2021 年四川省大学生程序设计竞赛

Prepared by SUA

May 30, 2021

题意

给定正整数 k，问有多少正整数对 (x, y) 满足 $x + y = k$ 且 $1 \leq x \leq y \leq 6$。

A. Chuanpai

- x 和 y 的可行范围很小，直接枚举即可。

B. Hotpot

题意

n 个人围成一圈吃火锅。从第一个人开始轮流行动。每个人有且仅有一种喜欢的食物。轮到一个人行动时，如果锅里有他喜欢的食物，他就全吃光，他的快乐值 $+1$。否则，他就会往锅里下一些他喜欢吃的食物，并结束行动。问所有人共行动 m 次后每个人的快乐值。

B. Hotpot

- 各种食物互相独立，可以分别求解。
- 注意到行动 $2n$ 步后一定会回到初始状态。
- 对于前面完整的若干个循环，直接模拟计算一个循环内喜欢当前食物的所有人的得分即可。对于最后一段不完整的行动，单独模拟。这样对于一种食物来说，时间复杂度是和喜欢这种食物的人数线性相关的。
- 总时间复杂度 $O(n)$。

C. Triangle Pendant

题意

有一个均匀的三角形薄片 ABC，三个顶点各有一条细线连到一个点 D 上。给定三边的长度和三条细线的长度，问提起 D 让三角形自然落下到稳定状态时，三个顶点距离 D 的垂直距离。

C. Triangle Pendant

- 稳定状态下，三角形的重心位于 D 的正下方。
- 在三条线全拉直的情况下，计算三棱锥把重心旋转到 D 正下方时 A, B, C 各自的坐标即可。
- 只能拉直一条线或两条线的答案比较好计算，但是要小心判断什么时候会发生这两种情况。

D. Rock Paper Scissors

题意

两个玩家各自有合计 n 张的剪刀、石头、布的牌，每一轮玩家 1 先打出一张剩余的牌，另一名玩家再打出一张剩余的牌，双方都想最大化自己赢的局数减去输的局数。问最终后手的得分。

D. Rock Paper Scissors

- 先手的决策没有意义，后手可以决定这 $2n$ 张牌的匹配情况。
- 作为后手，只需要先尽量贪心匹配所有自己赢的 Pattern（如尽量把自己的石头匹配对方的剪刀），再匹配平局的 Pattern，再匹配输的 Pattern 即可。
- 具体证明可以借助费用流的思想。

E. Don’t Really Like How The Story Ends

题意

给定一张无向图，问最少添加多少条边，可以使得 $1, 2, \cdots, n$ 是这张图的一个合法 DFS 序列。

E. Don't Really Like How The Story Ends

- 考虑什么时候不得不加边：如果在我们 dfs 的过程中走到了 i，并且不可能在下一步走到 $i+1$，那么我们必须添加一条边使得下一步能走到 $i+1$。
- 这等价于在当前的 DFS 栈中，最深的一个有未访问邻居的点与 $i+1$ 不相邻。
- 直接模拟这个过程即可。显然不按照这个过程加边不能使得答案变得更优。

F. Direction Setting

题意

一个 n 个点 m 条边的无向图，每个点有一个权值 a_i。要求给所有边定向，令 d_i 表示定向后 i 号点的入度，要求最小化

$$D = \sum_{i=1}^{n} \max(0, d_i - a_i)$$

F. Direction Setting

- 当 $d_i < a_i$ 时，点 i 的代价是 0。之后 d_i 每加 1，付出的代价就加 1。
- (u_i, v_i) 定向后可以恰好让 u_i 和 v_i 中的一个入度加一。考虑用费用流建模来描述这个贪心的过程。费用流图的点集由源点 S，汇点 T，m 个表示原图中边的点 $A_1, \cdots, A_m$，n 个表示原图中点的点 $B_1, \cdots, B_n$ 组成。
- 对于每个 A_i，S 连接一条边到 A_i，费用为 0，容量为 1。A_i 连接两条边到 B_{u_i} 和 B_{v_i}，费用均为 0，容量均为 1。这个过程描述了一条边恰好能使 u_i 和 v_i 中的一个入度加 1。
- 对于每个 B_i，连接两条边到 T。第一条的费用为 0，容量为 a_i。第二条的费用为 1，容量为无穷大。费用流增广的过程必定是先增广第一条，再增广第二条，这符合我们对最开始代价的描述。
- 这个图的最小费用最大流的费用即为答案。
- 也可以直接在费用为 0 的边形成的子图上使用最大流算法，答案即为 $m - maxflow$。

G. Hourly Coding Problem

题意

把一个长度为 n 的整数数组分成 k 个非空连续段。令各段内部的和是 $s_1, s_2, \cdots, s_k$，最小化 $\max(s_i)$。要求输出最优方案中各段的长度，如果有多组解，输出长度序列字典序最大者。

G. Hourly Coding Problem

- 二分答案，二分值确定后合法的 k 是个区间，树状数组优化 dp 维护 [i,n] 最多最少能分成几块。
- 输出答案时前缀和值域线段树维护目前能够分成的块数合法的下标（$min_i <= k <= max_i$），每次是前缀查询最大值 + 新增/删除合法下标 (k 减小 1 后维护合法下标)
- 时间复杂度 $O(n \log n \log(\sum a_i))$。

H. Nihongo wa Muzukashii Desu

题意

给定日语的一种动词变体规则，输出读入单词的变体。

H. Nihongo wa Muzukashii Desu

- 按题意模拟即可。注意仔细读题。

I. Monster Hunter

题意

玩家的攻击力序列是 $a_1, a_2, \cdots, a_n$ 无限循环，有 m 只怪物的血量是 $h_1, h_2, \cdots, h_m$。问击败所有怪物最少需要几次攻击。

$1 \leq n, m \leq 100000, 1 \leq a_i \leq 3$。

I. Monster Hunter

- 二分答案，转变为有若干个大小为 $1, 2, 3$ 之一的物体，问能不能填满大小为 $h_1, h_2, \cdots, h_m$ 的空间（可以溢出）。
- 如果没有 3 是一个简单的贪心，只要在不溢出的情况下尽量放 2，如果还有剩余的 2 再往剩余空间为 1 的格子里放，最后再放 1 即可。
- 有 3 的情况思想类似，就是尽量避免当前的浪费和之后过程的浪费。
- 具体过程如下：先往剩余空间为大于等于 3 的奇数的位置放一个 3；再往所有大于等于 6 的格子尽量多地放两个一组的 3。（尽量保持偶数可以减小放 2 和 1 时的浪费）。此时如果恰好只有一个 3 剩余，就把它放到剩余的最大的格子里。之后依次往剩余空间为 $4, 2, 1$ 的格子放 3 即可。之后就转化为了只有 $1, 2$ 的贪心。

J. Ants

题意

长度 $L = (10^9 + 1)$ 的数轴上 N 只蚂蚁，蚂蚁互相碰到之后各自转身，数轴两端各有一块挡板，一只蚂蚁撞到之后直接转身，挡板被撞 K 次就消失，问所有蚂蚁走出去的时间。

- 在挡板消失之前，每隔 2L 的的时间所有蚂蚁会回到初始状态，每块挡板被碰撞 n 次。因此可以直接模拟最后一轮的状态，这一轮中至多只碰撞 n 次。
- 蚂蚁的相对顺序不变。同时在算下一次碰撞时间时可以认为未发生碰撞，直接穿过。
- 两块挡板都维护一个队列，维护撞到这块挡板的蚂蚁的时间序列，每次在两个队头中选择一个最小值，模拟之后塞进另一个队列即可。

K. K-skip Permutation

题意

给定 n, k，构造一个 $1-n$ 的排列 $p_1, p_2, \cdots, p_n$，使得满足 $p_i + k = p_{i+1}$ 的下标 i 最多。

K. K-skip Permutation

- 将 $MOD\ k$ 相同的值放在一起，从小到大依次输出。显然可以达到理论最大值。

L. Spicy Restaurant

题意

无向图的每个顶点有一个属性 w_i，Q 个询问，第 i 个询问给定顶点 p 和阈值 a，问距离 p 最近的 $w_i \leq a$ 的 i 距离 p 有多远。

$1 \leq w_i \leq 100$。

L. Spicy Restaurant

- w 的范围很小，直接做 100 次 BFS，计算出 $d[i][j]$ 表示离 i 最近的属性值恰好为 j 的点的距离即可。时间复杂度 $O(100(n+m)+Q)$。

M. True Story

题意

有 n 个旅客在位置 0 同时出发赶飞机，飞机的位置是 x。第 i 个人以 s_i 的速度匀速运动。初始时，飞机预定在时刻 p_0 起飞。有 m 个广播，第 i 个广播在 t_i 时刻告诉所有旅客飞机延误到了 p_i，保证 t_i 和 p_i 是递增的。每个旅客在每个时刻都会根据当前自己的位置、当前自己的速度和当前预定的起飞时间来决定行动：如果赶得上飞机就继续移动，否则就原地停留。问最终有多少个旅客赶得上飞机。

M. True Story

- 注意到 p_i 是递增的，因此一个旅客开始移动后就一定会到达终点。
- 判断一个旅客 j 是否会开始移动，相当于判断是否存在一个 i，使得 $s_j * (p_i - t_i) \geq x$。因此可以由最大的 $p_i - t_i$ 判断。
- 找到最大的 $p_i - t_i$（令 $t_0 = 0$），依次判断每一个旅客能否在 $\max_i (p_i - t_i)$ 的时间里走完 x 的距离即可。
- 时间复杂度 $O(n + m)$。

Thank you!