### 第8题：棋局评估

#### 数据结构：

二维数组

#### 算法设计思想：

Alice（以下简称A）与Bob（以下简称B）都以最优策略下棋，用二维数组记录两人下棋的情况。先统计当前空格数（两人还没下的格子），然后进行深度优先搜索进行递归，搜索所有下棋的策略，当空格数为0时递归结束，并开始回溯，计算A得分情况并返回A得分情况，并把每一条策略进行比较，得出Alice的最优得分情况

每步递归根据当前递归下棋的对象改变下一步递归中下棋的对象。判断找寻当前对象的最优策略时需要比较函数回溯得来的A得分情况（B最优策略是使A得分情况最少，A最优策略是使A得分情况最多）

#### 源程序：

#include<iostream>

using namespace std;

#define MAX -99

#define MIN 99

//采用深度优先算法

// 0:null

// 1:Alice: X(first)

// 2:Bob: O

int board[3][3]; //数组形式记录棋盘

bool Judge (int side){

//判断谁胜利

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

//行连成一条线

if (board[i][0] == board[i][1] && board[i][1] == board[i][2] && board[i][2] == side )

return true;

//列连成一条线

if (board[0][i] == board[1][i] && board[1][i] == board[2][i] && board[2][i] == side )

return true;

}

//判断对角线

if (board[0][0] == board[1][1] && board[1][1] == board[2][2] && board[2][2] == side)

return true;

if(board[0][2] == board[1][1] && board[1][1] == board[2][0] && board[2][0] == side)

return true;

return false;

}

int DFS(int side){

int nullNum = 0;

int maxFinal = MAX;

int minFinal = MIN;

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

for (int j = 0; j < 3; j++){

if (board[i][j] == 0)

nullNum ++;

}

}

//返回得分情况

if (side == 1 && Judge(2))

return -nullNum-1;

else if (side == 2 &&Judge(1))

return nullNum+1;

else if (nullNum == 0)

return 0;

//进行下棋

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

for (int j = 0; j < 3; j++){

if (board[i][j] == 0){

board[i][j] = side;

if (side == 1) //当前为Alice下棋，计算 下一步Bob下棋Bob最优策略得分

maxFinal = max(maxFinal,DFS(2));

if (side == 2) //当前为Bob下棋，计算 下一步Alice下棋Alice最优策略得分

minFinal = min(minFinal,DFS(1));

board[i][j] = 0;//回溯时重新初始化

}

}

}

if (side == 1) return maxFinal;

if (side == 2) return minFinal;

}

int main() {

int group;

cin >> group;

while (group--){

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

for (int j = 0; j < 3; j++){

cin >> board[i][j];

}

}

cout << DFS(1) << endl;

}

return 0;

}

//代码总行数：73行

#### 测试数据和结果：

输入：

3

1 2 1

2 1 2

0 0 0

2 1 1

0 2 1

0 0 2

0 0 0

0 0 0

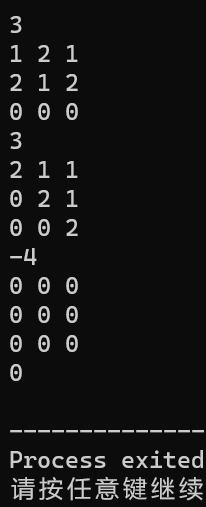
0 0 0

输出：

3

-4

0



#### 时间复杂度：

DFS时间复杂度O(n^2），判断输赢的函数时间复杂度O（n^2）

#### 该题代码行：

73行