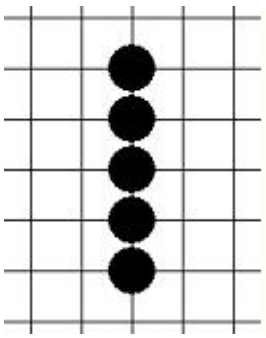
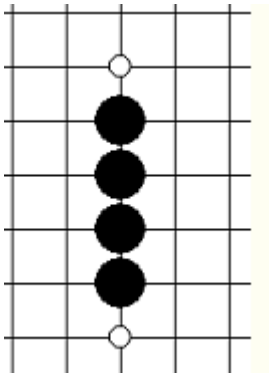
报告一

通过查阅资料，我们知道五子棋的基本棋型：连五、活四、冲四、活三、眠三、活二、眠二，如下图所示

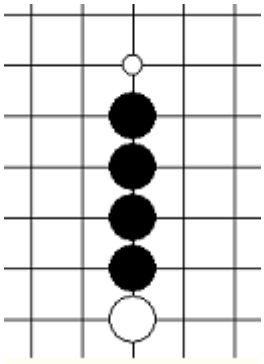
1) 连五

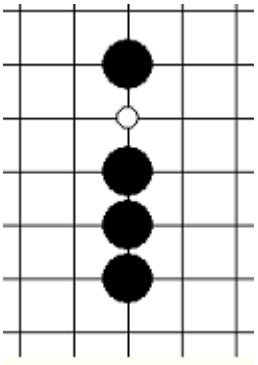


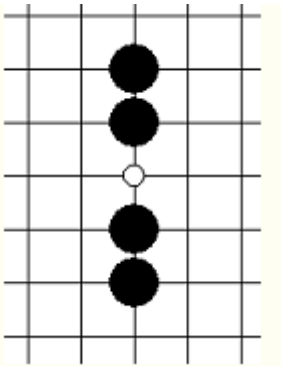
2) 活四



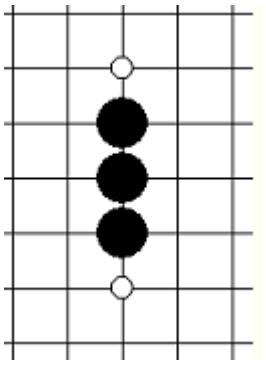
3) 冲四

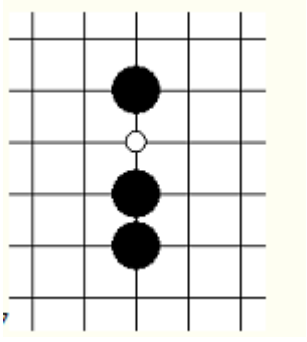




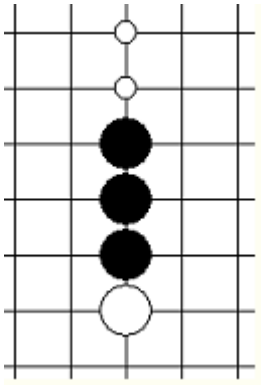
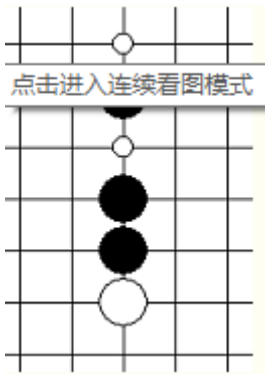


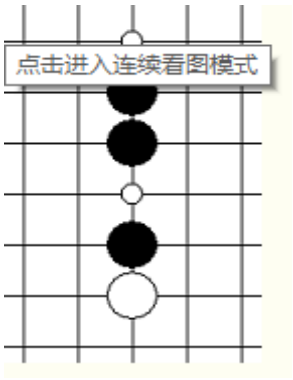
4) 活三

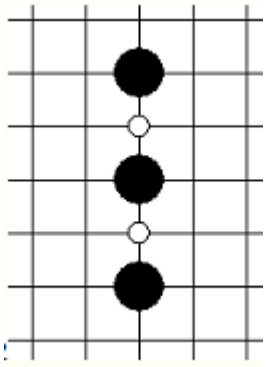
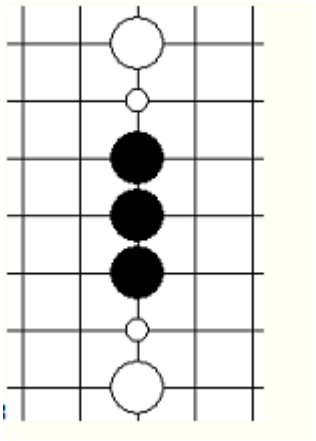
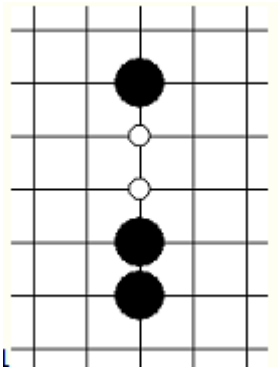




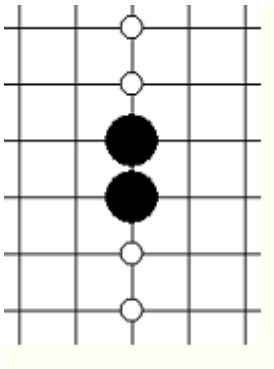
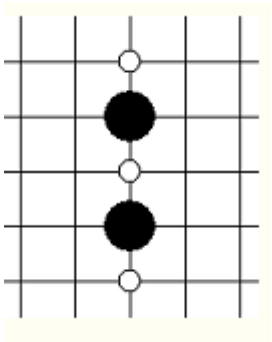
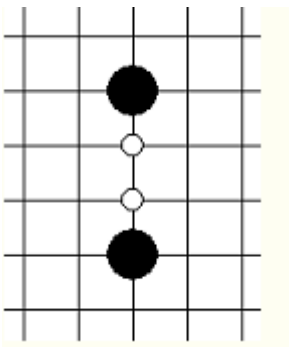
5) 眠三

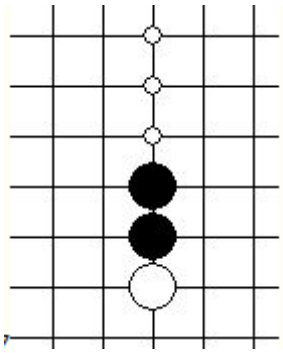
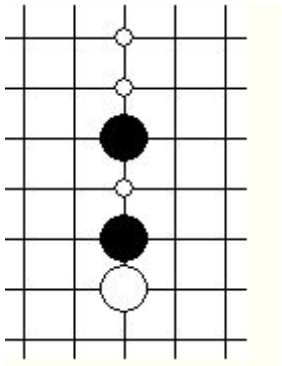
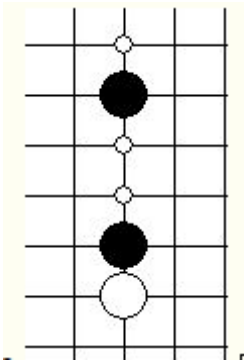


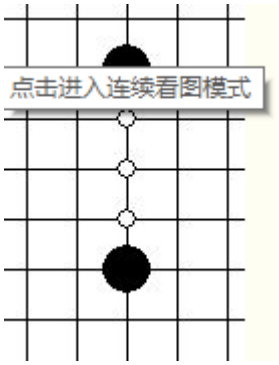




6) 活二

7) 眠二



我们打算给棋盘中空棋子的地方进行打分，存放在两个矩阵里面，一个为我方一个为敌方。找出我方形势的分数最大值 mymaxscore 及其对应的位置还有敌方形势的最大值 hismaxscore，然后判断是进攻还是防守。取出以空位置为中心的四个方向，上、下、左、右都判断一下其棋型，还有对角线的方向，综合这些情况，对该位置进行打分。

打分规定：活五给 100000，活四或者双死四或者死四活三给 10000，双活三给 5000，四三活三给 1000，死四给 500，低级死四给 400，单活三给 100，跳活三给 90，双活二给 50，活二给 10，低级活二给 9，死三给 5，死二给 2。为了能够更好的判断形势，将各种情况的分数差拉大。

但是，如果把这个方法应用到五子棋上，经过实验发现，这个方法不能保证必赢，下棋的时候只能考虑当前的情形，不会有长远的考量，很难评估胜率。而且，哪怕我们选择的是胜率比较大的地方落子，但是对手可以返回到胜率比较低的落子位置，这样会使胜率变小。而且当棋盘变大了以后反应时间变长。

后来，我们决定采用蒙特卡洛树搜索方法。它给出了一个局面评估，虽然它不准确，但是它会比较好地自动集中到更值得搜索的变化，如果发现了一个不错的看法，蒙特卡洛树搜索会比较快地把它看到很深，它结合了广度优先搜索和深度优先搜索，类似于启发式搜索。而且，随着训练的增加，蒙特卡洛树搜索可以保证在足够长的时间之后收敛到完美解

蒙特卡洛树搜索通过迭代来一步步地扩展博弈树的规模，UCT 树是不对称生长的，其生长顺序也是不能预知的。它是根据子节点的性能指标导引扩展的方向，这一性能指标便是 UCB 值。它表示在搜索过程中既要充分利用已有的知识，给胜率高的节点更多的机会，又要考虑探索那些暂时胜率不高的节点，这种对于“利用”和“探索”进行权衡的关系便体现在 UCT 着法选择函数的定义上，即子节点 N_i 的 UCB 值按如下公式计算：

$$\frac{W_i}{N_i}+\sqrt{\frac{C\times lnN}{N_i}}$$

W_i：子节点获胜的次数；

N_i：子节点参与模拟的次数；

N：当前节点参与模拟的次数

C：加权系数，需要通过实验确定。

蒙特卡洛树搜索（MCTS）仅展开根据 UCB 公式所计算过的节点，并且会采用一种

自动的方式对性能指标好的节点进行更多的搜索。具体步骤概括如下：

1.由当前局面建立根节点，生成根节点的全部子节点，分别进行模拟对局；

2.从根节点开始，进行最佳优先搜索；

3.利用 UCB 公式计算每个子节点的 UCB 值，选择最大值的子节点；

4.若此节点不是叶节点，则以此节点作为根节点，重复 2；

5.直到遇到叶节点，如果叶节点未曾经被模拟对局过，对这个叶节点模拟对局；否则为这个叶节点随机生成子节点，并进行模拟对局；

6.将模拟对局的收益（一般胜为 1 负为 0）按对应颜色更新该节点及各级祖先节点，同时增加该节点以上所有节点的访问次数；

7.回到 2，除非此轮搜索时间结束或者达到预设循环次数；

8.从当前局面的子节点中挑选平均收益最高的给出最佳着法。

由此可见 UCT 算法就是在设定的时间内不断完成从根节点按照 UCB 的指引最终走到某一个叶节点的过程。而算法的基本流程包括了选择好的分支、在叶子节点上扩展一层、模拟对局和结果回馈这样四个部分。

UCT 树搜索还有一个显著优点就是可以随时结束搜索并返回结果，在每一时刻，对 UCT 树来说都有一个相对最优的结果。

所以，我们最终决定采用这个方法来实现。