符号说明

|  |  |
| --- | --- |
|  | 赋权连通图 |
|  | 的子图 |
|  | 子图最佳回路 |

问题1（邻接矩阵）：

将地图抽象为一赋权连通图时，即会有task4.xlsx中所示矩阵（G中100000表示两点间无直接连通，即无穷远，其中，该图运行task4.m文件中为变量A）

问题2（避圈法的实现）：

将抽象的赋权连接图，经过floyd算法求齐全连接图后，在使用Kruskal 算法构造最小生成树。

Floyd算法伪代码如下：

for k=1:n

for i=1:n

for j=1:n

if dis(i,k)+dis(k,j)<dis(i,j)

dis(i,j)=dis(i,k)+dis(k,j);

end

end

end

end

Kruskal算法原理如下

（1）选，使得

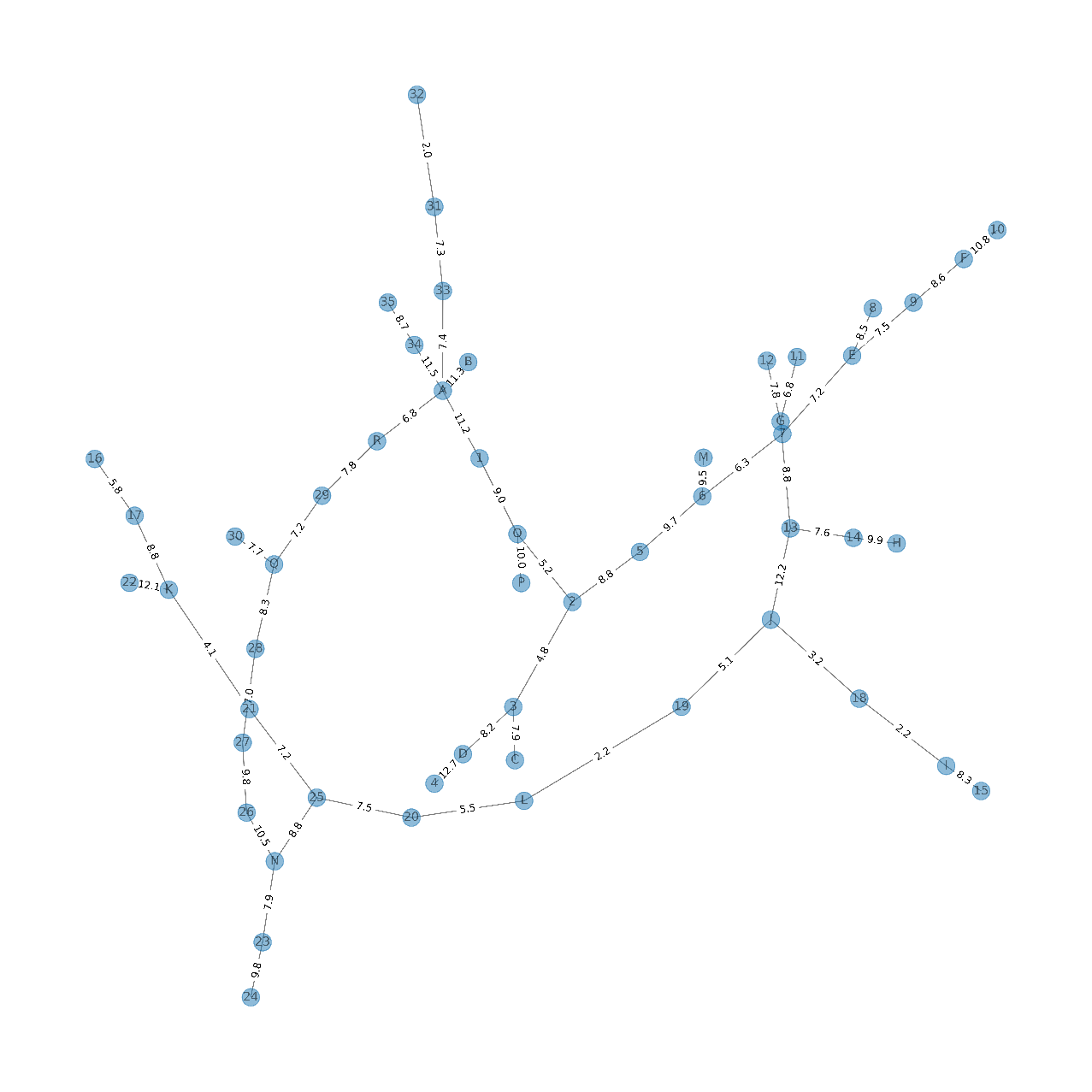
（2）若已选好，则从选取，使得

i）中无圈，且

ii）

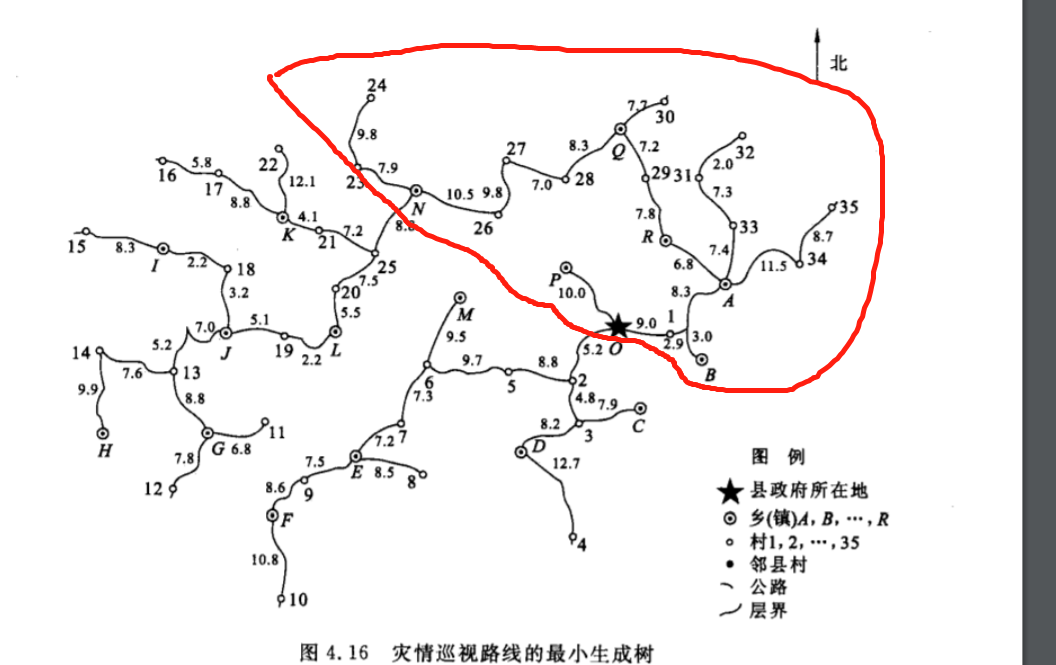
（3）直到选得为止

最小生成树如下图所示

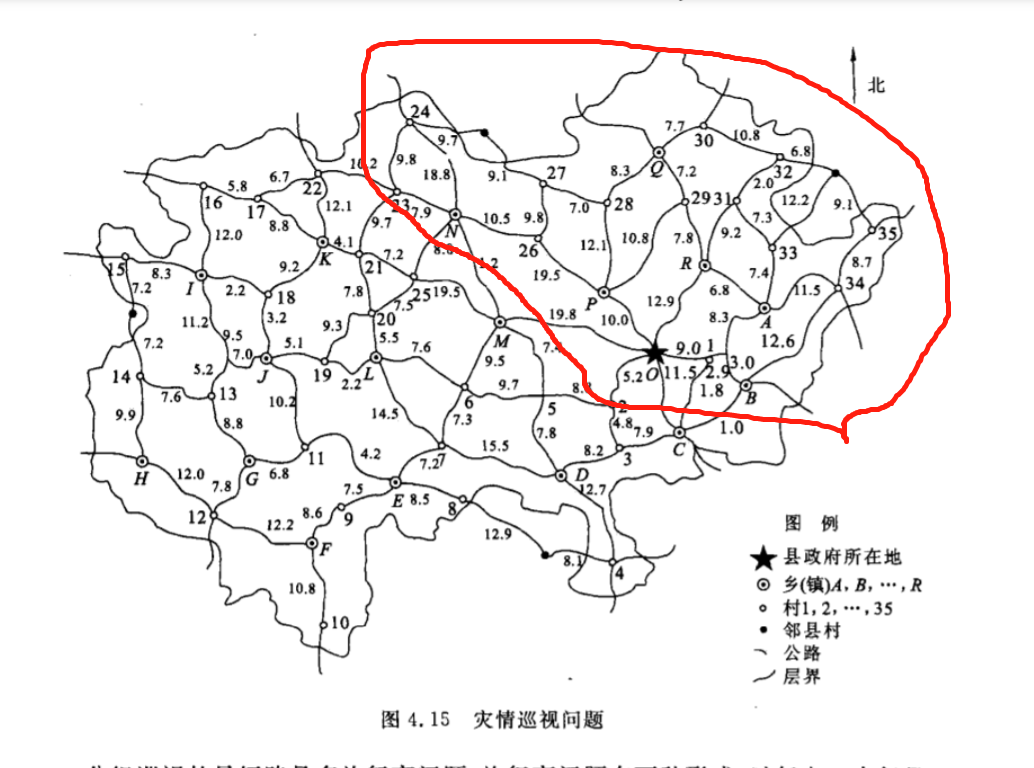


问题三（操作流程）：

借助最小生成树，按照各点的聚集方式把图分为3部分（以红色圈中为例子）



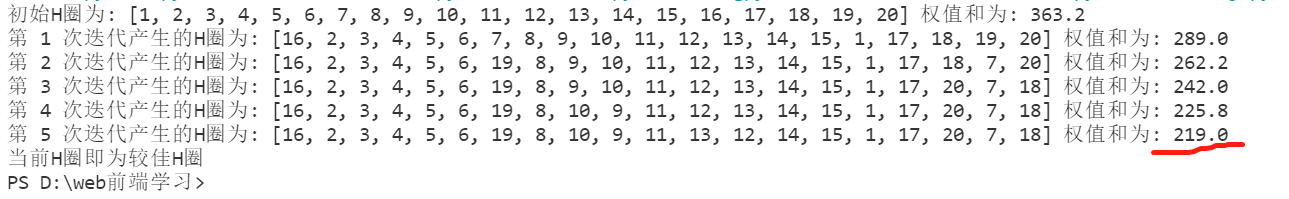
返回到原图中，求其全连接图（可在问题1中删除固定行列实现）



当划分子图后，问题近似最佳推销员回路问题。

根据新的全连接图，用逐次逼近法求其Hamilton图。

近似结果如下：



解近似，差14km,估测是在生成邻接矩阵时有数据错误，一时间无法完全还原

其余两个区域同理

相关代码：

1.最小生成树生成函数：

function MST=MST(W)

n=length(W);

T=zeros(n);

W2=W;

for i=1:n

for j=1:n

if W(i,j)==inf

W2(i,j)=0;

end

end

end

m=((nnz(W2))/2);

j=0;

for i=1:m

if j<(n-1)

minw=inf;

a=0;

b=0;

for k=1:n

for s=(k+1):n

if W(k,s)<=minw

minw=W(k,s);

a=k;

b=s;

end

end

end

T(a,b)=W(a,b);

T(b,a)=W(a,b);

f=0;

P=zeros(2,m);

y=0;

for i=1:n

for v=(i+1):n

if T(i,v)~=0

y=y+1;

P(1,y)=i;

P(2,y)=v;

end

end

end

for y=1:m

if P(1,y)<P(2,y)

for s=(y+1):m

if P(1,s)==P(2,y)

P(1,s)=P(1,y);

elseif P(2,s)==P(2,y)

P(2,s)=P(1,y);

end

end

P(2,y)=P(1,y);

elseif P(2,y)<P(1,y)

for s=(y+1):m

if P(1,s)==P(1,y)

P(1,s)=P(2,y);

elseif P(2,s)==P(1,y)

P(2,s)=P(2,y);

end

end

elseif (P(1,y)+P(2,y))~=0

f=1;

break;

end

end

if f==1

T(a,b)=0;

T(b,a)=0;

else

j=j+1;

end

W(a,b)=inf;

else

MST=T;

node=n

for i = 1:n

for j = 1:n

if MST(i,j)==inf

MST(i,j)=0;

end

end

end

MST;

weight=sum(sum(MST))/2

break;

end

end

if j<(n-1)

fprintf('该图没有最小生成树！')

end

2.floyd函数

function [dis,router]=floyd(a)

% floyd.m

% 采用floyd算法计算图a中每对顶点最短路

% d是矩离矩阵

% r是路由矩阵

n=size(a,1);

% 初始化距离矩阵

dis=a;

% 初始化路由矩阵

for i=1:n

for j=1:n

router(i,j)=j;

end

end

% Floyd算法开始

for k=1:n

for i=1:n

for j=1:n

if dis(i,k)+dis(k,j)<dis(i,j)

dis(i,j)=dis(i,k)+dis(k,j);

router(i,j)=router(i,k);

end

end

end

k;

dis;

router;

end

end

3.问题1.2的解答代码（task4.m）

clear;clc;

%%

% 原图邻接矩阵

n=53; %53 nodes

A=zeros(n,n);

%% 处理对角线

for i=1:n

for j=1:n

if(i==j)

A(i,j)=0;

else

A(i,j)=100000;

end

end

end

%% 输入数据

A(1,36)=10.3;A(1,37)=5.9 ;A(1,38)=11.2;A(1,50)=6;

A(2,3) =4.8; A(2,5) =8.3 ;A(2,50)=9.2;

A(3,38)=7.9; A(3,39)=8.2 ;A(4,8)=20.4 ;A(4,39)=12.7;

A(5,6) =9.7; A(5,39)=11.3;A(5,48)=11.4;

A(6,7) =7.3; A(6,47)=11.8;A(6,48)=9.5;

A(7,39)=15.1;A(7,40)=7.2 ;A(7,47)=14.5;

A(8,40)=8;

A(9,40)=7.8 ;A(9,41)=5.6;

A(10,41)=10.8;

A(11,40)=14.2;A(11,42)=6.8 ;A(11,45)=13.2;

A(12,41)=12.2;A(12,43)=10.2;

A(13,14)=8.6 ;A(13,42)=8.6 ;A(13,44)=16.4;A(13,45)=9.8;

A(14,15)=15 ;A(14,43)=9.9 ;

A(15,44)=8.8 ;

A(16,17)=6.8 ;A(16,44)=11.8;

A(17,22)=6.7 ;A(17,46)=9.8 ;

A(18,44)=8.2 ;A(18,45)=8.2 ;A(18,46)=9.2;

A(19,20)=9.3 ;A(19,45)=8.1 ;A(19,47)=7.2;

A(20,21)=7.9 ;A(20,25)=6.5 ;A(20,47)=5.5;

A(21,23)=9.1 ;A(21,25)=7.8 ;A(21,46)=4.1;

A(22,23)=10 ;A(22,46)=10.1;

A(23,24)=8.9 ;A(23,49)=7.9 ;

A(24,27)=18.8;A(24,49)=13.2;

A(25,48)=12 ; A(25,49)=8.8;

A(26,27)=7.8; A(26,49)=10.5;A(26,51)=10.5;

A(27,28)=7.9;

A(28,51)=12.1;A(28,52)=8.3 ;

A(29,51)=15.2;A(29,52)=7.2 ;A(29,53)=7.9;

A(30,32)=10.3;A(30,52)=7.7 ;

A(31,32)=8.1 ;A(31,33)=7.3 ;A(31,53)=9.2;

A(32,33)=19 ;A(32,35)=14.9;

A(33,35)=20.3;A(33,36)=7.4;

A(34,35)=8.2 ;A(34,36)=11.5;

A(36,37)=12.2;A(36,38)=21.5;A(36,53)=8.8;A(36,50)=16.3;

A(37,38)=11 ;A(37,50)=11.9;

A(38,39)=16.1;A(38,50)=11.5;

A(48,49)=14.2;A(48,50)=19.8;

A(50,51)=10.1;A(50,53)=12.9;

%% 保留原始矩阵(邻接矩阵)

A1=A;

for i=1:n

for j=1:n

if(A1(i,j)==100000)

A1(i,j)=inf;

end

end

end

%% 对称矩阵

for j=1:n

for i=1:j-1

A(j,i)=A(i,j);

end

end

%% 求最小生成树

[dis,router]=floyd(A);

MST=MST(A);

4.最小生成树画图代码（python实现）

import matplotlib.pyplot as plt # 导入 Matplotlib 工具包

import networkx as nx  # 导入 NetworkX 工具包

plt.figure(figsize=(15,15))

G2 = nx.Graph()  # 创建：空的 无向图

G2.add\_weighted\_edges\_from([('F', 10,10.8),

                            ('F',  9, 8.6),

                            ('E',  9, 7.5),

                            ('E',  8, 8.5),

                            ('E',  7, 7.2),

                            (  7,  6, 6.3),

                            ('M',  6, 9.5),

                            (  5,  6, 9.7),

                            (  5,  2, 8.8),

                            ('D',  4,12.7),

                            ('D',  3, 8.2),

                            ('C',  3, 7.9),

                            (  2,  3, 4.8),

                            ('O',  2, 5.2),

                            ('O','P',10.0),

                            ('O',  1, 9.0),

                            ('A',  1,11.2),

                            ('A','B',11.3),

                            ('A', 34,11.5),

                            ('A','R', 6.8),

                            ('A', 33, 7.4),

                            ( 34, 35, 8.7),

                            ( 33, 31, 7.3),

                            ( 31, 32, 2.0),

                            ('R', 29, 7.8),

                            ('Q', 28, 8.3),

                            ('Q', 29, 7.2),

                            ('Q', 30, 7.7),

                            ( 27, 28, 7.0),

                            ( 26, 27, 9.8),

                            ('N', 26,10.5),

                            ('N', 23, 7.9),

                            (23 , 24, 9.8),

                            ('N', 25, 8.8),

                            ( 21, 25, 7.2),

                            ('K', 21, 4.1),

                            ('K', 22,12.1),

                            ('K', 17, 8.8),

                            ( 16, 17, 5.8),

                            ( 20, 25, 7.5),

                            ('L', 20, 5.5),

                            ('L', 19, 2.2),

                            ('J', 19, 5.1),

                            ('J', 18, 3.2),

                            ('I', 18, 2.2),

                            ('I', 15, 8.3),

                            ('J', 13,12.2),

                            ( 13, 14, 7.6),

                            ('H', 14, 9.9),

                            ('G', 13, 8.8),

                            ('G', 11, 6.8),

                            ('G', 12, 7.8)

                                        ])

# 向图中添加多条赋权边: (node1,node2,weight)

# 两个指定顶点之间的最短加权路径

pos = nx.spring\_layout(G2)  # 用 FR算法排列节点

nx.draw(G2, pos, with\_labels=True, alpha=0.5)

labels = nx.get\_edge\_attributes(G2,'weight')

nx.draw\_networkx\_edge\_labels(G2, pos, edge\_labels = labels)

plt.savefig("plot1.png", dpi=600)

5.逐次逼近法代码实现（python实现）

import numpy as np

def feiring(D, H):

    # D是带权邻接矩阵，H是初始的H圈(本质上是1到n的一个排列)

    N = D.shape[0]

    # 初始化H圈权值和

    W = 0

    for i, j in zip(H, H[1:] + [H[0]]):

        W += D[i, j]

    print("初始H圈为:", [i + 1 for i in H], '权值和为:', W)

    # 初始化权值减小矩阵Y

    Y = np.zeros([N, N])

    # 初始化迭代次数

    k\_iter = 1

    while 1:

        # 计算权值减小矩阵Y

        for i in range(N - 1):

            for j in range(i + 1, N):

                # 一般的相邻情况：在H中两个点相邻

                if j == i + 1:

                    H1 = [H[i - 1] if i != 0 else H[-1]] + [H[i], H[j]] + [H[j + 1] if j != N - 1 else H[0]]

                    H2 = H1.copy()

                    H2[1], H2[2] = H2[2], H2[1]

                    Y[i, j] = (D[H1[0], H1[1]] + D[H1[1], H1[2]] + D[H1[2], H1[3]]) - (

                            D[H2[0], H2[1]] + D[H2[1], H2[2]] + D[H2[2], H2[3]])

                # 特殊的相邻情况：调换的两个点刚好是H的两个端点

                elif j == i + (N - 1):

                    H1 = [H[-2], H[-1], H[0], H[1]]

                    H2 = [H[-2], H[0], H[-1], H[1]]

                    Y[i, j] = (D[H1[0], H1[1]] + D[H1[1], H1[2]] + D[H1[2], H1[3]]) - (

                            D[H2[0], H2[1]] + D[H2[1], H2[2]] + D[H2[2], H2[3]])

                # 两个点不相邻的情况

                else:

                    H11 = [H[i - 1] if i != 0 else H[-1]] + [H[i], H[i + 1]]

                    H12 = [H[j - 1], H[j]] + [H[j + 1] if j != (N - 1) else H[0]]

                    H21 = [H[i - 1] if i != 0 else H[-1]] + [H[j], H[i + 1]]

                    H22 = [H[j - 1], H[i]] + [H[j + 1] if j != (N - 1) else H[0]]

                    Y[i, j] = (D[H11[0], H11[1]] + D[H11[1], H11[2]] + D[H12[0], H12[1]] + D[H12[1], H12[2]]) - (

                            D[H21[0], H21[1]] + D[H21[1], H21[2]] + D[H22[0], H22[1]] + D[H22[1], H22[2]])

        # 判断能否得到权和更小的H圈

        delta = np.max(Y)

        if delta > 0:

            pos = np.argwhere(Y == delta)[0]  # 得到权值减小矩阵中最大值的索引[i0,j0]

            # 计算新的H圈的权值和路线，并打印出来

            W -= delta

            H[pos[0]], H[pos[1]] = H[pos[1]], H[pos[0]]

            print('第', k\_iter, '次迭代产生的H圈为:', [i + 1 for i in H], '权值和为:', W)

            k\_iter += 1

        else:

            print('当前H圈即为较佳H圈')

            break

    return (H, W)

# 输入距离矩阵D

D = np.array([

[0   ,45  ,50.3,26.6,34.4,28.2,26.8,41.7,25  ,33.1,17.7,21.8,30  ,10.3,5.9 ,37.1,6   ,16.1,34  ,18.9],

[45  ,0   ,8.9 ,18.4,26.2,34.1,44.1,50.1,61.1,60.4,62.7,66.8,75  ,55.3,50.9,7.9 ,39  ,28.9,42.4,51.9],

[50.3,8.9 ,0   ,23.7,18.8,26.7,42.2,42.7,59.3,53  ,66.3,70.4,67.9,58.9,56.2,13.2,44.3,34.2,35  ,50.1],

[26.6,18.4,23.7,0   ,7.8 ,15.7,25.7,31.7,42.7,42  ,44.3,48.4,56.6,36.9,32.5,10.5,20.6,10.5,24  ,33.5],

[34.4,26.2,18.8,7.8 ,0   ,7.9 ,23.4,23.9,40.5,34.2,47.5,51.6,49.1,40.1,40.3,18.3,28.4,18.3,16.2,31.3],

[28.2,34.1,26.7,15.7,7.9 ,0   ,15.5,16  ,32.6,26.3,39.6,43.7,41.2,32.2,34.1,26.2,22.2,12.1,8.3 ,23.4],

[26.8,44.1,42.2,25.7,23.4,15.5,0   ,14.9,17.1,25.2,24.1,28.2,36.4,16.7,28.9,36.2,20.8,15.2,7.2 ,7.9 ],

[41.7,50.1,42.7,31.7,23.9,16  ,14.9,0   ,18.4,10.3,25.7,33.4,25.2,31.6,43.8,42.2,35.7,28.1,7.7 ,22.8],

[25  ,61.1,59.3,42.7,40.5,32.6,17.1,18.4,0   ,8.1 ,7.3 ,26.2,23  ,14.7,26.9,53.2,22.1,32.2,24.3,9.2 ],

[33.1,60.4,53  ,42  ,34.2,26.3,25.2,10.3,8.1 ,0   ,15.4,23.1,14.9,22.8,35  ,52.5,30.2,38.4,18  ,17.3],

[17.7,62.7,66.3,44.3,47.5,39.6,24.1,25.7,7.3 ,15.4,0   ,18.9,20.3,7.4 ,19.6,54.8,23.7,33.8,31.3,16.2],

[21.8,66.8,70.4,48.4,51.6,43.7,28.2,33.4,26.2,23.1,18.9,0   ,8.2 ,11.5,23.7,58.9,27.8,37.9,35.4,20.3],

[30  ,75  ,67.9,56.6,49.1,41.2,36.4,25.2,23  ,14.9,20.3,8.2 ,0   ,19.7,31.9,67.1,36  ,46.1,32.9,28.5],

[10.3,55.3,58.9,36.9,40.1,32.2,16.7,31.6,14.7,22.8,7.4 ,11.5,19.7,0   ,12.2,47.4,16.3,26.4,23.9,8.8 ],

[5.9 ,50.9,56.2,32.5,40.3,34.1,28.9,43.8,26.9,35  ,19.6,23.7,31.9,12.2,0   ,43  ,11.9,22  ,36.1,21  ],

[37.1,7.9 ,13.2,10.5,18.3,26.2,36.2,42.2,53.2,52.5,54.8,58.9,67.1,47.4,43  ,0   ,31.1,21  ,34.5,44  ],

[6   ,39  ,44.3,20.6,28.4,22.2,20.8,35.7,22.1,30.2,23.7,27.8,36  ,16.3,11.9,31.1,0   ,10.1,28  ,12.9],

[16.1,28.9,34.2,10.5,18.3,12.1,15.2,28.1,32.2,38.4,33.8,37.9,46.1,26.4,22  ,21  ,10.1,0   ,20.4,23  ],

[34  ,42.4,35  ,24  ,16.2,8.3 ,7.2 ,7.7 ,24.3,18  ,31.3,35.4,32.9,23.9,36.1,34.5,28  ,20.4,0   ,15.1],

[18.9,51.9,50.1,33.5,31.3,23.4,7.9 ,22.8,9.2 ,17.3,16.2,20.3,28.5,8.8 ,21  ,44  ,12.9,23  ,15.1,0   ]

])

# 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35

# 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53

# A  B  C  D  E  F  G  H  I  J  K  L  M  N  O  P  Q  R

H = list(range(D.shape[0]))

feiring(D, H)

#[16, 2, 3, 4, 5, 6, 19, 8, 10, 9, 11, 13, 12, 14, 15, 1, 17, 20, 7, 18]