在八数码问题中，需要一个节点位置的好坏用评估函数进行评估，A\*算法的评估函数可以表示为：f（n）=g（n）+h（n）。其中，f是节点总路径估价函数，g是起点到n的实际代价，h是n到目标的困难程度。

本实验中的估价函数用‘不在相应位置节点的个数’来评估，在八数码数组中，每有一个数不在相应位置，这个节点的估价就加一。代码如下：

void count(Node& node,int\* end)

{

node.h = 0;

for (int i = 0; i < 9; i++) //不在相应位置节点的个数

{

if (node.number[i] != end[i])

node.h++;

}

node.f = node.h + node.g;

}

**步骤**

首先读入初始数组与目标数组，通过比较两数组的逆序数奇偶值是否相同，判断八数码是否有解。

如果有解，定义两个表open和close。Open用于存放已经生成，并且已经用启发函数进行过评估但是没有产生它们的后继节点的那些节点，close表用于存放已经生成并且已经考察过的点。

把初始状态放入open表，设close为空。

当open表为空时表示失败，一直循环直到当前状态等于目标状态。

由小到大排序open表，取出估价值最小的节点加入到close表中。

分别对于上下左右这四个移动方向产生后继节点，进行判断。

后继节点的实际代价加一，建立父节点的联系，计算后继节点的评估值。

如果后继节点不在open也不在close表中，加入到open表。

如果后继节点在close表中，如果估价值比原来的小，加入到open表中，并且把close表中原来的值删除。

如果后继节点在open表中，如果估价值比原来的小，open表中原来的节点被代替，重新计算节点的估价值。

直到当前状态等于目标状态，把close表中的节点按顺序存入node数组中。

**数据结构**

struct Node

{

int number[9];//节点的数值

int Parent;//节点的父节点

int h;//节点到达目的的困难程度

int g;//节点的实际路径

int f;//节点的总路径

};

