2025 ICPC 国际大学生程序设计竞赛 全国邀请赛（武汉）

SUA 程序设计竞赛命题组

2025 年 4 月 27 日

赛题设计

- 基础思维能力与编码能力：A、F、I、L；
- 经典算法与套路的应用：C、E、G、M；
- 高级算法应用与思维能力：B、D、H、J、K。

A. Problem Setting

题意

- 给定序列 $a_1, a_2, \cdots, a_n$ 与 q 条限制，第 i 条限制是 a_{p_i} 必须在 l_i 到 r_i 之间。
- 每次可以把序列里一个元素增加或减少 1，求最少几次操作可以满足所有限制，或表明无法满足。
- $n, q \leq 100$，$a_i, l_i, r_i \leq 10^9$。

- 不同元素的要求不会互相影响，因此每个元素单独处理。
- 求出每种元素取值范围的交集（是一个连续区间），若元素值已在区间内则无需操作，若元素值小于区间最小值则把它加到最小值，若元素大于区间最大值则把它减到最大值。
- 复杂度 $\mathcal{O}(n+q)$。

I. Bingo 3

题意

- 一个 $n \times n$ 的网格，里面填了 1 到 n^2 的每种数。称 k 是它的 bingo 数，若至少存在一行或一列，使得格子里的所有数都小等于 k。
- 构造这样一个网格，使得最小 bingo 数恰好是 k。
- $n \leq 50$。

I. Bingo 3

- 首先考虑无解的情况。$k < n$ 显然无解，因为都凑不满一行或一列。另外 $k > n(n-1)+1$ 也不行，因为根据鸽笼原理，涂到 $n(n-1)+1$ 时一定至少有一行或一列已经是满的。
- 剩下的情况可以这样构造：把对角线留空，剩下的位置从 1 开始填。这样只要把 k 往对角线上一放，就能填上一行。
-

1	2	3	4	
5	6	7		8
9	10		11	$\cdots$
$\cdots$		$\cdots$	$\cdots$	$\cdots$
	$\cdots$	$\cdots$	$\cdots$	$\cdots$

L. Subsequence

题意

- 称一个序列是好的，若 (最小值 + 最大值) / 2 = 中位数，求最长好子序列。
- $n \leq 3000$。

- 给序列排个序，枚举最小值 a_i 和中位数 a_j，就能算出最大值。因为序列越长越好，因此有多个元素等于最大值的情况下，选择最右边的最大值 a_k。
- 这样 a_i 到 a_j 之间的元素都可以是序列的前半部分，a_j 到 a_k 之间的元素都可以是后半部分。如果序列的长度是奇数，那么序列的最大长度就是 $\min(j-i, k-j) \times 2 + 1$；如果序列的长度是偶数，那么序列的最大长度就是 $\min(j-i+1, k-j) \times 2$。
- 复杂度 $\mathcal{O}(n^2)$。

F. Knapsack

题意

- 有 n 种物品，第 i 种有 a_i 个，每个的重量是 2^{b_i}。现在把所有物品放进 k 个承重相同的背包，问背包的最小承重。
- $n \le 2 \times 10^5$，$a_i, b_i \le 10^9$。

- 贪心。每次把最重的物品放进最轻的背包里即可。因为物品数量很多，需要用除法加速计算。
- 证明：从 10^9 到 0 考虑每个 p，考虑所有重量大等于 2^p 的物品，并把每个物品拆成重量为 2^p 的若干份。假设一共有 t_p 份物品，那么答案不会小于 $\max\limits_{0 \le p \le 10^9} \lceil \frac{t_p \times 2^p}{k} \rceil$。上述做法产生的答案恰等于这个下界。

G. Path Summing Problem

题意

- 给定 $n \times m$ 的网格，每个格子有个颜色 $a_{i,j}$。考虑所有从左上角走到右下角的路径，每次只能往右或往下走，路径的价值是它经过的不同颜色种数。求所有路径价值之和。
- $n \times m \leq 10^5$。

G. Path Summing Problem

- 考虑贡献法：格子 (i,j) 的贡献，是从 $(1,1)$ 走到 (i,j) 的过程中，没有经过颜色 $a_{i,j}$ 的路径数。如果把同种颜色的格子看成障碍，就是从 $(1,1)$ 走到 (i,j) 不经过障碍的路径数。
- 这个问题有两个做法，$\mathcal{O}(nm)$ 的 DP 做法初学 DP 时一定都学过，$\mathcal{O}(k^2)$ 的容斥做法见 CF559C Gerald and Giant Chess，其中 k 是障碍数。
- 然而如果只使用其中一种做法，极端情况下的复杂度都是 $\mathcal{O}((nm)^2)$，怎么办？
- 使用 sqrt trick，当 $k \le \sqrt{nm}$ 时使用 $\mathcal{O}(k^2)$ 的做法，否则使用 $\mathcal{O}(nm)$ 的做法。整体复杂度 $\mathcal{O}(nm\sqrt{nm})$。

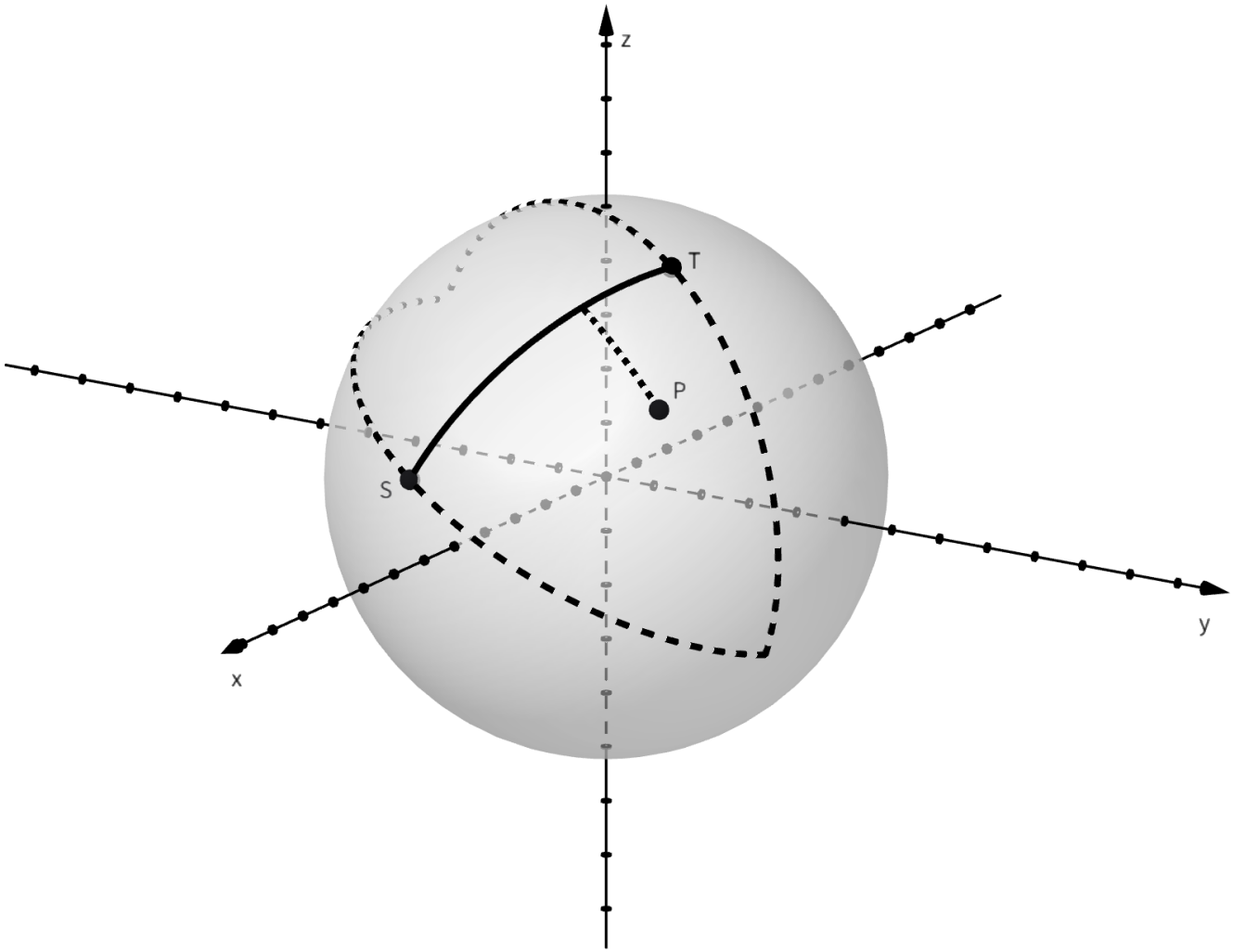
M. Flight Tracker

题意

- 给定球面上三点 P、S 和 T，计算球面上 P 到 $S \to T$ 的球面最短路的距离。球的半径 $r \leq 100$，要求相对误差不超过 10^{-4}，保证 $S \to T$ 的最短路唯一。
- 数据组数 $\leq 10^4$。

- 解法一：球面上 $S \to T$ 的最短路就是大圆劣弧 ST，点 P 到 ST 的最近点不在端点当且仅当 P 位于球面上由两个垂直于 ST 且分别过 S 和 T 的大圆半圆围成的区域内，如下图所示。

M. Flight Tracker



M. Flight Tracker

- 具体来说，记球心为 O，平面 OST 的法向量为 $\vec{n} = \overrightarrow{OS} \times \overrightarrow{OT}$，那么点 P 位于上图所示区域内当且仅当 $(\vec{n} \times \overrightarrow{OS}) \cdot \overrightarrow{OP} > 0$ 且 $(\vec{n} \times \overrightarrow{OT}) \cdot \overrightarrow{OP} < 0$。
- 此时点 P 到 ST 的最近距离就是 $\overrightarrow{OP}$ 与平面 OST 的夹角再乘以球的半径 r。
- 否则点 P 到 ST 的最近点在端点，最近距离就是 $\overrightarrow{OP}$ 和 $\overrightarrow{OS}$ 的夹角与 $\overrightarrow{OP}$ 和 $\overrightarrow{OT}$ 的夹角的最小值再乘以球的半径 r。
- 解法二：在 ST 上三分一个点 Q，计算 PQ 的距离。最近点不在端点的情况可以直接得到最优解，只需要再对最近点在端点的情况取最小值即可，注意调整三分次数以控制运行时间和答案精度。

M. Flight Tracker

- 解法三：旋转坐标轴，把劣弧 ST 转到 xOy 平面上，T 转到 x 轴上（进行三个二维平面上的旋转。先转 xOy，x 轴转到 OT 的投影；再转 xOz，x 轴转到 OT；再转 yOz，y 轴转到 OS 的投影），此时 ST 的方程就是 $y = \sqrt{r^2 - x^2}\ (t \le x \le r)$。
- 设 ST 上离 $P(x_0, y_0)$ 最近的点是 $Q(x, y)$，则根据弧长公式，距离等于 $r\theta$，其中 θ 是 $\overrightarrow{OP}$ 和 $\overrightarrow{OQ}$ 的夹角。
- 根据余弦公式，$r^2 \cos\theta = x_0 x + y_0 y = x_o x + y_0 \sqrt{r^2 - x^2}$。
- 函数在导数等于 0 以及边界时取到极值，导数 $x_0 - \frac{y_0}{\sqrt{r^2 - x^2}} x = 0$，解得 $x = \pm \frac{x_0 r}{\sqrt{x_0^2 + y_0^2}}$。
- 因此计算两个极值点的答案，再对比两个端点的答案即可。注意用 `acos` 求 θ 时，传入的 θ 要限制在 $[-1, 1]$ 里，避免因为精度问题让 $|\theta| > 1$ 导致算出 `NaN`。

C. One Must Imagine Sisyphus Happy

题意

- 西西弗斯在一片 n 格长的耕地上清理杂草，每格初始有杂草。他从 1 走到 n 再返回 1，每经过一格需 1 小时检查并清除杂草。移动时间忽略不计，一次来回耗时 $2n-1$ 小时。
- 每格在被清除后经过 a_i 小时又 1 分钟重新长出杂草（若重新长出时经过该格，会顺便清除）。
- 求前 m 次来回，西西弗斯每次清理了多少格杂草。
- $\sum n, \sum m \leq 10^6$。

C. One Must Imagine Sisyphus Happy

- 一次来回中每格杂草分别只有两个固定时间点会被清理，记一次来回是一周期，方向 $1 \to n$ 是“来”，方向 $1 \leftarrow n-1$ 是“回”。由于杂草生长时间也是固定的，每格杂草被清理的时间点有三种情况：
  1. 来 => 来 => 来 => 来 => .....；
  2. 来 => 回 => 来 => 回 => .....；
  3. 来 => 回 => 回 => 回 => .....。
- 对于第 i 格，”来 => 回”跨越的周期数是 $\lfloor \frac{a_i}{2n-1} \rfloor$，而 3 种其它方向跨越的周期数是 $\lfloor \frac{a_i}{2n-1} \rfloor + 1$。生长跨越周期数超过 m 的只会对第一个周期的答案产生贡献。
- 所以我们可以对每一种情况，统计每 i 来回一清理的格子数，它只对 $O(\frac{m}{i})$ 个答案有贡献。
- 复杂度 $O(n + m \log m)$。

E. Colorful Graph

题意

- 给定 n 个点 m 条边的无向图，给每条边涂一个颜色，使得每个点相邻的所有边中，每种颜色最多出现两次。求一个涂色方案，最小化 $\sum_{u=1}^{n} f_u$，其中 f_u 是节点 u 相邻的所有边中一共出现了几种颜色。
- $n, m \leq 2 \times 10^5$。

- 首先探求答案的下界。设点 u 的度数是 d_u，那么答案不可能小于 $\sum_{u=1}^{n} \lceil \frac{d_u}{2} \rceil$。
- 接下来构造能取到该下界的答案。

E. Colorful Graph

- 首先假设每个点的度数都是偶数，如果我们每次给一个环涂上同一种颜色，就能构造出答案。
- 度数均为偶数的图里，什么东西由很多个环组成？欧拉回路。
- 因此求出欧拉回路，然后把每个环拆出来即可。具体拆法可参考洛谷 P3520 [POI 2011] SMI-Garbage。
- 存在度数为奇数的点怎么办？每次取两个奇度数点连起来，就变回了全是偶度数点的情况。
- 复杂度 $\mathcal{O}(n+m)$。

J. Dictionary

题意

- 给定一个长度为 n 的字符串 $S = s_1 s_2 \cdots s_n$，处理 q 次操作，维护一个一开始为空的集合。每次操作把所有以 $S[l_i : r_i] = s_{l_i} s_{l_i+1} \cdots s_{r_i}$ 为开头的子串都放进集合里（集合不包含重复元素），然后询问集合的大小。
- $n, q \leq 2 \times 10^5$。

- 相同前缀的后缀在后缀数组里是排在一起的。考虑后缀数组里第 $(i-1)$ 和第 i 个后缀，后者有 `n - sa[i] + 1 - height[i]` 个前缀与前者是不同的。
- 因此我们可以维护每个后缀还有几个（与前一个后缀不同的）前缀没有放进集合。一次操作相当于把某个区间的前缀数清空，然后把区间开头的前缀数减少到特定值。这些操作可以用线段树，或并查集维护链表等方式来维护。
- 复杂度 $\mathcal{O}((n+q)\log n)$。

K. Las Vegas

题意

- 桌游《拉斯维加斯》的规则如下：有 n 个赌场，每个玩家在赌场中放入骰子，然后分别决出每个赌场的赢家。
- 决出每个赌场的赢家前，若两个或更多玩家在该赌场有相同数量的骰子，则他们的骰子全部被移除。之后，在该赌场中拥有最多骰子的玩家成为该赌场的赢家。
- 已知其它 m 名玩家在每个赌场里的骰子数，现在轮到您在每个赌场里放骰子，目标是胜利数至少和其它玩家一样多，且放入的总骰子数最少。
- $n, m \leq 50$。

K. Las Vegas

- 数据范围小，但还不到能状压或者折半搜索的地步，考虑网络流。
- 首先不知道自己需要取得几场胜利，因此先枚举一下自己胜利的场数，这样就能知道每个人至少要扣掉多少胜利，以及最多能增加多少胜利。
- 如果需要扣掉玩家 w 的 t 次胜利，从源点向 w 连容量为 t 的边。这条边必须流满。另外从源点向每个玩家连容量为 $+\infty$ 的边，表示每个玩家都能被扣掉更多胜利。
- 相应地，如果玩家 w 最多还能增加 t 次胜利，从 w 向汇点连容量为 t 的边。
- 另外自己必须取得这么多次胜利，因此自己向汇点连的边是必须流满的。

K. Las Vegas

- 每个赌场只有三种操作：要么不参加（代价为 0），要么和当前赢家平局（把胜利给别人，代价为 p_c），要么获得该赌场的胜利（代价为 q_c）。
- 考虑这样的连边方式：设赌场 c 的当前赢家为 w，第二名是 s，您是 y。
  - $w \to c$，表示玩家 w 在赌场 c 的胜利被夺取，容量为 1，代价为 0。
  - $c \to s$，表示和当前赢家平局的操作，胜利给了 s。容量为 $+\infty$，代价为 p_c。
  - $c \to y$，表示获得该赌场的胜利的操作，胜利给了自己。容量为 $+\infty$，代价为 q_c。
- 跑上下界费用流即可，可以在普通费用流里设置权值为 $-\infty$ 的边，表示必须要流的边。复杂度 $\mathcal{O}(n^2(n+m)^2)$，但是费用流的实际运行时间总是很快的。

H. WildFire, This Is for You!

题意

- 构造一组小于 10^{1500} 的正整数 x, y，使得所有坐标互质的整点中到 (x, y) 的最小曼哈顿距离恰好为 k。
- $1 \le k \le 20$。

- 指定 $x \pm i$ 和 $y \pm j$ 的最大公约数为质数 p_k。中国剩余定理合并即可。
- 卡位数的小技巧是，若某个位置 (i, j) 的最大公约数为 p_k，那么 $(i \pm np_k, j \pm mp_k)$ 的最大公约数都已经不为 1 了。可以少用很多质数填满整个表格。
- 具体实现是，在 $(0, 0)$ 位置放若干质数乘积，然后还没有被覆盖的位置再放单独的质数。
- 由于输入只有 20 种，可以在本地进行验证。

B. Black Red Tree

题意

- 有一棵 n 个点的树，一开始所有树边是黑色的。接下来依次把所有树边染成红色，每次染色后问树上有多少条简单路径恰好包含 k 条黑边，强制在线。
- $n \leq 2 \times 10^5, k \leq 10$

- 初始全黑的答案可以预处理算出来，简单 tree dp 即可
- 考虑每次染红一条树边 (u, v)，答案的 diff 是多少，显然我们只需要考虑经过了 (u, v) 的所有路径。
- 由于我们把这些路径上一条黑边染红了，对答案的改变就是经过了 (u, v) 的恰好包含 $k+1$ 条黑边的路径 - 恰好包含了 k 条黑边的路径。

B. Black Red Tree

- 考虑经过了 (u, v) 的恰好 $k+1$ 条黑边路径数量怎么算，不妨假设 u 是 v 的父亲，那么这 $k+1$ 条黑边可以拆成三部分：
  1. v 往下走 a 条黑边
  2. 黑边 (u, v)
  3. u 往上走 b 条黑边，接着往下走 c 条黑边

  其中 $a+b+c=k$
- 我们可以枚举 a 和 b，$c=k-a-b$，只需要维护一个 $f[x][a]$，表示点 x 往下走 a 条黑边的路径数量，就可以通过枚举 a, b，$O(k^2)$ 算出答案的 diff。
- 关于 f 的维护，由于 $a \le k$，只需要每次染色后，从 u 往上走 k 条黑边，依次更新经过的所有祖先的 f 数组即可。

B. Black Red Tree

- 现在有一个问题是计算答案和更新 f 的过程中，树上从 u 往上走的时候可能会经过很多红边，实际上需要遍历远超 $O(k)$ 的点，暴力走的复杂度会非常高。
- 因为树边染红之后不会再染黑，而红边可以视作合并相邻的节点，因此我们可以用并查集，每次染色 (u, v) 将两点合并，关于 $f[x][a]$ 的定义，修改为 x 所在连通块往下走 a 条黑边的路径数。每次往上跳的过程改为一个一个连通块跳。
- 总复杂度 $O(nk^2)$，空间复杂度 $O(nk)$。

D. Odd and Even

题意

- 以游程编码（run-length encoding）的方式给出一个序列 A，编码长度为 n，有 m 次询问，每次给定 r，将 A 分成 r 个不为空的连续子数组，设 p 表示和为奇数的子数组数量，q 表示和为偶数的子数组数量，分别最大化 p 和 q。
- $n, m \leq 2 \times 10^5$。

- 记 A 的前缀和奇偶性序列 $B = b_0, b_1, b_2, \cdots, b_k$，其中 $b_i = (\sum_{j=1}^{i} a_j) \bmod 2$。
- 相当于从 B 中选出 $r+1$ 个下标不同的数，并且 b_0 和 b_k 必选，其中有 p 对相邻数字不同，有 q 对相邻数字相同。

D. Odd and Even

- 记序列 B 的游程编码为 $(c_1, d_1), (c_2, d_2), \cdots (c_t, d_t)$，其中 $c_i \in \{0, 1\}, c_i \neq c_{i+1}, d_i > 0$。
- 先考虑最大化 p，观察到 $p \leq r, p \leq t-1, p \equiv t-1 \pmod 2$，只需要尽可能在不同的 (c_i, d_i) 里选数就能达到最大值。
- 再考虑最大化 q，也就是最小化 p，这需要尽量集中地在 (c_i, d_i) 里选数，直到一段全取完或者总共选到了 $r+1$ 个数。

D. Odd and Even

- 先不考虑 $p+q=r$ 的限制，而是考虑对每个 p 计算 r 的最大值，相当于最多能选多少个数，这样只要某一段 (c_i, d_i) 里有选数，整一段都应该直接选完，由于 b_0 和 b_k 必选，第 1 段和第 t 段都会被强制选完。
- 按照 p 从大到小考虑，初始 $p=t-1$ 时所有段都选了。观察到相邻两个段不会同时不选，否则可以直接选回至少一个段而不改变 p，这是一个经典的反悔贪心模型。

D. Odd and Even

- 每次找一个状态依次是“选-不选-选-不选-...-选”的连续且极长的段序列，将这些段全部取反，就能让 p 减少 2 的同时保持选出的个数最多。
- 实现上可以使用链表或者 std::set 维护这些段序列，使用堆维护这些段序列取反导致的个数减少量，每次从堆里弹出最小的减少量之后进行维护即可。

D. Odd and Even

- 观察到虽然 t 很大，但是只有 $O(n)$ 个 (c_i, d_i) 满足 $d_i > 1$，因此只有最后 $O(n)$ 次贪心的减少量会大于 1，只需要维护这 $O(n)$ 次贪心。
- 这样就在 $O(n \log n)$ 的复杂度下预处理出了每个 p 对应 r 的最大值，由于 r 的最大值关于 p 非降，每次询问对给定的 r 二分出 p 的最小值即可，总复杂度 $O((n+m)\log n)$。

Thank you!