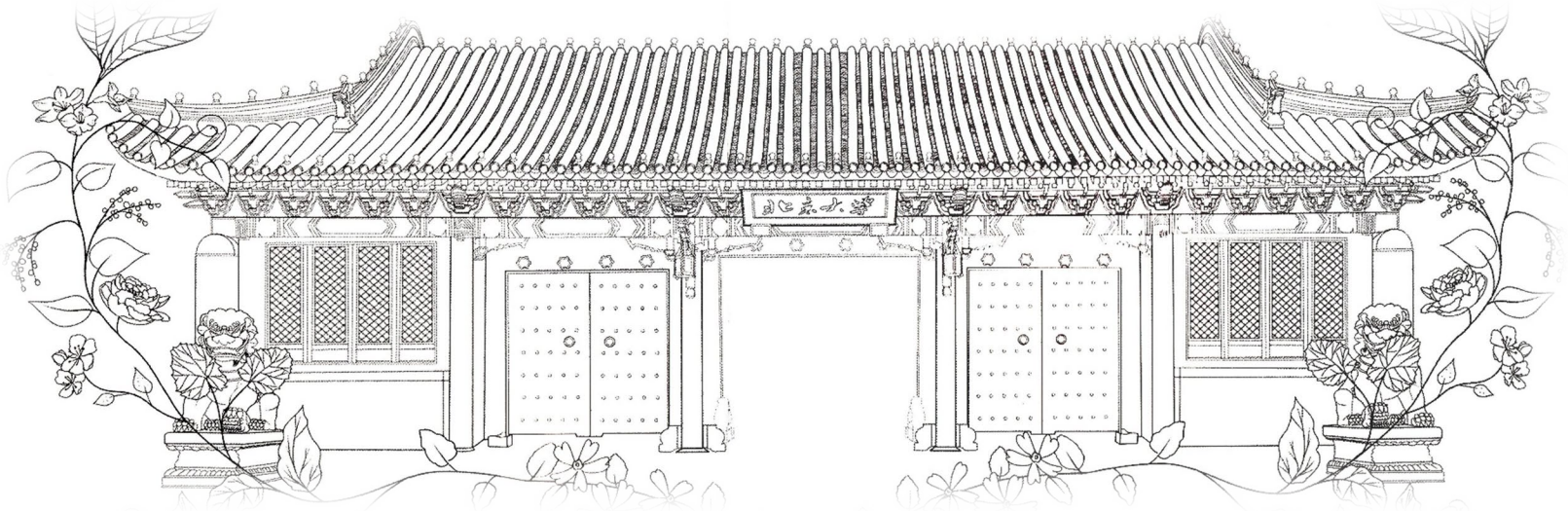
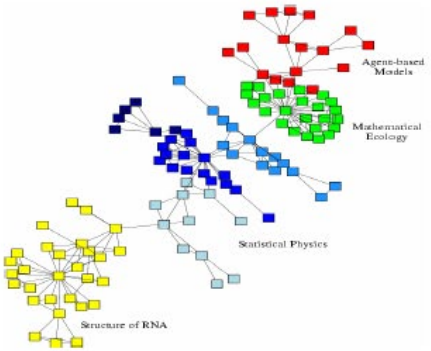
# 《Python数据分析》
# 应用篇：社会网络分析

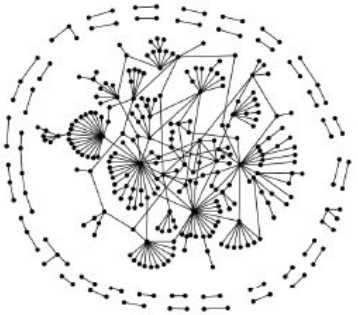


# 网络（network）

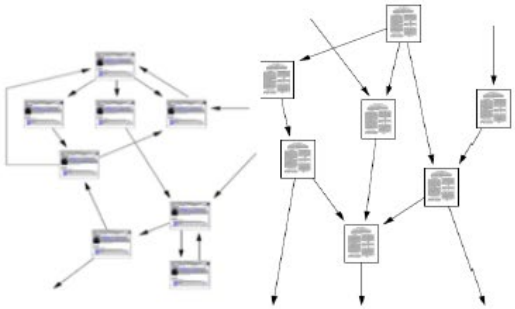
Social networks



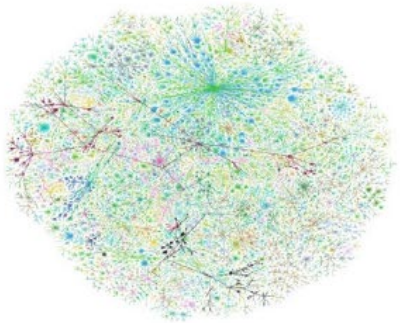

Economic networks



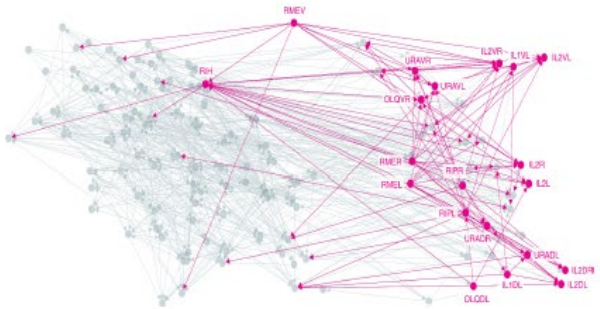
Biomedical networks



Information networks:
Web & citations



Internet



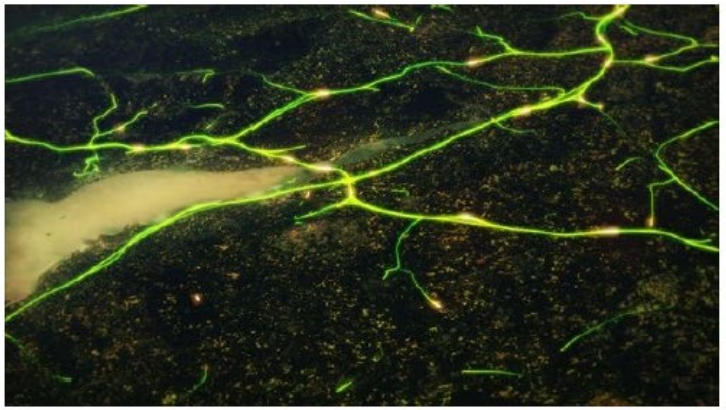

Networks of neurons

# 网络（network）

Data networks

Air traffic network

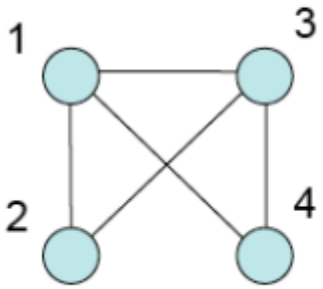
Telecommunications networks

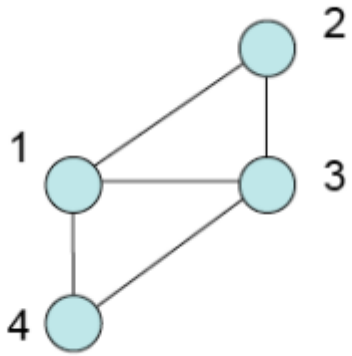


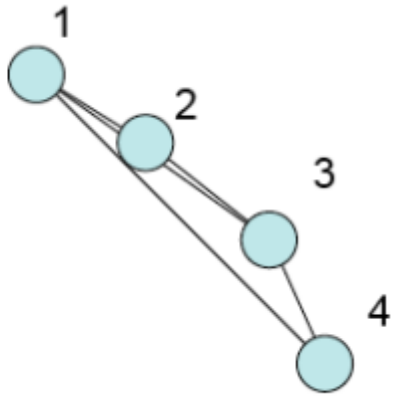
Shipping (sea) networks

# 网络的抽象化

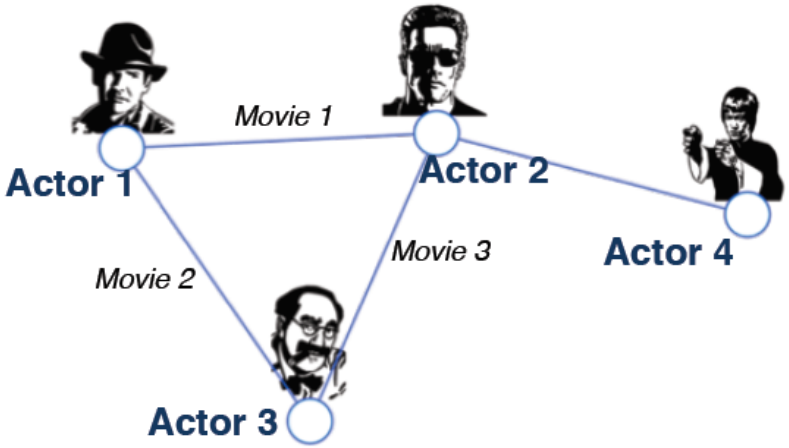
- 图G由一个顶点（或节点）集合V和一个边集合E组成： G={V,E}
- 每条边$e_{xy}$=(x, y)连接图G的两个顶点x, y
  - 例如： V={1,2,3,4}, E={(1,2),(1,3),(2,3),(3,4),(4,1)}

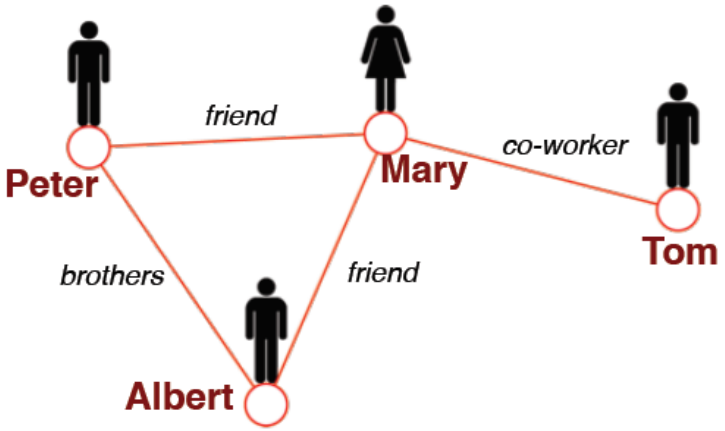


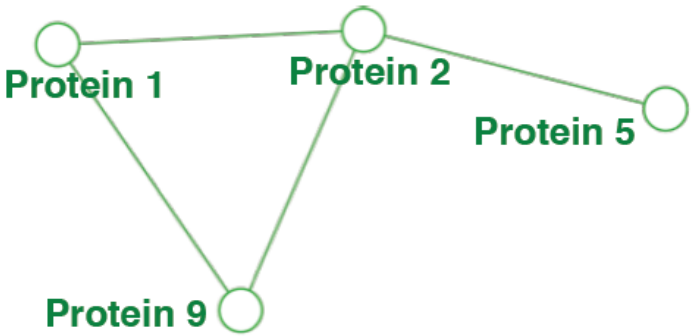



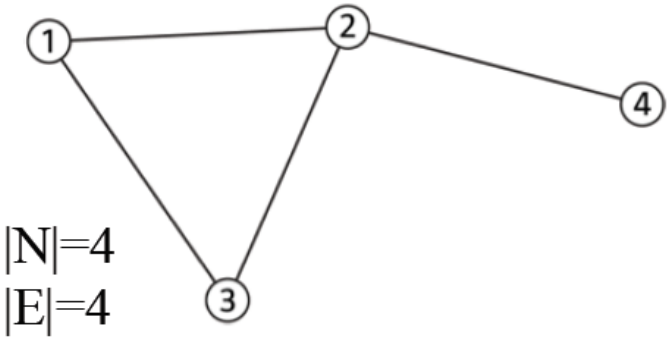



# 不同情景的数据都可化归为网络



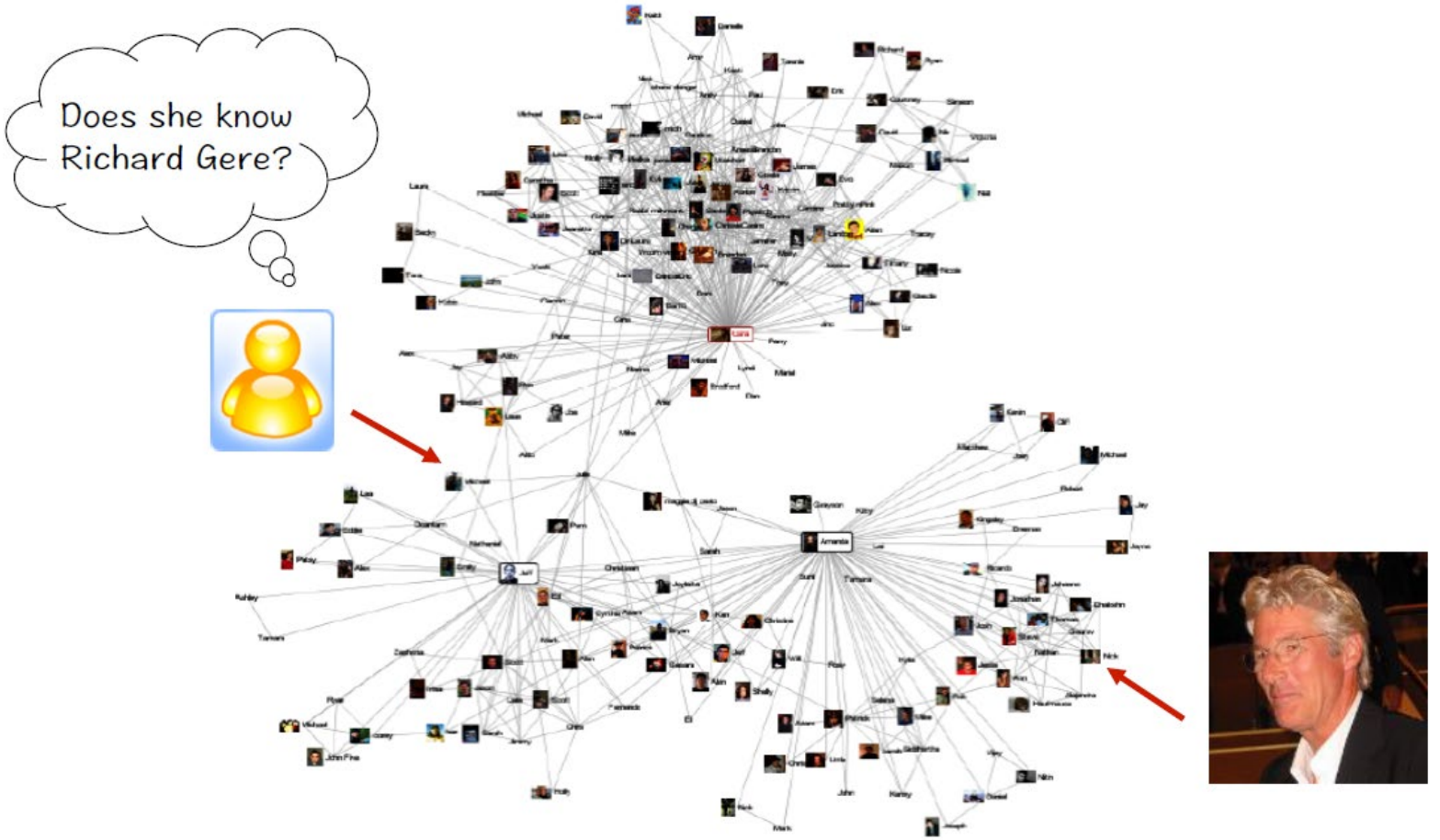



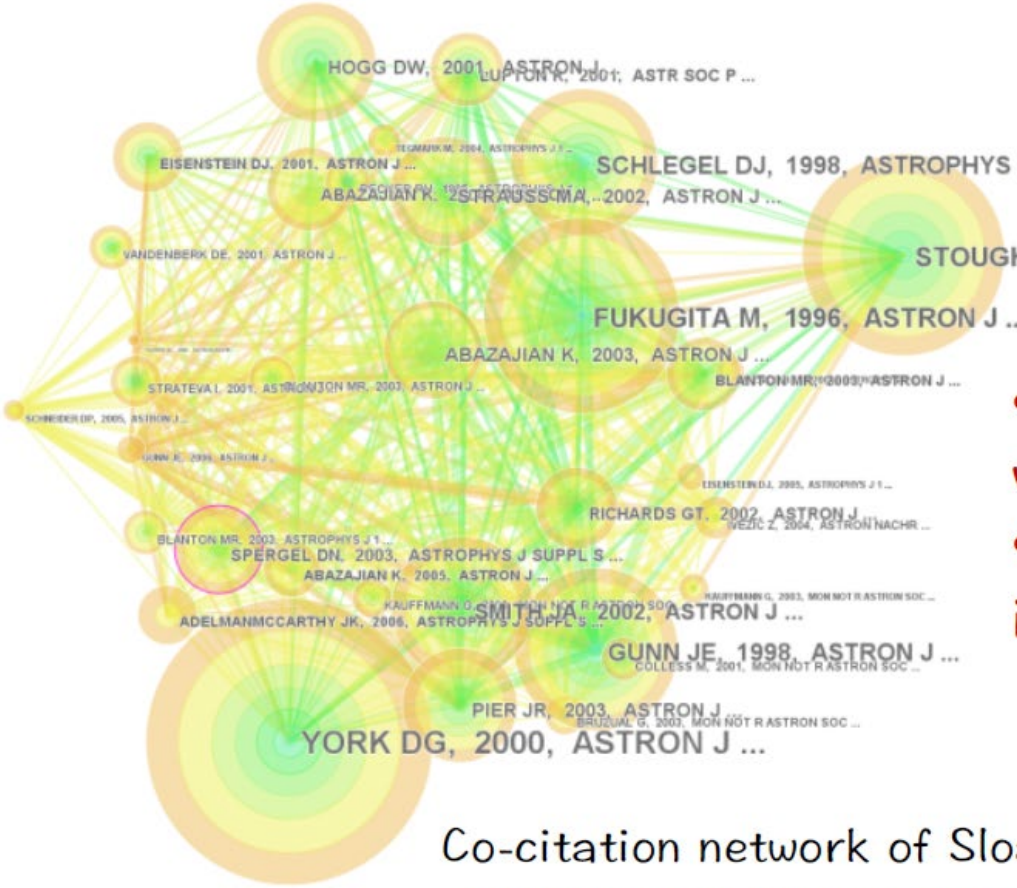
# 社会网络分析（Social network analysis，SNA）

- “社会网络”是作为节点的社会行动者（social actors）及其间关系的集合。
  - 一个“社会网络”是由多个点（社会行动者）和各点关系（行动者间的关系）组成的集合

# 社会网络分析的应用：链路预测 (Link prediction)



# 社会网络分析的应用：排名(Ranking)

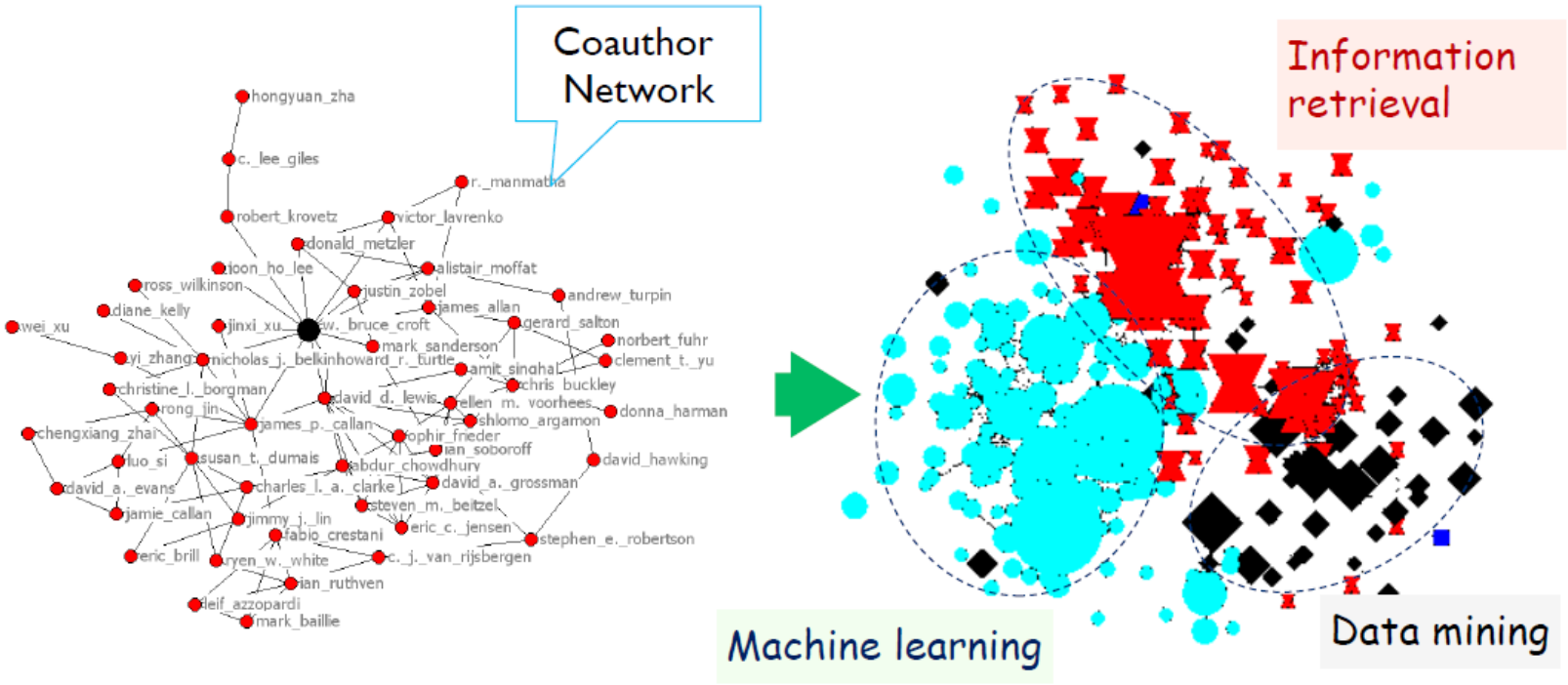



- Importance of vertices
- Which is the most influential paper?

Co-citation network of Sloan Digital Sky Survey

- http://nevac.ischool.drexel.edu/~james/infovis09/FP-tree-visual.html

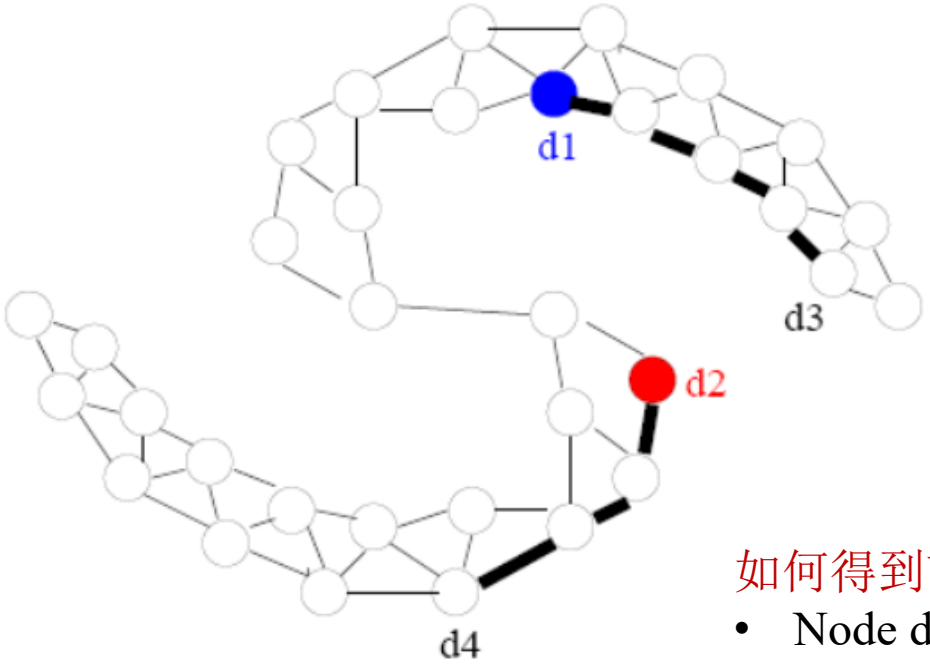
# 社会网络分析的应用：社区探测 (Community detection)




Who tend to work together?

- Q.Mei, D.Cai, D.Zhang, and C.Zhai, Topic Modeling with Hitting Time, WWW 2008

# 社会网络分析的应用：分类(Classification)/聚类 (Clustering)



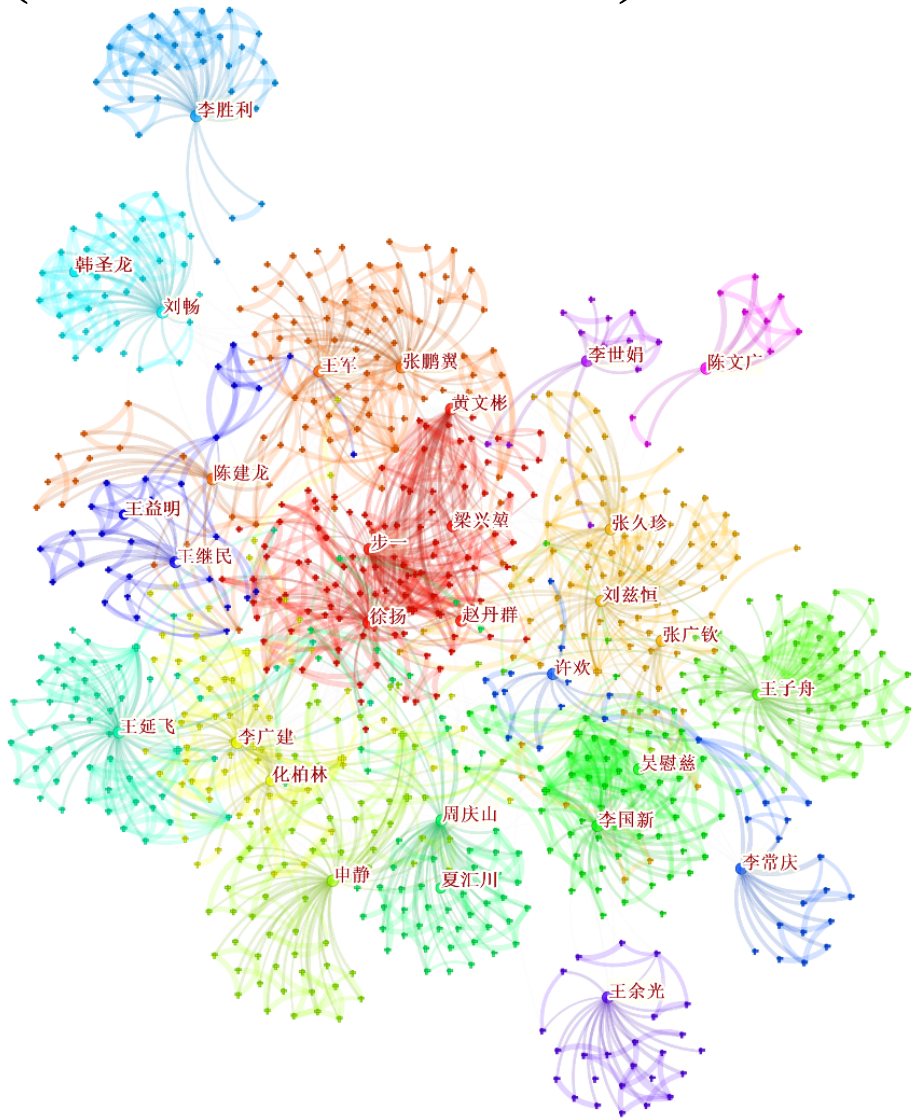

- d1 is democratic
- d2 is republican
- What can we say about d3 and d4?

如何得到节点的特征？

- Node degree, PageRank score, motifs, …
- Degree of neighbors, PageRank of neighbors, …

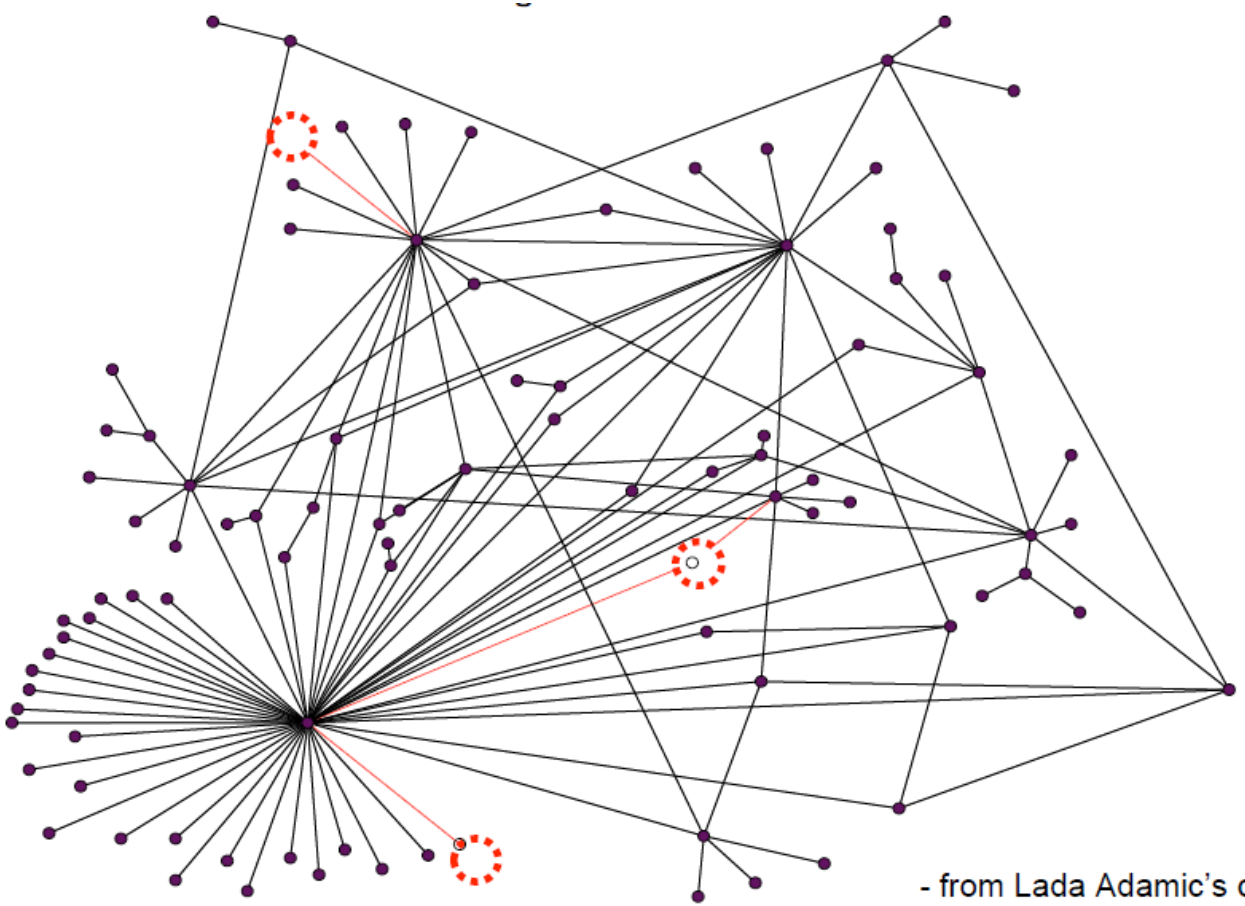
- Graph from Jerry Zhu's tutorial in ICML 07

# 社会网络分析的应用：分类(Classification)/聚类 (Clustering)




- 定价策略，商业模式，发布策略
- 信息检索行为，信息源使用，交互式信息检索
- 用户研究，信息搜寻行为，商品浏览，商品检索
- 数字阅读，数字阅读推广，图书馆建设
- 数据质量，滑动窗口模型，预测
- 文献计量，信息抽取，作者共引分析，共引网络
- 图书馆服务，图书馆资源，版权保护，开放存取
- 日志挖掘，机器学习，知识图谱
- 决策信息，情报学术，情报治理
- 信息分析，文本挖掘，情报方法
- 知识管理，知识服务，创新能力，创新评估
- 媒体规划，信息治理，信息公开，信息保护
- 公共文化服务，公共图书馆，公共文化服务体系
- 民间图书馆，图书馆功能，图书馆发展
- 图书，出版文化，出版业，阅读推广
- 文献学，图书馆学术史，图书馆学发展

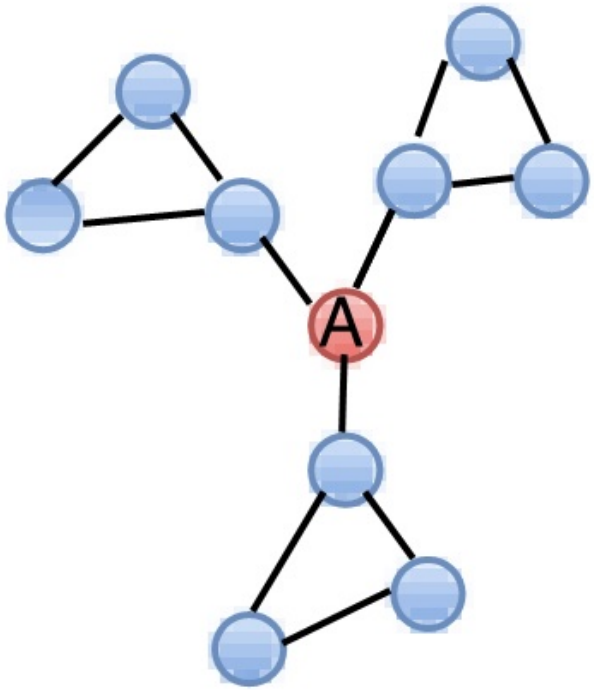
# 社会网络分析的应用：恢复能力(Resilience)

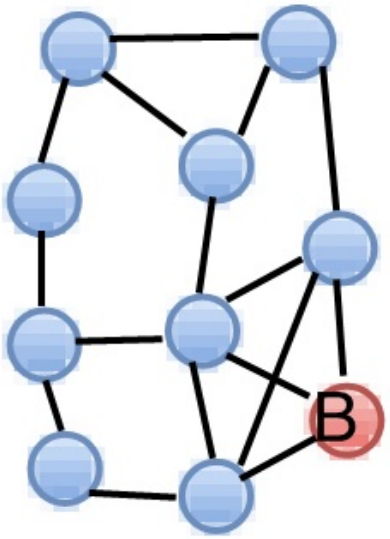


- from Lada Adamic's course

How robust are networks to random/targeted attacks?

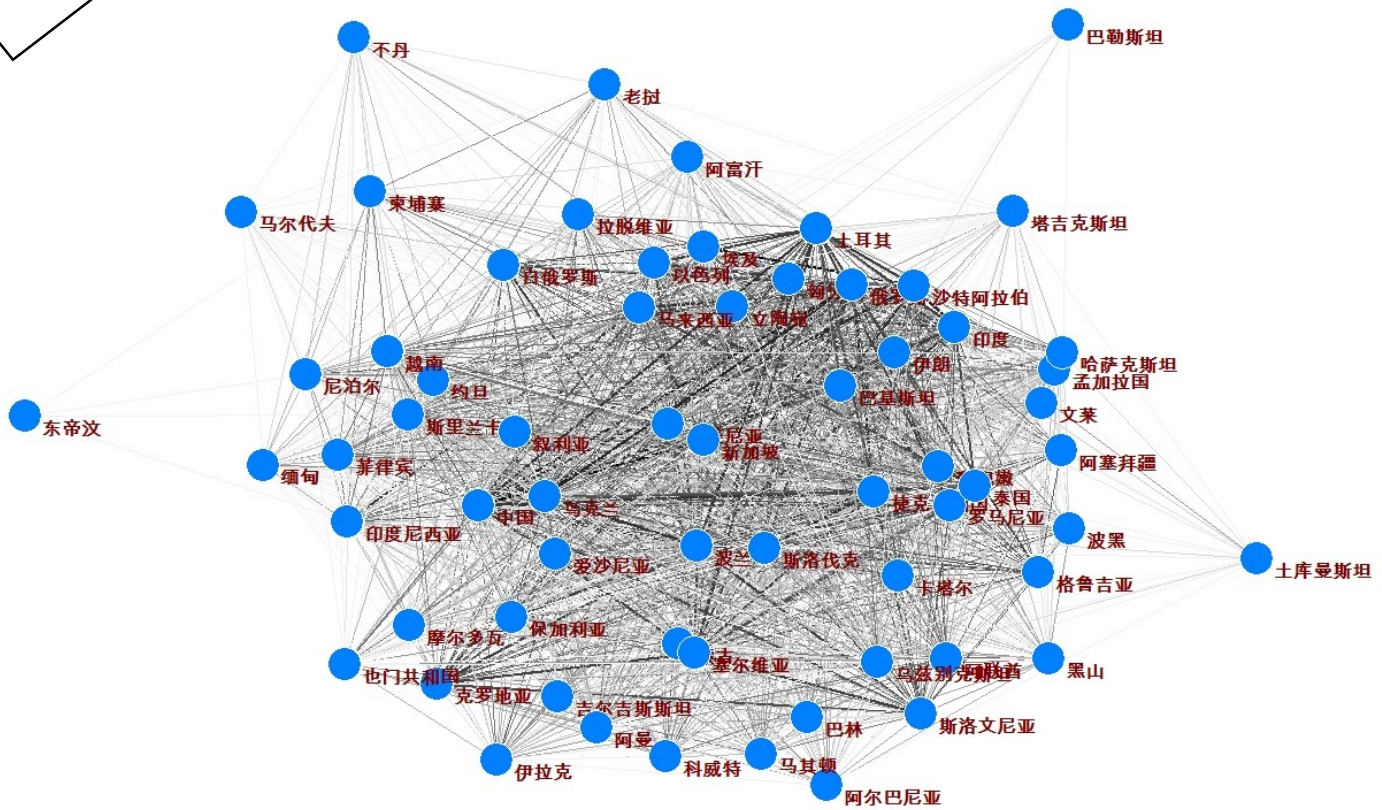
# 社会网络分析的应用：结构分析 (Structural analysis)



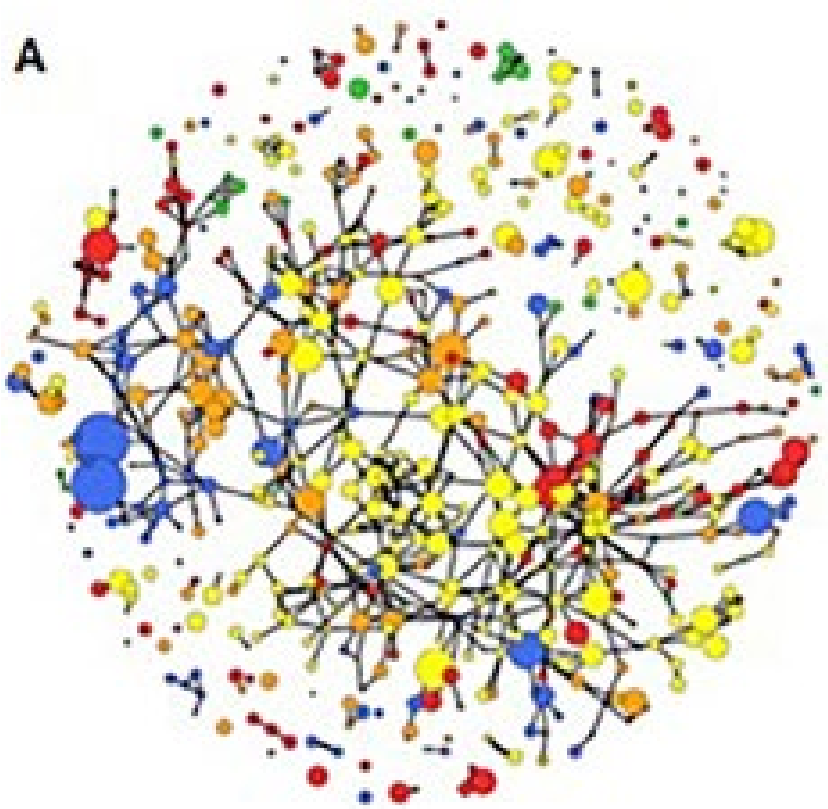
# 社会网络分析的应用：结构分析 (Structural analysis)

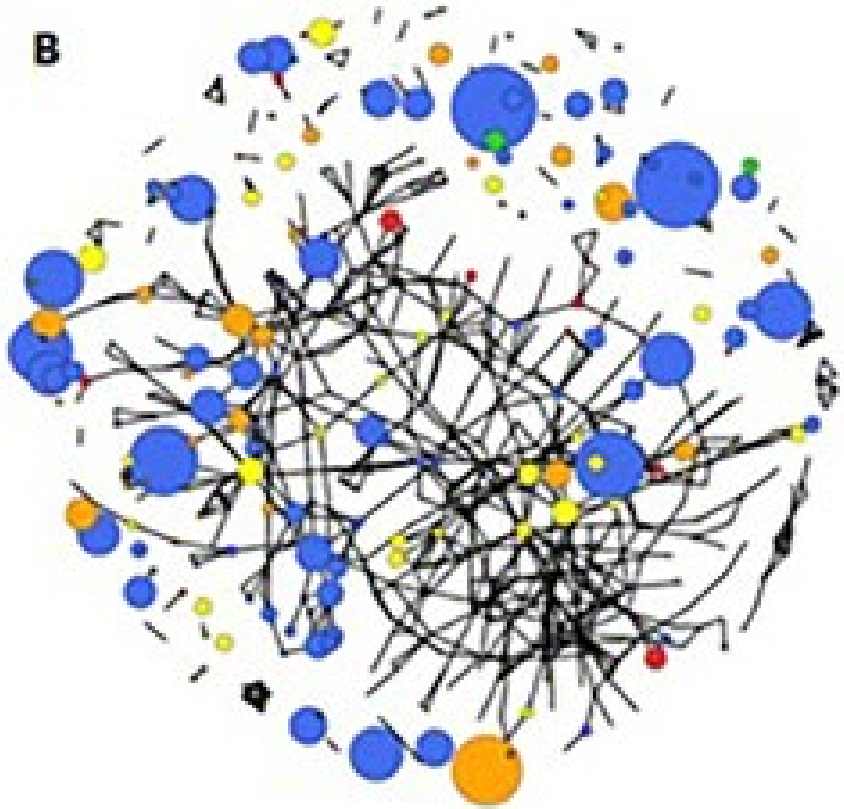
“一带一路”沿线国家科研合作网络的演化分析



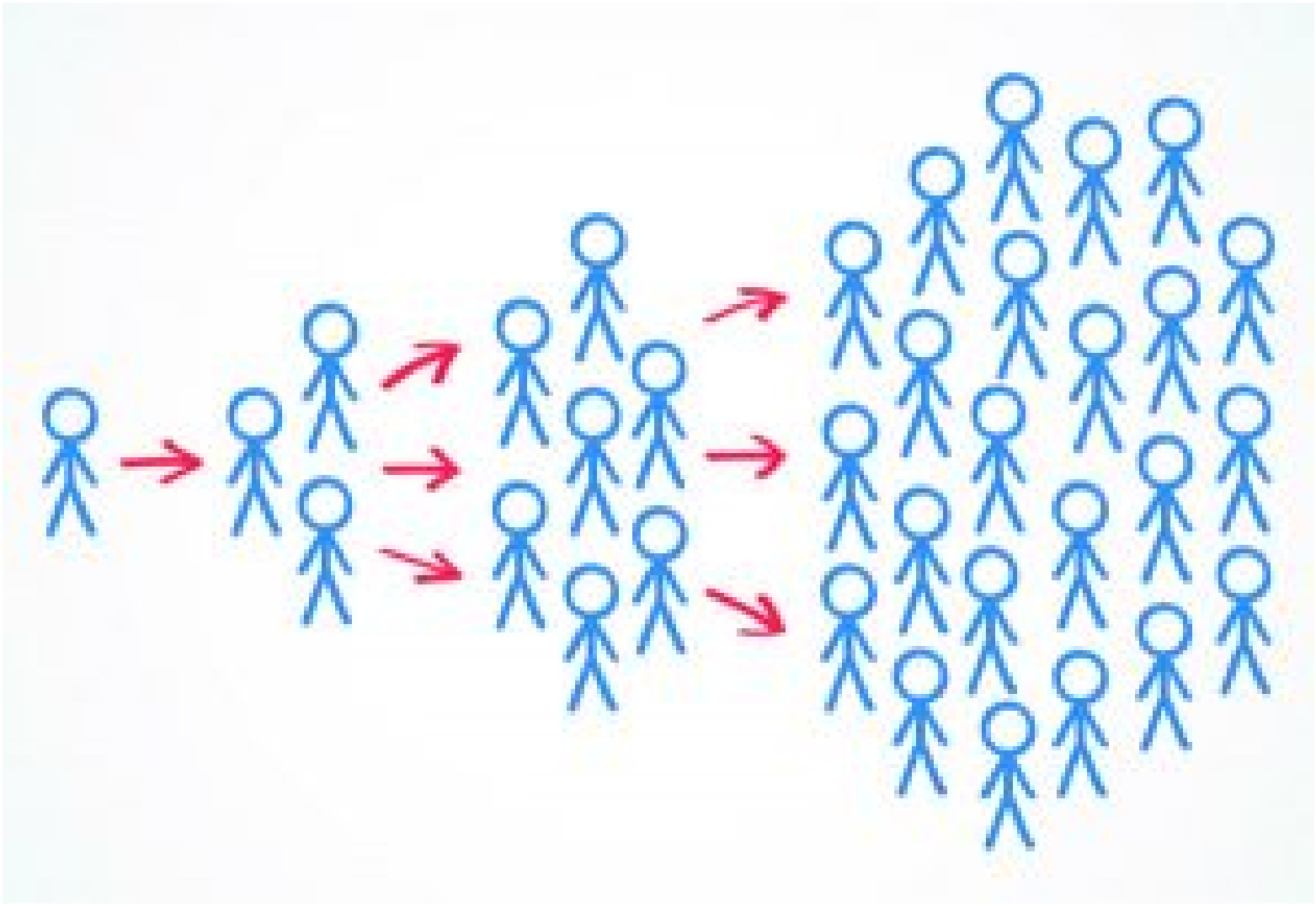
# 社会网络分析的应用：结构分析 (Structural analysis)

针对于国际（地区）合作网络的分析：区域国别政策意义

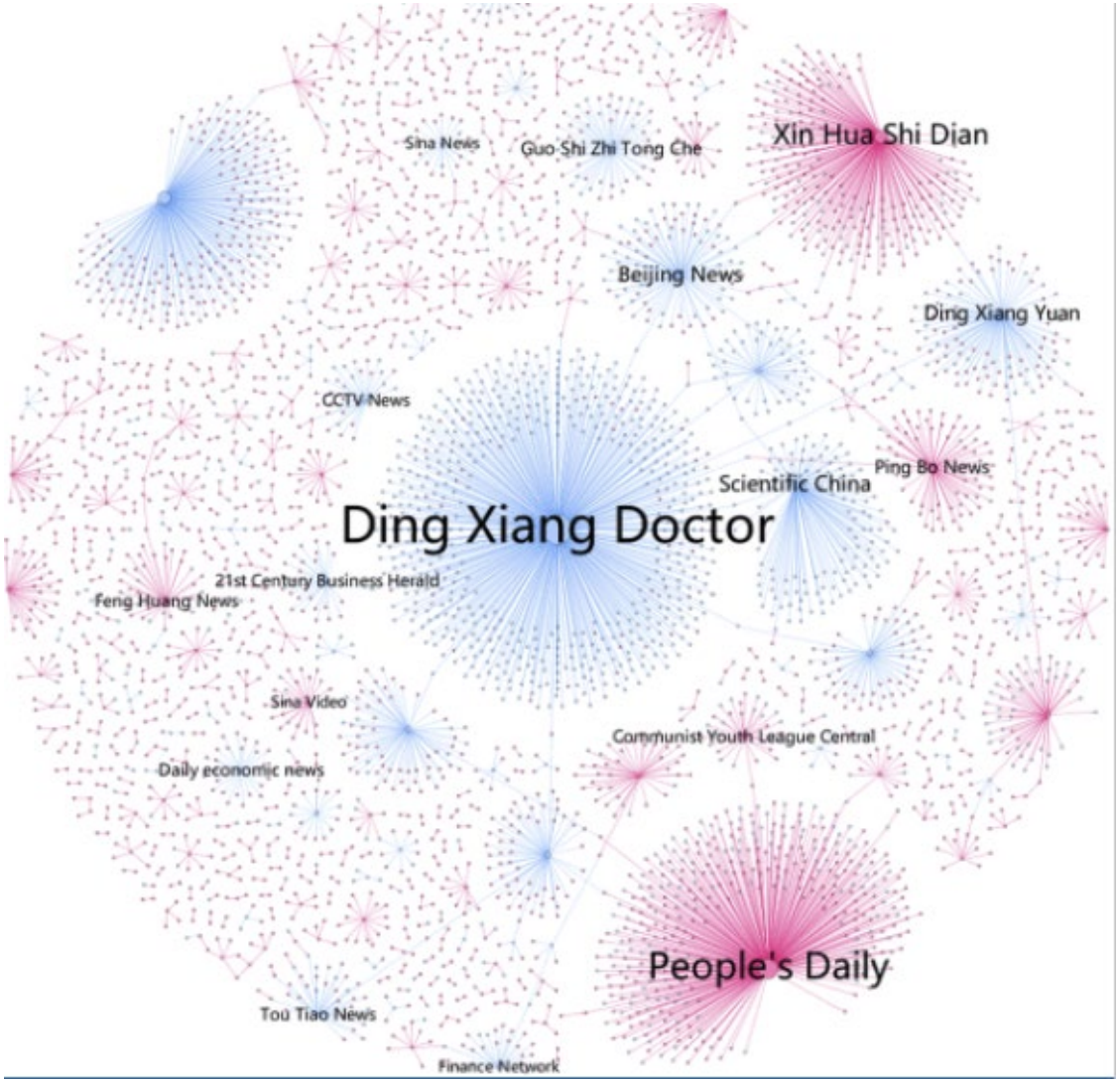


# 社会网络分析的应用：信息级联 (Information cascade)



# 社会网络分析的应用：信息级联 (Information cascade)



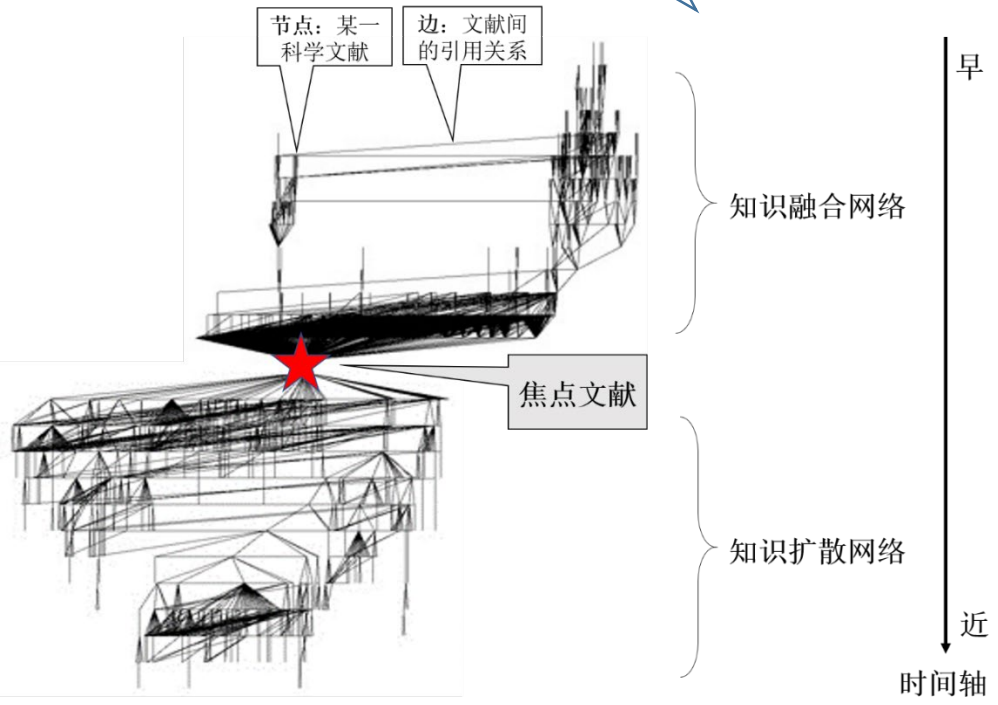

**新冠肺炎期间部分微博的信息级联图示**

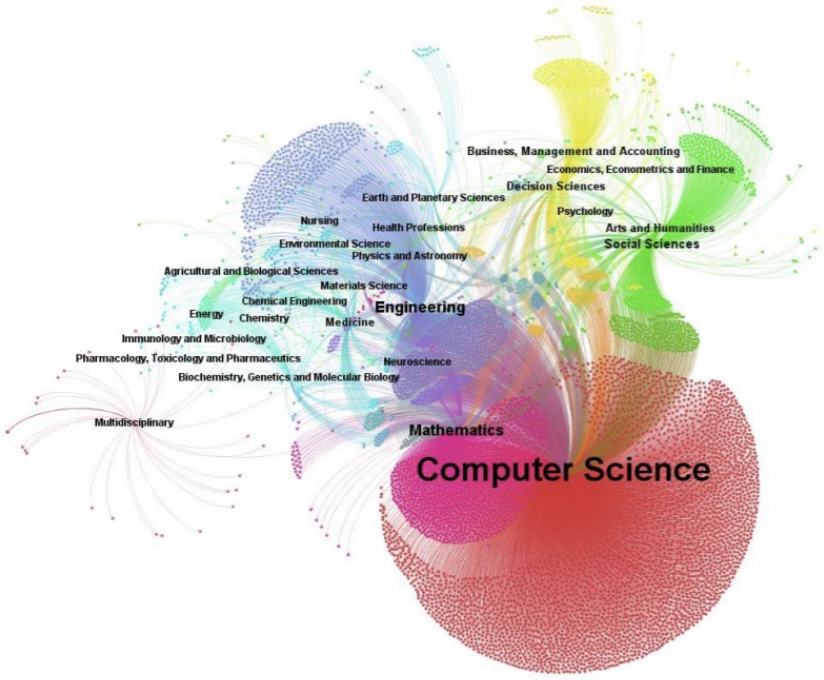
- **“意见领袖”**
- **谣言和真话哪个容易传播?**

# 社会网络分析的应用：信息级联 (Information cascade)

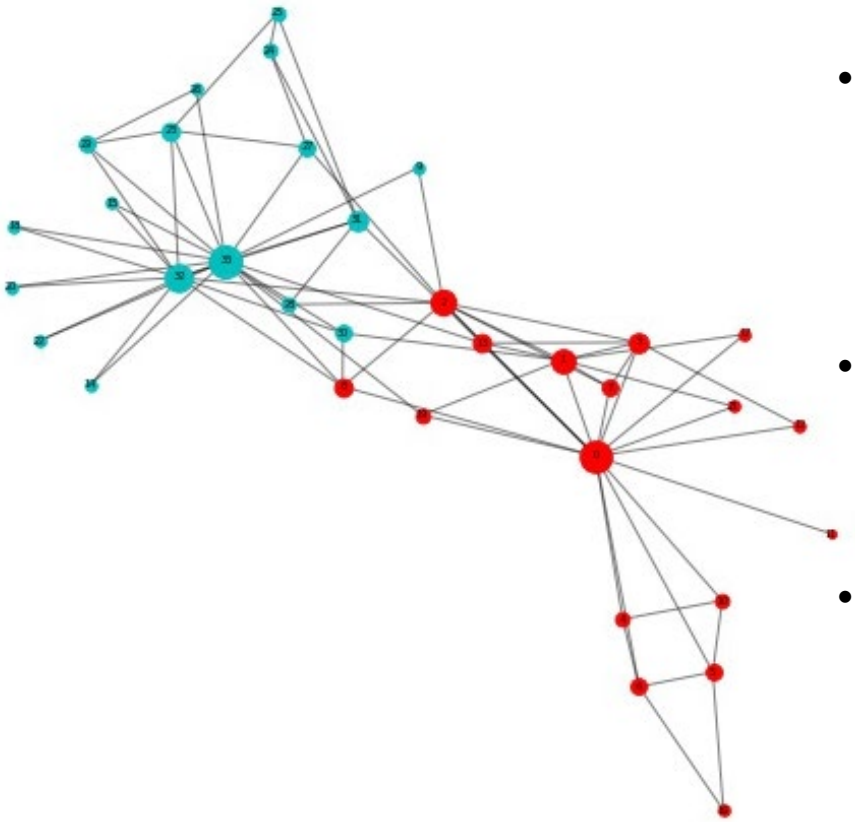
**交叉科学的“前世今生”：**
- **知识如何融合的?**
- **知识如何扩散和影响的?**

**以计算机学科的交叉科学知识扩散为例：**



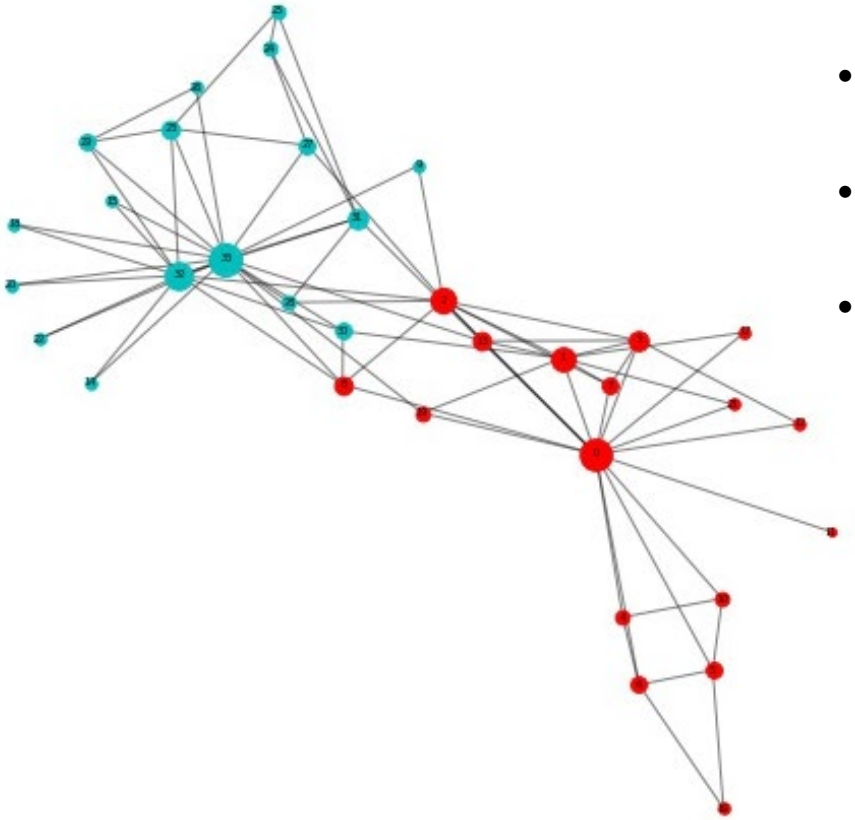



# 社会网络分析的应用：传染病模型(epidemic model)



- 易染状态S（Susceptible）：一个个体在被传染之前的状态，即可能会被与自己相连的节点传播（但不可能被与自己没有相连的节点传播，因为没有可致病接触）
- 感染状态I（Infected）：一个受感染的个体所处的状态，该个体可能被治愈，可能死亡，同时在该状态下的个体可以传播病毒给与其相邻的节点
- 免疫状态R（Recovered）：一个个体在被感染之后被治愈从而恢复健康后的状态，我们假定一个个体一旦从患病状态中被治愈，则就具有了对该病毒的抗体从而不会再次被感染

# 社会网络分析的应用：传染病模型 (epidemic model)



- 随机免疫 (Random immunization)，也称均匀免疫 (Uniform immunization)
- 目标免疫 (Targeted immunization )，也称选择免疫 (Selected immunization)
- 熟人免疫 (Acquaintance immunization)

# 社会网络分析的应用：传染病模型(epidemic model)

- 视频

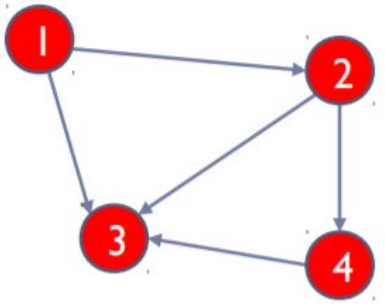
# 网络中的基本元素：节点和边

- 网络是一个有序对$G = \langle V, E \rangle$，$V$是节点集，$E$是边集
  - 节点（node/vertex）数目：$|V|$
  - 边（edge/tie/relation）数目：$|E|$
- 网络的分类：
  - 按照边有无方向分为：有向图（direct networks）和无向图（undirected networks）
  - 按照边是否有权重分为：有权图（unweighted networks）和无权图（weighted networks）
  - ……
- 节点的度数（degree）
  - 无向图：度数
  - 有向图：出度、入度

# 有向网络的表示

表示形式1：边列表

Graph (directed)




Edge list

| Vertex | Vertex |
|---|---|
| 1 | 2 |
| 1 | 3 |
| 2 | 3 |
| 2 | 4 |
| 3 | 4 |

表示形式2：邻接矩阵

Adjacency matrix

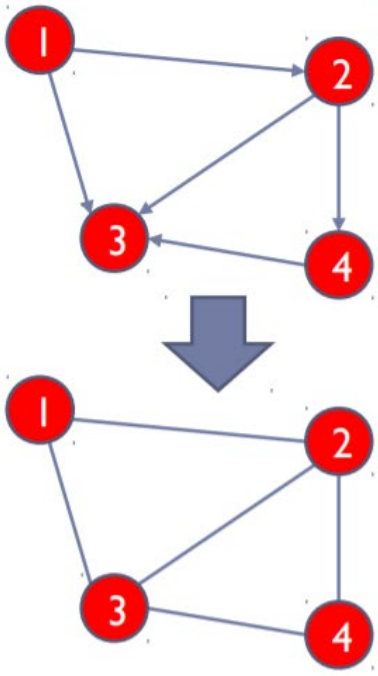
| Vertex | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | - | 1 | 1 | 0 |
| 2 | 0 | - | 1 | 1 |
| 3 | 0 | 0 | - | 0 |
| 4 | 0 | 0 | 1 | - |

# 无向网络的表示

表示形式1：边列表

**Directed**

*(who contacts whom)*




**Undirected**

*(who knows whom)*

**Edge list remains the same**

| Vertex | Vertex |
|---|---|
| 1 | 2 |
| 1 | 3 |
| 2 | 3 |
| 2 | 4 |
| 3 | 4 |

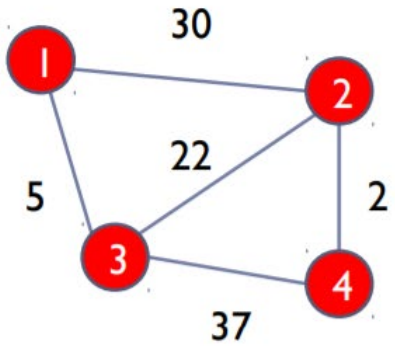
But interpretation is different now

表示形式2：邻接矩阵

**Adjacency matrix becomes symmetric**

| Vertex | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | - | 1 | 1 | 0 |
| 2 | 1 | - | 1 | 1 |
| 3 | 1 | 1 | - | 1 |
| 4 | 0 | 1 | 1 | - |

# 有权网络的表示




Weights could be:

- *Frequency of interaction in period of observation*
- *Number of items exchanged in period*
- *Individual perceptions of strength of relationship*
- *Costs in communication or exchange, e.g. distance*
- *Combinations of these*

边列表中增加了权重列

**Edge list: add column of weights**

| Vertex | Vertex | Weight |
|---|---|---|
| 1 | 2 | 30 |
| 1 | 3 | 5 |
| 2 | 3 | 22 |
| 2 | 4 | 2 |
| 3 | 4 | 37 |

邻接矩阵增加了权重信息

**Adjacency matrix: add weights instead of 1**

| Vertex | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | - | 30 | 5 | 0 |
| 2 | 30 | - | 22 | 2 |
| 3 | 5 | 22 | - | 37 |
| 4 | 0 | 2 | 37 | - |

# 网络中的多重边和自环

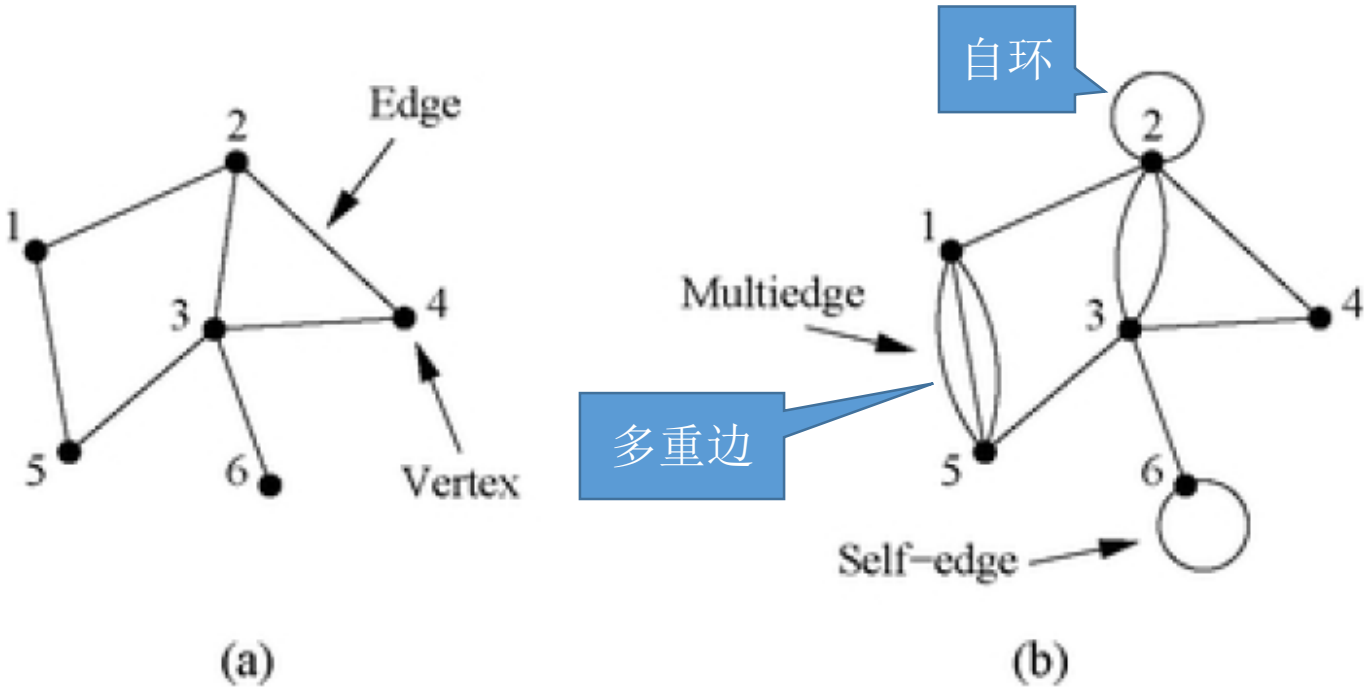



Figure 6.1: Two small networks. (a) A simple graph, i.e., one having no multiedges or self-edges. (b) A network with both multiedges and self-edges.

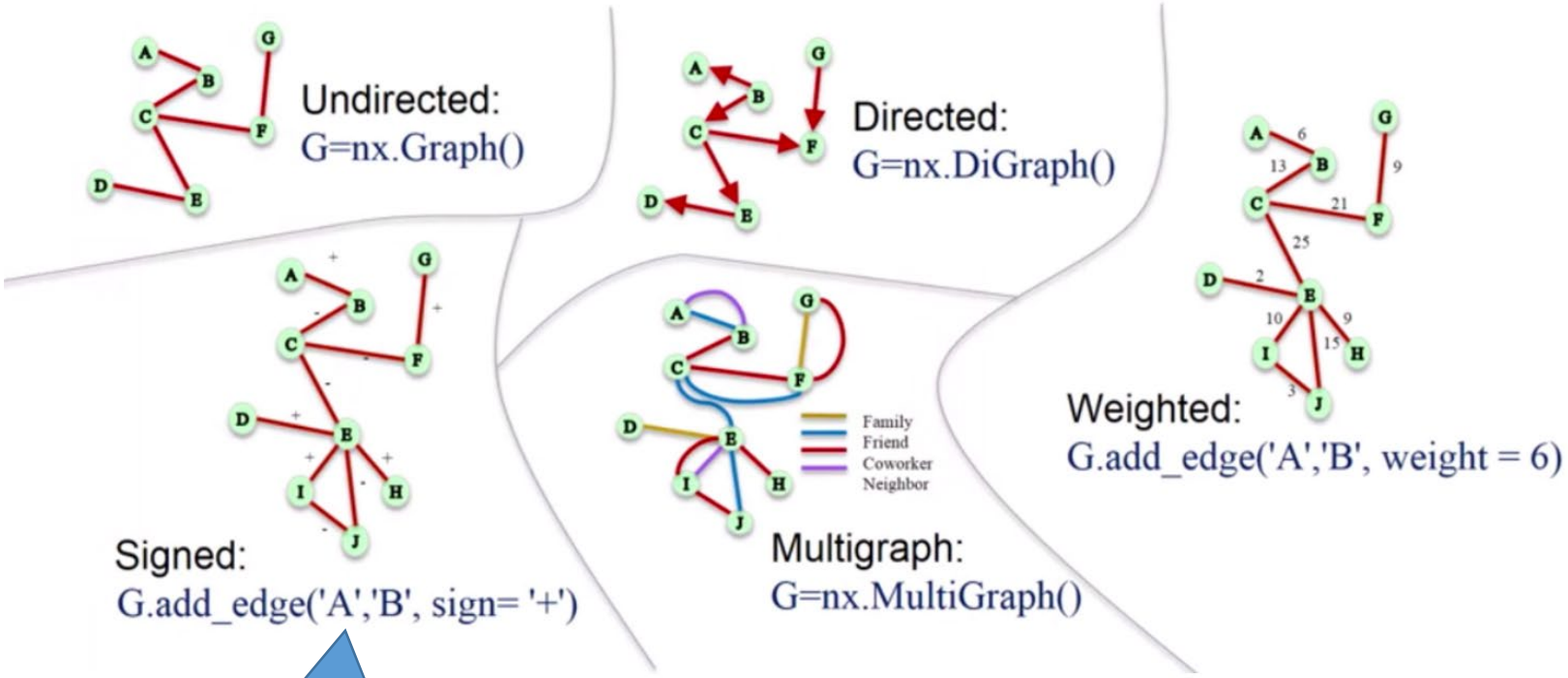
# Python中的networkx包

- 手动安装networkx包
  - http://pypi.python.org/pypi/networkx
- 使用Python内嵌的package安装工具：
  - $ easy install networkx
- 使用pip命令
  - $ pip install networkx
- ……

# 【实践】I. Network Basics

- 打开Jupyter Notebook

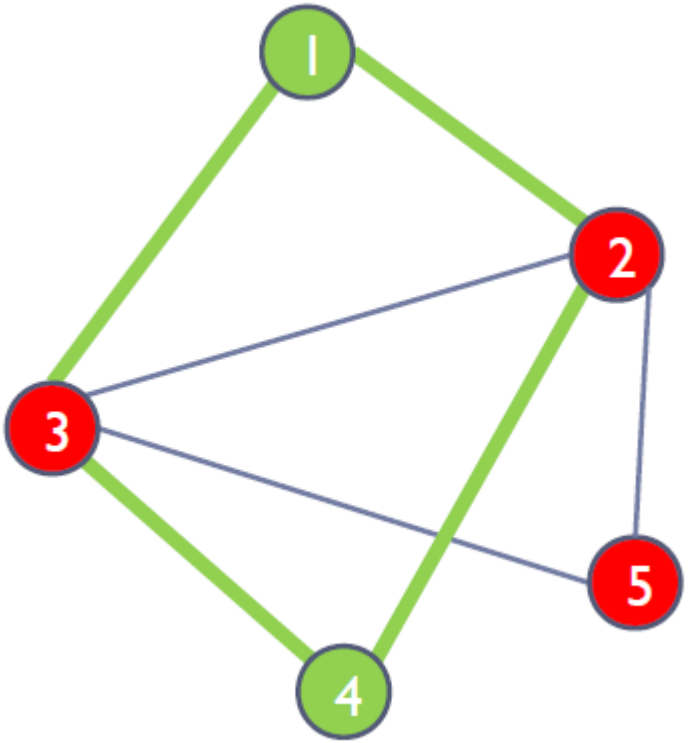
# 几种类型网络及其初始化




符号网络

# 路径（path）

- 路径、路径的长度
- 最短路径（距离）
- networkx中的路径实现

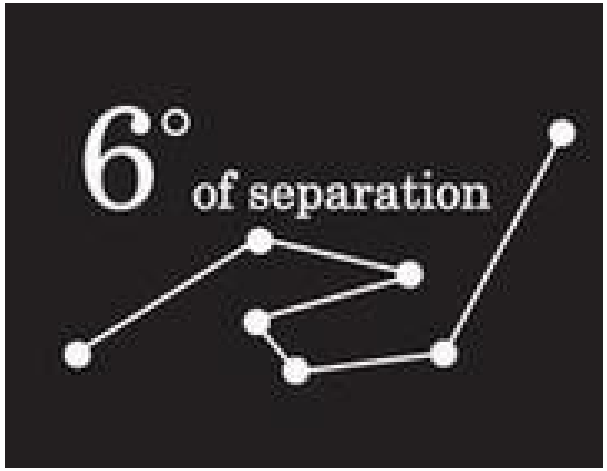


# 路径（path）

- 计算图中的最短路径
  - shortest_path(G[, source, target, weight, …])
- 计算图中最短路径的长度
  - shortest_path_length(G[, source, target, …])
- 计算图中的平均最短路径的长度
  - average_shortest_path_length(G[, weight, method])
- 如果G中有路径从source点到target点，那么返回True，反之False
  - has_path(G, source, target)

# 社会网络分析中的重要理论

六度分割理论




150定律（邓巴数字）

# 六度分割理论

- 意义：
  - 世界上所有互不相识的人只需要很少（6位）中间人就能建立起联系。称为小世界现象（又称小世界效应）
- 缘起：
  - 1967哈佛大学心理学教授 Stanley Milgram 为了解人际网络所进行的连锁信实验，发现了人与任何陌生人之间的间隔，平均来说不会超过6个。

# 六度分割理论

- 换言之，只要透过6个人就可以认识任何陌生人
- 实验：
  - 在美国两个州随机选择出其中三百多名志愿者。
  - 请他们邮寄一封信给一个陌生人，由于不可能直接送达，所以志愿者可以委请亲友代为邮寄，同时要求转寄者寄送通知给 Milgram 教授。
- 结果：
  - 共有 60 多封信被送达，且信件转寄的中间次数平均是 5。

# 六度分割理论

- 发展：
  - 1998年,美国哥伦比亚大学社会学系任教的Duncan Watts与其博士论文导师数学家Strogatz基于前人的理论提出了“Small-World Networks”，即W-S小世界网络模型
  - 他们在这一模型中认为现实世界中的许多网络既不可能是完全规则的封闭式环状网，也不会是完全随机的散点式网络，介于这两者之间的就是“小世界模型”

# 六度分割理论

- 小世界网络同时具有“高网络聚集度”和“低平均路径”的特性：
  - 聚集意味着如果个体有共同的朋友，则两个个体更有可能成为朋友
  - “低平均路径”意味着任意两个人可以通过比较短的中间连接取得联系
- 拓展：
  - Watts与Strogatz也先后在疾病传播、昆虫神经系统和电力系统中揭示出现实网络确实符合小世界模型

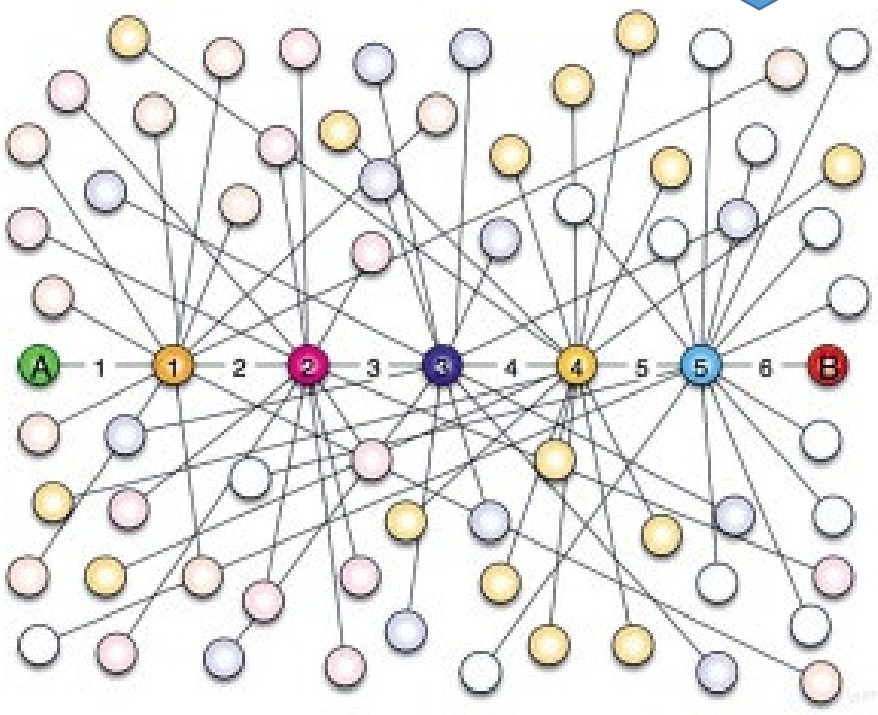
# 六度分割理论

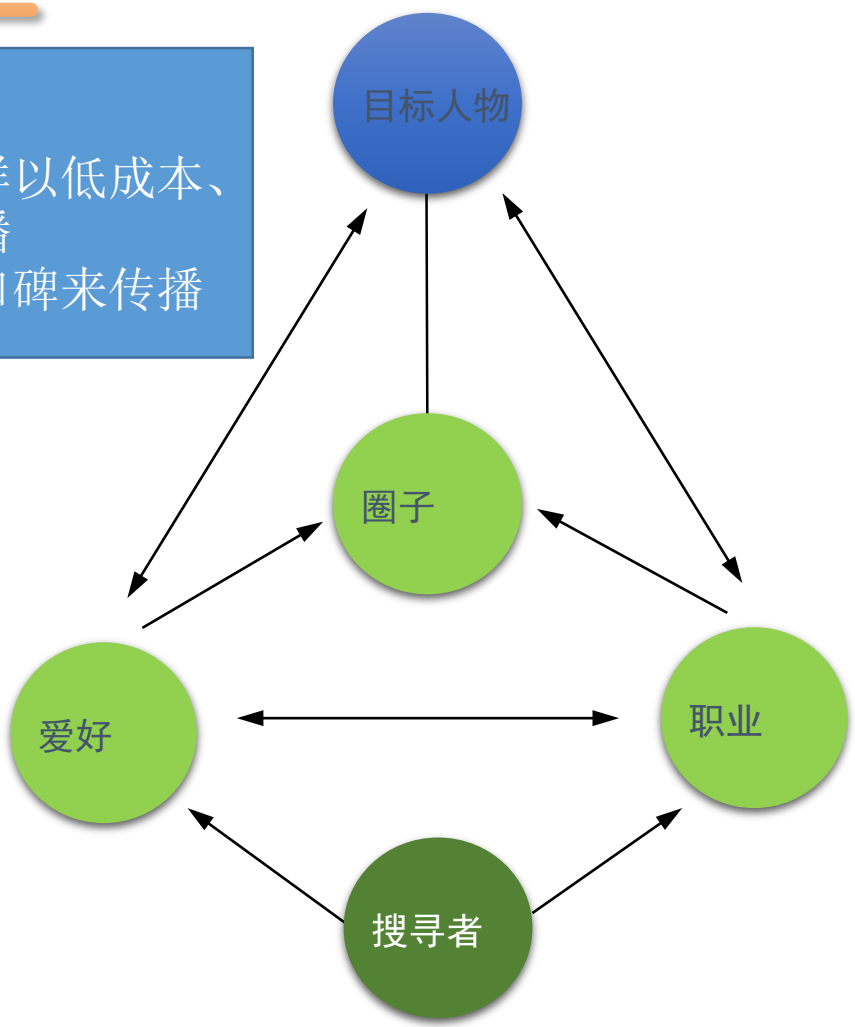
- 有何启示？

# 六度分割理论

- 有何启示？

病毒式营销：
- 像病毒一样以低成本、高效率传播
- 利用用户口碑来传播



# 常用指标

- 微观：节点层面
- 中观：社区层面
- 宏观：网络层面

# 常用指标：微观层面

- 哪个节点更为“重要”？
  - 中心性
    - 度数中心性
    - 中介中心性
    - 接近中心性
  - PageRank：二下《信息存储与检索》、三上《信息计量学》

# 度数中心性（degree centrality）

某点的度数中心度=
与这个点直接相连的其他点的个数 ÷ 图中点的最大可能的度数(即|V|-1)

| | |
|---|---|
| `degree_centrality`(G) | Compute the degree centrality for nodes. |
| `in_degree_centrality`(G) | Compute the in-degree centrality for nodes. |
| `out_degree_centrality`(G) | Compute the out-degree centrality for nodes. |

## networkx.algorithms.centrality.degree_centrality

`degree_centrality`(G) [source]

Compute the degree centrality for nodes.

The degree centrality for a node v is the fraction of nodes it is connected to.

**Parameters:** G (*graph*) – A networkx graph

**Returns:** **nodes** – Dictionary of nodes with degree centrality as the value.

**Return type:** dictionary

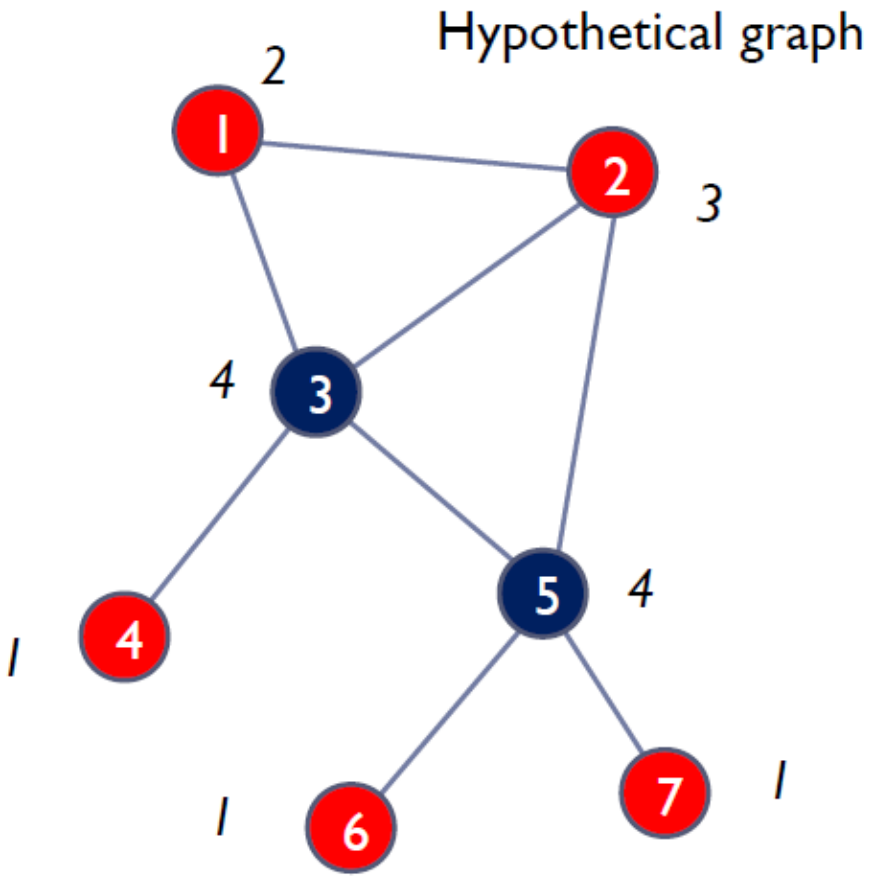
**See also**

`betweenness_centrality()`, `load_centrality()`, `eigenvector_centrality()`

Notes

The degree centrality values are normalized by dividing by the maximum possible degree in a simple graph n-1 where n is the number of nodes in G.

For multigraphs or graphs with self loops the maximum degree might be higher than n-1 and values of degree centrality greater than 1 are possible.



# 中介中心性（betweenness centrality）

- 某节点出现在其他节点间最短路径上的标准化次数

$$BC = \sum \frac{d_{st}()}{d_{st}}$$

## networkx.algorithms.centrality.betweenness_centrality

`betweenness_centrality`(*G, k=None, normalized=True, weight=None, endpoints=False, seed=None*) [source]

Compute the shortest-path betweenness centrality for nodes.

Betweenness centrality of a node $v$ is the sum of the fraction of all-pairs shortest paths that pass through $v$

$$c_B(v) = \sum_{s,t \in V} \frac{\sigma(s,t|v)}{\sigma(s,t)}$$

where $V$ is the set of nodes, $\sigma(s,t)$ is the number of shortest $(s,t)$-paths, and $\sigma(s,t|v)$ is the number of those paths passing through some node $v$ other than $s,t$. If $s=t$, $\sigma(s,t)=1$, and if $v \in s,t$, $\sigma(s,t|v)=0$ [2].

**Parameters:**

- **G** (*graph*) – A NetworkX graph.
- **k** (*int, optional (default=None)*) – If k is not None use k node samples to estimate betweenness. The value of k <= n where n is the number of nodes in the graph. Higher values give better approximation.
- **normalized** (*bool, optional*) – If True the betweenness values are normalized by `2/((n-1)(n-2))` for graphs, and `1/((n-1)(n-2))` for directed graphs where `n` is the number of nodes in G.
- **weight** (*None or string, optional (default=None)*) – If None, all edge weights are considered equal. Otherwise holds the name of the edge attribute used as weight.
- **endpoints** (*bool, optional*) – If True include the endpoints in the shortest path counts.
- **seed** (*integer, random_state, or None (default)*) – Indicator of random number generation state. See Randomness. Note that this is only used if k is not None.

**Returns:** **nodes** – Dictionary of nodes with betweenness centrality as the value.

**Return type:** dictionary

# 接近中心性（closeness centrality）

- 如果一个点与网络中所有其他点的“距离”都很短，则称该点具有较高的接近中心度。

**networkx.algorithms.centrality.closeness_centrality**

`closeness_centrality(G, u=None, distance=None, wf_improved=True)` [source]

Compute closeness centrality for nodes.

Closeness centrality [1] of a node `u` is the reciprocal of the average shortest path distance to `u` over all `n-1` reachable nodes.

$$C(u) = \frac{n-1}{\sum_{v=1}^{n-1} d(v,u)},$$

where `d(v, u)` is the shortest-path distance between `v` and `u`, and `n` is the number of nodes that can reach `u`. Notice that the closeness distance function computes the incoming distance to `u` for directed graphs. To use outward distance, act on `G.reverse()`.

Notice that higher values of closeness indicate higher centrality.

Wasserman and Faust propose an improved formula for graphs with more than one connected component. The result is “a ratio of the fraction of actors in the group who are reachable, to the average distance” from the reachable actors [2]. You might think this scale factor is inverted but it is not. As is, nodes from small components receive a smaller closeness value. Letting `N` denote the number of nodes in the graph,

$$C_{WF}(u) = \frac{n-1}{N-1}\frac{n-1}{\sum_{v=1}^{n-1} d(v,u)},$$

**Parameters:**
- **G** (*graph*) – A NetworkX graph
- **u** (*node, optional*) – Return only the value for node u
- **distance** (*edge attribute key, optional (default=None)*) – Use the specified edge attribute as the edge distance in shortest path calculations
- **wf_improved** (*bool, optional (default=True)*) – If True, scale by the fraction of nodes reachable. This gives the Wasserman and Faust improved formula. For single component graphs it is the same as the original formula.

**Returns:** **nodes** – Dictionary of nodes with closeness centrality as the value.

**Return type:** dictionary

$$CC = \frac{\sum_{t \in V} d(v,t)}{n-1}$$

# 三种中心性的比较

| Centrality measure | Interpretation in social networks |
| --- | --- |
| Degree | How many people can this person reach directly? |
| Betweenness | How likely is this person to be the most direct route between two people in the network? |
| Closeness | How fast can this person reach everyone in the network? |

# 三种中心性的比较

| Centrality measure | Other possible interpretations… |
| --- | --- |
| ▸ Degree | In network of music collaborations: how many people has this person collaborated with? |
| ▸ Betweenness | In network of spies: who is the spy though whom most of the confidential information is likely to flow? |
| ▸ Closeness | In network of sexual relations: how fast will an STD spread from this person to the rest of the network? |

# PageRank

- PageRank算法的基本算法：
  - 如果一个页面被多次引用，那么这个页面很可能是重要的；
  - 如果一个页面尽管没有被多次引用，但却被一个重要的页面引用，那么这个页面很可能是重要的；
  - 一个页面的重要性被均分，并传递到它所引用的页面。

# PageRank

PR值的决定因素：链入网页数、链入网页的质量、链入网页的链出网页数

- 假设一个由只有4个页面组成的集合：A，B，C和D。如果所有页面都链向A，那么A的PR（PageRank）值将是B，C及D的和。

$$PR(A) = PR(B) + PR(C) + PR(D)$$

- 继续假设B也有链接到C，并且D也有链接到包括A的3个页面。一个页面不能投票2次。所以B给每个页面半票。以同样的逻辑，D投出的票只有三分之一算到了A的PageRank上。

$$PR(A) = \frac{PR(B)}{2} + \frac{PR(C)}{1} + \frac{PR(D)}{3}$$

- 换句话说，根据链出总数平分一个页面的PR值。

$$PR(A) = \frac{PR(B)}{L(B)} + \frac{PR(C)}{L(C)} + \frac{PR(D)}{L(D)}$$

# PageRank

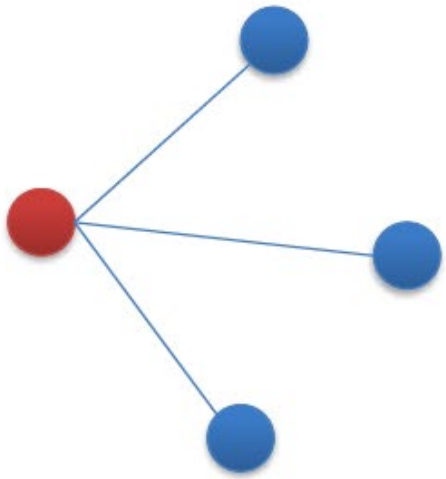
pagerank(G, alpha=0.85, personalization=None, max_iter=100, tol=1e-06, nstart=None, weight='weight', dangling=None) [source]

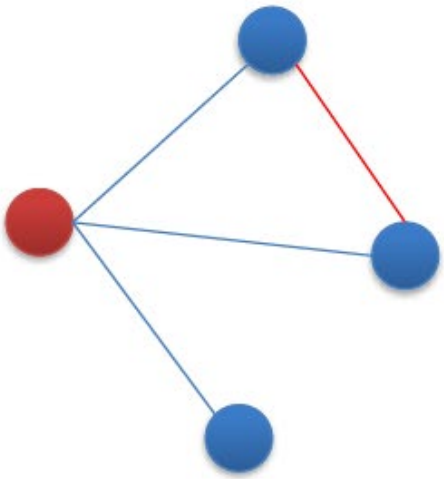
Return the PageRank of the nodes in the graph.

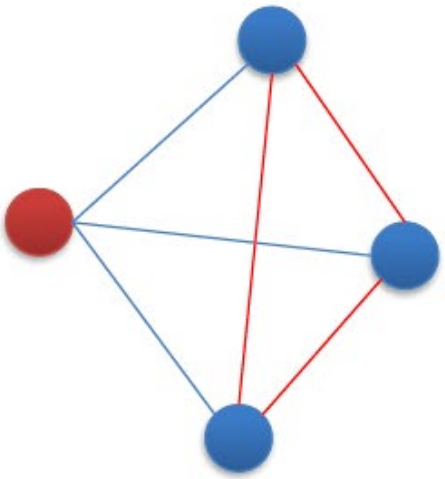
PageRank在networkx中

# 常用指标：中观层面

- 聚类系数（clustering coefficient）
  - $c_i = \dfrac{\text{节点}i\text{的邻居们相连对的数量}}{\text{节点}i\text{的邻居们形成对数的最大可能值}}$







# 常用指标：中观层面

"Unclustered" network

None of Ego's friends know each other*

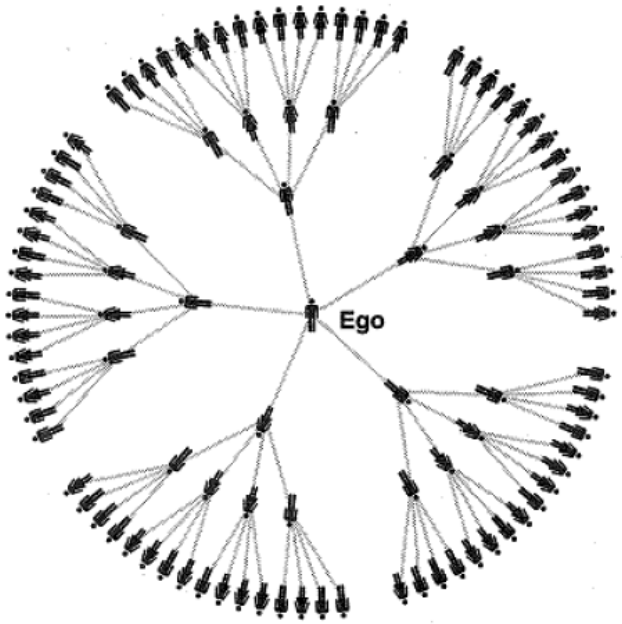



Figure 1.2. A pure branching network. Ego knows only 5 people, but within two degrees of separation, ego can reach 25; within three degrees, 105; and so on.

"Clustered" network

All of Ego's friends know each other

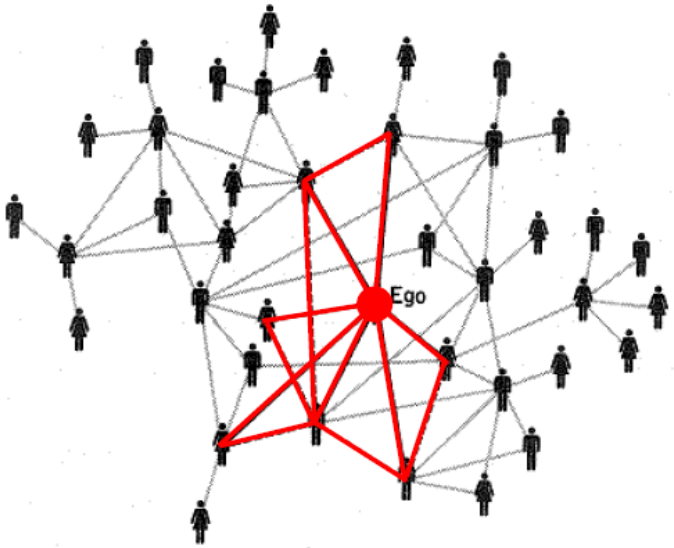



Figure 1.3. Real social networks exhibit clustering, the tendency of two individuals who share a mutual friend to be friends themselves. Here, Ego has six friends, each of whom is friends with at least one other.

# 常用指标：中观层面

- 聚类系数 -> 网络的平均聚类系数（average clustering coefficient）

| | |
|---|---|
| triangles(G[, nodes]) | Compute the number of triangles. |
| transitivity(G) | Compute graph transitivity, the fraction of all possible triangles present in G. |
| clustering(G[, nodes, weight]) | Compute the clustering coefficient for nodes. |
| average_clustering(G[, nodes, weight, …]) | Compute the average clustering coefficient for the graph G. |

# 常用指标：宏观层面

- 网络密度（network density）
  - 假设网络中有n个节点，实际边数为m
    - 无向网络：理论上边的最大可能值是n(n-1)/2，则其网络密度为：m/(n(n-1)/2)
    - 有向网络：理论上边的最大可能值是n(n-1)，则其网络密度为：m/(n(n-1))

```
nx.density(g)
```

# 常用指标：宏观层面

- 网络直径（network diameter）
  - 网络中任意两个节点的最长距离
- 度分布（degree distribution）

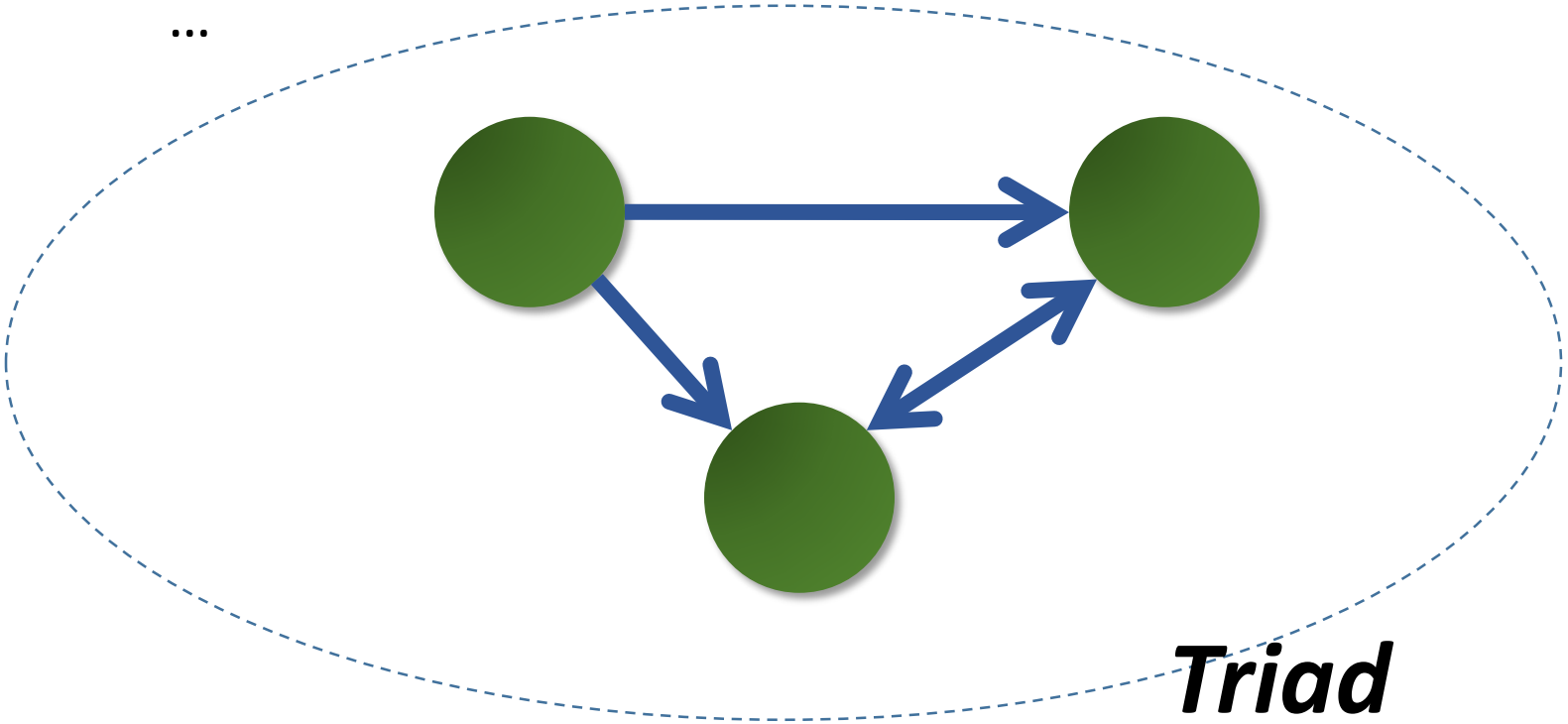
# SNA中的一些现象与规律

# 同质性（homophily）

- 相似属性的节点倾向于连接在一起
- 启示
  - 社交媒体
  - 推荐算法（电商平台）
  - 广告效应
  - …

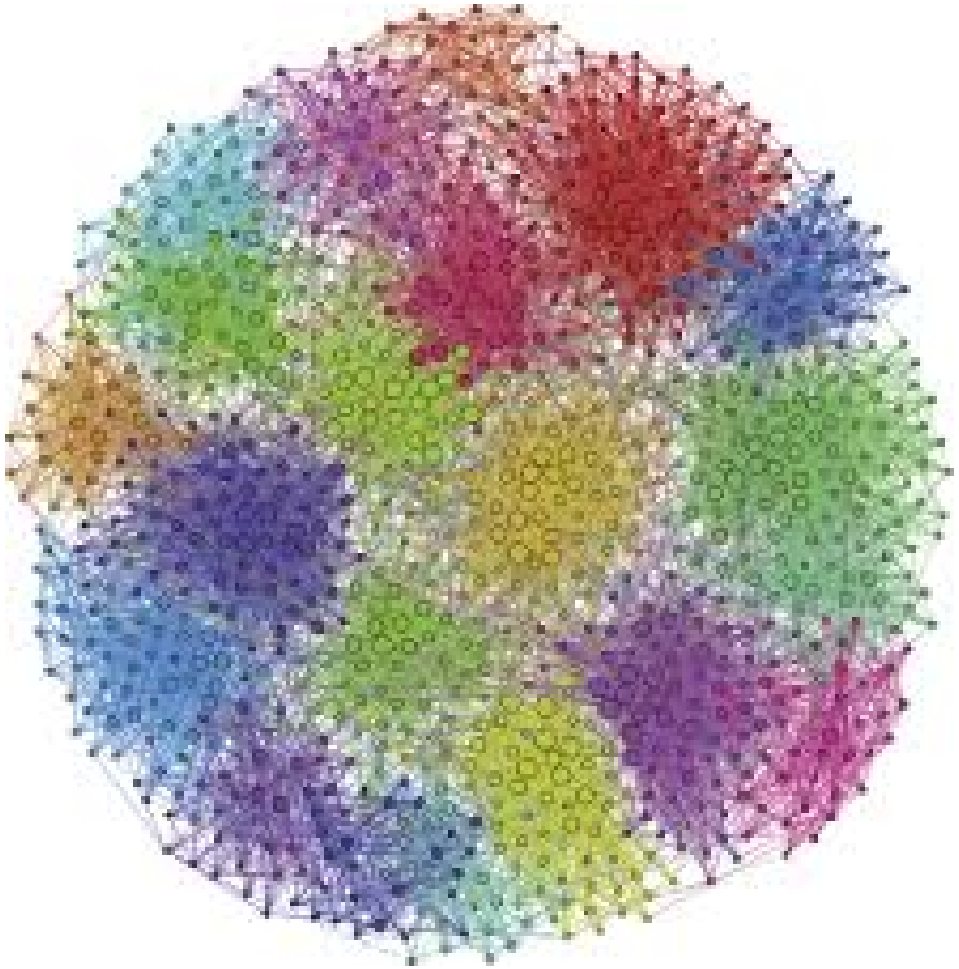
# 传递性

If A likes B and B likes C then A likes C  (transitivity)
If A likes B and C likes B then A likes C
…



***Triad***

# 社区划分



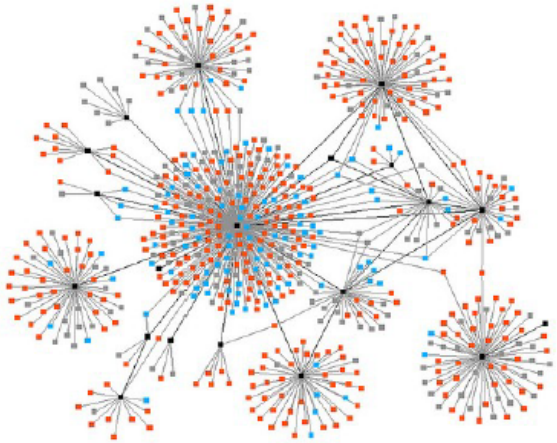
https://www.hindawi.com/journals/complexity/2019/6946189/

# 偏好连接性（preferential attachment）

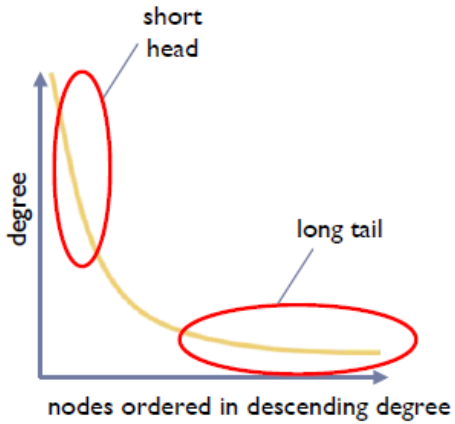
A property of some networks, where, during their evolution and growth in time, a the great majority of new edges are to nodes with an already high degree; the degree of these nodes thus increases disproportionately, compared to most other nodes in the network

- The result is a network with few very highly connected nodes and many nodes with a low degree
- Such networks are said to exhibit a *long-tailed* degree distribution
- And they tend to have a small-world structure!

*(so, as it turns out, transitivity and strong/weak tie characteristics are not necessary to explain small world structures, but they are common and can also lead to such structures)*

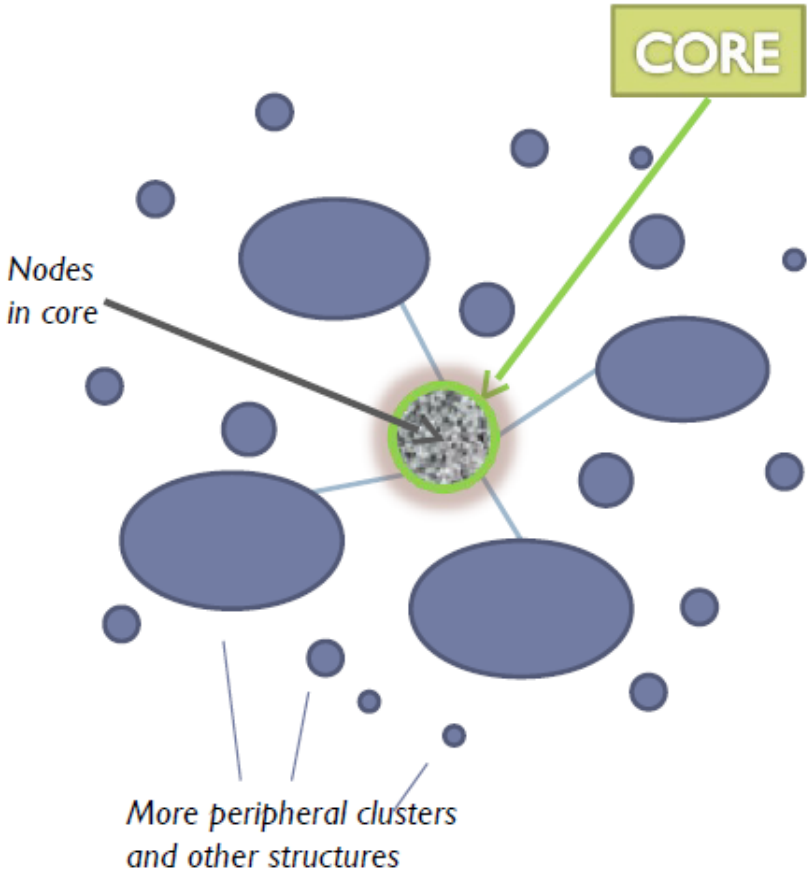


**Example of network with preferential attachment**



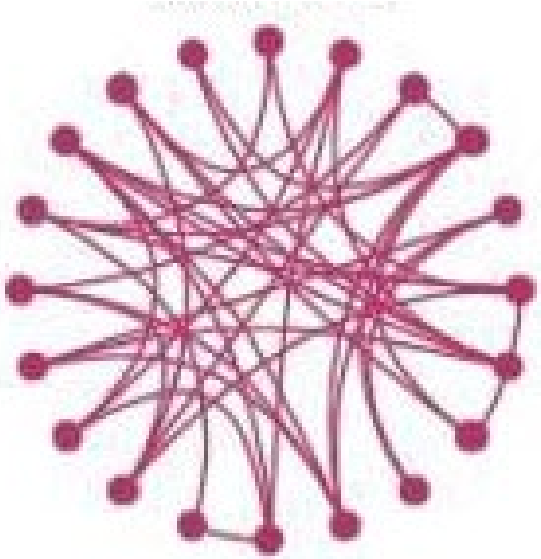

**Sketch of long-tailed degree distribution**

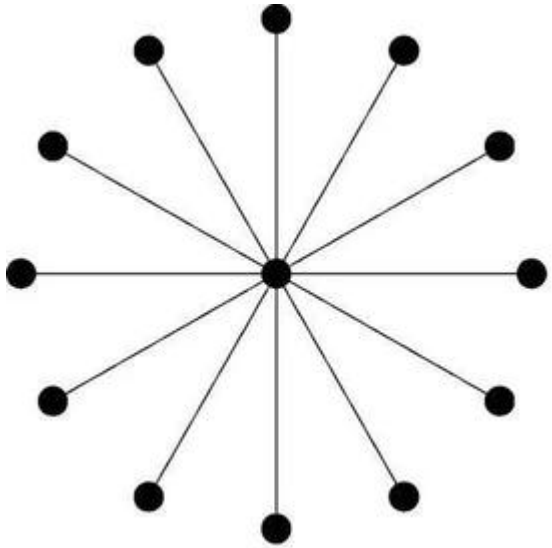
# 核心-边缘分布（core-peripheral）

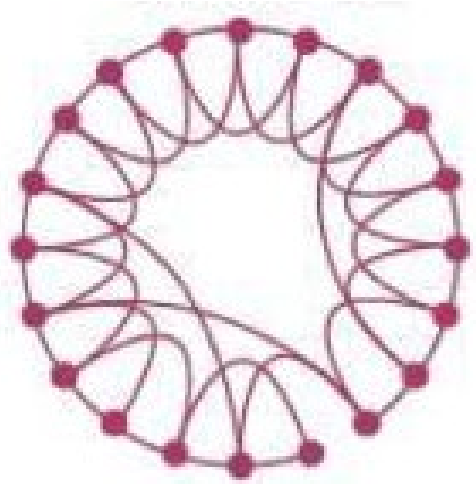


# 不同特质的网络

- 随机网络
- 星型网络
- 小世界网络







# 存储网络的几种文件格式

- GEXF

```
<?xml version="1.0" encoding="UTF-8"?>
<gexf xmlns="http://www.gexf.net/1.2draft" version="1.2">
    <meta lastmodifieddate="2009-03-20">
        <creator>Gexf.net</creator>
        <description>A hello world! file</description>
    </meta>
    <graph mode="static" defaultedgetype="directed">
        <nodes>
            <node id="0" label="Hello" />
            <node id="1" label="Word" />
        </nodes>
        <edges>
            <edge id="0" source="0" target="1" />
        </edges>
    </graph>
</gexf>
```

# 存储网络的几种文件格式

- GML

无节点标签

```
graph
[
  node
  [
   id A
  ]
  node
  [
   id B
  ]
  node
  [
   id C
  ]
   edge
  [
   source B
   target A
  ]
  edge
  [
   source C
   target A
  ]
]
```

有节点标签

```
graph
[
  node
  [
   id A
   label "Node A"
  ]
  node
  [
   id B
   label "Node B"
  ]
  node
  [
   id C
   label "Node C"
  ]
   edge
  [
   source B
   target A
   label "Edge B to A"
  ]
  edge
  [
   source C
   target A
   label "Edge C to A"
  ]
]
```

# 存储网络的几种文件格式

- CSV

### Edge List

The CSV example below represents a graph with two edges: "a" -> "b" and "b" -> "c".

```
a;b
b;c
```

### Adjacency List

All edges can be written as node pairs. It's also possible to write all node's connection on the same line. The example below represents a graph with 3 edges: "a" -> "b", "b" -> "c" and "b" -> "d"

```
a;b
b;c;d
```

# 存储网络的几种文件格式

- CSV

### Mixed

The following example shows various cases that CSV supports as well. Self-loops and mutual edges are supported. It's also possible to repeat an edge, "D" -> "E" is repeated twice in this example. **As a consequence the edge weight is incremented**. "D" -> "E" has a edge weight at two, whereas default value is one.

```
A,B
B,A
C,C
D,E
A,D
D,B,E
F,G,A,B
```

### Matrix

The sample below shows a graph with 5 nodes. An edge is created when the the cell is '1'.

```
;A;B;C;D;E
A;0;1;0;1;0
B;1;0;0;0;0
C;0;0;1;0;0
D;0;1;0;1;0
E;0;0;0;0;0
```

# 存储网络的几种文件格式

- 读/写的方式

- Adjacency List
  - Adjacency List
  - networkx.readwrite.adjlist.read_adjlist
  - networkx.readwrite.adjlist.write_adjlist
  - networkx.readwrite.adjlist.parse_adjlist
  - networkx.readwrite.adjlist.generate_adjlist
- Multiline Adjacency List
  - Multi-line Adjacency List
  - networkx.readwrite.multiline_adjlist.read_multiline_adjlist
  - networkx.readwrite.multiline_adjlist.write_multiline_adjlist
  - networkx.readwrite.multiline_adjlist.parse_multiline_adjlist
  - networkx.readwrite.multiline_adjlist.generate_multiline_adjlist

- Edge List
  - Edge Lists
  - networkx.readwrite.edgelist.read_edgelist
  - networkx.readwrite.edgelist.write_edgelist
  - networkx.readwrite.edgelist.read_weighted_edgelist
  - networkx.readwrite.edgelist.write_weighted_edgelist
  - networkx.readwrite.edgelist.generate_edgelist
  - networkx.readwrite.edgelist.parse_edgelist
- GEXF
  - Format
  - networkx.readwrite.gexf.read_gexf
  - networkx.readwrite.gexf.write_gexf
  - networkx.readwrite.gexf.generate_gexf
  - networkx.readwrite.gexf.relabel_gexf_graph
- GML
  - networkx.readwrite.gml.read_gml
  - networkx.readwrite.gml.write_gml
  - networkx.readwrite.gml.parse_gml
  - networkx.readwrite.gml.generate_gml
  - networkx.readwrite.gml.literal_destringizer
  - networkx.readwrite.gml.literal_stringizer

# 其他关于社会网络分析的内容

- 本科阶段：
  - 社会网络、图的整体介绍：《复杂网络理论与实践》课
  - 图的遍历：《数据结构与算法》课
  - 图的各种算法：《离散数学》《集合论与图论》《机器学习》《复杂网络理论与实践》课
- 研究生阶段：
  - 《社会网络分析》课

# 社会网络分析：课堂练习

- 详见Jupyter notebook

# 个人作业3

- 第一部分：社会网络分析
- 第二部分：机器学习基础

谢谢！