

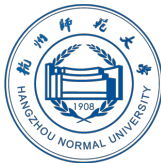
杭州师范大学第三届程序设计新生赛

题目解析

计算机科技协会

Prepared by Hangzhou Normal U ACM/ICPC Team

2020 年 12 月 13 日



- ## ② Solution

- 提交量: 4459
- 通过量: 1406
- 第一发 AC: 4 min
- 最后一发 AC: 299 min

- ## ② Solution

- 按时间顺序处理提交和询问。

- 按时间顺序处理提交和询问。
- 对于 `account`, `submitcount` 和 `teamcount` 简单维护一下就可以。

- 按时间顺序处理提交和询问。
- 对于 `account`, `submitcount` 和 `teamcount` 简单维护一下就可以。
- 对于 `teamstatus`, `minrank` 和 `maxrank` 可以每个时间点将当前每个队伍按规则排序一下，并且维护一下 `minrank` 和 `maxrank`。

- 按时间顺序处理提交和询问。
- 对于 `account`, `submitcount` 和 `teamcount` 简单维护一下就可以。
- 对于 `teamstatus`, `minrank` 和 `maxrank` 可以每个时间点将当前每个队伍按规则排序一下，并且维护一下 `minrank` 和 `maxrank`。
- 注意一下模拟细节。

Problem B BM 算日期

- 因为第二个日期算出来可能比第一个日期小。
- 先把第二个日期算出来，再比较两个日期的大小，把小的换到前面去。
- 就可以开始用 for 循环遍历年份，判断闰年。

题意

题解

- 题目中对不同的树的定义是存在某个点有不同数量的子节点，言下之意点与点之间是相同的。

题意

题解

- 题目中对不同的树的定义是存在某个点有不同数量的子节点，言下之意点与点之间是相同的。
- 按层考虑，对于每一层的节点，连向不同的上层节点就会产生不同的树。

题意

题解

- 题目中对不同的树的定义是存在某个点有不同数量的子节点，言下之意点与点之间是相同的。
- 按层考虑，对于每一层的节点，连向不同的上层节点就会产生不同的树。
- 因此我们只需要考虑上层节点每个节点分别连向了多少个下层节点。

题解

- 设上层有 x_0 个节点，下层有 x_1 个节点。

题解

- 设上层有 x_0 个节点，下层有 x_1 个节点。
- 问题就变成了把 x_1 个相同球随机分配到 x_0 个盒子里有多少分法，盒子里的球的数量可以为 0。

题解

- 设上层有 x_0 个节点，下层有 x_1 个节点。
- 问题就变成了把 x_1 个相同球随机分配到 x_0 个盒子里有多少分法，盒子球的数目可以为 0。
- 用插板法，将物品分成 x 份需要插 $x-1$ 块隔板，那么总分法就是：

$$\begin{pmatrix} x_1 + x_0 - 1 \\ x_0 - 1 \end{pmatrix}$$

Problem C BBpigeon Counting Trees

题解

- 设上层有 x_0 个节点，下层有 x_1 个节点。
- 问题就变成了把 x_1 个相同球随机分配到 x_0 个盒子里有多少分法，盒子球的数目可以为 0。
- 用插板法，将物品分成 x 份需要插 $x - 1$ 块隔板，那么总分法就是：

$$\binom{x_1 + x_0 - 1}{x_0 - 1}$$

- 根据乘法原理，只需要将每层之间的分法乘在一起即可。取模的部分参考乘法逆元和同余定理。

- 显然单个字符就是一个回文串。
- 只需要将所有字符串接起来，答案就是总的串长。

Problem E BM 充饥

- 注意「反斜杠」要用转义字符输出。

Problem F kita 买礼物

- 数据给的很水，只卡了 $\mathcal{O}(n^3)$ 次的做法。

Problem F kita 买礼物

- 数据给的很水，只卡了 $\mathcal{O}(n^3)$ 次的做法。
- 按照题目的意思：
 - 如果给你的金币数乘他的面额大于物品所需要的钱，那么就是一个完全背包。
 - 如果你的金币数乘以面额小于物品所需要的钱，那么就是一个 01 背包。

Problem F kita 买礼物

- 数据给的很水，只卡了 $\mathcal{O}(n^3)$ 次的做法。
- 按照题目的意思：
 - 如果给你的金币数乘他的面额大于物品所需要的钱，那么就是一个完全背包。
 - 如果你的金币数乘以面额小于物品所需要的钱，那么就是一个 01 背包。
- 倒是需要注意的是，直接循环 n 次的 01 背包是会 TLE 的。
- 想到用二分去取背包，就可以了。

Problem G New Game

- 关键信息为 A 赢 B, B 赢 C, 则排名为 $A \rightarrow B \rightarrow C$ 。
- 其实是一道「拓扑排序」的裸题。
- https://blog.csdn.net/dm_vincent/article/details/7714519

Problem G New Game

- 但是这题非常简单，就算没有这个前置知识，暴力两个 for 循环也能算出结果。
- 已知一些比赛结果，要给这些人排名，这里引入一个「度数」的概念。比如 A 赢 B，则让 B 的度数 +1。
- 我们则可以通过一个度数的比较，输出最后排名。

Problem G New Game

思路展开：

- 通过所有的比赛结果，记录好所有的人「初始度数」。
- 在这个基础上按从小到大的顺序去输出所有度数为 0 的人。
- 输了比赛的人才会度数 +1，最后度数为 0 的人表示没有输过，则排名靠前。
- 需要注意的是题目说明最后排名结果可能不唯一，需要从小到大输出排名。
- 比如 A 赢了 C，B 赢了 C，A 和 B 的排名并不确定，但是 A 比 B 小，则最后排名为 $A \rightarrow B \rightarrow C$ 。

Problem G New Game

思路继续展开：

- 在每次确定好一个人的排名时，对这名选手的所有「手下败将」进行度数 -1，后续则相同对所有度数为 0 的进行输出。
- 重复执行上一步，直到所有人的度数全部归为 0，则结束。
- 题目保证会有合法排名的。

Problem H Hsueh- Draw Progress

- 画一画就好了。

Problem I BM 旅游

题意：

- 输入 4 个数字，每个数字代表一个景点。
- 根据每个数的各个位之和是否等于 6 或者大于等于 16 判断是否去过这个景点。
- 若符合上述条件，则说明去过，否则没去过。

Problem I BM 旅游

两种思路：

- 使用字符串存各个数字。
- 使用 long long 类型存入各个数字（注意数字范围，会爆 int）。

Problem J 大扫除

- 引入一个访问标记 $vis[s]$ 的概念，对所有出现过的字符加上标记，每遍历到一个为访问过的字符则进行计数。

Problem K Kwords Find Kth Element

考虑给出一个 n 个数的序列 a , 怎么得到其中第 k 大的数?

- 直接排序, 然后第 k 个数就是答案, 时间复杂度 $\mathcal{O}(n \log n)$ 。
- 或者, 二分答案 res , 然后转化成判定问题:
 - 遍历序列得到小于等于 res 的数字个数 num 。
 - 如果 num 大于等于 k , 就调低上界限。
 - 否则, 增加下界限。
 - 时间复杂度 $\mathcal{O}(n \log(\max_{i=1}^n a_i))$ 。

Problem K Kwords Find Kth Element

- 现在给出 n 个序列，每次询问若干个序列 Union 起来的序列的第 k 大。
- 先将每个序列排序。这部分时间复杂度 $\mathcal{O}(n \log n)$ ，其中 n 表示所有序列的数字总数，题目保证不会超过 10^5 。
- 对于每个询问，二分答案 res ，然后对于每个指定的序列，通过二分定位出有多少个数字小于等于 res 。
- 时间复杂度为 $\mathcal{O}(m \cdot (\log \text{MaxValue}) \cdot (\log m))$ 。其中 m 表示所有询问中指定的序列个数之和，题目保证不会超过 10^5 。

Problem L 咆哮

- 这里假设群国为 A，吴国为 B。
- 求收到的伤害最小，有两种情况：
 - 先打 A，再打 B。
 - 计算出将 A 的血量降为 0 的最少回合数，然后将这回合数多出来的伤害拿来打 B，再计算后面将 B 的血量将为 0 的最小回合数。
 - 最后将总承受伤害计算出来即可。
 - 先打 B，再打 A。求承受伤害方法同上。

Problem L 咆哮

求答案本身不难，难点在于最后的字符串要字典序最小输出。

Problem L 咆哮

第一种情况，先打 A 再打 B。

- 假设好打完 A 需要最少 x 轮，则满足 $HP_A \leq \frac{(1+x) \cdot x}{2}$ ，可知经历了这 x 轮后，会有多余的伤害：

$$remain = \frac{(1+x) \cdot x}{2} - HP_A$$

- 这个时候需要把这部分多余的伤害留给 B。
- 证明：满足 $remain < x - 1$ ，所以在 $1 \sim x$ 回合里，必定可以将 $remain$ 的伤害留给 B。

Problem L 咆哮

第一种情况，先打 A 再打 B。

- 确定打倒 B 的回合数 y ，满足：

$$HP_B - remain \leq \frac{(1 + x + x + y) \cdot y}{2}$$

- 那么此策略的承受伤害是：

$$res1 = ATK_A \cdot x + ATK_B \cdot (x + y)$$

Problem L 咆哮

在此策略中，打怪的情况有两种：

- 若 $x + 1 \sim x + y$ 轮的伤害大于等于 $HP_B - remain$ ，则可以直接在前面 $1 \sim x$ 轮全部打 A，这时所确定的字符串字典序最小。
- 若不满足，则说明需要在 $1 \sim x$ 轮中选择其中一个来打 B。

Problem L 咆哮

第二种情况，先打 B 再打 A。

- 总的回合数求法同上，难点在于字符串的字典序输出。
- 假设开始打 B 的 x 回合所剩下的伤害：

$$remain = \frac{x \cdot (1 + x)}{2} - HP_B$$

- 恰好 $remain +$ 后面 y 回合 的伤害刚好能够杀死 A，这个时候就需要将 $remain$ 的伤害值全部分配给 A。
- 由于字典序中 $A < B$ 无疑，所以对这部分的伤害尽可能的靠前。

Problem L 咆哮

i.e.

- $remain = 6$, 需要在第 1、2、3 轮对 A 进行输出。
- $remain = 7$, 则需要 在第 1、2、4 轮对 A 进行输出。

Problem L 咆哮

若 $remain +$ 后面 y 回合 的伤害大于 HP_A 。这里有两种细化的情况：

- 取最大的回合数 x ，使得：

$$HP_A - y \text{ 轮的伤害值} \leq \frac{x \cdot (1 + x)}{2} \leq remain$$

- 这个时候需要将前 x 轮都用在 A 上，使构成的字符串字典序最小
- 若不满足上述条件，取 $tmp = HP_A - y$ 轮造成的伤害，将 tmp 按照最小字典序的方法全部进行分配即可。

Problem M XCPCIO Board CLI Hard

和 easy version 的不同点：

- 唯一的不同就是队伍总数 n 比较大。
- 我们发现困难点仅仅在于 `teamstatus`, `minrank` 和 `maxrank` 的计算。
- 因为在 easy version 中每个时间戳下对队伍进行排序就不适用了。

Problem M XCPCIO Board CLI Hard

- 对于每支队伍的名次，随便用个平衡树就能维护了。
- 对于 `minrank` 和 `maxrank`，是有什么牛逼的数据结构能够维护吗？

发掘性质

- 不用牛逼的数据结构。
- 考虑最小排名，只有当队伍过题的时候，队名才会变小。
- 考虑最大排名，最大排名肯定是此刻的排名，或者是之前每个过题时刻的前一刻的排名。
- 那么只需要对于每个时刻和下一个时刻过题的队伍更新一下 `minrank` 和 `maxrank` 即可。
- 更新的量级和提交量同级，不超过 10^6 。

Thank You!