2019 多校训练第七场题解

华东师范大学

2019 年 8 月 12 日

花絮

- 预估难度:
  - 简单: A, F, K
  - 中等: D, G, H, J
  - 难: B, C, E, I
- 实际情况: K > A > F > J > H > G > B > D > C > E = I. 其中 IE 0 AC.
- 这是敝校第一次出多校. 由于出题人都很忙 (摸), 导致出题过程断断续续拖了好久. 好在最后并没有夭折. 尽管对题面进行了好几遍的 review, 仍然出现了不少问题, 敬请谅解.
- 必须承认, 整场比赛没有非常有趣的 idea, 大部分题目的想法或是很套路, 或是很直接. 有好几题的难点在于实现的繁琐.
- I 是月赛原题 (`https://acm.ecnu.edu.cn/contest/174/`), 由于没人补, 所以再放送. 所以一定要来做我们月赛啊?
- I 当时出现在月赛的时候, 甚至 std 都是锅的.
- F 的题目背景一开始不是期末考, 改了以后变成了阅读理解. 改成了期末考的原因是期末考的首字母是 F.
- 一开始的题目预估会是最温暖的一场多校, 在一番操作, 强行增加了难度后, 似乎成为了没人 AC 题目数量最多的一场多校 (真的吗?).

A + B = C

解一 补零到 a, b, c 长度相等之后, 可能的情况只有四种: $b \mid (c-a), b \mid (10 \cdot c - a), a \mid (c-b), a \mid (10 \cdot c - b)$. 逐个判断.

解二 首先把 a, b, c 末尾的 0 都去掉得到 A, B, C, 方便处理. 去掉的 0, 显然是可以通过调整相对大小补回来的.

考虑 $C \cdot 10^k$ ， $k > 0$ 的情况只存在于 $A + B$, 因为如果是 $A \cdot 10^n + B = C \cdot 10^k$ $(n > 0, k > 0)$ 的情况, 显然 B 的末尾是存在 0 的, 这和我们上面已经去掉了末尾的 0 矛盾.

那么接下来, 显然就是 $k = 0$ 的情况, 这样的话, 显然 $n = 0$ 或者 $m = 0$, 因为 C 的最后以为是非 0 的. 确定一个数和 C 末尾对齐以后, 另一个要么是和 C 首位对齐, 要么就是和 C 第二位对齐 (发生了进位).

还有一种情况是, A (不失一般性) 和 C 长度相等, 这个时候 B 的位置是不确定的, 但我们可以通过从 A 的末尾开始和 C 比较, 不相同的最后以为就是 B 的末尾所在的位置.

技巧 这道题目其实可以不用到高精度. 判断两个数相加结果是否等于第三个数, 可以直接用 hash 判断.

Bracket Sequences on Tree

这题一开始是这样的: 给以 n 个分别为根形成的有向树划分等价类. 但是出完发现跟一个原题完全一样. 于是又稍微加了一点别的技 (原) 巧 (题).

不难看出, 原先那个题还是需要解决的. 我们先看一下如何划分等价类. 如果树够小的话, 我们可以将每一棵子树都与一个括号序列对应起来. 这个括号序列是一个 DFS 序列. 其遍历规则是: 遍历子树时按照子树对应的括号序列的字典序进行遍历（这样遍历的顺序就是唯一的）. 由于树比较大, 我们不可能将所有的子树的 DFS 括号序列都存下来, 我们可以退而求其次, 存下这些括号序列的哈希, 然后按照哈希值从小到大遍历. 这样我们就得到了固定根的哈希值.

还有一个问题就是求出所有根的哈希值. 对于这个问题, 我们可以看我们还少了什么信息. 假设 u 的父亲是 v, 那么以 u 为根时, 我们还需要知道 v 删掉 u 这棵子树, 加上 v 的父亲关于 v 的 “某一块” 的哈希值. 这个哈希值可以通过对第一步预处理得到的排序列表进行预处理前缀、增减元素得到. 注意到我们算出来的这个哈希值, 就是 v 关于 u 的 “某一块” 的哈希值. 这样, 我们可以自顶向下, 用一种类似于树形 DP 的方式, 求出每个节点的父亲关于它的 “某一块” 的哈希值. 然后我们就可以换根. 原问题就做完了.

现在我们考虑怎么计数. 对于本质不同的有根树, 对答案的贡献是:

- 分子上有: $deg(root) \prod_{i=1}^{n} (deg(i) - 1)!$.
- 分母上有: 每个节点的子树的哈希值每个出现了多少次的乘积.

注意到由于换根的时候子树会换, 这个乘积也需要在换根的时候动态维护. 到此为止这题就做完了.

这题时限开到了 HDU 最大允许的时限: 20 秒. 由于标程写得比较垃圾, 就算开了 20 秒, 也还是没达到两倍. 但是看比赛中的情况, 这个时限还是比较宽裕的.

另外哈希不需要双哈希. 只要自然溢出就能过了. 单哈希模素数试了好多次都不行 (可能是运气不大好).

Cuber Occurrence

$f(s')$ 就是把 s' 出现过的位置的集合拿出来, 如果相邻两个位置之差不超过 $|s'|$, 显然对答案的贡献 $+1$.

考虑 SAM 上每个节点的 endpos 的集合, 考虑该集合中所有相邻位置构成的差的集合 S. 为了满足 $f(s') = k$, 要有 S 中的第 k 大元素 $\leq |s'| < S$ 中的第 $k+1$ 大元素.

endpos 集合可以通过后缀树上启发式合并得到, 同时维护 endpos 差的集合 S, 由于需要求第 k 大, 所以需要用主席树（权值线段树）或者手写平衡树或者 `pb_ds` 中的平衡树.

由于需要的是字典序最小的串, 在启发式合并过程中判断每个节点可行性之后, 按照字典序遍历节点, 找到第一个可行的节点, 输出最短的串即可.

由于 HDU 最大内存限制为 512 M, 所以如果出现了卡内存, 我们也无能为力.

Data Structure Problem

最小点积的两个点一定都在凸包上.

简单证明: 假设里两个点都不在凸包上, 考虑把一个点换成凸包上的点（不动的那个点）, 不管你是要点积最大还是最小, 你都可以把那个不动的点跟原点拉一条直线, 然后把所有的点投影到这条直线上, 要找的无非就是这条直线最前面或者最后面的两个点. 这两个点不可能是不在凸包上的点. 同理我们可以把另一个点移到凸包上.

于是这道题目变成了动态维护凸包. 而且数据保证是随机的.

- 随机可以保证凸包上点的个数的期望是 $\log n$, 每次加点把原来凸包和新的点跑凸包;
- 删掉凸包上点的概率是 $\frac{\log n}{n}$, 出现这种情况的时候, 暴力重建, 重建次数是 $\frac{q \log n}{n}$.

于是总的复杂度是 $O(q \log^2 n)$.

这题有个问题是你在实现 erase 的时候会发现需要有一种东西可以查第 k 个还在的元素, 可能要使用某种数据结构 (Treap or BIT or `pb_ds`). 所以这个数据结构题要考察的压根就不是题目中所给的那种数据结构, 而是因为要解决问题不得不使用一种其他的数据结构. 这是题目标题的一个双关.

Equation

为了减少推导的复杂性, 降低题目难度, 因此把 2 给去掉了, 实际上含 2 的也可以以同样方式推导（如果找规律够厉害的话理论上是可以找到规律的）. 可能有多种推导方法, 提供一种方法.

解一 首先, 解的数量是积性函数. 因为对于任意一个 $\pmod p$ 和 $\pmod q$ 的解, 都可以通过中国剩余定理得到一个 $\pmod{pq}$ 的解.

我们令 $x_1^2+\cdots+x_n^2\equiv 0 \pmod{p^k}$ 的解的数量为 $a_n(p^k)$, $x_1^2+\cdots+x_{n-1}^2\equiv x_n^2 \pmod{p^k}$ 的解的数量为 $b_n(p^k)$. 原式可化为 $x_1^2+\cdots+x_{n-2}^2=(x_n-x_{n-1})(x_n+x_{n-1}) \pmod{p^k}$.

对于任意 $x_n-x_{n-1}\not\equiv 0 \pmod{p}$, 可以任意取 $x_1^2+\cdots+x_{n-2}^2$, 都存在唯一一个确定解, 因为可以将左边乘上逆元则可以确定 x_n+x_{n-1} 的值, 从而解出一组解. 对于任意 $x_n-x_{n-1}\equiv 0 \pmod{p}$, 考虑 $x_n-x_{n-1} \bmod p$ 的阶 t (即最高可以整除 p 的几次). 左式的阶必须大于等于 t. 因此, 此时右边可取方案数为 $a_{n-2}(p^t)\cdot p^{(k-t)(n-2)}$, 从而两边同时除掉 p^t 之后的方程便可以同上算出解的个数.

因此可以得到 $b_{n+2}(p^k)=(p^k-p^{k-1})(p^{kn}+p^{(k-1)n}a_n(p)+\cdots+p^n a_n(p^{k-1}))+p^k a_n(p^k)$.

接下去考虑如何求 $a_n(p^k)$.

首先, $a_1(p^k)$ 和 $a_2(p^k)$, 可以找规律或者 OEIS (当然也可以严格证明) 得到. 接下去分 $p \pmod 4\equiv 1$ 和 3.

当 $p\equiv 1 \pmod 4$ 时, $a_n=b_n$. 因为对于其二次剩余系而言 x^2 与 $-x^2$ 可以一一对应, 因此 $x_1^2+\cdots+x_{n-1}^2\equiv -x_n^2 \pmod{p^k}$ 与 $x_1^2+\cdots+x_{n-1}^2\equiv x_n^2 \pmod{p^k}$ 解数量相同 (左边随便取, 右边解的数量一致).

当 $p\equiv 3 \pmod 4$ 时, 对于其二次剩余系而言 x^2 与 $-x^2$ $(x\neq 0)$ 可以合并为一个完整的简化剩余系. 因此, 可以求出 a_n+b_n (左边随便取, 右边要么属于 a 的解要么属于 b 的解). 但是对于 $x_n^2\equiv 0 \pmod{p}$ 而言, 就需要特殊考虑. 因为在剩余系中不存在阶为奇数的数, 平方以后阶一定是偶数, 原该是奇数阶的数量都变成了对应的阶数 +1, 即需要减去奇数阶的数量, 加上偶数阶的数量. 因此 $a_n(p^k)+b_n(p^k)=2\cdot\left(p^{k(n+1)}+\sum_{i=1,\dots\frac{k}{2}}(a_{n-1}(p^{2i})p^{(k-2i)(n-1)}-a_{n-1}(p^{2i-1})p^{(k-2i+1)(n-1)})p^i\right)$.

这样就可以递推求出 $b_n(p^k)$.

解二 有这个问题的推广版的论文, 用那个公式也可以通过此题 (如果能看懂并实现优雅的话). arXiv 链接: `https://arxiv.org/abs/1404.4214`.

Final Exam

很抱歉由于出题和验题上的疏忽, 导致这题的题面至少存在三个问题. 在比赛的一开始耽误了大家的时间.

换位思考, 考虑如果我们是出题人会怎么让学生做不出 k 题, 即最坏情况. 显然, 我们会挑出学生复习得最少的 $n-k+1$ 道题, 让每道题的难度都等于他复习的时间. (田忌赛马的策略)

因此, 回到学生视角, 我们要让自己复习的最少的 $n-k+1$ 题复习时间总和 $>m$, 构造方式显然.

那么, QQ 小方还来得及复习吗?

Getting Your Money Back

此题的关键在于想清楚情况, DP 方程实际上很简单.

显然区间是没有用的, 可以仅保留长度, 但是又会发现, 本来 0 开头的的和本来不是 0 开头的计算方式不同, 那么就分两种状态, 然后可以写出平方的 DP 方程, 然后进行优化.

$f[0,x] = \min(\max(f[0,i-1]+b, f[0,x-i]+a))$ 表示区间长度为 x, 左端点为 0.

$f[1,x] = \min(\max(f[1,i-1]+b, f[0,x-i]+a))$ 表示区间长度为 x, 左端点不为 0.

注意初始值 $f[0,0]=0, f[1,0]=a$.

由单调性, 每次转移可以二分确定对于成功和失败最接近的位置. 或者直接对函数三分.

当然, 也可以优化到线性.

Halt Hater

解一 首先分析到达 (x,y) 时所有可能的左转次数, 直行次数组成的二元组 (s,t). 对于某一个 (s,t), 我们删掉所有满足 $s' \geq s$ 且 $t' \geq t$ 的二元组 (s,t), 留下的那些二元组 $\{(s,t)\}$ 组成一个集合. 我们将这个集合记为 $f(x,y)$.

通过简单打表以及大胆猜想 (也可以小心验证, 不过会相当繁琐), 我们可以得到如下结论.

- $f(x,y) = f(x,-1-y)$.
- $f(x,y) = f(1-x,y)$.
- $f(x,y) = \{(i, x-1-i+\max(y-i,0))|0 \leq i < x\}$ if $x \geq 0$ and $y < x$.
- $f(x,y) = \{(i, y-i+\max(x-i-1,0))|0 \leq i \leq y\}$ if $x \geq 0$ and $y \geq x$.

观察最后两个方程, 我们可以发现是一条性质很好的折线. 而我们要解决的最优化问题其实是以这条折线为约束的线性规划问题. 对决策有影响的必然是端点和折点. 这样的点顶多三个, 逐个枚举即可.

解二 验题人用了一个近似乱搞的做法.

本质上看起来还是个线性规划问题, 因为转向的次数是近似固定的.

首先配合左转和右转, 一个左转完全可以替代直走, 因此我们可以准确求出笔直向前走 x 步的最小代价. 之后是考虑在坐标轴上, 方法是枚举开局走的几步. 很容易推出只有很少的走法是有意义的. 比如在 x 正半轴上, 显然应该直接右转然后笔直走. 之后再考虑第一象限. 要么笔直走, 再右转 (尽量直走), 要么右转再左转 (尽量直走), 要么是先右转或先直走的不停左转右转走锯齿 (尽量右转).

注意这里的直走指的是我们刚刚求得的子问题. 因为线性规划问题就是尽量选性价比最高的方案, 不会证明这样搞是否严格是对的.

之后考虑其他象限, 当然也是枚举开始的有效走法, 比如第二象限可以左转或者直走再右转三次. 然后把他转成第一象限. 写法上把它们写成子问题就很容易实现.

Intersection of Prisms

我们只关心两个多边形被 $y = y_0$ 直线截取的长度 $len(y_0)$, 因此把两个多边形都按照节点 y 坐标排序, 统计有向边的斜率和, 特判有向边与 $y = y_0$ 平行的情况.

我们发现, $len(y_0)$ 是一个分段函数, 且每一段都是一次函数.

现在, 我们要求 $len_1(y_0) \cdot len_2(y_0)$ 的积分, 可以从左到右枚举两个函数的每一段, 然后求两个一次函数的积分, 加到答案里.

注意在统计斜率和时, 要删除已经不在答案中的有向边.

Just Repeat

每个人优先出的牌的颜色肯定是场上没出过的, 对方也持有的, 并且两个人手中持有数量最多的牌. 对方持有的越多意味着可以封掉更多的牌, 而自己手里的越多意味着可以防止自己更多的牌被封掉.

因此, 对所有两个人手里都持有的颜色的牌数进行统计, 从大到小依次分配给第一, 第二个玩家. 如果此时第一个玩家手里的牌数 > 第二个玩家, 则第一个玩家胜利, 否则第二个玩家胜利.

到此为止, 问题转换成另一个问题, 就是有一堆东西, 每个东西有两个值, A 拿到这个东西的收益是 a_i, B 拿到的收益是 b_i. 两人依次拿. 求最优策略下两人的各自收益. 这是一个经典问题, 答案就是按照 $a_i + b_i$ 排序模拟一下就好了. 至于这样做是正确的理由, 那就留给读者思考了.

由于出题人比较变态, 直接上 map 可能会 T, 单纯用 cin 是没什么问题的, 只要你别其他地方也都写得很慢, 还慢大常数倍的那种.

Kejin Player

设从 l 到 r 的期望为 $g(l, r)$, 这种期望满足减法 $g(l, r) = g(1, r) - g(1, l)$. 因为升级只能一级一级升, 所以要从 1 升级到 r, 必然要经过 l.

因此用期望方程推导一下 $g(1, x)$ 就好了.