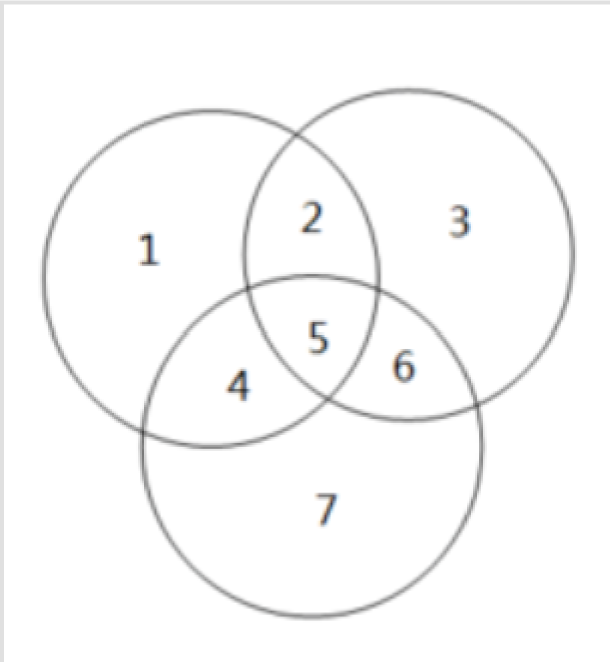
A. Yasaka with Lolita

算法：容斥原理

容斥原理：如果被计数的事物有A、B、C三类，那么，A类和B类和C类元素个数总和= A类元素个数+ B类元素个数+C类元素个数—既是A类又是B类的元素个数—既是A类又是C类的元素个数—既是B类又是C类的元素个数+既是A类又是B类而且是C类的元素个数。（$A\cup B\cup C=A+B+C-A\cap B-B\cap C-C\cap A+A\cap B\cap C$）。



推导公式如下：

$$|A_1\cup A_2\cup\cdots\cup A_m|=$$
$$\sum_{1\le i\le m}|A_i|-\sum_{1\le i<j\le m}\left|A_i\cap A_j\right|+\sum_{1\le i<j<k\le m}\left|A_i\cap A_j\cap A_k\right|-\cdots+(-1)^{m-1}|A_1\cap A_2\cap\cdots\cap A_m|$$

也可表示为，设S为有限集：

$$A_i\subseteq S\ (i=1,2,\cdots,n,n\ge 2)$$

则：

$$\left|\bigcup_{i=1}^{n}A_i\right|=\sum_{k=1}^{n}(-1)^{k-1}\sum_{1\le i_1<i_2<\cdots<i_k\le n}\left|A_{i_1}\cap A_{i_2}\cap\cdots\cap A_{i_k}\right|$$

由于：

$$\overline{A_1\bigcup A_2\bigcup\ldots\bigcup A_n}=\overline{A_1}\bigcap\overline{A_2}\bigcap\ldots\bigcap\overline{A_n}$$
$$\overline{A_1\bigcap A_2\bigcap\ldots\bigcap A_n}=\overline{A_1}\bigcup\overline{A_2}\bigcup\ldots\bigcup\overline{A_n}$$

所以：

$$\left|\overline{A_1}\bigcap\overline{A_2}\bigcap\ldots\bigcap\overline{A_n}\right| = N - \left|A_1\bigcup A_2\bigcup\ldots\bigcup A_n\right|$$

本题：根据容斥基本算法，应为：$a+b+c-ab-ac-bc+abc$。这样的话 $a+b+c$ 代表的是总的要摁的大门的按钮。而 ab, bc, ac代表的是这两个数的倍数，就是两次摁同一个按钮。这里我们应该想一下，如果只是减一个的话，那么还是和原来一样，交接的地方还是有剩余的。但是对一个按钮操作两次，大门就关了。这样就不需要计算这一块了，所以应该减两次。再看 $+abc$。我们将在前面已经 $-2\times(ab+bc+ac)$了，就相当于将 abc 这一块减了三次，但是对于 abc 这一块，三次操作大门是敞开着的。所以应该加上 $4\times(abc)$。所以最后的公式应该是 $a+b+c-2ab-2bc-2ac+4abc$。

B. Yasaka with Minimum Expectation

算法：数学期望+组合数打表

在概率论和统计学中，一个离散性随机变量的数学期望值，是试验中每次可能的结果乘以其结果概率的总和。换句话说，期望值像是随机试验在同样的机会下重复多次，所有那些可能状态平均的结果，便基本上等同“期望值”所期望的数。需要注意的是，期望值并不一定等同于常识中的“期望”——“期望值”也许与每一个结果都不相等。（换句话说，期望值是该变量输出值的平均数。期望值并不一定包含于变量的输出值集合里。）

例如，掷一枚公平的六面骰子，其每次“点数”的期望值是 3.5，计算如下：

$$E(X) = 1\cdot\frac{1}{6}+2\cdot\frac{1}{6}+3\cdot\frac{1}{6}+4\cdot\frac{1}{6}+5\cdot\frac{1}{6}+6\cdot\frac{1}{6} = 3.5.$$

不过如上所说明的，3.5 虽是“点数”的期望值，但却不属于可能结果中的任一个，没有可能掷出此点数。

因为求最小值的期望，所以显然排序后，枚举最小值统计即可;假设从小到大排序，那么答案即为：

$\sum_{i=0}^{n-m} s_i \times C_{n-i-1}^{m-1} \ \%\ (10^9+7)$

组合数可以通过打表实现:

$C_n^m = C_{n-1}^m + C_{n-1}^{m-1}$

C. Yasaka with Cups

算法：简单思维

题面非常简单，就是看给定的数列里面有几个逆序区间，如果只有一个，输出逆序区间的左端点和右端点；如果没有逆序区间输出0 0,；如果有多个逆序区间输出NO。

D~F

详见城哥的 PPT 。

G. Yasaka with Oneman233

你可以发现，这个插入是假的。

只要记下序列的长度 l。

对于 A 操作，$++l$ 单点修改。

对于 $Q\ L$ 操作，查询 $[l-L+1,l]$ 的区间最值。

线段树板子。

H. Yasaka with kr12138

线段树上维护四个东西(和最大子段和的左右端点)

区间和 sum

区间最大前缀和 max_prefix

区间最大后缀和 max_suffix

区间最大子段和 max_seg

区间和=左区间和+右区间和

区间的最大子段和= $max\{左区间最大字段和,右区间最大子段和,左区间最大后缀+右区间最大前缀\}$

区间最大前缀=$max\{左区间最大前缀,左区间和+右区间最大前缀\}$

区间最大后缀同上。

I. Yasaka with yxxxxxxx

如果当花了 g 钱升级时是可行的，那么花大于 g 的钱升级一定可行。

证明： 因为对于花 g 钱后的任何操作，花大于 g 的钱升级后都能做。前者是后者的子集。

也就是说答案具有单调性，我们可以二分答案。

问题变成了判定跳的距离数确定下来后，我们能否获得大于等于 k 的分数。

设 $dp[i]$ 表示跳到第 i 格时所能获得的最大分数。那么 $dp[i]=max\{dp[j]\}+s_i$ ，其中 s_i 是这格的分数，$j\in\{所有能跳到i格的格子\}$。

这样状态数量 $O(n)$ 转移时间 $O(n)$

总时间复杂度成了 $O(n^2logn)$

我们如何做得更快？

考虑转移方程的 $max\{dp[j]\}$部分。

我们发现它其实就是 K 题的滑动窗口。

所以我们可以维护一个单调队列保存最优的状态。

转移时间复杂度均摊 $O(1)$ 。

J. Yasaka and A+B

把素数看做物品的价值，然后完全背包计数。

设 $dp[i][j]$ 表示前 i 个素数选出若干个和为 j 的方案数。

初始状态 $dp[0][0] = 1, dp[i][j] = 0$

转移方程 $dp[i][j] = dp[i-1][j] + dp[i-1][j-p[i]]$，$p[i]$ 为第 i 个素数的值

答案 $dp[n\text{以内素数个数}][n]$

可以滚掉物品的维度。

K. Yasaka and window

滑动窗口，单调队列板子。

如果你补过 Day2 的 H ，这题应该不难。

如果你没补过，那可以参考这题：https://www.luogu.org/problemnew/show/P1886

L. Yasaka with card

任意时刻，如果 1 的位置小于等于 n ，它下次的位置就是在 $2 \times n$。否则它下次的位置在 $(\text{当前位置} - n) \times 2 - 1$。

1 再一次回到位置 1 为止所经过的步数就是答案。

时间复杂度 $O(ans)$