Machine Learning with Graphs (MLG)

# Network Community Detection (1)

## What are clusters in graphs and how to find them?

Cheng-Te Li (李政德)

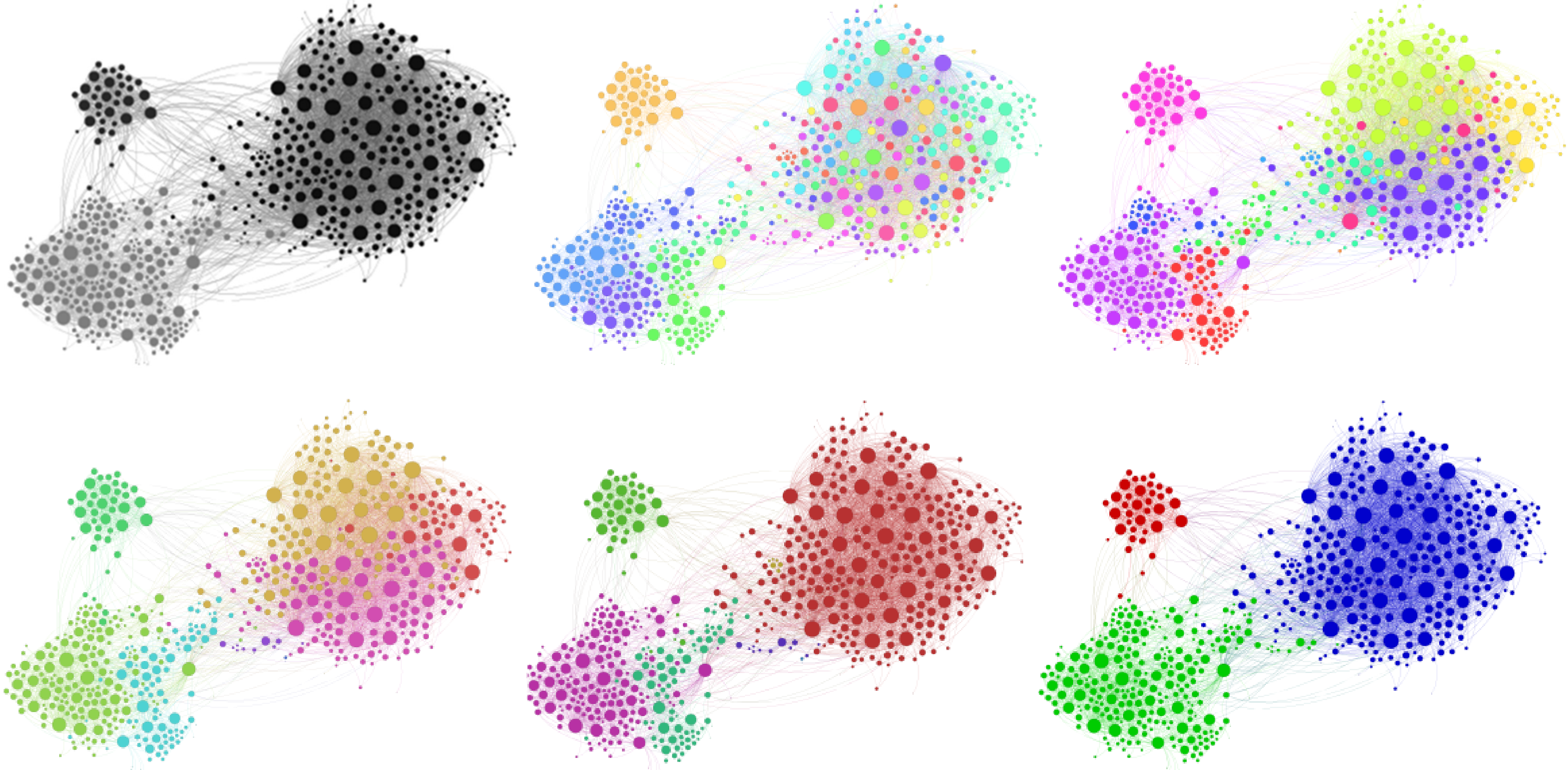
Institute of Data Science
National Cheng Kung University
chengte@mail.ncku.edu.tw

# Network Community Detection

- Given a social network, can we automatically identify a set of subgraphs with the best community structures?

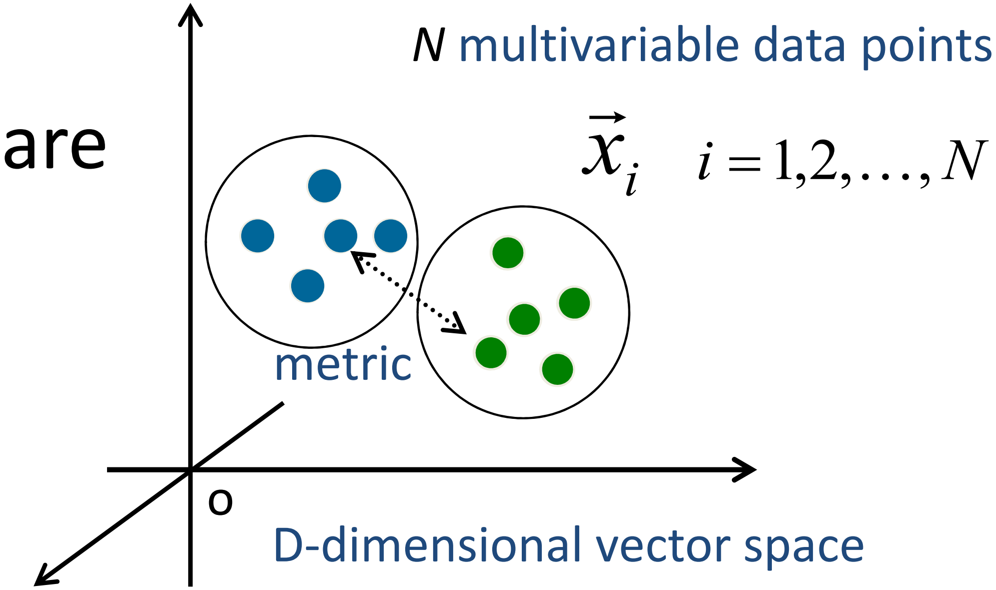
Applications:

- ✓ Friend Suggestion
- ✓ Viral Marketing
- ✓ Missing Value Inference
- ✓ Outlier Detection
- ✓ Multi-faceted Analysis
- ✓ Financial Support/Competition
- ✓ Brain Functionality

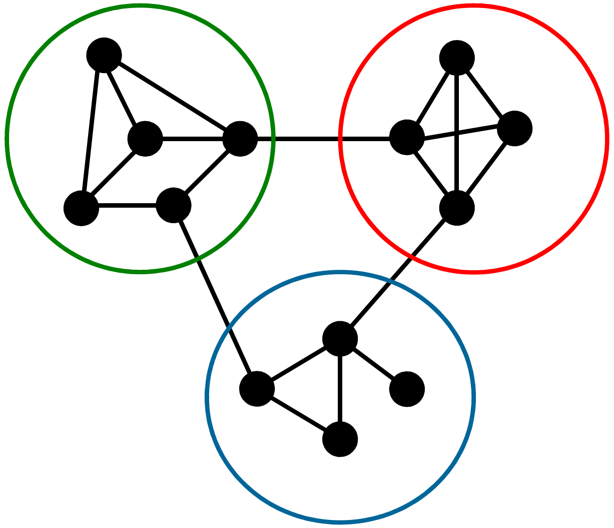


# Data Clustering

- Given a set of objects $V$, and a notion of similarity (or distance) between them, partition the objects into disjoint sets $S_1, S_2, \ldots, S_k$, such that objects within the each set are similar, while objects across different sets are dissimilar
    - Data in each subset share some common traits



# How about on Networks?



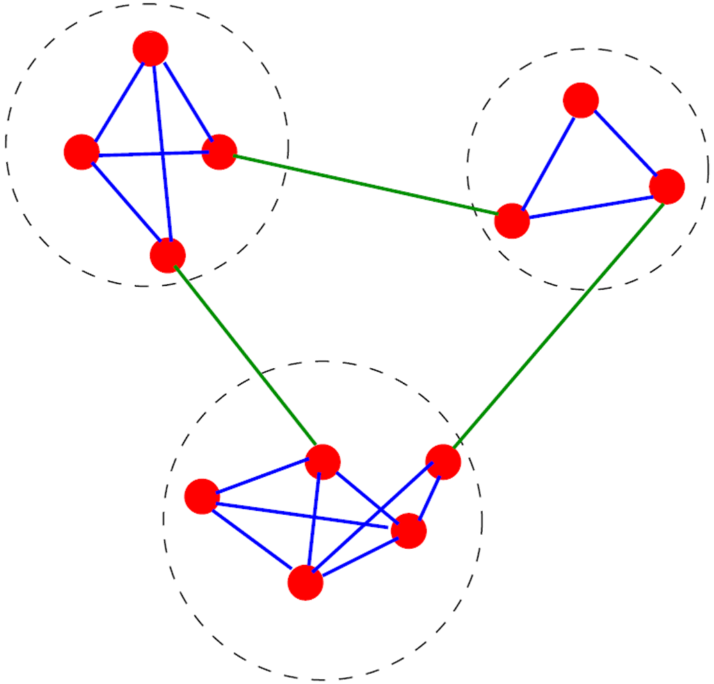
$N$ undifferentiated vertices (nodes)

NO prior information

Only know the edge (link) connectivity:
**Structural information**

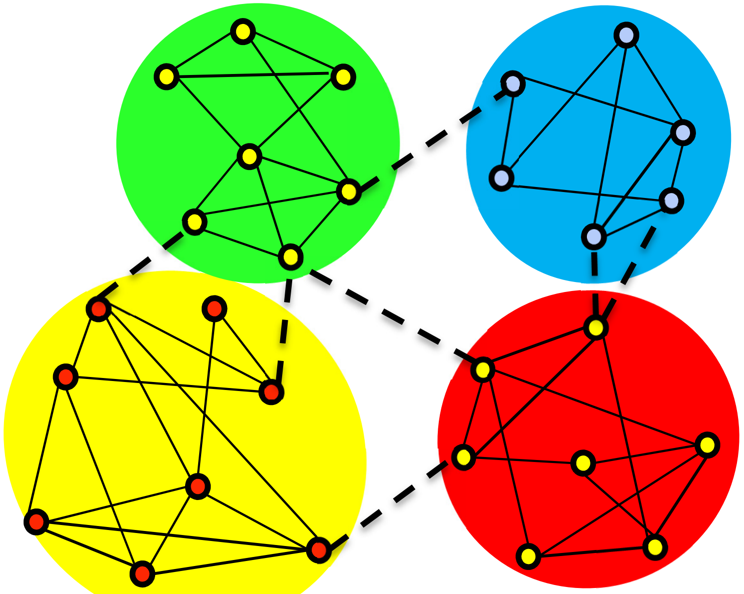
- How can we partition the network into **several parts**?
  - According to their structural connectivity
  - **Community structure**
- Applications
  - Topic/Domain Detection, Friend Suggestion, Viral Marketing, Graph Compression, Parallel Processing on graphs, etc.

# Community Structure

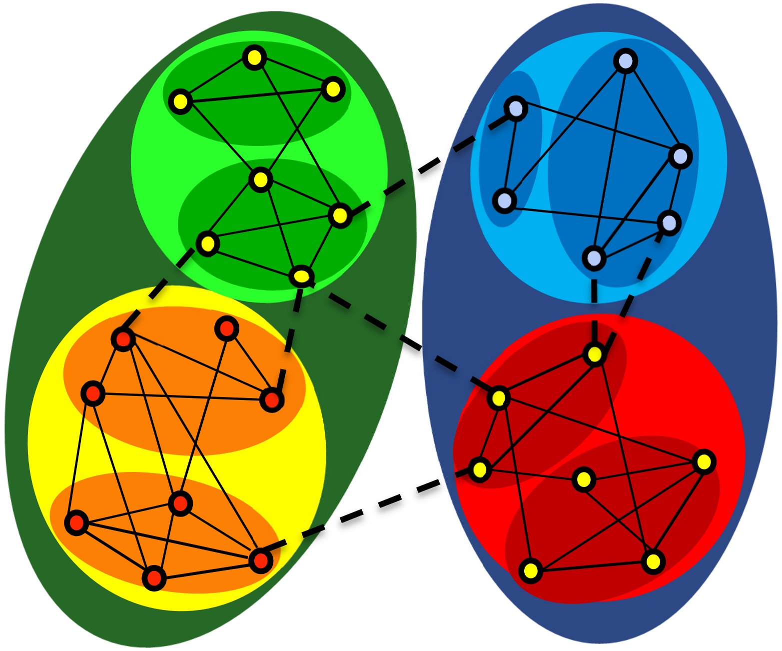


- Community structure
  - **Within** (intra-) group edges
    - **High** density
  - **Between** (inter-) group edges
    - **Low** density
  - The average path length among nodes is relatively small

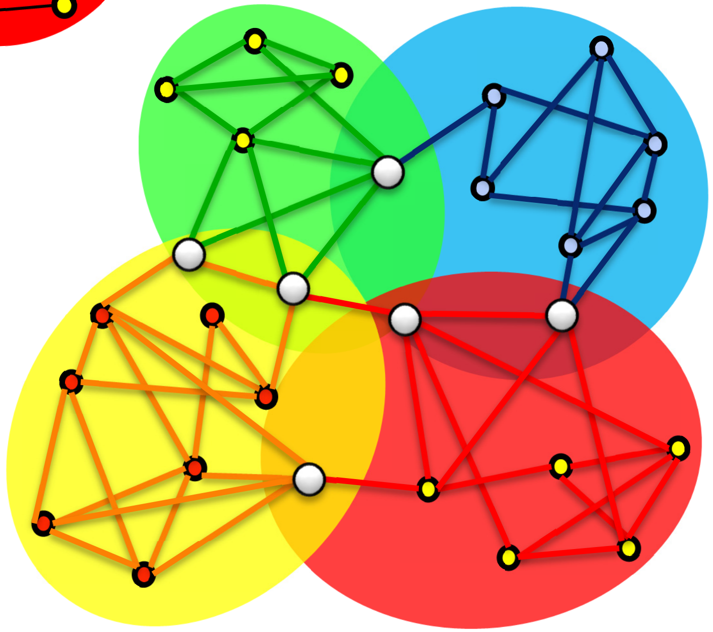
# Community Structure



Flat

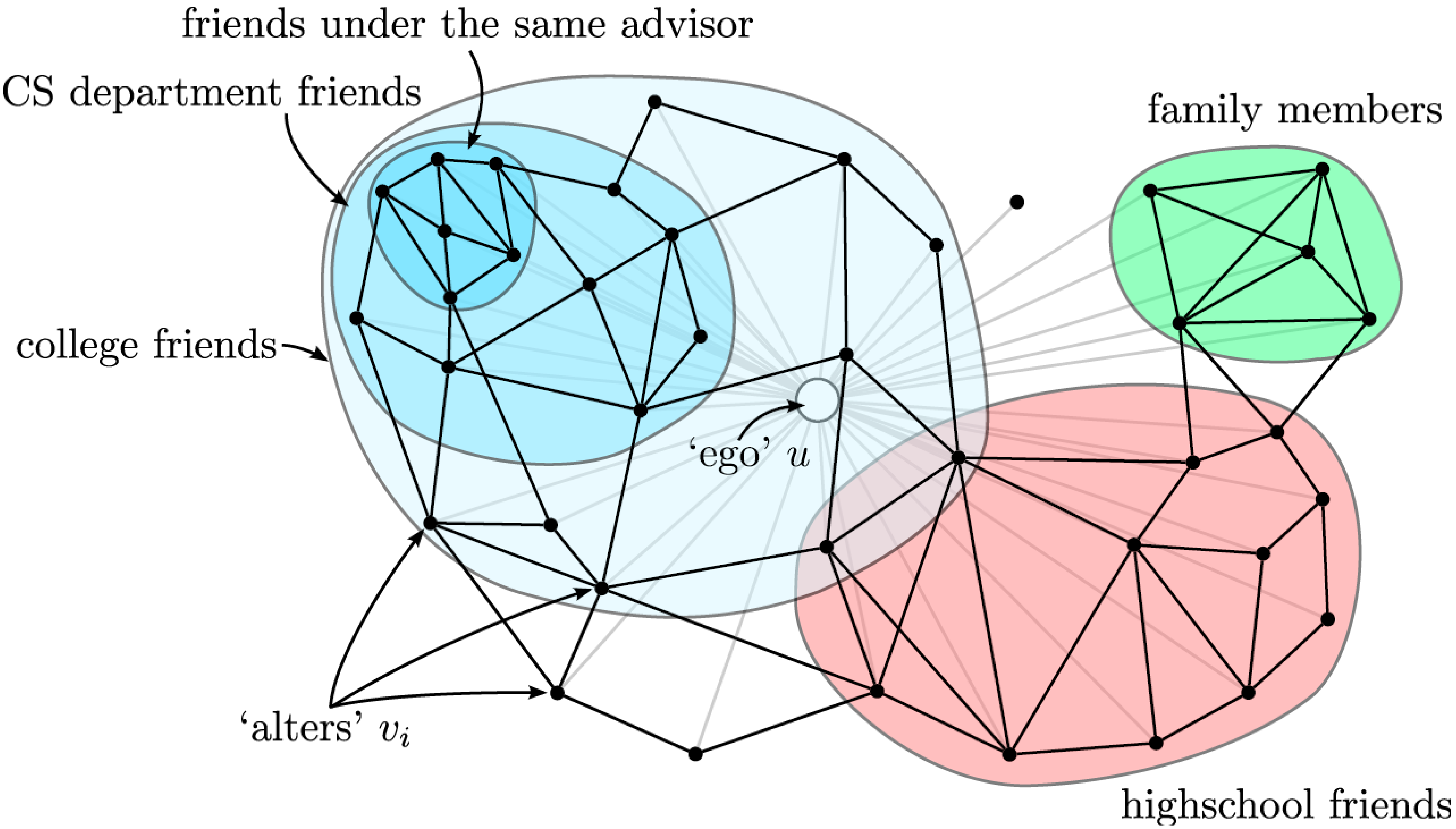


Hierarchical



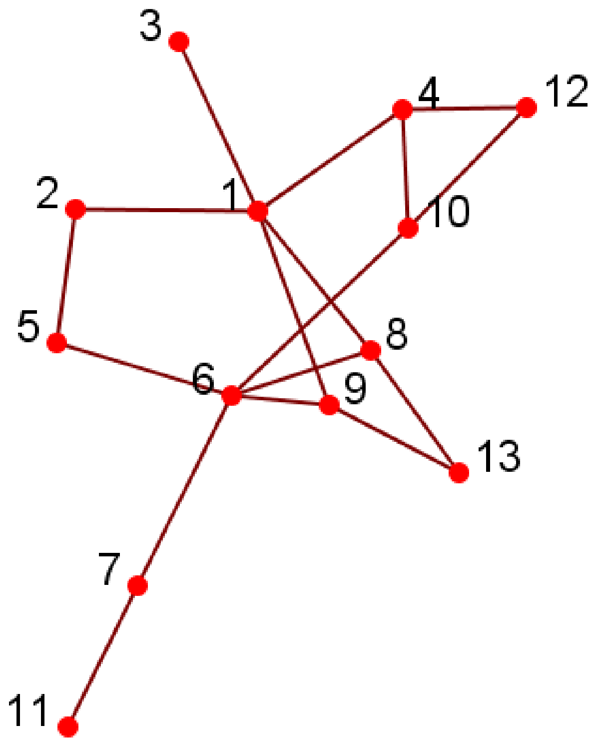
Overlapping

# Community Structure: Social Circle



# Node Similarity

- Node similarity is defined by how similar their interaction patterns are
- Two nodes are similar if they connect to the same set of actors
  - e.g., nodes 8 and 9 tend to belong to the same community
- In practice, use vector similarity
  - e.g., cosine similarity, Jaccard similarity



# Vector Similarity

a vector → 5

structurally equivalent: 8, 9

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 5 | | 1 | | | | 1 | | | | | | | |
| 8 | 1 | | | | | 1 | | | | | | | 1 |
| 9 | 1 | | | | | 1 | | | | | | | 1 |

Cosine Similarity:

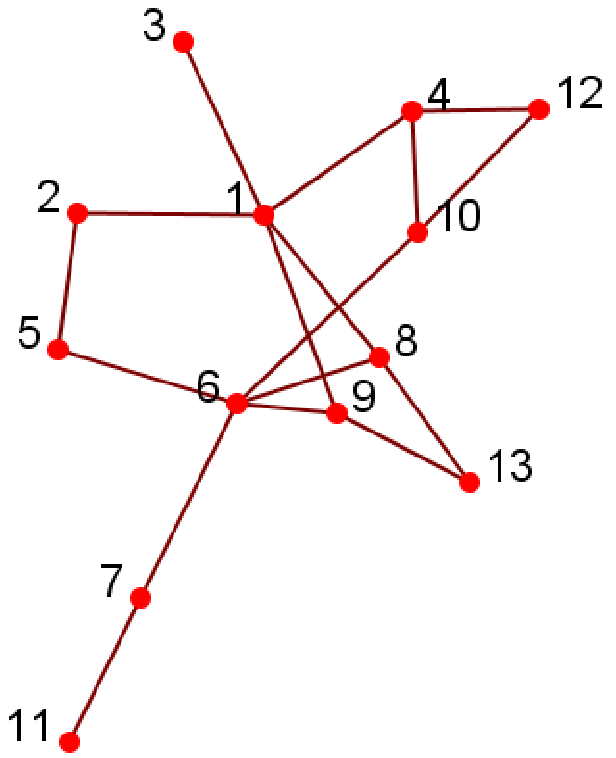
$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|}$$

$$sim(5,8) = \frac{1}{\sqrt{2} \times \sqrt{3}} = \frac{1}{\sqrt{6}}$$

Jaccard Similarity:

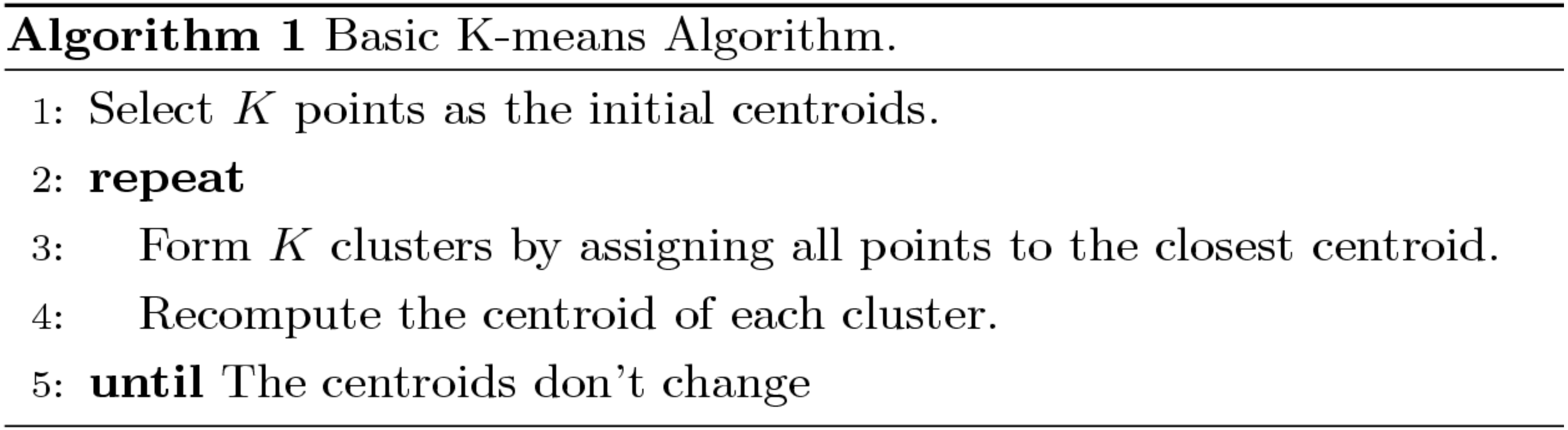
$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

$$J(5,8) = \frac{|\{6\}|}{|\{1,2,6,13\}|} = 1/4$$



# Clustering based on Node Similarity

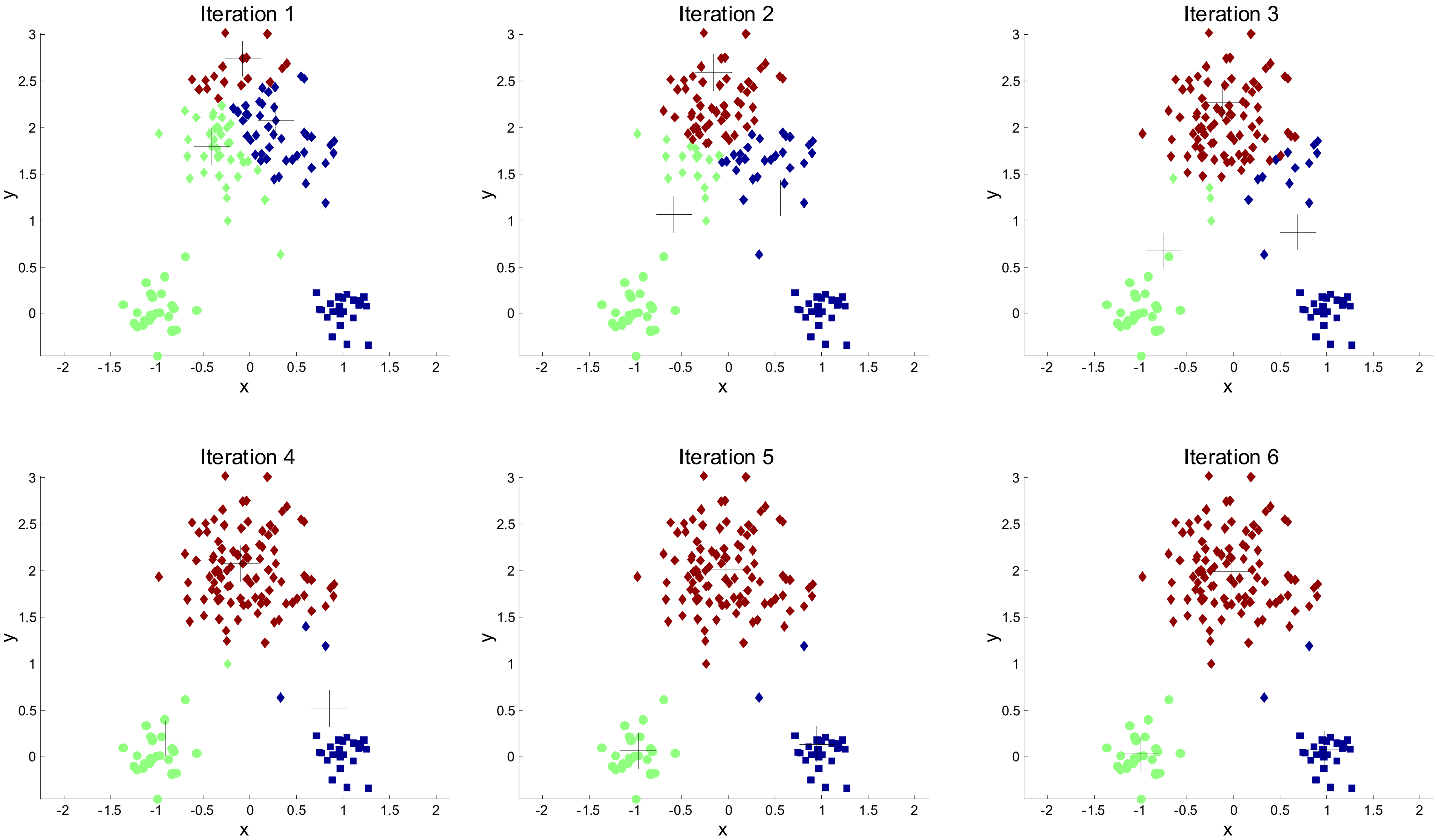
- For practical use with huge networks:
  - Consider the connections as features
  - Use Cosine or Jaccard similarity to compute vertex similarity
  - Apply classical k-means clustering Algorithm
- K-means Clustering Algorithm
  - Each cluster is associated with a centroid (center point)
  - Each node is assigned to the cluster with the closest centroid



**Algorithm 1** Basic K-means Algorithm.

```
Select $K$ points as the initial centroids.
repeat
    Form $K$ clusters by assigning all points to the closest centroid.
    Recompute the centroid of each cluster.
until The centroids don't change
```

# Illustration of k-means Clustering



http://etrex.tw/flash/kMeansClustering/kMeansClustering2.html

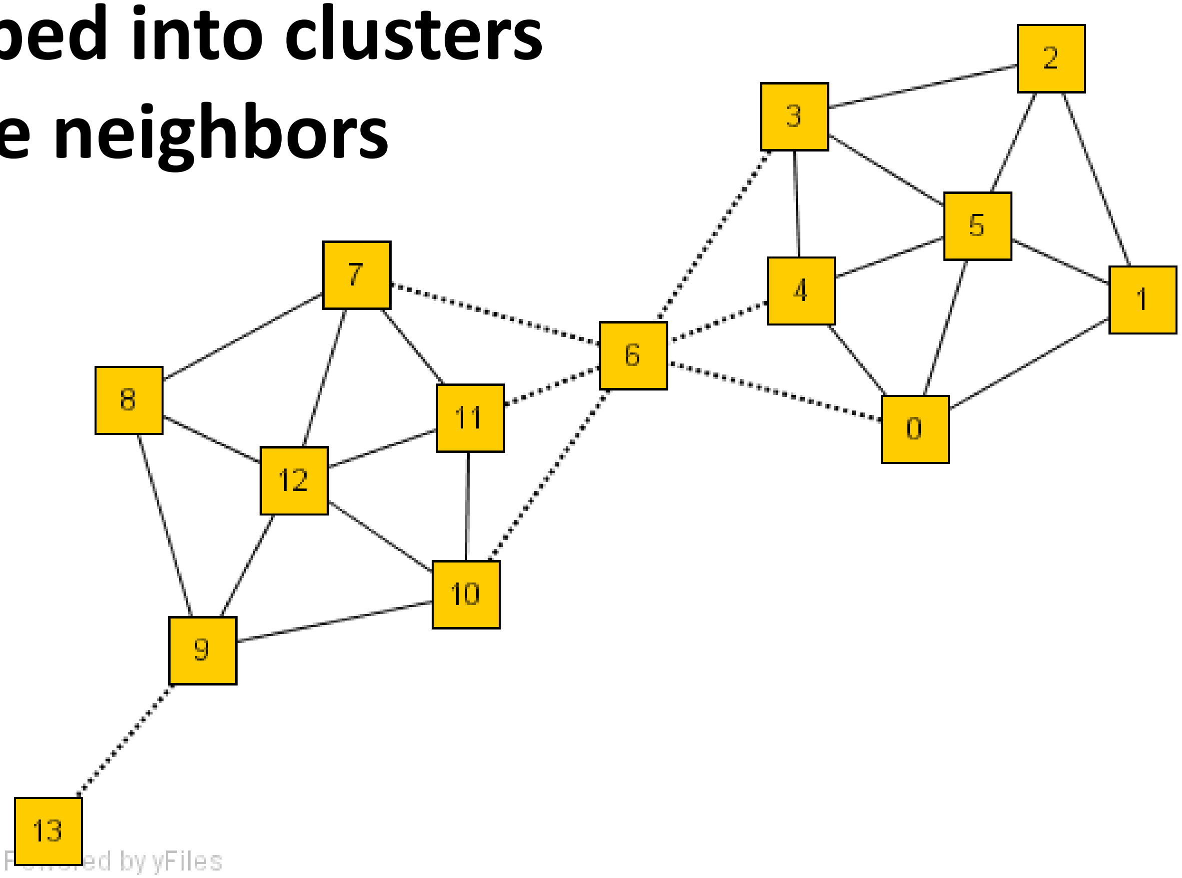
# Community Detection Approaches

- Propagation-based Method
  - Structural Clustering Algorithm for Networks (SCAN)
- Edge-Removal
  - Girvan-Newman Algorithm (GNA)
  - Fast Newman Algorithm
- Louvain Algorithm
- Label Propagation Algorithm

# Main Idea of SCAN

- Use the neighborhood of the vertices as clustering criteria
    - Instead of direct connections
    - **Vertices are grouped into clusters by how they share neighbors**

| | Neighbors |
|---|---|
| 0 | {**0,1,4,5**,6} |
| 5 | {**0,1**,2,3,**4,5**} |
| 9 | {8,**9**,10,12,**13**} |
| 13 | {**9,13**} |

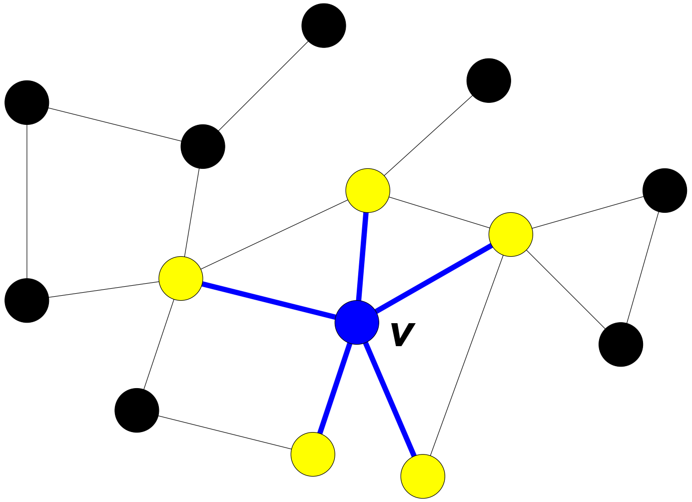


# The Neighbor of a Vertex

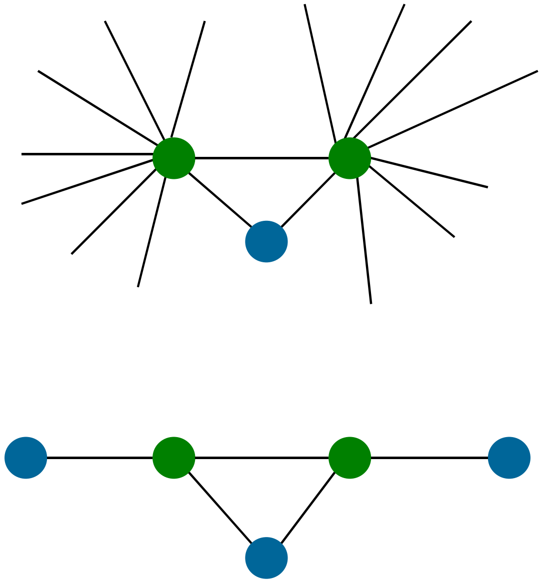
(Focus on simple, undirected and un-weighted graph)

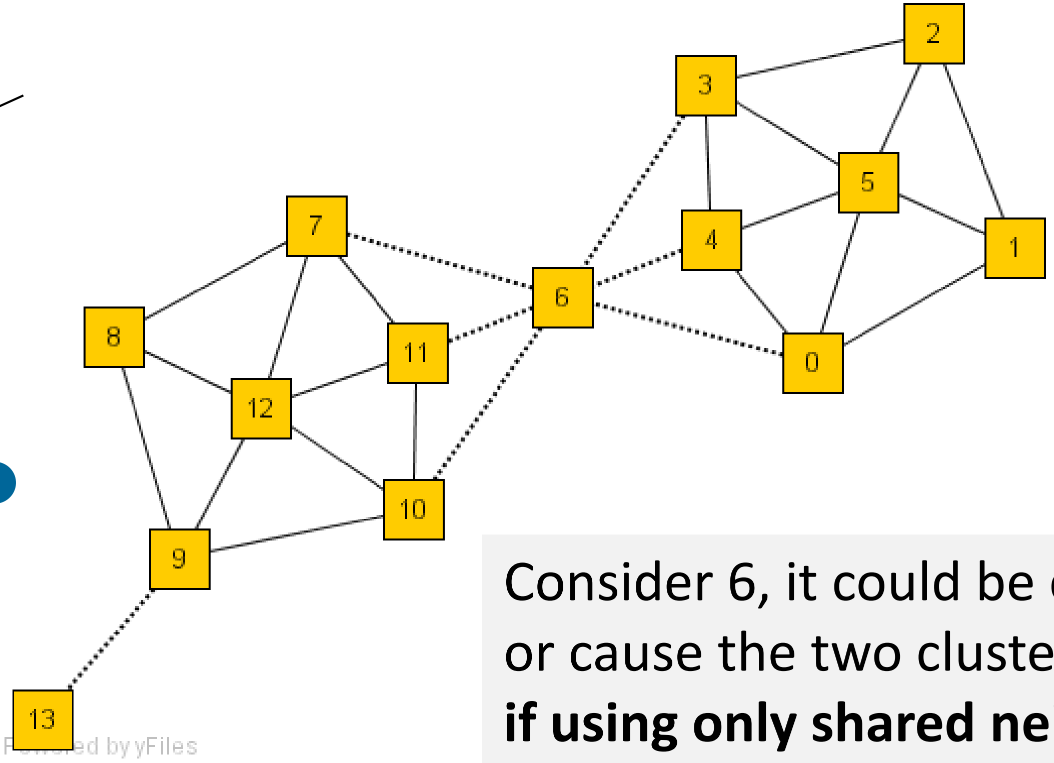
- Vertex structure
  - Define $\Gamma(v)$ as the immediate neighborhood of a vertex (i.e. the set of people an individual knows)

$$\Gamma(v) = \{w \in V \mid (v,w) \in E\} \cup \{v\}$$



# Structural Similarity





Consider 6, it could be clustered to either or cause the two clusters to merge
**if using only shared neighbors**

- **Normalize** the number of common neighbors by geometric mean of two neighborhoods' size
- Define structural similarity:

$$\sigma(v,w)=\frac{|\Gamma(v)\cap\Gamma(w)|}{\sqrt{|\Gamma(v)||\Gamma(w)|}}$$

# Basic Concepts (1/3)

- **$\varepsilon$-Neighborhood**

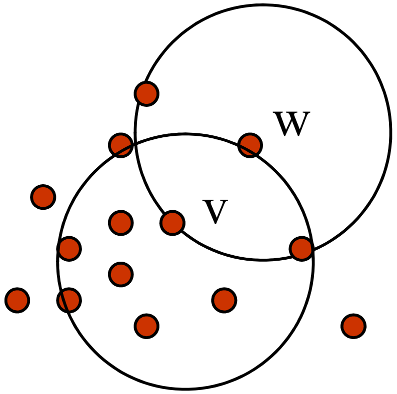
$$N_\varepsilon(v) = \{w \in \Gamma(v) \mid \sigma(v,w) \geq \varepsilon\}$$

- **Core$_{\varepsilon,\mu}$**

$$CORE_{\varepsilon,\mu}(v) \Leftrightarrow |N_\varepsilon(v)| \geq \mu$$

  - If a vertex is in $\varepsilon$-*neighborhood* of a core, it should be in the same cluster

- **Directly Structure Reachable**
  - w is directly structure reachable from v



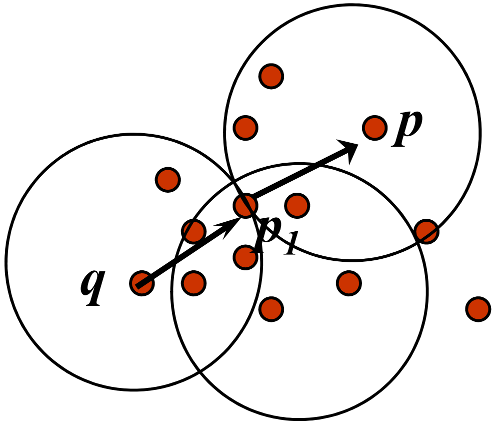
$$DirREACH_{\varepsilon,\mu}(v,w) \Leftrightarrow CORE_{\varepsilon,\mu}(v) \wedge w \in N_\varepsilon(v)$$

$\mu = 5$, $\varepsilon = 0.8$

# Basic Concepts (2/3)

- **Structure-reachable**

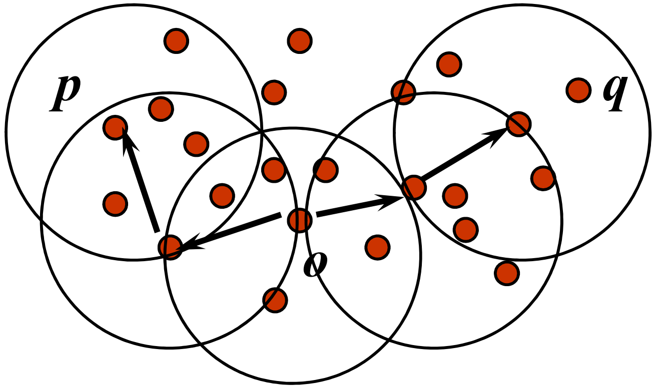
$$REACH_{\varepsilon,\mu}(v,w) \Leftrightarrow \exists v_1,\ldots v_n \in V : v_1 = v \wedge v_n = w \wedge \forall i \in \{1,\ldots,n-1\} : DirREACH_{\varepsilon,\mu}(v_i, v_{i+1}).$$



  - Transitive closure of direct structure reachability

- **Structure-connected**

$$CONNECT_{\varepsilon,\mu}(v,w) \Leftrightarrow \exists u \in V : REACH_{\varepsilon,\mu}(u,v) \wedge REACH_{\varepsilon,\mu}(u,w).$$



# Basic Concepts (3/3)

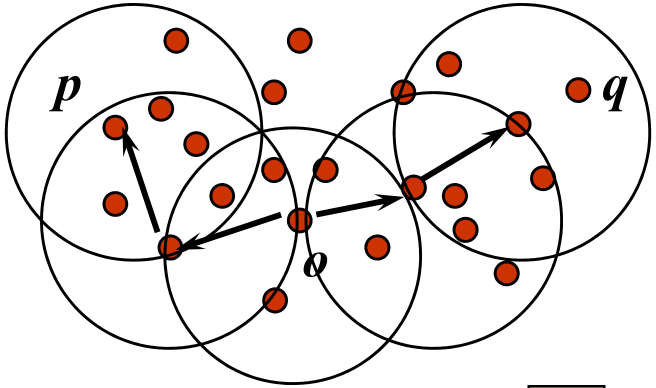
- **Structure-connected cluster**

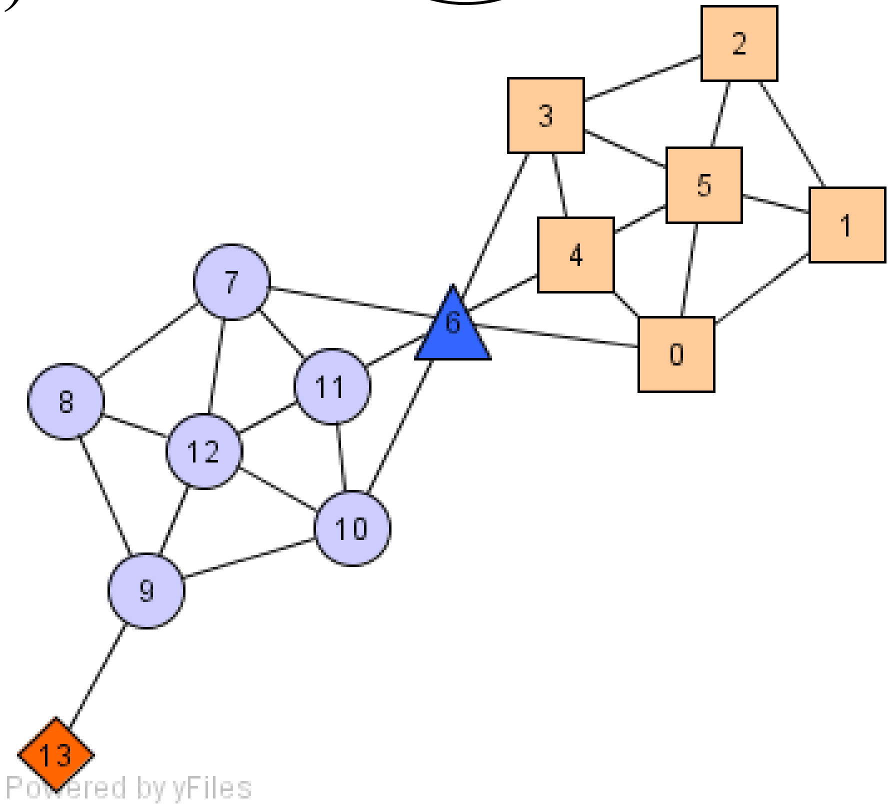
  $$CLUSTER_{\varepsilon,\mu}(C) \Leftrightarrow$$

  - Connectivity:

    $$\forall v, w \in C : CONNECT_{\varepsilon,\mu}(v,w)$$

- **Hub**
  - Not belong to any cluster
  - **Bridge to many clusters**
- **Outlier**
  - Not belong to any cluster
  - **Connect to few clusters**

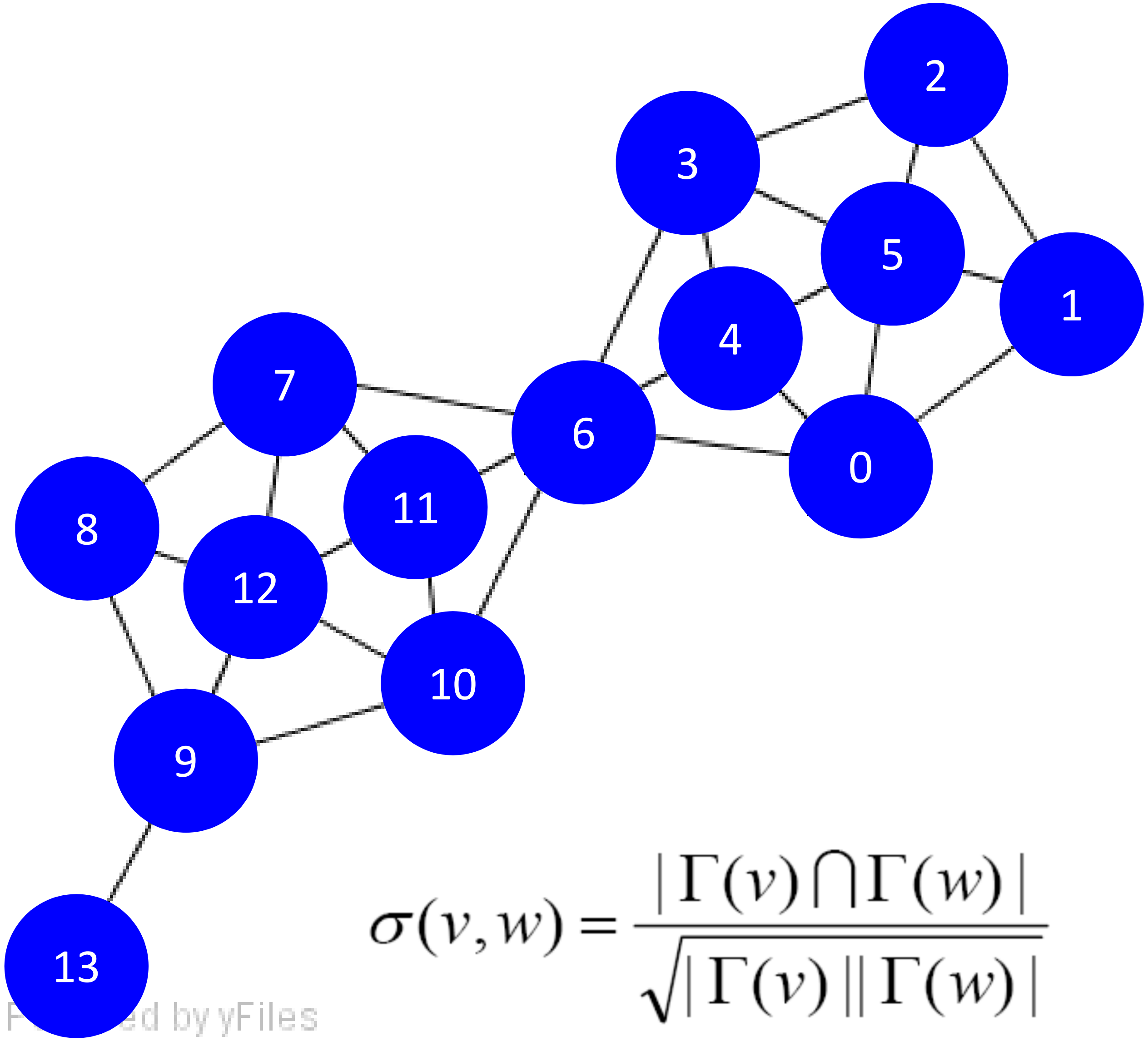




# SCAN Algorithm (1/13)

$\mu = 2$

$\varepsilon = 0.7$

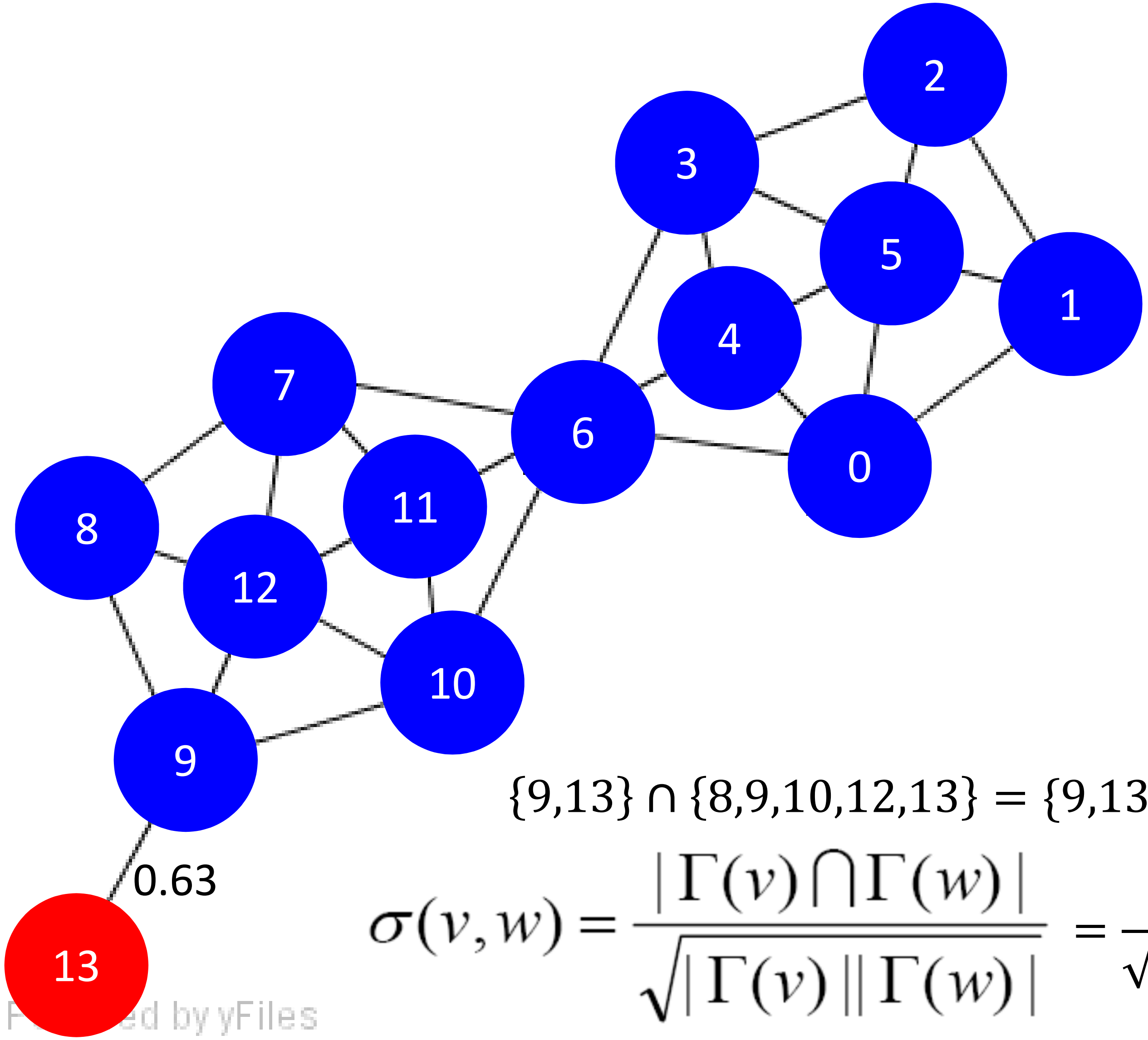


$$\sigma(v,w) = \frac{|\Gamma(v) \bigcap \Gamma(w)|}{\sqrt{|\Gamma(v)||\Gamma(w)|}}$$

# SCAN Algorithm (2/13)

$\mu = 2$

$\varepsilon = 0.7$



$\{9,13\} \cap \{8,9,10,12,13\} = \{9,13\}$

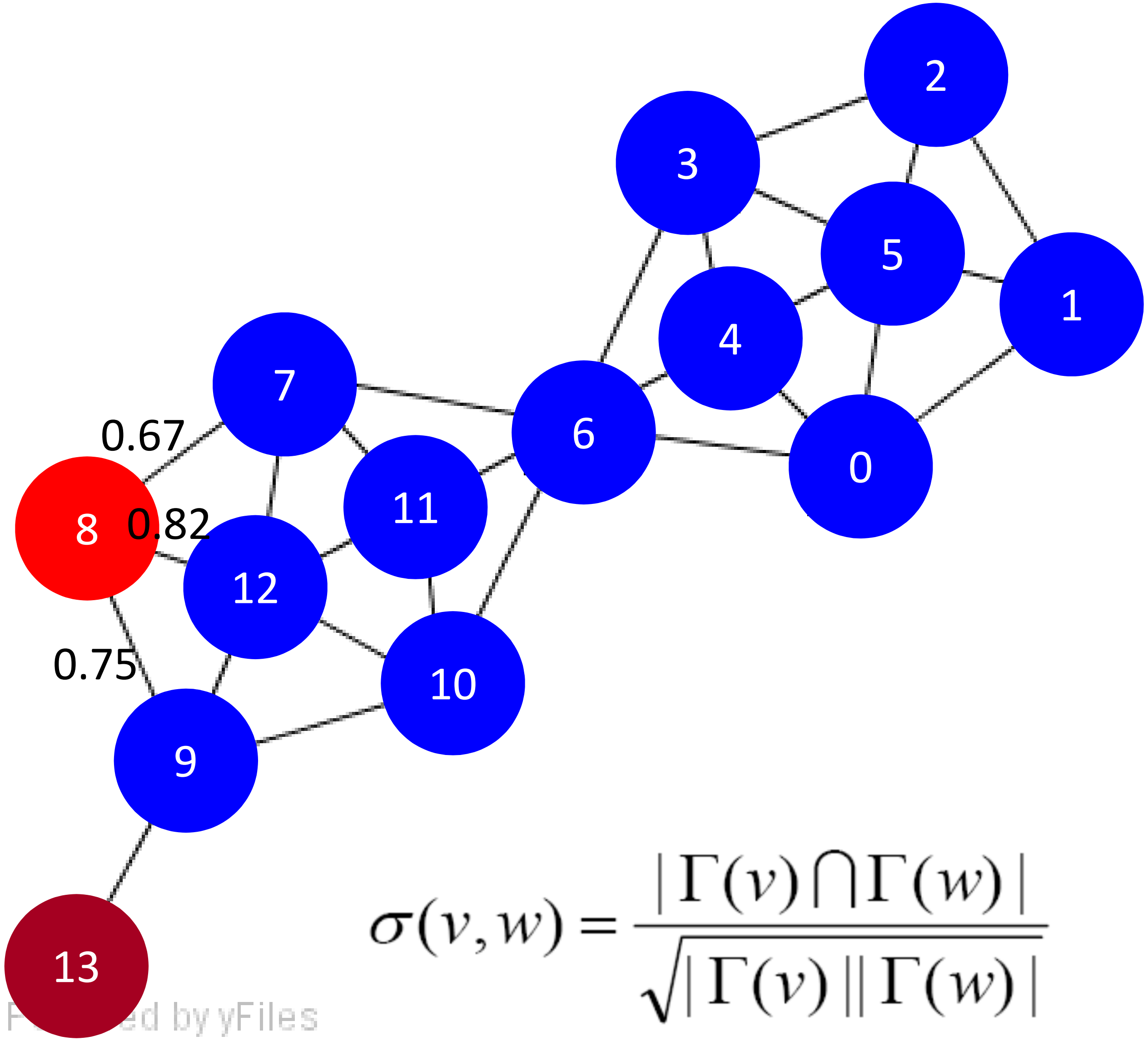
$$\sigma(v,w) = \frac{|\Gamma(v)\bigcap\Gamma(w)|}{\sqrt{|\Gamma(v)||\Gamma(w)|}} = \frac{2}{\sqrt{2\times 5}} = 0.63$$

$\{9,13\}$ $\{8,9,10,12,13\}$

# SCAN Algorithm (3/13)

$\mu = 2$

$\varepsilon = 0.7$

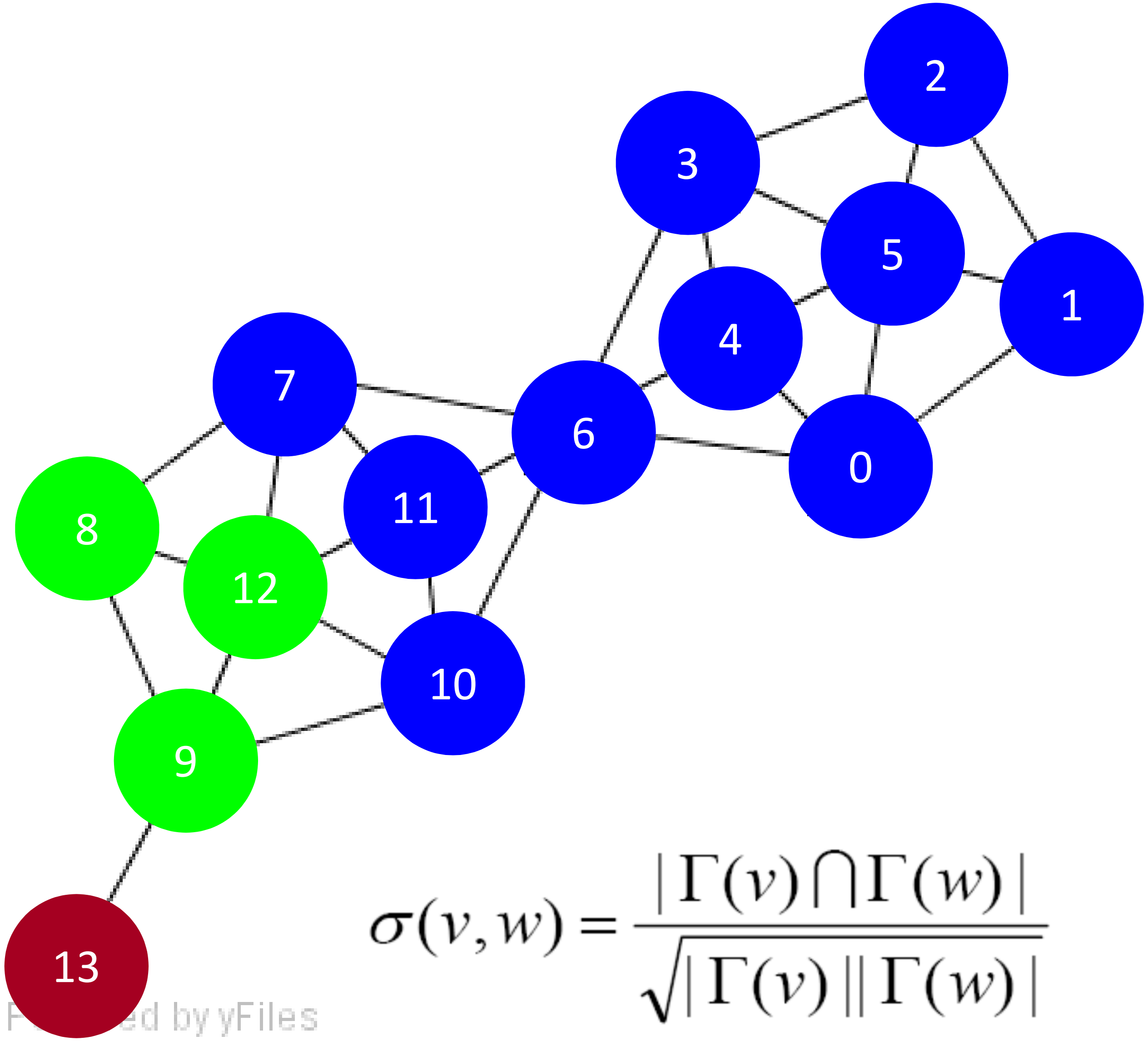


$$\sigma(v,w) = \frac{|\Gamma(v) \bigcap \Gamma(w)|}{\sqrt{|\Gamma(v)||\Gamma(w)|}}$$

# SCAN Algorithm (4/13)

$\mu = 2$

$\varepsilon = 0.7$

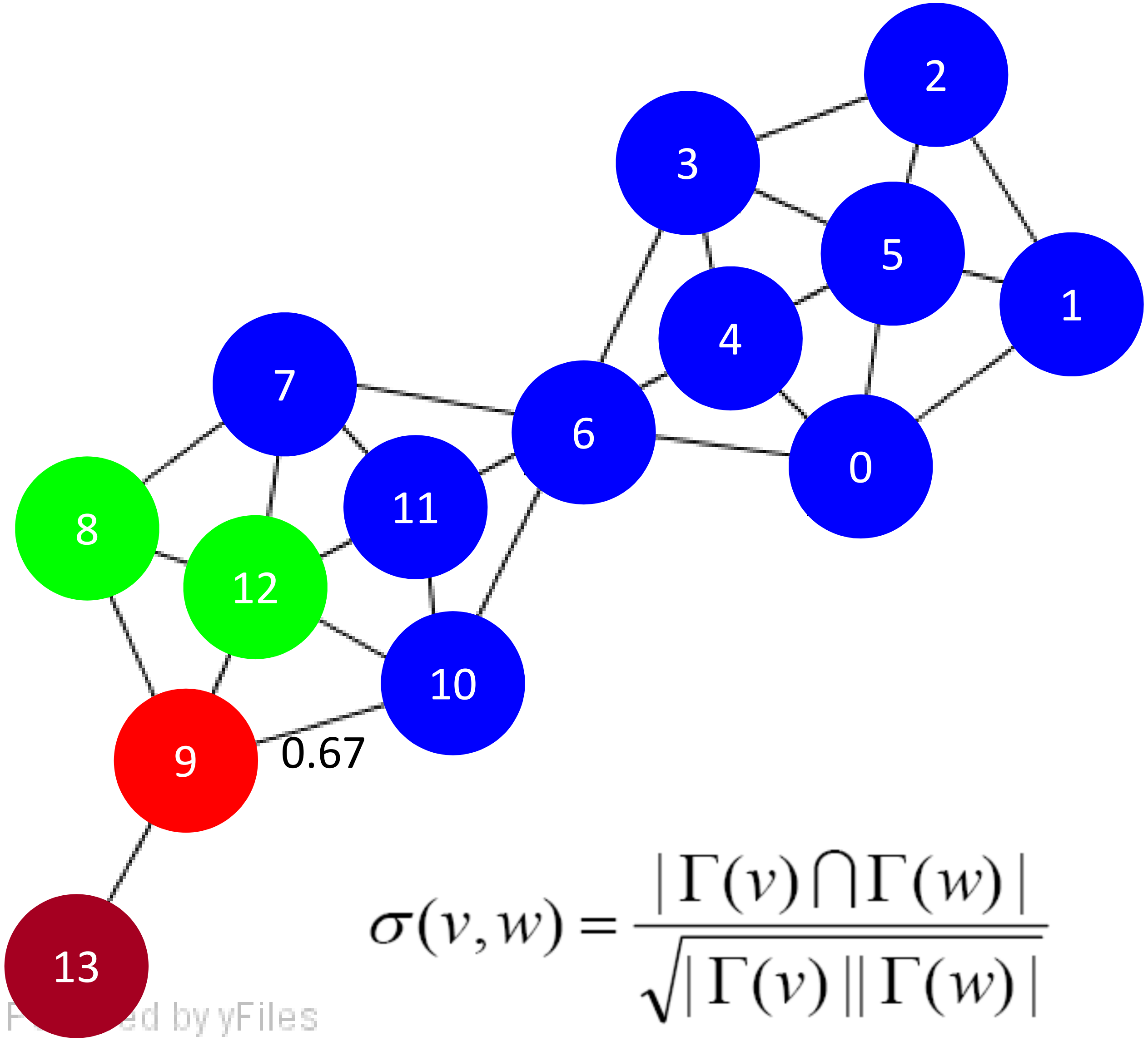


$$\sigma(v,w) = \frac{|\Gamma(v) \cap \Gamma(w)|}{\sqrt{|\Gamma(v)||\Gamma(w)|}}$$

# SCAN Algorithm (5/13)

$\mu = 2$

$\varepsilon = 0.7$

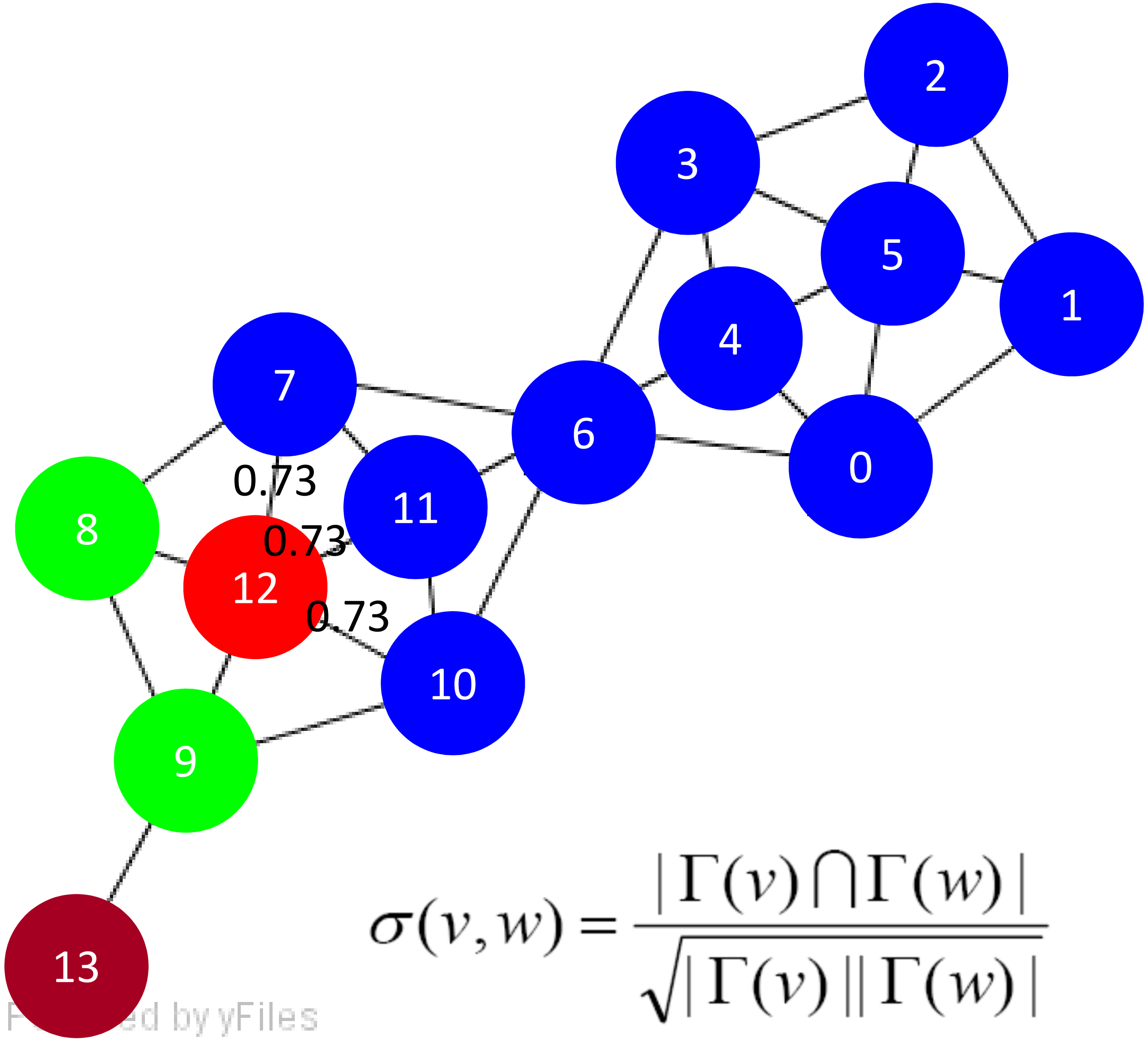


$$\sigma(v,w) = \frac{|\Gamma(v) \cap \Gamma(w)|}{\sqrt{|\Gamma(v)||\Gamma(w)|}}$$

# SCAN Algorithm (6/13)

$\mu = 2$

$\varepsilon = 0.7$

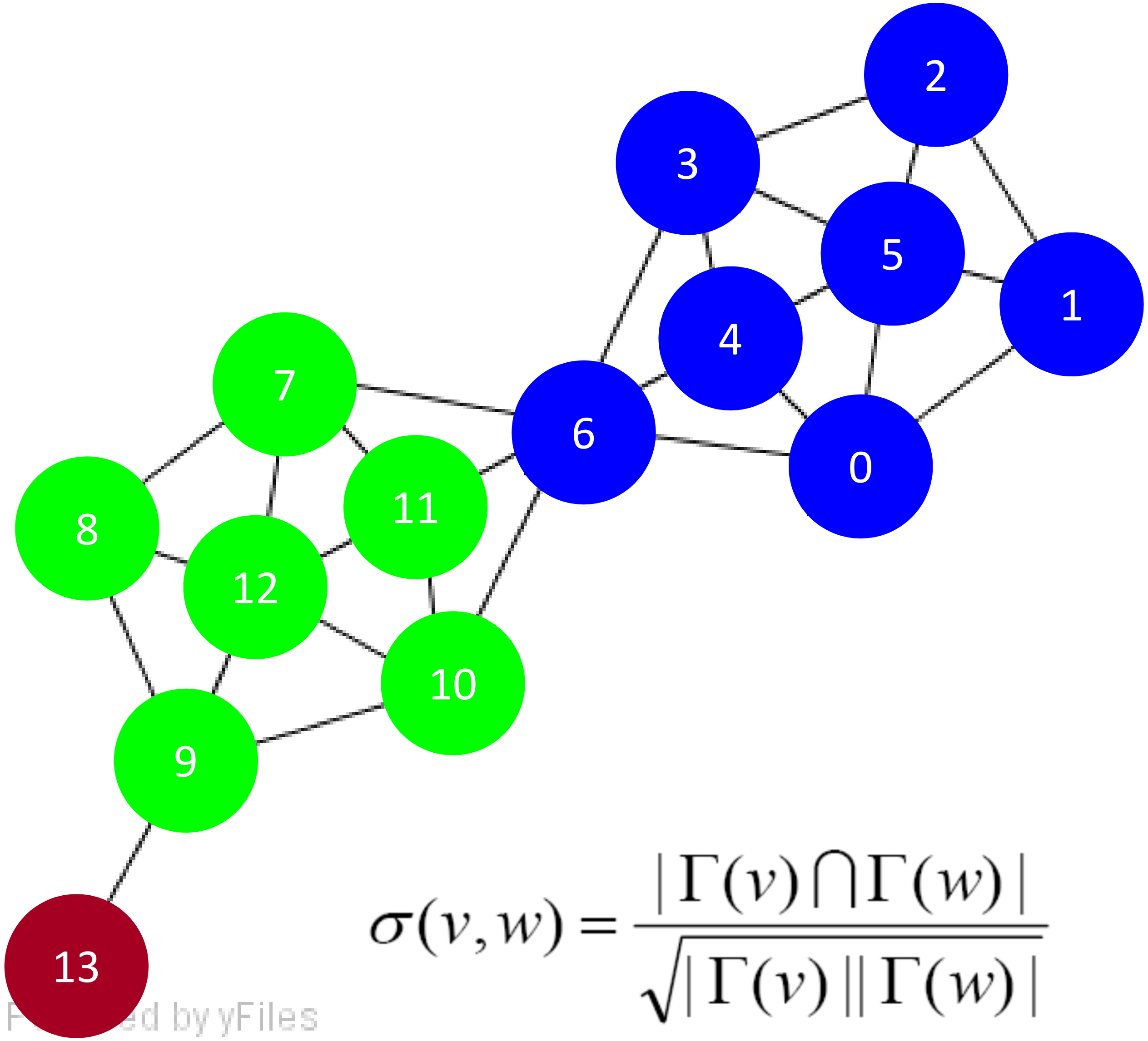


$$\sigma(v,w) = \frac{|\Gamma(v) \cap \Gamma(w)|}{\sqrt{|\Gamma(v)||\Gamma(w)|}}$$

# SCAN Algorithm (7/13)

$\mu = 2$

$\varepsilon = 0.7$

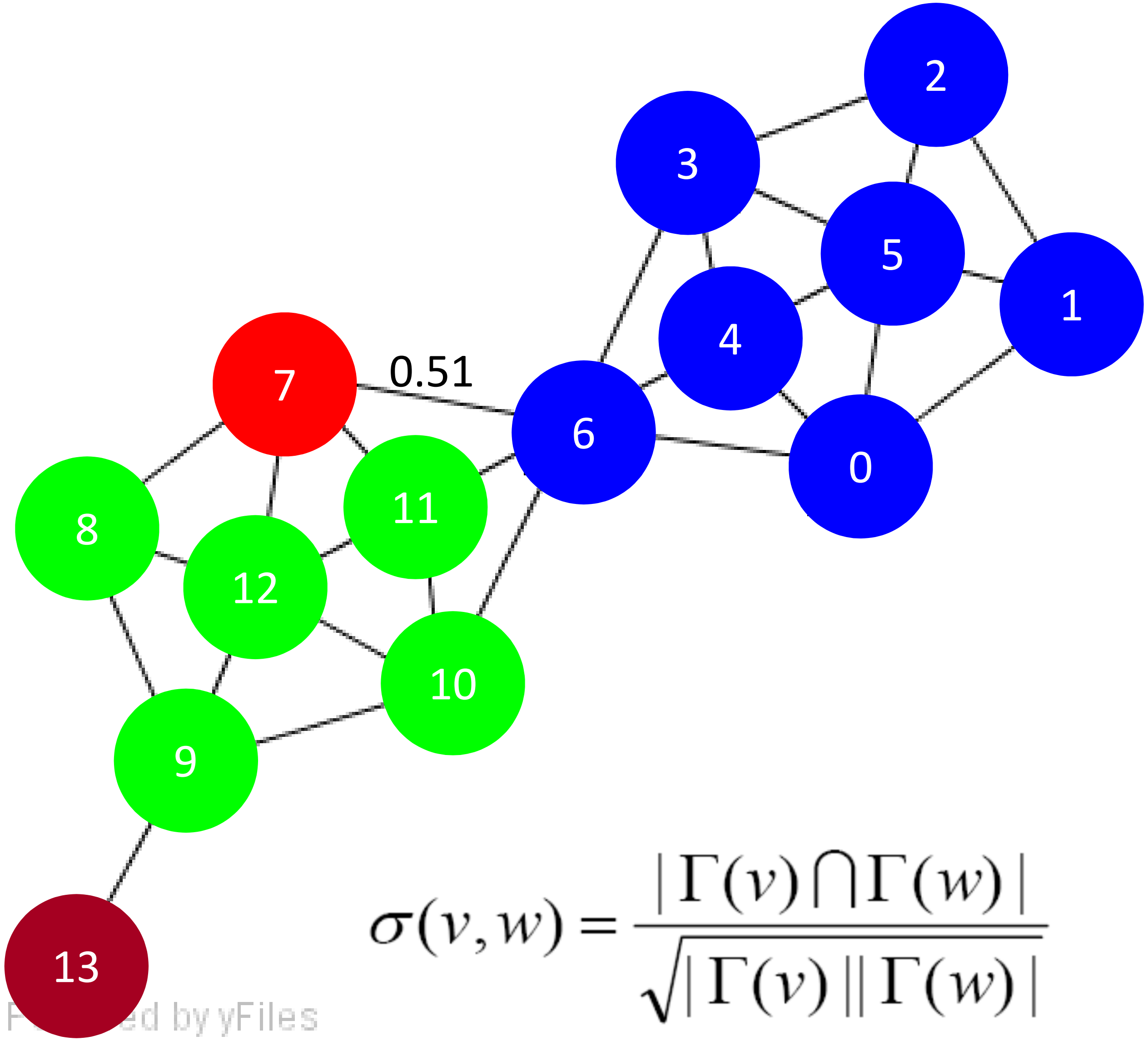


$$\sigma(v,w) = \frac{|\Gamma(v) \bigcap \Gamma(w)|}{\sqrt{|\Gamma(v)||\Gamma(w)|}}$$

# SCAN Algorithm (8/13)

$\mu = 2$

$\varepsilon = 0.7$

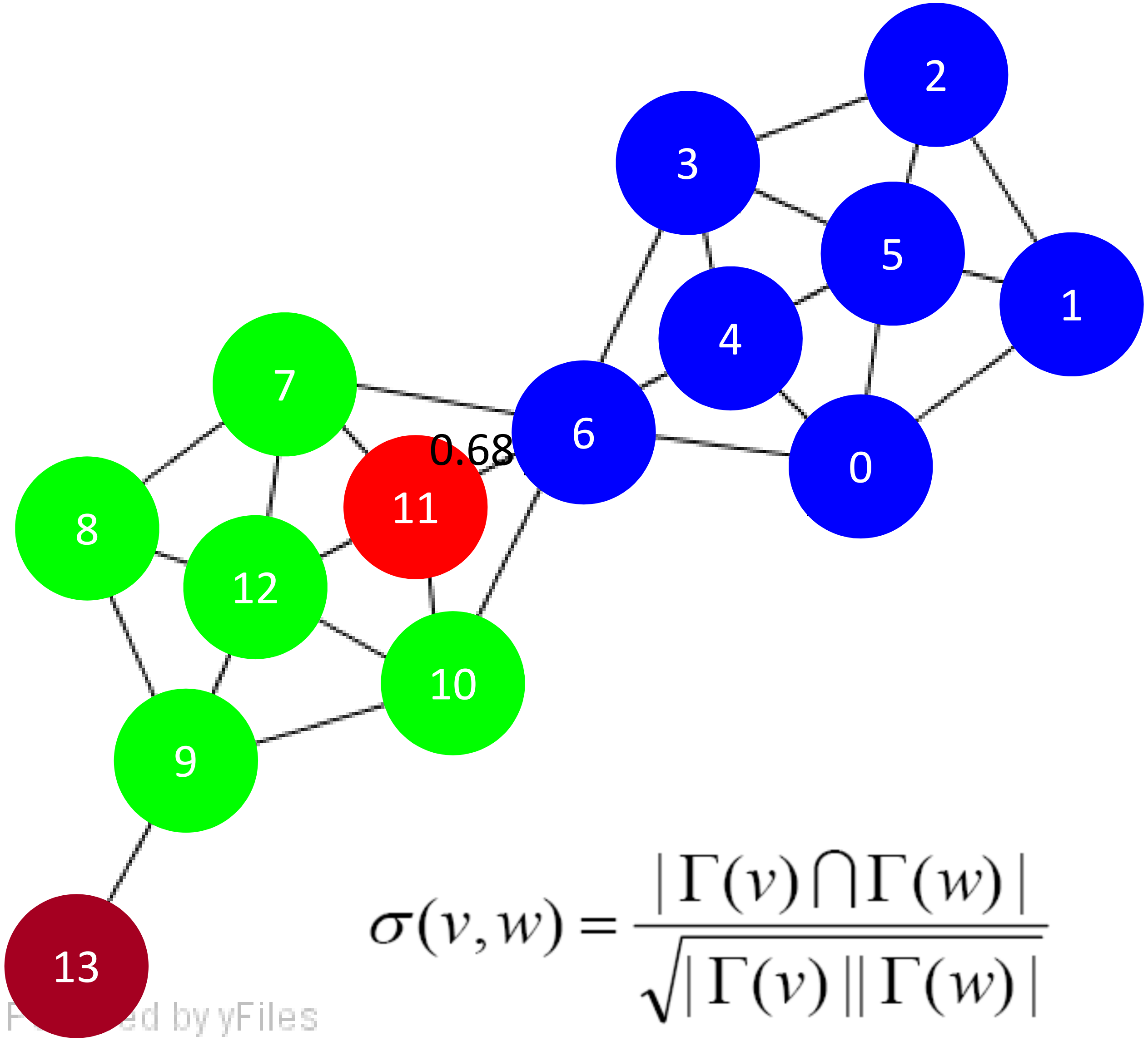


$$\sigma(v,w) = \frac{|\Gamma(v) \cap \Gamma(w)|}{\sqrt{|\Gamma(v)||\Gamma(w)|}}$$

# SCAN Algorithm (9/13)

$\mu = 2$

$\varepsilon = 0.7$

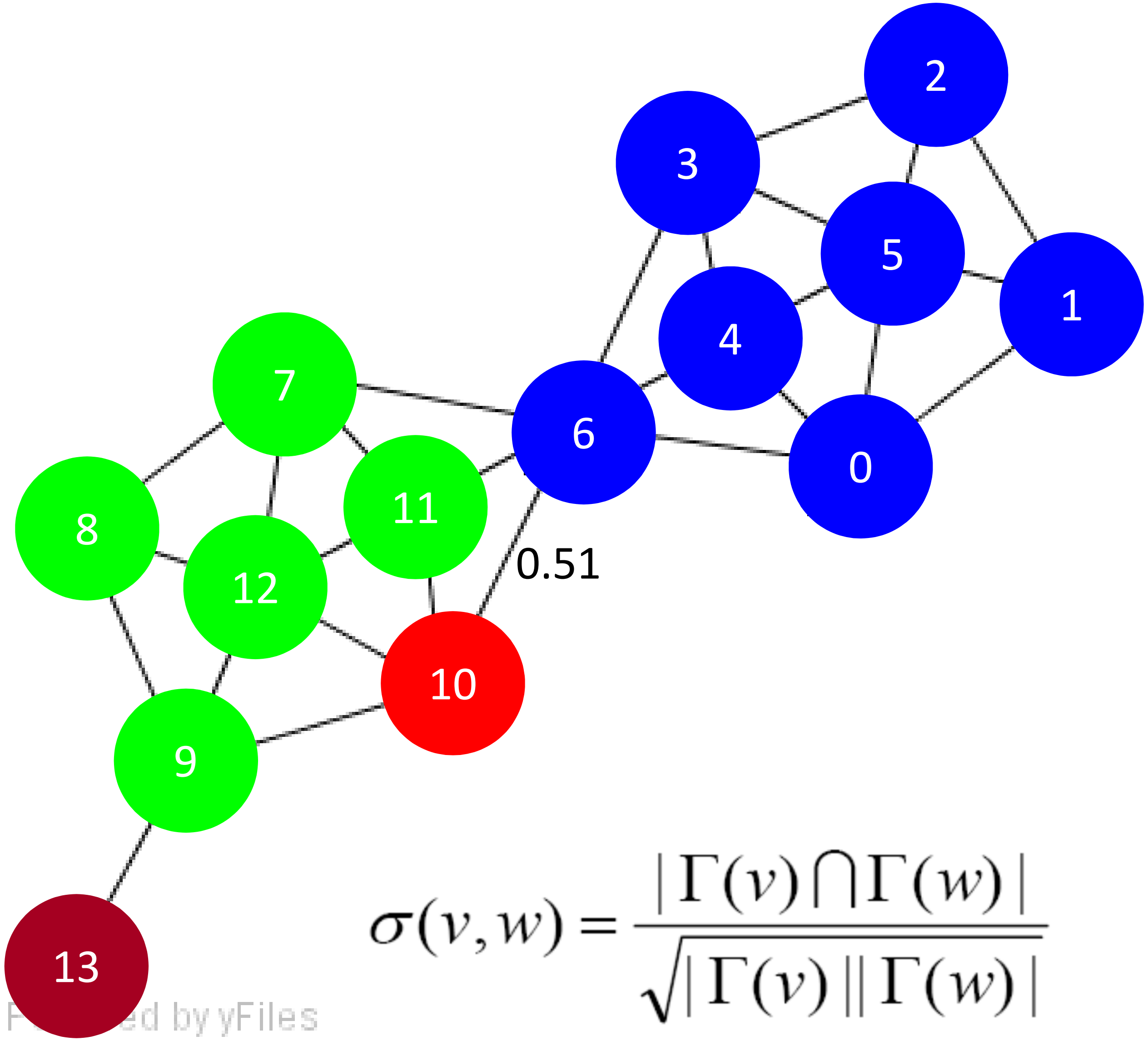


$$\sigma(v,w) = \frac{|\Gamma(v) \cap \Gamma(w)|}{\sqrt{|\Gamma(v)||\Gamma(w)|}}$$

# SCAN Algorithm (10/13)

$\mu = 2$

$\varepsilon = 0.7$

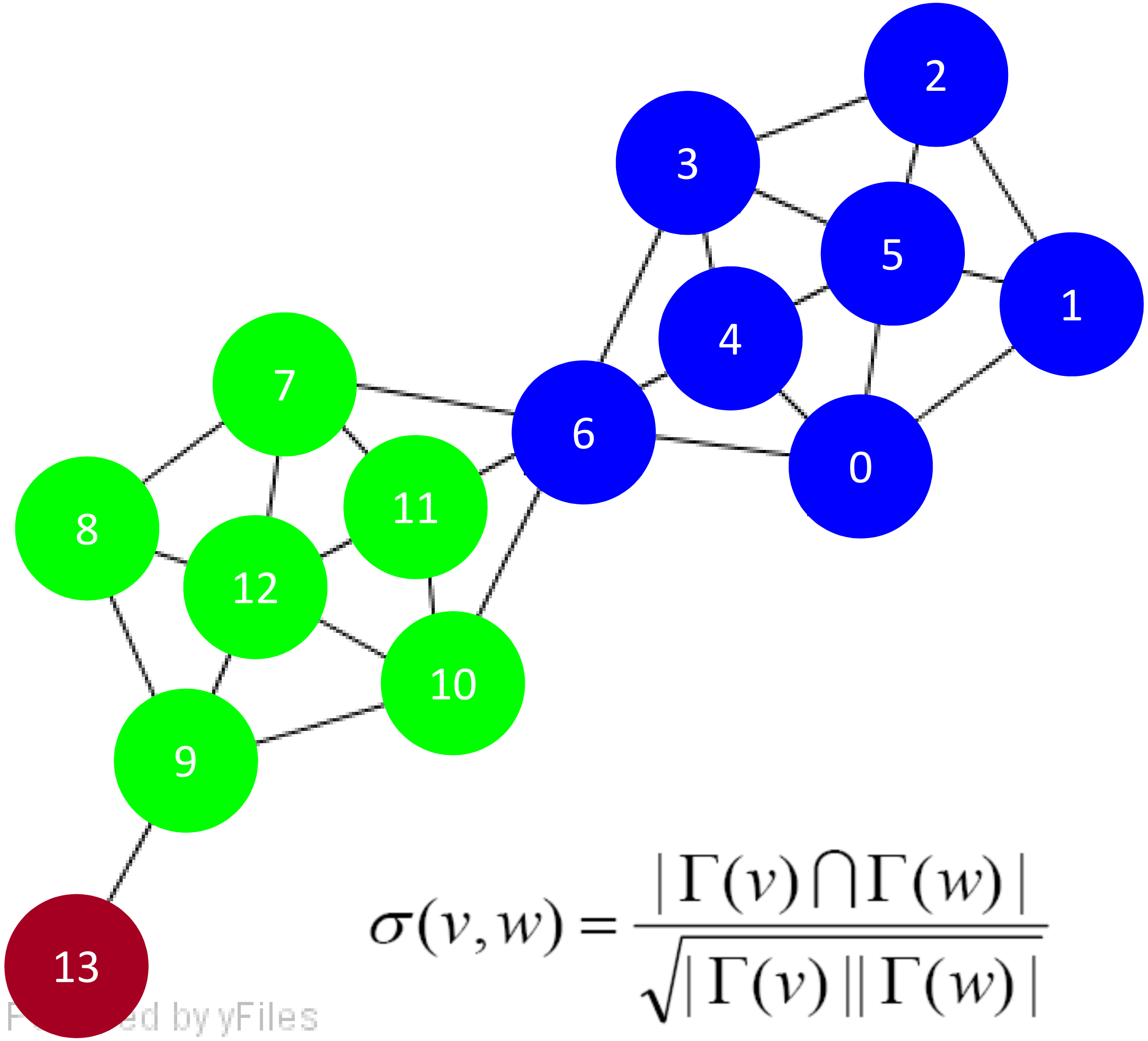


$$\sigma(v,w) = \frac{|\Gamma(v) \cap \Gamma(w)|}{\sqrt{|\Gamma(v)||\Gamma(w)|}}$$

# SCAN Algorithm (11/13)

$\mu = 2$

$\varepsilon = 0.7$

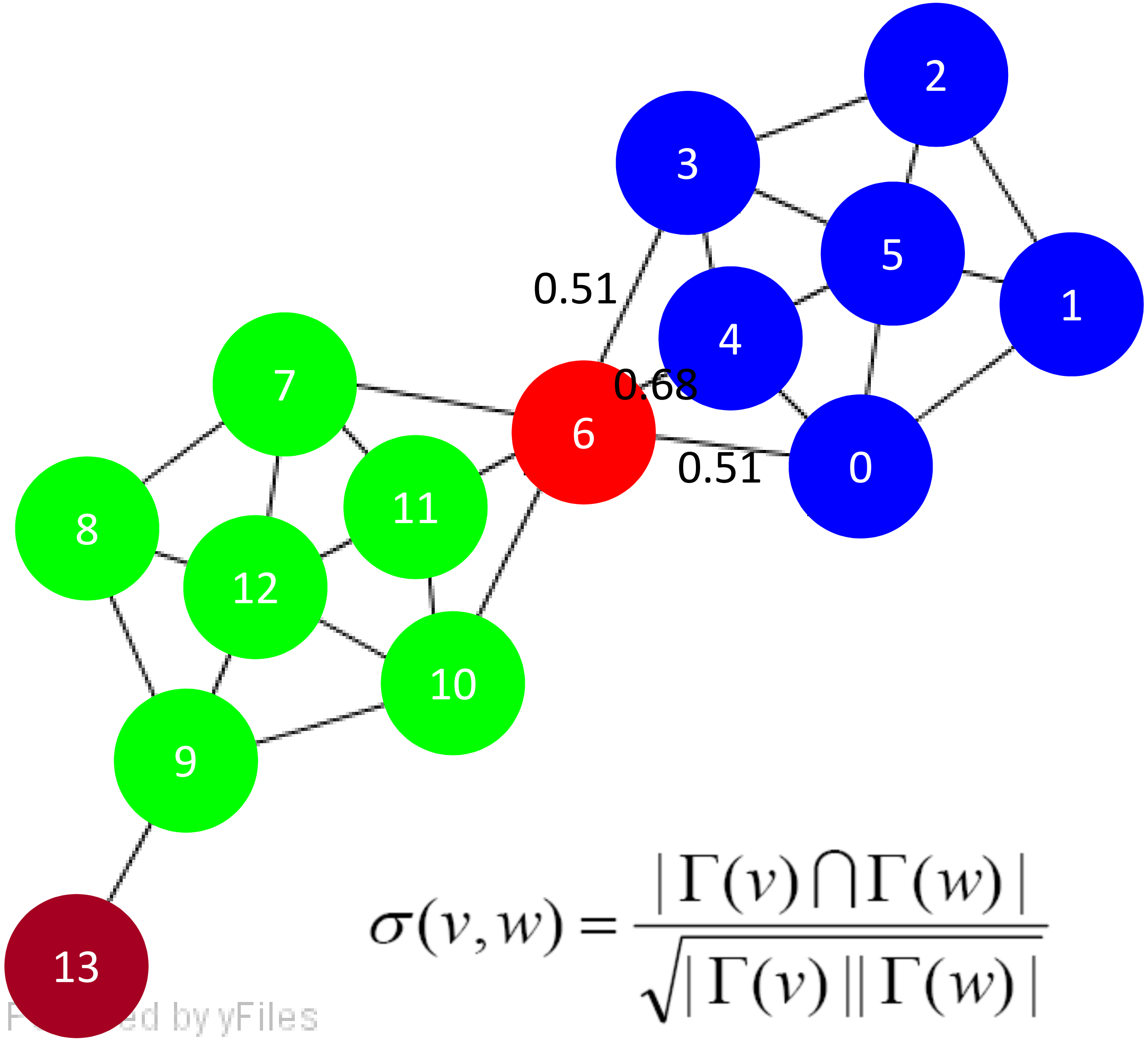


$$\sigma(v,w) = \frac{|\Gamma(v) \cap \Gamma(w)|}{\sqrt{|\Gamma(v)||\Gamma(w)|}}$$

# SCAN Algorithm (12/13)

$\mu = 2$

$\varepsilon = 0.7$

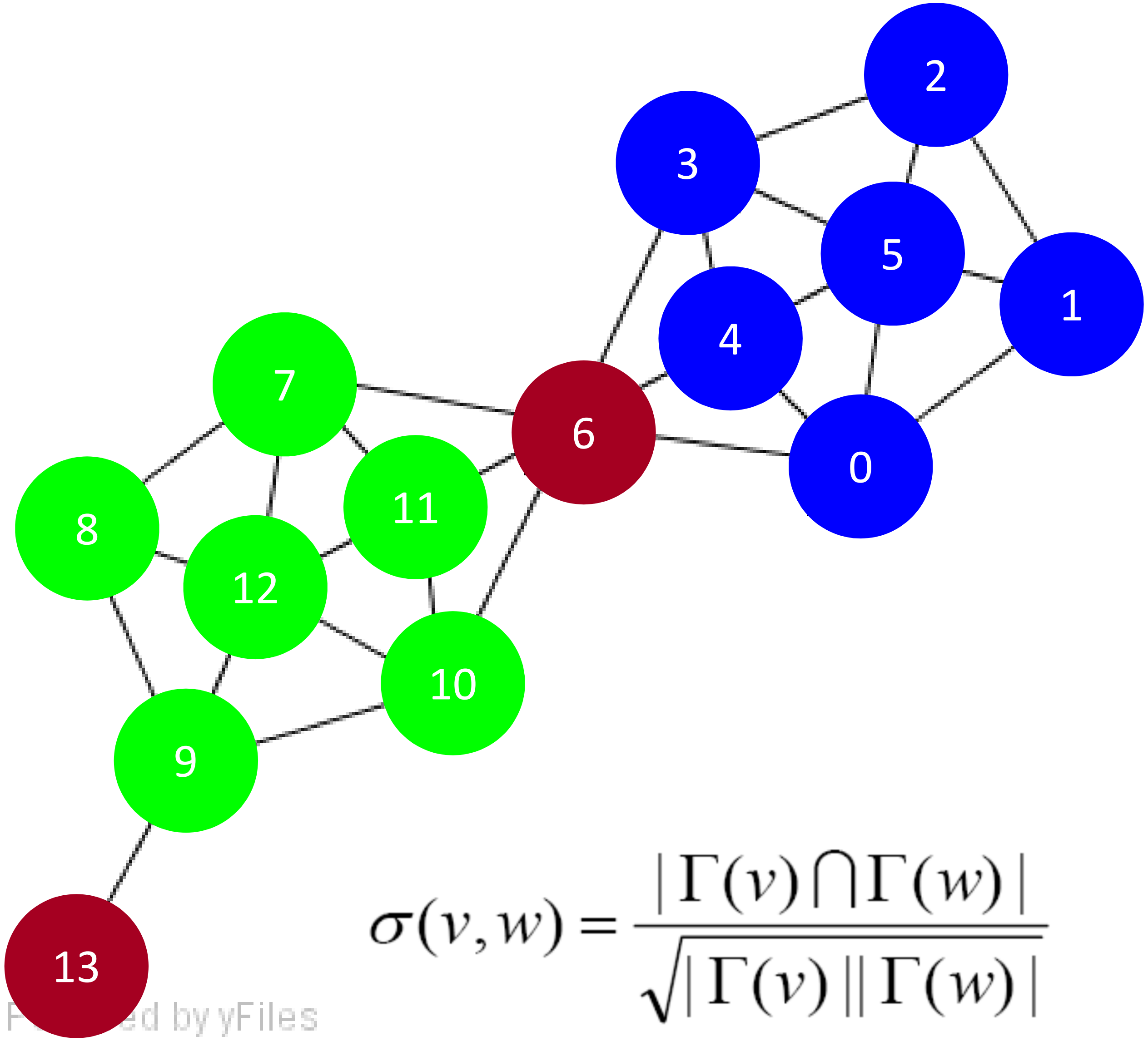


$$\sigma(v,w) = \frac{|\Gamma(v) \cap \Gamma(w)|}{\sqrt{|\Gamma(v)||\Gamma(w)|}}$$

# SCAN Algorithm (13/13)

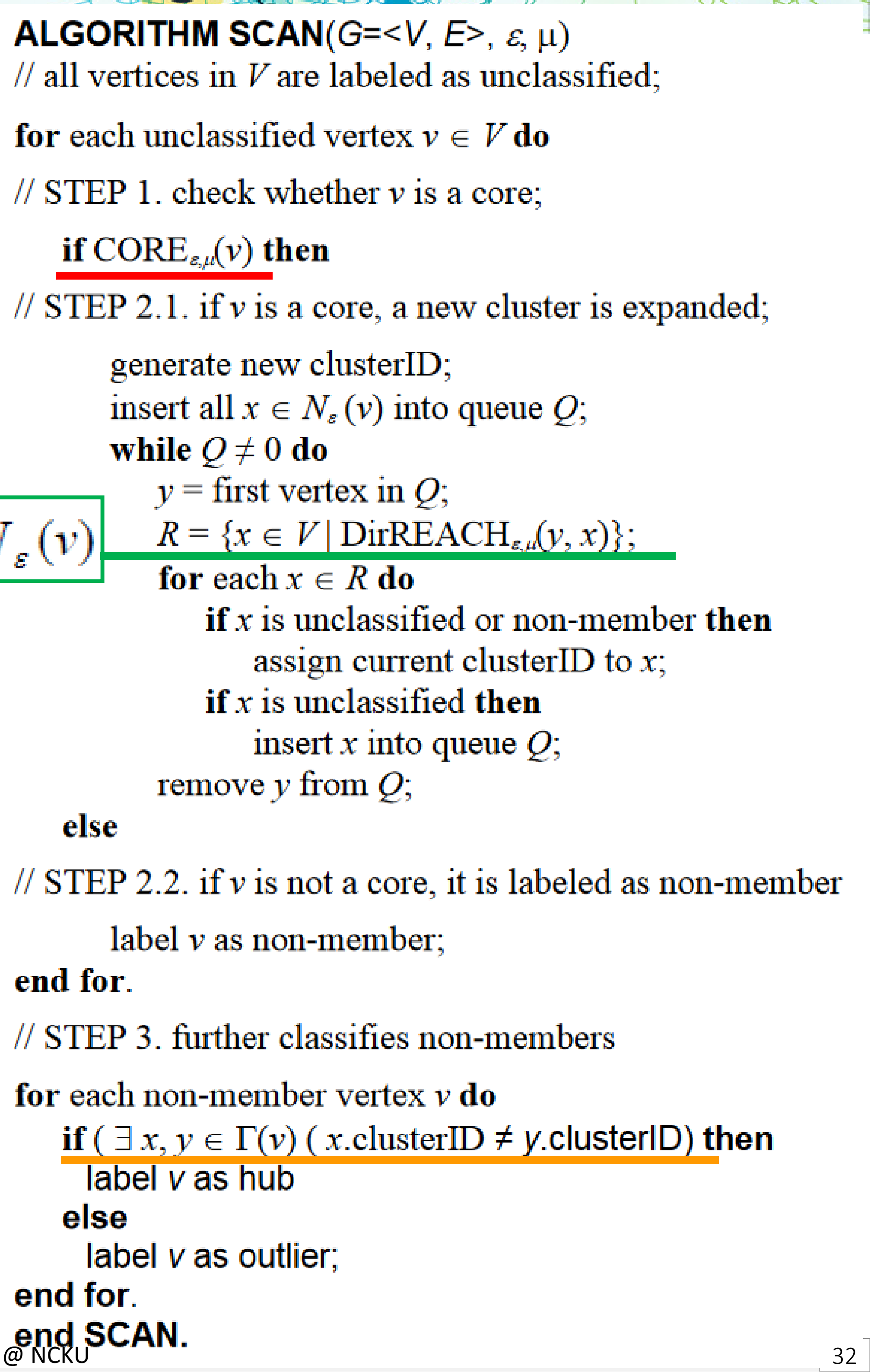
$\mu = 2$

$\varepsilon = 0.7$



$$\sigma(v,w) = \frac{|\Gamma(v) \bigcap \Gamma(w)|}{\sqrt{|\Gamma(v)||\Gamma(w)|}}$$

# SCAN Algorithm



```
ALGORITHM SCAN(G=<V, E>, ε, μ)
// all vertices in V are labeled as unclassified;
for each unclassified vertex v ∈ V do
// STEP 1. check whether v is a core;
    if CORE_{ε,μ}(v) then
// STEP 2.1. if v is a core, a new cluster is expanded;
        generate new clusterID;
        insert all x ∈ N_ε(v) into queue Q;
        while Q ≠ 0 do
            y = first vertex in Q;
            R = {x ∈ V | DirREACH_{ε,μ}(y, x)};
            for each x ∈ R do
                if x is unclassified or non-member then
                    assign current clusterID to x;
                if x is unclassified then
                    insert x into queue Q;
            remove y from Q;
    else
// STEP 2.2. if v is not a core, it is labeled as non-member
        label v as non-member;
end for.
// STEP 3. further classifies non-members
for each non-member vertex v do
    if ( ∃ x, y ∈ Γ(v) ( x.clusterID ≠ y.clusterID) then
        label v as hub
    else
        label v as outlier;
end for.
end SCAN.
```

$$DirREACH_{\varepsilon,\mu}(v,w) \Leftrightarrow CORE_{\varepsilon,\mu}(v) \wedge w \in N_{\varepsilon}(v)$$