CCPC 2023 网络赛题解

By Claris + quailty + tangjz

2023 年 8 月 20 日

A Almost Prefix Concatenation

Shortest judge solution: 1713 Bytes.

令 $f(i,j)$ 表示划分 $S_1S_2\cdots S_{i-1}$ 的所有方案的 $\binom{n}{j}$ 之和，其中 n 表示对应方案具有的段数。再令 $R(i)$ 表示最大的下标 j 满足 $S_iS_{i+1}\cdots S_j$ 是 T 的允许一次失配的前缀。根据 $\binom{n}{j}=\binom{n-1}{j-1}+\binom{n-1}{j}$，有 $f(i,j)=\sum_{R(i')\geq i-1}f(i',j-1)+f(i',j)$，可得两种类型的转移：$f(i',j)\to f(i,j)$ 或者 $f(i',j)\to f(i,j+1)$ $(i'<i\leq R(i')+1)$，均可以使用差分前缀和 $O(1)$ 完成转移。最后根据 $n^2=2\binom{n}{2}+\binom{n}{1}$，答案为 $2f(|S|,2)+f(|S|,1)$。

求出每个 $R(i)$ 等价于先求一次 LCP，跳过下一位，然后再求一次 LCP。求 LCP 可以使用二分 +Hash 或者后缀数组。

时间复杂度 $\mathcal{O}(n\log n)$，也可以做到线性。

B Palindromic Beads

Shortest judge solution: 2719 Bytes.

注意到所有颜色出现不超过两次，首先令答案为 1，接下来只需考虑所有长度至少为 2 的回文子序列。对于偶回文，它一定由若干对同色的珠子嵌套而成；对于奇回文，除了回文中心那一个珠子之外，剩下的部分也一定由若干对同色的珠子嵌套而成。对于一种出现恰好两次的颜色 c，设其对应的两个珠子的位置分别为 u 和 v，那么偶回文嵌套结构自外向内的 $dis(u,v)$ 是递减的。

将所有出现恰好两次的颜色按照 $dis(u,v)$ 从大到小排序，设 f_i 表示考虑了排序后前 i 种颜色，且第 i 种颜色的两个珠子都选择的情况下，偶回文序列的长度最大值，则 $f_i=\max\{f_j\}+2$，其中 $1\leq j<i$ 且 i 对应两点在树上的路径是 j 对应路径的子路径。最后检查一下 $dis(u,v)$ 是否不小于 2，若是的话则还可以再在路径上任取一个点作为奇回文中心，用 f_i+1 更新答案。

任取一点为根，求出这棵树的 DFS 序，记录每个点子树中 DFS 序的最小值 st 以及最大值 en。不失一般性，对于每对珠子 u,v，假设 $st_u<st_v$，有两种情况：

- 若 u 不是 v 的祖先，那么路径 (u,v) 是路径 (a,b) 的子路径当且仅当 a 在 u 的子树中，且 b 在 v 的子树中，即 $st_a\in[st_u,en_u]$ 且 $st_b\in[st_v,en_v]$。
- 若 u 是 v 的祖先，遍历 u 的所有邻居找到 u 到 v 方向的第一个点 p，那么路径 (u,v) 是路径 (a,b) 的子路径当且仅当一个点不在 p 的子树中，且另一个点在 v 的子树中，即

$st_a \in [1, st_p)$ 且 $st_b \in [st_v, en_v]$，或 $st_a \in [st_v, en_v]$ 且 $st_b \in (st_p, n]$。由于 u 互不相同，暴力找 p 的总复杂度为线性。

如果把一种颜色看成二维平面上的点 (st_u, st_v)，那么需要一个数据结构支持单点插入、矩形求最大值，可以使用线段树套线段树来实现。

时间复杂度 $\mathcal{O}(n \log^2 n)$。

C Clique Challenge

Shortest judge solution: 1166 Bytes.

枚举每个点 x 作为团中编号最小的点，那么剩下的点只能是 x 的邻居中编号大于 x 的点。如果将所有点按度数从小到大排序进行重标号，那么每个点最多往右连 $\sqrt{2m}$ 条边，这是因为如果一个点的度数超过了 $\sqrt{2m}$，那么它往右连的那些点的度数也必然超过 $\sqrt{2m}$，总度数超过了 $2m$，引出矛盾。

至此，每个点 x 往右的邻居数（即出度 out_x）被控制在较小的范围内。一张图团的数量等于补图独立集的数量。使用 Meet in the Middle 可在 $\mathcal{O}(2^{\frac{out_x}{2}})$ 的时间内求出点数为 out_x 的图的独立集的数量。

由于所有点的出度之和为 m，最坏情况下可以分成 $\frac{m}{\sqrt{2m}} = \mathcal{O}(\sqrt{m})$ 个出度为 $\sqrt{2m}$ 的点，因此时间复杂度为 $\mathcal{O}(\sqrt{m} \cdot 2^{\frac{\sqrt{2m}}{2}}) = \mathcal{O}(\sqrt{m} \cdot 1.414^{\sqrt{2m}})$。

D Discrete Fourier Transform

Shortest judge solution: 836 Bytes.

先按照题意模拟出调整前的 $F_0, F_1, \ldots, F_{n-1}$，每个 F_i 对应复平面上的一个点，模长 $|F_i|$ 是对应点到原点的距离，调整 f_k 的时候每个点会分别在某条直线上运动，每个点到原点的距离都是关于 f_k 的下凸函数，这些下凸函数的 max 仍然是下凸函数，因此可以直接三分 f_k 的值。

时间复杂度 $\mathcal{O}(n^2 + n \log A)$。

E Robot Experiment

Shortest judge solution: 600 Bytes.

按顺序依次模拟每条指令。处理每条指令时，如果目标格子尚未探测过，则对应有障碍和无障碍两个分支，最多 $\mathcal{O}(2^n)$ 种情况，对于每种情况计算最终坐标即可。

F Flying Ship Story

Shortest judge solution: 1411 Bytes.

将每个商品看作二维平面上的点，令格子 (x, y) 的权值 $f(x, y)$ 表示与 (x, y) 两维都不同的商品的价值的最大值。一开始，所有点的权值都为 0。

按照价值从大到小依次考虑每个商品。假设现在考虑到商品 (a, b, w)，它需要将除了直线 $x = a$ 以及直线 $y = b$ 之外的所有权值为 0 的格子的权值都赋值为 w。不难发现，如果这个商

品没有修改到任意一个格子，那么这个商品是无用的，可以直接删除。否则，权值为 0 的格子集合将缩小，只有以下几种可能的形态：全集、一个十字、一条横线、一条竖线、两个格子、一个格子、空集。

不同状态之间的转移显然，并且根据商品无用论，我们最多只需要保存 4 个有用的商品。每次加入新的商品时，将最多 5 个商品按价值从大到小排序，依次维护出权值为 0 的格子集合，并剔除无用商品。对于每个询问，暴力检查至多 4 个商品即可。

每次操作的时间复杂度为 $\mathcal{O}(1)$；空间复杂度为 $\mathcal{O}(1)$。

G GCD of Pattern Matching

Shortest judge solution: 795 Bytes.

令输入串包含的不同字符数为 k。因为输入保证有解，可得 $k \le m$。

如果 $m = 2$，唯一的映射方案是将字符 P_1 映射为 1，并将剩下的数（如果存在的话）映射为 0。否则，P_1 一定可以被映射为 1 或 2，在接下来的讨论中可等价地将其视为可以映射为 0。

现在开始，我们只考虑 $m > 2$ 的情况。

如果我们找到了一组映射方案，那么我们可以通过不断交换两种字符对映的映射来得到其它所有映射方案。进一步地，我们可以通过不断交换字典序上相邻的两个字符对映的映射来得到其它所有映射方案，总计 $\mathcal{O}(m)$ 种不同的基本交换。

构造出一组合法映射方案，将答案 ans 赋值为该方案对应的数字。对于每个字符 c 记录 f_c 表示如果将 c 映射为 1，并将其它所有字符映射为 0 时对应的数字。根据 $\gcd(a, b) = \gcd(a, b - a)$，交换 u 和 v 对应的映射时，数字变化量的最大公约数等于 $|f_u - f_v|$，因此对答案的贡献为 $ans = \gcd(ans, |f_u - f_v|)$。

边界情况：

- $k < m$：此时需要引入一种新字符，其 f 值为 0，表示未出现的数字。
- 初始方案对应数字的最高位应该越小越好，以保证运算过程中涉及的数值在 `long long` 范围内。

时间复杂度 $\mathcal{O}(m + |P| + k \log A)$。

H Hurricane

Shortest judge solution: 1153 Bytes.

如果直接从每个点出发跑一次单源补图最短路，时间复杂度为 $\mathcal{O}(n(n + m))$，不能接受。

观察到对于两个点 u 和 v，如果在补图上满足 $deg_u + deg_v < n$，那么一定有 $dis(u, v) \le 2$，这是因为原图中存在至少一个点与 u 和 v 均相连。考虑从所有满足 $2deg_u \ge n$ 的 u 出发跑一次单源补图最短路，这样的 u 只有 $\mathcal{O}(\frac{m}{n}) \le \mathcal{O}(\sqrt{m})$ 个，其余所有点对之间的最短路长度都不超过 2，并且当且仅当两个点之间在补图有边直接相连时，两点之间的最短路长度为 2。

时间复杂度 $\mathcal{O}(\frac{m}{n}(n + m)) = \mathcal{O}((n + m)\sqrt{m})$。

I Monster Generator

Shortest judge solution: 2719 Bytes.

对于固定的某一天，可以使用经典贪心算法求解。将怪物分为两类：

- $a \le b$ 的怪物，打完血量不会降低。
- $a > b$ 的怪物，打完血量会降低。

为了保持尽可能高的血量去挑战怪物，最优的打怪顺序是打完所有第一类怪物后再去打第二类怪物。对于第一类满足 $a \le b$ 的怪物，最优的打怪顺序是按照 a 从小到大去打，因为挑战它们所需的血量就是 a。对于第二类满足 $a > b$ 的怪物，是一个镜像的问题，将整个打怪的流程倒转后，又变成了第一类怪物的情况，因此最优的打怪顺序是按照 b 从大到小去打。使用基于比较的排序算法可以在 $\mathcal{O}(n \log n)$ 的时间内得到最优打怪顺序。

根据上述比较规则，可以发现最优打怪顺序可能发生变化的时间点只有以下三类：

- 某只怪物的 a 和 b 变得相等的日子。
- 某两只怪物的 a 变得相等的日子。
- 某两只怪物的 b 变得相等的日子。

枚举两只怪物，计算出所有 $\mathcal{O}(n^2)$ 个最优打怪顺序可能变化的时间点，将它们从小到大排序，那么对于相邻两个时间点之间的每一天，最优打怪顺序是固定的。排序得到对应的最优打怪顺序 $p_1 \to p_2 \to \cdots \to p_n$，那么所需的最少 HP 为

$$\max\{0, \sum_{j=1}^{i} a_{p_j} - \sum_{j=1}^{i-1} b_{p_j}\} \ (1 \le i \le n)$$

不难发现上式可以表示为 $n+1$ 条直线 $y = kx + b$，其中 x 表示日期，y 表示所需 HP，答案为每个整数 x 对应的最大的 y，求出凸壳后分段求和即可。

时间复杂度 $\mathcal{O}(n^3 \log n)$。

J Find the Gap

Shortest judge solution: 1052 Bytes.

首先特判单点、所有点共线、所有点共面等特殊情况，这些情况的答案都是 0。

对于一般情况，在最优情况下两个平面要么分别卡住两个点，要么一个卡住一个点、另一个卡住三个点。预处理出 $\mathcal{O}(n^2)$ 个点对之间的向量，然后枚举两个非共线的向量 $\vec{A}, \vec{B}$，叉乘得到平面的法向量 $\vec{C} = \vec{A} \times \vec{B}$，找到在该法向量上投影最大与最小的两点，将投影作差即是两平面之间的距离。

时间复杂度 $\mathcal{O}(n^5)$。

K Sequence Shift

Shortest judge solution: 1126 Bytes.

取 $lim =$ 序列 a 的第 k 大值 + 序列 b（包括后来 q 个数）的第 k 大值，令 ans_i 表示第 i 个操作的答案，一开始所有 $q+1$ 个 ans 值都为 0。

每次在序列 b 末尾添加新的数 v 时，按值从大到小遍历 a 中所有满足 $a_i + v \geq lim$ 的数，更新对应的 ans 值。如果现在要输出的答案至少为 lim，那么对应的 ans 值非 0，直接输出即可，否则暴力 $\mathcal{O}(n)$ 计算当前的答案。

如果 k 选得太小，那么暴力计算的次数太多；如果 k 选得太大，那么满足条件的 $a_i + v \geq lim$ 的数对又会太多。需要选取合适的 k 使得两者的期望都在合理的范围内。为了方便推导，令 $m = n + q$，不妨等价地认为所有数在 $[0, 1]$ 之间均匀随机，且求 max 变为求 min。此时 $lim \approx \frac{k}{n} + \frac{k}{m}$。

首先计算 $a_i + b_j \leq lim$ 的数对的期望数量 E：

$$
\begin{aligned}
E &= \sum_{i=1}^{n}\sum_{j=1}^{m}[a_i + b_j \leq lim] \\
&\approx \sum_{i=1}^{n}\sum_{j=1}^{m}[\frac{i}{n} + \frac{j}{m} \leq \frac{k}{n} + \frac{k}{m}] \\
&\approx \sum_{i=1}^{(1+\frac{n}{m})k} \frac{(k-i)m}{n} + k \\
&\approx \frac{\left(1+\frac{n}{m}\right)k\left(\frac{km}{n}+k\right)}{2} \\
&= \frac{\left(1+\frac{n}{m}\right)\left(\frac{m}{n}+1\right)k^2}{2} \\
&= \frac{k^2(n+m)^2}{2nm}
\end{aligned}
$$

那么一对数满足 $a_i + b_j \leq lim$ 的概率为 $P = \frac{E}{nm}$。由于计算一个答案需要考虑 n 对数，因此它们都不满足 $a_i + b_j \leq lim$ 的概率为 $(1-P)^n$，每次需要 $\mathcal{O}(n)$ 时间暴力计算一个答案，共 $q + 1 = m - n + 1$ 次，这部分期望时间复杂度为 $\mathcal{O}\left((m-n+1)\,n\,(1-P)^n\right)$。

总时间复杂度 T 为：

$$
\begin{aligned}
T &= E + (m-n+1)\,n\,(1-P)^n \\
&= E + (m-n+1)\,n\left(1-\frac{E}{nm}\right)^n
\end{aligned}
$$

为了最小化 T，考虑求 T 关于 E 的导数：

$$\begin{aligned}
1+(m-n+1)\,n^2\left(1-\frac{E}{nm}\right)^{n-1}\frac{-1}{nm} &= 0\\
1 &= (m-n+1)\,n^2\left(1-\frac{E}{nm}\right)^{n-1}\frac{1}{nm}\\
nm &= (m-n+1)\,n^2\left(1-\frac{E}{nm}\right)^{n-1}\\
\frac{m}{n\,(m-n+1)} &= \left(1-\frac{E}{nm}\right)^{n-1}\\
\left(\frac{m}{n\,(m-n+1)}\right)^{\frac{1}{n-1}} &= 1-\frac{E}{nm}\\
\frac{E}{nm} &= 1-\left(\frac{m}{n\,(m-n+1)}\right)^{\frac{1}{n-1}}
\end{aligned}$$

由此可以解出最优的 E，继而解出最优的 k。

因此 $T = nm\left(1-\left(\frac{m}{n(m-n+1)}\right)^{\frac{1}{n-1}}\right)+(m-n+1)n\left(\frac{m}{n(m-n+1)}\right)^{\frac{n}{n-1}}$。

令 $q=tn, m=n+q=(t+1)n$，取 $\frac{1}{n}\to 0$（即 $n\to\infty$）处的皮瑟级数展开，有

$$\begin{aligned}
E &= (t+1)n\log\frac{tn}{t+1}+o(1)\\
&= \mathcal{O}\left(m\log\frac{nq}{m}\right)\\
&= \mathcal{O}(m\log m)\\
T-E &= \frac{(t+1)(tn+1)}{t}+o(1)\\
&= \mathcal{O}\left(\frac{m(q+1)}{q}\right)\\
&= \mathcal{O}(m)
\end{aligned}$$

可得期望时间复杂度为 $\mathcal{O}(m\log m)$，而 $E\approx\frac{k^2(n+m)^2}{2nm}$，$k$ 的近似值为 $\frac{m}{n+m}\sqrt{2n\log\frac{nq}{m}}$。

注意到 E 的实际取值受 k 影响是一些离散值，但相邻数之差是 $\mathcal{O}(m)$ 级别，对复杂度影响不大，可以按预期的 E 去二分一个合适的 k。

L Partially Free Meal

Shortest judge solution: 1434 Bytes.

首先考虑如何计算固定的 k 的答案。将所有盘子按照 b 从小到大排序，枚举第 x ($k\le x\le n$) 个盘子作为选中的 b 最大的盘子，那么剩下的 $k-1$ 个盘子显然是贪心选择前 $x-1$ 个盘子中 a 最小的 $k-1$ 个。给定 k 和 x，可以通过可持久线段树在 $\mathcal{O}(\log n)$ 的时间内求出对应方案的值 $w(k,x)$。

令 $f(k)$ 表示使 k 取到最优解的 x。对于两个不同的决策 x,y ($x<y$)，若 $w(k,x)\ge w(k,y)$，那么增大 k 之后由于 y 的可选择范围严格包含了 x 的可选择范围，因此 y 新选的 a 值一定不大于 x 所选的，即 $w(k',x)\ge w(k',y)$ 对于 $k\le k'\le n$ 恒成立。由此可得 $f(1)\le f(2)\le f(3)\le\cdots\le f(n)$，最优决策具有单调性，可以分治求解，共需计算 $\mathcal{O}(n\log n)$ 个 $w(k,x)$ 的值。

时间复杂度 $\mathcal{O}(n\log^2 n)$。