2023 ICPC 国际大学生程序设计竞赛亚洲区域赛

题解

浙江大学

12.10.2023

M. V-Diagram

Description

给一个 V 图，求一个连续子序列平均值最大的 V 图.

M. V-Diagram

Solution

可以二分答案判断.
也可以观察到如果最小值在 i, 答案区间只能是 $[1, n]$, $[1, i+1]$ 或者 $[i-1, n]$.

J. Mysterious Tree

Description

一棵树，可能是链或者菊花，每次询问一条边存在性，确定是链还是菊花。询问次数 $\lceil \frac{n}{2} \rceil + 3$

J. Mysterious Tree

Solution

首先需要想办法询问到一条存在的边，不妨询问 $(1,2),(3,4),(5,6),\ldots,(n-1,n)$，如果全都不存在，则必然不是菊花。现在假设存在的边是 (u,v)，我们需要在至多三次询问内确定是链还是菊花。假设是菊花，(u,v) 中肯定有一个度数大于等于 3 的节点。由于 $n \geq 4$，则存在两个不同于 u,v 的节点，记为 p_1,p_2。

J. Mysterious Tree

Solution

先询问 (u, p_1) 和 (v, p_1)，这两条边中至多有一条存在。如果均不存在，则必然是链。现在假设 (u, p_1) 存在，则询问 (u, p_2)，如果存在，那 u 的度数就至少为 3，则为菊花。否则为链。

D. Operator Precedence

Description

构造一个序列满足表达式。

D. Operator Precedence

Solution

考虑到乘法不能乘得太大，那么可以构造类似 $x, 2, -1, 2, -1, \cdots, 2, -1, y$ 的序列，令 $y = 1$ 解出 x 即可.

G. Snake

Description

给定 $n \times m$ 地图上的一条长度为 k 的贪吃蛇。每次操作可以控制贪吃蛇移动一步或者缩短一格蛇身。对于每个位置，求从初始状态出发最少需要多少次操作使得蛇头到达该处。

G. Snake

Solution

每个操作都可以等价看作初始蛇身必然缩短 1，并且同时控制蛇头移动一步或者不移动，因此蛇头到达每个位置的时间越早越好。
最短路建图，相邻格子连代价 1 的边。
如果一个位置在初始状态下不被蛇身占据，考虑按 Dijkstra 最短路算法第一次访问该点的时刻，蛇头一定不会与蛇身相交。
否则，对于初始状态下第 i $(2 \leq i \leq k)$ 节蛇身所处的位置，显然必须操作 $k - i$ 次后才能访问该点。因此松弛该点时，需要将最短路长度与 $k - i$ 取 max。

G. Snake

Solution

直接堆优化 Dijkstra 的时间复杂度为 $O(nm\log(nm))$，可以进一步优化成 $O(nm)$。
除去取 max 操作，剩下的部分与 BFS 无异，可以直接队列维护。对于取 max 操作的 k 个点，使用另一个队列升序记录每个状态。那么 Dijkstra 算法的瓶颈——找距离最小的点可以通过比较两个队列的队首 $O(1)$ 得到。

H. Sweet Sugar II

Description

n 个数 $a_1, a_2 \ldots a_n$，有 n 个事件，每个事件形如：如果 $a_i < a_{b_i}$，则 $a_i = a_i + w_i$。事件随机顺序发生，问每个数的期望。

H. Sweet Sugar II

Solution

要求的就是每个事件发生的概率。事件的依赖关系构成一棵基环树。
如果 $a_i < a_{b_i}$，则事件 i 必然发生。若 $a_i \geq a_{b_i} + w_{b_i}$，则事件 i 必然不发生。否则，事件 i 发生当且仅当事件 b_i 在事件 i 之前发生。
则对这样子的事件而言，其发生的概率为 $\frac{1}{L_i!}$，其中 L_i 为基环树上 i 到最近的，必定发生的祖先事件的概率。如果离 i 最近的是一个必定不发生的事件，则事件 i 发生的概率为 0.
使用任何 $O(n)$ 或者 $O(n \log n)$ 的算法处理出 L_i 即可。

E. Period of String

Description

给 n 个串，重排列每个串使得 s_i 是 s_{i+1} 的 period。其中 a 是 b 的 period 定义为 $b_i = a_{i \bmod |a|}$。

E. Period of String

Solution

注意到 s_1 决定之后，所有的字符串都决定了，所以考虑将所有的限制转化到 s_1 上。

考虑 s_i 对 s_{i-1} 的限制，显然 s_i 中包含了完整的 $\left\lfloor \frac{|s_i|}{|s_{i-1}|} \right\rfloor$ 个 s_{i-1}，以及 s_{i-1} 的前 $|s_i| \bmod |s_{i-1}|$ 个字符。注意到这部分的字符集是可以确定的，所以 s_i 对 s_{i-1} 的限制是，要求 s_{i-1} 的前若干个字符，是某个特定的字符集 P。

E. Period of String

Solution

进一步的，我们可以发现这些限制是可以递归传递下去的。不妨假设现在的限制是 s_i 的前 k 个字符为字符集 P，则同样对 s_{i-1} 的限制为前 $k \bmod |s_i|$ 个字符为字符集 P'，其中 P' 为 P 扣除 $\left\lfloor \frac{k}{|s_{i-1}|} \right\rfloor$ 个 s_{i-1} 的字符。

将所有限制转化到 s_1 后检查是否合法即可。

由于每种长度不会取模超过 $\log |S|$ 次，则总复杂度为 $O(n \log |S| (\log n + 26))$.

A. Submissions

Description

给提交列表，问哪些队伍可以在修改至多一个提交的状态时获得金牌。

A. Submissions

Solution

注意到只有 35 个金牌. 可以直接枚举每个提交模拟判断. 但要注意并列的队伍可能很多.

另一种方法是分类讨论. 需要枚举每个队伍分数变高是否能到金牌. 然后对于差一名金牌的队伍, 还需要讨论金牌队伍分数变低,0 题队变 1 题使得金牌数量变多等情况是否能让该队伍获得金牌.

B. Festival Decorating

Description

给数轴上有 n 盏位置不同的路灯。给定每盏灯的位置和颜色，对于每个 d，找到编号最小的灯满足它右边距离 d 的位置存在另一盏与它颜色不同的灯。输出结果相对误差不超过 0.5 即视作正确。

B. Festival Decorating

Solution

对于一个 d，假设在 $[x, 3x]$ 内存在一盏灯满足条件，那么输出 $1.5x$ 即可保证结果正确。因此等价于找到最小的 k 满足答案位于 $[3^k, 3^{k+1})$，共 $\log_3 n$ 个区间。

对于一个区间 $[3^k, 3^{k+1})$，如何对于每个 d 判断是否存在一盏灯满足条件？

令 A_i 表示坐标为 i 且编号在 $[3^k, 3^{k+1})$ 中的灯的颜色，若不存在则为 0；

令 B_i 表示坐标为 i 的灯的颜色，若不存在则为 0。

则 d 有解当且仅当 $\sum_i [A_i > 0][B_{i+d} > 0](A_i - B_{i+d})^2 \neq 0$。反转 A 后拆掉平方为卷积的形式，FFT $O(d \log d)$ 求解即可。

时间复杂度 $O(d \log d \log n)$。

B. Festival Decorating

Solution2

枚举答案，维护当前已经出现的可行的距离 k，使用除了当前颜色之外的颜色的 `std::bitset` 可以计算出新增答案的位置，只枚举这些新增的位置来更新答案即可。需要使用 Big Small 来优化空间。

时间复杂度 $O\left(\frac{n^2}{w}\right)$。

F. Top Cluster

Description

给定一棵 n 个点的无根树，每个点的点权互不相同。q 次询问，每次询问 x 点的 k-邻域内所有点的点权的 mex。

F. Top Cluster

Solution

注意到所有点的点权互不相同，一个数不在 x 点的 k-邻域出现当且仅当它对应的那个点到 x 的距离大于 k。

将所有点按点权从小到大排序。对于每个询问二分答案，需要判断点权不超过 mid 的所有点中是否存在一个点到 x 的距离大于 k。

预处理出每个前缀的直径的两端点，显然只需要检查对应直径的两端点是否满足条件。

使用 $O(1)$LCA 来查询两点距离，总时间复杂度为 $O((n+q)\log n)$。

K. Card Game

Description

给一个牌序列，每次询问区间进行接龙游戏的剩余牌数。

K. Card Game

Solution

假设一次打牌的序列为 $b_1, b_2, \ldots, b_m$

考虑第一张牌的存在性。如果序列中存在和 b_1 一样的牌，则这两张牌中间的全部牌必然会被收走。否则，第一张牌必然保留。

令 nxt_i 为 i 之后的第一张等于 a_i 的牌，$ans[l, r]$ 表示区间 $[l, r]$ 的答案。

则 $ans[l, r] = \begin{cases} ans[nxt_l + 1, r] & r \geq nxt[l] \\ ans[l + 1, r] + 1 & other \end{cases}$

则令 $ans[l]$ 表示左端点为 l 的答案序列，我们可以发现，$ans[l]$ 由一部分的 $ans[l+1]$ 和另一部分的 $ans[nxt_l+1]$ 拼接而成。可以使用主席树维护版本序列。

具体来说，每次我们需要将 $ans[l+1]$ 版本的 $[l, nxt_l-1]$ 做一次区间加一，然后和 $ans[nxt_l+1]$ 版本的 $[nxt_l, n]$ 拼接成一个新的版本。上述操作均可以使用带区间修改的主席树完成。

C. Yet Another Shortest Path Query

Description

给定 n 个点 m 条边的无向平面图。q 次询问，每次给出 S 和 T，求 S 到 T 经过不超过 $k = 3$ 条边的最短路。

C. Yet Another Shortest Path Query

Solution

平面图满足 $m \leq 3n-6$，总度数小于 $6n$，因此一定存在一个点的度数不超过 5。每次取出一个度数不超过 5 的点，将其从图中删去，将删点序列从左往右记录下来。将每条边拆成两条单向边，那么有以下两种边：

- L 边：$v \leftarrow u$，其中 v 早于 u 删除。每个点最多只会作为 5 条 L 边的终点。
- R 边：$u \rightarrow v$，其中 v 晚于 u 删除。每个点最多只会作为 5 条 R 边的起点。

C. Yet Another Shortest Path Query

Solution

平面图满足 $m \leq 3n-6$，总度数小于 $6n$，因此一定存在一个点的度数不超过 5。每次取出一个度数不超过 5 的点，将其从图中删去，将删点序列从左往右记录下来。将每条边拆成两条单向边，那么有以下两种边：

- L 边：$v \leftarrow u$，其中 v 早于 u 删除。每个点最多只会作为 5 条 L 边的终点。
- R 边：$u \rightarrow v$，其中 v 晚于 u 删除。每个点最多只会作为 5 条 R 边的起点。

C. Yet Another Shortest Path Query

Solution

对于询问 S, T，不失一般性，假设 S 在 T 之前删除，考虑从 S 到 T 依次经过的每条边的类型，有以下几种情况：

- 第一条边为 R 边：枚举与 S 相连的 R 边，转化为至多 5 个 $k = 2$ 的子问题。
- 最后一条边为 L 边：枚举与 T 相连的 L 边，转化为至多 5 个 $k = 2$ 的子问题。
- LR/LRR：离线询问，固定 S，枚举第一条 L 边，再枚举两层 R 边，回答所有关于 S 的询问。
- LLR：离线询问，固定 T，枚举第一条 R 边，再枚举两层 L 边，回答所有关于 T 的询问。

C. Yet Another Shortest Path Query

Solution

对于前两种情况，一共可以得到 $10q$ 个 $k = 2$ 的子问题。对于后两种情况，第一层枚举的时间复杂度为 $O(\sum_i deg_i) = O(m)$，后两层为 5^2，总时间复杂度 $O(5^2 m + q) = O(n + q)$。

考虑 $k = 2$ 的子问题，不失一般性，假设 S 在 T 之前删除，依旧考虑从 S 到 T 依次经过的每条边的类型，有以下几种情况：

- LR/RR：离线询问，固定 S，枚举第一条边，再枚举一层 R 边，回答所有关于 S 的询问。
- RL：同上。

第一层枚举的时间复杂度为 $O(\sum_i deg_i) = O(m)$，第二层为 5，总时间复杂度 $O(5m + 10q) = O(n + q)$。

综上所述，总时间复杂度为 $O(n + q)$，常数略大。

设 $E(i,j)$ 表示 (i,j) 到终点的期望步数，则

$$\begin{aligned}E(i,j) =& l(i,j)E(i,(j-1) \bmod m) + r(i,j)E(i,(j+1) \bmod m) \\ &+u(i,j)E((i-1) \bmod n,j) + d(i,j)E((i+1) \bmod n,j) + 1\end{aligned}$$

如果假设到了终点还能继续游走，那么有

$$\begin{aligned}d(i,j)E((i+1) \bmod n,j) =& E(i,j) - l(i,j)E(i,(j-1) \bmod m) \\ -r(i,j)E(i,(j+1) \bmod m) &- u(i,j)E((i-1) \bmod n,j) - 1\end{aligned}$$

换言之，知道了 $E((i-2) \bmod n,j), E((i-1) \bmod n,(j-1) \bmod m)z$, $E((i-1) \bmod n,j)$ 以及 $E((i-1) \bmod n,(j+1) \bmod m)$ 的值，利用 $((i-1) \bmod n,j)$ 的方程即可推出 $E(i,j)$ 的值。

I. Dreamy Putata

Solution

对第 0 行至第 $n-1$ 行建立线段树，考虑表示第 a 行至第 b 行的线段树节点：如果知道了第 $a-1$ 行以及第 a 行的 $2m$ 个 E 的值，利用第 a 行至第 b 行共 $(b-a+1)m$ 个方程，可以推出第 b 行以及第 $b+1$ 行的 $2m$ 个 E 的值。因此在线段树对应节点上记录第 b 行以及第 $b+1$ 行的 $2m$ 个 E 的值关于第 $a-1$ 行以及第 a 行的 $2m$ 个主元的线性表示。信息满足区间合并性质，单次合并的时间复杂度为 $O(m^3)$。

I. Dreamy Putata

Solution

对于每个询问 $S(sx, sy), T(tx, ty)$：由于 (tx, ty) 为终点，因此 $E(tx+1, ty)$ 需要作为额外的第 $2m+1$ 个主元。求出区间 $[0, tx]$ 和 $[tx+1, n-1]$ 的信息，将 $[0, tx]$ 中第 $tx+1$ 行第 ty 个值更改为关于第 $2m+1$ 个主元的线性表示，然后与 $[tx+1, n-1]$ 合并，得到第 $n-1$ 行以及第 n 行关于 $2m+1$ 个主元的线性表示。此时有 $2m+1$ 个未知量，需要构造 $2m+1$ 个方程来解出所有未知量：

- 终点的方程：$E(tx, ty) = 0$。
- 第 1 个至第 m 个主元代表第 $n-1$ 行，列出每个主元与第 $n-1$ 行每个值相等的方程。
- 第 $m+1$ 个至第 $2m$ 个主元代表第 0 行，列出每个主元与第 n 行每个值相等的方程。

解出所有的未知量后，再根据 $sx \le tx$ 和 $sx > tx$ 两种情况分别正确地求出 $E(sx, sy)$ 的线性表示，即可求出 $E(sx, sy)$ 的值。

时间复杂度 $O(qm^3 \log n)$。

L. Master of Both V

Description

维护一个可重集合 S，支持插入线段和删除线段，每次询问后需要回答现在的线段集合是否可以被一个凸包的边界包含。

L. Master of Both V

Solution

注意到我们其实只需要将这些线段按照正确的顺序首尾相连，看看它们是否能构成凸包。

因为凸包上的边是按照极角序排序的，如果可以确定每个线段的方向，令所有其他线段在该线段的左侧，或者根据逆时针、顺时针确定，可以证明我们只需要判断极角序相邻的线段之间是否可以是凸包上相邻的边。

确定一个线段的方向需要选择一个其他线段。如果使用 set 或线段树维护极角序并且判断相邻是否合法，按照时间分治来将删除操作移除，那么每条线段都可以在插入时根据集合中其他线段确定方向，时间复杂度 $\Theta(n \log^2 n)$，难以通过。

L. Master of Both V

Solution

我们首先选择任意一个线段作为 base 来确定其他的方向，当这条线段被删除时，我们选择当前存在的线段中最晚被删除的线段作为新的 base，重构集合中所有线段的方向。
这样每条线段可以证明至多参加一次重构，总重构线段数量为 $\Theta(n)$，总复杂度为 $\Theta(n \log n)$，随机一个线段作为 base 重构时间复杂度期望也是 $\Theta(n \log n)$。

Thanks!