迭代加深算法的实现八数码问题

算法实现过程：先穿点一个三维数组迎来存放拼图，每次搜索都相当于空白块上下左右的运动，知道这个空白块经过几次运动后，整个拼图形成特定的形式，这就是八数码的实现思想，用迭代加深实现也就是把每次设定一个深度，规定空白块只能走这么多步骤，然后判断这些步骤里有没有能走出结果的，如果有那这个一定是最优解，如果没有就加深度，再让空白块找

具体代码实现：

#include <stdio.h>

enum direction{start,up,down,left,right};

int matrix[3][3];

int print\_matrix(int);

int all\_in\_place();

int iterative\_deepening(int k,int d,int i,int j,enum direction dir);

int main()

{

const int MAX\_DEPTH=10; //深度边界

int i,j,d;

for(i=0;i<9;i++) //输入0~9，0表示空格

scanf("%d",matrix[0]+i);

for(i=0; i<9&&matrix[0][i]; i++) //寻找空格位置

;

j = i%3;

i /= 3;

//迭代加深的DFS盲目搜索算法，深度边界从1到MAX\_DEPTH

for(d=1;d<MAX\_DEPTH && iterative\_deepening(1,d,i, j,start);d++)

;

return 0;

}

/\*

\* k : 当前深度

\* d : 深度边界

\* i,j : 空格位置

\* dir : 上一步移动方向

\*/

int iterative\_deepening(int k,int d,int i,int j,enum direction dir)

{

int ret=1;

if(k>d)

return ret;

if(all\_in\_place()) // 到达目标状态

return print\_matrix(k);

else

{

if(dir!=up&&i<2 ) //可以向下

{

matrix[i][j] = matrix[i+1][j];

matrix[i+1][j] = 0;

ret = iterative\_deepening(k+1,d,i+1,j,down);

matrix[i+1][j] = matrix[i][j];

matrix[i][j] = 0;

if(!ret)

return print\_matrix(k);

}

if(dir!=down&&i>0 ) //可以向上

{

matrix[i][j] = matrix[i-1][j];

matrix[i-1][j] = 0;

ret = iterative\_deepening(k+1,d,i-1,j,up);

matrix[i-1][j] = matrix[i][j];

matrix[i][j] = 0;

if(!ret)

return print\_matrix(k);

}

if(dir!=left&&j<2 ) //可以向右

{

matrix[i][j] = matrix[i][j+1];

matrix[i][j+1] = 0;

ret = iterative\_deepening(k+1,d,i,j+1,left);

matrix[i][j+1] = matrix[i][j];

matrix[i][j] = 0;

if(!ret)

return print\_matrix(k);

}

if(dir!=right&&j>0 ) //可以向左

{

matrix[i][j] = matrix[i][j-1];

matrix[i][j-1] = 0;

ret = iterative\_deepening(k+1,d,i,j-1,left);

matrix[i][j-1] = matrix[i][j];

matrix[i][j] = 0;

if(!ret)

return print\_matrix(k);

}

}

return ret;

}

int all\_in\_place()

{

const int object[3][3]={{1,2,3},{4,5,6},{7,8,0}};

int i;

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

if(matrix[0][i] != object[0][i])

return 0;

return 1;

}

int print\_matrix( int count)

{

int i,j;

printf("\n\nsetp %d\n",count);

for(i=0;i<3;i++,putchar('\n'))

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

printf("%d\t",matrix[i][j]);

return 0;

}

实验结果：