Binary

20pts $L \leq 10^5$

直接模拟即可，势能分析可得时间复杂度为$O(L \log L + n)$

20pts 满二叉树 $n \leq 2^{18}$

发现满二叉树的情况是有循环节的，且循环节为2^d，d 为二叉树的深度（根节点深度为0）。

将 L 对 2^d 取模后直接模拟即可。

时间复杂度 $O(n \log n)$

20pts 满二叉树 $n \leq 2^{20}$

对于一个深度为 d 的节点，我们考虑它会在哪几次投球被小球经过，发现这是以 2^d 为周期的，每个周期只会经过一次，我们考虑把它在每个周期内第几次被经过算出来。

具体来说，如果一个节点会在它的周期内第 k 次被经过，那么它初始情况下的重儿子会在周期内第 k 次被经过，轻儿子会在周期内的第 $k + 2^d$ 次被经过，我们可以在树上 dfs 得到每个点的会在周期内第几次被经过，然后计算出它在 L 次投球中会被经过多少次，即可得到它最终的参数值。

时间复杂度 $O(n)$

100pts $n \leq 2^{20}$

将上面的满二叉树做法拓展一下。

注意到二叉树的深度可能很大所以循环节长度可能会爆 $long\ long$，但是如果循环节太大了的话可以直接通过特判来得到小球经过这个点的次数，所以对循环节超过 L 的情况特判一下就好了。

时间复杂度 $O(n)$

100pts 简单做法

假设我们在点 v 处投了 L 次球。

首先根据 L 的奇偶性我们可以得到点 v 最终的参数。

然后我们知道小球会落入重儿子的子树 $\lceil \frac{L}{2} \rceil$ 次，落入轻儿子的子树 $\lfloor \frac{L}{2} \rfloor$ 次。

直接递归左右儿子即可。

String

30pts

暴力搜索即可.

时间复杂度 $O(3^n m)$.

10pts (x = 1)

用并查集把相交的有限制的区间合并成一个块, 把其他没有限制的位置也看成一个块. 设总块数为 k, 答案即为 3^k.

时间复杂度为 $O(\alpha n)$.

100pts

设 $f[i][j][k]$ 为: 已填好前 i 个字符, 从后往前数第一个与 s_i 不同的字符的位置为 j, 第一个与 s_i 不同且与 s_j 不同的字符的位置为 k 的方案数.
$(k < j < i)$

根据 s_{i+1} 所填的数, 有 3 种转移.

1. $s_{i+1} = s_i$, 则 $f[i][j][k] \to f[i+1][j][k]$.
2. $s_{i+1} = s_j$, 则 $f[i][j][k] \to f[i+1][i][k]$.
3. $s_{i+1} = s_k$, 则 $f[i][j][k] \to f[i+1][i][j]$.

m 个限制就相当于对 j 和 k 范围的限制.

假设存在限制 (l, r, x),

1. $x = 1$, 则当 $i = r$ 时, $j < l$.
2. $x = 2$, 则当 $i = r$ 时, $j \geq l$ 且 $k < l$.
3. $x = 3$, 则当 $i = r$ 时, $k \geq l$.

最终答案即为 $\sum_{j,k} f[n][j][k]$. (注意 i, j 也要符合上述的限制).

时间复杂度 $O(m + n^3)$.

100pts - extra

其实这题可以做到 $O(n^2)$,但是为了降低题目难度所以只出到了 $n \leq 200$ 的数据范围.（原题数据范围 $n \leq 5000$ ）

考虑对 DP 转移进行优化.

假设我们把 $f[i][j][k]$ 看成矩阵 i 上的第 j 行第 i 列, 那么第一种转移就相当于继承上一个矩阵的值, 而第二, 三中转移就相当于对矩阵的某一行或某一列求和, 而对 j, k 范围的限制就相当于划定矩阵的"有效范围" (即不在该范围内的元素一定为 0).

由于从矩阵 i 转移到矩阵 $i+1$ 时, 只有第 i 会获得新的值, 所以矩阵的有效范围是不断缩小的 (就是之前被清 0 的数将会一直为 0). 那么我们可以直接维护矩阵每一行的和, 每一列的和, 以及每一行的有效范围的左右端点, 转移的时候根据限制暴力维护就行了. 由于矩阵的有效范围是不断缩小的, 所以左右端点移动的总次数为 $O(n^2)$.

时间复杂度 $O(m+n^2)$.

Source： Comet OJ#12E Ternary String Counting

color

10pts $m = 1$

就是求合法染色数，即为 $c(c-1)^{n-1}$

20pts $m = n$

我们转化为求每种颜色被包含在点集中的方案数，补集转化为不存在这种颜色的方案数。

所以答案为 $c(c-1)(c-2)^{n-1}$

20pts 树为一条链

现在还是考虑算一种颜色x不出现在点集中的方案数，考虑 dp 。

设 $dp[i][0]$ 表示考虑前i个点，且第i个点的颜色为某种x以外的颜色的方案数， $dp[i][1]$表示第i个点的颜色为x的方案数，转移如下：

$dp[i][0] = (c-2)dp[i-1][0] + dp[i][1]$
$dp[i][1] = (c-1)dp[i-1][0]$

注意如果点i在点集中，那么$dp[i][1] = 0$

最后得到颜色x不在点集中的方案数 $sum = (c-1)dp[n][0] + dp[n][1]$

最后答案为 $c(c(c-1)^{n-1} - sum)$

100pts

把链的dp拓展到树上，设$dp[v][0/1]$为v子树内的方案数。

一样的转移：

$dp[v][0] = \prod_{u \in son_v} \big((c-2)dp[u][0] + dp[u][1]) \big)$
$dp[v][1] = \prod_{u \in son_v} (c-1)dp[u][0]$

$sum = (c-1)dp[1][0] + dp[1][1]$，一样计算答案即可。

Source: JXOI2019

Tree

30pts $n, m \leq 2000$

直接模拟即可。

20pts 树为一条链

用set维护连续段，每次修改暴力向左右合并颜色相同的段即可。

50pts $n, m \leq 50000$

这档分是给写根号算法的选手的。

先用并查集将每个连通块缩在一起，考虑一个不断合并的过程。

因为颜色相同的连通块一旦合并在一起就不会再分开，所以总合并次数是$O(n)$的。

我们只需要考虑如何在每次修改后快速找到与当前连通块相邻的颜色相同的连通块，以及快速合并两个连通块即可。

一种简单的实现是对每个连通块按度数$\sqrt{n}$大小分类，小的暴力扫每条出边，大的就用一个set维护所有出边通向的连通块的颜色。那么每次修改的时候还需要在相邻连通块的set里面修改，但是因为度数大于$\sqrt{n}$ 的连通块个数不会超过$\sqrt{n}$个，所以一次修改复杂度是$O(\sqrt{n}\log n)$的。至于合并两个连通块，可以采用启发式合并set，复杂度$O(n\log^2 n)$

总复杂度$O(n\sqrt{n}\log n)$，因为出题人没卡所以可能可以通过全部的数据。

100pts

在树上使用这种根号分治技巧显然多此一举。

考虑到树上每个连通块至多只有一个父亲连通块，我们只对每个连通块维护它的所有儿子连通块的set，那么每次修改的时候只需要在它父亲连通块的set里修改即可。而查询相邻的颜色相同的连通块只需要在set里查，然后再判断一下父亲连通块的颜色是否相同即可。

一样的启发式合并set，总复杂度$O(n \log^2 n)$。