云南大学数学与统计学院
《算法图论实验》上机实践报告

课程名称：算法图论实验	**年级：**2015 级	**上机实践成绩：**
指导教师：李建平	**姓名：**刘鹏	**专业：**信息与计算科学
上机实践名称：欧拉图判断与寻找欧拉回路	**学号：**20151910042	**上机实践日期：**2018-12-31
上机实践编号：5	**组号：**	

一、 实验目的

1. 了解欧拉图的来历、定义以及图论表述；
2. 能快速写出求最短路的最小插点问题的动态规划算法。

1.1 实验内容

1. 写出判定一个给定的图是否是欧拉图的算法；
2. 写出寻找含有欧拉回路的图的欧拉回路的算法。

二、 实验平台

Windows 10 Pro 1803；
MacOS Mojave。

三、 算法设计

3.1 图的一般概念

如果一个图不含有环（loop）和多重边（multiple edge），则称这种图为简单图（simple graph）。但是如果图中允许有环和多重边，那么这种图被称为伪图（pseudograph）。

给定伪图G，若存在一条简单链过图的每条边一次并且仅仅一次，则称这个链为欧拉链（Eulerian chain）。特别地，如果欧拉链变成了一个圈，那么称这个圈为欧拉圈（Eulerian cycle）。如果一个图有欧拉圈，则这个图是欧拉图（Eulerian graph）。如果连通伪图G是欧拉图，当且仅当G中不含奇点。

Fleury 提供了一个有效的在含欧拉圈的图中找出欧拉圈的算法[1]。

3.2 Fleury 算法

在本算法的叙述中，以$P_k=\left(v_{i_0},\ e_{i_1},\ v_{i_1},\ \cdots,\ e_{i_k},\ v_{i_k}\right)$表示在第$k$步得到的简单链。在这个算法中使用了连通性判断算法IS-CONNECTED(G)

Algorithm 求含欧拉圈图的欧拉圈，记此算法为FLEURY
Input $G=(V,\ E)$，允许这个图为任意的图

Output 图G的一个欧拉圈C，记$C=\text{FLEURY}(G)$
如果不含欧拉圈，就输出“This graph is not Eulerian graph”
Begin

Step 1 // 选择初始点

$\forall v_{i_0} \in V$

$P_0 = (v_{i_0})$

$G_0 = G\big(E - E(P_0)\big)$

Step 2 $k = 0$

while true:

  $\text{flag} = 0$

  for each edge $e \in \boldsymbol{\Phi}_{G_k}(v_{i_k})$:

    if IS-CONNECTED $\big((G(E - \{e\}))\big) = \textbf{True}$:

      $e_{i_{k+1}} = e$

      $v_{i_{k+1}} = e.\text{OTHER\_VERTEX}(v_{i_k})$

      $P_{k+1} = \big(v_{i_0},\ e_{i_1},\ v_{i_1},\ \cdots,\ e_{i_k},\ v_{i_k},\ e_{i_{k+1}},\ v_{i_{k+1}}\big)$

      $G_{k+1} = G\big(E - E(P_{k+1})\big)$

      $k = k + 1$

      $\text{flag} = 1$

      break

  if $\text{flag} = 0$:

    if $k = |G|$:

      $C = P_k$

    else:

      output This graph is not Eulerian graph

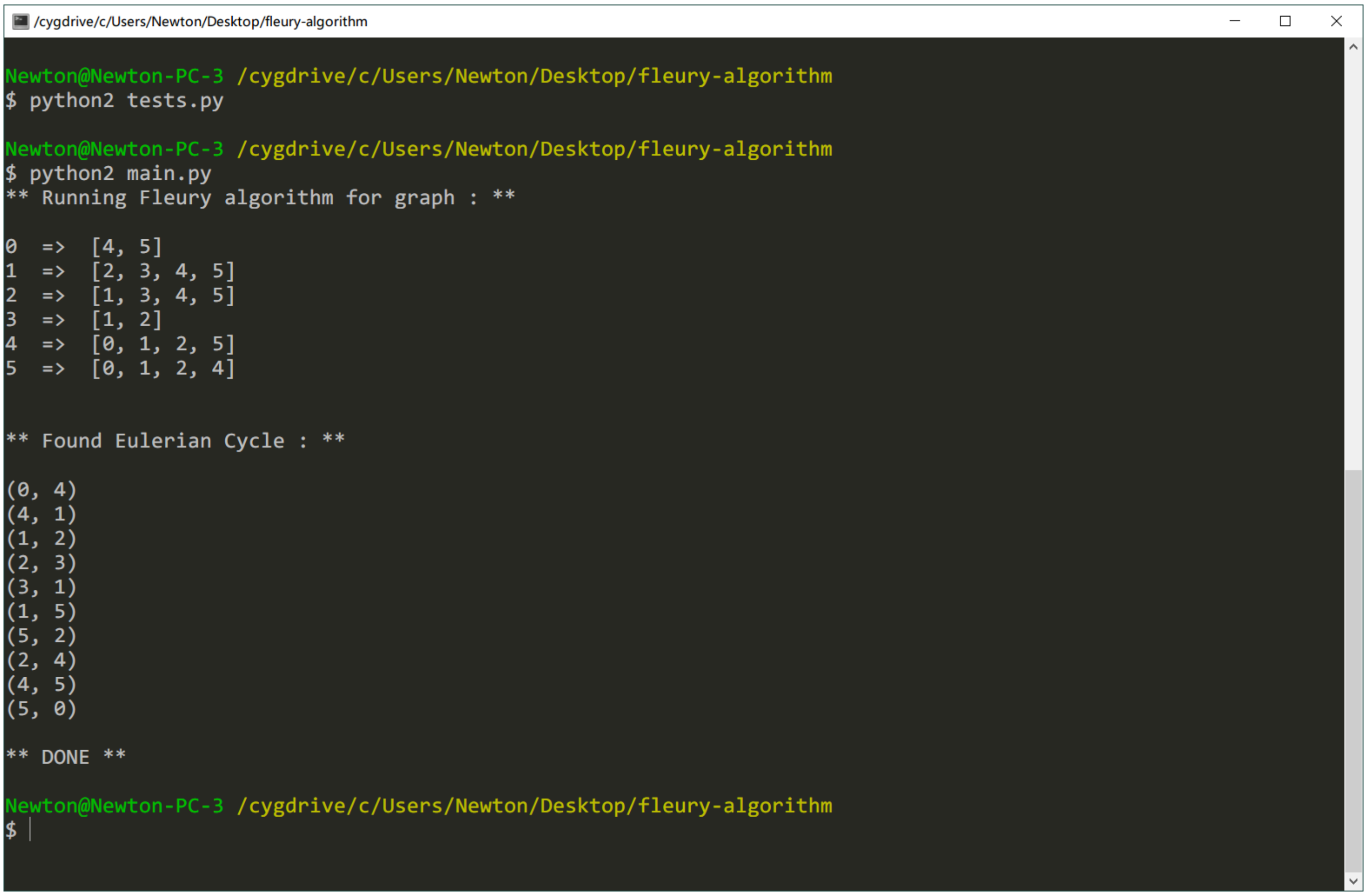
End

通过这个算法，可以比较简单地判定一个图是否是欧拉图，如果是欧拉图，还可以输出欧拉回路。

四、 程序代码

4.1 运行结果

以如图所示为例子，可以看到输出的结果。



4.2 程序代码

使用 Python 2 进行编码。具体的程序非常复杂，这里仅列举核心的FLEURY算法部分，具体代码参见相应的文件夹。

```python
# Fleury's Algorithm implementation


import copy

class FleuryException(Exception):
    def __init__(self, message):
        super(FleuryException, self).__init__(message)
        self.message = message

class Fleury:

    COLOR_WHITE = 'white'
    COLOR_GRAY  = 'gray'
    COLOR_BLACK = 'black'

    def __init__(self, graph):

        self.graph = graph

```

```
    def run(self):

        print "** Running Fleury algorithm for graph : ** \n"
        for v in self.graph:
            print v, ' => ', self.graph[v]
        print '\n'
        output = None
        try:
            output = self.fleury(self.graph)
        except FleuryException as (message):
            print message

        if output:
            print '** Found Eulerian Cycle : **\n'
            for v in output:
                print v
        print '\n** DONE **'

    def is_connected(self, G):

        start_node = list(G)[0]
        color = {}
        iterator = 0;
        for v in G:
            color[v] = Fleury.COLOR_WHITE
        color[start_node] = Fleury.COLOR_GRAY
        S = [start_node]
        while len(S) != 0:
            u = S.pop()
            for v in G[u]:
                if color[v] == Fleury.COLOR_WHITE:
                    color[v] = Fleury.COLOR_GRAY
                    S.append(v)
                color[u] = Fleury.COLOR_BLACK
        return list(color.values()).count(Fleury.COLOR_BLACK) == len(G)

    def even_degree_nodes(self, G):

        even_degree_nodes = []
        for u in G:
            if len(G[u]) % 2 == 0:
                even_degree_nodes.append(u)
        return even_degree_nodes


    def is_eulerian(self, even_degree_odes, graph_len):

        return graph_len - len(even_degree_odes) == 0

```

```python

    def convert_graph(self, G):

        links = []
        for u in G:
            for v in G[u]:
                links.append((u, v))
        return links


    def fleury(self, G):

        edn = self.even_degree_nodes(G)

        if not self.is_eulerian(edn, len(G)):
            raise FleuryException('This is not an Eulerian graph!')
        g = copy.copy(G)
        cycle = []

        u = edn[0]
        while len(self.convert_graph(g)) > 0:
            current_vertex = u
            for u in list(g[current_vertex]):
                g[current_vertex].remove(u)
                g[u].remove(current_vertex)

                bridge = not self.is_connected(g)
                if bridge:

                    g[current_vertex].append(u)
                    g[u].append(current_vertex)
                else:
                    break
            if bridge:

                g[current_vertex].remove(u)
                g[u].remove(current_vertex)
                g.pop(current_vertex)
            cycle.append((current_vertex, u))
        return cycle
```

五、 参考文献

[1] 田丰，张运清. 图与网络流理论 [M]. 2nd ed. 北京：科学出版社, 2015.

[2] https://github.com/dkulig/fleury-algorithm