Jed's std::pair

定位：签到题。事实上，这个题的核心在于整个输入序列合不合法在于它是不是一棵合法的**二叉树**。怎么理解呢？把 `std::pair` 视作非叶子节点，`int` 视为叶子结点，那么无论是什么“套娃”，最后一定是一颗二叉树的结构，而且出题人很好心地设置了 $n \le 20$，所以合法的序列甚至可以枚举出来，注意有一个很特别的情况就是 `1 int`，可能会让一些选手WA一发提交。时间复杂度可以做到 $O(n)$。

EC-Final 前夕

因为输入合法，最容易解决的情况是最终状态没有玩家有四个棋子在一条线上时，在这种情况下输出 `0`。否则，我们需要寻找游戏的有效中间状态——某时刻恰有一个玩家有四个棋子在一条线上。

考虑用 `列高度数组` 来辅助搜索。显然 `[6, 6, 6, 6, 6, 6, 6]` 表示游戏的最终状态。这给我们提供了大约$7^7 \approx 800,000$种可能状态的上限。我们可以进行BFS或DFS——这两种方法的时间复杂度都是$O(E)$，其中E是图的边数。每个状态最多有7个合法动作，因此最多差不多有$5,600,000$条边。主要难点在于写代码，细节非常多，一次写对很不容易。

压缩数列

考虑差分数组 $d[i] = a[i+1] - a[i]$，操作相当于交换 $d[i]$ 和 $d[i+1]$，对于答案而言肯定是 d 数组从小到大排列最好，操作次数就是逆序对数，归并排序求解即可。

爱丽丝和鲍勃（二）

可以证明，答案只和左边和右边的数相关，如果区间长度是奇数，就是 $\max(a[l], a[r])$；否则就是 $\min(a[l], a[r])$。

爱丽丝和鲍勃（三）

考虑全部都是 1 的情况，如果 n 是 $d+1$ 的倍数是后手胜，否则先手胜。因此这题的结论就是把 Nim 和这个结论拼起来。把数全部用二进制表示，用 $s[i]$ 表示第 i 位为 1 的数的个数，如果每一个 $s[i]$ 都是 $d+1$ 的倍数，就是后手胜，否则就是先手胜。

网络连接

考虑使用状压DP对每个 i 求出能到达的城市为 $\{2, \dots, i\}$ 的每个子集的概率 $dp_{i,st}$。考虑从 $dp_{i-1,st}$ 到 $dp_{i,st}$ 及 $dp_{i,1<<i|st}$ 的转移：第 i 座不能到达当且仅当 st 中的城市到 i 的道路全部关闭，即有

$$dp_{i,st} \leftarrow dp_{i-1,st} \cdot \prod_{j \in st}(1 - p_{j,i})$$

$$dp_{i,1<<i|st} \leftarrow dp_{i-1,st} \left(1 - \prod_{j \in st}(1 - p_{j,i})\right)$$

时间复杂度 $O(n2^n)$，可以通过。如果你被卡常了，可以采用子集DP的方式预处理乘积项优化至 $O(2^n)$。

猜数字

首先，每个人能确定自己的数是另两个的和或差。注意到如果有两个数相等，那么持有另一个数的人将可以确定自己的数。进一步，如果确定的两个数之一对应的询问次数比当前次数更少，就可以排除。由数学归纳法容易证明总是持有最大数的人最先确定。这样就可以得到一个递归算法：把最大的数替换成另两数之差，求出询问次数，再找到下一次问到最大数的人的次数就是当前答案。但这个算法会超时，

观察发现递归过程中较小的两个数在做辗转相除，于是可以用类欧几里得算法优化。时间复杂度 $O(T\log\max(a,b,c))$.

RVC

考虑到答案替换成更大的值也成立，即答案具有单调性，于是可以二分，不妨转化为判定性问题来做。假设判定的边长 a，那么可以贪心沿着底边尽可能地多码方形，码完一层后继续往上码，第一层能码的长度为 $a-2/\sqrt{3}$，第二层为 $a-4/\sqrt{3}$，一直往上加，直到长度为 0，所有长度向下取整加起来就是能放的方块数。