中山大学计算机学院 人工智能 本科生实验报告 Lab1

计算机科学与技术 21307289 刘森元

一、实验题目 最短路径搜索

罗马尼亚旅行问题

请基于最短路径程序，扩展实现一个搜索罗马尼亚城市间最短路径的导航程序，要求：

1. 出发城市和到达城市由用户在查询时输入
2. 对于城市名称，用户可以输入全称，也可以只输入首字母，且均不区分大小写
3. 向用户输出最短路径时，要输出途经的城市，以及路径总路程
4. 输出内容在直接反馈给用户的同时，还需追加写入一个文本文件中，作为记录日志
5. 为提升代码灵活性，你应在代码中合理引入函数和类（各定义至少一个）
6. 此外，将你定义的一些函数和类，存储在独立的模块文件中

二、实验内容 单源最短路径搜索

算法原理

Dijkstra

算法思想：从起点出发，使用贪心策略依次拓展节点，直到 无法再拓展/拓展到终点 为止

```
u_i = min(D[u_i])
S += u_i
D[v_i] = min(D[v_i], D[u_i] + d[u, v])
```

代码实现

Shortest-Path.py

```python
import os
import net

if __name__ == '__main__':
    DIR_NAME = os.path.dirname(__file__)
    Romania = net.net(DIR_NAME + '/Romania.txt',
                      directivity=1,
                      logger=1,
                      time_counter=0)
    Romania.readNet()

    print('Input \'stop\' to quit. ')
    while True:
        s = input()
        if s == 'stop':
            break

        if len(s.split()) != 2:
            print("ERROR: invalid input, requiring 2 arguments excepting",
                  len(s.split()))
            continue

        start, end = map(lambda val: val.capitalize(), s.split())
        for tar in Romania.nodes:
            if start == tar[0]: start = tar
            if end == tar[0]: end = tar

        # print(Romania.quest(start, end, method=net.Dijkstra))
        print(Romania.quest(start, end, method=net.SPFA))
```

net.py

```python
import time
import os
import counter


class net(object):

    def __init__(self, file='', directivity=0, logger=1, time_counter=0):
        # directivity=0 representing directed-map, =1 representing undirected-map
        # logger=0 representing log mode off, =1 representing log mode on
        # time_counter=0 representing running time counter off, =1 representing counter
on

        DIR_NAME = os.path.dirname(__file__)
        self.map_info = open(file, mode='r')
        self.m, self.n = map(int, self.map_info.readline().split())
        self.directivity = directivity
        self.edges = {}
        self.dist = {}
        self.nodes = set()
        self.logger = logger
        self.time_counter = time_counter
        self.avg_time = 0
        self.avg_count = 0
        LOG_LIMITS = 5
        # log files limit

        tmp = open(DIR_NAME + '/net_logfiles/IGNORED_FILES.txt', mode='r')
        IGNORED_FILES = tmp.read().split()

        FILES = os.listdir(DIR_NAME + '/net_logfiles')
        for ig in IGNORED_FILES:
            if ig in FILES:
                FILES.remove(ig)
        # keep files that should be presistently saved, stored in
'/net_logfiles/IGNORED_FILES'

        for i in range(len(FILES) - LOG_LIMITS + self.logger):
            os.remove(DIR_NAME + '/net_logfiles/' + min(FILES))
            FILES.remove(min(FILES))
        # remove excessive files

        if self.logger:
            self.log_file = open(time.strftime(
                DIR_NAME + "/net_logfiles/%Y%m%d-%H-%M-%S.log",
                time.localtime()),
                                 mode='w+')

    def __del__(self):
        self.map_info.close()

        if self.logger:
            self.log_file.close()

        if self.time_counter:
            print('Average running time:',
                  round(self.avg_time / self.avg_count * 1000, 3), 'ms')

    def readNet(self):
        for i in range(self.n):
            start, end, distance = self.map_info.readline().split()
            self.nodes |= set([start, end])
            distance = int(distance)
            for j in range(self.directivity + 1):
                if start not in self.edges: self.edges[start] = []
                self.edges[start].append([end, distance])
                start, end = end, start

    def quest(self, start, end, method):
        # [method] throw out an opportunity to modify shortest path algorithm
        if start not in self.nodes or end not in self.nodes:
            return "ERROR: invalid input, please check nodes\' name"

        count = counter.counter()
        if start not in self.dist:
            count.refresh()
            method(start, self)
            if self.time_counter:
                self.avg_time += count.print()
                self.avg_count += 1
        path = self.dist[start]
        # generate shortest path with [start] as start node

        distance = self.dist[start][end][0]
        pathOut = ''
        cur = end
        while cur != start:
            pathOut = ' -> ' + cur + pathOut
            cur = path[cur][1]
        pathOut = start + pathOut

        if self.logger:
            self.log_file.write(start + ' ' + end + '\n')
            self.log_file.write("The shortest path is: " + pathOut + '\n')
            self.log_file.write("The distance is: " + str(distance) + '\n')

        return "The shortest path is: " + pathOut + '\n' + "The distance is: " + str(
            distance)


def Dijkstra(start, net):  # Djikstra to get the shortest path
    queue = [start]
    path = {start: [0, ""]}
    while len(queue):
        cur = min(queue, key=lambda val: path[val][0])
        for tar, dis in net.edges[cur]:
            if tar not in path:
                path[tar] = [int(1E9), ""]
            if path[tar][0] > path[cur][0] + dis:
                path[tar] = [path[cur][0] + dis, cur]
                queue.append(tar)
        queue.remove(cur)
    net.dist[start] = path
    return


def SPFA(start, net):  # SPFA to get the shortest path
    queue = [start]
    path = {start: [0, ""]}
    while len(queue):
        cur = queue[0]
        for tar, dis in net.edges[cur]:
            if tar not in path:
                path[tar] = [int(1E9), ""]
            if path[tar][0] > path[cur][0] + dis:
                path[tar] = [path[cur][0] + dis, cur]
                if tar not in queue:
                    queue.append(tar)
        queue.pop(0)

    net.dist[start] = path
    return
```

counter.py

```python
import time


class counter(object):

    def __init__(self):
        self.refresh()

    def refresh(self):
        self.basetime = time.time()

    def print(self):
        return time.time() - self.basetime
```

增加了以下功能：

1. 记录文件的临时保存以及长期保存，自动删除溢出的临时文件，过滤了错误信息
2. 输入合法性的检查
3. 多种最短路算法切换接口
4. 新源点算法执行时间显示

三、实验结果及分析

实验结果展示

On terminal

```
Last login: Tue Feb 28 18:03:31 on ttys001
(base) MacBook-Pro ~ % python3 Shortest-Path.py
Input 'stop' to quit.
I z
The shortest path is: Iasi -> Vaslui -> Urziceni -> Bucharest -> Pitesti ->
RimnicuVilcea -> Sibiu -> Arad -> Zerind
The distance is: 812
c v
The shortest path is: Craiova -> Pitesti -> Bucharest -> Urziceni -> Vaslui
The distance is: 466
Pitesti Sibiu
The shortest path is: Pitesti -> RimnicuVilcea -> Sibiu
The distance is: 177
aldjwka fwa
ERROR: invalid input, please check nodes' name
Iasi Pitesti Sibiu
ERROR: invalid input, requiring 2 arguments excepting 3
stop
(base) MacBook-Pro ~ %
```

20230301-10-33-19.log

```
Iasi Zerind
The shortest path is: Iasi -> Vaslui -> Urziceni -> Bucharest -> Pitesti ->
RimnicuVilcea -> Sibiu -> Arad -> Zerind
The distance is: 812
Craiova Vaslui
The shortest path is: Craiova -> Pitesti -> Bucharest -> Urziceni -> Vaslui
The distance is: 466
Pitesti Sibiu
The shortest path is: Pitesti -> RimnicuVilcea -> Sibiu
The distance is: 177
```

运行时间：

```
stop
Average running time: 0.125 ms
```

实验平台：

Apple M1 Pro

Visual Studio Code

Python 3.11.2 Based on Anaconda

四、思考题

1. 可变元素不能作为字典的键，故会报错；元组作为字典的键，其中任一元素都可以找到对应字典值
2. *Test.py*

```python
if __name__ == '__main__':
    print('int')
    a = 1
    b = 1
    c = a
    print(id(1))
    print(id(a), id(b), id(c))

    print('float')
    a = 1.2
    b = 1.2
    c = a
    print(id(1.2))
    print(id(a), id(b), id(c))

    print('bool')
    a = True
    b = True
    c = a
    print(id(True))
    print(id(a), id(b), id(c))

    print('string')
    a = 'string'
    b = 'string'
    c = a
    print(id('string'))
    print(id(a), id(b), id(c))

    print('list')
    a = [1]
    b = [1]
    c = a
    print(id([1]))
    print(id(a), id(b), id(c))

    print('tuple')
    a = (1)
    b = (1)
    c = a
    print(id((1)))
    print(id(a), id(b), id(c))

    print('set')
    a = set([1])
    b = set([1])
    c = a
    print(id(set([1])))
    print(id(a), id(b), id(c))

    print('dict')
    a = {1: 1}
    b = {1: 1}
    c = a
    print(id({1: 1}))
    print(id(a), id(b), id(c))
```

On Terminal

```
(base) MacBook-Pro ~ % python3 Test.py
int
4370458928
4370458928 4370458928 4370458928
float
4373099056
4373099056 4373099056 4373099056
bool
4369545408
4369545408 4369545408 4369545408
string
4373798000
4373798000 4373798000 4373798000
list
4374910336
4373432640 4373432064 4373432640
tuple
4370458928
4370458928 4370458928 4370458928
set
4374904608
4373683776 4374903936 4373683776
dict
4373402816
4373402560 4373402752 4373402560
```

可见，int, float, bool, string, tuple 为不可变，list, set, dict 为可变。