

“规律未来” 2026 年广东工业大学 ACM 程序设计竞赛题解

26 校赛出题组

2026 年 03 月 22 日

A. 魔法石法阵

简明题意

给定 a 个黑点、 b 个白点，构造一个**简单连通无向图**，满足每个点：相邻黑点数 = 相邻白点数。要求边数最少，无解则输出 '-1'。

不妨设 $a \leq b$ 。

特殊情况只有一种：当 $a = 0, b = 1$ 时，图中仅有一个点，不连边即可满足连通性与条件要求，答案为 0。

对于其余情况，当且仅当 $\frac{a(a-1)}{2} \geq b + (b \bmod 2)$ 时有解，否则无解。

记黑黑边、白白边、黑白边数分别为 E_{bb}, E_{ww}, E_{bw} ，由题意可得 $2E_{bb} = E_{bw} = 2E_{ww}$ ，因此总边数最小时，本质上等价于让白点内部总度数最小。又由于每个白点都必须至少有一个白邻居，故白点内部总度数至少为 $b + (b \bmod 2)$ ，再结合黑点内部边数上界 $\binom{a}{2}$ ，即可得到上述判定条件。

A. 魔法石法阵（续）

一种构造方法是：

令白点内部达到最小需求，若 b 为偶数，则令每个白点内部度数为 1；否则令一个白点内部度数为 2，其余为 1。

于是白点还需连接的黑边数也随之确定，并将度数尽量均匀分配，也就是需要在 a 个黑点内构造一个 $b + (b \bmod 2)$ 边的简单图，且保证连通即可。对于只连一个白点的黑点，需要注意预期连接的白点不能已经连接过，否则可能形成多个连通块。

时间复杂度为 $O(a + b)$ 。

B. 凸包

简明题意

给定 n 个互不重合的整点，任意三点不共线。需要选出两个非空且不相交的点集 A 和 B ，使得它们的凸包完全不相交（内部和边界均无公共点）。保证有解，输出任意一种方案。

由于 $n \geq 6$ ，取所有点中 x 坐标最小的三个点作为集合 A ， x 坐标最大的三个点作为集合 B 。

因为任意三点不共线，所以 A 和 B 的凸包都是三角形，且 A 中所有点的 x 坐标严格小于 B 中所有点的 x 坐标（若存在相等的 x 坐标，则必有两个点 x 相同，再结合另一侧的某个点就会产生三点共线，矛盾）。

因此两个三角形被一条竖直线分隔，凸包完全不相交。输出对应的点编号即可。

时间复杂度为 $O(1)$ 。

C. 消失的元素

简明题意

本题为通信题。给定一个大小为 n 的严格递增集合 $S \subseteq [1, n + m]$, Alice 需要删去恰好一个元素后发送剩余的 $n - 1$ 个数; Bob 只根据收到的序列 S' 唯一确定被删去的元素。要求设计一种对任意数据都正确的策略。

设原集合为数列 $a_0, a_1, \dots, a_{n-1}$ (下标从 0 开始), 其元素和为 sum 。Alice 删除下标为 $k = sum \bmod n$ 的那个元素 a_k 。

C. 消失的元素（续）

这样 Bob 收到剩余 $n - 1$ 个数的数列 S' 后，再计算这些数的和 sum' ，只要枚举缺失的是哪个数 x ，就能得到完整数列的元素和为 $sum' + x$ ，于是被删元素在完整数列中的下标应为 $(sum' + x) \bmod n$ 只要这个值恰好等于 x 插回后所在的位置，就说明这个 x 就是被删除的数。

对于这个方案的正确性，可以假设若某个其他数列也满足条件，就要存在非零 y 使得有 $a_k + n + y$ 在 S' 的排名是 $k + y$ ，也就是 a_k 之前或之后要有至少 n 个空位。但一共最多也就 $n + m \leq n \times 2 - 1$ 个格子，而已经存在的数占了 n 个，所以不可能有其他合法的数列了，即 $a_k = x$ 唯一。

时间复杂度为 $O(n + m)$ 。

D. 收纳收纳袋

简明题意

一个袋子至多装一个袋子，且内层袋子的实际大小不能超过外层袋子的容量，装入后外层袋子的实际大小会变成“自身大小 + 内层实际大小”

区间询问所有能否最终只占一个背包格子；若不能，求把这段内所有袋子的容量同时增加多少值才能。

对**相邻**两个袋子 x, y ，若 $a_x + b_x \leq a_y + b_y$ ，交换可证把 x 放在 y 前面不会更劣。

所以最优顺序一定是按 $a_i + b_i$ 升序，令第 i 个收纳袋排名为 rk_i 。

D. 收纳收纳袋（续）

对于一个区间 $[L, R]$ ，对于某个收纳袋 j ，其需要增加的容量：

$$x_j = \text{区间 } [L, R] \text{ 内排名在 } j \text{ 前面所有 } a \text{ 的和} - b_j$$

答案即为：

$$\max(\text{区间 } [L, R] \text{ 内的所有 } x_j \text{ 的最大值}, 0)$$

考虑以排名为下标建线段树，再使用莫队即可得到此题 $O(n\sqrt{n}\log n)$ 的做法，但不足以通过。

考虑加一个收纳袋对 x_j 集合的影响，注意到加一个数相当于后缀减，维护个单调的结构即可求得所有 x_j 的最大值。

D. 收纳收纳袋（续）

考虑左端点大小为 B 的块内回滚，右端点单调递增的回滚莫队，按块内 $a_i + b_i$ 作为分界值可以将右侧的收纳袋分为 B 个单调的结构维护，使用并查集即可。

于是我们得到了回滚莫队，足以通过此题。
时间复杂度为 $O(n\sqrt{n}\alpha(n))$ 。

E. 周长

简明题意

有一个大小为 $n \times m$ 的矩阵，每个格子上有一个正整数权值 $a_{i,j}$ 。他想要圈定一块封闭图形，要求其周长恰好为 C ，且封闭图形内部的格子权值之和尽量大，请你输出一个和的最大可能值。

首先可以发现，因为权值为正整数，所以如果图形 A 能包住图形 B ，那么 A 一定不劣，由此可以推出封闭图形的形状是矩形时权值之和最大。

于是容易想到考虑枚举矩形。周长为 C 的矩形意味着长加宽为 $\frac{C}{2}$ ，所以可以枚举长和左上角的点，复杂度 $O(n^2m)$ ，当 $n = m = 10^3$ 时会被卡，所以我们需要一个更快的做法。

E. 周长（续）

当左上角的点固定时，考虑所有可行的右下角，它们组成了一条斜线——这是有依据的，因为对于左上角坐标 (x, y) ，其右下角坐标 (x', y') 一定满足 $x + y = x' + y' + \frac{C}{2}$ ；于是我们可以发现，枚举斜线 $x + y = k$ 上的所有点时，实际上我们在重复枚举另一条斜线 $x + y = k + \frac{C}{2}$ 的点。

这里存在一个针对重复枚举的优化：当 $x_1 > x_2$ 时， (x_2, y_2) 的最优右下角（即使得答案最大的右下角） (x'_2, y'_2) 相对 (x_1, y_1) 的最优右下角 (x'_1, y'_1) 一定有 $x'_1 < x'_2$ ！可以通过画图 + 剪切法证明这一结论，这里不做赘述。于是我们可以得到：这个决策具有单调性！

所以我们可以利用决策单调性分治来优化，只需要枚举斜线，然后对于每条斜线，在不超过 $O((n + m) \log n)$ （假设 $n > m$ ）的时间复杂度内完成枚举。

由此，我们可以得到一个总时间复杂度不超过 $O(nm \log n)$ 的做法。

F. 瞭望塔

简明题意

给定若干瞭望塔，每次可将某个塔高度减一，求使 $\sum k_i h_i^2 \leq X$ 的最少操作次数。

攻击某个塔带来的减少值为 $k_i \cdot (2h_i - 1)$ ，单调递减。

二分减少值的最小值 mid ，对每个塔都执行操作减少值大于 mid 的攻击，判断最后观测范围之和是否 $\leq X$ ；

注意最后操作减少值等于 mid 的攻击次数单独统计。

G. fufu 购买计划

简明题意

有面值 $1 \sim n$ 的硬币各一枚，需要选出若干枚，使其和恰好为 m 。

同时对每个 i 要满足：所选硬币中，面值不超过 i 的硬币数量至多为 a_i 。

需要判断是否有解，并输出任意一组合法方案。

可以先将限制数组化为为单调递增形式，即令前面的限制不超过后面的限制，这样不会改变可行性，便于统一处理。

可以先构造一个满足限制且总和尽量小的合法选币集合；如果其和已经超过 m ，则无解。若其和不超过 m ，再在不改变选币数量的前提下，将若干已选硬币，从大到小地尽量替换成更大的面值，从而逐步把总和补到 m 。

G. fufu 购买计划（续）

由于限制只约束“小于某一值的面值硬币选了多少个”，把已选硬币变大不会破坏合法性。

如果最后仍然补不到 m ，则无解；否则当前这组就是答案。

时间复杂度为 $O(n)$ 。

H. 死亡变速

简明题意

给定若干个分段变速点形成的 BPM 表，支持多次查询，在谱面时间（拍）与物理时间（秒）之间进行双向精确转换。

做法一

先预处理每个变速点 s_i 对应的累计物理时间 pre_i ，且 $pre_i = pre_{i-1} + (s_i - s_{i-1}) \cdot \frac{60}{b_{i-1}}$ 。

这样在第 i 段内，拍数到秒数的转换就是一次函数： $t = pre_i + (x - s_i) \cdot \frac{60}{b_i}$ 。

H. 死亡变速（续）

秒数到拍数则反解为 $beat = s_i + (t - pre_i) \cdot \frac{b_i}{60}$ 。

第一种做法是 **预处理 + 二分**：‘B x’ 在 s 数组中二分找到所在段，‘S x’ 在 pre 数组中二分找到所在段，再代公式即可。

时间复杂度为 $O(n + q \log n)$ 。

做法二

第二种做法是 **离线 + 累加**：把所有 ‘B’ 查询按拍数排序、所有 ‘S’ 查询按秒数排序，各用一个指针顺着分段线性扫描，因为查询值递增后所在分段也只会单调向后移动，所以不必每次二分。

时间复杂度为 $O(n + q \log q)$ 。

H. 死亡变速（续）

精度问题

本题前缀时间可达 $pre_i \leq 60 \times 10^9 = 6 \times 10^{10} \approx 2^{35}$ ，而 ‘double’ 尾数为 52 位，在一个约为 2^{35} 的数附近，相邻可表示数间距约为 $2^{35-52} = 2^{-17} \approx 7.6 \times 10^{-6}$ ，已经大于题目要求的 10^{-6} ；

同时最短一段时间却可能只有 $\frac{60 \cdot 1}{10^9} = 6 \times 10^{-8}$ ，小于这个间距，所以当前缀和已经很大时，再加这种极短时间段，‘double’ 直接加不进去。

测试点中存在 BPM 在极大极小反复横跳的数据，因此需要使用 ‘long double’ 才能通过。

I. 凹包

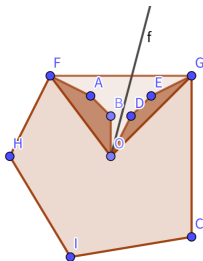
简明题意

选若干个顶点按顺序连接形成无交叉的多边形，使得存在一条从 s 出发的射线**不穿过这个多边形内部**

I. 凹包（续）

如图所示，求出 n 个点的凸包面积，然后枚举射线方向，则面积减去一个三角形，再加上射线两侧的**凸链**。

容易得到最优凹包上的所有点极角序是单调递增的，先极角排序，然后使用一个栈维护前后缀凸链面积即可。



J. 最远的彼此

简明题意

有一个未知的 $n \times n$ 01 矩阵，且至少存在一个 1。每次可以询问一个连续子矩阵，交互器返回该子矩阵内所有元素的按位或，即判断其中是否存在 1。

要求在不超过 $8n$ 次询问内，求出所有值为 1 的位置两两之间的最大曼哈顿距离。

设所有值为 1 的位置为 (x, y) 。记 $s = x + y$, $d = x - y$ ，则任意两点的曼哈顿距离可写为 $|x_1 - x_2| + |y_1 - y_2| = \max(|s_1 - s_2|, |d_1 - d_2|)$ 。

因此答案就是 $\max(\max s - \min s, \max d - \min d)$ ，所以问题转化为求所有 1 的点的 $x + y$ 和 $x - y$ 的最小值与最大值，也就是需要在一个方向上用 $2n$ 次求 $x + y$ 的最小值/最大值。

J. 最远的彼此（续）

具体方法比较多，一种较为简单的是：

可以用前缀子矩阵 $[1, x] \times [1, y]$ 进行扫描。固定 x 后，让 y 单调递减，询问该前缀中是否存在 1。这样可在总共不超过 $2n$ 次询问内求出一个方向上的最值。

总共进行 4 次扫描，询问次数不超过 $8n$ ，满足题目要求。

时间复杂度为 $O(n)$ 。

K. 共享单车

简明题意

未来 n 天，每天可用的单车公司类型为：‘A’（仅 A 可用）、‘B’（仅 B 可用）、‘C’（A、B 均可用）。

- 购买任一公司的套餐需支付 K 元，有效期为连续 L 天。
- 两家公司套餐不通用。
- 每天必须有对应的套餐可用。

求最少需要支付的总金额。

K. 共享单车（续）

本题采用 **动态规划 (DP)** 解决。核心思路是维护每一天结束时，所持有的套餐及其剩余有效期，并利用 'pair' 的性质在保证花费最少的同时，尽可能让有效期更长。

状态定义

定义 $f[j][i]$ 为一个 `'std::pair<int, int>'`，表示在第 i 天结束时，主要使用的公司为 j ($j = 0$ 代表 A, $j = 1$ 代表 B) 时的状态：

- **第一维 ('first')**：已购买套餐总次数的负值 ($-count$)。使用负值是为了方便直接使用 `'std::max'` 筛选出：首先花费最少，其次剩余天数最多的状态。
- **第二维 ('second')**：当前公司 j 套餐的剩余有效天数。

K. 共享单车（续）

状态转移

- **同公司延续/续费**：若当前公司 j 的套餐未过期 ($f[j][i].second > 0$)：

$$f[j][i+1] = \max(f[j][i+1], \{f[j][i].first, f[j][i].second - 1\})$$

若已过期：需支付 K 元购买新套餐，剩余天数重置为 $L - 1$ 。

$$f[j][i+1] = \max(f[j][i+1], \{f[j][i].first - 1, L - 1\})$$

- **跨公司切换**：这是本题 DP 的精髓：即便当前在用 $!j$ 公司的车，若今天要用 j 公司的车（比如 $S[i] = 'A'$ 而之前在骑 B），我们可以在今天买 j 公司的套餐。

如果 $!j$ 公司的套餐还能撑到 nxt 天，那么我们买的 j 公司套餐在 nxt 天时依然有效，剩余天数为 $L - (nxt - i)$ 。

$$\text{转移方程：} f[j][nxt] = \max(f[j][nxt], \{f[!j][i].first - 1, L - (nxt - i)\})$$

时间复杂度： $O(n)$ ，**空间复杂度：** $O(n)$ 。

L. PPIIIGG

简明题意

给定一个仅由 'P', 'I', 'G' 组成的字符串 S 。统计满足以下条件的子序列数量：

- 子序列形式为： a 个 'P' + b 个 'I' + c 个 'G' ($a, b, c \geq 1$)。
- 数量关系满足： $a + c - 1 \leq b$ 。
- 答案对 $10^9 + 7$ 取模。

考虑将 $b - a - c$ 的动态平衡值作为 DP 的一维。定义 'dp[j][state]' 为：

- 'j'：平衡值偏移量。令 $j = (b - a - c) + n + 1$ 。
- 'state'：当前子序列所处的阶段（0：正在选 'P'，1：正在选 'I'，2：正在选 'G'）。

满足条件的最终状态为 $b - a - c \geq -1$ ，对应到代码中的下标即为 $j \geq n$ 。

L. PPIIGG (续)

每个字符的加入都是一次状态转移，为了保证每个字符在一次转移中只被使用一次，需要根据平衡值的变化方向决定遍历顺序：

- 当加入一个 'P' 时，平衡值 $b - a - c$ 减 1，因此需要正序枚举 j ，使用 'dp[j+1]' 更新 'dp[j]'。
- 当加入一个 'I' 时，平衡值 $b - a - c$ 加 1，因此需要倒序枚举 j ，使用 'dp[j-1]' 更新 'dp[j]'。
- 当加入一个 'G' 时，平衡值 $b - a - c$ 减 1，因此需要正序枚举 j ，使用 'dp[j+1]' 更新 'dp[j]'。

所有阶段 2 且平衡值下标 $j \geq n$ 的状态总和即为答案。

M. 微软大战代码

简明题意

给定一棵 n 个节点的树，根为 1。每条边 (x_i, i) 有一个小写字母 ch_i 。

- 评价串：从根到节点 u 路径上的字符按顺序拼接。
- 排序规则：评价串字典序小的排在前面（前缀比长串小）。若评价串完全相同，则节点编号 i 小的排在前面。

要求输出所有节点排序后的编号序列。

考虑树的 DFS 遍历，优先访问评价串字典序较小的子节点。最终的访问顺序即为所求的排序结果。

需要注意的是，如果有两个儿子的边评价字符相同，需要合并处理。可以考虑使用字典树（Trie）来解决本题。时间复杂度为 $O(n)$ ，空间复杂度为 $O(n)$ 。

结语

Thank you!