

南 京 理 工 大 学

智能计算技术实验二

姓 名：蒋旭钊 学 号：918106840727

学院(系): 计算机科学与工程学院

专 业：计算机科学与技术

课 程：智能计算技术

2021年 11月

一. 问题重述

实现章节《5.6 博弈树的启发式搜索》中，5*5 格子的一字棋问题，要求 MAX 方和 MIN 方都用博弈树来决策，运用极大极小分析法，同时加入 α - β 剪枝策略。

二. 算法介绍

极大极小分析法：

- (1) 设博弈的双方中一方为 A，另一方为 B。极大极小分析法是为其中的一方（例如 A 方）寻找一个最优行动方案的方法。
- (2) 为了找到当前的最优行动方案，需要对各个方案可能产生的后果进行比较。
- (3) 为了计算得分，需要根据问题的特性信息定义一个估价函数，用来估算当前博弈树端节点的得分，称为静态估值。
- (4) 当端节点的估值计算出来后，再推算出父节点的得分。
- (5) 如果一个行动方案能获得较大的倒推值，则它就是当前最好的行动方案。

博弈树的启发式搜索算法：

- (1) $k=1$ ，初始棋局 $S_k = S_1$ ；
- (2) 如果棋局 S_k 是终止节点棋局，则算法成功终止；否则，由棋局 S_k 生成 A 方所有可能的或关系子节点 S_i ($i=1, 2, \dots, n$)。
- (3) 对每一个或关系子节点 S_i ，生成其 B 方所有可能的与关系子节点 S_j ($j=1, 2, \dots, m$)。生成节点数为 $n \times m + 1$ 的部分博弈树。
- (4) 计算每个与关系子节点 S_j 的启发函数值。
- (5) 分别由 m 个与关系子节点倒推计算其父节点（或节点）的启发函数值：

$$h_i = \min\{ h_j \mid j = 1, 2, \dots, m \}, \quad i = 1, 2, \dots, n$$

- (6) 由 n 个或关系子节点倒推计算其父节点 S_k （或节点）的启发函数值：

$$h_k = \max\{ h_i \mid i = 1, 2, \dots, n \}, \quad k = 1, 2, \dots, n$$

- (7) A 方从 S_k 的 n 个或关系子节点中选择节点 i 作为最优行动方案，获得棋局 S_i 。

- (8) B 方从 S_i 的 m 个与关系子节点中选择节点 j 作为最优行动方案，获得棋局 S_j 。

- (9) 若节点 j 是端节点，则算法终止；否则令 $k=j$ ，转步骤 (2)。

a- β 剪枝策略:

(1) 对于一个与节点 MIN, 若能估计出其倒推值的上界 β , 并且这个 β 值不大于 MIN 的父节点 (一定是或节点) 的估计倒推值的下界 α , 即 $\alpha \geq \beta$, 则就不必再扩展该 MIN 节点的其余子节点了。这一过程称为 α 剪枝。

(2) 对于一个或节点 MAX, 若能估计出其倒推值的下界 α , 并且这个 α 值不小于 MAX 的父节点 (一定是与节点) 的估计倒推值的上界 β , 即 $\alpha \geq \beta$, 则不必再扩展该 MAX 节点的其余子节点了。这一过程称为 β 剪枝。

三. 实现思路

主要有三个 java 类来实现:

- Class AIGobang

定义了棋盘大小 BOARD_SIZE, 以及无穷大 MAX_VAL、无穷小 MIN_VAL、平局 DRAW。

此外实现了 Human 和 AI 双方下棋位置的展示以及每个棋盘的显示, 最终给出双方的终局结果。

- Class BoardState

定义了博弈树中每个节点的状态, 包括生成该博弈树的角色 mainPlayerTurn、要落子的角色 playerTurn、结点的 alpha、beta 值, 以及博弈树搜索最大深度 MAX_DEPTH(=2)。

棋盘上 alpha、beta 值以及状态估值的更新, 通过 getScore() 函数递归更新, 剪枝通过:

```
if (this.alpha>=this.beta){  
    break;  
}进行判断。
```

- Class Utils

定义了要用到的各种函数, 包括根据棋盘落子算出静态估值的函数 getCost、生成新棋盘的函数 generateNewBoard() 等等。

四. 实验结果

本次实验不需要输入, 输出展示了下棋的棋盘状态以及走子情况, 最后判断双方的终局结果。

输出样例:

```
**Human First
**Human is X
**AI is O
**Chess is as below.....
Chessboard No.0:(The empty chessboard)
```


-----Let's start!-----

```
Next Human Position is (2,2)
Chessboard No.1:
```

			O		

```
Next AI Position is (0,0)
Chessboard No.2:
```

--	--	--	--	--	--

Next AI Position is (4,2)

Chessboard No.24:

	X		O		X		O		O	
--	---	--	---	--	---	--	---	--	---	--

	O		X		O		X		X	
--	---	--	---	--	---	--	---	--	---	--

	O		X		O		X		O	
--	---	--	---	--	---	--	---	--	---	--

	X		O		X		O		X	
--	---	--	---	--	---	--	---	--	---	--

	O		O		X				X	
--	---	--	---	--	---	--	--	--	---	--

Next Human Position is (4,3)

Chessboard No.25:

	X		O		X		O		O	
--	---	--	---	--	---	--	---	--	---	--

	O		X		O		X		X	
--	---	--	---	--	---	--	---	--	---	--

	O		X		O		X		O	
--	---	--	---	--	---	--	---	--	---	--

	X		O		X		O		X	
--	---	--	---	--	---	--	---	--	---	--

	O		O		X		O		X	
--	---	--	---	--	---	--	---	--	---	--

The Game Is Draw!

Process finished with exit code 0