A. Oops, It’s Yesterday Twice More

注意到我们可以通过 $n-1$ 次左（右）移，再加上 $n-1$ 次上（下）移将所有袋鼠移到一个角落，然后正常沿曼哈顿距离走到目标点即可。选择最靠近目标点的角落作为集结点，则可在限制步数内完成。

B. Puzzle in Inazuma

通过对两个图的每一条对应的边的权值做差，可以把题意转换为给定一个图 G ，询问是否可以通过有限次操作把所有边权都变成 0 .

我们先来讨论 $n \geq 6$ 时的情况。

首先不难发现，每次操作后总和必须一样。接下来我们说明只要总和一样的时候，图 G 一定可以变成图 H .

我们假设图中存在 6 个点 A, B, C, D, E, F ，我们进行如下一系列操作:

- 将 DA, DB, DE 边上的值 -1 ，将 AB, BE, EA 边上的值 $+1$.

- 将 DA, DB, DC 边上的值 $+1$ ，将 AB, BC, CA 边上的值 -1 .

- 将 EA, EC, ED 边上的值 $+1$ ，将 AC, CD, DA 边上的值 -1 .

- 将 EA, EB, EC 边上的值 -1 ，将 AB, BC, CA 边上的值 $+1$.

经过以上操作后，我们将 AB, AE 边上的值增大了 1 ，将 AC, AD 边上的值减小了 1 。

类似地，我们可以将 AB, AE 边上的值减小 1 ，将 AC, AF 边上的值增大 1 。

从而我们可以通过有限次操作将 AD 边上的值减小了 1 ，并将 AF 边上的值增大了 1 。

因此 $n \geq 6$ 的时候只要总和一样，则一定可以在有限步之内将图 G 变成图 H 。

接下来讨论 $n = 5$ 时的情况，

通过 4 次操作，我们可以将 AB, AE 边上的值增大了 1 ，将 AC, AD 边上的值减小了 1 。

类似地，我们可以将 AB, AC 边上的值增大了 1 ，将 AD, AE 边上的值减小了 1 。

从而我们可以通过有限次操作将 AD 边上的值减小了 2 ，并将 AB 边上的值增大了 2 。

因此我们只需考虑奇偶性，可以通过 $O(2^{\binom{n}{2}})$ 的复杂度完成。（其实可以优化成 $O(2^n)$ ）

最后讨论 $n = 4$ 时的情况，相当于 4 个未知数， 6 个方程，直接找有没有解即可。

C. Klee in Solitary Confinement

我们枚举最后众数为 x ，则每次只需要单独考虑 x 和 $x+k$ 。我们事先可以将每个数按数值大小，将位置插入vector，则可做到均摊 $O(n)$ 。如果使用map或者别的容器实现，则有运行超时的风险。

现在问题转化成有一个长度为 m 的序列，序列仅由 X 和 Y 组成，用 $X_{l,r}$ 和 $Y_{l,r}$ 表示区间 $[l,r]$ 里 X 和 Y 的个数，则我们需要选择一个区间 $[l,r]$ ，使得 $X_{1,l-1}+Y_{l,r}+X_{r+1..m}$ 最大。

简单转化一下，则对于每一个 r ，我们需要最大化 $X_{1,l-1}+Y_{l,r}+X_{r+1,m}=X_{1,l-1}+(r-l+1)-X_{l,r}+X_{r+1,m}=2\times X_{1,l-1}+(r-l+1)-X_{1,r}+X_{r+1,m}$ 。整理得到 $(2*X_{1,l-1}-l)+(r+1-X_{1,r}+X_{r+1,m})$ ，即最大化 $2*X_{1,l-1}-l$ ，记录前缀最大值转移即可，时间复杂度 $O(m)$ 。

综上，时间复杂度为 $O(n)$ 。

D. Paimon Sorting

先考虑序列所有元素各不相同的情况。

首先从第一个元素开始，找到并跳到当前元素右边第一个比它大的元素。称这些元素为``特殊元素''。可以发现外层循环的第一轮会把所有特殊元素``右移''一位，此时序列中最大的元素就到了序列第一个。

从外层第二轮循环开始，对于第 i 轮循环，序列前 $(i-1)$ 个元素已经是有序的。设此时第一个比 a_i 大的元素是 a_k ，那么第 i 轮循环将会发生 $(i-k)$ 次交换。因此，非特殊元素将会贡献``前面有几个数比它大''次交换（注意一开始特殊元素的右移对答案没有影响，因为虽然移走了一个比它大的元素，但是最大的元素移到了开头），而特殊元素将会贡献 2 次交换。

接下来考虑存在相同元素的情况。

对于非特殊元素，前面比它大的，但是相同的元素只会发生一次交换，因此需要对元素去重一下再统计。另外，如果这个非特殊元素恰好等于上一个特殊元素，那么上一个特殊元素的右移会导致该非特殊元素的贡献增加 1 。

实际解题过程中，由于暴力程序非常简单，可以考虑通过对拍发现并调整 corner case。

E. Paimon Segment Tree

我们可以通过差分询问时间，将单个询问 $[x, y]$ 转化为两次询问 $[0, y]$ 和 $[0, x-1]$ 的结果之差。再通过离线询问，我们不需要实际记录每个时间节点的信息，空间上就不容易超限了。同样，为了对齐时间，我们将 $\{[l, r], k\}$ 视作3次操作 $\{[1, l-1], 0\}, \{[l, r], k\}, \{[r+1, n], 0\}$ 。

我们考虑使用线段树及懒标记直接维护当前时间的区间信息，那么我们只需要考虑标记下放和区间合并。

对于每个节点 $[l, r]$ ，我们维护以下信息：

$x_1 = r - l + 1$ ，区间长度；

$x_2 = \sum_{i=l}^{r} a_i^t$ ，线性和；

$x_3 = \sum_{i=l}^{r} {a_i^t}^2$ ，平方和；

$x_4 = \sum_{j=0}^{t} \sum_{i=l}^{r} {a_i^j}^2$ ，历史平方和；

对于每次增量 k ，我们有以下的转移关系：

$x_1 = x_1$ ；

$x_2 = x_2 + x_1 \times k$ ；

$x_3 = x_3 + 2 \times k \times x_2 + k^2 \times x_1$ ；

$x_4 = x_4 + x_3 + 2 \times k \times x_2 + k^2 \times x_1$ ；

我们如果考虑一个向量 $\{x_1, x_2, x_3, x_4\}$ ，则可写出以下转移矩阵：

$$\begin{pmatrix} 1 & k & k^2 & k^2 \\ 0 & 1 & 2k & 2k \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

因此，我们只需要将转移矩阵作 4×4 为标记下放，即可用线段树维护当前信息，这样复杂度是 $q \log n$ 的。我们注意到矩阵大小为，则每次矩阵乘法运算次数为 4^3 ，常数比较大。经过分析，我们会发现有效转移只有从 x_i 到 $x_j (j > i)$ ，则实际只需要求出 $C(4,2) = 6$ 个系数即可，常数大幅减少。在题目的时限设置中，我们设置了一个让不需要任何矩乘优化就可勉强通过的时限。

F. Paimon Polygon

本题总共分为两种情况： A 和 B 包含或相离。

对于包含的情况，需要满足所有点都在某条直线的一侧。我们可以先对所有点构建一个严格凸包 A ，再利用剩下的点构建一个严格凸包 B ，同时要求不存在 B 内侧的点。注意需要特判共线的情况，即如果存在三点共线 x, y, z ， A 中会自动选取两侧的点 x, z ， B 中则选取了剩下的点 y ，但其实这种情况是不合法的（有除了 O 以外的新的交点了）。当然我们也不能简单地认为三点共线一定不合法，如果 y 外侧还有一个点 y' ，那么 A 选择 x, y', z 、 B 选择 y 是可能合法的。

对于相离的情况，我们首先可以尝试对所有点构建一个凸包。如果构建失败，违反的三个点处必须划成两个集合，然后以此为起点左右扫一遍即可；如果构建成功，我们可以做一个类似旋转卡壳的操作，把 360 °的点划成两段连续的集合，一段给 A 一段给 B ，同时要求每段的夹角不能大于等于 180 °。

G. Paimon’s Tree

考虑一个弱化的问题：假设在树上指定一条路径，求这条路径的最大长度。

树上除了在指定路径上的边，还有一些不在指定路径上的边可以让我们放``垃圾''。例如我们要求样例 1 中路径 $1 \to 3 \to 4$ 的最大长度，节点 3 就有 1 条边可以放垃圾，节点 4 有 2 条边可以放垃圾。如果指定的是路径 $1 \to 3 \to 2$ ，那么节点 3 就有 3 条边可以放垃圾，而节点 1 与 2 都没有边可以放垃圾。

设指定的路径有 k 个节点，其中第 i 个节点可以放 b_i 个垃圾，我们就可以进行区间 dp。记 $f(l,r,t)$ 表示已经完成了路径中第 l 到第 r 个节点之间的边的赋值，而且已经从序列 A 中用掉了 t 个数（也就是说树里已经放了 $(t-(r-l))$ 个垃圾）。有如下转移方程：

$$f(l,r,t) \to \begin{cases} f(l-1,r,t+1)+a_{t+1} & \text{if } l>1 \\ f(l,r+1,t+1)+a_{t+1} & \text{if } r<k \\ f(l,r,t+1) & \text{if } (t+1)-(r-l) \le \sum\limits_{i=l}^{r} b_i \end{cases}$$

分别是往左或右拓展一条边，或再塞一个垃圾到目前的节点里（要能塞得下才行），答案就是 $f(1,k,n)$ ，复杂度 $\mathcal{O}(n^3)$ 。

接下来考虑原问题，也就是把这个 dp 过程拓展到树上。此时会发现无法确定路径的两个端点能放多少垃圾，因为端点可以向它的任意邻居扩展。因此我们把``端点接下来要向哪里扩展''加入 dp 状态。dp 状态变成了 $f(u,v,t,mask)$ ， $mask \in [0,3]$ 。

$f(u,v,t,0)$ 表示路径 $u \rightsquigarrow v$ 的最大长度，而且路径不再向两端扩展。

$f(u,v,t,1)$ 表示路径 $u \rightsquigarrow p_v$ 的最大长度（ p_v 是路径 $u \rightsquigarrow v$ 中 v 的前一个节点），而且端点 u 不再扩展，但端点 p_v 接下来会向 v 扩展（也就是说， v 以及 v 的子树里都不能放垃圾）。

$f(u,v,t,2)$ 表示路径 $p_u \rightsquigarrow v$ 的最大长度，而且端点 p_u 接下来会向 u 扩展，但端点 v 不再扩展。

$f(u,v,t,3)$ 表示路径 $p_u \rightsquigarrow v$ 的最大长度，而且端点 p_u 接下来会向 u 扩展，端点 p_v 接下来会向 v 扩展。

转移与之前类似，答案就是 $\max f(u,v,n,0)$ ，复杂度 $\mathcal{O}(n^3)$ 。

H. Crystalfly

当我们初次到达某个节点 u 时，它的所有儿子会被激活，由于 $t_i \le 3$ ，所以我们的决策只有两种，一种是进入某个儿子 v 后继续向下，这样 u 的所有其它儿子的蝴蝶都无法被获取。另一种是进入某个儿子 v 获取 a_v 后立即回到 u ，然后进入另一个儿子 w 并获得 a_w ，这要求 $t_w = 3$ 。我们令 f_u 表示节点 u 在我们进入之前所有蝴蝶都已经消失，同时它的孩子在我们进入 u 之前都未被激活，这棵子树的最优答案，那我们要求的最终答案就是 $f_1 + a_1$ 。我们令 ch_u 表示 u 的儿子集合， sum_u 表示 u 的所有儿子 v 的 f 值之和。考虑两种决策对应的转移，第一种决策：

$$t_u = \max_{v \in ch_u} sum_u + a_v$$

第二种决策：

$$t_u = \max_{v,w \in ch_u, v \ne w, t_w = 3} sum_u - f_v + a_v + sum_v + a_w$$

第二种决策需要在转移时枚举 v ，并且找到符合条件的 w 中 a_w 最大的值，略微维护一下即可。总复杂度 $O(n)$.

I. Cloud Retainer’s Game

假设不存在挡板，那么小球的移动路线中，向右下移动的部分满足 $(x + y) \bmod 2H = k$ ，向右上移动的部分满足 $(2H - y + x) \bmod 2H = k$ 。

记 $f(k)$ 表示特征值为 k 的线路的最优答案。碰到金币 (x, y) 时， $f((x + y) \bmod 2H)$ 和 $f((2H - y + x) \bmod 2H)$ 均增加 1 ；碰到挡板 (x, y) 时，由于可以移除挡板， $f((x + y) \bmod 2H)$ 和 $f((2H - y + x) \bmod 2H)$ 均取二者中的最大值。

这真的是一个期望银铜难度的题目！！！！！！！！！！！！

J. Xingqiu’s Joke

如果只使用前两种操作，那么 $\delta = a - b$ 的值是不变的，而且 a 和 b 的公共质因数一定也是 δ 的质因数。而且当我们想除以一个质因数 g 时，我们一定会先通过前两种操作来到最近的 $g|a$ 且 $g|b$ 的状态然后直接除 g （先加减 k ，再除 g ，再加减 1 肯定优于加减 $2k$ ）。

因此记 $f(a, \delta)$ 表示从 $(a, a+\delta)$ 得到 1 的步数，枚举 δ 的质因数 g ， $f(a, \delta)$ 可以由 $f(\lfloor \frac{a}{g} \rfloor, \frac{\delta}{g})$ 和 $f(\lceil \frac{a}{g} \rceil, \frac{\delta}{g})$ 转移得到。因此状态的第二维只有 δ 的因数个。

但状态的第一维如果和除以因数的顺序有关，那状态数又超了。但无需担心，实际上对于整数 x 和 $a_1, a_2, \cdots, a_k$ ， x 以任意顺序被 a_i 进行整数除法（无论上取整或下取整），结果至多只有两种。因此总的状态数仍然是因数级别的。

通过以下方式完成证明:

若 x 以任意顺序被 a_i 进行下取整的整除得到的结果是 k_1 ，进行上取整的整除得到的结果是 k_2 ，那么

$$k_1 \prod_{i=1}^{n} a_i \le x \le (k_1 + 1) \prod_{i=1}^{n} a_i - 1$$

$$(k_2 - 1) \prod_{i=1}^{n} a_i + 1 \le x \le k_2 \prod_{i=1}^{n} a_i$$

该式可通过数学归纳法证明。

上面两个式子可以看作长度为 $(\prod_{i=1}^{n} a_i - 1)$ 的两个区间，显然只有当 $k_2 + 1 = k_1$ 或 $k_2 = k_1$ 时两个区间才有交集。

由于以任意顺序进行全部上取整的整除得到的结果一定是最大的，进行全部下取整的整除得到的结果一定是最小的，那么上下整除混合得到的结果自然在两者中间。该性质得证。

K. Ancient Magic Circle in Teyvat

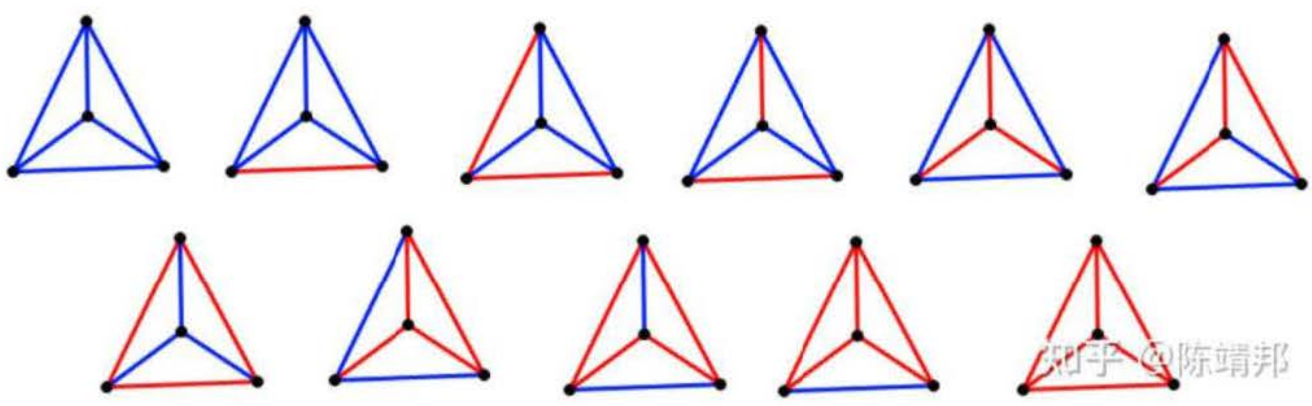
考虑容斥原理，记 U 为所有的四点组，由于四点组里两两连了六条边，记 A_i 表示第 i 条边是红色的四点组，那么有

$$\left|\bigcap_{i=1}^{6}\overline{A_i}\right| = |U| + \sum_{k=1}^{6}(-1)^k \sum_{1\le i_1<i_2<\ldots<i_k\le 6} |A_{i_1}\cap A_{i_2}\cap\cdots\cap A_{i_k}|$$

于是有

$$\left|\bigcap_{i=1}^{6}\overline{A_i}\right| - \left|\bigcap_{i=1}^{6}A_i\right| = |U| + \sum_{k=1}^{5}(-1)^k \sum_{1\le i_1<i_2<\ldots<i_k\le 6} |A_{i_1}\cap A_{i_2}\cap\cdots\cap A_{i_k}| \quad (*)$$

$\left|\left|\bigcap_{i=1}^{6}\overline{A_i}\right| - \left|\bigcap_{i=1}^{6}A_i\right|\right|$ 即为所求，那么需要计算出 $(*)$ 式右边的每一项，



首先列出所有本质不同的 11 种四点组之间连边情况，如果将红色边看成是有限制（对应边一定是红色），蓝色边看成是没有限制（对应边可以是红色也可以是蓝色），除去最后一种全是红色边的情况，其他 10 情况都可以在 $O(m\sqrt{m})$ 的复杂度，也就是枚举三元环和计数四元环的复杂度下计算出来，从而得到 $(*)$ 式右边每一项的值，也就得到答案了。

L. Secret of Tianqiu Valley

可以找规律把方程解出来，得到那些位置需要点奇数次那些位置偶数次。每次如果场上有一个奇数次的位置可以点，贪心点掉，这样是一步解决一个位置。如果场上没有：

场上有至少一个目前灭掉的灯，它一定需要被影响（直接+间接）奇数次；但同时它本身需要被点偶数次，有

```
目前情况：0
方程奇偶：0
```

在它左侧或右侧有一个需要被点的位置，但这个位置当前已经是亮着的了。

假设这个被点的位置在右侧，有

```
目前情况：01
方程奇偶：01
```

这个已经被点亮的位置需要被影响偶数次，因此再右侧一个位置一定是需要被点的；可以推得它一定也已经被点亮了，有

```
目前情况：011
方程奇偶：011
```

现在我们可以依次点第 1， 2， 1， 3 个位置，四步解决两个两个位置，因此最多 $2n$ 。

M. Windblume Festival

每个数对答案贡献只能为 1 或 -1 ，显然至少要有一个 1 ， $n>1$ 时至少要有 -1 。

对于任意一个符合上述条件的贡献方案，选一个 1 作为起点，之后的每个 1 可以和在其之前最近的 -1 合并，之后所有 -1 可以和起点 1 合并，因此所有符合条件的方案均合法。

答案是 $max - min + sum(rest)$ 。