排队

算法一

直接搜索，时间复杂度 $O(\text{能过})$ ，期望得分 30 。

算法二

当 h_i 互不相同时，每个人的目标位置是确定的，即他身高的排名，设为 p_i ，每个人消耗的体力实际上就是他的移动距离，所以第 i 个人消耗的体力不会少于 $|p_i-i|$ ，现在我们证明存在一种方案可以使所有人都达到这个下界。

对于一种局面，我们把人按照 p_i 和 i 的大小关系分成三类，先删去所有 $p_i=i$ 的人。之后如果存在两个相邻的人 i 和 $i+1$ ，使得 $p_i>i$ ， $p_{i+1}<i+1$ ，那么交换他们两个一定不劣，否则唯一的可能是存在一条分界线 x $(x=0,1,\dots,n)$ ，对于 $\forall i\le x$ ， $p_i<i$ ，对于 $\forall i>x$ ， $p_i>i$ ，容易发现这种情况并不存在。于是我们总可以这样交换，直到所有人的 p_i 都 $=i$ ，这种方案使每个人消耗的体力都达到了理论上的最小值，所以自然满足题目要求，答案为 $\sum|p_i-i|$ 。

直接排序计算即可，时间复杂度 $O(n\log n)$ ，期望得分 30 。

算法三

现在我们知道，一旦每个人的目标位置固定了，答案就随之确定了，于是问题变成了，如何合理的分配 p_i ，使得 $\max|p_i-i|$ 最小的前提下，$\sum|p_i-i|$ 最小。

对于两个身高相同的人初始时排在位置 i 和 j ，如果 $i<j$ 并且 $p_i>p_j$ ，经过简单的分类讨论可以证明，交换 p_i 和 p_j 一定不劣。

所以只要把以 h_i 为第一关键字、初始位置为第二关键字排序之后的排名作为 p_i 就可以得到最优方案。

时间复杂度 $O(n\log n)$ ，期望得分 100 。至于为什么只出到 50 ，主要是想放过一些乱搞，毕竟 NOIP 的 T1 送分还是要送到位的，希望没人被这个数据范围吓到。

STL 中的 sort 函数当待排序元素个数小于等于 16 时采用的是稳定的插入排序，这就是为什么算法二不拼暴力实际得分也是 60 。

扔骰子

令 $A = \max a_i$ 。

算法一

直接枚举每个骰子的点数并计算最大值，时间复杂度 $O(A^n)$ ，期望得分 10 。

算法二

设 $F_{i,j}$ 表示考虑前 i 个骰子，当前点数最大值为 j 的方案数，转移时直接枚举新的点数即可，时间复杂度 $O(nA^2)$ ，期望得分 30 。

算法三

先将 a_i 从小到大排序，设 $a_0 = 0$ ，当 Max 属于 $[a_{x-1}+1, a_x]$ 时，最大值等于 Max 的方案数为：

$$(\prod_{i=1}^{x-1} a_i) \times [Max^{n-x+1} - (Max-1)^{n-x+1}]$$

这个式子的含义是，前 x 个骰子的点数不可能达到或者超过 Max ，所以可以随便选，对于后 $n-x+1$ 个骰子，用所有点数都 $\le Max$ 的方案数减去所有点数都 $\le Max-1$ 的方案数就是最大值恰好为 Max 的方案数。

先枚举 x 再枚举 Max ，a 的前缀积可以便枚举边算，暴力计算 Max 和最大值为 Max 的方案数的乘积的和，时间复杂度 $O(n + A\log n)$ ，期望得分 60 。

算法四

当 Max 属于 $[a_{x-1}+1, a_x]$ 时，所有方案的最大值之和为：

$$(\prod_{i=1}^{x-1} a_i) \times \sum_{Max=a_{x-1}+1}^{a_x} Max \times [Max^{n-x+1} - (Max-1)^{n-x+1}]$$

注意到相邻两项之间有很大一部分是可以消去的，剩下的部分除了头尾是若干连续正整数的 $n-x+1$ 次方之和，这是经典问题，可以参考 $\mathrm{CF622F}$ ，有拉格朗日插值和第二类斯特林数两种做法，通过适当的预处理可以去掉老哥，时间复杂度 $O(n^2)$ ，期望得分 100 。

拉格朗日插值法虽然不在大纲里，但是在考试中时常出现，本身也并不复杂，推荐还不会的同学学一学。

约树

令 w_i 为 i 号点的点权。

算法一

搜索 / 打表，期望得分 $0 \sim 16$。

算法二

n 为 2 的幂是有提示性的，但实际做法并不依赖这一点。考虑一种 naive 的构造：$w_1 = 1 \times 2$，$w_2 = 2 \times 3$，$\cdots$，树的形态是一条链，用 w_i 和 w_{i+1} 构造 $i+1$，但是这样并不能构造出奇数。沿用这个思路，令：$w_1 = 1 \times 3$，$w_2 = 3 \times 5$，$\cdots$，就可以构造出所有 > 1 的奇数，然后给所有偶数都除以 2 并重复这一过程，之后对应的点权也要乘 2。为了解决所有 2 的幂，我们可以将 1 号节点的点权乘上 2 的若干次方，从第二轮开始每轮产生的链都接在 1 号点下方，具体实现有点麻烦就不细说了，需要特判较小的数据，这样构造出来的点权在 n^2 级别，期望得分 28。

算法三

令 $w_i = (\lfloor \frac{i}{2} \rfloor + 1) \times (n - \lfloor \frac{i-1}{2} \rfloor)$，树的形态是一条链，这样只靠相邻两个节点就可以产生 $2 \sim n$ 中的所有数，设 $t = \lfloor \frac{n}{2} \rfloor + 1$，则 $w_n = t^2$，$w_{n-2} = (t-1) \times (t+1)$，所以 $n-2$ 到 n 这三个点就可以产生 1，这样构造出的最大的点权就是 w_n，比 $\frac{n^2}{4}$ 稍大，期望得分 40。

这一做法来自秦兆阳学长。

算法四

树上的路径数量是平方级别的，而本题只需要构造 n 个数，这意味着树上有 n 个点这件事或许并不太重要，假如我们可以构造出点数 $< n$ 的方案，只需要随便连一些废点就可以了。

感觉上较小的数是比较容易产生的，我们尝试构造较大的数，并通过一些连边和完善产生较小的数，考虑设定阈值 B。

对于 $> B$ 的数，通过从后往前搜索每次找到剩下的数中满足 $\operatorname{lcm}(S) \le 11000$ 的元素个数最多的集合 S 并把这些数删去。设这些集合是 $S_1 \sim S_k$。

先对每个集合 S_i，建一个权值为 $\operatorname{lcm}(S_i)$ 的点 u，称作该集合的特殊点，再建 $|S_i|$ 个点，每个点点权是 S_i 中的一个元素，分别向 u 连边，这样可以得到 k 个菊花，对于 $> B$ 的所有数，只需要选择它所属集合中点权为它自身的点和特殊点的这条链就可以产生它。

想办法合并这 k 个菊花，让 $\le B$ 的数也符合条件。

通过直观感受或实际测试发现，连接非特殊点的优势不太大，记：

$$V_{i,j} = \{\gcd(a,b) \mid a \in S_i \cup \{\operatorname{lcm}(S_i)\}, b \in S_j \cup \{\operatorname{lcm}(S_j)\}\}$$

如果连接了集合 S_i 与集合 S_j 的特殊点，可以使 $V_{i,j}$ 中的数符合条件。不妨每次贪心地选出 $|V_{i,j}|$ 最大，且 i 和 j 还不在同一连通块中的边连接，过程类似 $kruskal$。最后还会剩下一些 $\le B$ 的 x 不能被表示，直接建出点权为 x 的点，当 $B \le \lfloor \frac{n}{2} \rfloor$ 时，一定存在 ≥ 2 的正整数 a 使得 $B < ax \le n$，让 x 向 ax 连边即可。

取 $B = \lfloor \frac{n}{2} \rfloor$，则集合个数 k 不超过 700，只要 B 不太小搜索的部分是很快的，后半部分时间复杂度为 $O(k^2 \log k)$，实际测试可以通过 $n \le 800$ 的全部数据，期望得分 64。

算法五

优化上面的做法，可以发现不同的 $V_{i,j}$ 会有重复元素，考虑每次选出能使新增的符合条件的数最多的边。可以用 set 维护每条边当前加入能新增的数的集合，时间复杂度 $O(k^3 \log k)$ ，由于 $V_{i,j}$ 中的元素其实很少，可以用堆维护所有边，每新增一个数在所有包含这个数的 V 中删去这个数并重新加入堆中，每个元素最多被删一次，时间复杂度 $O(k \log k + \sum |V_{i,j}| \log k)$ ，实际测试可以通过 $n \leq 2500$ 的全部数据，根据实现不同，期望得分 $84 \sim 100$ ，std 可以在 300ms 内得出答案。

本题由出题人的另一道题加强而来，为了沿用之前那道题的做法，最初构造的点权是可以有零的，过了很久我们才发现存在菊花图这种弱智的做法，为了进一步加强，正解就变的比较复杂，作为一道偏乱搞的题，相信大家有更优秀的做法。

遥不可及

记 $\oplus$ 表示按位异或运算。

算法一

暴力搜索出每个区间选哪个数。

枚举每位判断异或和是否回文。

时间复杂度 $O(2^{mn}m)$ 。

算法二

设 $f_{i,j}$ 表示已经确定 $a_1,\cdots,a_i$ 的值，$\bigoplus_{k=1}^{i}a_k$ 的值为 j 的方案数。

时间复杂度 $O(4^mn)$。

可以用 FWT 优化转移，时间复杂度为 $O(2^mmn)$ 。

算法三

对于非负整数 x，记 x_i 表示 x 在二进制下第 i 位（从第 0 位开始）。

$\bigoplus_{i=1}^{n}a_i$ 的前 m 位回文，即对于所有 $0\le k<\lfloor\frac{m}{2}\rfloor$ 的 k，有 $\bigoplus_{i=1}^{n}(a_i)_k=\bigoplus_{i=1}^{n}(a_i)_{m-k-1}$

亦即 $\bigoplus_{i=1}^{n}\left((a_i)_k\oplus(a_i)_{m-k-1}\right)=0$。

设函数 $h:\mathbb{N}\to\mathbb{N}$ 满足 $h(x)_k=\begin{cases}x_k\oplus x_{m-k-1} & 0\le k<\lfloor\frac{m}{2}\rfloor\\0 & k\ge\lfloor\frac{m}{2}\rfloor\end{cases}$

那么 $\bigoplus_{i=1}^{n}a_i$ 的前 m 位回文，等价于 $\bigoplus_{i=1}^{n}h(a_i)=0$。

设 $f_{i,j}$ 表示已经确定 $a_1,\cdots,a_n$ 的值，$\bigoplus_{i=1}^{k}h(a_k)$为 j 的方案数，这样状态数为 $O(2^{m/2}m)$ 。

预处理出 h 在 $[0,2^m)$ 的取值，对每个区间暴力枚举选哪个数转移，时间复杂度为 $O(2^{3m/2}m)$ 。

算法四

注意到 h 值域为 $[0,2^{\lfloor m/2\rfloor})$ 。

如果可以快速求出对于一个区间 $[l,r]$，在经过 h 的变换后，$[0,2^{\lfloor m/2\rfloor})$ 中每个值出现了多少次，就可以优化转移。

考虑形如 $[t2^{\lfloor m/2\rfloor},(t+1)2^{\lfloor m/2\rfloor})$ 这样的区间。对于该区间所有数 x，记函数 $hrev(x)$ 满足 $hrev(x)_k=\begin{cases}x_{m-k-1} & 0\le k<\lfloor\frac{m}{2}\rfloor\\0 & k\ge\lfloor\frac{m}{2}\rfloor\end{cases}$，那么 $h(x)$ 即为 x 的前 $\lfloor\frac{m}{2}\rfloor$ 位与 $hrev(x)$ 的异或和。

注意到在计算 $h(x)$ 时，$hrev(x)$ 是固定的，而 x_k 对应的前 $\lfloor \frac{m}{2} \rfloor$ 位的组合正好遍历 $[0, 2^{\lfloor m/2 \rfloor})$ 这个区间的所有数。 于是 $h\Big([t2^{\lfloor m/2 \rfloor}, (t+1)2^{\lfloor m/2 \rfloor})\Big) = \{a \oplus hrev(t2^{\lfloor m/2 \rfloor}) \mid a \in [0, 2^{\lfloor m/2 \rfloor})\} = [0, 2^{\lfloor m/2 \rfloor}]$，是一个连续的区间。

因此可以将所有区间 $[l, r]$ 拆成若干形如 $[t2^{\lfloor m/2 \rfloor}, (t+1)2^{\lfloor m/2 \rfloor})$ 的整区间和两个零散的区间。分成三部分转移，或者维护一个权值数组进行转移。

时间复杂度 $O(2^m n)$ 。

可以用 FWT 优化转移，时间复杂度为 $O(2^{m/2} mn)$ 。

算法五

算法四中，将所有区间 $[l, r]$ 拆成了若干形如 $[t2^{\lfloor m/2 \rfloor}, (t+1)2^{\lfloor m/2 \rfloor})$ 的整区间和两个零散的区间。

整区间是容易处理的，考虑如何处理两端的零散部分。

对于两端的部分，$hrev(x)$ 依然是固定的，在经过 h 的变换后，是类似 $\{a \oplus hrev(s) \mid a \in [0, r]\}$ 的形式，其中 $r < 2^{\lfloor m/2 \rfloor} - 1$，对于左边区间有 $s = l$，对于右边区间有 $s = r$。

注意到 $\{a \oplus hrev(s) \mid a \in [t2^d, (t+1)2^d)\}$ 仍然是一个连续的区间。于是连段零散的部分可以被拆成 $O(\log s) = O(m)$ 个区间。

而且这些区间恰好也是 $[t2^d, (t+1)2^d)$ 的形式。

注意到对于两个这样的区间 S_1, S_2，$\{x \oplus y \mid x \in S_1, y \in S_2\}$ 合并之后还是一个连续区间，并且每个数出现次数相同。

一个暴力的想法是，枚举每个数从哪个区间中选，合并这些区间，若最终区间包含 0，则计入答案。

时间复杂度 $O(m^n)$ 。

算法六

考虑维护出算法五在暴力合并过程中得到的区间。

对于每个 i，枚举区间 $[l_i, r_i]$ 所拆出的 $O(m)$ 个区间，与前 $i-1$ 个区间合并出的所有区间做合并。

时间复杂度 $O(m^n)$ 。

考虑一个维护可重区间集的抽象数据结构，支持与某个区间集做上述的合并操作。

上面的过程即在一个抽象数据结构中不断添加区间。

考虑维护两个上述数据结构，每次将信息数量更少的数据结构与当前区间拆出的区间集做合并。

因为查询仅查询有多少异或和为 0 的情况，所以可以枚举两个数据结构中对应的区间，计算答案。

时间复杂度 $O(m^{n/2})$ 。

算法七

考虑优化算法六中的抽象数据结构。

维护一棵线段树，对于每个节点，设其对应的区间为 $[l, r]$，在该节点上维护前面有多少种方案使得最后异或和在 $[l, r]$ 之间。

对于每次加入区间操作，假设当前要在节点 z 上加入区间 $[ql, qr]$，执行如下操作

- $mid = \lfloor \frac{l+r}{2} \rfloor$ 。

- 新建左、右子节点。
- 将当前节点左子节点与 $[ql,qr]\cap[l,mid]$ 做合并，贡献到新的左子节点上。
- 将当前节点右子节点与 $[ql,qr]\cap[mid+1,r]$ 做合并，贡献到新的左子节点上。
- 将当前节点左子节点与 $[ql,qr]\cap[mid+1,r]$ 做合并，贡献到新的右子节点上。
- 将当前节点右子节点与 $[ql,qr]\cap[l,mid]$ 做合并，贡献到新的右子节点上。

贡献是一个区间加操作，可以打标记。

初始时在线段树上加入一个 0。

考虑加入一个区间 $[l,r]$ 的操作，可以看成在 $[0,r]$ 上加，在 $[0,l)$ 上减，再将两棵树合并。所以节点数量最多 $\times 2$。实现时也可以按照该方法实现，常数上不会比直接加区间劣很多。

可以发现第 i 次与区间集合并的操作后树上有 $O(2^im)$ 个节点，与一个区间集合并会拆成进行 $O(m)$ 次加入区间操作，每次加入区间遍历树上（几乎）所有点，时间复杂度为 $O(2^{n/2}m^2)$ 。

算法八

注意到算法四中拆出的区间（差分后仅考虑前缀的情况下）正好是若干不交的线段树上的点所对应的区间。

于是不再需要先拆后加，可以在线段树上直接进行拆的操作。

事实上算法七也需要这个性质，它保证了第 i 次加入区间集的操作后树上节点数为 $O(2^im)$ ，从而保证了算法七的时间复杂度。

时间复杂度为 $O(2^{n/2}m)$ 。

如果对题目有什么建议欢迎来交流讨论。

祝大家在 NOIP2022 中都能取得理想的成绩！