2023 年牛客多校第十场题解

上海交通大学

2023 年 8 月 18 日

A. Mate in 33

- 首先提取出本题的核心逻辑：一个棋类游戏中存在白胜与和棋两种终局，黑白双方轮流行棋，求一个初始局面下双方对弈的结果。
- 这类问题的经典求解思路是从终局开始反过来 BFS：枚举所有黑方先走的局面，若该局面非题意规定的和棋终局（逼和或走一步吃子，造成无力可胜），则求出其所有出度（走一步可到达的局面）；若该局面是被将死的局面，则将其标记为“黑先必败，0 步结束”并放入队列中。

A. Mate in 33

此后每一步从队列头取出一个局面:

- 若该局面为“黑先必败、x 步结束”，则枚举所有可一步到达这个局面且**未被标记**的白方先走的局面，并将其标记为“白先必胜，最优步数为 $x+1$”并放入队列中。
- 若该局面为“白先必胜，最优步数为 x”，则枚举所有的可一步到达这个局面的黑方先走、且**不是和棋终局**的局面，将这个局面的出度减一。若这个局面的出度减为 0，则将这个局面标记为“黑先必败、x 步结束”，并放入队列中。

最后对于每个询问，若该局面未被 BFS 到，则必然是和棋；否则直接查询 BFS 得出的状态表即可。

A. Mate in 33

在实现时，如果直接预处理出每一个局面可以被哪些局面走一步到达并存下来，这样做的空间复杂度不可接受。注意到，在本题中，忽略“不可送王”的限制，局面 A 可一步到局面 B，等价于局面 B 可一步到局面 A。因此在枚举哪些局面 B 可达当前局面 A 时，可以从直接从局面 A 出发枚举其可一步到达的局面 B，并判断其是否满足“不可送王”等限制。

A. Mate in 33

记棋盘边长为 L，则每个局面的枚举复杂度最坏为 $\mathcal{O}(L)$ (象最多可以走到 $2L-2$ 个位置，其他棋子为 $\mathcal{O}(1)$)，局面总数为 $\mathcal{O}(L^8)$，因此总时间复杂度为 $\mathcal{O}(L^9)$。实际上由于 BFS 打表是必须的，因此只要在本地通过样例，基本上就可以通过本题。

B. Grid World

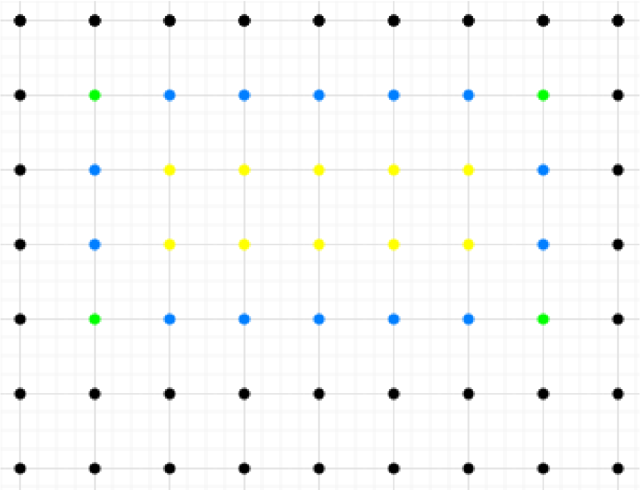
本题有多种做法，只要算法的逻辑设计合理，分类讨论完善，均可通过。不过无论具体逻辑如何，大致都需要分以下三步:

1. 找出所有的坏点形成的“洞”
2. 按照某种顺序还原出一个矩阵（可能是转置的）
3. 求出最小字典序的矩阵

B. Grid World

对于第一步，首先注意到一个最关键的限制条件：所有点度数大于等于 3，意味着坏点形成的“洞”一定是矩形。因此，算法的第一步是寻找矩形的边界。由其他几个限制条件，可以得出：所有矩形的边都是三度点（记为 A 类点），而矩形的四个角是恰好与两个三度点相邻的四度点（记为 B 类点）。除此以外的其他点，均为非边界上的四度点（记为 C 类点）

B. Grid World



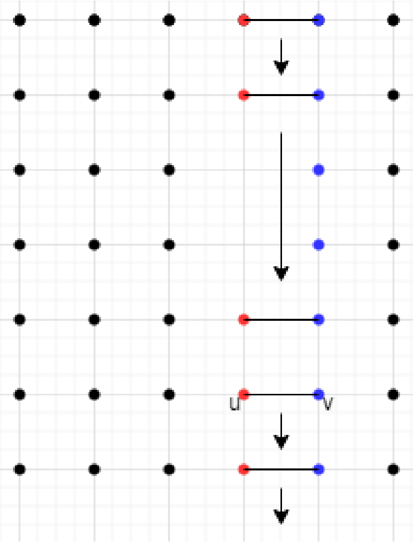
蓝色为 A 类点，绿色为 B 类点，黑色为 C 类点，黄色为坏点

B. Grid World

对于第二步，有很多种做法。出题人采取的做法如下：任意找一条边 (u, v)，将这条边沿着其垂直方向环绕一圈，即可得到两排点。具体方法是，先将当前 u 和 v 标记，然后：

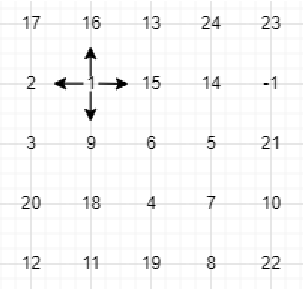
- 如果 (u, v) 位于某个洞的边缘，则找到洞的对面与之相对的边 (u', v') 并跳过去，标记中间的所有点，并执行下一步。否则直接执行下一步。
- 对于 u 的邻集 $N(u)$ 和 v 的邻集 $N(v)$，仅存在唯一的点对 (u', v') 满足：u' 与 v' 均未被标记, $u' \in N(u)$，$v' \in N(v)$, 且 u' 与 v' 相邻。将 (u, v) 赋值为 (u', v')，并标记之。
- 以上步骤中任何一步遇到了初始时标记的点，说明环绕了一圈，立即停止。

B. Grid World

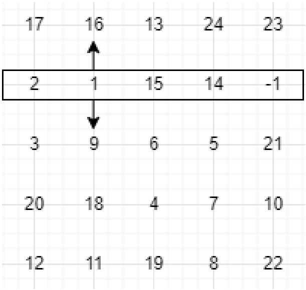


B. Grid World

对于第三步，由于要求字典序最小，因此 1 一定位于矩阵的左上角。找到矩阵中 1 的位置，然后就有四个方向可以选择第一行中的下一个数字；读取一排点以后，又有两个方向选择下一行的位置。因此，当 $r = c$ 时有 8 种情况，而当 $r \neq c$ 时有 4 种（剩下的 4 种矩阵为 $c \times r$，形状不匹配）。可以暴力求出上述若干种情况对应的矩阵，选择字典序最小且形状匹配的那个。



第一行第二个数字的四个方向



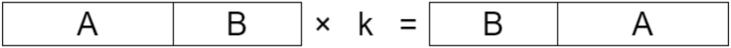
第二行的两个方向

B. Grid World

总时间复杂度为 $\mathcal{O}(rc)$，实现中使用 `std::map` 等增加一个 log 的复杂度也是可以接受的。

C. Multiplication

考虑枚举数字的位数；然后枚举前后移位的位数，这样将整个数字分成 A 和 B 两段，设左边长度为 a，右边长度为 b，两个长度均已知。



C. Multiplication

由题意可以得到式子

$$(A \cdot 10^b + B) \cdot k = B \cdot 10^a + A$$

整理得

$$\frac{A}{B} = \frac{10^a - k}{k \cdot 10^b - 1}$$

因此将右边的分式约分后，记结果为 $\frac{p}{q}$，则可以设 $A = pt$，$B = qt, t \in \mathbb{Z}$。使用整除求出最大可行的 t 即对应两侧长度分别为 a 和 b 时的答案。注意一些边界的讨论，例如前半部分相等时要比较后半部分。

C. Multiplication

本题需要使用高精度，建议使用 python 编写。由于约分时需要使用高精度除法的原因，时间复杂度上界大约为 $O(L^4 \log L)$，其中 L 是 n 的位数。当然实际上跑得非常快，只要高精度写得正确或者使用 python 均可通过。

D. Agnej

根据题意，每一行都是独立的，因此只需要求出每一层的 Grundy value （中文一般译为 SG 函数值），求出其异或和并判断是否为零即可。

D. Agnej

当 m 是偶数时，积木只有在左边和在右边两种，设左边积木数为 l, 右边积木数为 r，则不妨将 Grundy value 记作 $SG(l,\ r)$。显然

- 当该层积木总数为奇数时，抽走一个积木后就变为偶数
- 当总数为偶数时，抽走一个积木后就变为奇数，而结束条件为左右两边都只有一个积木，即 $SG(1,1)=0$

因此 Grundy value 仅与积木总数的奇偶性有关，即 $SG(l,\ r)=(l+r) \bmod 2$。

D. Agnej

当 m 是奇数时，积木有在左边、在中间和在右边两种，其中在中间的积木数只有 0 和 1 两种情况，同样设左边积木数为 l, 右边积木数为 r, Grundy value 记作 $SG(l,\ 0/1,\ r)$。稍作讨论：

- 若中间没有积木，则等价于 m 为偶数的情况，$SG(l,\ 0,\ r) = (l + r) \bmod 2$
- 否则，若左右两侧中有一边没有积木，则由于每次都只能从边上取积木直到只剩中间一个积木为止，因此 $SG(0,\ 1,\ x) = x \bmod 2$
- 否则，若左右两侧中有一边只有一块积木，不妨将此状态记为 $(1,\ 1,\ x)$。此状态可以到达以下三种状态：$(0,\ 1,\ x)$, $(1,\ 0,\ x)$, $(1,\ 1,\ x)$。注意到前二者的 Grundy value 分别为 $x \bmod 2$ 和 $(x + 1) \bmod 2$，必然为一个 0 和一个 1，由此即得 $SG(1,\ 1,\ x) = 2 + (x + 1) \bmod 2$
- 否则，基于第三条可以发现 $SG(2,\ 1,\ x) = (x + 1) \bmod 2$，进而 $SG(l,\ 1,\ r) = (l + r + 1) \bmod 2$

D. Agnej

上述规律可以由直接推导得出，也可以先打表找规律得出。总时间复杂度为 $\mathcal{O}(nm)$。

E. Geometry Problem

本题实际上需要求出圆的凸包，然后在凸包上旋转卡壳求解最终答案。

E. Geometry Problem

- 求圆的凸包有若干论文阐述做法，其中最容易实现的是分治法。
- 考虑一个单位向量 $(\cos t, \sin t)$，求出每个圆 $(x_i,\ y_i,\ r_i)$ 在该单位向量上的最大有向投影为 $f_i(t) = x_i \cos t + y_i \sin t + r$。
- 于是所有圆的凸包在极角为 t 时的投影为 $max_i \quad f_i(t)$。也就是说，我们只需求出 n 个函数 $f_i(t)$ 在 $[0, 2\pi)$ 中的上凸壳，就等价于求出了所有圆的凸包。
- 求这些函数的上凸壳可以采用分治法。

E. Geometry Problem

本题中，求出上凸壳后不必在几何上还原出圆的凸包再旋转卡壳。只需在求出函数上凸壳后，在上面扫描线即可求出答案，扫描线的过程本质上就是模拟了旋转卡壳。时间复杂度为 $O(n\log n)$。

F. IUPC

- 经典的状态压缩动态规划问题，不过直接暴力状压无法通过
- 考虑优化，可以发现在状态中无需记录已经通过了哪些题
- 记 $f(t, p, S)$ 表示时间为第 t 分钟，已经此前通过了 p 道题，且最近 k 分钟的过题状态为 S 时的合法方案数。关键点在于，此时如果又过了一道题，其具体是哪道题我们并不关心，因为剩下的 $n-p$ 道题是对称的，转移时乘以 $n-p$ 即可
- 时间复杂度 $O(tn \cdot 2^k)$

G. Palindrome Subsequence

对于最大字典序的问题，一个经典的思路是贪心。在本题中可以想到一个大致的思路：首先通过 DP 求出每一个区间 $[l, r]$ 是否能生成一个回文子序列，据此从前往后逐个贪心取尽可能大的字符。

G. Palindrome Subsequence

记 $f(l,r)$ 表示区间 $[l,r]$ 能否生成一个回文子序列，$nxt(x)$ 表示 x 后面第一个固定位置，$lst(x)$ 表示 x 前面第一个固定位置。考虑到此时我们只关注可行性，因此每次匹配的目的都是减少固定位置的限制。因此 $f(l,r)$ 有如下几种可能的转移方式:

- 用 $s_{nxt(l)}$ 匹配一个尽可能靠右的位置，减少左边一个固定位置的限制。
- 用 $s_{lst(r)}$ 匹配一个尽可能靠左的位置，减少右边一个固定位置的限制。
- 若 $s_{nxt(l)} = s_{lst(r)}$，则这二者匹配，减少两个固定位置的限制。

G. Palindrome Subsequence

在 DP 结束后可以进行贪心。在贪心中同样需要注意，每一步既可以取最靠外的某个字符，也可以取最靠外的固定位置。因此，长度为 l 的回文串，有可能对应于 l^2 种区间，我们无法判断其中哪个区间是最优的。

G. Palindrome Subsequence

因此本题时间复杂度为 $O(n^3)$，当然常数是非常小的。本题时限 7s 甚至允许部分 $O(10 \times n^3)$ 的解法通过。

H. Differential Equation

- 题目中提到了 Derivative Equations（微分方程），可以尝试用微分方程去解决这个问题。
- 我们可以把 $F_k(x)$ 看成一个序列 $F_0(x), F_1(x)\, F_2(x), \ldots$，那么定义这个序列的指数生成函数

$$H(z, x) = \sum_{k \geq 0} F_k(x) \frac{z^k}{k!}$$

H. Differential Equation

- 稍微改写一下原来的递推式

$$F_{k+1}(x) = (x^2 - 1)F'_k(x)$$

- 可以发现左边实际上就相当于我们对 z 求一个偏导:

$$\frac{\partial H(z,x)}{\partial z} = \sum_{k\geq 0} F_{k+1}(x)\frac{z^k}{k!}$$

- 而 $H(z,x)$ 对 x 求偏导的话自然就对应原来的 $F'_k(x)$，所以递推式等价于

$$\frac{\partial H(z,x)}{\partial z} = (x^2 - 1)\frac{\partial H(z,x)}{\partial x}$$

边界条件是 $H(0,x) = x$

H. Differential Equation

解方程得

$$H(z,x)=\frac{x\cosh z-\sinh z}{\cosh z-x\sinh z}$$

其中 sinh 和 cosh 分别代表双曲正弦、双曲余弦函数，定义为

$$\sinh z=\frac{e^z-e^{-z}}{2},\quad \cosh z=\frac{e^z+e^{-z}}{2}$$

如果 x 取某个值 x_0 的话，可以发现这时 $H(z,x_0)$ 就变成了一个关于 z 的形式幂级数，因此只需要做一个多项式求逆即可。可以发现无论 x 取什么值分母的常数项都是 1，所以不用担心特判的问题。

I. Hello

- 首先我们只知道输入的点一定在树上，但是并不知道这个点具体在哪条边上，距离端点的距离是多少。
- 注意到一个很重要的性质是保证所有边互不相交，所以如果用一个扫描线按 x 从小到大扫过去的话，y 的大小关系是不会变的。因此可以在扫描线的同时用一个 set 维护所有的边，这样就可以在 $O((n+m)\log n)$ 的时间里确定所有点的位置了。
- 判断边的大小关系时需要特别注意讨论竖直边的情况。

I. Hello

- 求出点在哪条边上后，可以将新点加进树中，重新建树。
- 此时问题转换为，给定一棵树和若干路径，求每条路径于多少条其他路径相交。
- 任选一个点作为根节点。两条路径 (a, b) 和 (c, d) 相交当且仅当 $lca(a, b)$ 在路径 (c, d) 上，或者 $lca(c, d)$ 在路径 (a, b) 上。
- 因此离线后树上差分即可。总时间复杂度为 $O((n+m)\log n)$，如果使用有理数类可能增加一个 log，不过同样可以通过。

J. Pumping

- 本题来源于正则语言中的泵引理（Pumping Lemma）
- 问题等价于找一条从 1 号点到某个终点的路径，满足中间有一个环（分割方案为令 y 为环的字符串，x 和 z 分别为将 y 提取出来后剩下的两段）。

J. Pumping

- 可以使用 Tarjan 算法求强连通分量求出每个点是否在某个简单环上，然后一次 DFS 求出一条从 1 号点到某个终点且经过一个在环上的点的路径，最后再 DFS 求出这个简单环。
- 此外，也可以先用 floodfill 求出所有从 1 号点可达的点集 S。在 S 的诱导子图上，用拓扑排序求出一些环上点，这些环上点满足 1 号点到这些点的所有路径上均没有环，可以证明只用这些环上点对于寻找答案是足够的。然后再用上述 DFS 求出答案。
- 当然还有很多其他类似的做法，此处不一一列举。实现时需要注意自环等细节。时间复杂度为 $\mathcal{O}(n+m)$。

K. First Last

注意讨论 $n = 1$ 的情况，直接输出

$$\left(\frac{\min\{n, 2\}}{n}\right)^m$$

即可。

L. Grayscale Confusion

题意：给定 n 个三元组 (r_i, g_i, b_i)。构造一个长度为 n 的数组 w，使得

- $w_1 = w_2$
- 对于任意 i，j，若 $r_i > r_j$，$g_i > g_j$，$b_i > b_j$，则 $w_i > w_j$

输出任意合法构造均可。

L. Grayscale Confusion

本题数据范围较小，可以直接建图后，采用广度优先搜索的拓扑排序输出拓扑序，此时复杂度为 $O(n^2)$。但是这不是正解。

L. Grayscale Confusion

- 考虑构造一个向量 (x, y, z) 满足 $x, y, z > 0$，$x + y + z = 1$，用点积表示最后的结果 $w_i = \lfloor (r_i, g_i, b_i) \cdot (x, y, z) \rfloor$，其中 $\lfloor\rfloor$ 下取整；换任意一种一致的取整方式亦可。
- 任意满足上述条件的向量，最终结果必然符合题意的偏序要求：每维至少差 1 则带权平均数至少差 1。
- 因此，若一开始对应两种颜色已经存在偏序则必然无解。否则可以证明，必然存在一个向量 (x, y, z) 满足 $(x, y, z) \cdot (r_1, g_1, b_1) = (x, y, z) \cdot (r_2, g_2, b_2)$，并且 x, y, z 中至少有一个为 0。
- 构造出来的向量可以为分母很小的有理数，因此不会有浮点数问题。
- 时间复杂度为 $O(n)$。该构造同样可以证明拓扑排序的正确性。

M. Fair Equation

- 按照题意处理字符串模拟即可。
- 如果使用 Python 甚至可以直接调用 `eval` 函数。

Thanks!