

**《人工智能导论》大作业**



**题 目：**  **A\*算法求解八数码问题**

**姓 名：**

**学 号：**

**班 级：**

**专 业：**

**学 院：**

# 研究背景和目的

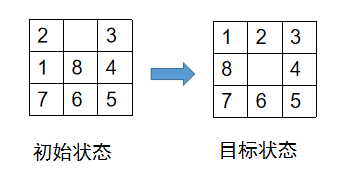
八数码问题是计算机科学中经典的搜索问题之一，它是一种拼图游戏，包括一个3x3矩阵和8个方块，每个方块上面有一个数字（1-8），另外一个空格可以与相邻的方块进行交换移动。目标是将所有方块移动到正确的位置以达到特定目标状态，解决八数码问题的主要目的在于探索搜索算法的应用，特别是启发式搜索算法。通过设计合适的搜索算法来求解八数码问题，可以有效提高搜索效率，并应用到更为复杂的问题中，例如路径规划、机器人导航、数据挖掘等领域。

此外，八数码问题还具有广泛的教育意义，它可以帮助人们训练逻辑思维和问题求解能力。其简单的规则和易于理解的概念使得它成为了人们常用的思维训练工具之一。因此，对于八数码问题的研究可以帮助我们更好地理解搜索算法的设计和应用，并为其他相关领域的研究提供有益的参考。

# 问题描述及分析

## 问题描述

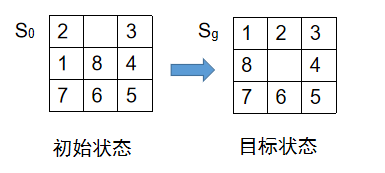
3х3九宫棋盘，放置数码为1~8的8个棋子，棋盘中留有一个空格，空格周围的棋子可以移动到空格中，从而改变棋盘的布局。

根据给定初始布局和目标布局，如何移动棋子才能从初始布局到达目标布局，找到一种最少步骤的移动方法。

## 问题分析

对于八数码问题，我们需要先判断其是否是可解问题，判断的方法如下所述：

将状态表示成一维形式后，计算除空格0之外所有数字的逆序数之和。逆序数是指每个数字前面比它大的数字的个数，将这些逆序数相加得到状态的逆序数。如果两个状态的逆序奇偶性相同，则这两个状态可以相互到达，否则不可相互到达。这是八数码问题的可解性判定结论。

 【思路1】采用BFS搜索，一边搜索结点一边生成新的子结点。将新生成的结点放入队列queue中，将访问过的结点放入集合set中。当某结点与目标状态相同时，那么最短路径长度就是该结点的长度。而最短路径可以通过存储父节点信息已经当前的操作算子得出。有两个关键点：如何计算BFS中结点的层数（进而得出最短路径长度），如何记录最短路径。

【思路2】启发式算法是在盲目搜索算法的改进，通过评价函数f(n) = g(n) + h(n)为每个状态打分，通过放入优先队列中，即得分低的状态结点优先访问。其他地方处理相同。其中g(n)是指到达这一状态的代价(即层数)，h(n)是指棋盘上与目标状态不同的棋子数目。

# 总体设计

## 输入输出设计

输入：初始状态和目标状态。

输出：从初始状态到目标状态的最短路径。

## 数据结构设计

状态：表示当前盘面的状态，可以使用一个包含九个数字的列表来表示。

节点：表示在搜索过程中的一个状态节点，包括当前状态、父节点、移动方式、路径成本等信息。

列表：包括开启列表和关闭列表，分别用于存储待扩展状态和已经扩展的状态。

其他辅助数据结构：如优先队列、哈希表等。

## 算法设计

启发式搜索算法：采用启发式函数计算估价值，以此评估当前状态的优先级。在搜索过程中，优先选取估价值最小的状态扩展。

A\*算法：结合启发式搜索和广度优先搜索的优点，以启发式函数为基础，选择最具优势的动作的下一步状态进行探索，同时考虑已探索的路径的总代价，以此指导搜索方向。

## 程序架构设计

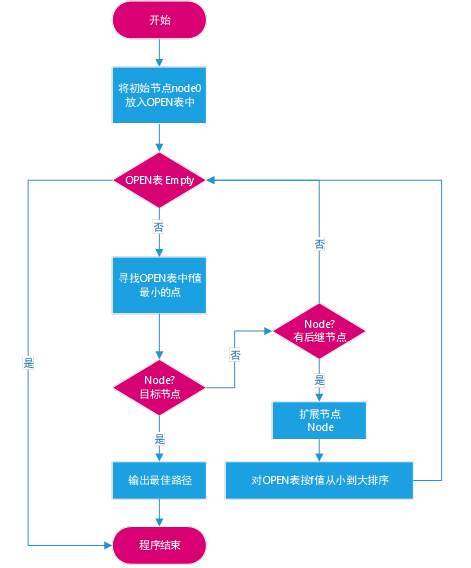
设计程序的主要模块：初始化、状态扩展、启发式函数计算、A\*算法流程等。

初始化：在该部分中，将初始状态保存至开启列表中，并设置结束状态和空格所在位置。

状态扩展：在该部分中，针对当前状态，生成其所有可能的下一步动作并计算各个状态的估价值。

启发式函数计算：在该部分中，利用估价函数计算当前状态到目标状态的估价值，这里采用曼哈顿距离作为启发函数。

A\*算法流程：综合以上三个部分，利用A\*算法进行搜索操作，找到最佳路径。程序的流程图如下：



## 用户接口设计

设计用户界面：提供简洁明了的交互界面，方便用户输入初始状态和目标状态，同时显示搜索结果。

用户体验：考虑用户使用体验，尽量减少不必要的操作和信息提示，提高用户满意度。

总之，针对八数码问题的总体设计需要充分考虑算法复杂度、程序稳定性和用户体验等方面的因素，以确保程序的正确性和易用性。

# 详细设计

## 输入输出设计

输入：初始状态和目标状态。

输出：从初始状态到目标状态的最短路径。

## 数据结构设计

* 状态表示:

需要定义一个合适的数据结构来表示八数码问题中的状态，通过一个9元素的列表来表示当前节点的状态，将空格用0来表示，例如：

[1, 2, 3,

4, 5, 6,

7, 8, 0]

* 节点表示:

节点表示一个状态，其中包括以下信息：当前状态、父节点、移动方式（由父节点到该节点）、路径成本（从初始状态到当前节点的路径成本）

* 列表表示

开启列表（存储待扩展状态节点）、关闭列表（存储已扩展状态节点）、优先队列（根据启发式函数和路径成本决定每个节点的扩展顺序）

* 其他辅助数据结构

哈希表（用于快速检查状态是否已经存在于关闭列表或开启列表中）

## 算法设计

* 启发式函数设计

使用曼哈顿距离作为启发函数评估状态节点优先级/估价值。曼哈顿距离指定状态与目标状态中各数字的位置差之和，例如：

对于状态：[1, 2, 3, 4, 5, 6, 7, 8, 0]

目标状态：[3, 1, 2, 4, 5, 6, 7, 8, 0]

计算每个数字到其目标位置的曼哈顿距离：

数字 1：2（目标位置为(0,0)，当前位置为(0,2)，曼哈顿距离=2）

数字 2：1（目标位置为(0,1)，当前位置为(0,1)，曼哈顿距离=1）

数字 3：2（目标位置为(0,2)，当前位置为(0,0)，曼哈顿距离=2）

数字 4：1（目标位置为(1,0)，当前位置为(1,0)，曼哈顿距离=1）

数字 5：0

数字 6：0

数字 7：0

数字 8：0

数字 0：0

因此该状态的曼哈顿距离为2+1+2+1=6。

* A\*搜索算法设计

使用A\*算法进行搜索过程，以启发式函数和路径成本决定每个节点的扩展顺序。

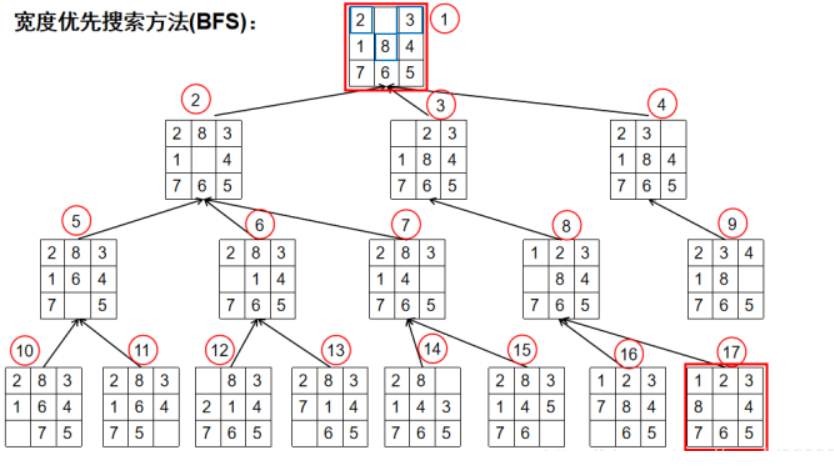
具体流程如下：

将初始状态保存至开启列表中，并设置结束状态和空格所在位置。

对于开启列表中的每个状态节点，按照估价值和路径成本的综合排序，并取出最优状态进行扩展操作，同时将该状态节点添加到关闭列表中。

对于当前状态节点，生成它所有可能的下一步状态，通过计算各个状态的估价值和路径成本，更新状态节点信息，并将其添加到开启列表中（前提是该状态没有出现在开启列表或关闭列表中）。

重复2-3步骤，直到找到目标状态。



此算法过程如下：

1. 从起始状态A开始，并且把它作为待处理点存入一个“开启列表”。开启列表就像一张购物清单。尽管现在列表里只有一个元素，但以后就会多起来。你的路径可能会通过它包含的其他状态，也可能不会。基本上，这是一个待检查方格的列表。

2. 寻找起点周围通过上下左右移动所有可到达的状态。也把他们加入开启列表。为所有这些状态保存状态A作为“父方格”。当我们想描述路径的时候，父方格的资料是十分重要的。后面会解释它的具体用途。

3.从开启列表中删除状态A，把它加入到一个“关闭列表”。

4.取出开启列表中fvalue最低的状态，判断该状态性质

(1) 如果在开启列表中出现，那么判断该状态的gvalue通过当前节点（从open中取出，扩展出该节点的节点）作为父节点带来的gvalue 是否小于原先的父节点，若小于则需要更改它的父节点以及gvalue.

(2) 如果在关闭列表中出现，那么判断该状态的gvalue通过当前节点（从open中取出，扩展出该节点的节点）作为父节点带来的gvalue是否小于原先的父节，若小于则需要从关闭列表中删除该节点，并且将其加入开启列表，并更改其父节点为当前节点。

(3) 既不在开启列表，也不在关闭列表，这样就很简单，直接加入开启列表中就行（保持开启列表升序排列）

(4) 如果取出的节点为目标节点，则成功退出

## 程序架构设计

* 初始化模块

将初始状态保存至开启列表中

设置目标状态

找到空格所在位置

* 状态扩展模块

遍历当前状态列表，找到空格0所在位置，然后分别与其相邻的四个方向进行交换，产生四个新状态。例如：

[2, 0, 3, 1, 5, 6, 4, 7, 8] -> [0, 2, 3, 1, 5, 6, 4, 7, 8] # 上移操作

对于每个新状态，检查其是否已经出现在开启列表或关闭列表中，如果不存在，则计算其估价值和路径成本，并将其加入优先队列中。

* 启发式函数计算模块

使用曼哈顿距离计算状态的估价值。

* A\*算法流程

综合以上三个模块，使用A\*搜索算法进行搜索操作，找到最佳路径。具体流程如下：

初始化：将初始状态保存至开启列表中，并设置结束状态和空格所在位置。

在每次循环中，取出开启列表中估价值加路径成本最小的节点进行扩展操作，并将该节点添加到关闭列表中。

对于当前状态节点，生成它所有可能的下一步状态，通过计算各个状态的估价值和路径成本，更新状态节点信息，并将其添加到开启列表中（前提是该状态没有出现在开启列表或关闭列表中）。

重复2-3步骤，直到找到目标状态。

在找到目标状态后，通过遍历父节点恢复完整的路径。

## 用户接口设计

提供简洁明了的交互界面，方便用户输入初始状态和目标状态，同时显示搜索结果。设计用户体验，尽量减少不必要的操作和信息提示，提高用户满意度。

以上是一个针对八数码问题的详细设计大纲，需要注意的是，实际实现中还需要处理各种边界情况、输入格式验证、错误处理等其他细节问题，以确保程序的正确性和稳定性。

# 编码实现

## 定义节点类

class Node:

count = 0

# node为八码数的list表示，parent为父节点

# cost为节点代价

# nd\_nums为不在位数

def \_\_init\_\_(self,node,parent,cost,nd\_nums):

self.node=node

self.parent=parent

self.cost=cost

self.nd\_nums=nd\_nums

Node.count += 1

def get\_children(self): # 获取一层子节点

children = []

for i in Nodes\_change(self.node):

# 逐个元素与0交换位置,生成子节点child

child = Node(node=get\_child(self.node, i), parent=self, cost=self.cost + 1,

nd\_nums=not\_digits(self.node, end\_arr))

# 将每一个交换结果（子节点）都存入children

children.append(child)

return children

## 计算逆序数

def AStar\_ND(initial\_arr, end\_arr):

open = [initial\_arr]

close = []

while len(open) > 0: # OPEN表是否为空表

open\_1 = [i.node for i in open] # 访问open节点内的node

close\_1 = [i.node for i in close]

n = open.pop(0) # 删除OPEN队头节点

close.append(n) # n注入CLOSE表

if n.node == end\_arr:

print('最优路径如下：')

print(np.array(n.node,dtype=object))

best\_path(n)

break

else:

for i in n.get\_children(): # 添加子节点进OPEN

if i.node not in open\_1:

if i.node not in close\_1:

open.insert(0, i)

open.sort(key=lambda x: x.nd\_nums + x.cost) # 按不在位数＋cost 进行排序

print('--' \* 20)

print('--' \* 20)

print('--' \* 20)

search\_line(close)

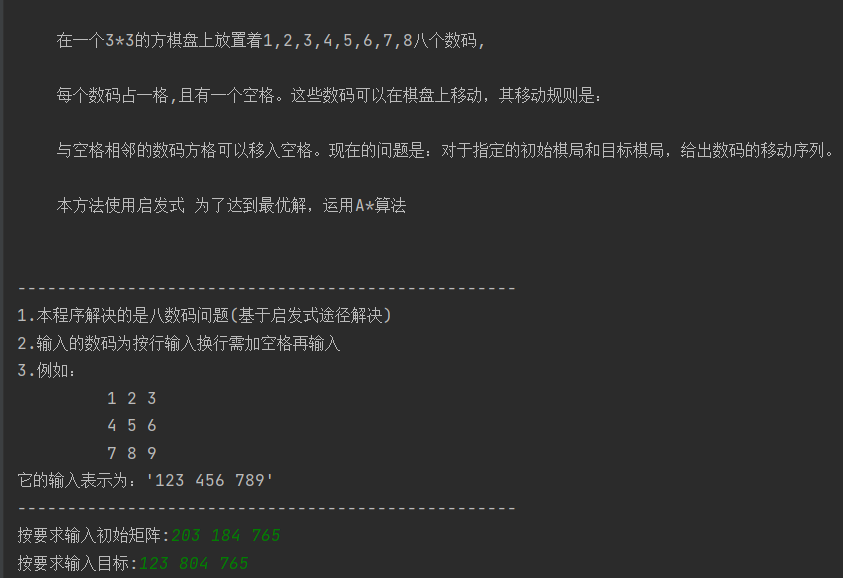
print('搜索步骤为：', len(close) - 1)

print('节点数为：', n.count)

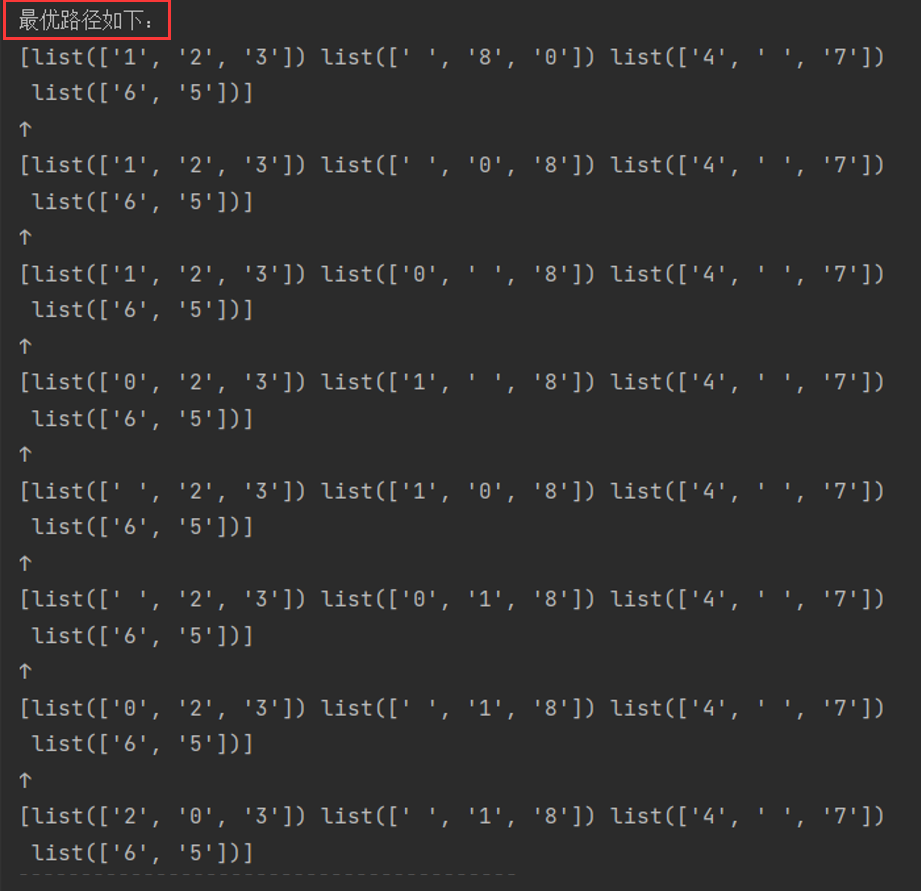
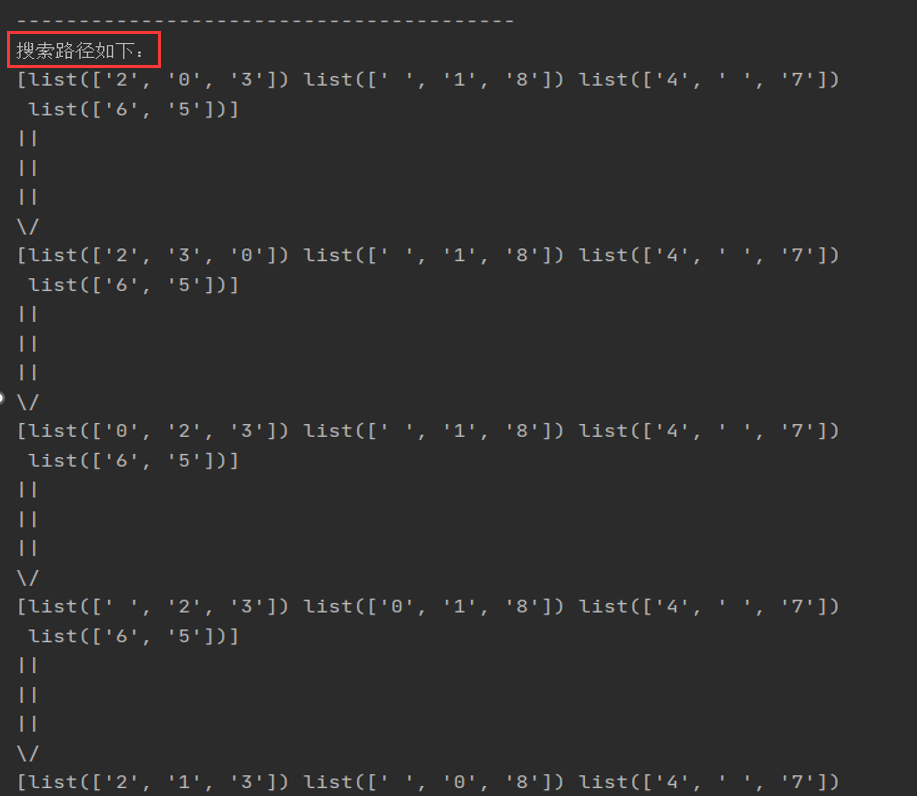
# 结果展示

## 输入：

输入序列为：203 184 765

目的序列为：123 804 765

## 输出：

 输出最优路径以及搜索路径：

# 总结

## 体会：

1、首先，本作品的设计在初始页面方面过于简单，仅仅着重于问题解决而忽略了交互性和互操作性。

2、在计算较为复杂的序列时，仍需花费较长时间，因此在一些特殊值上缺乏良好的匹配和适应算法，算法存在局限性，缺乏弹性。

3、程序设计逻辑不够清晰，存在模块之间的耦合度较高，代码较为冗余，缺少精简性。

## 课程建议

我认为这门人工智能导论课程非常有价值，对于进一步深入了解人工智能领域具有重要意义。以下是我的一些建议：

加强理论基础：人工智能是一门高度理论化的学科，需要建立坚实的数学和计算机基础。可以在课堂教学中增加相关的数学、概率论、统计学、计算机科学等基础知识的讲解。

拓宽应用广度：人工智能涉及到多个领域，如图像处理、自然语言处理、智能控制等。在课程安排上，建议涵盖更广泛的应用领域，引导学生关注应用前沿和未来趋势。

增加实践环节：人工智能是一个高度实践性的学科，在理论学习的同时，需要进行实际的操作和实验。建议在授课中加入实践环节，例如编写程序、实现算法等。

丰富教学资料：人工智能是一个快速发展的领域，新技术、新成果层出不穷。建议老师提供丰富的教学资料，引导学生了解最新的人工智能进展和趋势，并指导学生进行相关研究。

培养思考和创新能力：人工智能是一个需要创新思维的领域，建议课程中注重培养学生的思考和创新能力，引导学生深入思考问题，寻找创新解决方案。

总而言之，这门人工智能导论课程是非常有价值的，但也有一些可以改进的地方。通过加强理论基础、拓宽应用广度、增加实践环节、丰富教学资料以及培养思考和创新能力等措施，可以更好地为学生提供全面、系统和有针对性的人工智能教育。