Python图论算法实现工具——NetworkX（2）结点与边的操作

点击查看原文可进入我的个人博客（试运行）查看具有完整引用功能的文章~

本文是参考NetworkX官方文档[1]“Python图论算法实现工具——NetworkX”系列的第二篇文章，本系列往期内容：

【图文专辑】：Python图论算法实现工具——NetworkX

1. 访问边（edges）和邻居顶点（neighbor vertices）

在上一篇文章“Python图论算法实现工具——NetworkX （1）环境配置及图的创建”中曾提出，我们可以使用`Graph.edges()`和`Graph.adj()`方法来获取边（edges）和邻居顶点（neighbor vertices）的内容，我们同样可以使用下标（subscript notation）访问边（edges）和邻居顶点（neighbor vertices）。

```
>>> G[1]  # 添加新的邻居顶点，与G.adj[1]相同
AtlasView({2: {}})
>>> G[1][2]
{}
>>> G.edges[1, 2]
{}
```

如果边已经存在，可以使用下标来获取或设置边缘的相关属性：

```
>>> G.add_edge(1, 3)
>>> G[1][3]['color'] = "blue"
>>> G.edges[1, 2]['color'] = "red"
```

使用`G.adjacency()`或者`G.adj.item()`方法可以快速获取到所有$(node, adjacency)$信息。要注意，对于无向图（undirected graphs），每条边的邻接迭代结果会被显示两次。例如官方文档中的案例[2]：

```
>>> FG = nx.Graph()  # 创建一个空图FG
>>> FG.add_weighted_edges_from([(1, 2, 0.125), (1, 3, 0.75), (2, 4, 1.2), (3, 4, 0.375)])  # 添加带权的边
>>> for n, nbrs in FG.adj.items():
...     for nbr, eattr in nbrs.items():
...         wt = eattr['weight']  # 获得边的权值
...         if wt < 0.5: print('(%d, %d, %.3f)' % (n, nbr, wt))  # 输出权值小于0.5的（node，adjacency，weight）信息
(1, 2, 0.125)  # 这里可以看到“每条边的邻接迭代结果会被显示两次”
(2, 1, 0.125)
(3, 4, 0.375)
(4, 3, 0.375)
```

这里，我将图`FG`显示出来：

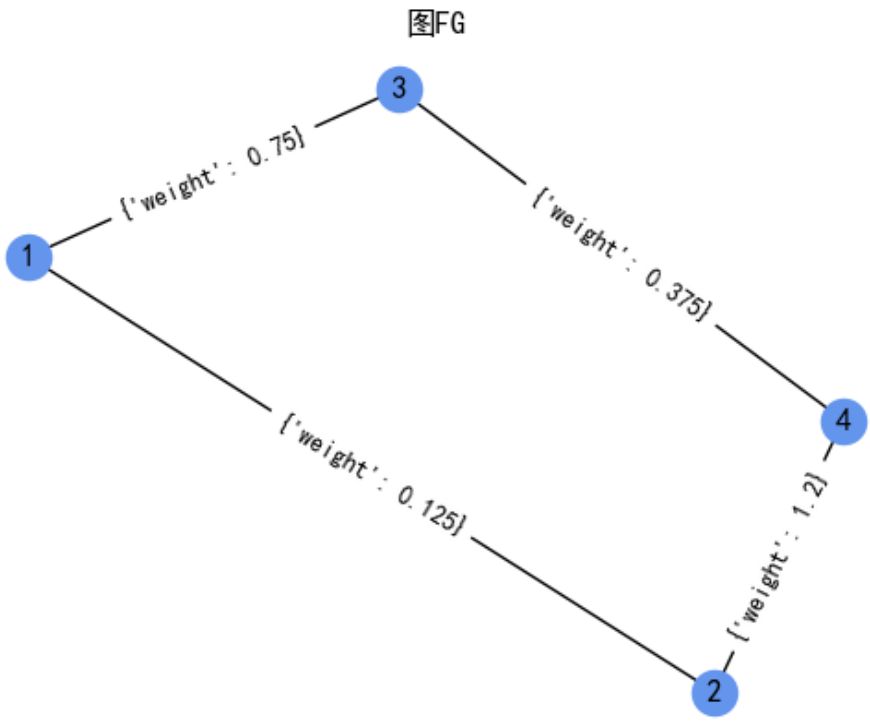


图1：图FG可视化显示

> 可视化代码：

```
pos = nx.spring_layout(FG)
plt.title('图FG')
nx.draw_networkx_edge_labels(FG, pos)
nx.draw(FG,pos, with_labels=True, node_color='#6495ED')
plt.show()
```

我们使用`edge`属性也可以轻松访问所有的边：

```
>>> for (u, v, wt) in FG.edges.data('weight'):  # FG.edges.data('weight')得到包含“weight”信息的（node，adjacency，weight）
...     if wt < 0.5: print('(%d, %d, %.3f)' % (u, v, wt)) # 输出权值小于0.5的（node，adjacency，weight）信息
(1, 2, 0.125)
(3, 4, 0.375)
```

2. 向图（graphs）、结点（nodes）、边（edges）添加信息

除了上述案例中对图`FG`的边进行赋权以外，在NetworkX中，可以将权重、标签、颜色信息等任何Python对象添加到图（graphs）、结点（nodes）、边（edges）中。

对于每一个图，结点和边都可以在与之相关的属性字典（attribute dictionary）存放属性的键值对（key/value）信息，例如图一中标注在结点1和结点3处的`{weight:0.75}`，要注意，键值对中的keys必须是可哈希的（hasable）[3]，关于python中可哈希（可散列,hasable）变量类型，也可参考CSDN文章“什么是可散列（hashable）的数据类型”[4]。在默认的情况下，属性字典是空的。但如果你需要添加或更改属性字典，可以使用`G.add_edge()`或`G.add_node()`方法，在添加结点或者边时添加相关的属性，或直接对图`G`的`G.graph`、`G.nodes`、`G.edges`属性进行操作。

（1）图的属性

我们可以在创建一个新图的时候，指定图的属性，例如：

```
>>> G = nx.Graph(day="Friday")  # 为图G创建一个“day属性”，并将图G的“day”属性指定为“Friday”
>>> G.graph
{'day': 'Friday'}
```

我们也可以在创建图之后设置图的相关属性：

```
>>> G = nx.Graph()  # 创建一个空图
>>> G.graph['day'] = "Monday"  # 为图G创建一个“day属性”，并将图G的“day”属性指定为“Friday”
>>> G.graph
{'day': 'Monday'}
```

(2) 结点的属性

我们可以使用`add_node()`或`add_nodes_from()`添加结点属性，或者利用`G.nodes`对结点属性进行操作。例如：

```
>>> G.add_node(1, time='5pm')  # 为图G添加一个结点
>>> G.add_nodes_from([3], time='2pm') # 为图G添加一个结点
>>> G.nodes[1]  # 查看结点"G.nodes[1]"的信息
{'time': '5pm'}
>>> G.nodes[1]['room'] = 714  # 为结点"G.nodes[1]"添加一个属性（键值对）
>>> G.nodes.data()
NodeDataView({1: {'time': '5pm', 'room': 714}, 3: {'time': '2pm'}})
```

> 注意：
>
> 将一个结点添加到`G.node`中并不是将其添加到图`G`中，若希望将结点添加到图`G`中应使用`G.add_node()`方法。边同理。

(3) 边的属性

我们可以使用`add_edge()`、`add_edges_from()`方法添加或改变边的属性，或者利用`G.edges`对结点属性进行操作（下标的方法）。例如：

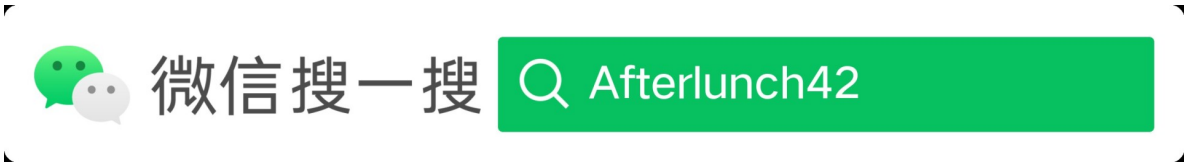
```
>>> G.add_edge(1, 2, weight=4.7 )
>>> G.add_edges_from([(3, 4), (4, 5)], color='red')
>>> G.add_edges_from([(1, 2, {'color': 'blue'}), (2, 3, {'weight': 8})])
>>> G[1][2]['weight'] = 4.7
>>> G.edges[3, 4]['weight'] = 4.2
```

> 注意：
>
> `weight`属性应当是数字，因为它在需要带权边的算法中使用。

> The special attribute `weight` should be numeric as it is used by algorithms requiring weighted edges.

本次的NetworkX工具介绍就到这里啦。如果喜欢这篇内容的话欢迎转发、收藏本文章，您的喜欢是我写作的最大动力！

欢迎关注我的微信公众号：



一位数学专业的在读大学生(菜鸡)

——————————————————

生活&音乐&学习&随笔

——————————————————

用文字记录平淡生活中每一个值得记录的瞬间。

感谢在茫茫人海中与你相遇。

做点温暖的事情，

愿你也能感受到身边的温暖。

参考资料：

[1] NetworkX2.4官方文档-install

> https://networkx.github.io/documentation/stable/index.html

[2] NetworkX2.4官方文档-accessing-edges-and-neighbors

> https://networkx.github.io/documentation/stable/tutorial.html#accessing-edges-and-neighbors

[3] Python3官方文档-hashable

> https://docs.python.org/3/glossary.html

[4] CSDN:"什么是可散列（hashable）的数据类型"

> https://blog.csdn.net/Kevin_Pei/article/details/79490298