****

**J I A N G S U U N I V E R S I T Y**

《网络科学基础》

第四次平时作业



学院名称： 计算机学院

专业班级： 物联网2303班

学生姓名： 邱佳亮

学生学号： 3230611072

教师姓名： 熊书明

2024年11月

# 题一代码

1. import numpy as np *#导包*
2. import networkx as nx
3. import matplotlib.pyplot as plt
4. *#%%*
5. G=nx.Graph()
6. n=15 *#节点数*
7. H=nx.path\_graph(n)*#添加节点*
8. G.add\_nodes\_from(H)
9. np.random.seed(10) *#定义随机种子*
10. def rand\_edge(vi,vj,p=0.6):
11. prob=np.random.rand()
12. if(prob>p): *#随机建立边 权重*
13. G.add\_edge(vi,vj,weight=(prob\*10))
14. i=0
15. while (i<n):
16. j=0
17. while j<i:
18. rand\_edge(i,j) *#添加边*
19. j+=1
20. i+=1
21. pos=nx.circular\_layout(G) *#可视化*
22. options={
23. "with\_labels":True,
24. "font\_size":10,
25. "font\_weight":"bold",
26. "font\_color":"white",
27. "node\_size":200,
28. "width":1
29. }
30. nx.draw(G,pos,\*\*options)
31. ax=plt.gca()
32. ax.margins(0.2)
33. plt.axis('off')
34. plt.show()
35. *#%%*
36. nx.dijkstra\_path(G,1,10) *#Dijkastra算法*
37. *#%%*
38. nx.bellman\_ford\_path(G,1,10) *#Bellman-Ford算法*

## 运行结果

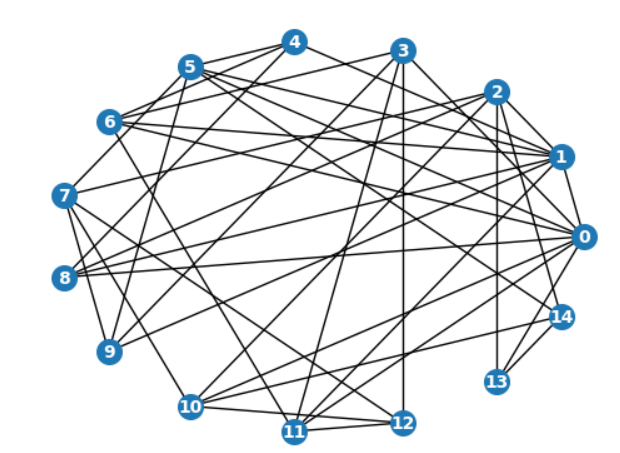


图 1 网络拓扑图

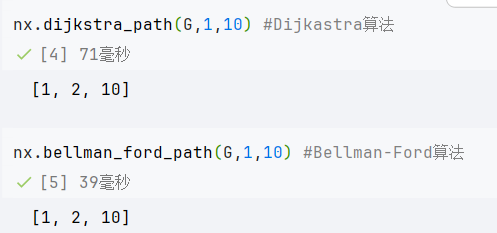


图 2 运行结果

## 算法比较

Dijkstra算法和Bellman-Ford算法都是用于求解图中单源最短路径问题的算法，它们都采用贪心策略来逼近最短路径。Dijkstra算法适用于边权全为非负的图，时间复杂度为O((V+E)logV)，实现相对复杂，不能处理负权边，也无法检测负权环。而Bellman-Ford算法可以处理含有负权边的图，能检测负权环，时间复杂度为O(VE)，实现简单，适用于可能包含负权边的图。所以如果图中含有负权边或需要检测负权环，Bellman-Ford算法效率更高；而当图的边权全为非负且图较为密集时，Dijkstra算法的效率更高。

# 题二代码

1. clc,clear;
2. a=zeros(6)
3. a(1,[2,3,5])=1;
4. a(2,[3,4])=1;
5. a(3,6)=1;
6. a(4,6)=1;
7. a=a+a*'*
8. [D,L,dist]=myAPL(a)
9. function[D,L,dist] =myAPL(a)
10. A=graph(a)
11. dist=distances(A);
12. D=max(max(dist));
13. Ldist=tril(dist);
14. he=sum(nonzeros(Ldist));
15. n=length(a);
16. L=he/nchoosek(n,2);
17. end

## 运行结果

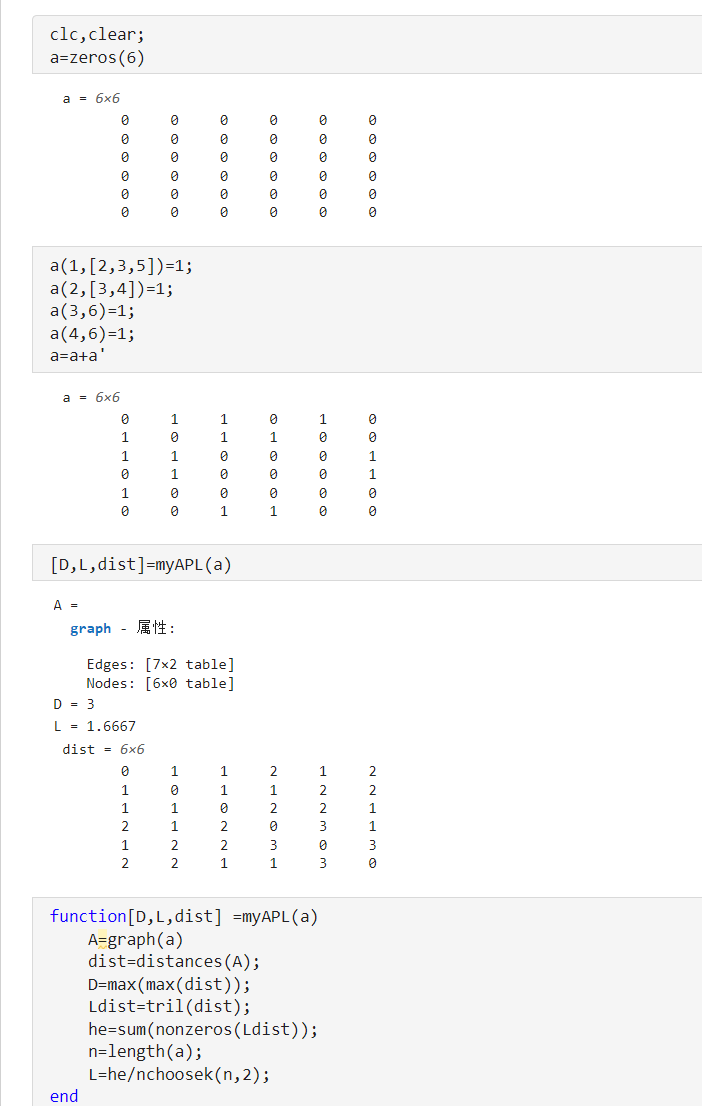


图 3 运行结果

## 查找帮助文档

### tril()

函数功能：返回矩阵 A 的下三角部分

实例：B提取了A的下三角部分，C仅提取了A的主对角线下的部分：



图 4 函数示例

### sparse()

函数功能：将矩阵转换为稀疏矩阵以节省内存；生成m×n的全零稀疏矩阵。

实例：通过sparse将A矩阵转换为稀疏矩阵，节约了内存；也可以建立一个10000\*5000的稀疏矩阵：

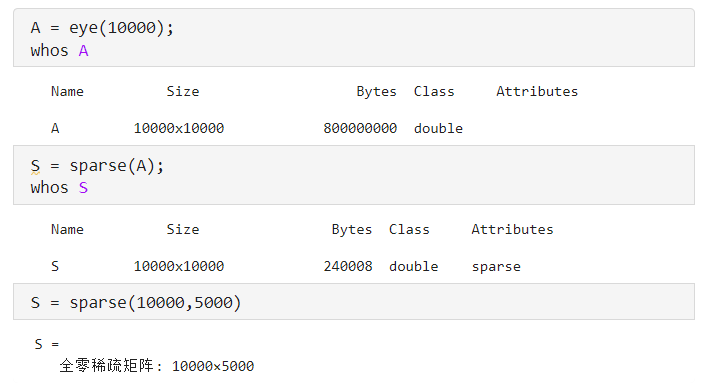


图 5 函数示例

### graphallshortestpaths()

函数功能：在图中找到所有最短路径。

实例：建立了一个稀疏矩阵G，并找到了其最短路径；graphallshortestpaths已在R2022b版本被删除，用distances代替。

1. G = sparse([6 1 2 2 3 4 4 5 5 6 1],[2 6 3 5 4 1 6 3 4 3 5],[41 99 51 32 15 45 38 32 36 29 21])
2. graphallshortestpaths(G)

### max()

函数功能：返回数组的最大元素。

实例：使用max函数返回了数组A的最大元素52：

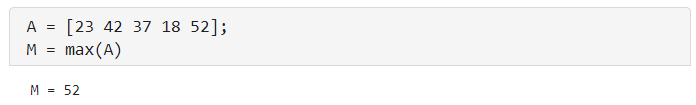


图 6 函数示例

### sum()

函数功能：返回沿数组一维度的元素之和。

实例：使用sum函数返回向量或矩阵A的元素之和，可以指定沿列方向或行方向计算：

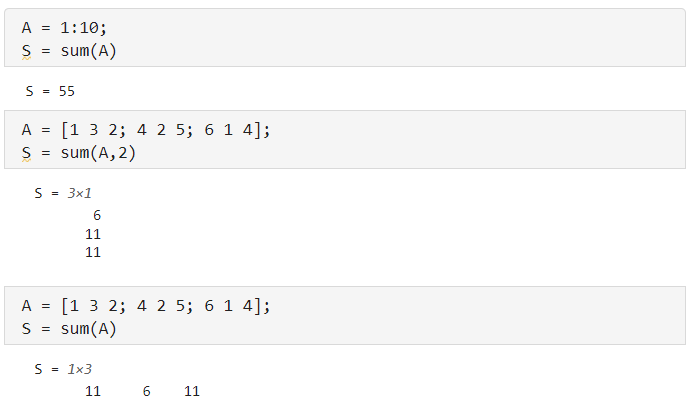


图 7 函数示例

### nonzeros()

函数功能：返回矩阵中非零元素的列向量，v 中的元素按列排序。

实例：建立了10\*10的稀疏矩阵A，用nonzeros函数返回了权不为0的边的权重，并按由小到大排序：

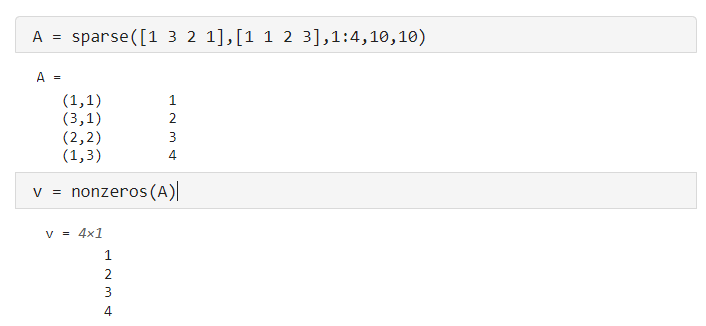


图 8 函数示例

### nchoosek()

函数功能：返回二项式系数。

实例：计算了二项式系数：

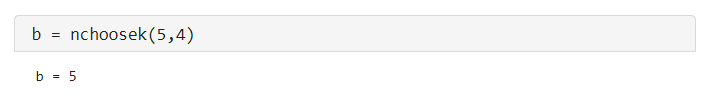


图 9 函数示例