

河海大学物联网工程学院

《人工智能》 课程设计报告

基于搜索求解的五子棋人机博弈

学年学期： 2021-2022 学年第二学期

授课班号： c0601036

专业年级： 计算机科学与技术专业 2019 级

指导教师： 万 建 武

报告撰写： 秦 骁

一、程序介绍

1. 课题介绍

采用搜索求解策略，编写一个五子棋程序。包含两个角色：玩家和电脑。玩家可自由下棋；程序读取棋盘状态，并自动计算出下一步棋子的位置。当五子连一线时，游戏结束。

该人机博弈程序满足以下条件：

- (1) 具有用户友好的可视化界面；
- (2) 用户落子交互采用鼠标点击，无需键盘输入落子位置；
- (3) 响应时间在可接受范围内；
- (4) 人机难度可设定；
- (5) 无禁手；
- (6) 玩家执黑先手，电脑可决定三手交换。
- (7) 可设置人机难度

2. 系统模块划分及功能介绍

可视化模块：

- (1) `void draw()`——绘制棋盘横纵坐标“1~19”和“A~S”，遍历 19×19 的棋盘绘制黑白子，同时标记最新的落子位置提醒玩家；
- (2) `void init()`——棋盘初始化，根据棋盘 19×19 的格子内部边缘、四角、内部绘制不同样式的网格线，按不同的棋盘内样式（棋盘上下边框、左右边框、四角边框、内部格线）绘制棋盘；

求解最优落子位置算法模块：

- (3) `location findbestlocation(int color, int c)`——程序的核心算法函数，在 `game()` 中被调用，利用估价函数，递归寻找最有利于己方、最不利于对方的落子位置，返回值为 `location` 对象；

辅助模块：

- (4) `void isWIN()`——根据相同颜色棋子连续长度，判断胜负，棋盘已满且未分胜负判定为平局，改变 `win` 值决定胜负结果；
- (5) `void game()`——五子棋游戏主函数，对各函数进行调用，每轮未落子前，利用 `GetMouseMsg()` 捕捉鼠标位置，落子后，调用 `isWIN()` 判断胜负，未分胜负时，利用 `goto LOOP` 交替循环双方下棋回合，分出胜负或者棋盘下满时，跳出循环，输出胜负情况；
- (6) `main(void)`——主函数，利用 `graphic` 库创建绘图环境，调用 `init()` 和 `game()` 初始化游戏界面；

3. 重要数据结构的设计与定义

定义数组 score，作为若干种落子情况的估价函数：

```
int score[3][5] = //评分表
{
    { 0, 90, 250, 500, 500 }, // 0个敌方棋子 0 1 2 3 4个己方棋子
    { 0, 0, 80, 300, 450 }, // 1个敌方棋子 0 1 2 3 4个己方棋子
    { 0, 0, 0, 70, 400 } // 2个敌方棋子 0 1 2 3 4个己方棋子
};
```

定义棋子类，成员变量包含落子的横纵坐标和该位置评价函数值：

```
// 落子位置和分数
class location
{
public:
    int i = 0; // y 坐标
    int j = 0; // x 坐标
    int number = 0; // 分数
};
```

定义棋盘格子类，提供draw方法绘制棋盘不同格子，成员变量包含坐标信息、

该位置落子信息、拟选中信息：

```
// 棋盘每个格子
class box
{
public:
    void draw(); // 绘制
public:
    int x = 0; // x 坐标
    int y = 0; // y 坐标
    int value = -1; // 值（黑棋：1，白棋：0，空位：-1）
    int background = 0; // 模式
    bool isnew = false; // 是否被选中
    COLORREF color = WHITE; // 棋盘背景色
};
```

4. 核心算法解释

(1) 系统总流程图 (Fig 1):

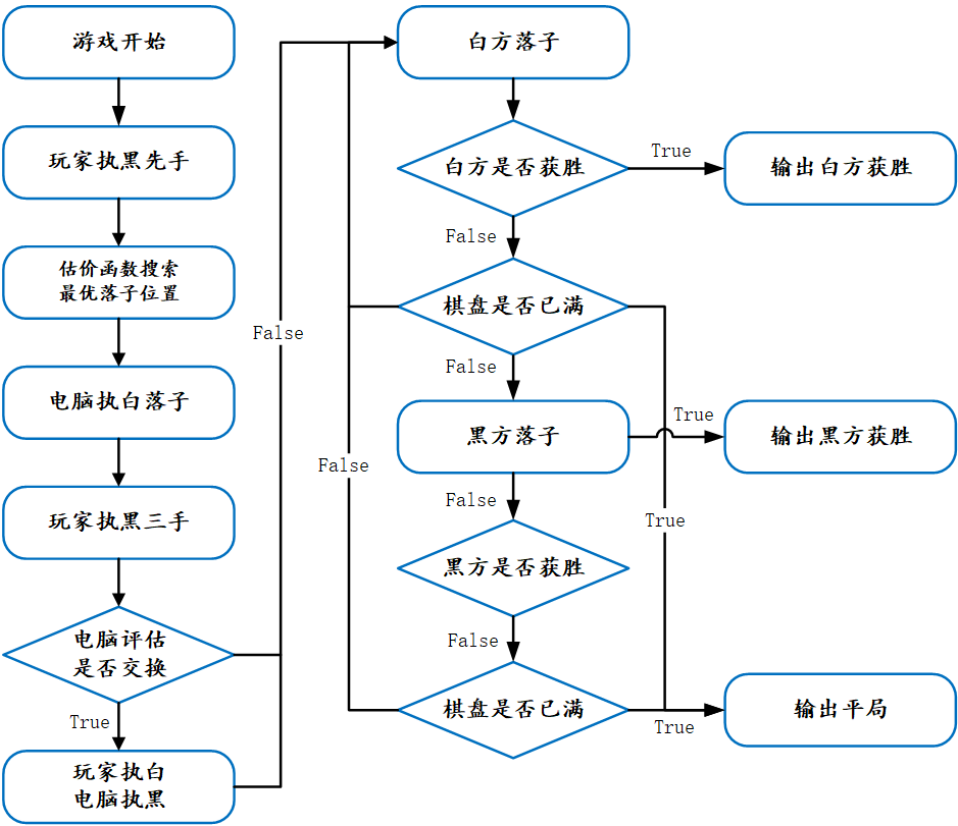


Fig 1

(2) 估价函数:

根据先验知识，可以为落子情形设置得分，越容易获胜，设置得分值越高，对于必胜的落子情况（例如活四、充五），设置绝对分值 500 作为必然落子位置。

估价函数值设置示意图 (Fig 2):










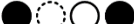


己方棋子数 敌方棋子数		1	2	3	4
1	 90	 250	 500	 500	
2	 0	 80	 300	 450	
3	 0	 0	 70	 200	

Fig 2

考虑到五子棋是零和游戏，是博弈决策的过程，所以应当有如下考虑：在做出最有利于自己决策的时候，要尽可能使得自己的决策让对方的最优落子情况最差。于是，估价函数设置为：

$$\text{nowscore}(i) = \text{this_score}(i) - \text{MAXscore}(i+1)$$

$$\text{MAXscore}(i+1) = \max(\text{nowscore}(i+1))$$

其中 $\text{this_score}(i)$ 是己方可选的最优落子位置得分， $\text{MAXscore}(i+1)$ 是第 $i+1$ 层敌方最优的落子位置得分， $\text{nowscore}(i)$ 是总的评价函数，示意图(Fig 5)如下：

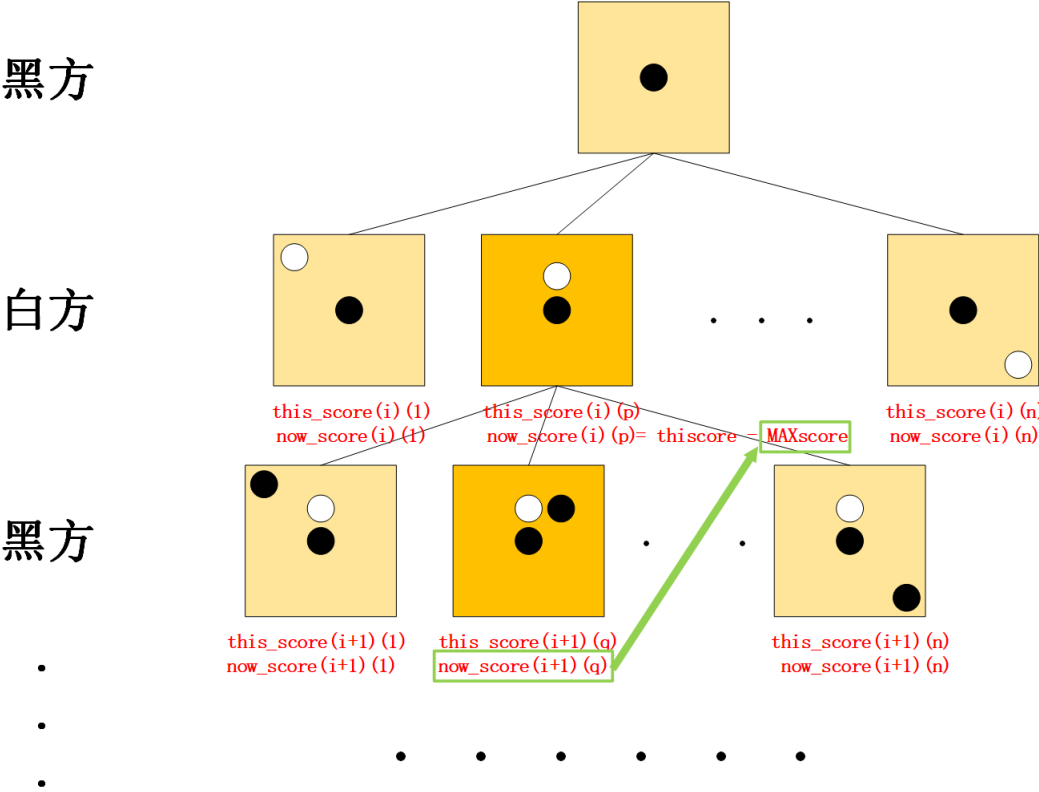


Fig 3

同时，对方也将如此考虑，使得己方的最优落子情况最坏，这是交替双方立场递归求解的过程，因而我们需要考虑未来若干步的可能性，估价函数需要设置为递归寻找最利于己方、最不利于对方的落子位置得分，定义为如下函数：

```
findbestlocation(mycolor, c){
    int nowscore = thescore - findbestlocation(enemycolor, c + 1).number
}
```

其中 c 为递归所在层数，层数越深，搜索到的最佳落子位置越好，但是搜索的时间代价指数增长。

为了综合考虑响应时间及电脑落子智能程度，我对递归层数进行试验，结果表明，递归层数超过 4 层，时间太长；少于 4 层，决策太差，因此该系统的递归层数设置为 4 层。

(3) 三手交换

考虑到五子棋存在必胜局开局（即先手无限制必胜，如寒星溪月等开局），五子棋比赛有三手交换（指对局双方开始对局后，甲方在棋盘的交叉点上落下 3 颗棋子（即前 3 手棋），乙方决定本局甲乙双方的执黑方与执白方，从第 4 手开始双方轮流落一子，直到对局结束）及禁手（指对局中禁止先行一方使用的战术，具体包括黑方一子落下时同时形成双活三、双四或长连等三种棋形。禁手只对黑方有效，白方无禁手。黑方禁手的位置称为禁手点）规则。

此项目暂只考虑三手交换，禁手受时间限制可后期再改进。设置落子计数器，在第二颗黑棋落子后，如果估价函数得分过高，则电脑决定交换，玩家执白子，见下图（Fig 4）。

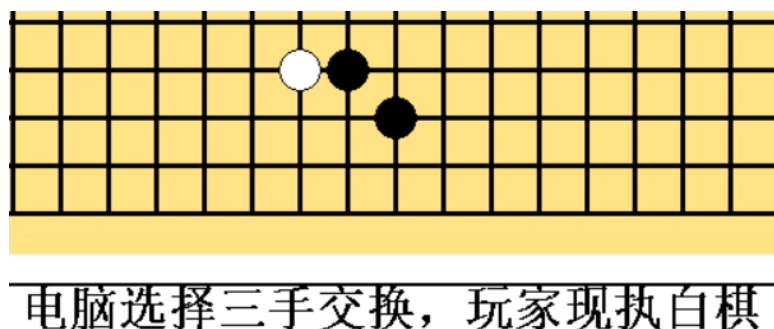


Fig 4

(4) 人机难度设置

通过设置递归层数的代价系数，决定人机难度。将递归的估价函数设置为：

$$\text{nowscore} = \text{thescore} - \text{findbestlocation}(!\text{color}, c + 1). \text{number} - k * c$$

其中 k 为递归层数的代价系数。上文已讨论过，递归层数越深，电脑落子的决策越好，所以此处 k 值越大，导致深层的递归对落子决策的优化变得越微弱。极端情况下， k 值极大（如超过 500）将导致搜索策略得优化结果变得很差如下图； k 值极小时，机器的智能决策是此系统搜索策略的最佳性能。

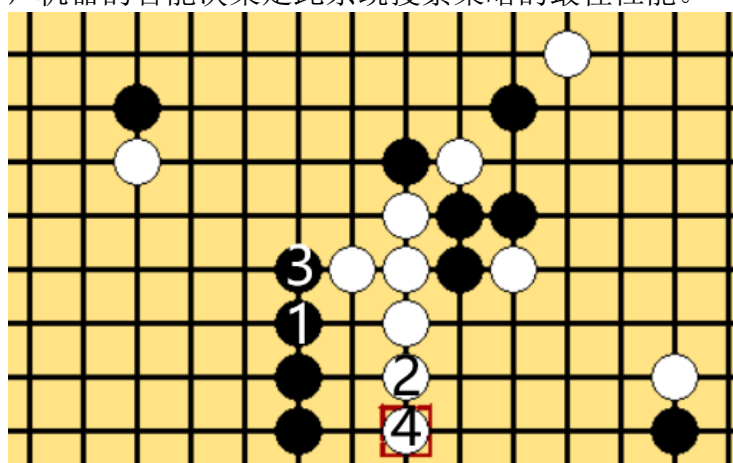


Fig 5

显然第 3 步电脑执黑应当堵白方活四，但实际决策受减分影响充己方活四。

总而言之，通过增大 k 值，可以降低游戏难度，但是会减慢系统响应时间（猜测是因为搜索到得分极大值的剪枝情况由于减分而失去作用）。

6. 程序实现（附）程序代码和相应注释说明

（代码附报告最后）

7. 附加说明

(1) 关于递归层数设置为 4 的实验结果附图及说明
设置递归为 3 层时，玩家白方不防守黑方，电脑黑方不速充 3 充 4 取胜，电脑过于愚蠢，见下图（Fig 6）

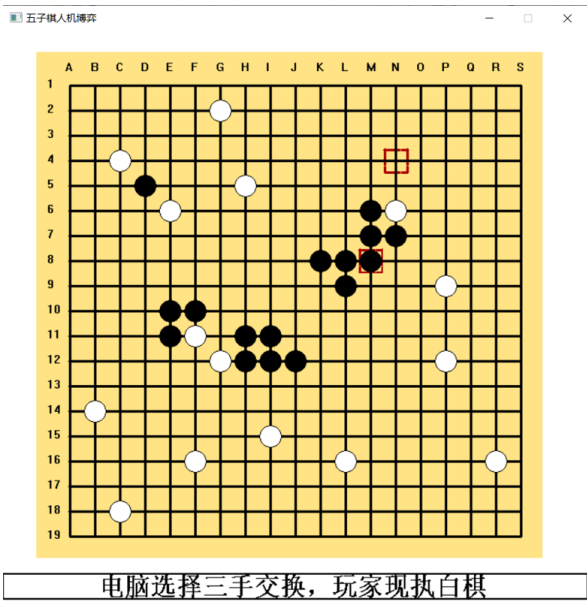


Fig 6

(2) 电脑决定三手交换，玩家白胜/电脑黑胜，见下图（Fig 7）:

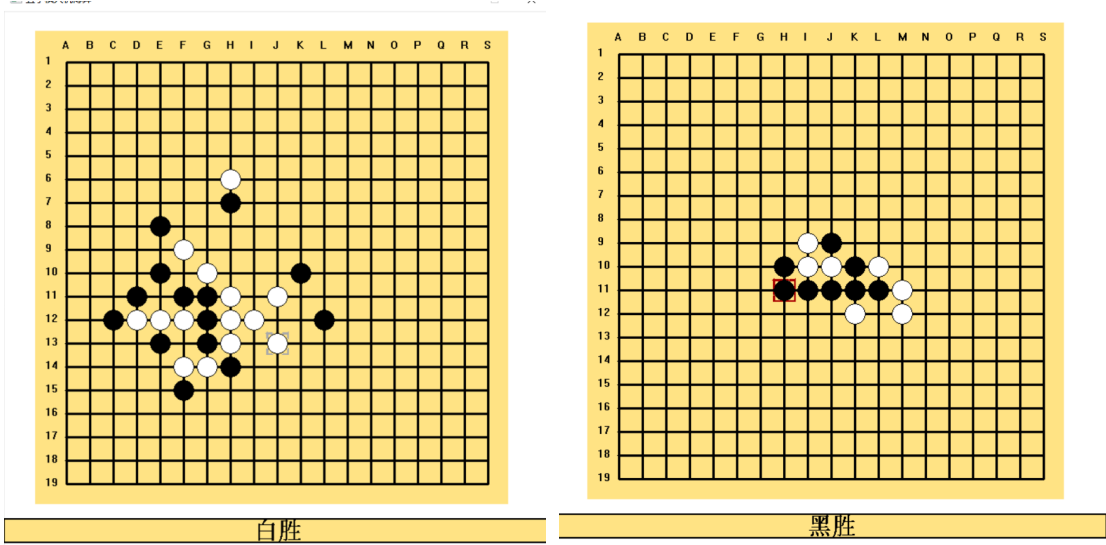


Fig 7

8. 改进方向

- (1) 必胜以及必堵落子位置可以写死, 这样在 k 值过高时, 不至于电脑太笨;
- (2) 三手交换存在 BUG, 实际情况中, 黑方边缘落子时, 白方不一定要交换, 因为边缘落子不存在先手优势;
- (3) 可以更细分活三、充四等情况, 设置更细的估价函数值;
- (4) 部分代码易读性不强, 后期可封装成更多函数;
- (5) UI 界面可更复杂点, 例如增加重新一轮游戏、难度设置等按钮;
- (6) 可能的话, 利用更高的算力, 递归更深层, 测试搜索策略的最佳性能。

二、总结

通过这次五子棋人机博弈设计, 我对《人工智能》这一课程所学内容, 尤其是搜索求解策略有了更深刻的理解。在设计程序的过程中, 将课程里的算法灵活应用于实际问题, 提高了自己的编程能力和解决问题的能力。

三、参考

1. [五子棋智能算法-博弈树算法思想详解 \(一\)_viafccy 的博客-CSDN 博客_五子棋算法](#)
(递归思路来源)
3. [五子棋人机对战完整代码_csuzhucong 的博客-CSDN 博客_五子棋代码](#)
(棋盘绘制思路、估价函数设置)

四、项目源码

```
#include <iostream>
#include <time.h>
#include <stdlib.h>
#include <graphics.h>
#include <conio.h>
using namespace std;

// 落子位置和分数
class location
{
public:
    int i = 0;        // y 坐标
    int j = 0;        // x 坐标
    int number = 0; // 分数
};

// 棋盘每个格子
class box
{
public:
    void draw();        // 绘制

public:
    int x = 0;          // x 坐标
    int y = 0;          // y 坐标
    int value = -1;      // 值（黑棋：1，白棋：0，空位：-1）
    int background = 0;  // 模式
    bool isnew = false;  // 是否被选中
    COLORREF color = WHITE; // 棋盘背景色
};

box BOX[19][19];        // 19X19 棋盘
int win = -1;           // 赢家（0：白棋，1：黑棋，2：平局）
int playcnt = 0;
bool changeflag = false;
int whosturn = 0;       // 轮到谁下棋了
int playercolor = 0;    // 玩家颜色
int dx[4]{ 1,0,1,1 }; // - | \ / 数组内的数值表示增量的四个方向（向右平移，向下平移，向
                    // 右下移，向右上移）同时取反可以扩展到 8 个方向
int dy[4]{ 0,1,1,-1 };
int score[3][5] = //评分表
{
    { 0, 90, 250, 500, 500 }, // 0 个敌方棋子 0 1 2 3 4 个己方棋子
```

```

        { 0, 0, 80, 300, 450 }, // 1 个敌方棋子 0 1 2 3 4 个己方棋子
        { 0, 0, 0, 70, 400 } // 2 个敌方棋子 0 1 2 3 4 个己方棋子
};
int bestx[361]; //最优 x 坐标
int besty[361]; //最优 y 坐标
int bestcnt = 0; //最优解数

// 函数声明
void draw(); // 绘制
void init(); // 初始化
location findbestlocation(int color, int c); // 寻找最佳位置
void isWIN(); // 判断输赢
void game(); // 游戏主函数

// main 函数
int main()
{
    initgraph(700, 700); // 初始化绘图环境
    setbkcolor(WHITE);
    cleardevice();
    setbkmode(TRANSPARENT); // 设置透明文字输出背景

    while (1)
    {
        init(); // 初始化
        game(); // 游戏开始
        cleardevice();
    }
}

// 类函数定义

// box 的 draw 方法绘制
void box::draw()
{
    COLORREF thefillcolor = getfillcolor(); // 备份填充颜色
    setlinestyle(PS_SOLID, 3); // 线样式设置
    setfillcolor(color); // 填充颜色设置
    solidrectangle(x, y, x + 30, y + 30); // 绘制无边框的正方形
    if (isnew)
    {
        // 如果是新下的
        // 绘制边框线
        setlinecolor(RED);
    }
}

```

```

line(x + 1, y + 2, x + 20, y + 2);
line(x + 2, y + 1, x + 2, y + 20);
line(x + 29, y + 2, x + 22, y + 2);
line(x + 29, y + 1, x + 29, y + 20);
line(x + 2, y + 29, x + 20, y + 29);
line(x + 2, y + 22, x + 2, y + 29);
line(x + 29, y + 29, x + 18, y + 29);
line(x + 29, y + 18, x + 29, y + 29);
}
setlinecolor(BLACK);
switch (background)
{
    // 以下是不同位置格子内部棋盘绘制的样式
case 0:
    line(x + 15, y, x + 15, y + 30);
    line(x - 1, y + 15, x + 30, y + 15);
    break;
case 1:
    line(x + 14, y + 15, x + 30, y + 15);
    setlinestyle(PS_SOLID, 3);
    line(x + 15, y, x + 15, y + 30);
    setlinestyle(PS_SOLID, 2);
    break;
case 2:
    line(x - 1, y + 15, x + 15, y + 15);
    setlinestyle(PS_SOLID, 3);
    line(x + 15, y, x + 15, y + 30);
    setlinestyle(PS_SOLID, 2);
    break;
case 3:
    line(x + 15, y + 15, x + 15, y + 30);
    setlinestyle(PS_SOLID, 3);
    line(x - 1, y + 15, x + 30, y + 15);
    setlinestyle(PS_SOLID, 2);
    break;
case 4:
    line(x + 15, y, x + 15, y + 15);
    setlinestyle(PS_SOLID, 3);
    line(x - 1, y + 15, x + 30, y + 15);
    setlinestyle(PS_SOLID, 2);
    break;
case 5:
    setlinestyle(PS_SOLID, 3);
    line(x + 15, y, x + 15, y + 15);

```

```

    line(x + 15, y + 15, x + 30, y + 15);
    setlinestyle(PS_SOLID, 2);
    break;
case 6:
    setlinestyle(PS_SOLID, 3);
    line(x + 15, y, x + 15, y + 15);
    line(x - 1, y + 15, x + 15, y + 15);
    setlinestyle(PS_SOLID, 2);
    break;
case 7:
    setlinestyle(PS_SOLID, 3);
    line(x - 1, y + 15, x + 15, y + 15);
    line(x + 15, y + 15, x + 15, y + 30);
    setlinestyle(PS_SOLID, 2);
    break;
case 8:
    setlinestyle(PS_SOLID, 3);
    line(x + 15, y + 15, x + 30, y + 15);
    line(x + 15, y + 15, x + 15, y + 30);
    setlinestyle(PS_SOLID, 2);
    break;
case 9:
    line(x + 15, y, x + 15, y + 30);
    line(x - 1, y + 15, x + 30, y + 15);
    setfillcolor(BLACK);
    setlinestyle(PS_SOLID, 1);
    fillcircle(x + 15, y + 15, 4);
    break;
}
switch (value)
{
case 0: // 白棋
    setfillcolor(WHITE);
    setlinestyle(PS_SOLID, 1);
    fillcircle(x + 15, y + 15, 13);
    break;
case 1: // 黑棋
    setfillcolor(BLACK);
    setlinestyle(PS_SOLID, 1);
    fillcircle(x + 15, y + 15, 13);
    break;
}
setfillcolor(thefillcolor); // 还原填充色

```

```

}

```

```
// 其他函数定义
```

```
// 绘制棋盘
```

```
void draw()
```

```
{
    int number = 0; // 坐标输出的位置
    // 坐标 (数值)
    TCHAR strnum[19][3] =
{   _T("1"),_T("2")   _T("3")   _T("4"),_T("5")   _T("6")   _T("7"),_T("8"),_T("9"),_T("10"),
    _T("11"),_T("12") _T("13") _T("14"),_T("15") _T("16") _T("17"),_T("18"),_T("19") };
    // 坐标 (字母)
    TCHAR strabc[19][3] =
{   _T("A"),_T("B")   _T("C")   _T("D"),_T("E")   _T("F")   _T("G"),_T("H"),_T("I"),_T("J"),
    _T("K"),_T("L") _T("M") _T("N"),_T("O") _T("P") _T("Q"),_T("R"),_T("S") };
    LOGFONT nowstyle;
    gettextstyle(&nowstyle);
    settextstyle(0, 0, NULL);
    for (int i = 0; i < 19; i++)
    {
        for (int j = 0; j < 19; j++)
        {
            BOX[i][j].draw(); // 绘制
            if (BOX[i][j].isnew == true)
            {
                BOX[i][j].isnew = false; // 把上一个下棋位置的黑框清除
            }
        }
    }
    // 标注坐标
    for (int i = 0; i < 19; i++)
    {
        outtextxy(75 + number, 35, strabc[i]);
        outtextxy(53, 55 + number, strnum[i]);
        number += 30;
    }
    settextstyle(&nowstyle);
}
```

```
// 棋盘初始化
```

```
void init()
```

```
{
    win = -1; // 谁赢了
    for (int i = 0, k = 0; i < 570; i += 30)
```

```

{
    for (int j = 0, g = 0; j < 570; j += 30)
    {
        int background = 0; // 棋盘样式
        BOX[k][g].value = -1;
        BOX[k][g].color = RGB(255, 227, 132); // 棋盘底色
        // x、y 坐标
        BOX[k][g].x = 65 + j;
        BOX[k][g].y = 50 + i;
        // 棋盘样式的判断
        if (k == 0 && g == 0)
            background = 8;
        else if (k == 0 && g == 18)
            background = 7;
        else if (k == 18 && g == 18)
            background = 6;
        else if (k == 18 && g == 0)
            background = 5;
        else if (k == 0)
            background = 3;
        else if (k == 18)
            background = 4;
        else if (g == 0)
            background = 1;
        else if (g == 18)
            background = 2;
        else
            background = 0;
        BOX[k][g].background = background;
        g++;
    }
    k++;
}
}

```

// 核心函数

// 寻找最佳位置

location findbestlocation(int color, int c)

```

{
    if (c == 0)
    {
        //如果是第一层
        //清空数组
    }
}

```

```

        bestcnt = 0;
    }
    int MAXnumber = INT_MIN;    //最佳分数
    for (int i = 0; i < 19; i++) {
        for (int j = 0; j < 19; j++) {
            if (BOX[i][j].value == -1) { // 空位 -1
                //遍历每一个空位置
                int length;        //当前方向长度
                int emeny;        //当前方向敌子
                int nowi = 0;      //现在遍历到的 y 坐标
                int nowj = 0;      //现在遍历到的 x 坐标
                int thescore = 0; //这个位置的初始分数

                //判断 5X5 的范围内有没有棋子
                int flag = 0;
                for (int k = 0; k < 4; k++)
                {
                    nowi = i;
                    nowj = j;
                    nowi += dx[k];
                    nowj += dy[k];
                    if (nowi >= 0 && nowj >= 0&& nowi <= 18 && nowj <=
18&& BOX[nowi][nowj].value != -1) // 测试某方向的棋子，在棋盘内，且非空
                    {
                        flag = 1;
                        break;
                    }
                    nowi = i;
                    nowj = j;
                    nowi += dx[k];
                    nowj += dy[k];
                    if (nowi >= 0 && nowj >= 0&& nowi <= 18 && nowj <=
18&& BOX[nowi][nowj].value != -1)
                    {
                        flag = 1;
                        break;
                    }
                    nowi = i;
                    nowj = j;
                    nowi -= dx[k];
                    nowj -= dy[k];
                    if (nowi >= 0 && nowj >= 0&& nowi <= 18 && nowj <=
18&& BOX[nowi][nowj].value != -1)
                    {

```

```

        flag = 1;
        break;
    }
    nowi = i;
    nowj = j;
    nowi -= dx[k];
    nowj -= dy[k];
    if (nowi >= 0 && nowj >= 0 && nowi <= 18 && nowj <=
18 && BOX[nowi][nowj].value != -1)
    {
        flag = 1;
        break;
    }
}
if (!flag)
{
    //如果周围没有棋子，查找下一个空位
    continue;
}

//己方
BOX[i][j].value = color; // 尝试下在这里
for (int k = 0; k < 4; k++) // 检测四个方向 自己棋子和对方棋子
的数量
{

    length = 0;
    emeny = 0;
    nowi = i;
    nowj = j;
    while (nowi <= 18 && nowj <= 18 && nowi >= 0 &&
nowj >= 0 && BOX[nowi][nowj].value == color) // 己方棋子
    {
        length++;
        nowj += dy[k];
        nowi += dx[k];
    }
    if (nowi < 0 || nowj < 0 || nowi > 18 || nowj > 18 ||
BOX[nowi][nowj].value == !color) // 对方棋子
    {
        emeny++;
    }

    nowi = i;

```



```

        nowj = j;
        while (nowi <= 18 && nowj <= 18 && nowi >= 0 &&
nowj >= 0 && BOX[nowi][nowj].value == color) // 己方棋子
        {
            length++;
            nowj -= dy[k];
            nowi -= dx[k];
        }
        if (nowi < 0 || nowj < 0 || nowi > 18 || nowj > 18 ||
BOX[nowi][nowj].value == !color) // 对方棋子
        {
            emeny++;
        }

        length -= 2; // 判断长度 （减 2 是因为从假设的 nowi,j 出
发本身算了两次，需要的 length 是已落子的实际长度）
        if (length > 4)
        {
            length = 4;
        }
        if (score[emeny][length] == 500)
        {
            BOX[i][j].value = -1;
            return{ i,j,score[emeny][length] };
        }
        thescore += score[emeny][length];
        length = 0;
        emeny = 0;
    }
}

```

```

//敌人（原理同上）
BOX[i][j].value = !color;
for (int k = 0; k < 4; k++)
{
    length = 0;
    emeny = 0;
    nowi = i;
    nowj = j;
    while (nowi <= 18 && nowj <= 18 && nowi >= 0 &&
nowj >= 0 && BOX[nowi][nowj].value == !color)
    {
        length++;
        nowj += dy[k];
    }
}

```

```

        nowi += dx[k];
    }
    if (nowi < 0 || nowj < 0 || nowi > 18 || nowj > 18 ||
BOX[nowi][nowj].value == color)
    {
        emeny++;
    }
    nowi = i;
    nowj = j;
    while (nowi <= 18 && nowj <= 18 && nowi >= 0 &&
nowj >= 0 && BOX[nowi][nowj].value == !color)
    {
        length++;
        nowj -= dy[k];
        nowi -= dx[k];
    }
    if (nowi < 0 || nowj < 0 || nowi > 18 || nowj > 18 ||
BOX[nowi][nowj].value == color)
    {
        emeny++;
    }
    length -= 2;
    if (length > 4)
    {
        length = 4;
    }
    if (score[emeny][length] == 500)
    {
        BOX[i][j].value = -1;
        return{ i,j,score[emeny][length] };
    }
    thescore += score[emeny][length];
    length = 0;
    emeny = 0;
}

BOX[i][j].value = -1;
if (c < 3) // 超过 4 层，时间太长；少于 4 层，决策太差
{

```

BOX[i][j].value = color; // 置顶该位置为 color，递归寻找对方！ color 的分数

```

int nowscore = thescore - findbestlocation(!color, c +
1).number - 50*c; // 递归求出对方在没有此位置后可获得的最优的分值 nowscore 代表（己

```

方分-对方分)

清空

交换棋子

```
BOX[i][j].value = -1;
if (nowscore > MAXnumber)
{
    MAXnumber = nowscore;
    if (c == 0)
    {
        bestcnt = 0; // 每个位置第一次递归时，重置数组
    }
}
if (c == 0)
{
    //第一层
    if (nowscore >= MAXnumber)
    {
        //把当前位置加入数组
        bestx[bestcnt] = i;
        besty[bestcnt] = j;
        bestcnt++;
    }
}
else {
    //如果递归到了第四层
    if (thescore > MAXnumber)
    {
        MAXnumber = thescore;
    }
}
}
}
if (MAXnumber == 500 && playcnt == 3) // 如果三手黑方优势，电脑决定

    changeflag = true;
    if (c == 0)
    {
        int mynum = rand() % bestcnt;
        return { bestx[mynum],besty[mynum],MAXnumber };
    }
    //其他层
    return { 0,0,MAXnumber };
}
```

```

// 判断输赢
void isWIN()
{
    bool isfull = true; // 棋盘是否满了
    for (int i = 0; i < 19; i++)
    {
        for (int j = 0; j < 19; j++)
        {
            if (BOX[i][j].value != -1)
            {
                // 遍历每个可能的位置
                int nowcolor = BOX[i][j].value; // 现在遍历到的颜色
                int length[4] = { 0,0,0,0 }; // 四个方向的长度
                for (int k = 0; k < 4; k++)
                {
                    // 原理同寻找最佳位置
                    int nowi = i;
                    int nowj = j;
                    while (nowi <= 18 && nowj <= 18 && nowi >= 0 &&
nowj >= 0 && BOX[nowi][nowj].value == nowcolor)
                    {
                        length[k]++;
                        nowj += dx[k];
                        nowi += dy[k];
                    }
                    nowi = i;
                    nowj = j;
                    while (nowi <= 18 && nowj <= 18 && nowi >= 0 &&
nowj >= 0 && BOX[nowi][nowj].value == 1 - nowcolor)
                    {
                        length[k]++;
                        nowj -= dx[k];
                        nowi -= dy[k];
                    }
                }
            }
            for (int k = 0; k < 4; k++)
            {
                if (length[k] >= 5) {
                    // 如果满五子
                    if (nowcolor == playercolor)
                    {
                        win = playercolor; // 玩家胜
                    }
                }
            }
        }
    }
}

```

```

        if (nowcolor == 1 - playercolor)
        {
            win = 1 - playercolor; // 电脑胜
        }
    }
}
else
{
    //如果为空
    isfull = false; //棋盘没满
}
}
}
if (isfull)
{
    // 如果棋盘满了
    win = 2; // 平局
}
}

```

// 游戏主函数

void game()

```

{
    bool isinit;
    // 上一个鼠标停的坐标
    int oldi = 0;
    int oldj = 0;

    playercolor = 1;
    // 绘制背景
    setfillcolor(RGB(255, 227, 132));
    solidrectangle(40, 25, 645, 630);
    // 设置字体样式
    settextstyle(30, 15, 0, 0, 0, 1000, false, false, false);
    settextcolor(BLACK);
    // 输出标示语
    if (playercolor == 0)
    {
        isinit = 1;
        outtextxy(150, 650, _T("玩家执白后行，电脑执黑先行"));
        whosturn = 1;
    }
    else

```

```

{
    isinit = 0;
    outtextxy(150, 650, _T("玩家执黑先行，电脑执白后行"));
    whosturn = 0;
}
draw(); // 绘制
while (1)
{
    NEXTPLAYER:
    // 玩家下棋
    if (whosturn == 0)
    {
        MOUSEMSG mouse = GetMouseMsg(); // 获取鼠标信息
        for (int i = 0; i < 19; i++)
        {
            for (int j = 0; j < 19; j++)
            {
                if (mouse.x > BOX[i][j].x && mouse.x < BOX[i][j].x + 30 //判断
                    && mouse.y > BOX[i][j].y && mouse.y < BOX[i][j].y +
                    30 //判断 y 坐标
                    && BOX[i][j].value == -1) //判断是否是空位置
                {
                    // 如果停在某一个空位置上面
                    if (mouse.mkLButton)
                    {
                        // 如果按下了
                        BOX[i][j].value = playercolor; // 下棋
                        BOX[i][j].isnew = true; // 新位置更新
                        oldi = -1;
                        oldj = -1;
                        // 下一个玩家
                        whosturn = 1;
                        goto DRAW;
                    }
                    // 更新选择框
                    BOX[oldi][oldj].isnew = false;
                    BOX[oldi][oldj].draw();
                    BOX[i][j].isnew = true;
                    BOX[i][j].draw();
                    oldi = i;
                    oldj = j;
                }
            }
        }
    }
}

```

```

    }
}
// 电脑下棋
else
{
    if (isinit)
    {
        // 开局情况
        isinit = 0;
        int drawi = 9;
        int drawj = 9;
        while (BOX[drawi][drawj].value != -1)
        {
            drawi--;
            drawj++;
        }
        BOX[drawi][drawj].value = 1 - playercolor;
        BOX[drawi][drawj].isnew = true;
    }
    else
    {
        location best;
        best = findbestlocation(1 - playercolor, 0); // 寻找最佳位置
        if (changeflag)
            break;
        BOX[best.i][best.j].value = 1 - playercolor; // 下在最佳位置
        BOX[best.i][best.j].isnew = true;
    }
    whosturn = 0;
    goto DRAW; // 轮到下一个
}
}

DRAW: // 绘制
isWIN(); // 检测输赢
draw();
oldi = 0;
oldj = 0;
if (win == -1)
{
    // 如果没有人胜利
    Sleep(500);
    playcnt          =          playcnt          +          1;
////////////////////
    if (changeflag) { // 三手交换

```

```

        playercolor = 0;
        whosturn = 0;
        setfillcolor(RGB(255, 255, 255));
        fillrectangle(000, 650, 700, 680);
        setfillcolor(RGB(255, 227, 132));
        outtextxy(115, 650, _T("电脑选择三手交换， 玩家现执白棋"));
    }
    changeflag = false;
    goto NEXTPLAYER; // 前往下一个玩家
}
// 胜利处理
settextcolor(RGB(0, 255, 0));
Sleep(1000);
if (win == 0)
{
    fillrectangle(000, 650, 700, 680);
    setfillcolor(RGB(255, 227, 132));
    settextcolor(BLACK);
    outtextxy(320, 650, _T("白胜"));
}
if (win == 1)
{
    fillrectangle(000, 650, 700, 680);
    setfillcolor(RGB(255, 227, 132));
    settextcolor(BLACK);
    outtextxy(320, 650, _T("黑胜"));
}
if (win == 2)
{
    fillrectangle(000, 650, 700, 680);
    setfillcolor(RGB(255, 227, 132));
    settextcolor(BLACK);
    outtextxy(320, 650, _T("平局"));
}
Sleep(50000);
return;
}

```