B. Node Reachability

The two extremes

Nodes are assumed to be in the same community

1. If there is a path between them (regardless of the distance) or
2. They are so close as to be immediate neighbors

How? Find using BFS/DFS

Challenge: most nodes are in one community (giant component)

How? Finding Cliques

Challenge: Cliques are challenging to find and are rarely observed

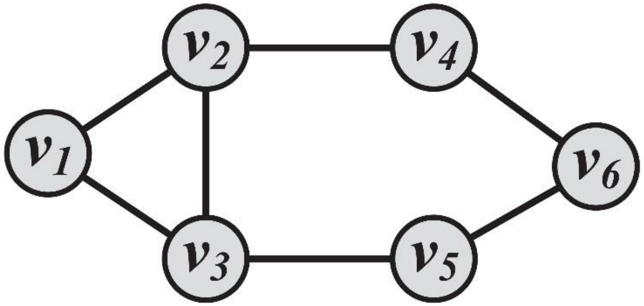
Solution: find communities that are in between cliques and connected components in terms of connectivity and have small shortest paths between their nodes

Special Subgraphs

1. *k*-**Clique**: a **maximal** subgraph in which the largest shortest path distance between any nodes is less than or equal to *k*

2. *k*-**Club**: follows the same definition as a *k*-clique
   - **Additional Constraint:** nodes on the shortest paths should be part of the subgraph (i.e., diameter)

3. *k*-**Clan**: a *k*-clique where for all shortest paths within the subgraph the distance is equal or less than *k*
   - All *k*-clans are *k*-cliques, but not vice versa

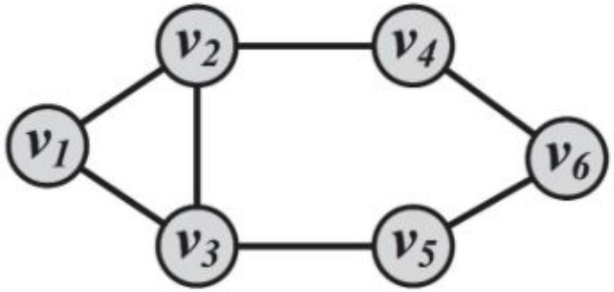


2-cliques : $\{v_1, v_2, v_3, v_4, v_5\}, \{v_2, v_3, v_4, v_5, v_6\}$

2-clubs : $\{v_2, v_3, v_4, v_5, v_6\}$

2-clans : $\{v_2, v_3, v_4, v_5, v_6\}$

K-clique



- Shortest path =1
  - v1v2, v1v3, v2v3, v2v4, v3v5, v5v6, v4v6
- Shortest path =2
  - v1v4, v1v5, v2v5, v2v6, v3v6, v3v4, v4v5
- Shortest path=3
  - v1v6

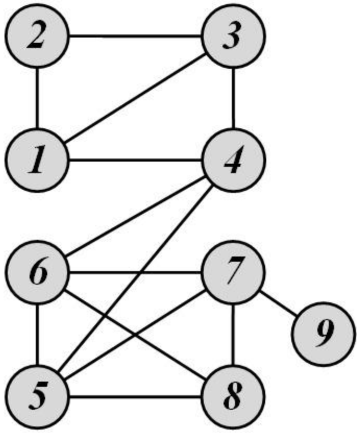
C. Node Similarity

- Similar (or most similar) nodes are assumed to be in the same community
  - A classical clustering algorithm (e.g., k-means) is applied to node similarities to find communities

- Node similarity can be defined
  - Using the similarity of node neighborhoods (**Structural Equivalence**) – Ch. 3
  - Similarity of social circles (**Regular Equivalence**) – Ch. 3

Structural equivalence: two nodes are structurally equivalent iff. they are connecting to the same set of actors

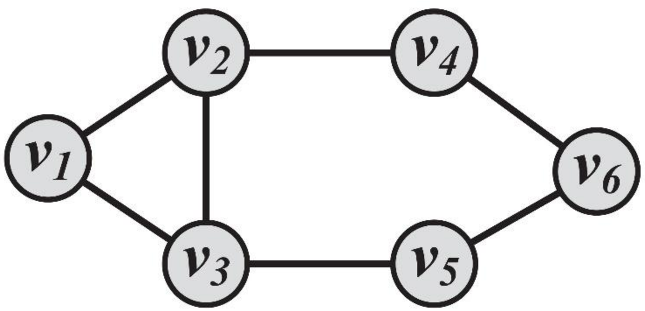
Nodes 1 and 3 are structurally equivalent, So are nodes 5 and 6.



Node Similarity (Structural Equivalence)

Jaccard Similarity $$\sigma_{\text{Jaccard}}(v_i, v_j) = \frac{|N(v_i) \cap N(v_j)|}{|N(v_i) \cup N(v_j)|}$$

Cosine similarity $$\sigma_{\text{Cosine}}(v_i, v_j) = \frac{|N(v_i) \cap N(v_j)|}{\sqrt{|N(v_i)||N(v_j)|}}$$



$$\sigma_{\text{Jaccard}}(v_2, v_5) = \frac{|\{v_1, v_3, v_4\} \cap \{v_3, v_6\}|}{|\{v_1, v_3, v_4, v_6\}|} = 0.25$$

$$\sigma_{\text{Cosine}}(v_2, v_5) = \frac{|\{v_1, v_3, v_4\} \cap \{v_3, v_6\}|}{\sqrt{|\{v_1, v_3, v_4\}||\{v_3, v_6\}|}} = 0.40$$

Group-Based Community Detection

Group-Based Community Detection

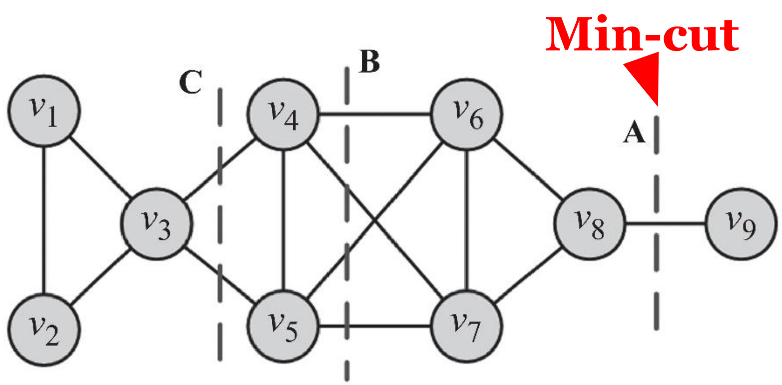
Group-based community detection: finding communities that have certain **group properties**

Group Properties:

I. **Balanced:** Spectral clustering
II. **Robust:** k-connected graphs
III. **Modular:** Modularity Maximization
IV. **Dense:** Quasi-cliques
V. **Hierarchical:** Hierarchical clustering

I. Balanced Communities

- Community detection can be thought of *graph clustering*
- **Graph clustering:** we cut the graph into several partitions and assume these partitions represent communities
- **Cut**: partitioning (*cut*) of the graph into two (or more) sets (*cutsets*)
  - **The size of the cut** is the number of edges that are being cut
- **Minimum cut (min-cut) problem**: find a graph partition such that the number of edges between the two sets is minimized



Min-cuts can be computed efficiently using the max-flow min-cut theorem

Min-cut often returns an imbalanced partition, with one set being a singleton

Ratio Cut and Normalized Cut

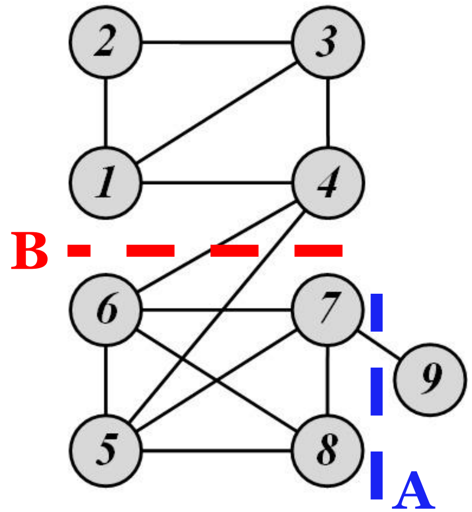
- To mitigate the min-cut problem we can change the objective function to consider community size

$$\text{Ratio Cut}(P) = \frac{1}{k}\sum_{i=1}^{k}\frac{\text{cut}(P_i,\bar{P}_i)}{|P_i|}$$

$$\text{Normalized Cut}(P) = \frac{1}{k}\sum_{i=1}^{k}\frac{\text{cut}(P_i,\bar{P}_i)}{\text{vol}(P_i)}$$

- $\bar{P}_i = V - P_i$ is the complement cut set
- $cut(P_i, \bar{P}_i)$ is the size of the cut
- $vol(P_i) = \sum_{v \in P_i} d_v$

Ratio Cut & Normalized Cut: Example



For Cut A

$$\text{Ratio Cut}(\{1,2,3,4,5,6,7,8\},\{9\}) = \tfrac{1}{2}(\tfrac{1}{1}+\tfrac{1}{8}) = 9/16 = 0.56$$
$$\text{Normalized Cut}(\{1,2,3,4,5,6,7,8\},\{9\}) = \tfrac{1}{2}(\tfrac{1}{1}+\tfrac{1}{27}) = 14/27 = 0.52$$

For Cut B

$$\text{Ratio Cut}(\{1,2,3,4\},\{5,6,7,8,9\}) = \tfrac{1}{2}(\tfrac{2}{4}+\tfrac{2}{5}) = 9/20 = 0.45 < 0.56$$
$$\text{Normalized Cut}(\{1,2,3,4\},\{5,6,7,8,9\}) = \tfrac{1}{2}(\tfrac{2}{12}+\tfrac{2}{16}) = 7/48 = 0.15 < 0.52$$

Both ratio cut and normalized cut prefer a balanced partition

Spectral Clustering

Reformulating ratio cut (or normalized cut) in matrix format

- Let $X_{ij} = 1$, when node i is member of community j; 0, otherwise
- Let $D = diag(d_1, d_2, \cdots, d_n)$ be the diagonal degree matrix
- The ith entry on the diagonal of $X^T AX$ is the number of edges that are inside community i.
- The ith element on the diagonal of $X^T DX$ is the number of edges that are connected to members of community i.
- The ith element on the diagonal of $X^T(D - A)X$ is the number of edges in the cut that separates community i from other nodes.

The ith diagonal element of $X^T(D - A)X$ is equivalent to $\mathrm{cut}(P_i, \bar{P}_i)$

Spectral Clustering

So ratio cut is

$$
\begin{aligned}
\text{Ratio Cut}(P) &= \frac{1}{k}\sum_{i=1}^{k}\frac{\text{cut}(P_i,\bar{P}_i)}{|P_i|} \\
&= \frac{1}{k}\sum_{i=1}^{k}\frac{X_i^T(D-A)X_i}{X_i^T X_i} \\
&= \frac{1}{k}\sum_{i=1}^{k}\hat{X}_i^T(D-A)\boxed{\hat{X}_i}
\end{aligned}
$$

$$\hat{X}_i = X_i/(X_i^T X_i)^{1/2}$$

Spectral Clustering

Both ratio/normalized cut can be reformulated as

$$\min_{\hat{X}} \mathrm{Tr}(\hat{X}^T L \hat{X})$$

$$L = \begin{cases} D - A & \text{Ratio Cut Laplacian, i.e., Unnormalized Laplacian} \\ I - D^{-1/2} A D^{-1/2} & \text{Normalized Laplacian for Normalized Cut.} \end{cases}$$

$D = diag(d_1, d_2, \cdots, d_n)$ is a diagonal degree matrix

- Spectral relaxation:

$$\min_{\hat{X}} \mathrm{Tr}(\hat{X}^T L \hat{X})$$
$$s.t.\ \hat{X}^T \hat{X} = I_k$$

Optimal Solution

$\hat{X}$ is the top eigenvectors with the smallest eigenvalues

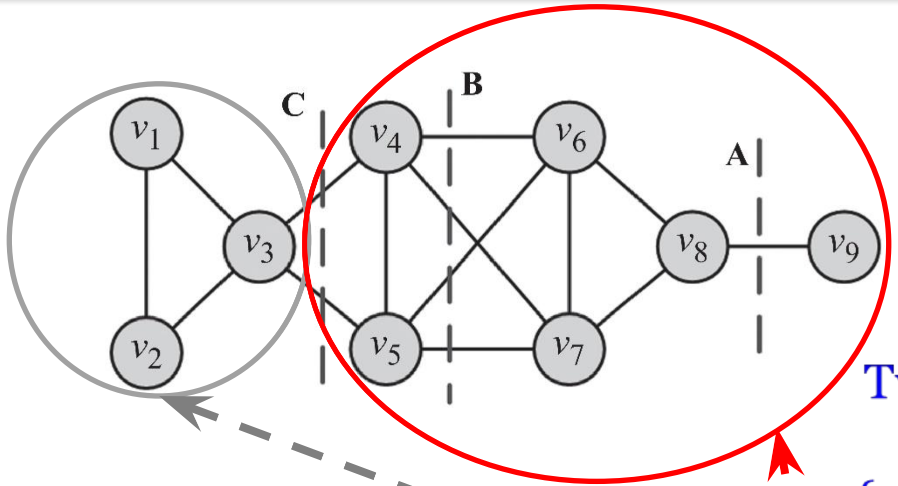
Recovering Integer Membership Values

- Because we performed spectral relaxation, the matrix obtained is not integer valued

- To recover X from $\hat{X}$ we can run k-means on $\hat{X}$

Spectral Clustering: Example

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$



Two communities:
$\{v_1, v_2, v_3\}$
$\{v_4, v_5, v_6, v_7, v_8, v_9\}$

$$D = diag(2, 2, 4, 4, 4, 4, 4, 3, 1)$$

$$L = D - A = \begin{bmatrix} 2 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 4 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 4 & -1 & -1 & -1 & 0 & 0 \\ 0 & 0 & -1 & -1 & 4 & -1 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 & 4 & -1 & -1 & 0 \\ 0 & 0 & 0 & -1 & -1 & -1 & 4 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & -1 & 3 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

2 Eigenvectors
i.e., we want
2 communities

$$\begin{bmatrix} 0.33 & \boxed{-0.46} \\ 0.33 & -0.46 \\ 0.33 & -0.26 \\ 0.33 & 1.16 \times 10^{-16} \\ 0.33 & 1.16 \times 10^{-16} \\ 0.33 & 0.13 \\ 0.33 & 0.13 \\ 0.33 & 0.33 \\ 0.33 & 0.59 \end{bmatrix}$$

***k*-means, *k* = 2**

II. Robust Communities

- The goal is find subgraphs robust enough such that removing some edges or vertices does not disconnect the subgraph

- ***k*-vertex connected (*k*-connected) graph:**
  - k is the minimum number of nodes that must be removed to disconnect the graph
- ***k*-edge connected:** at least k edges must be removed to disconnect the graph

- Examples:
  - Complete graph of size n: unique n-connected graph
  - A cycle: 2-connected graph

III. Modular Communities

Consider a graph $G(V, E)$, where the degrees are known beforehand however edges are not

- Consider two vertices vi and vj with degrees di and dj

What is an expected number of edges between v_i and v_j?

- For any edge going out of vi randomly the probability of this edge getting connected to vertex vj is

$$\frac{d_j}{\sum_i d_i} = \frac{d_j}{2m}$$

Modularity and Modularity Maximization

- Given a degree distribution, we know the expected number of edges between any pairs of vertices

- We assume that real-world networks should be far from random. Therefore, the more distant they are from this randomly generated network, the more structural they are

- Modularity defines this distance and modularity maximization tries to maximize this distance

Normalized Modularity

Consider a partitioning of the data $P = (P_1, P_2, P_3, \ldots, P_k)$

For partition P_x, this distance can be defined as

$$\sum_{i,j \in P_x} A_{ij} - \frac{d_i d_j}{2m}$$

This distance can be generalized for a partitioning P

$$\sum_{x=1}^{k} \sum_{i,j \in P_x} A_{ij} - \frac{d_i d_j}{2m}$$

The normalized version of this distance is defined as Modularity

$$Q = \frac{1}{2m} \sum_{x=1}^{k} \sum_{i,j \in P_x} A_{ij} - \frac{d_i d_j}{2m}$$

Modularity Maximization

Modularity matrix

$$B = A - dd^T/2m$$

$d \in \mathbb{R}^{n\times 1}$ is the degree vector for all nodes

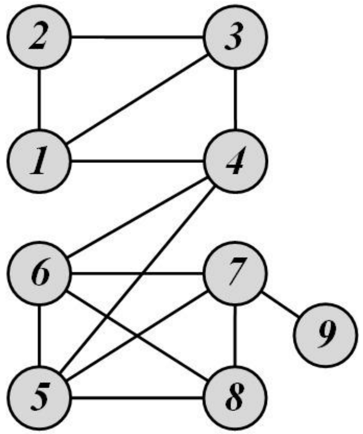
Reformulation of the modularity

$$Q = \frac{1}{2m}\mathrm{Tr}(X^T B X)$$

- $X \in \mathbb{R}^{n\times k}$ is the indicator (partition membership) function:
  - $X_{ij} = 1$ iff. $v_i \in P_j$

- Similar to Spectral clustering,
  - We relax X to be orthogonal, i.e., matrix $\hat{X}$
  - The optimal solution for $\hat{X}$ is the top k eigenvectors of B
  - To recover the original X, we can run k-means on $\hat{X}$

Modularity Maximization: Example



$$B = A - dd^T/2m$$
$$B_{ij} = A_{ij} - d_i d_j / 2m$$

$$B = \begin{bmatrix} -0.32 & 0.79 & 0.68 & 0.57 & -0.43 & -0.43 & -0.43 & -0.32 & -0.11 \\ 0.79 & -0.14 & 0.79 & -0.29 & -0.29 & -0.29 & -0.29 & -0.21 & -0.07 \\ 0.68 & 0.79 & -0.32 & 0.57 & -0.43 & -0.43 & -0.43 & -0.32 & -0.11 \\ 0.57 & -0.29 & 0.57 & -0.57 & 0.43 & 0.43 & -0.57 & -0.43 & -0.14 \\ -0.43 & -0.29 & -0.43 & 0.43 & -0.57 & 0.43 & 0.43 & 0.57 & -0.14 \\ -0.43 & -0.29 & -0.43 & 0.43 & 0.43 & -0.57 & 0.43 & 0.57 & -0.14 \\ -0.43 & -0.29 & -0.43 & -0.57 & 0.43 & 0.43 & -0.57 & 0.57 & 0.86 \\ -0.32 & -0.21 & -0.32 & -0.43 & 0.57 & 0.57 & 0.57 & -0.32 & -0.11 \\ -0.11 & -0.07 & -0.11 & -0.14 & -0.14 & -0.14 & 0.86 & -0.11 & -0.04 \end{bmatrix}$$

Modularity Matrix

2 eigenvectors

$$\begin{bmatrix} 0.44 & -0.00 \\ 0.38 & 0.23 \\ 0.44 & -0.00 \\ 0.17 & -0.48 \\ -0.29 & -0.32 \\ -0.29 & -0.32 \\ -0.38 & 0.34 \\ -0.34 & -0.08 \\ -0.14 & 0.63 \end{bmatrix}$$

k-means

Two Communities:
{1, 2, 3, 4}
and
{5, 6, 7, 8, 9}

IV. Dense Communities: *γ-dense*

- The density of a graph defines how close a graph is to a clique

$$\gamma = \frac{|E|}{\binom{|V|}{2}}$$

- A subgraph $G(V,E)$ is a *γ-dense* (or quasi-clique) if

$$|E| \geq \gamma \binom{|V|}{2}$$

- A 1-dense graph is a clique
- We can find quasi-cliques using the brute force algorithm discussed previously, but there are more efficient methods

V. Hierarchical Communities

- Previous methods consider communities at a single level
  - Communities may have hierarchies
    - Each community can have sub/super communities
  - Hierarchical clustering deals with this scenario and generates community hierarchies

- Initially n members are considered as either 1 or n communities in hierarchical clustering. These communities are gradually
  - merged (agglomerative hierarchical clustering) or
  - split (divisive hierarchical clustering)

Hierarchical Community Detection

- **Goal**: build a hierarchical structure of communities based on network topology

- Allow the analysis of a network at different resolutions

- Representative approaches:
  - Divisive Hierarchical Clustering
  - Agglomerative Hierarchical clustering

Divisive Hierarchical Clustering

- Divisive clustering
  - Partition nodes into several sets
  - Each set is further divided into smaller ones
  - Network-centric partition can be applied for the partition
- One particular example:

Girvan-Newman Algorithm: recursively remove the "weakest" links within a "community"

  - Find the edge with the weakest link
  - Remove the edge and update the corresponding strength of each edge

- Recursively apply the above two steps until a network is discomposed into a <u>desired number of connected components</u>.
- Each component forms a community

Edge Betweenness

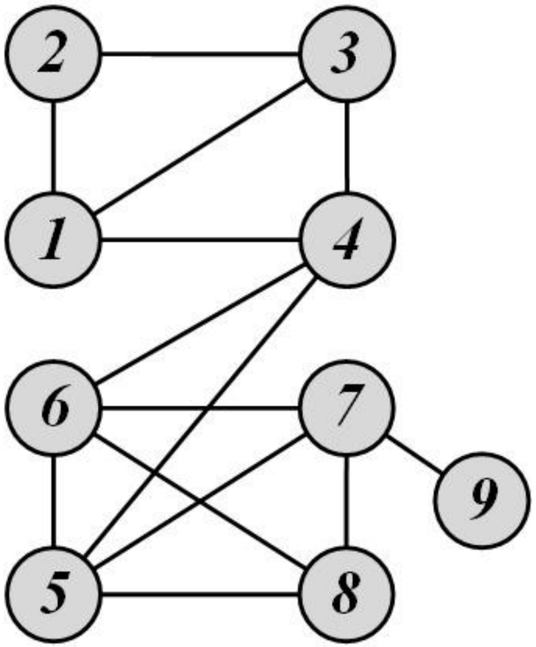
- To determine weakest links, the algorithm uses **edge betweenness**

Edge betweenness is the number of shortest paths that pass along with the edge

$$\text{edge-betweenness}(e) = \Sigma_{s<t} \frac{\sigma_{st}(e)}{\sigma_{s,t}}$$

- Edge betweenness measures the “bridgeness” of an edge between two communities
- The edge with **high** betweenness tends to be the bridge between two communities

Edge Betweenness: Example

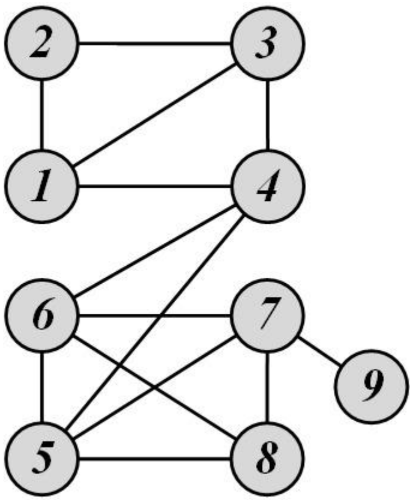


The edge betweenness of $e(1,2)$ is $6/2 + 1 = 4$, as all the shortest paths from 2 to $\{4,5,6,7,8,9\}$ have to either pass $e(1,2)$ or $e(2,3)$, and $e(1,2)$ is the shortest path between 1 and 2

The Girvan-Newman Algorithm

1. Calculate edge betweenness for all edges in the graph
2. Remove the edge with the **highest** betweenness
3. Recalculate betweenness for all edges affected by the edge removal
4. Repeat until all edges are removed

Edge Betweenness Divisive Clustering: Example



Initial betweenness value

	1	2	3	4	5	6	7	8	9
1	0	4	1	9	0	0	0	0	0
2	4	0	4	0	0	0	0	0	0
3	1	4	0	9	0	0	0	0	0
4	9	0	9	0	10	10	0	0	0
5	0	0	0	10	0	1	6	3	0
6	0	0	0	10	1	0	6	3	0
7	0	0	0	0	6	6	0	2	8
8	0	0	0	0	3	3	2	0	0
9	0	0	0	0	0	0	8	0	0

the first edge that needs to be removed is e(4, 5) (or e(4, 6))

By removing e(4, 5), we compute the edge betweenness once again; this time, e(4, 6) has the highest betweenness value: 20

This is because all shortest paths between nodes {1,2,3,4} to nodes {5,6,7,8,9} must pass e(4, 6); therefore, it has betweenness 4×5 = 20