171491218刘晓萌

人工智能实验七

1. 实验题目

八数码问题

1. 实验分析

八数码问题也称为九宫问题。在3×3的棋盘，摆有八个棋子，每个棋子上标有1至8的某一数字，不同棋子上标的数字不相同。棋盘上还有一个空格，与空格相邻的棋子可以移到空格中。要求解决的问题是：给出一个初始状态和一个目标状态，找出一种从初始转变成目标状态的移动棋子步数最少的移动步骤。所谓问题的一个状态就是棋子在棋盘上的一种摆法。解八数码问题实际上就是找出从初始状态到达目标状态所经过的一系列中间过渡状态。

1. 实验内容

1. 空白棋子用0代替

2.接受初始节点的信息和目标节点的信息

3.找到空白棋子很简单，直接遍历就好，但是如何返回它的x和y坐标，试着能不能使用一个数字代替，后来发现确实可以。

4. 从初始节点开始判断，然后扩展，即上下左右移动，当然我们要考虑具体的位置，比如说已经到边界了，就不能越出边界。还要考虑以前移动过的方向，所以记录下来以前移动过的方向，可以直接加在结构体里。

5. 如何实现判断相等，就是如何标志该状态的唯一性，方案1：9个数字拼接生成字符串直接判断是否相等。方案二：由于数也不大，所以直接使用9个数字直接生成一个数，使用long long 声明生成的数字即可。

6. 如果保证能够到达目标节点，可以一直搜索下去，如果不知道能不能，则可以设置搜索次数。

1. 实验代码

#include<stdio.h>

struct node

{

int xy[3][3];

int dir;

};

struct node sh[102], end;

int count = 1;

void init()

{

printf("输入起始节点的位置:\n");

int i, j;

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

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

scanf("%d", &sh[0].xy[i][j]);

sh[0].dir = -1;

printf("输入目标节点的位置:\n");

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

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

scanf("%d", &sh[101].xy[i][j]);

sh[101].dir = -1;

}

//找出0的位置

int loction(int num)

{

int i;

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

if (sh[num].xy[i / 3][i % 3] == 0) return i;

}

//进行标记

long long sign(int num)

{

long long sum;

sum = sh[num].xy[0][0]\*100000000 + sh[num].xy[0][1]\*10000000 + sh[num].xy[0][2]\*1000000 + sh[num].xy[1][0]\*100000 + sh[num].xy[1][1]\*10000 + sh[num].xy[1][2]\*1000 + sh[num].xy[2][0]\*100 + sh[num].xy[2][1]\*10 + sh[num].xy[2][2];

return sum;

}

void mobile(int num)

{

int temp;

int loc;

int up = 1, down = 1, left = 1, right = 1;

loc = loction(num);

int stand = sh[num].dir;

//dir的0 1 2 3分别代表左 上 右 下

if (loc / 3 != 0 && stand != 1)

{

sh[count] = sh[num];

temp = sh[count].xy[loc / 3][loc % 3];

sh[count].xy[loc / 3][loc % 3] = sh[count].xy[loc / 3 - 1][loc % 3];

sh[count].xy[loc / 3 - 1][loc % 3] = temp;

sh[count].dir = 3;

count++;

};

if (loc / 3 != 2 && stand != 3)

{

sh[count] = sh[num];

temp = sh[count].xy[loc / 3][loc % 3];

sh[count].xy[loc / 3][loc % 3] = sh[count].xy[loc / 3 + 1][loc % 3];

sh[count].xy[loc / 3 + 1][loc % 3] = temp;

sh[count].dir = 1;

count++;

}

if (loc % 3 != 0 && stand != 0)

{

sh[count] = sh[num];

temp = sh[count].xy[loc / 3][loc % 3];

sh[count].xy[loc / 3][loc % 3] = sh[count].xy[loc / 3][loc % 3 - 1];

sh[count].xy[loc / 3][loc % 3 - 1] = temp;

sh[count].dir = 2;

count++;

}

if (loc % 3 != 2 && stand != 2)

{

sh[count] = sh[num];

temp = sh[count].xy[loc / 3][loc % 3];

sh[count].xy[loc / 3][loc % 3] = sh[count].xy[loc / 3][loc % 3 + 1];

sh[count].xy[loc / 3][loc % 3 + 1] = temp;

sh[count].dir = 0;

count++;

}

}

void display(int num)

{

int i, j;

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

{

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

printf("%d ", sh[num].xy[i][j]);

printf("\n");

}

}

int search()

{

int i = 0;

while (1)

{

printf("\n");

display(i);

printf("\n");

if (i == 100)

{

printf("超出了上限次数\n");

return 0;

}

if (sign(i) == sign(101))

{

printf("在第%d次找到了", i);

display(i);

return i;

}

mobile(i);

i++;

}

}

int main()

{

init();

search();

return 0;

}

1. 实验截图













