2025 广西科技大学程序设计大赛题解

启迪数字学院 ACM 实验室命题

2025.12.14

预期难度

- Easy：G、H、I
- Easy-Medium：F、C、O、M
- Medium：A、E、N、K
- Medium-Hard：D、B
- Hard：L、J

相信我，我真是签到题

- 直接输出“guang xi ke ji da xue”即可。

送气球

- 小何每次只能从基地 $(0, 0)$ 出发，送完一个气球后必须返回基地。
- 从一个点移动到另一个点的**直线距离必须是正整数**，每走一步代价为 1。
- 目标是用最少的步数，把所有气球送到指定位置。

关键观察

- 对于一个气球坐标 (x, y)，考虑从 $(0, 0)$ 到 (x, y) 的距离：

$$d = \sqrt{x^2 + y^2}$$

- 若 $x^2 + y^2$ 是**完全平方数**，则可以一步直达目标点。
- 若不是完全平方数，则无法一步到达，至少需要**两步**（中途经过一个点）。
- 若 $(x, y) = (0, 0)$，无需移动，代价为 0。
- 若 $\sqrt{x^2 + y^2}$ 是整数：
  - 去程 1 步，回程 1 步
  - 总代价：2
- 若 $\sqrt{x^2 + y^2}$ 不是整数：
  - 去程至少 2 步，回程至少 2 步
  - 总代价：4
  - 总时间复杂度：$\mathcal{O}(n)$

问题分析

- 给定一个整数 n，需要判断它的二进制表示是否是回文。即其从左到右和从右到左的二进制字符串完全一致。
- 需要处理多个测试用例，判断每个数是否是“镜子友好”的。
- 解决方法是：对于每个数 n，将其转换成二进制字符串，判断该字符串是否为回文。

解题思路

- 将每个数转为二进制字符串。
- 判断该二进制字符串是否为回文：回文就是正着读和反着读都一样。
- 如果是回文，则输出"Yes"，否则输出"No"。
- 复杂度：对于每个测试用例，我们需要将数 n 转为二进制，二进制位数最多为 60 位（因为 $n \leq 10^{18}$）。所以对于每个数，时间复杂度是 $O(\log n)$，总时间复杂度为 $O(t \cdot \log n)$，其中 t 是测试用例的数量。

问题分析

- 题目描述了一个石子博弈的场景，两个玩家小方和小谢交替从石堆中取石子，每次可以取 1、2 或 3 个石子。小方先手，最终取得最后一颗石子的玩家获胜。
- 两个玩家都遵循最优策略，即每个玩家都会尽量使自己获得的石子最多，且最终获胜。
- 本题的目标是判断在给定的石子数量下，谁将获胜，并且输出获胜者所获得的石子数。

解题思路

- 本题可以通过博弈论的角度来分析。设定状态 n 代表当前剩余的石子数：
  - 如果 $n\%4 == 0$，则小方处于劣势，因为无论小方拿多少石子，小谢都能通过合理策略使得自己总是能拿到最后一颗石子。
  - 如果 $n\%4 \neq 0$，则小方处于优势，因为小方可以通过拿走恰当数量的石子，使得剩下的石子数是 4 的倍数，这样小谢会处于劣势。
- 当 $n\%4 == 0$ 时，小谢必定获胜；否则，小方获胜。
- 获胜者所获得的石子数量为 $n/4 + n\%4$。

妙手不如举手

本题本质：模拟 + 条件判断。

在第 $2 * i - 1$ 步需要同时检查三件事：

- 对手是否犯规：$r_i = 1$；
- 当前执子者是否处于劣势；
- 举报是否来得及：$i + t \leq T$。

分值定义：

$$\text{score}_P = \text{妙手} - \text{俗手}$$

若玩家 H 有 $\text{score}_H < \text{score}_C$，则处于劣势。

满足三个条件即可立即输出步数；若始终未满足，输出：

```
PLAY TO THE END
```

最大三角形面积·方法一：三分搜索

定义函数 $f(y)$ 为：当水平线取高度 y 时，小三角形 abc 的面积。

由相似三角形可得，此时底边长度为：

$$|ab| = \frac{y_0 - y}{y_0} \cdot |x_1 - x_0|$$

因此面积函数为：

$$f(y) = \frac{|ab| \cdot y}{2} = \frac{1}{2} \cdot \frac{y_0 - y}{y_0} \cdot |x_1 - x_0| \cdot y$$

这是在区间 $[0, |y_0|]$ 上的 单峰函数（先增后减），可以用三分搜索最大值。

最大三角形面积·方法二（推导）

设水平线为 $y = h$。

由相似三角形可得交点横坐标：

$$a_x = x_0\left(1 - \frac{h}{y_0}\right), \quad b_x = x_1\left(1 - \frac{h}{y_0}\right)$$

底边长度：$|a_x - b_x| = |x_1 - x_0|\left(1 - \frac{h}{y_0}\right)$

面积函数：$S(h) = \frac{1}{2}|x_1 - x_0| \cdot h\left(1 - \frac{h}{y_0}\right)$

这是开口向下的二次函数，最大值出现在：$h = \frac{y_0}{2}$

代回可得：$S_{\max} = \frac{|x_1 - x_0| \cdot |y_0|}{8}$

问题分析

- 题目给定了两个整数 a 和 b，目标是通过一系列异或操作将 a 变为 b。每次操作可以选择一个整数 x，满足 $0 \le x \le a$，并执行

$$a \leftarrow a \oplus x$$

- 我们的目标是判断是否可以在不超过 100 次操作内将 a 变为 b，并输出具体的操作步骤；如果无法实现，则输出 -1。

思路一：异或的性质

- 异或操作具有以下基本性质：
  - $a \oplus a = 0$，即任意数与自身异或的结果为 0；
  - $a \oplus 0 = a$，即任意数与 0 异或不会改变其值；
  - 异或运算满足交换律和结合律，可以任意调整运算顺序。
- 利用上述性质，可以构造如下解法：
  - 首先计算正整数 a 的二进制位数，生成与该位数相同的全 1 数 num (例如，若 a 的二进制为 3 位，则 $\text{num} = 0\text{b}111 = 7$，可通过位运算 $\text{num} = (1 \ll \text{bit\_length}(a)) - 1$ 计算，其中 $\text{bit\_length}(a)$ 表示 a 的二进制位数)；若 $b > \text{num}$，则问题无解，输出 -1；否则利用异或运算的按位取反性质，构造 $x = a \oplus \text{num}$ 和 $y = b \oplus \text{num}$ 两个数作为解 ($\oplus$ 表示按位异或运算)。

- 另一种思路是从二进制逐位分析异或操作的效果：
  - 将 a 和 b 转换为二进制表示，并逐位进行比较；
  - 若在某一位上 a 与 b 不同，则可以通过选择合适的 x，使得该位在异或后变为目标值。
- 每次操作通过精心构造 x，使 a 在若干位上向 b 靠近。

问题分析

- 本题要求根据提交记录计算最终的比赛排行榜，并确定奖项的获奖队伍数量。
- 排行榜排序规则为：解出题目数量降序，若相同，则按总罚时升序，若解出数量与罚时均相同，则按队名字典序升序。
- 总罚时的计算规则：每道题目首次 AC 的时间 $+$ 错误提交次数 $\times$ 20 分钟。
- 通过对输入数据进行处理，我们需要维护每个队伍的解题数、总罚时以及每个题目的错误提交次数。

解题思路

- 按照题目的描述来直接模拟就行。
- 首先遍历所有提交记录，记录每个队伍对每道题目的错误提交次数和解题情况。
- 对于每个队伍，如果某题目 AC 了，则计算罚时并记录解题数。
- 最后，根据解题数、罚时和队伍名字典序对队伍进行排序。
- 按照 ACM 竞赛规则计算金、银、铜奖的获奖队伍数量。

关键点总结

- 使用 map 记录每个队伍的提交情况，包括每道题的错误次数和是否已解题。
- 排序的关键在于按解题数、罚时和字典序进行多重排序。
- 对于奖项数量的计算，利用队伍总数和比例进行向上取整。

问题分析

- 本题要求从给定的能量节点中选取最多连续的 k 个节点，找到能量总和最大的情况。
- 具体而言，题目要求在一个长度为 n 的数列中找出一个子段，其长度不超过 m，使得子段中的所有元素和最大。
- 关键是通过合理的滑动窗口和前缀和方法，快速找到最大连续和。

解题思路

- 本题的关键在于如何高效地计算连续子数组的最大和。由于最多可以收集 m 个节点的能量，滑动窗口技术非常适合这类问题。
- 通过前缀和数组 ‘pre[i]‘ 来存储从第一个节点到第 i 个节点的累积能量，计算任意区间的和就可以通过前缀和的差来获得。
- 使用单调队列来优化滑动窗口操作，保持窗口内的前缀和递增，使得每次能够在 $O(1)$ 时间内计算当前子段的和。

算法实现

- 计算前缀和数组 ‘pre‘，使得 ‘pre[i]‘ 表示前 i 个节点的能量总和。
- 使用双端队列（deque）来存储有效的窗口范围。队列中保存的是前缀和的下标，保持队列中对应的前缀和递增。
- 每次在队列的头部弹出过期的下标（即距离当前下标超过 m 个位置的），然后计算当前子段的能量和，并更新最大值。
- 通过不断滑动窗口和更新最大值，最终可以得到最大连续和，复杂度是 $O(n)$。

问题分析

题目要求在不超过 100 次操作内，把给定的正整数 n 变成一个完全平方数。
每次操作可以：

- 选择当前 n 的一个正约数 x；
- 且每次选的 x 必须互不相同；
- 然后令 $n \leftarrow n + x$。

关键在于：如何保证操作次数少、且 n 不会超过 10^{18}。

解题思路

注意到一个非常重要的性质：

$$b = n\&(-n)$$

表示 n 的 **最低位二进制 1** 所对应的值。

- b 一定是 n 的正约数；
- 不同的 n 会得到不同的 b，因此不会重复使用约数；
- 每次加上 b，都会改变 n 的二进制结构，使其逐步“趋近”完全平方数。

问题分析

城市被抽象成一个 $n \times m$ 的网格，但输入并不是直接给出格子，而是一个 $(2n+1) \times (2m+1)$ 的字符矩阵：

- 奇数行、奇数列位置对应真实的“格子”；
- ‘|‘ 和 ‘-‘ 表示相邻格子之间不可通行的墙；
- 空格、‘S‘、‘T‘ 表示可通行。

本质上，这是一个 **带障碍的网格最短路径问题**，目标是从起点 S 到终点 T 的最短步数。

建图思路

将输入的字符矩阵 **还原为真正的** $n \times m$ **网格**：

- 用二维数组 (i, j) 表示第 i 行第 j 列的格子；
- 对于每个格子，检查上下左右四个方向：
  - 若对应位置是空格、S 或 T，说明该方向可以通行；
  - 否则存在墙，无法通过。

用 $b[i][j][k]$ 记录格子 (i, j) 是否能向第 k 个方向移动。

然后用 BFS，若 BFS 结束仍未访问到 T，则说明无法到达。

问题分析

赛道被抽象为一张 **无向带权图**：

- 节点表示检查点；
- 边表示赛道；
- 边权 w 表示该赛道的最高限速。

对于一次查询 (s, t)，需要从 s 到 t 选择一条路径，使得：

$$\max(\text{路径上边权}) - \min(\text{路径上边权})$$

最小。若无法连通，则输出 -1。问题等价于：

在所有能连通 s 和 t 的路径中，最小化最大边权与最小边权之差。

算法思路

采用 **排序** + **并查集**的枚举策略：

1. 将所有赛道按速度限制 w 从小到大排序；
2. 枚举路径中可能的最小边权 $a[i].w$；
3. 从第 i 条边开始，依次加入更大的边，用并查集合并端点；
4. 一旦 s 与 t 连通，当前区间

$$a[j].w - a[i].w$$

即为一种可行解。

- 对每个 i，首次连通时的 j 一定是当前最小的最大边权；
- 枚举所有 i，取最小差值即可。

问题分析

给定 n 个闭区间 $[l_i, r_i]$，你可以选择删除其中恰好一个区间（或者不删除），目标是让剩余所有区间的交集所包含的整数点数量（长度）最大。

利用 multiset 的自动排序特性分别维护所有线段的左、右端点集合，通过“删除当前线段端点 -> 查询剩余集合的最大左端点与最小右端点 -> 计算交集更新答案 -> 恢复端点”的流程，在 O(n $\log n$)

题意回顾与博弈模型

题意简述：

- 有 n 个箱子，且恰有 1 个有毒，其余安全；
- 小方先手，两人轮流操作，每回合选择一个未被移除的箱子进行检测；
- 检测准确率为 p，对安全/有毒箱的判定均以概率 p 正确；
- 检测为 TOXIC：箱子直接移除；
- 检测为 SAFE：当前玩家可以选择立刻开箱（赢/输立判），或不打开并移除该箱。

当场上只剩 1 个箱子时，轮到行动的玩家必须直接打开该箱，安全则胜，有毒则负。

额外规则：小方拥有一次 二次检测权：对某次选中的箱子检测两次，若至少一次为 TOXIC 则直接移除；若两次均为 SAFE，再决定开或不开。特权至多使用一次。

单次检测：后验概率计算

设当前剩 m 个箱子，其中 1 个有毒，$m-1$ 个安全。

先验概率：

$$\Pr(\text{选中安全箱}) = 1 - \frac{1}{m}, \qquad \Pr(\text{选中毒箱}) = \frac{1}{m}.$$

单次检测结果为 SAFE 的概率：

$$q_S^{(1)} = p\left(1 - \frac{1}{m}\right) + (1-p)\frac{1}{m}.$$

在结果为 SAFE 的条件下，该箱真实安全的后验概率：

$$\pi_1(m) = \Pr(\text{安全} \mid \text{SAFE}) = \frac{p\left(1 - \frac{1}{m}\right)}{q_S^{(1)}}.$$

若此时选择立刻开箱，当前玩家的胜率为 $\pi_1(m)$；若选择不打开，则移除该箱并将回合交给对手。

二次检测：后验概率与事件概率

二次检测同一箱子：

- 若两次中至少一次为 TOXIC：箱子被移除，回合交给对手；
- 若两次均为 SAFE：小方可以选择开箱或不打开。

两次检测均为 SAFE 的概率：

$$q_{ss} = \left(q_s^{(1)}\right)^2.$$

在“两次均为 SAFE”的前提下，该箱真实安全的后验概率：

$$\pi_2(m) = \Pr(\text{安全} \mid \text{SAFE, SAFE}) = \frac{p^2\left(1-\frac{1}{m}\right)}{p^2\left(1-\frac{1}{m}\right) + (1-p)^2\frac{1}{m}}.$$

此时若立刻开箱，当前玩家的胜率为 $\pi_2(m)$；否则仍然是移除该箱，回合交给对手。

状态定义：$V[m]$，$W[m]$

为方便处理回合对称性，定义：

状态 $V[m]$

$V[m]$ 表示：在剩余 m 个箱子、轮到当前玩家行动，且双方都没有二次检测权 时，当前玩家的最优胜率。

边界： $V[1] = 0$，因为只剩一个箱子，它必然是毒箱，当前玩家被迫打开而必输。

状态 $W[m]$

$W[m]$ 表示：剩余 m 个箱子，轮到 仍然拥有一次二次检测权 的小方行动时，她的最大获胜概率。

状态定义（无特权）：$V[m]$

单次检测的分支：

- 结果为 TOXIC（概率 $q_T^{(1)} = 1 - q_S^{(1)}$）：箱子移除，剩 $m-1$ 个箱子，轮到对手，当前玩家胜率为 $1 - V[m-1]$；
- 结果为 SAFE（概率 $q_S^{(1)}$）：当前玩家在“立即开箱”和“让回合给对手”之间二选一：

$$\max\big(\pi_1(m),\, 1 - V[m]\big).$$

因此（概念上的）递推关系：

$$V[m] = q_T^{(1)} \cdot (1 - V[m-1]) + q_S^{(1)} \cdot \max\big(\pi_1(m),\, 1 - V[m]\big).$$

实际实现时会根据 $\pi_1(m)$ 与 $1 - V[m]$ 的大小分类讨论，从而解出 $V[m]$。

有特权的状态：$W[m]$

小方可以选择：

- 直接按“无特权”的策略行事：胜率为 $V[m]$；
- 在当前回合使用二次检测权：
  - 两次检测至少一次为 TOXIC（概率 $1 - q_{SS}$）：箱子移除，局面变成“剩 $m-1$ 个箱子、轮到对手且无特权”，当前胜率 $1 - V[m-1]$；
  - 两次均为 SAFE（概率 q_{SS}）：小方可在 $\pi_2(m)$（立刻开箱）与 $1 - V[m]$（移除后让对手）之间取较优。

使用特权当回合的期望胜率：

$$E(m) = q_{SS} \cdot \max\big(\pi_2(m),\, 1 - V[m]\big) + (1 - q_{SS}) \cdot (1 - V[m-1]).$$

因此：$W[m] = \max\big(V[m],\, E(m)\big)$.

最终答案即为 $W[n]$。

题意分析

本题是经典的 Lights Out 模型：点击 (r, c) 会翻转其自身及上下左右的灯。

关键性质：

- 翻转操作在 $\mathbb{F}_2$ 上可交换，点击顺序无关；
- 每个格子点击 0 次或 1 次即可（因为两次等于没点）；
- 目标是找到一个 $N \times N$ 的 $0/1$ 点击矩阵，使最终状态全为 1。

一个重要事实：

- 第一行一旦确定，整张表的点击方案可以唯一从上到下推导出来。
- 只有最后一行无法继续被修正，因此它必须由第一行决定。

如何从上到下递推

我们把灯阵看作一层一层的面板。点击第 r 行会影响第 r 行本身，也会影响第 $r-1$ 行和第 $r+1$ 行（上下相邻）。

- 当第 r 行已经确定时，第 $r-1$ 行的最终状态必须是“全亮”；
- 因此第 r 行的点击方案 X_r 可以由 X_{r-1} 唯一决定；
- 也就是说：上一行的状态 强制 下一行必须如何点。

这意味着：

- 给定第一行 $X_0 = f$，可以按规则逐行计算：$X_1 \to X_2 \to \ldots \to X_{N-1}$
- 整张点击矩阵被唯一确定；
- 只有最后一行无法再被修补，因此最后一行的亮灭完全由 f 决定。

因此：第一行决定全局，递推过程完全是线性的逐层修正。

第一行与最后一行的关系

因为其余行都由顶部状态推进得到，并且每一步的翻转（异或）、左右移以及行间传递全都发生在 $\mathbb{F}_2$，整个系统天然保持线性结构。

因此最后一行的状态可以写成：

$$L(f) = f_1 L(e_1) \oplus f_2 L(e_2) \oplus \cdots \oplus f_N L(e_N) \oplus L(\mathbf{0})$$

其中：

- e_i 为“第一行只点第 i 列”的操作，
- $L(e_i)$ 表示该点击对最后一行造成的影响，
- $L(\mathbf{0})$ 为完全不点时产生的偏移量 d。

这给出了构造方程组的推导方式：

令矩阵 A 的第 i 列为 $L(e_i) \oplus d$，则想让最后一行全亮等价于

$$Af = \mathbf{1} \oplus d.$$

解得第一行 f 后，即可从上到下唯一恢复整个点击矩阵。

线性结构：构造 $Af = \mathbf{1} \oplus d$

令 e_i 为第一行仅第 i 列为 1 的点击向量。

步骤：

1. 计算 $L(e_i)$：若第一行只点第 i 列，最后一行会变成什么？
2. 将 A 的第 i 列定义为 $L(e_i) \oplus d$，其中 $d = L(\mathbf{0})$ 为不点任何灯时最后一行的偏移量。
3. 要求：

$$Af = \mathbf{1} \oplus d$$

这是一个 $\mathbb{F}_2$ 上的线性方程组，可用高斯消元（比特集优化）求解。

若有解 f，即可从第一行开始推导出完整点击矩阵，输出 YES。

算法流程与复杂度

完整流程：

1. 明确整个矩阵的状态可以由第一行决定；
2. 计算常量偏移 $d = L(\mathbf{0})$；
3. 对每个 i 求 $L(e_i)$，构造矩阵 A；
4. 解方程组 $Af = \mathbf{1} \oplus d$；
5. 若有解，从第一行向下恢复整个 $N \times N$ 的点击方案。

时间复杂度：

$$O(N^3/64)$$

使用 bitset / uint64 优化高斯消元后可轻松通过 $N = 300$。