博弈树算法：

#include<iostream>

#include <stdlib.h>

#include<windows.h>

using namespace std;

int list[8][8] = { 0 }; //棋盘状态,1是黑子，2是白子，0是空的

int p = 1, aip = 0, air, aic, lastr = 0, lastc = 0; //p是现在该谁下，1是黑方，2是白方

int p1, p2, p3, p4, temp1[8][8], temp2[8][8], temp3[8][8], temp4[8][8];//4层博弈对应的AI分数和棋局

const int dr[8] = { 0, 0, 1, 1, 1, -1, -1, -1 }, dc[8] = { 1, -1, 0, 1, -1, 0, 1, -1 };//8个方向向量

int book[65]; //棋谱

const int no = 1000;//AI的分数不可能达到的上界值

const int prio[8][8] =

{ //prio是每个位置的优先级，数值越高越好

{ 80, -20, 8, 6, 6, 8, -20, 80 },

{ -20, -20, 0, -1, -1, 0, -20, -20 },

{ 8, 0, 3, 5, 5, 3, 0, 8 },

{ 6, -1, 5, 1, 1, 5, -1, 6 },

{ 6, -1, 5, 1, 1, 5, -1, 6 },

{ 8, 0, 3, 5, 5, 3, 0, 8 },

{ -20, -20, 0, -1, -1, 0, -20, -20 },

{ 80, -20, 8, 6, 6, 8, -20, 80 }

};

bool playOK(int r, int c, int dr, int dc) //判断某个格子的某个方向能否下子

{

if (list[r][c] != 0)return false;

int tr = r, tc = c; //tr和tc分别表示该点通过行和列往特定方向移动后的坐标

while (tr + dr >= 0 && tr + dr < 8 && tc + dc >= 0 && tc + dc < 8 && list[tr + dr][tc + dc] == 3 - p)

{ //循环遍历，未到达边界或者右边的棋子是对方的则循环继续，否则循环退出

tr += dr, tc += dc; //移动坐标

}

//若使循环退出的那一格里，是对方的棋子，则(r,c)可落子，否则不可落子

if (tr == r && tc == c)return false; //难点，这一句不可少

if (tr + dr >= 0 && tr + dr < 8 && tc + dc >= 0 && tc + dc < 8 && list[tr + dr][tc + dc] == p)return true;

return false;

}

bool OK(int r, int c) //判断某个格子能否下子

{

if (list[r][c])return false;

for (int i = 0; i < 8; i++) //只要一个方向满足可以下的条件，就可以下

if (playOK(r, c, dr[i], dc[i]))return true; //调用

return false;

}

int num(int k) //统计棋子数目，1是黑子，2是白子

{

int s = 0;

for (int i = 0; i < 8; i++)for (int j = 0; j < 8; j++)if (list[i][j] == k)s++;

return s;

}

void display() //显示棋盘和棋子

{

system("cls");

for (int i = 0; i < 8; i++)

{

if (!i)

{

cout << " ";

for (int j = 0; j < 8; j++)cout << char('A' + j) << " ";

cout << endl;

}

cout << i + 1 << " ";

for (int j = 0; j < 8; j++)

{

if (list[i][j] == 2)cout << "○"; else if (list[i][j] == 1)cout << "●";

else if (OK(i, j))cout << "？"; else cout << " ."; //调用

cout << " ";

}

cout << endl << endl;

}

cout << "黑方：" << num(1) << " 白方：" << num(2) << " 轮到"; //调用

if (p == 1)cout << "黑方下\n"; else cout << "白方下\n";

cout << "候选项:";

for (int i = 0; i < 8; i++)for (int j = 0; j < 8; j++)if (OK(i, j)) //调用

cout << " " << char('1' + i) << char('A' + j);

cout << "\n最后一个落子位置是" << lastr + 1 << char(lastc + 'A');

}

void init() //初始化

{

list[3][3] = list[4][4] = 2;

list[3][4] = list[4][3] = 1;

while (aip != 1 && aip != 2 && aip != 3)

{

cout << "选择\n1，人机对战，AI先\n2，人机对战，玩家先\n3，双人对战\n";

cin >> aip;

}

display(); //调用

}

bool end\_() //判断游戏是否结束

{

for (int i = 0; i < 8; i++)for (int j = 0; j < 8; j++)if (OK(i, j))return false; //调用

p = 3 - p; //改变p的2个地方之一

for (int i = 0; i < 8; i++)for (int j = 0; j < 8; j++)if (OK(i, j))return false; //调用

display();

cout << "\n游戏结束\n";

if (num(1) < num(2))cout << "白方胜利"; //调用

else if (num(1)>num(2))cout << "黑方胜利"; //调用

else cout << "平局";

return true;

}

void turn(int tr, int tc, int dr, int dc) //吃子函数play()的一个方向

{

if (!playOK(tr, tc, dr, dc))return; //难点，这一句不可少 //调用

while (tr + dr >= 0 && tr + dr < 8 && tc + dc >= 0 && tc + dc < 8 && list[tr + dr][tc + dc] == 3 - p)

{

list[tr + dr][tc + dc] = p; //在该处换掉棋子的颜色

tr += dr, tc += dc;

}

}

void play(int r, int c)

{

lastr = r, lastc = c;

for (int i = 0; i < 8; i++)turn(r, c, dr[i], dc[i]); //调用

list[r][c] = p;

book[num(1) + num(2)] = r \* 8 + c; //调用

p = 3 - p; //改变p的2个地方之一

}

int getPoint()

{

int point = 0;

for (int i1 = 0; i1<8; i1++)for (int j1 = 0; j1<8; j1++)

{

if (list[i1][j1] == aip)point += prio[i1][j1]; //若是ai的子，则加上该位置的积分

if (list[i1][j1] == 3 - aip) point -= prio[i1][j1]; //若是玩家的子，则减去该位置的积分

}

return point;

}

void AI3()//函数里面会改变p，但是调用函数不会改变p

{

p4 = no;

for (int iiii = 0; iiii < 8; iiii++)for (int jjjj = 0; jjjj < 8; jjjj++) //第四层

{

if (!OK(iiii, jjjj))continue; //调用

for (int i = 0; i<8; i++)for (int j = 0; j<8; j++)temp4[i][j] = list[i][j];

play(iiii, jjjj);//调用

p = 3 - p;

if (p4 > getPoint())p4 = getPoint();//调用

for (int i = 0; i<8; i++)for (int j = 0; j<8; j++)list[i][j] = temp4[i][j];

}

if (p4 == no)p4 = getPoint(); //调用

if (p3 < p4)p3 = p4;

}

void AI2()//函数里面会改变p，但是调用函数不会改变p

{

p3 = -no;

for (int iii = 0; iii < 8; iii++)for (int jjj = 0; jjj < 8; jjj++) //第三层

{

if (!OK(iii, jjj))continue; //调用

for (int i = 0; i<8; i++)for (int j = 0; j<8; j++)temp3[i][j] = list[i][j];

play(iii, jjj);//调用

AI3();

p = 3 - p;

for (int i = 0; i<8; i++)for (int j = 0; j<8; j++)list[i][j] = temp3[i][j];

}

if (p3 == -no)

{

p = 3 - p;

AI3();

p = 3 - p;

}

if (p2 > p3)p2 = p3;

}

void AI() //函数里面会改变p，但是调用函数不会改变p

{

p1 = -no;

for (int i = 0; i<8; i++)for (int j = 0; j<8; j++) //第一层

{

if (!OK(i, j))continue; //调用

for (int i = 0; i<8; i++)for (int j = 0; j<8; j++)temp1[i][j] = list[i][j];

play(i, j);//调用

p2 = no;

for (int ii = 0; ii < 8; ii++)for (int jj = 0; jj < 8; jj++) //第二层

{

if (!OK(ii, jj))continue; //调用

for (int i = 0; i<8; i++)for (int j = 0; j<8; j++)temp2[i][j] = list[i][j];

play(ii, jj);//调用

AI3(); //调用

p = 3 - p;

for (int i = 0; i<8; i++)for (int j = 0; j<8; j++)list[i][j] = temp2[i][j];

}

if (p2 == no)

{

p = 3 - p;

AI2(); //调用

p = 3 - p;

}

if (p1 < p2)p1 = p2, air = i, aic = j; //难点，如果调用AI函数却1次都没有执行这条语句，那么AI就会下错位置

p = 3 - p;

for (int i = 0; i<8; i++)for (int j = 0; j<8; j++)list[i][j] = temp1[i][j];

}

}

void go() //落下一个子

{

display(); //调用

if (p == aip)

{

AI();//调用

play(air, aic);//调用

Sleep(1000);

return;

}

cout << "\n输入落子位置：行（1-8）和列（A-H） 输入00查看棋谱" << endl;

char x, y;

cin >> x >> y;

if (x == '0')

{

for (int i = 5; i <= num(1) + num(2); i++)cout << " " << book[i] / 8 + 1 << char(book[i] % 8 + 'A');

cout << "\n按任意键退出";

system("pause>nul");

return;

}

int r = x - '1', c = y - 'a';

if (y >= 'A' && y < 'Z')c = y - 'A';

if (!OK(r, c)) //调用

{

cout << "ERROR!";

return;

}

play(r, c); //调用

}

int main()

{

system("color f0");//白底黑字

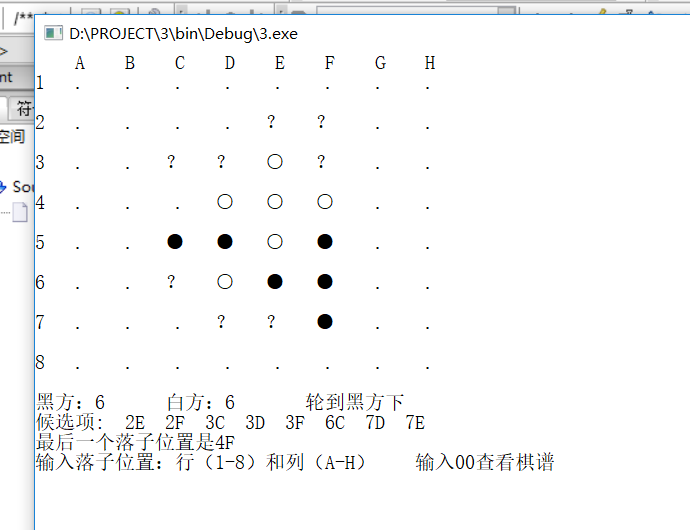
init();

while (!end\_())go();

system("pause>nul");

return 0;

}



对抗搜索：



#include<cstdio>

#include<cstdlib>

#include<algorithm>

using namespace std;

const int inf=1e9;

int n,a,b,c,d,f[2][69][21][21][21][21];

int dx[8]={1,0,-1,0,2,0,-2,0};

int dy[8]={0,1,0,-1,0,2,0,-2};

int dfs(int x,int y,int a,int b,int c,int d)

{if(y>3\*n) return inf;

if(a==c && b==d){if(x) return inf; return 0;}

if(f[x][y][a][b][c][d]) return f[x][y][a][b][c][d];

int ans,xx,yy;

if(x)

{ans=inf;

for(int i=0;i<=7;i++)

{xx=c+dx[i],yy=d+dy[i];

if(1<=xx && xx<=n && 1<=yy && yy<=n)

ans=min(ans,dfs(0,y+1,a,b,xx,yy));

}

}

else

{ans=0;

for(int i=0;i<=3;i++)

{xx=a+dx[i],yy=b+dy[i];

if(1<=xx && xx<=n && 1<=yy && yy<=n)

ans=max(ans,dfs(1,y+1,xx,yy,c,d));

}

}

ans++;

return f[x][y][a][b][c][d]=ans;

}

int main()

{scanf("%d%d%d%d%d",&n,&a,&b,&c,&d);

if(abs(a-c)+abs(b-d)<=1) printf("WHITE 1");

else printf("BLACK %d",dfs(0,0,a,b,c,d));

return 0;

}

