Solution

魔法阵(magic)

显然，最开始的时候，所有的魔术回路构成了若干个环。

我们可以发现，所谓的重新排列一个魔法阵节点内的宝石，就是将经过同一个魔法阵节点的两个魔术回路的环连接起来。

因此，我们可以将初始的环处理出来，看成一个大的节点，对于所有的大的节点，每一个魔法阵节点的重新排列意味着可以在新的两个大节点之间花费c_i的代价连接一条边，这就变成了一个非常经典的最小生成树问题，可以使用Kruskal算法简单解决。

时间复杂度 $O(N \log N)$

猫(cat)

注意到行列之间互不影响，每行每列内部 L, R 的格子与 U, D 的格子也并不影响。

以行为例，设 $f_{i,j}, g_{i,j}$ 表示前 i 个格子，还有 j 个 R 没匹配的方案数与权值和。

假设不选当前格子：

$$f_{i,j} \leftarrow f_{i-1,j}, g_{i,j} \leftarrow g_{i-1,j}$$

如果当前是 L 并且选，那么有:

$$f_{i,j-1} \leftarrow f_{i-1,j} \times j, g_{i,j-1} \leftarrow (g_{i-1,j} + f_{i-1,j} \times a_i) \times j$$

如果当前是 R 并且选，那么有:

$$f_{i,j+1} \leftarrow f_{i-1,j}, g_{i,j+1} \leftarrow g_{i-1,j} + f_{i-1,j} \times a_i$$

时间复杂度 $O(N^3)$。

随机图生成器 (random)

我们用 $G_{a,b}$ 表示图 G 中是否存在从结点 a 到结点 b 的边，$T(a,b,c)$ 表示 (a,b,c) 或 (a,c,b) 是否为三元环，则我们要求的是

$$\sum_G \sum_{1\le a<b<c\le n} T(a,b,c) = \sum_{1\le a<b<c\le n} \sum_G (G_{a,b}G_{b,c}G_{c,a} + G_{a,c}G_{c,b}G_{b,a})$$

枚举 a,b,c 然后求有多少个图以 (a,b,c) 或 (a,c,b) 为三元环。时间复杂度 $\mathcal{O}(n^3)$。

由于 $(G_{a,b}+G_{b,a})(G_{b,c}+G_{c,b})(G_{c,a}+G_{a,c}) = 1$

可得 $T(a,b,c) = 1 - G_{a,b}G_{a,c} - G_{b,a}G_{b,c} - G_{c,a}G_{c,b}$。

于是

$$\begin{aligned}\sum_G \sum_{1\le a<b<c\le n} T(a,b,c) &= \sum_{1\le a<b<c\le n} \sum_G (1 - G_{a,b}G_{a,c} - G_{b,a}G_{b,c} - G_{c,a}G_{c,b}) \\ &= \binom{n}{3}2^m - \sum_{1\le a\le n} \sum_G \binom{\deg_G^+(a)}{2}\end{aligned}$$

现在只需对每个点 a 和每个 x 统计有多少个图 G 满足 $\deg_{G_0}^+(a) = x$，答案减去图的个数乘 $\binom{x}{2}$。

这一步很简单，设 a 在 G_0 中的出度为 $c = \deg_{G_0}^+(a)$，d 为 G_0 中和 a 不关联的结点数。则生成的 d 条和 a 关联的边中有 x 条从 a 连出的方案数为 $\binom{d}{x}2^{m-d}$。故

$$\sum_G \binom{\deg_G^+(a)}{2} = 2^{m-d}\sum_{x=0}^{d}\binom{d}{x}\binom{c+x}{2}$$

时间复杂度 $\mathcal{O}(n^2)$。这里等号右边的计算可以优化到 $\mathcal{O}(1)$，不过输入是 n^2 的。

棋盘(chessboard)

为了方便，我们先将横纵坐标奇偶相同的格子异或 1，这样限制就变成了不能有同色的 2×2 子矩形。

那么如果存在一个 2×2 的矩形，其中三个格子颜色确定且相同，那么就能判断出另外一个格子的颜色。

先用一次 bfs 把所有能推的格子推出来，如果矛盾则无解，否则我们声称如下过程一定能构造出一组解：

- 如果所有格子都有数那么终止进程，否则任意选一个空格子填入任意一种数。
- 进行一次 bfs 把所有能推的格子推出来。
- 重复上面两步直到所有格子都有数。

下面证明上面的操作不会产生矛盾。

不妨设第一步任意填的位置为 (x,y)，且填的数为 0。不妨考虑右下角的情况。

如果 $(x+1,y+1)$ 能确定，那么 $(x,y+1),(x+1,y)$ 填的都是 0，而 $(x+1,y+1)$ 会被填 1。因为 $(x,y+1)$ 和 $(x+1,y)$ 填的数不同，所以只能往 $(x+2,y+2)$ 扩展。

以此类推，新被确定的数一定形如 $(x+k,y+k)$。如果在 $(x+k,y+k)$ 处产生矛盾，因为 $(x+k-1,y+k),(x+k,y+k-1)$ 一定与 $x+k,y+k$ 异色，所以产生矛盾的一定是以它为左上角的矩形，但是这种情况下它应该在之前就被确定了，矛盾。

模拟即可做到 $O(nm)$。

圣诞树(tree)

对于 $p=1$ 的问题就是动态插入直线，查询某个位置最大值。可以用李超树实现，复杂度 $\mathcal{O}(n\log n)$ 。

对于 $a=0$ 的问题，等价于动态查询到根路径上的最大值。可以离线下来 dfs 用数据结构维护以时间为下标的，支持单点修改，查询区间最大值的线段树。

也可以在加入小熊玩偶的时候对子树取 $\max$ ，单点查询。时间复杂度 $\mathcal{O}(n\log n)$。

对于链上的问题，等价于有一个二维偏序的限制。考虑 CDQ 分治去掉其中一维。比如先对时间那一维分治。然后把询问按照深度顺序排序，然后就可以按照 $p=1$ 的方法做了。复杂度 $\mathcal{O}(n\log^2 n)$。

这启发我们通过分治去掉一维。不妨去掉时间使得所有小熊玩偶都能对小猫玩偶造成贡献。

一只小熊玩偶可以影响 dfn 一段区间的小猫玩偶。那么采用树套树，将其对应的 dfn 区间拆成 $\mathcal{O}(\log n)$ 个线段树上的区间，在这些节点上插入一个一次函数。

关于一次函数，要么采用排序后建凸包，询问时在凸包上二分，要么采用李超树。时间复杂度都是 $\mathcal{O}(n\log^3 n)$。

李超树的 $\log$ 去不掉。考虑如何优化建凸包和询问时的复杂度。

如果初始加入的小熊玩偶满足 a 有序，就不需要排序。同理，如果询问的小猫玩偶满足 x 有序，就可以使用双指针求得答案。那么直接 CDQ 分治的时候归并排序即可。时间复杂度即可降为 $\mathcal{O}(n\log^2 n)$ 。