“睿琪杯”第 23 届浙江省大学生程序设计竞赛

题目分析

2026 年 4 月 25 日

H. ucup-team7610

Description

给定若干个账号名，判断它是否形如 `ucup-teamXXXX`。其中 `XXXX` 必须是至少三位、无前导零的十进制正整数。

H. ucup-team7610

Solution

直接按格式检查即可。

设固定前缀为 `ucup-team`。若字符串不以前缀开头，或者剩余长度小于 3，则不合法。否则要求剩余部分全为数字，且第一位不能为 `0`。

每个字符串长度不超过 50，逐字符判断即可。

时间复杂度 $O(|s|)$。

I. 飞行艇

Description

给定一个 $n \times m$ 矩阵。可以分别重排每一行，最大化每一列最小值之和。

I. 飞行艇

Solution

将每一行从小到大排序。令第 j 列答案上界为

$$c_j = \min_{1 \le i \le n} a_{i,j}.$$

任意方案中，把每一列的最小值也从小到大排序为 $b_1, \ldots, b_m$。对于任意一行，它至少要有 $m - j + 1$ 个数不小于 b_j，因此该行排序后的第 j 个数不小于 b_j，所以 $b_j \le c_j$。

另一方面，把所有行都按升序对齐即可让第 j 列最小值恰好为 c_j。故答案为 $\sum c_j$。

时间复杂度 $O(nm \log m)$。

L. 玛瑟尔大冒险

Description

三维空间中有若干单位立方体障碍物。角色每次先按给定向量传送，再在同一 (x, y) 柱内自由落体到最高的可站立位置。求从起点到终点的最少传送次数。

L. 玛瑟尔大冒险

Solution

可到达的位置一定是某个障碍物正上方的一格，即 $(x, y, z+1)$，再加上起点、终点。把这些位置作为图上的点。

对每个 (x, y) 维护该柱内所有障碍物的 z 坐标有序表。若当前在 (x, y, z)，使用向量 (a, b, c)，先得到目标柱 $(x+a, y+b)$ 和高度 $h = z + c$：

- 若 $(x+a, y+b, h)$ 本身是障碍物，则该传送非法；
- 否则在该柱中二分出最大的 $z_0 < h$ 且 $(x+a, y+b, z_0)$ 为障碍物，落点为 $(x+a, y+b, z_0+1)$；
- 若不存在这样的 z_0，则会一直下落，传送非法。

在上述隐式图上 BFS 即可。点数 $O(n)$，每个点枚举 m 种传送，复杂度 $O(nm \log n)$。

G. 构造

Description

给定 a, b, n, s，构造长度为 n、每项为 a 或 b 的序列，使得不存在和为 s 的连续子段。

G. 构造

Solution

先除以 $g = \gcd(a, b)$。若 s 不能被 g 整除，则任意序列都合法。之后令 $\gcd(a, b) = 1$。

若全填 a 合法，或全填 b 合法，直接输出。剩下的情况中，s 同时是 a, b 的倍数，且 $\frac{s}{a}$ 和 $\frac{s}{b}$ 均小于等于 n。

此时，如果 $s = a$ 或者 $s = b$ 则判定无解。

G. 构造

Solution

若 $a > 1$，且 $\{a, b\} \neq \{2, 1\}$，取 $k = \frac{s}{a} - 1$，循环输出块 $a^k b$。对于任意一个子段

- 如果不包含 b，其和不超过 s
- 如果包含一个 b 且和等于 s，则说明 $b + pa = qa$（p, q 均为整数），此时必然有 b 是 a 的倍数，即 $a = 1$，与前提矛盾。
- 如果包含两个 b，则必然是 $2b + ka = s = qa$，推出 $2b$ 是 a 的倍数，即 $\{a, b\} = \{2, 1\}$，与前提矛盾。
- 如果包含三个及以上的 b，则必然包含超过 $k + 1$ 个 a，此时和必然大于 s。

G. 构造

Solution

若 $a = 1, b = 2$，可以通过抽屉原理证明最长合法长度为 $s - 1$，当前情形无解。

若 $a = 1, b \geq 3$：交换 a, b 后使用第一部分的构造即可。

最后把构造出的数乘回 g 即可。

J. 签到题

Description

初始有一个 1×1 的矩阵，唯一元素为 c。接下来进行 q 次操作：可以在某一行后插入一行并将新行赋值为 y，可以在某一列后插入一列并将新列赋值为 y，也可以询问当前矩阵中 $M_{x,y}$ 的值。要求输出所有询问的答案。

$q \leq 5 \times 10^5$

J. 签到题

Solution

- 一个格子 $M_{x,y}$ 的值只会被它所在行的插入操作和所在列的插入操作影响，因此其值等于二者中插入时间较晚的那个操作的赋值。问题转化为：在任意时刻，快速找到当前第 k 行或第 k 列对应的是哪一次插入。
- 对行和列分别离线处理。设某一维最终有 N 个元素，先从后往前扫描所有操作，用树状数组维护当前仍存在的最终位置，初始 $1 \sim N$ 均为 1。若逆序遇到一次“在第 x 个元素后插入”的操作，则该新元素在当时序列中是第 $x+1$ 个元素，用树状数组上二分找到第 $x+1$ 个 1 的位置，将该最终位置分配给这次插入，并在树状数组中删去它。

J. 签到题

Solution

- 逆序结束后剩下的位置就是初始元素的位置。这样可以得到每一行、每一列的最终位置及其插入时间和值。随后正序扫描操作，再用两个树状数组分别维护当前已经出现的行和列的最终位置；插入时将预处理出的最终位置加入树状数组，查询时通过树状数组上二分找到当前第 x 行和第 y 列对应的最终位置，比较二者插入时间，输出较晚者的赋值即可。
- 本题也可以使用 Treap 来直接做。时间复杂度为 $O(q \log q)$.

F. 厌倦

Description

给定序列 a_i 和修改代价 c_i。修改一个位置后可把它改成任意值。求最小代价，使每个前缀和模 r 都不为 0。

F. 厌倦

Solution

设原前缀和为 p_i。一次修改等价于从该位置开始，把之后所有前缀和整体加上一个可任意选择的偏移量。

于是序列被若干个块分开：第一块偏移量固定为 0，之后每个块的偏移量任意。一个非第一块合法，当且仅当块内的 p_i 没有覆盖全部 r 种余数；这样总能选一个没有出现过的相反数作为偏移量。

令 dp_i 表示处理完前 i 个位置的最小代价。若最后一块从 l 开始，则转移为

$$dp_i = \min(dp_{l-1} + c_l).$$

其中 l 必须使区间 $[l, i]$ 没有覆盖全部余数。用双指针维护当前右端点 i 时最靠右的非法左端点，再用单调队列/线段树/RMQ 等数据结构维护所有合法 l 的 $dp_{l-1} + c_l$ 最小值。

第一块只需额外判断是否出现过 $p_i = 0$。总复杂度 $O(n \log n)$。

E. 这也是题？

Solution:

递归处理一个点集 S 中的所有边。

首先在 S 中贪心构造一个极大独立集 I:

- 初始 $I = \varnothing$;
- 依次考虑 $v \in S$，询问 $I \cup \{v\}$ 是否独立;
- 若回答为是，则把 v 加入 I;
- 若回答为否，则 v 与当前 I 中至少一个点有边，把 v 放入 $R = S \setminus I$。

由于 I 内部已经被询问确认为独立，所以当前递归层不需要再处理 I 内部的边。

接下来找出所有跨越 I 和 R 的边，最后递归处理 R 内部的边。

E. 这也是题？

Solution

固定 $v \in R$，我们要找它在 I 中的所有邻点。

因为 I 本身是独立集，所以对任意 $X \subseteq I$，询问

$$X \cup \{v\}$$

是否独立，等价于判断 v 是否连向 X 中的某个点。

于是可以像二分查找一样找一个邻点：

- 已知当前集合 X 中存在邻点；
- 把 X 分成两半，询问左半加上 v；
- 若非独立，邻点在左半；否则在右半；
- $O(\lceil \log_2 n \rceil)$ 次询问找到一个具体邻点。

找到一个邻点后将其从候选集合中删除，继续判断是否还有其它邻点。

E. 这也是题？

Solution:

计数方式如下：

- 每个点最终会在某一层成功加入独立集，贡献一次“加入成功”的询问，总计不超过 n 次；
- 每次加入失败，必然会在之后找到至少一条跨边，可把这次失败询问计入某条边；
- 每找到一条边，需要至多 $\lceil \log_2 n \rceil$ 次二分询问；
- 对同一个 v，每找到一个邻点后，还需要一次询问判断是否存在下一个邻点；最后一次“没有了”的询问也可以由已找到的边数均摊承担。

因此每条真实边最多贡献

$$2 + \lceil \log_2 n \rceil$$

次询问，总次数不超过题目限制。

给定一个排列 P。对每个前缀 $A = P^{(i)}$，求最多能在末尾追加多少个两两不同的实数，使得新序列 B 满足：

$$\mathrm{LIS}(B) = \mathrm{LIS}(A), \qquad \mathrm{LDS}(B) = \mathrm{LDS}(A).$$

需要输出每个前缀的答案。

M. Rounddog

Solution:

对当前前缀 A，令：

- f_i 表示长度为 i 的上升子序列的最小结尾；
- g_j 表示长度为 j 的下降子序列的最大结尾。

若当前 $\mathrm{LIS} = L$、$\mathrm{LDS} = D$，则答案为

$$\boxed{\sum_{i=1}^{L}\sum_{j=1}^{D}[f_i > g_j]}.$$

也就是说，每一对满足 $f_i > g_j$ 的 (i, j) 都对应一个可追加的“槽位”。

下面只需要在线维护 f 数组、g 数组以及这个二重和。

M. Rounddog

Solution:

考虑一个追加到末尾的新数 x。

若 $x > f_i$，那么原序列中存在长度为 i 且结尾为 f_i 的上升子序列，可以用 x 接在后面，得到更长的上升子序列。因此，若 x 处在第 i 个上升层，它必须满足 $x < f_i$。

类似地，若 $x < g_j$，那么原序列中存在长度为 j 且结尾为 g_j 的下降子序列，可以用 x 接在后面，得到更长的下降子序列。因此，若 x 处在第 j 个下降层，它必须满足 $x > g_j$。

所以一个槽位 (i, j) 能放入新数，当且仅当存在实数 x 满足

$$g_j < x < f_i,$$

也就是 $f_i > g_j$。

不同追加数不能占用同一个槽位；反过来，所有非空槽位可以按层次顺序选择不同实数并追加。因此最大可追加数量正是上述槽位数。

排列元素依次加入，设当前加入 x。

维护 f：它是严格递增数组。用标准 LIS 做法，找到第一个 $f_p > x$ 的位置，把 f_p 改为 x；若不存在，则在末尾新增 x。

维护 g：它是严格递减数组。g_j 表示长度为 j 的下降子序列最大结尾。找到最后一个满足 $g_j > x$ 的位置，设为 j，则 x 可以形成长度 $j+1$ 的下降子序列，于是更新 $g_{j+1} = x$；若 $j = D$，则新增一层。

每次更新都只会改变 f 中的一个位置和 g 中的一个位置。

记当前答案为

$$ans = \sum_i \sum_j [f_i > g_j].$$

当某个 f_p 从旧值 old 变成 x 时，只需修正这一行的贡献：

$$ans \mathrel{-}= \#\{j : g_j < old\}, \qquad ans \mathrel{+}= \#\{j : g_j < x\}.$$

当某个 g_q 从旧值 old 变成 x 时，只需修正这一列的贡献：

$$ans \mathrel{-}= \#\{i : f_i > old\}, \qquad ans \mathrel{+}= \#\{i : f_i > x\}.$$

由于 f 递增、g 递减，上述两个计数都可以二分得到。

每加入一个数只做常数次二分，单次复杂度 $O(\log n)$，总复杂度 $O(n \log n)$。

B. 拜云台

Description

给定按编号排列的无向带权边。每次询问只保留编号区间 $[l, r]$ 内的边时，所有桥边的权值和，答案按 2^{64} 取模。

B. 拜云台

Solution

本题有许多做法，下面介绍其中的一种。

对于一张图，如果我们删除他的一个任意生成树（记为 T_1），并且再在剩余的图中删除一棵任意生成树（记为 T_2），则这两个生成树的并集包含了图中所有的边双信息（即，仅在这两个生成树的并集上求桥边，所得的桥边与原图桥边等价）。

考虑离线所有询问，对于一个右端点 r，求出 $[1, r]$ 中边标号的最大生成树，和去掉这个最大生成树之后的最大生成树。则对于一个左端点 l，其对应的 T_1 和 T_2 即为这两棵最大生成树中边权大于等于 l 的边，单独拿出来使用一次 tarjan 算法即可。

B. 拜云台

Solution

在 r 移动到 $r+1$ 时，暴力在最大生成树上找到需要替换的边，并将替换的边再插入另一棵生成树，找到需要替换的边即可。

时间复杂度 $O(n \times (m+q)))$。

C. 回文串文回

Description

给定一个字符串。每次随机选择一个尚未选择的位置并改成字符 c，直到字符串成为回文串。求停止时间的期望。

C. 回文串文回

Solution:

对于每一对字符 (s_i, s_{n+1-i})，有四种可能的情况：

- $s_i = s_{n+1-i} = c$，此时是否操作 i 和 $n+1-i$ 无关紧要。
- $s_i \neq s_{n+1-i}$，且存在 i 或者 $n+1-i$ 为 c，此时必须操作另一个不为 c 的。
- $s_i \neq s_{n+1-i}$，且均不为 c，此时两者必须都被操作。
- $s_i = s_{n+1-i} \neq c$，此时两者必须同时被操作或者同时不被操作。

C. 回文串文回

Solution

对于无关紧要的点，可以忽略，只需要在最后的期望里乘一个系数即可。

对于剩下两种点：必须被操作（令有 a 个）和必须成对操作（令有 b 个），设 f_i 表示有多少种操作序列，满足其恰好包含 i 个成对的点，和所有 a 个必须要操作的点，并且不存在一个前缀也是合法的操作序列。求出 f 序列则容易求答案。

C. 回文串文回

Solution

显然有 $f_0 = a!$。对于 f_i，有 $(2 \times i + a)!$ 种方案，我们枚举其第一个合法的前缀，设为 j，则对应方案数为 $\binom{i}{j} \times f_j \times (2 \times (i-j))!$。

所以有递推式 $f_i = (2 \times i + a)! - \sum_{j=1}^{i-1} \binom{i}{j} \times f_j \times (2 \times (i-j))!$。这个形式是个半在线卷积的形式，分治 NTT 求即可。

K. 灵魂链接

Description

很多敌人，等级为 1 到 n ，等级为 i 的敌人有 a_i 只，血量为 2^{i-1} 。

每次可以消耗 m 点魔力，将一个敌人添加标记，或者消耗 1 点魔力攻击，对所有被标记的敌人造成 1 点伤害。

杀死一个敌人可以获得 m 点魔力。问杀死所有敌人最少需要多少初始魔力。

K. 灵魂链接

Description

首先，容易发现：我们一旦发动一次攻击，就会一直攻击直到至少一个敌人死亡，才会进行下一次标记操作。

结论：在最优策略下，一旦我们开始攻击敌人，此后只会在杀死一个敌人后，对至多一个敌人施加标记。

这是因为，杀死一个敌人只返还 m 点能量，因此如果我们需要对两个（或以上）敌人施加标记，那么其中一个可以在攻击前就施加，这样一定不劣。

K. 灵魂链接

Solution

那么，如果我们在杀死一个血量为 h_1 的敌人后对血量为 h_2 的敌人施加了标记，这两个敌人就等价于一个血量为 $h_1 + h_2$ 的敌人。

于是我们的策略就变成，决定一开始进行标记的次数 t，然后把所有敌人“合成”为 t 个大的敌人，然后一直攻击它们。

假设这 t 个敌人的血量从小到大排序为 $h_1, h_2, \ldots, h_t$。由于杀死一个敌人会获得 m 点能量，我们需要的初始能量将会是

$$t \times m + \max_{i=1}^{t} h_i - (i-1) \times m = m + \max_{i=1}^{t} h_i + (t-i) \times m$$

二分这个取 max 的部分（记为 x），则 $t = 1 + \lfloor \frac{x}{m} \rfloor$ 且要求 $h_i \le H_i$，其中 $H_i = x - (t-i) \times m$ 为一个等差数列。

K. 灵魂链接

Solution

问题变成，能否将所有的敌人“放入” t 个桶中，且第 i 个桶中的血量总和不超过 H_i。

一个贪心的判断方法是：按血量从高到低考虑所有敌人，对于每个敌人，随便找一个可以放的桶放进去。

这个贪心总是最优的，这是因为敌人的血量都是 2 的次幂，这意味着如果一堆血量小于 2^i 的敌人血量总和超过了 2^i，我们总是能找到一个子集使得其血量总和恰好等于 2^i。

考虑加速这个贪心的过程，我们只需要从大到小枚举等级 l，计算大小为 2^{l-1} 的“槽位”个数（即 $\sum_{i=1}^{t}\lfloor\frac{H_i}{2^{l-1}}\rfloor$，可以用类欧计算），看看能不能放下所有等级大于等于 l 的敌人即可。

单组数据时间复杂度 $O(n\log^2 S)$，其中 S 是怪物血量总和。

D. GG

Description

从 $[1, n]$ 中选择若干区间，要求所有端点两两不同，并且选出的区间中同时存在长度至少 a 的嵌套链和长度至少 b 的同向链。求方案数。

D. GG

Solution

首先不妨枚举区间数量 k，并只考虑在 $[1, 2k]$ 中选择 k 个区间的情况。只需要额外乘以一个组合数即为答案。

从 $[1, n]$ 中选择端点两两不同的区间，等价于选择一个对合 f(i.e，满足 $f^2 = I$ 的置换)。可以注意到，a 为这个对合的最长下降子序列长度的一半，b 为这个对合的最长上升子序列。

关于对合，有结论是对合的数目等于 n 个格子构成的杨表个数。具体来说，对于任意置换 f，将 f 和 f^{-1} 应用 RSK 算法插入杨表后，他们的 (插入杨表, 记录杨表) 恰好分别为 (P, Q) 和 (Q, P)。而对合有 $f^{-1} = f$，故有杨表 $P = Q$。

D. GG

Solution

更进一步的是，如果对合中含有 m 个不动点（i.e. $f_i = i$），则杨表形状中恰好含有 m 个长度为奇数的列。而从 $[1, 2k]$ 中选 k 个区间，会导致得到的对合完全不含不动点，故每一列的长度均为偶数。因此，行 $2i$ 和行 $2i+1$ 的长度一定是一致的。

根据这个观察，我们可以枚举所有大小为 $\frac{k}{2}$ 的杨表形状，将其每行复制一份之后得到一个合法的杨表形状。a 和 b 的限制等价于限制了这个新杨表的行数和列数。最后利用钩子公式求出这个杨表的填数方案即可。

时间复杂度 $O(p(\frac{n}{2}))$，其中 p 为拆分数。

A. R 司的故事

Description

给定一张连通图。每个点是一个员工，类型为 A 或 B，对应工作时长 a 或 b，且 $a < b$。

可以任意安排每个员工的上班时间 s_i，下班时间为 $s_i + a$ 或 $s_i + b$。

一个员工满意当且仅当：

- 存在邻居上班时间严格早于自己；
- 存在邻居下班时间严格晚于自己。

要求最小化不满意人数，并输出一种安排。

A. R 司的故事

Solution

考虑如果所有人的类型都相同应该怎么做。对于一张图，显然他至少有两个点不满足条件，即上班时间最早的和上班时间最晚的两个人。此外，如果图是点双联通的，则一定可以做到仅有两个人不满意：考虑求出任意一组 s, t 双极定向，则按照双极定向的拓扑序赋值时间，可以保证每个中间点都有一个相邻的更早的邻居和更晚的邻居。

对于图不是点双的情况，求出圆方树后，圆方树上的每个充当叶子节点的点双都至少会贡献一个不满意的点，而选取一个叶子节点自上而下的进行双极定向可以达到这个下界。

A. R 司的故事

Solution

考虑对于存在 A 和 B 两种人的点双，首先，我们一定可以按照之前的方法得到一个不满意人数为 2 的解。另外，容易发现答案至少为 1，并且答案为 1 当且仅当最早上班的人和最晚下班的人是同一个人，这意味着存在一个人，他的工作区间覆盖剩下所有人的工作区间。

显然，这只能在点双中恰有一个 B 节点时取到。此外，恰有一个 B 节点时，求出这个点到任意一个邻居的双极定向，按顺序安排即为一个合法的解。

A. R 司的故事

Solution

现在考虑整个圆方树上的情况。令叶子点双个数为 L，同样的，我们也可以用之前的方法得到一个不满意人数至多为 $\max(L, 2)$ 的解。

我们发现，我们仅在一种情况下可以减小不满意人数的数目，即存在叶子点双恰有一个 B 节点，并且这个 B 节点是这个点双中唯一的对外割点。对于所有这样的叶子节点，我们按照之前所述的方法将其定向为仅有 B 节点不满足，然后忽略这个叶子节点即可。

之后只需要选取任意一个剩余的叶子节点为根，并自上而下的对点双进行双极定向。

注意实现上的细节，和点双个数 ≤ 2 的情况。

谢谢！