题目：交通地理信息系统
最短路径作业分析报告

姓　　名：王　倩　妮

学　　号：2015112956

班　　级：交通 2015-02 班

任课教师：谢军

2017 年 11 月 16 日

一、 问题描述

现有 Sioux Falls network、Chicago Sketch network 两个不同规模的路网，现需达成以下目标：

- 编程读入符合一定编写规则的 txt 格式原始文件；
- 编程实现 Label Setting 和 Label Correcting 两个最短路径算法；
- 在两个不同规模的路网上运行，对比、分析两种算法的效率；
- 打印输出 Chicago Sketch network 上从 1 点到 866 点的最短路径；
- 完成分析报告。

二、实践方法

- **数据读入**
  a) 导入程序所需模块，开始计时；
  b) 打开 txt 文件；
  c) 按行读取文件。读取文件首行，获取 node 数，继续读取 node 行数的 node 信息，依靠符号替换，将每行的数据进行分割，构成 node 矩阵；
  d) 再继续读取一行，获得 link 数，继续读取 link 行数的 link 信息，同理进行预处理，构成 link 矩阵。

  注：因 Python 索引从 0 开始引用，为保证矩阵索引数与标号一致，我们在接下来的操作中都设法废掉 0 位的信息。
- **LS 算法**
  a) 初始化。对 L、P、Q 进行初始化。
  b) 节点选择。判断是否继续循环，若继续，则从 Q(候选节点集)中选择并移出标号最小的节点。(此处创建 L_copy 帮助寻找最小标号节点)
  c) 节点扩展。针对选择的节点，验证该节点出发的每一条弧段是否有 $L_i + length_{(i,j)} < L_j$，若有，则修改 L、P、Q 中的参数(修改节点 j 标号、重置前驱节点、插入候选节点)。
  d) 终止条件。若已确定目标点，则循环停止条件为“选择的点为目标点”。此处，假设没有确定具体目标点，循环停止条件为“Q 为空”。
  e) 倒序打印最短路。依靠 P 中的记录，追踪最短路，存入 passnode 中。

f) 停止计时，输出运行时间。

- LC 算法

a) 初始化。对 L、P 进行初始化。

b) 不断调整标号直至所有标号都满足 $L_j \le L_i + length_{(i,j)}$。若存在不满足的标号时，进行两项调整：令 $L_j = L_i + length_{(i,j)}, p[j] = i$。(本程序中依靠“whethercontinue”变量控制是否继续循环)

c) 倒序打印最短路。依靠 P 中的记录，追踪最短路，存入 passnode 中。

d) 停止计时，输出运行时间。

三、LS 算法与 LC 算法在两路网中的运行效率分析

路网名/算法(运行时间)	LS 算法	LC 算法
Sioux Falls Network	0.167295 s	0.159163 s
Chicago Sketch network	1.868675 s	0.342213 s

注：程序每次运行时间有一定差异。且由于 print 操作占用较长时间，此时间为仅打印(print)初始点到终点(1-24 或 1-933)一条最短路的运行时间。

经观察应用于同一路网时两种算法的运行时间，可以发现 LC 算法的运行效率高于 LS 算法。与理论相一致。

由于没有设定 LS 算法的终点（目标点），本人设计的 LS 算法也是全局状态。实际上求解 1:1 的最短路径 LS 算法更为合适，因为到达目标点后算法即停止。而求解 1：n 的最短路径 LC 算法更为合适。在实际应用过程中应就具体问题选择合适的方法解决问题。

四、输出结果

路网名/最短路径、长度	初始点到终点的最短路	最短路长度
Sioux Falls Network	[1, 3, 12, 13, 24]	3.75km
Chicago Sketch network	[1, 547, 549, 551, 563, 564, 565, 569, 573, 528, 526, 527, 543, 534, 933]	13.25km

题目要求：输出 Chicago Sketch network 的 1-866 节点的最短路。

最短路径为：[1, 547, 549, 550, 560, 556, 624, 623, 627, 486, 480, 483, 539, 409, 410,

700, 695, 696, 698, 810, 866]

最短路长度为：14.75km

五、运行环境

- 操作系统：Windows 8.1 中文版
- 编程语言：Python3
- Python 环境：利用 Python3.6 编程实现，使用 Python 自带 IDLE
- 使用第三方库：numpy、copy、time
- （numpy 需下载 Python3.6 并配置环境变量后，在命令提示行通过 pip install numpy 进行安装；copy 与 time 模块为下载 Python 时的自带模块）
- 硬件环境：

✓ 处理器:Intel(R)Core(TM)i5-5200U CPU @ 2.20GHz 2.20GHz

✓ 安装内存(RAM):4.00GB

✓ 系统类型:64 位操作系统，基于 x64 的处理器

六、程序清单

6.1 芝加哥路网 LC 算法

```
#########################LC 算法的 933 节点 Chicago Sketch
network#######################
import time
start = time.clock()
import numpy as np
import copy
#读取文件
file=open("cs.1.txt",'r')
#####读 node
node_count=int(file.readline())#读点数量
node=np.array([0,0])#第一行不用，用 0 占位
for _ in range(node_count):
    nodes=file.readline()
    if len(nodes)==7:
        ls=int(nodes[:2].replace("\t",""))
        le=int(nodes[2:].replace("\n","").replace("\t",""))
        node=np.row_stack((node,[ls,le]))
    elif len(nodes)==9:
        ls=int(nodes[:3].replace("\t",""))
```

```
            le=int(nodes[3:].replace("\n","").replace("\t",""))
            node=np.row_stack((node,[ls,le]))
        elif len(nodes)==11 or len(nodes)==12:
            ls=int(nodes[:4].replace("\t",""))
            le=int(nodes[4:].replace("\n","").replace("\t",""))
            node=np.row_stack((node,[ls,le]))
        else:
            ls=int(nodes[:5].replace("\t",""))
            le=int(nodes[5:].replace("\n","").replace("\t",""))
            node=np.row_stack((node,[ls,le]))
#print("nodearray=",node)
#####读 link
link_count=int(file.readline())#读弧段数量
link=np.array([0,0,0,0,0])#第一行不用，用 0 占位
for _ in range((node_count+1),(node_count+1+link_count)):
        links=file.readline()
        links=links.replace('\t',',').replace('\n','')
        links=links.split(",")
        for ii in range(len(links)):
            links[ii]=float(links[ii])
        fn=links[0]
        tn=links[1]
        capacity=links[2]
        length=links[3]
        fs=links[4]
        link=np.row_stack((link,[fn,tn,capacity,length,fs]))
#print("linkarray=",link)
####初始化 LPQ
L=np.zeros(node_count+1)
P=np.zeros(node_count+1)
Q=[]
i=1
L[0]=10000000#0 位置不用，放一个足够大的数
L[2:]=10000000#将无穷大归为一个够大的数
P[0]=np.nan#0 位置不用，用空值占位
P[1]=-1
Q.append(i)
##########循环部分##########
whethercontinue=1
waz=[]
while whethercontinue==1:
        for li in range(1,link_count+1):
            i=int(link[li,0])
            j=int(link[li,1])
```

```
            length=link[li,3]
            if L[j]>L[i]+length:
                L[j]=L[i]+length
                P[j]=i
                waz.append(1)
            else:
                waz.append(0)
        #print(waz)
        if 1 in waz:
            waz=[]
        else:
            whethercontinue=0
#####打印到各点的最短路(经过的节点、最短路径长度)
print("These are 933 node 2950 link LC program's results:")
for n in range(1,node_count+1):
        passnode=[]
        passlink=[]
        passnode.append(n)
        pn=int(n)
        while passnode[-1]!=-1:
            passnode.append(P[pn])
            pn=int(P[pn])
        passnode.pop()
        passnode.reverse()
        for s in range(len(passnode)):
            passnode[s]=int(passnode[s])
print("sp passnode to node %s:"%n,passnode,"leastlength=",L[int(n)])#此处只输出初始点到终点（1-933）的最短路，若希望打印其他则缩进一格
end = time.clock()
print ("time to run this LC Program: %f s" % (end - start))
```

6.2 芝加哥路网 LS 算法

```
#########################LS 算法的 933 节点 Chicago Sketch network#########################
import time
start = time.clock()
import numpy as np
import copy
#读取文件
file=open("cs.1.txt",'r')
#####读 node
node_count=int(file.readline())#读点数量
node=np.array([0,0])#第一行不用，用 0 占位
for _ in range(node_count):
```

```
        nodes=file.readline()
        if len(nodes)==7:
            ls=int(nodes[:2].replace("\t",""))
            le=int(nodes[2:].replace("\n","").replace("\t",""))
            node=np.row_stack((node,[ls,le]))
        elif len(nodes)==9:
            ls=int(nodes[:3].replace("\t",""))
            le=int(nodes[3:].replace("\n","").replace("\t",""))
            node=np.row_stack((node,[ls,le]))
        elif len(nodes)==11 or len(nodes)==12:
            ls=int(nodes[:4].replace("\t",""))
            le=int(nodes[4:].replace("\n","").replace("\t",""))
            node=np.row_stack((node,[ls,le]))
        else:
            ls=int(nodes[:5].replace("\t",""))
            le=int(nodes[5:].replace("\n","").replace("\t",""))
            node=np.row_stack((node,[ls,le]))
#print("nodearray=",node)
#####读 link
link_count=int(file.readline())#读弧段数量
link=np.array([0,0,0,0,0])#第一行不用，用 0 占位
for _ in range((node_count+1),(node_count+1+link_count)):
        links=file.readline()
        links=links.replace('\t',',').replace('\n','')
        links=links.split(",")
        for ii in range(len(links)):
            links[ii]=float(links[ii])
        fn=links[0]
        tn=links[1]
        capacity=links[2]
        length=links[3]
        fs=links[4]
        link=np.row_stack((link,[fn,tn,capacity,length,fs]))
#print("linkarray=",link)
####初始化 LPQ
L=np.zeros(node_count+1)
P=np.zeros(node_count+1)
Q=[]
i=1
L[0]=10000000#0 位置不用，放一个足够大的数
L[2:]=10000000#将无穷大归为一个够大的数
P[0]=np.nan#0 位置不用，用空值占位
P[1]=-1
Q.append(i)
```

```
##########循环部分###########
while len(Q)!=0:#终止条件（本身应该是到目标点截止，现在是全局条件）
#####节点选择
    L_copy=copy.deepcopy(L)
    for q in range(len(L_copy)):
        if q in Q:
            pass
        else:
            L_copy[q]=1000000
    i=np.argmin(L_copy)
    Q.remove(i)
#####节点扩展
    node_link=np.arange(node[i,0],node[i,1]+1)
    for l in node_link:
        fn=int(link[l,0])
        tn=int(link[l,1])
        length=link[l,3]
        if L[i]+length<L[tn]:
            L[tn]=L[i]+length
            P[tn]=i
            Q.append(tn)
#####打印到各点的最短路(经过的节点、最短路径长度)
print("These are 933 node 2950 link LS program's results:")
for n in range(1,node_count+1):
    passnode=[]
    passlink=[]
    passnode.append(n)
    pn=int(n)
    while passnode[-1]!=-1:
        passnode.append(P[pn])
        pn=int(P[pn])
    passnode.pop()
    passnode.reverse()
    for s in range(len(passnode)):
        passnode[s]=int(passnode[s])
print("sp passnode to node %s:"%n,passnode,"leastlength=",L[int(n)])#此处仅输出头到尾的最短路，若输出全局则缩进一格
end = time.clock()
print ("time to run this LS Program: %f s" % (end - start))
```

6.3Sioux Falls NetworkLC 算法

```
########################LC 算法的 6 节点 13 弧段实现#######################
import time
start = time.clock()
```

```
import numpy as np
import copy
#读取文件
file=open("Sioux Falls Network.txt.txt",'r')
#####读 node
node_count=int(file.readline())#读点数量
node=np.array([0,0])#第一行不用，用 0 占位
for _ in range(node_count):
    nodes=file.readline()
    ls=int(nodes[:8].replace(" ",""))
    le=int(nodes[8:].replace(" ",""))
    node=np.row_stack((node,[ls,le]))
#print("nodearray=",node)
#####读 link
link_count=int(file.readline())#读弧段数量
link=np.array([0,0,0,0,0])#第一行不用，用 0 占位
for _ in range((node_count+1),(node_count+1+link_count)):
    links=file.readline()
    fn=float(links[:6].replace(" ",""))
    tn=float(links[6:13].replace(" ",""))
    capacity=float(links[13:26].replace(" ",""))
    length=float(links[26:35].replace(" ",""))
    fs=float(links[35:].replace(" ",""))
    link=np.row_stack((link,[fn,tn,capacity,length,fs]))
#print("linkarray=",link)
file.close()
####初始化 LPQ
L=np.zeros(node_count+1)
P=np.zeros(node_count+1)
Q=[]
i=1
L[0]=10000000#0 位置不用，放一个足够大的数
L[2:]=10000000#将无穷大归为一个够大的数
P[0]=np.nan#0 位置不用，用空值占位
P[1]=-1
Q.append(i)
##########循环部分###########
whethercontinue=1
waz=[]
while whethercontinue==1:
    for li in range(1,link_count+1):
        i=int(link[li,0])
        j=int(link[li,1])
        length=link[li,3]
```

```
            if L[j]>L[i]+length:
                L[j]=L[i]+length
                P[j]=i
                waz.append(1)
            else:
                waz.append(0)
        #print(waz)
        if 1 in waz:
            waz=[]
        else:
            whethercontinue=0
#####打印到各点的最短路(经过的节点、最短路径长度)
print("These are 24 node 76 link LC program's results:")
for n in range(1,node_count+1):
        passnode=[]
        passlink=[]
        passnode.append(n)
        pn=int(n)
        while passnode[-1]!=-1:
            passnode.append(P[pn])
            pn=int(P[pn])
        passnode.pop()
        passnode.reverse()
        for s in range(len(passnode)):
            passnode[s]=int(passnode[s])
print(" passnode to node %s:"%n,passnode,"leastlength=",L[int(n)])
end = time.clock()
print ("time to run this LC Program: %f s" % (end - start))
```

6.4 Sioux Falls NetworkLS 算法

```
########################LS 算法的 24 节点 76 弧段实现######################
import time
start = time.clock()
import numpy as np
import copy
#读取文件
file=open("Sioux Falls Network.txt.txt",'r')
#####读 node
node_count=int(file.readline())#读点数量
node=np.array([0,0])#第一行不用，用 0 占位
for _ in range(node_count):
        nodes=file.readline()
        ls=int(nodes[:8].replace(" ",""))
        le=int(nodes[8:].replace(" ",""))
        node=np.row_stack((node,[ls,le]))
```

```
#print("nodearray=",node)
#####读 link
link_count=int(file.readline())#读弧段数量
link=np.array([0,0,0,0,0])#第一行不用，用 0 占位
for _ in range((node_count+1),(node_count+1+link_count)):
    links=file.readline()
    fn=float(links[:6].replace(" ",""))
    tn=float(links[6:13].replace(" ",""))
    capacity=float(links[13:26].replace(" ",""))
    length=float(links[26:35].replace(" ",""))
    fs=float(links[35:].replace(" ",""))
    link=np.row_stack((link,[fn,tn,capacity,length,fs]))
#print("linkarray=",link)
file.close()
####初始化 LPQ
L=np.zeros(node_count+1)
P=np.zeros(node_count+1)
Q=[]
i=1
L[0]=10000000#0 位置不用，放一个足够大的数
L[2:]=10000000#将无穷大归为一个够大的数
P[0]=np.nan#0 位置不用，用空值占位
P[1]=-1
Q.append(i)
##########循环部分###########
while len(Q)!=0:#终止条件（本身该条件应该是到目标点截止，现在是全局条件）
#####节点选择
    L_copy=copy.deepcopy(L)
    for q in range(len(L_copy)):
        if q in Q:
            pass
        else:
            L_copy[q]=1000000
    i=np.argmin(L_copy)
    Q.remove(i)
#####节点扩展
    node_link=np.arange(node[i,0],node[i,1]+1)
    for l in node_link:
        fn=int(link[l,0])
        tn=int(link[l,1])
        length=link[l,3]
        if L[i]+length<L[tn]:
            L[tn]=L[i]+length
            P[tn]=i
```

```
                Q.append(tn)
#####打印到各点的最短路(经过的节点、最短路径长度)
print("These are 24 node 76 link LS program's results:")
for n in range(1,node_count+1):
    passnode=[]
    passlink=[]
    passnode.append(n)
    pn=int(n)
    while passnode[-1]!=-1:
        passnode.append(P[pn])
        pn=int(P[pn])
    passnode.pop()
    passnode.reverse()
    for s in range(len(passnode)):
        passnode[s]=int(passnode[s])
print("sp passnode to node %s:"%n,passnode,"leastlength=",L[int(n)])#若希望打印所有，则缩进一位
end = time.clock()
print ("time to run this LS Program: %f s" % (end - start))
```