A. rain（easy version）

本题困难版的题解见专业组题解 K 题 rain（hard version）。

先将所有降水区间离散化后按右端点排好序。

对于 $k=1$ 的情况，设 $f_{i,j}$ 为只考虑前 i 个区间，降水量为 1 的最右边的城市编号为 j 的情况下的最大降水量，假如第 i 个区间为 $[l_i,r_i]$，则加入该区间的转移有 $f_{i,r_i}=\max\limits_{t=1}^{l_i-1}\{f_{i-1,t}\}+(R_i-L_i+1)$，其中 L_i,R_i 为离散化前的值。

同理，对于 $k=2$ 的情况，设 $f_{i,j,k}$ 为只考虑前 i 个区间，降水量大于等于 1 的最右边的城市编号为 j，降水量为 2 的最右边的城市编号为 k 的情况下的最大降水量，假如第 i 个区间为 $[l_i,r_i]$，则加入该区间的转移有

$$f_{i,r_i,j}=\begin{cases}\max\limits_{t=1}^{j}\{f_{i-1,j,t}\}+(R_i-L_i+1) & j\geq l_i\\ \max\limits_{t=j}^{l_i-1}\{f_{i-1,t,j}\}+(R_i-L_i+1) & j<l_i\end{cases}$$

B. rain（hard version）

考虑费用流。

某一个点最多被覆盖 k 次的这个条件似乎在把我们引向拆点，但是此时存在一个问题，那就是选择了某一个降雨安排 $[l,r]$ 这个条件在网络流的图中很难表示。

但是，有一个明显的限制，就是就是永远不会选择 k 个以上交于一点的线段，即如果两条线段没有交集，就可以同时选，否则，就会存在 k 的限制。所以我们把每个点看出一条边，即把线段 $[l,r]$ 变换成 $[l,r+1]$。

所以，按照如下方式构图，从超级源点 S' 向源点 S，连接一条 $[k,c]$ 的边。然后从 S 向每个区间左端点连接一条 $[1,0]$ 的边，从一个线段起点向其终点连接一条 $[1,-len]$ 的边，代表一条线段只能选择一次。最后，从每个线段的终点向汇点 T 连接 $[1,0]$ 的边。

等等，我们是不是忽略了 “如果两条线段没有交集，就可以同时选” 这个条件，所以如果两条线段不交，即可从前面一条的终点向后一条的起点连接一条 $[1,0]$ 的边，代表这两条边互相选没有影响。

最后跑最小费用可行流即可。

C. There are many books and books

最后书本的摆放一定是 $[1,1,\ldots,2,2]$ 或者 $[2,2,\ldots,1,1]$ 的形式。

那么我们可以枚举每个位置 i，在将最后书本的摆放定是 $[1,1,\ldots,2,2]$ 的情况下 $[1,i]$ 中所有编号为 2 的书移到最后，将 $[i+1,n]$ 中所有编号为 1 的书移到前面，另一种形式同理。

最后算出移动的最小书本数量即可。

区间内某一类书的数量可以通过前缀和加快计算。

时间复杂度 $O(\sum n)$，空间复杂度 $O(\sum n)$。

D. We need more and more OR numbers

根据按位或运算的性质可以知道，一个小于等于 10^9 的数字求过两次或数之后，它的值就会小于 10。此时，它的或数就是它自己。

所以我们可以使用某种方法，快速记录每个数字被操作了多少次。然后在询问这个数字的时候，我们暴力地去计算操作后地值，因为最多会求 2 次或数。

我是使用树状数组记录每个数字被操作的次数，同时，本题还可以使用线段树，map 容器等。

时间复杂度 $O(\sum q \log \sum n + \sum q)$，空间复杂度 $O(\sum n)$。

E.不减的数组

在这里，我们使用 $-$ 代表负数，$+$ 代表非负数。

数组 a 的形式必然可以表示为 $[-,-,+,-,+,+]$ 或着类似的形式。第一个 $+$ 和最后一个 $-$ 之间的元素必须都变成 0.而前面一堆负数和后面一堆非负数都是从两端向中间贪心，每个数字在满足题目的要求的情况下尽可能少除。

时间复杂度 $O(\sum n \log(\max a))$，空间复杂度 $O(\sum n)$。

F.乘法与加法

对于 $\sum_{i=x}^{y} \sum_{j=x}^{y} a_i \times a_j - \sum_{i=x}^{y} \sum_{j=x}^{y} (a_i + a_j)$ 进行化简：

$$\sum_{i=x}^{y} \sum_{j=x}^{y} a_i \times a_j - \sum_{i=x}^{y} \sum_{j=x}^{y} (a_i + a_j) =$$

$$\sum_{i=x}^{y} a_i \times \sum_{i=x}^{y} a_i - 2 \times (j - i + 1) \sum_{i=x}^{y} a_i$$

令 x 表示 $\sum_{i=x}^{y} a_i$，那么最终结果为 $x^2 - 2(j - i + 1)x$。

这是一个二次函数的形式，所以当 x 最小或最大的时候会去到最大值。

那么如何知道对于某一个询问 $f(x, y, k)$ 来说，x 的最小值与最大值可以取到多少呢。

我们令 $c_i = b_i - a_i$，那么此时是不是使得 x 加上区间中 c 的前 k 大的和，或者加上 c 的前 k 小的和（为负数）即可。

使用主席树即可。

注意答案可能会超过 64 位整数，所以需要 128 位整数来计算答案。

时间复杂度 $O(q \log n)$，空间复杂度 $O(kn)$。

G. 除法与取模

签到题，模拟即可，时间复杂度 $O(\max(\sum a, \sum b))$，空间复杂度 $O(1)$。

提高版：考虑在 $0 \leq x, a, b \leq 10^9; t \leq 10^5$ 的限制下怎么解决这个问题。

H.打饭

这题需要使用 dp 的方法解决，这一点还是比较明显的。

假设我们令 dp_i 表示在消耗 i 点体力的情况下可以获取的饭的质量的最大值，那么我们就可以向背包那么转移，并且在 $O(1)$ 的时间内获取每个询问的答案。但是，这种方案存在一个问题，那就是 $\sum i \times a_i$ 非常大，最大为 10^{11}，那么很明显，我们此时无法这么做。

好的，那我们现在来更改一下 dp 的状态，我们令 dp_i 表示在在打了 i 克饭的情况下消耗体力的最小值，那我么就可以得到如下两个转移：

$$dp_j = \min(dp_j, dp_{j-a_i} + a_i \times i)\ j \neq a[i]$$
$$dp_j = \min(dp_j, (a_i + 2) \times i)\ j = a[i]$$

但是，此时时间复杂度仍然为 $O(n \max(a))$，但是为了回到原点，在位置 i 打的饭的质量不会超过 $\frac{\max(x)}{i}$。此时倒序枚举每个 a_i 转移即可。

在回答每个询问时，在 dp 数组上二分答案就可以了。

I.大力出奇迹

本题中，可以直接使用贪心去做，不过，前提是需要观察到树的高度不会超过 18。

我们可以对于第一棵树，自底向上合并。因为在第一棵树中对于某个节点来说，它的所有子叶子节点两两之间的距离是固定的，所以我们此时只需要知道这些节点在第二棵树上之间的距离最大值。

好的，如果你是这样分析的，恭喜你，此时你已经需要数据结构了，所以，我们现在必须在第二棵树上开始。

对于第二棵树上的每个节点，我们记录在其的左子树和右子树中深度为 dep_i 与 dep_j 的节点与第一棵树的叶子节点的连边中叶子节点编号的最小值与最大值。那么此时枚举左右子树的 dep 值，即可计算出最大环的大小。

时间复杂度 $O(n^2 2^n)$，空间复杂度 $O(n2^n)$。

J.大兴土木

本题的数据范围已经为我们提供了一个思路，那就是区间 dp，然后考虑如何进行 dp 转移。

不难发现，对于任意限制 l, r ，可以知道 $h_i (l < i < r)$ 一定不是该区间内的最大值，那我们就可以将 $[l + 1, r - 1]$ 覆盖住表明这一块的位置都不能作为最大值，按照这个思路，我们可以将最大值作为突破口做 dp 转移。

设 $dp[l][r]$ 为在区间 $[l, r]$ 内只考虑完全被包含在这个区间的限制时的方案数，保证区间内的数时 1 到 $r - l + 1$ 的排列。

根据之前关于最大值的推断，排除在 $[l, r]$ 内所有被限制所确定不能作为最大值的位置，然后枚举可以的位置选定为最大值进行转移，将区间分为左右两个区间。

由于被选定的位置没有被覆盖，也就是说不存在某个限制 l 在左区间且 r 在右区间，所以左右区间是相互独立的，可以直接拼凑起来。

所以 dp 转移式就是 $dp[l][r] = \sum\limits_{k} \binom{r-l}{k-l} dp[l][k-1] \times dp[k+1][r]$ 。

至于实现细节可以简单粗暴的线段树，复杂度为 $O(nm\log n+n^3)$ ，我是对每个位置存储了一个双端队列维护从该位置为开头的区间内可以作为最大值的位置，复杂度 $O(n^3)$ 。

K.美丽角对

使用 Graham扫描法 求出最大凸包后，枚举每个角度，算出其 θ 值，并且记录每个值的数量，最后两层循环枚举每对角，即可得到答案。求角度值可以用叉积来求。

时间复杂度 $O(n\log n+180^2)$，空间复杂度 $O(n+180)$。

L.生活在树上

首先，考虑一个括号序列，什么时候是一个好的括号序列：

1. 括号序列长度必须是偶数。

这是因为假如括号序列长度为 len，而你每次向左侧走，那么最终序列的长度会增加 $2k$。而一个匹配的括号序列长度一定是偶数，所以上述结论成立。

2. 括号序列第一个字符是左括号。
3. 括号序列最后一个字符是右括号。

这两点就不过多阐述。

现在让我们来看看其他条件，因为可以向左侧走，假如有连续的左括号 "((" 或者连续的右括号 "))" 那我们就能产生无穷的偶数个左括号或者无穷个偶数右括号。

所以这就是最后一个条件，连续的左括号对一定出现在连续的右括号对前面（或者同时不存在连续的左括号对与连续的右括号对）。

好的，现在让我们回到问题，如何快速处理树上的询问 u,v 呢？我们此时可以采用树上倍增的方法维护每一个括号段段的长度，括号段连续的左括号第一次出现的位置和括号段连续的右括号最后一次出现的位置，每次询问时在树上倍增查询并且根据上述条件回答即可。

时间复杂度 $O(n\log n+q\log n)$，空间复杂度 $O(n\log n)$。

M.染色游戏（easy version）

最后一步染色肯定不会被其他染色覆盖，所以最后一步染色一定会占满一行或一列。对于每一行与每一列，循环检查所有格子颜色是否相同即可。但是注意，0 不能作为一种颜色。

最后把所有的答案排序输出即可。

时间复杂度 $O(tn^2+tn\log n)$，空间复杂度 $O(n^2)$。

N.染色游戏（hard version）

很明显，最后一次染色必然是染满某一行或某一列，而在这次染色之前，上一次染色的位置一定是被这次染色覆盖掉的某些位置，所以继续在这一次染色的位置的那一行或者那一列来搜索即可。

搜索过程可以使用 bfs 完成，初始时将染满某一行或某一列的颜色入队。

假如最后还有搜索不到但是存在的颜色，那就找不到解。

如果有解，最多染色 $2n$ 次。

时间复杂度 $O(tn^2)$，空间复杂度 $O(n^2)$。

O.至少一半要相等

我们可以知道，答案一定是某两个数的差的绝对值的因子 x，并且只要数组 a 的所有数字模 x 的值有一半就相等即可满足题目条件。

那么我们可以枚举两个数字和它们的差，然后按照上述方法检查，答案一定正确，但是此方法的时间复杂度为 $O(n^3\sqrt{a})$，无法通过题目。

那么我们是否可以随机选取两个数字并计算答案呢？

假如我们随机选取两个数字，那么它们都在最后的“答案那一半”的概率为 $0.5 \times 0.5 = 0.25$，那么在 k 次之后，仍然错误的概率为 0.75^k。k 可以取 100，基本认为不会出错。

PS：$0.75^{100} = 3.21 \times 10^{-13}$，如果你还错了，去买彩票吧。

时间复杂度 $O(k\sum n\sqrt{a})$，空间复杂度 $O(\sum n)$。