牛客竞赛
AC.NOWCODER.COM

2024 牛客
暑期多校训练营 1

quailty, 葫芦,Gromah

A - A Bit Common

题目大意

- 求有多少长为 n 的元素是 $[0, 2^m)$ 的整数序列
- 满足存在一个非空子序列的 AND 和是 1
- 答案对输入的正整数 q 取模
- $1 \leq n, m \leq 5000, 1 \leq q \leq 10^9$

题解

- 二进制最低位为 0 的数一定不在 AND 和是 1 的子序列里，这些数除了最低位都可以任选
- 对于二进制最低位为 1 的数，如果存在一个子序列的 AND 和是 1，那么包含这个子序列的子序列的 AND 和也是 1
- 只要所有二进制最低位为 1 的数 AND 和是 1 就能满足条件

题解 (cont'd)

- 记二进制最低位为 1 的数有 k $(k > 0)$ 个，这些数除了最低位的 AND 和都要是 0，也就是每一位上这些数都至少有一个是 0
- 方案数是 $\binom{n}{k}2^{(n-k)(m-1)}(2^k-1)^{m-1}$，对 k 从 1 到 n 求和即可
- 时间复杂度是 $O(n(n+m))$

B - A Bit More Common

题目大意

- 求有多少长为 n 的元素是 $[0, 2^m)$ 的整数序列
- 满足存在**两个不同的**非空子序列的 AND 和都是 1
- 答案对输入的正整数 q 取模
- $1 \leq n, m \leq 5000$, $1 \leq q \leq 10^9$

题解

- 二进制最低位为 0 的数一定不在 AND 和是 1 的子序列里，这些数除了最低位都可以任选
- 对于二进制最低位为 1 的数，如果存在一个子序列的 AND 和是 1，那么包含这个子序列的子序列的 AND 和也是 1
- 考虑有且仅有一个非空子序列的 AND 和 1 的情况，只能是所有二进制最低位为 1 的数构成的子序列的 AND 和是 1，并且如果有多于一个数，在任意移除一个数之后 AND 和都不是 1

题解 (cont'd)

- 记二进制最低位为 1 的数有 $k\,(k>0)$ 个，这些数除了最低位的 AND 和都要是 0，也就是每一位上这些数都至少有一个是 0
- 如果 $k=1$，那么这个数除了最低位都是 0，否则 $k>1$
- 如果某一位上有至少两个 0，那么移除任意一个数之后这一位还是 0
- 如果某一位上恰有一个 0，那么移除这个 0 所在的数之后这一位是 1，这个位称为“特殊位”
- 要使得移除任意一个数之后 AND 和都不是 1，也就是总有某一位是 1，需要每个数对应至少一个“特殊位”

题解 (cont'd)

- 考虑 $dp_{i,j}$ 表示 i 个数有总共对应了 j 个“特殊位”的方案数
- 新加进来的“特殊位”可以被这 i 个数之一所对应，移除这一位之后可能还有 i 个数有对应“特殊位”，也可能只剩 $i-1$ 个数有对应“特殊位”
- $dp_{i,j} = i \times (dp_{i,j-1} + dp_{i-1,j-1})$

题解 (cont'd)

- 记这 k 个二进制最低位为 1 的数总共对应了 t 个“特殊位”
- 方案数是 $\binom{n}{k}2^{(n-k)(m-1)}\binom{m-1}{t}dp_{k,t}(2^k-k-1)^{m-1-t}$，对 t 从 k 到 $m-1$ 求和，再对 k 从 2 到 n 求和，额外再加上 $k=1$ 时的方案数即可
- 最后用 A 题的答案减去上述求得的结果即为本题的答案
- 时间复杂度是 $O((n+m)^2)$
- 非常量取模时使用高效的模乘算法可以得到更快的运行效率，例如 barrett reduction (atcoder::dynamic_modint 使用了这一算法)

C - Sum of Suffix Sums

题目大意

- 维护一个初始为空的非负整数序列，支持 q 次操作
- 每次操作移除末尾的若干个整数，然后在末尾加入一个整数 v
- 每次操作后输出当前序列所有后缀和的**总和**
- 答案对 $1\,000\,000\,007$ $(= 10^9 + 7)$ 取模
- $1 \leq q \leq 5 \times 10^5$, $0 \leq v \leq 10^9$

题解

- 记当前序列为 $a_1, a_2, \ldots, a_n$
- 不难发现 a_i 会出现在所有起始位置 $\leq i$ 的后缀和里，在总和里出现 i 次
- 总和就是 $\sum_{i=1}^{n} i \times a_i$，直接维护即可
- 时间复杂度是 $O(q)$

D - XOR of Suffix Sums

题目大意

- 维护一个初始为空的非负整数序列，支持 q 次操作
- 每次操作移除末尾的若干个整数，然后在末尾加入一个整数 v
- 每次操作后输出当前序列所有后缀和的 **XOR 和**
- 答案对 $2\,097\,152\ (=2^{21})$ 取模
- $1 \le q \le 5 \times 10^5$, $0 \le v \le 10^9$

题解

- 记当前序列为 $a_1, a_2, \ldots, a_n$
- 定义前缀和序列为 $p_i = \sum_{j=1}^{i} a_j$，其中 $p_0 = 0$
- 从 i 位置开始的后缀和为 $p_n - p_{i-1}$
- 所求即为 $\oplus_{i=1}^{n}(p_n - p_{i-1})$
- 每次操作仍然是移除末尾的若干个整数，然后在末尾加入一个整数

题解 (cont'd)

- 分别计算答案二进制表示的每一位，由于答案对 2^{21} 取模，只有低 21 位有用，假设当前在考虑第 d $(0 \le d < 21)$ 位
- 要求出有奇数还是偶数个 i 满足 $p_n - p_{i-1}$ 的二进制表示的第 d 位是 1，也就是 $(p_n - p_{i-1}) \bmod 2^{d+1} \ge 2^d$
- 带修改的场景就是支持增删数 $x = (-p_{i-1}) \bmod 2^{d+1}$，查询有多少个数 x 满足 $(p_n + x) \bmod 2^{d+1} \ge 2^d$，由于值域不大，可以直接用树状数组维护
- 时间复杂度是 $O(q \log^2 Mod + Mod)$

E - Product of Suffix Sums

题目大意

- 维护一个初始为空的非负整数序列，支持 q 次操作
- 每次操作移除末尾的若干个整数，然后在末尾加入一个整数 v
- 每次操作后输出当前序列所有后缀和的**乘积**
- 答案对 $1\,004\,535\,809$ $(= 479 \times 2^{21} + 1)$ 取模
- $1 \leq q \leq 1.3 \times 10^5$, $0 \leq v \leq 10^9$

题解

- 记当前序列为 $a_1, a_2, \ldots, a_n$
- 定义前缀和序列为 $p_i = \sum_{j=1}^{i} a_j$，其中 $p_0 = 0$
- 从 i 位置开始的后缀和为 $p_n - p_{i-1}$
- 所求即为 $\prod_{i=1}^{n}(p_n - p_{i-1})$
- 每次操作仍然是移除末尾的若干个整数，然后在末尾加入一个整数

题解 (cont'd)

- 将操作离线下来可以得到一棵树，每个节点 u 记录一个值 t_u，使得根到每个节点的路径都对应了某次操作后的前缀和序列
- 在根节点 rt 记个单位多项式 $f_{rt}(x) = 1$
- 在非根节点 u 记个一次多项式 $f_u(x) = x - t_{fa_u}$
  - 其中 fa_u 表示 u 的父亲节点

题解 (cont'd)

- 转换成对每个点 u 计算 $\prod_{v \in path_u} f_v(t_u)$
  - 其中 $path_u$ 表示从 u 到根的路径
- 也可以表示为 $\prod_{v \in path_u} f_v(x) \bmod (x - t_u)$
- 在节点 u 再记个一次多项式 $g_u(x) = x - t_u$

题解 (cont'd)

- 对操作树做重链剖分，记 u 的重儿子为 son_u，先预计算一些多项式
- 对每个节点 u 计算 $pg_u(x) = \prod_{v \in sub_u - sub_{son_u}} g_v(x)$
  - 其中 sub_u 表示以 u 为根的子树
- 对每个重链顶 top 计算 $pgt_{top}(x) = \prod_{v \in sub_{top}} g_v(x)$
- 根据重链剖分的性质，这些多项式的总次数是 $O(n \log n)$

题解 (cont'd)

- 实际计算由两个过程交替进行：
  - 将节点 u 的轻儿子（也是重链顶）top 的结果相乘再乘上 $g_u(x)$
  - 将重链的每个节点 u 的结果相乘得到重链顶 top 的结果
- 这两个过程都可以通过 (带权) 分治多项式乘法快速完成，套用全局平衡二叉树的分析，这部分时间复杂度是 $O(q \log^2 q)$

题解 (cont'd)

- 完成上述预计算后，考虑计算答案
- 对每个节点 u 计算 $pf_u(x) = \prod_{v \in path_u} f_v(x) \bmod pg_u(x)$
  - 由于 $g_u(x) \mid pg_u(x)$，可以根据 $\prod_{v \in path_u} f_v(t_u) = pf_u(t_u)$ 得到答案
- 对每个重链顶 top 计算 $\prod_{v \in path_{fa_{top}}} f_v(x) \bmod pgt_{top}(x)$
- 这些多项式的总次数仍然是 $O(n \log n)$

题解 (cont'd)

- 实际计算仍然是两个过程交替进行：
  - 将节点 u 的结果分别对每个轻儿子（也是重链顶）top 的 $pgt_{top}(x)$ 取模，这部分可以通过（带权）分治多项式取模快速完成
  - 将重链顶 top 的结果分别对重链的每个节点 u 先乘上 $\prod_{v \in path_u - path_{fa_{top}}} f_v(x)$ 再对 $pg_u(x)$ 取模，这部分需要通过改造 (带权) 分治多项式取模的过程来快速完成
- 仍然套用全局平衡二叉树的分析，这部分时间复杂度也是 $O(q \log^2 q)$
- 实际上也可以显式地建出操作树的全局平衡二叉树，基于树结构实现上述所有计算过程，时间复杂度也是一样的。

题解 (cont'd)

- 不难发现整个过程和多项式多点求值的过程非常类似，实际上就是基于操作树构建的全局平衡二叉树用作多点求值的分治树，分治递归时用到了经典多点求值算法中用到的多项式取模。
- 基于转置原理的多点求值算法则不需要多项式取模，但是直接通过转置原理分析似乎比较困难，另一个分析方式是使用形式洛朗级数，具体可以参考 A simple way to understand the transposed algo of multi-eval，常数相比需要多项式取模的做法会小一点。

F - Career Path

题目大意

- 小 Q 是打工人，当前工龄 X 年，还有 N 年退休，市面上有 M 家公司可以入职工作。
- 给出每家公司的薪酬模式，例如现金，年终奖，股票，裁员补偿，竞业补偿等等。
- 计算小 Q 剩余职业生涯的最大收益。
- $0 \leq N, M \leq 100$

题解

- $f(i,j,k,0/1)$ 表示在第 i 年结束的时候，小 Q 有 $X+j$ 年工作经验，且上一家公司是 k，离职之后是否有 gap 年，所能获得的最大收益。
- 考虑 $O(N^2M)$ 先枚举 i,j,k，表示第 i 年结束的时候，小 Q 有 $X+j$ 年工作经验，且准备入职 k 公司。那么接下来可以 $O(M)$ 枚举上一家公司 k'，如果 $k'=k$ 或者 k' 对 k 有竞业，那么只能从 $f(i,j,k',1)$ 转移，否则 $f(i,j,k',0/1)$ 都可以转移，这样就可以得到入职之前的最大可能收益值。
- 然后 $O(N)$ 枚举在新公司 k 的工作时间 z，维护好各种收益、股票、补偿等，可以均摊 $O(1)$ 计算每个 $f(i+z+0/1,j+z,k,0/1)$。一个细节是可以选择额外 gap，即从 $f(i,j,k,0/1)$ 转移到 $f(i+1,j,k,1)$，虽然基本上不优。

题解 (cont'd)

- 具体收益计算的话，现金、年终奖、赔偿金就按照题意模拟即可，股权收益的话可以用维护一个股价单调递减的栈，因为如果后面可以卖得更贵，肯定就不会在前面便宜的时候卖，然后每个元素是一个（股价，股票数量）的二元组，表示在当前股价卖了多少数量的股票。每次枚举一个新的 z，首先可以算出这一年结束可以获得多少股票，然后 pop 单调栈，同时维护股票余量和股票总收益，最后根据新股价再 push 单调栈和计算股票收益即可。
- 时间复杂度：$O(N^2M(N+M))$。

G - 3/2 Square Strings

题目大意

- 给出两个字符串 S 和 T。
- 取 S 的子串 $p = S[a, b]$，以及 T 的子串 $q = T[c, d]$，使得 $q = p + p$。
- 统计四元组 (a, b, c, d) 的个数。
- $1 \leq |S|, |T| \leq 2 \times 10^5$

题解

- 先在 T 串内找到所有本质不同平方串，并统计他们的出现个数，可以使用 *Runs* 或者 *Lyndon* 系列算法在 $O(n\log^2 n)$ 或 $O(n\log n)$ 内完成求解。
- 因为一个平方子串一定会出现在某个 $Run(l,r,p)$ 中，且长度一定是 $2p$ 的倍数，所以可以枚举所有 *Run*，枚举平方子串的长度 $2kp$，再枚举左端点 $L\in[l,l+p)$，就可以覆盖所有的平方子串，这样枚举的平方子串都可以对应到不同的本原平方子串 $[L+2(k-1)p,L+2kp-1]$，然后一个串的本原平方子串个数是 $O(n\log n)$ 的，所以上述方式枚举平方子串的复杂度是 $O(n\log n)$ 的。还有一个结论：一个串的本质不同的平方子串的个数是 $O(n)$ 的，所以可以进行进一步的去重。

题解

- 前面枚举到的平方子串的个数，去重前后分别是 $O(n\log n)$ 和 $O(n)$，接下来就是对于每个平方子串，求它的一半在 S 中的出现次数。
- 可以使用 SAM 上根据子串长度倍增定位的方法，实现 $O(n\log n)$ 预处理和 $O(\log n)$ 单次查询，也可以使用对 $S\#T$ 构造 SA，向左右二分确定范围的方法，复杂度相同。
- 整体复杂度由第一步决定，$O(n\log^2 n)$ 可以过，但卡掉了自然溢出哈希。

H - World Finals

题目大意

- 有两场 World Finals 同时举办，每场都有若干出线队伍，两场出线名单可能有重复的队伍，而一个队伍只能选择其中的一场参加，然后每场比赛的每支队伍都有一个预测成绩（过题数 + 罚时）。
- lzr010506 在两场 World Finals 都出线了，他想知道如果所有队伍的最终成绩就是预测成绩，并且可以任意分配两场都出线了队伍的比赛场次的选择，他们队最高可能的排名是多少。
- $1 \leq number\ of\ teams \leq 10^5$

题解

- 最优情况要么是 lzr010506 参加第一场，所有其他队选择第二场；要么是 lzr010506 参加第二场，所有其他队选择第一场，两者取最优即可。
- 时间复杂度：$O(number\ of\ teams)$。

I - Mirror Maze

题目大意

- 有一个 $n \times m$ 的矩形镜子迷宫，镜子有 “\, /, -, |” 四种，每种镜子有特定的光线反射方向，注意直接通过镜子的情况不算被反射。
- 有 q 个询问，每个询问给定一个点光源 (x, y, dir)，表示在 (x, y) 位置向 dir 方向发射一束光线，问经过足够的时间之后，这束光线被多少个不同的镜子反射过。
- $1 \leq n, m \leq 1000, 1 \leq q \leq 10^5$

题解

- 因为光路可逆，所以不可能有光束分叉或者汇合的情况，所以所有的光束会构成若干个环和若干条链。
- 所以我们要做的就是把这些光环和光链抽出来，然后预处理出每个点光源的答案，询问就查表 $O(1)$ 回答即可。
- 时间复杂度： $O(nm+q)$。

J - 2D Travel

题目大意

- 给一个 $n \times m$ 的二维矩形区域和一个长为 k 的指令序列，每条指令由“LRUD”中的一个字符 c 和一个整数 t 构成，表示沿着 c 方向走 t 步，其中“LRUD”分别表示向左/右/上/下走一步，如果要走出矩形区域就停着不动。有 q 个询问，每个询问给定 x, y, l, r，表示从 (x, y) 出发，依次按照第 $l, l+1, l+2, \ldots, r$ 条指令移动，问最后会停在哪个位置以及实际走了多少步。
- $1 \leq n, m, t \leq 10^9, 1 \leq k, q \leq 10^5$

题解

- 注意到两个方向是独立的，可以变成一维的来处理，接下来以 n 这一维为例来讨论。
- 如果没有卡边界的情况，那么问题就是一个简单的前缀和计算，实际走的步数就是 n 这一维的操作个数。先考虑这样一个问题：给定 x, l，从位置 x 出发，依次按照第 $l, l+1, l+2, \ldots$ 条指令移动，什么时候会走到边界位置。考虑把 L 当成 -1，R 当成 1，U 和 D 当成 0，再乘以对应的次数 t，能得到一个序列 A，要求的就是满足 $x + \sum_{i=l}^{r} A_i \leq 0$ 或者 $x + \sum_{i=l}^{r} A_i \geq n$ 的最小 r。

题解 (cont'd)

- 考虑对每个 l，维护两组前缀和的最值 $Min_{l,r} = \min_{a=l}^{r} \sum_{i=l}^{a} A_i$ 和 $Max_{l,r} = \max_{a=l}^{r} \sum_{i=l}^{a} A_i$，这样就可以二分出第一个满足 $x + Min_{l,p} \leq 0$ 或者 $x + Max_{l,p} \geq n$ 的位置 p，这个就是后面第一次走到边界时对应的操作。可以按 l 从大到小离线下来用线段树维护，也可以直接用可持久化线段树维护。具体来说，以 $Min_{l,r}$ 为例，这个最值序列显然是单调递减的，所以从 Min_l 转移到 Min_{l-1} 时的后缀加和后缀取 min 就可以先找到取 min 的分界点，然后一边是区间加，一边是区间赋值。

题解 (cont'd)

- 再对每个 l 预处理从位置 $0/n$ 出发依次按照第 $l, l+1, l+2, \ldots$ 条指令移动，什么时候又走到边界位置，以及是哪个边界。回答询问的话就是先求出走到的第一个边界，之后再用倍增快速转移到能走到的最后一个边界，最后把剩下一段的位移和算出来再加上去即可，同时用倍增以及前缀和来快速计算倍增转移和区间操作时实际走的步数。区间操作的时候可能会出现在最后一个操作的中途走到边界的情况，这时已知前面的操作不会走出边界，所以可以先无视边界去走，最后再把超出边界的部分减掉就行。
- 时间复杂度：$O((k+q)\log k)$。

K - Matching Problem

题目大意

- 给一棵无向基环树 G 和一棵无根树 H，问 H 是否为 G 的子图。多组询问。
- $3 \leq |G|, \sum |G| \leq 1000, 1 \leq |H| \leq |G|$

题解

- 一个做法是选一条 G 中的环边 (u, v) 作为关键边，有两种情况：
- H 到 G 的匹配中不包含 (u, v)：那么把这条边去掉之后问题就变成一棵树是否为另一棵树的子树了。一个做法是先变成有根树，然后求 $Dp[i][j]$ 表示在 i 跟 j 相匹配的情况下，G 中以 i 为根的子树是否可以匹配 H 中以 j 为根的子树，转移的话就基于儿子节点之间的 Dp 值做一次最大匹配就行。再求 $Up[i][j]$，表示在 $Fa[i]$ 与 j 相匹配的情况下，G 中以 i 为根的子树**以外的节点**是否可以匹配 H 中以 j 为根的子树，这个也是类似的做一次最大匹配来求解。最后再枚举 G 中的每个节点作为 H 根节点所对应的节点，用预处理的 Dp 和 Up 值做最大匹配来求是否有解。

题解 (cont'd)

- H 到 G 的匹配中包含 (u, v)：那么就考虑枚举 H 中的所有边 (a, b) 与其对应，不妨设 u 对应 a，v 对应 b，且 a 是 b 的父亲节点，v 跟 b 能否对应就看以 v 为根的情况下 $Dp[v][b]$ 的值，u 跟 a 的话可以再求一个 $Du[i][j]$，表示在 i 与 $Fa[j]$ 对应的情况下，G 中以 i 为根的子树是否可以匹配 H 中以 j 为根的子树**以外的节点**，转移也是大同小异，最后就看以 u 为根的情况下 $Du[u][b]$ 的值即可。
- 我们注意到上述做法实际上就是分别从 u 和 v 开始选了两个连通块来对应 (a, b) 的两边连通块，这里选出的连通块可能会有重叠，所以还要限制分别从 u 和 v 开始的沿环链长之和必须小于环大小。

题解 (cont'd)

- 一种可能的补丁就是让 $Dp[i][j]/Du[i][j]$ 表示可以匹配的情况下的最小沿环链长，以 $Dp[i][j]$ 为例，求解的话首先看 i 是不是在环上，不是的话就跟之前一样的算，是的话就先找到环儿子 $rson_i$，然后二分答案，把 $Dp[rson_i][son_j]$ 值在答案以内的 j 的儿子 son_j 算作与 $rson_i$ 匹配，在答案以外的算作不匹配，最后再算整体的匹配，当然也可以用最大权匹配来做。
- 上面的做法是一种为了描述问题而写的比较朴素的做法，下面来把其中的各个部分来讨论和优化。

题解 (cont'd)

- $Dp[i][j]$ 的优化：上面的做法是进行 $O(\log d_{G,i})$ 次 i 子树和 j 子树的匹配，但实际上只需要进行一次匹配，然后再按 $Dp[ring\_child][j]$ 从小到大去枚举 i 的环儿子与 j 的儿子的对应，然后问题就变成检查一组对应是否可能在最大匹配上，在残量网络上 dfs 即可。
- $Up[i][j]$ 的优化：朴素的做法是对每个 $Up[i][j]$，考虑 i 的父亲的父亲和其他儿子，实际上我们可以批量求解：做一次 fa 和 j 的匹配，对于 fa 的每个儿子 i，如果去掉它之后，fa 的父亲和剩余的儿子是能跟 j 的所有儿子全部匹配的，那么 $Up[i][j]$ 为 true，否则就是 false。这个也是可以在残量网络上 bfs 找到每个 $Up[i][j]$ 为 true 的 i。

题解 (cont'd)

- $Du[i][j]$ 的优化：在批量求解的思路上跟 $Up[i][j]$ 类似，就是做一次 i 和 fa 的匹配，然后枚举 fa 的每个儿子 j，如果去掉它之后，i 的所有儿子能与 fa 的父亲和剩余儿子全部匹配，那么 $Du[i][j]$ 至少是 true。至于具体的值，一个做法是不考虑 $ring\_child$ 做一次匹配，然后把 $ring\_child$ 涉及到的边按照 Dp/Du 值分组，对每一组都进行一次残量网络上的 bfs，那么对于每个 j，就可以根据使得 $Du[i][j]$ 不为 false 的第一组 Dp/Du 值来求 $Du[i][j]$。因为组数是 $\sqrt{|H|}$ 级别的，所以这部分的总复杂度是 $O(|G||H|\sqrt{|H|})$。另外还有基于分类讨论 +Dijkstra 的 $O(|G||H|\log|H|)$的做法，但是常数可能比较大，不一定比上面的做法快，这里就不展开了。

题解 (cont'd)

- 时间复杂度（省略大 O 符号，$d_{G,u}$ 表示 G 中 u 节点的度数，$d_{H,u}$ 同理，$F(x,y)$ 表示一边 x 个点，一边 y 个点的二分图匹配的复杂度）：

$$\sum_{i=1}^{|G|}\sum_{j=1}^{|H|}(F(d_{G,i},d_{H,j})+d_{G,i}d_{H,j}\log d_{H,j}) \tag{1}$$

$$=F(|G|,|H|)+|G||H|\log|H| \tag{2}$$

THANKS!

AC.NOWCODER.COM