四子棋实验报告

计82　洪昊昀

1. 算法思路

本实验中，我的方法是“蒙特卡洛方法”和“信心上限树算法”的结合，主要过程是构建 （或小幅度修改）UCT树，随机模拟对战过程，最后计算收益，从而确定最佳落子点。

每次以当前状态作为根节点，建立 UCT 树。根的子节点即为每一列可落子的状态。

从根出发，若可以扩展出新的子节点，则扩展出新扩展的点；若当前点无法扩展出新的子节点，则选择它的得分最高的子节点。得分规则为如下UCB1函数：

$$\frac{Q(v)}{N(v)}+C\sqrt{\frac{2\ln(N(root(v)))}{N(v)}}$$

其中 v 为考察的子节点，root(v)代表它的根。N(v) 为 v 被回溯的次数，Q(v) 为 v
回溯的总得分。

然后从选中的节点v开始，随机模拟玩家落子的过程，直到得到结果。然后将收益（胜为 1，负为 -1，平为 0）回溯给祖先。在时限内不断进行该随机过程，最后选择最优的子节点作为答案。

2. 思路优化

2.1 避免频繁建树

第一次轮到我方下棋时，创建UCT树，而后面的每一步只需要利用我写的`moveRoot()`函数，将UCT树的根节点移到当前状态，这样可以避免频繁建树和删除节点，而且在移动表示根的指针前，还可以把没有用的之前的根的兄弟节点释放了，把析构的时间分摊到每一步棋，并及时回收内存。

2.2 最优子节点的选取

不同于选择UCB1值最大的子节点，我最后选择的是被回溯最多次的节点，因为教材中提到，在实际系统中具有最高收益的子节点和访问次数最多的子节点往往是同一个节点，虽然这一点并不能保证，但是如果参数C的选取不当，多次利用UCB1函数可能会导致出现较大偏差，所以我选择被回溯最多次的节点，最后也得到了较好的结果。

2.3 加入四子棋的知识

因为UCT算法可能无法总是找到最优解除了使用UCT算法，我还加入了四子棋局面判断的知识：

1. 如果我方走某步棋，可以让我方取得胜利，那么一定下这步棋；
2. 如果对方占领某个格子会让对方胜利，那么我方这步棋一定要下在这个格子中，阻止对方胜利；
3. 如果对方占领某个格子，导致对方有至少两个方向，仅差一个棋子就能胜利，那么虽然对方不会马上胜利，它下一局也一定会胜利，那么我方也一定要下在这个格子中，阻止对方胜利；
4. 如果我方占领某个格子，导致我方有至少两个方向，仅差一个棋子就能胜利，那么一定下这步棋，这样在我方的下一步中，我方一定胜利；
5. 如果我方下了一步目前看来最好的棋，但是会导致对方仅需一步棋就能获胜，那么我方绝对不能下这步棋。

这些知识都比较“短视”，不过结合蒙特卡洛模拟也能得到比较理想的结果。

从逻辑上来看，越早得到结果的模拟，它带来的信息应当越重要，所以我也尝试过记录每一次模拟得到结果需要的步数n，将它作为参数，然后用函数$1+\exp(1-n)$来代替回溯的delta=1（胜为 delta，负为 -delta，平为 0），但是效果不明显，这应该与函数的选取关系很大，这也是未来可以考虑的。还有就是delta与参数C的比值问题，我最终取delta=1，c=1.4，因为有文献“*Improved Monte-Carlo Search*”中提到$\sqrt{2}$的比值会让蒙特卡洛模拟性能更好，但是因为测试的批量有限，所以我也无法证明这一点，不过这也是未来可以考虑的。

3. 测试结果

批量测试 #1769　总胜率86%

AI标号	胜	负	平
44，56，60，80，86，96	1	1	0
84，94，98，100	0	2	0
其它偶数	2	0	0