GFOJ8701 元素反应

Statement

Mitama 正在玩原神。

元素反应是原神的特色之一，深化区域元素协调发展，打好元素反应组合牌，对玩好原神具有极深远的意义。目前 Mitama 在进行超绽放反应的研究。

Mitama 给出一张 n 个点的树，如果他选择三个互不相同的节点 A, B, C 并在这三个位置分别放置一个草元素技能，一个水元素技能，一个雷元素技能，那么超绽放能发生当前仅当 $\text{dis}(A, B) \le \max\{\text{dis}(A, C), \text{dis}(B, C)\}$。请求出有多少三元组 (A, B, C) 能发生超绽放反应。

$1 \le n \le 5 \times 10^5$。

Solution

QOJ7206

为了方便，下文记 $|AB| = \text{dist}(A, B)$，

把原问题容斥一下，变成 $n(n-1)(n-2) - \sum_{A,B,C}[|AB| > \max(|AC|, |BC|)]$，

考虑一个点 X，满足 X 同时位于 $A \to B, B \to C, C \to A$ 的路径上，显然，这样的 X 是唯一的。那么有

$$\begin{aligned}
&[|AB| > \max(|AC|, |BC|)] \\
=&[|AX| + |BX| > \max(|AX| + |CX|, |BX| + |CX|)] \\
=&[\min(|AX|, |BX|) > |CX|]
\end{aligned}$$

先套路点分治,然后考虑对于分治重心 R, A, B, C 不同时位于它的一颗子树的所有情况（为了方便，我们视 R为单独一颗子树）：

- A, B, C 位于不同的三颗子树。

  此时 X 即为 R，直接枚举 C 位于那颗子树，$S1(d) =$ 其他子树中 $|UR| \ge d$ 的 U 的数量，$S2(d) =$ 其他子树中 $\min(|UR|, |VR|) \ge d, U \ne V$ 的 (U, V) 的数量，于是可以轻松使用容斥和后缀和维护。

- A, C 位于同一颗子树。

  B, C 位于同一颗子树是相同的,把这部分的答案乘 2 再加回去就行了。

  还是枚举 A, C 所属子树, 考虑枚举 X 和 $|CX|$, 经过思考后可以发现,因为需要满足 $|AX| > |CX|$, 故这和长剖有几乎等价的转移方式, 而 $|BX| = |BR| + |RX| < |CX| \to |BR| < |CX| - |RX|$, 故可以使用第一种情况中的 $S1$ 来协助维护。

- A, B 位于同一颗子树。

  第二种情况的解决对我们颇有启发意义，考虑 $|CX| < \min(|BX|, |AX|)$，于是在做第二种情况时我们可以同时枚举 $\min(|BX|, |AX|)$ 来计算答案. 这里看起来需要使用前缀和, 但实际上可以直接暴力移动端点维护前缀和。

综上，我们用一种非常优雅的方法通过了这道题，总复杂度就是点分治的复杂度 $O(n \log n)$，而且不需要任何高级数据结构。

GFOJ8689 蛋糕

Statement

有一个长得像蛋糕一样的单调不降序列 a，一次“吃蛋糕”可以是下面两种操作之一：

1. 选择一个 $1 \le i \le n$，$a_i \ge 2$，令 $a_i \leftarrow a_i - 2$；
2. 选择一个 $1 \le i < n$，$a_i = a_{i+1} \ge 1$，令 $a_i \leftarrow a_i - 1$，$a_{i+1} \leftarrow a_{i+1} - 1$。

需要保证操作完之后序列仍然不降。求不同的操作方式使得 a 变为全 0 的方案数。

$1 \le n, a_i \le 1000$。

Solution

把蛋糕的轮廓画在一个二维平面上，然后把 $(0,0)$ 到 (n, a_n) 的路径用一个 01 串表示出来（0 为横着走，1 为坚着走），那么两种吃法即代表了： $a_i = 1$ 且 $a_{i+2} = 0$，交换 a_i, a_{i+2}，并且最终希望不存在逆序对。

于是我们把奇数位置的串和偶数位置的串分别提出来，变成两个独立的问题。判掉 $ans = 0$ 的情况，获得 40% 的分数。然后问题变为给定 01 串，每次可以交换相邻的 1,0，问有多少种方案使得其不存在逆序对。考虑用这个 01 串重新转换回蛋糕的轮廓，那么我们每次操作就变成了删除一个格子。把每个格子删除的时间写在格子中，发现其实就是杨表填数方案数，直接钩子公式即可。复杂度 $O\left(\sum a_i\right)$。

GFOJ8684 体育课

Statement

给定 n, m, r, c，求将 $n \times m$ 矩阵染色为 $0/1$ 的方案数，使得所有 $r \times c$ 连续子矩阵的 1 数量相同。

$1 \le n, m \le 10^9$，$1 \le r, c \le 4$。

Solution

考察左上角在 $(i,j), (i+1,j), (i,j+1), (i+1,j+1)$ 的 $r \times c$ 的子矩形，计算 $(i,j) + (i+1,j+1) - (i+1,j) - (i,j+1)$，其中的运算指子矩形的对位相加相减，可得 $a_{i+r,j+c} = a_{i+r,j} + a_{i,j+c} - a_{i,j}$。

将下标从 0 开始标号，可以推出对于 $i \ge r$，$j \ge c$，$a_{i,j} = a_{i \bmod r, j} + a_{i, j \bmod c} - a_{i \bmod r, j \bmod c}$。即确定了前 r 行和前 c 列的元素后，可以求出所有元素，但可能无解：

对于所有 $x \in [0,r), y \in [0,c)$，记 $b_{i,j} = a_{ir+x, jc+y}$，$a_{i,j} \in \{0,1\}$ 会带来限制: $\forall i, [b_{0,0} = b_{i,0}]$ 或 $\forall j, [b_{0,0} = b_{0,j}]$，即 b 要么每行一样，要么每列一样。

枚举每个 (x,y) 是每行一样还是每列一样，考察原题限制，计算 $(x,j+1)-(x,j)$ 和 $(i+1,y)-(i,y)$

具体来说，分别用 col_j 和 row_i 表示第 j 列上“列相同的位置”有几个、第 i 行上“行相同的位置”有几个。

1. 先算 “列相同的情况”，即同一行的可以随便填，每一列枚举对应的位置（注意只考虑 “列相同” 的这 col_i 的位置) 有几个填 1，接下来后面对应位置列 1 的个数都要和当前列相同，记 $num = \lfloor \frac{m-i+c-1}{c} \rfloor$ 为这些列的个数，方案相加后每列求积即可，即 $\prod_{i=0}^{c-1} \sum_{j=0}^{col_i} \binom{c_j l_i}{j}^{num}$；
2. 对于 “行相同的情况”，这里需要容斥，即在计算时要减去可以作为 “列相同” 的情况。用 $row_i = 3$举例：类似的考虑前 m 列的每一行随便填，首先一行 (对应位置上) 的值不能为 0 或 3，不然所有位置都要填 $0/1$，和 “列相同” 一样，然后还要考虑某一位置一列下来都一样的情况，该位置也是 “列相同 “了，这边也要做个小容斥 (3 个都 “列相同”)，最后的权值即为 $2 \times 3^{num} - 6 \times 2^{num} + 6$ 其它 row_i 情况类似。

时间复杂度 $O(2^{rc}rc)$。

QOJ8010 Hierarchies of Judges

Statement

计数满足以下条件的 n 个节点的有标号有根树的数量：

- 每个节点有颜色黑色或白色。
- 对于每个节点，其本身以及其儿子中，至少一半的节点是白色的。

两棵树被认为不同当且仅当：

- 对于某个节点，其在两棵树中颜色不同。
- 对于某个节点，其儿子的**集合**不同。
- 对于某个节点，其**白色**儿子的**相对顺序**不同。

答案对 998244353 取模。$1 \le n \le 2 \times 10^5$。

Solution

考虑 EGF。设根为黑色的 EGF 为 $F_0(x)$，根为白色的 EGF 为 $F_1(x)$，则答案是 $\left[\frac{x^n}{n!}\right](F_0(x) + F_1(x))$。

那么列出关于 $F_0(x), F_1(x)$ 的方程，枚举黑色/白色的儿子数量：

$$
F_0(x) = x \sum_{i=0}^{\infty} \frac{F_1(x)^i}{i!} \sum_{j=0}^{i-1} F_0(x)^i
$$

$$
F_1(x) = x \sum_{i=0}^{\infty} \frac{F_1(x)^i}{i!} \sum_{j=0}^{i+1} F_0(x)^i
$$

把后面的 $F_0(x)^i$ 求和用等比数列求和重写，然后整个式子又可以用 exp 重写：

$$
\begin{aligned}
F_0(x) &= x\sum_{i=0}^{\infty}\frac{F_1(x)^i}{i!}\cdot\frac{F_0(x)^i-1}{F_0(x)-1}\\
&= \frac{x(\exp(F_0(x)F_1(x))-\exp F_1(x))}{F_0(x)-1}\\
F_1(x) &= x\sum_{i=0}^{\infty}\frac{F_1(x)^i}{i!}\cdot\frac{F_0(x)^{i+2}-1}{F_0(x)-1}\\
&= \frac{x(F_0(x)^2\exp(F_0(x)F_1(x))-\exp F_1(x))}{F_0(x)-1}
\end{aligned}
$$

那么我们把方程组整理一下，直接用 F_0, F_1 表示 $F_0(x), F_1(x)$:

$$
\begin{aligned}
G_0(F_0,F_1) &= x(\exp(F_0F_1)-\exp F_1)-F_0(F_0-1)\\
G_1(F_0,F_1) &= x(F_0^2\exp(F_0F_1)-\exp F_1)-F_1(F_0-1)
\end{aligned}
$$

$$
G_0(F_0,F_1)=G_1(F_0,F_1)=0
$$

考虑使用**多元函数的牛顿迭代法**。原方程组的 Hessian 矩阵为:

$$
H=\begin{bmatrix}\frac{\partial G_0}{\partial F_0} & \frac{\partial G_0}{\partial F_1}\\ \frac{\partial G_1}{\partial F_0} & \frac{\partial G_1}{\partial F_1}\end{bmatrix}
$$

设 $F_0(x), F_1(x)$ 为 $\bmod x^n$ 的结果，F_0^*, F_1^* 为 $\bmod x^{2n}$ 的结果，那么:

$$
\begin{bmatrix}F_0^*\\ F_1^*\end{bmatrix}=\begin{bmatrix}F_0\\ F_1\end{bmatrix}-H^{-1}\begin{bmatrix}G_0(F_0,F_1)\\ G_1(F_0,F_1)\end{bmatrix}
$$

经计算可以验证，H 总是存在逆，故迭代可以进行。该方法正确性的证明和传统的多项式牛顿迭代是类似的。

在计算过程中，需要进行多项式乘法、多项式求逆、多项式 exp，因此复杂度 $T(n)=T\left(\frac{n}{2}\right)+O(n\log n)=O(n\log n)$。常数很大，但是足以通过。

GFOJ8706 赛博航行

Statement

给定长度为 n 的序列 a，给出代码:

```
const int maxn=1e6+5;
int b[maxn],ans[maxn],p[maxn],a[maxn],pos[maxn],n,k,cnt;
void rev(int l,int r) {
    for(int i=l;i<=r;i++) b[i]=ans[i];
    ans[l]=b[r];
    for(int i=l+1;i<=r;i++) ans[i]=b[i-1];
}
void solve() {
    ans[1]=p[1];
    for(int i=2;i<=n;i++) {
        cnt=0;
        if(ans[i-1]<=p[i]) {
            for(int j=1;j<i;j++)
```

```
                if(ans[j]<=p[i])
                    pos[++cnt]=j;
        }
        else {
            for(int j=1;j<i;j++)
                if(ans[j]>p[i])
                    pos[++cnt]=j;
        }
        for(int j=1;j<=cnt;j++) {
            rev(pos[j-1]+1,pos[j]);
        }
        ans[i]=p[i];
    }
}
void get(int l,int r) {
    n=r-l+1;
    for(int i=l;i<=r;i++) p[i-l+1]=a[i];
    solve();
}
```

有 m 次操作：

1. 给定 l,r,x，将 $l \le i \le r$，$a_i \leftarrow a_i + x$；
2. 给定 l,r，求如果进行上述代码的 `get(l,r)` 后，$ans[1:n]$ 的逆序对数。

$1 \le n,m \le 10^6$。

Solution

考虑加入一个 p_i 对答案的贡献。

经推导可得，如果 $p_i \ge p_{i-1}$，那么贡献是 0，否则贡献是 $i-1$。

树状数组维护 $p_i - p_{i-1} < 0$ 的位置的和即可。

[CF757G Can Bash Save the Day?]

Statement

给定 n 个点的树和一个 $1 \sim n$ 的排列 p，进行 q 次操作：

1. 给定 l,r,x，求 $\sum_{i=l}^{r} \operatorname{dis}(p_i,x)$。
2. 给定 x，将 p_x,p_{x+1} 交换。

强制在线。$1 \le n,q \le 2\times 10^5$。

Solution

记排列 p 的逆排列为 p^{-1}，那么查询相当于只有 $p_u^{-1} \in [l,r]$ 的 u 才会产生贡献。

首先考虑树分块。那么我们需要查询的就是块内有贡献点的个数及到界点距离之和，这是一个单点修改区间求和的问题，使用 $O(1)$ 查询 $O(\sqrt{n})$ 修改的分块维护。总时间复杂度 $O(n\sqrt{n})$。

然后不难发现这个树分块的问题可以继续 reduce，所以可以把树分块改成某种树分治（如点分树），然后每个分治节点用线段树维护单点修改区间求和。时空复杂度均为 $O(n\log^2 n)$。

在做法上不太有优化空间了，所以考虑题目性质。注意到修改交换的是相邻元素，也就是说 p^{-1} 的变化量只有 1，而上面的做法都没有依赖于该性质。在每个点分树节点上维护子树内 p_u^{-1} 的相对顺序，那么每次修改至多交换一对相邻的数，于是可以直接前缀和维护那个区间求和问题。查询时需要在 log 个点分树节点上二分 l, r 对应的 rank，这里可以用类似分散层叠的小技巧优化成 $O(\log n)$。总时间复杂度 $O(n \log n)$。

QOJ4829 Mark on a Graph

Statement

邪恶的在线法官给了你一张 n 个点 m 条边的简单无向图。你需要判断这张图是以下两类中的哪一类:

1. 它是一张随机图，即边集在所有可能中等概率均匀随机。

   如果这样，你需要对图进行至多 5 次加边或删边后还给法官。

2. 它是一张被你在 1 中操作过的图（图的标号和边的顺序会被随机打乱）。

   如果这样，你只需要告诉法官这一点即可。

对所有数据，$n = 1000, 2000 \le m \le 5000$。

Solution

做法一

`Mark` 的时候找出图中度数最大的 5 个点（并列随便选一个），把它们以某种顺序串成一个环，发现这样做完度数最大的 5 个点还是度数最大的 5 个点，并且没有同为第 5 大的情况。判断的时候看看度数最大的 5 个点是否被串成一个环了即可。

做法二

`Mark` 的时候任意找到 5 个点使得它们之间的连边有至少 5 条，然后把它们加边形成 K_5。判断的时候检测是否存在 K_5 即可。以上两步题解都声称可以暴力搜索剪枝实现，正确率是因为 5 个点之间有 5 条边很易出现，但是 K_5 很难出现。

做法 *

Draw a pentagon or a hexagon without inner edges? Or perhaps some other shape.

Consider vertex of maximum degree, add 5 to that degree.

... Use your imagination!

做法 Ω

~~`sleep(1); cout<<(time(0)&1?"mark\n0":"ok");`~~

CF1201E2 Knightmare

Statement

$n \times m$ 的国际象棋棋盘，n, m 是偶数。一开始有一个白马在 (x_1, y_1)，有一个黑马在 (x_2, y_2)，白马先手。白马目标是到达 $(n/2, m/2)$ 或把对方吃掉，黑马目标是到达 $(n/2+1, m/2)$ 或把对方吃掉。如果黑马先到了终点，但是下一步它就被白马吃了，也算白马赢。你要和交互库下棋，自选白马黑马，要赢。

$6 \le n, m \le 1000$。

Solution

看到棋盘和马第一反应想到的是二分图。以下约定 W 为白马初始位置，T_W 为白马目标，B 为黑马初始位置，T_B 为黑马目标，

考虑吃子这个最麻烦的条件，如果不允许吃子这题就是简单的最短路比速度。进一步思考发现：如果两个马初始异色，那么只有白马可能吃黑马；如果两个马初始同色，那么只有黑马可能吃白马。以下假设初始两个马异色，同色的情况同理。

现在只有白马能吃黑马，所以如果白马跑得比黑马快就直接不管不顾冲到终点就行了，黑马阻止不了它。

如果白马比黑马慢，那仅仅比速度肯定比不过，所以白马要想办法吃掉黑马。进一步，如果白马有可能吃到黑马，就说明**白马一定可以在黑马之前到达 T_B**。这里“之前”定义为“至多晚一步”，即 $\operatorname{dis}(W, T_B) \le \operatorname{dis}(B, T_B) + 1$。

发现黑色目标和白色目标很接近，这应当有用。事实上，当白马到达 T_B 的时候，黑马与白马的距离至少为 2（因为它们现在同色），所以下一步黑马不能使得它与白马距离为 1（否则白马一步就把它吃了），从而下一步黑马与白马距离为必然为 3。惊奇地发现 $\operatorname{dis}(T_B, T_W) = 3$，所以白马直接 3 步冲到终点就赢了。

另一方面，如果白马不能在黑马之前到达 T_B，就说明白马一定不可能吃到黑马，那黑马不管不顾冲到终点，白马就输了。所以白马能在黑马之前到达 T_B 是充要的（当白马不能直接赢，即 $\operatorname{dis}(W, T_W) > \operatorname{dis}(B, T_B)$ 的时候）。

跑一遍最短路即可。注意判一些边界情况，例如能吃掉对方就直接吃掉并记得结束程序。还有如果 WA 了发现 Jury 的输出中有类似于 `winning position` 的字眼说明你大概率选错边了 Jury 确保自己能赢。没有的话大概率是被 Jury 吃了或操作不合法。

GFOJ8697 回文

Statement

给出 n 对 (x_i, y_i)，构造一个次数为 k 次，最高次项不为 0 的多项式 $f(x)$，满足 $f(x_i) \equiv y_i \pmod{998244353}$，且 $f(x)$ 是回文的，即 $[x^i]f(x) = [x^{k-1-i}]f(x)$，要求 $k \le 10^4 + 1$。$n \le 1000$。

Solution

sub1

直接输出 $d = 1, a_0 = y$。

sub2

枚举 d，每次大力跑高斯消元，复杂度 $O\left(Tnd^3\right)$，注意不一定有唯一解，此时需要调整出 $a_{d-1} \ne 0$ 的一组解。

sub3

首先特判掉 $x_i = y_i = 0$ 的限制，因为 $A(0) = 0 \Longrightarrow a_0 = a_{d-1} = 0$。

一种经典的翻转多项式系数的方法是代入 x^{-1} 再乘上 x^{d-1}，即 $A\left(x^{-1}\right)x^{d-1} = \sum x^i a_{d-i-1}$。由此可知，一个多项式是回文的当且仅当 $A(x) = A\left(x^{-1}\right)x^{d-1}$ 在 $x \ne 0$ 时恒成立。

因此，如果已知 $A(x) = 0$，则也可以得到 $A\left(x^{-1}\right) = 0$，对于每个输入的 $(x_i, 0)$，如果 x_i^{-1} 没有在输入数据中，就加入一条新的限制 $\left(x_i^{-1}, 0\right)$。

这样会得到至多 $2n$ 条限制，可以构造 $A(x)=f(x)+f\left(x^{-1}\right)x^{3000}$，其中 $f(x)=\prod\left(x-x_i\right)$，3000 是乱写的一个 $>2n$ 的值，因为需要足够大才能保证 $f\left(x^{-1}\right)x^{3000}$ 在 $x=0$ 时取 0，从而让 $A(0)=f(0)=\prod\left(-x_i\right)\neq 0$。

sub4

沿用 sub3 的思路，由于 $n^2<10^9+9$，所以不用担心限制出现矛盾的情况，可以随便定一个足够大的 d，比如 5000。

同样地，对于每个输入的 (x_i,y_i)，把 $\left(x_i^{-1},\frac{y_i}{x_i^{d-1}}\right)$ 也加入限制。

先考虑插值出一个符合限制的多项式 $f(x)$，然后构造 $A(x)=\dfrac{f(x)+f\left(x^{-1}\right)x^{d-1}}{2}$。但插值出来的东西可能不满足 $f(0)\neq 0$， 此时只需要让 $f(x)\leftarrow f(x)+k\prod\left(x-x_i\right)$ 即可，其中 k 是随便扔的一个非零常数。

sub5

由于此时可能出现两个输入的 x_i,x_j 互为逆元，当 $\dfrac{y_i}{x_i^{d-1}}\neq y_j$ 时这个 d 是不合法的，因此需要枚举 d。

做法还是和 sub4 一致，不过需要注意的是可能存在 $x_i=0$ 的限制，对此的解决方案是:

- 特判 $x_i=y_i=0$ 为无解，
- 一开始不考虑此限制，直到最后一步调整 $f(x)\leftarrow f(x)+k\prod\left(x-x_i\right)$ 时，通过合理地设置 k 来满足该限制。

最后还有个小问题，如果最后一步 $f(x)$ 加上 $k\prod\left(x-x_i\right)$ 之后次数超过 $d-1$ 了怎么办? 实际上，可以直接判定当前的 d 是无解的，因为对于一个给定的点集 $\{(x_i,y_i)\}$，所有满足 $h\left(x_i\right)=y_i$ 的多项式 $h(x)$ 都形如 $h(x)=f(x)+g(x)\prod\left(x-x_i\right)$，其中 $f(x)$ 是对这个点集插值插出来的多项式。也就是说，所有这样的 $h(x)$ 要么等于 $f(x)$，要么次数大于等于 $\Pi\left(x-x_i\right)$，出现上面的情况说明 $f(x)$ 不符合条件，$\prod\left(x-x_i\right)$ 的次数又太大，故当前的 d 肯定不存在答案。

正解

实际上，如果写一个从大到小枚举 d，用 $O(n)$ 的复杂度判断是否存在 $x_i=x_j^{-1},y_i\neq\dfrac{y_j}{x_j^{d-1}}$，如果存在直接跳过这个 d，否则跑 sub5 的做法，找到一组解时直接结束的程序，可以惊奇地发现直接通过了本题。原因如下:

先找到所有 $x_i=x_j^{-1}$ 的 (i,j) 对，它们对于 d 的限制就是 $x_j^{d-1}=\dfrac{y_j}{y_i}$，解出来会形如一个 $d\equiv a\pmod b$ 的限制，而把所有这些限制合并后，也必然得到形如 $d\equiv a\pmod b$ 的限制。因此，如果 D 以内只有一个符合该限制的 d，那么只需要对一个值尝试构造，否则一定存在一个大于 5000 的满足该限制的 d，由于它远大于 $2n$，所以构造一定有解。因此，至多只会一个 d 尝试构造，总复杂度实际为 $O\left(T\left(nd+n^2\right)\right)$。