2025 年广东工业大学 ACM 新生程序设计竞赛（初赛）题解

广东工业大学 ACM 集训队

2025 年 10 月 19 日

题目难度分布预测

题目难度分布预测：

- 签到题：B、I、E
- Easy：F、G、H
- Easy-Medium A、J
- Medium C、D

接下来的题解将按照预测难度顺序展示。

B. 我不是 AI

简明题意

给定一行字符串组成的句子，判断该句子是否 **严格等于** "Are you AI"。

如果是，输出"I am not AI"；否则，输出"I do not know"。

签到题，按题意判断输入并按规则输出即可。

可以使用 cin 或 scanf 等方法按顺序读取三个字符串并判断。

也可以使用 getline 一次性读取整行判断。

时间复杂度 $O(1)$ 。

I. 2-冲突数对

简明题意

给定一个大小为 n 的数组 a。求出满足 $1 \le i < j \le n$ 且 $a_i \ne a_j$ 的所有 **值不同** 的 **有序下标对** (i, j) 的数量。

容斥原理求解：

- 所有满足 $i < j$ 的下标对总数为 $\frac{n(n-1)}{2}$。
- 我们减去 $a_i = a_j$ 的“非冲突”对数。
- 统计每个数字出现的次数 $count[x]$，则该数字贡献的非冲突对数为 $\frac{count[x] \times (count[x]-1)}{2}$。
- 答案 $= \frac{n(n-1)}{2} - \sum \frac{count[x](count[x]-1)}{2}$。

时间复杂度 $O(n)$。

E. 稳定区间

简明题意

给定一个长度为 n 的序列 a 和 q 个询问。每个询问给定一个区间 $[l, r]$，判断该区间内是否存在一个 **非空子序列**，其元素总和是 **3 的倍数**。若存在，则称区间 $[l, r]$ 是 **稳定的**。

我们只需要关注区间内每个数除 3 的余数：

- 若区间内存在 0（即 3 的倍数），直接满足。
- 若区间内存在 1 和 2，则 $1 + 2 \equiv 0 \pmod 3$，满足。
- 若区间内有三个 1，则 $1 + 1 + 1 \equiv 0 \pmod 3$，满足。
- 若区间内有三个 2，则 $2 + 2 + 2 \equiv 6 \equiv 0 \pmod 3$，满足。

综上，只要区间长度 ≥ 3，总是"Yes"，否则暴力判断即可。

时间复杂度 $O(q)$ 。

F. 【模板】Pollard-Rho

简明题意

这是一道 **交互题**。给定一个正整数 n $(2 \le n \le 2^{60}-1)$，交互器虽然声称要你求 n 的最大质因子 p，但实际上交互器已经内部算好了 p。你有 64 次尝试机会，每次输出一个整数 m，交互器返回 $d = p \oplus m$ 的 **二进制表示中数位 1 的个数**。当交互器输出 0 时，表示 $m = p$，此时程序应终止。

题目的名称和输入 n 是误导，这实际上是一道通过位运算交互来猜数的题目，完全不需要实现 Pollard-Rho 算法进行质因数分解（当然写了也能过）。

交互器持有目标值 p，我们需要逐位确定 p 的二进制位。由于 $p \le n < 2^{60}$，我们需要确定 60 个二进制位。给定的 64 次询问机会绰绰有余。

F. 【模板】Pollard-Rho

按如下步骤可以逐步确定出 p：

第一步：询问 $m = 0$。

- 由于 $p \oplus 0 = p$，交互器返回的便是 p 中二进制 1 的个数，记为 $base$。

第二步：从 0 到 59 枚举二进制位 k 。

- 对于每个 k，询问 $m = 2^k$（即只有第 k 位为 1，其余为 0）。
- 交互器返回 $cnt = \text{popcount}(p \oplus 2^k)$。
- 若 $cnt < base$，说明 p 的第 k 位原本是 1（异或后变成了 0，导致 1 的总数减少）。
- 若 $cnt > base$，说明 p 的第 k 位原本是 0（异或后变成了 1，导致 1 的总数增加）。

第三步：根据每一位的判断结果构造出 p 。

- 然后直接输出该值即可（此时返回值为 0，通过）

注意：如果任何时候读取到返回值 0 ，及时终止程序。

总询问次数为 $1 + 60 = 61$ 次，满足限制。

G. 互异排列——扩展

简明题意

给定一个 1 到 n 的排列 p。将数字 $n+1$ 插入到 p 的某个位置得到新排列 q（长度为 $n+1$）。要求新排列 q 的所有前缀和 $S_k = \sum_{j=1}^{k} q_j \bmod (n+1)$ 互不相同。判断是否存在这样的插入位置，如果存在，输出任意一个可行的排列 q，否则输出 -1。

分析前缀和的性质：

- 若将 $n+1$ 插入到 p 的中间或末尾（第 k 位，$k > 1$），由于 $n+1 \equiv 0 \pmod{n+1}$，会导致 $S_k \equiv S_{k-1} + 0 \equiv S_{k-1}$，即出现重复的前缀和。
- 因此，唯一的合法插入位置是 **开头**。
- 插入开头后，前缀和序列变为 $\{0, P_1, P_2, \ldots, P_n\} \pmod{n+1}$（其中 P_i 是原排列的前缀和）。
- 只需检查 $\{P_1 \bmod (n+1), \ldots, P_n \bmod (n+1)\}$ 是否包含 0 以及是否有重复元素。

时间复杂度 $O(n)$ 。

H. 互异排列——生成

简明题意

给定整数 n，求是否存在一个 1 到 n 的排列 p，使得在 p 中插入 $n+1$ 得到的排列 q 的所有前缀和对 $(n+1)$ 取模的结果 **互不相同**（即“互异排列扩展”问题有解）。如果存在，输出任意一个可行的排列 p，否则输出 -1。

根据 G 题结论，我们需要构造排列 p，使得其前缀和 $P_1, \ldots, P_n$ 模 $n+1$ 后是 $1, \ldots, n$ 的一个排列：

- 若 n 是偶数，$\sum_{i=1}^{n} i = \frac{n(n+1)}{2} \equiv 0 \pmod{n+1}$，这要求 P_n 为 0，但我们需要 P_n 非零，故无解，输出 -1。
- 若 n 是奇数，可以构造之字形的排列 p ，形如 $1, n, 2, n-1, 3, \ldots$ 。

时间复杂度 $O(n)$ 。

A. 最大子串匹配删除操作

简明题意

给定两个字符串 S 和 T。每回合可以执行“删除操作”（删除 T 开头的第一个字符）或“匹配删除操作”（若在 S 中存在子串 $S[i,j]$ 与当前的 T 完全相同，则删除 S 的前缀 $S[1,j]$ 并删除 T 开头的第一个字符）。目标是计算 **最多** 可以执行多少次匹配删除操作，直到 T 变为空字符串。

题目要求最大化匹配次数。每次操作（无论删除还是匹配删除）都会消耗 T 的一个字符。

设我们需要匹配 k 次，为了消耗最少的 S，我们应该贪心地匹配 T 的最后 k 个后缀（即长度为 $k, k-1, \ldots, 1$ 的后缀）。

根据贪心策略，较短的后缀应尽可能匹配在 S 的**末尾**，从而为前面较长的后缀预留更多的空间。

问题转化为：在 S 中 **倒序**依次寻找 T 的长度为 $1, 2, 3, \ldots$ 的后缀。

A. 最大子串匹配删除操作

从长度 $len = 1$ 开始按 **倒序**暴力匹配即可：

- 每次在 S 的剩余部分中寻找当前需要的目标后缀（长度为 len）。
- 找到后，将 S 中匹配位置之后的字符“删除”（即指针左移），答案计数器加 1，并尝试匹配长度为 $len + 1$ 的后缀。
- 由于第 k 次匹配消耗长度为 k 的子串，总长度满足 $1 + 2 + \cdots + k \leq |S|$，故每次匹配的最坏匹配次数为 $\sqrt{2|S|}$。

每次都暴力匹配即可，时间复杂度 $O(|S|\sqrt{|S|})$ ，足以通过。

使用 KMP 、Z 函数或哈希等字符串匹配算法匹配可进一步优化至 $O(|S| + |T|)$ 。

J. 收藏宝石

简明题意

给定 n 个宝石（节点 1 到 n）和房梁（节点 0）的连接图。每个宝石 i 有价值 a_i。**合法操作**是选择一个宝石切断所有与其相连的绳索，若**恰好**使得**仅有一个**宝石失去与房梁的连接（即被取下），则该操作合法。问是否存在一种取下顺序，能使每次都能**合法取下当前剩余宝石中价值最大的宝石之一**。

解题思路：时光倒流 $+$ 贪心。

J. 收藏宝石

正向思考判断“是否为割点”较为困难，我们可以考虑**逆向过程**：

- 正向是“拆除”，要求每次拆除一个非割点（保持剩余图连通）。
- 逆向则是“组装”，等价于从节点 0 开始，每次选择一个与当前连通块相连的节点加入连通块。

在逆向过程中，我们要加入的宝石价值序列 $v_n, v_{n-1}, \ldots, v_1$ 必须是**非递减**的。

考虑使用并查集，每次把 **所有价值最小的**宝石全部加入，然后检查此次加入的所有宝石是否 **全部与房梁相连**即可。

时间复杂度 $O(n\alpha(n))$。

J. 收藏宝石

我们还可以使用搜索：

- 题目限制等价于每个宝石在被摘下时，必须通过价值相同或更小的宝石连向房梁，我们可以建一个新的图。
- 对于每条 x 连向 y 的边，如果 x 的价值小于等于 y 的价值，则连一条 x 到 y 的边（反之亦然）。
- 从 0 号节点（房梁）出发，如果可以沿着上述边到达所有宝石，则 “YES”，使用任意一种搜索（如 DFS/BFS）即可判断。

时间复杂度 $O(n+m)$。

D. 师出同门

简明题意

玄真国和幻月国各有 n 个宗派的修士，数量分别为 a_i 和 b_i。每次战斗双方各选一个宗派 i, j 出战。若 $i = j$，双方不攻击；若 $i \neq j$，双方各损失一名修士（有效战斗）。求是否存在一系列战斗，使得战斗结束后，**至少有一方** 的所有修士 **全部阵亡**（即所有 $a_i = 0$ 或所有 $b_i = 0$）。如果存在，输出"YES"，并输出 **第一次有效战斗** 中双方选择的宗派编号 i 和 j $(i \neq j)$。

这是一道构造/贪心题目。若要让一方（如 A 方）全军覆没，需要满足两个条件：

- 总兵力条件：B 方总人数需大于等于 A 方总人数，即 $\sum b_i \geq \sum a_i$。
- 众数限制：对于任意宗派 i，A 方该宗派的人数 a_k 不能太多，必须能被 B 方其他宗派的人数消灭。具体条件为 $a_k \leq \sum b_i - b_k$。

显然，上述条件是必要条件。

D. 师出同门

大胆猜想其为充要条件，以下给出证明：

- 若存在 $a_k > \sum b_i - b_k$，显然无解。
- 否则若 **存在** $a_k = \sum b_i - b_k$，令 a_k 匹配所有其它 b_j ，其它所有 a_j 匹配 b_k 即可。
- 否则若 **全部** $a_k < \sum b_i - b_k$ ，即全部 $a_k + b_k < \sum b_i$ 由于任意一次操作后 $\sum b_i$ 减 1 ，且所有 $\forall k \to a_k + b_k$ 不增加，随意匹配直到出现存在 $a_k + b_k = \sum b_i$ ，即可转化为情况 2。

对于输出第一步操作，如果是上述第二种情况，则找到满足 $a_k = \sum b_i - b_k$ 的 a_k 匹配 b_j ，否则可以随意匹配。

C. 逃出生天

简明题意

在一个 n 行 m 列的网格猪圈中，一只小猪从右上角 $(1, m)$ 移动到左下角 $(n, 1)$ 的出口。网格中每行有一只老虎巡逻。**每时刻**，**小猪先执行移动**，**老虎后执行移动**。小猪可以移动到相邻格子或停留在原地。老虎沿当前方向移动，若遇到边界则反向再移动一格。小猪在任何时刻都 **不能** 移动到网格外，也 **不能** 与任何老虎处在同一格子中。判断小猪是否能够到达出口。

数据范围较小 $(n, m \leq 100)$，可以使用广度优先搜索（BFS）求解。

- 老虎的位置随时间 t 呈周期性变化，周期为 $2m - 2$。
- 定义状态 (r, c, t) 表示 t 时刻小猪位于 (r, c)，可以先模拟出每个状态是否合法。
- 状态转移时，枚举小猪的 5 种移动策略（上下左右停），并爆炸小猪移动到的目标格子在时刻 t（移动前）和时刻 $t + 1$（移动后）都不能有老虎。
- 若搜索到 $(n, 1)$ 则输出 YES。

时间复杂度 $O(nm^2)$ 。

结语

Thank you!