Networkx库介绍

常鏐鏐

2020.5

1. 顶点和边

顶点（node）是社会分析的任何单位 ，如个人 、群体 、组织和社区等，点的特征指的是这些单位本身的特征 。

边（edge）：顶点之间的某种关系。

比如在社交媒体上，每个人账户可以看做一个节点，关注与被关注则可以看做用户之间的关系。

社会网络分析就是包括测量与调查点的特征和点之间的相互关系（边），将其用网络的形式表示出来 ，然后分析其关系的模式与特征。

相关代码

```
import networkx as nx
from matplotlib import pyplot as plt
```

建图

G = nx.Graph() *创建空的无向图*　G = nx.DiGraph()*创建空的有向图*

创建节点

G.add_node(1) *加入1这个点*

G.add_nodes_from([1,2,3])*用一个列表批量加入点*

删除节点

G.remove_nodes from([1,2,3])

创建边

G.add_edge(1, 2)　*（1和2有一条边）*

G.add_edges_from([(1,2),(2,3)]) 用包含元组的列表创建边

G.add_edge(2, 3, weight=0.9)　*（权重0.9）*

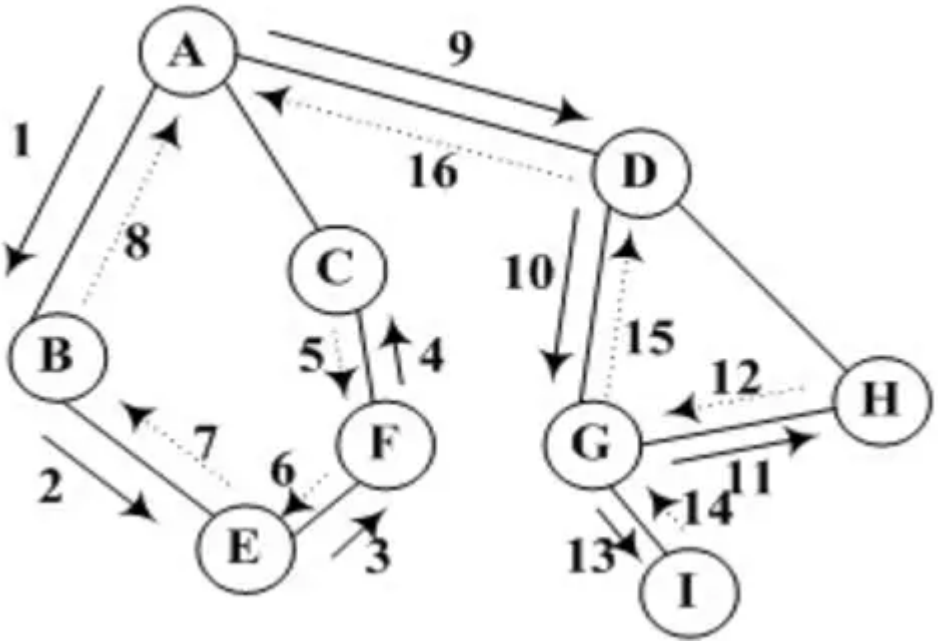
画图

```
nx.draw(G)
plt.show()
```

2.图的遍历

从图的某一个顶点出发，沿着边访遍图中所有的顶点，并且每个点只访问一次。
遍历的实质：找每个顶点邻接点的过程。
辅助数组visited[n]标记被访问过的顶点，0表示没有访问过，1表示访问过

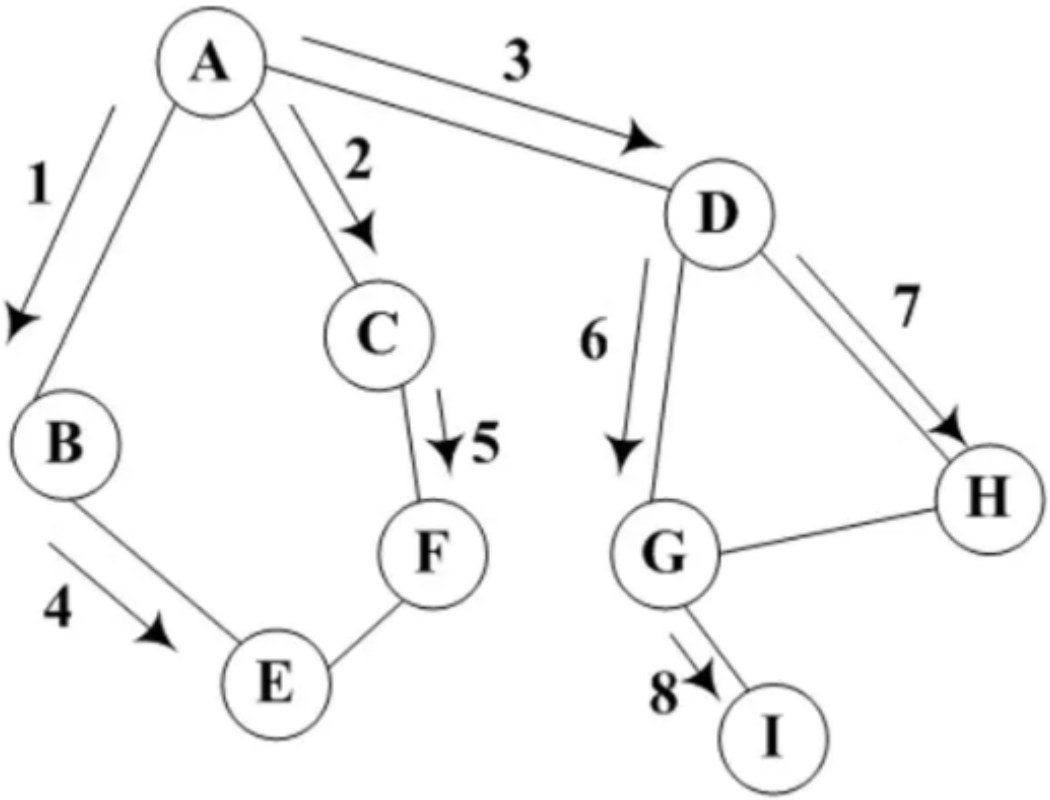
常用的遍历方法：深度优先、广度优先
可以固定遍历顶点的顺序



深度优先搜索（depth-first search, DFS）

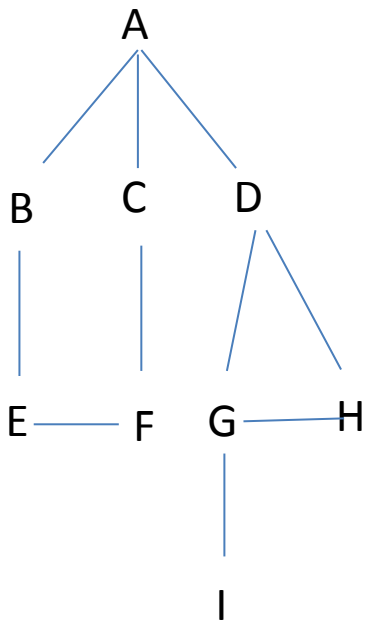
用树来比方，深度优先是沿着深度的方向访问树的节点，直到遍历该节点的所有分支，然后再回溯至该点，访问与之相邻的节点，直到遍历所有可达的节点。

A-B-E-F-C-D-G-H-I



广度优先搜索（bread-first traversal or search, BFS）

首先访问最邻近的节点，然后继续访问与该节点相邻的部分。



3. 图的应用

最短路径：　在图中A点（源点）到达B点（终点）的多条路径中，寻找各边权值之和最小的路径，即为最短路径。

两点（源点到终点）间的最短路径 --Dijkstra算法（迪杰斯特拉算法）

某源点到其他各点的最短路径—Floyd算法（佛洛依德算法）

两点（源点到终点）间的最短路径 --Dijkstra算法

初始化：先找出源点v_0达到各终点的*直达路径*（v_0,v_k）,即通过一条线到达的路径

选择：从找出的路径中找出一条长度最短的路径（v_0,u）

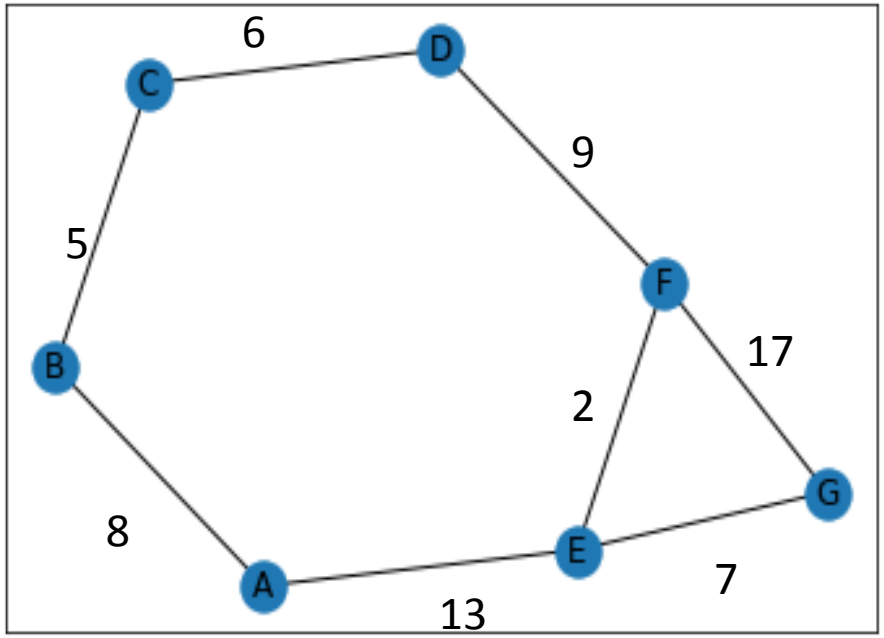
更新：对其余各条路径进行调整

如果图中存在线（u, v_k），且（v_0，u）+（u，u_k）（经过某点的路径）<（v_0,v_k）（原先找到的直达路径），就以路径（v_0,u,v_k）代替（v_0,v_k）

调整路径后，再找最短路径，以此类推。

将所有顶点放在集合V中，S代表已求出最短路径顶点的集合。

T=V-S，尚未确定最短路径的顶点集合



A为源点，求到其他各点的最短距离
V（所有点的集合）
S （已求出的最短顶点的集合{B}
T=S-V

	i=1	i=2	i=3	i=4	i=5	i=6	
B	8						
C	无穷大	13					
D	无穷大	无穷大	19	19	19		
E	13	13	13				
F	无穷大	无穷大	无穷大	15			
G	无穷大	无穷大	无穷大	20	20	20	
找到顶点	B	C	E	F	D	G	
距离	8	13	13	15	19	20	

相关代码

```
path=nx.dijkstra_path(G, source='H', target='F')
print('节点H到F的路径：', path)

distance=nx.dijkstra_path_length(G, source='H', target='F')
print('节点H到F的最短距离为：', distance)
```

4.社会网络分析中的常用指标

度中心度（ degree centrality ）测量网络中一个节点与其他节点相联系的程度。分为绝对值和标准值。对于绝对值的计算，在拥有g个节点的无向图中，节点i的程度中心性就是与其他g-1个节点的直接联系总数。对于标准值的计算：

	度中心性
绝对中心性值	InDegree(v)
标准化中心性值	$\frac{\text{InDegree(v)}}{(V-1)}$

分子：所要计算的节点与其他节点的直接联系总数

分母：该图中节点有可能的最大连接数V减一

中介中心度（betweenness centrality），计算经过一个点的最短路径的数量。经过一个点的最短路径的数量越多，就说明它的中介中心度越高。在网络中，中介中心度越高，那么在节点之间传播信息的能力就越强。一般跨界者会在多个社会网络之间体现出较强的中介中心度。

得到中介中心度的三个步骤：

1、计算每对节点（s,t）之间的最短路径，需要得到具体的路径信息

2、分析得到的所有最短路径，判断某节点是否在最短路径上

3、将刚刚的判断进行累加，得到从s到t的最短路径经过该节点的数量，其中，用经过该节点的最短路径除以所有节点对的最短路径总数，这个比率就是该点的中介中心性

接近中心度（ closeness centrality ），计算的是一个点到其他所有点的距离的总和，这个总和越小就说明这个点到其他所有点的路径越短，也就说明这个点距离其他所有点越近。
接近中心度体现的是一个点与其他点的近邻程度。

这个点的紧密中心性 基于该节点到网络中其他所有节点的最短路径之和。如果进行归一化，那么就是求这个节点到其他所有节点的平均最短距离。一个节点的平均最短距离(di)越小，那么该节点的紧密中心性越大。di越小，意味着节点vi更接近网络中的其他节点，于是把di的倒数定义为节点vi的紧密中心性，即CC(i)。

$$d_i = \frac{1}{n-1}\sum_{j \neq i} d_{ij},$$

$$CC(i) = \frac{1}{d_i} = \frac{n-1}{\sum_{j \neq i} d_{ij}}.$$

相关代码：

```
A=nx.degree_centrality(G)
B=nx.betweenness_centrality(G)
C=nx.closeness_centrality(G)
```