牛客竞赛
AC.NOWCODER.COM

2024 牛客
暑期多校训练营

Nemesis ++

HR.NOWCODER.COM

A - Surrender to My Will

题目大意

- 基于英雄联盟的投票系统中四票赞成通过投降的规则，判定一个给定的票型是否能确定投降结果。

A - Surrender to My Will

- 四个 `Y`: 成功;
- 两个 `N`: 失败;
- 否则不确定。

B - pair<pair<int,int>,int>[1]

题目大意

- 给定基于 double 和 int 的 pair 嵌套的变量声明，回答查询某个变量的成员访问。
- $n, q \leq 1000$, 每行字符串长度不超过 5000.

[1]因为牛客会把 <> 当做 html 标签，标题更改为 “std::pair”。

B - pair<pair<int,int>,int>

- 任何线性于一行最大字符数量回答询问的方法都可以接受，包括
  - 建出表达式的二叉树，在表达式树上找到对应节点后 dfs 遍历输出类型；
  - 每次查询对类型串扫描，当与要查询的位置匹配上时，输出对应字符串。
- 一个比较天才的想法是：使用 Python 的 tuple 类型，对输入和查询串进行字符串替换：
  - `pair<` → `(`, `>` → `)`, `int` → `1`, `double` → `2`
  - `.first` → `[0]`, `.second` → `[1]`

  然后使用 `eval()` 实现类型转换和查询。但由于 CPython 解释器对 tuple 有深度限制，Pypy 跑这个任务比较慢，所以很遗憾不太能通过。

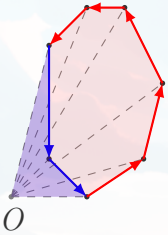
C - Capability Expectation

题目大意

- n 个二维平面上的点，每个点有 p_i 概率消失，概率独立。
- 问凸包的期望面积。答案要求精度 10^{-6}.
- $\sum n \leq 2000$, $|x_i|, |y_i| \leq 10^9$, 保证不存在三点共线。

C - Capability Expectation

- 凸包的面积可以用叉积计算线积分。线积分可以直观地理解为：从任意点出发，计算每条逆时针有向边和其构成的有向三角形的面积和。



C - Capability Expectation

- 一条有向边在凸包上出现时贡献固定为有向三角形面积，我们只需要计算出每条有向边在凸包上的概率。两个点构成的有向边有贡献当且仅当
  - 两个顶点出现了，且
  - 其右侧所有点都没有出现。
- 枚举有向边第一个点，以它为参考对所有其他点极角排序，扫描线维护线段右侧均不出现的概率乘积即可。
- 为了使得一定出现的点（对连乘贡献 0）以及浮点数连乘精度正确，可能的实现方式是维护 0 的个数和取 $\log$ 后的其他权值的和。
- 时间复杂度 $O(n^2 \log n)$.

D - Is it rated?

题目大意

- 给定每场比赛的 rating 更新方式为

$$r \leftarrow k \cdot p + (1 - k) \cdot r,$$

  其中 p 为比赛的表现分。
- 计算从某一个初始 rating 出发，在 n 场比赛里至多让 m 场不计分 (unrated) 的情况下，最后 rating 的最大值。
- $1 \leq m \leq n \leq 10^5$, $0.1 \leq k \leq 1$.

D - Is it rated?

- 注意到 $0.9^{200} \approx 7.055 \times 10^{-10} < 10^{-9}$, 这意味着在进行了两百场 rated 比赛后，初始 rating 的影响会在误差范围内消失。
- 当 $n \geq m + 200$ 时，最后 $m + 200$ 场里至少要选 200 场比赛算分，而在这之前的比赛都已无所谓，因此可以视为全选。
- 结合 $n < m + 200$ 的情况，令 $d = \min(200, n - m)$, 则在最后 $m + d$ 场里选择至少 d 场计分，使用 $f_{i,j}$ 表示前 i 场比赛里选择 j 场的最高分，答案为 $f_{m+d,d}$.
- 时间复杂度 $O(n \cdot \log_{1-k} \epsilon)$, 其中 $\log_{1-k} \epsilon \leq 197$.

E - Cherry on Top[2]

题目大意

- 敌人血量为 h，我方使用有 3 种花色的琼玉牌进行回合制战斗。
- 战斗开始时手牌为空，我方每回合依次进行如下行动：
  1. 抽 2 张牌。如果已经持有 3 张牌，则只会抽 1 张。
  2. 攻击敌人。如果持有四张相同花色，那么就会把它们全部弃掉，并造成 a 点伤害；否则，弃掉（本花色数量，编号）最小的牌，并造成 b 点伤害。

[2]“Cherry on Top” 的意思是 “杠上开花”。

E - Cherry on Top

- 每场战斗都会有一个 **循环的** 抽牌堆，抽牌时按照这个顺序抽取。
- 服务器中有一个长为 n 的花色序列，每场战斗选一个区间作为抽牌堆。
- 有 m 次操作，分为两种：
  - 修改：修改花色序列中的某个位置。
  - 查询：询问选择某个区间作为抽牌堆时，需要多少回合才能击败敌人。
- $1 \leq n, m \leq 10^5,\ 1 \leq h \leq 10^{13},\ 1 \leq a, b \leq 10^9.$

E - Cherry on Top

- 每回合都要弃牌，所以回合结束时最多有 3 张手牌，状态数最多只有 $\binom{6}{3} = 20$ 种。
- 考虑到抽一张或者抽两张都有可能，可以再加一维 $0/1$ 表示是否需要再抽一张牌，那么每次抽牌之后就最多有 40 种状态。
- 根据鸽巢原理，循环节最多只有抽牌堆大小的 40 倍，所以询问时只要找到循环节就都好说了。

E - Cherry on Top

- 因为需要支持单点修改、区间询问，可以用线段树维护。
- 对每个区间，维护以每种状态进入后，出来时的状态，以及在区间内打出 a 和 b 的次数。
- 两个区间合并时复杂度是 40，所以总的复杂度就是 $O(40(n+m)\log n)$。
- 为了减小常数，可以预处理 40×40 的转移表，而不是每次都模拟一遍。
  - 实际上这个 40 的常数也是可以优化的，因为并非所有状态都可达。
  - 例如 $\{1,2,3\}$ 或者 $\{1,1,2\}$ 并且不需要再抽一张的情况就是不可达的。

F - Collinear Exception

题目大意

- 给定一个 $\{1 \dots n\} \times \{1 \dots n\}$ 的二维平面点的排列，从前到后依次插入一个初始为空集合。如果插入后会导致三点共线，那么会跳过这次插入。
- 输出每次插入是否成功。
- $n \leq 1000$.

F - Collinear Exception

- 每行每列至多只有 2 个点，因此至多插入 $2n$ 个点[3]。
- 如果一个点插入时会导致三点共线，我们可以找到两个点使得经过这两个点的直线穿过当前点。
- 将某个点判断不能被插入提前到每次成功插入点时：维护被经过已有点的直线覆盖的点是哪些，在成功插入点时去更新这个集合。

[3]题外话：一定能找到 $2n$ 个点吗？答案是 $n \leq 46$ Yes, $n > 46$ Unknown. (Reference)

F - Collinear Exception

- 插入点时，一种可能的更新方式是枚举已有的点，去标记直线上的点。
- 看起来单次操作是 $O(n^2)$ 的，实际上如果是按照斜率跳跃的话会更快。
- 考虑将直线的斜率表示为 $(\Delta x, \Delta y)$, 其中 $\gcd(\Delta x, \Delta y) = 1$, $\Delta x > 0$ 或 $\Delta y = 1$. 由于不存在三点共线，因此斜率互不相同。给定斜率集合 S, 标记点数

$$P = \sum_{(\Delta x, \Delta y) \in S} \left\lfloor \frac{n}{\max(|\Delta x|, |\Delta y|)} \right\rfloor.$$

- $n = 1000, |S| = 2000$ 时，只限制 S 内互不相同的最坏情况为 $P = 97784$.
- 总复杂度为 $O(nP)$.

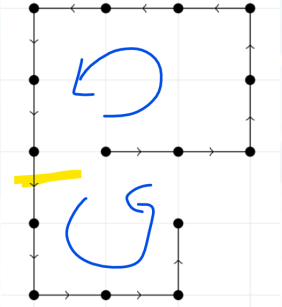
G - TNT Run

题目大意

- 给定网格图上 n 个整点。
- 选择一个点开始，每次要么直走一步，要么左转 90 度。
- 不能连续左转，不能经过一个点多次。
- 要求经过且只经过所有给定点，构造方案。
- $2 \leq n \leq 10^5$。

G - TNT Run

- 可以证明移动路线一定是：
  - 两个逆时针螺旋拼起来，其中一个向外，一个向内；
  - 且一定存在某条水平或竖直的直线可以把图形切成这两个螺旋。



G - TNT Run

- 枚举在边界处切断路径的直线，从边界出发分别检查两侧是否都是螺旋。
- 为了快速检测螺旋，记录每个点最多向四个方向行走多远，就可以在直线上进行快速移动。
- 螺旋从内到外线段越来越长，$n = 1 + 2 + \ldots + O(\sqrt{n})$，因此转弯次数是 $O(\sqrt{n})$ 的，直接模拟此过程即可。
- 总复杂度 $O(n\sqrt{n})$ 。

H - All-in at the Pre-flop

题目大意

- 两个玩家分别拥有 a 和 b 的筹码，在一场公平的游戏里互相全押。当赢家筹码不少于输家时，游戏立即结束；否则输家向赢家支付等同于赢家筹码的筹码。
- 问两个玩家的胜率 $\bmod 998244353$.
- $1 \leq a, b, a + b < 998244353$.

H - All-in at the Pre-flop

- 输出 $\frac{a}{a+b}, \frac{b}{a+b}$. 为什么?

H - All-in at the Pre-flop

- 输出 $\frac{a}{a+b}, \frac{b}{a+b}$. 为什么?
- ~~因为数值模拟打个表出来就是这样的。~~
- 一个玩家的胜率只和其筹码量有关。令 $f(x)$ 表示拥有 x 筹码的玩家在总筹码固定为 n 的游戏里的胜率。
  - $f(0) = 0, f(n) = 1.$
  - 当 $x < n/2$ 时，$f(x) = \frac{1}{2} \cdot f(0) + \frac{1}{2} \cdot f(2x)$;
  - 当 $x \geq n/2$ 时，$f(x) = \frac{1}{2} \cdot f(n) + \frac{1}{2} \cdot f(2x - n)$.
- 可以验证 $f(x) = \frac{x}{n}$ 是一组解。
- 同时，由于线性方程满秩，亦是唯一解。

I - Riffle Shuffle

题目大意

- 给定一个 n 张牌的排列，使用 Riffle Shuffle 将 $1 \dots n$ 洗成给定排列。
- Riffle Shuffle 的定义是：将牌任意切为两叠，以任意顺序归并。
- 最小化洗牌次数并构造方案。
- $2 \leq n \leq 52, \sum n \leq 2 \times 10^5$.

I - Riffle Shuffle

- 倒着考虑这个过程：给定一个排列，使用以下操作进行排序。
  - 选择一个子序列，将这个子序列按顺序放在牌堆顶，其余牌顺序不变。
- 如果有最后一步，选择的这个排列一定是形如 $1 \dots i$, 没选择的部分一定是 $i+1 \dots n$. 也就是说，一步能到的排列一定能拆成这两个连续子序列。
- 如果有倒数第二步，在操作后的排列一定要能拆成这两个连续子序列。
- 基于以上观察，我们给出归纳结论。

I - Riffle Shuffle

Definition (上升序列)

一个排列中的子序列被称作上升序列，当其元素为按顺序排列的连续自然数。

Lemma (下界)

如果排列 p 能被拆分为 r 个极长上升序列，至少要用 $\lceil \log_2 r \rceil$ 次 *Riffle Shuffle* 才能从 $1 \dots n$ 洗出 p。

Lemma (上界)

如果排列 p 能被拆分为 r 个极长上升序列，那么能用 $\lceil \log_2 r \rceil$ 次 *Riffle Shuffle* 从 $1 \dots n$ 洗出 p。

I - Riffle Shuffle

证明：下界.

考虑一次 Riffle Shuffle 对排列中的一个上升序列带来的影响：其在第一叠和第二叠的部分分别构成上升序列，极长上升子序列数量至多翻倍。 □

证明：上界.

考虑从目标排列倒着做，合并上升序列：选择出第 $1, 3, 5 \dots$ 个上升序列的元素放在牌堆顶，这样第 $(1, 2), (3, 4), (5, 6) \dots$ 个上升序列就合并在一起了，极长上升序列数量除二上取整。 □

- 利用上界证明的构造模拟即可。时间复杂度 $O(n \log n)$.

I - Riffle Shuffle

- 举例：对于排列 $[8,3,1,4,2,6,5,7]$，倒数第一步：
  - 其上升序列为 $[1,2],[3,4],[5],[6,7],[8]$;
  - 将第 $1,3,5$ 个上升序列取出来：$8,3,1,4,2,6,5,7$;
  - 进行 Riffle Shuffle 的逆操作：$8,1,2,5,3,4,6,7$
- 倒数第二步：
  - 其上升序列为 $[1,2,3,4],[5,6,7],[8]$;
  - 将第 $1,3$ 个上升序列取出来：$8,1,2,5,3,4,6,7$
  - 进行 Riffle Shuffle 的逆操作：$8,1,2,3,4,5,6,7$
- 倒数第三步把 8 放到最后即可。

J - Doremy's Starch Trees

题目大意

- 给定无根树 X，有根树 Y（但根未知）。
- 判定 Y 是否可能是 X 的淀粉树（点分树）。
- $1 \leq n \leq 10^6$。

J - Doremy's Starch Trees

- 如果给定根如何判定是否是淀粉树?
- 事实上只需要按照淀粉树的构造方法判定即可。
- 在 T_2 上 DFS，回溯时检查 x 在 T_1 上的所有已经被 DFS 到过的邻居 y 是否可以通过 T_2 已经回溯过的部分与 x 连通，使用并查集维护即可。
- 于是目标转化为寻找根。

J - Doremy's Starch Trees

Definition

如果一条边同时在两棵树中出现，称其为重边。

Lemma (1)

若 x 是合法的根，且有重边 (x, y)，那么 y 也是合法的根。

Lemma (2)

若 x 是合法的根，那么其在两棵树上的度数相同。

J - Doremy's Starch Trees

Lemma (3)

把重边相连的点缩成一个点，并且与它们相关的边也对应地连到同一个点上（下称缩点）之后，所有合法的根会被缩成同一个点；不合法的根不会被缩到这个点里。

Lemma (4)

设一个点在两棵树中度数相同，那么该点是合法的根等价于，**缩点后**任意（Y 上的）两条边都来自不同的（X 上的）子树，且与它缩到一起的其他点的度数也各自在两棵树上相同。

J - Doremy's Starch Trees

- 先承认以上 Lemma，我们可以给出如下算法。
- 用 Lemma 4 判定是否有合法根，如果没有，一定不存在合法根。若有，则任选其一为根。
- 判定一个点是否可能是合法根，需要求 $O(\text{该点度数})$ 次 $x \to y$ 路径上倒数第二个点是谁，可以离线下来 $O(n)$ 解决。
- 验题人则给出了多种不同做法，包括换根、线段树合并等等，由于时限比较松弛，$O(n \log n)$ 做法也可通过。

K - Doremy's IQ 2

题目大意

- 给定数轴上 n 个整点，你初始在原点。
- 每次你选择一个点，向它的方向移动一格；但如果选择的点与你当前重合，则加一分。选择后该点消失。
- 最大化分数。
- $1 \leq n \leq 10^5$，点的位置 $[-n, n]$，可以重复。

K - Doremy's IQ 2

- 省流，算法如下：
  - 最优方案中移动的方向最多改变一次。若到达过的区间是 $[l, r]$，那么存在一种最优方案使得每次移动所用的都在 (l, r) 外。
  - 考虑转向一次的最优方案。枚举先左还是先右，然后枚举左右最远到达何处，用前缀和检查是否可行，如果可行，计算相应的答案。
  - 然后考虑只向一个方向移动的最优方案。此时，如果向左移动，那么所有 > 0 的点都没有意义；反之亦然。类似转向一次的做法求解即可。
- 直接暴力计算复杂度是 $O(n^2)$ 的，但是随着 l 减小，r 也是非严格减小的，所以可以用双指针 / 二分优化到 $O(n)$ / $O(n \log n)$。

K - Doremy's IQ 2

- 首先排除掉站在原地不动的情况。然后改写一下题面，允许在任何时间停止，这不会改变答案。

Lemma (1)

最优方案中 *IQ* 移动的方向最多改变一次。

- 证明：如有一种移动方式方向改变超过一次，我们找到 IQ 移动到的最左边 l 和最右边 r。如果 l 比 r 先到达，那么可以把移动方案调整成一步一步走到 l 然后到 r；反之亦然。

K - Doremy's IQ 2

Lemma (2)

在所有方向改变一次的最优方案中，若 *IQ* 到达过的区间是 $[l, r]$，必然存在一种使得 l 和 r 处都存在得分。

- 证明：若 l 处不存在得分，则没有必要走到这个位置；对 r 同理。

K - Doremy's IQ 2

Lemma (3)

若 *IQ* 到达过的区间是 $[l, r]$，那么存在一种方向改变一次的最优方案使得每次移动所用的都在 (l, r) 外。

- 证明：假设有某个在 x （$l < x < r$）的点被用于移动，那么不如改用 l 处或 r 处的点移动，那么 x 这个点可以得分，得分总数不变。由 Lemma (2) 知这样的点一定存在。

L - Tada!

题目大意

- 一个 n 位 $0 \sim 9$ 的转轮密码锁，拨动方式是每次选择一个区间，顺时针或者逆时针转动一位。
- 从某个特定的密码出发，给定 m 条转动了 t 次到达了状态 s 的记录，问密码是否有解，如果有唯一解输出唯一解。
- $n \leq 5, m \leq 50, \sum 10^n \leq 3 \times 10^5$.

L - Tada!

- 从 00000 出发, 我们可以知道到每个状态需要几步。
- 由于操作可逆，将每次的终态 s 看做 00000, 我们可以知道哪些位置能在 t 步内到达 s:
  - 首先最短路一定不能超过 t;
  - 如果 x 步能到，$x+2$ 步一定能到，因为可以来回旋转相同的格子来浪费时间。注意 $x+1$ 不一定能到：例如 $t=1$ 时最短路为 0 的情况，以及 $n=1$ 的情况。

  因此距离的奇偶分开维护最短路，用最短路求出可行答案集合，对 m 个大小至多为 10^n 的集合求交即可。
- 最短路复杂度 $O(10^n n^2)$, 集合求交复杂度 $O(10^n m)$.

M - Patchouli and Magic Formations

题目大意

- 有 m 种不同的结点，结点之间可以通过有向边连接。
- 每个结点最多拥有一个前驱，第 i 种结点最多有 a_i 个 **不同** 的后继栏位。
- 每个结点的第一个后继栏位是特殊的 – 这个栏位必须非空，除非这个结点没有后继。
- 问 $1 \dots n$ 个结点分别可以连成多少个本质不同的（无标号）图形，$\bmod 998244353$. 可以有环，也可以不连通。
- $1 \leq n, m, a_i \leq 10^5$.

M - Patchouli and Magic Formations

- 因为是无标号计数，所以直接求可以不连通的方案数是行不通的。
- 考虑先求连通的方案数，最后再转换成答案。
- 每个结点至多有一个前驱，所以每个连通块要么是一个基环树，要么是一棵有根树。

M - Patchouli and Magic Formations

- 先求有根树的方案数。
- 设有根树的 OGF 是 $F(x)$，方便起见令其没有常数项。
- 设 S 是 a_i 的可重集，可以写出方程:

$$F(x) = x \sum_{k \in S} \left(1 + F(x)\left(1 + F(x)\right)^{k-1}\right)$$

- 含义即为枚举根结点种类后，要么没有后继，要么第一个栏位非空，其它栏位随便。

M - Patchouli and Magic Formations

- 移项得

$$\frac{F(x)}{\sum_{k\in S}\left(1+F(x)\left(1+F(x)\right)^{k-1}\right)}=x$$

- 左边只与 $F(x)$ 有关，所以可以写出 $F(x)$ 的复合逆

$$F^{(-1)}(x)=\frac{x}{\sum_{k\in S}\left(1+x\left(1+x\right)^{k-1}\right)}$$

- 因为后面需要用到前 $(n+1)$ 项的所有系数，所以需要跑一个 $O(n\log^2 n)$ 的多项式复合逆。

M - Patchouli and Magic Formations

- 接下来考虑基环树的方案数。
- 一个基环树必定是若干个有根树排成一个有向环。
- 因为是无标号计数，所以需要用 Polya 定理求出旋转同构的方案数。

M - Patchouli and Magic Formations

- 需要注意的是，因为环上的连边有可能是特殊栏位，需要分情况考虑。
- 设环上的一棵树的 OGF 是 $G(x)$，方便起见也没有常数项，那么有：

$$\frac{G(x)}{x} = \left(\sum_{k\in S} (1+F(x))^{k-1}\right) + F(x)\sum_{k\in S}[k>1](k-1)\,(1+F(x))^{k-2}$$

- 因为 S 是输入给出的，只能跑一个 $O(n\log^2 n)$ 的多项式复合。
- 求完 $F(x)$ 和 $G(x)$，现在可以求一个连通块的方案数了。

M - Patchouli and Magic Formations

- 设一个连通块的 OGF 是 $H(x)$，枚举环长 i 和旋转的位数 j:

$$
\begin{aligned}
H(x) =& F(x) + \sum_{i\geq 2} \frac{1}{i} \sum_{j=1}^{i} G\left(x^{\frac{i}{\gcd(i,j)}}\right)^{\gcd(i,j)} \\
=& F(x) - G(x) + \sum_{i\geq 1} \frac{1}{i} \sum_{j=1}^{i} G\left(x^{\frac{i}{\gcd(i,j)}}\right)^{\gcd(i,j)} \\
=& F(x) - G(x) + \sum_{i\geq 1} \frac{1}{i} \sum_{j|i} \varphi\left(\frac{i}{j}\right) G\left(x^{\frac{i}{j}}\right)^{j}
\end{aligned}
$$

M - Patchouli and Magic Formations

$$
\begin{aligned}
H(x) =& F(x) - G(x) + \sum_{i\geq 1} \frac{1}{i} \sum_{j|i} \varphi\left(\frac{i}{j}\right) G\left(x^{\frac{i}{j}}\right)^j \\
=& F(x) - G(x) + \sum_{i\geq 1} \frac{\varphi(i)}{i} \sum_{j\geq 1} \frac{G\left(x^i\right)^j}{j} \\
=& F(x) - G(x) - \sum_{i\geq 1} \frac{\varphi(i)}{i} \ln\left(1 - G(x^i)\right)
\end{aligned}
$$

- 后面的和式只需要求一次 $\ln(1 - G(x))$，然后枚举倍数就行了。

M - Patchouli and Magic Formations

- 最后考虑如何转换成可以不连通的方案数。
- 因为是无标号计数，所以不能直接 $\exp$，而是要做 Euler 变换。
- 设答案的 OGF 是 $A(x)$，那么有：

$$A(x) = \exp\left(\sum_{i\geq 1} \frac{H(x^i)}{i}\right)$$

- 至此我们就得到了答案，总复杂度 $O(n\log^2 n)$.
- 难点主要在于过程中用到的 $O(n\log^2 n)$ 多项式复合逆和多项式复合。

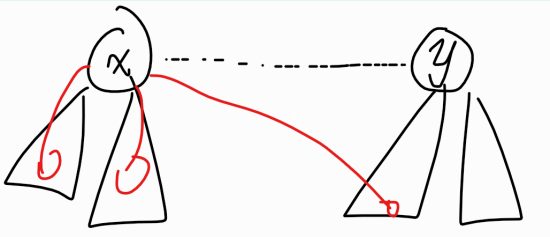
Omitted proofs for Problem J

Proof of Lemma (3).

- 引理的后半部分直接由引理 1 可得。
- 前半部分考虑反证法。假设存在两个点都是合法的根且它们不能缩成同一个点，那我们找到在 X 上最近的这样的点对 x 和 y。由于最近，那么 x 到 y 的路径上不存在其他点是合法的根。
- 然后在 Y 上考虑 $x \to y$ 。
- Case 1: x, y 在 X 上直接相连。那么 x, y 不在 Y 上直接相连（否则会有重边，它们可以缩起来），考察以 x 为根的淀粉树的第一层如图。

Omitted proofs for Problem J

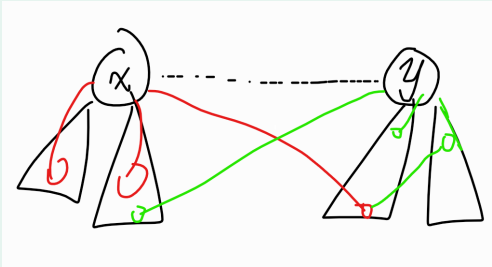
Proof of Lemma (3) (Cont.).



Omitted proofs for Problem J

Proof of Lemma (3) (Cont.).

- 再考察 y 为根的淀粉树。



- 发现这一定会导致矛盾。

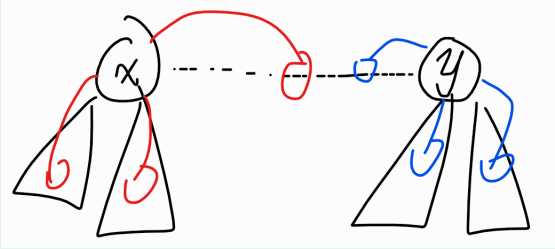
Omitted proofs for Problem J

Proof of Lemma (3) (Cont.).

- Case 2.1 x, y 在 X 上不直接相连，且淀粉树第一层都进入对方子树。那么与 Case 1 讨论类似，不赘述;
- Case 2.2 x, y 在 X 上不直接相连，但在 Y 上直接相连。那么以 x 为根看，y 在 Y 上的度数就会等于 X 上的度数加一，但这与 Lemma (2) 矛盾。
- Case 2.3 x, y 在 X 上不直接相连的其他情况。不妨令以 x 为根的淀粉树第一层不进入对方子树。如图:

Omitted proofs for Problem J

Proof of Lemma (3) (Cont.).



Omitted proofs for Problem J

Proof of Lemma (3) (Cont.).

- 考察 $x \to y$ 的路径从 y 的哪一个子树进来。
- 若从 x 方向进来，则 x, y 不是最近满足条件的二元组（可以取 $x \to y$ 的倒数第二个元素代替 y），矛盾。
- 若从其他子树方向进来，那么以 y 为根可以看出总有一条边跨越了子树，于是不可能是淀粉树。

Omitted proofs for Problem J

Proof of Lemma (3) (Cont.)

- 严格来说，以上只证明了 x 和 y 不是叶子的情况。但事实上，x 和 y 是叶子的情况也与上面证明过程类似，故不赘叙。

□

Omitted proofs for Problem J

Proof of Lemma (4).

- 首先一个方向（合法则每条边必然来自不同子树）的证明只需一句话：若两条边来自相同子树，就不满足淀粉树条件。
- 要证另一个方向（来自不同子树则必然合法），可以证明逆否命题（不合法则存在两条边来自相同子树）。
- 我们可以直接考虑淀粉树的结构。

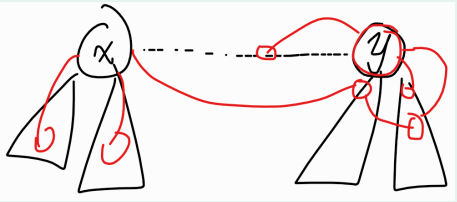
Omitted proofs for Problem J

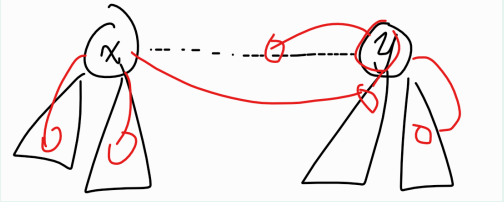
Proof of Lemma (4) (Cont.).

- 设一个合法根是 x，那么即需要证明除 x 外所有点满足如下之一：
  1. 缩点后与 x 在同一个点；
  2. 在两棵树上的度数不同；
  3. 存在两条边来自相同子树。
- 筛去前两种情况后我们只需要讨论度数相同（注意是在原树上而非缩点后）且缩点后不与 x 在一起的点。设这个点为 y，注意到，如果想让 y 的度数在两棵树上相同，则只有如下两种方法：

Omitted proofs for Problem J

Proof of Lemma (4) (Cont.).





Omitted proofs for Problem J

Proof of Lemma (4) (Cont.)

- 对于第一种方法，可以发现存在两个点来自相同子树。
- 对于第二种方法，现在已知 y 的度数满足条件，我们找到 $x \to y$ 路径上倒数第二个点 z，此时 (y, z) 是重边，我们将 y, z 两个点合并（注意合并后不会改变合法性），递归地证明。由于树的大小有限，总会中止（实际上在对树大小归纳）。

□

Omitted Explanations for Problem M

Bostan-Mori 算法

- **问题** 已知两个 n 次多项式 $f(x)$ 与 $g(x)$，求 $\frac{f(x)}{g(x)}$ 的第 k 项系数。
- 分子分母同乘 $g(-x)$，则分母的 $g(x)g(-x)$ 只有偶数项，那么分子的次数在乘完之后奇偶性是不变的。
- 按照 n 的奇偶性只取出对应的一半，那么 k 就折半了，n 不变，一直做到 $k = 0$ 然后输出常数项即可。
- 复杂度 $O(n \log n \log k)$。
- 后面虽然没有直接用到这个过程，但是思路是类似的。

Omitted Explanations for Problem M

多项式复合逆

- **问题** 已知两个常数项为 0 的形式幂级数 $f(x)$ 与 $g(x)$，满足 $f(g(x)) = x$，求 $f(x)$ 的 $1 \dots n$ 项系数。
- 由拉格朗日反演可以推导出结论:

$$f(x) \equiv \frac{x}{g_1} \left(\sum_{k=1}^{n} \frac{n}{k} \left(\frac{x}{g_1} \right)^{n-k} [z^n] \left(\frac{g(z)}{g_1} \right)^k \right)^{-1/n} \pmod{x^{n+1}}$$

- 其中 g_1 代表 $g(x)$ 的一次项。

Omitted Explanations for Problem M

- 唯一的难点在于 $[x^n]\left(\frac{g(x)}{g_1}\right)^k$。
- 考虑更一般的问题：给定 n 和 $A(x)$，对所有 $k \in [1, n]$ 求 $[x^n]A^k(x)$。
- 引入另一个自变量 y 代表 k 这一维：

$$\sum_i x^i \sum_j y^j [x^i] A^j(x) = \sum_j y^j A^j(x) = \frac{1}{1 - yA(x)}$$

- 那么 x^n 项的系数即为所求。

Omitted Explanations for Problem M

- 对这个子问题再考虑更一般的情况，即求 $[x^n]\frac{P(x,y)}{Q(x,y)}$。
- 仿照 Bostan-Mori 算法，上下同乘 $Q(-x,y)$，就可以把 x 的最高次数（也就是 n）折半，递归直到 $n=0$ 即可。
- 注意每次相乘之后 y 的最高次数会翻倍，所以总的项数一直是 $O(n)$ 的。
- 总复杂度 $O(n\log^2 n)$ 。

Omitted Explanations for Problem M

多项式复合

- **问题** 已知两个 n 次多项式 $F(x)$ 与 $G(x)$，求 $F(G(x))$ 的 $0 \dots n$ 项系数。
- 设 $H(x) = F(G(x))$，即

$$H(x) = \sum_{i=0}^{n} f_i G^i(x)$$

- 点积并不好做，可以翻转系数变成卷积。设 $r_i = f_{n-i}$，那么

$$H(x) \equiv \sum_{i=0}^{n} r_{n-i} G^i(x) \pmod{x^{n+1}}$$

Omitted Explanations for Problem M

$$H(x) \equiv \sum_{i=0}^{n} r_{n-i} G^i(x) \equiv \sum_{i=0}^{n} r_{n-i} \left[y^i\right] \frac{1}{1 - yG(x)} \equiv [y^n] \frac{R(y)}{1 - yG(x)} \pmod{x^{n+1}}$$

- 和多项式复合逆的子问题很像，不过这里要求的是 y^n 项的系数。
- 仍然仿照 Bostan-Mori 算法，上下同乘 $Q(-x, y)$：

$$\begin{aligned}
[y^n] \frac{R(y)}{Q(x, y)} &\equiv [y^n] \frac{R(y)}{Q(x, y) Q(-x, y)} Q(-x, y) \\
&\equiv [y^n] \frac{R(y)}{V(x^2, y)} Q(-x, y) \pmod{x^{n+1}}
\end{aligned}$$

Omitted Explanations for Problem M

$$[y^n]\frac{R(y)}{Q(x,y)} \equiv [y^n]\frac{R(y)}{V(x^2,y)}Q(-x,y) \pmod{x^{n+1}}$$

- $\frac{R(y)}{V(x^2,y)} \pmod{x^{n+1}}$ 是一个 x 的最高次数（n）减半，而 y 的最高次数翻倍的子问题。
- 可以递归直到 $n=0$，这时 y 的最高次数一定至少是 n。
- 因为 $R(y)$ 是不变的，递归的时候先不管它，递归到底再乘上然后回溯。
- 递归边界是 $\frac{R(y)}{Q(x,y)} \equiv R(y)Q(0,y)^{-1} \pmod{x^1}$。

Omitted Explanations for Problem M

- 回溯时要乘上 $Q(-x, y)$，这样 x, y 的最高次数都要翻倍。
- x 直接把超过当前层 n 的部分截掉就行了，重点是 y。
- 因为最终只需要 y^n 项，设当前层 $Q(-x, y)$ 中 y 的最高次数是 k，那么当前层返回的结果的 y^i 项最多只会影响到最终的第 $i + 1 + 2 + 4 + \cdots + 2^{k-1} = i + 2^k - 1$ 项。
- 所以 y 这一维只需要保留最高的 2^k 项，较低项可以舍去。
- 和多项式复合逆的分析类似，这样做每层的总项数仍然是 $O(n)$ 的。
- 总复杂度 $O(n \log^2 n)$。

THANKS!

AC.NOWCODER.COM