以下是本场的难度分布：

- Easy：A, F, H
- Medium：C、E、G、I
- Hard：B、D、J

Easy

难度低于正式赛题目，用于检测大家的代码语言基本功。

A. 战斗爽（模拟）

有一些细节但不是很复杂的模拟题。

需要注意对于每个回合，都是小 hua 先进攻，再是攻击力最高的存活敌人进攻，小 hua 的进攻可能导致当时攻击力最高的敌人死亡。还需要注意 $k = 1$ 之类的 corner cases。

能够一遍 AC 此题，说明较为细心，且拥有不错的代码实现能力。

F. 进步（线段树 or 树状数组）

一个考察数据结构基础是否扎实的，接近模板题。

本质是：单点修改，前缀查询。

线段树能够胜任，但是最方便的是使用树状数组（常数小且核心代码不到十行哦，具体可以看 std ^v^）。

H. 制衡（简单 DP）

基础的 DP 题，注意 $n \times k \leq 10^6$。

不妨考虑一种直接的状态设计。

$f(i, j)$ 表示当前考虑了序列位置 $[1, i]$，且已经分了 j 段。每次转移：

- 可以划分出一个空段，对应 $f(i, j + 1) \leftarrow f(i, j)$。
- 可以将下一位融入到当前段，对应 $f(i + 1, j) \leftarrow f(i, j) + a_{i+1, j}$。

注意使用 vector 的 resize 功能动态开数组可以避免空间问题。

Medium

难度接近正式比赛的签到题或铜牌题，是选手们需要较快反应并做出的。

C. 洞察（二进制、二分）

给定**正整数** k, b, c, v，请问有多少**非负整数** x 满足 $(kx + b) \oplus c \leq v$。

第一个问题，答案是否可能超过 long long 范围？

注意 $2^{60} > 10^{18}$，所以如果 $x > 2^{61}/k$，那么 $(kx+b) \oplus c$ 的最高非 0 二进制位一定比 v 大，也就一定会 $> v$，所以 x 有一个显然的上界。

考虑拆位，从高到低考虑二进制位，设当前考虑到第 t 位，当前需要考虑的 x 在范围 $[l, r]$ 中。（初始 $l = 0, r = \lfloor 2^{61}/k \rfloor$）

下面约定以 $[x]_t$ 表示 x 二进制表示下的第 t 位。

设使得 $[(kx+b) \oplus c]_t = 0$ 的 x 的个数为 p，$[(kx+b) \oplus c]_t = 1$ 的个数自然为 $(r-l+1) - p$。

核心性质：当前 $[l, r]$ 中，使得 $[(kx+b) \oplus c]_t = 0$ 的 x 是 $[l, r]$ 的长度为 p 的前缀或后缀。更具体的，若 $[c]_t = 0$ 则是前缀，否则为后缀。因为 x 在当前范围内，第 $t+1$ 到最高位都一样（这个性质可以根据后续做法归纳证明），从而第 t 位的差距可以明显分出大小关系（类似考虑 $[0, 7]$，其中 $[0, 3]$ 的第二位二进制位为 0，而 $[4, 7]$ 都为 1）。

- 若 $[v]_t = 0$，直接不用考虑 $[(kx+b) \oplus c]_t = 1$ 的区间，$[(kx+b) \oplus c]_t = 0$ 的区间则需要继续比较。缩小 $[l, r]$ 到那个长度为 p 的区间，并继续考虑 $t-1$ 位。
- 若 $[v]_t = 1$，$[(kx+b) \oplus c]_t = 0$ 的区间最终一定 $< v$，故答案直接加上 p。对于 $[(kx+b) \oplus c]_t = 1$ 的区间则需要继续考虑，缩小 $[l, r]$ 到那个长度为 $(r-l+1) - p$ 的区间并继续考虑 $t-1$ 位。

如何确定 p，因为 $k > 0$，$kx + b$ 有单调性，$kx + b$ 第 t 位是 0 还是 1 也有单调性，可以二分答案。实际上如果熟悉二进制，也有 $O(1)$ 确定的方法。

最终复杂度 $O(T \log^2 V)$ 或 $O(T \log V)$。V 即值域 10^{18}。

E. 持家（贪心）

较为简单的贪心，只需要观察到几个性质：

- 先打折再减价，一定优于减价后再打折。打折内部顺序不重要，减价内部顺序也不重要。
- 只会用到最小的几个折扣，以及最高的几个减价。

问题是怎么确定用几个折扣、几个减价？

直接枚举即可，枚举使用 a 个折扣，$k-a$ 个减价。预先排好序并预处理 前缀和/前缀积，可以在 $O(n+k)$ 的时间内得到答案。

G. 童年（博弈论、有向图）

最最经典的博弈论建模，也非常具有普适性。

对 (x, y, x', y') 的所有情况建图，如果一个状态 p 能够变化到状态 q，则连有向边 $p \to q$，得到一个有向图。

双方实际就是在图上轮流决定状态走到下一个状态，谁率先走到 $(x, y, 0, 0)$（注意走这一步的人在走完这一步之后变为后手，所以他对应后面的 $(0, 0)$）谁就获胜。

考虑经典的博弈思路：

- 一个点能到达必败态，那么它是必胜态

- 一个点只能到达必胜态，那么它是必败态
- 否则它为平局态

考虑一个逆向迭代的过程，每次将非平局态的点压入队列，如果从队列取出的点为必败态，那么将所有**能到它**的点标记为必胜态。如果取出的为必胜态，那么查看每个能到它的点是否满足必败态的条件。

具体实现只要预处理同时存下反向边即可。复杂度 $O(n+m)$，这里是 10^4。

I. 荣耀的羁绊（搜索、状压）

这题的最终难度比预设难度简单，因为放宽了时限。

预处理每种英雄选择方案（共 $\binom{n}{5}$ 种）的答案数（至多 $5!$ 种轮换），查询时直接枚举，复杂度 $O(n^5q)$。当然这没有算预处理复杂度。

预处理在 $O(n^5)$ 枚举后直接暴力搜索在放宽时限后也能通过，std 采用的是小型的状态压缩的 DP（经测试效果确实更好），单次复杂度为严格 2^5。（具体见 std，大概是将 2^5 种二进制按照 1 的个数分层枚举，减少了很多无效枚举）

Hard

难度接近正式比赛的中档题，银牌题左右。

B. 溯源（树形 DP）

大概是这套题中难度最高的。

考虑修改一下问题，将权值改为 1，也就是单纯计数。

可以预处理每个点的深度 d_u，以及能连通的最浅深度 m_u，DP 时设计状态 $f(u,k)$，表示考虑了以 u 为根的子树，k 表示当前 u 所在连通块中的点最浅深度，对 u 的每个儿子 v，转移分两种：

- 对 $k\le d_u$，$f(u,\max(m_u,k))\leftarrow f(v,k)$。将转移分为 $k\ge m_u$ 和 $k<m_u$ 并前缀和优化即可。
- 不限制 k，当前 u 可以断开和儿子 v 的边，此时 $f(u,m_u)\leftarrow f(v,k)$。

总复杂度 $O(n^2)$。

而权值为所有连通块大小乘积其实有非常经典的技巧。

那就是考虑组合意义，相当于在每个连通块选择一个关键点。具体的，考虑有两个集合 A,B，一个大小分别为 $|A|,|B|$ 且 $A\cap B=\varnothing$，那么从 A,B 各选一个元素的方案数即为 $|A|\times|B|$。（某种意义上来说，是最简单的乘法原理）

那么 DP 数组多开一维 0/1 表示当前连通块有没有选关键点即可。（如果做过同类 DP 大概能迅速反应过来）

具体的，最终状态设计为 $f(u,k,0/1)$，每次转移额外判断当前连通块有无关键点（注意每个连通块必须**恰好**选择一个关键点）。

这样最终复杂度仍为 $O(n^2)$。值得注意的是，空间复杂度也为 $O(n^2)$，虽然足够通过本题（在刻意开大空间减小难度的情况下）但是略显吃紧，可以通过外层 dfs 套内层 dfs 的技巧节省空间，做到 $O(n)$ 空间 $O(n^2)$ 时间的优秀复杂度。

D. 充实（ad-hoc、数论）

非常典型的思维题，代码难度很低。

这种计数题都先考虑合法的充要条件，只有足够简洁的性质是能够去计数的。

对 $a_1 < a_2 < \cdots < a_n$，考虑序列 $\{a_i\}$ 充实的充要条件。

考虑差分，不难发现 $a_i, a_j (i < j)$ 奇偶性相同，等价于 $a_j - a_i$ 是偶数，而插入 $\frac{a_i+a_j}{2}$，无非是将它们的差 $/2$。举个例子：

- `1 5`
- `1 3 5`
- `1 2 3 4 5`

对应的差分数组：

- `4`
- `2 2`
- `1 1 1 1`

注意加入数越多越不劣，故每当差分数组中有偶数，将它不断 $/2$ 是不劣的。故只需要考虑差分全为奇数的情况。

最终目标就是将差分数组变为全 1。

那么如果相邻的 $a_{i+1} - a_i$ 都是 2 的次幂，那么答案显然为 Yes。当然即使并非 2 的次幂也可能是 Yes，例如：

- `1 4 6`
- `1 4 5 6`
- `1 3 4 5 6`（选择 1 5）
- `1 2 3 4 5 6`

因为我们每次显然不一定要选择相邻位置的两个数。仍然观察差分数组：

- `3 2`
- `3 1 1`
- `2 1 1 1`
- `1 1 1 1 1`

注意 `3 1 1` 到 `2 1 1 1` 这一步是非常规的，而其他步骤是常规的 $/2$。

非常规的原因是，它选择了原序列中并不相邻的 `1 5` 操作，对应差分数组中的 `3 1`，不难发现，这样操作的本质是**让较大的数减去了较小的数**。注意两个奇数相减必得偶数，故一定可以继续 $/2$。

学习计算 $\gcd$ 辗转相除法之后，对 $(a, b) \to (a - b, b)$ 就要有相当的敏感性，不难发现我们最终一定能得到 $\gcd(a, b)$ 的差分，而不难证明这也是能得到的做好的了（熟悉基础数论知识的话会好理解很多）。

故令 $d_i = a_{i+1} - a_i, \forall i = 1, 2, \cdots, n-1$，那么序列 $\{a_i\}$ 合法的充要条件是 $\gcd(d_1, \cdots, d_{n-1})$ 是 2 的次幂！

这个性质的发现令人兴奋，因为这意味着从根出发做一次 dfs 顺便记录 gcd 即可。

实际上，这题可以进行任意树上路径查询，也不需要是小根堆。利用 $\gcd$ 的性质不难得到这一点，有兴趣的同学可以自行思考这题的加难版（(●'◡'●)）。

J. 图之图（计数、图论、根号分治）

将二元组 (a_i, a_j) 连无向边，将点 $1, 2, \cdots c$ 按照度数（连边数）分类，分为 $\geq k$ 的和 $< k$ 的。

从 $1 \to n$ DP，设当前到 i：

- 如果 a_i 的度数 $< k$ 就直接暴力枚举对应出边 (a_i, a_j)，并将方案加到颜色上。
- 如果 a_i 的度数 $\geq k$，就依靠预处理。反向存边，并在每次查询的时候顺便特别考虑 $\geq k$ 的那些颜色并特殊计数，遇到时直接使用特别计数的结果。

复杂度：$O(n(k + m/k))$，取 $k = \sqrt{m}$ 时得到最佳复杂度 $O(n\sqrt{m})$。

如果接触过根号分治，这是非常自然的技巧。结合 std 食用更佳喵(❁´◡`❁)