迭代加深搜索

问题描述：

在一个5×5的棋盘上有12个白色的骑士和12个黑色的骑士， 且有一个空位。在任何时候一个骑士都能按照骑  
士的走法（它可以走到和它横坐标相差为1，纵坐标相差为2或者横坐标相差为2，纵坐标相差为1的格子）移动到空  
位上。 给定一个初始的棋盘，怎样才能经过移动变成如下目标棋盘： 为了体现出骑士精神，他们必须以最少的步  
数完成任务。

输入：　第一行有一个正整数T(T<=10)，表示一共有N组数据。接下来有T个5×5的矩阵，0表示白色骑士，1表示黑色骑  
士，\*表示空位。两组数据之间没有空行

输出：对于每组数据都输出一行。如果能在15步以内（包括15步）到达目标状态，则输出步数，否则输出－1

代码：

#include<iostream>

#include<cstdio>

#include<cstdlib>

#include<cmath>

#include<ctime>

#include<cctype>

#include<cstring>

#include<string>

#include<algorithm>

using namespace std;

const char goal[5][5]={{'1','1','1','1','1'},

{'0','1','1','1','1'},

{'0','0','\*','1','1'},

{'0','0','0','0','1'},

{'0','0','0','0','0'}};

int sx,sy,T,step;

char map[5][5];

struct node

{

int gox;

int goy;

}go[8];

inline void pre()

{

go[0].gox=1,go[0].goy=2;

go[1].gox=1,go[1].goy=-2;

go[2].gox=2,go[2].goy=1;

go[3].gox=2,go[3].goy=-1;

go[4].gox=-1,go[4].goy=2;

go[5].gox=-1,go[5].goy=-2;

go[6].gox=-2,go[6].goy=1;

go[7].gox=-2,go[7].goy=-1;

}

inline bool jud(int x,int y)

{

return x>=0&&x<=4&&y>=0&&y<=4;

}

inline int check()

{

int dif=0;

for(int i=0;i<=4;i++)

for(int j=0;j<=4;j++)

if(map[i][j]!=goal[i][j]) dif++;

return dif;

}

inline bool dfs(int x,int y,int now)

{

if(now==step+1)

{

if(check()==0) return true;

else

{return false;}

}

if(now+check()-3>step) {return false;}

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

{

int tx=x+go[i].gox;

int ty=y+go[i].goy;

if(jud(tx,ty))

{

swap(map[x][y],map[tx][ty]);

if(dfs(tx,ty,now+1)) return true;

swap(map[x][y],map[tx][ty]);

}

}

return false;

}

int main()

{

//freopen("a.in","r",stdin);

printf("输入将要进行的组数：");

scanf("%d",&T);

pre();

while(T--)

{

sx=sy=0;

for(int i=0;i<=4;i++)

scanf("%s",map[i]);

bool flag=false;

for(i=0;i<=4;i++)

{

if(flag==true) break;

for(int j=0;j<=4;j++)

if(map[i][j]=='\*')

{

sx=i,sy=j;

flag=true;

break;

}

}

step=1;

for(i=1;i<=15;i++)

{

if(dfs(sx,sy,1)) break;

step++;

}

if(step==16) cout<<"-1"<<endl;

else cout<<step<<endl;

}

return 0;

}

输出结果：

