**武汉大学计算机学院**

**本科生实验报告**

**人工智能引论大作业实验报告**

专业名称：计算机科学与技术

课程名称：人工智能引论

指导教师：彭敏 教授

学生学号：2018302110388

学生姓名：李政民

二○一九 年 十二月

**郑 重 声 明**

本人呈交的实验报告，是在指导老师的指导下，独立进行实验工作所取得的成果，所有数据、图片资料真实可靠。尽我所知，除文中已经注明引用的内容外，本实验报告不包含他人享有著作权的内容。对本实验报告做出贡献的其他个人和集体，均已在文中以明确的方式标明。本实验报告的知识产权归属于培养单位。

本人签名： 李政民 日期： 2019.12.13

摘要

本次实验主要探讨A\*算法和IDA\*算法在15数码求解问题中的应用。

实验内容主要包括：

①使用A\*算法求解15数码问题。

②使用IDA\*算法求解15数码问题。

**关键词：** A\*算法，IDA\*算法，15数码问题

目录

**[实验一 使用A\*算法求解15数码问题 1](#_Toc3262)**

**[1实验目的和意义 1](#_Toc20153)**

[1.1 实验目的 1](#_Toc3747)

[1.2 实验意义 1](#_Toc15618)

**[2实验设计 1](#_Toc10370)**

[2.1 问题概述 1](#_Toc32601)

[2.2算法描述 2](#_Toc30527)

[2.3实验演示 6](#_Toc253)

**[3总结 10](#_Toc7006)**

[3.1实验中遇到的问题及解决方法 10](#_Toc16800)

[3.2结论 10](#_Toc23068)

**[实验二 使用IDA\*算法解15数码问题 11](#_Toc27791)**

**[1 实验目的与意义 11](#_Toc26743)**

[1.1实验目的 11](#_Toc17963)

[1.2实验意义 11](#_Toc14137)

**[2实验设计 11](#_Toc2626)**

[2.1 问题概述 11](#_Toc4219)

[2.2算法描述 12](#_Toc27138)

[2.3实验演示 13](#_Toc6728)

**[3总结 16](#_Toc3734)**

[3.1实验中遇到的问题及解决方法 16](#_Toc12095)

[3.2结论 16](#_Toc28086)

**[教师评语评分 17](#_Toc17772)**

**实验一 使用A\*算法求解15数码问题**

# 1实验目的和意义

## 实验目的

掌握A\*算法的基本思想。

## 实验意义

加深本人对于课堂所学算法的理解，提高本人算法与程序设计的综合能力，锻炼本人代码和自学能力，并能够进一步加深本人对于人工智能引论的兴趣，为将来的专业择项和就业规划产生影响。

# 2实验设计

## 2.1 问题概述

在4×4的棋盘上，将数字1，2，3，…，14，15以任意顺序置入棋盘的各个方格中，棋盘中留有一个空格，空格用0来表示。空格周围的棋子可以移到空格中。要求解的问题是：给出一种初始布局（初始状态）和目标布局，找到一种最少步骤的移动方法，实现从初始布局到目标布局的转变，如下表所示：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 10 | 1 | 11 | 14 | ----> | 1 | 2 | 3 | 4 |
| 2 | 3 |  | 5 | ----> | 5 | 6 | 7 | 8 |
| 8 | 9 | 4 | 15 | ----> | 9 | 10 | 11 | 12 |
| 12 | 13 | 6 | 7 | ----> | 13 | 14 | 15 |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 5 | 1 | 2 | 4 | ----> | 1 | 2 | 3 | 4 |
| 9 | 6 | 3 | 8 | ----> | 5 | 6 | 7 | 8 |
| 13 | 15 | 10 | 11 | ----> | 9 | 10 | 11 | 12 |
| 14 |  | 7 | 12 | ----> | 13 | 14 | 15 |  |

移动规则是：每次只能在空格周围的四个数字中任选一个移入空格,若在5000步以内不能完成，输出“该问题无解”，否则输出移动过程，如2.3.3运行结果所示。

## 2.2算法描述

### 2.2.1 A\*算法简介

A\*（A-Star）算法是求解最短路最有效的直接搜索方法，也是许多其他问题的常用启发式算法。

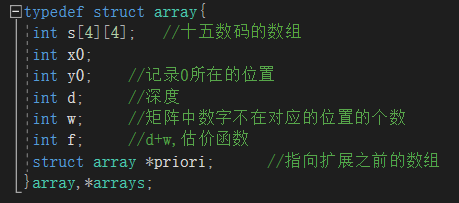
其基本思想是：定义一个评价函数f，对当前的搜索状态进行评估，找出一个最有希望的结点来扩展。评价函数形式为：f(n)=g(n)+h(n)。其中f(n)是从初始状态经由状态n到目标状态的代价估计，g(n) 是在状态空间中从初始状态到状态n的实际代价，h(n) 是从状态n到目标状态的最佳路径的估计代价。

A\*算法要求启发函数中的h(n)是处在h\*(n)的下届范围，即满足h(n)<=h\*(n).

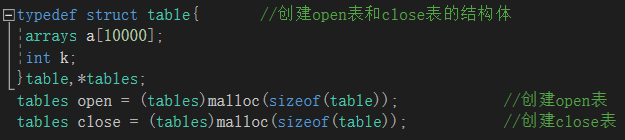
### 2.2.2 定义状态空间

十五数码问题的一个状态就是十五个数字在棋盘上的一种放法。每个棋子用它上面所标的数字表示，并用0表示空格，这样就可以将棋盘上棋子的一个状态存储在一个二维数组 s[4][4]中。

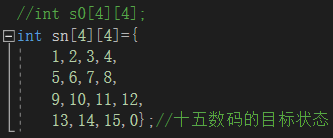
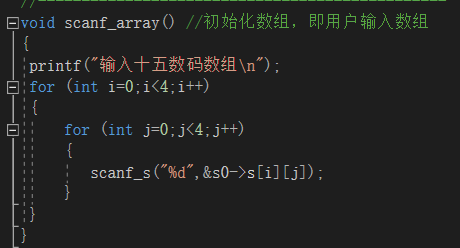
建立矩阵array结构体，s[4][4]为4\*4的十六宫格，x0为矩阵中空白位置的横坐标，y0为矩阵中空白位置的纵坐标，d表示搜索的深度，w表示矩阵中数字不在对应的位置的个数，f表示估价函数。



建立open和close的结构体，创建open表和close表。



确定初始状态和目标状态。



### 2.2.3 选择启发式函数

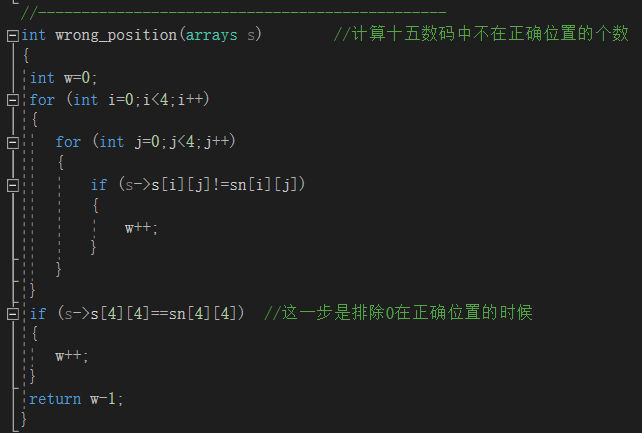
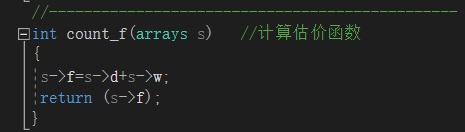
A\*算法评价函数形式为：f(n)=g(n)+h(n)。

本作业取f(s) = d(s) + w(s)。其中，d(s)为从初始结点到节点s的实际代价，w(s)是从结点n到目标结点的最佳路径的估计代价。这里取f(n)=f(s), g(n)=d(s), h(n)=w(s)，d(s)为搜索树的深度，w(s)是错放位置的数字个数，并且w(s)<=w\*(s), 因为w(s)只估计放错位置的数字个数，还远未考虑从放错位置到正确位置要移动的困难程度（移动的次数可能多于1次）。

这样必须满足两个条件：

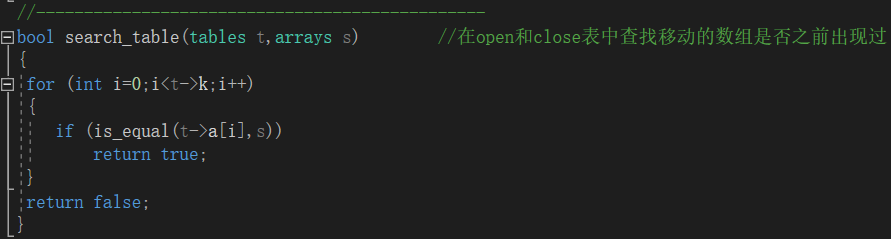
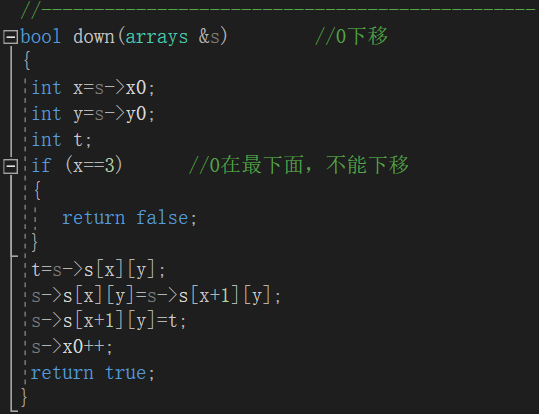
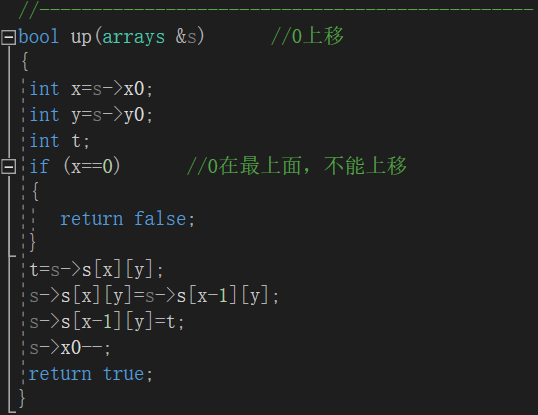
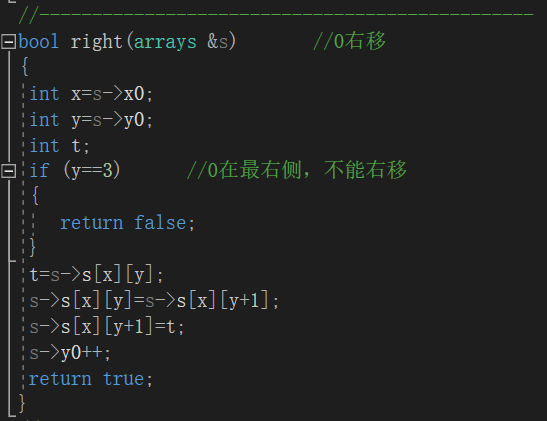
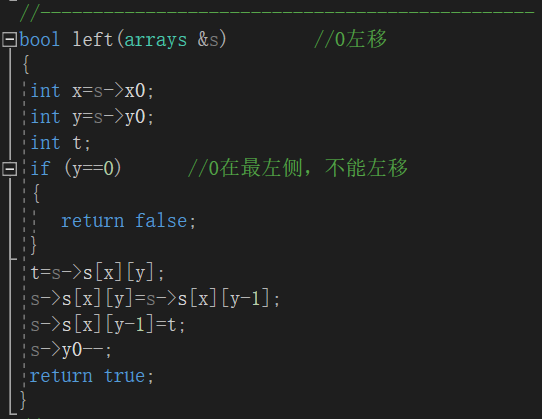
（1）d(s)>=d\*(s)（大多数情况下都是满足的，可以不用考虑），且f必须保持单调递增。

（2）w必须小于等于实际的从当前节点到达目标节点的最小耗费w(s)<=w\*(s)。第二点特别的重要。可以证明应用这样的估价函数是可以找到最短路径的。



估价函数中，主要是w(s)体现了搜索的启发信息，因为d(s)是已知的，计算w(s)也成为了本算法的重点。

### 2.2.4 定义操作



### 2.2.5 算法流程

（1）生成一个只包含开始节点s0的搜索矩阵S，把s0放在一个叫OPEN的列表上。

（2）生成一个列表CLOSED，它的初始值为空。

（3）如果OPEN表为空，则失败退出。.  
（4）选择OPEN 上的第一一个 节点，把它从OPEN中移入CLPSED，称该节点为s。  
（5）如果s是目标节点，顺着S中，从s到s0的指针找到一条路径， 获得解决方案，成功退出(该指针定义了一个搜索树，在第7步建立)。  
（6）扩展节点s，生成其后继结点集P，在S中，s的祖先不能在P中。在S中安置P的成员，使他们成为s的后继。  
（7）从P的每一个不在S中的成员建立一个指向s的指针(例如，既不在0PEN中，也不在CLOSED中)。把P的这些成员加到OPEN中。对P的每一个已在OPEN中或CLOSED中的成员p，如果到目前为止找到的到达p的最好路径通过s, 就把它的指针指向s。对已在CLOSED中的P的每个成员，重定向它在S中的每个后继，以使它们顺着到目前为止发现的最好路径指向它们的祖先。  
（8）按递增f\*值，重排OPEN (相同最小f值可根据搜索树中的最深节点来解决)。（9）返回第3步。  
  
      在第7步中，如果搜索过程发现一条路径到达一个节点的代价比现存的路径代价低，就要重定向指向该节点的指针。已经在CLOSED中的节点子孙的重定向保存了后面的搜索结果，但是可能需要指数级的计算代价。

## 2.3实验演示

### 2.3.1实验平台

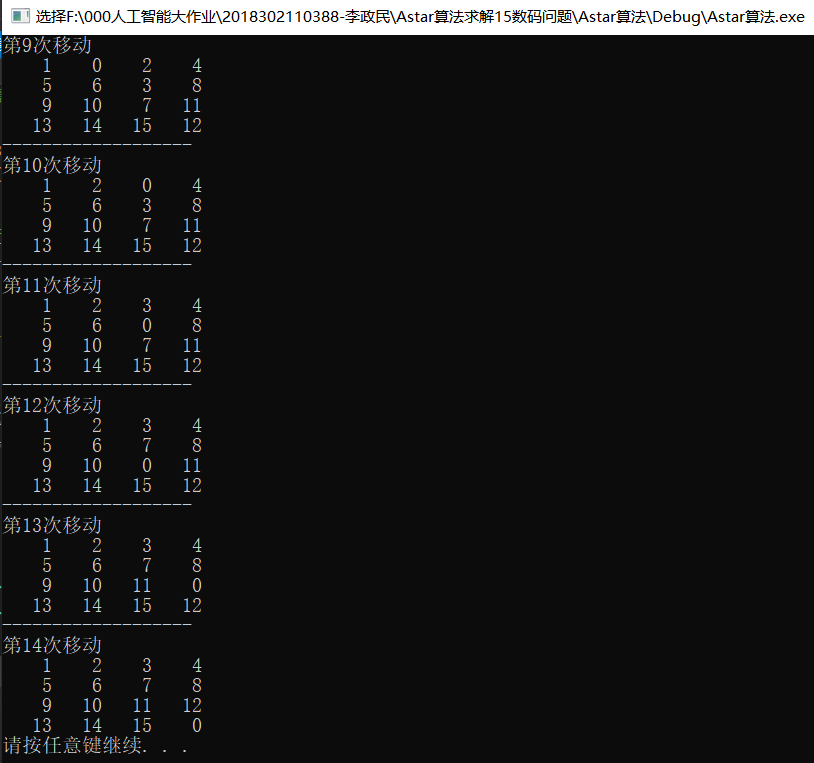
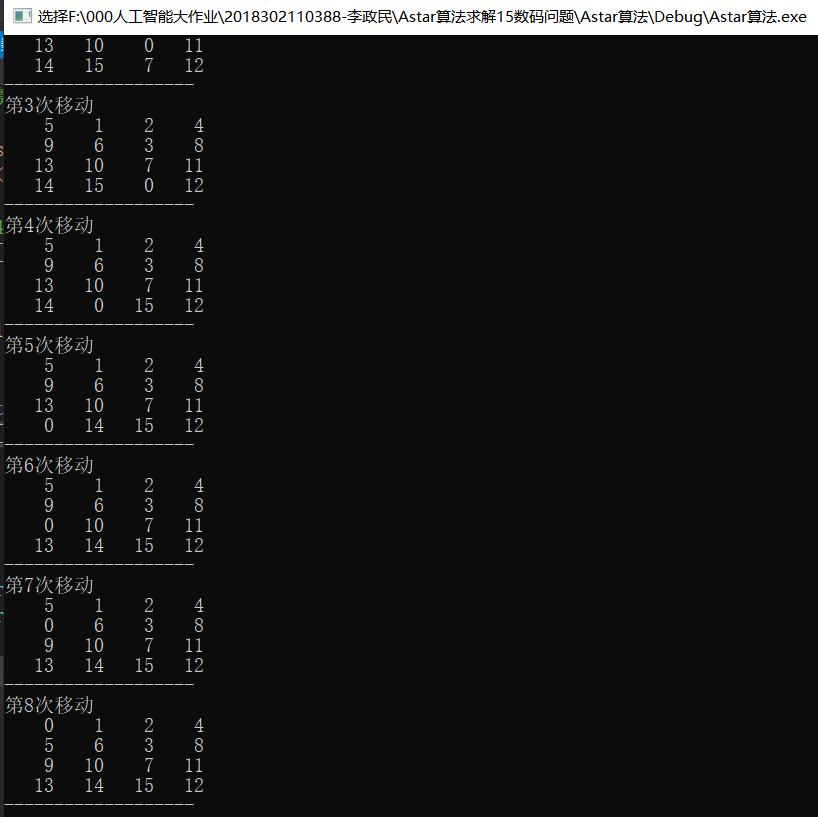
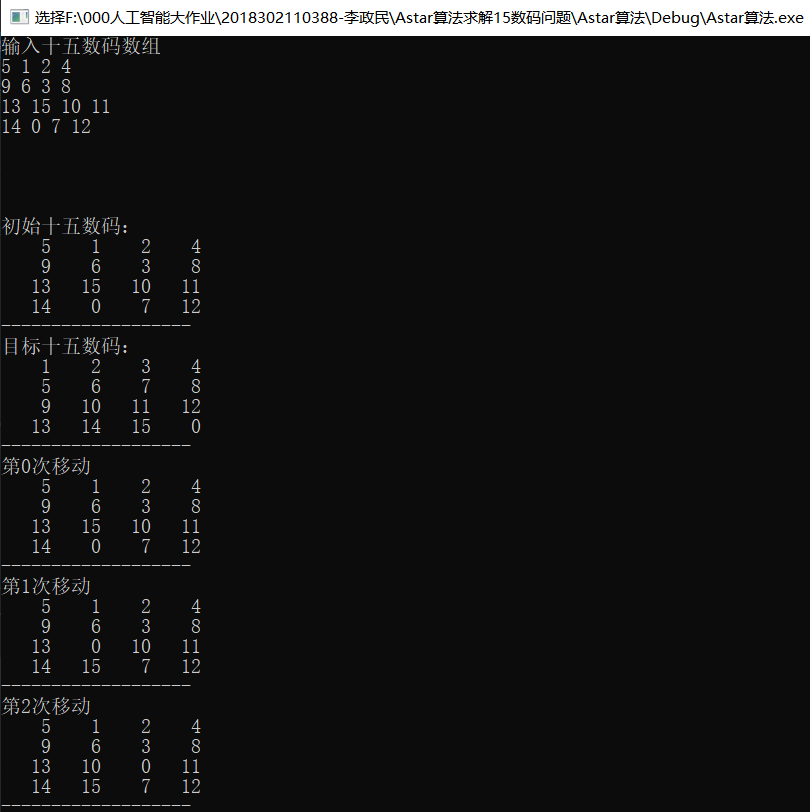
Visual Studio 2017编辑器 C++语言/C语言

### 2.3.2项目结构

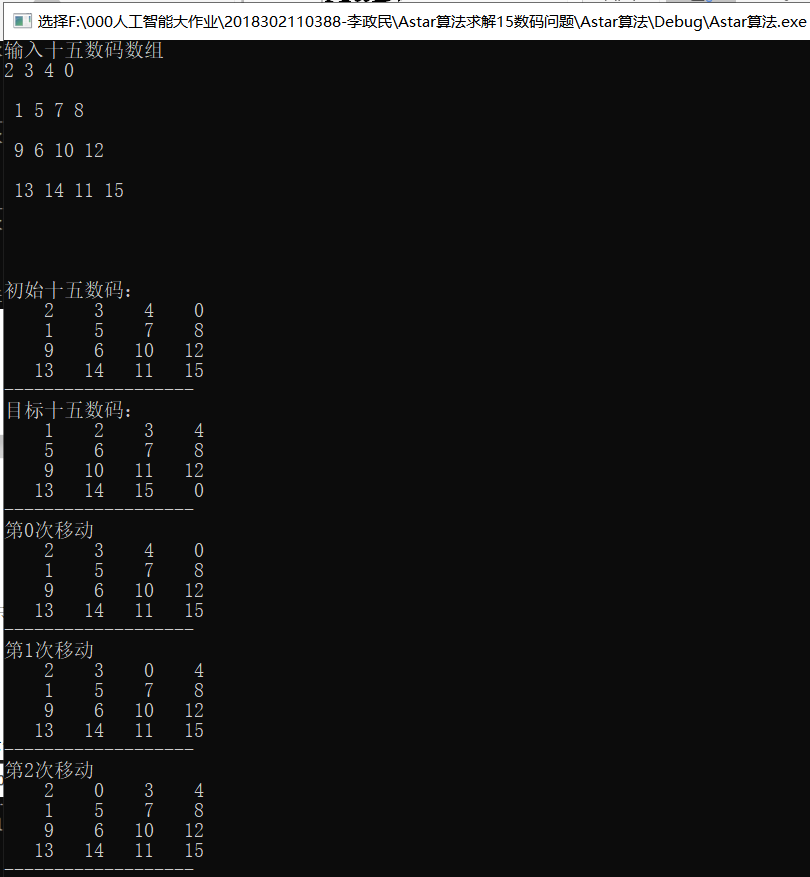
Astar算法求解15数码问题.cpp

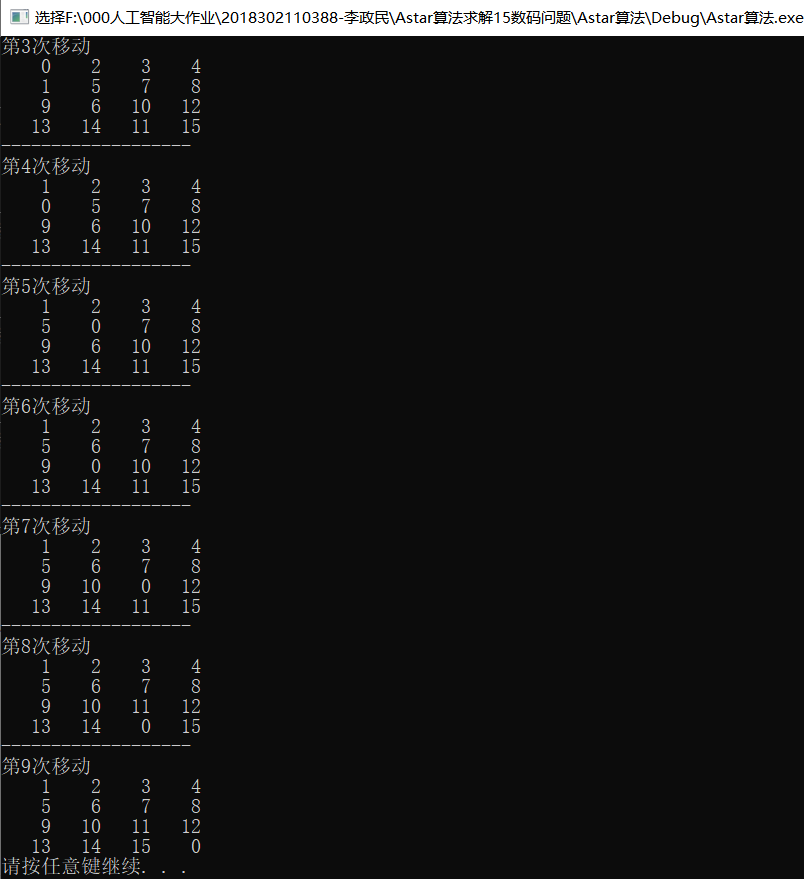
### 2.3.3运行结果

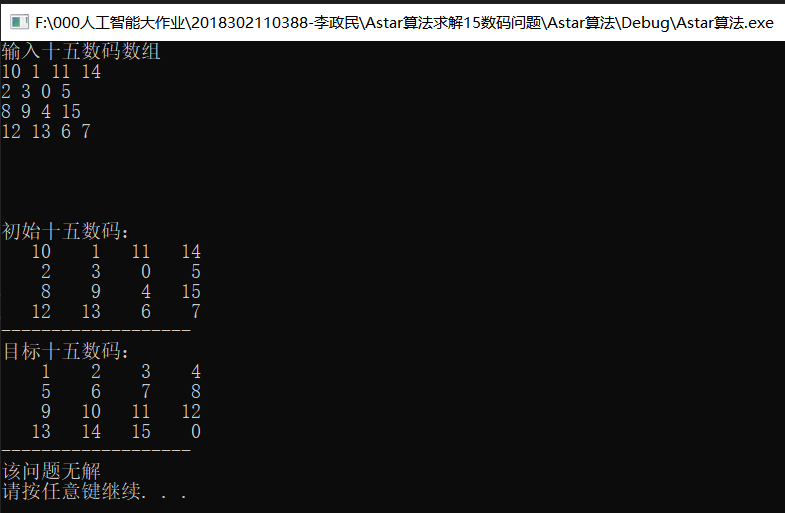
有解示例1：共14步完成。

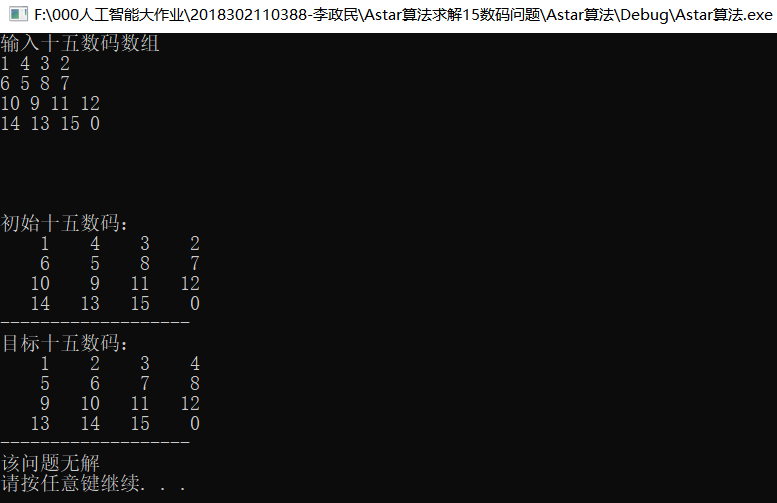


有解示例2：共9步完成。



无解示例1：

无解示例2：



# 3总结

## 3.1实验中遇到的问题及解决方法

问题1：通过搜索资料发现牛人都是通过逆序奇偶性来判断N数码是否有解的，本人感觉方法有点复杂，不知道如何去实现。

解决方案：由于回忆起A\*算法具有可采纳性，如果问题有解，即S0->Sg存在一条路径，A\*算法一定能够找到最优解。于是我决定通过增大循环判断的次数（5000次），来确保如果优解就能得出最优解，从而判断问题是有解还是无解。

问题2：由于15数码问题涉及的存储空间较大，在open表中用到的指针数量很多，本人在书写这些指针代码的时候很容易弄混含义。

解决方案：耐心地逐语句或者逐过程对程序进行调试，在容易混淆含义的指针旁边备注以提醒自己，在关键之处观察Open表前驱后继的指针变化的情况，找到出问题的函数并进行修正。

问题3：书本上介绍的都是8数码问题，当我想做15数码问题的时候没有什么头绪。

解决方案：通过仔细温习课本知识，本人发现知识都是融会贯通的。15数码问题无非是8数码问题的延伸，将3\*3的地图拓宽至4\*4的地图，数据更大了一些，其余的解决方案和方法没有明显的区别！

## 3.2结论

当15数码问题有解时，程序可以正确的给出问题的解，即可以显示每一步标记的变换路径。

当15数码问题无解时，程序会显示该问题无解。

本人从课本上了解到了，经典算法A\*算法在不限时间和空间的前提下，是一定能找到最优解的，这也是A\*算法给我最深刻的印象。但是由于时间和空间的限制，本实验求得的只是局部最优解。在课堂上习得的知识通过本次大作业实验的巩固，让我对于A\*算法的思想有了更深一步的理解。在代码的实际实践中我也发现了书本上通俗易懂的知识在现实实现中会遇到很多意想不到的困难，但也正是这些困难督促我们自发寻找资料，去探索去钻研，我觉得这是这次课程实践给我最大的收获。

# 实验二 使用IDA\*算法解15数码问题

# 1 实验目的与意义

## 1.1实验目的

本实验旨在加深自己对人工智能经典算法A\*算法的优化算法—IDA\*算法的理解与使用。

## 1.2实验意义

通过使用A\*算法的优化算法——IDA\*算法，解决15数码问题，提升自我求解问题的能力。

# 2实验设计

## 2.1 问题概述

在4×4的棋盘上，将数字1，2，3，…，14，15以任意顺序置入棋盘的各个方格中，棋盘中留有一个空格，空格用0来表示。空格周围的棋子可以移到空格中。要求解的问题是：给出一种初始布局（初始状态）和目标布局，找到一种最少步骤的移动方法，实现从初始布局到目标布局的转变，如下表所示：

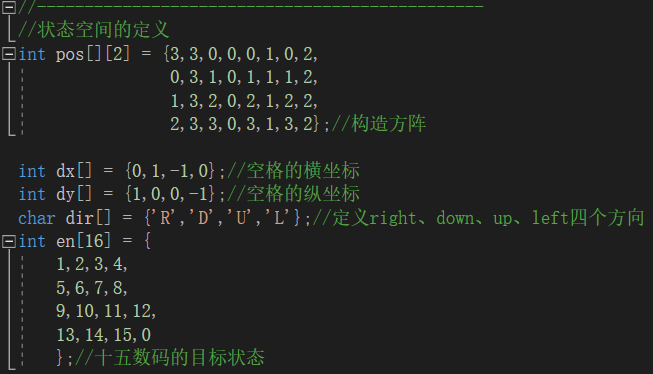
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 10 | 1 | 11 | 14 | ----> | 1 | 2 | 3 | 4 |
| 2 | 3 |  | 5 | ----> | 5 | 6 | 7 | 8 |
| 8 | 9 | 4 | 15 | ----> | 9 | 10 | 11 | 12 |
| 12 | 13 | 6 | 7 | ----> | 13 | 14 | 15 |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 5 | 1 | 2 | 4 | ----> | 1 | 2 | 3 | 4 |
| 9 | 6 | 3 | 8 | ----> | 5 | 6 | 7 | 8 |
| 13 | 15 | 10 | 11 | ----> | 9 | 10 | 11 | 12 |
| 14 |  | 7 | 12 | ----> | 13 | 14 | 15 |  |

这里用IDA\*算法来解决这个问题。

## 2.2算法描述

### 2.2.1 状态的表示



### 2.2.2 启发式函数的选择

A\*算法评价函数形式为：f(n)=g(n)+h(n)。

这里了定义h函数，为每个数字到目标位置的曼哈顿距离之和。

### 2.2.3 算法流程

同八数码一样，我们定义h函数，为每个数字到目标位置的曼哈顿距离之和。然后这些写完，编译运行后运算的时间太长了，显然还需要优化。

首先第一个优化，是对无解的判断。当移动空格时可以发现，左右移动，是不改变这15个数字对应的序列的。而上下移动，会将4个数字的位置进行改变，这样会导致逆序数的奇偶性改变（+-3或+-1）。同时我们还要让空格移动到右下角的位置。

所以，15数码有解的条件是：15个数字的逆序数加上空格移动到右下角的需要的行数的和目标状态的逆序数的奇偶性相同。因为最终的状态的逆序数是偶数，所以给出的局面也应该是偶数。

第二个优化，在dfs中优化，也是更重要的优化，就是在下一步禁止向上一步的反方向移动。如果不禁止下一步向上一步的反方向移动，就回到了原局面，因为在IDA\*中没有判重函数，这样会大大增加搜索树的分支，降低效率。

## 2.3实验演示

### 2.3.1实验平台

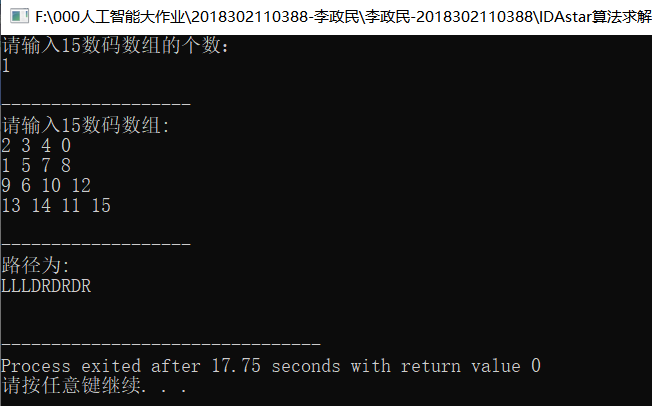
Visual Studio 2017编辑器 C++语言/C语言

### 2.3.2项目结构

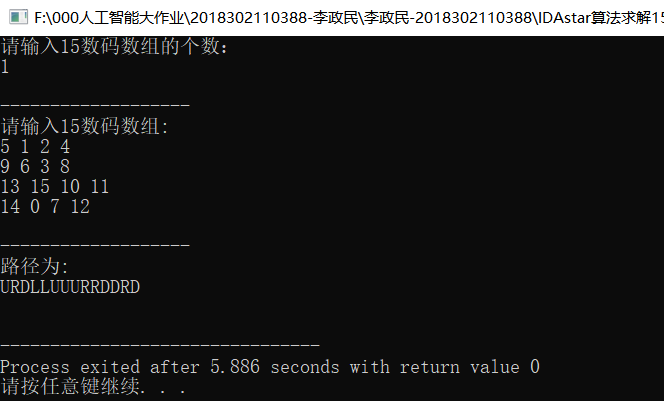
IDEstar算法求解15数码问题.cpp

### 2.3.3运行结果

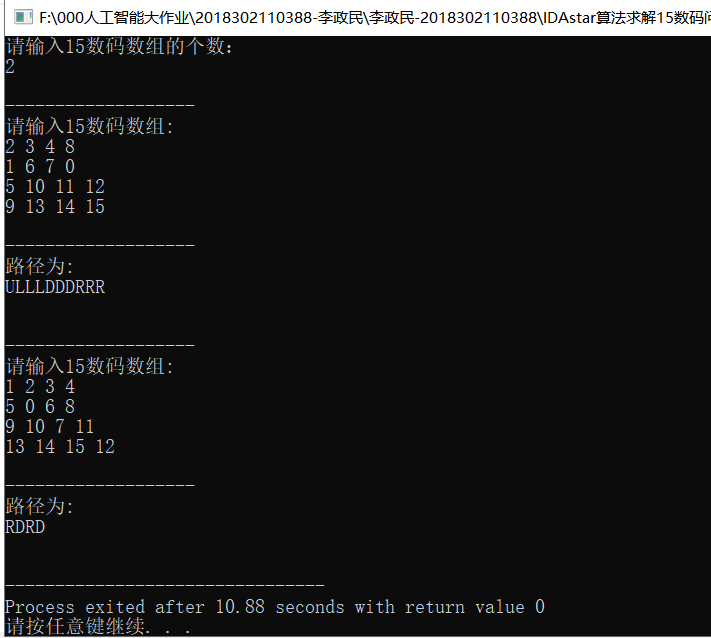
有解示例1：共9步完成。



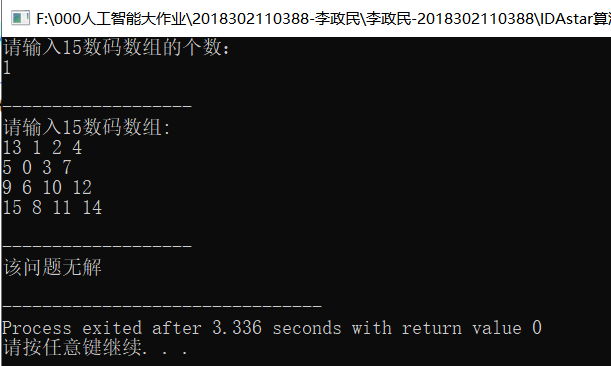
有解示例2：共14步完成。



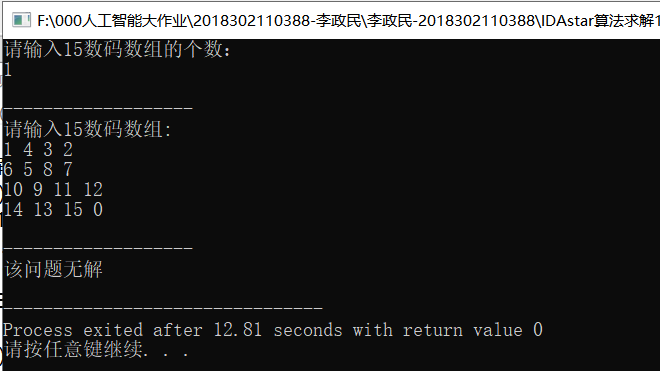
有解示例3：分别用10步和4步完成。



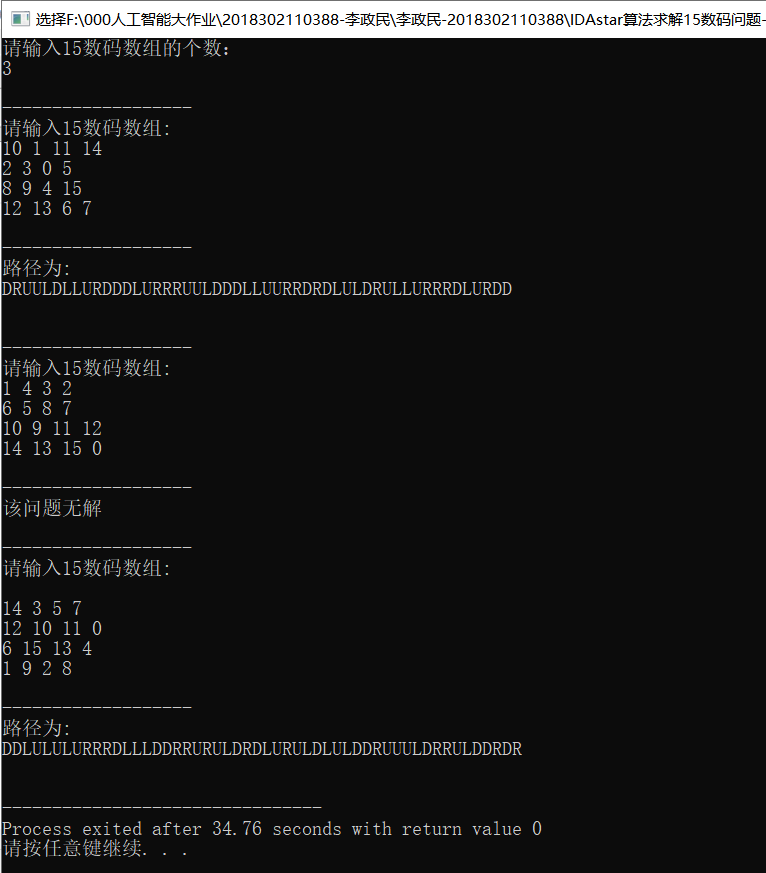
无解示例1：



无解示例2：



混合示例：



# 3总结

## 3.1实验中遇到的问题及解决方法

问题：这次实验最大的难点在于对有无解的判断，上个实验中没有用特别好的办法去解决，如何把上个实验的局部最优转变为全局最优，是IDA\*算法需要解决的问题。

解决方案：通过查阅逆序数奇偶的相关知识点和一步一步摸索，惊讶地发现当左右移动空格时，15个数字对应的序列是不会改变的，由此总结出了15数码有解的条件是：15个数字的逆序数加上空格移动到右下角的需要的行数的和目标状态的逆序数的奇偶性相同。

## 3.2结论

经过多次测试，在选取合理的样本示例后，本算法可以找到符合题意的全局最优解。总体来讲，优化的IDA\*算法还是可以起到很好的效果，可以有效解决传统A\*算法解决不了的问题。自己也对于全局最优和局部最优的概念有了更为深刻的认识。

对于IDA\*算法的核心思想也有了进一步的认识：设置每次可达的最大深度depth，若没有到达目标状态则加深最大深度。采用估值函数，剪掉f(n)大于depth的路径。采取了深度优先遍历和剪枝的知识，全方面提高的本人的能力。

教师评语评分

评语：

评分：

评阅人：

年 月 日

（备注：对该实验报告给予优点和不足的评价，并给出百分之评分。）