2020 年“联想杯”全国高校程序设计在线邀请赛题解

金轩城　任灏贇

上海理工大学

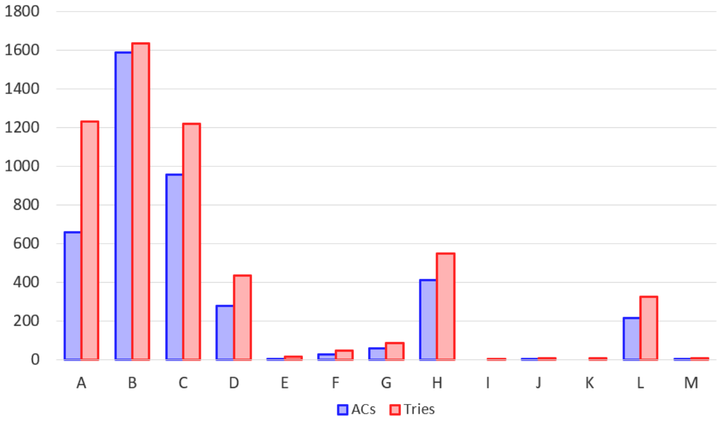
2020 年 5 月 30 日

预期难度

- Very Easy: ABC
- Easy: DHL
- Medium Easy: GEJM
- Medium Hard: FIK
- Hard: ?

实际难度

A	B	C	D	E	F	G	H	I	J	K	L	M
661	1589	958	279	6	28	59	412	0	2	0	215	3



B. Bamboo Leaf Rhapsody

- First solved: 徐本奇 (河南农业大学) 0:03 (+)
- 题意: 在三维直角坐标系内给出 n 个点，求距离原点最近的距离
- 扫一遍求 $\min\limits_{1\le i\le n}\{\sqrt{{x_i}^2+{y_i}^2+{z_i}^2}\}$ 即可
- 题目背景是真实的事件

A. Archmage

- First solved: 顾奕臻 (杭州电子科技大学) 0:04 (+)
- 题意: 大法师蓝量上限为 n ，每秒可以花费 x 释放一次技能，每秒会自动回蓝 y 点，后结算回蓝，问 m 秒内能最多放几次技能

观察

- $x + y \leq n$
- 如果有 $x \leq y$, 则他显然每秒都能释放一次技能
- 如果有 $x > y$, 则前 $m - 1$ 秒内恢复的魔法值都可以被利用上

- 所以答案是 $\min(\lfloor \frac{n+(m-1)y}{x} \rfloor, m)$
- 乘的过程可能会爆 int

C. Cheat Sheet

- First solved: Hamsterw 0:04 (+)
- 题意: 你有一张能最多写 n 个字符的纸，m 个可能重复的单词，你需要挑选一些不重复的写到纸上，单词之间必须用空格分隔，问最多能写下几个不同的单词

- 可以暴力或者使用 stl::set 保留不重复的字符串，我们只关心不重复串的长度
- 可以开个桶来存长度，也可以排序
- 先塞短的一定更优，贪心地模拟一遍即可

L. Lottery Tickets

- First solved: 张晴川 (University of Auckland) 0:23 (+1)
- 题意: 0 到 9 每个数字有 c_i 个，不要求全部用完，求能组成的不含前导 0 且能被 4 整除的最大整数。

- 枚举最后两位，从中挑个最大的即可
- 有一些细节需要处理，比如前导零、单独一个 $0, 4$ 或 8 等情况 (出题人和验题人都想到了这点但都写挂了，最神奇的是还拍上了)
- 给大家造成了不好的做题体验，非常抱歉

H. Hay Mower

- First solved: HexHexHex 0:34 (+2)
- 题意：生草题。$n \times m$ 网格图，每个格子内的草每秒增加 $a_{i,j}$，接下来 k 个操作，每个操作会在某个时间把某一列或某一行的草割光，求最终割掉的草的总和。

观察

- 每个格子的贡献只取决于它最后一次被割的时间
- 事实上只需要维护每行与每列最后一次被割的时间, 记为 r_i 和 c_j
- 那么 (i, j) 最后一次被割的时间就是 $\max(r_i, c_j)$

- 可能的坑：时间需要比大小，所以不能先取模

- First solved: 刘铭洁 (杭州电子科技大学) 0:22 (+1)
- 题意: n 点 m 边无向联通图，第 i 个点的点权是斐波那契数列第 i 项，边权是两端点点权之和，求原图的一个最小生成树使得度数最大的点的度数最小

观察

- 点权和边权过大，不能直接求出
- 斐波那契数列任意两项和都不相等，即边权互不相同
- 一个人尽皆知的性质：最小生成树的边权集合是唯一的，所以这张图的最小生成树是唯一的
- 边 (x, y) 的大小关系等价于 pair $(\max(x, y), \min(x, y))$ 的大小关系

- (Xuzhou 2018 A 既视感？)

J. JXC!!

- First solved: 蓝天朗 (浙江省衢州第二中学) 3:37 (+2)
- 题意: $n \times m$ 的 01 矩阵，你需要在不多于 2000 次询问内推出整个矩阵。有两种询问，第一种询问一个不小于 2×2 的子矩阵，交互器回答子矩阵内相邻且相等的元素对数，第二种询问矩阵单点值，不能超过 5 次。

- 观察询问小规模矩阵能得出的信息
- 考虑一个 2×3 的子矩阵 $\begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix}$
- 只需利用两次询问一得到 $\begin{bmatrix} a & b \\ d & e \end{bmatrix}$ 和 $\begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix}$ 的相同对数就可以知道中间的元素 b 和 e 相不相同
- 两者之差为奇数则相同，为偶数则不同

J. JXC!!

- $\begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix}$
- 一个更简单的做法是，询问左 2×2，得到的是 $(a = b) + (a = d) + (b = e) + (d = e)$，询问右 2×2，得到的是 $(b = c) + (b = e) + (e = f) + (b = e)$，询问整个 2×3，得到的是 $(a = d) + (b = e) + (c = f) + (a = b) + (b = c) + (d = e) + (e = f)$
- 前两式相加，减第三式，只剩一项 $b = e$。亦即，通过 2×3 内的询问，可以确定中心两个元素是否相等

- 观察角落，例如左上角，所有相关的询问中，涉及左上角的，只有 $(1,1)=(1,2)$ 和 $(1,1)=(2,1)$ 两项，而若 $(1,2)\neq(2,1)$，则无论 $(1,1)$ 如何取值，两式必一真一假
- 例如 $\begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix}$ 和 $\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$ 的相同对数均为 2 ，此时即使知道其他所有元素，左上角仍是无法获知的，因此每个角落需要花费 1 次查询 2

J. JXC!!

- 考虑推广 2×3 中的结论，每次相当于获知两个相邻点是否相等，将这些相等或不相等的信息连起来，在图中构成一棵树，只需要知道其中一个点，即可推出全图，刚好还剩余 1 次查询 2 的机会
- 分析所消耗的次数，不同的 2×3 块之间，共享对 2×2 子矩阵的询问，这部分共有 $(n-1)\times(m-1)$ 次，此外，每进行 1 次 2×3 或 3×2 的查询，都能添加一条连通相邻 2 个格子的边，除角落外，共有 $n\times m-4$ 个点需要连通，共需 $n\times m-5$ 次查询
- 总查询次数 (包含回答)，为 $5+(n-1)\times(m-1)+n\times m-5+1$ 次，对于 $n=m=32$ ，为 1986

E. Eight Digital Games

- First solved: 顾奕臻 (杭州电子科技大学) 1:25 (+2)
- 题意：一个只包含 1 到 8 的 n 长串，每个逆序对会产生相应的代价，可以花费特定的代价使所有的 x 变成 y，y 变成 x ，最小化两部分代价的和

- 所有的变换操作其实都是把一个 8 的排列映射到另一个 8 的排列上
- 数对只有 $8 \times 8 = 64$ 种，可以 $8n$ 的代价得到所有的数对出现的次数，这样我们就能快速计算变换过后逆序对产生的代价了
- 把每一次操作看做一条边，每个排列看做一个点，就得到了一个 $8!$ 个点 $\frac{8\times7\times8!}{4}$ 条边的无向图，跑单源最短路即可

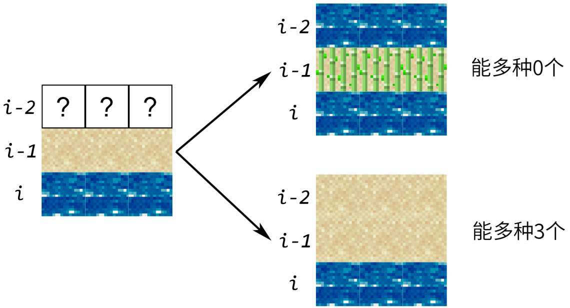
- First solved: 吉兴龙 (杭州电子科技大学) 3:37 (+)
- 题意: $n \times m$ 网格种甘蔗，沙子可以被替换成水源，石头不能被替换成水源，甘蔗能种在水源四联通域内的沙子上，问最多能种多少格甘蔗，构造一个解

观察

- m 很小，一眼状压 DP

- 第 i 行放完了水后，能新种下的甘蔗数量取决于 $i-1$ 行和 $i-2$ 行的状态
- 令 $dp_{i,j}$ 表示放完前 i 行且最后两行状态为 j 时最多能种下的数量
- 决策就是枚举第 i 行水是如何放的
- 构造解只需要记录每个状态从哪里转移过来的即可，最后从终态回溯一遍就能搞出来

- 有点抽象？为啥是和前两行有关？举个栗子



- 我们在第 i 行放水有可能能使 $i-1$ 行的某个格子能够放上甘蔗了，但如果不知道 $i-2$ 行的状态的话是没办法确定该格子是否已经被放过了的

M. MITE

- 考虑状态压缩，一个最简单的思路是，每一列的最后两行有 4 种状态，可以压成 4 进制共 4^m 种状态
- 如果你卡常能力够强应该能勉强卡进，不然你就 T 飞了...
- 如果第 $i-1$ 行是水，那我们就不关心第 $i-2$ 行是什么了，这样考虑的话每列只有 3 种状态，压成 3 进制就是 3^m 种状态
- 一共有 n 行，总状态数 $n3^m$ ，决策是 2^m 枚举放水的，单次转移代价是 $O(m)$ 的，所以总时间复杂度 $O(nm6^m)$ 大约 $4e8$ 左右
- 顾奕臻用看上去 $O(nm3^m)$ 复杂度的算法爆踩了标程

G. Gentle Jena

- First solved: 董适 (Wish) 2:08 (+9)
- 题意：动态往序列末加数字，每次添加完求序列内所有子区间的 RMQ 之和，强制在线，求一个线性的做法
- 了解笛卡尔树能在一定程度上减小思维难度，但不了解也能做

观察

- 考虑某个位置上的数 x 作为区间最小值出现的次数，一旦插入了一个比 x 大的数，x 的贡献就不会再变化了
- 当插入一个数时，只有从末尾往前单调增的数的贡献才会变化，容易想到用一个单调栈维护（其实就是笛卡尔树当前最右边的一条链）

- 假设在插入完了一个数后，单调栈中的下标为 $x_1, x_2, \cdots, x_k$，特殊地，令 $x_0 = 0$，也就是序列长这样

$$\cdots, b_{x_1}, \cdots, b_{x_2}, \cdots, b_{x_k}$$

- 可以看出，插入这个数之前，以 b_{x_i} 为最小值的子区间左端点在 $(x_{i-1}, x_i]$ 里，右端点在 $[x_i, x_k)$ 里；而插入后以它为最小值的子区间左端点取值不变，右端点变成了 $[x_i, x_k]$
- 也就是对于 b_{x_i} 来说，它对答案的贡献增加了 $b_{x_i} \times (x_i - x_{i-1})$
- 又因为之前的观察得出，不在单调栈里的值的贡献不会变化
- 所以只要维护住 $\sum_{i=1}^{k} b_{x_i} \times (x_i - x_{i-1})$ 就行，这个很容易在每次入栈出栈时在常数的时间内计算出来
- 每个数至多入栈一次、出栈一次，总时间复杂度严格 $O(n)$
- 于是就得到了一个优雅且简短的线性做法

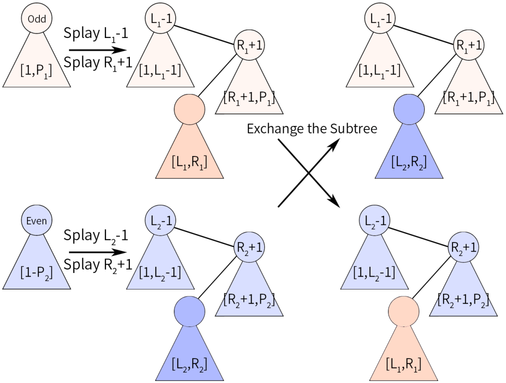
K. K-Shift Array

- 题意: n 长数组 m 个操作，每次选择一个区间，使每 $k \leq 3$ 个数为一组，循环左移一格，或者询问区间和

- k 是变化的情况下不太容易想清楚，不妨固定 $k = 2$
- 每次操作就变成了交换区间内相邻的奇数下标和偶数下标对应的数
- 开 2 棵平衡树分别维护奇数下标序列和偶数下标序列
- 操作 $[l, r]$ 等价于交换 2 棵平衡树内，下标在 $[l, r]$ 内的结点
- 用 Splay 或无旋 Treap 等能提取中序遍历上一段区间的平衡树来维护

K. K-Shift Array

- 假设奇下标平衡树内的结点重新编号为 $1, 2, \cdots, p_1$，偶下标平衡树内的结点重新编号为 $1, 2, \cdots, p_2$，对于修改操作 $[l, r]$，分别求出在奇平衡树内和偶平衡树内对应的区间 $[L_1, R_1]$ 和 $[L_2, R_2]$，像下图这样操作就行（以 Splay 为例），交换完了别忘记更新一下子树的和



K. K-Shift Array

- 对于 k 变化时的做法就呼之欲出了，我们只需要开 $\mathrm{lcm}(1,\cdots,k)$ 棵平衡树即可
- 所有的交换操作都是以 $\mathrm{lcm}(1,\cdots,k)$ 为周期的
- 对于询问区间和操作，直接把每棵平衡树在区间内的和加起来就行
- 总时间复杂度 $O(n\log\frac{n}{\mathrm{lcm}(1,\cdots,k)}+q\ \mathrm{lcm}(1,\cdots,k)\log\frac{n}{\mathrm{lcm}(1,\cdots,k)})$
- 对于无旋 Treap 而言，一个 log 应该是对的
- 对于 Splay，由于出题人太菜了，分析不来子树结构变换了之后是不是还是均摊一个 log 的
- 不过你只要写对了，两种都是能过的

F. Fake Algorithm

- First solved: 吕熠强 (华东师范大学) 0:36 (+)
- 题意：把 n 个数分成若干组，组内互质，问最少要分几组。题目给了一个贪心的假算法，你需要构造一组输入和对应的解使得你分的组数恰好比贪心的少 k ，不要求你分的也是最少的

观察

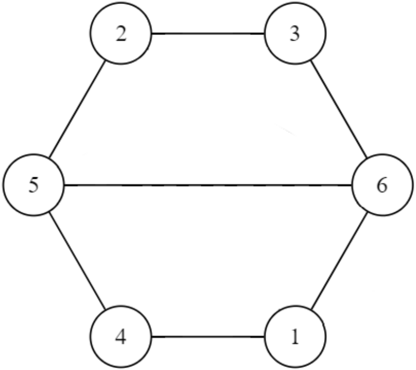
- 令每个数都对应一个顶点，如果两个数不互质就在对应顶点上连一条边
- 于是原问题被规约成了图的染色

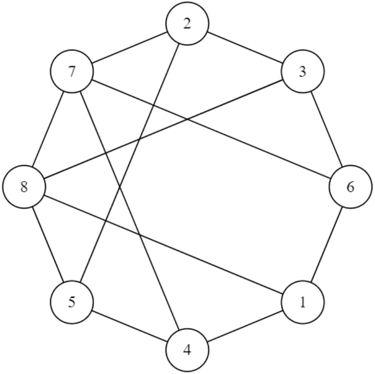
- 假设假算法得到的染色数是 x ，我们要做的就是找到一张可以被 $x-k$ 染色的图
- 往边上填不同的质数，顶点的权值是相邻边权的乘积，就能构造回来

- 如果能输出重复数字，那么复读样例 k 遍就做完了
- 可惜出题人太菜了，根本没看出来，向大家谢罪了
- 然后出题人就造了一个奇怪的 STD

F. Fake Algorithm

- 如果要求数字不能相同呢？介绍一个出题人的做法，理论上是点数最少的，虽然没有证明
- 一个比较容易想到的想法就是在二分图上不断加偶环，使之在假算法下变成 $2+k$ 染色图，可以小规模手玩然后观察规律
- 只要不乱往边上填素数，答案就不容易爆 $1e18$,k 是卡着这个边界出的





I. Immortal Trees

- 题意：数树题。n 个点，每个点有度数限制，有一些边必须选，求把它补成一个生成树的方案数

观察

- 看到度数和生成树个数，容易想到 Prufer 序列
- 但是有必选的边，直接搞搞不太定
- 不妨把每个联通块缩成一个点

- 假设有 m 条不重且不成环的边，那么 n 个点被分成了 $n-m$ 个连通块，相当于一个 $n-m-2$ 长的 Prufer 序列
- 用 DP 来计数即可

- 设 $F_{i,j}$ 表示前 i 个联通块已经在 Prufer 序列的 $n-m-2$ 个坑里占了 j 个坑的方案数
- 容易得到

$$F_{i,j+k} = \sum_{k=0}^{n-m-2-j} \binom{n-m-2-j}{k} \times F_{i-1,j} \times g(i,k+1)$$

- 其中 $g(i,k+1)$ 表示第 i 个联通块除去块内已经选上的边，往其他联通块接了 $k+1$ 条边的方案数
- 发现这也是一个类似的问题，我们再套一个 DP

- 设 $G_{i,j,k,l}$ 表示第 i 个联通块总共往外要连 j 条边，其中前 k 个点往外连了 l 条边的方案数
- 设第 i 个联通块有 sz_i 个点，其中第 j 个点为 $p_{i,j}$
- 令 L_i, R_i 表示第 i 个点去掉已经存在的边后的度数下限与上限
- 容易得到

$$G_{i,j,k,l+o} = \sum_{o=L_{p_{i,k}}}^{\min(R_{p_{i,k}},j-l)} \binom{j-l}{o} \times G_{i,j,k-1,l}$$

- 那么有 $g(i,k+1) = G_{i,k+1,sz_i,k+1}$，预处理出它然后代回第一个 DP
- 其实本质上就是个带顺序的背包计数，总时间复杂度 $O(n^4)$

- 一些可能的坑:
- 给的限制条件可能有重复，比如有重边，或者某个点的度数限制有多条
- 输入恰好是一棵完整的树，那么此时 Prufer 序列的长度就为 -1 了，实际上是 DP 不动的，所以要特判一下
- 输入的边有环，那么答案一定是 0

The End