25 秋季 C 语言程序设计课程期末作业

人机对战版五子棋程序设计

提交时间：2026 年 1 月 15 日 23:59 前

重要提示

- 本次作业为 **C 语言课程期末作业**，占课程总评成绩的 50%
- 请务必独立完成，严禁直接抄袭他人代码，一经发现按零分处理
- 本次作业代码评分采用**人工评阅**方式，需提交**完整项目源代码**和**主程序可执行文件**；对战得分由现场最终实际名次决定
- 提交截止时间后原则上不再接受补交，请合理安排时间
- 程序必须能够**正常编译运行**，无法运行的程序将严重影响评分

目录

1 项目简介

期末作业要求在期中作业的基础上，加入五子棋“AI”的设计，实现一个功能完整的**人机对战版五子棋程序**。五子棋，又称连珠、五目、五目碰、五格等，是起源于中国古代的传统黑白棋种之一。现代五子棋称为连珠，是一种两人对弈的纯策略型棋类游戏。

1.1 基本介绍

五子棋使用围棋棋盘和棋子，两人对局，各执一色，轮流下一子。棋子下在棋盘的交叉点上，棋子下定后不再移动。**率先在棋盘上形成横向、竖向或斜向连续五个同色棋子的一方获胜**。

棋盘正中一点为“天元”。棋盘两端的横线称端线。棋盘左右最外边的两条纵线称边线。从两条端线和两条边线向正中发展而纵横交叉在第四条线形成的四个点称为“星”。

以持黑方为准，棋盘上的纵轴线从左到右用英文字母 A~O 标记。横行线从近到远用阿拉伯数字 1~15 标记。纵横轴上的横纵线交叉点分别用横纵线标记的名称合写成。如“天元”为 H8，四个“星”分别为 D4、D12、L12、L4 等。

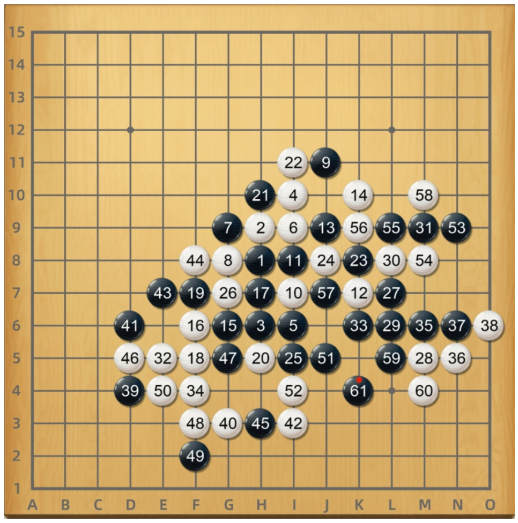

图 1: 五子棋棋盘示意图

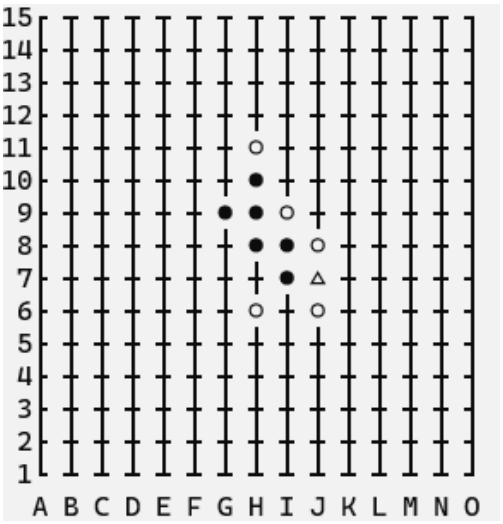

图 2: 控制台棋盘实现示例

1.2 项目目标

通过本次作业，要求学生：

- **掌握 C 语言程序设计能力**：能够熟练运用 C 语言的基本语法、数据结构（数组、结构体等）、函数设计等编程技巧，完成一个功能完整的应用程序
- **理解并实现五子棋规则**：深入理解五子棋的基本规则及禁手规则（三三禁手、四四禁手、长连禁手），能够通过代码准确实现规则判断
- **设计并实现 AI 算法**：学习和运用经典的博弈树搜索算法（如极大极小搜索、Alpha-Beta 剪枝等），设计具有一定棋力的五子棋 AI
- **提高模块化编程能力**：学会将复杂程序拆分为多个模块（如棋盘管理、规则判断、AI 决策、界面显示等），实现代码的高内聚、低耦合
- **培养代码规范和文档撰写能力**：养成良好的代码风格和注释习惯，能够撰写清晰的技术文档（README）
- **锻炼算法优化思维**：在保证 AI 棋力的同时，关注程序运行效率，思考如何在有限时间内做出更优决策

2 功能要求

程序必须实现以下基本功能：

1. **使用 C 语言编写**
   - 必须使用**标准 C 语言**（C99 及以上标准）编写
   - **严禁使用 C++** 或其他语言
2. **命令行界面**
   - 程序必须在**命令行终端**中运行，按图2所示的方式显示棋盘，**不要改动行列编号顺序**
   - **不允许使用图形界面**（如 GTK、Qt 等）
   - 使用标准输入输出（`stdin/stdout`）进行交互
   - 界面应清晰美观，易于理解（期末评分的重要依据之一）
3. **标准键盘输入**
   - 只能使用**键盘**进行输入
   - **不使用鼠标**进行交互
   - 输入方式采用“字母 + 数字”式的坐标输入（如：`H7`）
4. **人机对战模式**
   - 实现**人机对战**功能，黑白双方轮流落子
   - 明确显示当前轮到哪一方落子
   - 提供清晰的落子提示（如输入引导文字）
   - 实现机器落子功能，**机器落子后需要明确输出机器本轮思考所用时长（最大允许时长为 15s）**以及**本对局中机器落子思考的平均时长（包括本轮落子思考用时）**，单位为秒，**精确到小数点后两位**
5. **禁手规则实现**
   - 实现**黑棋禁手**规则（详见第3节）
   - 白棋无禁手限制
   - 程序应能自动判断禁手并在黑棋尝试下禁手点时给出明确提示，对黑棋判负
6. **最后一步特殊显示**
   - 最新落下的棋子应有**特殊标记**，可以用不同的符号、颜色来区分，如实心三角形/空心三角形
   - 便于玩家识别最近的操作
7. **输赢判断**
   - 能够正确判断五连（横、竖、斜向）
   - 判断黑棋禁手导致的输局
   - 判断棋盘下满的和局
   - 胜负确定后显示明确的结果信息，而不是直接退出程序

3 五子棋规则详解

3.1 基本规则

1. **对局双方各执一色**，黑先、白后，从天元（棋盘中心）开始相互顺序落子
2. **白棋第一手**应在天元为界自己一侧布子，之后双方可任意行子
3. **最先在棋盘**横向、竖向、斜向形成连续的相同色五个棋子的一方为胜
4. **黑棋禁手判负、白棋无禁手**
   - 黑棋禁手包括：**“三三禁手”、“四四禁手”、“长连禁手”**
   - 黑方只能通过“四三”取胜
   - 黑方出现禁手，无论是自愿或被迫走出，判白方胜
   - 若白方在黑方出现禁手后，未立即指出又落下一白子，则黑方禁手不再成立
5. 如分不出胜负（棋盘下满），则定为**平局**
6. **五连与禁手同时形成**，先五为胜（五连优先于禁手）

3.2 棋型说明

理解以下棋型定义对于实现禁手规则至关重要：

1. **长连**
   - 定义：在一条直线或斜线上，连续下成**六个或以上**的同色棋子
   - 说明：黑棋形成长连即为禁手，白棋长连视为胜利
2. **活三**
   - 定义：若对方不进行必要防守，本方**下一手可取得活四**的基本子力形式（包括连活三和跳活三）
     - **连活三**：在一条横线、竖线或斜线上紧紧相连的同色三子，两端均有空交叉点，且至少有一端有两个或以上的空交叉点（对于黑方则还需左右相邻的两个交叉点在此三形成的同时至少有一个不为禁点，且从两端向外数至少在各两个交叉点上均无黑子）所构成的基本子力形式
     - **跳活三**：中间间隔一子的活三，且两端均有空交叉点（对于黑方则还需在此三形成的同时中间的空交叉点不为禁点，且从两端向外数至少在各两个交叉点上均无黑子）所构成的基本子力形式
   - 特别说明：1、3、5 式的中间空出两个子的三不叫跳活三，而只能称为跳三。跳三不认为属于活三的范围，所以不纳入禁手判断考虑的情况
   - 示例：
     - 连活三：___●●●___（两端均为空，至少一端有两个空位）
     - 跳活三：_●_●●_（中间间隔一子）
3. **活四**
   - 定义：在一条直线或斜线上连续相邻的四枚同色棋子且有两个空白交叉点可以成五的棋型（对于黑方则还需左右相邻的两个交叉点在此四形成的同时至少有一个不为禁点，且从两端向外数至少在各两个交叉点上均无黑子）
   - 示例：___●●●●___（两端均有两个空位）

4. **冲四**

   - 定义：指除“活四”外的，再下一着棋（下子的位置不能是黑棋的禁点）便可形成五连，并且存在五连的可能性的局面
   - 示例：○●●●●___ 或 _●_ ●●●○

5. **五连**

   - 定义：五个同色棋子在一条线上连成一线
   - 这是获胜的基本条件

3.3 禁手详解

黑棋先行具有较大优势，因此对黑棋施加了禁手限制。**白棋无任何禁手限制**。

3.3.1 禁手的类型

1. **三三禁手**

   - 定义：由于黑方走一着在无子交叉点上同时形成两个或两个以上黑方“活三”的局面
   - 注意：**此子必须是形成的至少两个活三的共同构成子**
   - 如图4所示，A 点黑棋一子落下同时形成了两个活三，注意三个黑 1 组成的三，在黑棋 A 点落子后由眠三（假活三）成为活三，这是由于 A 点落下黑子后，B 点被解禁了。但 A 点落子不是这两个活三的共同构成子，所以 A 点不是三三禁手。**请同学们务必仔细揣摩**。

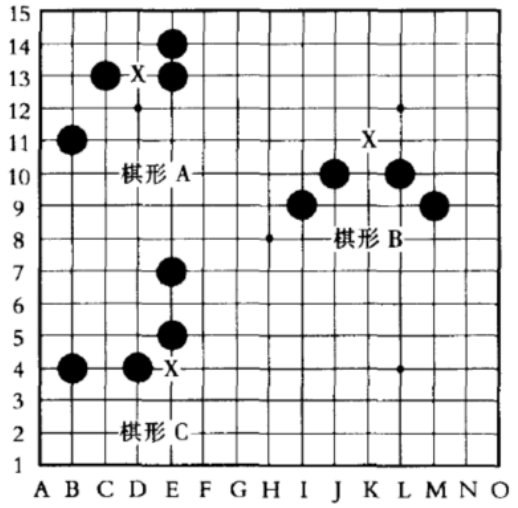

图 3: 三三禁手示例（X 点为禁手点）

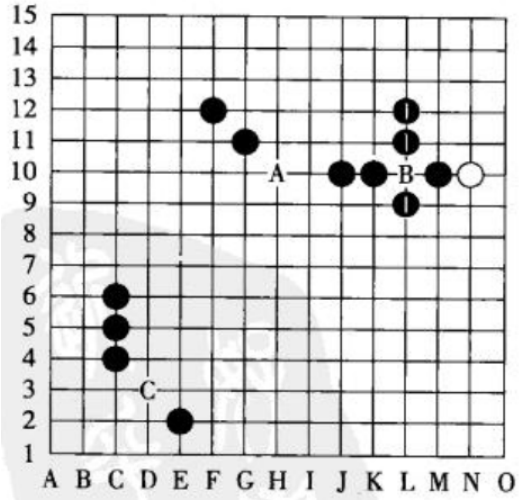

图 4: 三三禁手反例（A、C 点不为禁手点）

2. **四四禁手**

   - 定义：当黑棋走一步棋，**同时形成两个或两个以上的“四”**（包括活四和冲四）且没有形成“五连”时，该落子点为禁手点
   - 注意：只要是“四”的棋型都算，不区分活四还是冲四
   - 如果同时形成了五连，则五连优先，不算禁手

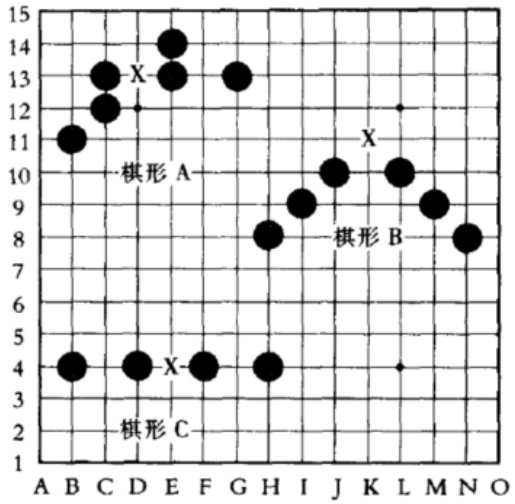

图 5: 四四禁手示例（X 点为禁手点）

3. **长连禁手**

   - 定义：当黑棋走一步棋，**形成六个或六个以上连续同色棋子**，该落子点为禁手点
   - 注意：只有黑棋受此限制，白棋长连视为胜利

3.4 胜负判定

3.4.1 胜局条件

1. **五连胜利**

   - 最先在棋盘上形成五连的一方获胜
   - 白棋长连（六个或以上连续棋子）视同五连，白方胜

2. **黑棋禁手判负**

   - 黑方走出禁手点，白方胜
   - 但如果黑方同时形成五连与禁手，禁手失效，黑方胜

3. **对方认输或超过规定时限**

   - 一方主动认输，对方获胜
   - 一方该回合落子思考时长超过规定时限（**针对机器思考而言，单回合时限为 15s**），对方获胜

3.4.2 和局条件

1. **棋盘下满**

   - 全盘均下满，已无空白交叉点
   - 双方均未形成五连，判为和局

2. **双方同意和棋**

   - 对局双方同一回合均放弃行棋权
   - 对局双方一致同意和棋

4 五子棋 AI 模块设计

本部分将会给出一些五子棋 AI 模块的设计策略建议，仅供同学们参考。在很多主流方法的基础上，加入一些特殊的优化手段，往往会有意想不到的效果。

4.1 局面评估函数

在介绍搜索算法之前，首先需要理解**局面评估函数**的概念。评估函数用于对当前棋盘局面进行打分，评估对某一方的有利程度。一个好的评估函数是 AI 棋力的基础。

常见的评估策略包括：

- **棋型统计**：统计己方和对方的活四、冲四、活三、眠三、活二等棋型数量。
- **分数量级**：不同棋型的分数应呈**指数级差异**。例如，活四（必胜）的分数应远大于活三，而活三的分数应远大于活二。若权重设置过于线性，AI 容易为了贪图几个活二而忽视对方的必杀棋。
- **位置权重**：棋盘中心位置通常比边缘更有价值，可以使用位置价值表（Position Map）进行加权。
- **禁手判断**：必须在评估时检测黑棋是否形成“三三”、“四四”或“长连”。若落入禁手点，应直接判负或赋予极低的惩罚分。

4.2 极大极小搜索（Minimax）

极大极小搜索是博弈论中最经典的算法之一，基本思想是：

- 假设双方都采取对自己最有利的策略。
- **极大层**（Max 层）：轮到己方落子时，选择使评估值最大的走法。
- **极小层**（Min 层）：轮到对方落子时，假设对方会选择使评估值最小（对己方最不利）的走法。
- 通过递归搜索博弈树，最终在根节点选出最优的落子位置。

4.3 Alpha-Beta 剪枝

极大极小搜索的问题在于需要遍历所有可能的走法，计算量随着搜索深度呈指数增长。**Alpha-Beta 剪枝**是对极大极小搜索的优化，通过维护两个界限值 $[\alpha, \beta]$，在不影响最终结果的情况下大幅减少搜索节点。

基本原理：

- **Alpha (α)**：当前搜索路径上，极大层（Max）已经找到的最好的结果（下界）。任何低于此值的走法 Max 层都不会考虑。
- **Beta (β)**：当前搜索路径上，极小层（Min）已经找到的最好的结果（上界，即对 Max 层来说最坏的结果）。任何高于此值的走法 Min 层都不会让其发生。
- **剪枝**：当某个节点的评估值导致 $\alpha \geq \beta$ 时，发生剪枝（Cut-off），停止搜索该节点的剩余子节点。

Listing 1: Alpha-Beta 剪枝伪代码

```
int alphabeta(int depth, int alpha, int beta, int isMaxPlayer) {
    // 达到搜索深度或游戏结束（分出胜负）
    if (depth == 0 || gameOver())
        return evaluate();
```

```
5
6       if (isMaxPlayer) {
7           int maxEval = -INFINITY;
8           for (each possible move) {
9               makeMove(move);
10              int eval = alphabeta(depth - 1, alpha, beta, 0);
11              undoMove(move);
12
13              maxEval = max(maxEval, eval);
14              alpha = max(alpha, eval);
15
16              // Beta剪枝：对方之前已经找到了更坏的局面，不会让当前局面发生
17              if (beta <= alpha)
18                  break;
19          }
20          return maxEval;
21      } else {
22          int minEval = +INFINITY;
23          for (each possible move) {
24              makeMove(move);
25              int eval = alphabeta(depth - 1, alpha, beta, 1);
26              undoMove(move);
27
28              minEval = min(minEval, eval);
29              beta = min(beta, eval);
30
31              // Alpha剪枝：己方之前已经找到了更好的局面，不需要考虑当前分支
32              if (beta <= alpha)
33                  break;
34          }
35          return minEval;
36      }
37  }
```

4.4 蒙特卡洛树搜索（MCTS）

蒙特卡洛树搜索通过大量随机模拟（Play-out）来评估走法，在围棋中表现优异。其包含选择、扩展、模拟、回溯四个阶段。

1. **选择（Selection）**：从根节点开始，根据 UCB 公式选择子节点，直到到达叶子节点
2. **扩展（Expansion）**：在叶子节点处，添加一个或多个新的子节点
3. **模拟（Simulation）**：从新节点开始，进行随机对局直到游戏结束
4. **回溯（Backpropagation）**：将模拟结果沿路径回传，更新各节点的统计信息

UCB（Upper Confidence Bound）公式用于平衡探索与利用：

$$UCB = \frac{w_i}{n_i} + c\sqrt{\frac{\ln N}{n_i}}$$

其中 w_i 是节点获胜次数，n_i 是节点访问次数，N 是父节点访问次数，c 是探索常数。

注意：五子棋属于**“骤死型”博弈**（Sudden Death），一步疏忽（如漏看冲四）就会直接导致失败。纯粹的 MCTS 在有限计算量下往往难以敏锐捕捉到局部的必杀形状。对于初学者，**建议优先完善 Alpha-Beta 搜索体系**，除非有能力结合神经网络或针对性的剪枝策略。

4.5 搜索优化技巧

为了在有限的时间（如 15 秒）内让 AI 做出更深远的决策，以下优化策略至关重要：

1. **启发式走法排序**：这是 Alpha-Beta 效率的关键。优先搜索“最有希望”的走法（如就在刚落子的棋子附近、或能形成活三的点），能更早触发剪枝。
2. **算杀模组（VCF/VCT）**：单纯的固定深度搜索（如搜 6 层）很难看到 15 步之后的连杀。建议在叶子节点处，如果发现有连续进攻机会（冲四、活三），则开启**算杀搜索**（Victory by Continuous Four/Three），专门验证是否必胜。
3. **迭代加深（Iterative Deepening）**：不要固定只搜 N 层。应从 2 层开始，逐步加深到 4, 6, 8... 这样即使超时，也能返回上一层搜索到的最佳结果。
4. **置换表（Transposition Table）**：使用 Zobrist Hashing 记录局面。如果当前局面之前搜过，直接复用结果，避免重复计算。
5. **邻域裁剪**：五子棋的有效落子通常都在现有棋子的“邻域”内（如距离 2 格以内）。只生成这些位置的候选走法，可大幅缩小分支因子。

4.6 其他进阶方法

除了上述经典方法，还有一些其他技术可供参考（部分实现难度较高，量力而行）：

- **威胁空间搜索（Threat-Space Search）**：专门针对五子棋设计的搜索算法，重点搜索具有威胁性的走法序列
- **证明数搜索（Proof-Number Search）**：用于求解胜负的专用算法，可以快速判断当前局面是否必胜
- **神经网络评估**：使用机器学习方法训练评估函数，但实现复杂度较高，且在 C 语言中实现有一定难度

建议：对于大多数同学，建议从**极大极小搜索 + Alpha-Beta 剪枝**入手，配合一个合理的评估函数和基本的优化技巧，就能实现一个具有较强棋力的 AI。在此基础上，可以逐步添加更多优化手段来提升 AI 的表现。如果关于 AI 策略的实现有其他独特的想法，欢迎随时与助教交流讨论。

注意：无论采用何种方法，**必须严格遵守本文档中关于 AI 模块设计的所有要求（如思考时长限制）**。

5 对战规则

5.1 比赛说明

最终的五子棋比赛将于本学期最后一次实验课（北京时间 **2026 年 1 月 15 日晚**）上随堂进行。所有同学必须在指定时间现场参与五子棋对战，方可获得该部分对战分数，**缺席将必然导致失去该部分所有分数**。

每两位同学之间的对战应遵循以下规则：

- 选择“人机对战”模式，由双方各自设计的 AI 进行对弈，人类负责口头传递和输入对方机器落子的坐标
- 共对弈两局，双方机器各执黑一局，白一局
- 每轮对局结束后，按照3.4节的胜负判定规则对结果进行判定，胜者得 2 分，负者得 0 分
- 原则上每盘棋有且仅有一个胜者，若棋局需要判定为和局，必须请现场老师进行认定，和局情况下双方各得 1 分

所有在现场参与五子棋比赛的同学将会随机分成若干个小组，每个小组先进行内部赛，小组内部两两对战，得分最高的两位同学获得小组出线资格，参加后续的淘汰赛。淘汰赛中，每组出线选手进行对战，胜者晋级，负者直接淘汰，决赛中胜者获得冠军，负者获得亚军，以此类推。

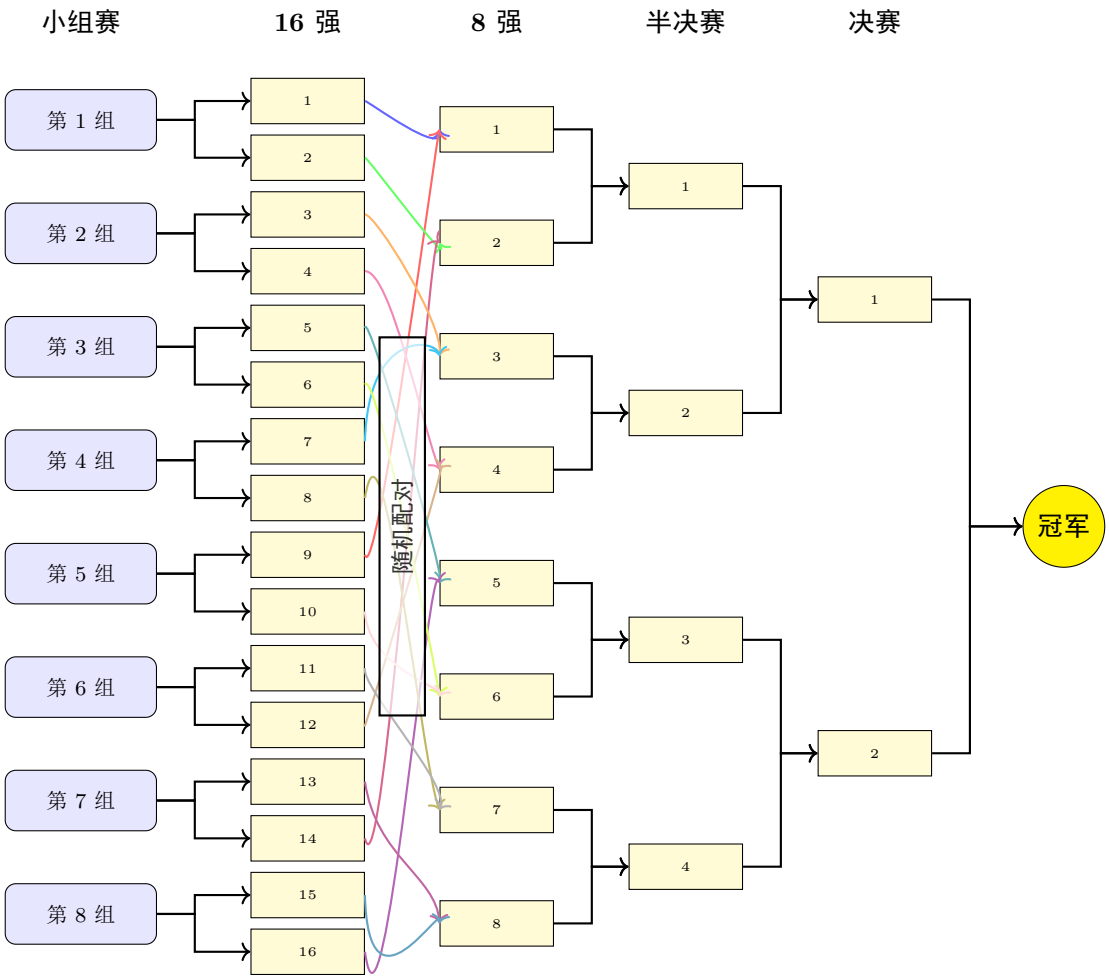

图 6: 五子棋比赛赛制示意图（8 组）

淘汰赛中的晋级规则：

- 共对弈两局，双方各执黑一局，白一局
- 两局均获胜利者，可直接晋级，若两局均失败，则直接淘汰
- 若出现两人得分相等的情况（比如“各胜一局”），则**通过比较截至最后一步落子时双方机器思考平均用时以决定晋级者**，用时少者晋级；若用时也完全相同，则视情况由现场老师进行裁决或在时间充裕的情况下加赛
- 若代码中没有实现机器思考时长输出功能，则不能获得晋级资格
- 若对输出机器思考时长有弄虚作假行为，一经发现直接取消晋级资格

5.2 补充说明

1. 棋盘的坐标必须按照图2格式显示，此外任何情形均视为不合，如：字母和数字坐标颠倒、下标从 0 开始、棋盘格数不对等等
2. 棋子必须下在棋盘线的交点上，不可下在格子中
3. 棋盘的输入只能以键盘输入，为方便比赛，规定统一以字母为先，如：F8
4. 任何一方的落子输入坐标与实际输出的棋盘落子坐标不符，判其为负
5. 人机对战时，**计算机算出的落子坐标需以字符的方式输出到屏幕**，如输出“机器最后落子 E3”，这个坐标将作为对手的输入项，如果输出坐标值与棋盘实际落子不符，则判负
6. 人机对战必须有人先和机先两种模式可供选择（即执黑者的选择）
7. 棋局的输赢以计算机的判定为准，如果实际已经胜出，但胜出方的计算机算法没有识别出，则不算胜出，如双方都没有识别到胜出则继续下棋，如输家识别到对方胜出，判定为和局
8. 比赛过程中如一方的程序异常退出，则判其为负
9. 棋盘直到下满都没有决出胜负，判为和局
10. 考试时不到场参赛的同学无法获得对战分，若其他同学抽到了与不到场的同学进行对战，到场的同学两局都计 2 分，共得 4 分

6 提交要求

6.1 基本要求

本次作业需要提交一个以学号命名的 **zip 压缩包**，包含以下内容：

1. **源代码文件（必需）**
   - 所有.c 和.h 源文件（可以建立子文件夹，但结构和命名应该清晰规范）
   - 代码必须有**良好的注释**（期末评分的重要依据之一）
   - 代码风格应**规范统一**（期末评分的重要依据之一）
   - 如果项目包含多个文件，请在 README 中说明文件组织结构
2. **可执行文件（必需）**
   - 编译好的可执行程序（Windows 下为.exe 文件，Linux 下为可执行文件）
   - 必须保证可执行文件能够直接运行
   - 如果需要特殊的运行环境，请在 README 中说明
3. **README 文档**（必需，期末评分的重要依据之一）
   - 可以是 README.txt 或 README.md，也可以是 tex 文件或 pdf 格式
   - 内容包括：
     - 项目简介（可以简单介绍自己的设计思路和方法，项目的亮点与优势）
     - 编译与运行方法（具体的编译命令等等）
     - 操作说明（如何输入坐标等）
     - 实现的功能列表
     - 已知问题（如果有）
     - 其他说明
4. **Makefile**（可选，但强烈建议）
   - 提供编译脚本，方便评阅
   - 应包含 make、make clean 等目标
5. **精彩对局记录**（可选）
   - 与 AI 对弈的精彩对局截图与说明，或者其他任何能展现你所设计的 AI 水平的证据（如与现有主流 AI 对战时的胜率等）

6.2 提交方式

- 提交平台：国科大在线
- 提交方式：直接以附件形式提交即可
- 如有特殊情况，请提前与助教沟通

7 评分标准

本次作业总分为 **100 分**，占本课程总评成绩的 50%。参考评分标准如下：

7.1 基本功能 (30 分)

评分项	评分要点	分值
程序可运行性	• 程序能够正常编译，且正常运行不崩溃（5 分）	5 分
界面显示	• 棋盘显示清晰美观、坐标显示正确（3 分） • 最后一步有特殊标记（1 分）	4 分
基本对局功能	• 能够正常落子，有落子合法性检查（2 分）	2 分
禁手规则	• 实现禁手判断（3 分） • 禁手提示信息（1 分）	4 分
AI 功能	• AI 能够自动落子（5 分） • 正确输出机器本轮思考时长和平均思考时长（2 分） • 支持人先/机先两种模式（1 分） • AI 落子位置合理，能和普通人类玩家正常对弈多个回合保持无明显失误（7 分）	15 分

7.2 代码质量 (30 分)

评分项	评分要点	分值
代码规范	• 命名规范（变量、函数命名清晰有意义）（3 分） • 代码格式规范（缩进、空格、括号使用一致）（2 分） • 注释充分（关键逻辑有注释说明）（3 分）	8 分
程序结构	• 函数划分合理（单一职责，避免超长函数）（3 分） • 模块化设计（多文件组织，头文件使用得当）（1 分） • 逻辑清晰，代码简洁高效（2 分）	6 分
AI 设计质量	• 算法选择合理（4 分） • 算法实现正确（4 分） • 有一定的优化措施（如剪枝、走法排序等）（2 分）	10 分
文档完整性	• 提供完整的 README 文档（3 分） • AI 算法说明清晰（简单介绍设计思路和方法）（3 分）	6 分

注：学术诚信是每个人都应具有的良好品质。我们鼓励同学们学习借鉴往届大神或者身边同学的优秀思路，但这并不意味着抄袭。借鉴思路时，请记得在注释中说明参考来源并致谢。不加以注释地原样照抄他人代码，一经发现将按零分处理。

7.3 对战得分 (40 分)

本部分分数由现场比赛成绩决定，参加比赛的同学均可获得一定的基础分，比赛名次越高得分越高。

8 常见问题解答

8.1 技术问题

8.1.1 Q1: 为什么我的棋盘输出会有乱码？

A: 极大可能是由于编码问题导致的，请确保你的代码文件和运行时的控制台/终端使用同种编码。对于 Windows 环境，推荐统一使用 GB2312 编码（UTF-8 亦可）。如果使用 UTF-8 编码，Windows 终端可能需要先执行 `chcp 65001` 命令切换代码页。

8.1.2 Q2: 如何输出机器落子思考所用的时长以及平均时长？

A: 机器思考时长指的是 AI 从开始计算到输出落子位置所经过的时间。在 C 语言中，可以使用 `<time.h>` 头文件中的计时函数来实现：

Listing 2: 计时示例代码

```c
#include <time.h>
// 在AI开始思考前记录时间
clock_t start = clock();
// AI进行决策计算...
int bestMove = ai_think();
// 在AI思考结束后记录时间
clock_t end = clock();
// 计算耗时（单位：秒）
double duration = (double)(end - start) / CLOCKS_PER_SEC;
printf("本轮思考用时：%.2f秒\n", duration);
// 累计总时间并计算平均时长
totalTime += duration;
moveCount++;
printf("平均思考用时：%.2f秒\n", totalTime / moveCount);
```

注意：需要使用全局变量或静态变量来记录累计思考时间和落子次数，以便计算平均时长。不同操作系统下 clock() 行为可能略有不同，建议在开发环境下测试确认，比赛以实际运行时间为准。

8.1.3 Q3: 比赛过程中五子棋程序是否可以联网来辅助我的 AI 进行落子决策？

A: 不可以。如果确有需要，必须事先说明情况并征得助教同意，否则将视为作弊。

8.1.4 Q4: 是否可以在程序中调用大语言模型 API 来做决策？

A: 不可以。我们的作业是要求大家自行设计“AI”，而不是比拼谁调用的 API 更强大。

8.1.5 Q5: 对复杂禁手判断的实现有缺陷，是否影响期末评分？

A: 禁手判断不作为期末评分的主要依据。但请注意，如果禁手的判断存在极大漏洞，则很可能导致你在比赛中失利。

8.1.6 Q6: 是否可以使用诸如 CNN、RNN 等神经网络模型来做决策？

A: 原则上允许，但**不推荐**。本课程要求使用**标准 C 语言**编写，在纯 C 环境中实现神经网络非常困难；如果使用 Python 训练模型后导出权重，在 C 中进行推理，实现复杂度很高。经验表明，对于五子棋这类规则明确的博弈问题，传统的搜索算法（如 Alpha-Beta 剪枝）往往比简单的神经网络更有效。

8.1.7 Q7: AI 搜索深度设置多少比较合适？

A: 这取决于你的评估函数复杂度和优化程度。一般来说，在没有剪枝优化的情况下，2-3 层搜索可能就会超时；使用 Alpha-Beta 剪枝通常可以达到 4-6 层；如果加上走法排序等优化，可以尝试更深的搜索。建议使用**迭代加深**策略，从浅层开始逐步加深，在时间限制（15 秒）内尽可能搜索更深。

8.1.8 Q8: 如何防止 AI 思考超时？

A: 建议在思考过程中定期检查计时器，如果发现即将超过 15 秒，应该尽快停止思考。但请注意，强行掐断搜索有时候不是一个很好的选择。

8.2 提交与比赛相关问题

8.2.1 Q9: 可以使用多个.c 文件吗？

A: 完全可以。我们**强烈建议**将代码分成多个文件，例如：

- `main.c`：主程序
- `board.c/board.h`：棋盘相关函数
- `rule.c/rule.h`：规则判断函数
- `ai.c/ai.h`：AI 决策相关函数
- `display.c/display.h`：显示相关函数

提交时记得包含所有源文件，并在 README 中说明编译方法。

8.2.2 Q10: 代码中的注释需要写到什么程度算是“充分”？

A: 这是一个因人而异的问题，没有固定的评判标准。写注释是为了清晰地表达代码的意图和实现过程，如果代码本身意图足够清晰，注释就可以相对简洁甚至略去。

8.2.3 Q11: 现场比赛是使用自己的电脑还是教师机？我的笔记本电脑允许插电吗？

A: 使用自己的电脑。允许插电，但现场的插座数量可能不足，建议笔记本需要插电的同学们提早到场。

祝各位同学顺利完成作业！