图的社区检测：Louvain算法&标签传播（LPA）

覃雄派

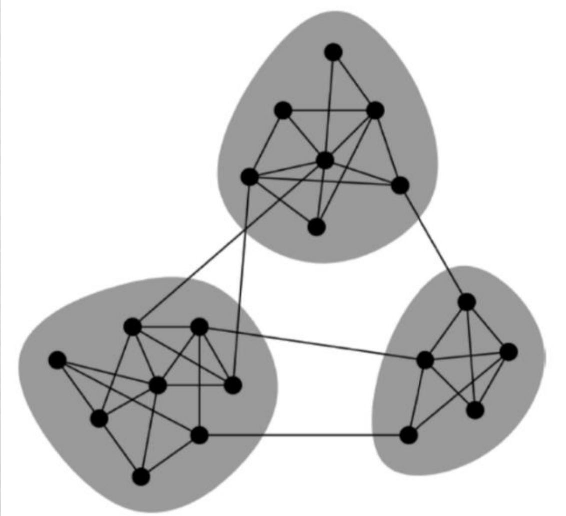
图的社区检测：
Louvain算法&标签传播（LPA）

提纲

- 弱连接理论（Weak Tie）
- 社区检测、问题定义
- Louvain社区检测算法
    - 模块度的定义
    - 课堂练习：模块度计算
    - Louvain算法的大流程与示例
    - 局部优化模块度、公式推导、实例
    - https://www.jianshu.com/p/f8aa22d33e2a
    - 课堂练习：局部优化模块度
- Louvain社区检测算法实践

图的社区检测：Louvain算法&标签传播（LPA）

- 弱连接理论（Weak Tie)
- 真实网络中为什么能形成社区？
  - 在分析数据之前，首先要对数据的性质有足够的认识
  - 很多时候，我们认为真实网络（如社交网络）长这样



家庭之间、单位之间……

果真如此吗？如果是真的话，
为什么会是这样？

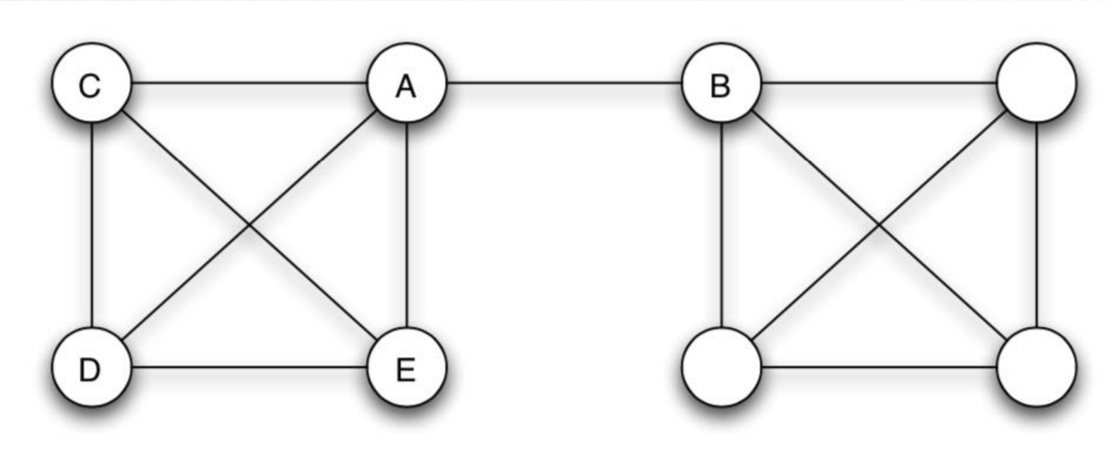
图的社区检测：Louvain算法&标签传播（LPA）

- 弱连接理论（Weak Tie）
- Mark Granovetter的研究
- 上世纪60年代末， Mark Granovetter在做他博士论文研究
  - 研究题目：人们是如何找到新的工作的
  - 发现1：人们通常是通过人际关系获取了新工作的信息
  - 发现2：获取新工作的人际关系通常是“点头之交” （casual acquaintances）而并非“亲密好友” （close friends）

- 这个发现很让人惊讶
  - 一般认为，亲密好友对你的帮助应该大于点头之交的熟人

图的社区检测：Louvain算法&标签传播（LPA）

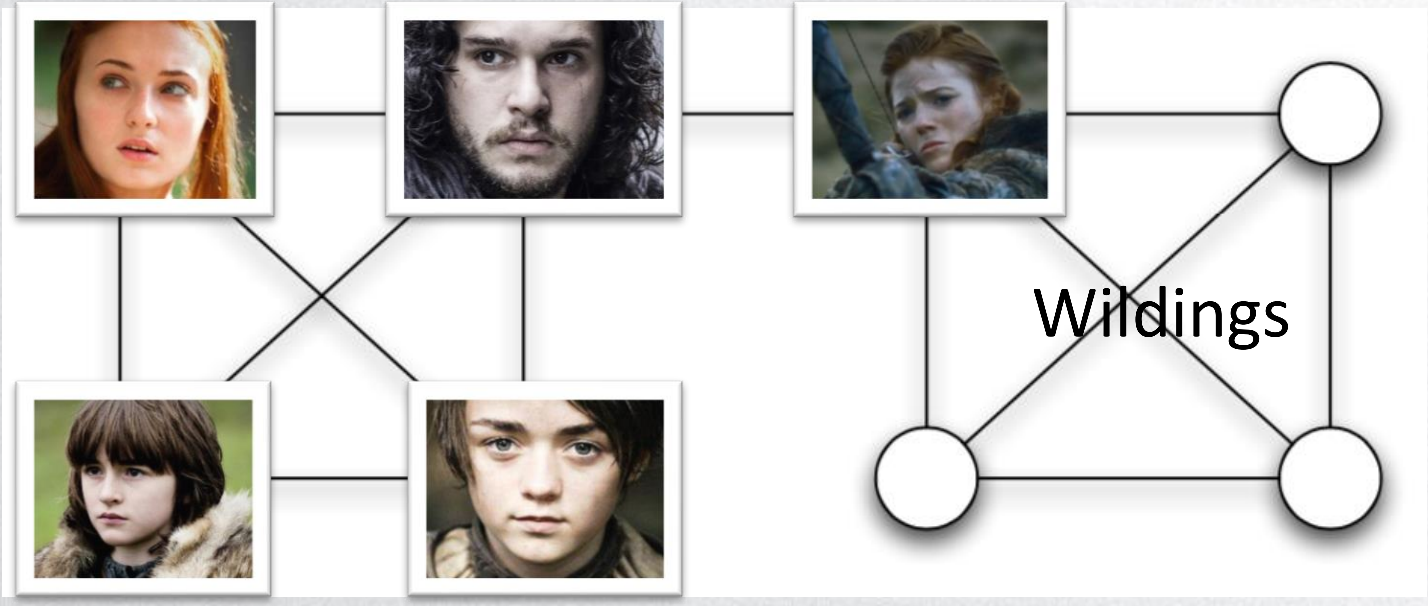
- 弱连接理论（Weak Tie)：强关系 vs. 弱关系
  - 怎么解释下图中A和B之间的关系?



没有A-B的关系，两个族群就没有联系了

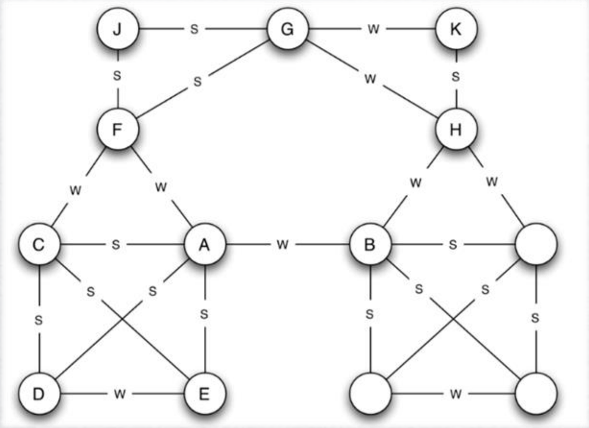
图的社区检测：Louvain算法&标签传播（LPA）

- 弱连接理论（Weak Tie)：强关系 vs. 弱关系
  - 怎么解释下图中A和B之间的关系?



图的社区检测：Louvain算法&标签传播（LPA）

- 弱连接理论（Weak Tie）：强关系 vs. 弱关系
  - Granovetter从结构和社交功能两个角度将边分为
    - 强关系 Strong Tie
    - 弱关系 Weak Tie



- 结构角度
  - 强关系意味着社交紧密
  - 弱关系链接网络不同部分
- 社交功能角度
  - 弱关系让你从不同角度获取信息，从而找到新工作
  - 强关系在新信息获取方面的作用十分有限

图的社区检测：Louvain算法&标签传播（LPA）

- 弱连接理论（Weak Tie)
- 真实数据中的强弱关系
- 测量了电话通信网络
  - 边的强弱表示打电话的次数

Structure and tie strengths in mobile communication networks

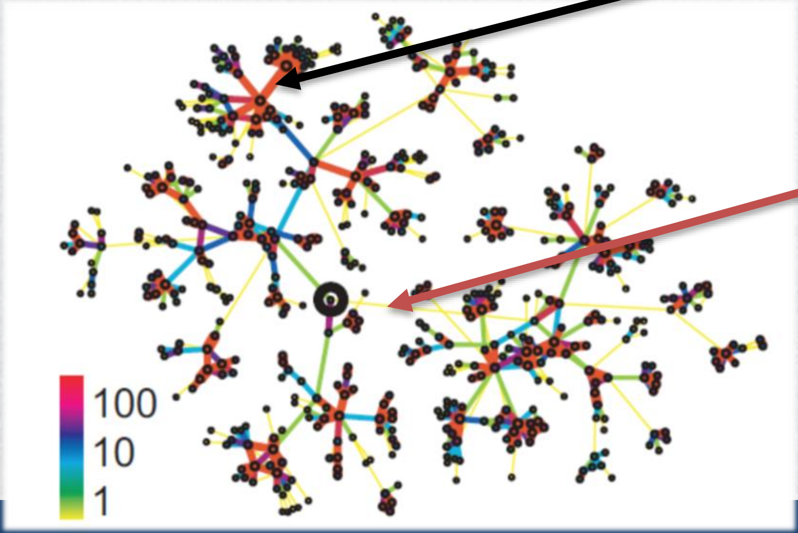
J.-P. Onnela, J. Saramäki, J. Hyvönen, G. Szabó, D. Lazer, K. Kaski, J. Kertész, and A.-L. Bara…

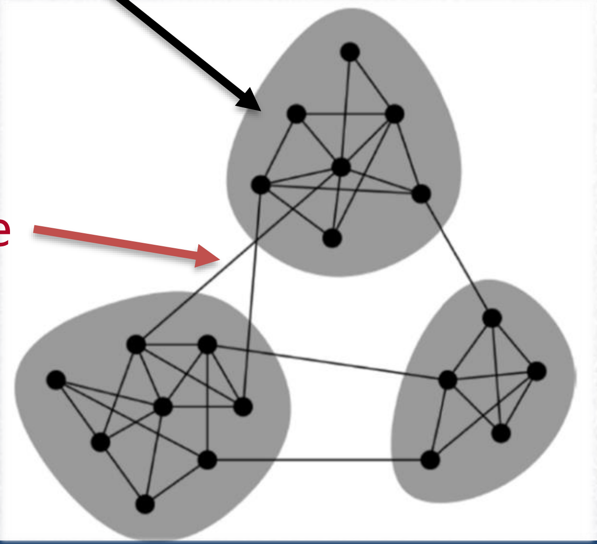
+ See all authors and affiliations

PNAS May 1, 2007 104 (18) 7332-7336; https://doi.org/10.1073/pnas.0610245104

Strong Tie

Weak Tie

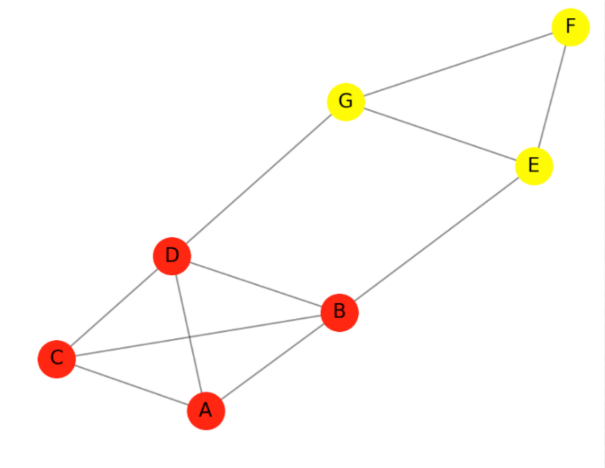




图的社区检测：Louvain算法&标签传播（LPA）

图的社区检测：Louvain算法&标签传播（LPA）

- 看一个小例子
  - 构造一个简单的社交网络
- 思考
  - 如何**自动地**将右图中红色的点与黄色的点分开?




```
def sample_community_graph ():
    G = nx.Graph()
    G.add_edges_from([(‘A’,‘B’), (‘A’,‘C’) ,
                ('A','D'), ('B','C',),('B','D'),
                ('C','D'), ('B','E'), ('D','G'),
                ('E', 'G'), ('E','F'), ('F','G')])
    return G
```

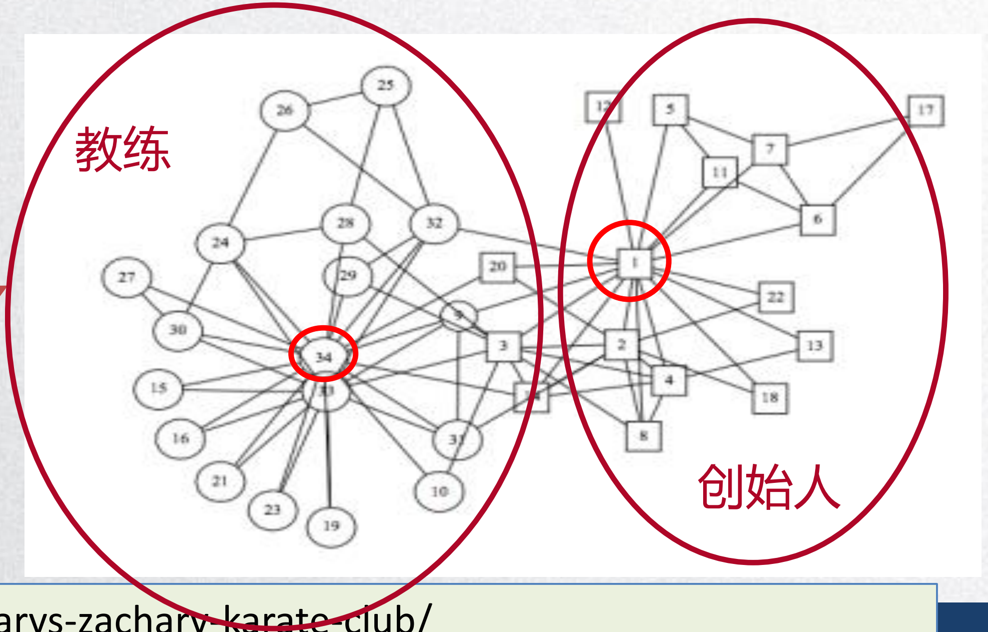
图的社区检测：Louvain算法&标签传播（LPA）

- 社区检测（Community Detection）
- 为什么使用图模型对数据建模?
  - 图提供了一种观察数据结构特征的视角

一个空手道俱乐部中34个成员之间朋友关系形成的图
你能发现什么特点?

最终这个俱乐部分裂成两个对立的空手道俱乐部

检测出图中的社区结构

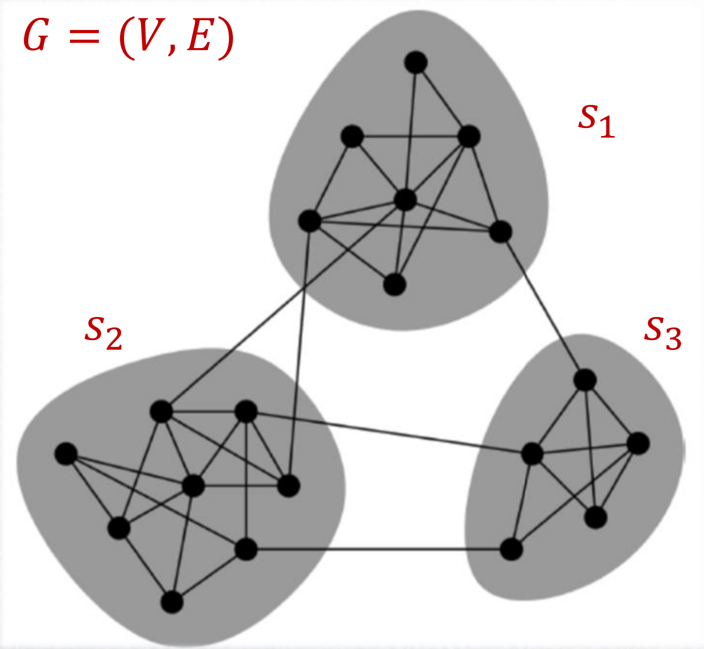


https://petterhol.me/2018/01/28/zacharys-zachary-karate-club/

图的社区检测：Louvain算法&标签传播（LPA）

- 问题定义
- 输入
  - 一个无向图$G = (V, E)$
- 输出
  - 一组点的划分S
  - $\forall s \in S, s \subseteq V$
  - $\forall_{i,j} s_i \cap s_j = \emptyset$ and $\cup_i s_i = V$

s_i为子集



- 设计优化目标！
  - 给定图G，评价S的质量
  - 你会怎么定义？

图的社区检测：Louvain算法&标签传播（LPA）

图的社区检测：Louvain算法&标签传播（LPA）

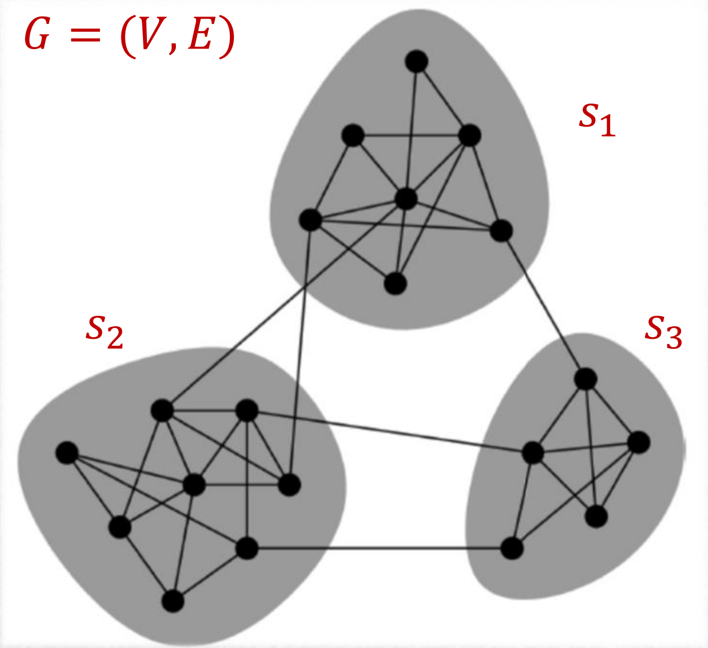
- Louvain社区检测算法
- 模块度定义
  - 模块度用来度量一个网络划分成社区的程度
  - 它的思想是：
    - 一个好的社区一定是内部的连接
    - 要比随机连接情况下的连接更紧密

图的社区检测：Louvain算法&标签传播（LPA）

- Louvain社区检测算法
- 模块度定义
  - 给定图G，度量一组划分 S的质量
  - 用模块度（Modularity Q ）
    - 度量划分S的紧凑性

$$Q \propto \sum_{s \in S} [\ (\#\ \text{edges within group } s) - \underbrace{(\text{expected}\ \#\ \text{edges within group } s)\ ]}_{\text{Need a null model!}}$$

如何计算expected# edges Within group s

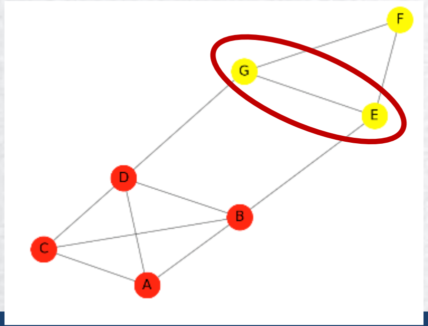


图的社区检测：Louvain算法&标签传播（LPA）

- 给定一个真实的图G（无向图），包含n个节点和m条边
  - 边的双向都算，那么边的总数为2m
  - 为了度量任意两点之间期望的边数，构造一个图G'，使其满足
    - G'与G有着相同的点的度数分布，但点之间的连接是随机的
    - G'为多重图（Multigraph）
  - 节点j连接到任意一个节点的概率是$\frac{k_j}{2m}$，现在节点i的度数为k_i，因此在随机情况下节点i与节点j的边的数量的期望值为$k_i\frac{k_j}{2m}$

$$\boxed{k_i \cdot \frac{k_j}{2m} = \frac{k_i k_j}{2m}}$$

  - 比如
    - （1）节点G连接到任何节点的概率是
      - 3/(2*11)=3/22
    - （2）节点E的度为3
    - （3）G和E的边的数量的期望值为
      - 3*3/22



n=7个节点
m=11条边

图的社区检测：Louvain算法&标签传播（LPA）

- 定义Modularity函数

$$Q(G,S)=\frac{1}{2m}\sum_{s\in S}\boxed{\sum_{i\in s}\sum_{j\in s}\left(A_{ij}-\frac{k_ik_j}{2m}\right)}$$

每个划分s内部
单独计算

  - 备注：A为邻接矩阵
  - （1）Modularity函数的范围是[-1,1]
  - （2）当社区内部边数大于预期边数的时候，模块度Q为正
  - （3）如果Modularity函数介于0.3-0.7，成为“显著社区结构”（significant community structure)

社区发现的思路：优化modularity函数

Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, Etienne Lefebvre. Fast unfolding of communities in large networks. https://arxiv.org/abs/0803.0476, 2008.

图的社区检测：Louvain算法&标签传播（LPA）

- 定义Modularity函数
- 模块度函数的另一种写法

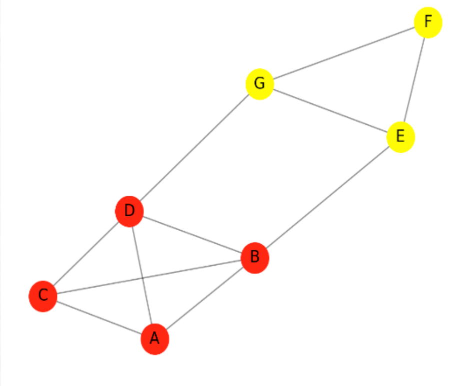
$$Q=\frac{1}{2m}\sum_{i,j}[A_{ij}-\frac{k_ik_j}{2m}]\delta(c_i,c_j)$$

$$\delta\left(c_i,c_j\right)=1, if\, c_i=c_j; \delta\left(c_i,c_j\right)=0, if\, c_i\neq c_j$$

- 式中A_{ij}为节点i和节点j之间的边的权重(这里以无向图为例)，当一个图不是带权的图的时候，所有边的权重为1
- $k_i=\sum_j A_{ij}$表示节点i相连的边的权重的和(即度数，一个节点和多少个其他节点相连，那么这个节点的度就是多少)
- c_i表示节点i所属的社区，δ(c_i,c_j)用来判断节点i和节点j是否在同一个社区内，如果在同一个社区内δ(c_i,c_j)=1，否则δ(c_i,c_j)=0
- $m=\frac{1}{2}\sum_{i,j}A_{ij}$表示所有边的权重的和(边的数量)
  - 每个节点都计算一次出度，由于每条边对应两个节点，所以在Q的计算中m要乘以2

图的社区检测：Louvain算法&标签传播（LPA）

- 课堂练习
- 计算右图中以下划分的Modularity分值
  - {{A}, {B}, {C}, {D}, {E}, {F}, {G}}
  - {{A,C}, {B,D}, {E,G}, {F}}
  - {{A,B,C,D}, {E,F,G}}
  - {{A,B,C,D,E,F,G}}



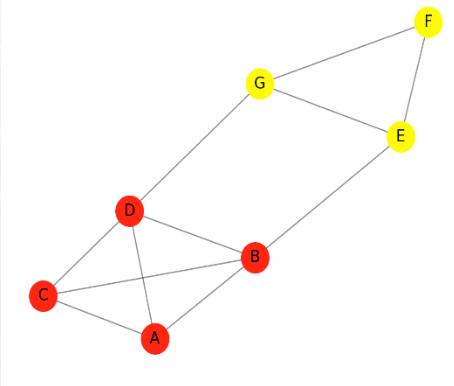
n=7个节点
m=11条边

$$Q = \frac{1}{2m}\sum_{i,j}[A_{ij} - \frac{k_i k_j}{2m}]\delta(c_i, c_j)$$

$$\delta(c_i, c_j) = 1, if\, c_i = c_j; \delta(c_i, c_j) = 0, if\, c_i \neq c_j$$

图的社区检测：Louvain算法&标签传播（LPA）

- 计算右图中以下划分的Modularity分值
- **{{A,B,C,D},** {E,F,G}}



n=7个节点
m=11条边

$$Q = \frac{1}{2m}\sum_{i,j}[A_{ij} - \frac{k_i k_j}{2m}]\delta(c_i, c_j)$$

$$\delta(c_i, c_j) = 1, if\, c_i = c_j; \delta(c_i, c_j) = 0, if\, c_i \neq c_j$$

- **{A,B,C,D}**
- AA 0-3*3/22, AB 1-3*4/22, AC 1-3*3/22, AD 1-3*4/22
- BB 0-4*4/22, BA 1-4*3/22, BC 1-4*3/22, BD 1-4*4/22
- CC 0-3*3/22, CA 1-3*3/22, CB 1-3*4/22, CD 1-3*4/22
- DD 0-4*4/22, DA 1-4*3/22, DB 1-4*4/22, DC 1-4*3/22

- **{E,F,G}**
- EE 0-3*3/22, EF 1- 3*2/22, EG 1-3*3/22
- FF 0-2*2/22, FE 1-2*3/22, FG 1-2*3/22
- GG 0-3*3/22, GE 1-3*3/22, GF 1-3*2/22

- **{A,B,C,D}**
- 12 - 12/22 - 9/22 - 12/22
- -12/22 - 12/22 - 16/22
- -9/22 -12/22 -12/22
- -12/22 - 16/22 - 12/22 - 50/22
- =12 - 146/22 - 50/22= 68/22

- **{E,F,G}**
- 6 -6/22 – 9/22 – 6/22 – 6/22 – 9/22 – 6/22 -22/22=
- 6 – 42/22 -22/22 = 68/22

- {A,B,C,D} 68/22
- {E,F,G} 68/22
- Sum = 136/22
- Q = 136/22/22
- =136/484

图的社区检测：Louvain算法&标签传播（LPA）

- 数据科学的必备能力之一：优化思维

maximize Objective　　or　　minimize Loss

图的社区检测：Louvain算法&标签传播（LPA）

图的社区检测：Louvain算法&标签传播（LPA）

- Louvain算法大流程
- 算法的基本思想
  - 通过贪心法最大化Modularity

- 算法的优点
  - 快：时间复杂性$O(n \log n)$
  - 好：在很多真实网络上能够得到较高的Modularity
  - 支持边上有权重的图
  - 提供层次化的划分

图的社区检测：Louvain算法&标签传播（LPA）

- Louvain算法大流程
  - （1）刚开始的时候，所有的顶点都是一个小小的类簇 – `init`
  - （2）Phase 1：以局部方式，优化模块度函数，将每个顶点归到“最好”的类簇中，直到所有的顶点所属的类簇不再变化为止 – `one_level`
  - （3）Phase 2：把一个类簇中的所有顶点聚集抽象为一个顶点，重建一个网络，其中的每个顶点对应一个社区 – `induced_graph`
  - （4）看抽象以后的网络图，是否还有优化的可能性，如果有，则迭代执行上述(2)、(3)步骤。

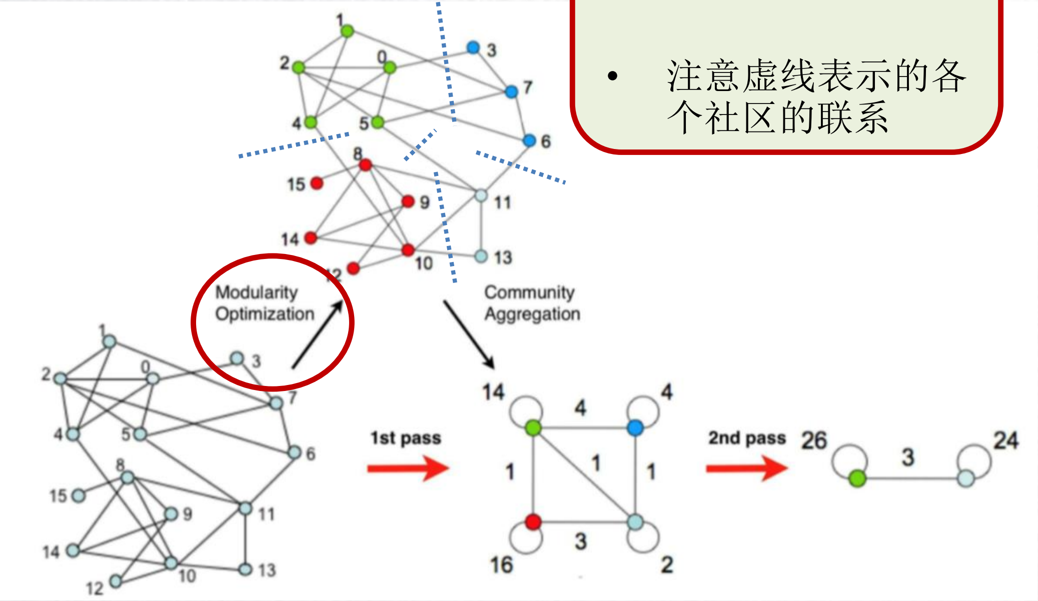
图的社区检测：Louvain算法&标签传播（LPA）

- Louvain算法大流程
- Phase 1：以局部方式，优化模块度函数，将每个顶点归到“最好”的类簇中，直到所有的顶点所属的类簇不再变化为止 – **one_level**
  - 计算将节点i合并到邻居j所在社区的modularity增益ΔQ
  - 将节点i合并到能够产生最大增益ΔQ的节点j的社区中
  - 循环执行上述步骤，直到合并操作不再产生modularity的增益

如何计算modularity增益ΔQ，在后续展开；这里关注大流程

图的社区检测：Louvain算法&标签传播（LPA）

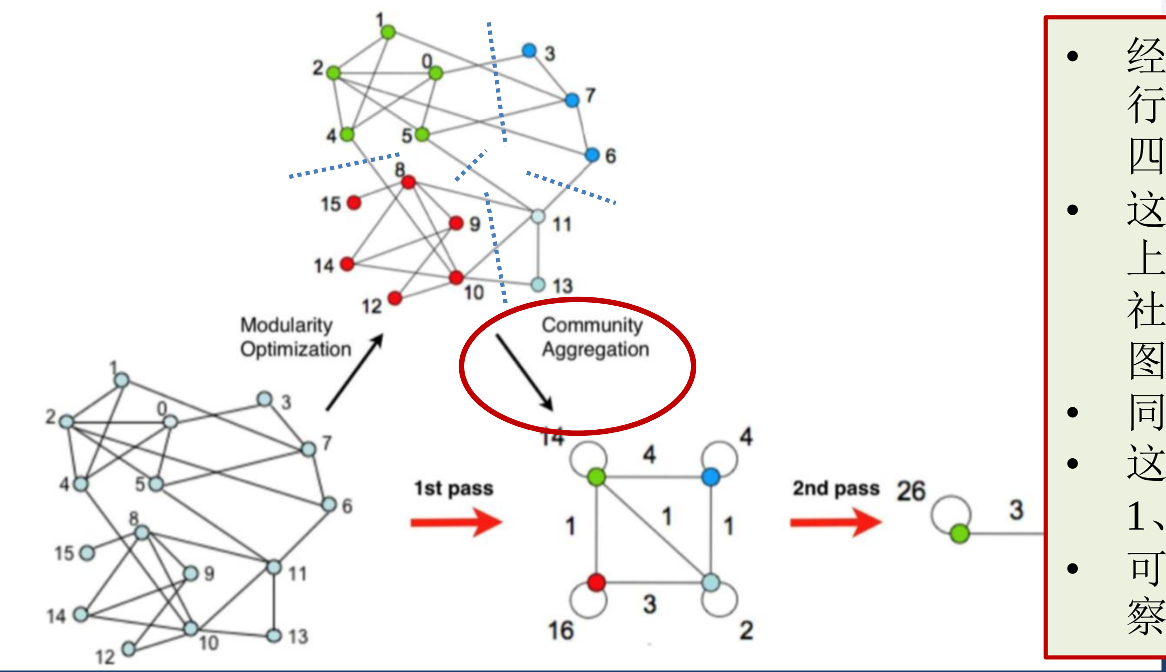
- Louvain算法大流程
- Phase 1：以局部方式，优化模块度函数，将每个顶点归到“最好”的类簇中，直到所有的顶点所属的类簇不再变化为止 – **one_level**
  - 计算将节点i合并到邻居j所在社区的modularity增益ΔQ
  - 将节点i合并到能够产生最大增益ΔQ的节点j的社区中
  - 循环执行上述步骤，直到合并操作不再产生modularity的增益



- 目前分成4个子社区，即红色、绿色、浅蓝、蓝色
- 注意虚线表示的各个社区的联系

图的社区检测：Louvain算法&标签传播（LPA）

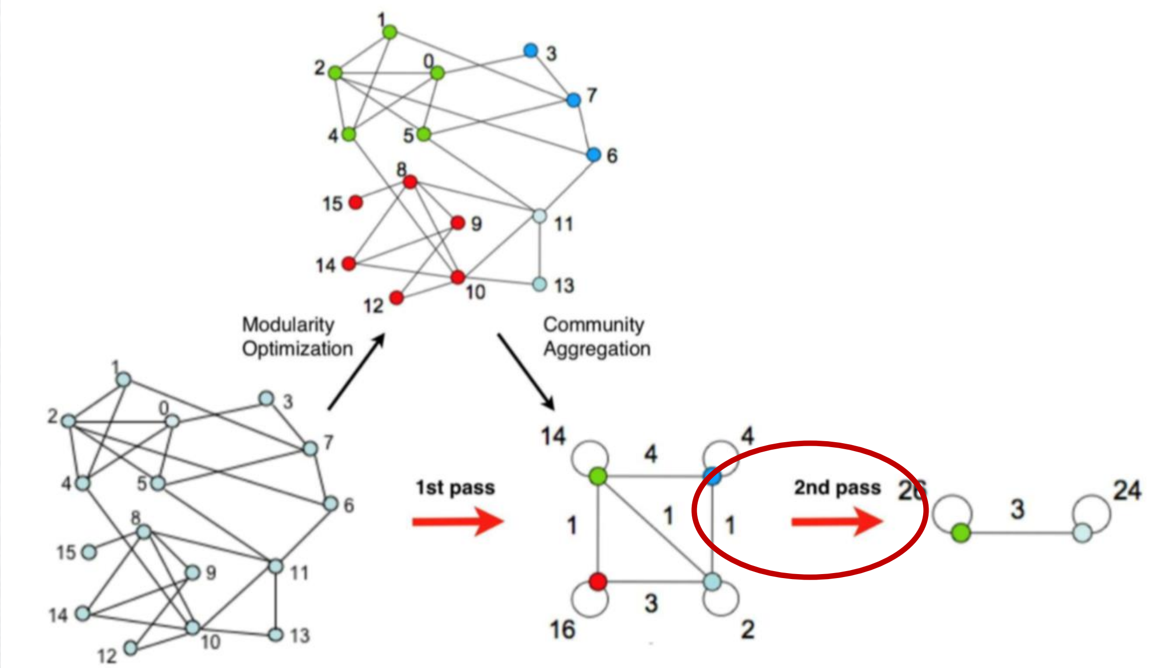
- Louvain算法大流程
  - Phase 2：把一个类簇中的所有顶点聚集抽象为一个顶点，重建一个网络，其中的每个顶点对应一个社区 – **induced_graph**



- 经过第一阶段的模块度优化后，进行折叠，4个社区各折叠为1个顶点，四个顶点的标识为14、4、16、2等
- 这些标识是如何计算的呢？子图左上角顶点的标识为14，表示第一个社区内部的连接数为7，由于是无向图，所以是双向连接，14=7×2
- 同理4=2×2, 16=8×2, 2=1×2
- 这些折叠过的顶点的连线的标识为4、1、1、1、3，表示社区间的连接数，
- 可以在图上沿着（蓝色）虚线来观察和验证

图的社区检测：Louvain算法&标签传播（LPA）

- Louvain算法大流程
  - 下一趟迭代，仍然包含（1）模块度优化、（2）社区聚集两个阶段



图的社区检测：Louvain算法&标签传播（LPA）

图的社区检测：Louvain算法&标签传播（LPA）

- Louvain算法的Phase 1
  - 以局部方式，优化模块度函数，将每个顶点归到“最好”的类簇中，直到所有的顶点所属的类簇不再变化为止 – `one_level`
  - 如何计算将i合并到社区C中的modularity增益ΔQ？
  - 需要从模块度计算公式做一些推导

$$Q = \frac{1}{2m}\sum_{ij}\left[A_{ij} - \frac{k_i k_j}{2m}\right]\delta(c_i, c_j)$$

$$\delta(u,v) = \begin{cases} 1 when\ u==v \\ 0\ else \end{cases}$$

图的社区检测：Louvain算法&标签传播（LPA）

- Louvain算法的Phase 1
  - 以局部方式，优化模块度函数，将每个顶点归到“最好”的类簇中，直到所有的顶点所属的类簇不再变化为止 – **one_level**
  - 如何计算将i合并到社区C中的modularity增益ΔQ？
  - 需要从模块度计算公式做一些推导

$$Q = \frac{1}{2m}\sum_{ij}\left[A_{ij} - \frac{k_i k_j}{2m}\right]\delta(c_i, c_j)$$

$$\delta(u,v) = \begin{cases}1 & when\ u==v\\ 0 & else\end{cases}$$

$$Q = \frac{1}{2m}\sum_{i,j}[A_{ij} - \frac{k_i k_j}{2m}]\delta(c_i, c_j)$$

$$= \frac{1}{2m}[\sum_{i,j}A_{ij} - \frac{\sum_i k_i \sum_j k_j}{2m}]\delta(c_i, c_j)$$

$$= \frac{1}{2m}\sum_c[\Sigma in - \frac{(\Sigma tot)^2}{2m}]$$

$$Q = \sum_c[\frac{\Sigma in}{2m} - (\frac{\Sigma tot}{2m})^2]$$

验算一下
(2+3+3)(2+3+3)
=8*8=64
=4+6+6
+6+9+9
+6+9+9

- $\sum in$为社区c内节点之间的边的权重之
- $\sum tot$表示社区c内节点所有的边的权重之和

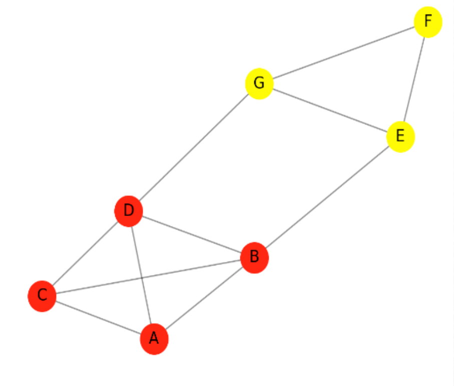
π

图的社区检测：Louvain算法&标签传播（LPA）

- Louvain算法的Phase 1
  - 以局部方式，优化模块度函数，将每个顶点归到“最好”的类簇中，直到所有的顶点所属的类簇不再变化为止 – **one_level**
  - 如何计算将i合并到社区C中的modularity增益ΔQ?

$$Q = \sum_c \left[\frac{\Sigma in}{2m} - \left(\frac{\Sigma tot}{2m}\right)^2\right]$$

- $\sum in$为社区c内节点之间的边的权重之
- $\sum tot$表示社区c内节点所有的边的权重之和



- **{A,B,C,D}**
- 12/22 – (14*14)/(22*22)
- =264/484 – 196/484
- =68/484

- **{E,F,G}**
- 6/22 – (8*8)/(22*22)
- =132/484 – 64/484
- =68/484

- Sum = 136/484

- Q = 208/22/22
- = 136/484

比对一下前面另一种计算结果?
一样的

图的社区检测：Louvain算法&标签传播（LPA）

- 从Q到ΔQ
  - 把节点i分配到邻居节点j所在的社区c时
  - 模块度的变化量为：

$$Q=\sum_c\left[\frac{\Sigma in}{2m}-\left(\frac{\Sigma tot}{2m}\right)^2\right]$$

  - $$\Delta Q=\underbrace{\left[\frac{\sum in+k_{i,in}}{2m}-\left(\frac{\sum tot+k_i}{2m}\right)^2\right]}-\underbrace{\left[\frac{\sum in}{2m}-\left(\frac{\sum tot}{2m}\right)^2+\frac{0}{2m}-\left(\frac{k_i}{2m}\right)^2\right]}$$

- 这个公式的前面一部分，表示把节点i加入社区c之后的c的模块度；后一部分，是节点i作为一个独立社区的模块度和社区c本身的模块度
- 节点i移动前的模块度为$\frac{\sum in}{2m}-\left(\frac{\sum tot}{2m}\right)^2+\frac{0}{2m}-\left(\frac{k_i}{2m}\right)^2$，移动后的模块度为$\frac{\sum in+k_{i,in}}{2m}-\left(\frac{\sum tot+k_i}{2m}\right)^2$

- 式中，$\sum in$为社区c内节点之间的边的权重之和
- $k_{i,in}$表示节点i与社区c内节点的边的权重之和
- k_i表示节点i与所有节点的边的权重之和
- $\sum tot$表示社区c内节点所有的边的权重之和

图的社区检测：Louvain算法&标签传播（LPA）

- 该公式式展开后，很多项互相抵消(请读者参考相关资料)，最后得到简化形式如下：

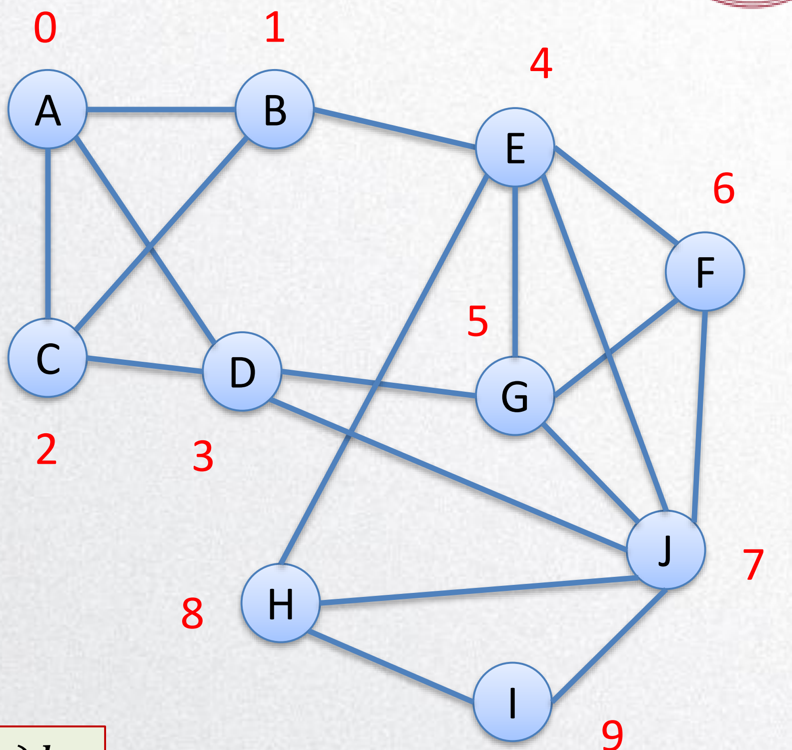
$$\Delta Q = \left[\frac{k_{i,in}}{2m} - \frac{(\sum tot)k_i}{2m^2}\right] = \frac{1}{2m}[k_{i,in} - \frac{(\sum tot)k_i}{m}]$$

  - 把公共系数$\frac{1}{2m}$提取出来，在Louvain算法的具体实现时
  - 计算 $\boxed{k_{i,in} - \frac{(\sum tot)k_i}{m}}$ 即可
  - 特别注意，把节点i从社区D出来，放到社区C
  - 总的模块度变化量，由两部分构成
    - $\Delta Q(i \to C)$，即把i加入社区C
    - $\Delta Q(D \to i)$，即把i从社区D拿掉
      - 注意把节点从社区拿掉，就是把节点加入社区的反动作

图的社区检测：Louvain算法&标签传播（LPA）

图的社区检测：Louvain算法&标签传播（LPA）

- Louvain算法的Phase 1实例
- 考虑下面的示例图作为输入
- 维护以下数据结构
  - Node → Community



A	B	C	D	E	F	G	H	I	J
0	1	2	3	4	6	5	8	9	7

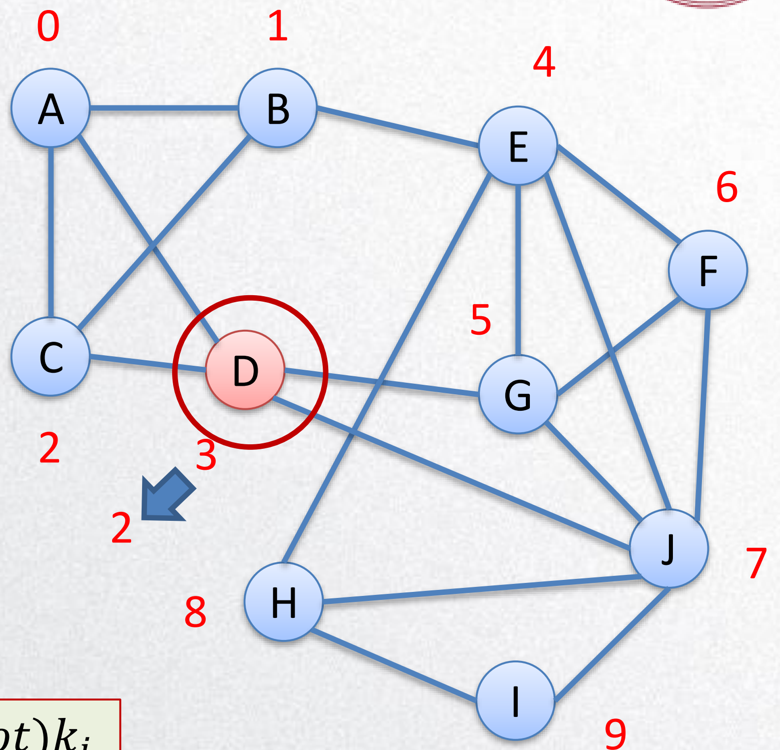
- 随机生成一个节点访问的序列
  - D, G, E, C, H, I, B, A, J, F

模块度变化公式 $k_{i,in} - \frac{(\sum tot)k_i}{m}$

图的社区检测：Louvain算法&标签传播（LPA）

- Louvain算法的Phase 1实例
- 考虑红色标注的节点
  - **D**, G, E, C, H, I, B, A, J, F
- 邻居社区：
  - {0, 2, 5, 7}
- 按随机的顺序访问邻居，计算分值
  - $\Delta Q(C_3 \rightarrow D) = -\left(0 - \frac{0*4}{18}\right) = 0$
  - $\Delta Q(D \rightarrow C_5) = 1 - \frac{4\times 4}{18} = \frac{2}{18}$
  - $\Delta Q(D \rightarrow C_2) = 1 - \frac{4\times 3}{18} = \frac{6}{18}$
  - $\Delta Q(D \rightarrow C_7) = 1 - \frac{4\times 6}{18} = -\frac{6}{18}$
  - $\Delta Q(D \rightarrow C_0) = 1 - \frac{4\times 3}{18} = \frac{6}{18}$
- **选择将D并入社区2**

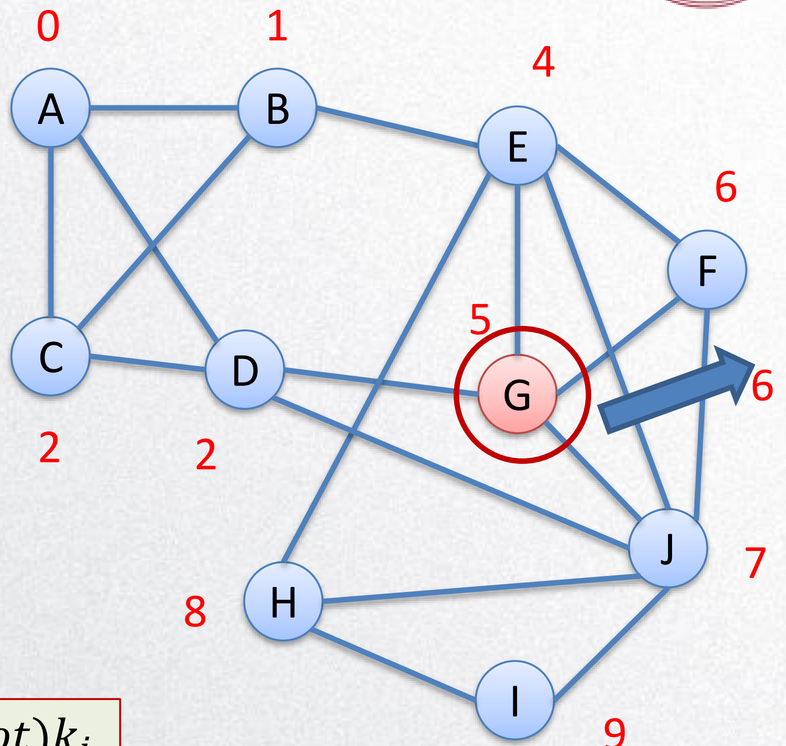
$$k_{i,in} - \frac{(\sum tot)k_i}{m}$$



图的社区检测：Louvain算法&标签传播（LPA）

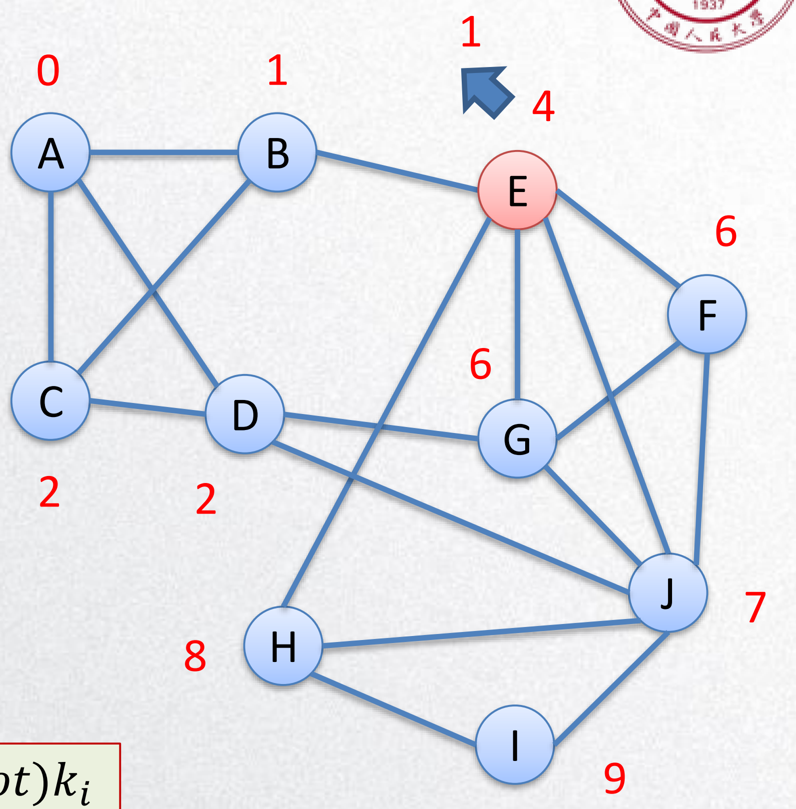
- Louvain算法的Phase 1实例
- 考虑红色标注的节点
  - D, G, E, C, H, I, B, A, J, F
- 邻居社区：
  - {2, 4, 6, 7}
- 按随机的顺序访问邻居，计算分值
  - $\Delta Q(C_5 \to \mathrm{G}) = -\left(0 - \frac{0*4}{18}\right) = 0$
  - $\Delta Q(G \to C_2) = 1 - \frac{4\times7}{18} = \frac{-10}{18}$
  - $\Delta Q(G \to C_4) = 1 - \frac{4\times5}{18} = \frac{-2}{18}$
  - $\Delta Q(G \to C_6) = 1 - \frac{4\times3}{18} = \frac{6}{18}$
  - $\Delta Q(G \to C_7) = 1 - \frac{4\times6}{18} = \frac{-6}{18}$
- **选择将G并入社区6**

$$k_{i,in} - \frac{(\sum tot)k_i}{m}$$



图的社区检测：Louvain算法&标签传播（LPA）

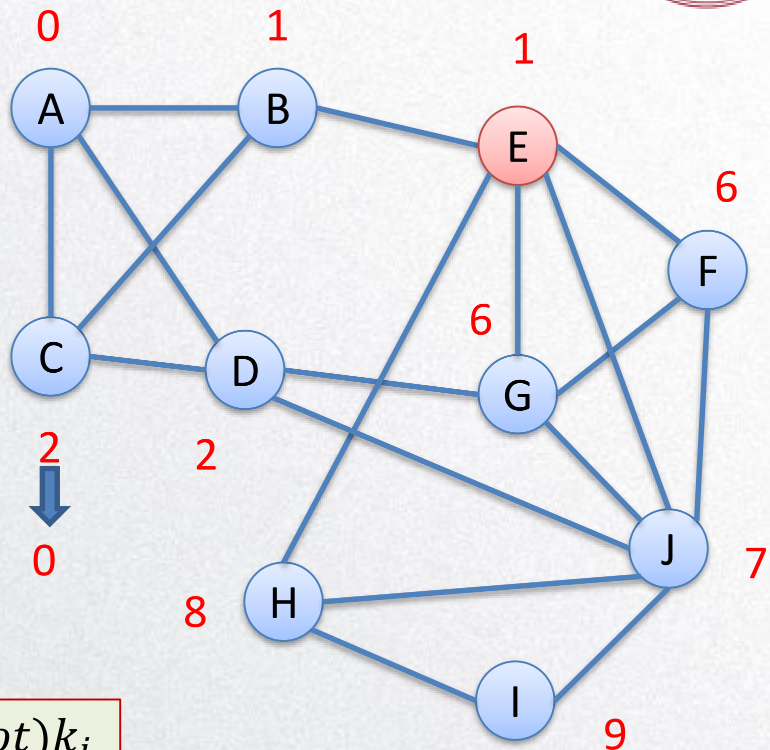
- Louvain算法的Phase 1实例
- 考虑红色标注的节点
  - D, G, E, C, H, I, B, A, J, F
- 邻居社区：
  - {1, 6, 8, 7}
- 按随机的顺序访问邻居，计算分值
  - $\Delta Q(C_4 \rightarrow E) = -\left(0 - \frac{0*5}{18}\right) = 0$
  - $\Delta Q(E \rightarrow C_1) = 1 - \frac{5\times3}{18} = \frac{3}{18}$
  - $\Delta Q(E \rightarrow C_6) = 2 - \frac{5\times7}{18} = \frac{1}{18}$
  - $\Delta Q(E \rightarrow C_7) = 1 - \frac{5\times6}{18} = \frac{-12}{18}$
  - $\Delta Q(E \rightarrow C_8) = 1 - \frac{5\times3}{18} = \frac{3}{18}$
  - **选择将E并入社区1**



$$k_{i,in} - \frac{(\sum tot)k_i}{m}$$

图的社区检测：Louvain算法&标签传播（LPA）

- Louvain算法的Phase 1实例
- 考虑红色标注的节点
  - D, G, **E**, C, H, I, B, A, J, F
- 邻居社区：
  - {0, 1, 2}
- 按随机的顺序访问邻居，计算分值
  - $\Delta Q(C_2 \to C) = -\left(1 - \frac{3*4}{18}\right) = -\frac{6}{18}$
  - $\Delta Q(C \to C_0) = 1 - \frac{3\times3}{18} = \frac{9}{18}$
  - $\Delta Q(C \to C_1) = 1 - \frac{3\times3}{18} = \frac{9}{18}$
  - $\Delta Q(C \to C_2) = 1 - \frac{3\times4}{18} = \frac{6}{18}$
  - **选择将C并入社区0**

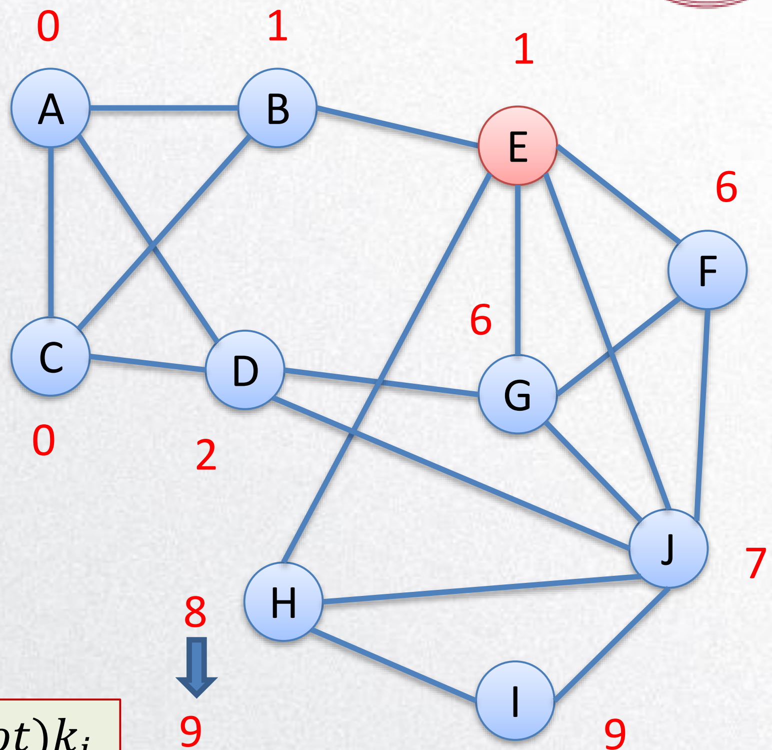


$$k_{i,in} - \frac{(\sum tot)k_i}{m}$$

图的社区检测：Louvain算法&标签传播（LPA）

- Louvain算法的Phase 1实例
- 考虑红色标注的节点
  - D, G, E, C, H, I, B, A, J, F
- 邻居社区：
  - {1, 7, 9}
- 按随机的顺序访问邻居，计算分值
  - $\Delta Q(C_8 \rightarrow H) = -\left(0 - \frac{0*3}{18}\right) = 0$
  - $\Delta Q(H \rightarrow C_1) = 1 - \frac{3\times8}{18} = -\frac{6}{18}$
  - $\Delta Q(H \rightarrow C_7) = 1 - \frac{3\times6}{18} = \frac{0}{18}$
  - $\Delta Q(H \rightarrow C_9) = 1 - \frac{3\times2}{18} = \frac{12}{18}$
  - **选择将H并入社区9**

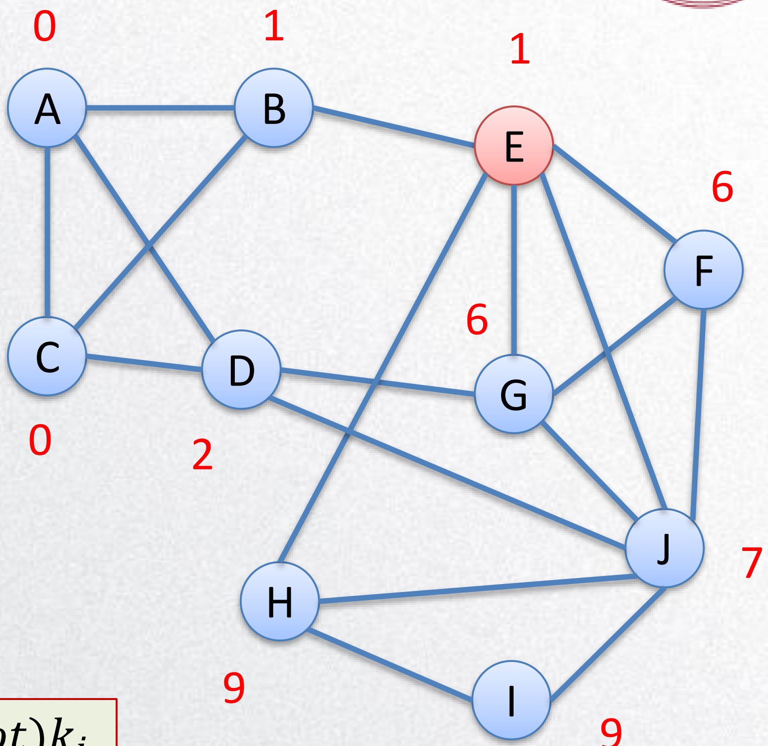
$$k_{i,in} - \frac{(\sum tot)k_i}{m}$$



图的社区检测：Louvain算法&标签传播（LPA）

- Louvain算法的Phase 1实例
- 考虑红色标注的节点
  - D, G, E, C, H, I, B, A, J, F
- 邻居社区：
  - {7， 9}
- 按随机的顺序访问邻居，计算分值
  - $\Delta Q(C_9 \rightarrow \mathrm{I}) = -\left(1 - \frac{2*3}{18}\right) = -\frac{12}{18}$
  - $\Delta Q(\mathrm{I} \rightarrow C_7) = 1 - \frac{2\times 6}{18} = \frac{6}{18}$
  - $\Delta Q(\mathrm{I} \rightarrow C_9) = 1 - \frac{2*3}{18} = \frac{12}{18}$
  - **选择将I不动**

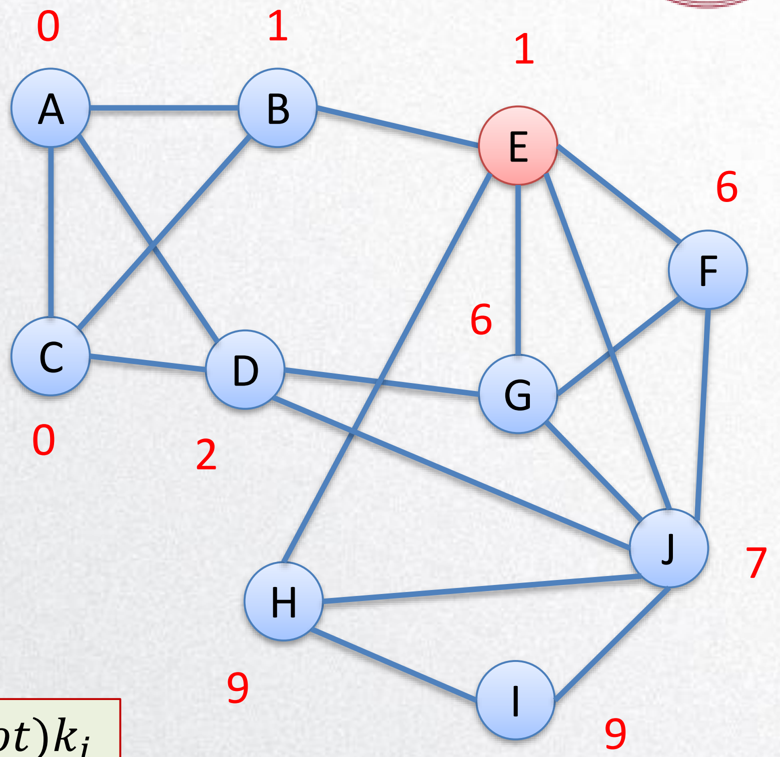
$$k_{i,in} - \frac{(\sum tot)k_i}{m}$$



图的社区检测：Louvain算法&标签传播（LPA）

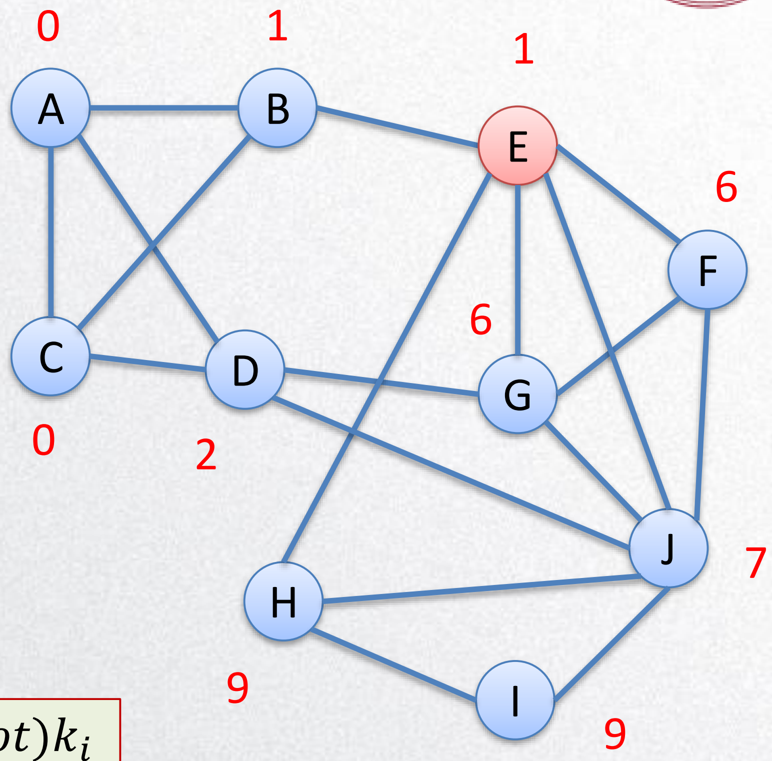
- Louvain算法的Phase 1实例
- 考虑红色标注的节点
  - D, G, E, C, H, I, B, A, J, F
- 邻居社区：
  - {0,1}
- 按随机的顺序访问邻居，计算分值
  - $\Delta Q(C_1 \rightarrow B) = -\left(1 - \frac{3*5}{18}\right) = -\frac{3}{18}$
  - $\Delta Q(B \rightarrow C_0) = 1 - \frac{3\times 6}{18} = \frac{0}{18}$
  - $\Delta Q(B \rightarrow C_1) = 1 - \frac{3*5}{18} = \frac{3}{18}$
  - **选择将B不动**

$$k_{i,in} - \frac{(\sum tot)k_i}{m}$$



图的社区检测：Louvain算法&标签传播（LPA）

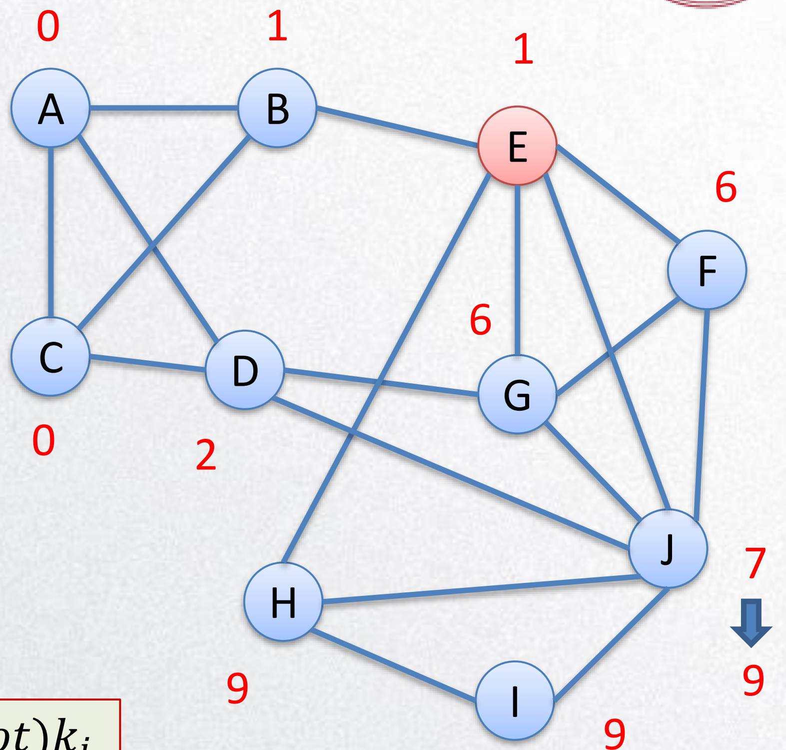
- Louvain算法的Phase 1实例
- 考虑红色标注的节点
  - D, G, E, C, H, I, B, A, J, F
- 邻居社区：
  - {0,1,2}
- 按随机的顺序访问邻居，计算分值
  - $\Delta Q(C_0 \rightarrow A) = -\left(1 - \frac{3*3}{18}\right) = -\frac{9}{18}$
  - $\Delta Q(A \rightarrow C_0) = 1 - \frac{3\times 3}{18} = \frac{9}{18}$
  - $\Delta Q(A \rightarrow C_1) = 1 - \frac{3\times 8}{18} = -\frac{6}{18}$
  - $\Delta Q(B \rightarrow C_2) = 1 - \frac{3*4}{18} = \frac{6}{18}$
  - **选择将A不动**



$$k_{i,in} - \frac{(\sum tot)k_i}{m}$$

图的社区检测：Louvain算法&标签传播（LPA）

- Louvain算法的Phase 1实例
- 考虑红色标注的节点
  - D, G, E, C, H, I, B, A, J, F
- 邻居社区：
  - {1,2,6,9}
- 按随机的顺序访问邻居，计算分值
  - $\Delta Q(C_7 \to J) = -\left(0 - \frac{6*0}{18}\right) = 0$
  - $\Delta Q(J \to C_1) = 1 - \frac{6\times 8}{18} = -\frac{30}{18}$
  - $\Delta Q(J \to C_2) = 1 - \frac{6\times 4}{18} = -\frac{6}{18}$
  - $\Delta Q(J \to C_6) = 2 - \frac{6\times 7}{18} = -\frac{6}{18}$
  - $\Delta Q(J \to C_9) = 2 - \frac{6*5}{18} = \frac{6}{18}$
  - **选择将J，加入9**

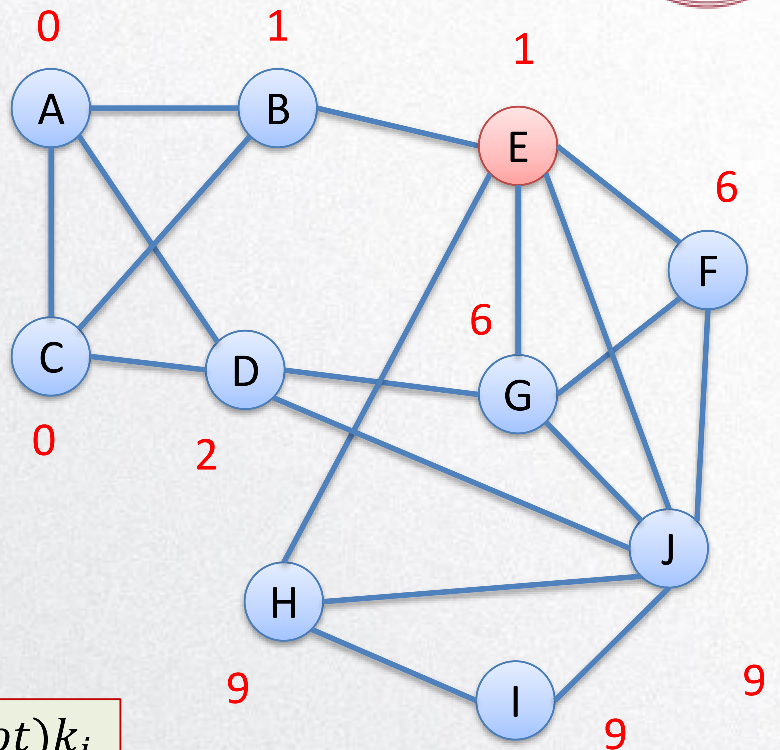


$$k_{i,in} - \frac{(\sum tot)k_i}{m}$$

图的社区检测：Louvain算法&标签传播（LPA）

- Louvain算法的Phase 1实例
- 考虑红色标注的节点
  - D, G, E, C, H, I, B, A, J, F
- 邻居社区：
  - {1，6，9}
- 按随机的顺序访问邻居，计算分值
  - $\Delta Q(C_6 \rightarrow \mathrm{F}) = -\left(1 - \frac{3*4}{18}\right) = -\frac{6}{18}$
  - $\Delta Q(\mathrm{F} \rightarrow C_1) = 1 - \frac{3\times 8}{18} = -\frac{6}{18}$
  - $\Delta Q(F \rightarrow C_6) = 1 - \frac{3\times 4}{18} = \frac{6}{18}$
  - $\Delta Q(F \rightarrow C_9) = 1 - \frac{3\times 11}{18} = -\frac{15}{18}$
  - **选择将F不动**

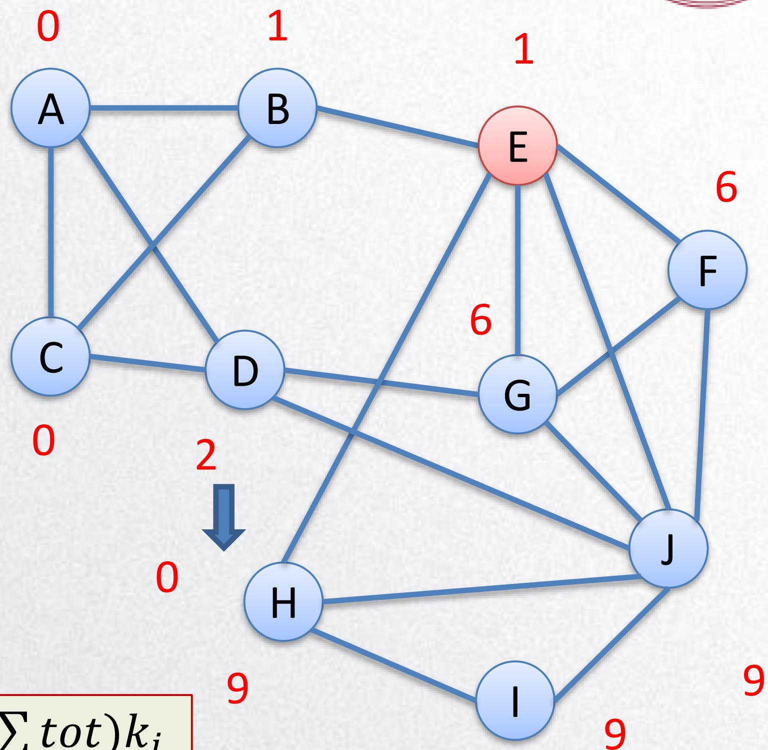
$$k_{i,in} - \frac{(\sum tot)k_i}{m}$$



图的社区检测：Louvain算法&标签传播（LPA）

图的社区检测：Louvain算法&标签传播（LPA）

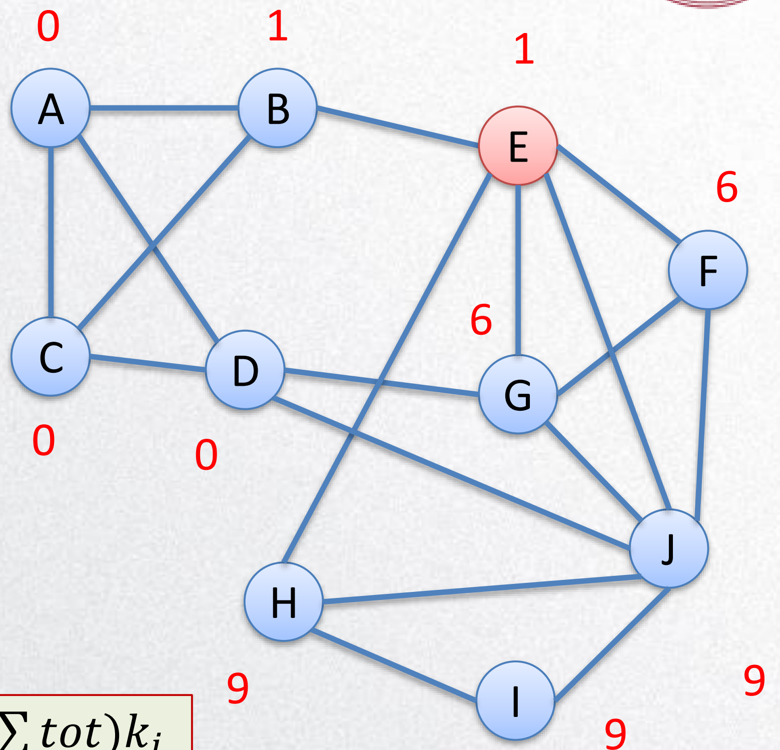
- Louvain算法的Phase 1实例
- **继续调整，看有没有改进余地**
- 考虑红色标注的节点
  - D, G, E, C, H, I, B, A, J, F
- 邻居社区：
  - {0,6,9}
- 按随机的顺序访问邻居，计算分值
  - $\Delta Q(C_2 \rightarrow \mathrm{D}) = -\left(0 - \frac{0\times4}{18}\right) = 0$
  - $\Delta Q(D \rightarrow C_0) = 2 - \frac{4\times6}{18} = \frac{12}{18}$
  - $\Delta Q(D \rightarrow C_6) = 1 - \frac{4\times7}{18} = -\frac{10}{18}$
  - $\Delta Q(D \rightarrow C_9) = 1 - \frac{4\times11}{18} = -\frac{26}{18}$
  - **选择将D，加入0**



$$k_{i,in} - \frac{(\sum tot)k_i}{m}$$

图的社区检测：Louvain算法&标签传播（LPA）

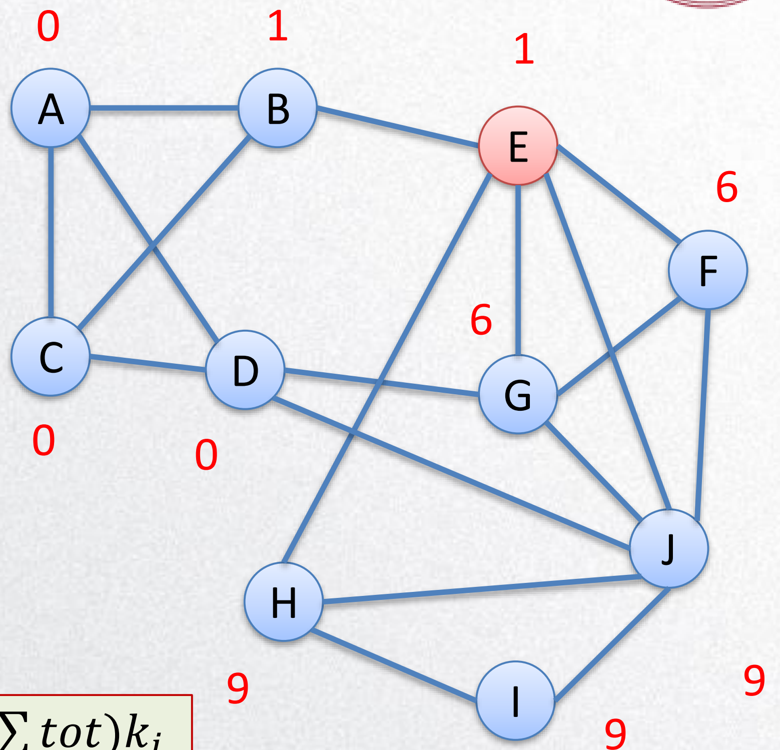
- Louvain算法的Phase 1实例
- **继续调整，看有没有改进余地**
- 考虑红色标注的节点
  - D, G, E, C, H, I, B, A, J, F
- 邻居社区：
  - {0，1,6,9}
- 按随机的顺序访问邻居，计算分值
  - $\Delta Q(C_6 \rightarrow G) = -\left(1 - \frac{4\times 3}{18}\right) = -\frac{6}{18}$
  - $\Delta Q(G \rightarrow C_0) = 1 - \frac{4\times 10}{18} = -\frac{22}{18}$
  - $\Delta Q(G \rightarrow C_1) = 1 - \frac{4\times 8}{18} = -\frac{14}{18}$
  - $\Delta Q(G \rightarrow C_6) = 1 - \frac{4\times 3}{18} = \frac{6}{18}$
  - $\Delta Q(G \rightarrow C_9) = 1 - \frac{4\times 11}{18} = -\frac{26}{18}$
  - **选择将G不动**



$$k_{i,in} - \frac{(\sum tot)k_i}{m}$$

图的社区检测：Louvain算法&标签传播（LPA）

- Louvain算法的Phase 1实例
- **继续调整，看有没有改进余地**
- 考虑红色标注的节点
  - D, G, E, C, H, I, B, A, J, F
- 邻居社区：
  - {1,6,9}
- 按随机的顺序访问邻居，计算分值
  - $\Delta Q(C_1 \rightarrow E) = -\left(1 - \frac{5\times 3}{18}\right) = -\frac{3}{18}$
  - $\Delta Q(E \rightarrow C_1) = 1 - \frac{5\times 3}{18} = \frac{3}{18}$
  - $\Delta Q(E \rightarrow C_6) = 2 - \frac{5\times 7}{18} = \frac{1}{18}$
  - $\Delta Q(E \rightarrow C_9) = 2 - \frac{5\times 11}{18} = -\frac{19}{18}$
  - **选择将E保留**

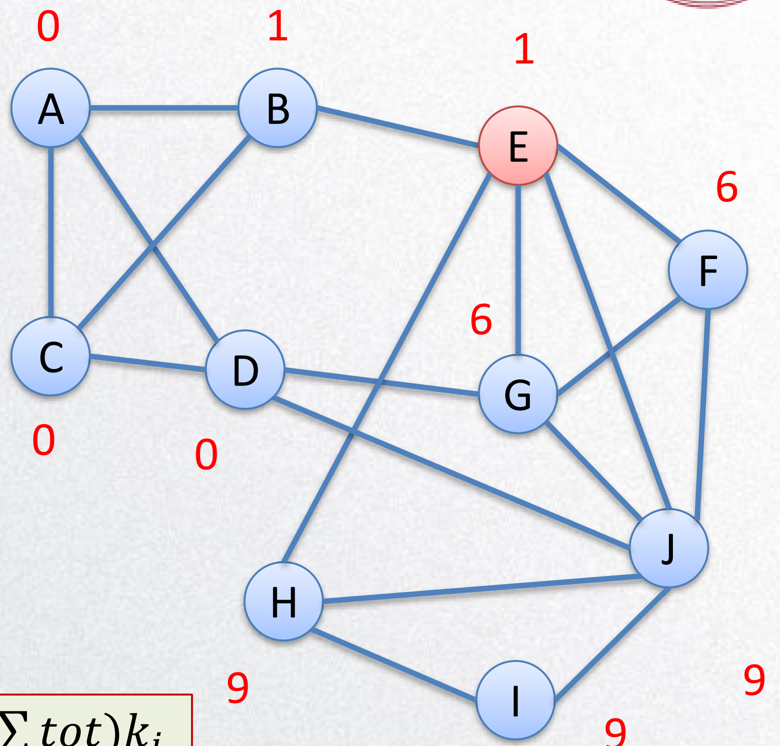


$$k_{i,in} - \frac{(\sum tot)k_i}{m}$$

图的社区检测：Louvain算法&标签传播（LPA）

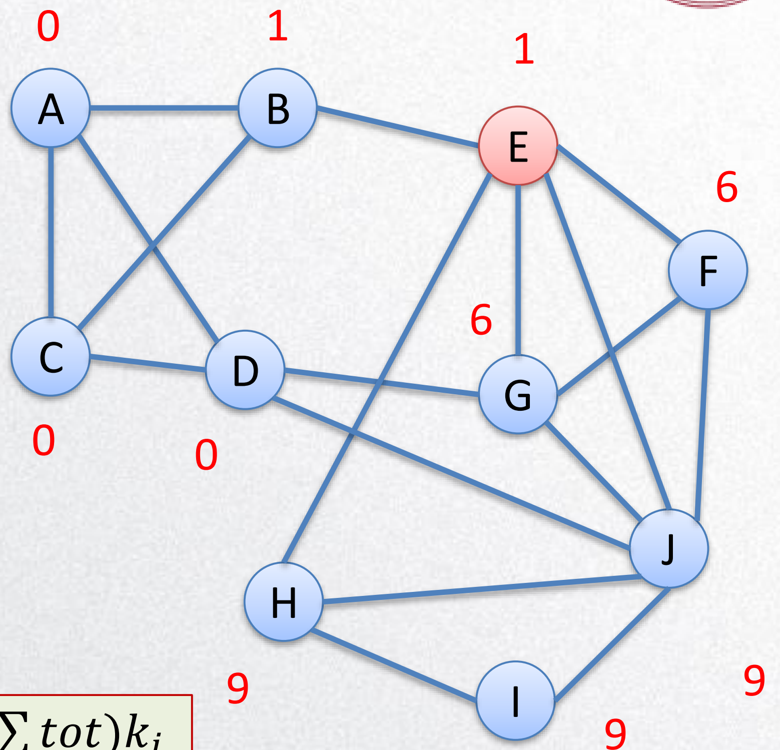
- Louvain算法的Phase 1实例
- **继续调整，看有没有改进余地**
- 考虑红色标注的节点
  - D, G, E, C, H, I, B, A, J, F
- 邻居社区：
  - {0，1}
- 按随机的顺序访问邻居，计算分值
  - $\Delta Q(C_0 \rightarrow \mathrm{C}) = -\left(2 - \frac{3*7}{18}\right) = -\frac{15}{18}$
  - $\Delta Q(\mathrm{C} \rightarrow C_0) = 2 - \frac{3\times 7}{18} = \frac{15}{18}$
  - $\Delta Q(\mathrm{C} \rightarrow C_1) = 1 - \frac{3\times 3}{18} = \frac{9}{18}$
  - **选择将C不动**

$$k_{i,in} - \frac{(\sum tot)k_i}{m}$$



图的社区检测：Louvain算法&标签传播（LPA）

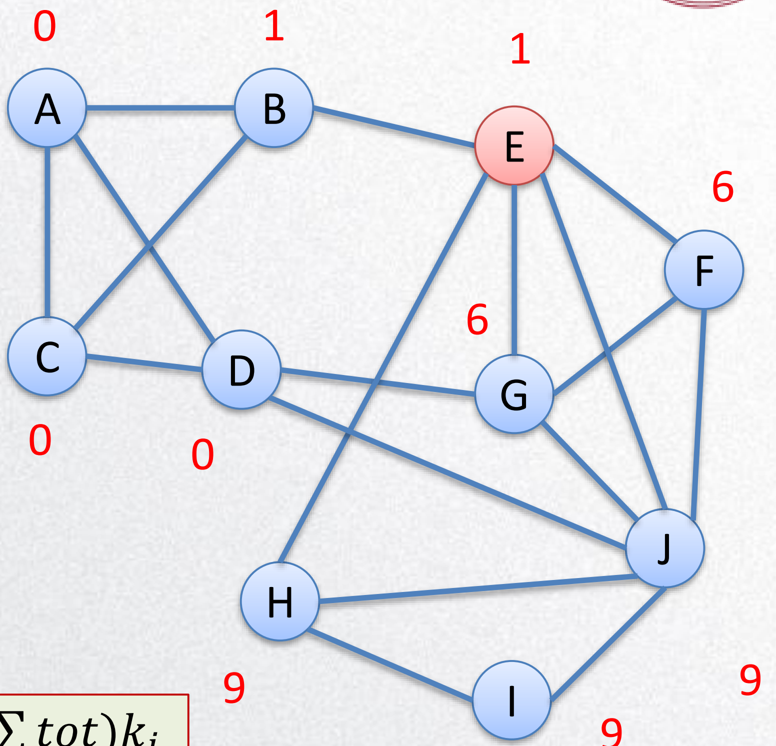
- Louvain算法的Phase 1实例
- **继续调整，看有没有改进余地**
- 考虑红色标注的节点
  - D, G, E, C, H, I, B, A, J, F
- 邻居社区：
  - {1， 9}
- 按随机的顺序访问邻居，计算分值
  - $\Delta Q(C_9 \rightarrow H) = -\left(2 - \frac{3*8}{18}\right) = -\frac{12}{18}$
  - $\Delta Q(H \rightarrow C_1) = 1 - \frac{3\times 8}{18} = \frac{-6}{18}$
  - $\Delta Q(H \rightarrow C_9) = 2 - \frac{3\times 8}{18} = \frac{12}{18}$
  - **选择将H不动**



$$k_{i,in} - \frac{(\sum tot)k_i}{m}$$

图的社区检测：Louvain算法&标签传播（LPA）

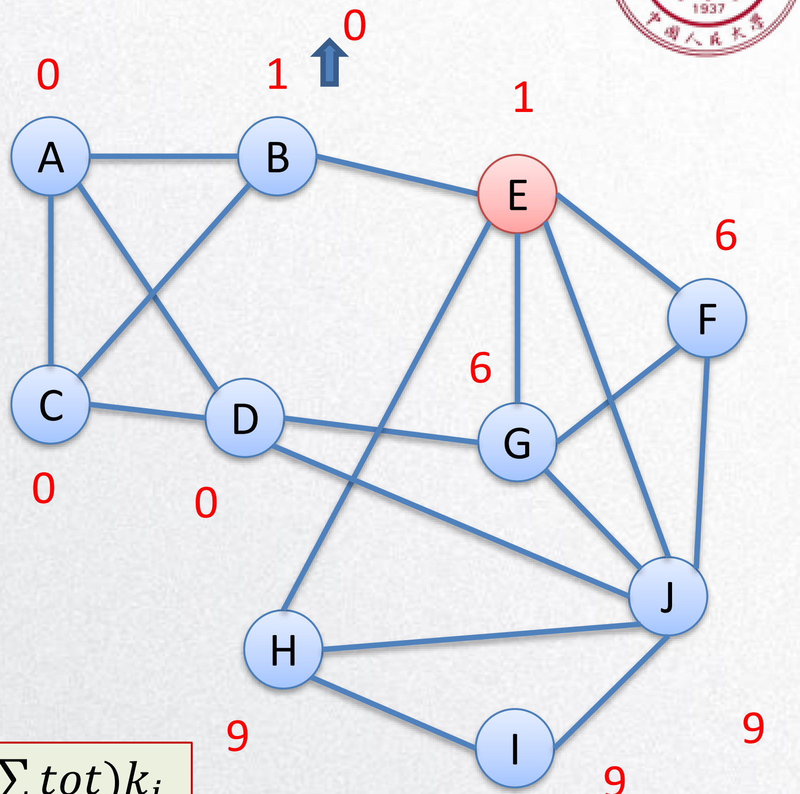
- Louvain算法的Phase 1实例
- **继续调整，看有没有改进余地**
- 考虑红色标注的节点
  - D, G, E, C, H, I, B, A, J, F
- 邻居社区：
  - {9}
- 按随机的顺序访问邻居，计算分值
  - $\Delta Q(C_9 \rightarrow I) = -\left(2 - \frac{2*9}{18}\right) = -2$
  - $\Delta Q(I \rightarrow C_9) = 2 - \frac{2\times 9}{18} = 2$
  - **选择将I不动**



$$k_{i,in} - \frac{(\sum tot)k_i}{m}$$

图的社区检测：Louvain算法&标签传播（LPA）

- Louvain算法的Phase 1实例
- **继续调整，看有没有改进余地**
- 考虑红色标注的节点
  - D, G, E, C, H, I, B, A, J, F
- 邻居社区：
  - {0,1}
- 按随机的顺序访问邻居，计算分值
  - $\Delta Q(C_1 \rightarrow B) = -\left(1 - \frac{3*5}{18}\right) = -\frac{3}{18}$
  - $\Delta Q(B \rightarrow C_0) = 2 - \frac{3\times 10}{18} = \frac{6}{18}$
  - $\Delta Q(B \rightarrow C_1) = 1 - \frac{3*5}{18} = \frac{3}{18}$
  - **选择将B加入0**

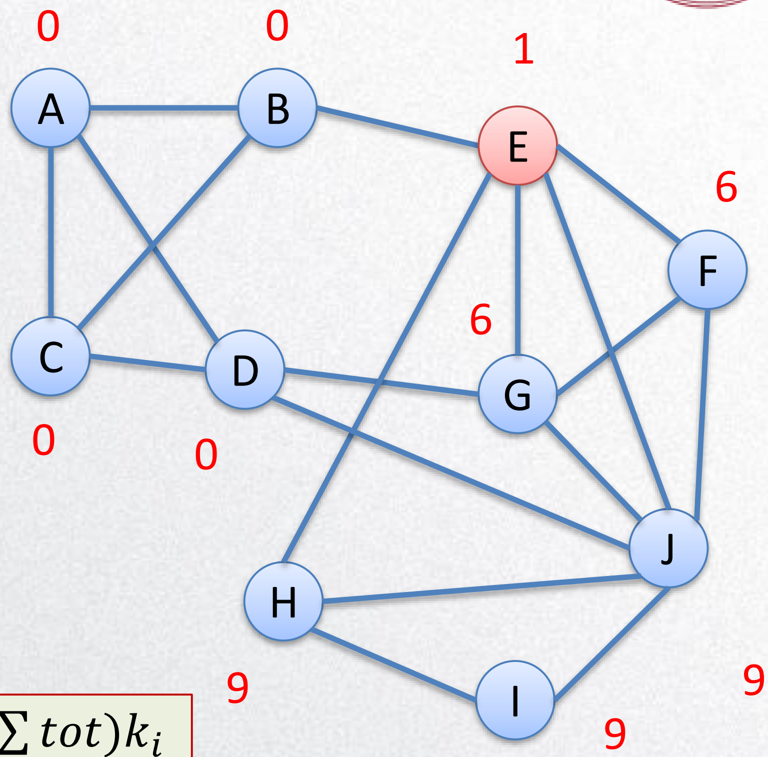


$$k_{i,in} - \frac{(\sum tot)k_i}{m}$$

图的社区检测：Louvain算法&标签传播（LPA）

- Louvain算法的Phase 1实例
- **继续调整，看有没有改进余地**
- 考虑红色标注的节点
  - D, G, E, C, H, I, B, A, J, F
- 邻居社区：
  - {0}
- 按随机的顺序访问邻居，计算分值
  - $\Delta Q(C_0 \rightarrow A) = -\left(3 - \frac{3*10}{18}\right) = -\frac{24}{18}$
  - $\Delta Q(A \rightarrow C_0) = 3 - \frac{3*10}{18} = \frac{24}{18}$

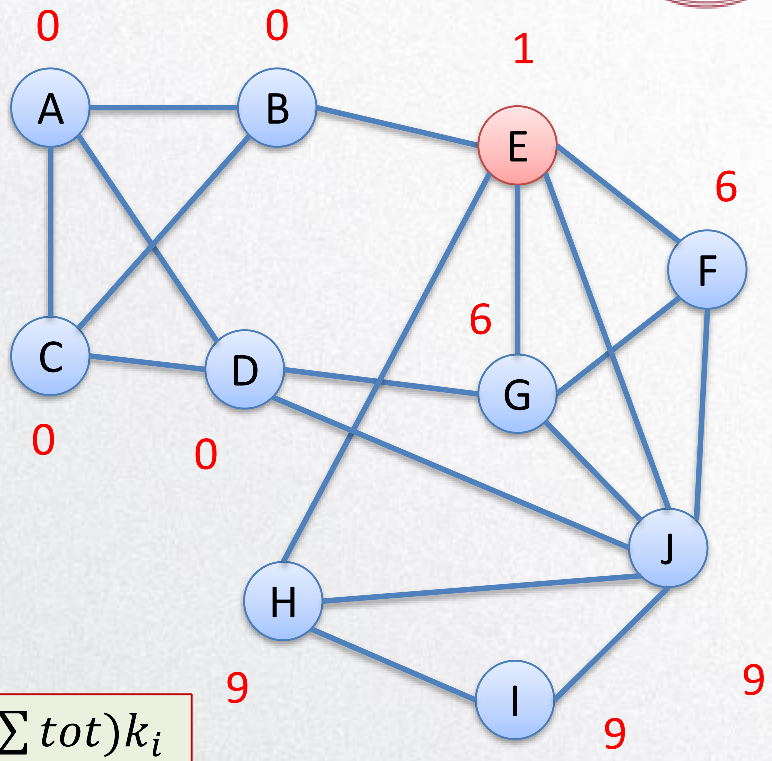
  - **选择将A不动**



$$k_{i,in} - \frac{(\sum tot)k_i}{m}$$

图的社区检测：Louvain算法&标签传播（LPA）

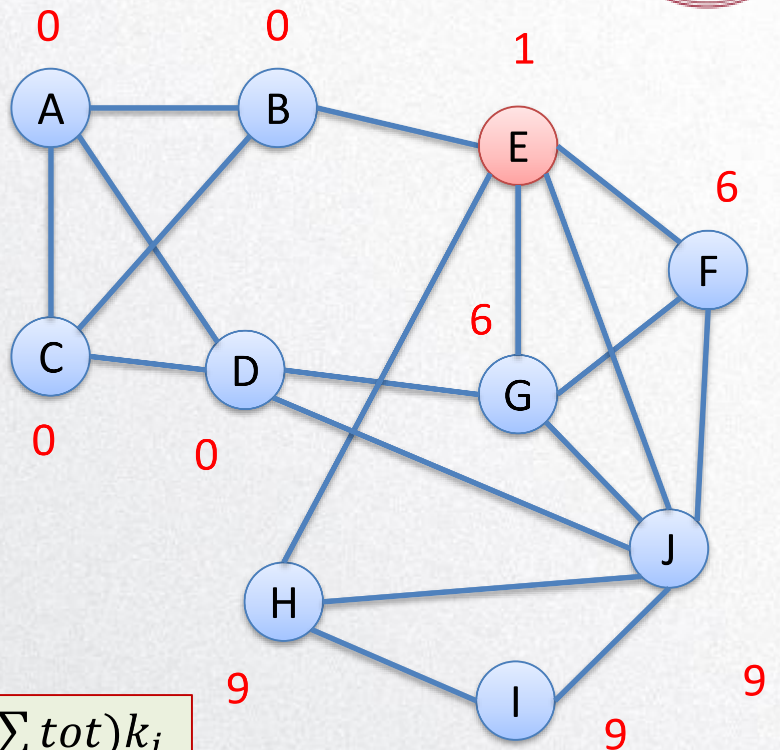
- Louvain算法的Phase 1实例
- **继续调整，看有没有改进余地**
- 考虑红色标注的节点
  - D, G, E, C, H, I, B, A, J, F
- 邻居社区：
  - {0,1，6，9}
- 按随机的顺序访问邻居，计算分值
  - $\Delta Q(C_9 \rightarrow J) = -\left(2 - \frac{6*5}{18}\right) = -\frac{6}{18}$
  - $\Delta Q(J \rightarrow C_0) = 1 - \frac{6\times13}{18} = -\frac{60}{18}$
  - $\Delta Q(J \rightarrow C_1) = 1 - \frac{6\times5}{18} = -\frac{12}{18}$
  - $\Delta Q(J \rightarrow C_6) = 2 - \frac{6\times7}{18} = -\frac{6}{18}$
  - $\Delta Q(J \rightarrow C_9) = 2 - \frac{6*5}{18} = \frac{6}{18}$
  - **选择将J不动**



$$k_{i,in} - \frac{(\sum tot)k_i}{m}$$

图的社区检测：Louvain算法&标签传播（LPA）

- Louvain算法的Phase 1实例
- **继续调整，看有没有改进余地**
- 考虑红色标注的节点
  - D, G, E, C, H, I, B, A, J, F
- 邻居社区：
  - {1，6，9}
- 按随机的顺序访问邻居，计算分值
  - $\Delta Q(C_6 \to \mathrm{F}) = -\left(1 - \frac{3*4}{18}\right) = -\frac{6}{18}$
  - $\Delta Q(\mathrm{F} \to C_1) = 1 - \frac{3\times 5}{18} = \frac{3}{18}$
  - $\Delta Q(F \to C_6) = 1 - \frac{3\times 4}{18} = \frac{6}{18}$
  - $\Delta Q(F \to C_9) = 1 - \frac{3\times 11}{18} = -\frac{15}{18}$
  - **选择将F不动**



$$k_{i,in} - \frac{(\sum tot)k_i}{m}$$

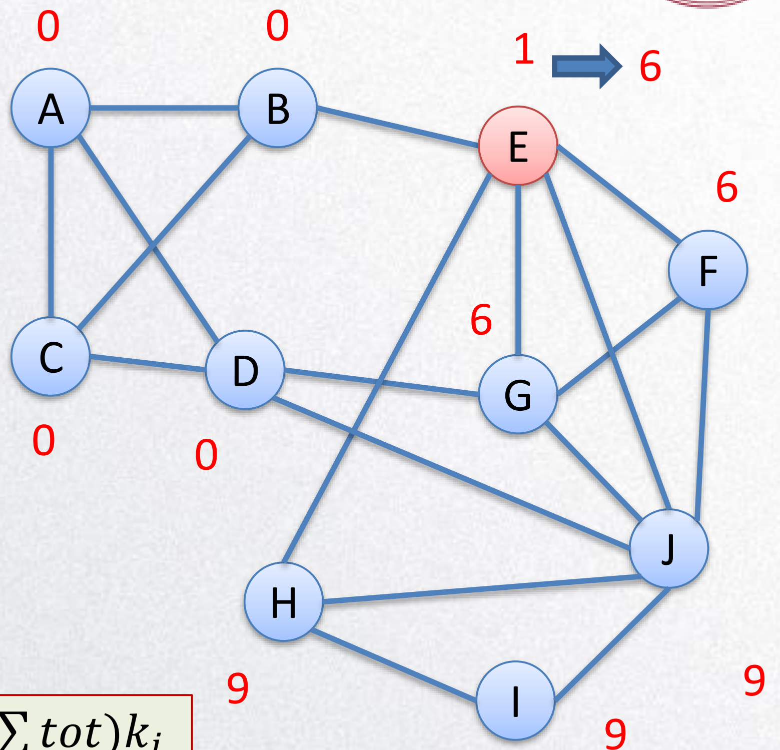
图的社区检测：Louvain算法&标签传播（LPA）

图的社区检测：Louvain算法&标签传播（LPA）

- Louvain算法的Phase 1实例
- **继续调整，看有没有改进余地**
- 考虑红色标注的节点
  - D, G, $\boxed{\text{E,}}$ C, H, I, B, A, J, F
- 邻居社区：
  - {0， 6， 9}
- 按随机的顺序访问邻居，计算分值
  - $\Delta Q(C_1 \to \mathrm{E}) = -\left(0 - \frac{5*0}{18}\right) = 0$
  - $\Delta Q(\mathrm{E} \to C_0) = 1 - \frac{5\times 13}{18} = -\frac{47}{18}$
  - $\Delta Q(\mathrm{E} \to C_6) = 2 - \frac{5\times 7}{18} = \frac{1}{18}$
  - $\Delta Q(\mathrm{E} \to C_9) = 2 - \frac{5\times 11}{18} = -\frac{19}{18}$

  - **选择将E，加入6**

仅仅展示E



$$k_{i,in} - \frac{(\sum tot)k_i}{m}$$

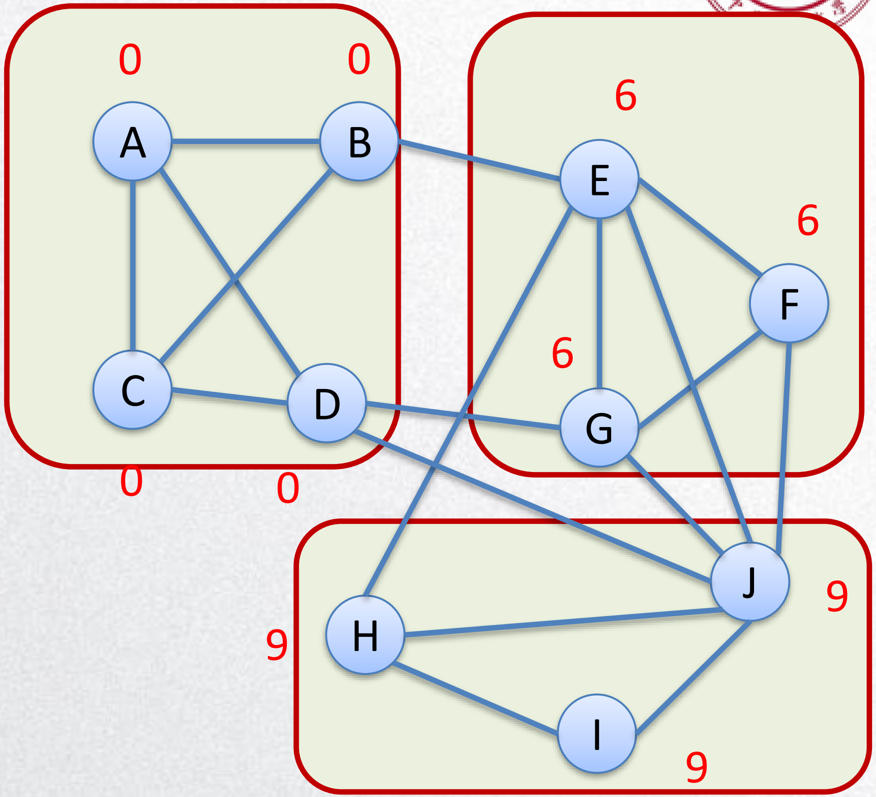
图的社区检测：Louvain算法&标签传播（LPA）

- 这一轮迭代终止
- 得到社区划分结果

{'A': 0, 'B': 0, 'C': 0, 'D': 0,

'E': 6, 'G': 6, 'F': 6,

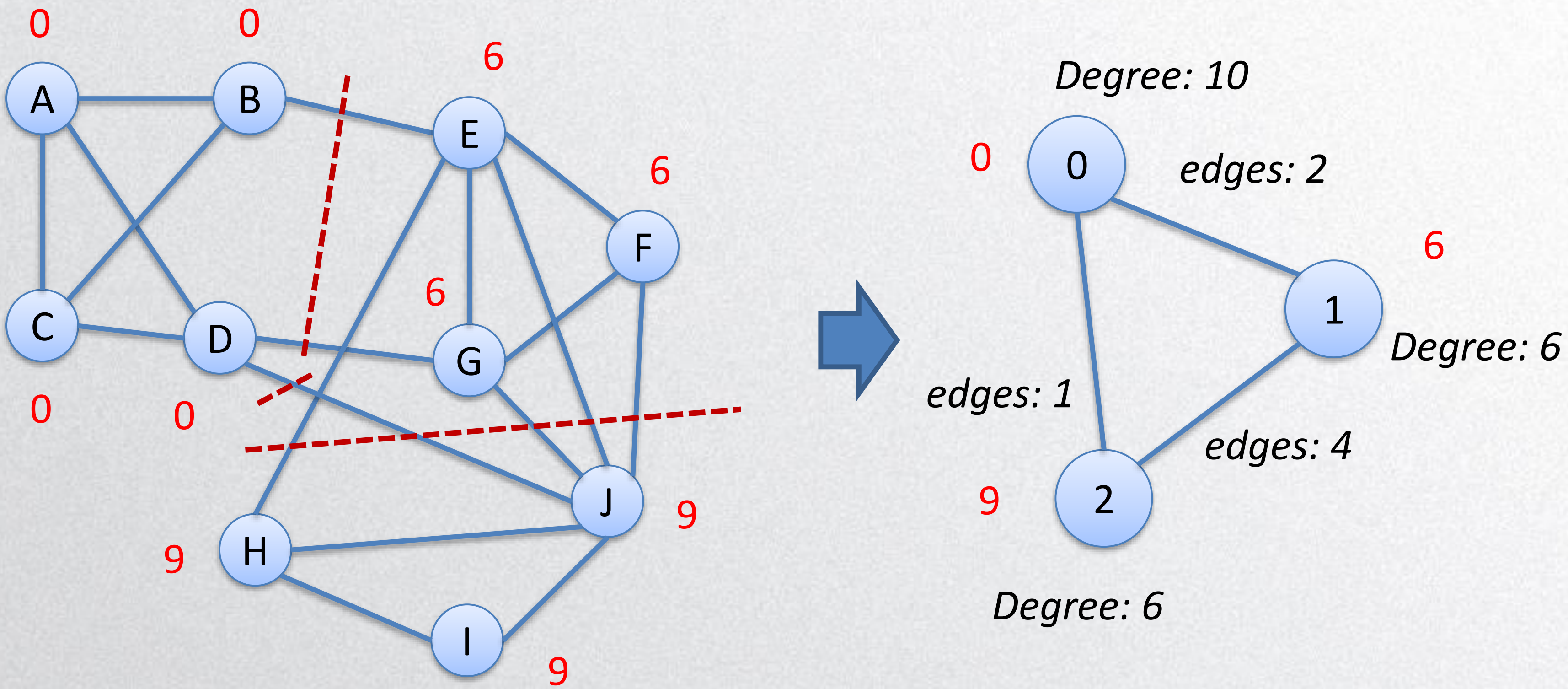
'J': 9, 'H': 9, 'I': 9}



图的社区检测：Louvain算法&标签传播（LPA）

图的社区检测：Louvain算法&标签传播（LPA）

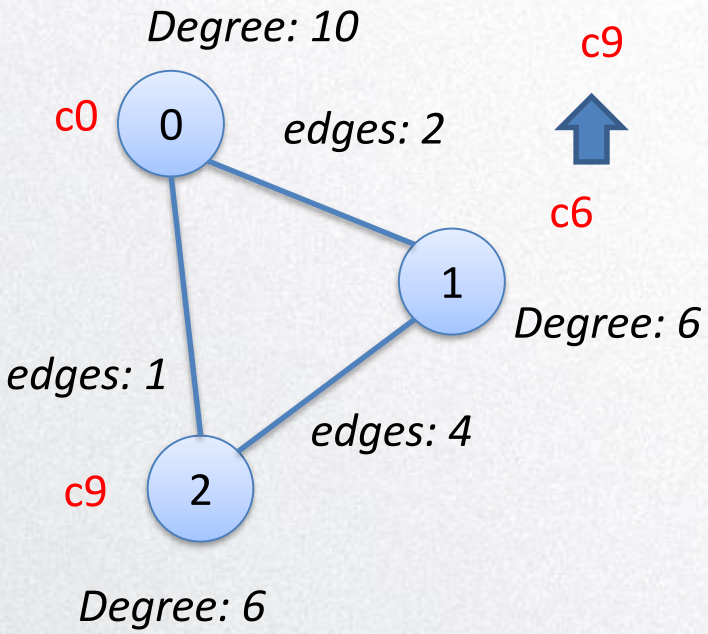
- 将社区抽象成super node



图的社区检测：Louvain算法&标签传播（LPA）

$$k_{i,in} - \frac{(\sum tot)k_i}{m}$$

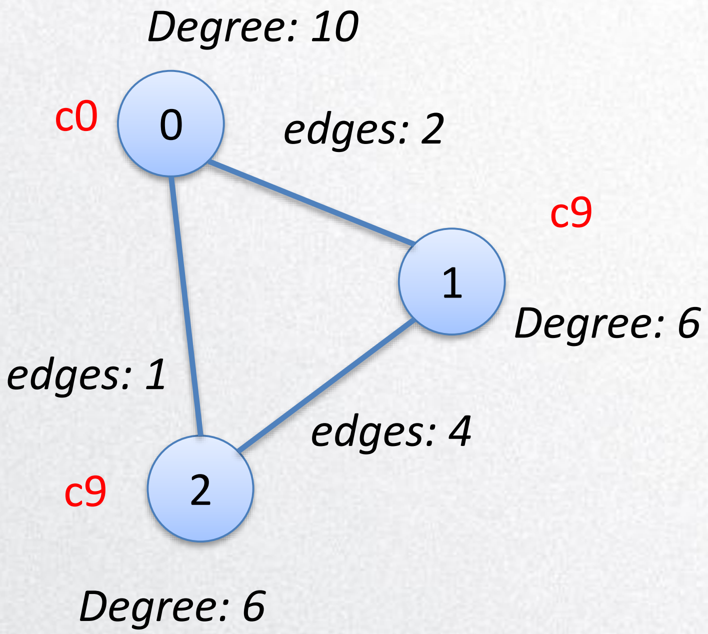
- 将社区抽象成super node
- 枚举一个节点的序列
  - 节点：1, 2, 0
- 邻居社区：
  - 社区：{0, 9}
- 模块度计算
  - $\Delta Q(C_6 \to 1) = 0 - \frac{0*(2+4)}{18} = 0$
  - $\Delta Q(1 \to C_0) = 2 - \frac{(10+2+1)*(2+4)}{18} = \frac{-42}{18}$
  - $\Delta Q(1 \to C_9) = 4 - \frac{(6+1+4)*(2+4)}{18} = \frac{6}{18}$
- 将节点1合并到社区9



图的社区检测：Louvain算法&标签传播（LPA）

$$k_{i,in} - \frac{(\sum tot)k_i}{m}$$

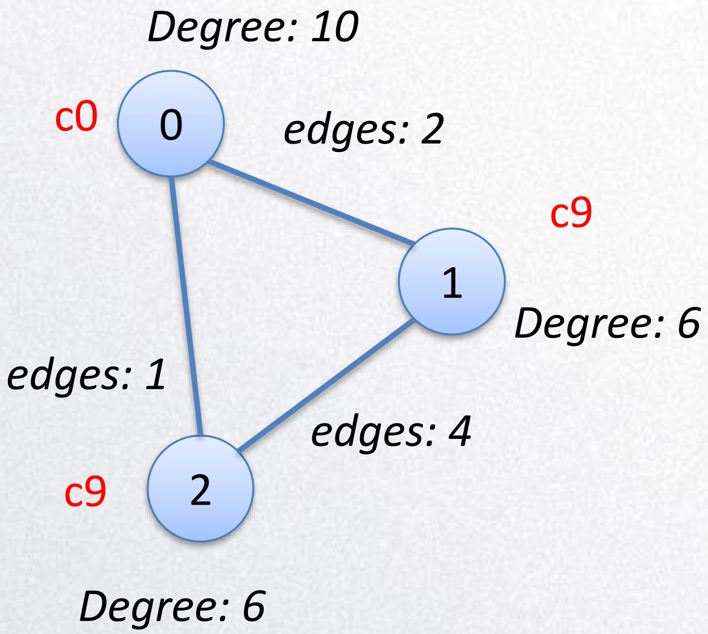
- 将社区抽象成super node
- 枚举一个节点的序列
  - 节点：1, 2, 0
- 邻居社区：
  - 社区：{0，9}
- 模块度计算
  - $\Delta Q(C_9 \to 2) = -\left(4 - \frac{(6+2+4)*(1+4)}{18}\right) = -\frac{12}{18}$
  - $\Delta Q(2 \to C_0) = 1 - \frac{(10+2+1)*(1+4)}{18} = \frac{-47}{18}$
  - $\Delta Q(2 \to C_9) = 4 - \frac{(6+2+4)*(1+4)}{18} = \frac{12}{18}$
- 将节点2不动



图的社区检测：Louvain算法&标签传播（LPA）

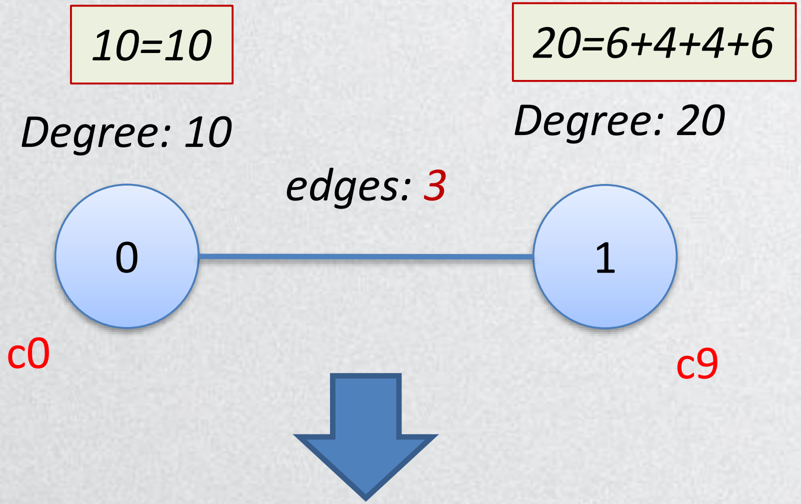
$$k_{i,in} - \frac{(\sum tot)k_i}{m}$$

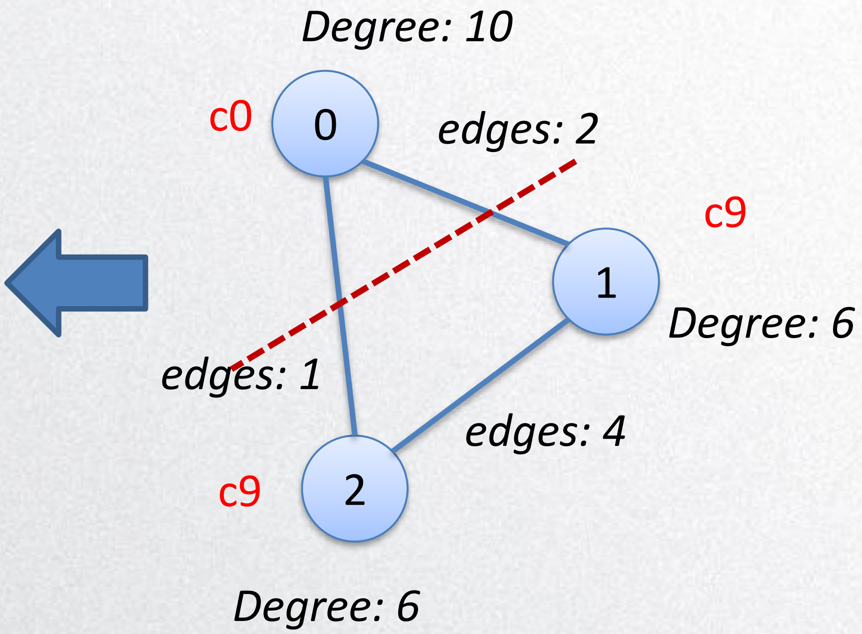
- 将社区抽象成super node
- 枚举一个节点的序列
  - 节点：1, 2, 0
- 邻居社区：
  - 社区：{9}
- 模块度计算
  - $\Delta Q(C_0 \to 0) = -\left(0 - \frac{(0)*(2+1)}{18}\right) = 0$
  - $\Delta Q(0 \to C_9) = 3 - \frac{(6+4+2+6+4+1)*(2+1)}{18} = -\frac{15}{18}$
  - 所以节点0不动



图的社区检测：Louvain算法&标签传播（LPA）

- 将社区抽象成super node

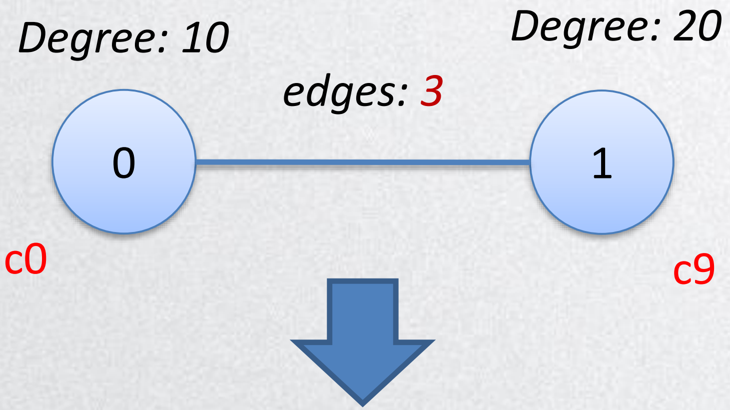




练习：计算将节点0和节点1合并时Modularity变化

图的社区检测：Louvain算法&标签传播（LPA）

- 将社区抽象成super node



练习：计算将节点0和节点1合并时Modularity变化

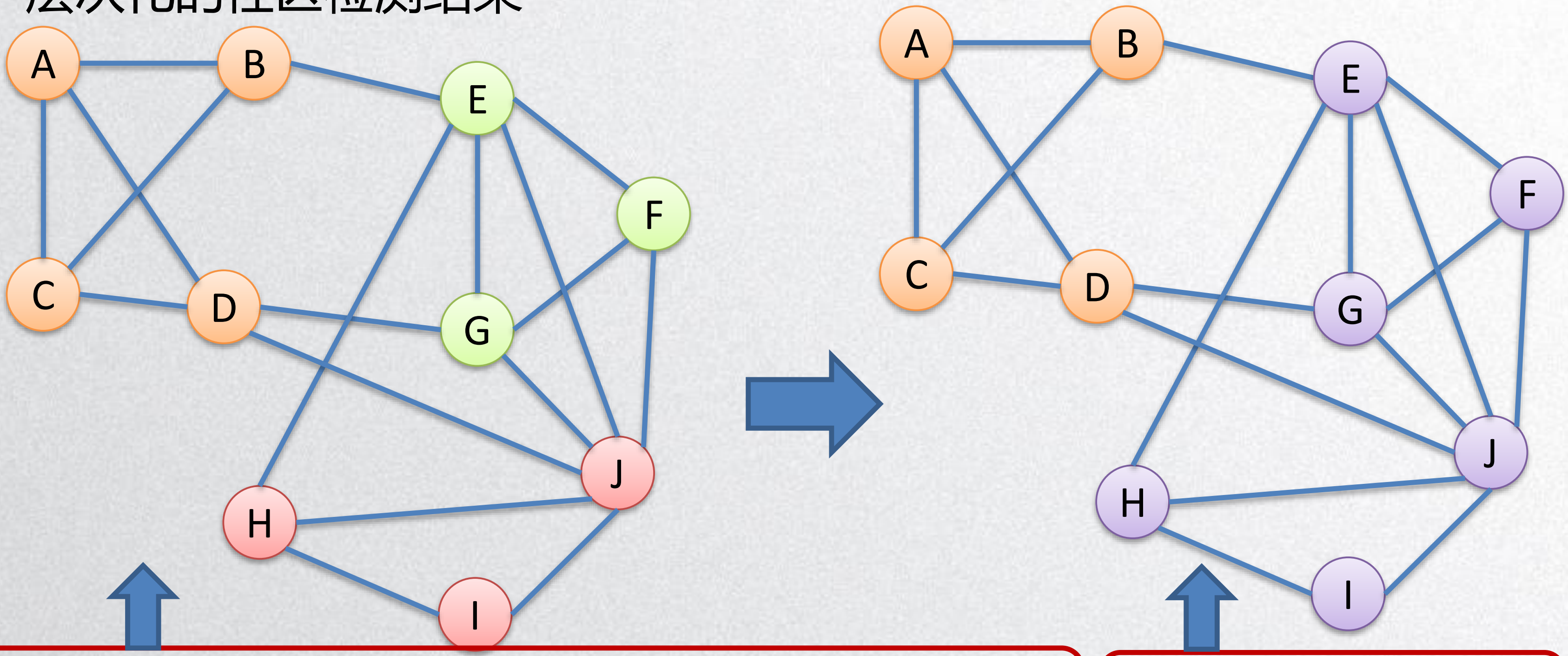
$$k_{i,in} - \frac{(\sum tot)k_i}{m}$$

$$\Delta Q(C_0 \to 0) = -\left(0 - \frac{(10)*(3)}{18}\right) = 0$$

$$\Delta Q(0 \to C_9) = 3 - \frac{(20+3)*(3)}{18} = -\frac{15}{18}$$

算法运行示例

- 层次化的社区检测结果

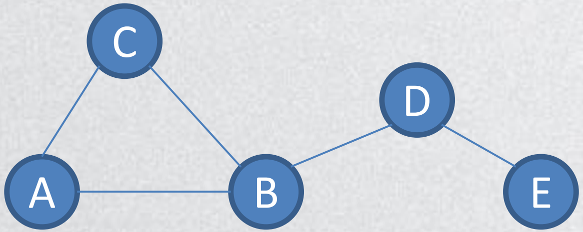


[{'A': 0, 'B': 0, 'C': 0, 'D': 0, 'E': 1, 'G': 1, 'F': 1, 'J': 2, 'H': 2, 'I': 2}, {0: 0, 1: 1, 2: 1}]

图的社区检测：Louvain算法&标签传播（LPA）

图的社区检测：Louvain算法&标签传播（LPA）

- 练习
- 请针对下图运行Louvain算法，得到社区检测的结果



1.刚开始，每个节点一个社区
2.生成随机节点访问序列
3.依次访问这些节点
  针对每个节点的社区的变化
  计算模块度变化
  适时进行社区调整
4.看看有没有继续调整的必要

图的社区检测：Louvain算法&标签传播（LPA）

图的社区检测：Louvain算法&标签传播（LPA）

- Louvain算法
  - 安装相关Python库

```
$ pip install python-louvain
```

  - 代码中装载louvain库

```
import community
```

名称	修改日期	类型	大小
01louvain_test.py	2021/10/28 18:22	Python File	2 KB

图的社区检测：Louvain算法&标签传播（LPA）

- Louvain算法
  - 创建图

```python
import networkx as nx
import matplotlib.pyplot as plt
import community

def simple_graph():
    G = nx.Graph()
    G.name = "Simple Graph"
    node_list = ['A','B','C','D','E','F','G','H','I','J']
    G.add_nodes_from(node_list)
    edge_list  = [('A','B'), ('A','C'),('A','D'),
              ('B','C'),('B','E'),
              ('C','D'),
              ('D','G'),('D','J'),
              ('E','F'),('E','G'),('E','H'),('E','J'),
              ('F','G'), ('F','J'),
               ('G','J'),
               ('H','I'), ('H','J'),
               ('I','J'),
              ]
    G.add_edges_from(edge_list)
    pos = {'A': [0.1, 0.1], 'B': [0.3, 0.1], 'C': [0.1, 0.4], 'D': [0.3, 0.4],
           'E': [0.5, 0.2], 'F': [0.7, 0.3], 'G': [0.5, 0.4],
           'H': [0.4, 0.5], 'I': [0.7, 0.5], 'J': [0.7, 0.4]}
    return G, pos
```

图的社区检测：Louvain算法&标签传播（LPA）

- Louvain算法
  - 绘制图

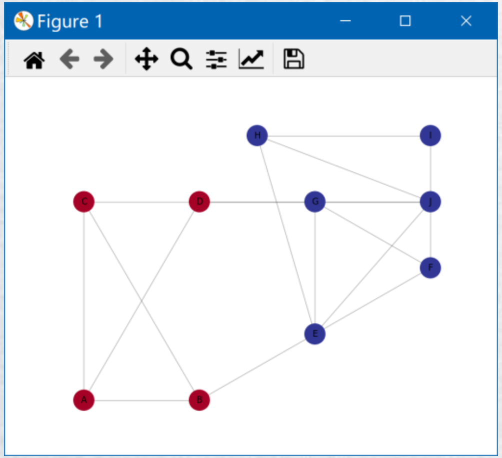
```
def draw_graph(G, pos,  partition, num):
    plt.figure(num)
    plt.axis('off')
    nx.draw_networkx_nodes(G, pos, node_size=800,
                           cmap=plt.cm.RdYlBu, node_color=list(partition.values()))
    nx.draw_networkx_labels(G, pos)
    nx.draw_networkx_edges(G, pos, alpha=0.3)
    plt.show()

```

图的社区检测：Louvain算法&标签传播（LPA）

- Louvain算法
  - 社区检测

```python
G,pos = simple_graph()
partition = community.best_partition(G)  # compute communities
draw_graph(G,pos,partition,1)
```



图的社区检测：Louvain算法&标签传播（LPA）

- Louvain算法
  - 空手道俱乐部社区检测示意

```python
G = nx.karate_club_graph()
pos = nx.spring_layout(G)  # compute graph layout
partition = community.best_partition(G)  # compute
draw_graph(G,pos,partition,2)
```