

A O L O E O M OO H O F O G OO I O D O C OOO J OOOO B OOOOO K OOOO

郑州轻工业大学“筑梯杯” 第十七届程序设计大赛暨省内高校邀请赛 解题报告

郑州轻工业大学



2025 年 4 月 6 日

致谢

感谢 **Dorothy__**、**Zinc-acetate**、**Ice_teapoy**、**Zero_L**、**Cai_Guang**、**HaPpY1213**、**WaO.o** 与 **Caiki** 为本场比赛验题并提供建议。

难度分布

预估难度分布：

- Easy: AL
- Easy-Medium: MHFG
- Medium: EID
- Medium-Hard: CJ
- Hard: BK

封榜前过题分布：

- Easy: AL
- Easy-Medium: EHGM
- Medium: F
- Medium-Hard: DJ
- Hard: BCJK

注：题解 M 题为牛客同步赛 B 题（超椭圆）。

Problem A - No idea

题目大意

- 给定一个含小写字母和数字的字符串，将字符串中所有 2024 替换为 2025。
- 关键词：模拟

Problem A - No idea

题目大意

- 给定一个含小写字母和数字的字符串，将字符串中所有 2024 替换为 2025。
 - 关键词：模拟
-
- 直接模拟即可，注意不要单独将 24 变为 25 。

Problem L - 割方术

题目大意

- 给你一个左上角被切掉一块区域的矩形，现在用边长为 k 的正方形填满这个剩余图形，令 k 为何值时，所需的正方形个数最小？
- 关键词：gcd、贪心

Problem L - 割方术

题目大意

- 给你一個左上角被切掉一塊區域的矩形，現在用邊長為 k 的正方形填滿這個剩餘圖形，令 k 為何值時，所需的正方形個數最小？
 - 关键词：gcd、贪心
-
- 對於剩下的 L 型範圍，想要正方形個數最小，即要求 k 取值最大，且要 $n, m, N - n, M - m$ 都是 k 的倍數；

Problem L - 割方术

题目大意

- 给你一個左上角被切掉一塊區域的矩形，現在用邊長為 k 的正方形填滿這個剩餘圖形，令 k 為何值時，所需的正方形個數最小？
- 关键词：gcd、贪心
- 对于剩下的 L 型范围，想要正方形个数最小，即要求 k 取值最大，且要 $n, m, N - n, M - m$ 都是 k 的倍数；
- 因此答案即为 $\gcd(n, m, N - n, M - m) = \gcd(n, m, N, M)$ 。

Problem E - 小刻与奇怪队列

题目大意

- 请你实现一种队列，满足从两端入队，从中间处出队。并根据给定的多次操作进行模拟。
- 关键词：双端队列

Problem E - 小刻与奇怪队列

题目大意

- 请你实现一种队列，满足从两端入队，从中间处出队。并根据给定的多次操作进行模拟。
- 关键词：双端队列
- 类似对顶堆的思想，动态维护两个双端队列 $q1, q2$ ，使得两队列的长度满足 $\text{len}_1 = \text{len}_2 + 1$ 或 $\text{len}_1 = \text{len}_2$ 。

Problem E - 小刻与奇怪队列

题目大意

- 请你实现一种队列，满足从两端入队，从中间处出队。并根据给定的多次操作进行模拟。
 - 关键词：双端队列
-
- 类似对顶堆的思想，动态维护两个双端队列 $q1, q2$ ，使得两队列的长度满足 $\text{len}_1 = \text{len}_2 + 1$ 或 $\text{len}_1 = \text{len}_2$ 。
 - 此时 $q1$ 与 $q2$ 拼接起来的队列即为满足要求的队列。实现两端入队即从 $q1$ 队头入队，从 $q2$ 队尾入队。实现从中间处出队即要从 $q1$ 队尾处出队。

Problem M - 超椭圆

题目大意

- 给定一个超椭圆的方程，求图形的面积。
- 关键词：数学

Problem M - 超椭圆

题目大意

- 给定一个超椭圆的方程，求图形的面积。
- 关键词：数学
- 本题数据范围较小，可以直接利用微分的思想，将图形分割成很多个小梯形/矩形。

Problem M - 超椭圆

题目大意

- 给定一个超椭圆的方程，求图形的面积。
 - 关键词：数学
-
- 本题数据范围较小，可以直接利用微分的思想，将图形分割成很多个小梯形/矩形。
 - 考虑到超椭圆在四个象限是完全对称的，因此我们可以求出第一象限的面积。

Problem M - 超椭圆

题目大意

- 给定一个超椭圆的方程，求图形的面积。
 - 关键词：数学
-
- 本题数据范围较小，可以直接利用微分的思想，将图形分割成很多个小梯形/矩形。
 - 考虑到超椭圆在四个象限是完全对称的，因此我们可以求出第一象限的面积。
 - 在第一象限内，椭圆方程有如下化简： $y = b \times \sqrt[n]{1 - (\frac{x}{a})^n}$

Problem M - 超椭圆

题目大意

- 给定一个超椭圆的方程，求图形的面积。
 - 关键词：数学
-
- 本题数据范围较小，可以直接利用微分的思想，将图形分割成很多个小梯形/矩形。
 - 考虑到超椭圆在四个象限是完全对称的，因此我们可以求出第一象限的面积。
 - 在第一象限内，椭圆方程有如下化简： $y = b \times \sqrt[n]{1 - (\frac{x}{a})^n}$
 - 因此我们可以模拟微分，令每一步长为 eps，以 eps 为底，再在该步范围内取一点的函数值作为高，对所有求得的小矩形面积求和即可模拟答案。当 eps 在 10^{-6} 左右即可通过本题。

Problem M - 超椭圆

- 此外，超椭圆还可以根据伽马函数得出精确解：

- $$S = 4ab \cdot \frac{\Gamma(1 + \frac{1}{n})^2}{\Gamma(1 + \frac{2}{n})}$$

- 对于 C/C++，可以使用 `tgamma` 函数得出答案。

Problem H - ARK no NIGHTS

题目大意

- 给定两个日期与时间，判断时间差是否超过 8 小时。
- 关键词：思维

Problem H - ARK no NIGHTS

题目大意

- 给定两个日期与时间，判断时间差是否超过 8 小时。
 - 关键词：思维
-
- 首先，将时间转换为秒数形式： $t = 3600 \cdot h + 60 \cdot m + s$;

Problem H - ARK no NIGHTS

题目大意

- 给定两个日期与时间，判断时间差是否超过 8 小时。
 - 关键词：思维
-
- 首先，将时间转换为秒数形式： $t = 3600 \cdot h + 60 \cdot m + s$;
 - 如果两个日期相同，直接判断 $t_b - t_a \geq 8 \times 3600$ 是否成立；

Problem H - ARK no NIGHTS

题目大意

- 给定两个日期与时间，判断时间差是否超过 8 小时。
 - 关键词：思维
-
- 首先，将时间转换为秒数形式： $t = 3600 \cdot h + 60 \cdot m + s$;
 - 如果两个日期相同，直接判断 $t_b - t_a \geq 8 \times 3600$ 是否成立;
 - 如果两个日期相邻，可以令 $t_b = t_b + 24 \times 3600$ 然后判断;

Problem H - ARK no NIGHTS

题目大意

- 给定两个日期与时间，判断时间差是否超过 8 小时。
 - 关键词：思维
-
- 首先，将时间转换为秒数形式： $t = 3600 \cdot h + 60 \cdot m + s$;
 - 如果两个日期相同，直接判断 $t_b - t_a \geq 8 \times 3600$ 是否成立;
 - 如果两个日期相邻，可以令 $t_b = t_b + 24 \times 3600$ 然后判断;
 - 如果两个日期不相邻，说明相隔一定在 24 小时以上。

Problem H - ARK no NIGHTS

题目大意

- 给定两个日期与时间，判断时间差是否超过 8 小时。
 - 关键词：思维
-
- 首先，将时间转换为秒数形式： $t = 3600 \cdot h + 60 \cdot m + s$;
 - 如果两个日期相同，直接判断 $t_b - t_a \geq 8 \times 3600$ 是否成立;
 - 如果两个日期相邻，可以令 $t_b = t_b + 24 \times 3600$ 然后判断;
 - 如果两个日期不相邻，说明相隔一定在 24 小时以上。
 - 另外，也可以通过调用 `mktime` 函数或者手动转换成时间戳来解决。

Problem F - 你好多宝宝，你开幼儿园算了

题目大意

- 给你一棵树，请你任意连接一条边（不能出现重边与自环），使得连接后的基环树的最小生成树权值和最小。
- 关键词：贪心、构造

Problem F - 你好多宝宝，你开幼儿园算了

题目大意

- 给你一棵树，请你任意连接一条边（不能出现重边与自环），使得连接后的基环树的最小生成树权值和最小。
- 关键词：贪心、构造
- 可以证明新加入的边一定是边权为 0 时最优，于是我们考虑在何处链接边。

Problem F - 你好多宝宝，你开幼儿园算了

题目大意

- 给你一棵树，请你任意连接一条边（不能出现重边与自环），使得连接后的基环树的最小生成树权值和最小。
 - 关键词：贪心、构造
-
- 可以证明新加入的边一定是边权为 0 时最优，于是我们考虑在何处链接边。
 - 对于一棵基环树，只要去掉它环上的任意一条边，都是该基环树的一棵生成树。于是我们考虑让原树的最大边在基环树的环上，则该基环树的最小生成树的总权值和一定为原树权值和与最大边权的差，此时最小生成树的权值和最小。

Problem G - 数组笑传之查查表

题目大意

- 给定两个数组 a, b ，求每一个 a_i 的前 b_i 个比它小的元素下标集合和后 b_i 个比他小的元素下标集合的交集大小。
- 关键词：排序

Problem G - 数组笑传之查查表

题目大意

- 给定两个数组 a, b ，求每一个 a_i 的前 b_i 个比它小的元素下标集合和后 b_i 个比他小的元素下标集合的交集大小。
- 关键词：排序
- 对于每个 a_i ，小于它的数的数量是固定的，并且题目只要求我们找到集合大小，而不是交集元素。原题意等价于对于一个固定大小的线段，前 b_i 的长度和后 b_i 的长度的重叠部分。假设小于 a_i 的元素数量为 k ，则 ans_i 即为 $\max(0, 2 \times b_i - k)$ 。

Problem G - 数组笑传之查查表

题目大意

- 给定两个数组 a, b ，求每一个 a_i 的前 b_i 个比它小的元素下标集合和后 b_i 个比他小的元素下标集合的交集大小。
- 关键词：排序
- 对于每个 a_i ，小于它的数的数量是固定的，并且题目只要求我们找到集合大小，而不是交集元素。原题意等价于对于一个固定大小的线段，前 b_i 的长度和后 b_i 的长度的重叠部分。假设小于 a_i 的元素数量为 k ，则 ans_i 即为 $\max(0, 2 \times b_i - k)$ 。
- 对原数组排序一下，记录每个数在原下标中的位置，找到 k 即可。

Problem G - 数组笑传之查查表

- 数据结构解法:
- 线段树 + 二分



Problem I - #define int bigint

题目大意

- 给定 $a = p_1^{e_1} \times p_2^{e_2} \times \cdots \times p_n^{e_n}$ 和 b , 求 $\lfloor \frac{a}{b} \rfloor$ 对 $10^9 + 7$ 取模后的结果。
- 关键词: 逆元、快速幂

Problem I - #define int bigint

题目大意

- 给定 $a = p_1^{e_1} \times p_2^{e_2} \times \cdots \times p_n^{e_n}$ 和 b , 求 $\lfloor \frac{a}{b} \rfloor$ 对 $10^9 + 7$ 取模后的结果。
 - 关键词: 逆元、快速幂
-
- 先不考虑 a 的计算, 单独考虑对于一个大整数 x , $\lfloor \frac{x}{b} \rfloor$ 对 $10^9 + 7$ 取模后如何处理。

Problem I - #define int bigint

题目大意

- 给定 $a = p_1^{e_1} \times p_2^{e_2} \times \cdots \times p_n^{e_n}$ 和 b , 求 $\lfloor \frac{a}{b} \rfloor$ 对 $10^9 + 7$ 取模后的结果。
 - 关键词: 逆元、快速幂
-
- 先不考虑 a 的计算, 单独考虑对于一个大整数 x , $\lfloor \frac{x}{b} \rfloor$ 对 $10^9 + 7$ 取模后如何处理。
 - 令 $P = 10^9 + 7$, 有
$$\lfloor \frac{x}{b} \rfloor \bmod P = (x \bmod P - x \bmod b) \bmod P \times \text{inv}(b)$$

Problem I - #define int bigint

题目大意

- 给定 $a = p_1^{e_1} \times p_2^{e_2} \times \cdots \times p_n^{e_n}$ 和 b , 求 $\lfloor \frac{a}{b} \rfloor$ 对 $10^9 + 7$ 取模后的结果。
 - 关键词: 逆元、快速幂
-
- 先不考虑 a 的计算, 单独考虑对于一个大整数 x , $\lfloor \frac{x}{b} \rfloor$ 对 $10^9 + 7$ 取模后如何处理。
 - 令 $P = 10^9 + 7$, 有
$$\lfloor \frac{x}{b} \rfloor \bmod P = (x \bmod P - x \bmod b) \bmod P \times \text{inv}(b)$$
 - 因此只需要得到 $a \bmod P$ 与 $a \bmod b$ 的值即可, 在用快速幂模拟模运算时计算这两个值即可。

Problem D - 双影奇境

题目大意

- 给定一张地图、两个起点和一个终点，判断两人是否能同时到达终点。
- 关键词：思维、并查集、搜索

Problem D - 双影奇境

题目大意

- 给定一张地图、两个起点和一个终点，判断两人是否能同时到达终点。
- 关键词：思维、并查集、搜索
- 首先使用并查集或 BFS 进行地图的连通块预处理，找出每个位置所属的连通块。

Problem D - 双影奇境

题目大意

- 给定一张地图、两个起点和一个终点，判断两人是否能同时到达终点。
- 关键词：思维、并查集、搜索
- 首先使用并查集或 BFS 进行地图的连通块预处理，找出每个位置所属的连通块。
- 如果两个起点和终点都位于同一个连通块内，那么两人才能都到达终点。

Problem D - 双影奇境

题目大意

- 给定一张地图、两个起点和一个终点，判断两人是否能同时到达终点。
- 关键词：思维、并查集、搜索
- 首先使用并查集或 BFS 进行地图的连通块预处理，找出每个位置所属的连通块。
- 如果两个起点和终点都位于同一个连通块内，那么两人才能都到达终点。
- 接着，判断两人的起点到终点的曼哈顿距离的奇偶性是否相同，只有当两者奇偶性一致时，两人才有可能同时到达终点。

Problem D - 双影奇境

题目大意

- 给定一张地图、两个起点和一个终点，判断两人是否能同时到达终点。
- 关键词：思维、并查集、搜索
- 首先使用并查集或 BFS 进行地图的连通块预处理，找出每个位置所属的连通块。
- 如果两个起点和终点都位于同一个连通块内，那么两人才能都到达终点。
- 接着，判断两人的起点到终点的曼哈顿距离的奇偶性是否相同，只有当两者奇偶性一致时，两人才有可能同时到达终点。
- 特殊情况：若某一人的第一步只能移动到终点，而另一人的距离超过 1，则两人无法同时到达终点。

Problem C - 简单题

题目大意

- 给定一个由 n 个正整数组成的序列 a ，求其所有连续子序列中按字典序排序后的第 k 大的连续子序列。
- 关键词：离散化、枚举

Problem C - 简单题

题目大意

- 给定一个由 n 个正整数组成的序列 a ，求其所有连续子序列中按字典序排序后的第 k 大的连续子序列。
- 关键词：离散化、枚举
- 通过观察可以发现，对于首元素不同的连续子序列之间，首元素较小的序列小于首元素较大的序列，由此规律我们可以先从小到大枚举答案 ans 的首元素是多少。

Problem C - 简单题

题目大意

- 给定一个由 n 个正整数组成的序列 a ，求其所有连续子序列中按字典序排序后的第 k 大的连续子序列。
- 关键词：离散化、枚举
- 通过观察可以发现，对于首元素不同的连续子序列之间，首元素较小的序列小于首元素较大的序列，由此规律我们可以先从小到大枚举答案 ans 的首元素是多少。
- 枚举之前我们先求第 k 大转化成求第 p 小，其中 $p = \frac{n \times (n+1)}{2} - k + 1$ ，我们从小到大枚举所有元素的集合。

Problem C - 简单题

- 当枚举到某一元素 x 时可以先求出该元素为首元素的连续子序列有 res 个，如果 $res < p$ 则让 $p = p - res$ ，若 $res \geq p$ 则可以确定 ans 的首个元素是当前枚举到的这个元素并让 $p = p - count$ ，其中 $count$ 为连续子序列 $[x]$ 的数量。

Problem C - 简单题

- 当枚举到某一元素 x 时可以先求出该元素为首元素的连续子序列有 res 个，如果 $res < p$ 则让 $p = p - res$ ，若 $res \geq p$ 则可以确定 ans 的首个元素是当前枚举到的这个元素并让 $p = p - count$ ，其中 $count$ 为连续子序列 $[x]$ 的数量。
- 首元素找到后，我们从小到大枚举首元素后面紧接着的第一个元素组成的集合，然后通过上面枚举首元素的方式，就可以枚举 ans 的第二个元素，以此类推……当 $p \leq 0$ 时，就代表 ans 的所有元素已经被枚举完了。

Problem C - 简单题

- 由于元素的值域为 $[1, 10^9]$, 我们想要从小到大枚举元素时就需要对元素进行排序, 如果使用 sort 进行排序的话, 会导致时间复杂度较大, 这里我们可以先在枚举之间将所有的元素进行离散化, 这样我们就能将元素的值域变成 $[1, 5000]$ 左右, 这时我们就能直接利用桶排对元素进行排序时间复杂度是线性的。
- 时间复杂度 $O(n^2)$
- 空间复杂度 $O(n)$ 。

Problem J - C 属性大爆发

题目大意

- 给定一个足球场地，包含了墙壁、球门、障碍物和足球，我们需要根据干员踢球的方向来推算足球的最终位置。
- 关键词：模拟

Problem J - C 属性大爆发

题目大意

- 给定一个足球场地，包含了墙壁、球门、障碍物和足球，我们需要根据干员踢球的方向来推算足球的最终位置。
- 关键词：模拟

主要思路：

- ① 读取场地和干员信息；
- ② 根据干员的方向计算足球的合力方向；
- ③ 模拟足球的运动，考虑反弹、阻挡等情况；
- ④ 判断足球是否进入球门。

Problem J - C 属性大爆发

输入处理：

- ① 读取场地信息，标记足球的初始坐标；
- ② 给场地的上下界各添加一行墙壁 #；
- ③ 给场地的左右界各添加一列空地 .，作为球门；
- ④ 根据干员的踢球方向计算足球的合力方向，使用 dx 与 dy 变量进行维护；
- ⑤ 如果某对相对方向均有障碍物，则清空该方向的变量，使其不再影响运动。

Problem J - C 属性大爆发

每一时刻，足球的运动过程包括以下几个步骤：

- ① 根据足球当前位置 (x, y) 计算新位置 $(x + dx, y + dy)$;
- ② 若 $(x + dx, y)$ 与 $(x, y + dy)$ 都为障碍物，则反向弹回;
- ③ 若 $(x + dx, y)$ 与 $(x, y + dy)$ 其中一个是障碍物，则进行镜像反弹;
- ④ 若 $(x + dx, y + dy)$ 为障碍物，则反向弹回;
- ⑤ 否则，足球沿合力方向向前移动到新坐标，并增加移动次数;
- ⑥ 特殊情况：若足球在某一位置不断改变方向却无法移动，视为卡死状态;
- ⑦ 进球处理：若足球进入球门范围，则跳出循环并输出结果。

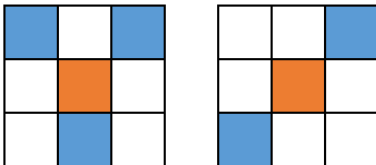
Problem J - C 属性大爆发

可以证明，足球只会在初始状态时卡死。

Problem J - C 属性大爆发

可以证明，足球只能在初始状态时卡死。

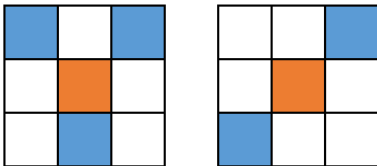
两种可能的卡死情况（均为斜向移动）：



Problem J - C 属性大爆发

可以证明，足球只会在初始状态时卡死。

两种可能的卡死情况（均为斜向移动）：



时间复杂度为 $O(n \times m + t)$ ，其中 $n \times m$ 为场地大小， t 为给定的最大移动时间。

Problem B - 并非无法战胜

题目大意

- 与 AI 进行井字棋对战。
- 关键词：交互、博弈、暴力、Minimax

Problem B - 并非无法战胜

题目大意

- 与 AI 进行井字棋对战。
- 关键词：交互、博弈、暴力、Minimax

对战流程如下：

Problem B - 并非无法战胜

题目大意

- 与 AI 进行井字棋对战。
- 关键词：交互、博弈、暴力、Minimax

对战流程如下：

- ① 使用一个二维数组维护棋盘状态；
- ② 每次接收到 AI 的坐标，更新棋盘；
- ③ 判断是否已有胜负或平局；
- ④ 计算当前局面的最佳落子位置，输出并刷新。

Problem B - 并非无法战胜

我们可以将所有的局面构造成一棵博弈树，而 Minimax 算法正是用于在这种博弈树上搜索最优策略的一种方法。它假设你和对手都“完美博弈”，总是做出对自己最有利的决策。

Problem B - 并非无法战胜

我们可以将所有的局面构造成一棵博弈树，而 Minimax 算法正是用于在这种博弈树上搜索最优策略的一种方法。它假设你和对手都“完美博弈”，总是做出对自己最有利的决策。

在井字棋中：

- 玩家视角（你）是 Max，目标是最大化得分。
- AI 视角是 Min，目标是最小化你的得分。
- 每一步我们在当前状态下枚举所有可能的落子，递归判断对手的响应，选择最终收益最优的一步。

Problem B - 并非无法战胜

我们根据以下规则对每一个局面进行打分：

- 如果你赢了 (O 三连)，返回 +1；
- 如果 AI 赢了 (X 三连)，返回 -1；
- 如果平局，返回 0；
- 若游戏还未结束，继续搜索子状态。

Problem B - 并非无法战胜

如何使用 Minimax 找到最优解？

① 在当前棋盘局面下，枚举每一个空格（即尚未落子的格子），尝试在该位置落下我方棋子（0）；

Problem B - 并非无法战胜

如何使用 Minimax 找到最优解？

- ① 在当前棋盘局面下，枚举每一个空格（即尚未落子的格子），尝试在该位置落下我方棋子（0）；
- ② 对于每一种尝试，递归模拟接下来的博弈过程，双方轮流落子，始终遵循各自的最优策略（我方最大化得分，对方最小化得分）；

Problem B - 并非无法战胜

如何使用 Minimax 找到最优解？

- ① 在当前棋盘局面下，枚举每一个空格（即尚未落子的格子），尝试在该位置落下我方棋子（0）；
- ② 对于每一种尝试，递归模拟接下来的博弈过程，双方轮流落子，始终遵循各自的最优策略（我方最大化得分，对方最小化得分）；
- ③ 根据递归结果评估当前选择的得分：
 - 若我方胜利，得分为 +1；
 - 若对方胜利，得分为 -1；
 - 若平局，得分为 0。

Problem B - 并非无法战胜

如何使用 Minimax 找到最优解？

- ① 在当前棋盘局面下，枚举每一个空格（即尚未落子的格子），尝试在该位置落下我方棋子（0）；
- ② 对于每一种尝试，递归模拟接下来的博弈过程，双方轮流落子，始终遵循各自的最优策略（我方最大化得分，对方最小化得分）；
- ③ 根据递归结果评估当前选择的得分：
 - 若我方胜利，得分为 +1；
 - 若对方胜利，得分为 -1；
 - 若平局，得分为 0。
- ④ 最后，从所有备选落子中选出得分最高的那一格，作为当前局面的最优决策。

Problem B - 并非无法战胜

虽然 Minimax 是一种暴力搜索算法，但由于棋盘仅为 3×3 ，且
 首步由 AI 先行，最多只需搜索 $8! = 40\,320$ 种局面。

Problem B - 并非无法战胜

虽然 Minimax 是一种暴力搜索算法，但由于棋盘仅为 3×3 ，且
 首步由 AI 先行，最多只需搜索 $8! = 40\,320$ 种局面。

实际上，许多对局在胜负未分前就已结束，因此搜索空间远小于
 理论上限，完全可以在时间限制内完成搜索。

Problem B - 并非无法战胜

虽然 Minimax 是一种暴力搜索算法，但由于棋盘仅为 3×3 ，且首步由 AI 先行，最多只需搜索 $8! = 40\,320$ 种局面。

实际上，许多对局在胜负未分前就已结束，因此搜索空间远小于理论上限，完全可以在时间限制内完成搜索。

Minimax 在这种小规模棋类中能够实现“必不败”的策略，是极为有效的博弈手段。

Problem B - 并非无法战胜

虽然 Minimax 是一种暴力搜索算法，但由于棋盘仅为 3×3 ，且首步由 AI 先行，最多只需搜索 $8! = 40320$ 种局面。

实际上，许多对局在胜负未分前就已结束，因此搜索空间远小于理论上限，完全可以在时间限制内完成搜索。

Minimax 在这种小规模棋类中能够实现“必不败”的策略，是极为有效的博弈手段。

优化： $\alpha - \beta$ 剪枝或选择最快获胜的路径。

Problem K - 计算机的末路

题目大意

- 给你一张图与 k 对接送外卖的地址，请问接送完所有外卖所需的最短时间。
- 关键词：最短路、状压 dp

Problem K - 计算机的末路

题目大意

- 给你一张图与 k 对接送外卖的地址，请问接送完所有外卖所需的最短时间。
- 关键词：最短路、状压 dp

- ① 由于我们的目的是不断接送外卖，因此移动路径一定是从一个外卖点到另一个外卖点， k 的数量级不大且外卖点至多只有 $2k$ 个，很容易想到用状压 dp 处理最优路径。

Problem K - 计算机的末路

题目大意

- 给你一张图与 k 对接送外卖的地址，请问接送完所有外卖所需的最短时间。
- 关键词：最短路、状压 dp

- ① 由于我们的目的是不断接送外卖，因此移动路径一定是从一个外卖点到另一个外卖点， k 的数量级不大且外卖点至多只有 $2k$ 个，很容易想到用状压 dp 处理最优路径。
- ② 在外卖点与外卖点之间的移动路径一定是按照两点间最短路移动，因此先预处理出来 $2k$ 个外卖点到其余点的最短路。

Problem K - 计算机的末路

- ① 定义 $dp_{st,i}$ 为目前外卖状态为 st 且目前在 i 点时的最短时间。 st 为 $2k$ 位 2 进制数，前 k 位分别代表外卖是否取走，后 k 位代表外卖是否送出。

Problem K - 计算机的末路

- ① 定义 $dp_{st,i}$ 为目前外卖状态为 st 且目前在 i 点时的最短时间。 st 为 $2k$ 位 2 进制数，前 k 位分别代表外卖是否取走，后 k 位代表外卖是否送出。
- ② 状态转移 $dp_{st,i} = \min(dp_{st,i}, dp_{lst,j} + dist_{i,j})$ 。其中 lst 为 st 的上一个状态，数值上为 st 的二进制上某位 1 取 0 的结果。

Problem K - 计算机的末路

- ① 定义 $dp_{st,i}$ 为目前外卖状态为 st 且目前在 i 点时的最短时间。 st 为 $2k$ 位 2 进制数，前 k 位分别代表外卖是否取走，后 k 位代表外卖是否送出。
- ② 状态转移 $dp_{st,i} = \min(dp_{st,i}, dp_{lst,j} + dist_{i,j})$ 。其中 lst 为 st 的上一个状态，数值上为 st 的二进制上某位 1 取 0 的结果。
- ③ 转移时间复杂度为 $O(2^{2k} \times k^2) \approx 10^8$ ，复杂度过大，不能接受。

Problem K - 计算机的末路

- ① 因此可以考虑优化状态，由于 k 个外卖每个的状态只有未被取，被取走但没被送达，被送达三种状态，可将状态优化为 k 位三进制数。

Problem K - 计算机的末路

- ① 因此可以考虑优化状态，由于 k 个外卖每个的状态只有未被取，被取走但没被送达，被送达三种状态，可将状态优化为 k 位三进制数。
- ② $3^{10} = 59\,049 < 2^{20} = 1\,048\,576$ ，因此对时间和空间的优化是显著的。经优化后的时间复杂度已经足够通过本题。

Problem K - 计算机的末路

- ① 因此可以考虑优化状态，由于 k 个外卖每个的状态只有未被取，被取走但没被送达，被送达三种状态，可将状态优化为 k 位三进制数。
- ② $3^{10} = 59\,049 < 2^{20} = 1\,048\,576$ ，因此对时间和空间的优化是显著的。经优化后的时间复杂度已经足够通过本题。
- ③ k 对外卖点之间可能有重复，为了避免 dp 状态有误，我们需要先对外卖点进行离散化。

Problem K - 计算机的末路

- 除此之外，dist 数组在枚举时可能会出现卡 cache 的情况，常数较大。注意到 dist 的终点也最多只会有 $2 \times k$ 个点，因此可以考虑将 dist 压缩成 $2k \times 2k$ 的大小，也可以在不优化三进制状态的情况下通过。