人工智能第四次实验

（用极大极小值算法实现井字棋游戏）

极大极小值算法  
x极大极小值算法又名MiMaX算法,是一种找出失败的最大可能性中的最小值算法。算法常用于棋类等由两方较量的游戏和程序,这类程序由两个游戏者轮流,每次执行一个步骤。我们众所周知的五子棋、象棋等都属于这类程序,所以Minimax算法是基于搜索的博弈算法的基础。该算法是一种零总和算法,即一要在可选的选项中选择将其优势最大化的选择,而另一方则选择令对手优势最小化的方法。

极大极小值算法κ我们知道,常用的博弈算法都是基于搜索的博弈算法,所有可能的下棋步骤构成一个树的结构,然后对局面进行价值评估。极大极小值算法是这样做的(假设现在要为A选择最佳走法):如果当前局面是A的局面那么就选择最大值vaue=Avaue-Bvaue,如果当前是B走后形成的局面,那么就选择

的最小值,也就是最小化A的利盖,其实就是最大化B的利益

井字棋”游戏的规则与“五子棋”十分类似，“五子棋”的规则是一方首先五子连成一线就胜利；“井字棋”是一方首先三子连成一线就胜利。 游戏时一方是电脑，另一方是玩家。

程序：#include "stdio.h"

#include "malloc.h"

#define SIZE 3

#ifndef FALSE

#define FALSE 0

#endif

#ifndef TRUE

#define TRUE 1

#endif

#define NONE 0

#define PLAYER\_A 1

#define PLAYER\_B 2

#define WARNNING 255

#define COMPETITOR 200

#define WINNER -1

char chessboard[SIZE][SIZE];

struct CHESS\_MAN

{

int row;

int col;

};

/\*PLAYER可能胜利的方法\*/

int get\_value(int player)

{

int i,j,ret=0;

int row,col,inc;

int bNONE=FALSE;

/\*检查所有行\*/

for(i=0;i<SIZE;i++)

{

row=SIZE;

bNONE=FALSE;

for(j=0;j<SIZE;j++)

{

if(chessboard[i][j]==player) row--; /\*如果该位置有player的棋子占据\*/

if(chessboard[i][j]==NONE) bNONE=TRUE; /\*如果该位置还没有棋子占

据，则返回bNONE为TRUE\*/

}

if(row==1&&bNONE==TRUE) return WARNNING; /\*电脑:该行中有一个

空位且有对手下的2个旗子,则可能会输掉该局，返回WARNNING值）\*/

else if(row==SIZE) ret++; /\*电脑:该行中没有player的棋子，

ret+1\*/

}

/\*检查所有列\*/

for(i=0;i<SIZE;i++)

{

col=SIZE;

bNONE=FALSE;

for(j=0;j<SIZE;j++)

{

if(chessboard[j][i]==player) col--; /\*如果该位置有player的棋子占据\*/

if(chessboard[j][i]==NONE) bNONE=TRUE; /\*如果该位置还没有棋子占

据，则返回bNONE为TRUE\*/

}

if(col==1&&bNONE==TRUE) return WARNNING; /\*电脑:该列中有一个

空位且有对手下的2个旗子,则可能会输掉该局，返回WARNNING值\*/

else if(col==SIZE) ret++; /\*电脑:该列中没有player的棋子，

ret+1\*/

}

/\*检查左对角线\*/

inc=SIZE;

bNONE=FALSE;

for(i=0,j=0;i<SIZE;i++,j++)

{

if(chessboard[i][j]==player) inc--; /\*如果该位置有player的棋子占据\*/

if(chessboard[i][j]==NONE) bNONE=TRUE; /\*如果该位置还没有棋子占

据，则返回bNONE为TRUE\*/

}

if(inc==1&&bNONE==TRUE) return WARNNING; /\*电脑:左对角线中有

一个空位且有对手下的2个旗子,可能会输掉该局，返回WARNNING值\*/

else if(inc==SIZE) ret++; /\*电脑:左对角线中没有player的棋子，

ret+1\*/

/\*检查右对角线\*/

inc=SIZE;

bNONE=FALSE;

for(i=0,j=SIZE-1;i<SIZE;i++,j--)

{

if(chessboard[i][j]==player) inc--; /\*如果该位置有player的棋子占据\*/

if(chessboard[i][j]==NONE) bNONE=TRUE; /\*如果该位置还没有棋子占

据，则返回bNONE为TRUE\*/

}

if(inc==1&&bNONE==TRUE) return WARNNING; /\*电脑:右对角线中有

一个空位且有对手下的2个旗子,可能会输掉该局，返回WARNNING值\*/

else if(inc==SIZE) ret++; /\*电脑:右对角线中没有player的棋子，

ret+1\*/

return ret;

};

/\*显示棋盘图形边框\*/

void disp\_chess\_board(void)

{

int i,j;

/\*显示棋盘最顶层边框\*/

for(i=0;i<SIZE\*4+1;i++)

printf("-");

printf("\n");

/\*显示3层棋盘格落子情况及其左、右和下边框\*/

for(i=0;i<SIZE;i++)

{

printf("|");

for(j=0;j<SIZE;j++)

{

if(chessboard[i][j]==PLAYER\_A) printf(" o |"); /\*如果是PLAYER\_A落

子则用o表示棋子\*/

else if(chessboard[i][j]==PLAYER\_B) printf(" x |"); /\*如果是PLAYER\_B

落子则用x表示棋子\*/

else printf(" |");

}

printf("\n");

/\*输出该层下边框\*/

for(j=0;j<SIZE\*4+1;j++)

printf("-");

printf("\n");

}

return;

};

/\*棋盘初始状况\*/

void init\_chess\_board(void)

{

int i,j;

for(i=0;i<SIZE;i++)

for(j=0;j<SIZE;j++)

chessboard[i][j]=NONE;

return;

};

int enter\_chess\_man(int row, int col, int player)

{

if(row>=SIZE||col>=SIZE) return FALSE; /\*输入位置超出棋盘坐标，不能

落子\*/

if(chessboard[row][col]!=NONE) return FALSE; /\*输入当该位置已有棋子占

据，不能落子\*/

chessboard[row][col]=player; /\*玩家落子\*/

return TRUE;

};

/\*判断胜利状况\*/

int chk\_winner(int player)

{

int i,j;

int col,row,inc;

/\*占满一行\*/

for(i=0;i<SIZE;i++)

{

row=TRUE;

for(j=0;j<SIZE;j++)

{

if(chessboard[i][j]!=player) row=FALSE;

}

if(row==TRUE) return TRUE;

}

/\*占满一列\*/

for(i=0;i<SIZE;i++)

{

col=FALSE;

for(j=0;j<SIZE;j++)

{

if(chessboard[j][i]!=player) col=FALSE;

}

if(col==TRUE) return TRUE;

}

/\*占满左对角线\*/

inc=TRUE;

j=0;

for(i=0;i<SIZE;i++)

{

if(chessboard[i][i+j]!=player) inc=FALSE;

}

if(inc==TRUE) return TRUE;

/\*占满右对角线\*/

inc=TRUE;

j=SIZE-1;

for(i=0;i<SIZE;i++)

{

if(chessboard[i][j-i]!=player) inc=FALSE;

}

if(inc==TRUE) return TRUE;

/\*还没有一方胜利\*/

return FALSE;

};

/\*最佳的一步棋\*/

int get\_best\_chess(struct CHESS\_MAN \*best\_chess, int player, int other)

{

int chess\_value[9];

struct CHESS\_MAN chess[9];

int i,j,cur=0;

for(i=0;i<SIZE;i++)

{

for(j=0;j<SIZE;j++)

{

chess[cur].row=i;

chess[cur++].col=j;

}

}

for(i=0;i<9;i++)

{

if(enter\_chess\_man(chess[i].row,chess[i].col,player)==TRUE)

{

chess\_value[i]=get\_value(other); /\*\*/

if(chk\_winner(player)==TRUE) chess\_value[i]=WINNER; /\*玩家未胜利，

则chess\_value[i]为WINNER\*/

chessboard[chess[i].row][chess[i].col]=NONE; /\*玩家落子位置错误，

棋盘为0\*/

}

else chess\_value[i]=COMPETITOR; /\*

玩家落子位置正确，\*/

}

/\*选择值为最低的chess\_value\*/

cur=0;

for(i=0;i<9;i++)

{

if(chess\_value[cur]>chess\_value[i]) cur=i;

}

/\*\*/

best\_chess->row=chess[cur].row;

best\_chess->col=chess[cur].col;

return chess\_value[cur];

};

/\*检查是否还有未落子的棋格\*/

int chk\_full(void)

{

int i,j;

for(i=0;i<SIZE;i++)

for(j=0;j<SIZE;j++)

{

if(chessboard[i][j]==NONE) return FALSE;

}

return TRUE;

};

int main(void)

{

int i;

struct CHESS\_MAN best\_chess;

int player=PLAYER\_A; /\*玩家先手\*/

int competitor=PLAYER\_B; /\*电脑后手\*/

int bEND=FALSE; /\*初始bEND的值\*/

int row,col; /\*玩家输入所下棋子的位置\*/

init\_chess\_board(); /\*初始棋盘数据\*/

disp\_chess\_board(); /\*绘制棋盘边框\*/

while(bEND==FALSE) /\*若bEND为FALSE，则判定棋局结束\*/

{

if(player==PLAYER\_A)

{

/\*轮到玩家下棋时，显示玩家坐标输入提示\*/

do

{

printf("] 请输入您落子的位置 : \n");

printf("] 行坐标为: ");

scanf("%d",&row);

printf("] 列坐标为: ");

scanf("%d",&col);

if(enter\_chess\_man(row-1,col-1,player)==TRUE) /\*玩家落子正确，棋

盘坐标显示，并结束该循环\*/

{

printf("] 您落子的位置是 [%d][%d]\n",row,col);

break;

}

else printf("] 您输入的棋盘坐标错误，请重新输入\n"); /\*玩家落子坐标

错误提示\*/

}while(TRUE);

}

else

{

/\*电脑选择最佳位置下棋并显示落子的棋盘坐标提示\*/

get\_best\_chess(&best\_chess,player,competitor);

enter\_chess\_man(best\_chess.row,best\_chess.col,player);

printf("] 电脑落子的位置是[%d][%d]\n",best\_chess.row+1,best\_chess.col+1);

}

/\*显示当前棋盘落子状况\*/

disp\_chess\_board();

/\*判断胜负后，显示该棋局的胜利者提示\*/

bEND=TRUE;

if(chk\_winner(player)) printf("] 胜利者是Player %d.\n",player);

else if(chk\_winner(competitor)) printf("] 胜利者是Player %d.\n",competitor);

else if(chk\_full()) printf("] 平局.\n");

else bEND=FALSE;

competitor=player;

if(player==PLAYER\_A) player=PLAYER\_B;

else player=PLAYER\_A; };

printf("\n\n本棋局结束.\n\n"); /\*该局已结束\*/

return 0;

};

运行截图：

