The 2025 FJNU Programming Contest

Solution

FJNUACM Problem Setter Team

FJNUACM 集训队

2025 年 5 月 18 日

写在前面

这套题难度和去年差不多，题量为 14 题，因为难度原因，K2 没放入正式赛中。

难度预估

Easy: A K1 L

Easy-Mid: D F M

Mid: G J

Mid-Hard: B C E H

Hard: I K2

A. One Must Imagine Time Tight,

签到，模拟。

读取输入后输出最小值即可。

K1. MST(Most Shortened Terms)

签到，模拟。

读取输入后判断三个单词是否分别以 M, S, T 开头即可。

L. We Luv Stamina

模拟。

从下往上枚举每一行，判断底下一行左右手分别有没有键即可。

D. Where’ s My Money?

签到，模拟。

首先我们记三个人的总花费分别为 x, y, z 。

D. Where’s My Money?

解法 1

以 A 为例，实际上他只需付 $\frac{x}{3}$ 元，剩下的 $\frac{2x}{3}$ 由另外两个人 B, C 平摊，也就是说 $B \to A = C \to A = \frac{x}{3}$。这对剩下两个人都是成立的。

D. Where’s My Money?

解法 2

可以发现，剔除无意义转账后，最多出现 2 笔转账。

记 $AA = \dfrac{x+y+z}{3}$，分类讨论：

1. 只有一个人付的钱超过 AA，剩下两个人将钱打给这个人；
2. 只有一个人付的钱小于 AA，这个人将钱打给另外两个人；
3. 三人刚好 AA，不需要转账。

F. Imbalanced Bracket

构造，思维，前缀和。

首先，不能存在子序列 “()”。为此，有两种解法。

F. Imbalanced Bracket

解法 1

为了避免出现，满足条件的形式为若干连续 “)” 与若干连续 “(” 的拼接。合法构造总共有 $n+1$ 种。

可以从左到右枚举分割点的位置，简单统计计算，也可以考虑前缀和。

F. Imbalanced Bracket

解法 2

考虑让所有括号对失配，只需模拟括号配对，统计可以配对的括号数量即可。

M. The Journey Onwards...

贪心。

将所有点从小到大排序，从 0 出发每次尝试跳下一个点。

如果到下一个点的距离不大于 x，那么不用技能，直接跳到下一个点即可。否则，我们希望用了技能后，这一次能够跳得尽可能远。此时从左往右遍历，找到下一个能跳到的点并把答案加 1 即可。跳不到下一个点时无解。

G. Still No Money?

博弈论。

$x \le k$ 时先手有必胜策略：全拿光即可。后手没得拿，先手获胜。

所以 $x > k$ 时，双方一定都会尽量避免自己操作之后出现 $x \le k$ 的情况。除非 $k = 0$，否则双方总是可以通过只拿 1 根竹签，使得 x 和 k 都减少 1，依然满足 $x > k$。

因而 $x > k$ 时的胜负就取决于轮到谁操作时 k 变成 0。不难发现 k 为奇数时先手取到最后一根，k 为偶数时后手取到最后一根。

综上，先手必胜当且仅当 $x \le k$ 或 k 为奇数。

J. Spirit of Cola.

搜索。

每个杯子最多只有 301 种不同状态，三个杯子最多 $301^3 \approx 2.7 \times 10^7$ 种不同状态，而且其中很多状态是取不到的。所以可以考虑搜索。

从初始状态开始 BFS，通过题目中的两种操作进行转移即可。

B. Uchiage Hanabi

dp，单调队列优化。

令 $dp_{i,j}$ 为只考虑前 i 次烟花对答案的贡献，且放第 i 次烟花时停在点 j 时的最大总价值。

转移时考虑所有满足在时刻 t_i 时能走得到 j 的状态 $dp_{i-1,j'}$，取最大值更新答案即可。

暴力转移的单次复杂度高达 $O(n)$，无法通过。注意到能够走到 j 的位置，构成一个区间，且 j 右移一位的时候合法区间也跟着右移一位，那么转移过程就转化为经典的滑动窗口问题，用单调队列优化即可做到 $O(1)$ 转移，可以通过本题。

C. Sequential Sequence

数据结构，哈希，前缀和。

考虑这样来判断是否合法：

1. 记区间最小值为 A，最大值为 B，那么 $B - A + 1$ 等于区间长度；
2. 区间内不包含重复数字。

条件 1 为 单点改 区间查最值，可用线段树维护。

条件 2 做法很多，比如说离线后用莫队查询。此处给出一种在线的做法。

C. Sequential Sequence

由于值域很小，不妨将值域内的所有数进行随机哈希映射。

用线段树查询出区间的最大值 A 和最小值 B 后，如果 区间内所有数 的哈希值的异或和 与 数值区间 $A \sim B$ 内所有数 的哈希值的异或和 相等，那么该区间为 $A \sim B$ 内所有数字的任意排列，满足条件 2。

区间异或和可以用线段树顺便维护，或者使用树状数组。数值区间的异或和查询可用前缀和解决。

E. Binary Banter: Counting Combinatorial Bits

数论，数位 dp。

先考虑第二重求和。由 Lucas 定理（或强大的注意力）可知 $C_i^j \bmod 2 = ((i \,\&\, j) == j)$，即 i 在二进制下包含 j。问题转化为求 i 二进制表示下的子集个数。

记 $f(i)$ 为 i 的二进制表达中 1 的个数，则 $\sum_{j=0}^{i} \left(C_i^j \bmod 2 \right) = 2^{f(i)}$。

那么，原式可化简为：$\sum_{i=0}^{n} 2^{f(i)}$。

E. Binary Banter: Counting Combinatorial Bits

考虑求解子问题: $\sum_{i=0}^{2^k-1} 2^{f(i)}$ 。

对于一个 k 位二进制数，选取 $f(i)$ 位并置 1，方案数共有 $C_k^{f(i)}$ 。

转而枚举 $f(i)$，原式化为 $\sum_{i=0}^{k} C_k^i \cdot 2^i$ 。

可根据二项式定理进一步化简，得到: $\sum_{i=0}^{2^k-1} 2^{f(i)} = 3^k$ 。

E. Binary Banter: Counting Combinatorial Bits

假设二进制表示下，n 的第 k 位为 1。考虑这样的二进制数 x：第 k 位取 0，并以第 k 位为分界线，高位与 n 一致，低位任取。

此为数位 dp 的经典 trick，可将 n 拆成若干个长度为 2^k 的区间。

令 p 为高位的值，即需求: $\sum_{i=0}^{2^k-1} 2^{f(i+p)}$。由于高位和低位独立，$2^{f(i+p)} = 2^{f(i)+f(p)} = 2^{f(i)} \cdot 2^{f(p)}$。化简可得 $\sum_{i=0}^{2^k-1} 2^{f(i+p)} = 2^{f(p)} \cdot 3^k$。

那么从高位向低位枚举，记录高位 1 的数量并累计求和即可。

H. Yet Another Tree Problem

贪心，二分，前缀和。

首先把方差的表达式展开，得到 $\frac{k\sum_{i=1}^{k} b_i^2 - (\sum_{i=1}^{k} b_i)^2}{k^2}$。分式上面被减的一项是确定的，因此方差的值就只和剩余非零数的数量和剩余这些数的平方和有关。

先将给出的数从小到大排序。注意到所有的策略都可以等价为“把最小的若干个数砍成 0，然后去砍其他的数”。感性理解就是如果小的数没砍成 0 但是大的数砍成 0 了，那调整一下砍的次数，把小的数砍成 0，相当于两个数交换了一下，对最后的方差没有影响。

故考虑枚举剩余数的数量，每次贪心地砍最大的数并更新答案。

H. Yet Another Tree Problem

由于 s 很大，暴力模拟贪心过程不可行，需要优化。

考虑找到最后一个没被砍过的数。在它之前所有的数（除了被砍成 0 的）都没被砍过；在它之后所有的数一定是平均分配的（可能有余数）。由于 s 是固定的，只要找到这个位置就能唯一确定被砍过的所有数的和，从而求出它们的平方和。

预处理出 a_i 和 a_i^2 的前缀和，二分找最后一个没被砍过的位置，对前后分别计算平方和并更新答案即可。细节比较多；注意精度问题。

I. DJ Mr. Spin

计算几何，离散化，差分。

对于圆中的每一个点，可以分别算出 P 开始移动的时刻 t 的可行时间范围，需要考虑：

1. 向量 OP 足够长，可以 “碰到” 这个点；
2. 向量 OP 不会和圆相交

求出这些范围后，只需离散化后差分，就能找到可以到达最多点数的对应开始时刻。

将圆固定，视为点 P 在顺时针旋转。不妨考虑处理时刻 t 对应于向量 OP 与坐标轴的夹角，找到合适夹角后除掉 v_1 即可。

K2. MST(a.k.a. Most Stupid Technique)

最小生成树，LCA，DFS，树上差分。

显然原图的最小生成树是唯一的，不妨先把最小生成树跑出来。

注意到算法给出的是一棵 DFS 树。由于无向图上跑出来的 DFS 树是没有横叉边的，因此如果某个点为根时最小生成树对应的某条非树边是横叉边，那这个点一定不符合条件。

另一方面，如果某棵最小生成树对应的非树边全部都是返祖边，那么根据最小生成树的性质，这些非树边与树边形成的环上，非树边的边权一定是最大的，因此不可能在 DFS 的时候被遍历到。

问题等价为：给定原图的一棵生成树和若干非树边，求以哪些点为根时，所有非树边连接的两个顶点在树上都是祖先-后代关系。

K2. MST(a.k.a. Most Stupid Technique)

考虑如何求解转化后的问题。直接暴搜每一个点并检查合法性的话复杂度太大，需要优化。

任选一个点作为根，遍历每条非树边 (u, v)。不难发现满足条件的点不能夹在两个点之间。手玩一下可以发现结论：

1. 若 u, v 在树上是祖先-后代关系，不妨设 u 是 v 的祖先。令 w 是从 u 到 v 路径上的第二个点，则满足条件的是在 v 子树或不在 w 子树内的所有点；
2. 如果 u 和 v 在树上没有祖先后代关系，则满足条件的是在 u 子树或在 v 子树内的所有点。

树上差分维护所有结点的合法性，遍历完所有边之后 DFS 下去统计答案即可。

Thank you!