Assignment 2

赵嘉 ZY1806418
北京航空航天大学 计算机学院

1. 问题描述

某公司于乙城市的销售点急需一批成品，该公司成品生产基地在甲城市。甲城市与乙城市之间共有 n 座城市，互相以公路连通。甲城市、乙城市以及其它各城市之间的公路连通情况及每段公路的长度由矩阵 M1 给出。每段公路均由地方政府收取不同额度的养路费等费用，具体数额由矩阵 M2 给出。

请给出在需付养路费总额不超过 1500 的情况下，该公司货车运送其产品从甲城市到乙城市的最短运送路线。

具体数据参见文件：

- M1.txt: 各城市之间的公路连通情况及每段公路的长度矩阵(有向图)； 甲城市为城市 Num.1，乙城市为城市 Num.50。
- M2.txt: 每段公路收取的费用矩阵（非对称）。

2. 问题分析

2.1 解题思想

首先使用 floyid 算法求出所有结点对之间的最短路长和最小费用，以便之后剪枝使用。然后从起点开始寻找到终点的路径，具体操作如下：

初始化一个堆栈,首先把 0 结点（起点）放入堆栈。

每次取出栈顶的结点，并检查与此节点相接的节点，若某个相接的节点不满足界限要求则被剪枝和回溯，否则将其放入堆栈令其作为栈顶节点继续扩展。

找到一个解后,进行保存,然后回溯.如此进行,直到栈空。

2.2 剪枝策略

剪枝策略有两条：

1. currentDist+distance_arr[cur][i]+min_distance_arr[i][49]>=distBound
2. currentFee+fee_arr[cur][i]+min_fee_arr[i][49]> feeBound:

即当前路径的距离超过最优路径的距离时剪枝以及当前路径所花费养路费超过 1500 时剪枝，并进行回溯。

2.2 复杂度分析

算法中一开始的 Floyd 算法的时间复杂度为 $O(n^3)$，后面的搜索路径部分的时间复杂度为 $O(n + e)$，因此总时间复杂度为 $O(n^3)$，其中 n 是所有城市的个数、e 是所有城市间互相联通的有向图边的个数。

3. 代码实现

以下为程序的代码实现：

```
# coding: utf-8

import os
import numpy as np
import time

distance_dir=os.path.join(os.getcwd(),'m1.txt')
fee_dir=os.path.join(os.getcwd(),'m2.txt')
print('distance_dir:',distance_dir)
print('fee_dir:',fee_dir)

def readfile(file_dir):
    res=[]
    with open(file_dir,'r') as f:
        for line in f.readlines():
            linelist=[int(i) for i in line.split()]
            res.append(linelist)
    return res

distance_arr=np.array(readfile(distance_dir))
fee_arr=np.array(readfile(fee_dir))
print(distance_arr)
print(fee_arr)

def floyd(D):
    lengthD = len(D)
    P =np.array(D)

    for k in range(lengthD):
        for i in range(lengthD):
            for j in range(lengthD):
                if(P[i,j] > P[i,k]+P[k,j]):
                    P[i,j] = P[i,k]+P[k,j]
    return P

min_distance_arr=floyd(distance_arr)
min_distance_arr[49][49]=0
print(min_distance_arr.tolist())
#print(distance_arr.tolist())
min_fee_arr=floyd(fee_arr)
min_fee_arr[49][49]=0
```

```
#print(min_fee_arr.tolist())

stack=[0 for i in range(51)]
visited=[0 for i in range(51)]
depth=0
stack[depth]=0
stack[depth+1]=0
currentDist=0
distBound=9998
currentFee=0
feeBound=1500
bestPath=[]
shortestDist=9999
minimumFee=9999

startTime=time.time()
while(depth>=0):
    cur=stack[depth]
    next_=stack[depth+1]
    nextnode=0
    for i in range(next_+1,50):
        if distance_arr[cur][i]==9999 or visited[i]==1:
            continue
        if
currentDist+distance_arr[cur][i]+min_distance_arr[i][49]>=distBound or
currentFee+fee_arr[cur][i]+min_fee_arr[i][49]>feeBound:
            continue
        nextnode=i
        break
    if(nextnode==0):
        depth=depth-1
        currentDist-=distance_arr[stack[depth]][stack[depth+1]]
        currentFee-=fee_arr[stack[depth]][stack[depth+1]]
        visited[stack[depth+1]]=0
    else:
        #print(nextnode)
        #print(currentDist,' ',distance_arr[cur][i],'
',min_distance_arr[i][49],'
',currentDist+distance_arr[cur][i]+min_distance_arr[i][49])
        currentDist+=distance_arr[cur][nextnode]
        currentFee+=fee_arr[cur][nextnode]
        visited[nextnode]=1
        depth+=1
        stack[depth]=nextnode
```

```
        stack[depth+1]=0
        if nextnode==49:
            bestPath.clear()
            for i in range(depth+1):
                bestPath.append(stack[i]+1)
            #print('found a solution:',bestPath,' the dis
is:',currentDist)
            shortestDist=currentDist
            minimumFee=currentFee
            distBound=currentDist
            depth-=1
            currentDist-=distance_arr[stack[depth]][stack[depth+1]]
            currentFee-=fee_arr[stack[depth]][stack[depth+1]]
            visited[stack[depth+1]]=0
costTime=time.time()-startTime

print('shortestDist:',shortestDist)
print('minimumFee:',minimumFee)
print('bestPath:',bestPath)
print('costTime:',costTime)
```

4. 运行结果

运行代码的机器 CPU 配置为:Intel(R) Xeon(R) CPU E5-4607 v2 @ 2.60GHz

算法的运行在 0.3s 之内完成，运行结果如下图所示：

```
In [9]: print('shortestDist:',shortestDist)
        print('minimumFee:',minimumFee)
        print('bestPath:',bestPath)
        print('costTime:',costTime)

        shortestDist: 464
        minimumFee: 1448
        bestPath: [1, 3, 8, 11, 15, 21, 23, 26, 32, 37, 39, 45, 47, 50]
        costTime: 0.29264187812805176
```

即结果为：

最短路径总长度：464

走最短路径所花费的养路费：1448

最短路径节点序列：[1, 3, 8, 11, 15, 21, 23, 26, 32, 37, 39, 45, 47, 50]

算法运行花费时间：0.29264187812805176

5. 附件说明

本作业提交包含以下 6 个文件，分别是：

Question.pdf：作业题目的 PDF

m1.txt：距离矩阵文件

m2.txt：养路费矩阵文件
Assignment2.ipynb：代码文件的 Jupyter 版本
Assignment2.py：代码文件的 Python 版本
Assignment2.pdf:本说明文档

若要运行代码，请将 m1.txt 与 m2.txt 放置在同一目录下。Python 版本的运行需要安装 Python3, Jupyter 版本的运行需要安装 Jupyter。