Solution

Outline

染颜色

染颜色

- 将边两端的点颜色相同的边视为实边

染颜色

- 将边两端的点颜色相同的边视为实边
- 一次新增颜色操作实际上就是 LCT 中的 access 操作

染颜色

- 将边两端的点颜色相同的边视为实边
- 一次新增颜色操作实际上就是 LCT 中的 access 操作
- 每次虚实边切换，对应于一个子树中所有结点的代价值改变

染颜色

- 将边两端的点颜色相同的边视为实边
- 一次新增颜色操作实际上就是 LCT 中的 access 操作
- 每次虚实边切换，对应于一个子树中所有结点的代价值改变
- 利用线段树或树状数组维护代价值

染颜色

- 将边两端的点颜色相同的边视为实边
- 一次新增颜色操作实际上就是 LCT 中的 access 操作
- 每次虚实边切换，对应于一个子树中所有结点的代价值改变
- 利用线段树或树状数组维护代价值
- 时间复杂度 $O(n \log^2 n)$

染颜色

- 将边两端的点颜色相同的边视为实边
- 一次新增颜色操作实际上就是 LCT 中的 access 操作
- 每次虚实边切换，对应于一个子树中所有结点的代价值改变
- 利用线段树或树状数组维护代价值
- 时间复杂度 $O(n \log^2 n)$
- 对于树随机生成的点，树高是 $O(\log n)$ 级别，因此暴力向父亲方向修改直到根即可

Outline

最大割

最大割

- 前 20%：用 unsigned int 存数，暴力枚举所有点集计算即可。$O(2^n \times m)$

最大割

- 前 20%：用 unsigned int 存数，暴力枚举所有点集计算即可。$O(2^n \times m)$
- 度数不超过 1：所有边集都是割，因此实际上就是给定若干个数，求一个子集使得异或值最大

最大割

- 前 20%：用 unsigned int 存数，暴力枚举所有点集计算即可。$O(2^n \times m)$
- 度数不超过 1：所有边集都是割，因此实际上就是给定若干个数，求一个子集使得异或值最大
- 经典的线性基应用，每次 $O(l^2)$ 维护基，总时间复杂度 $O(ml^2)$，可以使用 bitset 进一步压常数

最大割

- 前 20%：用 unsigned int 存数，暴力枚举所有点集计算即可。$O(2^n \times m)$
- 度数不超过 1：所有边集都是割，因此实际上就是给定若干个数，求一个子集使得异或值最大
- 经典的线性基应用，每次 $O(l^2)$ 维护基，总时间复杂度 $O(ml^2)$，可以使用 bitset 进一步压常数
- 将每个点的点权定义为它所连所有边的权值异或和

最大割

- 前 20%：用 unsigned int 存数，暴力枚举所有点集计算即可。$O(2^n \times m)$
- 度数不超过 1：所有边集都是割，因此实际上就是给定若干个数，求一个子集使得异或值最大
- 经典的线性基应用，每次 $O(l^2)$ 维护基，总时间复杂度 $O(ml^2)$，可以使用 bitset 进一步压常数
- 将每个点的点权定义为它所连所有边的权值异或和
- 问题转为给定若干个数 (所有点的点权)，求子集最大异或和，需要支持修改操作

最大割

最大割

- 维护基的同时，再维护每一行是由哪些向量 (权值) 异或而成

最大割

- 维护基的同时，再维护每一行是由哪些向量 (权值) 异或而成
- 修改一个向量时，找到由它异或而成的行中 1 的位数最低的那个行 (有零行就选择零行)，将其他包含修改向量的行消去它

最大割

- 维护基的同时，再维护每一行是由哪些向量 (权值) 异或而成
- 修改一个向量时，找到由它异或而成的行中 1 的位数最低的那个行 (有零行就选择零行)，将其他包含修改向量的行消去它
- 将选出的此行异或上修改值，并重新添加进基中

最大割

- 维护基的同时，再维护每一行是由哪些向量 (权值) 异或而成
- 修改一个向量时，找到由它异或而成的行中 1 的位数最低的那个行 (有零行就选择零行)，将其他包含修改向量的行消去它
- 将选出的此行异或上修改值，并重新添加进基中
- 这样做能保证不影响其他基中的元素

最大割

- 维护基的同时，再维护每一行是由哪些向量 (权值) 异或而成
- 修改一个向量时，找到由它异或而成的行中 1 的位数最低的那个行 (有零行就选择零行)，将其他包含修改向量的行消去它
- 将选出的此行异或上修改值，并重新添加进基中
- 这样做能保证不影响其他基中的元素
- $O(\frac{nm(l+n)}{64})$

Outline

开锁

开锁

- 每个盒子向它能开启的盒子连边，由于每个盒子都有且仅有一把钥匙，因此这张图会是几个环

开锁

- 每个盒子向它能开启的盒子连边，由于每个盒子都有且仅有一把钥匙，因此这张图会是几个环
- 每个环中选择一个点即可全部打开

开锁

- 每个盒子向它能开启的盒子连边，由于每个盒子都有且仅有一把钥匙，因此这张图会是几个环
- 每个环中选择一个点即可全部打开
- $dp_{i,j}$ 表示前 i 个环选了 j 个点并且全部打开的方案数

开锁

- 每个盒子向它能开启的盒子连边，由于每个盒子都有且仅有一把钥匙，因此这张图会是几个环
- 每个环中选择一个点即可全部打开
- $dp_{i,j}$ 表示前 i 个环选了 j 个点并且全部打开的方案数
- 枚举每个环选择了几个点进行转移

开锁

- 每个盒子向它能开启的盒子连边，由于每个盒子都有且仅有一把钥匙，因此这张图会是几个环
- 每个环中选择一个点即可全部打开
- $dp_{i,j}$ 表示前 i 个环选了 j 个点并且全部打开的方案数
- 枚举每个环选择了几个点进行转移
- 总方案数除以 $\binom{n}{k}$ 即可

开锁

- 每个盒子向它能开启的盒子连边，由于每个盒子都有且仅有一把钥匙，因此这张图会是几个环
- 每个环中选择一个点即可全部打开
- $dp_{i,j}$ 表示前 i 个环选了 j 个点并且全部打开的方案数
- 枚举每个环选择了几个点进行转移
- 总方案数除以 $\binom{n}{k}$ 即可
- 若担心精度问题，则把概率分配到转移中即可

开锁

- 每个盒子向它能开启的盒子连边，由于每个盒子都有且仅有一把钥匙，因此这张图会是几个环
- 每个环中选择一个点即可全部打开
- $dp_{i,j}$ 表示前 i 个环选了 j 个点并且全部打开的方案数
- 枚举每个环选择了几个点进行转移
- 总方案数除以 $\binom{n}{k}$ 即可
- 若担心精度问题，则把概率分配到转移中即可
- 时间复杂度 $O(Tnk)$