IDS算法：

import pandas as pd  
from pandas import Series, DataFrame  
# 城市信息：city1 city2 path\_cost  
\_city\_info = None  
# 已探索集合  
\_explored = []  
  
# 节点数据结构  
class Node:  
 def \_\_init\_\_(self, state, parent, action, path\_cost):  
 self.state = state  
 self.parent = parent  
 self.action = action  
 self.path\_cost = path\_cost

def main():  
 global \_city\_info, \_explored  
 import\_city\_info()  
 while True:  
 src\_city = input('input src city:')  
 dst\_city = input('input dst city:')  
 result = iterative\_deepening\_search(src\_city, dst\_city)  
 if result == "failure" or result == "cutoff":  
 print('from city: %s to city %s search failure' % (src\_city, dst\_city))  
 else:  
 print('from city: %s to city %s search success' % (src\_city, dst\_city))  
 path = []  
 while True:  
 path.append(result.state)  
 if result.parent is None:  
 break  
 result = result.parent  
 size = len(path)  
 for i in range(size):  
 if i < size - 1:  
 print('%s->' % path.pop(), end='')  
 else:  
 print(path.pop())

def import\_city\_info():  
 global \_city\_info  
 data = [{'city1': 'Oradea', 'city2': 'Zerind', 'path\_cost': 71},  
 {'city1': 'Oradea', 'city2': 'Sibiu', 'path\_cost': 151},  
 {'city1': 'Zerind', 'city2': 'Arad', 'path\_cost': 75},  
 {'city1': 'Arad', 'city2': 'Sibiu', 'path\_cost': 140},  
 {'city1': 'Arad', 'city2': 'Timisoara', 'path\_cost': 118},  
 {'city1': 'Timisoara', 'city2': 'Lugoj', 'path\_cost': 111},  
 {'city1': 'Lugoj', 'city2': 'Mehadia', 'path\_cost': 70},  
 {'city1': 'Mehadia', 'city2': 'Drobeta', 'path\_cost': 75},  
 {'city1': 'Drobeta', 'city2': 'Craiova', 'path\_cost': 120},  
 {'city1': 'Sibiu', 'city2': 'Fagaras', 'path\_cost': 99},  
 {'city1': 'Sibiu', 'city2': 'Rimnicu Vilcea', 'path\_cost': 80},  
 {'city1': 'Rimnicu Vilcea', 'city2': 'Craiova', 'path\_cost': 146},  
 {'city1': 'Rimnicu Vilcea', 'city2': 'Pitesti', 'path\_cost': 97},  
 {'city1': 'Craiova', 'city2': 'Pitesti', 'path\_cost': 138},  
 {'city1': 'Fagaras', 'city2': 'Bucharest', 'path\_cost': 211},  
 {'city1': 'Pitesti', 'city2': 'Bucharest', 'path\_cost': 101},  
 {'city1': 'Bucharest', 'city2': 'Giurgiu', 'path\_cost': 90},  
 {'city1': 'Bucharest', 'city2': 'Urziceni', 'path\_cost': 85},  
 {'city1': 'Urziceni', 'city2': 'Vaslui', 'path\_cost': 142},  
 {'city1': 'Urziceni', 'city2': 'Hirsova', 'path\_cost': 98},  
 {'city1': 'Neamt', 'city2': 'Iasi', 'path\_cost': 87},  
 {'city1': 'Iasi', 'city2': 'Vaslui', 'path\_cost': 92},  
 {'city1': 'Hirsova', 'city2': 'Eforie', 'path\_cost': 86}]  
 \_city\_info = DataFrame(data, columns=['city1', 'city2', 'path\_cost'])  
 # print(\_city\_info)  
  
def iterative\_deepening\_search(src\_state, dst\_state):  
 for i in range(100):  
 result = depth\_limited\_search(src\_state, dst\_state, i)  
 if result != "failure" and result != "cutoff":  
 return result  
 return "cutoff"  
  
def depth\_limited\_search(src\_state, dst\_state, limit):  
 global \_explored  
 \_explored = []  
 node = Node(src\_state, None, None, 0)  
 return recursive\_dls(node, dst\_state, limit)  
  
def recursive\_dls(node, dst\_state, limit):  
 *"""* ***:param*** *node:* ***:param*** *dst\_state:* ***:param*** *limit:* ***:return****: "failure"：失败."cutoff"：被截至.node：成功  
 """* global \_city\_info, \_explored  
 if node.parent is not None:  
 print('node state:%s parent state:%s' % (node.state, node.parent.state))  
 else:  
 print('node state:%s parent state:%s' % (node.state, None))  
 \_explored.append(node.state)  
 # 目标测试  
 if node.state == dst\_state:  
 print('this node is goal!')  
 return node  
 elif limit == 0:  
 print('this node is cutoff!')  
 return "cutoff"  
 else:  
 cutoff\_occurred = False  
 # 遍历子节点  
 for i in range(len(\_city\_info)):  
 dst\_city = ''  
 if \_city\_info['city1'][i] == node.state:  
 dst\_city = \_city\_info['city2'][i]  
 elif \_city\_info['city2'][i] == node.state:  
 dst\_city = \_city\_info['city1'][i]  
 if dst\_city == '':  
 continue  
 child = Node(dst\_city, node, 'go', node.path\_cost + \_city\_info['path\_cost'][i])  
 # 过滤已探索的点  
 if child.state in \_explored:  
 continue  
 print('child node:state:%s path cost:%d' % (child.state, child.path\_cost))  
 result = recursive\_dls(child, dst\_state, limit - 1)  
 if result == "cutoff":  
 cutoff\_occurred = True  
 print('search failure, child state: %s parent state: %s limit cutoff' %  
 (child.state, child.parent.state))  
 elif result != "failure":  
 print('search success')  
 return result  
 if cutoff\_occurred:  
 return "cutoff"  
 else:  
 return "failure"

if \_\_name\_\_ == '\_\_main\_\_':  
 main()

运行结果：



