西安交通大學

XI'AN JIAOTONG UNIVERSITY

复杂网络动力学基础 #1

班级：计试 91
姓名：施劲松
学号：2193512032

10/02/2022

1 问题 1

1.1 问题描述

在图 1.1 所示的无向网络中，共有 6 个节点 7 条边，$N = 6$，$M = 7$，试利用程序求解 6 个节点的介数 $B_i(i = 1 \dots 6)$ 和 7 条边的介数 $B_{ij}(e_1 \dots e_7)$，并要求手算 B_1 和 B_{15} 。

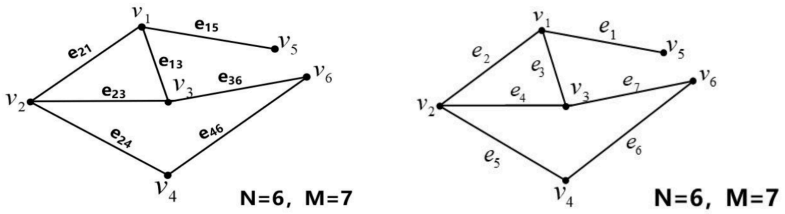

图 1.1.1: 问题 1 无向网络

1.2 实验原理

节点介数 B_i 表征最短路径途径节点 v_i 的程度。其定义如下：

$$B_i = \sum_{1 \le j < l \le N \wedge j \ne i \ne l} \frac{n_{jl}(i)}{n_{jl}}$$

其中 n_{jl} 为节点 v_j 和 v_l 之间的最短路径条数，$n_{jl}(i)$ 为节点 v_j 和 v_l 之间的最短路径经过节点 v_i 的条数。

同理，边介数 $\tilde{B}_{ij}$ 表征最短路径途径边 (v_i, v_j) 的程度。其定义如下：

$$\tilde{B}_{ij} = \sum_{1 \le l < m \le N \wedge (l,m) \ne (i,j)} \frac{N_{lm}(e_{ij})}{N_{lm}}$$

其中，N_{lm} 为节点 v_l 与 v_m 之间的最短路径条数；$N_{lm}(e_{ij})$ 为节点 v_l 和 v_m 之间的最短路径且经过边 e_{ij} 的条数。

所以只要求出网络中任意两点之间的最短路径以及对应的最短路径的条数，接着遍历每个点、每条边，就能计算各点及各边的介数。而这用 Floyd 算法就能解决。用 $dp[i][j]$ 表示点 v_i 至 v_j 的最短距离，在转移时多维护一个 $dp\_cnt[i][j]$ 表示最短路径条数即可，时间复杂度：$O(N^3)$。

1.3 求解过程

注意到状态转移：

$$dp[i][j] = \begin{cases} \min_{k=0,1,2,\dots,n}\{dp[i][k] + dp[k][j]\}, i \ne j \\ 0, i = j \end{cases}$$

以及

$$dp\_cnt[i][j] = \begin{cases} \sum_{dp[i][k]+dp[k][j]=dp[i][j] \wedge k \neq j} dp\_cnt[i][k] \times dp\_cnt[k][j], i \neq j \\ 1, i = j \end{cases}$$

故可简单实现如下 (C++):

```
for (int k = 0; k < n; ++k) {
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            if (i == k || j == k) continue;
            if (dp[i][k] + dp[k][j] < dp[i][j]) {
                dp[i][j] = dp[i][k] + dp[k][j];
                dp_cnt[i][j] = dp_cnt[i][k] * dp_cnt[k][j];
            } else if (dp[i][k] + dp[k][j] == dp[i][j]) {
                dp_cnt[i][j] += dp_cnt[i][k] * dp_cnt[k][j];
            }
        }
    }
}
```

手算求解 B_1 以及 $\tilde{B}_{15}$ 的过程如下:

解. 由无向网络知:

$$\begin{aligned} n_{25} &= N_{25} = 1 \\ n_{35} &= N_{35} = 1 \\ n_{45} &= N_{45} = 3 \\ n_{56} &= N_{56} = 1 \end{aligned}$$

故:

$$\begin{aligned} B_1 &= \sum_{i<j} \frac{n_{ij}(1)}{n_{ij}} \\ &= \frac{n_{25}(1)}{n_{25}} + \frac{n_{35}(1)}{n_{35}} + \frac{n_{45}(1)}{n_{45}} + \frac{n_{56}(1)}{n_{56}} \\ &= 1 + 1 + 1 + 1 \\ &= 4 \end{aligned}$$

$$\begin{aligned} \tilde{B}_{15} &= \sum_{i<j} \frac{N_{ij}(1)}{N_{ij}} \\ &= \frac{N_{25}(1)}{N_{25}} + \frac{N_{35}(1)}{N_{35}} + \frac{N_{45}(1)}{N_{45}} + \frac{N_{56}(1)}{N_{56}} \\ &= 1 + 1 + 1 + 1 \\ &= 4 \end{aligned}$$

1.4 实验数据

在 bash/zsh 中输入：

```
$ g++ task_1.cpp -o task_1
$ ./task_1
```

再输入该网络：

```
6 7 # 点数、边数

1 5
1 2
1 3
2 3
2 4
4 6
3 6
```

得到：

```
node betweenness:
node#1: 4.00
node#2: 2.50
node#3: 2.50
node#4: 0.50
node#5: 0.00
node#6: 0.50
edge betweenness:
edge#1: 4.00
edge#2: 3.00
edge#3: 3.00
edge#4: 1.00
edge#5: 3.00
edge#6: 1.00
edge#7: 3.00
```

也即，点 $v_1, v_2, \ldots, v_6$ 的介数分别为：$4, 2.5, 2.5, 0.5, 0, 0.5$，而边（编号依图 1.1）$e_1, e_2, \ldots, e_7$ 的介数分别为：$4, 3, 3, 1, 3, 1, 3$。

1.5 实验结果与分析

实验结果见表 1.5 和表 1.5。

结点编号	1	2	3	4	5	6
B_i	4	2.5	2.5	0.5	0	0.5

表 1.5.1: 问题 1 点介数表

由上述结果可以得出一些直观性结论：高介数节点通常扮演“枢纽”，往往是图的拓扑中心且会影响很多节点的连接性；割点往往是高介数节点。高介数边通常扮演“必经之路”，一旦去掉会影响很多节点的连接性；桥往往是高介数边；而低介数边一旦去掉，影响连通性的概率较小，所以点（边）介数的大小表现了其在网络中的重要性。

边编号	1	2	3	4	5	6	7
B_i	4	3	3	1	3	1	3

表 1.5.2: 问题 1 边介数表

2 问题 2

2.1 问题描述

在互联网的网页链接和搜索引擎中，互联网可以看成一个有向图 2.1（无权有向网络），每一个网页是图的一个节点（顶点），网页间的每一次超链接是图的一个边，Google 公司使用了 PageRank 算法作为搜索引擎的核心算法。下图描述了一个互联网搜索引擎的网页链接关系图，其中有 7 个节点（网页）和 18 条边（超链接），$N = 7, M = 18$。

基于复杂网络特征向量中心性、Pagerank 算法和书本例 3.6 的学习，试求:

1. 网络的邻接矩阵 B、特征向量中心性中最大特征值 λ 和所对应的特征向量以及归一化后的特征向量，以及该特征向量中心性的分值;
2. 网络的 Markov 链的状态转移概率矩阵 P 及其转置矩阵 P^{T}，最大特征值 $\lambda_{\max}$，Markov 链的平稳分布，PageRank 值及其直方图，模型参数 d 可选 0.85。

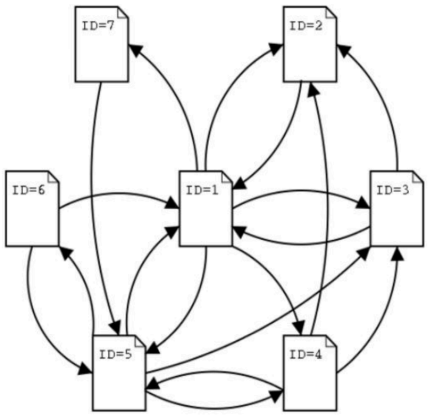

图 2.1.1: 问题 2 有向网络

2.2 实验原理

矩阵 B 由网络结构决定:

$$\mathrm{B}_{ij} = \begin{cases} 1, (v_i \to v_j) & \text{exists;} \\ 0, (v_i \to v_j) & \text{does not exist.} \end{cases}$$

故由图 2.1 即得：

$$\mathrm{B}=\begin{pmatrix}0&1&1&1&1&0&1\\1&0&0&0&0&0&0\\1&1&0&0&0&0&0\\0&1&1&0&1&0&0\\1&0&1&1&0&1&0\\1&0&0&0&1&0&0\\0&0&0&0&1&0&0\end{pmatrix}$$

根据定义，满足 $\lambda\vec{x}=\mathrm{B}\vec{x}$ 的最大特征值 $\lambda_{\max}$ 即为 B 的特征向量中心性的最大特征值，其对应的归一化后的特征向量的分量即为中心性的分值。而求解特征值即求

$$|\mathrm{B}-\lambda\mathrm{I}|=0$$

的代数方程。由特征值再求特征向量即求齐次线性方程解。由于高次代数方程极难求解，问题 2 用 `python` 来完成。

在互联网的 PageRank 算法中，设网页的浏览行为是 Markov 过程，定义 Markov 链的状态转移概率矩阵为 P。则有：

$$\mathrm{P}_{ij}=\frac{1-d}{N}+d\frac{\mathrm{B}_{ij}}{\sum_l \mathrm{B}_{il}}$$

而该 Markov 链的平稳分布 x^{T} 则满足：

$$\mathrm{P}^{\mathrm{T}}x=x$$

即 P^{T} 的最大特征值为 1。故要求 2 与要求 1 的求解过程是相仿的，实现时可写成同一函数。

2.3 求解过程

由上一节的分析，求解的关键在于如何写一个可求矩阵最大特征值以及对应归一化后的特征向量的函数。借由 `python` 中的 `numpy` 库，这则可以轻松实现：

```
1  import numpy as np
2  # 函数，接收一个矩阵 A，返回最大特征值以及归一化后的特征向量
3  def get_eigen(A):
4      # 获取特征值与特征向量
5      eigen_values, eigen_vectors = np.linalg.eig(A)
6
7      i = np.argmax(eigen_values) # 获得最大特征值的索引
8      max_eigen_value = eigen_values.max() # 获得最大特征值
9      max_eigen_vector = eigen_vectors.T[i] # 获得对应特征向量
10
11     # 最大特征值
12     max_eigen_value = np.real(max_eigen_value) # 实化
13
14     # 归一化后的特征向量
15     max_eigen_vector = np.real(max_eigen_vector) # 实化
16     max_eigen_vector = max_eigen_vector / max_eigen_vector.sum() # 归一化
17
```

```
    # 返回矩阵，最大特征值以及对应归一化后的特征向量
    return A, max_eigen_value, max_eigen_vector.T
```

2.4 实验数据

由于程序已将邻接矩阵输入，故只需在 bash/zsh 中输入：

```
$ python3 task_2.py
```

得到：

```
邻接矩阵：
[[0 1 1 1 1 0 1]
[1 0 0 0 0 0 0]
[1 1 0 0 0 0 0]
[0 1 1 0 1 0 0]
[1 0 1 1 0 1 0]
[1 0 0 0 1 0 0]
[0 0 0 0 1 0 0]]
最大特征值：2.83117721
归一化后的特征向量：
[[0.22038096]
[0.07784075]
[0.10533488]
[0.14254021]
[0.22038096]
[0.15568150]
[0.07784075]]

状态转移概率矩阵：
[[0.02142857 0.19142857 0.19142857 0.19142857 0.19142857 0.02142857 0.19142857]
[0.87142857 0.02142857 0.02142857 0.02142857 0.02142857 0.02142857 0.02142857]
[0.44642857 0.44642857 0.02142857 0.02142857 0.02142857 0.02142857 0.02142857]
[0.02142857 0.30476190 0.30476190 0.02142857 0.30476190 0.02142857 0.02142857]
[0.23392857 0.02142857 0.23392857 0.23392857 0.02142857 0.23392857 0.02142857]
[0.44642857 0.02142857 0.02142857 0.02142857 0.44642857 0.02142857 0.02142857]
[0.02142857 0.02142857 0.02142857 0.02142857 0.87142857 0.02142857 0.02142857]]
状态转移概率矩阵的转置：
[[0.02142857 0.87142857 0.44642857 0.02142857 0.23392857 0.44642857 0.02142857]
[0.19142857 0.02142857 0.44642857 0.30476190 0.02142857 0.02142857 0.02142857]
[0.19142857 0.02142857 0.02142857 0.30476190 0.23392857 0.02142857 0.02142857]
[0.19142857 0.02142857 0.02142857 0.02142857 0.23392857 0.02142857 0.02142857]
[0.19142857 0.02142857 0.02142857 0.30476190 0.02142857 0.44642857 0.87142857]
[0.02142857 0.02142857 0.02142857 0.02142857 0.23392857 0.02142857 0.02142857]
[0.19142857 0.02142857 0.02142857 0.02142857 0.02142857 0.02142857 0.02142857]]
最大特征值：1.00000000
平稳分布：
[0.28028780 0.15876449 0.13888182 0.10821960 0.18419813 0.06057067 0.06907750]
```

2.5 实验结果与分析

由程序运行结果可知：邻接矩阵 B 的最大特征值 $\lambda_{\max}$ 为 2.83117721，其对应的归一化后的特征向量为：

$$x^* = \begin{pmatrix} 0.22038096 \\ 0.07784075 \\ 0.10533488 \\ 0.14254021 \\ 0.22038096 \\ 0.15568150 \\ 0.07784075 \end{pmatrix}$$

每一维即为特征向量中心性的分值。此处由于是有向图的邻接矩阵，方程 $\mathrm{B}\vec{x} = \lambda\vec{x}$ 描述的是链出节点对该点中心性的影响，也就是说，键出的边越多，中心性越大，该点越重要。

而状态转移概率矩阵：

$$\mathrm{P} = \begin{pmatrix} 0.02142857 & 0.19142857 & 0.19142857 & 0.19142857 & 0.19142857 & 0.02142857 & 0.19142857 \\ 0.87142857 & 0.02142857 & 0.02142857 & 0.02142857 & 0.02142857 & 0.02142857 & 0.02142857 \\ 0.44642857 & 0.44642857 & 0.02142857 & 0.02142857 & 0.02142857 & 0.02142857 & 0.02142857 \\ 0.02142857 & 0.30476190 & 0.30476190 & 0.02142857 & 0.30476190 & 0.02142857 & 0.02142857 \\ 0.23392857 & 0.02142857 & 0.23392857 & 0.23392857 & 0.02142857 & 0.23392857 & 0.02142857 \\ 0.44642857 & 0.02142857 & 0.02142857 & 0.02142857 & 0.44642857 & 0.02142857 & 0.02142857 \\ 0.02142857 & 0.02142857 & 0.02142857 & 0.02142857 & 0.87142857 & 0.02142857 & 0.02142857 \end{pmatrix}$$

其转置的最大特征值为 $\lambda_{\max} = 1$，对应的归一化后的特征向量（平稳分布的转置）：

$$\vec{x}^{\mathrm{T}} = \begin{pmatrix} 0.28028780 \\ 0.15876449 \\ 0.13888182 \\ 0.10821960 \\ 0.18419813 \\ 0.06057067 \\ 0.06907750 \end{pmatrix}$$

PageRank 能很好地体现一个节的的链入能力，亦即高价值的节点。这与其入度的关系大致为正相关，但若仅考虑入度是较为片面的，因为这会夸大中间"索引"节结的能力。直方图见图 2.5，其重要性一目了然，可见 v_1 和 v_5 的 PageRank 值较高，故其重要程度也高，链入能力越强；反之 v_6 和 v_7 则较弱。

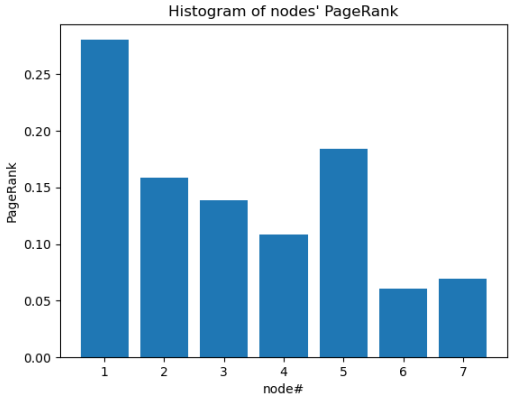

图 2.5.1: PageRank 直方图

3 感想与建议

复杂网络动力学的学习是网络科学研究中极为重要的一个环节，通过课上课下的学习我对其认知已有一个初步的轮廓，通过本次实验，我更进一步地了解了一些复杂网络中的重要概念与标度，对介数、特征向量中心性以及 PageRank 的作用都有了更深的理解。对今后的学习提供了莫大的帮助。

另外我觉得实验的数据可以加强，实验指导书可以只放一两个简单例子供同学校验。验收可用些强数据，方便检验是否是真正实现了正确的算法，例如用一些随机的种子生成数据是十分方便有效的。

4 源码

问题 1 见：`./src/task_1.cpp`。
问题 2 见：`./src/task_2.py`。

5 运行

对于问题 1，请确保安装了 GNU GCC，而后在 `bash/zsh` 输入：

```
$ g++ task_1.cpp -o task_1
$ ./task_1
```

编译运行，实验数据在源码最下方。若是 Windows 用户，在 `cmd` 中输入：

```
> g++ task_1.cpp -o task_1
> .\task_1.exe
```

对于问题 2，请确保安装了 `python3` 以及依赖包 `numpy`（可使用 conda 或 pip 安装），conda 安装命令并运行方法：

```
$ conda create -n lab1 numpy
$ conda activate lab1
$ python3 task_2.py
```

另外，源码采用 utf-8 编码，若开启编辑器中文乱码，请切换至 utf-8 编码重新打开。