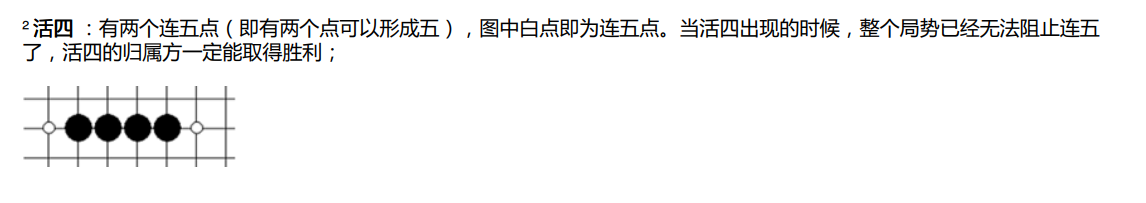
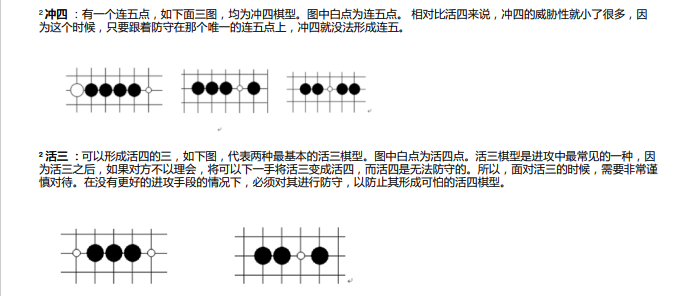
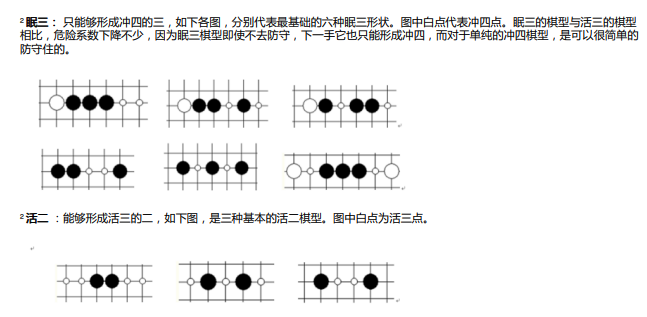
按钮功能：

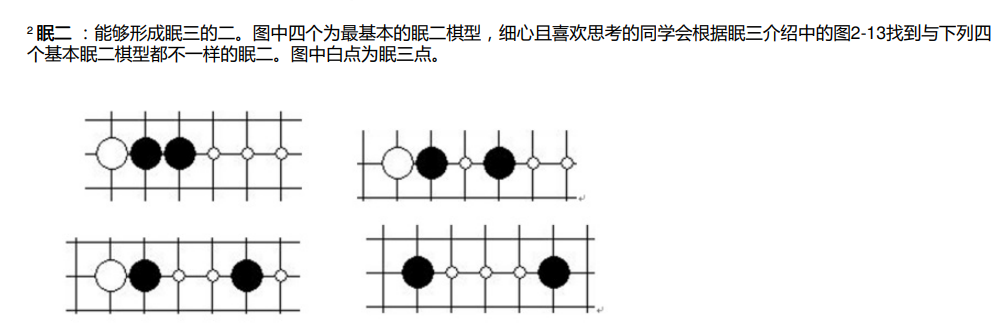
1. 设定AI先手或玩家先手
2. 重置棋局
3. 退出程序

棋局的种类：







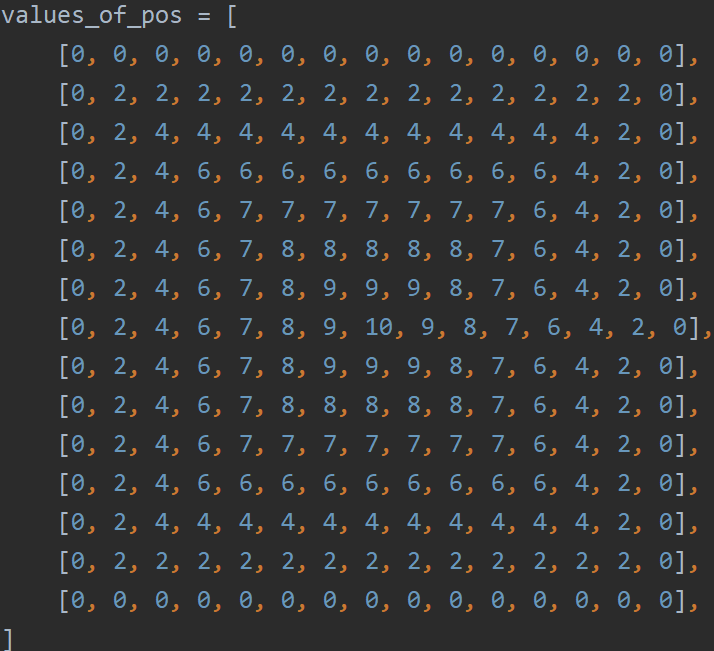


不同棋局得分的设定：

|  |
| --- |
| 连五 |
| "11111": 999900000, |
| 活四 |
| "011110": 33300000, |
| 冲四 |
| "211110": 6250000, |
| "11101": 6250000, |
| "11011": 6250000, |
| 活三 |
| "01110": 625000, |
| "011010": 625000, |
| 眠三 |
| "211100": 12500, |
| "211010": 12500, |
| "210110": 12500, |
| "11001": 12500, |
| "10101": 12500, |
| "2011102": 12500, |
| 活二 |
| "001100": 250, |
| "01010": 250, |
| "010010": 250, |
| 眠二 |
| "211000": 25, |
| "210100": 25, |
| "210010": 25, |
| "10001": 25, |
| 其他 |
| "others": 8 |

位置权重，越靠近中间的位置越重要，给予更大的权重。

（该图片可以换成棋盘表格，在每个方格上进行标注）



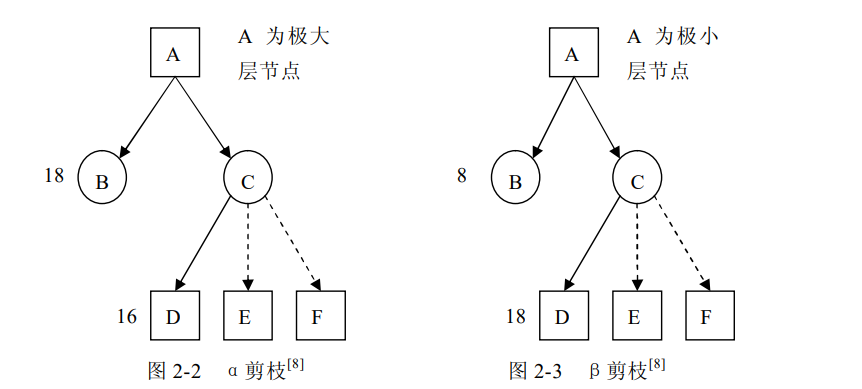
一些重要参数的设定：

搜索深度：3

搜索范围：4（AI下一步走的棋的位置不超过当前棋局范围四格）

Alpha-beta剪枝：

博弈树α-β剪枝的原理图示（摘自哈工大毕业论文）



程序使用递归的方式实现这一功能，alpha记录当前节点最小值，beta记录当前节点最大值，若 alpha>=beta，则进行剪枝操作。

主要变量和函数含义：

me\_first/ai\_fisrt ：记录玩家/ai先手情况

Myturn/AIturn ：记录玩家/ai回合

game\_state : 记录游戏状态 0表示未分胜负，1表示黑子win，-1表示白子win

whiteq ：存储白棋位置

blackq ：存储黑棋位置

chess ：用来判断棋盘某处是否有棋,1为黑棋, -1为白棋, 0为空

更新chess状态 renew\_chess(chess, blackq, whiteq)

画棋盘背景draw\_background(screen)

画棋子draw\_gos(screen, blackq)

获取以p点为中心的9个点的连线棋形 getline(chess, px, py, dir\_offset, discolor)

分析棋形并得到该方向的分值,最后的参数是,有利分为1不利分为-1

get\_line\_score(line, adcolor, discolor, weight, px, py)

评估函数，评估当前棋局的分值Evaluate(chess)

获取下一步所有可能走法的点位get\_next\_pos(tchess, b\_num)

将p点放入棋盘布局 set\_newchess(new\_chess, p, max\_or\_min)

alpha-beta剪枝搜索 alpha\_beta(depth, max\_or\_min, chess, alpha, beta)

获取对应value值的走法的位置 get\_point(value)

AI下棋 draw\_ai(blackq, whiteq)

获取棋子在某个方向上连续的最大个数 one\_dir\_num(x, y, dx, dy, chess, b\_or\_w)

判断胜负 check\_win()

算法运行图示

