北 京 航 空 航 天 大 学

# 模式识别实验报告（三）

## 图搜索问题求解


专业名称 自动化

专业方向 自动控制与模式识别

班级学号 16711094

学生姓名 李翰韬

指导教师 王 田

2019 年 11 月 18 日

# 一、实验目的

1、使学生加深对图搜索技术的理解

2、掌握图搜索基本编程方法

3、运用图搜索技术解决一些应用问题

# 二、实验要求

1、用广度优先搜索算法实现八数码问题。

2、编程语言不限。

3、程序运行时，应能清晰直观演示搜索过程。

# 三、实验内容

(一) 原理分析

广度优先搜索算法分析:

步 1 把初始节点 $S_0$ 放入 OPEN 表中。

步 2 若 *OPEN* 表为空，则搜索失败,退出。

步 3 取 *OPEN* 表中前面第一个节点 *N* 放在 CLOSED 表中，并冠以顺序编号 n。

步 4 若目标节点 $S_g$=N,则搜索成功，结束。

步 5 若 *N* 不可扩展，则转步 2。

步 6 扩展 *N*，将其所有子节点配上指向 *N* 的指针依次放入 OPEN 表尾部，转步 2。

(二) 实验内容

在 3*3 的方格棋盘上，分别放置了标有数字 1、2、3、4、5、6、7、8 的八张牌，初始状态 S，目标状态 Sg，如下图所示。可以使用的操作有:

**空格左移，空格上移，空相右移，空格下移**

即只允许把位于空格左、上、右、下方的牌移入空格。要求应用广度优先搜索策略寻找从初始状态到目标状态的解路径。

## 四、实验步骤

具体工作及步骤为：

1、设计问题的知识表示方法：矩阵表示法，或向量表示法

| | | |
|---|---|---|
| $X_1$ | $X_2$ | $X_3$ |
| $X_8$ | $X_0$ | $X_4$ |
| $X_7$ | $X_6$ | $X_5$ |

我们将棋局用向量

$$A = (X_0, X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8)$$

表示，$X_i$为变量，$X_i$的值就是所在方格内的数字。于是，向量 $A$ 就是该问题的状态表达式。

设初始状态和目标状态分别为

$$S_0 = (0, 2, 8, 3, 4, 5, 6, 7, 1)$$

$$S_g = (0, 1, 2, 3, 4, 5, 6, 7, 8)$$

2、根据相应的知识表示方法，表示数码移动规则；

0 组规则：

$$r_1(X_0 = 0)\Lambda(X_2 = n) \to X_0 \Leftarrow n\Lambda X_2 \Leftarrow 0;$$
$$r_2(X_0 = 0)\Lambda(X_4 = n) \to X_0 \Leftarrow n\Lambda X_4 \Leftarrow 0;$$
$$r_3(X_0 = 0)\Lambda(X_6 = n) \to X_0 \Leftarrow n\Lambda X_6 \Leftarrow 0;$$
$$r_4(X_0 = 0)\Lambda(X_8 = n) \to X_0 \Leftarrow n\Lambda X_8 \Leftarrow 0;$$

1 组规则：

$$r_5(X_1 = 0)\Lambda(X_2 = n) \to X_1 \Leftarrow n\Lambda X_2 \Leftarrow 0;$$
$$r_6(X_1 = 0)\Lambda(X_8 = n) \to X_1 \Leftarrow n\Lambda X_8 \Leftarrow 0;$$

2 组规则：

$$r_7(X_2=0)\Lambda(X_1=n)\rightarrow X_2\Leftarrow n\Lambda X_1\Leftarrow 0;$$
$$r_8(X_2=0)\Lambda(X_3=n)\rightarrow X_2\Leftarrow n\Lambda X_3\Leftarrow 0;$$
$$r_9(X_2=0)\Lambda(X_0=n)\rightarrow X_2\Leftarrow n\Lambda X_0\Leftarrow 0;$$
$$\vdots$$

8 组规则：

$$r_{22}(X_8=0)\Lambda(X_1=n)\rightarrow X_8\Leftarrow n\Lambda X_1\Leftarrow 0;$$
$$r_{23}(X_8=0)\Lambda(X_0=n)\rightarrow X_8\Leftarrow n\Lambda X_0\Leftarrow 0;$$
$$r_{24}(X_8=0)\Lambda(X_7=n)\rightarrow X_8\Leftarrow n\Lambda X_7\Leftarrow 0;$$

3、实现搜索方法；

4、系统调试与测试。

## 五、实验结果

使用 python 编写程序，最终程序可以实现实验目标，以下为操作步骤：

1、输入原始格局与目标格局，顺序为从左至右，从上至下。

```
"123405678"#输入原始格局
"123804567"#输入目标格局
```

2、能够自行求初始格局和目标格局逆序数，然后在比较两者的逆序数的奇偶性是否相同，判断两个格局是否可达。若目标格局不可达，则输出提示。

3、若目标格局可达，则计算后用图形表示出路径结果。

```
Step Number:14
-------------
| 1 | 2 | 3 |
-------------
| 8 | 6 | 4 |
-------------
| 5 | 0 | 7 |
-------------

Step Number:15
-------------
| 1 | 2 | 3 |
-------------
| 8 | 0 | 4 |
-------------
| 5 | 6 | 7 |
-------------

======================
本次求解共15步
```

# 附录：程序代码

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Mon Nov 18 14:23:51 2019

@author: hantao.li
"""

g_dict_layouts = {}
g_dict_shifts = {0:[1, 3], 1:[0, 2, 4], 2:[1, 5],
                 3:[0,4,6],          4:[1,3,5,7],
5:[2,4,8],
                 6:[3,7],    7:[4,6,8], 8:[5,7]}
#----------------------------------------------定义九宫格中每一个方格可以移动的位置

def swap_chr(a, i, j):
    if i > j:
        i, j = j, i
    b = a[:i] + a[j] + a[i+1:j] + a[i] + a[j+1:]
    return b
#----------------------------------------------两个方格交换后的数组

def solvePuzzle_depth(srcLayout, destLayout):
    src=0;dest=0
    for i in range(1,9):
        fist=0
        for j in range(0,i):
            if  srcLayout[j]>srcLayout[i]  and
srcLayout[i]!='0':
                fist=fist+1
        src=src+fist

    for i in range(1,9):
        fist=0
        for j in range(0,i):
            if destLayout[j]>destLayout[i] and
destLayout[i]!='0':
                fist=fist+1
        dest=dest+fist
    if (src%2)!=(dest%2):
        return -1, None
#----------------------------------------------求初始格局和目标格局逆序数，然后在比较两者的逆序数的奇偶性是否相同
    g_dict_layouts[srcLayout] = -1
    stack_layouts = []
    stack_layouts.append(srcLayout)

    while len(stack_layouts) > 0:
        curLayout = stack_layouts.pop(0)
        if curLayout == destLayout:
            break
#----------------------------------------------判断当前格局是否为目标格局
```

```
            ind_slide = curLayout.index("0")
            lst_shifts = g_dict_shifts[ind_slide]
#----------------------------------------------找到能交换的方格
            for nShift in lst_shifts:
                newLayout = swap_chr(curLayout, nShift, ind_slide)

                if g_dict_layouts.get(newLayout) == None:# 判断交换后的状态是否已经查询过
                    g_dict_layouts[newLayout] = curLayout

                    stack_layouts.append(newLayout)#存入集合

        lst_steps = []
        lst_steps.append(curLayout)
        while g_dict_layouts[curLayout] != -1:
            curLayout = g_dict_layouts[curLayout]
            lst_steps.append(curLayout)
#----------------------------------------------将结果存入路径
        lst_steps.reverse()
        return 0, lst_steps



if __name__ == "__main__":
    srcLayout  = "123405678"# 输入原始格局
    destLayout = "123804567"# 输入目标格局

    retCode, lst_steps = solvePuzzle_depth(srcLayout, destLayout)
    if retCode != 0:
        print("目标布局不可达")
    else:
        for nIndex in range(len(lst_steps)):
            print("Step Number:" + str(nIndex + 1))
            print('-------------')
            print('| '+lst_steps[nIndex][0]+' | '+lst_steps[nIndex][1]+' | '+lst_steps[nIndex][2]+' |')
            print('-------------')
            print('| '+lst_steps[nIndex][3]+' | '+lst_steps[nIndex][4]+' | '+lst_steps[nIndex][5]+' |')
            print('-------------')
            print('| '+lst_steps[nIndex][6]+' | '+lst_steps[nIndex][7]+' | '+lst_steps[nIndex][8]+' |')
            print('-------------\n')

print('======================\n 本次求解共'+str(nIndex+1)+'步')
```