**深度优先搜索解决八数码问题**

**臧泽林**

# 代码块1

# -\*- coding: utf-8 -\*-

"""

Created on Sun May 21 10:15:39 2017

本程序用于解决，人工智能（王万良）书中深度优先书中的P166也的八数码问题，使用深度优

先搜索策略

@author: jon zang，杭州，15168307480，zangzelin@gmail.com

"""

import numpy as np

import matplotlib.pyplot as plt

import random as rd

from scipy.special import comb, perm

import copy

import winsound

class Point(object):#搜索树的节点定义为类

def \_\_init\_\_(self,state,father,son = 0,neighbors = 0):

self.state = state

self.father = father

self.son = son

self.neighbors = neighbors

# def CreatPoint(self,state)

def ShowThisPoint(self):

print(self.state,self.father,self.son,self.neighbors)

# 移动空间子函数，用于对八数码问题的数字进行操作

def MoveTheNumber(direction, a):

state\_of\_current = copy.deepcopy(a.state)#获得当前状态的复制体，方便进行修改和变换

index\_of\_zero = np.zeros(2)#初始化检索矩阵

for i in range(3):#检索到零的位置

for j in range(3):

if state\_of\_current[i,j] == 0:

index\_of\_zero = [i,j]

break

if direction == 1:#根据输入的移动动作制作索引

a = index\_of\_zero[0] - 1#上

b = index\_of\_zero[1]

elif direction == 2:

a = index\_of\_zero[0] + 1#下

b = index\_of\_zero[1]

elif direction == 3:#左

a = index\_of\_zero[0]

b = index\_of\_zero[1] - 1

elif direction == 4:#右

a = index\_of\_zero[0]

b = index\_of\_zero[1] + 1

if a >= 0 and a <= 2 and b >= 0 and b <= 2 : #判断索引是否合理， 合理则进行交换

state\_of\_current[index\_of\_zero[0],index\_of\_zero[1]] = state\_of\_current[a,b]

state\_of\_current[a,b] = 0

else:

state\_of\_current=np.array([[-1, -1, -1],#不合理则输出错误

[-1, -1, -1],

[-1, -1, -1]])

return state\_of\_current

def GetSon(fatherpoint): #制作子节点的方法，输入父节点，制作其所有的子节点

son1 = Point(MoveTheNumber(1,fatherpoint),fatherpoint)

son2 = Point(MoveTheNumber(2,fatherpoint),fatherpoint)

son3 = Point(MoveTheNumber(3,fatherpoint),fatherpoint)

son4 = Point(MoveTheNumber(4,fatherpoint),fatherpoint)

son1.neighbors = son2

son2.neighbors = son3

son3.neighbors = son4

son4.neighbors = 0

fatherpoint.son = son1

sonlist = [son1,son2,son3,son4]

return sonlist

def issame(a,b):#判断两个状态是否相同的方法

count = 0

for i in range(3):

for j in range(3):

if a[i,j] == b[i,j]:

count = count+1

if count == 9:

out= 1

else:

out = 0

return out

def findpoint(point,deep,maxdeep,state\_of\_end):#递归思想完成的寻找点的方法

p = copy.deepcopy(point)

if p.state[0,1] != -1:

#print(p.state)

if issame(p.state,state\_of\_end ):

print('已经找到最优路径，路径长度为',deep,'，从最低端到最顶端打印如下：')

print(p.state)

while deep > 0:

p = p.father

print('第',deep,'步：转化为')

print(p.state)

deep = deep -1

elif deep < maxdeep:

sonlist = GetSon(p)

findpoint(sonlist[0],deep+1,maxdeep,state\_of\_end)

findpoint(sonlist[1],deep+1,maxdeep,state\_of\_end)

findpoint(sonlist[2],deep+1,maxdeep,state\_of\_end)

findpoint(sonlist[3],deep+1,maxdeep,state\_of\_end)

else:

return 0

state\_of\_start = np.array([[2, 0, 8],#设置开始状态

[1, 6, 3],

[7, 5, 4]])

state\_of\_end = np.array([[1, 2, 3],#设置结束状态

[8, 0, 4],

[7, 6, 5]])

p1 = Point(state\_of\_start,0)#初始化开始节点

deep = 0#初始化节点深度

maxdeep = 9#初始化最大节点深度

findpoint(p1,deep,maxdeep,state\_of\_end)