哈密顿问题

周游世界问题

- 1859 年，数学家哈密顿将正十二面体的每个顶点比作一个城市，连接两个顶点之间的边看作城市之间的交通线，提出周游世界问题：能否从某个城市出发沿交通线经过每个城市一次并且仅一次，最后回到出发点？

哈密顿图

- 哈密顿通路: 经过图中所有顶点的初级通路
- 哈密顿回路: 经过图中所有顶点的初级回路
- 哈密顿图: 有哈密顿回路的图

1. 必要条件

定理 ①: 若 G 是哈密顿图，则对于结点集 $V(G)$ 的任一非空真子集 $S \subset V(G)$ 有 $\omega(G-S) \le |S|$。其中 $G-S$ 表示在 G 中删去 S 中的结点后所构成的图，$\omega(G-S)$ 表示 $G-S$ 的连通分支数。

证明: 设 C 是 G 的一条哈密顿回路，C' 视为 G 的子图，在回路 C 中，每删去 S 中的一个结点，最多增加一个连通分支，且删去 S 中的第一个结点时分支数不变，所以有 $\omega(C-S) \le |S|$。又因为 C 是 G 的生成子图，所以 $C-S$ 是 $G-S$ 的生成子图，且 $\omega(G-S) \le \omega(C-S)$，因此 $\omega(G-S) \le |S|$。

这不是充分条件，反例是彼得松图，这段代码通过暴力搜索验证了彼得松图不是哈密顿图。

Python 代码验证

```
import networkx as nx
from itertools import permutations

def is_hamiltonian_cycle(graph, cycle):
    """
    检查给定的顶点排列是否构成哈密顿环
    :param graph: 待检查的图
    :param cycle: 顶点的排列
    :return: 如果是哈密顿环返回 True，否则返回 False
    """
    # 检查排列长度是否与顶点数相同
    if len(cycle) != len(graph.nodes):
        return False
```

```
    # 检查排列中的每对相邻顶点是否有边连接
    for i in range(len(cycle)):
        if not graph.has_edge(cycle[i], cycle[(i + 1) % len(cycle)]):
            return False

    return True

def main():
    # 生成彼得森图
    G = nx.petersen_graph()

    # 获取彼得森图的所有顶点
    nodes = list(G.nodes)

    # 尝试所有顶点排列，检查是否有哈密顿环
    for perm in permutations(nodes):
        if is_hamiltonian_cycle(G, perm):
            print("彼得森图有哈密顿环：", perm)
            return

    # 如果循环结束没有找到哈密顿环，输出彼得森图没有哈密顿环
    print("彼得森图没有哈密顿环")

if __name__ == "__main__":
    main()
```

2.简单的充分条件

设 G 是 $n(n\geq 3)$ 阶简单图，且对每一对顶点 $v, v^{'}$ 有

$$d(v)+d(v^{'})\geq n-1,$$

则图 G 有哈密顿链。

证明 先证明 G 是连通图。若 G 有两个或两个以上的连通部分，设其中之一有 n_1 个顶点，另一部分有 n_2 个顶点。分别从中各取一顶点 v_1、 v_2 ，则 $d(v_1)\leq n_1-1, d(v_2)\leq n_2-1$ 。故

$$d(v_1)+d(v_2)\leq n_1+n_2-2<n-1,$$

这与题设矛盾，所以 G 是连通图。

现证明存在哈密顿链。证明的方法实际上给出一种哈密顿链的构造步骤。设在 G 中有一条从 v_1 到 v_p 的链: $v_1v_2\cdots v_p$ 。如果有 v_1 或 v_p 与不在这条链上的一个顶点相邻，我们可扩展这条链，使它包含这个顶点。否则， v_1 和 v_p 都只与这条链上的顶点相邻，这时存在一个圈包含顶点 $v_1, v_2, \cdots, v_p$ 。假设与 v_1 点相邻的顶点集是 $\{v_{j_1}, v_{j_2}, \cdots, v_{j_k}\}$ ，这里 $v_{j_1}, v_{j_2}, \cdots, v_{j_k}$ 都是链 $v_1v_2\cdots v_p$ 中的点，且 $p<n$ 。

如果 v_1 与 v_p 相邻，则显然存在一个圈 $v_1 v_2 \cdots v_p v_1$。

如果 v_1 和 v_p 不相邻，则必然存在一点 $v_l (2 \leq l \leq p)$ 和 v_1 相邻，而 v_{l-1} 和 v_p 相邻。因为否则 v_p 最多只和 $p-k-1$ 个顶点相邻，即排除 v_{j_1-1}、 v_{j_2-1}、 $\cdots$、 v_{j_k-1} 和 v_p 自身，这样

$$d(v_1)+d(v_p) \leq k+(p-k-1)=p-1<n-1,$$

这与假设矛盾。因而存在 $v_1, v_2, \cdots, v_p$ 的圈 $v_1 v_l v_{l+1} \cdots v_p v_{l-1} v_{l-2} \cdots v_2 v_1$。 若 $p=n$，实际上已经存在一个哈密顿圈。若 $p<n$，由于 G 是连通的，所以必定存在一个不属于此圈的顶点 v' 与 $v_1 v_2 \cdots v_p$ 中的某个相邻。不断重复以上步骤直到得到一条 n-1 的链为止.

(Ore)设 G 是 $n(n \geq 3)$ 阶简单图，且对每一对不相邻的顶点 v, v' 有

$$d(v)+d(v') \geq n,$$

那么图 G 有哈密顿圈。

证明：当 $n=3$ 时，由所给条件知 G 一定是完全图 K_3，命题成立。设 $n \geq 4$，用反证法证明。设 G 是有 n 个顶点且满足度数条件却没有哈密顿圈的图。不妨设 G 是具有这种性质的边数最大的图，也就是说 G 添上一条边就具有哈密顿圈（否则 G 可以添加一些边，直到不能再添为止，加边后顶点的度数条件仍满足），由此得出在图 G 中有一条包含图中每一个顶点的哈密顿链，记为 $v_1 v_2 \cdots v_n$。则 v_1 与 v_n 不相邻，于是 在 $v_2, v_3, \cdots, v_{n-1}$ 中必有一点 v_i 使 v_1 与 v_i 相邻，v_n 与 v_{i-1} 相邻，如图所示。否则，有 $d(v_1)=k$ 个点 $v_{i_1}, v_{i_2}, \cdots, v_{i_k} (2 \leq i_1 \leq i_2 \leq \cdots \leq i_k \leq n-1)$ 与 v_1 相邻，而 v_n 与 $v_{i_1-1}, v_{i_2-1}, \cdots, v_{i_k-1}$ 都不相邻，从而

$$d(v_n) \leq n-1-k,$$

则

$$d(v_1)+d(v_n) \leq k+n-1-k=n-1<n,$$

这与条件矛盾。故 G 存在一条哈密顿圈 $v_1 v_2 \cdots v_{i-1} v_n v_{n-1} \cdots v_i v_1$。这又与假设矛盾，从而命题得证。

Python 验证代码

```
# -*- coding: utf-8 -*-
"""
Created on Tue Jun 11 21:06:35 2024

@author: 蔚元利
"""

import numpy as np
import networkx as nx
import matplotlib.pyplot as plt

def generate_ore_graph(n):
    """
```

```
    生成一个满足 Ore 性质的随机图
    """
    G = nx.Graph()
    G.add_nodes_from(range(n))

    # 随机生成边，确保满足 Ore 性质
    for u in range(n):
        for v in range(u + 1, n):
            if np.random.rand() < 0.5:
                G.add_edge(u, v)

    # 强制确保满足 Ore 性质
    for u in range(n):
        for v in range(u + 1, n):
            if not G.has_edge(u, v):
                if G.degree[u] + G.degree[v] >= n:
                    G.add_edge(u, v)

    return G

def find_hamiltonian_cycle(graph, n):
    def extend_path(path):
        while True:
            extended = False
            for v in range(n):
                if v not in path and graph.has_edge(path[-1], v):
                    path.append(v)
                    extended = True
                    break
            if not extended:
                break
        return path

    def find_cycle_from_path(path):
        y1, ym = path[0], path[-1]

        # 检查 y1 和 ym 是否相邻
        if graph.has_edge(y1, ym):
            if len(path) == n:
                return path
            else:
                return None

        # 找到一个不在 path 中的顶点 z，使得 z 与 y1 相邻
        z_candidates = [z for z in range(n) if z not in path and
graph.has_edge(z, y1)]
        if z_candidates:
            z = z_candidates[0]
            path = [z] + path
            path = extend_path(path)
```

```
        else:
            # 找到一个顶点 yk，使得 y1 和 yk 邻接，并且 ym 和 yk-1 邻接
            for k in range(1, len(path)):
                if graph.has_edge(y1, path[k]) and graph.has_edge(ym, 
path[k - 1]):
                    path = path[:k] + [ym] + path[k:]
                    path = extend_path([y1] + path)
                    break
            else:
                return None

        return find_cycle_from_path(path)

    path = extend_path([0])
    cycle = find_cycle_from_path(path)

    return cycle

def main():
    # 用户输入图的顶点数
    n = int(input("请输入顶点数（n > 3）："))
    if n <= 3:
        print("顶点数必须大于 3")
        return

    # 生成满足 Ore 性质的随机图
    while True:
        graph = generate_ore_graph(n)
        if all(graph.degree[u] + graph.degree[v] >= n for u in 
range(n) for v in range(u + 1, n) if not graph.has_edge(u, v)):
            break

    # 查找哈密顿链
    cycle = find_hamiltonian_cycle(graph, n)

    if cycle:
        print("找到哈密顿链：", cycle)
    else:
        print("未找到哈密顿链")

    # 绘制图形
    pos = nx.spring_layout(graph)
    nx.draw(graph, pos, with_labels=True, node_color='lightblue', 
edge_color='gray')
    if cycle:
        edges = [(cycle[i], cycle[(i+1)%n]) for i in 
range(len(cycle))]
        nx.draw_networkx_edges(graph, pos, edgelist=edges, 
edge_color='r', width=2)
    plt.show()
```

```
if __name__ == "__main__":
    main()
```

3. 对于平面图的必要条件

如果一个平面图有哈密顿圈 c, 用 f_i' 表示在 c 的内部的 i 边形的个数, 用 f_i'' 表示在 c 的外部的 i 边形的个数, 则 (1) $1\cdot f_3'+2\cdot f_4'+3\cdot f_5'+\cdots=n-2$; (2) $1\cdot f_3''+2\cdot f_4''+3\cdot f_5''+\cdots=n-2$; (3) $1\cdot\left(f_3'-f_3''\right)+2\cdot\left(f_4'-f_4''\right)+3\cdot\left(f_5'-f_5''\right)+\cdots=0.$

其中 n 为 G 的顶点数, 显然也是 c 的长. 证明 设 c 的内部有 d 条边. 由于 G 是平面图, 它的边都不相交, 所以一条边把它经过的面分成两部分. 设想这些边是一条一条地放进图里去的, 每面. 于是 c 的内部的面的总数为

$$f_2'+f_3'+f_4'+f_5'+\cdots=d+1.$$

在 c 内每个 i 边形中记上数字 i, 各面所记数字之和就是围成这些面的边的总数, c 内部的每一条边都被数了两次, 而 c 上的 n 条边, 每条边都只数了一次,于是

$$2f_2'+3f_3'+4f_4'+5f_5'+\cdots=2d+n.$$

(2)式减去(1)式的两倍, 得

$$1\cdot f_3'+2\cdot f_4'+3\cdot f_5'+\cdots=n-2.$$

类似地可以推得

$$1\cdot f_3''+2\cdot f_4''+3\cdot f_5''+\cdots=n-2.$$

(3)、(4)两式相减即得

$$1\cdot\left(f_3'-f_3''\right)+2\cdot\left(f_4'-f_4''\right)+3\cdot\left(f_5'-f_5''\right)+\cdots=0$$

Python 验证代码

```
import networkx as nx
import matplotlib.pyplot as plt
from itertools import permutations

def generate_random_planar_graph(num_nodes):
    """
    生成一个随机平面图
    :param num_nodes: 节点数
    :return: 平面图
    """
    G = nx.erdos_renyi_graph(num_nodes, 0.4)
    while not nx.check_planarity(G)[0]:
        G = nx.erdos_renyi_graph(num_nodes, 0.4)
```

```python
    pos = nx.planar_layout(G)
    return G, pos

def is_hamiltonian_cycle(graph, cycle):
    """
    检查给定的顶点排列是否构成哈密顿环
    :param graph: 待检查的图
    :param cycle: 顶点的排列
    :return: 如果是哈密顿环返回 True，否则返回 False
    """
    if len(cycle) != len(graph.nodes):
        return False

    for i in range(len(cycle)):
        if not graph.has_edge(cycle[i], cycle[(i + 1) % len(cycle)]):
            return False

    return True

def find_hamiltonian_cycle(graph):
    """
    尝试找到图中的哈密顿环
    :param graph: 待检查的图
    :return: 如果找到哈密顿环，返回环中的节点顺序，否则返回 None
    """
    nodes = list(graph.nodes)
    for perm in permutations(nodes):
        if is_hamiltonian_cycle(graph, perm):
            return perm
    return None

def verify_hamiltonian_theorem(graph, cycle):
    """
    根据定理验证一个平面图是否是哈密顿图
    :param graph: 待检查的图
    :param cycle: 哈密顿环
    :return: True 如果满足定理条件, 否则 False
    """
    if not cycle:
        return False

    n = len(graph.nodes)
    f_prime = [0] * (n + 1)
    f_double_prime = [0] * (n + 1)

    # 获取所有的面
    faces = nx.cycle_basis(graph)
    internal_faces = []
    external_faces = []
```

```
    # 根据哈密顿环将面分为内部和外部
    for face in faces:
        if all(node in cycle for node in face):
            internal_faces.append(face)
        else:
            external_faces.append(face)

    for face in internal_faces:
        f_prime[len(face)] += 1

    for face in external_faces:
        f_double_prime[len(face)] += 1

    # 计算内部和外部的定理值
    sum_internal = sum(i * f_prime[i] for i in range(3, n + 1))
    sum_external = sum(i * f_double_prime[i] for i in range(3, n + 1))

    return sum_internal == n - 2 and sum_external == n - 2

def main():
    num_nodes = int(input("请输入平面图的节点数: "))
    G, pos = generate_random_planar_graph(num_nodes)

    # 找到哈密顿环
    cycle = find_hamiltonian_cycle(G)

    if cycle:
        print("找到哈密顿环:", cycle)
        # 验证定理
        if verify_hamiltonian_theorem(G, cycle):
            print("根据定理，该图是哈密顿图")
        else:
            print("根据定理，该图不是哈密顿图")
    else:
        print("没有找到哈密顿环，该图不是哈密顿图")

    # 绘制图
    nx.draw(G, pos, with_labels=True)
    plt.show()

if __name__ == "__main__":
    main()
```

4.邦迪定理和邦迪-萨瓦达定理

Bondy's Theorem (1972)

设 G 为 n 阶简单无向图，且 $n \geq 3$，则图 G 是 Hamilton 图当且仅当图 G 的闭包 $Cl(G)$ 是 Hamilton 图.

闭包: 图 G 的闭包 $Cl(G)$ 是指由下述方法得到的一个图，即反复连接 G 中度数之和 $\geq n$ 的不相邻的两顶点，直到没有这样的顶点对存在为止.

Proof:

引理: 图 G 的闭包 $Cl(G)$ 是唯一确定的.

证明: 假设 G 有两个闭包 G_1 和 G_2，设 $\{e_i\}_{i=1}^m$ 是经由上述操作所添加的边，$\{f_i\}_{i=1}^k$ 是 G_2 所添加的边。令 $H = G + \{e_i\}_{i=1}^m$，不妨设 $e_{k+1} = uv$ 是 $\{e_i\}$ 中第一条不属于 G_2 的边，则 $H \subseteq G_2$，故

$$d_{G_2}(u) + d_{G_2}(v) \geq n$$

由 G_1 的定义可得

$$d_H(u) + d_H(v) \geq n$$

矛盾!

因此每条 e_i 都是 G_2 的边，同理可得每条 f_i 都是 G_1 的边，故 $G_1 = G_2$，即 G 的闭包 $Cl(G)$ 是唯一确定的.

下面证明 G 是 Hamilton 图。在构作闭包的过程中，每添加一条边都用一次 Ore 定理即可。证毕。

推论: 若 $Cl(G) \cong K_n$（$n \geq 3$），则 G 是 Hamilton 图。

显然 K_n 是 Hamilton 图，它的最外圈 C_n 即为其 Hamilton 回路，故由 Bondy 定理得证。

Bondy-Chvátal Theorem (1976)

设 G 为 n 阶简单无向图，且 $n \geq 3$。设 G 的度序列为 $(d_i)_{i=1}^n, n \geq 3$，且 $d_1 \leq d_2 \leq \cdots \leq d_n$。若不存在 $m \leq l\frac{n}{2}$¿¿，使得 $d_m \leq m \wedge d_{n-m} \leq n-m$，则 G 是 Hamilton 图。

Proof:

我们先来证明 $Cl(G) \cong K_n$。

假设 $Cl(G) \ncong K_n$，设 u, v 是 $Cl(G)$ 中两个不相邻的顶点，有

$$d_{Cl}(u) \leq d_{Cl}(v)$$

且 $d_{Cl}(u) + d_{Cl}(v)$ 充分大，并且有

$$\forall u, v \in Cl(G) \wedge uv \notin Cl(G) \text{s.t.} \, d_{Cl}(u) + d_{Cl}(v) < v$$

记

$$S=\{v_i \mid v_i \in Cl(G) \wedge v_i v \notin Cl(G)\}$$
$$T=\{v_i \mid v_i \in Cl(G) \wedge v_i u \notin Cl(G)\}$$

则有

$$|S|=n-1-d_{Cl}(v)$$
$$|T|=n-1-d_{Cl}(u)$$

由于 $\Delta(S)\le d_{Cl}(u),\Delta(T\cup\{u\})\le d_{Cl}(v)$，令 $d_{Cl}(u)=m$，则 $V(Cl(G))\ge m,\Delta(Cl(G))\le m$ 且 $|\{v_i \mid d_{Cl}(v_i)\le n-m\}|\ge n-m$。因为 G 是 $Cl(G)$ 的生成子图，所以上述结论对 G 也成立，即 $d_m\le m,d_{n-m}\le n-m$，又由 $(1)(2)$ 可知 $m<\frac{n}{2}$，这与 $Cl(G)\not\cong K_n$ 矛盾！

故 $Cl(G)\cong K_n$。再由 Bondy 定理的推论可得 G 是 Hamilton 图。

5.posa 定理

Pósa’s Theorem (1962)

设 G 为 n 阶简单无向图，且 $n\ge 3$。定义 $V_k:=\{v\in V: deg(v)\le k\}$，若对于每一个 $k\in Z^{+¿¿}$ 满足 $1\le k<\frac{n-1}{2}$，$|V_k|<k$，且当 n 为奇数时满足 $k=\frac{n-1}{2}$，$|V_k|\le k$ 则 G 是 Hamilton 图。

证明

我们首先来证明 $Cl(G)\cong K_n$。假设 $Cl(G)\not\cong K_n$，取 $Cl(G)$ 中不相邻的两顶点 u,v 使得 $d_{Cl}(u)+d_{Cl}(v)$ 最大，则有 $d_{Cl}(u)+d_{Cl}(v)\le n-1$。我们不妨设 $d_{Cl}(u)\le d_{Cl}(v)$，设 $d_{Cl}(u)=k\le\frac{n-1}{2}$，则有 $d_{Cl}(v)\le n-k-1$。设 $S=\{x\in Cl(G): xv\notin Cl(G), x\ne v\}$，所以 $u\in S$。若 $w\in S$，则 $d_{Cl}(x)\le k$，否则

$$d_{Cl}(w)+d_{Cl}(v)>d_{Cl}(u)+d_{Cl}(v)$$

这与 u,v 的定义矛盾。因此 $\Delta(S)\le k,|S|\le k-1$，则

$$d_{Cl}(v)\ge(n-1)-(k-1)=n-k$$

这就产生了矛盾。故 $Cl(G)\cong K_n$。由 Bondy 定理可知 G 显然为 Hamilton 图。