天津大学


# 八数码求解实验报告

学院名称 智能与计算学部

年　　级 2019 级

专　　业 计算机科学与技术

班　　级 3 班

学生姓名 王红阳

实验名称 八数码求解

成　　绩

# 一、实验内容：使用 A 算法求解八数码问题

## 1. 八数码问题介绍：

在一个 3 X 3 的九宫中有 1 - 8 这 8 个数字以及一个空格随机摆放在其中的格子里。将该九宫格调整到目标状态。

规则：每次只能将与空格（上、下、左、右）相邻的一个数字移动到空格中。试编程实现这一问题的求解。

备注：为了程序中表示方便，可用 0 代替空格。

## 2. A 算法

在搜索的每步利用评估函数 f(n)=g(n)+h(n)，从根结点开始对其子结点计算评估函数，按函数值大小，选取小者向下扩展，直到最后得到目标结点，此种搜索算法称为 A 算法

由于评估函数中带有问题自身的启发性信息，因此 A 算法是一种启发式搜索算法

## 3. 评估函数

评估函数 f(n) 定义为从初始结点 S0 经过结点 n 到达目标结点的最小代价路径的代价评估值，它形式为： f(n) = g(n) + h(n)

g(n) 为初始结点 S0 到结点 n 是已实际付出的代价；

h(n) 是从结点 n 到目标结点 SB 最优路径的估计代价，而搜索的启发式信息主要由 h(n) 决定，在本题中老师所给的 h(n) 为不在位数码个数。

也可定义 h(n)为错位的牌必须要移动的距离之和。该方法需表示出某个特定元素从棋盘的某位置到其最终位置的距离，我将它们存入 position_to_position 列表中。这样的 h(n)启发效果更好。

## 二、实验原理与步骤

### 1. A 算法实现步骤

①将初始节点装入 OPEN 表；

②如果 OPEN 表为空，则失败，退出；否则，取出 OPEN 表 中第一个节点，加入到 CLOSE 表中；

③如果节点是目标节点，则成功，退出；

④如果节点可扩展,将节点的扩展节点加入到 OPEN 表中，将 OPEN 表按照估价函数由小到大排列；否则跳转第 2 步。

注：去除重复节点

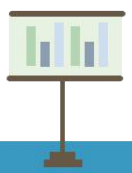
### 2. 伪代码

我最开始的实验没有使用伪代码，导致的结果就是逻辑不清，找不出自己的 bug 在哪里，程序陷入死循环。（由于节点数太多了，难以在 debug 环节找到自己的问题）

因此，我决定先把伪代码写出来，再将其改写为真实代码，实验证明这是相当有效的。几乎没有出现那样死循环的情况。代码很快搞好了。

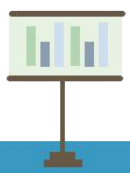
结合八数码问题性质，书写伪代码如下：





```
Input start , goal
open = [start];closed=[ ],f(s) = g(s) + h(s)
While open != [ ] do:
    从 open 表中删除第一个节点，称其为 n
    If n = goal
        return(success)
    生成 n 的所有子节点
    If n 没有任何子节点
        continue
    For n 的每个子节点 do
        If 子节点 is already on open 表 or closed 表
            计算该子节点的估值函数值
            将该子节点加入 open 表中
        If 子节点已经在 open 表中
            If 子节点是沿着一条比 open 表已有的更短路径而到达
                记录更短路径及其估价函数值
        将 n 放入 closed 表中：
        根据估价函数值，从小到大重新排列 open 表；
//此时，open 表中已无节点
Return false
```





### 3. 去重：解决表中含有重复节点的问题

在实验初期，选择编程语言时，我选择了 python 语言，而不是 C 语言。主要是因为最近经常使用 python 语言，而且 python 语言中也有很多已经封装好的函数。

比如，我把棋盘（节点）定义为字符串，用一维来模拟二维。把节点放入表中，只需 current_node not in openlist 语句，即可判断是否当前节点是否在表中，从而实现去重，防止因为不停地走重复节点的死循环情况。

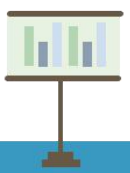
### 4. 无解情况

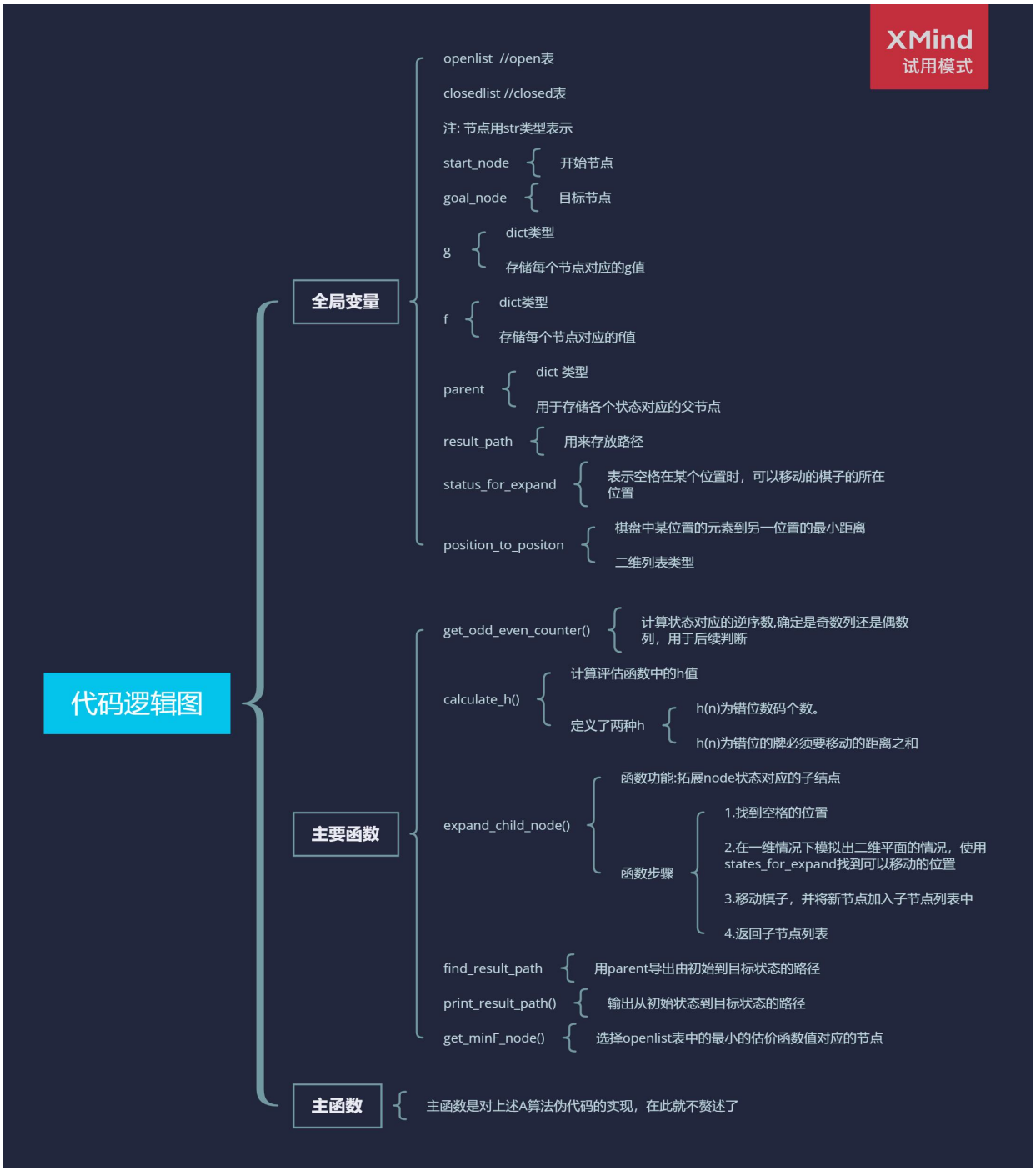
并不是所有情况都是有解的。

由数学知识可知每移动一次棋子，该棋盘的逆序数奇偶性都不会变。也就是说，如果初始节点与目标节点的逆序数的奇偶状态不一致，那么怎么移动都没有用。所以，在正式使用 A 算法解决该问题前需先判断逆序数的情况，如果相同才能继续运算；否则直接退出程序

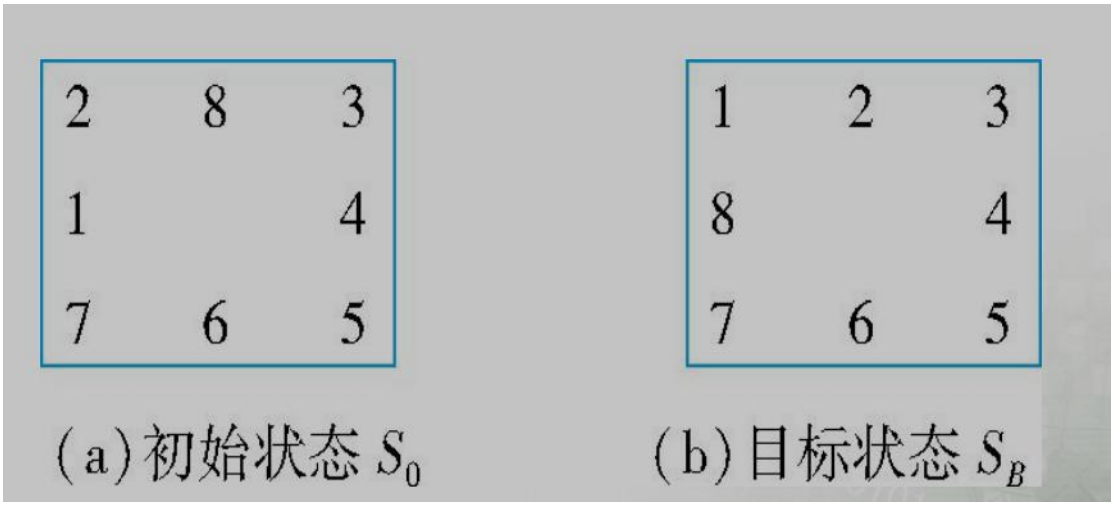
### 5. 代码逻辑图






XMind
试用模式
代码逻辑图
全局变量
openlist //open表
closedlist //closed表
注: 节点用str类型表示
start_node
开始节点
goal_node
目标节点
g
dict类型
存储每个节点对应的g值
f
dict类型
存储每个节点对应的f值
parent
dict 类型
用于存储各个状态对应的父节点
result_path
用来存放路径
status_for_expand
表示空格在某个位置时，可以移动的棋子的所在位置
position_to_positon
棋盘中某位置的元素到另一位置的最小距离
二维列表类型
主要函数
get_odd_even_counter()
计算状态对应的逆序数,确定是奇数列还是偶数列，用于后续判断
calculate_h()
计算评估函数中的h值
定义了两种h
h(n)为错位数码个数。
h(n)为错位的牌必须要移动的距离之和
expand_child_node()
函数功能:拓展node状态对应的子结点
函数步骤
1.找到空格的位置
2.在一维情况下模拟出二维平面的情况，使用states_for_expand找到可以移动的位置
3.移动棋子，并将新节点加入子节点列表中
4.返回子节点列表
find_result_path
用parent导出由初始到目标状态的路径
print_result_path()
输出从初始状态到目标状态的路径
get_minF_node()
选择openlist表中的最小的估价函数值对应的节点
主函数
主函数是对上述A算法伪代码的实现，在此就不赘述了

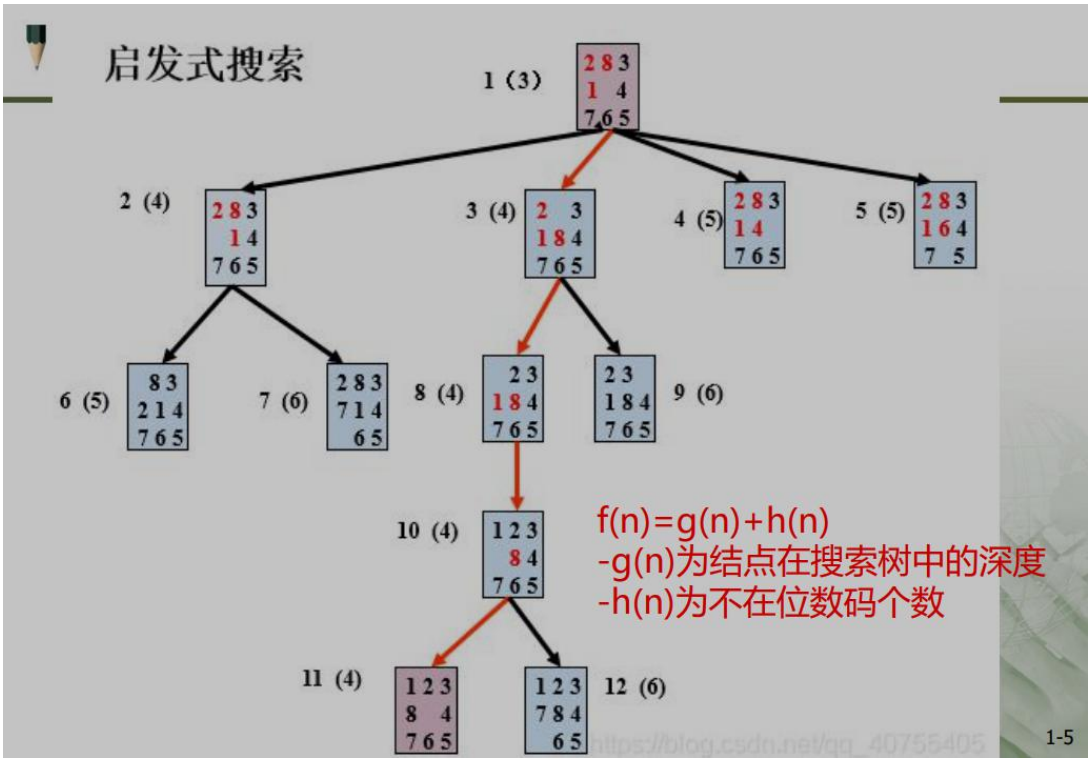
# 三、实验结果与分析




在本实验中假定输入数据符合要求，均为正常的八数码棋盘状态。将棋盘以字符串输入，即上图中的 S0 可表示为：283104765，目标状态可表示为：123804765

实验结果如下：


上述移动方式，符合老师所给的搜索情况。




至此，本次实验到此结束。

教师签字：

年　月　日