Problem A. Tokitsukaze and a+b=n (easy)

签到题。

枚举 $a = L_1$ 到 R_1，$b = n - a$，再判断 b 是否在区间 $[L_2, R_2]$中，如果是，答案 $+1$。

Problem B. Tokitsukaze and a+b=n (medium)

根据 a 的取值范围 $[L_1, R_1]$，我们求出能满足 $a + b = n$ 的 b 的范围是 $[n - L_1, n - R_1]$，但是合法的 b 的范围是 $[L_2, R_2]$ 。所以答案是区间 $[n - L_1, n - R_1]$ 与 $[L_2, R_2]$ 取交后的区间长度。

区间取交的区间长度可以这么计算：

两个区间 $[a, b]$，$[c, d]$ 取交的区间长度为：$max(0, min(d, b) - max(a, c) + 1)$

Problem C. Tokitsukaze and a+b=n (hard)

同样的，可以根据 a 的范围 $[L_i, R_i]$ 能算出 b 的范围 $[n - R_i, n - L_i]$，再与合法的 b 的范围取交即可。由于这题变成了 n 个区间，那么合法的 b 的范围总共有 $n - 1$ 个（因为有 1 个已经用来当成 a了）。但遍历每一个 b 的合法范围时间复杂度是 $O(n^2)$ 的，需要进行优化。

设 d_i 表示 $b = i$ 时的区间个数。由于区间范围是 10^5 级别，我们可以将 n 个区间利用 差分前缀和 计算出 d 数组。再做一次前缀和计算出 bit 数组。那么 a 的范围是 $[L_i, R_i]$时，可能的 b 的数量可以用通过 bit 计算出：$bit_{n-R_i} - bit_{n-L_i-1}$。但是要注意这样会把 $[L_i, R_i]$ 与 $[n - R_i, n - L_i]$ 相交的长度算进去，所以要减去。

计算时要记得取模。由于存在减法运算，要注意最后答案不要变成负数。

时间复杂度 $O(n + m)$

PS：本题还可以使用 树状数组/线段树 等数据结构进行区间求和。

Problem D. Tokitsukaze and Energy Tree

可以发现当第 i 个能量球放置在节点 x 时，它的贡献为：节点 1 到节点 x 的高度 h_x · 能量球的能量 v_i

于是我们可以贪心。求出高度 h 后，分别对高度 h 和 能量 v 排序，之后大的乘大的，再全部加起来即可。

Problem E. Tokitsukaze and Function

通过暴力打表 或者 对 $f(x)$ 求导后，发现在 $x = \sqrt{n}$ 处 $f(x)$ 最小。但不确定是在 $\sqrt{n}$ 处还是在 $\lceil \sqrt{n} \rceil$ 处。

我们可以分别求出 $f(l)$, $f(r)$, $f(\sqrt{n})$, $f(\lceil \sqrt{n} \rceil)$ 进行排序，求出最小的 $f(x)$。

令 $f(t')$ 为最小的 $f(x)$，现在要求最小的 t。显然在 $[1, t']$ 内的 f 是单调递减的，所以我们可以在 $[L, t']$ 范围内二分求出 t。时间复杂度 $O(T \cdot \log n)$

另外需要注意的是，本题无法使用三分解决，因为三分要求在极小值左边和右边都是**严格单调**的，而由于本题的 f 函数有取整运算，可能使得函数的一些部分不是严格单调的，使用三分有可能找不到极小值点。

Problem F. Tokitsukaze and Gold Coins (easy)

显然，走进怪物格子一定不是最优解，所以我们把有怪物的格子当成障碍物来看。所以题目转化为：求 $(1, 1)$ 到 $(n, 3)$ 的所有路径能覆盖的格子数量。

我们可以通过两次 DFS/BFS 来搜索答案：

1. 从起点 $(1, 1)$ 开始，向下/右走，标记所有能到达的格子。
2. 从终点 $(n, 3)$ 开始，向上/左走，标记所有能到达的格子。

在这之后，只要统计哪些格子被标记过两次即可。

时间复杂度 $O(3 \cdot n)$。

Problem G. Tokitsukaze and Gold Coins (hard)

显然，走进怪物格子一定不是最优解，所以我们把有怪物的格子当成障碍物来看。所以题目转化为：求 $(1,1)$ 到 $(n,3)$ 的所有路径能覆盖的格子数量。

我们可以通过第一列和第三列来确定第二列的可行区域 $[L,R]$。

首先找到第一列向下的第一个障碍物，坐标为 $(r,1)$。接着，我们通过 r 来寻找可行区域的最大值 R。我们在第二列找到第一个大于等于 $r-1$ 的障碍物，坐标为 $(x_r,2)$，然后找到小于 x_r 的第一个空白处，坐标为 $(R,2)$。

同理，我们先找到第三列向上的第一个障碍物，坐标为 $(l,3)$。接着我们通过 l 来寻找可行区域的最小值 L。我们在第二列找到第一个大于等于 $l+1$ 的空白处，坐标为 $(xl,2)$，然后找到小于 x_l 的第一个障碍物，坐标为 $(L-1,2)$。

可以用set分别维护障碍物和空白处，完成上述操作。

现在第一列的可行区域为 $[1, min(r, R+1)-1]$;

第二列的可行区域为 $[L,R]$,

第三列的可行区域为 $[max(l, L-1)+1, n]$。

如果 $L>R$ 或者 $l>R$ 或者 $r<L$，则无法从 $(1,1)$ 走到 $(n,3)$，所以答案是 0 。否则答案为

$$r+(n-l+1)+(R-L+1-count(L,R))$$

$count(L,R)$ 表示第二列 L 行到 R 行的障碍物个数，这个可以使用树状数组来维护第二列的障碍物得到。

总时间复杂度 $O(n\log n)$，常数较大。

Problem H. Tokitsukaze and K-Sequence

可以发现每种数字的贡献可以分开计算。我们按照每种数字出现的次数进行讨论。

假设数字 x 出现的次数为 cnt_x:

- 如果 $cnt_x \le k$，我们可以贪心地将每个 x 都分到某个子序列中，使得每个子序列要么只包含 1 个 x，要么不包含 x。所以此时数字 x 的贡献为 cnt_x
- 如果 $cnt_x > k$，我们按照上面的方法分配完 k 个 x，多出来的 x 必须分配到某个子序列中，导致那个子序列中，数字 x 没有贡献。所以此时数字 x 的贡献为 $k-1$

答案就是每种数字的贡献求和。

现在题目求 $k=1\dots n$ 的答案。我们对 cnt 从小到大排序，枚举 $k=1\dots n$，答案为 $cnt_x \le k$ 的 cnt_x 求和，加上 $cnt_x > k$ 的 cnt_x 的个数乘上 $(k-1)$

时间复杂度 $O(n\log n)$

Problem I. Tokitsukaze and Musynx

先对 x 排序。总共有 5 个区间，枚举每个 x 在哪个区间，二分出每个区间内包含几个 x，再乘上对应区间的 v，求和，即是当前情况的答案。最后对所有情况的答案取 $\max$ 即可。

具体的，我们可以枚举每个 x 在哪个分界点上，来计算答案。实现的时候为了方便，求出当前枚举的分界点与第一个分界点的差值 $delta$，将所有 x 平移 $delta$ 即可。

时间复杂度 $O(n\log n)$

Problem J. Tokitsukaze and Sum of MxAb

看到带有绝对值的式子，一般先展开绝对值。我们对 a_i 和 a_j 的正负进行讨论：

- $a_i < 0, a_j < 0$ 时，$\max(|a_i - a_j|, |a_i + a_j|) = |a_i + a_j| = |a_i| + |a_j|$
- $a_i < 0, a_j > 0$ 时，$\max(|a_i - a_j|, |a_i + a_j|) = |a_i - a_j| = |a_i| + |a_j|$
- $a_i > 0, a_j < 0$ 时，$\max(|a_i - a_j|, |a_i + a_j|) = |a_i - a_j| = |a_i| + |a_j|$
- $a_i > 0, a_j > 0$ 时，$\max(|a_i - a_j|, |a_i + a_j|) = |a_i + a_j| = |a_i| + |a_j|$

所以 $\max(|a_i - a_j|, |a_i + a_j|) = |a_i| + |a_j|$
那么原式 $\sum_{i=1}^{n} \sum_{j=1}^{n} MxAb(i,j)$
$= \sum_{i=1}^{n} \sum_{j=1}^{n} (|a_i| + |a_j|)$
$= \sum_{i=1}^{n} (n \cdot |a_i| + \sum_{j=1}^{n} |a_j|)$
$= \sum_{i=1}^{n} n \cdot |a_i| + n \cdot \sum_{j=1}^{n} |a_j|$
$= n \cdot \sum_{i=1}^{n} |a_i| + n \cdot \sum_{i=1}^{n} |a_i|$
$= 2 \cdot n \cdot \sum_{i=1}^{n} |a_i|$

Problem K. Tokitsukaze and Synthesis and Traits

题意转化为：给一张可能不连通的无向图，q 次查询，每次给出 k 个点，查询 $\frac{k \cdot (k-1)}{2}$ 条边中，有多少条边在原图中出现过。

Solution 1：给边定向

考虑给原图的边定向。设 d_x 表示 节点 x 的度，则对于原图中的一条边 (u, v)，若 $d_u < d_v$, 则连接 $u \rightarrow v$，否则连接 $u \leftarrow v$。给原图的边定向后，每个点的出边 $\leq \sqrt{m}$。对于每次查询，用桶记录每个点，然后直接遍历每个查询的点的所有出边即可，常数较小。总时间复杂度为 $O(\sqrt{m} \cdot \sum k)$。

证明：对于一个查询的点 x，设 cnt_x 表示 x 的出边数量。 若 $cnt_x < \sqrt{m}$，则显然遍历所有出边的时间复杂度小于 $O(\sqrt{m})$；若 $cnt_x \geq \sqrt{m}$，由于 $x \rightarrow y$ $(d_x < d_y)$，则总边数最多为 $cnt_x \cdot d_y = m$，所以 cnt_x 不超过 $\sqrt{m}$。

PS:如果看不太懂，画个图就明白了。

Solution 2：根号分治

对每组查询的点数按照 $w = \sqrt{\sum k}$ 进行分类：

- 若 $k \leq w$，则暴力枚举 $\frac{k \cdot (k-1)}{2}$ 条边，用 hash 判定是否在原图中出现。
  暴力的时间复杂度为 $O(w^2)$，这一情况出现的次数为 $\frac{\sum k}{w}$，因此总时间复杂度不超过 $O(w \cdot \sum k)$
- 若 $k > w$，则暴力枚举所有查询点的出边。
  暴力的时间复杂度最坏为 $O(m)$，这一情况出现的最大次数为 $\frac{\sum k}{w}$，因此总时间复杂度不超过 $O(w \cdot m)$。

由于 $\sum k$ 与 m 的数量级一致，所以两种做法的时间复杂度区别不大，但是根号分治需要多一步 hash，常数较大。

如果你是手写hash，那么是可以过的：（提交记录）

如果你是用gp_hash_table，那么也是可以过的：（提交记录）

但如果你是用unordered_map，那么残念，会TLE（也可能是我写臭了）：（提交记录）

Problem L. Tokitsukaze and Three Integers

Solution 1：容斥

dp_x 表示 $a_i \cdot a_j \equiv x \pmod p$ $(i \neq j)$ 的个数，可以在 $O(n^2)$ 的时间复杂度内计算出来。

接下来枚举 k ，计算 ans_x 满足 $(a_i \cdot a_j) + a_k \equiv x \pmod p$ $(i \neq j)$，时间复杂度为 $O(np)$。

但这样会把 $i = k$ 或者 $j = k$ 的情况计算进答案，我们需要把这部分减掉。枚举 $i = k$ 的情况，减去 ans_x 满足 $(a_i \cdot a_j) + a_i \equiv x \pmod p$ $(i \neq j)$， $j = k$ 的情况同理。时间复杂度为 $O(n^2)$。

总时间复杂度为 $O(n^2 + np)$。

Solution 2：dp

我们让 i, j, k 有序，分两种情况：

1. k 在左边或者右边，即 $[i, j, k]$ 或者 $[k, i, j]$
2. k 在中间，即 $[i, k, j]$

由于 $a_i \cdot a_j = a_j \cdot a_i$，所以不需要管 i 和 j 的顺序，最后答案乘 2 即可。

- 对于情况1，只需要正着做一遍dp，倒着做一遍dp即可：
  $dp_{x,y,0}$ 表示前 x 个数，已经选了 a_i，且 $a_i \equiv y \pmod p$ 的方案数；
  $dp_{x,y,1}$ 表示前 x 个数，已经选了 a_i 和 a_j，且 $a_i \cdot a_j \equiv y \pmod p$ 的方案数；
  初始化 $dp_{x,a_x \bmod p,0} = 1$，表示选 a_x 作为 a_i
  转移：
  - $dp_{x,y,0}$ += $dp_{x-1,y,0}$，表示不选 a_x
  - $dp_{x,y,1}$ += $dp_{x-1,y,1}$，表示不选 a_x
  - $dp_{x,(y \cdot a_x) \bmod p,1}$ += $dp_{x-1,y,0}$，表示选 a_x 作为 a_j
  - $ans_{(y+a_x) \bmod p}$ += $dp_{x-1,y,1}$，表示选 a_x 作为 a_k

  发现 dp 的第一维 x 只跟 $x - 1$ 有关，可以用滚动数组优化掉这一维。
- 对于情况2，我们依然枚举 k，$dp2_y$ 表示合法的 $(a_i \cdot a_j) \bmod p = y$ 的数量。
  先将 $a_1 \cdot a_2$, $a_1 \cdot a_3$,. . ., $a_1 \cdot a_n$ 更新进 $dp2$。
  枚举合法的 k，从 2 到 $n - 1$。假设当前 $k = x$：
  1. 把 a_x 作为 a_j 的不合法贡献从 $dp2$ 中删去
  2. 选 a_x 作为 a_k 计入答案：$ans_{(y+a_x) \bmod p}$ += $dp2_y$
  3. 把 a_x 作为 a_i 的合法贡献加入 $dp2$

总时间复杂度同样是 $O(n^2 + np)$