实验报告

## 八数码问题求解实验报告

**班级：10011703 姓名：李昕 学号：2017302629**

**一 实验题目**

图搜索问题中的八数码问题求解

**二 实验目的**

1 熟悉和掌握启发式搜索/A\*搜索的定义、估价函数和算法过程；

2 理解和掌握搜索过程，能够用选定的编程语言求解八数码问题，理解求解流程和搜索顺序；

3 比较并分析图搜索策略的实质，通过实验理解启发式搜索/A\*搜索的意义。

**三 实验要求**

1以九宫问题/八数码问题为例，以某种启发式搜索/A\*搜索策略编程演示其搜索过程；

2 定义启发式函数,能正确求解出从初始状态到目标状态的移动路线；

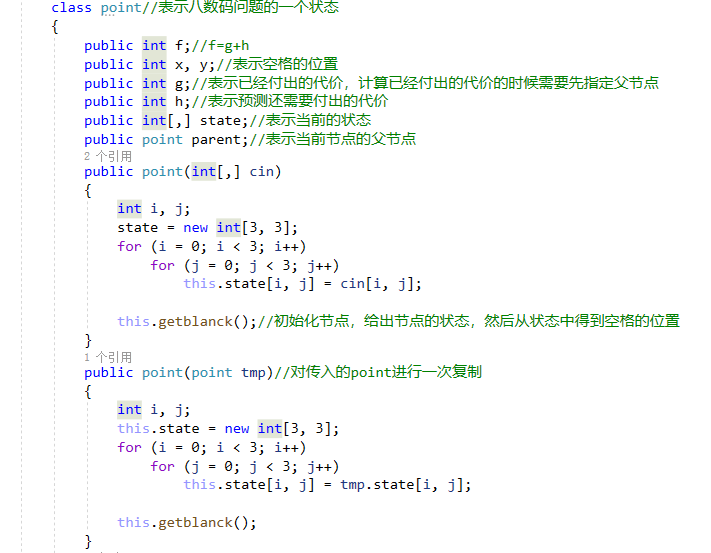
3 对不可达状态能进行正确识别；

4 对所采用的启发式函数做出性能分析。

**四 数据结构**

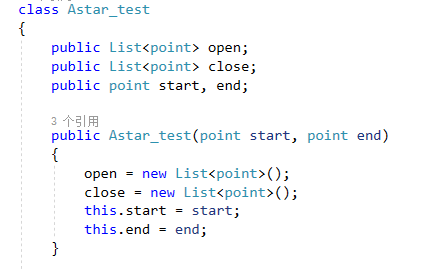
请说明八数码状态、OPEN表和CLOSE表是如何定义的。

1)八数码的状态表示：八数码中每一个节点的状态我采用Point类的一个实例来表示



这个数据结构表示八数码问题的一个状态，九个格子用一个二维数组state表示，除了一些必要的信息如g,h,f,parent，为了方便后面的计算，我提取出了空格的位置用(x，y)表示，重载了两个构造函数，一个根据输入的二维数组来初始化state，另一个是对输入的一个状态进行复制，方便后面子节点的生成。

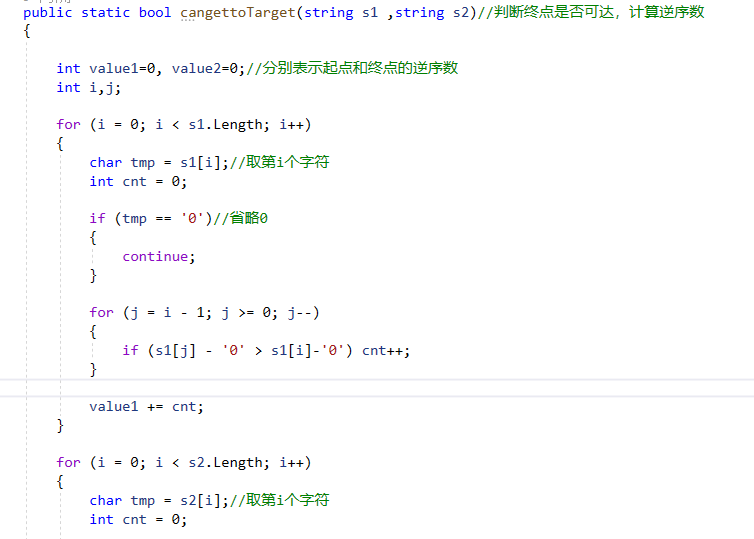
2)OPEN表和CLOSE表的数据结构

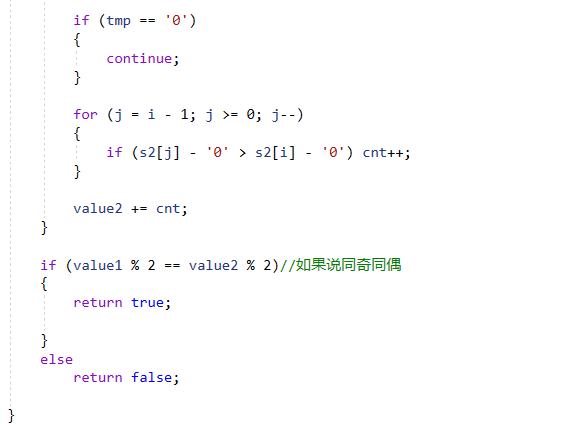


这里我直接采用了C#提供的List类来实现的，在构造函数中传入用户输入的初始和目标节点，然后初始化open表的close表

**五 实验算法**

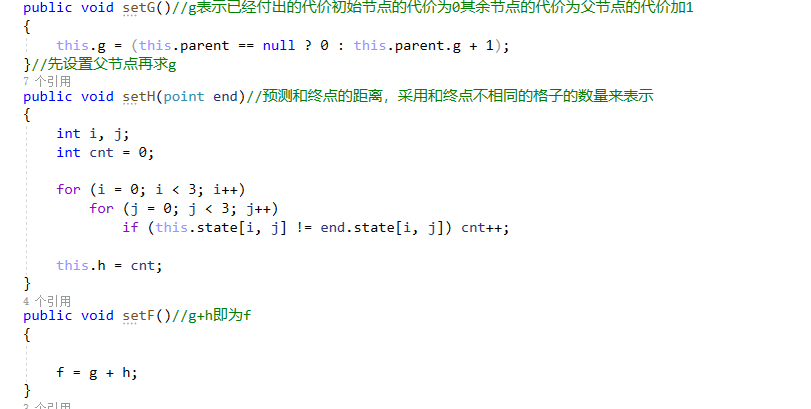
1 说明有解和无解如何判定





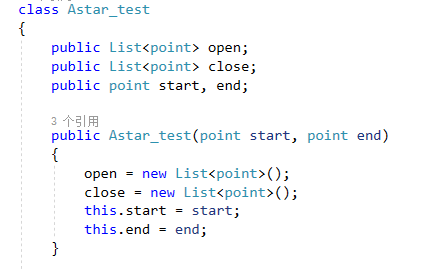
Value1和Value2分别表示起点和终点的逆序数，每次提取出一个字符，然后与前面的字符进行比较，如果前面的字符比当前的字符大的话就产生了一个逆序，逆序数加1，对终点和起点分别计算逆序数，如果说是同奇同偶的话就是可以到达，如果不是就不可到达

2 说明启发式函数如何设定；



这里我的启发函数设定是f=g+h，当然在这个问题中我还实现了广度优先算法和全局择优算法，采用面向对象的思想，使用继承将启发函数分别设置为f=g和f=h即可。

3 说明open表和close表如何实现；



Open和close表我在这里采用的是C#提供的List类，其中的每一个元素为一个状态节点

4 说明实验中采用的搜索算法。

本实验采用了A\*算法和全局择优和广度优先算法，

以A\*算法为例，首先将初始节点加入到open表中，然后进入循环，每次从open表中取出代价最小的(f最小)节点进行扩展，每次可以有四种方式，如果说是父节点则舍弃，如果说不可到达，则舍弃，如果说可以到达又不是父节点，则继续观察如果在open表中，则找到了一条更优路径，更新open表中的节点的父节点设置为当前正在扩展的节点，重新设置g，f，如果在close表中，则表明又找到了一条更优路径并且这个节点之前已经到达过了，并且已经产生了子节点，因此需要把这个在close表中的节点的父节点设置为当前正在扩展的节点重新设置g，f然后把它从close表中移除，加入到open表中，如果既不在open表中也不再close表中，则直接加入到open表中。直到最后找到终点

全局择优的f设置为h即可

广度优先的f设置为g即可

**六 实验结果**

1 要求有实验运行结果截图，以及必要的说明；

A\*算法运行结果：



点击单步执行



可以看到A\*算法可以找到最优解，速度适中(2s)

全局择优算法运行结果：



点击单步执行：



可以看到，全局择优往往不是找到的最优路径，但是收敛速度非常快(ms)

广度优先算法结果：



点击单步执行：  
 

可以看到广度优先算法也可以找到最优路径，但是由于解空间巨大收敛速度非常慢(1分钟)

同时我还加入了游戏功能，即通过用户输入w表示向上移动方块，s表示向下移动方块，a表示向左移动方块，d表示向右移动方块，如下图所示



输入d



在输入d



2 对不可达状态能进行正确识别；

状态无法到达的情况：



3 对所采用的策略进行性能分析。

通过之前的三种算法比较，很能明显地看出，全局择优最快，A\*算法适中，广度优先最慢，即f=g+h当h所占的比重越大则搜索的速度越快，但是也越难得到最优解。其中A\*和广度优先可以找到最优解，全局择优一般找不到最优解

**七 实验总结及体会**

本次实验我实践了智能搜索算法A\*算法，在编写程序的过程中仔细体会了该算法的思想，其中遇到了不少问题，由于是第一次使用面向对象的语言实现算法，之前写算法的时候都是用的C语言使用的数据结构是结构体，在面向对象的语言中使用的是类，有一点不习惯，但是同时我也体会到了其便利性，直接使用C#提供的List类就可以直接实现open表和close表的数据结构了，如果是放在之前用C语言实现还需要自己写一个链表结构来实现open表和close表无疑增加了复杂性。同时利用面向对象中的继承思想，通过继承A\*算法，可以很便捷地实现广度优先和全局择优，同时我也犯了一个错误，使得代码看起来比较不规范，由于我将计算代价f的函数写在了point类中，但是继承的时候是继承的Astar\_test这个类，我就只能在A\*算法中把用到计算f的地方用f=g和f=h代替，但是如果在写代码之前已经规划好，把这个函数写在Astar\_test中，继承的时候只需要覆盖这个函数即可，这就启发我以后在写代码之前要将具体的框架思路构建好再开始。