2023牛客暑期多校训练营#2 题解

A. Link with Checksum

首先观察到，在CRC运算中，每个输入位的翻转，会让最终checksum异或一个固定的值。不妨设翻转从低到高第 i 位时，最终checksum会异或 f_i 。 f_i 可以从低位到高位递推求得，复杂度 $O(32n)$ 。

接下来，考虑填入的答案对checksum的贡献。设固定输入数据对checksum的贡献为 a ，填入的答案第 i 位为 b_i $(0 \le i < 32, 0 \le b_i \le 1)$ ，则该位对checksum贡献的异或值为 f_{i+n_2} 。因此可以列出如下的方程：

$$(\oplus_{i=0}^{31} b_i f_{i+n_2}) \oplus a = \Sigma_{i=0}^{31} b_i \cdot 2^i$$

上述方程中，各个二进制位是独立的，因此上述方程可以拆分为 32 个子方程求解。使用高斯消元、乃至线性基等方法均可求解上述方程。此部分复杂度不超过 $O(32^3)$ 。

> 花絮：
>
> 1. 本题事实上是没有无解情况的，根据出题人对 10^5 以内的 n_2 的枚举，产生的所有方程均是满秩的。但是由于出题人线性代数水平有限，暂时无法给出其必定满秩的证明。若有高手可以证明，欢迎分享给出题组。
> 2. 在验题过程中，有验题人提供了折半搜索的算法，似乎是利用了CRC算法的某些神秘性质，详情可以看萌新版题解。此种算法复杂度与CRC位数相关，使用更多位的CRC可以卡掉，但出题人懒为了让更多同学能通过本题，没有卡掉该种做法。

B. Link with Railway Company

考虑使用最大权闭合子图的算法解决此问题。

为每个运营线路，树上的每条边建一个点。源点向每条运营线路连边，每条运营线路向运营线路所需的铁路连边，每条铁路向汇点连边。运营线路赋 $x_i - y_i$ 的权值，铁路赋 $-c_i$ 的权值。问题即转化为一个标准的最大权闭合子图问题，最大权闭合子图问题的解法在此不再赘述。

在建图时，直接建图会产生至多 nm 条边，可以采用树剖线段树的方式，将边的数量压缩到 $m \log^2 n$ 条，同时可以在重链上建前缀链，实现 $m \log n$ 条边建图。但测试结果表明，上述两种建图方式对算法性能没有明显影响，均可通过本题。

> 花絮：
>
> 1. 出题人本来想把 $m \log^2 n$ 方案建图的算法卡掉，但由于网络流算法性能较不可控，最终放弃了这个想法。

C. graph

考虑一种01赋值的方式，使得后续的导出子图每个点都是偶度点。

可以递归构造：对于原图G,若找不到任何奇度点，那么全部归到一个集合里。

如果找到了奇度点，记这个点为u,由于没有自环，记它所连的点的集合为P,现将P形成的导出子图取补图，删去u,将原图变为G',递归求解G'的一个划分，使得每个导出子图都是偶度点，记P中被赋0的点集合为A,被赋1的点集B，那么$A \cup B = P, A \cap B = \emptyset$,由于$u$是奇度点，因此$|A| + |B|$是奇数，那么$|A|, |B|$必然一奇一偶。不妨设$|A|$是偶数，如果将$u$归入$A$的划分， 那么对于$A$中任何一个点$v$, $deg(v) = |A| - 1 - deg'(v) + 1$是偶数，对于$B$中任何一个点$v$,$deg(v) = |B| - 1 - deg'(v)$也是偶数，因此只要这样构造即可。

D. The Game of Eating

题目大意

- 一共有m道菜，n个人轮流点，一共点k道。
- 第i个人对第j道菜的喜爱程度$A_{i,j}$公开，一个人点了菜所有人都可以吃到。
- 每个人都希望最大化自己的喜爱程度之和，求最终的点菜集合。
- $1 \le n, m \le 2000, 1 \le A_{i,j} \le 10^9, \forall 1 \le x \ne y \le m, A_{i,x} \ne A_{i,y}$

题解

通过样例容易发现，每次贪心地选择自己最喜欢的菜是不优的，因为这道菜也可能成为后面的人的选择，这样我们就浪费了一次机会。考虑如何不浪费这样的机会：

- 假设最后一个人最喜欢的菜在最后还没被选，则最后一个人一定会选它;
- 因此其它人不会浪费机会去选择这道菜;
- 同理，在剩下的菜中倒数第二个人一定会选择他最喜欢的;
- 以此类推，倒过来贪心即可;

猜到这样贪心就可以AC本题，但是正确性的证明比较复杂。下面是由OMG_link给出的一个证明：

- 假设所有人都知道，当剩余菜集合为A时，最后k个人一定会保证$S[A, k]$（即我们的策略）集合内的所有菜都被选。进行归纳:

1. $k = 1$情况显然;
2. $k > 1$时:
   记A中除$S[A, k-1]$以外到数第k个人最喜欢的菜为x:
   1. 假设第k个人选x，得到集合$S[A, k]$
   2. 假设第k个人选$S[A, k-1]$和x以外的菜，根据归纳前提，剩下$k-1$个菜一定是$S[A, k-1]$，因此第k个人一定亏
   3. 假设第k个人选$S[A, k-1]$内的菜，则根据归纳前提，剩余$k-1$人中恰有一人选择$S[A, k-1]$以外的菜，若:
      1. 这个人选的是菜x，则得到$S[A, k]$
      2. 这个人选的不是菜x，则第k个人亏

E. Square

题目大意

多组询问，给定整数x，问是否存在整数$0 \le y \le 10^9$使得y^2在十进制下以x开头。$0 \le x \le 10^9$

题解

枚举y^2的位数，开根判断即可。注意精度。

F. Link with Chess Game

本题是一道诈骗题/猜结论题/分类讨论题。

对于“不能经过重复状态”的博弈问题，通常首先考虑其是否为二分图，若是，则可以使用二分图博弈算法。在本题中，状态转移构成的图是一个边长为 n 的三维立方体，因此显然是二分图。

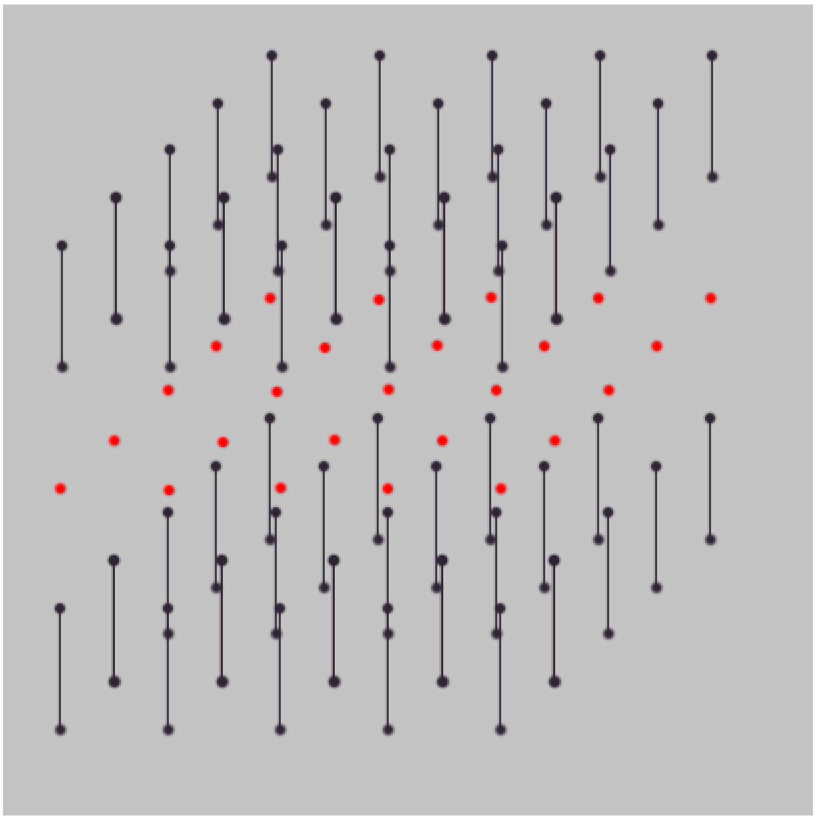
> 为方便不了解二分图博弈的选手阅读，这里给出二分图博弈的结论：
>
> 若起始状态必定位于该二分图的最大匹配上，则先手必胜。否则先手必败。

但是，对于二分图博弈而言，其本身是基于网络流求解的，题目中的 $n = 10^5$ 显然无法用网络流求解。因此考虑观察图中是否有某些特殊性质，可以快速求解博弈的结果。

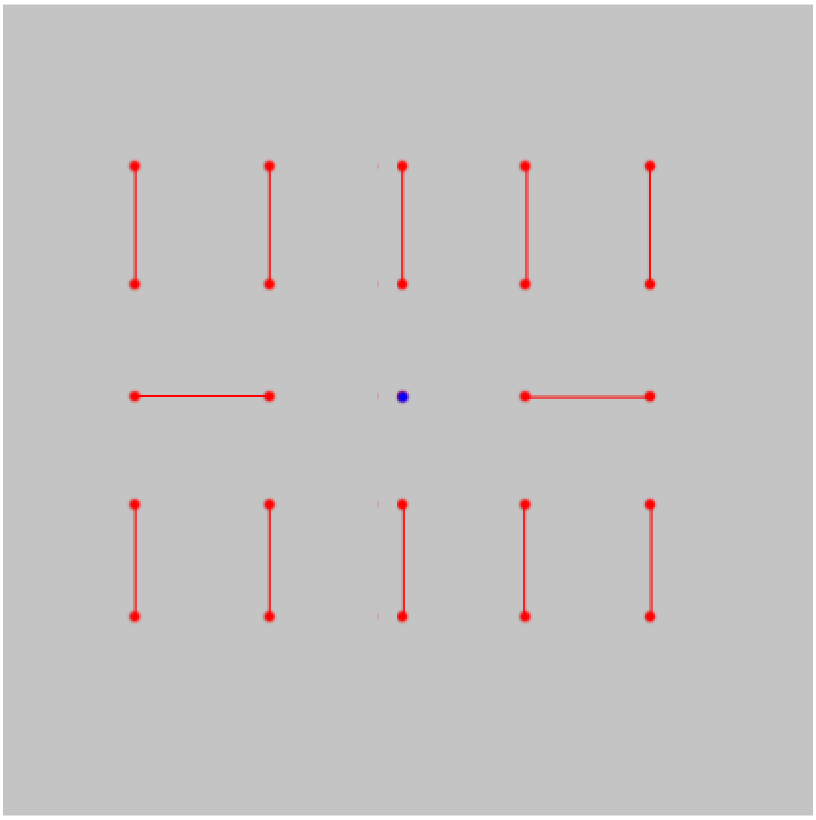
对于 n 是偶数的情况，发现其最大匹配一定是满的，即所有点都位于最大匹配上。因此先手必胜。

对于 n 是奇数的情况，容易构造出只剩下一个点的匹配，即二分图中，数量较少那一部一定全部在最大匹配上，数量较多的那一部有一个点不在最大匹配上。起点在数量较少的一部，先手必胜。下面证明，数量较多的那一部的任意一个点去掉后，最大匹配依然不变，即**任何一个点都不必然在最大匹配上。**

对于数量较多的一部中的任意一个点 (r_0, g_0, b_0) ，其必然满足 $r_0 + g_0 + b_0$ 为奇数。因此，r_0, g_0, b_0 三者中必然有一个是奇数。不妨假设 r_0 为奇数，容易将 $r \neq r_0$ 的部分匹配满，如下图所示：



这样，就去除了 r 这一维，只需要考虑 g_0, b_0 。若 r_0, b_0 均为奇数，则匹配方案如下图所示：



若 r_0, b_0 均为偶数，则可以按上述方案，将除以 (r_0, b_0) 为中心的 3×3 网格以外的部分匹配满。对于 3×3 的网格，显然存在不包含中心点的匹配 4 对点的方案。

综上，证明了任意的一个 (r_0, g_0, b_0) 都可以不在最大匹配上，因此从这类点出发时，后手必胜。

> 花絮：
>
> 1. 题目中的 $\sum n$ 是用来误导你的。甚至有验题人要求将 n 的范围改成 100，让它看起来更像是某种神秘复杂度的做法。
> 2. 有验题人用一堆分类讨论通过了本题，想必正式赛上肯定也有这么过的。
> 3. 这题对棋子的数量**应该**没有要求，**应该**都能构造出满的匹配。

G. Link with Centrally Symmetric Strings

参照Manacher算法的思路，先在每两个字母中间加上#，然后求出新串以每个字符为中心的最长中心对称串，求解方法与Manacher算法相同，只需要将原算法中字符相等的判断改为中心对称的判断即可。

得到上述信息后，一个简单的思路是：从前往后，考虑每个前缀是否是"好的"，对于一个好的前缀，拼接一个中心对称串后，转移到下一个好的前缀。这种DP的思路存在多种复杂度为 $O(n \log n)$ 的转移方法，但由于本题并不打算让此类算法通过，在此就不描述了。

为了在线性时间内解决本题，需要引出一个结论：在从前往后转移的过程中，对于每个好的前缀，**只需要选择其后最短的中心对称串转移**即可。我们将在本题题解的最后附上该结论的证明。

拥有上述结论后，在类Manacher算法的运行过程中，一旦发现某个中心字符的对称半径覆盖到了串的开头，便将该中心对称串截下用于转移，而后忽略该串的所有字符，从下一个没有被对称半径覆盖的字符开始，继续运行类Manacher算法。Manacher算法是线性的，因此上述算法也是线性复杂度的。

下面证明上文用到的结论：

考虑一个中心对称串 $S = AB$，其中 A 是该串最短的中心对称前缀。

首先证明 $|A| \le \frac{1}{2}|S|$：假设 A 的长度超过了 S 长度的一半，不妨将 S 写做 $B'CB$，其中 B 与 B' 中心对称。由于 S 是中心对称串，C 也是中心对称串。由 A 的中心对称性，A 的前缀中含有 C 的中心对称串 C'，于是 A 不是 S 最短的中心对称前缀。

于是可以有 $S = ABA'$，其中 A, B, A' 都是中心对称串。

根据题面中的形式化表述，对于"好的"串的任何一种拆分方式 $S = T_1T_2\cdots T_n$，若 T_i 中含有更短的中心对称前缀，则可以将其拆分为三个更短的中心对称串，直到所有 T_i 都不含有更短的中心对称前缀。此时，按照 T_1 到 T_n 的顺序依次转移，即可判定出原串是"好的"串。

H. 0 and 1 in BIT

题意：给定一个长为 n 的只含 A, B 两种字符的字符串，给定 Q 次询问 (l, r, x) 表示二进制字符串 x 经过字符串 (l, r) 这段区间后变成什么。之中，A 操作反转该二进制字符串（$0, 1$ 互换），B 操作将该二进制字符串视为数字计算 $x = x + 1$（溢出的位舍弃）。数据范围都在2×10^5级别，且询问要求**强制在线**。

根据这个转化，我们发现，可以考虑每个字符对于最终答案的影响，例如 $BABA$ 中第一个 B 对答案的影响是 $+1$（因为后面有两个 A），第一个 A 对答案的影响是 $+1$（因为后面有一个 A），第二个 B 对答案的影响是 -1（因为后面有一个 A），第二个 A 对答案的影响是 -1（因为后面什么都没了）。且顺便也能发现对于一段确定的区间 (l, r) 无论输入的 x 是什么，变化都是 x 先乘以 1 或 -1（取决于这段里 A 的奇偶性），再（在模意义下）加或减同一个常数，我们就是要求出每个区间的这个常数，记要求的东西为 $f(l, r)$。

转化后的问题可以用矩阵加线段树或加倍增或前缀求矩阵逆来维护，不过这里介绍一种十分阳春白雪的前缀和做法。

记 $cnt(l, r)$ 表示 (l, r) 中 A 的数量，可以前缀和预处理。同时我们也可以前缀和预处理出前缀的答案 $f(1, 1), f(1, 2), \dots, f(1, n-1), f(1, n)$。

对于一次询问 (l, r)：若 $cnt(l, r)$ 是偶数，则有 $f(1, l-1) + f(l, r) = f(1, r)$；若 $cnt(l, r)$ 是奇数，则有 $-f(1, l-1) + f(l, r) = f(1, r)$，因此都可以解出 $f(l, r)$。另外，注意 $cnt(l, r)$ 是奇数时，要把输入的 x 先乘以 -1。

本题的强制在线主要是为了防止有人使用 AGC044C 题的做法一下复制黏贴就过了，这也是个很有意思的题（其实也确实是本题灵感来源，经典之想错的题==出了新题），有兴趣可以了解一下。本来不强制在线的，验题时 gkjj拉了这题板子直接秒了，并派出bbg嘲笑我出原题，令出题人恼羞成怒，加了强制在线。

I. Link with Gomoku

签到题。

可能的 Wrong Answer 原因包括：

- 黑子数量不是 $\lceil \frac{n*m}{2} \rceil$
- 白子数量不是 $\lfloor \frac{n*m}{2} \rfloor$
- 存在横、竖、两个对角线方向的五子连珠

一种构造模式如下：

```
x....x....x....
...x....x....x.
.x....x....x...
....x....x....x
x....x....x....
...x....x....x.
.x....x....x...
....x....x....x
```

在上述方案中，无论如何摆放白子，白字都不可能五子连珠。

再加入白子：

```
xo...xo...xo...
...xo...xo...xo
.xo...xo...xo..
o...xo...xo...x
xo...xo...xo...
...xo...xo...xo
.xo...xo...xo..
o...xo...xo...x
```

这样，黑子也无法形成五子连珠。

剩余空位按照双方棋子的剩余数量随意摆放即可。

J. Smoke

题目大意：

给出 $f_{0,1}, \cdots, f_{0,m}$ 以及递推式 $f_{i,j} = pA_{j-1}f_{i-1,j-1} + (1-p)(B_j + C_i)f_{i-1,j}$ 的各个系数，对每个 $n = 1, 2, \cdots, N$ 求 $\sum_j f_{n,j}$。对 998244353 取模。

$1 \leq N, m \leq 10^5$。

题解：

不妨设 n, m 同阶。

题意等价于 $f_{i,j} = pA_{j-1}f_{i-1,j-1} + (1-p)(B_j + C_i)f_{i-1,j}$，求 $\sum_j f_{n,j}$。注意到 $p, 1-p$ 仅仅是常数，且期望不重要，只要在初始值除以 m 即可。为了接下来书写方便，我们不妨设 $f_{i,j} = A_{j-1}f_{i-1,j-1} + (B_j + C_i)f_{i-1,j}$。

进一步化简可以发现，$f_{i,j} = A_{j-1}f_{i-1,j-1} + B_j f_{i-1,j} + C_i f_{i-1,j}$，可以注意到 C_i 的作用仅仅是整体乘上某个定值，并不会影响 $f_{i,j}$ 之间的递推结果，于是如果设 $g_{i,j} = A_{j-1}g_{i-1,j-1} + B_j g_{i-1,j}$，以及 $G(x) = \sum_n (\sum_j g_{n,j})x^n$ 的话，此时 $\sum_j f_{n,j} = [x^n]G(x)\prod_{i=1}^n (1 + C_i x)$。如果我们能求出 $G(x)$，注意到右边的多项式是要乘到 n，此时我们考虑对分治结构进行维护。做 CDQ 分治。具体而言，设 $Solve(l, r, P)$ 表示分治到 $[l, r]$ 时的情况，考虑中点为 mid，设 $P' = P_0 \sim P_{mid-l+1}, F = P * \prod_{i=l}^{mid}(1 + C_i x), P'' = F_{mid-l+1} \sim F_{r-l+1}$，其中 $F_x \sim F_y$ 表示多项式 F 第 x 项到第 y 项按序排列起形成的多项式。先递归 $Solve(l, mid, P')$，再递归 $Solve(mid+1, r, P'')$，对于 $l = r$ 时，答案即为 $[x^1]P * (1 + C_i x)$。此时的总复杂度是 $O(n \log^2 n)$。

于是只用考虑 $G(x)$ 如何求得。

考虑 $G_k(x)=\sum_{i=0}^{\infty} g_{i,k}x^i$，于是有递推式子：

$$
\begin{gathered}
g_{i,j}x^i = xA_{j-1}g_{i-1,j-1}x^{i-1} + xB_jg_{i-1,j}x^{i-1} \\
\sum_{i=1}^{\infty} g_{i,j}x^i = xA_{j-1}G_{j-1}(x) + xB_jG_j(x) \\
\implies (1-xB_j)G_j(x) = g_{0,j} + xA_{j-1}G_{j-1}(x) \\
G_0(x) = \frac{g_{0,0}}{1-B_0x} \\
\implies G_k(x) = \sum_{j=0}^{k} g_{0,j}\frac{1}{h_k(x)}\prod_{i=j}^{k-1}\frac{f_i(x)}{h_i(x)}. \\
f_i(x) = xA_i, h_i(x) = 1-B_ix
\end{gathered}
$$

于是答案等价于

$$
\begin{aligned}
&[x^n]\sum_{k=0}^{m}\sum_{j=0}^{k} g_{0,j}\frac{1}{h_k(x)}\prod_{i=j}^{k-1}\frac{f_i(x)}{h_i(x)} \\
&= [x^n]\sum_{j=0}^{m} g_{0,j}\sum_{k=j}^{m}\frac{1}{h_k(x)}\prod_{i=j}^{k}\frac{f_i(x)}{h_i(x)} \\
&= [x^n]\sum_{j=0}^{m} g_{0,j}\sum_{k=j}^{m}F_k(x)
\end{aligned}
$$

可以发现等价于是每个区间都乘起来，同时左右端点处有特殊贡献。

考虑分治，每个节点维护一个四元组，分别表示 (区间多项式积的和 A，固定左端点到区间最右的多项式积的和 L，固定右端点到区间最左的多项式积的和 R，区间多项式的积 S)，注意这些多项式本身都是二元组，即有分子和分母。发现可以合并，即设两个多元组为 $(A_l,L_l,R_l,S_l),(A_r,L_r,R_r,S_r)$，则可以合并成 $(A_l+A_r+L_l*R_r, L_r+L_l*S_r, R_l+R_r*S_l, S_l*S_r)$，时间复杂度为 $O(n\log^2 n)$。

具体实现时，不选择维护二元组而选择统一维护分母并通分是比较合理的，此时多项式的次数将会大幅减少。

K. Box

题目大意：

给出一个长度为 n 的 01 串以及每个位置都有权值 $a_i\ge 0$，每个 1 最多移动一次且最多移动一位。若一个位置有 1 则可以获得权值，问最大可能的权值和。

$n\le 10^6$。

题解：

$f_{i,0/1,0/1/2}$ 表示前 i 个位置，当前位置是否已经盖有 1，下一个位置是否也盖有 1，有的话是否从当前位置移动过去的最大权值和，转移讨论即可。时间复杂度 $O(n)$。

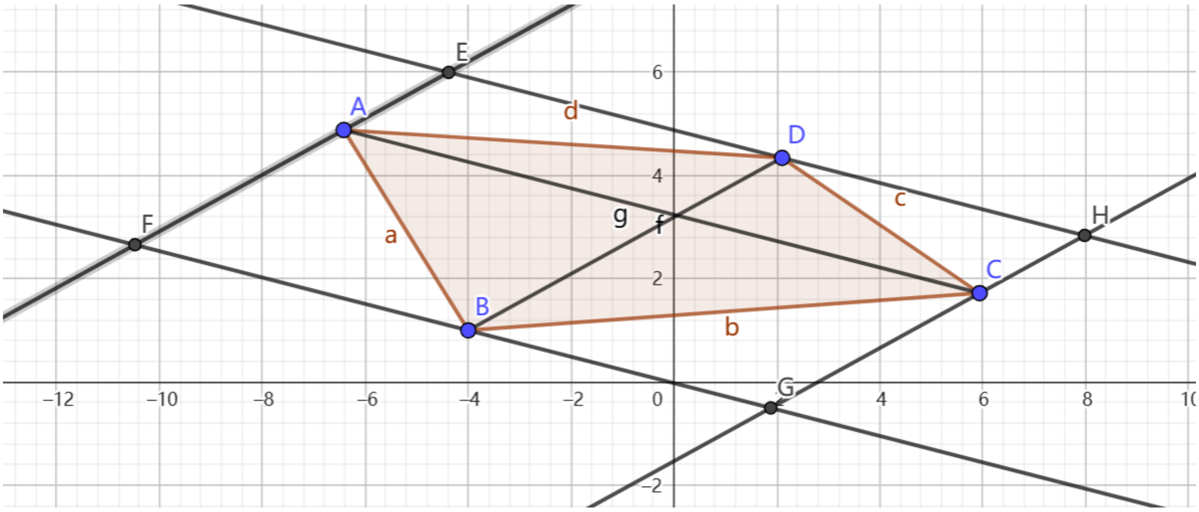
L. Link doesn't want to cut tree

题目大意

多组询问，给定矩形，构造外接菱形使其面积位矩形两倍。需要判断是否无解。

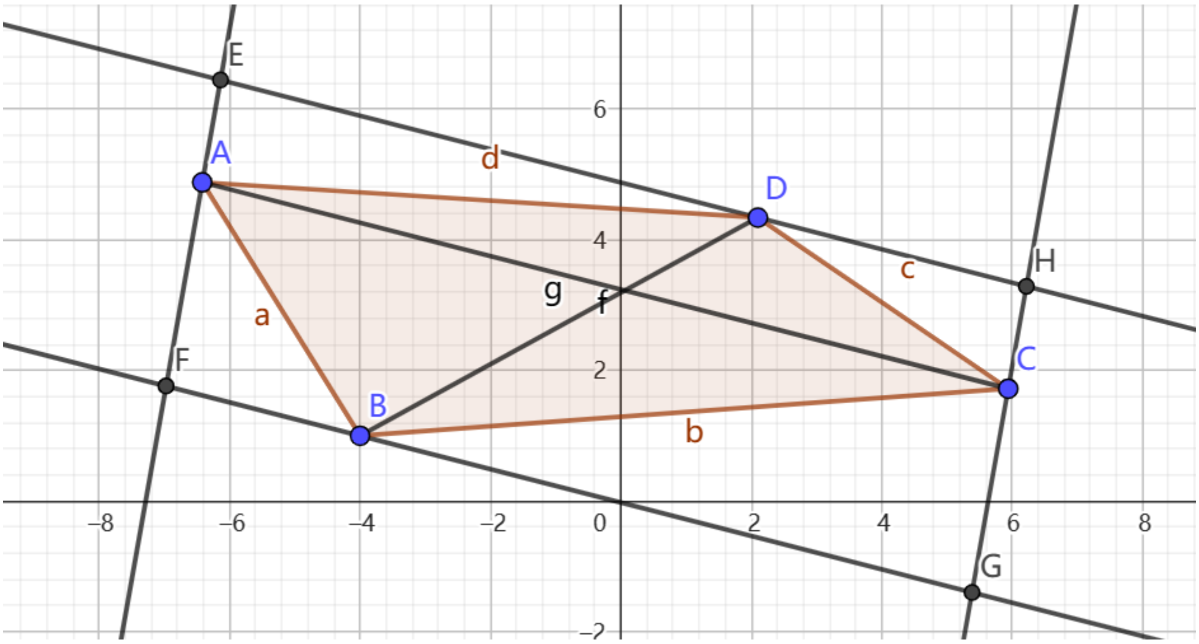
题解

设原矩形面积为S。直接构造菱形比较困难，但是构造一个面积为$2S$的平行四边形相对简单，可以直接通过平移对角线完成：

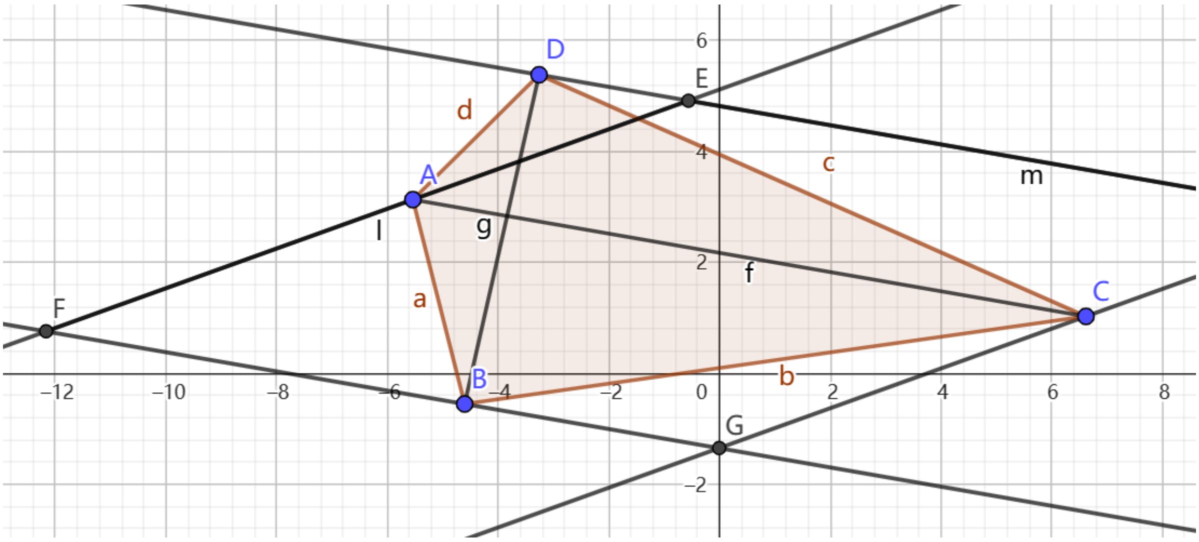


容易证明，平行四边形EFGH的面积为ABCD的二倍。

接下来分析如何利用EFGH来构造结果。发现旋转一条对边可以使平行四边形面积不变，并且边长发生变化，下图为旋转EF,GH两条边后的图形：



通过上述旋转操作，计算出$|EF| = |GH|$时对应的旋转角度即可得到$A'B'C'D'$。需要枚举两组对边*2个旋转方向共四种情况，并判断是否会出现下图中的非法情况，合法的直接输出：

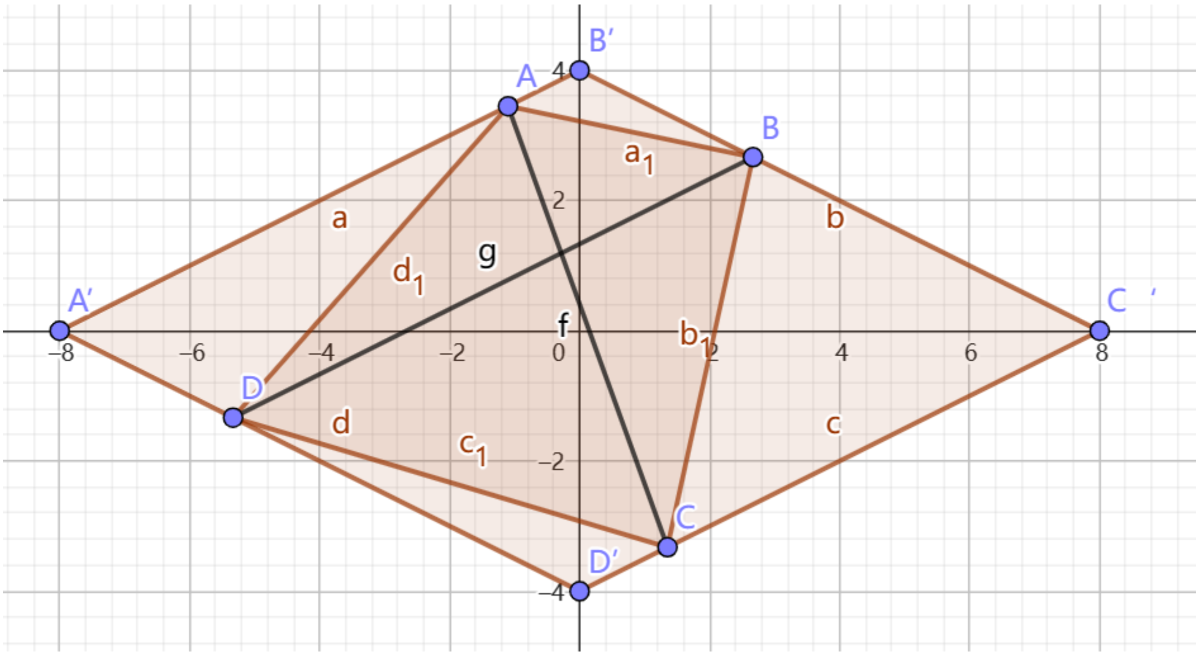


下面说明通过上述方法一定可以构造出解：

- 在$A'B'C'D'$的一组对边与$ABCD$的一条对角线平行时，上述方法都可以构造出结果。
- 那么只要证明"$S_{A'B'C'D'} = 2S_{ABCD}$时，$A'B'C'D'$一组对边一定与$ABCD$的一条对角线平行"即可。

我们首先固定$A'B'C'D'$，然后尝试构造$ABCD$使$S_{ABCD} = \frac{1}{2}S_{A'B'C'D'}$

- 当$AC//B'C'$时，显然有$S_{ABCD} = \frac{1}{2}S_{A'B'C'D'}$
- 当AC与$B'C'$不平行时，对任意给定的B，S_{ABC}面积固定，S_{BCD}随D的变化单调变化，因此仅当$BD//A'B'$时$S_{ABCD} = \frac{1}{2}S_{A'B'C'D'}$



M. Fundamental Skills in Data Structures

题目大意

n个点的有根树，每个点上有点权a_i为0或1，要求进行两种操作：

- 链点权覆盖为0或1
- 查询子树内满足 $x < y$ 且 $a_x \oplus a_y \oplus a_{lca(x,y)}$ 的点对 (x, y) 的数目

思路一

hint 1

使用什么思路统计答案?

考虑到点对的限制条件与lca相关，我们可以考虑在lca处统计点对的数量。如果进行了树链剖分的话，我们就能将点对按照所在位置分为重-轻和轻-轻两类，并分别统计数目。

hint 2

回答询问的计算方式是什么?

由于我们要求的是子树的答案和，我们可以在每一条重链上直接统计lca在该链上的合法点对的数量和。子树和即为该子树内部的每条重链和的总和。

hint 3

有什么数据结构能够更好的处理该问题?

树剖线段树当然能够处理，然而，考虑到复杂度与操作灵活性的问题，我们在该题目中选择使用[Link Cut Tree]()。我们在虚边上传递子树的答案以及维护答案所需的额外信息，在实边上计算链贡献总和。

hint 4

如何维护操作对答案的影响?

考虑到链覆盖只有覆盖为0和覆盖为1两种，一种比较方便的处理方法是，每条实链上直接同时维护全0情况下的信息和全1情况下的信息，在链覆盖时直接使用对应的信息替换当前信息。

做法

在以上思考的基础上，我们考虑如何维护实链涉及的答案和。

容易发现，点对分为两类:

- lca与某个端点重合，或者两个端点均位于lca虚儿子的子树中：此时我们只需要在虚边传递的过程中直接计算即可。
- lca其中一个端点位于实儿子的子树中，另一个端点位于虚儿子的子树中：此时我们考虑，将虚儿子处端点的信息绑定到lca上，另一边剩下一个实端点。对于整条实链，使用类似分治的思想，即每次考虑跨过某个节点的lca-实端点点对的数量，最终求和。

由于lct本身一条实链即为一个splay维护的二叉树结构，天然地形成了一个分治结构，因此我们统计实链上答案的时候，在向上合并信息时，计算当前左子树（即较浅的一边）含有的lca和右子树含有端点的合法点对数目即可。

具体的，我们在每个节点处，统计$pcnt_0$和$pcnt_1$，分别表示虚子树内部节点和当前节点的权值异或值为0和1的节点数目。同时维护cnt_0和cnt_1，代表当前虚子树中权值为0和1的节点数目。

在向上传递信息的过程中，我们对这两个信息进行求和，这样得到的就是实链上某一段的节点权值统计数和与lca异或后的节点权值统计数。在当前节点x，我们取出左子树的$pcnt$和右子树的cnt对应相乘，即可得到在这一段实链上跨过点x的答案总和。注意计算时还要额外考虑端点为当前节点的答案与其他信息。

在虚边处，我们将该实链统计出的实链总信息向上传递到父节点中，父节点同时处理第一类点的答案计算。

为了处理子树询问，我们还需沿虚边向上传递子树答案的信息，注意虚实边切换细节的处理。实际回答时，我们只需要对子树根节点u进行access操作，将u下面的边全部变为虚边，并利用虚边上传递的信息计算出答案。

最后，我们使用一个比较方便实现的做法维护链覆盖，即同时维护全0和全1的各类信息，在覆盖操作的时候直接进行信息的替换。这里有一个细节是，我们可以通过两段access到根节点，但仅进行lca的后缀处理的方式来取代原本lct中先换根再access的做法，这样可以避免换根操作需要维护的额外代价。

最终做法的时间复杂度为 $O(n \log n)$，且LCT相比同一个思路的树剖做法更加好写。

思路二

鸣谢RDDCCD与lzoilxy提供的更加简单的其他思路。

hint 1

使用什么思路统计答案?

考虑到点对的限制条件与lca相关，我们可以考虑在lca处统计点对的数量。

hint 2

回答询问的计算方式是什么?

考虑lca节点处的权值情况，我们发现，若lca处权值为0，实际上是统计有多少对权值相同的点位于不同子树内，若lca处权值为1，则统计的是权值相反的点对。

统计点对数可以使用容斥的思想，以lca处0权值为例，可以先统计出整棵子树内权值相同的点对数量，再减去每棵子树内部的点对数量。而子树和即为子树内每个点处计算出的值的和。

hint 3

该统计方式有什么额外的性质?

观察计算过程，我们发现，如果一个点和父亲具有相同的权值，则在表达式中它自己的点对数量被抵消掉了。也就是说，实际上答案的贡献仅在每一条连接两个不同权值的点的边上产生。我们只需要分别维护子树内对应权值的节点数量信息，以及每条边产生的答案贡献，即可求得答案。

hint 4

有什么数据结构能够更好的处理该问题?

我们可以使用树剖线段树或LCT来维护边的贡献，同时还需要一个类似的数据结构来维护子树内不同权值的节点数量。

hint 5

如何维护操作对答案的影响?

考虑到每次执行链覆盖，都相当于将一部分贡献边转化为非贡献边，同时在边缘产生一些额外的贡献边。我们可以在重链上直接进行贡献清除，同时在虚边处维护可能产生的额外贡献，使用节点数量的信息进行计算。同时，还需要将节点数量信息的变化向上直接传递到根节点。