CF538G

重要的一步是将坐标系旋转 45°，那么 `UDLR` 转化为 $(1,1),(-1,-1),(1,-1),(-1,1)$，于是两维独立。

但是每一步 ±1 还比较麻烦，视为当成每一时刻固定 -1 后选择是否 $+2$，可以转化为每一步选择不动还是 $+1$。

转化为一维坐标上的问题，假设一个周期内移动了 K，那么条件“t 时刻在 x”等价于条件 $t \bmod l$ 时刻在 $x-\lfloor\frac{t}{l}\rfloor K$”，于是 $t \bmod l$ 相邻的两个事件之间坐标之差属于一个区间，每个区间对 K 产生一个限制，看所有限制的交是否为空，对应构造即可。

P7320

看到图和独立集考虑 dfs 树（因为 dfs 树的树边和返祖边好刻画相连关系）。

第一个条件是 树-friendly 的，因为它可以忽略掉多余的边。

事实上，一棵树能被 k 条路径覆盖的充要条件是这棵树至多有 $2k$ 个叶子。

> 证明：随便选一个非叶节点为根进行 dfs，将叶子按 dfs 序标号，则选出所有叶子 i 到叶子 $i+k$ 的路径即可。
>
> 为什么这是对的？首先，将树画出来，可以发现这些路径之间两两有交；然后，假设一个点没有被覆盖，那么这个点一定不是叶子，将这个点删去后，树分成多个部分，每一部分一定都有叶子，由于该点未被覆盖，所以每部分的叶子互相配对，于是不同部分之间的路径之间无交，矛盾！

然后发现 dfs 树的所有叶子构成一个独立集，就做完了。

ARC103F

由于给的信息是每个点关于树的整体信息，所以应考虑树上的特殊点，具体地就是剥叶子。

事实上，整个树中 d_i 最大的点一定是叶子，然后发现树上相邻两个点之间 d_i 的差值是可以简单计算的（只与这条边切开后两边的 $size$ 有关），所以可以判断叶子与哪个点相连，递归处理即可。

注意这里我们只用到了 d_i 间的差值的信息，所以最后要随便找个点验一遍。

CF611H

如果把形成树的条件刻画成“连 $n-1$ 条边使整张图连通”就寄了，应刻画成“连 $n-1$ 条边使整张图中没有环”。

（似乎用后者的条件刻画树总是好过前者的，因为“环”的判定比“连通”更具有局部性）

把十进制位数相同的点包装成一个元素，模仿 Hall 定理，可以证明，合法当且仅当每个子集内部的边数 $<$ 该子集中的点数（记为限制 σ）。

> 证明：如果（除了全集外）没有一个子集内部边数 $+1=$ 该子集中点数，那就随便连一条边；否则，随便取出一个满足“子集内部边数 $+1=$ 该子集中点数”的子集 S，归纳可证 S 内部一定可连成一棵树，那么枚举外面的元素 A，设 S 和 A（即若干个十进制位数相同的点）连了 k 条边，则将 A 中拉出 $k-1$ 个单点和 S 连边，之后与 S 相邻的元素都只连了一条边，因此可以把 S 缩成一个点。而新的图必然仍满足限制 σ：新图中的子集 T 满足限制 σ 等价于原图中子集 $S\cup T$ 满足限制 σ。归纳即可。

故判断是否合法可以 $2^{\log_{10} n}\approx n^{0.3}$（即 2^6）枚举子集数边数。

但是没法直接构造，因此可以每一轮暴力枚举所有边尝试删掉后判断是否合法，复杂度 $O(n^{1.3}\log_{10}^2 n)$，应当足以通过。

CF1508D

"线段互不相交"是个很复杂的条件，考虑将其简单化。思路是先把图画出来，然后再根据 a_i 规划画线段的顺序。

怎样的图保证线段互不相交？最简单的是一个菊花。

巧的是，如果排列 a_i 仅形成一个置换环，那一个菊花就够了，足以将 a_i 复位，方案自行思考。

问题是原图不一定只有一个置换环，自然的想法是每个置换环分别做，但是这样很难保证不同菊花之间不相交，于是考虑改为使用几次操作把所有 a_i 并入一个置换环内再操作。

只需要合并置换环的操作不与菊花相交（自己之间也不相交）即可，考虑到每个点只要被碰过就可以进入同一置换环，所以在菊花外面至多包一圈就可以保证不相交。

实现上就是找出一个定点，按极角顺序依次枚举每个点看是否在置换环中，如果否就与上一个点连一条边并加入。

但是"菊花外面包一圈可以保证不相交"是有条件的，当选的这个定点是 x 坐标最小的时才能保证这一点。反例留做练习/doge。

CF741C

经典题目，就不说怎么想到了。属于必须见过才会的题。

将每对点之间连边，则要求连边的点颜色不同。

加连边 $(1,2),(3,4),\dots,(2n-1,2n)$，则"连边的点颜色不同"即保证了相邻三个点颜色不全相同。

由于现在这张图由两个匹配叠加而成，所以图是若干个偶环。给每个偶环黑白染色即可。

ARC131E

由于题目只有一个 n，考虑递归处理。

如果一个点连出去的边全是同一种颜色，那这个点可以直接删掉，规约到 $n-1$ 的情况。

但是这会破坏三种边数量相等的限制，所以每次连删 6 个点规约到 $n-6$，这 6 个点依次向剩下的点连 $1,2,3,3,2,1$ 的边，这就可以保证与这 6 个点相关的 $1,2,3$ 边相等。

小情况直接暴搜/手玩即可。

CF1270E

将问题强化的同时简化：我们要求 P 与 Q 中的数在模意义下不同。

要想弄出"取模"，一个自然的想法是黑白染色。这个想法也可以从欧几里得距离（的平方）的表达式 $(x_1-x_2)^2+(y_1-y_2)^2$ 走过来：众所周知 $a^2\equiv 0,1 \pmod 4$，所以模 4 是一个好的主意，因此需要对 x_1-x_2 和 y_1-y_2 的奇偶性分讨，这就会联想到黑白染色。

事实上，如果点集中既有黑点又有白点，那直接黑白染色就是对的。可以自行计算。

否则不妨设点集中只有黑点，那么将坐标系旋转 45° 后缩小至 $1/\sqrt{2}$ 倍，递归处理即可。

AGC027D

如果谁 mod 谁不固定的话构造会比较复杂，因此将棋盘黑白染色，我们钦定每个黑格 mod 周围的白格为定值。

（由中国剩余定理）这就等价于每个黑格 mod 周围白格的 lcm 为定值。

所以我们只需要给每个白格赋一个值，使得任意黑格相邻的四个白格的 lcm 小于 10^{15} 即可。

但注意题目还要求格子里的数互不相同，所以黑格中数的大小不大容易控制。要想控制 lcm，一个自然的办法是考虑素数。同时两个数不同当且仅当它们素数分解不同，因此考虑我们直接控制每个白格的素因数分解。

如果每个白格都是素数，那我们需要 $O(n^2)$ 个素数，黑格中的数是 $O(n^8)$ 的，无法接受。

为了减少素数的使用，我们需要白格有相同的素数。为了这一相同能够 pass on 到黑格上使黑格数也比较小，我们希望把有相同素数的白格靠近放。

接下来可能是一个应记住的想法~~其实我自己估计想不到~~，就是我们应将素数按对角线摆放，这样只用了 $O(n)$ 个素数，每个黑格也只接触了 2 个素数。

为了解决白格中的数有重复，将素数按正副对角线摆放，每个白格为正副对角线上的素数的乘积，每个黑格的数是 $O(n^4)$ 的，可以通过。

CF804E

由于题目只有一个 n，考虑递归处理。

首先用奇偶性可知 $n \bmod 4 = 1$ 或 0。

子操作：对于 $1, 2, \ldots, k$，将 k 与 $k-1, k-2, \ldots, 1$ 依次操作一遍可以达到将 k 循环移位至开头的效果。

于是如下操作：

1. 将 $1, \ldots, n-4$ 操作一遍
2. 将 $n-3$ 与 $n-4, \ldots, 1$ 交换一遍，此时 $n-3$ 到了开头
3. 将 $n-3$ 与 $n-2$ 交换
4. 将 $n-2$ 与 $1, \ldots, n-4$ 交换一遍，此时排列为 $1, 2, \ldots, n-4, n-2, n-3, n-1, n$
5. 如前法炮制，得到排列为 $1, 2, \ldots, n-4, n-2, n-3, n, n-1$

此时我们只需要将 $n-2, n-3, n, n-1$ 交换至原位即可，其中 $(n-2, n-3)$ 和 $(n, n-1)$ 已交换过。这就是 $n = 4$ 的情况，手玩解决即可。

AGC037D

初状态杂乱而末状态整齐，考虑操作逆序。

最后一步操作逆过来，就是变成第一行是 $1, 2, \ldots, m$ 的排列，第二行是 $m+1, m+2, \ldots, 2m$ 的排列，$\ldots$，第 n 行是 $(n-1)m+1, \ldots, nm$ 的排列。

再将第二步操作逆过来，就是变成每一列有 $1 \sim m, m+1 \sim 2m, \ldots, (n-1)m+1 \sim nm$ 的数各一。

将 $1 \sim m$ 染成颜色 1，$m+1 \sim 2m$ 染成颜色 2，$\ldots$，$(n-1)m+1, \ldots, nm$ 染成颜色 n，则我们的要求变为：将初始状态的每一行重排，使得每一列有颜色 $1 \sim n$ 各一。

仔细想想这其实就是个二分图匹配，n 个左部点 m 个右部点，每个左部点连出 n 条边，这是一个正则二分图匹配，由 Hall 定理一定有解。暴力求解或使用随机化优化即可。

AGC012C

构造 10^{12} 级别的数，考虑 $\times 2+1$ 或者二进制拆分。

但是"子序列是平方串"这个条件极其麻烦：一方面，添加一个数会增加 2^n 个子序列，难以保证其中几个是平方串；另一方面，很难将两个串拼接起来做到加法。

由于二进制拆分往往需要很"大"的构造，而本题串长限制较紧，所以主要考虑 $\times 2+1$。

这题我们是比较自由的，所以我们要自加限制。

一个想法是对象切分：我们要满足"平方串"，那我们就把串切成两部分 A 和 B，要求 A 和 B 内分别无平方串，那我们数的就只是 A 和 B 的公共子序列个数了。

事实上，这要求 A 和 B 都是排列。不妨设 B 为 $(1,2,\dots,100)$，那我们就只需要构造 A 满足 A 的上升子序列个数为 n 即可。

这是经典问题：从小到大往 A 中加数，加在后面就是 $\times 2$，加在前面就是 $+1$。

Note：即使到最后一步，如果用很显然的二进制拆分，也会发现构造的串长太大（当 $n=2^{12}-2$ 是会爆）。这也体现了 $\times 2+1$ 往往比二进制拆分优越。