Contents

1 Pastoral Oddities

题目来源：https://codeforces.com/contest/603/problem/E

1.1 题目大意

N 个点，M 次操作，每次操作加入一条带权无向边，然后询问当前图是否存在一个生成子图，满足所有点的度数都是奇数；如果有，输出所有满足条件的生成子图中最大边权的最小值 (没有则输出-1)。

1.2 数据范围与约定

对于所有数据，保证 $2 \le N \le 10^5, 1 \le M \le 3 \times 10^5$, 边权 w_i 满足 $1 \le w_i \le 10^9$。

时间限制：$4s$

空间限制：$256MB$

1.3 解题过程

首先，我们证明如下定理：

一个图满足要求，当且仅当图中所有连通块大小都是偶数。

证明：显然定理成立当且仅当其对每个连通块分别成立，因此可以分别考虑每个连通块。每条边都会使点度数和加 2，并不改变奇偶性，而奇数大小满足条件的图如果存在，度数和一定是奇数，这就产生了矛盾，因此奇数大小的连通块一定不满足要求。对于一个偶数大小的连通块，我们任意选择它的一棵生成树，并任意选择一个点为根，然后做如下操作：

1. 选择一个叶子节点，如果它在当前生成子图中的度数是偶数，则将它与它父亲之间的边加入生成子图，否则什么都不做；

2. 在生成树中删除上一步所选择的点，然后回到过程 1，直到生成树只剩下一个点。

显然此时除了树根之外的所有点度数都是奇数，同时由于向生成子图中加边并不改变度数和的奇偶性，根节点的度数也一定是奇数。至此，我们就证明了上面的定理，可以开始着手解决这道题了。

1.3.1 对于单组询问做法

首先我们尝试解决一个弱化版的问题：只需要回答加入最后一条边之后的询问。注意到加入边并不会使答案变劣，我们可以把所有边按权值从小到大排序，依次加入图中，同时使用并查集维护连通块及其大小，直到所有连通块大小都是偶数，此时最后一条加入的边的边权就是答案。时间复杂度为 $O(nlog(n))$。

接下来，我们需要把这个做法拓展到多组询问。这里提供四种解法。

1.3.2 解法一

由于新加入的边不会使答案变劣，在任意时刻，对于每个连通块我们都只需要留下任意一棵生成树上的边即可，其它边不会对答案产生影响。使用 Link-Cut Tree 维护当前生成树森林结构以及大小，同时用大根堆维护所有在森林中的边。每当我们加入一条边后，我们不断地尝试删除当前森林中边权最大的边，直到删除了某条边后出现了奇数大小的连通块，那么这条边的边权就是答案。由于答案是单调不增的，已经被删除的边不可能被重新加入，因此每条边都只会产生 $O(1)$ 次 LCT 上的操作，总复杂度是 $O(mlog(n))$ 的。需要注意的是由于我们需要知道当前是否存在奇数大小的连通块，这里的 LCT 还需要额外维护子树大小。

1.3.3 解法二

与上一个解法类似，这个解法同样需要使用 LCT 来维护当前森林。

首先如果 n 是奇数，答案一定全是-1，这种情况先特判掉。对于一个偶数大小连通块的最小生成树上的边，我们给他们分配一个 0/1 值 val，表示这条边连接的两个部分大小的奇偶性。如果一条边的 val 值为 0，那么这条边就可以被删去，也就是说所有 val 值为 1 的边的边权最大值就是这个连通块的答案。

一开始的时候，我们先加入 $n-1$ 条权值为 $+\infty$ 的边，把整张图连成一棵树，那么整张图就只剩下一个连通块，且大小一定是偶数。如果转化之后某一次询问求出的答案是 $+\infty$，就意味着真正的答案是-1。每当我们将一条边 (u, v, w) 加入时，我们求出当前生成树上 u, v 之间路径上边权最大的一条边 (u', v', w')，如果 $w < w'$，就说明我们需要更新当前的最小生成树，删去边 (u', v', w')，加入边 (u, v, w)，同时更新一些边的 val 值。

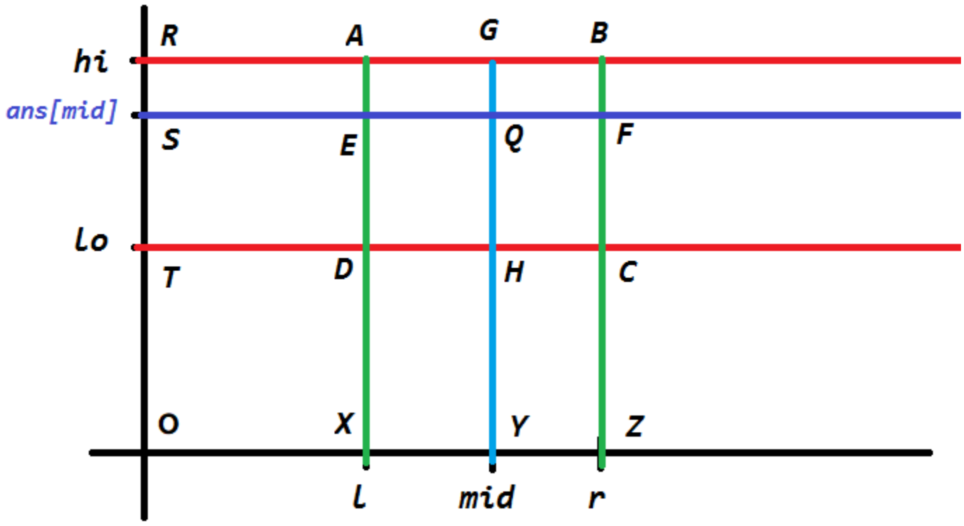
经过观察发现，如果删去的边 val 值为 0，则其它边的 val 值不会发生任何变化；如果删去的边 val 值为 1，则 u 到 v 路径上其它边的 val 值会全部翻转。在 LCT 上维护 val 值翻转的懒标记以及按 val 值分类的边权最大值，即可在 $O((n+m)log(n))$ 的时间复杂度内解决本题。

1.3.4 解法三

如果把-1 看作 $+\infty$，那么答案是单调不增的，因此考虑按时间分治。

如果把一条边的加入时间看作一个点的 x 坐标，边权看作一个点的 y 坐标，那么所有边都对应平面上的某一个点；如果把一个询问的时间看作一个点的 x 坐标，答案看作一个点的 y 坐标，那么每个询问也对应平面上的一个点。维护一个可回退化并查集，维护当前时刻图的连通性。假设当前分治的时间区间为 $[l, r]$，答案区间为 $[lo, hi]$。

开始时，并查集内维护的是由所有在矩形 OXDT 内的边构成的图。首先求出时间 mid 时的答案 $ans[mid]$，具体方法是首先全部加入矩形 DHYX 内的边，然后按边权从小到大加入矩形 HTRG 内的边，直到求出答案，并在求出答案后撤销刚才的并查集操作。



那么 $[l, mid-1]$ 的答案区间为 $[ans[mid], hi]$，对应矩形 AEQG；$[mid+1, r]$ 的答案区间为 $[lo, ans[mid]]$，对应矩形 CFQH。

对于对应矩形 AEQG 的分治区间，矩形 DEST 内的边可以直接全部加入，然后递归求解答案，最后撤销并查集操作；对于对应矩形 AEQG 的分治区间，矩形 DEST 内的边可以直接全部加入，然后递归求解答案，最后撤销并查集操作。

如何分析这个算法的复杂度？分开考虑每条边，它对答案产生贡献的时间是一个区间，类似于线段树的复杂度分析，它最多会被加入并查集 $O(log(m))$ 次。由于并查集需要支持撤销操作，所以复杂度只能做到每次操作 $O(log(n))$，因此总复杂度是 $O(mlog(m)log(n))$ 的，但由于常数小，实际跑得比前面两个 LCT 解法都要快。

1.3.5 解法四

发现每条边产生贡献的时间是一个区间，考虑使用线段树分治。但是怎么求出每条边产生贡献的区间呢？不难发现，如果我们从后往前处理询问，当一条边第一次对答案产生贡献时，就可以确定它对答案产生贡献的区间就是从它出现到现在时刻。同样维护一个可回退化并查集，按时间建立线段树，在线段树上进行 DFS（先走右孩子再走左孩子），从后往前确定答案。

每当我们走到一个节点时，我们向并查集内加入所有标记在这个节点上的边，在离开这个节点时再撤销这些操作，回到走进来时的状态。

每当我们走到一个叶子节点时，从小到大依次加入还没有确定贡献的边，直到确定答案，同时得到了这些边的贡献区间。假设当前时刻为 j，被加入的边为 i，那么就在线段树上 $[i, j-1]$ 这段区间打上标记，表示在处理这段区间内询问时可以直接加入边 i。当我们离开叶子节点时，再撤销之前加入的边。

由于每条边只会被加入 $O(log(m))$ 次，总复杂度为 $O(mlog(m)log(n))$。

1.4 参考文献

https://codeforces.com/blog/entry/21885

https://codeforces.com/blog/entry/21914

https://www.cnblogs.com/galaxies/p/cf603E.html

2 ABC String

题目来源：https://atcoder.jp/contests/agc036/tasks/agc036_e

2.1 题目大意

一个字符集大小为 3 的字符串 S，求出它的一个最长子序列，满足每种字符都出现了同样多次，且任意相邻两个字符不相同。

2.2 数据范围与约定

对于所有数据，保证 $|S| \le 10^6$。

时间限制：$2s$

空间限制：$1024MB$

2.3 解题过程

首先把原串中相邻的相同字符缩成一个。记当前串中 **A**,**B**,**C** 的出现次数分别为 cnt_A, cnt_B, cnt_C，不失一般性地，我们只考虑 $cnt_A \le cnt_B \le cnt_C$ 的情况。

如果把 A 看作分隔符，那么当前串被分成了至多 $cnt_A + 1$ 个段，每段只由 B 和 C 组成。记同时出现了 B 和 C 的段数为 $s1$，只由单个 B 组成的段数为 $s2$，只由单个 C 组成的段数为 $s3$，如果 $s1 + s2 \ge s3$，那么可以按如下方法构造，取得答案的上界 $3cnt_A$：

1. 首先让 $cnt_C = cnt_B$。每次找到一个 C，满足它的两侧分别是 A 和 B，然后把它删去，直到 $cnt_C = cnt_B$。由于所有同时含有 B 和 C 的段对 $cnt_C - cnt_B$ 的贡献都可以通过这个操作减到 -1，同时 $s1 + s2 \ge s3$，这个操作一定可以完成。

2. 接下来，不断找到相邻的 B 和 C，满足删去它们后序列内相邻字符依然不相同，然后把它们删去，直到 $cnt_A = cnt_B = cnt_C$。为什么这样做是合法的？如果当前的序列已经不可以操作，开头和结尾的两段长度最多只剩下 1，中间的 $cnt_A - 1$ 长度至多剩下 2，也就是 $cnt_B + cnt_C \le 2cnt_A$，此时一定有 $cnt_A \ge cnt_B$。

否则，说明如果不删去一些 A，cnt_B 与 cnt_C 永远不可能相等。可以发现删去一个 A 至多只会使 $s3$ 减少 1，且 $s1, s2$ 也是不增的，因此任意删去 $s3 - s1 - s2$ 个 ACA 结构中的 AC 是一种最优的做法。做完这些操作后，现在满足 $s1 + s2 \ge s3$，再使用上面的做法即可。

使用链表维护当前的字符串，总复杂度 $O(n)$。

2.4 参考文献

https://img.atcoder.jp/agc036/editorial.pdf

3 Ball Eat Chameleons

题目来源：https://atcoder.jp/contests/agc021/tasks/agc021_e

3.1 题目大意

N 只变色龙，初始时都是蓝色，依次喂给它们 K 个球，每个球会被随机一只变色龙吃掉。每当一只变色龙吃下的一种颜色的球数量大于另一种时，它就会变成对应的颜色。问有多少种给球染色的方案，满足按这种方案把球依次染色并喂给变色龙后，变色龙有可能全部变为红色，答案对 998244353 取模。

3.2 数据范围与约定

对于所有数据，保证 $1 \le N, K \le 5 \times 10^5$。

时间限制：$2s$

空间限制：$256MB$

3.3 解题过程

首先可以观察得到一个性质：一只变色龙是红色当且仅当它吃的红球数大于蓝球数，或者红球数等蓝球数且最后一个吃的球是蓝球。接下来枚举红色球的总数 R，蓝球总数 B 就是 $K-R$，然后对于每个 (R,B) 分别求解。

我们把 (R,B) 按照大小关系分类：

3.3.1 $R < B$

显然无解。

3.3.2 $R = B$

这种情况下一种方案合法当且仅当最后一个球是蓝球，且能够选出 N 对红球和蓝球，满足每对球内部红球都在蓝球前面出现，且任意一个球都至多只在一对中出现。

如何证明？显然一种合法的方案一定满足这个条件，因此必要性是满足的。关于充分性可以这样构造：除了 N 对中蓝球出现位置最靠后的那对，其余 $N-1$ 对分别喂给 $N-1$ 只变色龙，剩下的那只变色龙吃掉其余所有球。

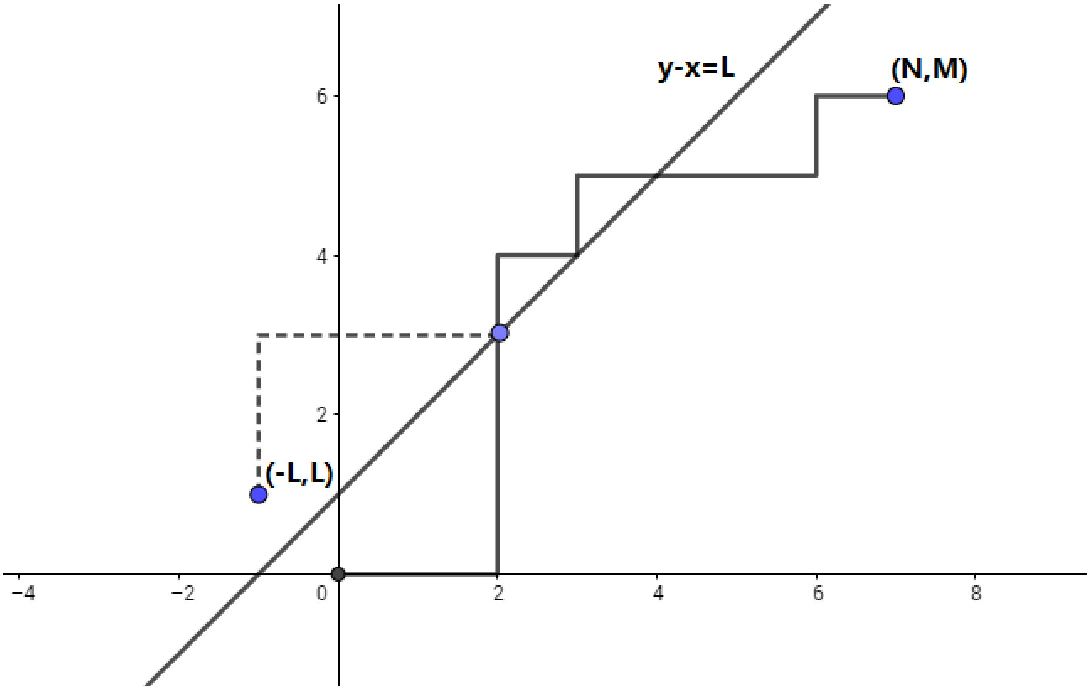
接下来，问题就可以转化为满足条件的方案数量。而满足方案的数量也可以对应到这样一个问题：在平面中，从 $(0,0)$ 走到 (R,B) 且中途经过 $(R,B-1)$，每次只能够向 x 轴或 y 轴正方向走一步，同时要求任意时刻满足 $y-x < B-N+1$，求这样的路径数量。同时由于 (R,B) 处一定满足 $y-x < B-N+1$，因此只需要考虑 $(0,0)$ 走到 $(R,B-1)$ 即可。

3.3.3 $R > B$

与上一种情况类似，这种情况下方案合法当且仅当能够选出 $max(0, N-(R-B))$ 对红球和蓝球，满足每对球内部红球都在蓝球前面出现。满足条件的方案数量同样可以对应到这样一个问题：在平面中，从 $(0,0)$ 走到 (R,B)，每次只能够向 x 轴或 y 轴正方向走一步，同时要求任意时刻满足 $y-x<R-N+1$，求这样的路径数量。

接下来，我们只要能够解决这样一个问题，本题就可以得到解决：在平面中，从 $(0,0)$ 走到 (N,M)，每次只能够向 x 轴或 y 轴正方向走一步，同时要求任意时刻满足 $y-x<L$，求这样的路径数量。

这个问题的解法类似于求卡特兰数，用所有路径数量减去会碰到直线 $y-x=L$ 的方案来计算答案。而对于所有会碰到直线 $y-x=L$ 的路径，我们把从起点到第一次碰到直线的部分沿 $y-x=L$ 翻折 (如下图所示)，那么原本每一条会碰到直线的路径都会唯一对应一条从 $(-L,L)$ 出发到达 (N,M) 的路径，同时由于 $(-L,L)$ 与 (N,M) 不在直线的同侧，所有 $(-L,L)$ 出发到达 (N,M) 的路径都可以对应一条原本会碰到直线的方案。因此答案就是 $\binom{N+M}{M}-\binom{N+M}{M-L}$。



通过预处理阶乘及阶乘逆，可以在 $O(1)$ 的时间内求出一组 (R,B) 的答案，总复杂度 $O(N+K)$。

3.4 参考文献

https://img.atcoder.jp/agc021/editorial.pdf