人工智能实验（十）

实验报告

姓名：钟怡宁

学号：17341217

班级：计科八班

专业：计算机科学与技术

一：实验目的

掌握博弈树算法，实现基于alpha+beta剪枝的五子棋人机对战程序，在运行前，可以指定任意方为先手。其中人执黑棋，电脑执白棋保持不变。

二：实验原理

1. 博弈树

在博弈树搜索中，内部节点和叶子节点表示问题的状态，每一步的行动就是在扩展节点， 对弈双方轮流扩展节点，两个玩家的行动逐层交替出现，需要设计一个评价函数，对当 前节点的优劣进行评分，博弈树搜索的目的是找出双方都是最优的子节点的值。双方利 益关系对立。就零和博弈来说，博弈双方知道信息所处状态的全部信息，结局由三种可 能，玩家A获胜，玩家B获胜，或者平局，一方的损失相当于另一方的收益。

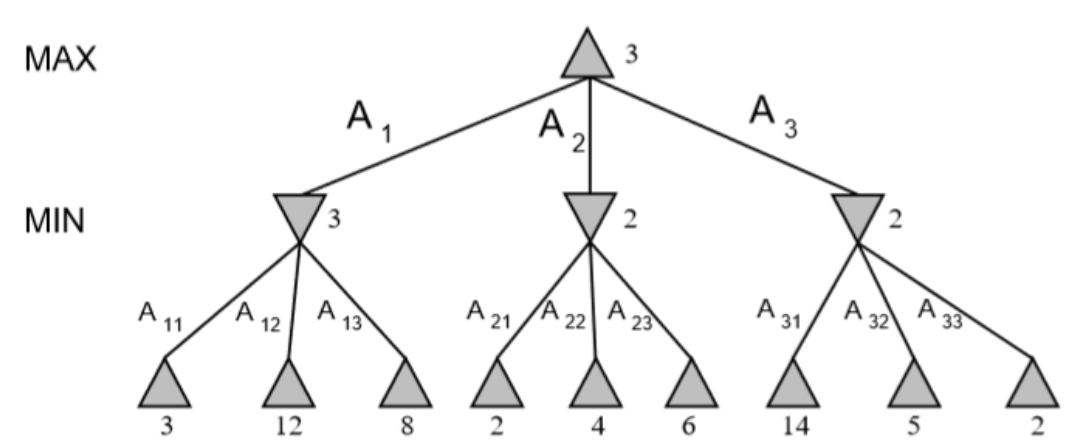
1. alpha+beta剪枝

由于Minimax搜索必须检查的游戏状态的数目随着博弈的进行呈指数增长，所以要 用到alpha+beta剪枝操作，尽可能的消除部分搜索树，来提高运算速度。

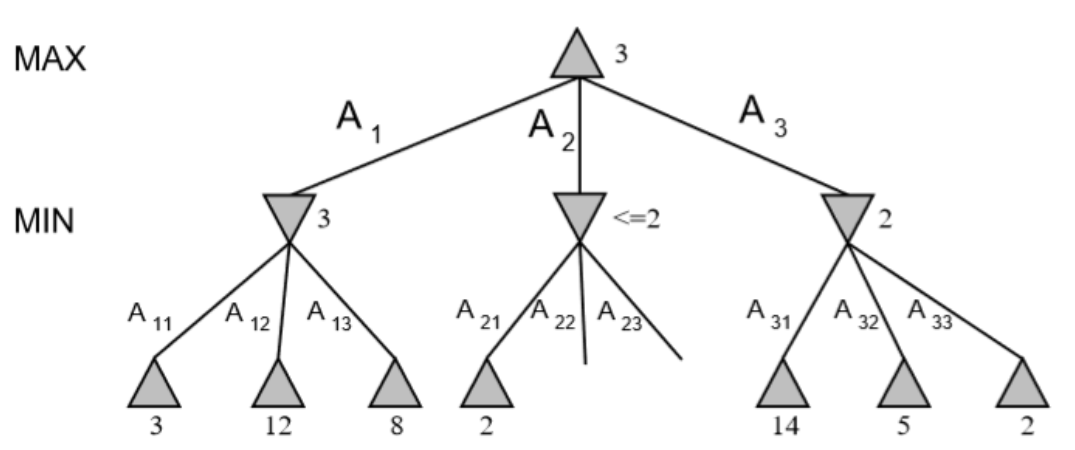
它基本的原理是：当一个 Min 节点的 β值≤任何一个父节点的α值时 ，剪掉该节点 的所有子节点，当一个 Max 节点的 α值≥任何一个父节点的β值时 ，剪掉该节点的 所有子节点。

下面通过两个图来对比剪枝前后的效果

未使用alpha+beta剪枝：



使用alpha+beta剪枝：

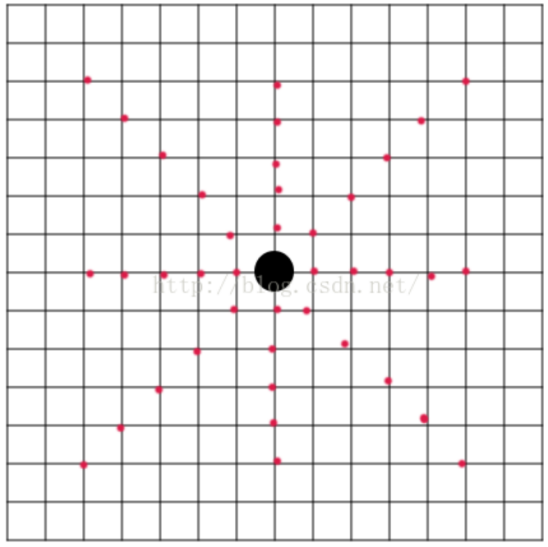
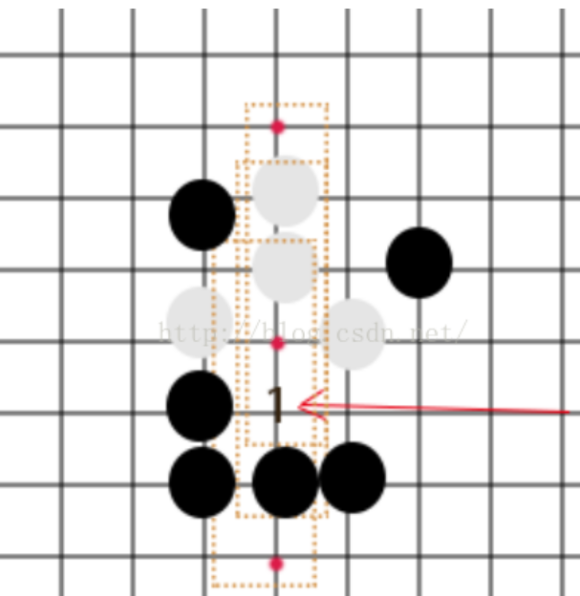


分析：首先第一层是MAX层，第二层是MIN层，由A11,A12,A13可知，A1分支的cost 最佳为3，则MAX层的根节点的alpha值大于等于3，在遍历了A21分支之后，可以知 道A2分支节点的beta值小于等于2，小于其父节点的alpha值，因而其A22和A23子 节点被剪掉了。因而完成了剪枝操作。其他剪枝操作与之类似，alpha+beta剪枝的基础 是深度搜索，可以看出遍历子节点的顺序会影响剪枝的效果。

1. 搜索得分形状

在搜索中，会利用alpha+beta剪枝来减去多余的，不需要再遍历搜索的节点，然后利 用minimax搜索策略来找对于当前玩家来说，最佳的落子点。基本思想：在遍历每一个 节点时，进行窗口滑动式形状搜索，搜寻4个方向上所有可能的得分形状。

如下图所示：对于一个点的上下，我们认为是一个方向，因此一共有四个方向，对于待 选位置1，我们搜索以它为底端的五元组，即第一个矩形，然后是第二个，第三个，一 直到以1号位置为顶端的五元组，这样我们就统计了上下这一方向的所有的可能得分的 形状。由于在实际的五子棋评分操作中，要考虑活四的情况，会有6个节点参与得分 的评定，所以在滑动窗口时，一般会往上，往下各扩展一个节点，即以1号位置为顶端 的六元组，然后根据实际需要从六元组中截取自己需要的部分。

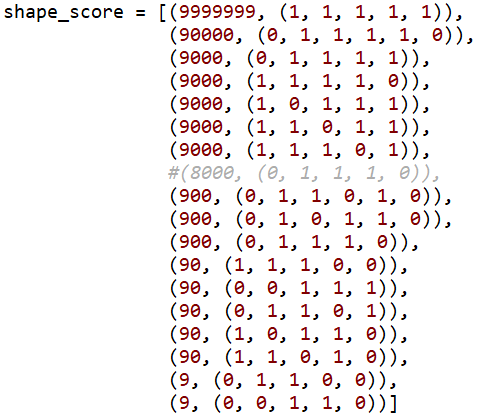
 

搜索得分形状部分即是如此操作。对于当前局面中的每一个我方棋子和当前候选位置都 要进行该搜索操作，以便计算当前候选节点如果落子，当前局面会得到的分数。

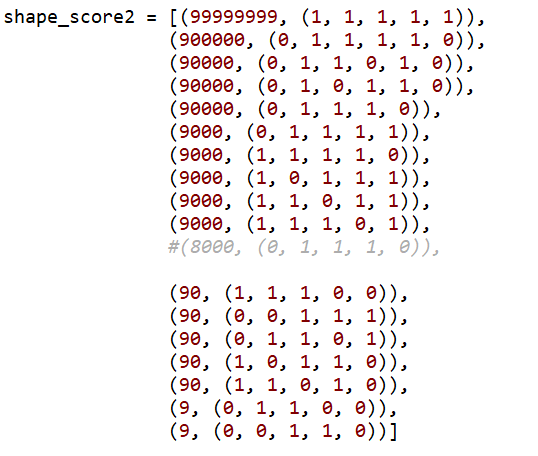
1. 评价函数设计

首先需要一张得分形状对应表，我们用1来表示己方棋子，用2表示敌方棋子，用0 表示可落子，由此来设计我们自己的得分形状对应表。考虑到五子棋的一般棋形分布， 我们考虑这几种情况，活二，活三，活四，死三，死四，可转为活四的类死三，连棋这 几种基本情况，在设置分数时，我们以10的倍数的形式不断增长，考虑到五子棋的有 攻有防，这个度可能不好把握，所以我设计了两张得分形状对应表，使得己方暂时对 敌方活二，死三，死四的状况稍加以放置，而对其他情况相对重视，防止对方出现 活三，活四，连棋等情况，具体的思路是增加敌方得分对应表中相关得分情况的值，我 增加了一个数量级，即跟己方得分表相比，对应的相应形状，得分扩大了10倍，设计 的得分表如下：

己方得分表：



敌方得分表：

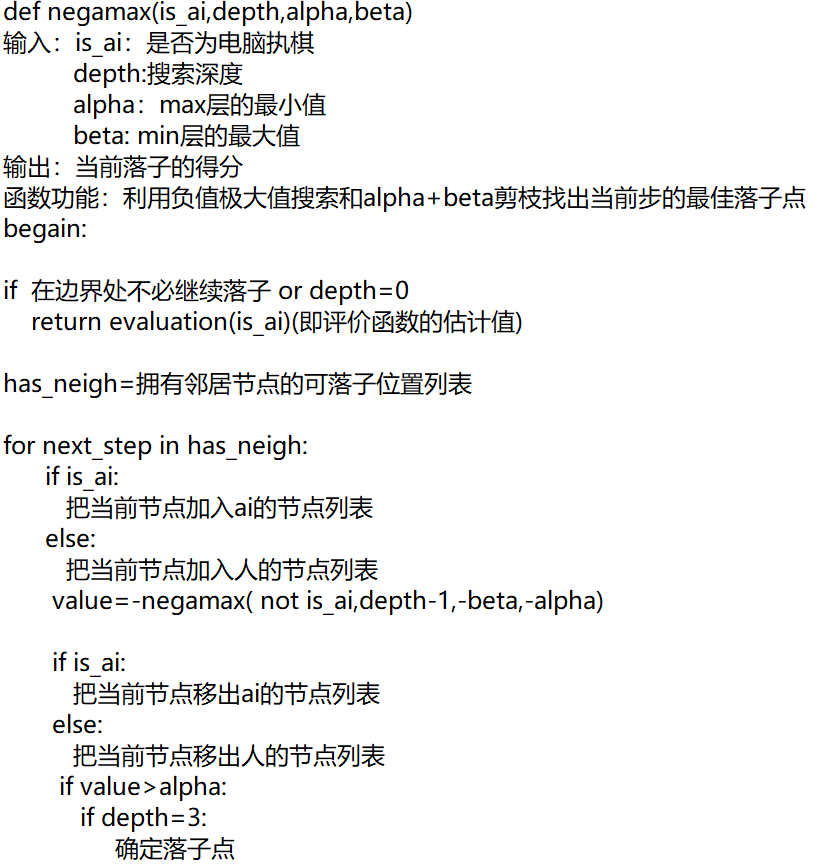


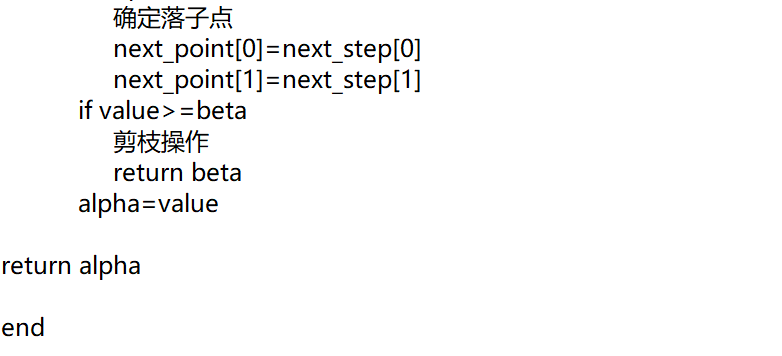
利用搜索得分形状中的方法，在找到当前局面所有的得分形状之后，就可以利用提 供的得分形状表，找到对应的分数。利用此方式，找到每个节点每个方向上对应的最大 的敌方和我方分数，再求四个方向上己方和敌方的分数总和，然后做差，得出的结果为 最终得分，该得分为当前落子的得分，并非总分，该程序只计算单步落子的得分，并以 此为依据找到最佳的落子点。

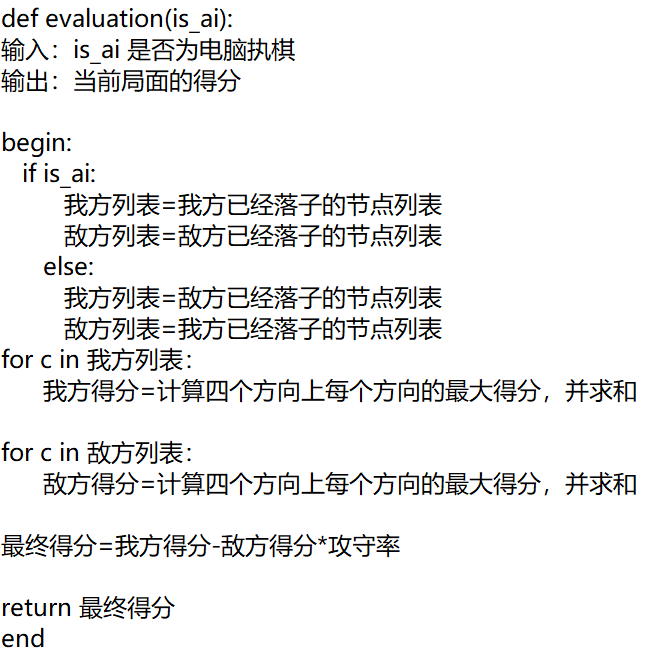
在搜索最佳落子点的过程中，使用负值极大算法搜索 和alpha + beta剪枝，搜索深度必 须为奇数，这里设为3，其实就是Minimax搜索算法，原理是一样的。这里使用递归算 法。再进行搜索之前，进行了一些预处理，主要是两部分，第一，把当前候选节点中有 邻居的分出来，没有邻居的去除，第二把最后和倒数第二落子的节点的的邻居节点提到 候选节点的最前面，因为最后落子的节点的邻居节点很有可能是新一轮的最佳落子点。

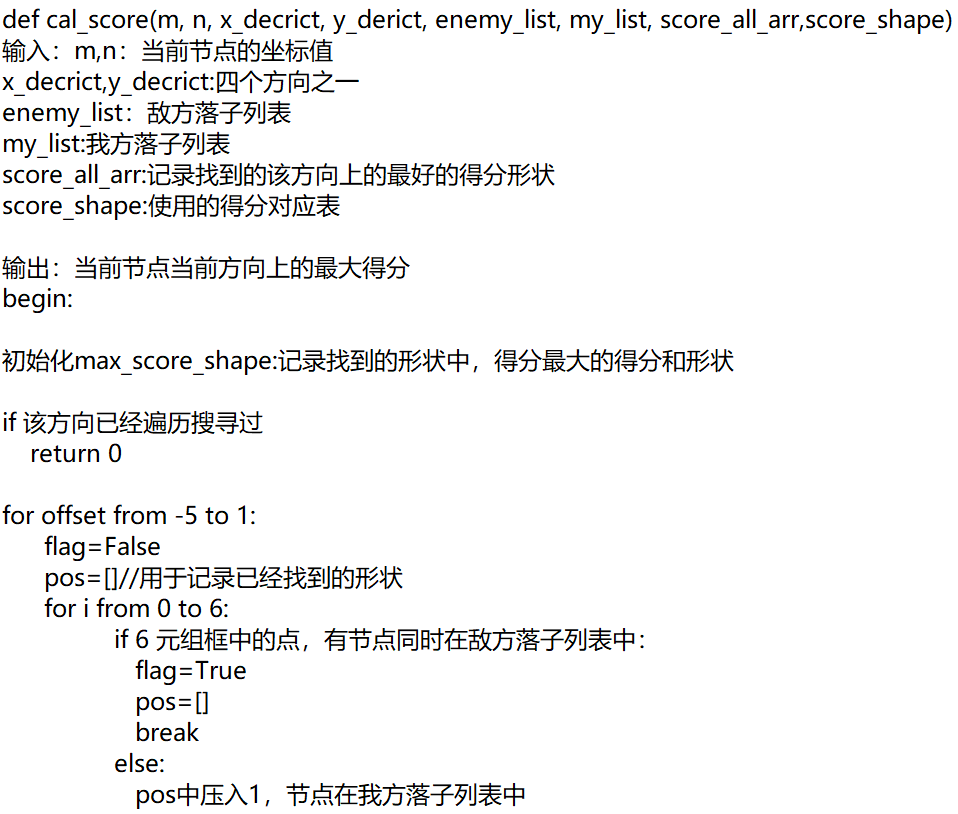
在计算每个方向上的得分的值时，如果该方向上的得分形状与其他方向上的得分形状有 交叉点，则该方向上的得分翻倍增长，使得棋子落在尽可能会得分的位置，更加容易连 棋。

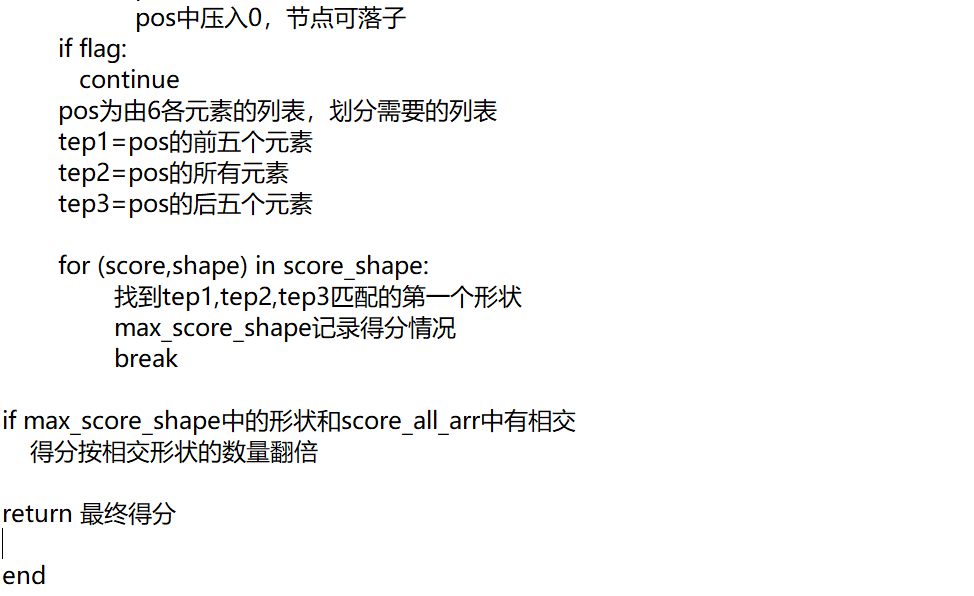
三：伪代码展示



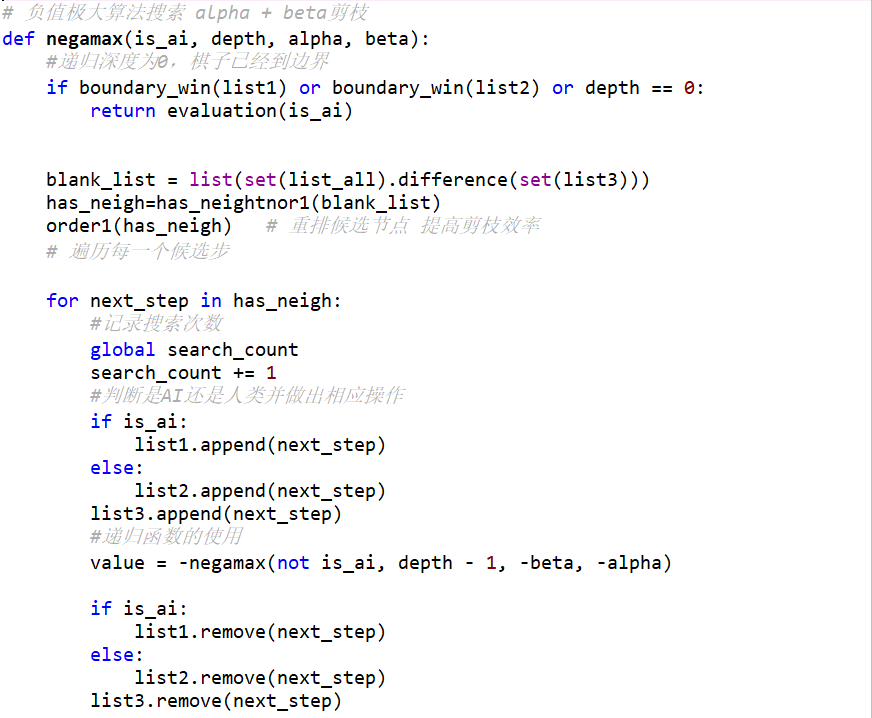


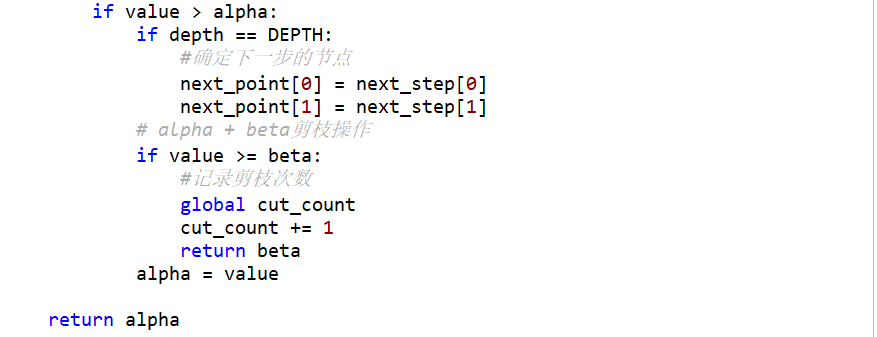


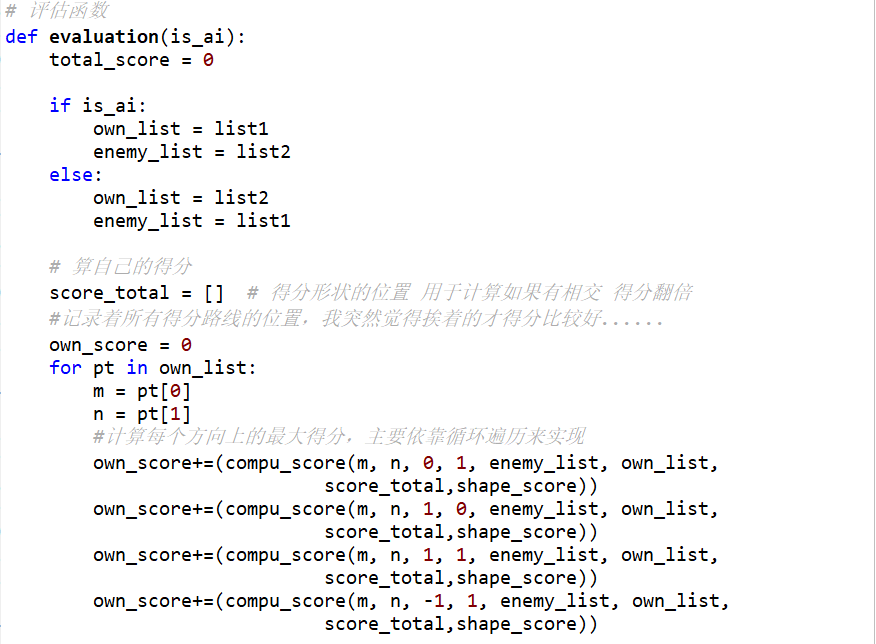


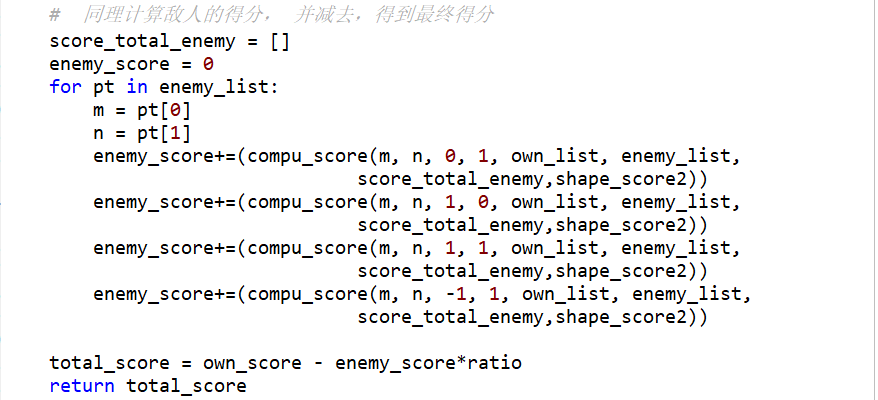


四：关键代码展示







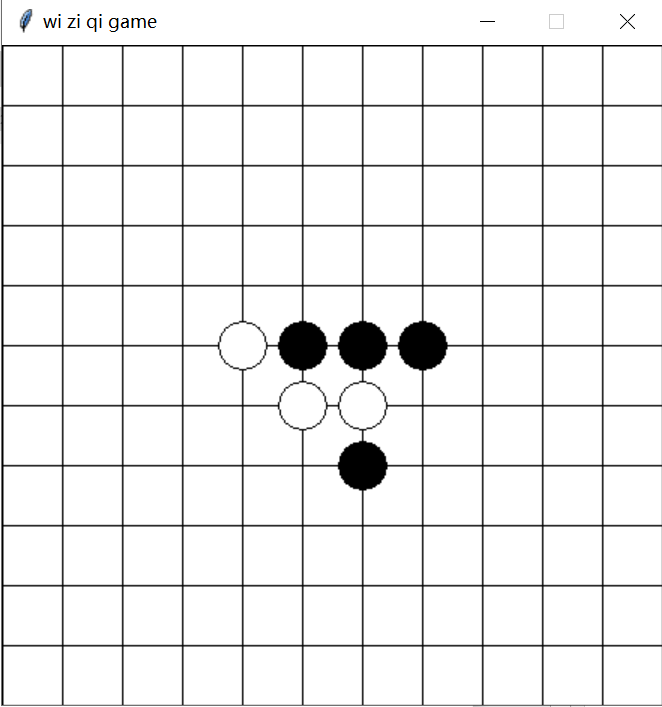
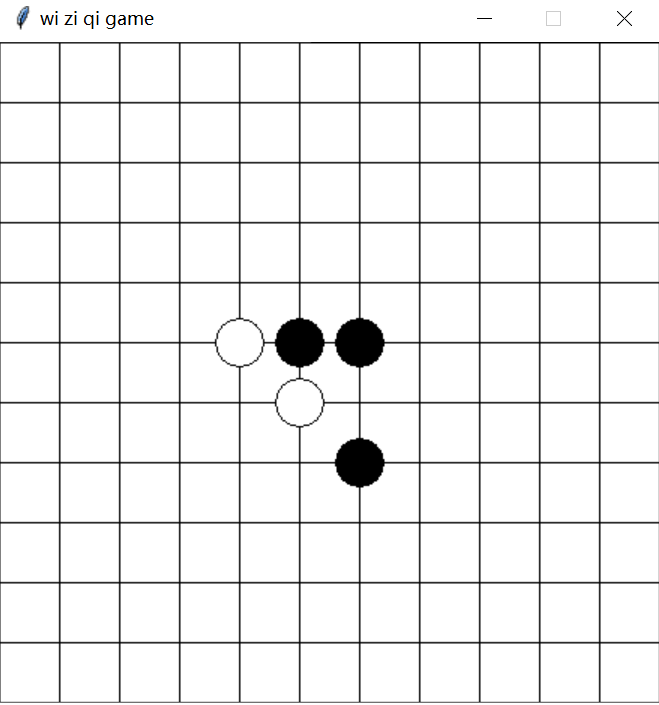


另外一个函数有些冗长，这里不再展示。

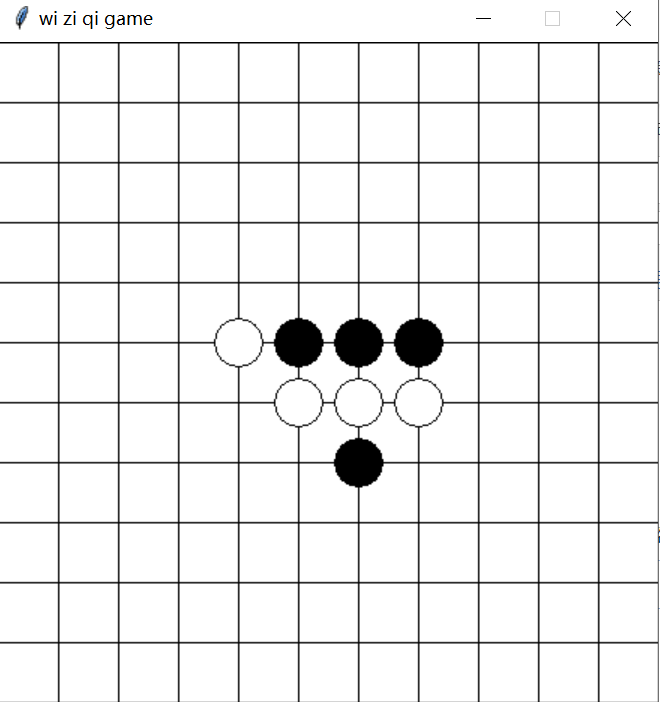
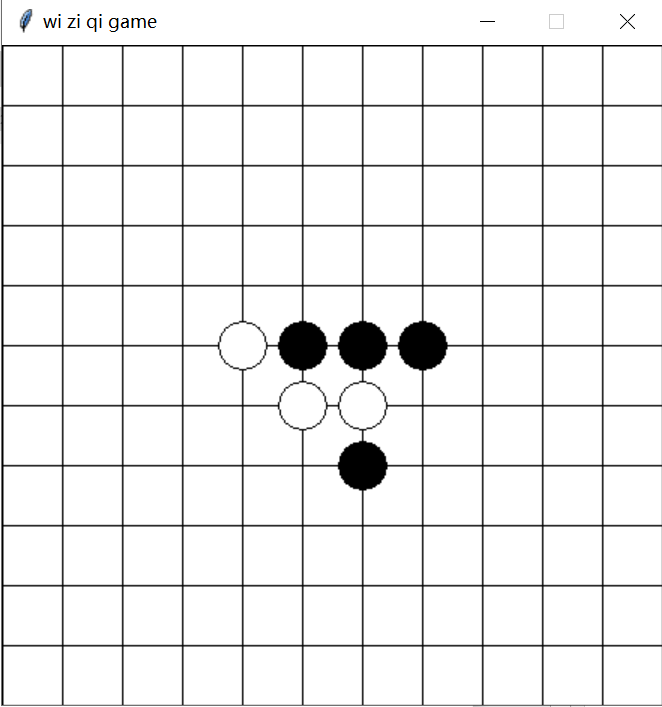
五：结果展示

1. 电脑先手（先手执黑棋）

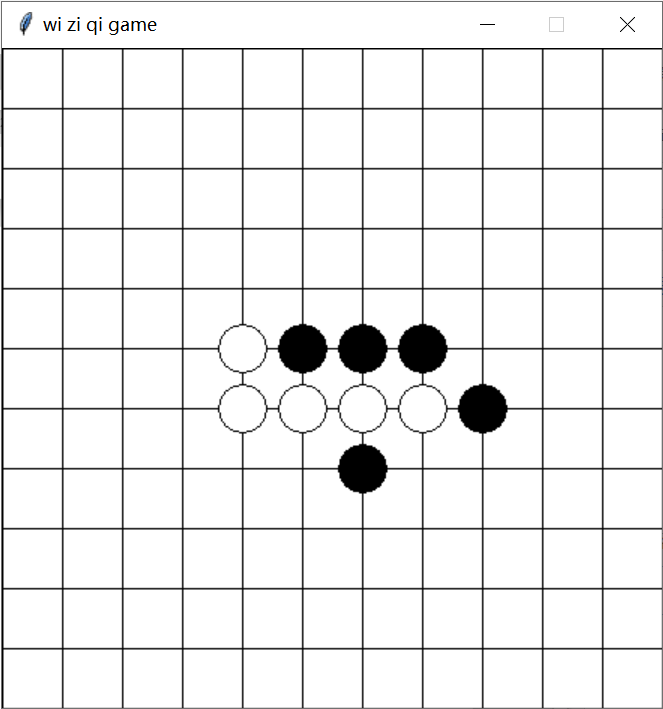
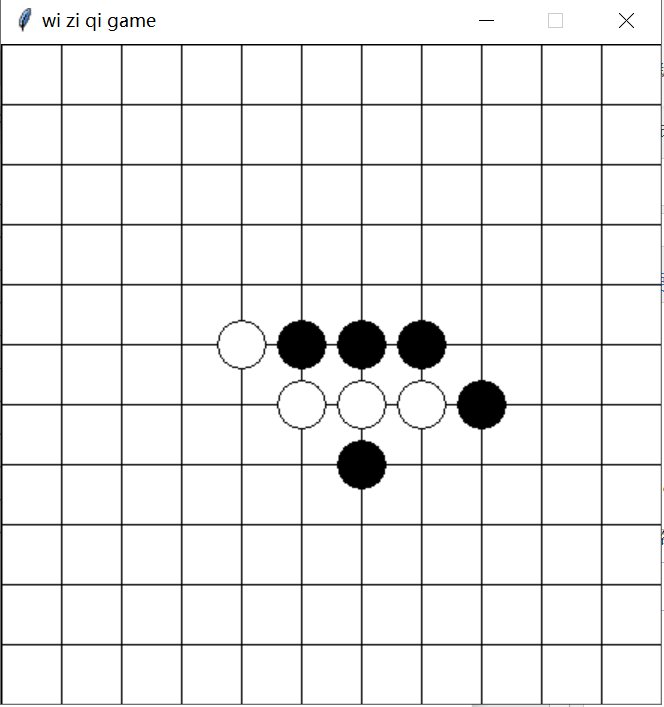
第一回合：



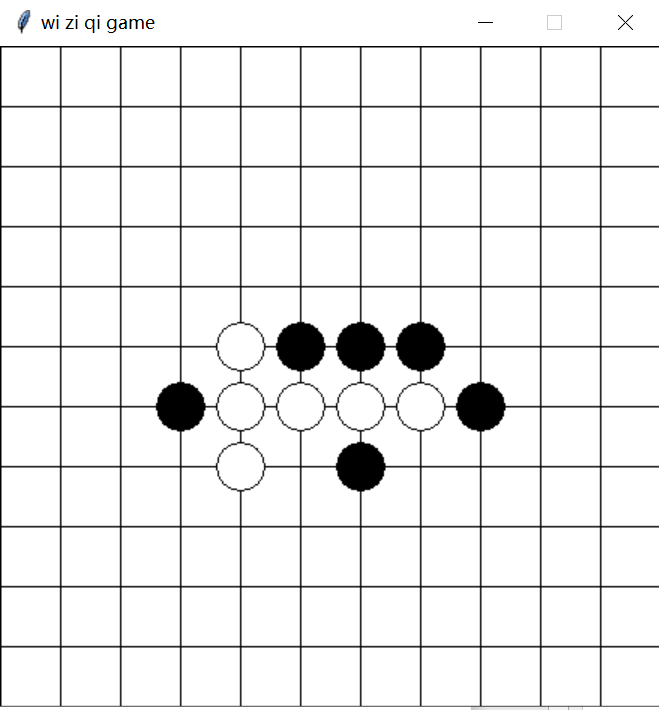
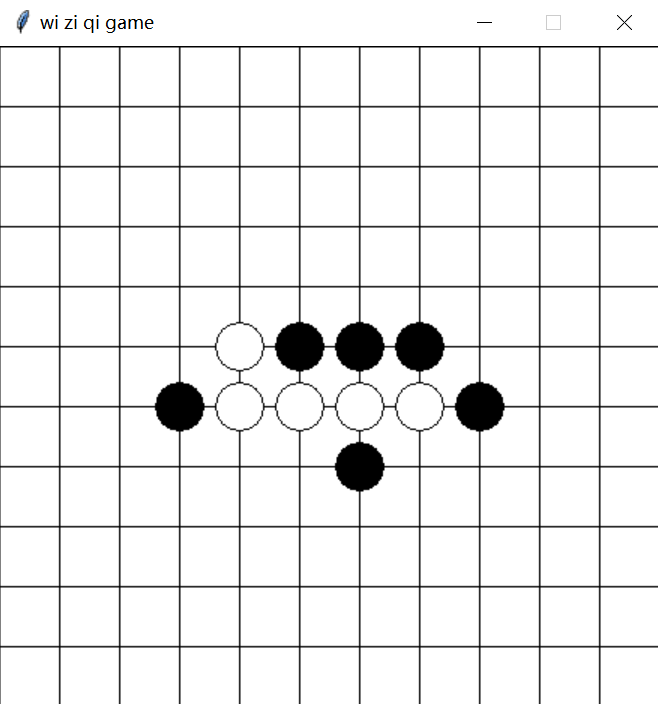
第二回合



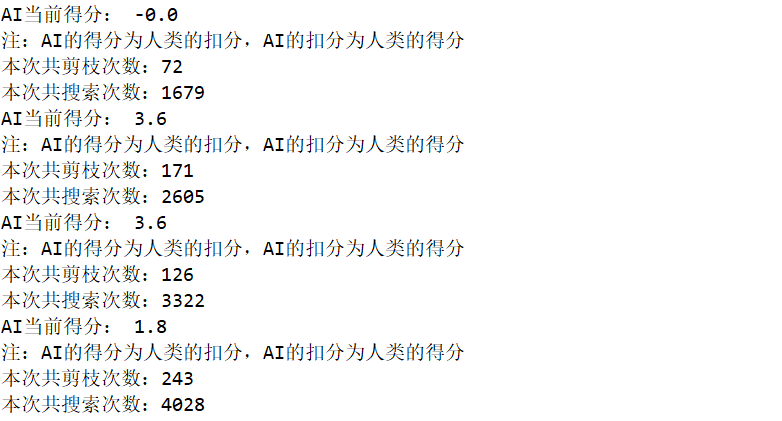
第三回合



第四回合

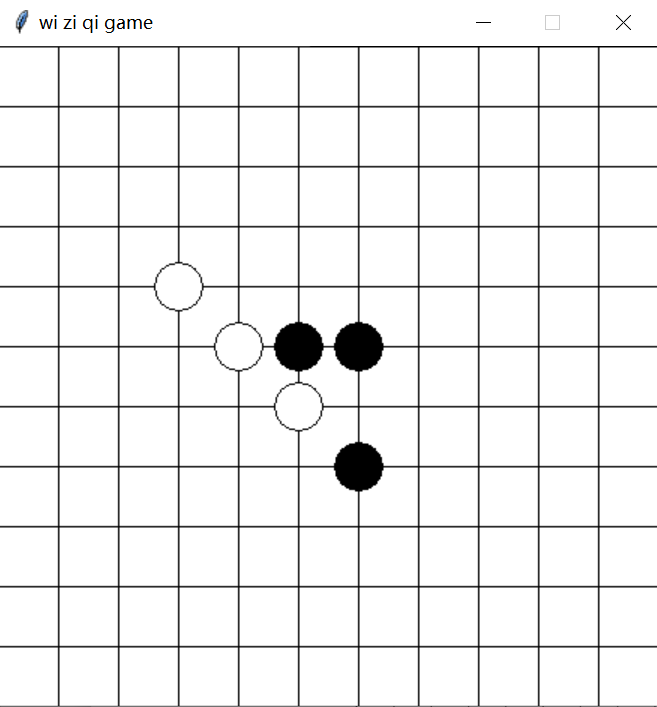
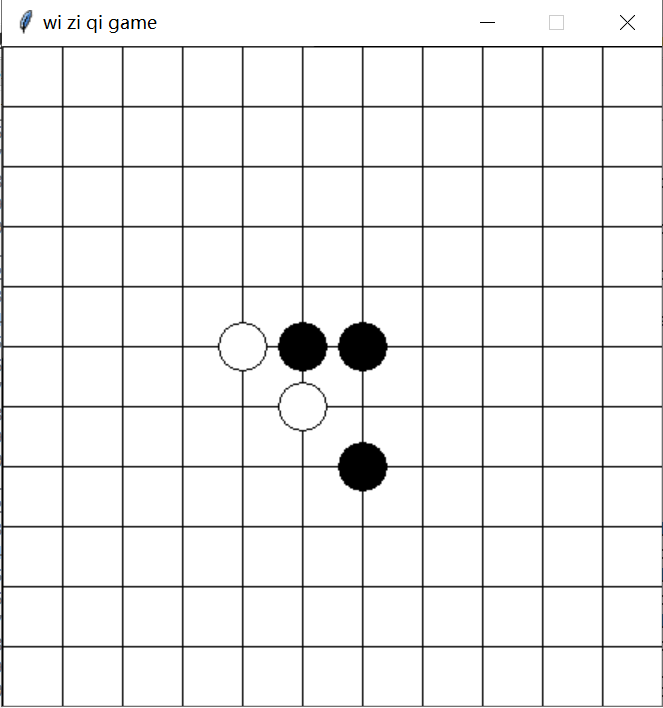


各局得分情况：

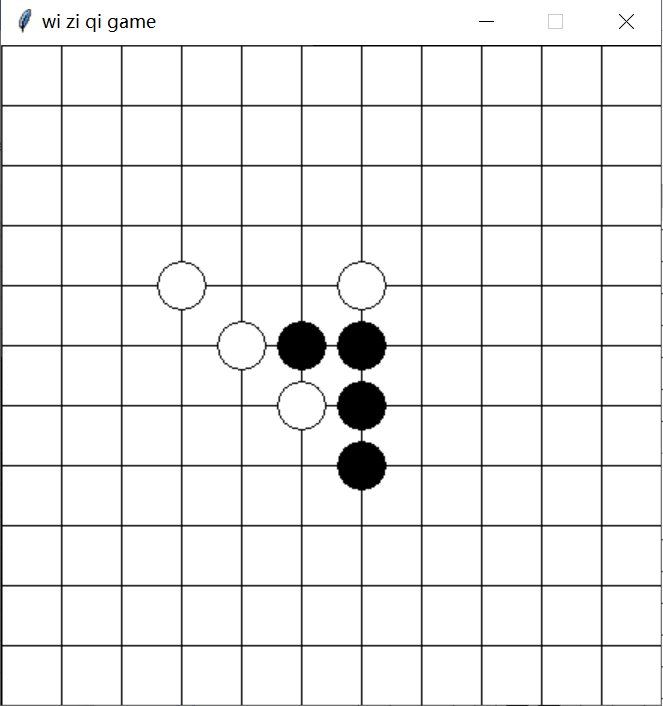
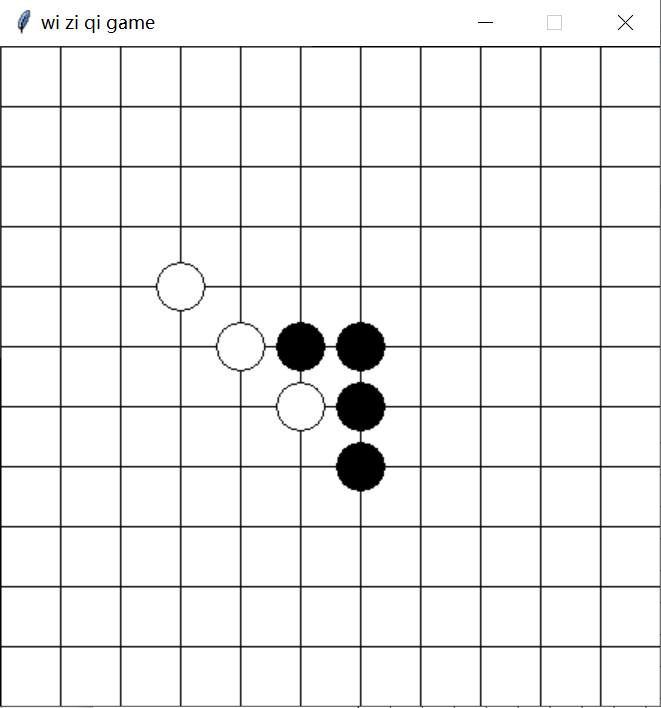


1. 人类先手：

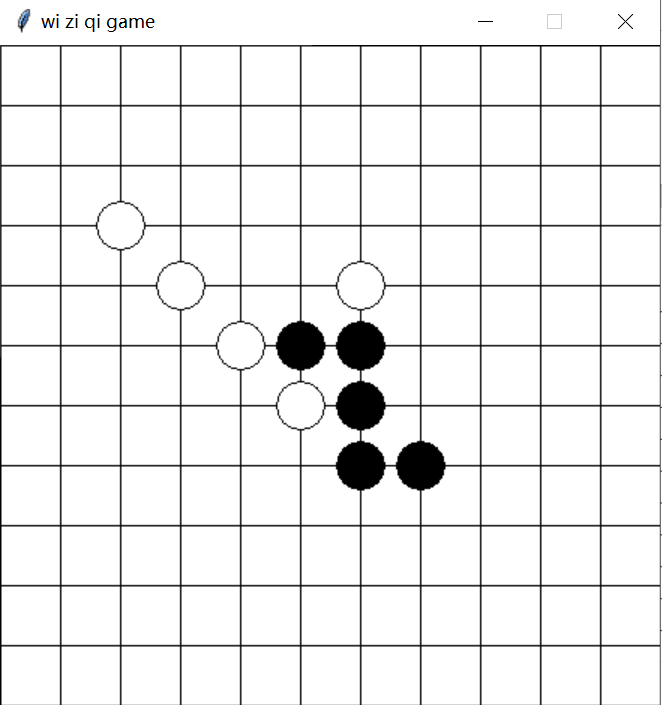
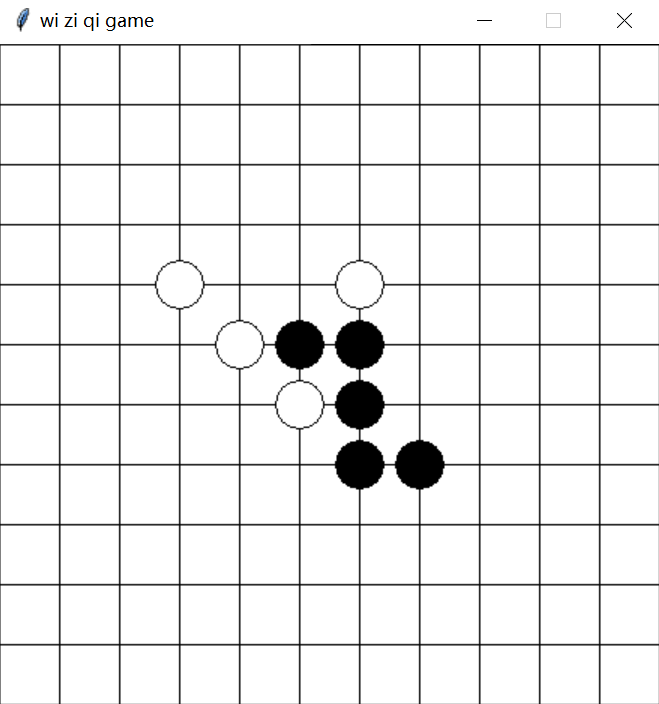
第一回合：



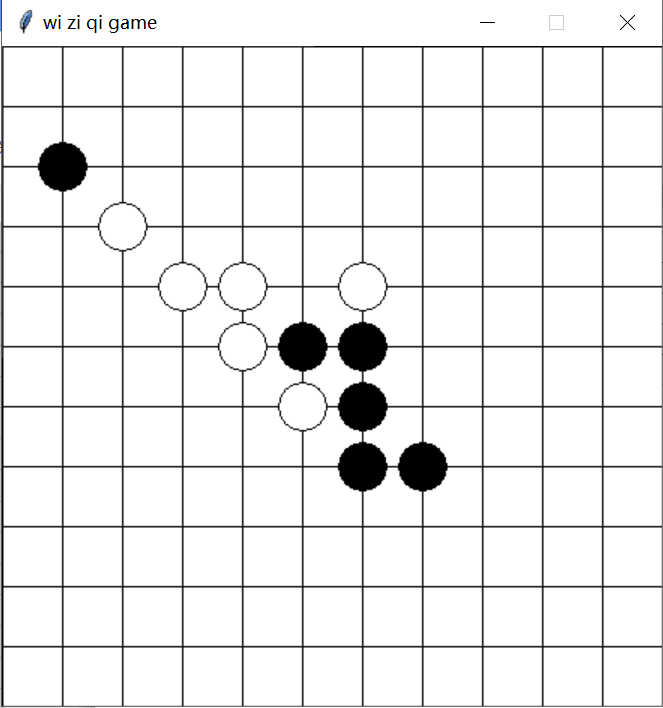
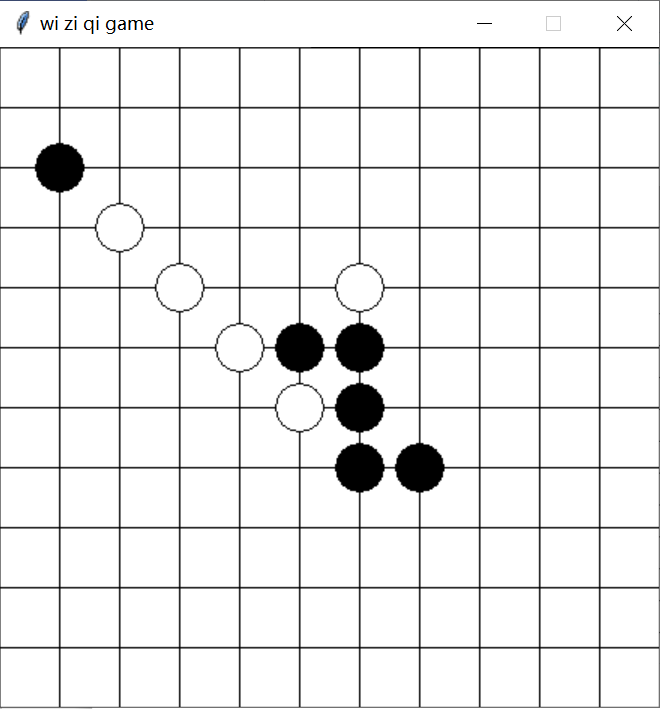
第二回合



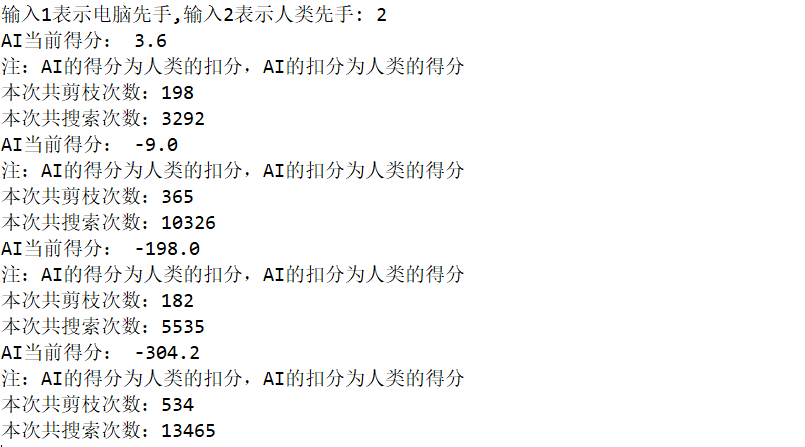
第三回合



第四回合



各局得分如下：



结果展示完毕。

六：思考题

无

七：参考资料

1. PPT
2. https://blog.csdn.net/starh4/article/details/52604649?locationNum=4&fps=1
3. https://github.com/colingogogo/gobang\_AI/blob/master/gobang\_AI.py

加了自己的想法和特有创新