A. 树异或价值

显然每位情况独立，所以答案为 $k=1$ 时的答案的 k 次方，故现在只考虑 $k=1$ 的情况。

考虑对 a 价值的式子处理：

$$\sum_{i=1}^{n}\sum_{j=1}^{n}(a_i\oplus a_j)\times dep_{LCA(i,j)}$$
$$=\sum_{i=1}^{n}\sum_{j=1}^{n}(a_i\oplus a_j)\sum_{x=1}^{n}[i\in subtree(x)][j\in subtree(x)]$$
$$=\sum_{x=1}^{n}\sum_{i\in subtree(x)}\sum_{j\in subtree(x)}(a_i\oplus a_j)$$

故实际上价值为树上所有点的子树内两两之间的异或值之和的和。由于 $a_i\in\{0,1\}$，显然我们会考虑让子树内 0 的数量和 1 的数量尽量接近，实际上总有方法，使得每个点子树内的 0 的数量与 1 的数量绝对值差 ≤ 1。

假设 dp_x 表示 $a_x=0$ 时 x 子树内合法的填数方案，$sone_x$ 表示 x 子树大小为偶数的儿子的数量，$sono_x$ 表示 x 子树大小为奇数的儿子的数量。则有转移式子：

$$dp_x=\prod_{y\in son(x)}dp_y\times 2^{sone_x}\times(\binom{sono_x}{\lfloor\frac{sono_x}{2}\rfloor}+\binom{sono_x}{\lfloor\frac{sono_x}{2}\rfloor-1}\times[2\mid sono_x])$$

$k=1$ 时的答案即为 $2\times dp_1$，总答案即为 $(2\times dp_1)^k$。

时间复杂度 $O(n)$。

B. 树上询问

首先通过线段树或 st 表的方式找到区间 $[l,r]$ 内距离最远的两点 x 和 y，再判断 x 和 y 之间路径是否为 $[l,r]$ 中所有点即可。

可以使用哈希判断，也可以使用路径上的 max 与路径上的 min 以及路径上的点数判断。

时间复杂度 $O(n\log^2 n)$ 或者 $O(n\log n)$。

C. 黑洞合并

容易看出对于任意次序合并，答案一致，直接计算即可。

时间复杂度 $O(n)$。

D.亡语

简要题意：你有 n 个随从，每个随从都具有三个亡语效果中的其中一种。若强制第一个随从死亡，求所有随从最后的状态。

- 考虑使用集合 s 维护当前存活的随从，内部按照随从的最大生命值 h 排序。再维护一个队列，记录死了但还没有触发亡语的随从。
- 记录一个值 sum 表示当前所有已触发的第一类亡语造成的伤害之和。显然在触发一次第一类亡语后，新死亡的随从是集合 s 中所有 $h\le sum$ 的。
- 对于第二类随从新产生的随从，将其最大生命值加上此时刻的 sum 再加入集合 s 中即可。

其余部分按题意模拟。

注意第二类随从可能召唤的随从数量会很大，需要开__int128。

E. 怪物猎人

不妨设 $x \le y$，

假设所有的攻击都造成 x 伤害，怪物在第 i 轮死亡，

那么有：$x \cdot (i-1) < k$，$x \cdot i \ge k$

考虑逐次将攻击的伤害由 x 替换成 y, 直到所有攻击的伤害都造成 y 伤害，

分两种情况讨论：

- 若 $y \cdot (i-1) \ge k$，

  那么，一定存在 q （$1 \le q \le i-1$），满足：

  $x \cdot q + y \cdot (i-1-q) < k$，$x \cdot (q-1) + y \cdot (i-q) \ge k$

  也就是说，怪物可能在第 $i-1$ 轮死亡，因此两只宠物都可能给予怪物最后一击。
- 若 $y \cdot (i-1) < k$，

  结合 $x \cdot i \ge k$ 可知，怪物一定在第 i 轮死亡，

  因此，若 i 为奇数，则只有第一只宠物能给予最后一击；

  若 i 为偶数，则只有第二只宠物能给予最后一击。

时间复杂度：$O(1)$

F. 融合矿石

由于价值体系保证价值随着金辉石占比单调不减，

在质量一定的情况下，优先选择金辉石质量较大的矿石。

由于矿石数量无限，可以通过一次 完全背包，

处理出 f_i： 通过融合能够得到的所有质量为 i （$1 \le i \le m$）的矿石中，含有金辉石质量最大为 f_i 。

那么最后选择装入背包的矿石，若其质量为 x ，则其金辉石质量一定为 f_x，

也就是说，只有最多 m 种矿石可能装入背包中。

问题转化为：物品为 m 种矿石， 总重量为 m 的完全背包。

时间复杂度：$O(m(n+m))$

G. 小猫钓鱼

不妨记双方手上各有一张的牌为单牌，两张在一方手上的牌为双牌。

结论：如果所有牌都为单牌，则先手必败，否则先手必胜。

- 都为单牌：显然不论先手打出什么，后手都可以出同样的单牌，直到先手的手牌为空，后手必胜。
- 存在双牌：一个观察是，单牌是没有意义的。因为无论哪一方出单牌，另一方只需要出同样的单牌，

  这样先手失去一张单牌，后手获得一张单牌以及 2 分，且后续的先后手没有发生变化，这对于先手而言显然不优。

因此先手只会打出一张双牌 X，而后手同理不会打出一张单牌，只会打出另一张双牌 Y。

先手再打出另一张 X，先手得到三张牌：两张 X 和一张 Y。（简单分析可知先手打出另一张双牌 Z 也是不优的）

考虑这样一轮的影响：先手得到一张单牌；后手失去一对双牌，得到一张单牌；先后手交换。

因为单牌没有意义，所以简单来说每轮后手会失去一对双牌。

初始时先手和后手手上双牌的数量相等，一定是后手先失去完所有双牌，因此先手必胜。

时间复杂度：$O(n)$

H. 最佳选手

首先考虑判断一位选手（记为 i） 能否成为最佳选手，

所有的对决可以分为两类：

- 选手 i 参与的对决。在对选手 i 最有利的情况下，这些对决的得分都可以确定：所有的未知得分 z 都由选手 i 获得 。

  此时，选手 i 的最终得分也唯一确定，记这个值为 val_i 。

- 选手 i 未参与的对决。记对决的双方为 a 和 b，上半场得分分别为 x 和 y （不妨假设 $x \le y$），未知得分为 z，这些对决根据得分可以分为四类：
  1. 无论如何分配得分 z， 这场比赛双方的得分都大于等于 val_i，也即： $x \ge val_i, y \ge val_i$；
  2. 通过分配得分 z， 可以使得 a 的得分小于 val_i，也即： $x < val_i, y \ge val_i$；
  3. 通过分配得分 z， 可以使得 a/b 的得分小于 val_i，也即： $x < val_i, y < val_i$ 且 $\max\{y, \lfloor \frac{x+y+z+1}{2} \rfloor\} \ge val_i$ ；
  4. 通过分配得分 z， 可以使得 a, b 的得分同时小于 val_i，也即： $\max\{y, \lfloor \frac{x+y+z+1}{2} \rfloor\} < val_i$

将整场比赛变成一张无向图， n 位选手为图上的 n 个点，

对于选手 i 未参与的 4 类对决， 分别处理如下：

1. 不做处理
2. 节点 a 向自身连一条边 （自环）
3. 节点 a 与 节点 b 之间连一条边
4. 节点 a 向自身连一条边， 节点 b 向自身连一条边

对于选手 i 参与的对决，记对决的另一方为 b, 其上半场得分为 y,

若 $y < val_i$， 则 节点 b 向自身连一条边 （自环）， 否则不做处理。

于是得到一张可能有自环/重边的无向图，那么，节点 i 可能成为最佳选手的充要条件是：

这张图上，除了节点 i 外，对于所有的连通块，都满足：边数大于等于节点数。

遍历所有对决，分类连边处理后，通过简单的搜索可以实现 $O(n)$ 判定选手 i 是否可能成为最佳选手，

总时间复杂度： $O(n^2)$ 。

考虑优化，先预处理出每位选手的 val，按照 val 从小到大考虑每位选手是否能成为最佳选手，

记当前考虑的选手为 i，

对于一场对决，随着 val_i 的递增，其类型一定遵循 选手 i 不参与 $\to$ 选手 i 参与 $\to$ 选手 i 不参与 的变化（注意这些 i 不同，且这两次变化显然是连续发生的）；

对于选手 i 参与的对决，其类型一定遵循 $1 \to 2 \to 3 \to 4$ 的变化，

也就是说，一次对决所表示的边最多经历 5 次变化，

那么，比较暴力的实现中，可以利用 ETT，LCT 或 线段树分治 + 可撤销并查集 维护，支持 加/删边，判断每个连通块是否满足边数大于等于节点数。

时间复杂度：$O(n \log^2 n)$

进一步优化，可以通过打标记的方式避免删边操作：

对于节点 a 连一个自环的操作，

将其变为：向节点 a 所在的连通块（记为 f_a）打上标记，也即：$cnt_{f_a} \leftarrow cnt_{f_a} + 1$，

同理，删除这个自环的操作为：$cnt_{f_a} \leftarrow cnt_{f_a} - 1$

那么，对于选手 i 参与的对决的变化 $1 \to 2 \to 3 \to 4$：

- $1 \to 2$：$cnt_{f_a} \leftarrow cnt_{f_a} + 1$
- $2 \to 3$：$cnt_{f_a} \leftarrow cnt_{f_a} - 1$，$a$ 和 b 之间连一条边
- $3 \to 4$：$cnt_{f_a} \leftarrow cnt_{f_a} + 1$，$cnt_{f_b} \leftarrow cnt_{f_b} + 1$（这里删除 a 和 b 之间的边没有意义，因此直接保留即可）

而对于 选手 i 不参与 $\to$ 选手 i 参与 的变化，

首先 $cnt_{f_i} \leftarrow cnt_{f_i} + 1$，

然后根据对决另一方 b 的得分 y, 若 $y < val_i$，则 节点 b 连一个自环，也即 $cnt_{f_b} \leftarrow cnt_{f_b} + 1$，否则不做处理。

同理处理 选手 i 不参与 $\to$ 选手 i 参与 的变化即可。

（实际处理中，由于这两次变化连续发生，这一变化的处理等同于撤销 选手 i 不参与 $\to$ 选手 i 参与 的变化）

具体实现中，利用扫描线的思路，将类型变化看成修改事件（时间点为变化临界的 val），判定作为查询事件（时间点为 val_i），

随着 val 的递增，用并查集维护连通性，每个连通块的边数，节点数，以及标记数。

时间复杂度仅来源于 排序 和 并查集的常数。

时间复杂度：$O(n \log n)$

I. 长期素食

首先考虑朴素的 $\mathcal{O}(n^2)$ dp，用 $f_{i,j}$ 表示当前进行到第 i 天，并且在上一天收获了 j 号田中的黄瓜时候能获得的最大幸福值。可以显然的发现在同一行即相同的 i 的 dp 值当中只有最大值和次大值会被用到，于是考虑只维护这两个值。 同时可以证明每个时刻可能会被收获的只有提供幸福值前 3 大的黄瓜田。

考虑如何找出：维护三层凸壳，第一层凸壳维护每个时刻供幸福值最大的瓜田（即最上面的直线）。第二层维护除去出现在第一层中出现的直线以外的直线构成的凸壳。第三层同理。容易发现对于每一层的所有凸壳，在某一个时刻上的取值是单峰的。于是就可以用于维护提供幸福值前 3 大的黄瓜田，只需在每层的最大值左右的地方寻找次大值和第三大值即可。

时间复杂度：$O(n \log n)$ 或 $O(n)$。

J. 收集签名

考虑 DP，假设 sz_x 表示 x 的子树大小，lef_x 表示 x 子树叶子节点数量，$dp_{x,s}(-lef_x \le s \le sz_x)$ 表示：

$$dp_{x,s} = \begin{cases} x \text{ 子树内存在 } -s \text{ 个还未处理的超级技能终点，遍历的最少时间的 } 1/2 \ (s \le 0) \\ x \text{ 子树内存在 } s \text{ 个还未处理的超级技能起点，遍历的最少时间 } 1/2 \ (s > 0) \end{cases}$$

则有 DP 式子：

$$dp'_{x,s} = \min(dp_{y,i} + dp_{x,s-i} + [i > -lef_y] \times w_{x,y} + |i| \times k)$$

由于遍历的第二维下标范围在 $O(sz_x)$ 内，所以总时间复杂度为 $O(n^2)$。

K. 地牢谜题：更高还是更低

H 和 L 两种情况没有区别，接下来以 H 为例。

首先考虑如何的 $[l, r]$ 区间能被一次杀完。对于一个固定 r，合法的 l 一定是一段后缀。因此，我们可以预处理每个 r 对应的最小的合法的 l，假设为 L_r。

所以询问 $[l, r]$ 就相当于令一个 $x = r$，不断的将 x 变为 L_x，直到 $x \le l$ 需要多少次操作。这个问题可以通过倍增+二分处理。

时间复杂度 $O((n + m) \log n)$。

L. 地牢谜题：三个怪人

很抱歉本题题意对大家造成的误解。其实本题直接给出形式化题意会更好，下次本出题组一定重视注意相关问题。

形式化题意：

给定 n, m，询问有多少个不同的长度为 n 的非负整数数组 $a_1, a_2, \ldots, a_n$，以及长度为 $m + 1$ 的非负整数数组 $b_0, b_1, b_2, \ldots, b_m$ 满足：

- $$\sum_{i=1}^{n} a_i + \sum_{j=0}^{m} b_j = m$$
- $$b_0 = 0$$
- 存在且唯一存在 $x \ (1 \le x \le n)$，使得存在 $y \ (0 \le y \le m)$ 满足：
  $a_x + b_y = y$，且不存在 $x' \ (1 \le x' \le n, x' \ne x)$ 满足 $a'_x + b_y = y$。

考虑按 $y-b_y$ 从大到小考虑。假设 $dp_{i,j,t,s,0/1}$ 表示考虑到 $y-b_y=i$，有 j 个这样的 y，已经考虑过 t 个 a_x，当前的 $\sum a+\sum b=s$ 的当前有没有合法的 a_x 的方案数。

转移分两步考虑，第一步考虑将部分 $y-b_y$ 的 b_y++，第二步则考虑钦定部分 a_x 为当前的 $y-b_y$。考虑分析时间复杂度：

- i 的范围为 1 到 m。
- j 要满足 $j\times(j-1)/2\le m$。
- t 要满足 $j\times(j-1)/2+t\times i\le m$。
- s 要满足 $s\ge j\times(j-1)/2+t\times i$

所以总状态数应该为 $O(m^{5/2}\ln m)$。考虑转移的时间复杂度：

对于第一部分转移，会多枚举一维 j，时间复杂度为 $O(m^3\ln m)$。

对于第二部分转移，会多枚举一维 t，时间复杂度 $O(m^{7/2})$。

由于常数较小，可以通过 $m=200$ 的数据。时间复杂度 $O(m^{7/2}+m^3\ln m)$。