**五子棋AI实验报告**

09016424 季宇鑫

刚看到这个题目的时候，我有点无从下手的感觉。近年来人工智能一直是计算机科学发展的一个热门方向，Alpha Go打败人类世界顶级棋手的新闻近几年也是轰动一时。现在要让一个从未接触过人工智能算法的人写一个五子棋AI，是否算一个艰巨的任务呢？在查阅网上的一些资料，了解了一些算法逻辑后，我对这个任务有了一定的理解，开始着手从0开始制作自己的AI。

1. **任务分析**

做一个五子棋AI，首先要实现五子棋的底层逻辑算法，搭建一个人机按规则对战的平台；其次，要建立一个完善的AI设计方案，有一个明确的对弈思路，包括何时进攻、何时防守；然后就是从算法性能效率以及综合胜率上对AI算法和思路进行优化改进，这一任务可作为最终的目标。

1. **设计思路**

在短短两周时间内实现一个较为完备的AI算法是很难的，但我选择从写一个能完成基本对弈的AI模型做起。

AI在得到棋盘反馈的状况后，如何对当前局势进行判断并做出对自己最有利的决策？这是我要解决的主要问题。

我想到的第一个方案是枚举出棋盘上每一个可能出现的情况，然后求出下在每个位置上我方可能获胜的概率，选择在获胜率最高的点落子。但这一方案很快被否决了，因为它对时间的开销实在太大，就15\*15的五子棋棋盘而言，如果我下了第一步棋，AI需要穷举出的棋盘上可能出现的情况数是一个天文数字，可能几十年后AI才能做出最合理的判断，该枚举算法仅适用于井字棋等棋盘规模很小的游戏。在查阅了一些有关资料后，我选择从蒙特卡洛树搜索算法中提取了一些思想，并结合查阅到的五子棋的相关知识，建立了自己AI的雏形。

蒙特卡罗树搜索是一种通过随机的对游戏进行推演来逐渐建立一棵不对称的搜索树的过程。蒙特卡罗树算法大概可以被分成四步：选择，拓展，模拟，反向传播。简单来说，就是在当前状态的基础上，选择一个备选动作，再从当前备选动作按照一定策略进行模拟，直到游戏结束为止，再统计本次备选动作产生的回报，由此来获得回报最高的备选动作并执行。

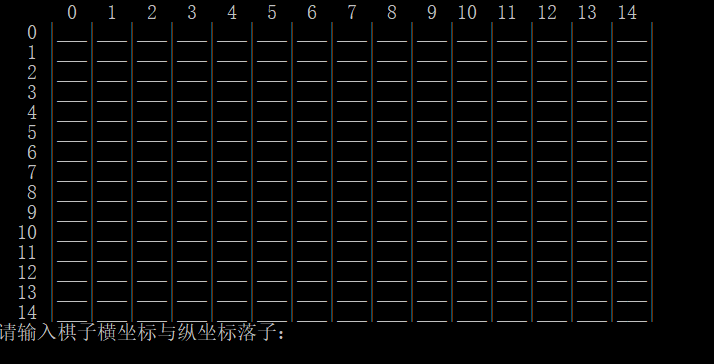
蒙特卡洛树的特点是，AI不需要知道游戏的一些定式，只需要知道怎样操作算是合法的，便可以得出理论上胜率最高的方案。还有一个比较值得学习的特点就是，算法在进行一次模拟后反馈的不是单纯的胜负结果，而是一个“回报”值。五子棋决策过程的本质，是在棋盘某一位置预落子后，对当前操作可能带来的局势改变进行评估，并确定一定的权值，通过权值比较来确定是应该进攻还是应该防守。

由此思想，可以确立一个思路：用一个15\*15的二维矩阵来表示棋盘当前每个格子上的状态，是空的还是被黑子或者白子占据；除此之外，每个格子处用两个整型数值分别代表我方以及对方在该处落子的优势权值，以判断本回合是应该防守还是应该主动进攻。以玩家先手为例，在玩家落子之后，程序对棋盘进行一次遍历，并在该处进行“预落子”，即假设此处下了黑子之后，会形成怎样的棋形，离形成五子相连的情况越相近，则赋以更高的权值，例如在此处落子后会形成四子相连的情形，其权值是大于形成三子相连情形的。从该位置四个方向进行探查后，对每个方向上会出现的棋形权值进行线性叠加，并记为防守权值（因为此时AI执白，玩家执黑）。计算完防守权值后，将之前预落子的黑子去掉，换上白子，用同样的方法计算当前可能出现的有利于白方的棋形，并用同样的方法从四个方向计算权值并进行线性叠加，并记录为进攻权值。两个权值都计算完成后，用两个临时整型变量Max\_Defend和Max\_Attack分别来储存当前最大的防守以及进攻权值，并对取最大权值的棋子位坐标进行记录。这两个最大权值不等时，可确定将棋子下在权值更大的位置，以获取对当前局势最有利的决策；当最大进攻权值与最大防守权值相等时，可设想这样一种情况，对手已经四子相连，自己也已经四子相连（先不考虑其他棋形对当前局势的影响），此时的最大进攻权值和最大防守权值是相等的，但我方只需要在四子相连处再补上一个子形成五子即可获胜结束游戏，而不是去堵对方的子。因此，应优先下在最大进攻权值处。

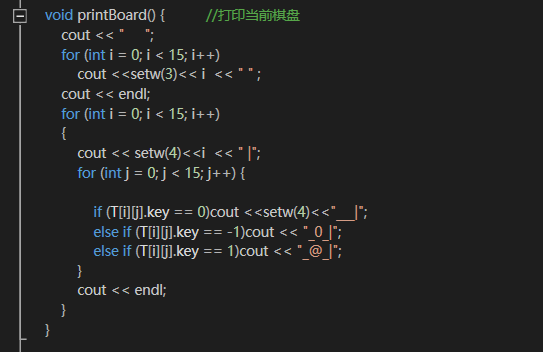
至于权值设定，我参考了网络上搜索到的资料，了解到对本算法有决定性作用的大致有六种棋形：连五、活四、冲四、活三、眠三、活二、眠二。针对每种棋形，我计划分别用几个不同的函数来对每种棋形的出现进行判断，若检测到该棋形则返回相应权值，未出现则返回0。计算总权值时，只需要将每个方向上检测到的棋形进行线性叠加即可。

1. **代码实现**

为了实现人与AI的交互以及棋盘的显示，我选用C++对话框作为交互的平台，并搭建了一个五子棋游戏的UI，如下图所示：

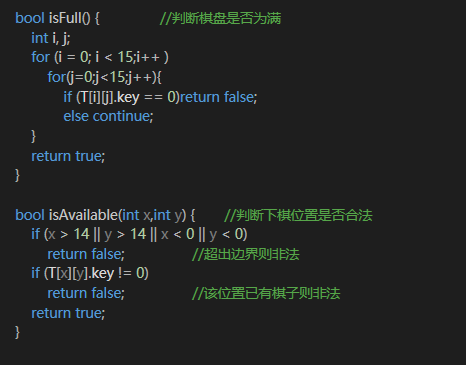


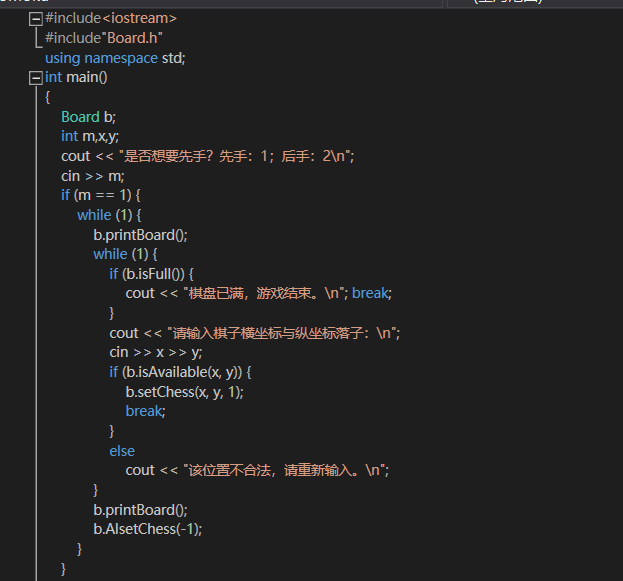
玩家可通过输入棋子的横坐标和纵坐标来在棋盘上落子，用＇＠＇代表黑子，＇０＇代表白子，此部分代码如下：

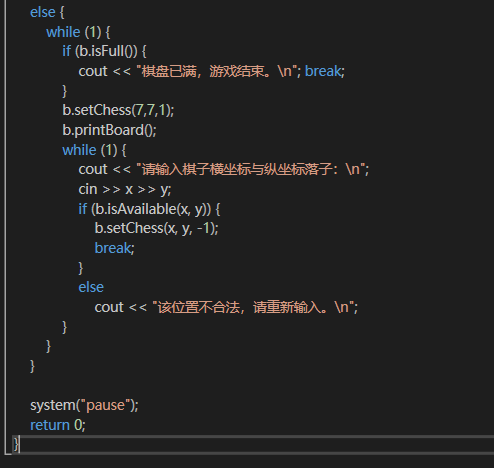


我用面向对象的方法，构造了一个Board类来形成棋盘，其中包括了一个private型chess结构体数组T[15][15]。选用结构体的原因是，棋盘位除了储存当前棋子的颜色以及是否为空状态外，还可以存储当前的进攻以及防守权值。Key表示当前棋子位的状态，1为黑子，-1为白子，0为空。

主函数中玩家可自行选择电脑先手还是玩家先手。当选择玩家后手时，AI第一步默认下在天元（7，7）处。除此之外，如果胜负未决出但棋盘已满，也要结束游戏，所以用一个判满函数isFull()来对棋盘是否还有空余位置进行判定，并用一个判合法函数isAvailable(int x,int y)对落子位置进行合法性判断。代码如下：







目前为止，基本的对弈平台已经搭建完成，接下来就是对AI算法进行完善和优化处理。

但用我之前提出的逐点从四个方向求权值的方法，运算效率特别慢，AI每做出一步决策需要对整个棋盘进行一次搜索，当棋盘上棋子数量很少时，也要对大部分空白的位置从四个方向进行探索，这一方法是非常不合算的。所以我写了一个HasNeighbor函数，用来判断棋子位周围是否存在别的棋子，距离2范围内没有棋子的话，对这个棋子位求权值是没有很大意义的，在这里可以筛选掉很多冗余的操作。

另外，由于每回合落子只改变该位置周围棋子位的权值，可由此着手，对权值进行一系列的简化运算，省略对一些不必要位置的重复计算。由于时间有限，我打算将这一项工作放在下周实行。