2 n)$，可以通过子任务 $1,2,3,4,5,6,7,8,9$，可以得到 85 分。

5.9 算法九

由于模数不一定为质数，求逆无法直接定义，维护消元的做法扩展 $\bmod p^k$ 不方便处理，而且需要进行 excrt 合并。考虑直接顺序将走 i 步到达的所有位置的带权和求出，令其为 ans_i，那么需要到 i 时维护出 $[0,i-1]$ 的特征多项式，而我们需要求的是 $[0,i-1]$ 的特征多项式与 $[0,i-1]$ 的答案的点积，这个看似是十分困难的，因为不同位置特征多项式的变换是一个乘矩阵的变换。

但实际上这仍然是可以维护的，考虑将特征多项式的 z 看作 z^{-1}，最后求多项式 $[z^0]$ 处的取值，考虑建立一颗线段树，对于每一个节点 $[l,r]$，我们需要求出 $[0,l-1]$ 的部分在作用 $[0,l-1]$ 的矩阵的线性变换所得到的结果，并截断仅保留指数在 $[0,r-l]$ 的部分，而在下传时左侧的已求出，先递归计算 $[l,mid]$ 求解，在得到 ans 在 $[l,mid]$ 的取值后为了计算 $[mid+1,r]$，我们需要知道 $[l,mid]$ 的部分对 $[mid+1,r]$ 部分的影响，这部分为 $\sum_{i=l}^{mid} ans_i z^i$ 作用 $[0,mid]$ 的线性变换，先将其化为 $\sum_{i=l}^{mid} ans_i z^i$ 作用 $[0,l-1]$ 的线性变换再与原先要下传的信息统一作用 $[l,mid]$ 的线性变换。

注意到求解 $\sum_{i=l}^{mid} ans_i z^i$ 作用 $[0,l-1]$ 的线性变换中我们需要将 $[l,mid]$ 变为 $[0,r-l]$,则有用的 $[0,l-1]$ 的线性变换的指标取值范围为 $[-mid,r-2l]$，而 $[0,l-1]$ 有效的部分为 $[-l,0]$，即我们仅需保留 $[-l,r-2l]$ 这个长度为 $r-l$ 的部分即可。将 z^{-1} 与 1 互换可以将其转化为保留 $[-(r-l),0]$ 的部分，也就是说对于每一个节点 $[l,r]$ 要维护 $[0,l-1]$ 的线性变换在一个长度为 $r-l$ 的区间的截断，这个是可以快速维护的。

在处理完后需要将 $[l,mid]$ 所产生的贡献与对 $[l,r]$ 的贡献合并后，需要作用 $[l,mid]$ 的线性变换后再将贡献下放给 $[mid+1,r]$，这就需要一开始在线段树上处理出每个节点的矩阵积用于查询 $[l,mid]$ 的线性变换的作用效果，注意的是每一次在下传标记时维护的都是一个 1×2 的向量。

时间复杂度为 $O(n\log^2 n)$，可以通过所有子任务，可以得到 100 分。

5.10 算法十

实际上由于路径并非特别复杂，可以支持合并，还存在一种分治维护路径信息的做法，对于线段树上的每一个区间 $[l,r]$，实际上是可以把 $[l,r]$ 内

的所有路径信息全部保留下来的。因为路径只会在 $[l,r]$ 中，根据 $[l,r]$ 中的特征多项式，我们仅需保留步数在 $[0,r-l]$ 之内的信息就可以将 $[r-l,\infty]$ 的部分推出来。

令当前考虑的区间为 $[l,r]$，对于 $r-l+1\geq 3$，令 $mid=\lfloor\frac{l+r}{2}\rfloor$，将其分为 $[l,mid-1]$ 与 $[mid+1,r]$ 两个子区间分别求出步数在 $[0,mid-l-1]$ 与在 $[0,r-mid]$ 的路径信息，并用特征多项式将两侧所有路径信息均补全至 $[0,r-l]$。令多项式 $G[l,r]_{0/1,0/1/2}$ 有 $[x^k]G[l,r]_{0/1,0/1/2}$ 表示从 l/r 出发走 k 步最终到达 l/r 的方案数或任意位置的 F 值和，如果一条路径没有经过 mid，那么可以直接由 $G[l,mid-1]$ 与 $G[mid+1,r]$ 转移而来，否则其一定是从出发点走到 mid 再走了若干条 mid 回到 mid 的路径后最终从 mid 到终点。

对于出发点到 mid 与 mid 到终点的路径都是比较容易由 $G[l,mid-1]$ 与 $G[mid+1,r]$ 的信息得来的，关键在于 mid 回到 mid 的信息。实际上由于 mid 第一次回到 mid 的多项式 H 是容易用 $G[l,mid-1]$ 与 $G[mid+1,r]$ 的信息推出来的，最终只要求出 $\frac{1}{1-H}$ 即可得到 mid 回到 mid 的信息。

时间复杂度为 $O(n\log^2 n)$，可以通过所有子任务，可以得到 100 分。

6 命题过程与致谢

本题在命制时的初始想法为算法七，在经过一定思考后，本人得到了可以求出每一个终点答案的算法九，在与来自杭州市学军教育集团文渊中学的周康阳同学进行讨论后，本人发现了算法八与算法十。

本题在命制过程中，来自长沙市长郡中学的杨鑫和同学，程楷轩同学参与了本题的验题工作，并提出了宝贵的改进意见。

感谢以上同学的帮助。

7 参考资料

https://oiwiki.org/math/linear-algebra/char-poly/