

# A\*算法

## 求解八数码问题



学 号 \_\_\_\_\_

姓 名 \_\_\_\_\_

专 业 \_\_\_\_\_ 计算机科学与技术专业 \_\_\_\_\_

授课老师 \_\_\_\_\_

## 目 录

1 实验概述 .....	1
1.1 实验目的 .....	1
1.2 实验内容 .....	1
2 实验方案设计 .....	2
2.1 总体设计思路与总体框架 .....	2
2.1.1 总体设计思路 .....	2
2.2.1 总体框架 .....	2
2.2 核心算法及基本原理 .....	2
2.2.1 核心算法—A*算法 .....	2
2.2.2 算法基本原理 .....	2
2.2.3 启发式函数算法设计 .....	2
2.3 模块设计 .....	2
2.4 其他创新内容或优化算法 .....	2
3 实验过程 .....	9
3.1 环境说明 .....	9
3.2 源代码清单 .....	9
3.2.1 AStar_head.h .....	9
3.2.2 AStar_function.cpp .....	9
3.2.3 AStar_show.cpp .....	9
3.2.4 AStar_main.cpp .....	9
3.3 实验结果展示 .....	39
3.3.1 基础界面展示 .....	39
3.3.2 较小数据量展示 .....	39
3.3.3 较大数据量展示 .....	39
3.3.4 实验结论 .....	39
4 总结 .....	50
4.1 实验中存在的问题及解决方案 .....	50
4.2 心得体会 .....	50
4.3 后续改进方向 .....	50
参考文献 .....	51

### 1 实验概述

#### 1.1 实验目的

熟悉和掌握启发式搜索策略的定义，评价函数  $f(n)$  和算法过程，并利用 A\* 算法求解 8 数码问题，理解求解流程和搜索顺序

#### 1.2 实验内容

##### 1.2.1 内容

以 8 数码问题为例，实现 A\* 算法的求解程序（编程语言不限）。要求设计至少两种不同的启发函数  $h(n)$ 。

设置相同的初始状态和目标状态，针对不同的评价函数求得问题的解，比较它们对搜索算法性能的影响，包括拓展结点数，生成结点数和运行时间等。画出结果比较的图表，并进行性能分析。

##### 1.2.2 要求

要求界面显示初始状态，目标状态和中间搜索步骤。

要求显示搜索过程，画出搜索过程生成的搜索树，并在每个节点显示对应节点的评价值  $f(n)$ ，并标注出最终结果所选用的解路线。

撰写实验报告，提交源代码（进行注释），实验报告、汇报 PPT

## 2 实验方案设计

### 2.1 总体设计思路与总体框架

#### 2.1.1 总体设计思路

本程序的总体设计思路是：

首先设计 AStarEightDigital 实现对八数码问题的求解，对外存在一个初始状态和中止状态的接口，计算结果返回一个 TreeList 数组，存储了搜索树信息。外部直接调用本类，分别传入起始和中止两个状态数组信息，调用类中对应计算函数，即可得到最终的搜索树信息，根据 TreeList 数组便可得到整个搜索过程中的所有信息。

再设计一个 EasyXSHOW 实现对八数码问题的起始和中止状态输入输出的可视化界面设置，运用 EasyX 库较好地实现可视化页面的展示。最终达到的效果是全程仅使用鼠标与程序进行交互，达到较好的交互便捷度。

程序设计安排再 4 个 cpp 文件中实现，分别为：

AStar\_head.h：存储整个项目类、结构体、函数、宏定义的声明，作为项目的头文件使用。

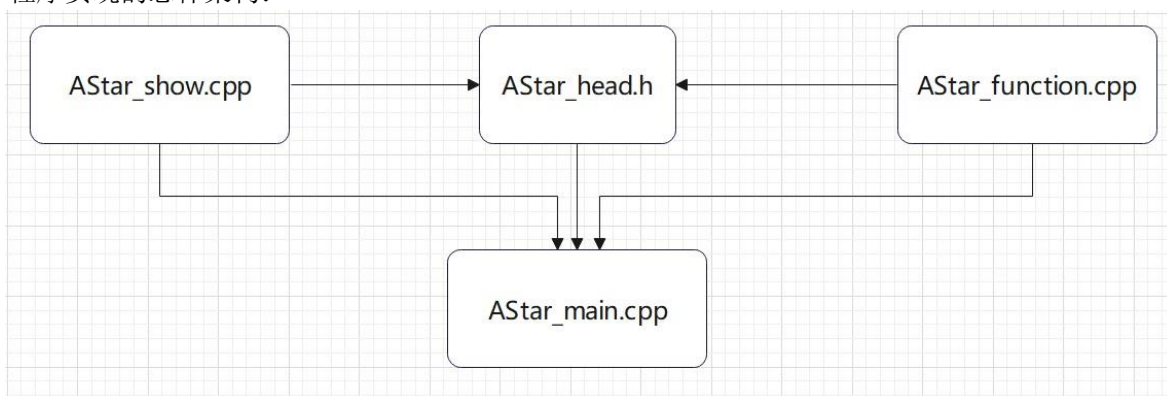
AStar\_function.cpp：内含 AStarEightDigital 类的具体实现函数，实现八数码问题的内部核心算法的求解。

AStar\_show.cpp：内含 EasyXSHOW 类的具体实现函数，实现八数码问题的可视化界面输入输出。

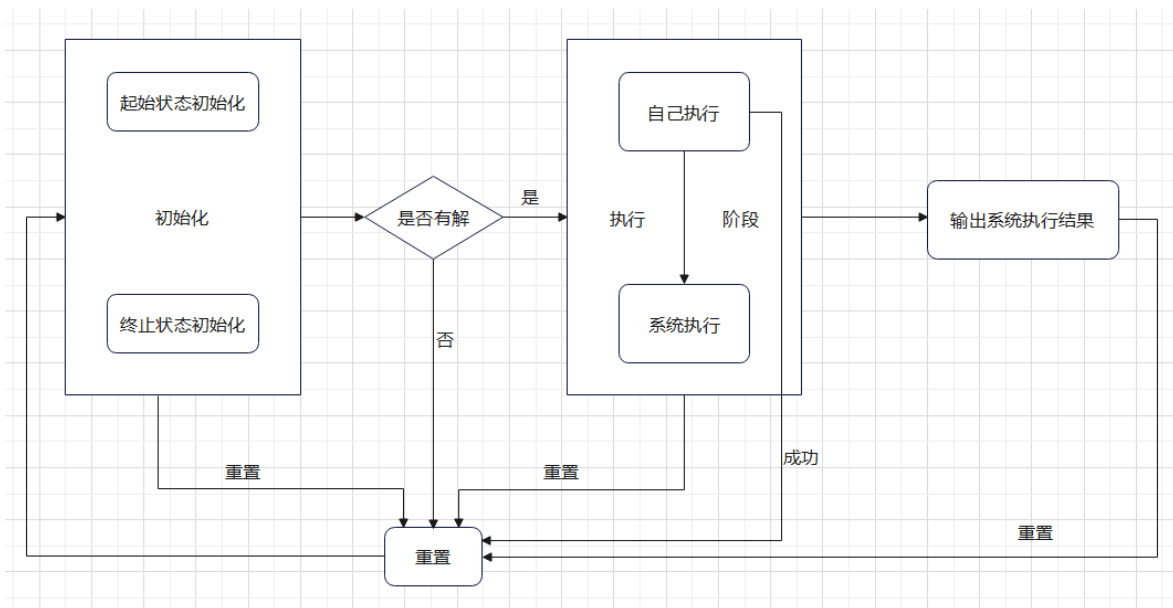
AStar\_main.cpp：程序主函数所在地，调用各个类进行对八数码问题的求解。

#### 2.1.2 总体架构

程序实现的总体架构：



根据此总体程序实现过程，以期望实现的程序运行流程如下：



## 2.2 核心算法及基本原理

### 2.2.1 核心算法—A\*算法

A\*可以认为是添加了启发式函数的 Dijkstra 算法，在 Dijkstra 算法的基础上，构造一个函数  $h(n)$ ， $n$  为当前扩展结点， $h(n)$  返回结点  $n$  到终点的开销估计。然后建立函数  $f(n)=g(n)+h(n)$ ，其中  $g(n)$  为从起点到结点  $n$  已经使用的代价，所以  $f(n)$  可以理解为“从起点出发经过结点  $n$  再到终点的代价估计”<sup>[1]</sup>。

A 星算法是一种启发性的算法，即由通过设定评估函数，全面性地对网格的各个节点进行评估。而每个节点就是机器人所到达的位置，对每个位置点都进行智能化评估，找到最好的位置，从而最终找到目标位置<sup>[2]</sup>。

关于  $h(n)$  的构造返回值的问题会对 A\* 算法造成影响：

①如果构造  $h(n) \equiv 0$ ，那么该 A\* 算法就已经退化为了 Dijkstra 算法，一定能解得最优解但是运行效率最低。

②如果构造  $h(n) \equiv h^*(n)$ ，则该 A\* 算法不仅能够保证一定能解得最优解，而且运行效率在所有能保证解得最优解的 A\* 算法中是最高的。其中  $h^*(n)$  表示结点  $n$  到终点的实际代价。

③如果构造的  $h(n)$  对所有的  $n$  有  $h(n) \leq h^*(n)$ ，则该 A\* 算法能保证一定能得到最优解，但是效率略低于上述  $h(n) \equiv h^*(n)$  的情况。

④如果构造  $h(n)$  存在  $n$  使得  $h(n) > h^*(n)$ ，则该 A\* 算法不一定能得到最优解（当然运气好的时候也有可能解得最优解，但是不能保证）。

所以在构造 A\* 算法时，着重关注  $h(n)$  的启发式函数的选择。

启发式函数听起来很有学问，其实可以很简单的理解为从源点到目标的所需要消耗的总代价  $f(n)$ （和适应度函数比较相像），这个总代价可以分成两个部分从源点到中间节点（搜索的中间状态）已经消耗的实际代价  $g(n)$ ，另一个部分就是对从中间节点到目标的预测  $h(n)$ 。

通常来说，这里的代价一般是指各种距离，像欧式距离，曼哈顿距离等等，这个根据你所求

解的实际问题决定。

另一个值得指出的就是预测，预测值直接影响了问题求解的效率以及能否求得合理的解。这里给出一个结论：对于任意预测值  $h(n)$  均小于等于实际值的话，我们可以说最终解就是问题的最优解。

### 2.2.2 算法基本原理

算法基本过程可以简述为：

①设置一个 OPEN 表用于存放那些搜索图上的叶节点，也就是已经被生成出来，但是还没被拓展的节点；初始化 CLOSE 表用于存放图中的非叶节点，也就是不但被生成出来，还已经被拓展的节点。

②OPEN 中的节点按照  $f$  值从小到大排列。每次从 OPEN 表中取出第一个元素  $n$  进行扩展，如果  $n$  是目标节点，则算法找到一个解，算法结束，否则扩展  $n$ 。

③对于  $n$  的子节点  $m$ ，如果  $m$  既不在 OPEN 也不在 CLOSE，则将  $m$  加入 OPEN；如果  $m$  在 CLOSE，说明从初始节点到  $m$  有两条路径，如果新路径耗散值大，什么都不做；如果较小，则删除原来在 CLOSE 的，将新找到的放入 OPEN。

④重复①，直到找到一个解结束；或者 OPEN 为空算法以失败结束，说明无解。

利用 A\*算法根据以上步骤进行求解，便轻松得到在算法层面利用 A\*算法求解八数码问题了。

### 2.2.3 启发式函数算法设计

根据本题的八数码问题，特设计一下四个启发式函数进行比较：

①计算当前状态的不在位棋子数作为  $h(n)$

②将当前状态的各个棋子实际位置与期望位置的欧拉距离和作为  $h(n)$

③将当前状态的各个棋子实际位置与期望位置的曼哈顿距离和作为  $h(n)$

④将当前状态的各个棋子实际位置与期望位置的带系数的曼哈顿和作为  $h(n)$ ：

即此时，将当前和期望的八数码棋盘各自展开成一个数列，比较对应数列，各个棋子的距离，将距离和作为  $h(n)$ 。即相当于将原有的曼哈顿距离带上一定的系数。

## 2.3 模块设计

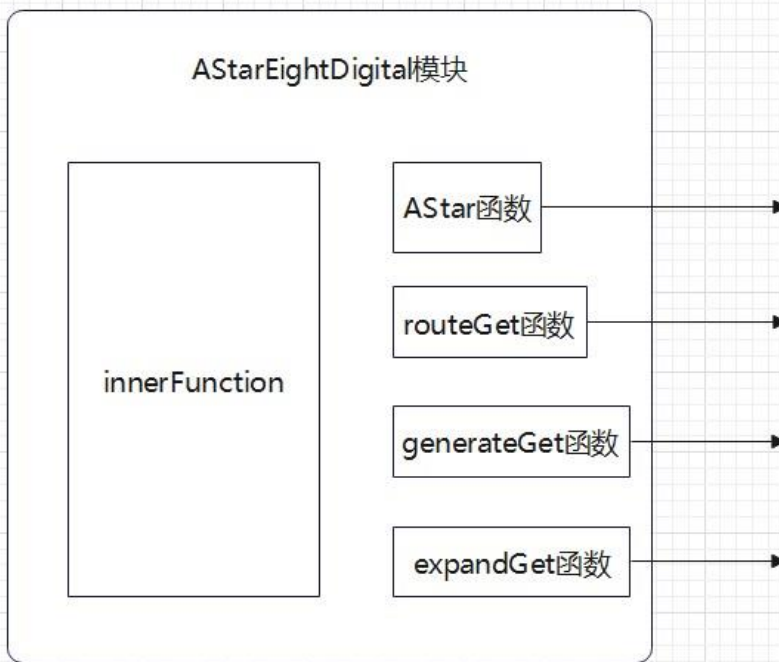
### ①AStarEightDigital 模块

```
class AStarEightDigital {
public:
    char start [BoardSize][BoardSize] = { '\0' }; //起始八数码形式
    char end   [BoardSize][BoardSize] = { '\0' }; //终点八数码形式
    vector<AStarQueNote> OpenStorage; //open数组vector
    vector<AStarQueNote> CloseStorage; //close数组vector
    vector<AStarTreeNote> TreeList; //存储搜索过程中的线索树

public:
    AStarEightDigital(); //构造函数
    int setup(char(*s)[BoardSize], char(*e)[BoardSize]); //初始化
    int DataChangeGraph(unsigned long long& in, char(*arry)[BoardSize]); //unsigned long long 转 char格式化数组数据
    int GraphChangeData(char(*arry)[BoardSize], unsigned long long& in); //unsigned long long 转 unsigned long long
    int setInspire(AStarQueNote& in); //获取启发式函数的值
    int FindOpenMin(int& site);
    bool FindOpen(unsigned long long& in, int& site);
    bool FindClose(unsigned long long& in, int& site);
    int AStarMove(char(*arry)[BoardSize], AStarTreeNote& father);
    int FindFather(AStarQueNote& node, AStarTreeNote& father);
    int AStar(void);

    int routeGet(vector<unsigned long long>& routeResult); //根据求出解路径
    int generateGet(int& get); //返回生成结点数
    int expandGet(int& get); //返回拓展结点数
};
```

该模块的整体模块设计为:



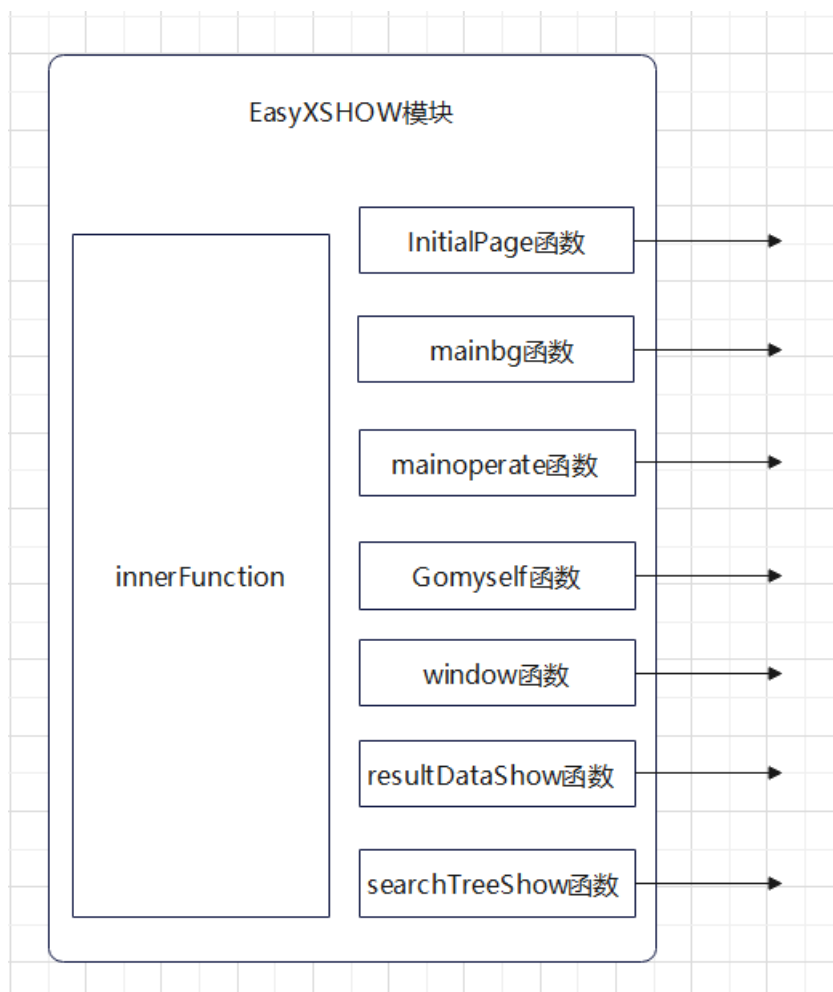
②EasyXSHOW 模块

```
class EasyXSHOW {
public:
    IMAGE num[9];
    AREA General, cherkboard, dataout, stateshow, buttonoperate;
    AREA eightborad, startstate, endstate;
    AREA systemgo, myselfgo;
    AREA upbutton, rightbutton, downbutton, leftbutton;
    AREA reset, initialbegin, initialend;
public:
    EasyXSHOW(); //可视化初始构造函数
    int Initialbg(void);
    int Boardshow(char(*array)[BoardSize]);
    int blockshow(int x, int y, char(*array)[BoardSize]);
    int block(int x, int y, int num);
    int button(int x, int y, int length, int width, TCHAR text[]);
    int stateinitial(char(*st)[BoardSize], char(*end)[BoardSize]);
    int moveinitial(char(*state)[BoardSize], AREA& site);
    int InitialPage(char(*start)[BoardSize], char(*end)[BoardSize]);
    int mainbg(char(*start)[BoardSize], char(*state)[BoardSize], char(*end)[BoardSize]);
    int mainoperate(void);
    int Gomyself(char(*state)[BoardSize], char(*end)[BoardSize]);
    void winshow(void);

    int processShow(vector<unsigned long long>& routeResult);
    int resultDataShow(clock_t time, int expandNum, int generateNum, vector<unsigned long long> routeResult);
    int searchTreeShow(vector<AStarTreeNode>& TreeList, unsigned long long end);
    int searchTreeShowIn(vector<AStarTreeNode>& TreeList, unsigned long long end);
    int searchTreeShowNode(int x, int y, AStarTreeNode& node, unsigned long long end);
    int resultNone(void);
};
```

该模块的整理模块设计为:



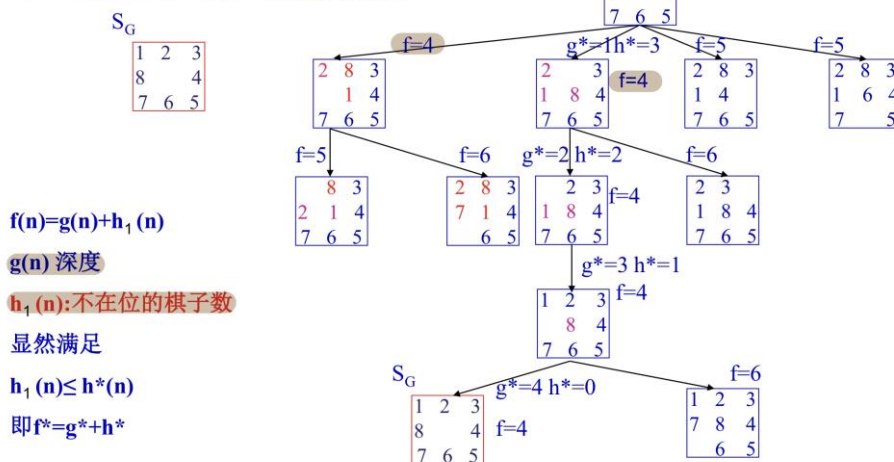


## 2.4 其他创新内容或优化算法

由于本题采用了 OPEN 和 CLOSE 表实现 A\*算法，每次均需要从 OPEN 表中提取  $f(n)$  最小的元素，因此，若每次进行遍历寻找最小值，当程序数据量增大时，程序运行效率将大大降低。因此可以使用优先队列的方式对 OPEN 表进行存储，每次只需要 pop 出最优节点即可。

但是这样也存在一个问题：

## A\*算法应用举例 八数码难题



八数码难题 $h(n)=h_1(n)$ 的搜索树

当 OPEN 表中存在两个相同最小值时，按照逻辑严密性进行分析，应当将两个最小节点均进行遍历。避免出现：随机遍历多个最小值之一，忽略另一个最小值子节点可能出现更小  $f(n)$  的情况。因此，使用优先队列的过程中也应该进行适当的修改。

## 3 实验过程

### 3.1 环境说明

操作系统: Window  
 开发语言: C++  
 编译平台: Visual Studio 2022  
 核心库 : <graphics.h> //EasyX 库  
           <conio.h> //EasyX 库  
           <vector> //使用 vector 数组  
           <queue> //优先队列  
           <windows.h> //sleep 函数  
           <time.h> //计算运行时间

### 3.2 源代码清单

#### 3.2.1 AStar\_head.h

```
/*
 * @author : gonzalez
 * @time : 2022.3.28-2022.4.12
 * @function: 实现 A 星求解八数码问题的头文件
 */

#pragma once
#include <vector> //vector 数组
#include <queue> //队列
#include <graphics.h> //EasyX 库
#include <conio.h> //EasyX 库
#include <windows.h> //sleep 函数
#include <time.h> //记录执行时间
#include <string> //使用 string 字符串
using namespace std; //使用 vector 数组

#define FINE 1 //正常返回
#define FAIL -1 //错误返回
#define BoardSize 3 //八数码数组存储范围

/*条件编译选择启发式函数*/
//#define OUFOFPOSITION //不在位的棋子数
#define COEFFICIENTMANHANTTAN //系数曼哈顿距离法--->可能存在高估的情况,
因此并非可采纳的, 结果可能不是最优解
//#define EULERDISTANCE //欧拉距离法
//#define DISTANCETOTARGET //曼哈顿距离法所有棋子到其目标位置的距离和

/*****
 * @author : */
 * @time : */
 * @function : */
```

```

/*****
struct AStarTreeNode {                                //A*遍历树存储结构体
    unsigned long long FatherNode = 0;
    unsigned long long Treelayout = 0;
    int Treeprice = 0;
    int Treeinspire = 0;
    bool Find = false;                                //是否被遍历
};
struct AStarQueNote{                                  //open/close 表存储结构体
    unsigned long long Quelayout = 0;
    int Queprice = 0;                                  //代价函数值
    int Queinspire = 0;                                //激励函数值
};

/*默认 char 型字符存储八数码方式如下(0 表示空格)
    * * * * *
    * 0 * 1 * 2 *
    * * * * *
    * 3 * 4 * 5 *      --转换--》    012345678
    * * * * *
    * 6 * 7 * 8 *
    * * * * *

*/

class AStarEightDigital {
public:
    char start [BoardSize][BoardSize] = { '\0' };    //起始八数码形式
    char end   [BoardSize][BoardSize] = { '\0' };    //终点八数码形式
    vector<AStarQueNote> OpenStorage;                  //open 数组 vector
    vector<AStarQueNote> CloseStorage;                 //close 数组 vector
    vector<AStarTreeNode> TreeList;                   //存储搜索过程中的线索

    树
public:
    AStarEightDigital();                              //构造函数
    int setup(char(*s)[BoardSize], char(*e)[BoardSize]); //初始化
    int DataChangeGraph(unsigned long long& in , char(*arry)[BoardSize]); //unsigned long long
    转 char 格式化数组数据
    int GraphChangeData(char(*arry)[BoardSize], unsigned long long& in); //unsigned long long
    转 unsigned long long
    int setInspire(AStarQueNote& in);                  //获取启发式
    函数的值
    int FindOpenMin(int& site);
    bool FindOpen(unsigned long long& in, int& site);
    bool FindClose(unsigned long long& in, int& site);
    int AStarMove(char(*arry)[BoardSize], AStarTreeNode& father);
    int FindFather(AStarQueNote& node, AStarTreeNode& father);
    int AStar(void);

    int routeGet(vector<unsigned long long>& routeResult); //根据求出解路
    径
    int generateGet(int& get);                          //返回生成

```

```

结点数
    int expandGet(int& get); //返回拓展
结点数

};

/*****
/*@author    : */
/*@time      : */
/*@function   : */
*****/
struct AREA{ //可视化区域存储
    int left = 0;
    int top = 0;
    int right = 0;
    int bottom = 0;
};
struct TreeShow { //可视化树型打印存储结构体
    int x = 0;
    int y = 0;
    unsigned long long dad = 0;
    unsigned long long lay = 0;
};

#define BKCOLOR    WHITE
#define PICWIDE    120
#define BKLENGTH  800
#define BKWIDTH    600
#define BLOCKSIZE  40
#define BUTTONHIGHT 15
#define SYGO       1
#define MYGO       0
#define SLEEPTIME  1000//单步演示之间的间隔时长
#define WIN        6

class EasyXSHOW {
public:
    IMAGE num[9];
    AREA General, cherkerboard, dataout, stateshow, buttonoperate;
    AREA eightborad, startstate, endstate;
    AREA systemgo, myselfgo;
    AREA upbutton, rightbutton, downbutton, leftbutton;
    AREA reset, initialbegin, initialend;
public:
    EasyXSHOW(); //可视化初始构造函数
    int Initialbg(void);
    int Boardshow(char(*array)[BoardSize]);
    int blockshow(int x, int y, char(*array)[BoardSize]);
    int block(int x, int y, int num);
    int button(int x, int y, int length, int width, TCHAR text[]);
    int stateinitial(char(*st)[BoardSize], char(*end)[BoardSize]);
    int moveinitial(char(*state)[BoardSize], AREA& site);
    int InitialPage(char(*start)[BoardSize], char(*end)[BoardSize]);

```

```
int mainbg(char(*start)[BoardSize], char(*state)[BoardSize], char(*end)[BoardSize]);
int mainoperate(void);
int Gomyself(char(*state)[BoardSize], char(*end)[BoardSize]);
void winshow(void);

int processShow(vector<unsigned long long>& routeResult);
int resultDataShow(clock_t time, int expandNum, int generateNum, vector<unsigned long long>
routeResult);
int searchTreeShow(vector<AStarTreeNode>& TreeList, unsigned long long end);
int searchTreeShowIn(vector<AStarTreeNode>& TreeList, unsigned long long end);
int searchTreeShowNode(int x, int y, AStarTreeNode& node, unsigned long long end);
int resultNone(void);
};

/*****
/* @author : */
/* @time : */
/* @function : */
*****/

int Simple_show(unsigned long long& in);
int arrayclear(char(*array)[BoardSize]);
bool answerExist(char(*arr1)[BoardSize], char(*arr2)[BoardSize]);
int TreePrint(vector<AStarTreeNode>& TreeList);
```

### 3.2.2 AStar\_function.cpp

```
/*
 * @author : gonzalez
 * @time : 2022.3.28-2022.3...
 * @function: 实现类的功能函数部分
 */

#include<iostream>
#include<queue>
#include<math.h>
#include"AStar_head.h"

using namespace std;

/*****
/* @author : */
/* @time : */
/* @function : */
*****/

//@function : AStarEightDigital 中默认构造函数
AStarEightDigital::AStarEightDigital()
{
    ;//暂时均无需构造
}

//@function : AStar 算法中的始末状态构造函数
int AStarEightDigital::setup(char(*s)[BoardSize], char(*e)[BoardSize])
{
```

```

//状态数组初始化
for (int i = 0; i < BoardSize; i++) {
    for (int j = 0; j < BoardSize; j++) {
        start[i][j] = s[i][j];
        end[i][j] = e[i][j];
    }
}

//优先队列初始化
AStarQueNote note;
note.Queprice = 0;
GraphChangeData(start, note.Quelayout);
setInspire(note);
OpenStorage.push_back(note);

//线索树列表初始化
AStarTreeNode root;
root.FatherNote = 0;
root.Find = false;
root.Treeinspire = note.Queinspire;
root.Treeprice = 0;
root.Treelayout = note.Quelayout;
TreeList.push_back(root);

return FINE;
}

//@function : unsigned long long 转换成 char 型可视化数据
int AStarEightDigital::DataChangeGraph(unsigned long long &in, char(*arry)[BoardSize])
{
    unsigned long long data = in;
    char temp;
    //数据转字符
    for(int i= BoardSize-1;i>=0;i--)
        for (int j = BoardSize - 1; j >= 0; j--) {
            temp = '0' + data % 10;
            arry[i][j] = char(temp);
            data = data / 10;
        }
    return FINE;
}

//@function : char 型可视化数据 转换成 unsigned long long
int AStarEightDigital::GraphChangeData(char(* arry)[BoardSize], unsigned long long& in)
{
    unsigned long long data = 0;
    //字符转数据
    for(int i=0;i<BoardSize;i++)
        for (int j = 0; j < BoardSize; j++) {
            data = data * 10;
            data = data + (int)(arry[i][j] - '0');
        }

    in = data;
}

```

```

return FINE;
}

/* !!! 影响算法关键因素的启发式函数 !!! */

//@function : 不在位的棋子数作为启发式函数值
#ifdef OUFOFPOSITION
int AStarEightDigital::setInspire(AStarQueNote& in)
{
    char st[BoardSize][BoardSize];
    DataChangeGraph(in.Quelayout,st);
    int price = 0;
    for (int i = 0; i < BoardSize; i++)
        for (int j = 0; j < BoardSize; j++)
            if (st[i][j] != '0')//仅讨论棋子数
                if (st[i][j] != end[i][j])
                    price++;

    in.Queinspire = price;

    return FINE;
}
#endif // OUFOFPOSITION

//@function : 所有棋子到其目标位置的距离和作为启发式函数值（曼哈顿法）
#ifdef DISTANCETOTARGET
int AStarEightDigital::setInspire(AStarQueNote& in)
{
    char st[BoardSize][BoardSize];
    DataChangeGraph(in.Quelayout, st);
    int price = 0;
    char TempState;
    for(int i=0;i<BoardSize;i++)
        for (int j = 0; j < BoardSize; j++)
            if (st[i][j] != '0') { //仅考虑为棋子的情况
                bool sign = false;
                int ti = 0;
                int tj = 0;
                TempState = st[i][j];
                for (ti = 0; ti < BoardSize; ti++) {
                    for (tj = 0; tj < BoardSize; tj++) {
                        if (end[ti][tj] == TempState)
                            sign = true;
                        if(sign)
                            break;
                    }
                    if (sign)
                        break;
                }
                price += abs(ti - i) + abs(tj - j);
            }
    in.Queinspire = price;
}

```



```

return FINE;

}
#endif // DISTANCETOTARGET

//@function : 所有棋子到其目标位置的直线距离(欧氏距离)作为启发式函数值
#ifdef EULERDISTANCE
int AStarEightDigital::setInspire(AStarQueNote& in)
{
    char st[BoardSize][BoardSize];
    DataChangeGraph(in.Quelayout, st);
    int price = 0;
    char TempState;
    for (int i = 0; i < BoardSize; i++)
        for (int j = 0; j < BoardSize; j++)
            if (st[i][j] != '0') { //仅考虑为棋子的情况
                bool sign = false;
                int ti = 0;
                int tj = 0;
                TempState = st[i][j];
                for (ti = 0; ti < BoardSize; ti++) {
                    for (tj = 0; tj < BoardSize; tj++) {
                        if (end[ti][tj] == TempState)
                            sign = true;
                        if (sign)
                            break;
                    }
                    if (sign)
                        break;
                }
                price += int(sqrt((ti - i) * (ti - i) + (tj - j) * (tj - j))); //欧式距离
            }
    in.Queueinspire = price;
    return FINE;
}
#endif //EULERDISTANCE

//@function : 系数曼哈顿距离法
#ifdef COEFFICIENTMANHANTTAN
int AStarEightDigital::setInspire(AStarQueNote& in)
{
    char st[BoardSize][BoardSize];
    DataChangeGraph(in.Quelayout, st);
    int price = 0;
    vector<int>nowState(10, 0);
    vector<int>endState(10, 0);
    for (int i = 0; i < BoardSize; i++)
        for (int j = 0; j < BoardSize; j++) {
            nowState[i * BoardSize + j] = st[i][j] - '0';
            endState[i * BoardSize + j] = end[i][j] - '0';
        }

    for (int i = 0; i < 9; i++) {
        if (nowState[i] != 0) {

```

```

        for (int j = 0; j < 9; j++) {
            if (nowState[i] == endState[j])
                price += abs(j - i);
        }
    }

    in.Queinspire = price;
    return FINE;
}
#endif//COEFFICIENTMANHANTTAN

//@function : 找到 open 数组中的最小值
int AStarEightDigital::FindOpenMin(int& site)
{
    int Value = OpenStorage[0].Queinspire + OpenStorage[0].Queprice;
    int siteget = 0;
    for (int i = 0; i < (int)OpenStorage.size(); i++)
        if (OpenStorage[i].Queinspire + OpenStorage[i].Queprice <= Value) {
            Value = OpenStorage[i].Queinspire + OpenStorage[i].Queprice;
            siteget = i;
        }
    site = siteget;//site from [0 -- OpenStorage.size()-1]
    return FINE;
}

//@function : 找到返回 true;未找到返回 false
bool AStarEightDigital::FindOpen(unsigned long long& in,int&site)
{
    for (int i = 0; i < (int)OpenStorage.size(); i++)
        if (OpenStorage[i].Quelayout == in) {
            site = i;
            return true;
        }
    site = -1;
    return false;
}

//@function : 找到返回 true;未找到返回 false
bool AStarEightDigital::FindClose(unsigned long long& in, int& site)
{
    for (int i = 0; i < (int)CloseStorage.size(); i++)
        if (CloseStorage[i].Quelayout == in) {
            site = i;
            return true;
        }
    site = -1;
    return false;
}

//@function : 进行八数码 A*算法中的棋子移动
int AStarEightDigital::AStarMove(char(*arry)[BoardSize], AStarTreeNode&father)
{

```

```

//新建初始化
unsigned long long num;
GraphChangeData(array, num);
AStarQueNote quenote;
quenote.Queueprice = father.Treeprice + 1;
quenote.Quelayout = num;
setInspire(quenote); //获取启发式函数值
AStarTreeNode treenote;
treenote.FatherNote = father.Treeprice;
treenote.Treeprice = quenote.Queueprice;
treenote.Treeinspire = quenote.Queueinspire;
treenote.Find = false;

//根据 open/close 表中有无相关数据分别执行内容
int site;
if (FindOpen(num, site)) { //OpenStorage 中存在相同值
    if (quenote.Queueprice + quenote.Queueinspire < OpenStorage[site].Queueprice +
        OpenStorage[site].Queueinspire) {
        OpenStorage[site].Queueprice = quenote.Queueprice; //显示较小的 f(n)值
        OpenStorage[site].Queueinspire = quenote.Queueprice; //显示较小的 f(n)值
        TreeList.push_back(treenote); //送入搜索树
    }
}
else if (FindClose(num, site)) { //CloseStorage 中存在相同值
    if (quenote.Queueprice+quenote.Queueinspire >= CloseStorage[site].Queueprice +
        CloseStorage[site].Queueinspire) {
        ;
    }
    else {
        CloseStorage.erase(CloseStorage.begin() + site); //删除 close 中原结点
        TreeList.push_back(treenote); //进入搜索树中
        OpenStorage.push_back(quenote); //新的较小结点送入 open 队
    }
}
else { //均不存在相同值
    OpenStorage.push_back(quenote);
    TreeList.push_back(treenote);
}

return FINE;
}

//@function: 在搜索树 vector 中找到对应结点
int AStarEightDigital::FindFather(AStarQueNote& node, AStarTreeNode& father)
{
    for (int i = 0; i < (int)TreeList.size(); i++) {
        if (TreeList[i].Treeprice==node.Queueprice) //布局一致
        if (TreeList[i].Treeinspire == node.Queueinspire) { //代价值一致
            if (TreeList[i].Treeinspire == node.Queueinspire) { //启发函数值一致
                TreeList[i].Find = true; //已经被遍历
            }
        }
    }
}

```

```

        father.FatherNode = 0;
        father.Find = true;
        father.TreeLayout = TreeList[i].TreeLayout;
        father.TreeInspire = TreeList[i].TreeInspire;
        father.TreePrice = TreeList[i].TreePrice;
        break;
    }
}

return FINE;
}

//@function : "A*"算法
int AStarEightDigital::AStar(void)
{
    AStarQueNode temp;
    AStarTreeNote FatherNode;
    unsigned long long EndState; //结果状态信息
    GraphChangeData(end, EndState); //结果信息状态转换

    while (true) {
        if (OpenStorage.size() == 0) //未找到解路径
            return FAIL;
        int MinNum = 0;

        //计算最小值的个数
        int site;
        FindOpenMin(site);
        int value = OpenStorage[site].QueInspire + OpenStorage[site].QuePrice;
        for (int i = 0; i < (int)OpenStorage.size(); i++) {
            if (OpenStorage[i].QueInspire + OpenStorage[i].QuePrice <= value)
                MinNum++;
        }

        //开始拓展
        for (int i = 1; i <= MinNum; i++) {

            //open-->close
            int popsite;
            FindOpenMin(popsite);
            temp = OpenStorage[popsite];
            OpenStorage.erase(OpenStorage.begin() + popsite);
            CloseStorage.push_back(temp);
            FindFather(temp, FatherNode); //找到线索树中的该点所对应的点

            if (temp.QueLayout == EndState) //找到结果
                return FINE;

            //begin to extend
            char st[BoardSize][BoardSize];
            char move;
            int ZeroSite[2] = { 0, 0 };
            DataChangeGraph(temp.QueLayout, st);
        }
    }
}

```

```

for(int i= 0;i<BoardSize;i++)                //找到空格位置
    for (int j = 0; j < BoardSize; j++)
        if (st[i][j] == '0') {
            ZeroSite[0] = i;
            ZeroSite[1] = j;
        }

//Up
if (ZeroSite[0] > 0) {
    //move
    move = st[ZeroSite[0] - 1][ZeroSite[1]];
    st[ZeroSite[0] - 1][ZeroSite[1]] = st[ZeroSite[0]][ZeroSite[1]];
    st[ZeroSite[0]][ZeroSite[1]] = move;

    AStarMove(st,                            FatherNode);

//moving

    //move back
    move = st[ZeroSite[0] - 1][ZeroSite[1]];
    st[ZeroSite[0] - 1][ZeroSite[1]] = st[ZeroSite[0]][ZeroSite[1]];
    st[ZeroSite[0]][ZeroSite[1]] = move;
}
//Down
if (ZeroSite[0] < 2) {
    //move
    move = st[ZeroSite[0] + 1][ZeroSite[1]];
    st[ZeroSite[0] + 1][ZeroSite[1]] = st[ZeroSite[0]][ZeroSite[1]];
    st[ZeroSite[0]][ZeroSite[1]] = move;

    AStarMove(st,                            FatherNode);

//moving

    //move back
    move = st[ZeroSite[0] + 1][ZeroSite[1]];
    st[ZeroSite[0] + 1][ZeroSite[1]] = st[ZeroSite[0]][ZeroSite[1]];
    st[ZeroSite[0]][ZeroSite[1]] = move;
}
//Left
if (ZeroSite[1] > 0) {
    //move
    move = st[ZeroSite[0]][ZeroSite[1] - 1];
    st[ZeroSite[0]][ZeroSite[1] - 1] = st[ZeroSite[0]][ZeroSite[1]];
    st[ZeroSite[0]][ZeroSite[1]] = move;

    AStarMove(st,                            FatherNode);

//moving

    //move back
    move = st[ZeroSite[0]][ZeroSite[1] - 1];
    st[ZeroSite[0]][ZeroSite[1] - 1] = st[ZeroSite[0]][ZeroSite[1]];
    st[ZeroSite[0]][ZeroSite[1]] = move;
}
//Right
if (ZeroSite[1] < 2) {

```

```

        //move
        move = st[ZeroSite[0]][ZeroSite[1] + 1];
        st[ZeroSite[0]][ZeroSite[1] + 1] = st[ZeroSite[0]][ZeroSite[1]];
        st[ZeroSite[0]][ZeroSite[1]] = move;

        AStarMove(st,                                     FatherNode);
//moving

        //move back
        move = st[ZeroSite[0]][ZeroSite[1] + 1];
        st[ZeroSite[0]][ZeroSite[1] + 1] = st[ZeroSite[0]][ZeroSite[1]];
        st[ZeroSite[0]][ZeroSite[1]] = move;
    }
}

}

}

//@function : 求出解路径
int AStarEightDigital::routeGet(vector<unsigned long long>& routeResult)
{
    unsigned long long startShow;
    unsigned long long endShow;
    GraphChangeData(start, startShow);
    GraphChangeData(end, endShow);

    unsigned long long lastSite;
    lastSite = endShow;
    while (true) {
        routeResult.push_back(lastSite);
        for (int i = 0; i < (int)TreeList.size(); i++)
            if (TreeList[i].Find) {
                if (TreeList[i].Treelayout == lastSite) {
                    lastSite = TreeList[i].FatherNode;
                    break; //跳出 for 循环
                }
            }
        if (lastSite == startShow) {
            routeResult.push_back(lastSite);
            break; //跳出 while 循环
        }
    }

    return FINE;
}

//@function : 求出生成结点数
int AStarEightDigital::generateGet(int& get)
{
    get = TreeList.size();
    return FINE;
}

//@function : 求出拓展结点数

```

```

int AStarEightDigital::expandGet(int& get)
{
    int num = 0;
    for (int i = 0; i < (int)TreeList.size(); i++){
        if (TreeList[i].Find)
            num++;
    }
    get = num - 1;//终止结点不算拓展结点
    return FINE;
}

/*****
/*@author      : */
/*@time        : */
/*@function    : */
*****/
//@function:显示一个八数码问题界面
int Simple_show(unsigned long long& in)
{
    unsigned long long divied = 100000000;
    unsigned long long temp = 0;
    for (int i = 0; i < BoardSize; i++) {
        for (int j = 0; j < BoardSize; j++) {
            temp = in / divied;
            if (temp)
                cout << temp << ' ';
            else
                cout << "  ";
            in = in - in / divied * divied;
            divied = divied / 10;
        }
        cout << endl;
    }
    return 0;
}

//@function : 清除初始化数组标记
int arrayclear(char(*array)[BoardSize])
{
    for (int i = 0; i < BoardSize; i++)
        for (int j = 0; j < BoardSize; j++)
            array[i][j] = '*';
    return FINE;
}

//@function : 判断是否存在结果-->start 和 end 的逆序数不变
bool answerExist(char(*arr1)[BoardSize], char(*arr2)[BoardSize])
{
    vector<int>list1;
    vector<int>list2;

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

```

```

        for (int j = 0; j < BoardSize; j++) {
            list1.push_back(arr1[i][j] - '0');
            list2.push_back(arr2[i][j] - '0');
        }
int num1 = 0;
int num2 = 0;
int Temp = 0;
for (int i = 0; i < (int)list1.size(); i++) {
    if (list1[i] != 0) { //空格不检查
        Temp = list1[i];
        for (int j = 0; j < i; j++) {
            if (list1[j] > Temp)
                num1++;
        }
    }
}
for (int i = 0; i < (int)list2.size(); i++) {
    if (list2[i] != 0) { //空格不检查
        Temp = list2[i];
        for (int j = 0; j < i; j++) {
            if (list2[j] > Temp)
                num2++;
        }
    }
}

if (num1 % 2 == num2 % 2)        //奇偶性一致
    return true;
else                             //奇偶性不同
    return false;
}

//@function : cmd 显示结果树（如遇可视化界面不足的情况）
int TreePrint(vector<AStarTreeNode>& TreeList)
{
    cout << "搜索树为为: " << endl;
    cout << "*****" << endl;
    for (int i = 0; i < (int)TreeList.size(); i++) {
        cout << "父亲结点为 : " << TreeList[i].FatherNode << endl;

        cout << "是否被拓展 : " << boolalpha << TreeList[i].Find << endl;
        cout << "代价函数值 : " << TreeList[i].Treeprice << endl;
        cout << "激励函数值 : " << TreeList[i].Treeinspire << endl;
        cout << "该结点信息 : " << endl;
        Simple_show(TreeList[i].Treelayout);
        cout << "-----" << endl;
    }

    return FINE;
}

```



### 3.2.3 AStar\_show.cpp

```

/*
 * @author : gonzalez
 * @time : 2022.3.28-2022.3...
 * @function: 实现界面的可视化功能
 */

#include <iostream>
#include "AStar_head.h"

using namespace std;

/*****
/* @author : */
/* @time : */
/* @function : */
*****/

// @function: 打印一个字符框
int EasyXSHOW::block(int x, int y, int num)
{
    int length = BLOCKSIZE;
    int HIGHT = BLOCKSIZE - 10;
    setbkmode(TRANSPARENT);
    setfillcolor(LIGHTGRAY);
    fillroundrect(x, y, x + length, y + length, 10, 10);
    int mx = x + 2;
    int my = y + 2;
    length = length - 4;
    fillroundrect(mx, my, mx + length, my + length, 10, 10);
    settextrstyle(HIGHT, 0, _T("黑体")); //设置字体
    TCHAR text[5] = _T(" ");
    if (num != 0)
        _stprintf_s(text, _T("%d"), num);
    int tx = mx + (length - textwidth(text)) / 2;
    int ty = my + (length - textheight(text)) / 2;
    outtextxy(tx, ty, text);

    return FINE;
}

// @function: 打印字符格子
int EasyXSHOW::blockshow(int x, int y, char(*array)[BoardSize])
{
    int temp_x = x;
    int temp_y = y;
    for (int i = 0; i < BoardSize; i++) {
        for (int j = 0; j < BoardSize; j++) {
            if (array[i][j] != '*') { //处于*时 不显示
                temp_x = x + j * BLOCKSIZE;
                temp_y = y + i * BLOCKSIZE;
                block(temp_x, temp_y, (int)(array[i][j] - '0'));
            }
        }
    }
}

```

```

    }
}
return FINE;
}

// @function: EasySHOE 的预处理函数
EasyXSHOW::EasyXSHOW()
{
    /*照片预处理*/
    loadimage(&num[1], _T("./image/1.jpg"), PICWIDE, PICWIDE, false);
    loadimage(&num[2], _T("./image/2.jpg"), PICWIDE, PICWIDE, false);
    loadimage(&num[3], _T("./image/3.jpg"), PICWIDE, PICWIDE, false);
    loadimage(&num[4], _T("./image/4.jpg"), PICWIDE, PICWIDE, false);
    loadimage(&num[5], _T("./image/5.jpg"), PICWIDE, PICWIDE, false);
    loadimage(&num[6], _T("./image/6.jpg"), PICWIDE, PICWIDE, false);
    loadimage(&num[7], _T("./image/7.jpg"), PICWIDE, PICWIDE, false);
    loadimage(&num[8], _T("./image/8.jpg"), PICWIDE, PICWIDE, false);
    loadimage(&num[0], _T("./image/0.jpg"), PICWIDE, PICWIDE, false);
    //总界面位置信息
    General.left = 0;
    General.right = BKLENGTH;
    General.top = 0;
    General.bottom = BKWIDTH;
    //棋盘左边位置信息
    cherkerboard.left = 0;
    cherkerboard.right = 360;
    cherkerboard.top = 0;
    cherkerboard.bottom = 420;
    //右侧数据框位置信息
    dataout.left = 360;
    dataout.right = 800;
    dataout.top = 0;
    dataout.bottom = 420;
    //右下侧起始中止状态位置信息
    stateshow.left = 360;
    stateshow.right = 800;
    stateshow.top = 420;
    stateshow.bottom = 600;
    //按键区域
    buttonoperate.left = 0;
    buttonoperate.right = 360;
    buttonoperate.top = 420;
    buttonoperate.bottom = 600;
    //八数码棋盘位置信息
    eightborad.left = 0;
    eightborad.right = 360;
    eightborad.top = 30;
    eightborad.bottom = 390;
    //左下角起始状态位置信息
    startstate.left = stateshow.left + 10 + 60;
    startstate.right = startstate.left + 120;
    startstate.top = stateshow.top + 15;
}

```

```

startstate.bottom = startstate.top + 120;
//左下角终止状态位置信息
endstate.left = startstate.right + 60;
endstate.right = endstate.left + 120;
endstate.top = stateshow.top + 15;
endstate.bottom = endstate.top + 120;

//各个按钮位置信息
//系统执行
systemgo.left = 50;
systemgo.right = 50 + 80;
systemgo.top = 420 + 40;
systemgo.bottom = 420 + 40 + 30;
//自己执行
myselfgo.left = 50;
myselfgo.right = 50 + 80;
myselfgo.top = 600 - 70;
myselfgo.bottom = 600 - 40;
//上行
upbutton.left = 180 + 15 + 50;
upbutton.right = upbutton.left + 50;
upbutton.top = 420 + 15;
upbutton.bottom = upbutton.top + 50;
//下行
downbutton.left = 180 + 15 + 50;
downbutton.right = downbutton.left + 50;
downbutton.top = 420 + 15 + 100;
downbutton.bottom = downbutton.top + 50;
//左行
leftbutton.left = 180 + 15;
leftbutton.right = leftbutton.left + 50;
leftbutton.top = 420 + 15 + 50;
leftbutton.bottom = leftbutton.top + 50;
//右行
rightbutton.left = 180 + 15 + 100;
rightbutton.right = rightbutton.left + 50;
rightbutton.top = 420 + 15 + 50;
rightbutton.bottom = rightbutton.top + 50;
//重置按键
reset.left = 0;
reset.right = 40;
reset.top = 580;
reset.bottom = 600;
//起始初始化
initialbegin.left = 130;
initialbegin.right = 130 + 100;
initialbegin.top = 420 + 30;
initialbegin.bottom = initialbegin.top + 45;
//终止初始化
initialend.left = 130;
initialend.right = 130 + 100;
initialend.top = 600 - 75;
initialend.bottom = initialend.top + 45;

```

```

}

// @function: 起始状态背景显示
int EasyXSHOW::Initialbg(void)
{
    int mid = 5;
    setbkcolor(BKCOLOR);

    setfillcolor(RGB(128, 64, 0));
    fillroundrect(cherkerboard.left, cherkerboard.top, cherkerboard.right, cherkerboard.bottom, 10,
10);
    fillroundrect(cherkerboard.left, cherkerboard.top + mid, cherkerboard.right, cherkerboard.bottom -
mid, 10, 10);

    setfillcolor(BLACK);
    fillrectangle(dataout.left, dataout.top, dataout.right, dataout.bottom);

    setfillcolor(RGB(123, 123, 123));
    fillrectangle(stateshow.left, stateshow.top, stateshow.right, stateshow.bottom);
    fillrectangle(stateshow.left + mid, stateshow.top + mid, stateshow.right - mid, stateshow.bottom -
mid);

    settextstyle(20, 0, _T("黑体"));                                //设置字体
    setbkmode(TRANSPARENT);
    settextcolor(WHITE);
    outtextxy(startstate.left+ 20, startstate.bottom+10, _T("起始状态"));
    outtextxy(endstate.left + 20, endstate.bottom + 10, _T("终止状态"));

    TCHAR text1[] = _T("重置");
    button(reset.left, reset.top, reset.right- reset.left,reset.bottom- reset.top, text1);
    TCHAR text2[] = _T("起始初始化");
    button(initialbegin.left, initialbegin.top, initialbegin.right- initialbegin.left, initialbegin.bottom-
initialbegin.top, text2);
    TCHAR text3[] = _T("终止初始化");
    button(initialend.left, initialend.top, initialend.right - initialend.left, initialend.bottom -
initialend.top, text3);

    return FINE;
}

// @function: 在 eightborad 区域打印图片情况
int EasyXSHOW::Boardshow(char(*array)[BoardSize])
{
    int tx, ty;
    int size = PICWIDE;
    int get = 0;
    for(int i=0;i<BoardSize;i++)
        for (int j = 0; j < BoardSize; j++) {
            tx = eightborad.left + j * size;
            ty = eightborad.top + i * size;
            if (array[i][j] != '*') {
                get = array[i][j] - '0';
            }
        }
}

```

```

        putimage(tx, ty, &num[get]);
    }
    else {
        setfillcolor(RGB(128, 64, 0));
        fillroundrect(tx, ty, tx + PICWIDE, ty + PICWIDE, 10, 10);
    }
}
return FINE;
}

// @function: 按钮输出函数
int EasyXSHOW::button(int x, int y, int length, int width, TCHAR text[])
{
    int HIGHT = BUTTONHIGHT;
    setbkmode(TRANSPARENT);
    setfillcolor(BLUE);
    fillroundrect(x, y, x + length, y + width, 10, 10);
    settextrstyle(HIGHT, 0, _T("黑体"));
    int tx = x + (length - textwidth(text)) / 2;
    int ty = y + (width - textheight(text)) / 2;
    outtextxy(tx, ty, text);

    return FINE;
}

// @function: 移动界面实现
int EasyXSHOW::moveinitial(char(*state)[BoardSize], AREA& site)
{
    ExMessage msg;
    char original[BoardSize][BoardSize] = { {'1','2','3'},{'4','5','6'},{'7','8','0'} };
    Boardshow(original);
    blockshow(site.left, site.top, state);

    int press = 0;
    int goal = 0;
    int tx = 0, ty = 0;
    while (press < 9) {
        while (1) {
            if (peekmessage(&msg, EM_MOUSE)) {
                if (msg.message == WM_LBUTTONDOWN) {
                    if (msg.x >= reset.left && msg.x <= reset.right && msg.y >= reset.top &&
msg.y <= reset.bottom) {
                        return FAIL;
                    }
                    int tx = (msg.y - eightborad.top) / 120;
                    int ty = (msg.x - eightborad.left) / 120;
                    if (tx >= 0 && tx < BoardSize && ty >= 0 && ty < BoardSize) {
                        if (original[tx][ty] != '*') {
                            state[goal / 3][goal % 3] = original[tx][ty];
                            original[tx][ty] = '*';
                            press++;
                            goal++;
                            Boardshow(original);
                            blockshow(site.left, site.top, state);
                        }
                    }
                }
            }
        }
    }
}

```

```

        break;//跳出循环
    }
}
}
}
}
return FINE;
}

// @function: 初始化状态选择
int EasyXSHOW::stateinitial(char(*start)[BoardSize], char(*end)[BoardSize])
{
    bool start_flag = false;
    bool end_flag = false;
    ExMessage msg;
    while (!start_flag||!end_flag) {
        if (peekmessage(&msg, EM_MOUSE)) {
            if (msg.message == WM_LBUTTONDOWN) {
                if (msg.x >= reset.left && msg.x <= reset.right && msg.y >= reset.top && msg.y
<= reset.bottom) {
                    return FAIL;
                }
                else if (msg.x >= initialbegin.left && msg.x <= initialbegin.right && msg.y >=
initialbegin.top && msg.y <= initialbegin.bottom) { //起始状态
                    if (!start_flag) {
                        setfillcolor(BKCOLOR);
                        fillrectangle(initialbegin.left,      initialbegin.top,      initialbegin.right,
initialbegin.bottom);

                        if (moveinitial(start, startstate) == FAIL)//reset
                            return FAIL;
                        start_flag = true;
                    }
                }
                else if (msg.x >= initialend.left && msg.x <= initialend.right && msg.y >=
initialend.top && msg.y <= initialend.bottom) { //结束状态
                    if (!end_flag) {
                        setfillcolor(BKCOLOR);
                        fillrectangle(initialend.left,      initialend.top,      initialend.right,
initialend.bottom);

                        if (moveinitial(end, endstate) == FAIL)//reset
                            return FAIL;
                        end_flag = true;
                    }
                }
            }
        }
    }
    return FINE;
}

// @function: 初始化总界面
int EasyXSHOW::InitialPage(char(*start)[BoardSize], char(*end)[BoardSize])
{

```

```

cleardevice();
setbkcolor(BKCOLOR);
Initialbg();
while (1) {
    if (stateinitial(start, end) == FINE)
        return FINE;
    else {
        cleardevice();
        setbkcolor(BKCOLOR);
        arrayclear(start);
        arrayclear(end);
        Initialbg();
    }
}
return FAIL;
}

// @function:
int EasyXSHOW::mainbg(char(*start)[BoardSize], char(*state)[BoardSize], char(*end)[BoardSize])
{
    int mid = 5;
    setbkcolor(BKCOLOR);

    setfillcolor(RGB(128, 64, 0));
    fillroundrect(cherkerboard.left, cherkerboard.top, cherkerboard.right, cherkerboard.bottom, 10,
10);
    fillroundrect(cherkerboard.left, cherkerboard.top + mid, cherkerboard.right, cherkerboard.bottom -
mid, 10, 10);

    setfillcolor(BLACK);
    fillrectangle(dataout.left, dataout.top, dataout.right, dataout.bottom);

    setfillcolor(RGB(123, 123, 123));
    fillrectangle(stateshow.left, stateshow.top, stateshow.right, stateshow.bottom);
    fillrectangle(stateshow.left + mid, stateshow.top + mid, stateshow.right - mid, stateshow.bottom -
mid);

    settextstyle(20, 0, _T("黑体"));
    setbkmode(TRANSPARENT);
    settextcolor(WHITE);
    outtextxy(startstate.left + 20, startstate.bottom + 10, _T("起始状态"));
    outtextxy(endstate.left + 20, endstate.bottom + 10, _T("终止状态"));

    TCHAR text1[] = _T("重置");
    button(reset.left, reset.top, reset.right - reset.left, reset.bottom - reset.top, text1);
    TCHAR text2[] = _T("系统执行");
    button(systemgo.left, systemgo.top, systemgo.right - systemgo.left, systemgo.bottom -
systemgo.top, text2);
    TCHAR text3[] = _T("自己执行");
    button(myselfgo.left, myselfgo.top, myselfgo.right - myselfgo.left, myselfgo.bottom -
myselfgo.top, text3);
    TCHAR text4[] = _T("↑");
    button(upbutton.left, upbutton.top, upbutton.right - upbutton.left, upbutton.bottom - upbutton.top,
text4);
}

```

```

TCHAR text5[] = _T("↓");
button(downbutton.left, downbutton.top, downbutton.right - downbutton.left, downbutton.bottom -
downbutton.top, text5);
TCHAR text6[] = _T("→");
button(rightbutton.left, rightbutton.top, rightbutton.right - rightbutton.left, rightbutton.bottom -
rightbutton.top, text6);
TCHAR text7[] = _T("←");
button(leftbutton.left, leftbutton.top, leftbutton.right - leftbutton.left, leftbutton.bottom -
leftbutton.top, text7);

for (int i = 0; i < BoardSize; i++)
    for (int j = 0; j < BoardSize; j++)
        state[i][j] = start[i][j];
Boardshow(state);
blockshow(startstate.left, startstate.top, start);
blockshow(endstate.left, endstate.top, end);

return FINE;
}

//return 1 --> 系统执行
//return 0 --> 自己执行
//return -1 --> reset
int EasyXSHOW::mainoperate(void)
{
    ExMessage msg;
    while (true) {
        if (peekmessage(&msg, EM_MOUSE)) {
            if (msg.message == WM_LBUTTONDOWN) {
                if (msg.x >= reset.left && msg.x <= reset.right && msg.y >= reset.top && msg.y
<= reset.bottom) { //reset
                    return FAIL;
                }
                else if (msg.x >= systemgo.left && msg.x <= systemgo.right && msg.y >=
systemgo.top && msg.y <= systemgo.bottom) { //系统执行
                    setfillcolor(BKCOLOR);
                    fillrectangle(systemgo.left, systemgo.top, systemgo.right, systemgo.bottom);
                    return SYGO;
                }
                else if (msg.x >= myselfgo.left && msg.x <= myselfgo.right && msg.y >=
myselfgo.top && msg.y <= myselfgo.bottom) { //自己执行
                    setfillcolor(BKCOLOR);
                    fillrectangle(myselfgo.left, myselfgo.top, myselfgo.right, myselfgo.bottom);
                    return MYGO;
                }
            }
        }
    }
}

// @function: 自己执行游戏界面
int EasyXSHOW::Gomyself(char(*state)[BoardSize], char(*end)[BoardSize])

```



```

{
    ExMessage msg;
    int x, y;
    char move;
    while (true) {
        if (peekmessage(&msg, EM_MOUSE)) {
            if (msg.message == WM_LBUTTONDOWN) {
                bool win = true;
                if (msg.x >= reset.left && msg.x <= reset.right && msg.y >= reset.top && msg.y
<= reset.bottom) { //reset
                    return FAIL;
                }
                else if (msg.x >= systemgo.left && msg.x <= systemgo.right && msg.y >=
systemgo.top && msg.y <= systemgo.bottom) { //系统执行
                    setfillcolor(BKCOLOR);
                    fillrectangle(systemgo.left, systemgo.top, systemgo.right, systemgo.bottom);
                    return SYGO;
                }
                else {
                    //找到空格所在位置
                    for(int i=0;i<BoardSize;i++)
                        for(int j=0;j<BoardSize;j++)
                            if (state[i][j] == '0') {
                                x = i;
                                y = j;
                            }
                    if (msg.x >= upbutton.left && msg.x <= upbutton.right && msg.y >=
upbutton.top && msg.y <= upbutton.bottom) { //向上
                        if (x < 2) {
                            move = state[x + 1][y];
                            state[x + 1][y] = state[x][y];
                            state[x][y] = move;
                            Boardshow(state);
                        }
                    }
                    else if (msg.x >= rightbutton.left && msg.x <= rightbutton.right && msg.y >=
rightbutton.top && msg.y <= rightbutton.bottom) { //向右
                        if (y > 0) {
                            move = state[x][y - 1];
                            state[x][y - 1] = state[x][y];
                            state[x][y] = move;
                            Boardshow(state);
                        }
                    }
                    else if (msg.x >= downbutton.left && msg.x <= downbutton.right &&
msg.y >= downbutton.top && msg.y <= downbutton.bottom) { //向下
                        if (x > 0) {
                            move = state[x - 1][y];
                            state[x - 1][y] = state[x][y];
                            state[x][y] = move;
                            Boardshow(state);
                        }
                    }
                    else if (msg.x >= leftbutton.left && msg.x <= leftbutton.right && msg.y >=

```

```

leftbutton.top && msg.y <= leftbutton.bottom) { //向左
    if (y < 2) {
        move = state[x][y + 1];
        state[x][y + 1] = state[x][y];
        state[x][y] = move;
        Boardshow(state);
    }
}

for (int i = 0; i < BoardSize; i++)
    for (int j = 0; j < BoardSize; j++)
        if (state[i][j] != end[i][j])
            win = false;

if (win)
    break; //若到达成果状态-->跳出循环
}
}

return WIN;
//成功走到
}

// @function: 自己执行成功界面打印
void EasyXSHOW::winshow(void)
{
    TCHAR text[] = _T("成功");
    button(360, 285-50, 80, 30, text);
    TCHAR text1[] = _T("按下 重置 以重启");
    button(300, 350-50, 200, 30, text1);
    ExMessage msg;
    while (true) {
        if (peekmessage(&msg, EM_MOUSE)) {
            if (msg.message == WM_LBUTTONDOWN) {
                if (msg.x >= reset.left && msg.x <= reset.right && msg.y >= reset.top && msg.y
<= reset.bottom) { //reset
                    return;
                }
            }
        }
    }
}

// @function: 展现游戏过程
int EasyXSHOW::processShow(vector<unsigned long long> & routeResult)
{
    int num = routeResult.size();
    char array[BoardSize][BoardSize] = { {'*', '*', '*'}, {'*', '*', '*'}, {'*', '*', '*'} };
    AStarEightDigital AT;
    for (int i = num - 1; i >= 0; i--) {
        AT.DataChangeGraph(routeResult[i], array);
        Boardshow(array);
        Sleep(SLEEPTIME);
    }
}

```

```

    }
    return FINE;
}

// @function:显示游戏数据
int EasyXSHOW::resultDataShow(clock_t time, int expandNum, int generateNum,vector<unsigned long long> routeResult)
{
    const int HIGHT = 18;
    const int BUFFER = 30;
    setbkmode(TRANSPARENT);
    settextrcolor(LIGHTCYAN);
    settextrstyle(HIGHT, 0, _T("黑体")); //设置字体

#ifdef OUFOFPOSITION
    TCHAR s1[] = _T("使用方法为      : 不在位棋子数");
#endif
#ifdef DISTANCETOTARGET
    TCHAR s1[] = _T("使用方法为      : 目标距离之和");
#endif
#ifdef EULERDISTANCE
    TCHAR s1[] = _T("使用方法为      : 欧拉距离法");
#endif
#ifdef COEFFICIENTMANHANTTAN
    TCHAR s1[] = _T("使用方法为      : 系数曼哈顿法");
#endif

    TCHAR s2[BUFFER];
    _sprintf_s(s2, _T("运行时间为      : %d ms"), time);
    TCHAR s3[BUFFER];
    _sprintf_s(s3, _T("生成结点数      : %d"), generateNum);
    TCHAR s4[BUFFER];
    _sprintf_s(s4, _T("拓展结点数      : %d"), expandNum);
    TCHAR s5[BUFFER];
    _sprintf_s(s5, _T("解路径步数      : %d"), routeResult.size() - 1);

    int leftInterval = 10;
    int topInterval = 10;

    outtextxy(dataout.left + leftInterval, dataout.top + topInterval, s1);
    outtextxy(dataout.left + leftInterval, dataout.top + topInterval + HIGHT * 1, s2);
    outtextxy(dataout.left + leftInterval, dataout.top + topInterval + HIGHT * 2, s3);
    outtextxy(dataout.left + leftInterval, dataout.top + topInterval + HIGHT * 3, s4);
    outtextxy(dataout.left + leftInterval, dataout.top + topInterval + HIGHT * 4, s5);

    TCHAR s6[]=_T("左侧开始执行解路径步骤...");
    outtextxy(dataout.left + leftInterval, dataout.top + topInterval + HIGHT * 6, s6);

    return 0;
}

// @function:搜索树显示页面
int EasyXSHOW::searchTreeShow(vector<AStarTreeNode>& TreeList,unsigned long long end)

```

```
{
    //显示按钮
    int length = 100;
    int width = 40;
    int x = dataout.right - length - 10;
    int y = dataout.bottom - width - 10;
    TCHAR text[] = _T("显示搜索树");
    button(x, y, length, width, text);

    ExMessage msg;
    while (true) {
        if (peekmessage(&msg, EM_MOUSE)) {
            if (msg.message == WM_LBUTTONDOWN) {
                if (msg.x >= reset.left && msg.x <= reset.right && msg.y >= reset.top && msg.y
<= reset.bottom) {//重新开始
                    return FAIL;
                }
                else if (msg.x >= x && msg.x <= x + length && msg.y >= y && msg.y <= y +
width)
                    break;
            }
        }
    }

    //显示搜索树...
    searchTreeShowIn(TreeList, end);

    while (true) {
        if (peekmessage(&msg, EM_MOUSE)) {
            if (msg.message == WM_LBUTTONDOWN) {
                if (msg.x >= reset.left && msg.x <= reset.right && msg.y >= reset.top && msg.y
<= reset.bottom) {//重新开始
                    return FAIL;
                }
            }
        }
    }

    return FINE;
}

// @function:搜索树显示页面
int EasyXSHOW::searchTreeShowIn(vector<AStarTreeNode>& TreeList, unsigned long long end)
{
    setfillcolor(BLACK);
    fillrectangle(cherkerboard.left, cherkerboard.top, stateshow.right, stateshow.bottom);
    TCHAR text1[] = _T("重置");
    button(reset.left, reset.top, reset.right - reset.left, reset.bottom - reset.top, text1);

    //...依次选择区域打印...
    int len = TreeList.size();
    int blockHigh = 60;
    int blockWide = 45;
```

```

int blockMid = 30;
int bottom = dataout.bottom;
int right = dataout.right;

//构建打印树的数据
vector<TreeShow>Tree;
int maxLayer = 0;
for (int i = 0; i < len; i++)
    if (maxLayer < TreeList[i].Treeprice)
        maxLayer = TreeList[i].Treeprice;
int numLayer = 0;
int interval = 0;
int site = 0;
for (int i = 0; i <= maxLayer; i++) { //每层遍历
    numLayer = 0;
    site = 0;
    for(int j=0;j<len;j++){
        if (TreeList[j].Treeprice == i)
            numLayer++;
    }
    interval = right / (numLayer + 1); //每层间隔确定
    for (int j = 0; j < len; j++) {
        if (TreeList[j].Treeprice == i) {
            site += interval;
            TreeShow temp;
            temp.x = site;
            temp.y = i * (blockHigh + blockMid);
            temp.dad = TreeList[j].FatherNote;
            temp.lay = TreeList[j].Treelayout;
            Tree.push_back(temp);
            searchTreeShowNode(temp.x, temp.y, TreeList[j], end); //打印结点
        }
    }
}
//打印连线
int x1 = 0, y1 = 0, x2 = 0, y2 = 0;
int half = 23;
unsigned long long father = 0;
for (int i = 0; i < (int)Tree.size(); i++) {
    if (Tree[i].dad != 0) {
        father = Tree[i].dad;
        x1 = Tree[i].x;
        y1 = Tree[i].y;
        for (int j = 0; j < (int)Tree.size(); j++) {
            if (Tree[j].lay == father) {
                x2 = Tree[j].x;
                y2 = Tree[j].y;
                break;
            }
        }
        line(x1 + half, y1, x2 + half, y2 + 60);
    }
}
}

```

```

settextcolor(LIGHTCYAN);
settextstyle(15, 0, _T("黑体")); //设置字体
TCHAR notice1[25] = _T("红色表示拓展结点");
outtextxy(650, 10, notice1); //显示提示信息
TCHAR notice2[25] = _T("黄色表示目标结点");
outtextxy(650, 25, notice2); //显示提示信息
return FINE;
}

// @function:搜索树显示一个搜索情况
int EasyXSHOW::searchTreeShowNode(int x, int y, AStarTreeNode& node, unsigned long long end)
{
    char array[BoardSize][BoardSize] = { 0 };
    AStarEightDigital Astar;
    Astar.DataChangeGraph(node.TreeLayout, array);

    int length = 15;
    int HIGHT = 15;
    int tempX = x;
    int tempY = y;
    setbkmode(TRANSPARENT);
    settextstyle(HIGHT, 0, _T("黑体")); //设置字体
    setfillcolor(LIGHTGRAY);
    if(node.TreeLayout==end)
        settextcolor(YELLOW); //终止结点展示
    else {
        if (node.Find)
            settextcolor(RED); //拓展结点展示
        else
            settextcolor(BLACK); //生成结点展示
    }
    //显示状态
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            tempX = x + j * length;
            tempY = y + i * length;
            fillrectangle(tempX, tempY, tempX + length, tempY + length);
            TCHAR text[5] = _T(" ");
            if (array[i][j] != '0')
                _sprintf_s(text, _T("%d"), array[i][j] - '0');
            int tx = tempX + (length - textwidth(text)) / 2;
            int ty = tempY + (length - textheight(text)) / 2;
            outtextxy(tx, ty, text);
        }
    }

    //显示 f(n)
    settextstyle(10, 0, _T("黑体")); //设置字体
    TCHAR text[25] = _T(" ");
    _sprintf_s(text, _T("f(n)=%d"), node.Treeinspire + node.Treeprice);
    tempX = x;
    tempY = y + 3 * length;
    fillrectangle(tempX, tempY, tempX + length * 3, tempY + length * 1);

```

```

outtextxy(tempX, tempY + (length - textheight(text)) / 2, text);

return FINE;

}

// @function:不存在解的情况提示
int EasyXSHOW::resultNone(void)
{
    TCHAR text[] = _T("此时无解 点击重置重新开始");
    button(290, 270, 220, 60, text);
    ExMessage msg;
    while (true) {
        if (peekmessage(&msg, EM_MOUSE)) {
            if (msg.message == WM_LBUTTONDOWN) {
                if (msg.x >= reset.left && msg.x <= reset.right && msg.y >= reset.top && msg.y
<= reset.bottom) {
                    //reset
                    return FINE;
                }
            }
        }
    }
}

```

### 3.2.4 AStar\_main.cpp

```

/*
 * @author   : gonzalez
 * @time     : 2022.3.28-2022.3...
 * @function: 实现A星求解八数码问题的主函数
 */

#include<iostream>
#include"AStar_head.h"
using namespace std;

/*int main()
{
    AStarEightDigital Astar;
    //*****
    /*  2 8 3
    /*  1  4
    /*  7 6 5
    //*****
    char ST[BoardSize][BoardSize] = { { '2','8','3'},{ '1','0','4'},{ '7','6','5' } };
    //char ST[BoardSize][BoardSize] = { { '2','8','3'},{ '1','6','4'},{ '7','0','5' } };
    //char ST[BoardSize][BoardSize] = { { '0','1','2'},{ '3','4','5'},{ '6','7','8' } };

    //*****
    /*  1 2 3
    /*  4  5
    /*  6 7 8
    //*****

```

```

char EN[BoardSize][BoardSize] = { {'1','2','3'},{'8','0','4'},{'7','6','5'} };
//char EN[BoardSize][BoardSize] = { {'1','2','3'},{'8','0','4'},{'7','6','5'} };
//char EN[BoardSize][BoardSize] = { {'8','7','6'},{'5','4','3'},{'2','0','1'} };

Astar.setup(ST, EN);
Astar.AStar();

//show the result
for (int i = 0; i < (int)Astar.CloseStorage.size(); i++) {
    cout << Astar.CloseStorage[i].Quelayout << ' ';
    cout << Astar.CloseStorage[i].Queprice << ' ';
    cout << Astar.CloseStorage[i].Queinspire << endl;
}
cout << "Close数组: " << Astar.CloseStorage.size() << endl;
cout << "*****" << endl;
for (int i = 0; i < (int)Astar.TreeList.size(); i++) {
    cout << Astar.TreeList[i].FatherNote << '-';
    cout << boolalpha << Astar.TreeList[i].Find << '-';
    cout << Astar.TreeList[i].Treeprice << '-';
    cout << Astar.TreeList[i].Treeinspire << endl;
    Simple_show(Astar.TreeList[i].Treelayout);
}
cout << "Tree数组: " << Astar.TreeList.size() << endl;

return 0;
}*/

int main()
{
    initgraph(BKLENGTH, BKWIDTH, EW_SHOWCONSOLE);
    setbkcolor(BKCOLOR);
    char start[BoardSize][BoardSize] = { {'*','*','*'},{'*','*','*'},{'*','*','*'} };
    char end[BoardSize][BoardSize] = { {'*','*','*'},{'*','*','*'},{'*','*','*'} };
    char TempState[BoardSize][BoardSize] = { {'*','*','*'},{'*','*','*'},{'*','*','*'} };
    while (true) {
        arrayclear(start); // 数组重置
        arrayclear(end);
        arrayclear(TempState);
        EasyXSHOW show;
        show.InitialPage(start, end); // PageFirst
        show.mainbg(start, TempState, end);
        int sign = show.mainoperate();
        if (sign == FAIL) // 重置
            continue;
        if (sign == MYGO) { // 自己
            int result = show.Gomyself(TempState, end);
            if (result == WIN)
                show.winshow();
            if (result != SYGO)
                continue;
        }

        if (answerExist(start, end))
    }
}

```



```
{ // 系统执行阶段...

    AStarEightDigital Astar;
    clock_t startTime, endTime;
    int expandNum, generateNum;
    unsigned long long endstate;

    Astar.setup(start, end);
    startTime = clock();
// 计时开始
    Astar.AStar();
    endTime = clock();
// 计时结束
    vector<unsigned long long> routeResult;
    Astar.routeGet(routeResult);
    Astar.expandGet(expandNum);
    Astar.generateGet(generateNum);
    show.resultDataShow(endTime - startTime, expandNum, generateNum, routeResult);
// 显示搜索信息

    show.processShow(routeResult);
// 显示解过程
    Astar.GraphChangeData(end, endstate);
    show.searchTreeShow(Astar.TreeList, endstate);
// 显示搜索树
    if (Astar.TreeList.size() <= 100)
// 若搜索树过多，则没有打印的必要
        TreePrint(Astar.TreeList);
// cmd打印搜索树
    }
    else
//此时无解
        show.resultNone();
        cout << "success ! ! ! " << endl;
    }


    closegraph();

    return 0;
}
```

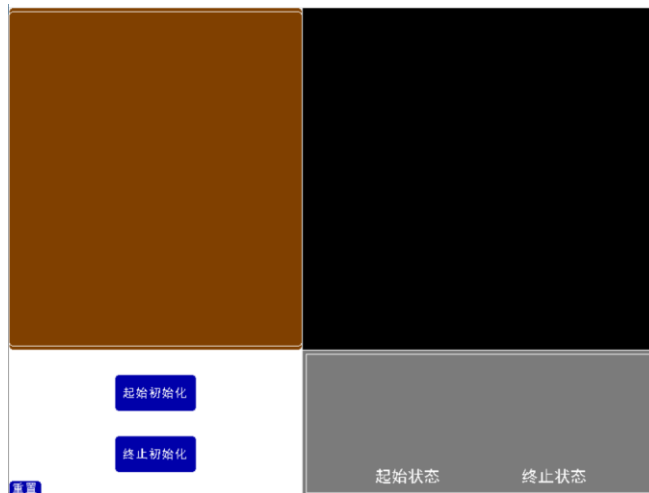
### 3.3 实验结果展示

#### 3.3.1 基础界面展示

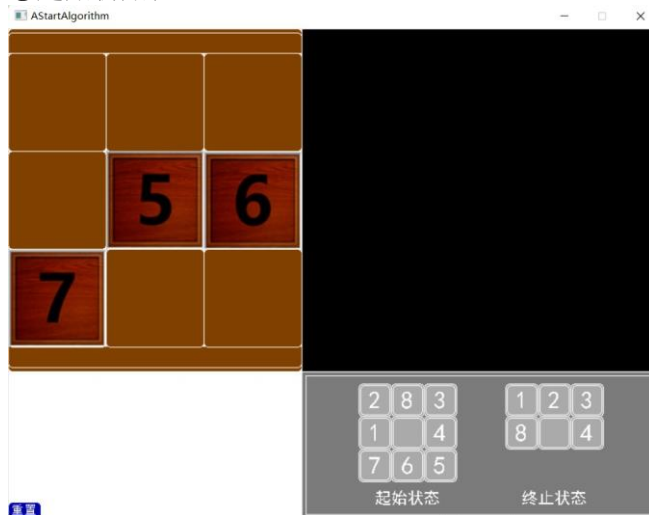
以教材页面展示



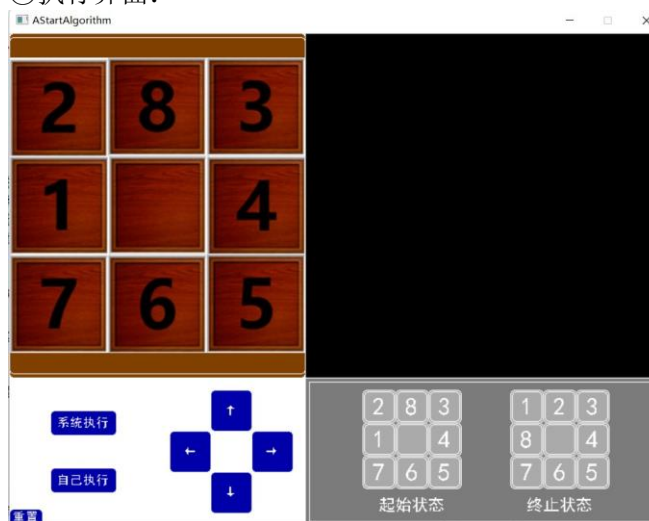
①初始界面:



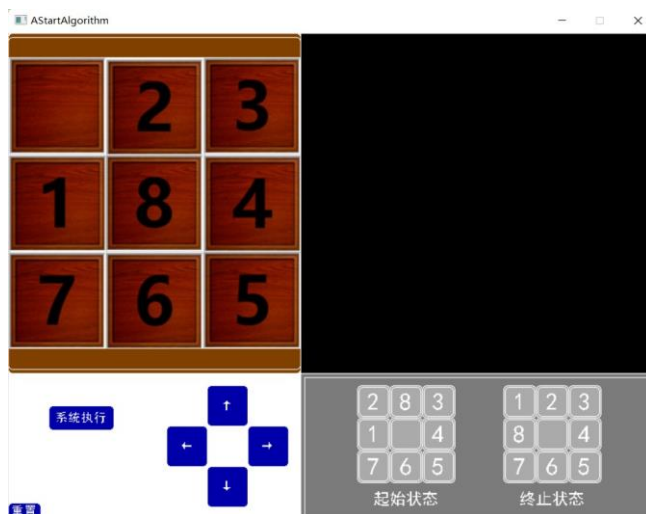
②起始初始化:



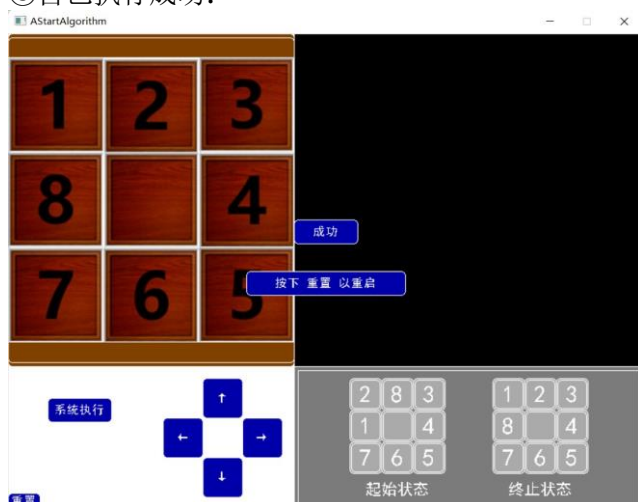
③执行界面:



④自己执行:



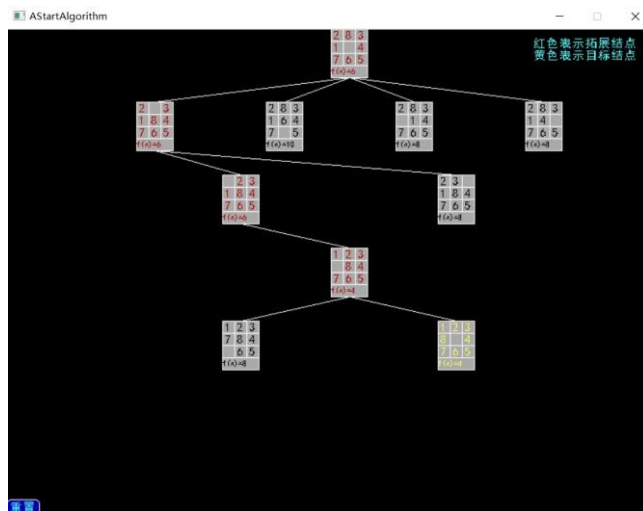
⑤自己执行成功:



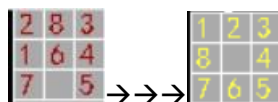
⑥系统执行:



⑦系统显示搜索树:

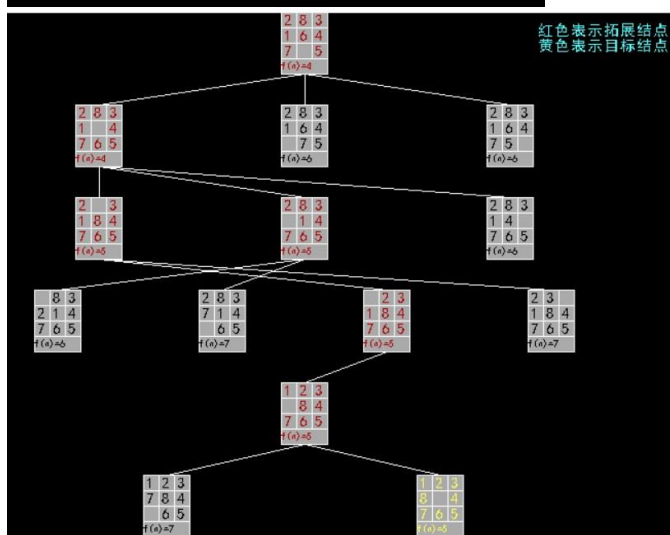


### 3.3.2 较小数据量展示



①  $h(n)$ =不在位棋子数

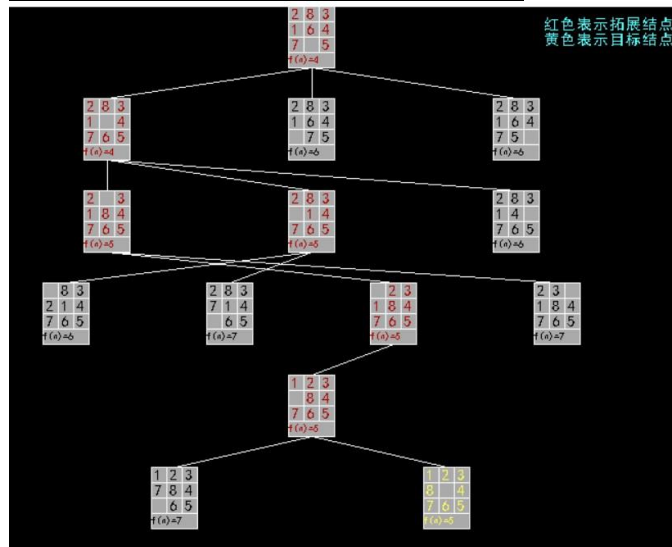
使用方法为 : 不在位棋子数  
 运行时间为 : 0 ms  
 生成结点数 : 14  
 拓展结点数 : 6  
 解路径步数 : 5  
 左侧开始执行解路径步骤...



②  $h(n)$ =欧拉距离

使用方法为 : 欧拉距离法  
运行时间为 : 1 ms  
生成结点数 : 14  
拓展结点数 : 6  
解路径步数 : 5

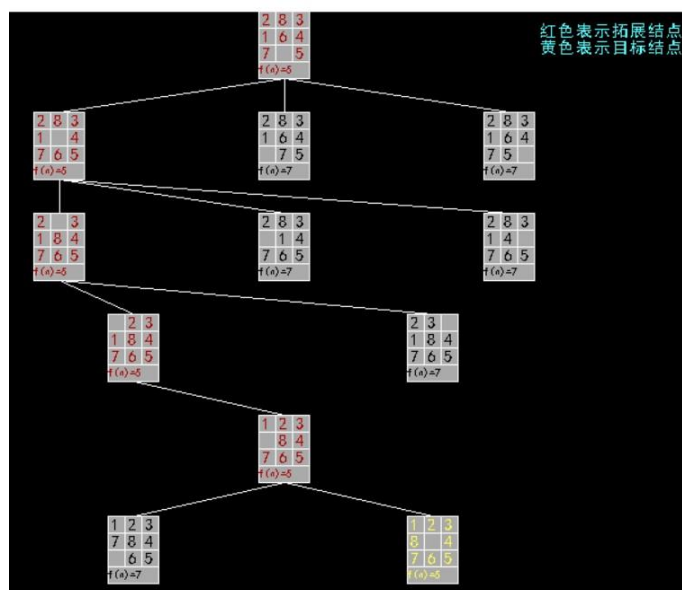
左侧开始执行解路径步骤...



③  $h(n)$ =曼哈顿距离和

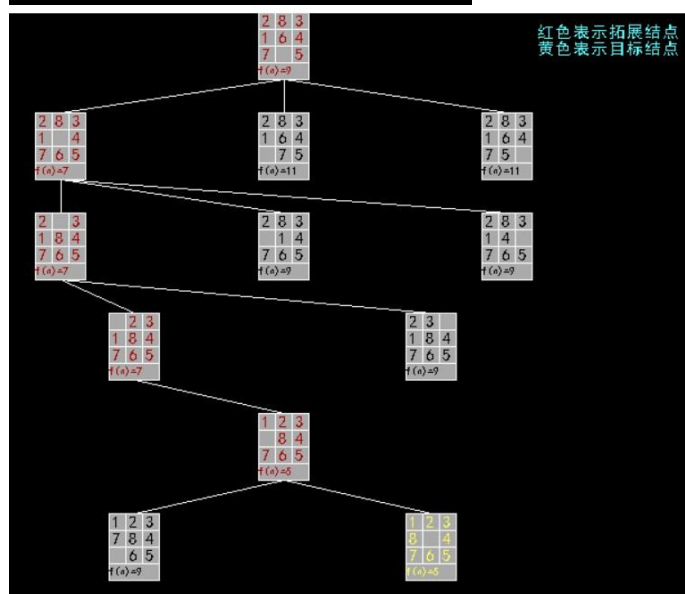
使用方法为 : 目标距离之和  
运行时间为 : 0 ms  
生成结点数 : 12  
拓展结点数 : 5  
解路径步数 : 5

左侧开始执行解路径步骤...



④  $h(n)$ =系数曼哈顿距离

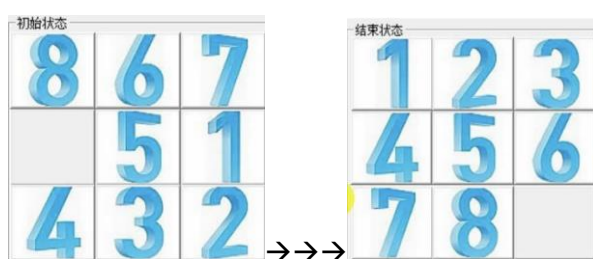
使用方法为 : 系数曼哈顿法  
 运行时间为 : 0 ms  
 生成结点数 : 12  
 拓展结点数 : 5  
 解路径步数 : 5  
 左侧开始执行解路径步骤...



当数据量较小时，我们可以发现不在位棋子数和欧拉距离的求解结果一致；曼哈顿距离和系数曼哈顿距离法的求解结果一致。四种方法，均可以求出最优解。

其中，不在位棋子数和欧拉距离的算法效率更低，拓展和生成的节点数也更多，这也符合预期，因为其  $h(n)$  相较于另外两组，更加远离实际  $h^*(n)$ 。

### 3.3.3 较大数据量展示



①  $h(n)$  = 不在位棋子数

使用方法为 : 不在位棋子数  
 运行时间为 : 28395 ms  
 生成结点数 : 28340  
 拓展结点数 : 18767  
 解路径步数 : 25

左侧开始执行解路径步骤...

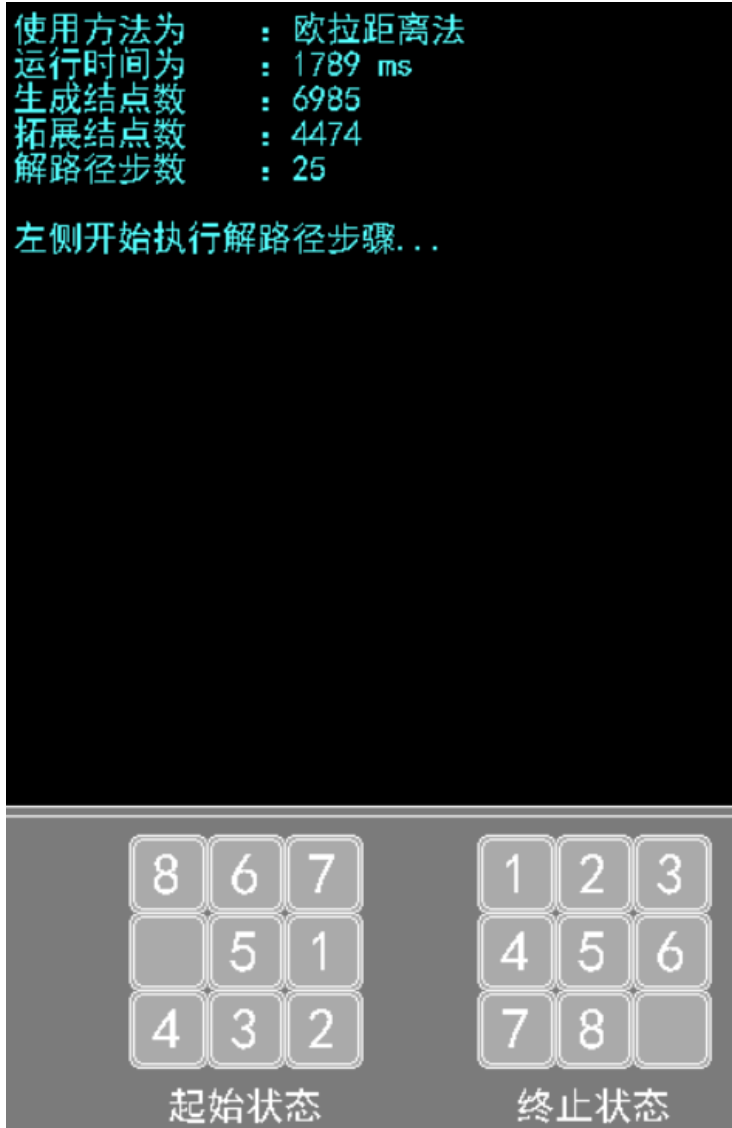
8	6	7
	5	1
4	3	2

起始状态

1	2	3
4	5	6
7	8	

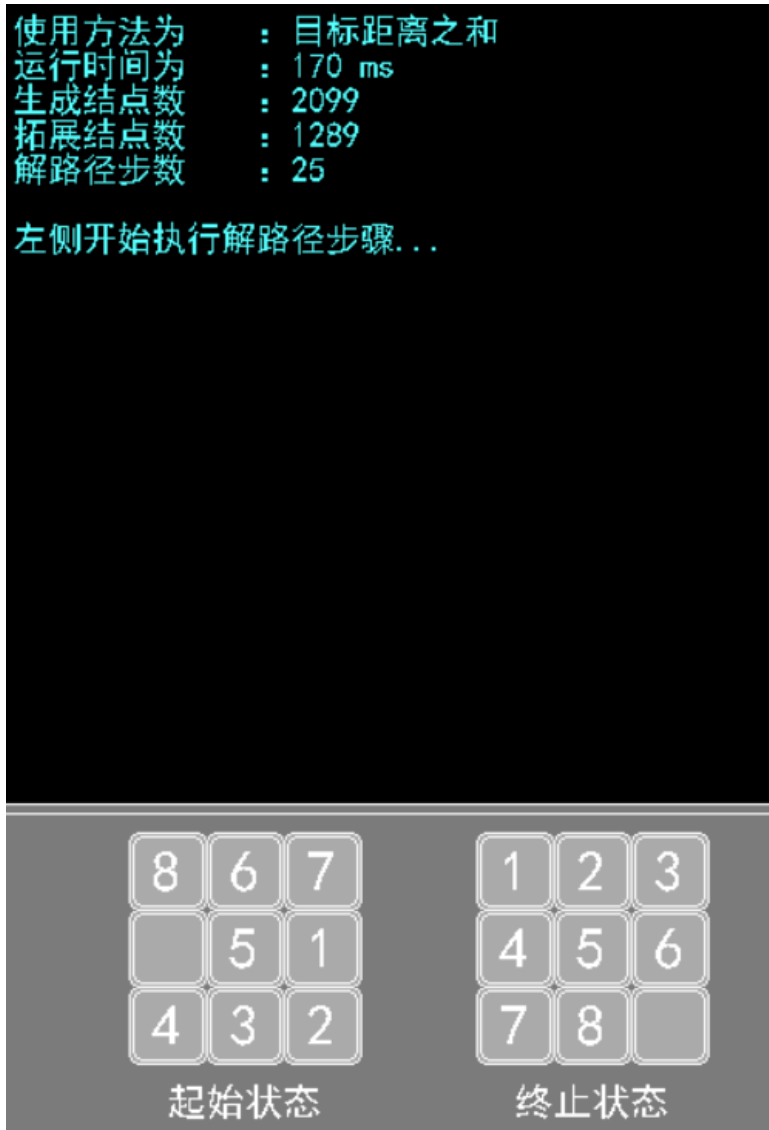
终止状态

② $h(n)$ =欧拉距离和

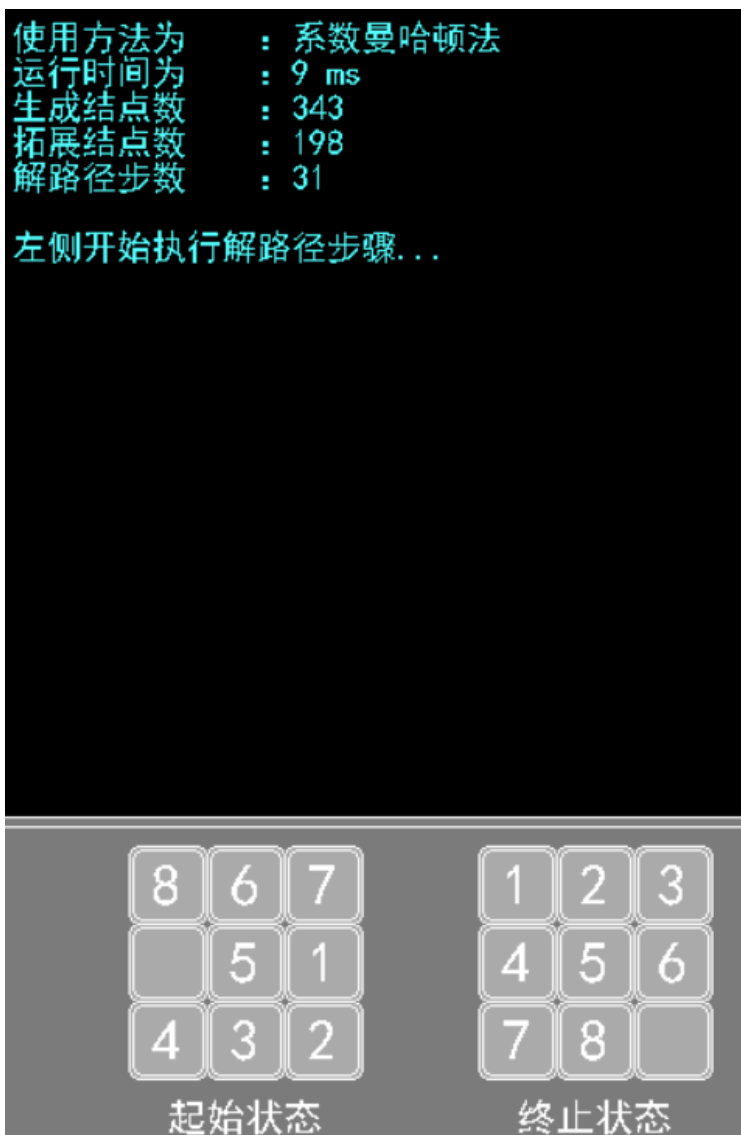


③ $h(n)$ =曼哈顿距离和





④ $h(n)$ =系数曼哈顿距离



加大数据量之后，我们能清晰地看到各个方法之间的差距

方法名	运行时间	生成结点	拓展结点	解路径步骤
不在位棋子数	28395ms	28340	18767	25
欧拉距离和	1789ms	6985	4474	25
曼哈顿距离和	170ms	2099	1289	25
系数曼哈顿法	9ms	343	198	31

### 3.3.4 实验结论

根据上列表格信息，我们可以得出以下结论：

首先系数曼哈顿法并非可采纳的，求得解路径并非最优结果，这也是可以推证的：

1	2	3
4	5	6
7	8	

1	2	3	4	5	6	7	8	
---	---	---	---	---	---	---	---	--

1	2	3
4	5	
7	8	6

1	2	3	4	5		7	8	6
---	---	---	---	---	--	---	---	---

我们可以发现，以上两状态真实距离只需要一步，但是系数曼哈顿法算出距离为 3， $h(n) > h^*(n)$ ，因此也就出现了系数曼哈顿法无法得到最优解，其并非可采纳的。

而其余三种方法，我们也可发现，均是可采纳的，但**曼哈顿距离法** 优于 **欧拉距离法** 优于 **不在位棋子数法**。这也符合，三种启发式函数同真实  $h^*(n)$  之间的接近关系。随机推算多个状态对比可知，曼哈顿距离法  $h(n)$  更加接近于实际距离，欧拉距离法次之，不在位棋子数相差最远。

综上，实验得出结论有：

- ① 系数曼哈顿法是不可采纳的，其余  $h(n)$  均可采纳
- ② 启发式函数性能对比：**曼哈顿距离法** > **欧拉距离法** > **不在位棋子数法**

### 4 总结

#### 4.1 实验中存在的问题及解决方案

整个实验过程中存在的问题不多，A\*算法的实现过程也较为顺利。主要问题在于考虑到算法实现的逻辑严密性上。即当 OPEN 表中最小值有多个时，如果随机选择一个进行遍历，而其子节点小于此最小值，则下一次遍历将在子节点产生。因此将出现同为最小值，但由于遍历的随机性问题，导致某些节点在整个过程中出现未被遍历的情况，这样是否会对最后的结果产生影响？对于这个问题思考了许久，仍无法得到令人信服的证明。

最后，将每层的多个最小值均进行遍历，以排除这样逻辑上的漏洞。但对于这种情况是否会出现，仍百思不得其解，无法想到具体证明其存在或不存在的推证。

#### 4.2 心得体会

经过本次实验，我对于 A\*算法的实现过程有了更加清晰的认识。启发式函数的选择极大程度影响 A\*算法的效率，并且如果选择不可采纳的算法，尽管最终结果可能并非最优解，但可以极大程度地提升效率，降低运行时间。因此，我们在使用 A\*算法时，应当考虑当时的实际情况，选择恰当地启发式函数，以达到更好的实际效果。

经过本次实验，对 C++ 地 EasyX 库有了更加清晰的了解，告别了 cmd 地小黑框，能实现较好的可视化效果。

#### 4.3 后续改进方向

后续可以在算法效率上提升：本次采用 OPEN 表和 CLOSE 表进行 A\*过程的搜索，并且还添加了一个 TreeList 表记录搜索过程。我们可以考虑是否可以仅使用一个表格即可完成 A\*搜索，并且记录搜索树，这将极大地降低 A\*算法实现地空间复杂度。

除此之外，在算法的时间复杂度方面，也可以查阅更多的相关文献，提升算法效率。

## 参考文献

- [1] 邓伟.基于 C#WinForm 的 AStar 寻路算法交互设计[J].信息技术与信息化,2021(08):80-83.
- [2] 谭威,胡新荣,雷伟.基于 Unity 的 A-Star 算法在游戏中的具体实现[J].计算机产品与流通,2018(11):121.陈桂娥,樊行雪,许振良.线性滴定中稳定常数测定方法比较[J].华东理工大学学报, 1996, 22(5): 620-625.
- [3] 刘好. 基于优化 Astar 与 MPC 的自动驾驶车辆路径规划研究 [D]. 东南大学,2019.DOI:10.27014/d.cnki.gdnau.2019.001055.

装

订

线