牛客竞赛 AC.NOWCODER.COM | NIO

2022 牛客

暑期多校训练营

黑冰茶出题组

A - Falfa with Polygon

- 给定一个 n 个顶点的凸包和一个常数 k
- 选择这 n 个顶点中的 k 个，使得它们按顺序连线形成的凸包周长最大，输出这个周长
- $3 \le k \le n \le 2000$

题解

- 首先考虑暴力怎么做
- 设 $d[i][j]$ 表示 (i,j) 这条边的长度，实际上问题等价于：从任意一点出发，走 k 条边回到原点的最长路（不超过起点）。
- 如果不考虑回到原点的这个限制，这是一个经典问题。

题解

- 设 $f_m[i][j]$ 表示走了 m 条边，从 i 走到 j 的最长路，那么我们有以下等式：
- $$f_0[i][j] = d[i][j] \tag{1}$$
- $$f_{m+1}[i][j] = \max_k f_m[i][k] + f_0[k][j], m \geq 0 \tag{2}$$

题解

- 这个东西和乘法的形式是一样的，而且是 $\{\max, +\}$ 矩阵的形式，可以用矩阵快速幂加速
- 这里就不细讲了，大家应该都会（网上资料也很多）
- 但是这样做的复杂度是 $O(n^3 \log k)$ 的，并不能满足要求

题解

- $$f_{r+c}[i][j]=\max_k f_r[i][k]+f_c[k][j], m\geq 0 \tag{3}$$
- 稍微改写一下形式，这个函数 f 满足四边形不等式，设 $p[i][j]$ 表示上式取到最大值的 k ，那么：
- $$p[i][j-1]\leq p[i][j]\leq p[i+1][j] \tag{4}$$
- 可以利用决策单调性进行优化，这样矩阵乘法的复杂度就变成 $O(n^2)$ 了

题解

- 最后一个问题是解决回到原点的限制
- 如果我们给凸包顶点标号，在我们选出的 k 个点中，有且仅有一条边是从大编号顶点连向小编号顶点的
- 于是在做矩阵快速幂时，我们可以仅考虑小编号连向大编号的边，最后枚举这条大编号顶点连向小编号的边求答案即可
- 总的复杂度是 $O(n^2 \log k)$

B - light

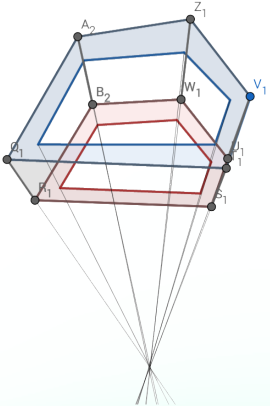
- 给定一个有厚度的凸多边形围墙和一个点光源，问围墙内有多少面积有光

题解

- 这是一个计算几何模拟题：
- 首先我们考虑怎么得到围墙的内层，我们先在 2D 上解决这个问题：把围墙外层的每一条边向内偏移相等的距离（围墙厚度）即可得到。这里推荐使用半平面交来处理这个问题以避免对一些 conner-case 的讨论。

题解

- 接下来，考虑点光源的投影：可知投影和原形状相似，也为凸包，对围墙顶面，其投影在地面也是一个有宽度的凸包
- 其中红色为围墙底面（接地处），蓝色为围墙顶面在地面的投影，灰色为围墙侧面投影。光源在直线相交处。
- 故两个凸包的交面交便是答案，这里同样可以使用半平面交来实现（如果不想写凸包交）



C - Link with Nim Game

- 有 n 堆石头，第 i 堆石头有 a_i 个。Alice 和 Bob 轮流进行操作，Alice 先。每次操作可以从某一堆石头中取出正整数个石头，无法完成操作者输。
- 在双方都采取最优策略的情况下，必败的一方希望尽可能慢的结束游戏，必胜的一方希望尽可能快的结束游戏。
- 求“游戏结束时进行的操作数量”和“第一轮可能进行的操作种类数”

题解

- 首先给出以下结论：
- 对于某个局面，当且仅当所有堆石头数量的异或和不为 0 时，先手必胜。
- 对于必败的一方，其可以保证在之后的操作中双方都只能取一个石头。

题解

- 基于以上结论：
- 若先手必败，则游戏轮数等于石头的总数量，首轮操作方案数可以由第二个结论的证明计算出。
- 若先手必胜，则先手会尽可能多的取石头，并使得剩余石头的数量异或和为 0 。游戏轮数即为第一轮操作后剩余石头数量 $+1$，首轮操作方案数即为满足上述条件的取法数。

结论 2 证明

- 下面我们通过构造的方式证明结论 2。
- 对于一个必败态，其各堆石头数量的异或和为 0 。
- 考虑从 $\operatorname{lowbit}(a_i)$ （记为 v）最小的那堆石头中取出一个石头，此时异或和变为 $2v-1$ 。必胜方只能从另一个 $\operatorname{lowbit}(a_i)=v$ 的堆中取出一个石头，才能保证异或和依然为 0 。

结论 2 证明

- 但是！
- 这只是证明了存在一种方案可以只取一个，并不代表只取一个的方案只有这一类。
- 需要枚举所有堆中取出一个石头的方案，判断下一步是否只能取一个石头。
- 取出一个石头后的异或值只有 log 种可能，因此只需要判断这 log 种异或值是否在下一步只能取一个石头即可。
- 出题人和验题团队都没有考虑到上面这个问题，导致在赛时进行了一次重测，我们对此深表歉意。

D - Link with Game Glitch

- 给定 m 个物品合成的方式，求一个最大的合成损耗参数 w ，使得所有物品都无法通过无限合成的方式无限获得。

题解

- 考虑建图
- 对于每个物品建点，每个合成方式由 b_i 向 d_i 建有向边，边权为 c_i/a_i 。
- 原问题实际上是要求一个最大的 w ，使得在每条边的边权乘上 w 之后，不存在一个乘积大于 1 的环。
- 首先二分答案，check 的问题类似于求负环。由于边权乘积较大，需要对其取对数。

E - Falfa with Substring

- 对于所有的 $0 \le i \le n$, 求长度为 n 的字符串中恰好出现了 i 个"bit" 子串的字符串数量，答案对 998244353 取模.
- $1 \le n \le 10^6$.

题解

- 看到“恰好”就想到容斥.
- 先考虑至少出现了 k 个"bit"的串的数量 F_k ，如果将每个"bit"看成一整个特殊点，其他字符看成可选任意字符的普通点的话，容易发现它等价于从 $n-2k$ 个点中选出 k 个特殊点的方案数乘上字符选择方案数, 即为 $F_k=\binom{n-2k}{k}26^{n-3k}$.
- 直接套上容斥，得到答案 $G_k=\sum_{j\ge k}\binom{j}{k}(-1)^{j-k}F_j$.
- 分离系数得到 $k!G_k=\sum_{j\ge k}(j!F_j)\frac{(-1)^{j-k}}{(j-k)!}$.
- 发现它是卷积的形式，设 $P_i=i!F_i$ ，$Q_i=\frac{(-1)^{n-i}}{(n-i)!}$ ，$R_{n+i}=i!G_i$ ，则有 $R=P\times Q$ ，所以我们直接使用 ntt 加速一下式子计算就做完了.

F - NIO with String

- 给定一个字符串 s 和 n 个匹配串 t_i，Q 次操作，有以下四种：
- 操作 1：在一个匹配串 t_i 的，末尾加上一个字符 c
- 操作 2：删除 s 的末尾 p 个字符
- 操作 3：在 s 的末尾增加 k 个字符 c
- 操作 4：询问有多少个匹配串的字典序严格小于 s
- $n, Q \leq 2 \times 10^5$，仅包含小写字母

题解

- 首先不考虑对 t 的修改，如果 s 每次只增加/删除一个字符怎么做
- 一个朴素的方法是，我们对所有匹配串 t 建出 Trie，按字典序 dfs
- 比 Trie 上一个节点所代表的字符串小的所有串，对应 dfs 序先于它的所有节点
- 于是可以 dfs 一遍，预处理出所有节点的答案，这样我们只需要考虑 s 匹配到字典树上哪个节点就行了。如果每次增加/删除一个，可以直接暴力爬树

题解

- 现在考虑 t 后面加一个字符的操作
- 我们可以离线建出所有操作完成后的 Trie，如果 Trie 上一个节点 x 对应某一个匹配串，这个串会使得所有 dfs 序大于节点 x 的位置答案 $+1$
- 按 dfs 序建线段树，每增加一个字符对应一段区间 $+1$，一段区间-1，简单维护即可（也可以是单点加，维护前缀和）

题解

- 再考虑对 s 的操作，问题的关键是如何快速找到 s 在 Trie 上对应的节点
- 对于删除操作，显然可以用倍增往上跳
- 对于增加操作，由于加的是同一个字符，我们可以预处理出 $c_{i,j}$ 表示节点 i 一直往下走 j 这个字符边能走到哪个节点，走过头了倍增跳回去即可
- 注意考虑 s 是恰好位于 Trie 上某个节点还是超出了，总复杂度大概是 $O((n+q)\log n)$ 级别

题解（在线做法）

- 考虑到可以在 trie 上操作，我们需要每次在询问串的后面加入若干个字母，因此我们可以先对 trie 进行一个剖分，将所有相同字母的点合并成一条链，并在所有的点上记录链首标号，然后在链首使用 vector 记录整条链的标号. 这样我们就能够做到在 trie 上一次跳过一整段相同字母。

题解

- 再考虑如何维护字典序比某个串小的串的数量。我们可以在 trie 上某个串每一个分支处，将所有该分支处字符比它大的点的计数器加一，然后再将这个串的最后一个点的所有儿子计数器加一。容易发现，计数器相当于是记录了每个串在什么位置开始比询问串字典序更小，因此询问串在 trie 上经过的所有的点的计数器之和即为答案。
- 值得注意的是，询问串有可能末端不在 trie 上，这时我们需要额外计算询问串在 trie 上的末端分支处出现的字典序更小的串数量，可以通过记录一些额外信息简单维护。

题解

- 下面我们考虑每个操作会产生什么影响。
- 对于询问串添加若干个相同字符，我们可以在 trie 上一次性跳过一整段的点，但还需要一次性将一整段的点上的答案和算出来，因此我们可以再在链首处建一个树状数组来维护答案和，总复杂度为 $O(n\log n)$ 。
- 对于询问串删除若干个字符，我们可以开一个栈来维护字符类型、数量、答案、trie 上对应的点标号等信息，然后删除时直接将栈尾弹出，直到删除了足够的字符，弹出数量和插入数量一致，因此总复杂度是 $O(n)$ 。值得注意的是，可能有一整段相同字符在删除时被删去了一部分，因此最后可能要重构栈尾。

题解

- 对于原串添加字符，相当于在 trie 上某个点下增加了一个新儿子，我们需要在某条剖分链后添加一个新点，而 vector 和树状数组都可以做到这个操作。另外我们需要重新维护一遍计数器的答案。这让我们可能需要 26×2 个位置的计数器值，但我们发现，某个点只会存在于其中一条剖分链中，因此，实际上至多只有两个树状数组需要修改内部的点值，而其他树状数组都是修改头部的点值，那么我们只需要将头部分开来计算即可，复杂度为 $O(52 + \log n)$ 。

题解

- 修改完 trie 中的信息后，我们还需要注意到，部分询问串涉及到的点信息改变了，因此询问串的答案也发生了变化，那么我们就需要重构询问串中对应位置的点的答案。我们在栈上二分找到深度对应可能被涉及到的点，然后重构这一部分栈的信息即可。复杂度 $O(\log n)$ 。
- 综上，这种在线做法的总复杂度为 $O((n+q)\log n)$ 。

G - Link with Monotonic Subsequence

- 构造一个排列，使其 $\max(\text{lis}(p), \text{lds}(p))$ 最小。

题解

- 首先给出以下结论：
- 排列权值的最小值为 $\lceil \sqrt{n} \rceil$
- 只需要构造形如 $3, 2, 1, 6, 5, 4, 9, 8, 7$ 的排列即可

结论证明

- 对于排列中的每个元素，我们记一个二元组 $(\text{lis}_i, \text{lds}_i)$ ，其中 lis_i 为以第 i 个数结尾的 lis ，lds_i 为以第 i 个数结尾的 lds 。
- 对于某个排列生成的所有二元组，其必定是两两不同的。（证明考虑反证法，此处略）
- 因此所有二元组中的最大值至少为 $\lceil \sqrt{n} \rceil$ 。

H - Take the Elevator

- n 个人坐电梯，楼高 m ，每个人有起始楼层和目标楼层。
- 电梯有载客量限制 k ，上升时可以上升到任意层并随时下降，但是下降要一直下降到一层才能再上升。
- 电梯每秒运行一层，换方向和上下人不占用时间，问电梯最短运行时间。
- $n, m \leq 2 \times 10^5, k \leq 10^9$

题解

- 其实一开始 idea 题目限制了一个人不能中途下电梯，不过验题的时候发现答案是一样的，这里按原 idea 说了
- 解题的关键在于下降时一定要到一层的限制，这样我们其实希望上升的高度尽可能小
- 考虑上行，目标楼层的那个人一定要安排一趟，本着不能浪费的原则，我们一定会往电梯里塞到达楼层尽可能高的人
- 于是我们贪心地选择上电梯的人即可
- 事实上“中途能不能下电梯结果一样”这个结论也可以类似这样简单推出来

题解

- 具体来说，我们可以把每个人看成一条线段 $[l, r]$：
- 如果当前电梯内人数不足 k，只需要找一条还未选择的 r最大且 $r \leq r_{\max}$的线段（起始楼层不能低于已经上了电梯的人）
- 如果当前电梯内人数为 k，只需要找一条还未选择的 r最大且 $r \leq l_{\max}$的线段（有人下电梯）
- 下行其实是一样的，只不过变成贪心选择起始楼层最高的
- 每轮上行下行取一个 max 即可，最后复杂度是 $O(n \log n)$的

I - let fat tension

- 每个功夫大师有能力值和技能值，都为向量；
- 功夫大师们要互相学习技能，一个功夫大师学习后的技能值为所有功夫大师技能值的加权和；
- 两个功夫大师间的学习效率（学技能的加权权值）为能力值的余弦相似度；
- 求所有功夫大师学习后的技能值。

- 数据范围：
- 功夫大师的数量不超过 10^4，所有向量长度不超过 50

题解

- 提示 0：显然，本题的问题是学习效率就有 n^2对，根本算不出来；
- 提示 1：一个 $n*m$的矩阵与 $m*d$的矩阵相乘复杂度为 $O(nmd)$；
- 提示 2：数据范围看上去暗示 $O(nkd)$的做法；
- 提示 3：所以，有没有一种可能，题目中操作可以写成矩阵的形式；

- 最终思路：
- 题目中操作可以写成三个矩阵的相乘，通过矩阵结合律调整运算顺序，使得可以在 $O(nkd)$复杂度内完成操作

题解

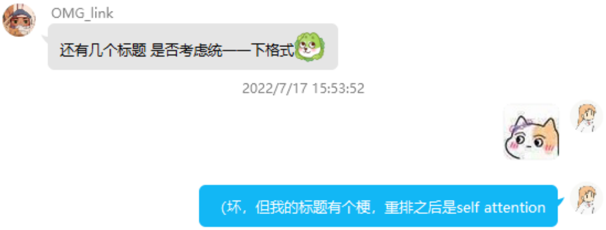
- 矩阵 X表示输入的 $n*k$矩阵，矩阵 X_{norm}^T 表示矩阵 X沿行归一化（即，X_{norm}^T 每行元素的平方和为 1）后得到的矩阵；
- 矩阵 Y表示输入的 $n*d$矩阵，则所求答案可以表示为：

$$M_{answer} = X_{norm}X_{norm}^T Y$$

- （因为 $X_{norm}X_{norm}^T$得到的 $n*n$矩阵里 i行 j列的元素就是 $le(i,j)$）
- 上述三个矩阵大小分别为 $(n*k),(k*n),(n*d)$；
- 从左到右顺序计算，运算次数为 n^2k+n^2d；
- 先算 $X_{norm}^T Y$，则运算次数为 $knd+nkd$，十分合理，复杂度 $O(nkd)$

花絮

- 一个烂梗：这个很奇怪的标题（让脂肪紧张？）字母重排后可以组成 self-attention——某种深度学习相关的科技，也是本题的灵感来源（十分抱歉破坏了整齐的标题们.jpg）；



花絮

- 具体来说，self-attention 可以抽象为对于一个矩阵 A^{n*d}之中 $n >> d$，要计算：$f(AA^T)A$，之中 $f(\cdot)$为作用于分别作用于矩阵各个元素的非线性函数，显然该过程运算次数为 $2n^2d$；而如果没有 $f(\cdot)$，则可以仅用 $2nd^2$次运算算出；这是本题灵感来源，即去掉 $f(\cdot)$（激活函数）的 self-attention；
- 实际 self-attention 中不能简单粗暴的去掉 $f(\cdot)$，导致复杂度不好优化，有兴趣（真的有人有兴趣吗）的同学可以了解线性 Transformer，如 LinFormer、FlowFormer 等内容

J - Link with Arithmetic Progression

- 给定一个数列 a ，将其修改为一个等差数列 b，代价为 $\sum_{i=1}^{n}(a_i - b_i)^2$
- 最小化代价

题解

- 给出三种做法:
- 1. 设 $b_i = b_0 + id$ ，使用三分套三分求 b_0 和 d 。
- 2. 在上一做法的基础上，注意到每一层三分的事实上都是一个二次函数，可以直接求出极值。
- 3. 此题的本质是一个线性回归，直接套式子即可。

K - Link with Bracket Sequence I

- 已知括号序列 a 是一个长度为 m 的合法括号序列 b 的子序列，求可能的序列 b 的数量。

题解

- 记 $dp_{i,j,k}$ 表示在序列 b 的前 i 位中，与 a 的 lcs 为 j，且左括号比右括号多 k 个的方案数。
- 转移时枚举下一位填写的是哪种括号即可。

L - Link with Level Editor I

题目大意

- 有 n 个世界，每个世界是一张简单有向图。从这些世界中选择一个子段进行游戏，规则为从 1 出发，每个世界可以原地不动或经过一条边，到达点 m 即为胜利。
- 要求选出一个尽可能短的子段，使得存在至少一种方案可以胜利。

题解

- 记 $dp_{i,j}$ 表示在第 i 个世界中到达点 j，最晚可以从哪个世界出发。
- 使用滚动数组优化掉第一维即可。

牛客竞赛
AC.NOWCODER.COM
NIO

THANKS!

AC.NOWCODER.COM