

14

Types of Graphs

常见图

用 NetworkX 绘制常见图，并了解特性



今天，是别人的；我苦苦耕耘的未来，是我的。

The present is theirs; the future, for which I really worked, is mine.

—— 尼古拉·特斯拉 (Nikola Tesla) | 发明家、物理学家 | 1856 ~ 1943



- networkx.balanced_tree() 创建平衡树图
- networkx.barbell_graph() 创建哑铃型图
- networkx.binomial_tree() 创建二叉图
- networkx.bipartite.gnmk_random_graph() 创建随机二分图
- networkx.bipartite_layout() 二分图布局
- networkx.circular_layout() 圆周布局
- networkx.complete_bipartite_graph() 创建完全二分图
- networkx.complete_graph() 创建完全图
- networkx.complete_multipartite_graph() 创建完全多向图
- networkx.cycle_graph() 创建循环图
- networkx.ladder_graph() 创建梯子图
- networkx.lollipop_graph() 创建棒棒糖型图
- networkx.multipartite_layout() 多向图布局
- networkx.path_graph() 创建路径图
- networkx.star_graph() 创建星型图
- networkx.wheel_graph() 创建轮型图

本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger: <https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com



14.1 常见图类型

本书前文已经介绍了几种图，本章将用 NetworkX 可视化工具来帮助我们理解常用图类型及基本性质。图 1 给出几个用 NetworkX 绘制的图。下面让我们聊聊其中比较常用的几种图。

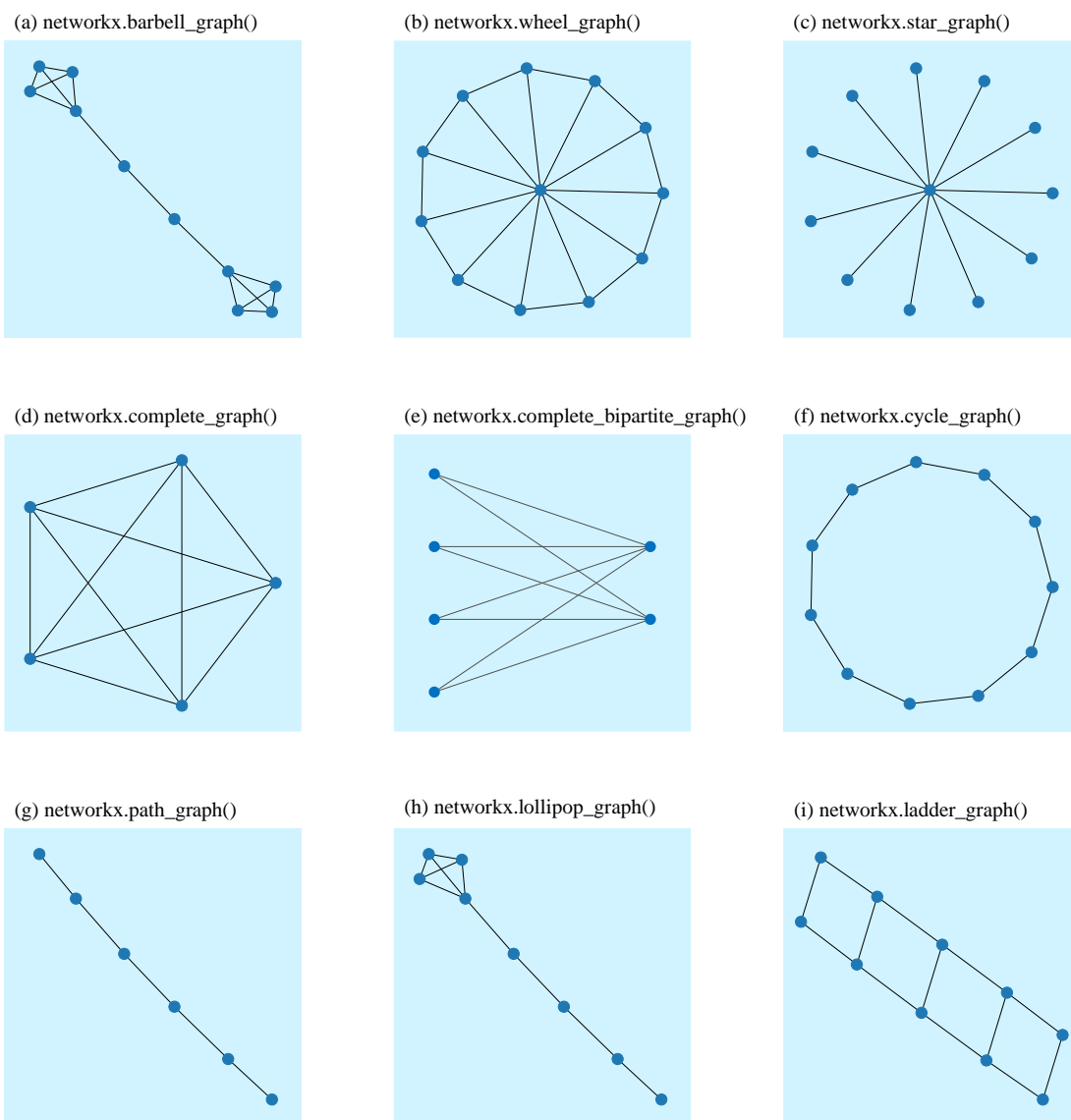


图 1. NetworkX 可视化的一些常见图类型

14.2 完全图

完全图 (complete graph) 是指每一对不同的节点都有一条边相连，形成了一个全连接的图。换句话说，如果一个无向图中的每两个节点之间都存在一条边，那么这个图就是一个完全图。

本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

一个完全图有很多边，具体边的数量可以通过组合学的方式计算。如果一个完全图有 n 个节点，每个节点对应 $(n-1)$ 条边，那么这个完全图将有 $n(n-1)/2$ 条边。

比如，一个有 3 个节点的完全图，它包含 $3(3-1)/2 = 3$ 条边，每一对节点都有一条边连接。

图 2 所示为一组完全图；注意，这些图都是无向图。

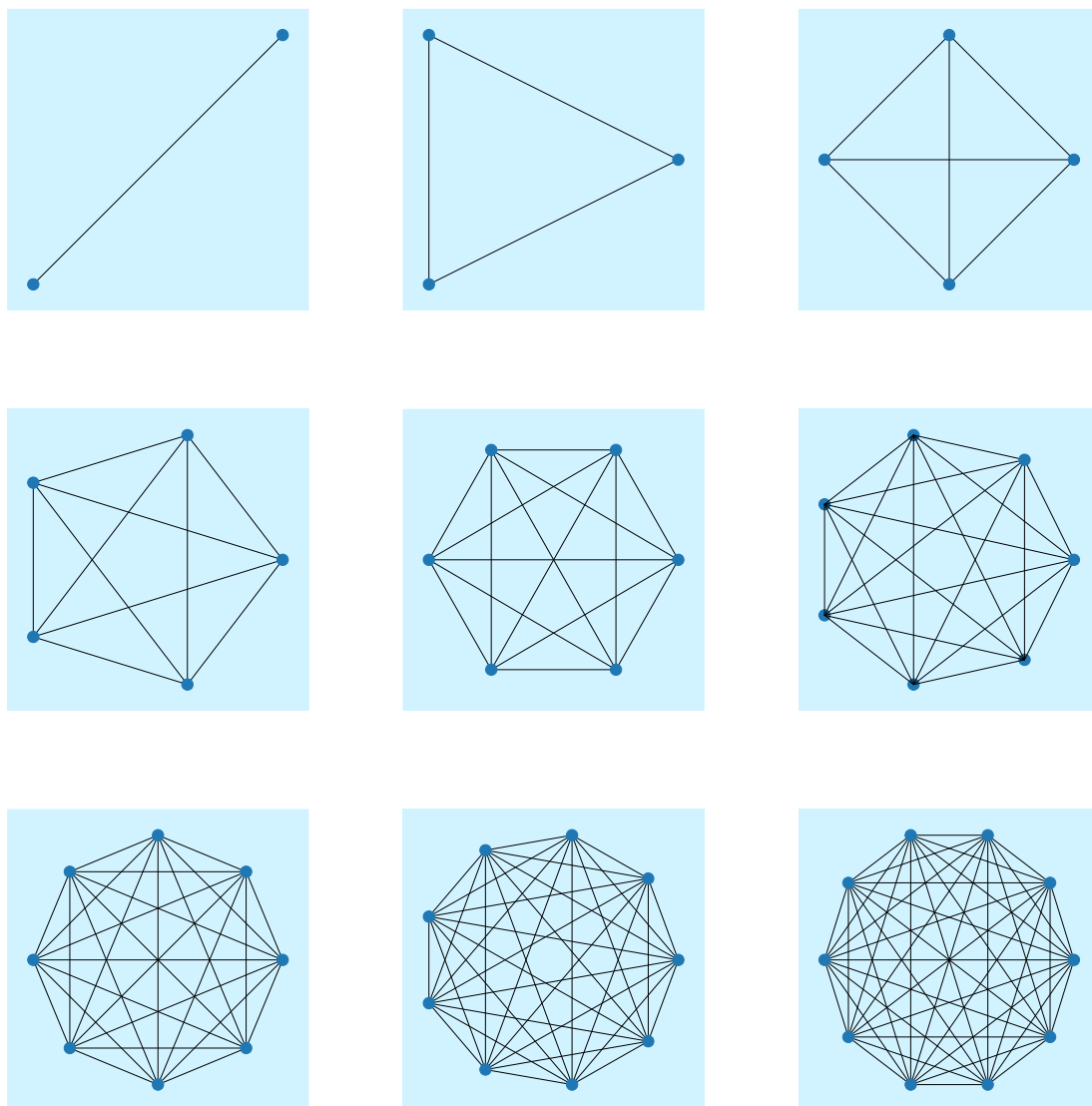


图 2. 一组完全图

代码 1 绘制图 2，下面聊聊其中关键语句。

- a** 用 `networkx.complete_graph()` 创建完全图。
- b** 用 `networkx.circular_layout()` 构造环形布局位置，结果为字典。
- c** 用 `networkx.draw_networkx()` 绘制完全图。

```

import networkx as nx
import matplotlib.pyplot as plt

# for循环，绘制9幅完全图
for num_nodes in range(2,11):

    # 创建完全图对象
    G_i = nx.complete_graph(num_nodes)

    # 环形布局
    pos_i = nx.circular_layout(G_i)

    # 可视化，请大家试着绘制 3 * 3 子图布局
    plt.figure(figsize = (6,6))
    nx.draw_networkx(G_i,
                     pos = pos_i,
                     with_labels = False,
                     node_size = 28)
    plt.savefig(str(num_nodes) + '_完全图.svg')

```

代码 1. 绘制完全图 | Bk6_Ch14_01.ipynb

图 3 所示为对完全图不同边分别着色的案例，请大家自行学习。

https://networkx.org/documentation/stable/auto_examples/drawing/plot_rainbow_coloring.html

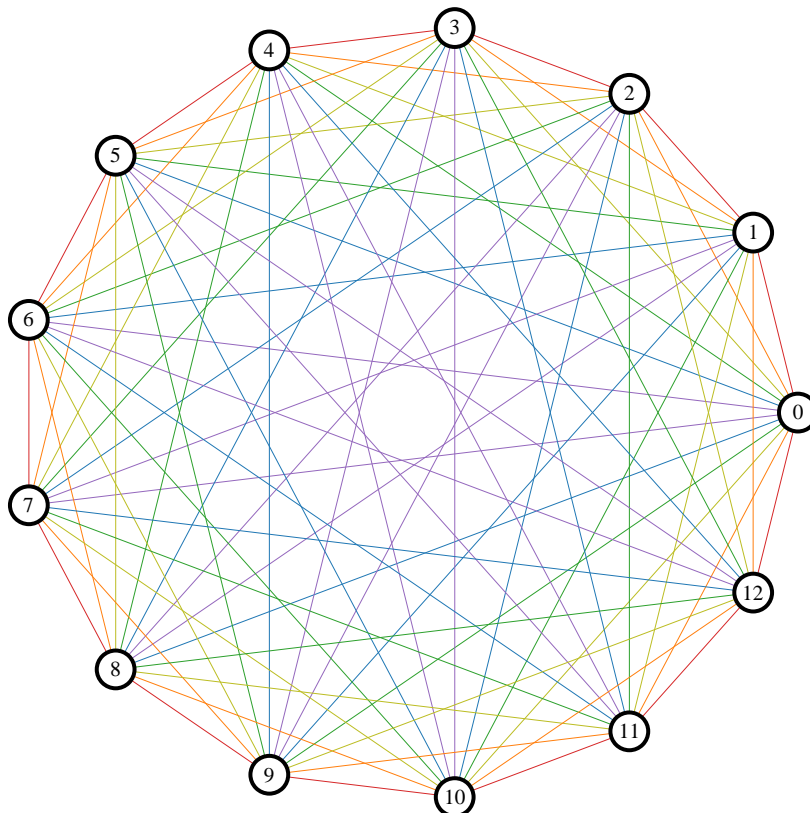


图 3. 对完全图边着色

有了完全图，我们就可以很容易定义简单图的**补图** (graph complement)。两幅图有相同节点，两者没有相同边，但是把两幅图的边结合起来是一幅完全图；这样我们就称两者互为补图。图 G 的补图记作 \bar{G} 或 G^c 。图 4 展示图、补图、完全图三者边的关系。图 5 可视化用 NetworkX 计算并可视化图和补图。

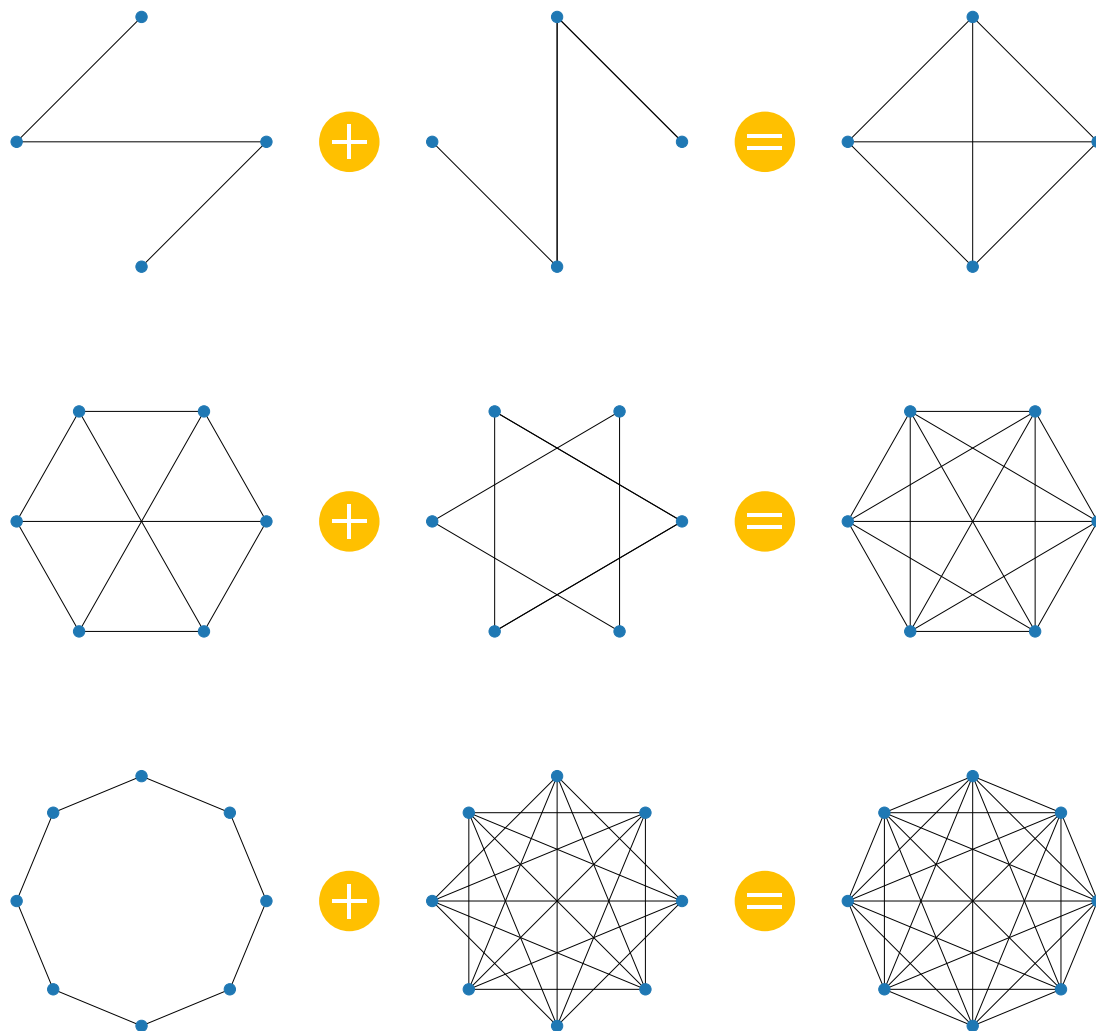
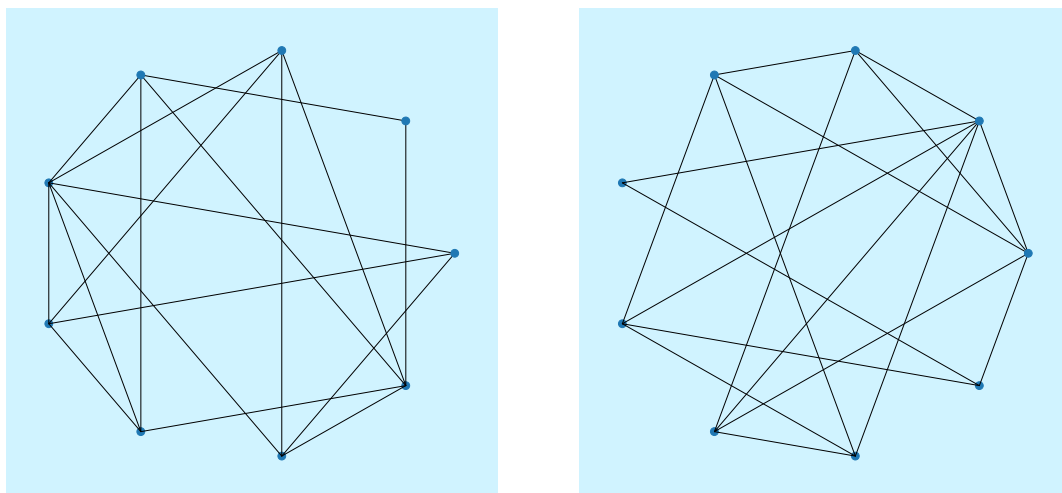


图 4. 图、补图、完全图三者关系，只考虑边的叠加



本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger: <https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

图 5. 用 NetworkX 计算并可视化图和补图

代码 2 绘制图 5，下面聊聊其中关键语句。

- a** 用 `random.sample()` 从由边构成的 `list` 中随机抽取 18 条边，正好是 9 个节点完全图（36 条边）边数的一半。
- b** 将随机抽取的 18 条边删除。
- c** 用 `networkx.complement()` 计算图 G 的补图。

```
import networkx as nx
import matplotlib.pyplot as plt
import random

# 创建完全图
G = nx.complete_graph(9)
print(len(G.edges))

# 随机删除一半边
a edges_removed = random.sample(list(G.edges), 18)
b G.remove_edges_from(edges_removed)

# 环形布局
pos = nx.circular_layout(G)

plt.figure(figsize = (6,6))
nx.draw_networkx(G,
                 pos = pos,
                 with_labels = False,
                 node_size = 28)
plt.savefig('图.svg')

c G_complement = nx.complement(G)
# 补图

# 可视化补图
plt.figure(figsize = (6,6))
nx.draw_networkx(G_complement,
                 pos = pos,
                 with_labels = False,
                 node_size = 28)
plt.savefig('补图.svg')
```

代码 2. 绘制图和补图 |  Bk6_Ch14_02.ipynb

14.3 二分图

在无向图中，**二分图** (bipartite graph, biograph)，也叫二部图，是指可以将图的所有节点划分为两个互不相交的子集，使得图中的每条边都连接一对不同子集中的节点。

本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

换句话说，如果一个无向图的节点集合 V 可以被分为两个子集 U 和 W ，使得图中的每条边 (u, v) 都满足 u 属于 U ， v 属于 W ，或者反过来，即 u 属于 W ， v 属于 U ，那么这个图就是一个二分图。

图 6 所示为 4 个二分图的例子。以图 6 (a) 为例，这幅无向图的所有边都在蓝色节点和黄色节点之间。并且，蓝色节点之间不存在任何边；同样，黄色节点之间也不存在任何边。

图 6 (c) 和 (d) 的这两幅二分图则显得有些不同；在这两幅无向图中，任意一对蓝色节点和黄色节点之间都存在一条边。我们管这类二分图叫做完全二分图。

二分图在实践中很常用。以图 6 为例，蓝色点可以代表若干人选，黄色点可以代表不同任务；边则代表人-任务的匹配关系。

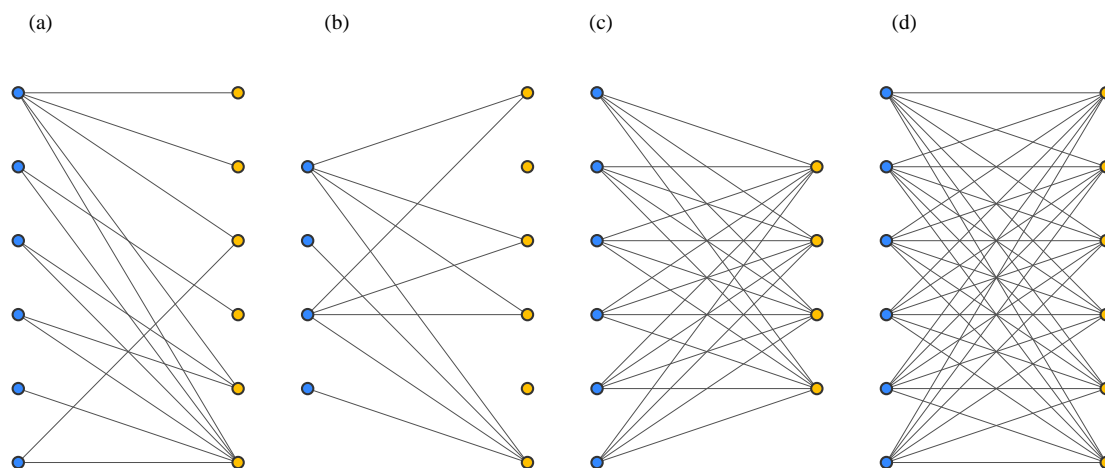


图 6. 4 个二分图

完全二分图 (complete bipartite graph) 在二分图基础上更进一步， U 中任意节点与 W 中任意节点均有且仅有唯一一条边相连。显然， U 中任意两个节点之间不相连， W 中任意两个节点之间也不相连。

代码 3 绘制图 7，节点和边都是手动设置。下面聊聊这段代码。

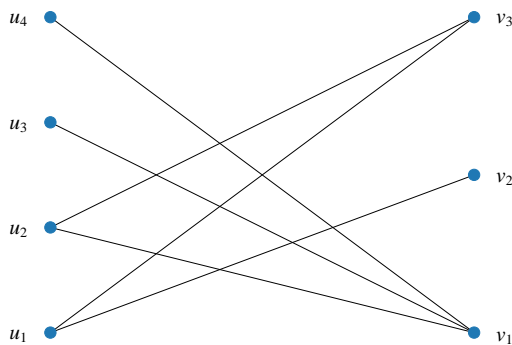


图 7. 用 NetworkX 绘制的二分图，手动设置节点和边

a 增加二分图第一节点集中的节点。

b 增加二分图第二节点集中的节点。

本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger: <https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

- c 增加二分图边。
- d 用 `networkx.algorithms.is_bipartite()` 判断图是否为二分图。
- e 用 `networkx.bipartite_layout()` 设置二分图节点布局，输入中包含一个子集集合。

图 6 (c) 和 (d) 两幅完全二分图则可以用 `networkx.complete_bipartite_graph()` 创建，请大家自行学习使用这个函数。

```
import networkx as nx
import matplotlib.pyplot as plt
from networkx.algorithms import bipartite

G = nx.Graph()


# 增加节点
a G.add_nodes_from(['u1', 'u2', 'u3', 'u4'],
                    bipartite=0)

b G.add_nodes_from(['v1', 'v2', 'v3'],
                    bipartite=1)

# 增加边
c G.add_edges_from([('u1', 'v3'),
                    ('u1', 'v2'),
                    ('u4', 'v1'),
                    ('u2', 'v1'),
                    ('u2', 'v3'),
                    ('u3', 'v1')])

# 判断是否是二分图
d bipartite.is_bipartite(G)

# 可视化
e pos = nx.bipartite_layout(G,
                           ['u1', 'u2', 'u3', 'u4'])
nx.draw_networkx(G, pos = pos, width = 2)
```

代码 3. 绘制二分图，手动添加节点和边 |  Bk6_Ch14_03.ipynb

代码 4 绘制图 8，下面聊聊其中关键语句。

- a 用 `networkx.bipartite.gnmk_random_graph(6, 8, 16, seed=88)` 生成随机二分图。其中，6 是二分图第一子集节点数，8 是第二子集节点数，16 为边数，seed 设置随机数种子。
- b `networkx.bipartite.sets(G)[0]` 取出二分图第一子集节点集合；同理，`networkx.bipartite.sets(G)[1]` 取出二分图第二子集节点集合。
- c 用 `networkx.bipartite_layout()` 生成二分图节点布局。z

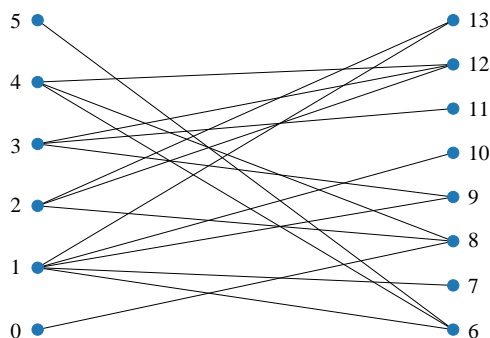


图 8. 用 NetworkX 绘制的二分图，随机生成

```
import networkx as nx
import matplotlib.pyplot as plt
from networkx.algorithms import bipartite

a G = nx.bipartite.gnmk_random_graph(6, 8, 16,
                                     seed=88)

# 判断是否是二分图
bipartite.is_bipartite(G)

# 取出节点第1子集
b left = nx.bipartite.sets(G)[0]
# {0, 1, 2, 3, 4, 5}

# 生成二分图布局
c pos = nx.bipartite_layout(G, left)

# 可视化
nx.draw_networkx(G, pos = pos, width = 2)
```

代码 4. 绘制二分图，随机生成 | Bk6_Ch14_04.ipynb

二分图可以进一步推广得到**多分图** (multi-partite graph); 同样，完全二分图也可以推广到**完全多分图** (complete multi-partite graph)。

图 9 所示为用 NetworkX 绘制的多分图；准确来说，这是一幅三分图。从左到右三个节点子集的节点数分别为 3、6、9。请大家自行学习 Bk6_Ch14_05.ipynb，并试着绘制调整参数，绘制其他多分图。

图 10 所示为多图布局，并且对不同分层节点分别着色，请大家自行学习这个案例。

https://networkx.org/documentation/stable/auto_examples/drawing/plot_multipartite_graph.html

请大家注意，图 10 并不是严格意义的多分图。

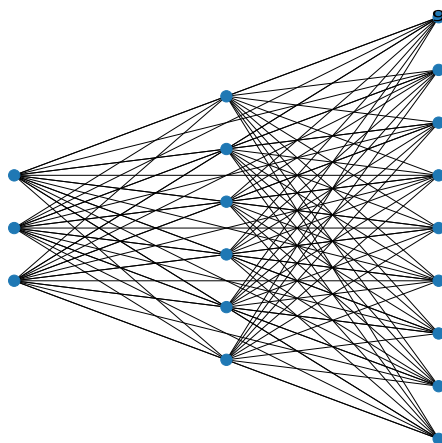


图 9. 用 NetworkX 绘制的多分图

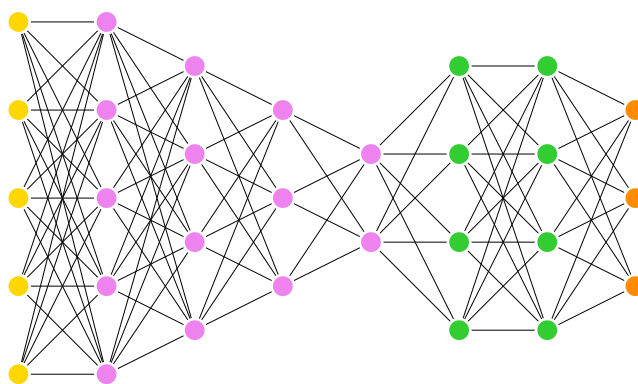


图 10. 用 NetworkX 绘制的多图布局

14.4 正则图

对于无向图，**正则图** (regular graph) 是指图中每个节点的度都相同的图。如果一个图是正则的，那么它被称为正则图，并且其度被称为图的正则度。图 11 所示为一组正则图。

正则图可以分为两种类型：

- ▶ **正则图** (regular graph)：如果所有节点的度都相同，那么这个图被称为正则图。
- ▶ **k -正则图** (k -regular graph)：如果所有节点的度都是 k ，那么这个图被称为 k -正则图。

图 12 所示为用 NetworkX 生成的一组随机正则图，请大家自行学习 Bk6_Ch14_06.ipynb。

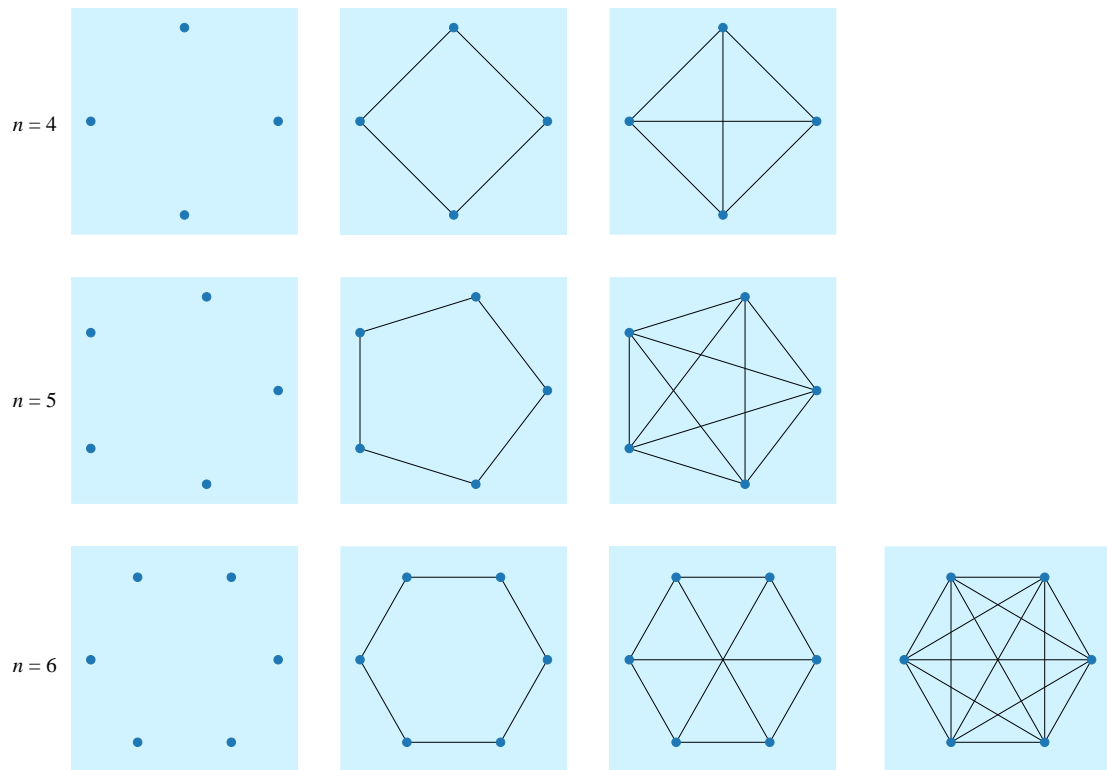


图 11. 一组正则图

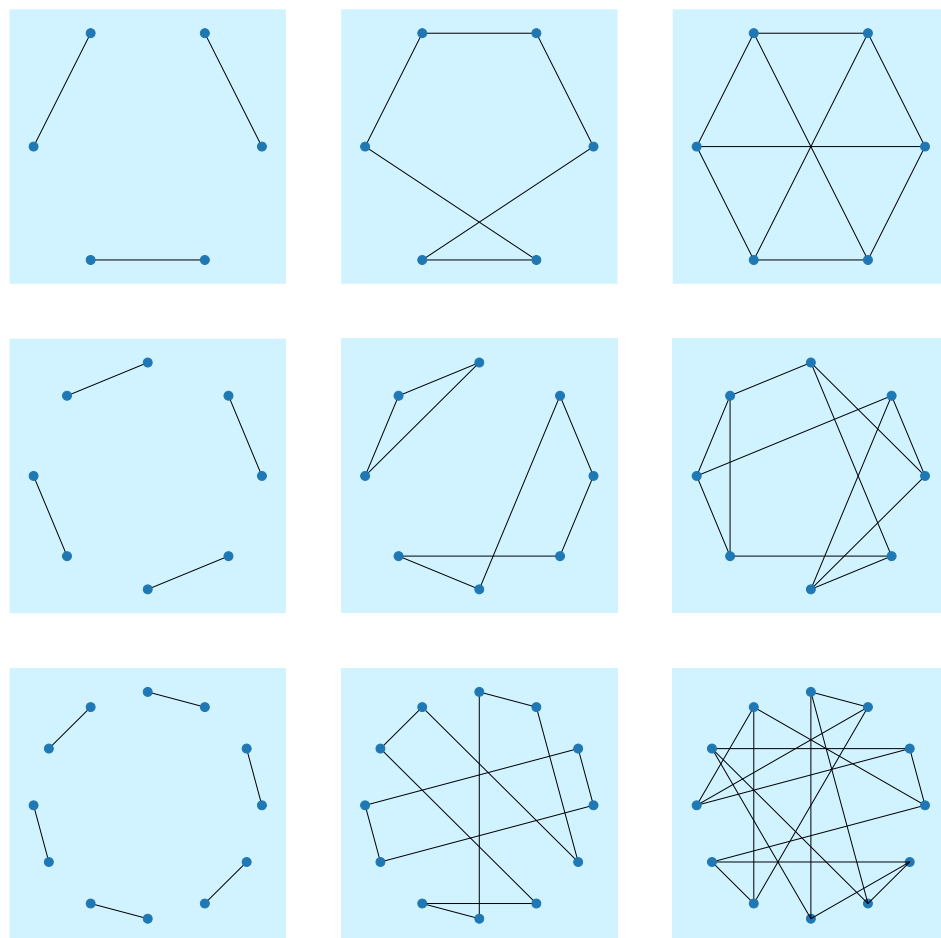


图 12. 用 NetworkX 生成的随机正则图

本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

14.5 树

在无向图中，一个**树**(tree)是一种特殊的无环连通图。换句话说，一个无向图是树，当且仅当图中的每两个节点之间存在唯一的路径，并且图是连通的，即，从任意一个节点出发，都可以到达图中的任意其他节点)。

树形数据再常见不过。地球物种分类树是一种用来组织和分类地球上生物多样性的树形结构。这个树形结构基于物种的共同特征，将生物按照层次分类，从大类别(域，domain)到小类别(物种，species)。这种分类系统有助于科学家理解生物之间的亲缘关系，以及它们是如何进化和演化的。

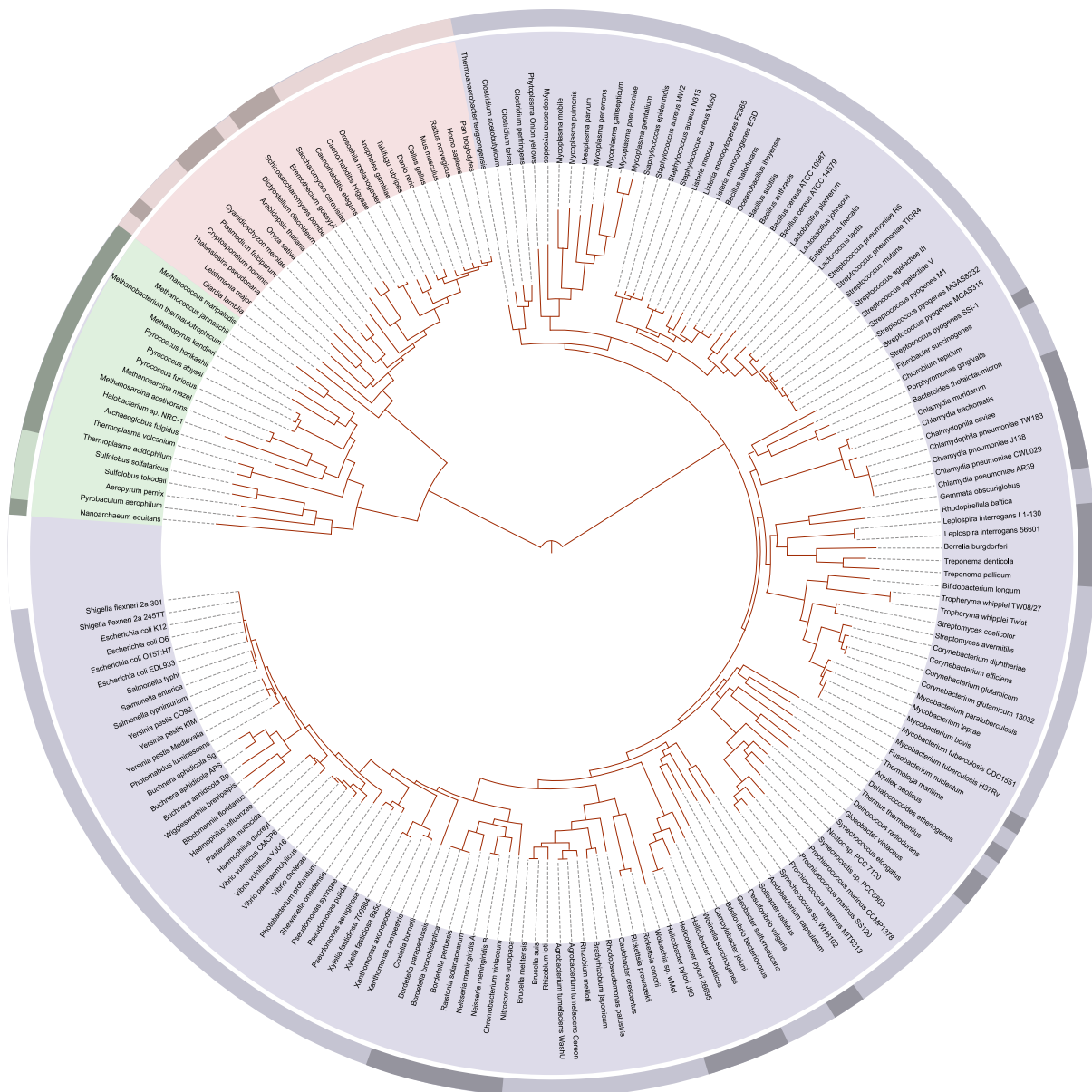


图 13. 生命之树，图片来自 wikipedia.org

本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger: <https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

树在计算机科学和图论中有广泛应用，例如在数据结构中作为搜索树、在算法中作为树的遍历和操作等。特别是，无向树的每一对节点之间都有唯一的简单路径，这使得树结构具有一些良好的性质，方便在算法和数据结构中使用。

下一章将专门介绍树，特别是其在机器学习算法中的应用。

图 14 所示为环形布置的树。

图 15 是用 `networkx.balanced_tree()` 创建的平衡树，请大家自行学习 `Bk6_Ch14_07.ipynb`。

图 16 是用 `networkx.binomial_tree()` 创建的二叉树，请大家自行学习 `Bk6_Ch14_08.ipynb`。

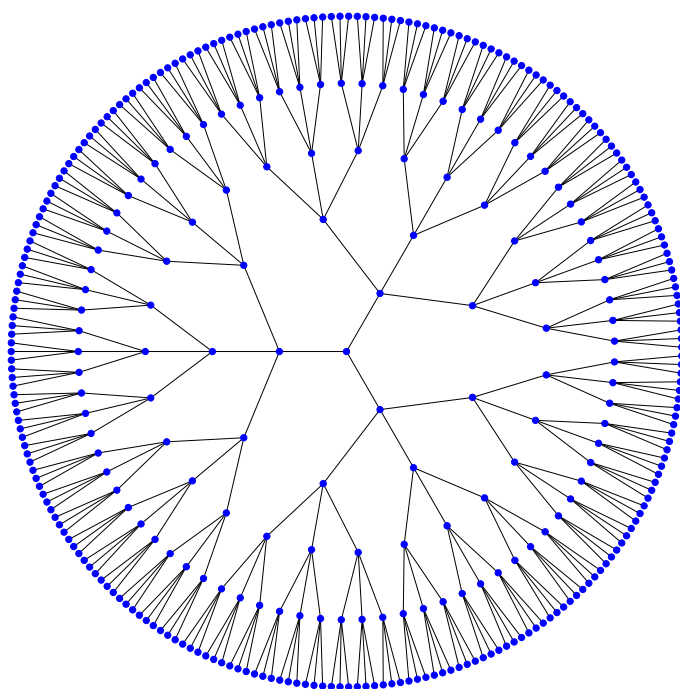


图 14. 环形布置的树

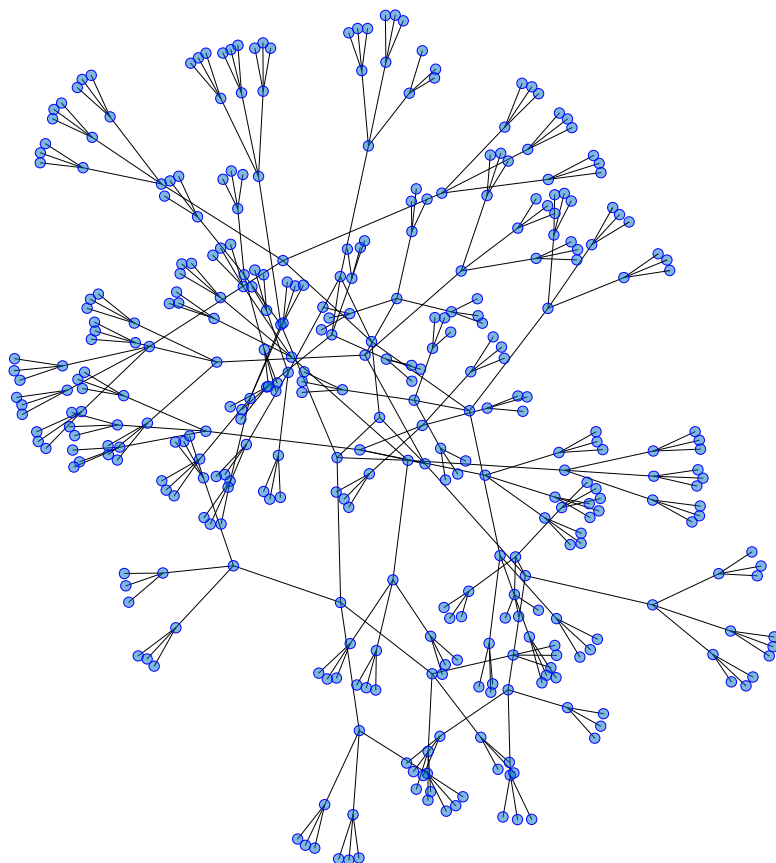


图 15. 平衡树，用 `networkx.balanced_tree()` 创建

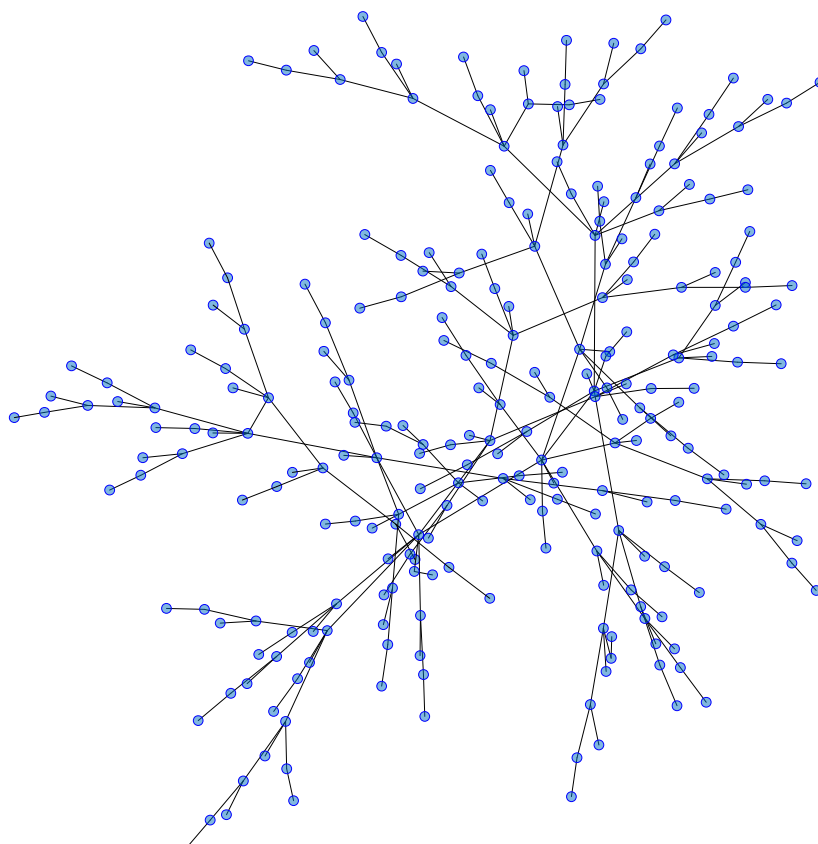


图 16. 二叉树，用 `networkx.binomial_tree()` 创建

本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger: <https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

星图 (star graph) 相当于一种特殊的树，图 17 给出一组示例。Bk6_Ch14_09.ipynb 绘制图 17，请大家自行学习。

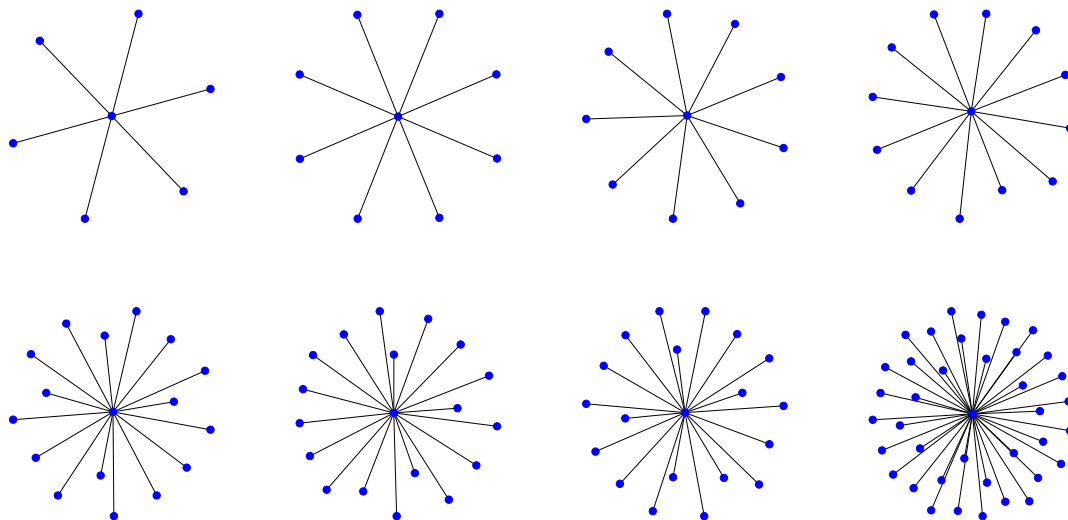


图 17. 星图

链图 (path graph) 也可以看成一种特殊的树，图 18 给出一组示例。Bk6_Ch14_10.ipynb 绘制图 18，请大家自行学习。

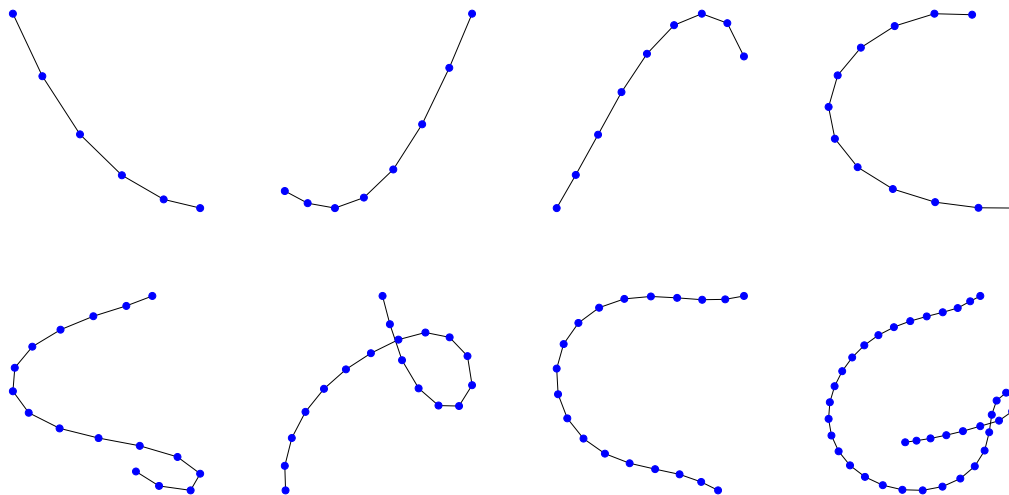


图 18. 链图

14.6 柏拉图图

《数学要素》介绍过**柏拉图立体** (Platonic solid)，也叫正多面体。正多面体的每个面全等，均为**正多边形** (regular polygons)。图 19 所示为五个柏拉图立体，包括**正四面体** (tetrahedron)、**正六面体** (cube)、

本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

正八面体 (octahedron)、**正十二面体** (dodecahedron) 和**正二十面体** (icosahedron)。图 20 所示为五个正多面体展开得到的平面图形。

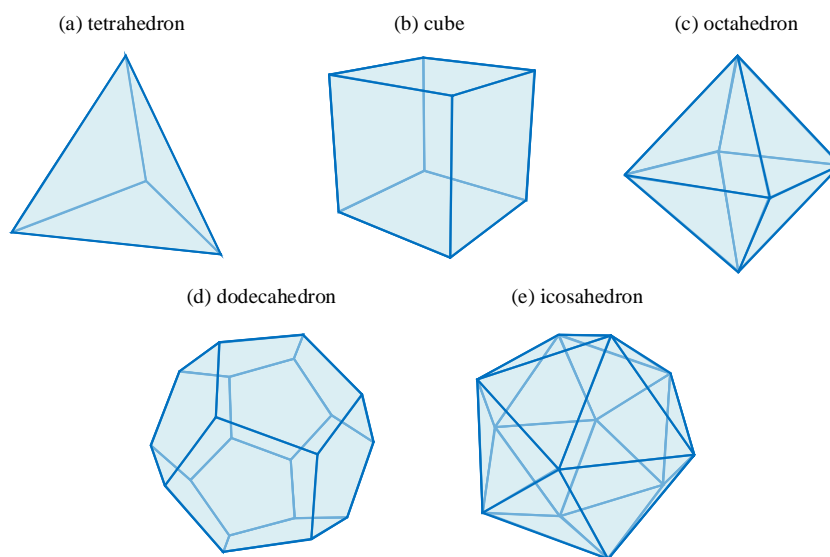


图 19. 五个正多面体，图片来自《数学要素》

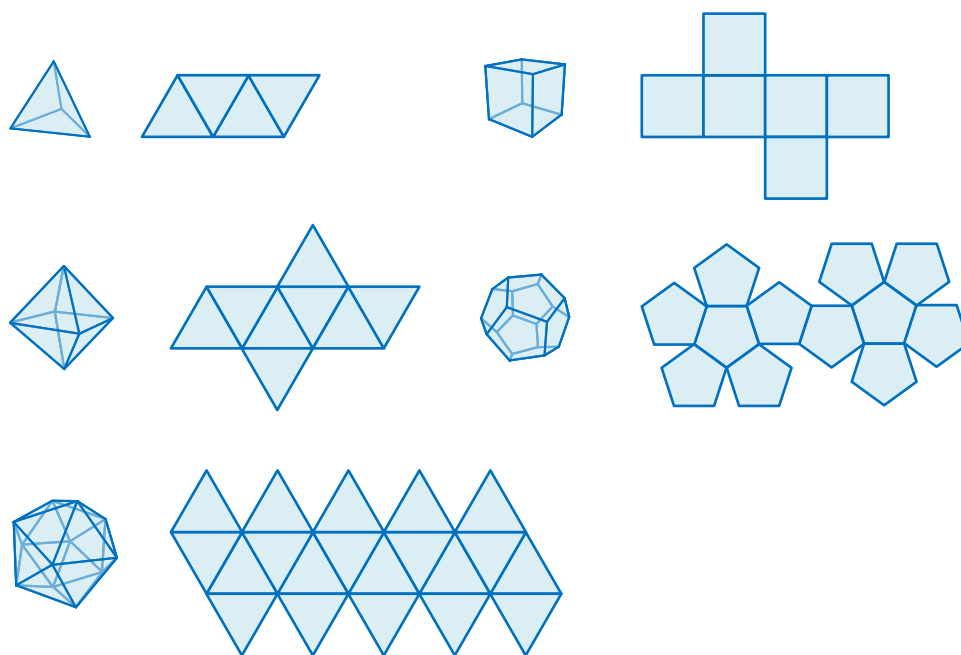


图 20. 五个正多面体展开得到的平面图形，图片来自《数学要素》

而本节要介绍的柏拉图图 (Platonic graph) 就是基于柏拉图立体骨架的图。

图 21 展示了五种柏拉图——**正四面体图** (tetrahedral graph)、**正六面体图** (cubical graph)、**正八面体图** (octahedral graph)、**正十二面体图** (dodecahedral graph) 和**正二十面体图** (icosahedral graph)。

图 21 所示为用 NetworkX 绘制的五个柏拉图图。表 1 总结五个正多面体的结构特征。

本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger: <https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

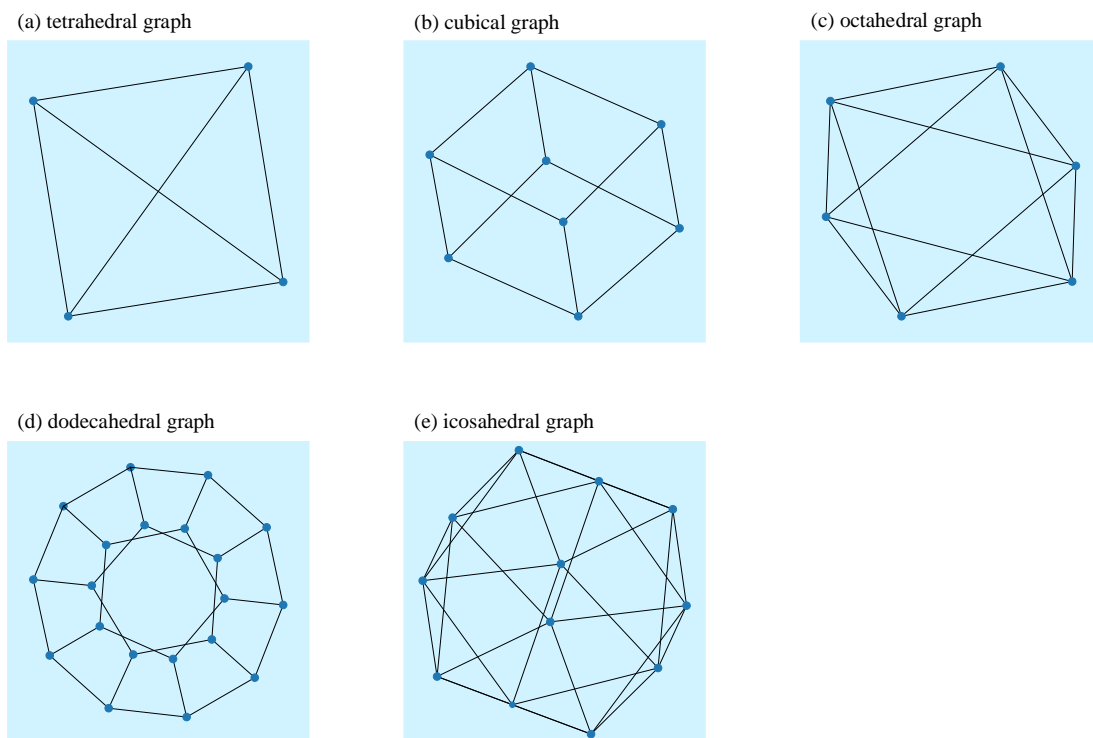


图 21. 用 NetworkX 绘制的五个柏拉图图

表 1. 柏拉图图的特征

柏拉图图	柏拉图立体	节点数	边数
Tetrahedral graph	Tetrahedron	4	6
Cubical graph	Cube	8	12
Octahedral graph	Octahedron	6	12
Dodecahedral graph	Dodecahedron	20	30
Icosahedral graph	Icosahedron	12	30

代码 5 绘制图 21，下面聊聊其中关键语句。

- a** 用 `networkx.tetrahedral_graph()` 创建正四面体图。
- b** 用 `networkx.cubical_graph()` 创建正六面体图。
- c** 用 `networkx.octahedral_graph()` 创建正八面体图。
- d** 用 `networkx.dodecahedral_graph()` 创建正十二面体图。
- e** 用 `networkx.icosahedral_graph()` 创建正二十面体图。

```

import networkx as nx
import matplotlib.pyplot as plt

# 自定义可视化函数
def visualize_G(G, fig_name):
    plt.figure(figsize = (6,6))
    nx.draw_networkx(G,
                     pos = nx.spring_layout(G),
                     with_labels = False,
                     node_size = 28)
    plt.savefig(fig_name + '.svg')

# 正四面体图
a tetrahedral_graph = nx.tetrahedral_graph()

# 可视化
visualize_G(tetrahedral_graph,
            'tetrahedral_graph')


# 正六面体图
b cubical_graph = nx.cubical_graph()

# 正八面体图
c octahedral_graph = nx.octahedral_graph()

# 正十二面体图
d dodecahedral_graph = nx.dodecahedral_graph()

# 正二十面体图
e icosahedral_graph = nx.icosahedral_graph()

```

代码 5. 创建并绘制柏拉图图 |  Bk6_Ch14_11.ipynb

平面化：任意两条不交叉

对于一个画在平面上的连通图，如果除在节点外，任意两边不交叉，这种图叫做**平面图** (planar graph)。

以这个标准看图 21，这 5 图柏拉图图似乎都不是平面图；但是，实际上，柏拉图图都是平面图。也就是说，它们都可以**平面化** (planar)。请大家自行学习 Bk6_Ch14_12.ipynb。

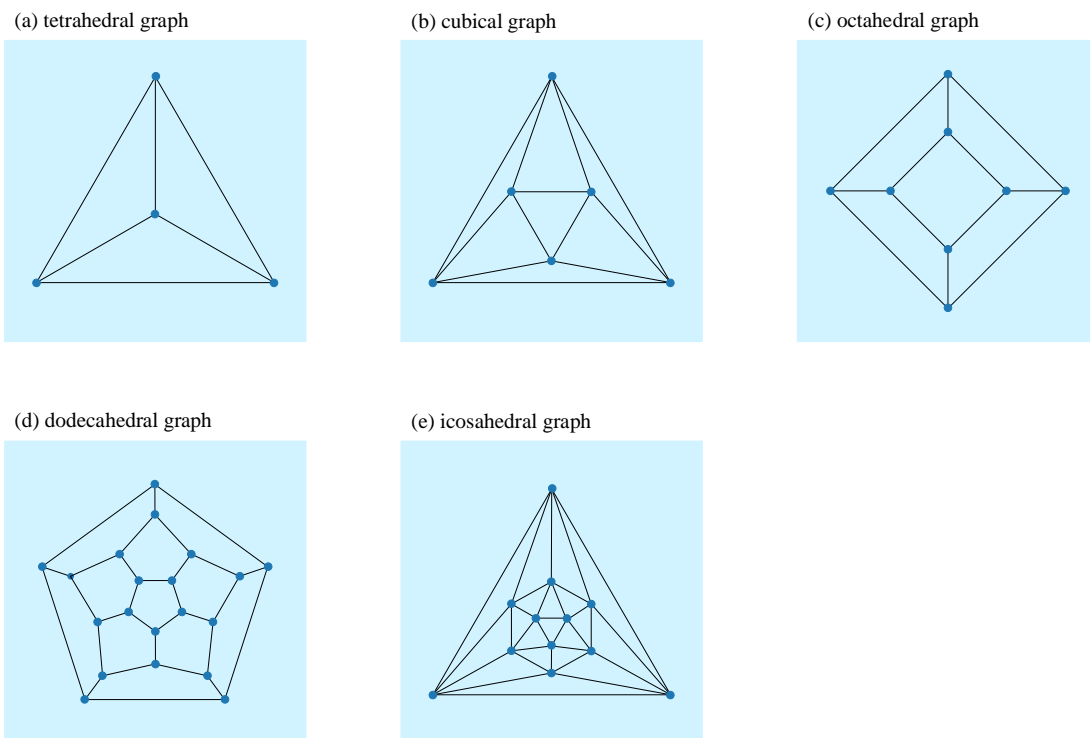


图 22. 五个柏拉图图，平面化

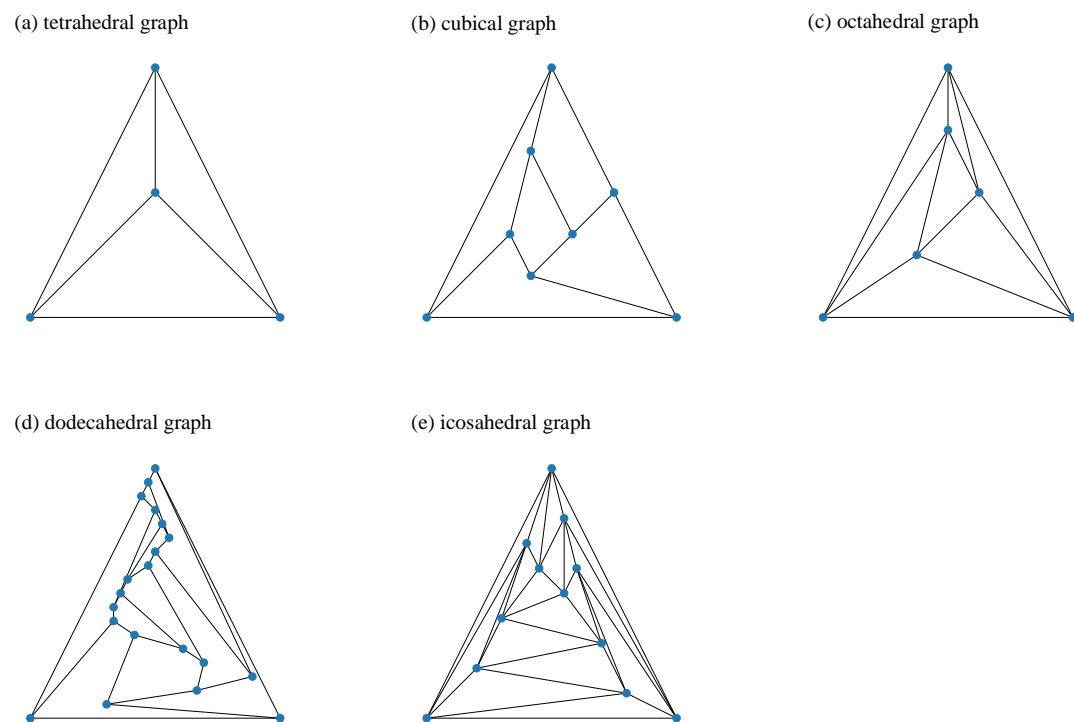


图 23. 五个柏拉图图，用 NetworkX 判断并完成平面化

本章介绍了几种常见的图以及它们的性质，并且和大家探讨如何用 NetworkX 完成这些图的可视化。本书后续还将介绍树在机器学习算法中的应用案例。