八数码

\_\_author\_\_ = 'ysc'  
import numpy as np  
class State:  
 def \_\_init\_\_(self, state, directionFlag=None, parent=None):  
 self.state = state  
 self.direction = ['up', 'down', 'right', 'left']  
 if directionFlag:  
 self.direction.remove(directionFlag)  
 self.parent = parent  
 self.symbol = ' '  
 def getDirection(self):  
 return self.direction  
 def showInfo(self):  
 for i in range(3):  
 for j in range(3):  
 print(self.state[i, j], end=' ')  
 print("\n")  
 print('->')  
 return  
 def getEmptyPos(self):  
 postion = np.where(self.state == self.symbol)  
 return postion  
 def generateSubStates(self):  
 if not self.direction:  
 return []  
 subStates = []  
 boarder = len(self.state) - 1  
 row, col = self.getEmptyPos()  
 if 'left' in self.direction and col > 0:  
 s = self.state.copy()  
 temp = s.copy()  
 s[row, col] = s[row, col-1]  
 s[row, col-1] = temp[row, col]  
 news = State(s, directionFlag='right', parent=self)  
 subStates.append(news)  
 if 'up' in self.direction and row > 0:  
 s = self.state.copy()  
 temp = s.copy()  
 s[row, col] = s[row-1, col]  
 s[row-1, col] = temp[row, col]  
 news = State(s, directionFlag='down', parent=self)  
 subStates.append(news)  
 if 'down' in self.direction and row < boarder:  
 s = self.state.copy()  
 temp = s.copy()  
 s[row, col] = s[row+1, col]  
 s[row+1, col] = temp[row, col]  
 news = State(s, directionFlag='up', parent=self)  
 subStates.append(news)  
 if self.direction.count('right') and col < boarder:  
 s = self.state.copy()  
 temp = s.copy()  
 s[row, col] = s[row, col+1]  
 s[row, col+1] = temp[row, col]  
 news = State(s, directionFlag='left', parent=self)  
 subStates.append(news)  
 return subStates  
 def solve(self):  
 openTable = []  
 closeTable = []  
 openTable.append(self)  
 steps = 1  
 while len(openTable) > 0:  
 n = openTable.pop(0)  
 closeTable.append(n)  
 subStates = n.generateSubStates()  
 path = []  
 for s in subStates:  
 if (s.state == s.answer).all():  
 while s.parent and s.parent != originState:  
 path.append(s.parent)  
 s = s.parent  
 path.reverse()  
 return path, steps+1  
 openTable.extend(subStates)  
 steps += 1  
 else:  
 return None, None  
if \_\_name\_\_ == '\_\_main\_\_':  
 symbolOfEmpty = ' '  
 State.symbol = symbolOfEmpty  
 originState = State(np.array([[2, 8, 3], [1, 6 , 4], [7, symbolOfEmpty, 5]]))  
 State.answer = np.array([[1, 2, 3], [8, State.symbol, 4], [7, 6, 5]])  
 s1 = State(state=originState.state)  
 path, steps = s1.solve()  
 if path:  
 for node in path:  
 node.showInfo()  
 print(State.answer)  
 print("Total steps is %d" % steps)