《人工智能课程设计》实验二：五子棋人机博弈

装 订 线

学　　号 _____

姓　　名 _____

专　　业 计算机科学与技术

授课老师 _____

目录

装 订 线

1 实验概述

1.1 实验目的

熟悉和掌握博弈树的启发式搜索过程、α－β 剪枝算法和评价函数，并利用 α－β 剪枝算法开发一个五子棋人机博弈游戏。

1.2 实验内容及要求

（1）以五子棋人机博弈问题为例，实现 α－β 剪枝算法的求解程序（编程语言不限），要求设计适合五子棋博弈的评估函数。

（2）要求初始界面显示 15*15 的空白棋盘，电脑执白棋，人执黑棋，界面置有重新开始、悔棋等操作。

（3）设计五子棋程序的数据结构，具有评估棋势、选择落子、判断胜负等功能。

（4）撰写实验报告，提交源代码（进行注释）、实验报告、汇报 PPT。

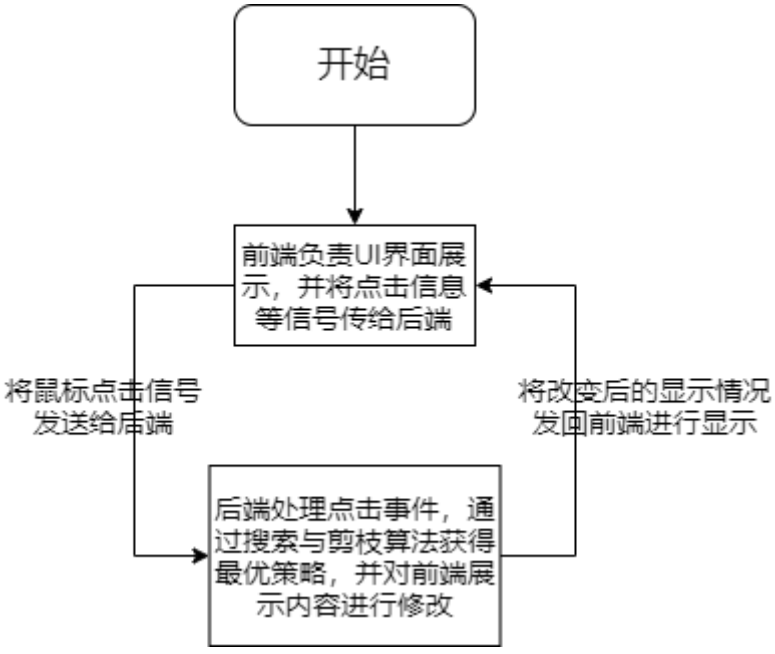
2 实验方案设计

2.1 总体设计思路与总体框架

总体设计思路：

通过设计开发 web 应用进行五子棋博弈游戏的展示，其中算法逻辑部分使用 JavaScript 语言编写，通过启发式搜索与α－β 剪枝算法得到当前最优决策并进行相应操作，同时对鼠标事件等进行逻辑判断，完成游戏的后端功能部分；前端 UI 展示部分使用 html 和 css 语言进行 UI 设计，使人机交互体验更加友好。

总体思路流程图如下：

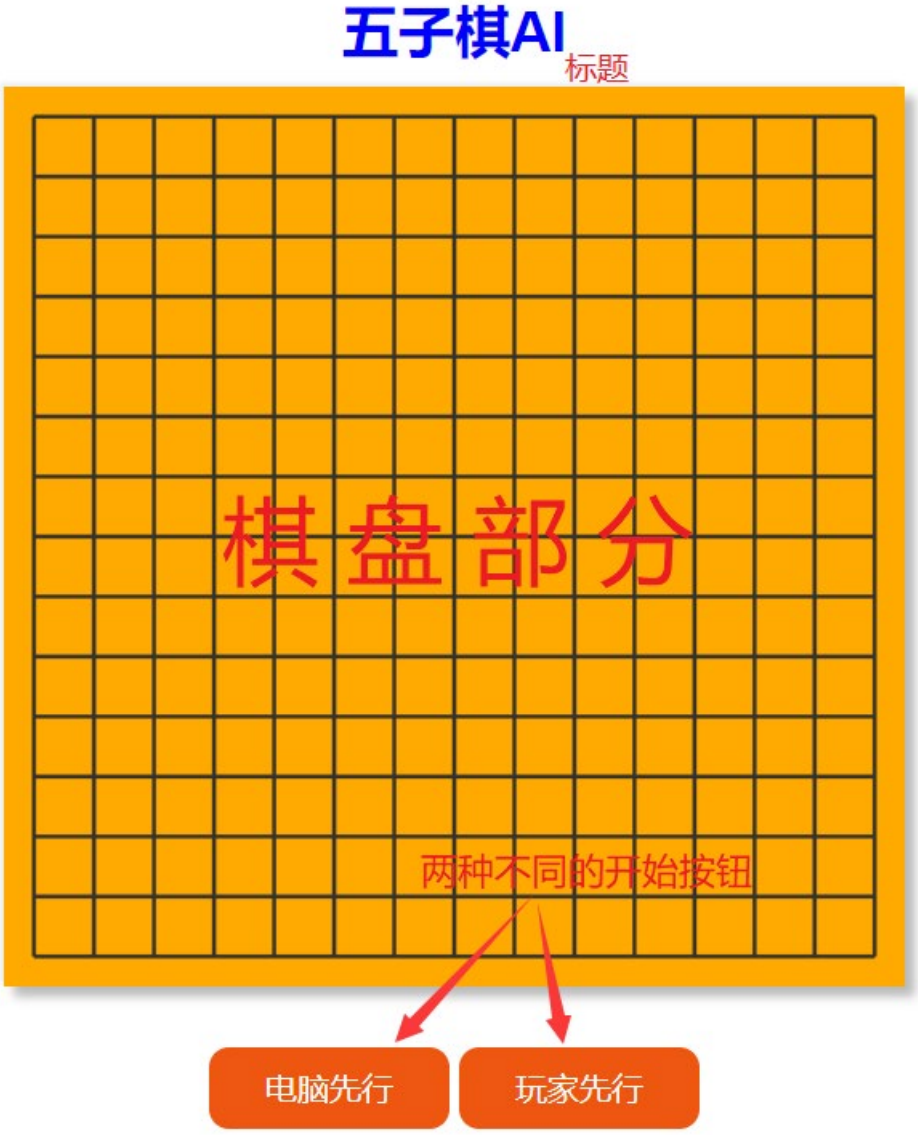


装 订 线

2.1.1 前端 UI 设计思路

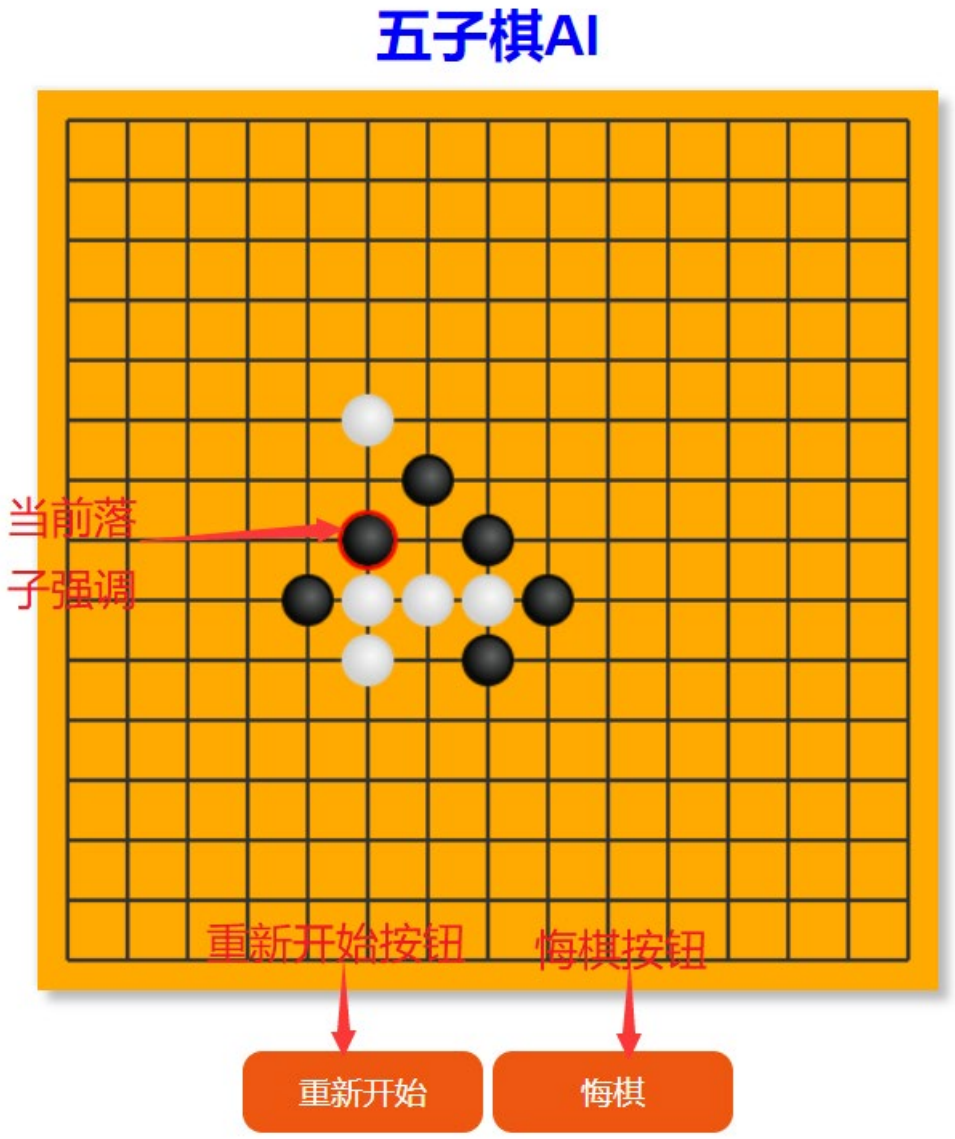
与上次八数码展示程序的设计思路不同，这次五子棋的人机交互程序前端仅进行 UI 展示，而不需要对点击事件进行处理，也不需要在前端进行信息输入，前端仅有的点击事件仅将信号传给后端，所有逻辑由后端进行处理。

UI 界面初始状态如下：



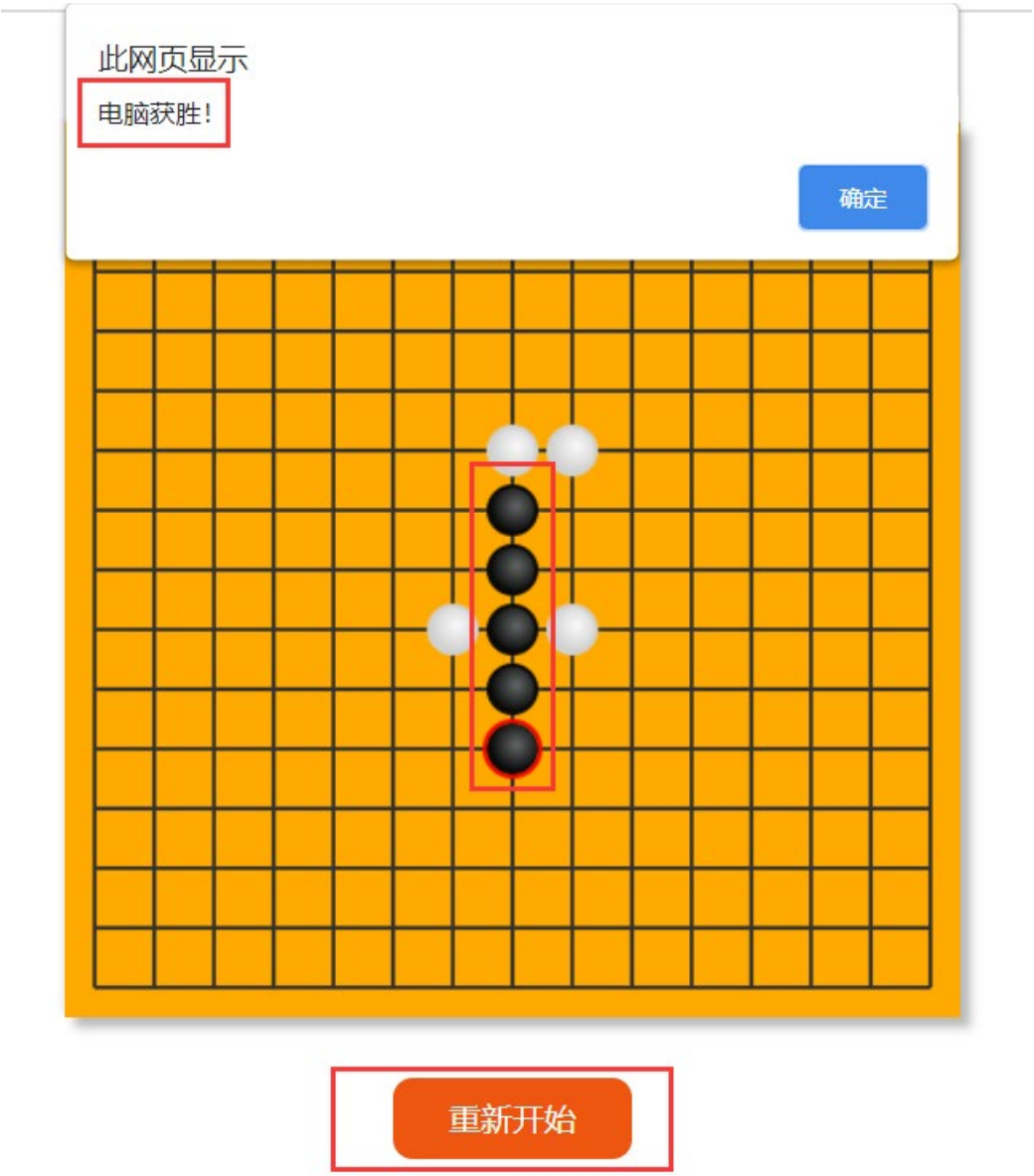
界面上最上方为标题：五子棋 AI；中间是棋盘画布，游戏开始后玩家在此进行点击下棋操作；下方是两个按钮，分别代表“电脑先”和“玩家先”两种不同的游戏开始情况。

点击一种开始按钮，可以开始游戏，游戏过程中的界面展示如下：



界面中原本的两个开始按钮变为了重新开始和悔棋按钮，点击可以完成相应操作；点击棋盘上的相应位置可以实现落子，同时对于最后一步的落子进行标红强调。

游戏结束后界面如下：



其中会给出弹窗提示某一方（电脑或玩家）获胜，同时下方按钮只剩“重新开始“按钮；此时即使关闭弹窗，点击棋盘也不会再进行下棋操作，只有点击”重新开始“后才能重开游戏继续进行操作。

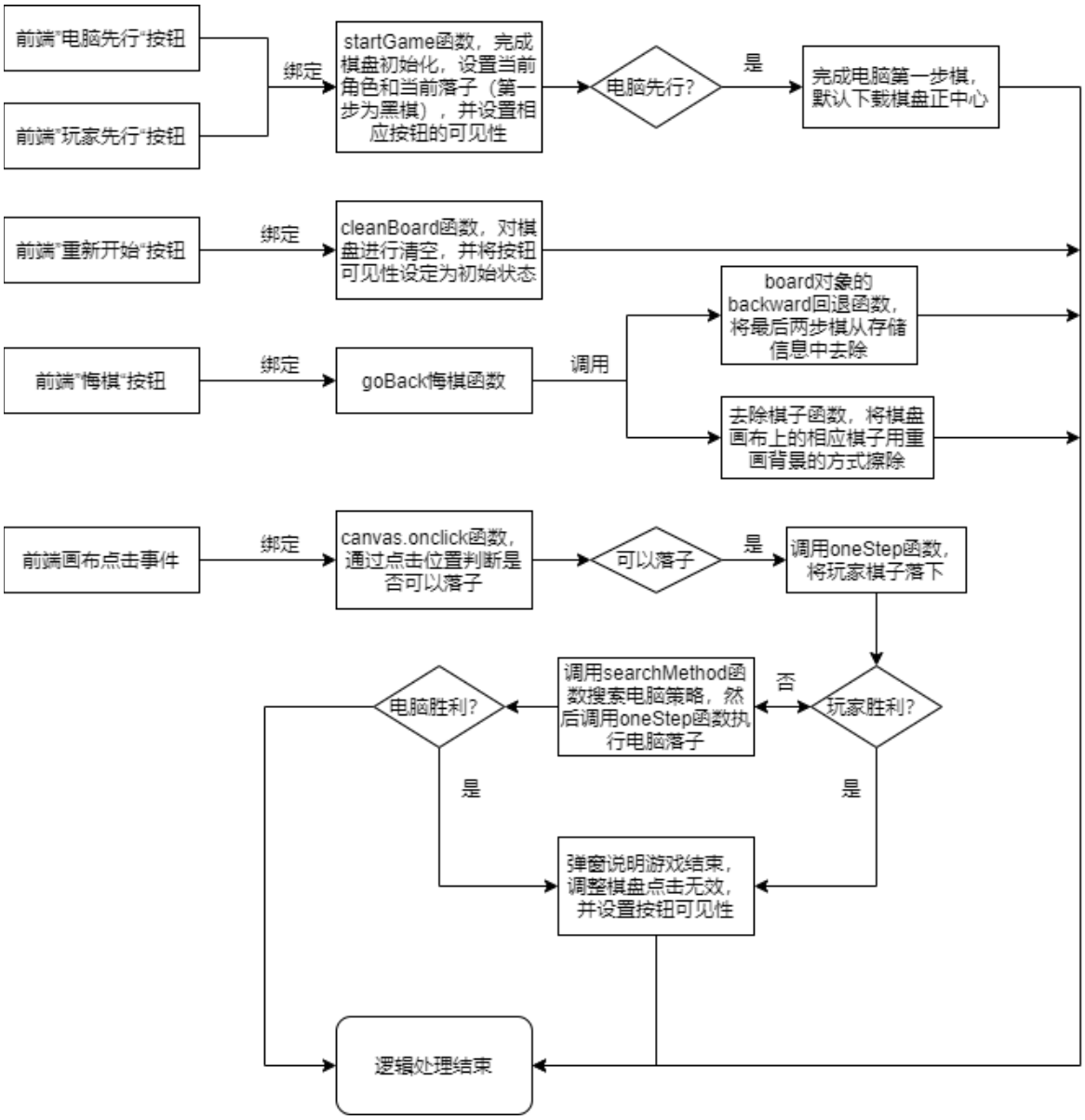
2.1.1 后端部分设计思路

后端部分执行所有的搜索算法，并对前端传回的信号进行逻辑判断以进行相应操作。

逻辑判断部分：

对于 UI 传回的信号，每个信号绑定一个处理函数，根据不同的信号调用相应函数进行相应的逻辑处理，同时可以根据信号中包含的函数传入参数对逻辑判断进行进一步细化。

逻辑判断流程图如下：



搜索算法部分：

（1） 根据当前玩家落子更新棋盘中所有受影响点的得分情况，因为当前落子只会影响棋盘中米子方向距离小于 5 的一系列为空的点，因此只对这些点的得分进行更新；同时可以对得分更新进一步细分，将一个点的得分拆分成四种不同方向的得分，因此当前落子只会影响周围点某一方向的得分，只更新该方向得分即可。

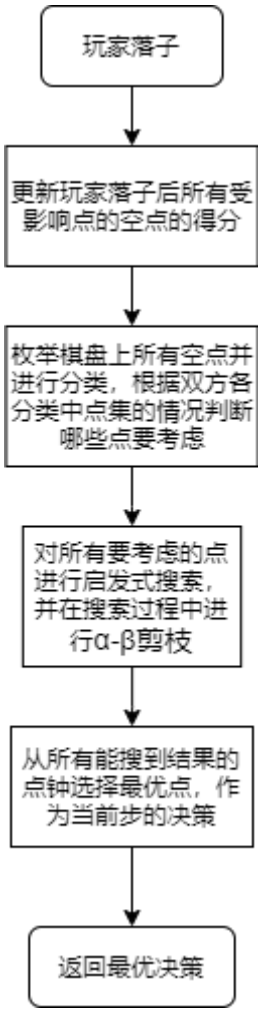
（2） 搜索棋盘上的所有点，对于其中有邻居且目前为空的点进行分类，分别为如果在该处落子能够我方成五、对方成五、我方成四、对方成四……等类，然后根据我方点类的情况和对方点类的情况进行启发式判断，找到所有有价值考虑的点。

（3） 上一步获得的所有点构成极小极大搜索第一层的点集，对该点集进行极小极大搜索（此处使用极小极大搜索结合迭代加深搜索的方式，目的是获取最快的获胜途径，

并在不能获胜时尽可能拖延），并结合α－β 剪枝剪掉不能使结果更优的点，以此加快搜索速度。

（4） 从所有能搜到叶节点的第一层点集中找到最优方案，将此方案返回，作为电脑这一次的决策。

算法流程图如下：



2.2 核心算法及基本原理

本实验的核心算法为极大极小算法和α－β 剪枝算法。

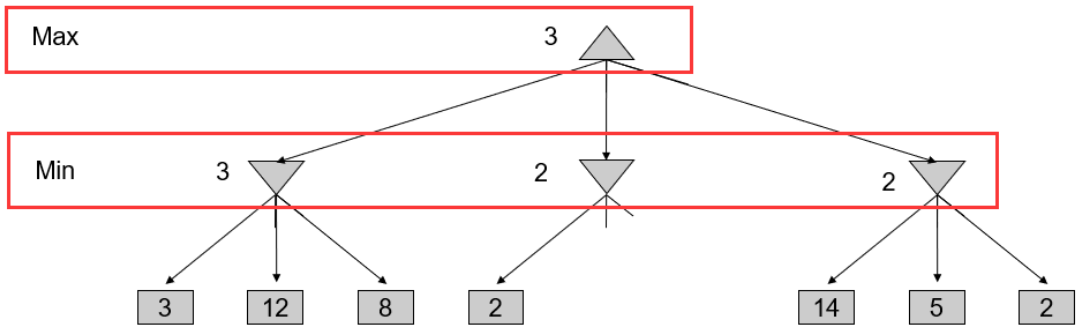
2.2.1 极大极小搜索算法

极大极小原理常用于二人博弈游戏，目的是寻找最优方案使自己的利益最大化。基本思想是假设自己（甲）足够聪明，总是选择最有利于自己的方案，同时对手（乙）也足够聪明，总是选择对自己最不利的方案。

因此将决策树分为两类，一类是自己做决策的层，称为 MAX 层，因为在此层中我方总是选择得分最高的决策；另一类使对手做决策的层，称为 MIN 层，因为在此层中对手总是选择得分最低

的决策。注意，所有的得分都是针对我方而言，同时 MAX 层和 MIN 层一定交替出现（因为我方和对手总是交替决策）。

以下为极小极大搜索的决策树样例，其中第一层我方决策，选择第二层对方决策后得分最高的决策，因此第一层为 MAX 层；同理第二层对方决策，选择第三层我方方决策后得分最低的决策，因此第二层为 MIN 层



2.2.2 α-β 剪枝算法

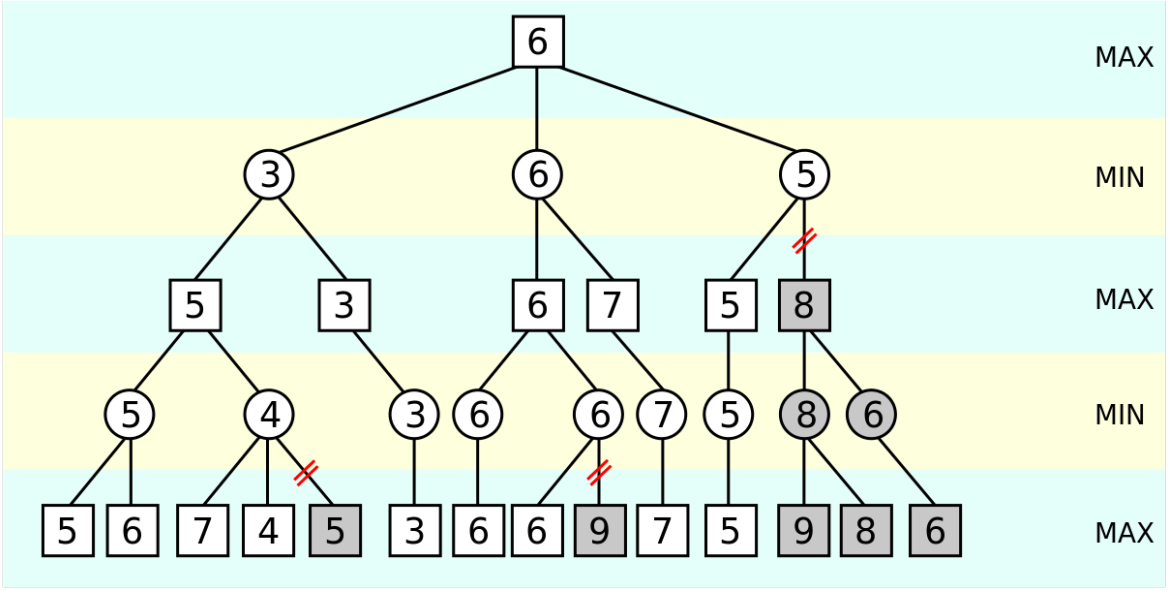
对于极大极小搜索，虽然能够得到最优决策，但是随着搜索层数加深，搜索状态数迅速增大，时间和空间都难以承受，因此有必要在搜索过程中进行剪枝。

α－β 剪枝的策略是，如果当前层某节点的得分一定不能使上一层父节点的选择更优，则可以将该节点减去，不再继续向下搜索。

具体做法是，记录 MAX 层已经找到的当前最大值 α（该值此时为该 MAX 层最终选择值的下界）和 MIN 层已经找到的当前最小值 β（该值此时为该 M 层最终选择值的上界），在搜索 MAX 层节点的子节点时，如果发现 2 层后的某一孙节点的估值低于下界 α 值，则可以剪掉此枝（该孙节点所在的子节点对应的子树的剩余部分），此为 α 剪枝；同理，在搜索 MIN 层节点的子节点时，如果发现 2 层后的某一孙节点的估值高于上界 β 值，则可以剪掉此枝，此为 β 剪枝。

传统的 α－β 剪枝算法对于 MAX 层和 MIN 层各自要写一个函数，互相调用以实现搜索剪枝功能，但是实际上两个函数的逻辑类似，只有在判断极大值和极小值时有所差异，因此可以对其进行简化，合并为一个函数。具体做法是，在 MAX 层搜索完成进行下一层 MIN 的搜索时，不调用新的函数，而是将决策角色进行转换，并将当前 α 值和 β 值取相反数并互换位置后，进行下一层的调用，这样，下一层的搜索就能达到 MIN 搜索的效果；注意，的到下一层的返回值后为保证逻辑正确性，需要将该返回值取相反数。与传统 MIN-MAX 函数有所区别，一般称这种方式的搜索为 NEGAMAX 函数。

α－β 剪枝的示例图如下，灰色节点为被剪枝的子树：



装 订 线

2.3 模块设计

2.3.1 后端部分模块设计

后端部分将棋盘信息封装为一个 Board 类，用于存储棋盘上双方棋子及空位的情况，以及不同位置的得分情况；同时该类中还有负责落子、悔棋、初始化棋盘等操作的相应函数，其他搜索函数和启发式函数等都是对该类的实例化对象进行操作，因此该类的设计为本次实验模块设计的核心（详情见注释）

```
//board 棋盘类
class Board {
    /**
     * 初始化棋盘函数
     * @param size  棋盘大小
     */
    init(size) {
        this.current_steps = [] //当前搜索的步骤
        this.all_steps = []     //总步骤
        this.count = 0  //已落子数量
        this.board = array.create(size, size)   //棋盘信息二维数组
        this.size = size    //棋盘大小
        this.com_score = array.create(size, size)   //电脑得分
        this.hum_score = array.create(size, size)   //玩家得分
        // score_cache[role][dir][row][col]
        this.score_cache = [
            [],
            [
                array.create(size, size),
                array.create(size, size),
                array.create(size, size),
                array.create(size, size)
            ],
            [
```

```
                array.create(size, size),
                array.create(size, size),
                array.create(size, size),
                array.create(size, size)
            ]
        ]
    }

    /**
     * 更新一个点附近的分数
     * @param p      描述棋子的对象
     */
    updateScore(p) { }

    /**
     * 落子
     * @param p      描述棋子的对象
     * @param role   当前角色
     */
    put(p, role) { }

    /**
     * 移除棋子
     * @param p      描述棋子的对象
     */
    remove(p) { }

    /**
     * 回退(悔棋)
     */
    backward() { }

    /**
     * 为某一角色估分
     * @param role   当前角色
     */
    evaluate(role) { }

    /**
     * 判断某个位置周围是否有足够的邻居
     * @param r      该位置行数
     * @param c      该位置列数
     * @param dis    距离
     * @param cnt    所需邻居数量
     */
    haveNeighbor(r, c, dis, cnt) { }
}
```

2.3.2 前端部分模块设计

前端部分使用 html+css 对 UI 进行展示，html<body>代码如下，从中可以看出游戏界面的大致情况。

```
<body>
    <header>
        <h2>五子棋 AI</h2>
    </header>
```

```
    <canvas id="chessboard" width="450px" height="450px"></canvas>
    <header>
        <div id="button" onclick="startGame(true)" name="meFirst" style="displa
y:inline-block">电脑先行</div>
        <div id="button" onclick="startGame(false)" name="computerFirst" style=
"display:inline-block">玩家先行
        </div>
        <div id="button" onclick="cleanBoard()" name="restart" style="display:n
one">重新开始</div>
        <div id="button" onclick="goBack()" name="goBack" style="display:none">
悔棋</div>
    </header>
</body>
```

2.4 其他创新内容及优化算法

（1）采用改进的极大极小搜索算法 negamax 搜索算法，减少一个搜索函数，有利于提高代码可维护性（不需要像原来一样每次修改搜索逻辑要修改两个函数）。

（2）使用迭代加深搜索优化极大极小搜索，可以在复杂度增加不大的情况下达到能获胜时找到最快获胜方案、必输时找到尽可能拖延方案的效果。

（3）引入某一方向得分的概念，可以将一个一个点的得分按照四种不同的延伸方向分为四个不同的得分，每次更新时只需对相应方向进行更新，优化棋盘得分更新速度。

（4）对于一个空点只有在该点有邻居的情况下才对该点进行考虑，避免引入过多低分点影响搜索速度。

（5）当极大极小搜索到一定深度时，只考虑能成三个以上相连的情，因为一般情况下不会在最优决策几层后依然没有三连的情况，由此可以减少搜索状态数，提高搜索速度。

3 实验过程

3.1 环境说明

3.1.1 操作系统

Windows 10

3.1.2 开发语言

JavaScript、html、css

3.1.2 开发环境

编辑器：VS code（当前最新版）

浏览器：Chrome（当前最新版）

核心使用库：canvas 用于画图

3.2 源代码文件清单及主要函数清单

(1) index.html

前端网页，是 UI 显示的主要部分。

(2) style.css

css 文件，包含网页中各种部件的显示格式。

(3) web-design.js

后端与前端相连的 js，本项目逻辑判断的核心，包含 startGame、canvas.click 等与前端按钮及点击事件相绑定的函数，以及 paintPiece、drawBoardLines 等和棋盘在画布上显示的函数，还有 oneStep、isOver 等将前端消息逻辑判断整合到后端处理的函数。

(4) board.js

包含项目的核心类 Board 及对棋盘得分进行调整的 fixScore 函数。

(5) nagamax.js

搜索决策模块，包含极大极小搜索函数 negamax 和搜索决策函数 searchMethod。

(6) evaluate-point.js

对一个点计算得分的模块，包括两部分，统计该点附近同色点的分布情况和将分布情况转化为得分。

(7) Mymath.js

包含对得分进行比较的一系列函数功能，由于对得分进行比较实际上是对该得分包含的情况进行比较，为了防止较低得分情况的不同对比较结果的可能的影响，该比较为模糊的比较，即误差在一定范围内即认为相等。

(8) config.js

包含项目中与搜索剪枝相关的几个主要参数。

(9) array.js

包含一个构造二维数组并初始化为 0 的函数。

(10) score.js

包含各种情况下的得分，例如成五得分最高，为 10000000，而单点得分很低，仅有 10。

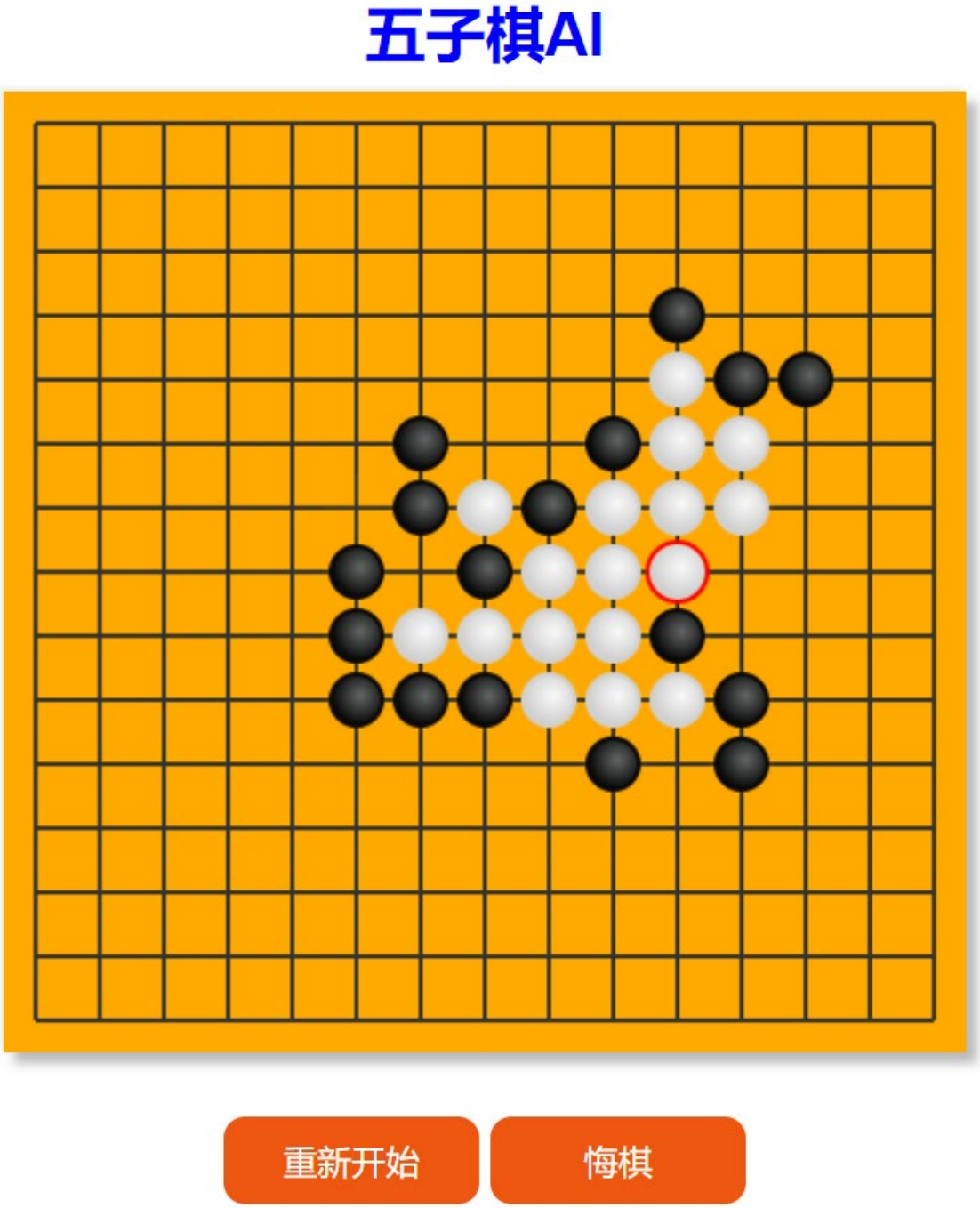
(11) role.js

包含棋盘落点数字所代表的含义，0 代表空点，1 代表落点为电脑，2 代表落点为玩家。

3.3 实验结果展示

3.3.1 程序展示

可以通过浏览器打开 index.html 运行该游戏看到实际 UI，前面 UI 设计部分也有一定的展示，因此此处仅给出一种运行中截图：



3.3.1 实验结论

本次实验按照实验要求，成功利用 α－β 剪枝算法完成了一个五子棋人机博弈游戏的开发。

4 总结

4.1 实验中存在问题及解决方案

（1）最初进行实验时没有注意区分内外层函数的 this 对象，导致搜索结果混乱，后来在查阅资料后得知内层不能直接使用外层函数的 this 对象，需要先将外层 this 赋值给 self 后进行使用。

（2）最初试验时 AI 下的棋不合常理，且判断胜负莫名其妙，经过观察，如果下的棋按照棋盘对角线对称后是合理的，于是经过分析代码，发现有一处地方画布 xy 坐标和 Board 对象中的二维数组下标对应关系搞反了，导致画布上绘制的棋子位置并不正确，修改后解决。

（3）最初进行实验时搜索返回的策略经常不够优，在经过仔细调试后发现得分只返回了当前方向的得分，而不是四个方向的得分之和，在进行相应修改后解决。

（4）最初进行实验时没有采用迭代加深的搜索策略，导致 AI 在能够立刻取胜的时候仍会使用冲四等方法浪费时间，这是因为搜索固定步长找到的获胜策略不一定步数最少；经过查阅资料将搜索修改为迭代加深搜索后解决。

（5）最初棋盘上悔棋将棋子擦除后补画的黑线会比其他地方的黑线更“黑”，视觉效果不好，查阅资料后得知是黑线与背景色重合的原因，因此先画一遍白线再补画黑线得到解决。

4.2 心得体会

（1）本次实验使用极大极小搜索和α－β 剪枝算法实现了一个五子棋 AI，在算法的实际使用中大大增强了对这两个算法的认识与理解

（2）实验中对于五子棋 AI 的启发式函数进行了设计，学会了许多根据实际问题设计启发式函数的方法。

（3）实验中学习使用 js、html、css 前端三件套开发 web 应用，并根据 web 应用的特点进行 UI 设计，学到了许多开发人机交互界面的新方法。

（4）实验中为了解决 AI 的 bug 和棋力较弱的问题我查阅了一些资料，学到了许多搜索过程中的优化技巧。

4.3 后续改进方向

（1）可以在现有 AI 算法中添加现实五子棋比赛的“黑棋禁手”规则，使游戏更加公平合理。

（2）可以在程序中添加双人对战功能，虽然这与本课程无关，但是可以使五子棋游戏更加符合实际需求。

（3）由于本人第一次使用前端三件套进行 web 应用开发，代码中的许多地方的实现不够合理，需要进一步优化。

（4）本次实验没有使用 ES6 中 js 代码的 import 和 export 功能，因此 js 模块的封装性不好，需要改进。

（5）本次实验较多借鉴了网上的一个开源项目，虽然我完全是在读懂源项目后自己动手写的，但是受原项目影响，我的代码中的许多函数名和模块名和原项目十分相似，而且含有一些原项目的缺点，需要进一步改进。

4.4 实验总结

通过本次实验，我在实践中对极大极小搜索和α－β 剪枝算法的原理与实现有了更深刻的认识，并通过实际实现算法学会了许多剪枝技巧。同时通过实验中对 JavaScript、html、css 的学习与研究，我对于 web 应用的开发有了许多了解，并学会了设计 web 应用满足人机交互的需求。

5 参考文献

[1] Stuart J. Russell，Peter Norvig. 世界计算机教材精选 人工智能 一种现代的方法 第3版. 北京：清华大学出版社，2018.07.

[2] https://github.com/lihongxun945/gobang

6 成员分工与自评

6.1 成员分工

成员：X(组长)

分工：由组长 X 同学实现本次实验的全部工作（包括程序设计、前后端代码编写实现、报告撰写、汇报 PPT 制作等）

6.2 自评

本次实验总体完成情况良好，能够完成实验要求实现的全部内容。

但是由于我没有选择自己习惯的程序语言(C++)，而是选择学习新的语言进行开发，导致实验时间超出预期：本来实验已经完成了，但是我为了优化搜索修改了一些地方且没有备份，导致程序出现重大问题，而我对于 js 代码的调试不够熟悉，导致了一些时间的浪费，最终时间花费较大。

总体来看，在独立完成本次实验的过程中我对于实验的整体有了全方面的认识，同时对于多种能力都有所提升，缺点是没有控制好新语言的特性导致时间规划不够好。