Python图论算法实现工具——NetworkX（3）有向图、多图等图生成器

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”系列的第三篇文章，本系列往期内容：

【图文专辑】：Python图论算法实现工具——NetworkX

1. 有向图（Directed graphs）

`NetworkX`的`DiGraph`类提供了针对于有向边（directed edges）的额外属性，例如：`DiGraph.out_edges()`、`DiGraph.in_degree()`、`DiGraph.predecessors(n)`(返回n的前驱结点的迭代器)、`DiGraph.successors(n)`(返回n的后继结点的迭代器)等。

为了使算法在两类图（有向图与无向图）上可以轻松使用，当`degree`等于`in_degree`与`out_degree`的和时（即使有时不能保证一致），有向图版本的`neighbors()`方法与`successors()`方法等价。

我们可以使用`nx.DiGraph()`创建一个空的有向图：

```
DG = nx.DiGraph()  # 创建一个空的有向图
DG.add_weighted_edges_from([(1, 2, 0.5), (3, 1, 0.75)])  # 向空有向图中添加结点
out_degree = DG.out_degree(1, weight='weight')  # 查看DG的出度
degree = DG.degree(1, weight='weight')  # 查看DG的度
successors = list(DG.successors(1))  # 查看结点1的后继结点列表
neighbors = list(DG.neighbors(1))  # 查看结点1的邻接结点列表
print('out_degree:', out_degree)
print('degree:', degree)
print('successors:', successors)
print('neighbors:', neighbors)
```

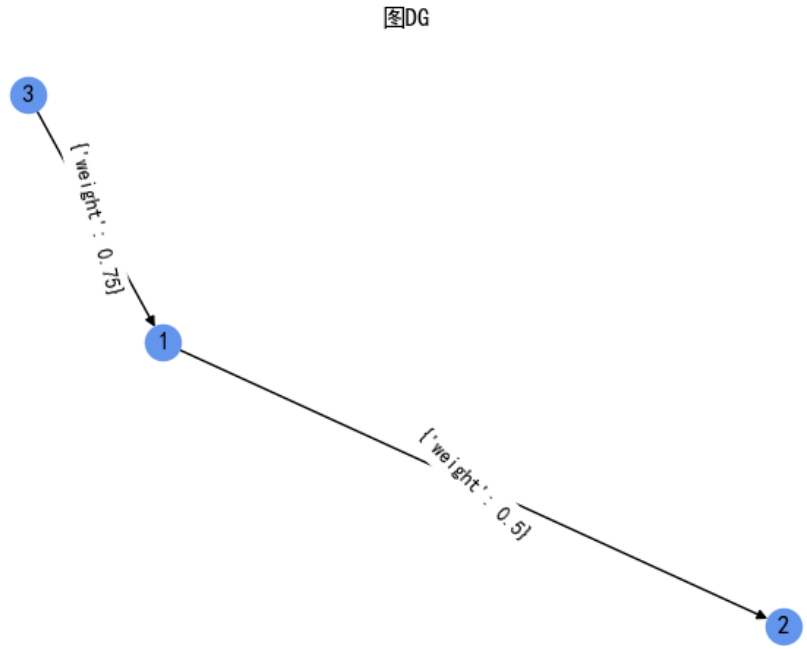


图 1：有向图DG的可视化展示

输出：

```
out_degree: 0.5
degree: 1.25
successors: [2]
neighbors: [2]
```

注意到有一些算法仅适用于有向图，如果需要将有向图转换为无向图进而进行某些操作，可以使用`Graph.to_undirected()`方法，即：

```
G = nx.DiGraph()
G.to_undirected()
```

或使用:

```
G = nx.DiGraph()
H = nx.Graph(G)
```

2. 多图（Multigraphs）

NetworkX提供了允许在任意一对结点之间存在多个边的类。在MultiGraph和NultiDiGraph类中，允许添加两条的相同边（比如两条边连接的结点相同但边存储数据不同）。对于某些应用情况来说这也许很有效，但许多算法在这些图上不能很好的定义。对于可以进行明确定义的方法，例如：MultiGraph.degree()可以在NetworkX实现。否则，应当用一种使得方法定义清晰的方式转换为标准的图。

```
>>> MG = nx.MultiGraph()  # 生成一个空的多图
>>> MG.add_weighted_edges_from([(1, 2, 0.5), (1, 2, 0.75), (2, 3, 0.5)])
>>> dict(MG.degree(weight='weight'))  # MG的度的字典
{1: 1.25, 2: 1.75, 3: 0.5}
>>> GG = nx.Graph() # 将多图MG转换成一般的图GG
>>> for n, nbrs in MG.adjacency():  # 对MG的结点进行迭代
...     for nbr, edict in nbrs.items():  # 迭代某一结点的所有邻接结点
...         minvalue = min([d['weight'] for d in edict.values()])  # 记录某结点的邻接结点中权值最小的结点权值
...         GG.add_edge(n, nbr, weight = minvalue)  # 将minvalue添加到图GG中
...
>>> nx.shortest_path(GG, 1, 3)  # 求图GG的最短路径
[1, 2, 3]
```

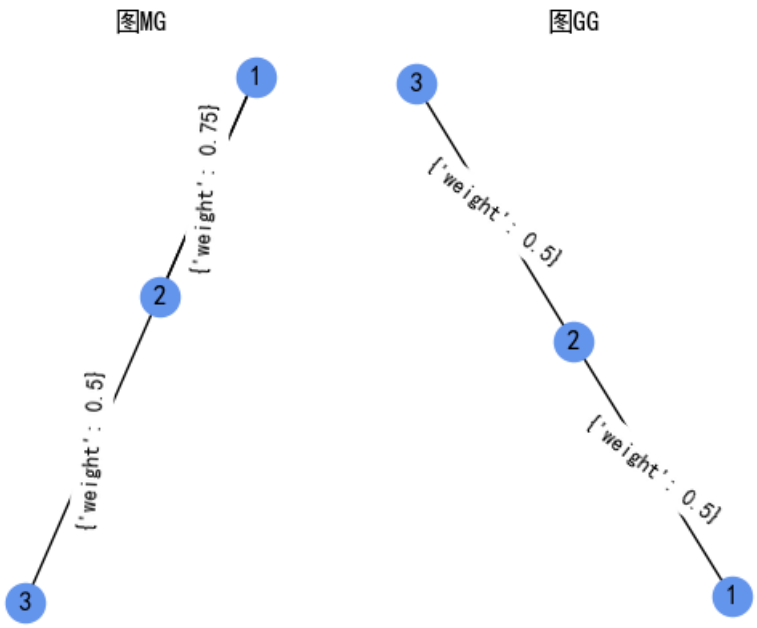


图 2：多图MG&图GG

> 注意：这里发现了一个关于NetworkX使用`nx_draw()`方法绘制多图时无法绘制相同边的技术问题[2],若您有解决该问题的方案，麻烦您和我取得联系（@673235106@qq.com）。

3. 图生成器和图操作

除了根据结点或边构造图之外，也可以通过以下方式生成图：

1. 应用经典的图操作，例如：

方法	具体含义
`subgraph(G, nbunch)`	由图G的nbunch[3]所生成的子图
`union(G1, G2)`	图G_1与图G_2的并集
`disjoint_union(G1,G2)`	假设所有结点都不相同时，图G_1与图G_2的并集
`cartesian_product(G1,G2)`	返回一个笛卡尔积[4]的图
`compose(G1,G2)`	标识两个图共同结点的组合图
`complement(G)`	图G的补图[5]
`create_empty_copy(G)`	返回一个相同图种类的空白拷贝
`to_undirected(G)`	返回一个图G的无向表示形式
`to_directed(G)`	返回一个图G的有向表示形式

2. 使用经典的小图声明方法，例如：

方法	具体含义
`nx.petersen_graph()`	生成一个petersen图[6]
`nx.tutte_graph()`	生成一个tutte图[7]
`nx.sedgewick_maze_graph()`	生成一个sedgewick maze图
`nx.tetrahedral_graph()`	生成一个四面体图[8]

3. 使用经典图的生成器，例如：

案例	具体含义
`K_5 = nx.complete_graph(5)`	生成一个5阶完全图K_5
`K_3_5 = nx.complete_bipartite_graph(3, 5)`	生成一个完全二部图$K_{3,5}$
barbell = nx.barbell_graph(10, 10)	生成一个barbell图[9]
lollipop = nx.lollipop_graph(10, 20)	生成一个lollipop[10]

4. 使用随机图生成器，例如：

```
>>> er = nx.erdos_renyi_graph(100, 0.15)
>>> ws = nx.watts_strogatz_graph(30, 3, 0.1)
>>> ba = nx.barabasi_albert_graph(100, 5)
>>> red = nx.random_lobster(100, 0.9, 0.9)
```

5. 使用常见的图格式（例如边列表、邻接列表、GML、GraphML、pickle模块、LEDA等）读取存储在文件中的图

```
>>> nx.write_gml(red, "path.to.file")
>>> mygraph = nx.read_gml("path.to.file")
```

4. 图的分析

图G的结构可以使用许多种图方法的python函数来分析，例如：

```
>>> G = nx.Graph()
>>> G.add_edges_from([(1, 2), (1, 3)])
>>> G.add_node("spam")       # 添加结点 "spam"
>>> list(nx.connected_components(G))  # 连通分量
[{1, 2, 3}, {'spam'}]
>>> sorted(d for n, d in G.degree())  # 图G的度的排序
[0, 1, 1, 2]
>>> nx.clustering(G)  # 聚类
{1: 0, 2: 0, 3: 0, 'spam': 0}
```

同时对于在需要遍历二元组：（结点，值）的使用场景中，可以使用Python的`dict`数据结构存储相关信息，例如：

```
>>> sp = dict(nx.all_pairs_shortest_path(G))
>>> sp[3]
{3: [3], 1: [3, 1], 2: [3, 1, 2]}
```

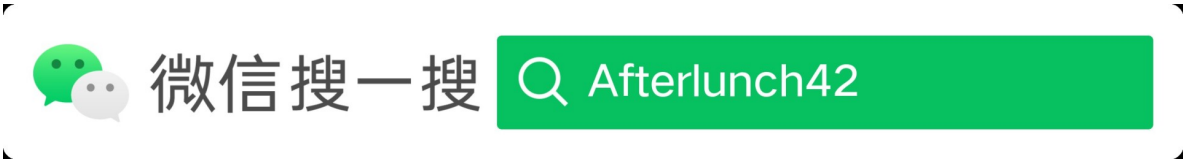
NetworkX已实现的图论相关算法可以参考：NetworkX官方文档-reference-algorithms[11]

到这里，NetworkX工具的所有入门知识就到这里结束啦，完结撒花~

在NetworkX中，关于图的可视化部分在之前的文章中已有简单提及。事实上NetworkX调用了Matplotlib的相关API，熟悉Matplotlib的朋友应该很快就能上手。这里就不过多提及了。

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] 相关问题（stack overflow）：Plotting directed graphs in Python in a way that show all edges separately

> https://stackoverflow.com/questions/10379448/plotting-directed-graphs-in-python-in-a-way-that-show-all-edges-separately

[3] NetworkX 1.9.1 reference-glossary-nbunch

> https://networkx.github.io/documentation/networkx-1.9.1/reference/glossary.html

[4] wolfram: Graph Cartesian Product

> https://mathworld.wolfram.com/GraphCartesianProduct.html

[5] wolfram: Graph Complement

> https://mathworld.wolfram.com/GraphComplement.html

[6] wolfram: Petersen Graph

> https://mathworld.wolfram.com/PetersenGraph.html

[7] wolfram: Tuttes Graph

> https://mathworld.wolfram.com/TuttesGraph.html

[8] wolfram: Tetrahedral Graph

> https://mathworld.wolfram.com/TetrahedralGraph.html

[9] wolfram: Barbell Graph

> https://mathworld.wolfram.com/BarbellGraph.html

[10] wolfram: Lollipop Graph

> https://mathworld.wolfram.com/LollipopGraph.html
> 噗~

[11] NetworkX官方文档-reference-algorithms

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