【题解】牛客 CSP-S 提高组赛前集训营 1

T1 仓鼠的石子游戏

定位：签到题，tag：博弈，思维

题意：n 个石圈，两个人轮流对石子进行染色，相邻不能同色，不能操作的人输，问谁赢。

10pts

白送，测试点 6 主要作用是引导选手发现堆大小为 1 时后手必败。

40pts

暴力 dfs 或者状压 dp 都可以。实际上拿 40 分不需要打代码，在纸上左右互博一下，然后把答案打个表直接输出即可。

60pts

如果大胆猜想，每一堆的胜负状态其实不随两个人的决策而改变，就会发现输赢其实之和必败堆的奇偶数目有关，所以只要统计必败堆的个数，用必败堆的奇偶数目判断即可。

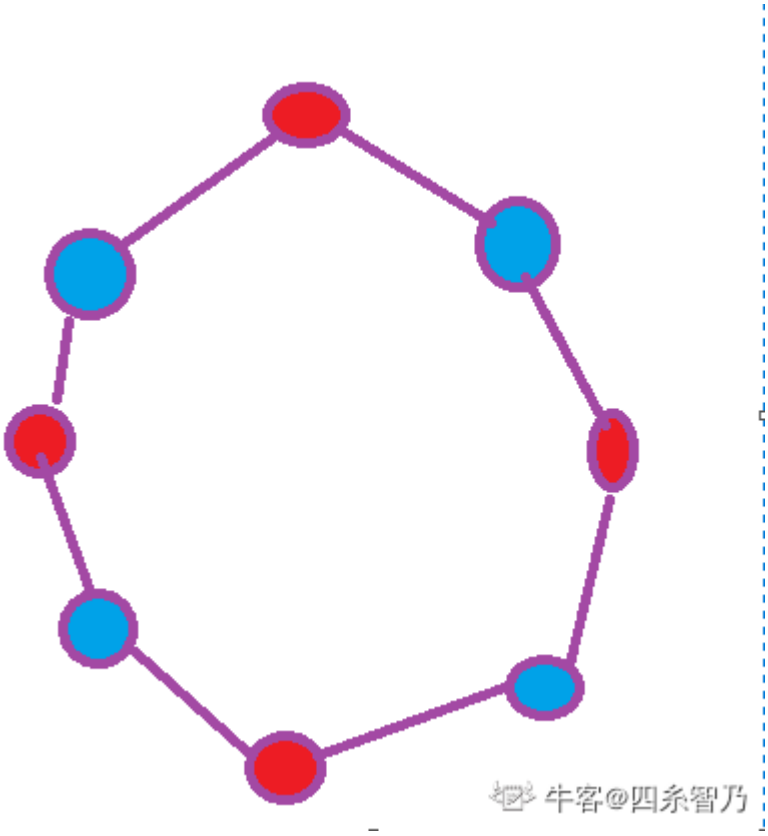
100pts

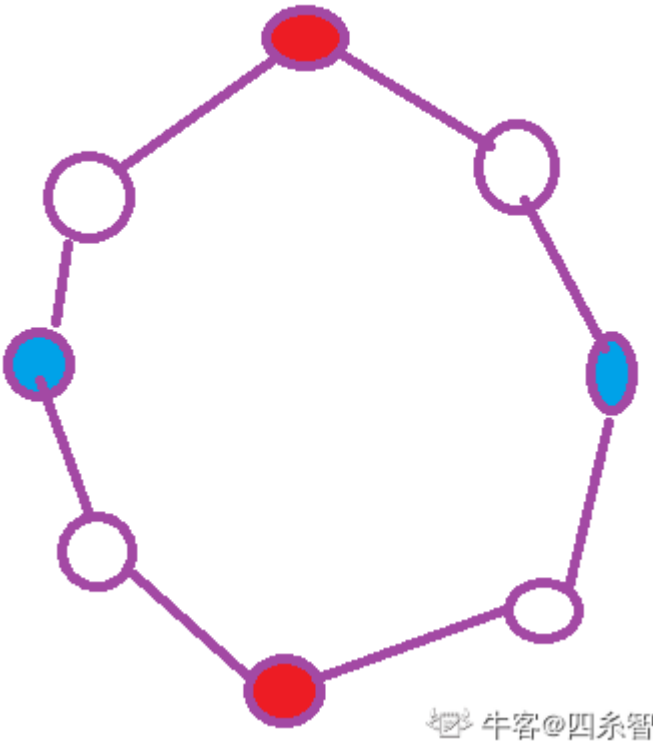
无论是通过暴力还是纸上自己推一下画画，都不难发现，当只有一圈石子的时候，除非大小为 1，不然后手一定必胜。所以只需要统计 1 的个数即可，1 的个数为奇数则先手必胜，否则后手必胜。

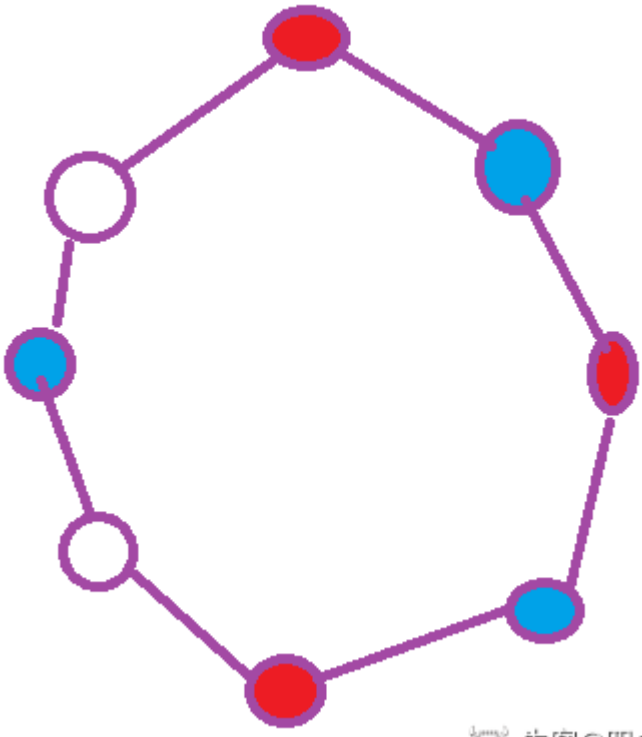
简单证明一下为什么这个博弈跟先手与后手的操作完全无关。

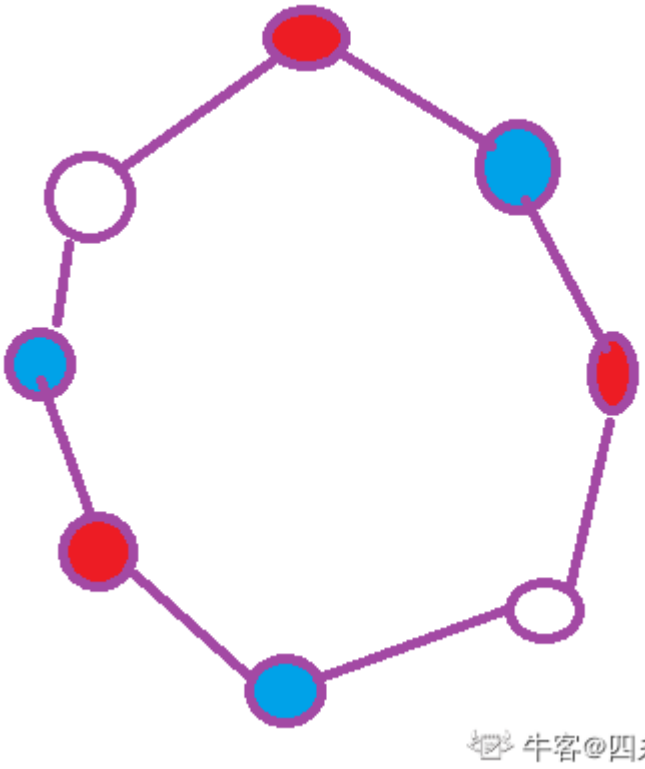
根据题意：不能有相邻同色的石子，考虑终态，终态一定不存在这种情况。

我们先考虑一个圈的情况，当一个圈无法被涂上任何颜色的时候。




牛客@四系智乃


牛客@四系智乃



我仅画出了几种情况。

可以推断，(其实也是必然啦)——颜色一定是交替出现的。

因为如果两个相同颜色连续，必定中间存在至少一个空位。

所以，当一个圈无法涂上任何颜色时，颜色出现的数量一定是偶数。所以后手一定会赢。

当且仅当，石圈的大小为 1 时，先手可以赢。

扩展到 n 个圈其实也是相同的，只不过每存在一个大小为 1 的石圈，都会导致胜负状态发生转换，所以只需要统计大小为 1 的石圈数目即可。

T2 乃爱与城市拥挤程度

定位：中等题，tag：树 DP，换根法/树 DP 的 up and down。

题意：给一颗树，问你每个节点作为根节点时，距离它小于 k 的节点数目，以及这些节点动态权值的乘积。

10pts

对于测试点 10，因为是单链，所以可以 O(1)直接算。

40pts

对于测试点 1，2，3，枚举每个点，写个 dfs 跑一下按题意模拟统计一下即可。

70pts

对于测试点 4,5，可以写一些对于测试点特化的 dp，或者利用树结构随机进行一些 dfs 上的优化。

80pts

单链其实也可以 dfs…因为没分叉所以跑的很快，优秀点的暴力应该拿到 80 了吧。

100pts

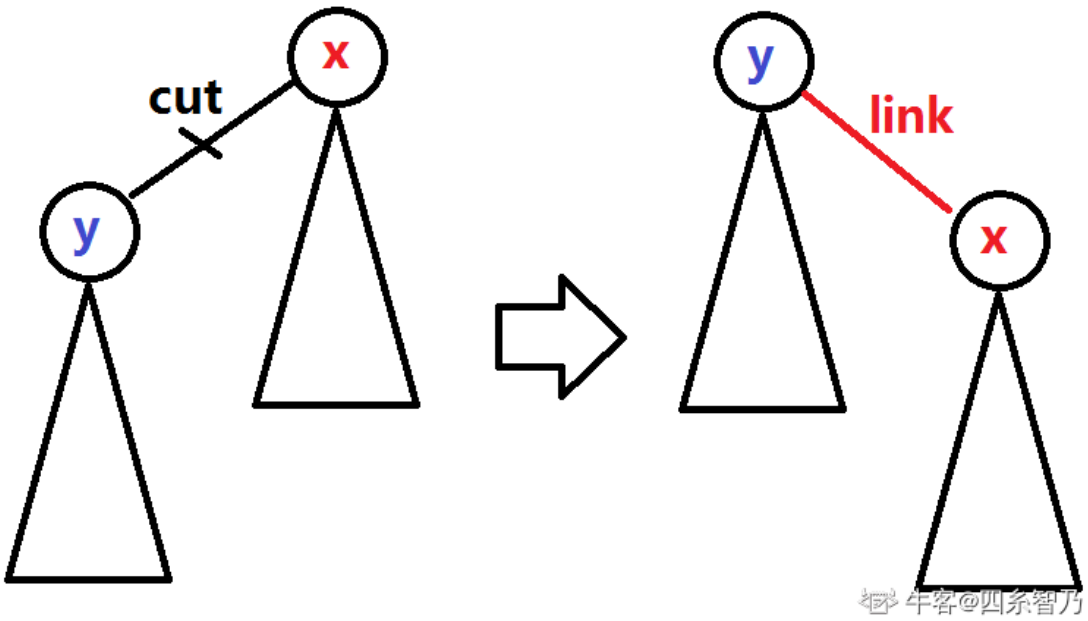
一看就是树 DP，然后也就没什么滑头搞就完了，然后这道题是无根树 DP，属于树 DP 中稍麻烦一点的情况。

对于无根树的 dp 套路其实就两种，一种是换根法，另一种是树 DP 的 up and down。

换根法：

先随便找一个点作为根节点使用，得到整个树每个节点的 dp 信息并且储存起来。

考虑相邻节点换根，也就是假设根节点为 x，然后整颗树的 dp 信息是以 x 为根的基础上进行构建的，y 节点为 x 节点的直接孩子。我们发现当整棵树的根节点由 x 转换为 y 时，改变的 dp 信息其实非常少，往往只影响到 x,y 两个节点。



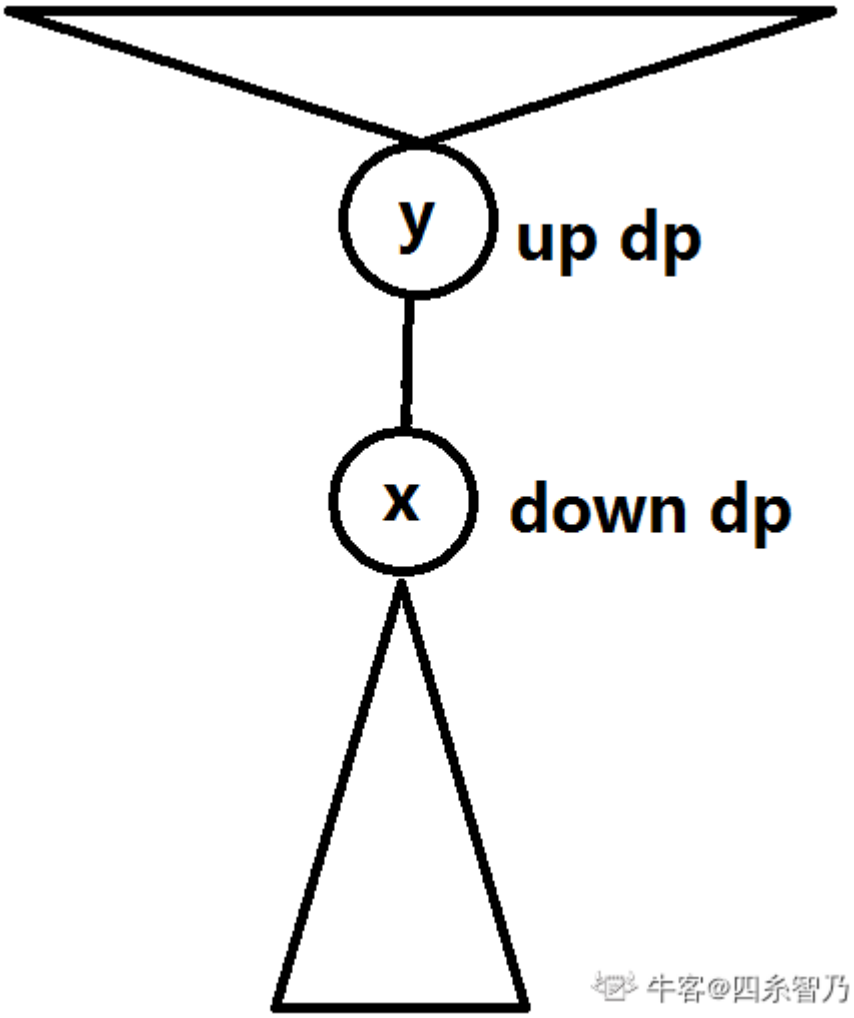
如果你 DP 方程是那种加加减减，乘乘除除的话就更适合使用换根法了。

因为它们有对应的“逆操作”。将 DP 方程反着转移就可以抵消这种转移带来的影响，因为我们可以写一个 link 一个 cut，对应转移和逆转移。利用 link cut，我们可以像平衡树一样把所有的点都拎上来当根。

如果方程没有对应的逆操作，比如 max,min，就比较麻烦了，一般解决方法是借助数据结构(一般常用的方法是开 n 个 multiset/multimap 或者手写平衡树)，或者维护前 k 大/小值来做到，这个时候推荐使用下面的方法。

up and down：

先做一个自下而上的 up dp，再做一个自上而下的子树 down dp，然后在每个节点处合并两个 dp 数组的信息，得到以每个点作为根节点的 dp 信息。



x 表示当前节点，y 节点是 x 节点的父节点，那么以 x 作为根节点的 dp 信息为 x 的 down dp 合并其父节点 y 的 up dp。

上述这两种方法换根法的常数比较大（大概慢 2~3 倍），好处是可以很方便的枚举所有根，有时题目要求导致不得不使用这种方法，建议都掌握，平时用 up and down 的技巧。

T3 小 w 的魔术扑克

定位：中等偏难题，tag：图论模型，并查集，树状数组，离线算法

题意：k 张正反面的卡片，打出时只能选择一面，查询 q 次，每次问是否能组成

l 到 r 的顺子。

10pts

对于测试点 4，搞个前缀和判一下就行了。

40pts

写个暴力 dfs，或者状压 DP 随你搞…反正数据范围很小。甚至暴力网络流都可以 40 分

100pts

如果能想到图论模型，这个题就解了一半了。对于每个面值，由于在一个查询中它最多需要一次。所以我们可以把每个面值都当成是一个节点，对于一张牌，我们都建一条连接它正反两面两个面值的边。建好模型以后，我们不难发现，如果对于一个大小为 N 的连通块有至少 N 条边，那么这个连通块一定能满足保证每个面值都能够被提供至少一次。只有一种连通块特殊，也就是当连通块的大小为 N，并且它具有 N-1 条边时，无论怎么调整，都会漏掉一个面值无法打出，大小为 N，具有 N-1 条边的连通块，那也就是树。

所以我们先找树，dfs 并查集啥玩意都 ok，总之找出所有的树就可以了。

然后我们想这个问题的反面：什么样的顺子是不能满足的，我们发现，如果你的顺子完全包含了一颗树，那这个顺子就由于上述的原因无法构造。问题来了？如

何判断一个顺子是否完全包含树?

先求出每一颗树上节点编号的最小值与最大值，构造一个约束线段，约束线段的左端点为树上节点编号的最小值，右端点为树上节点编号的最大值。

接下来问题转化成了线段覆盖问题:

即：给定查询线段 l,r，问你查询线段是否至少完整的覆盖了任意一个约束线段。如果存在则输出 No，反之输出 Yes。

这个就是个经典问题了，按照线段的右端点排序然后离线树状数组。对于每一个约束线段都将它的左端点赋成 1，然后查询，查询区间和是否为 0 即可。

当然你也可以使用桶排序或者链式前向星处理查询，然后离线扫描线 for 过去维护下界。这样做是近似 O(n) 复杂度的算法，因为第一步使用了并查集，所以全局复杂度不是 O(n)，而是 O(nlogn)，只不过并查集常数太小了所以一般近似 O(n)。