作业二： 重力四子棋

计76 沈诣博 2017011427

1 算法思路：

在该实验中我采用的是“蒙特卡洛方法”和“信心上限树算法”；通过构建 UCT树、随机模拟对战过程并且计算收益，从而确定最佳落子点并落子。

在算法的具体执行步骤中，每次我以函数传入的当前状态作为根节点，建立 UCT 树。每一个状态的子节点为被搜索点在该状态上合法落子产生的其它状态。

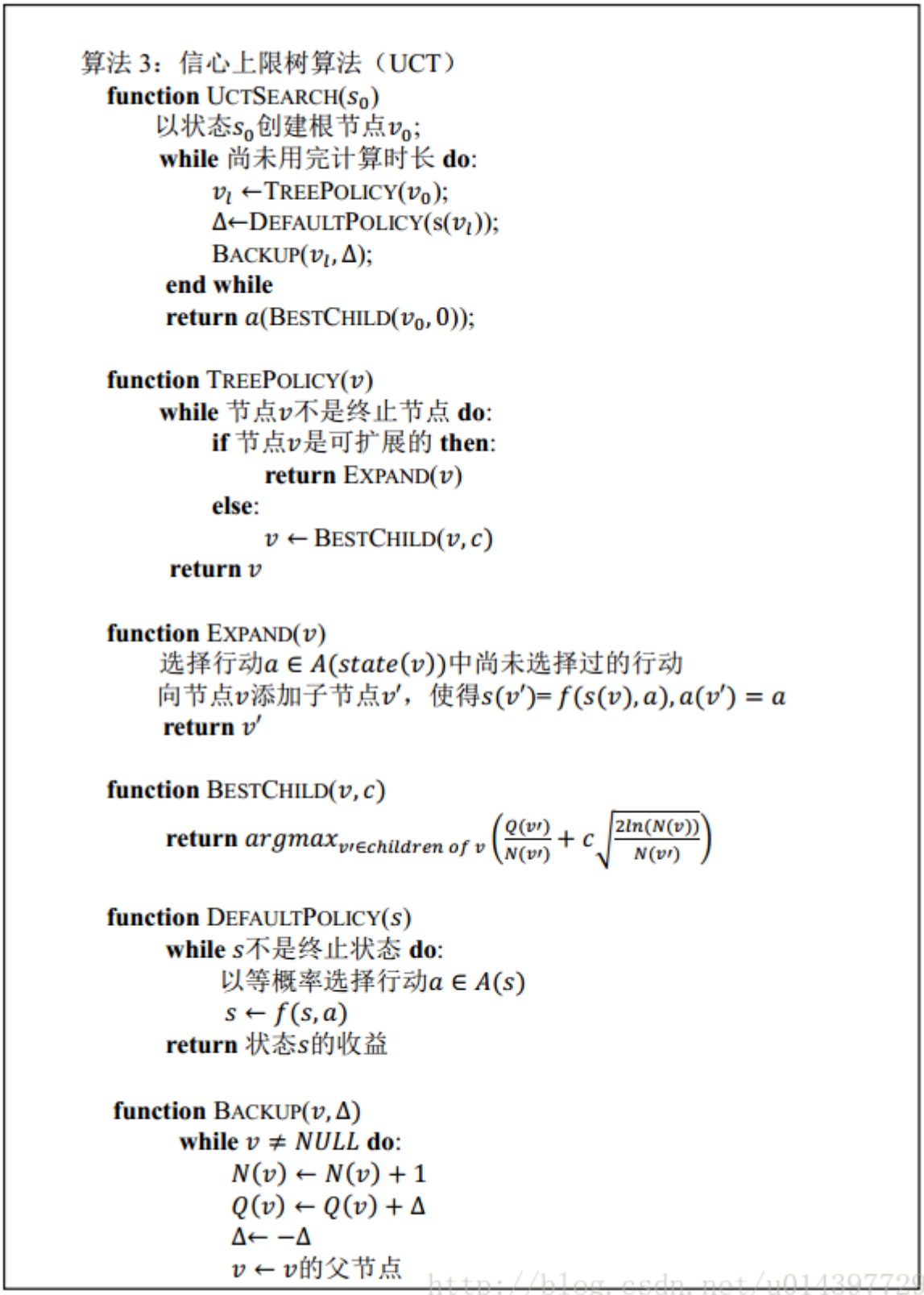
首先进行状态选中：每一轮从根出发，若可以扩展新的子节点，则扩展并选中新扩展的点；若当前点无法扩展新的子节点，则下一轮搜索当前点的得分最高的叶子节点，直到被搜索点可以分出胜负或平局。然后若选中状态并未停止，则再其的基础上进行随机落子直到分出胜负或平局。

得分规则如下：

$$Score = \frac{S(v)}{N(v)} + \sqrt{\frac{2ln(N(u))}{N(v)}}$$

其中u为当前节点，v为叶子节点；*S(v)*为叶子节点方胜利次数和失败次数之间的差值。*N(u),N(v)*分别为自己和叶子节点被访问的次数。

多次进行上述的模拟之后，选中根节点的得分最高子节点作为落子点。



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**:
$v_l \leftarrow$TREEPOLICY(v_0);
$\Delta \leftarrow$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**:
$N(v) \leftarrow N(v) + 1$
$Q(v) \leftarrow Q(v) + \Delta$
$\Delta \leftarrow -\Delta$
$v \leftarrow v$的父节点

图为算法的伪代码，引用自https://blog.csdn.net/u014397729。

2 具体实现

除了原有的 Strategy 类外,我新增了两个类 Status 和 Uct。Status 类为节点类,存储棋局信息;Uct 类为信心上限树算法类。

Status 类的成员如下：

```
char chessboard[MAXI*MAXI]; //存储棋盘
char topNode[MAXI];         //存储每列最高点
char expandableNode[MAXI];  //存储每个状态可扩展的点
char expandableNodeSize;    //可扩展的点个数
char _preX,_preY;           //父状态落子点
int visitedNum;             //状态被访问的次数
double profit;              //收益
bool player;                //状态对应执子玩家
bool stop;                  //状态是否平局或胜负
Status *father;             //父状态的指针
Status *descendents[MAXI];  //子状态的指针数组
```

函数如下：

```
bool Stops()                        //返回状态是否平局或胜负
Status* BestChild()                 //返回最佳子状态
void init ( char, char , Status *)  //根据父状态以及其落子点初始化一个子节点
Status* Expand()
```

UCT类的成员和成员函数如下：

```
Status * _root;                 //根节点
Status * TreePolicy(Status* )   //确定选中的开始搜索的状态
void DefaultPolicy(Status* )    //从搜索状态开始随机模拟
Point* UCTSearch()              //返回落子节点
```

另外，有两个非成员函数：

```
void backup( Status*, double )  //更新搜索路径上的收益和访问次数
bool End()                      //判断随机落子是否平局或分出胜负
```

3 实验结果

这里只列出与 96、98、100 这 3个 AI 的对战结果，CPU为i5-8500 3GHz单核，和评测机较为类似

AI名称	我的胜率
96.dll	100%
98.dll	100%
100.dll	90%

4 总结

本次完成大作业的过程中，学习到了蒙特卡洛模拟方法、信心上限树算法，并将其应用到对战策略中，取得了较为理想的测试结果。同时，本人还在常数优化上做了一定的工作，使得自己的算法有更高的效率。

感谢马老师和助教的悉心指导！