2024 ICPC 国际大学生程序设计竞赛 全国邀请赛 (昆明)

SUA 程序设计竞赛命题组

2024 年 5 月 26 日

概况

- 阶段一（基础代码能力）：B、G、I、A
- 阶段二（经典算法理解）：E、M、J、L、F
- 阶段三（高级算法、思维能力）：H、K、C、D

B. 金牌

题意

- 给定正整数序列 $a_1, a_2, \cdots, a_n$ 以及整数 k，找出非负整数序列 $b_1, b_2, \cdots, b_n$，使得 $\sum b_i = m$，并最大化

$$\sum_{i=1}^{n} \lfloor \frac{a_i + b_i}{k} \rfloor$$

- $n \le 100$，$a_i, k, m \le 10^9$。

- 为了让 $\lfloor \frac{a_i+b_i}{k} \rfloor$ 增加 1，一开始 b_i 需要是 $(k - a_i \bmod k)$，之后 b_i 要增加 k 才能让值增加 1。
- 所以一开始先按 $(k - a_i \bmod k)$ 排序并增加 b_i（同时增加答案，减少 m）。如果还有剩下的 m，则答案还能增加 $\lfloor \frac{m}{k} \rfloor$。
- 复杂度 $\mathcal{O}(n \log n)$。

G. 乐观向上

题意

- 构造 0 到 $(n-1)$ 的排列，使得每个前缀异或和都不是 0。求字典序最小的排列。
- $n \leq 10^6$。

- $n = 1$ 和 $n \bmod 4 = 0$ 都不行，因为所有数异或起来就是 0。这启发我们每四个数看一下。
- 样例里已经展示了 $n = 3$ 的答案是 $1, 0, 2$。
- 3 不能直接放在后面，中间要用一个别的数 “垫” 一下，所以继续构造 $1, 0, 2, 4, 3, 5, 6$。
- 7 不能直接放在后面，中间要用一个别的数 “垫” 一下，所以继续构造 $1, 0, 2, 4, 3, 5, 6, 8, 7, 9, 10, \cdots$。
- 复杂度 $\mathcal{O}(n)$。

I. 左移

题意

- 给定字符串，可以将它左移任意次。问左移之后，最少要改掉几个字符，才能让相邻字符不相同。
- $n \le 5 \times 10^5$。

- 先假设整个字符串第一个字符和最后一个字符不同。
- 如果有一段长度为 l 的相同字符，那么需要从中改掉 $\lfloor \frac{l}{2} \rfloor$ 个字符。
- 接下来考虑左移操作带来的影响。左移操作可能会把一段相同字符拆成两段，如果一个偶数拆成了两个奇数，那么答案还能减小 1。
- 所以先把字符串左移到第一个字符与最后一个字符不同的位置，再检查是否能让答案减小 1 即可。
- 复杂度 $\mathcal{O}(n)$。

A. 两星级竞赛

题意

- 有 n 个竞赛，每个竞赛有 m 种属性，取值范围从 0 到 k。第 i 个竞赛评级为 s_i，得分是它的所有属性之和。
- 现在有的属性值缺失，请填充所有属性值，使得若 $s_i > s_j$，则 i 的得分严格大于 j 的得分。
- $nm \le 4 \times 10^5$，$k \le 10^9$。

- 《ACM-ICPC 国际大学生程序设计竞赛无参赛数据，相关指标按最小值估算》
- 因此（?）按评级从小到大填充属性值。每个属性值尽可能填小，给后面的评级留下更多空间。

A. 两星级竞赛

- 每个评级的属性值一起填充，需要维护每个评级最低的最大分数。设上一个评级最低的最大分数是 v，则本评级最低的最大分数是 $\max(v+1, \max \sum p)$，其中 $\sum p$ 是本评级一个竞赛已知的属性值之和。
- 由于每个属性取值范围从 0 到 k，所以一个竞赛的得分从 $\sum p$ 到 $\sum p + ke$ 中的任意值都能取到，其中 e 是缺失的属性值数量。所以只要该评级每项竞赛都满足 $\sum p + ke > v$ 即可。
- 复杂度 $\mathcal{O}(n \log n + nm)$。

E. 学而时习之

题意

- 给定正整数序列 $a_1, a_2, \cdots, a_n$ 与非负整数 k，可以执行至多一次操作：选择一个连续子树组，并把所有元素加 k。最大化整个序列的最大公因数。
- $n \le 3 \times 10^5$，$k \le 10^{18}$。

- 假设选择的区间是 $[l, r]$，那么整个序列的最大公因数就是以下四项的最大公因数

$$
\begin{gathered}
\gcd(a_1, a_2, \cdots, a_{l-1}), \\
\gcd(|a_l - a_{l+1}|, |a_{l+1} - a_{l+2}|, \cdots, |a_{r-1} - a_r|), \\
a_r + k, \\
\gcd(a_{r+1}, a_{r+2}, \cdots, a_n)
\end{gathered}
$$

- 如果 r 的值是确定的，那么后两项的值都是确定的。接下来考虑 l 的值如何影响前两项。

E. 学而时习之

- 注意到前缀 gcd 的值只有 $\log X$ 种。
- 而且因为 r 是确定的，所以 l 越大，第二项越大，且变大后是之前的倍数。
- 所以对于固定的前缀 gcd，l 越大越好。因此枚举 $\log X$ 个前缀 gcd 即将变小的地方，再枚举所有 r 即可。含计算 gcd 的复杂度，均摊后 $\mathcal{O}(n \log X)$。

M. 意大利美食

题意

- 凸包包含一个圆，选择凸包的两个顶点切一刀，使得不经过圆且不包含圆的那部分面积尽可能大。
- 顶点数 10^5。

- 枚举切割线的一个端点 A，另一个端点离它越远越好。
- 因此使用类似旋转卡壳的方式，在枚举 A 的同时维护逆时针最远的 B，使得 AB 位于圆心固定的一侧，且圆心到 AB 的距离大于等于半径。顺时针最远的 C 同理。
- 复杂度 $\mathcal{O}(n)$。

J. 冲向黄金城

题意

- 给一张无向图，每条边有长度和颜色。从节点 1 开始一共要走 k 轮，每一轮可以一次性走完颜色均为 a_i 且总长不超过 b_i 的边。问 k 轮走完以后能走到哪些点。
- 节点数、边数、$k \le 5 \times 10^5$。

- 其实就是最短路问题，但距离需要设计一下。
- 以 "走到了第 r 轮" 为第一关键字，"这一轮走了距离 d" 为第二关键字，作为最短路的距离。
- 拓展一条边时，如果这条边颜色 c 和 a_r 相同，且 d 加上边长不超过 b_r，则可以直接转移 $(r, d) \to (r, d + \text{len})$。
- 否则需要找到 r 之后的，颜色为 c 的，且 $b_r \ge \text{len}$ 的最早的一轮转移过去。
- 可以每个颜色预处理一个 rmq，然后二分查找。因为 dijkstra 算法每条边只会拓展一次，所以只会二分 $\log k$ 次。复杂度就是 $\mathcal{O}(m(\log m + \log k) + k \log k)$。

L. 漫步野径

题意

- 二维平面每个整点 (x,y) 都有连向 $(x+1,y)$ 和 $(x,y+1)$ 的两条无向路径。另外还有 n 条特殊的无向路径连接 (x_i,y_i) 和 (x_i+1,y_i+1)。
- 设 $f(x,y)$ 表示从 $(0,0)$ 走到 (x,y) 需要的最少路径数。给 p 和 q，求 $\sum_{x=p}\sum_{y=q} f(x,y)$。
- 多组数据，$\sum n \le 10^6$，$p,q \le 10^6$。

- 注意到只会往右和往上走，因为路径没有权值，绕路没意义。
- 所以 $f(x,y)$ 就是 $x+y$，减去最多经过几条斜线。

L. 漫步野径

- 把所有斜线按 x_i 升序为第一关键字，y_i 降序为第二关键字排个序，走到 (x, y) 最多经过几条斜线，就是满足 $x_i + 1 \leq x$ 且 $y_i + 1 \leq y$ 的最长上升子序列的长度。
- 算最长上升子序列的时候，我们要维护一个二分数组 f_i，表示当前 LIS 长度是 i 的最小元素是多少。对应到本题里，就是当前最多经过 i 条斜线的 y 最小是多少。所以对于固定的 x，答案就是 $\sum(q - f_i)$。随着 f 值的变化维护这个和即可。
- 复杂度 $\mathcal{O}(n \log n)$。

F. 收集硬币

题意

- 数轴上有两个机器人，每个机器人每秒最多移动 v 的距离。接下来会掉落 n 枚硬币，第 i 枚硬币在 t_i 秒掉在坐标 d_i，这个时候这个位置必须至少有一个机器人才能把它收集到。问收集所有硬币需要的最小的 v。
- $n \le 10^6$，$t_i, d_i \le 10^9$。

- 设 $f(i)$ 表示已知有一个机器人 A 第 t_i 秒在坐标 d_i，另一个机器人 B 可能在哪些位置，接下来归纳证明这些位置形成一个连续区间 $[l_i, r_i]$。

F. 收集硬币

证明

- 归纳初始条件：$l_1 = -\infty$，$r_1 = +\infty$。
- 机器人可以移动的距离是 $w = (t_{i+1} - t_i)v$。
- 若 $d_i - w \le d_{i+1} \le d_i + w$，也就是 A 可以收集到下一枚硬币，则 B 可能的范围会扩大到 $[l_i - w, r_i + w]$。
- 若 $l_i - w \le d_{i+1} \le r_i + w$，也就是 B 可以收集到下一枚硬币，则 A 可能的范围会变成 $[d_i - w, d_i + w]$。
- 若两个条件同时满足，由于两个区间均包含 d_i，所以两个区间是相交的，并起来仍然是一个区间。

- 因此二分 v，并递推 $f(n)$，检查是不是空区间即可。复杂度 $\mathcal{O}(n \log X)$。

H. 子数组

题意

- 给定长度为 n 的整数序列 $a_1, a_2, \cdots, a_n$，称一个连续子数组 $a_l, a_{l+1}, \cdots, a_r$ 是好的，若子数组里的最大元素在子数组里恰好出现了 k 次。对每个 $1 \le k \le n$ 计算好的子数组的数量。
- $n \le 4 \times 10^5$。

- 首先考虑对一个 k 求答案怎么做。
- 枚举最大值是哪种数，用单调栈可以求出这种数的每一数，在哪个区间里是最大值。这些区间显然是不相交的，因此可以每个区间分别计算答案，再把答案加起来。
- 设最大值是 v，区间内部是 { t_0 个其它数，v，t_1 个其它数，v，t_2 个其它数，…，t_p 个其它数 }，答案就是 $t_0 t_k + t_1 t_{k+1} \cdots + t_{p-k} t_p$。
- 接下来考虑对每个 k 怎么求答案。令 $t'_i = t_{p-i}$，答案就是 $t_0 t'_{p-k} + t_1 t'_{p-k-1} + \cdots$，这就是序列 t 和 t' 的卷积，FFT/NTT 算一下即可。
- 复杂度 $\mathcal{O}(n \log n)$。

K. 排列

题意

- 有一个隐藏的 1 到 n 的排列。每次可以询问一个序列（不一定是排列），得知这个序列和隐藏的排列之间有几个位置是一样的。
- $n \leq 1000$，在 6666 次询问内找出隐藏的排列。

- 考虑递归解决如下问题：已知位置区间 $[l, r]$ 中的元素恰为集合 S，求每个元素的具体位置。
- 设 $m = \lfloor \frac{l+r}{2} \rfloor$，每次从 S 里选出两个元素 x 和 y，构造询问 $a_1 = \cdots = a_m = x$，$a_{m+1} = \cdots = a_n = y$。返回值只可能是 0，1，2：
  - 如果返回值是 0 或 2，就知道了 x 和 y 具体在哪边。未知数的数量减少 2。
  - 如果返回值是 1，就知道了 x 和 y 在同一边，以后只问其中一个数即可。未知数的数量减少 1。

K. 排列

- 因此期望上，我们可以用 $\frac{2}{3}(r-l+1)$ 次询问确定 S 中所有元素在左边还是右边。之后递归处理。
- 一共会递归 $\lceil \log n \rceil$ 层，然后 $l=r$ 这一层不需要询问，所以期望的询问次数是 $\frac{2}{3}(n\log n-n)=6000$。
- 但这是期望，最差情况怎么办？
- 由于排列具有对称性（没有哪个元素是特殊的）。每次从 S 里随机抽两个数，相当于数据随机生成。本地测一下就能发现没问题。

C. 阻止城堡 2

题意

- 棋盘上有 n 个城堡（车）和 m 个障碍物。处于同一行/列，且中间没有其它城堡或障碍物的一对城堡可以互相攻击。现在要拿走 k 个障碍物，最小化能互相攻击的城堡对数。
- $n, m \leq 10^5$，坐标范围 $[1, 10^9]$。

- 拿走 k 个障碍物，等于往没有障碍物的棋盘的规定位置放 $(m-k)$ 个障碍物。
- 放一个障碍物最多可以阻止一横一竖两对城堡互相攻击，称这种放障碍物的位置为好位置。
- 把所有处于同一行，且可以互相攻击的一对城堡，看成二分图左边的点；同样地，把所有处于同一列，且可以互相攻击的一对城堡，看成二分图右边的点。这样所有好位置就连接了一个二分图左边的点，和一个二分图右边的点。

C. 阻止城堡 2

- 为了阻止尽量多对的城堡，我们要选尽可能多的好位置。求这张二分图的最大匹配即可。Dinic 算法在单位网络的复杂度是 $\mathcal{O}(m\sqrt{m})$。
- 添加完二分图对应的好位置之后，如果还有选障碍物的名额，剩下的位置最多只能阻止一对城堡互相攻击。枚举剩下的所有位置，看它能否阻止一对城堡。如果名额还剩，那么剩下的障碍物随意选择。
- 如何维护方案？可以给每一行以及每一列都维护一个 set，保存这一行/列哪些列/行有城堡/障碍物。这样就能快速将障碍物插入到 set 里，以及检查某一行/列是否有两座城堡之间没有障碍物。

D. 生成字符串

题意

- 给一个模板字符串，一个生成字符串是一个由模板串的若干段子串连接起来的字符串。
- 维护一个字符串的可重集合，支持以下操作：
  - 加入生成字符串。
  - 删除之前加入的生成字符串。
  - 给两个生成字符串 *pre* 和 *suf*，问可重集合里有多少字符串满足以 *pre* 为开头，以 *suf* 为结尾。
- 所有生成字符串的段数总和不超过 3×10^5。

D. 生成字符串

- 先考虑普通字符串（而不是生成字符串）怎么做。
- 用所有插入串和所有 *pre* 询问串建一棵字典树 T_1；用所有插入串倒过来和所有 *suf* 询问串倒过来建一棵字典树 T_2。
- 满足询问的串在 T_1 里节点一定位于 *pre* 的子树内；在 T_2 里的节点一定位于 *suf* 的子树内。子树关系可以用 dfs 序区间维护。
- 由于还有删除操作，所以还额外要求询问的操作顺序在增加和删除之间。因此这里一共有三维限制，通过 cdq 分治可以在 $\mathcal{O}(n \log^2 n)$ 的复杂度内解决三维偏序问题。

D. 生成字符串

- 接下来考虑生成字符串怎么处理。
- 因为生成字符串太长了，不能直接建字典树。但我们可以改造一下字典树，让它的每个节点保存模板串的一个区间。
- 这样就需要实现两个字典树节点的 merge 函数，实现时需要求两个子串的 lcp，所以还需要维护一个后缀数组。
- 另外不能按顺序将生成字符串加入字典树内，考虑 ab, aab, aaab, …，这种字符串按顺序加入字典树的复杂度可能达到 $\mathcal{O}(n^2)$，所以需要递归建树。
- 合并两棵字典树的复杂度是节点数之和，所以递归建树的复杂度是 $\mathcal{O}((\sum k + \sum m) \log q)$。

最后

- 没听明白？没关系。
- 访问 https://sua.ac/wiki/ ，有文字版题解与带注释的参考代码。

Thank you!