四子棋实验报告

1. 算法策略

为了解决这个问题，我采用了 UCT 算法，程序伪代码如下：

算法 3：信心上限树算法（UCT）

function UCTSEARCH(s_0)
以状态s_0创建根节点v_0;
while 尚未用完计算时长 **do**:
v_l ←TREEPOLICY(v_0);
Δ←DEFAULTPOLICY(s(v_l));
BACKUP(v_l, Δ);
end while
return a(BESTCHILD(v_0, 0));

function TREEPOLICY(v)
while 节点v不是终止节点 **do**:
if 节点v是可扩展的 **then**:
return EXPAND(v)
else:
$v \leftarrow$ BESTCHILD(v, c)
return v

function EXPAND(v)
选择行动$a \in A(state(v))$中尚未选择过的行动
向节点v添加子节点v'，使得$s(v')= f(s(v), a), a(v') = a$
return v'

function BESTCHILD(v, c)
return $argmax_{v' \in children\ of\ v}\left(\frac{Q(v')}{N(v')} + c\sqrt{\frac{2ln(N(v))}{N(v')}}\right)$

function DEFAULTPOLICY(s)
while s不是终止状态 **do**:
以等概率选择行动$a \in A(s)$
$s \leftarrow f(s, a)$
return 状态s的收益

function BACKUP(v, Δ)
while $v \neq NULL$ **do**:
$N(v) \leftarrow N(v) + 1$
$Q(v) \leftarrow Q(v) + \Delta$
Δ← −Δ
$v \leftarrow v$的父节点

首先，我定义了一个 Node 类，用作信心上限树中的节点，在一个节点内储存当前的棋盘信息，前一轮对手的落子信息，访问次数，收益，以及子节点的指针。在 UCT 类中，我按照以上方法实现了各个函数，其中对于收益我做了额外处理：节点中的收益值都为正，在计算信心上界时再乘以轮次（即对方取负，我方取正），同时取 c=0.5，计算得到结果。在搜索时，我限制搜素轮数不超过 300000，以避免超时。

2. 效果

通过 compete 与 100.dll 对战时，单次决策最大时间约为 2.3 秒，胜率约为 60%。