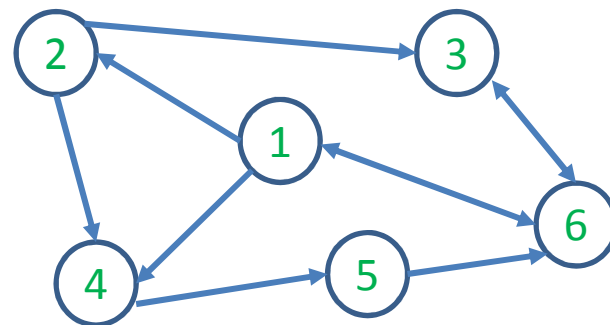
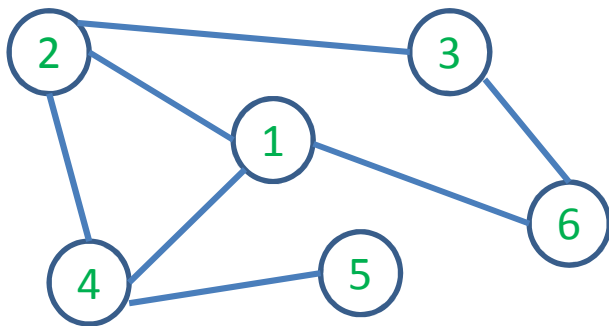


# Graph Data

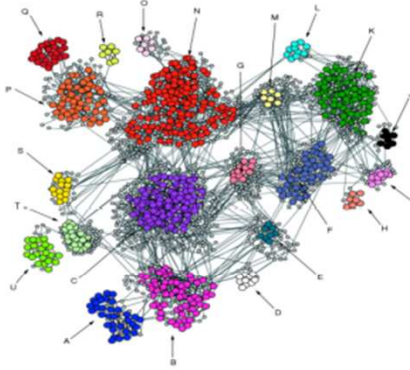
# What is a Graph

- Graphs, denoted as  $G = (V, E)$ , are structures formed by a set of vertices,  $V$  (also called nodes) and a set of edges,  $E = \{v, w\}$ , that are connections between pairs of vertices.



# Graphs are everywhere

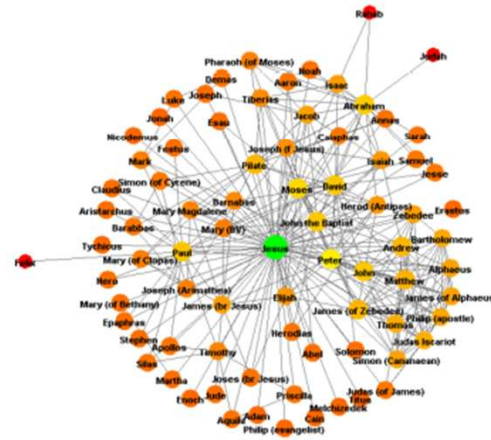
Magwene et al. *Genome Biology* 2004 5:R100



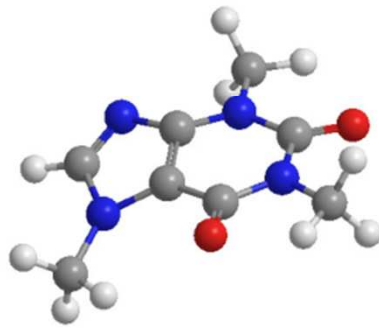
Co-expression Network



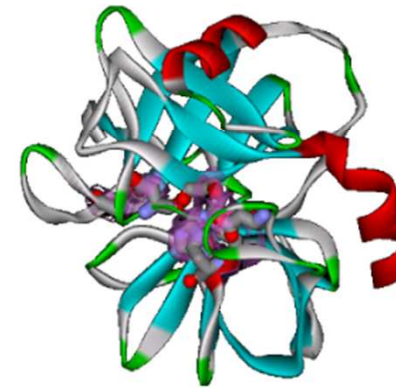
Program Flow



Social Network



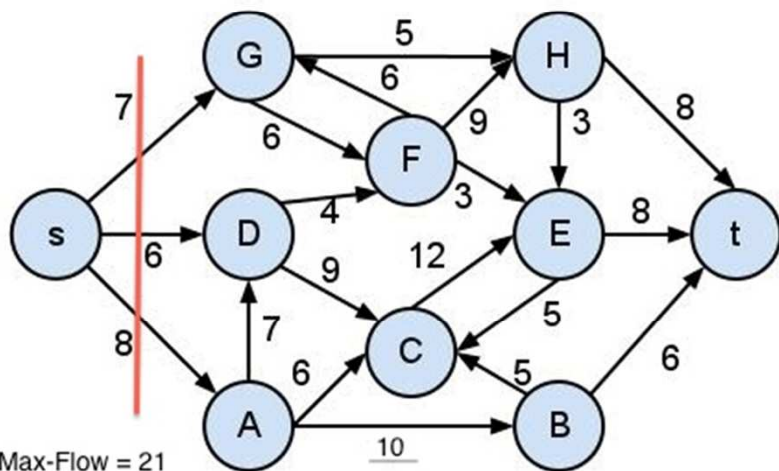
Chemical Compound



Protein Structure

# Basic Concepts

- The order of the graph  $G$ ,  $n = |V|$
- The size of the graph  $G$ ,  $m = |E|$
- A graph is planar if it can be drawn in a plane without any of the edges crossing



Max-Flow = 21  
 Min-Cut = 21

Image source: <http://people.seas.harvard.edu/~joshlee/>

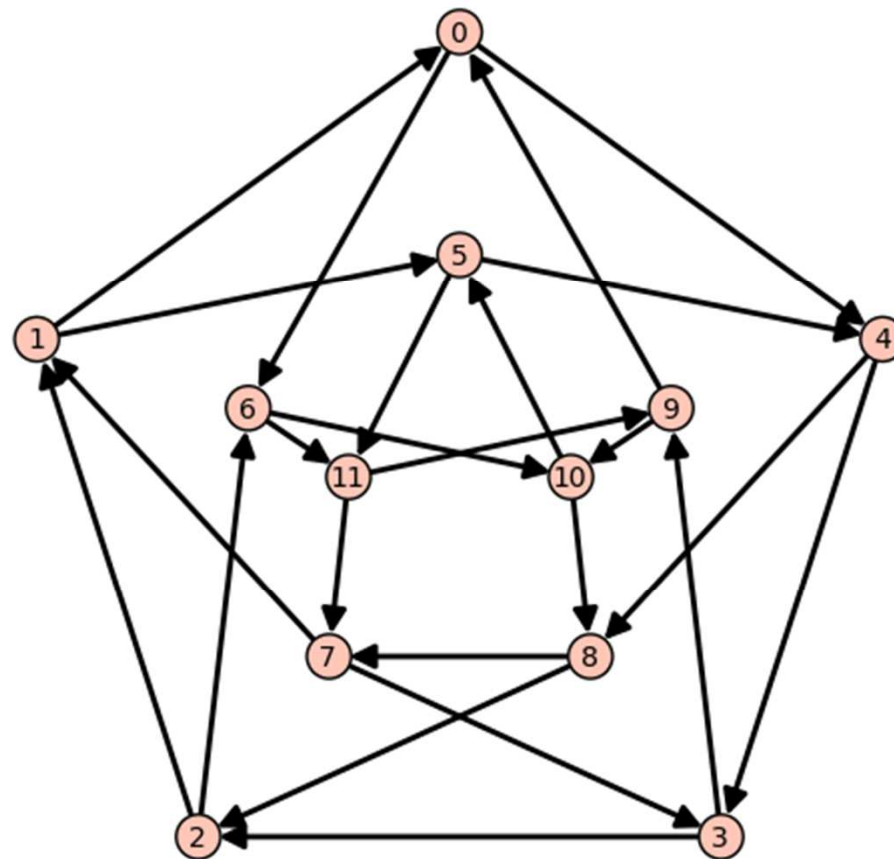
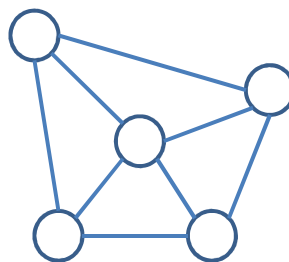
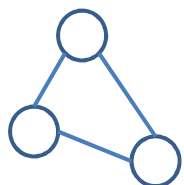
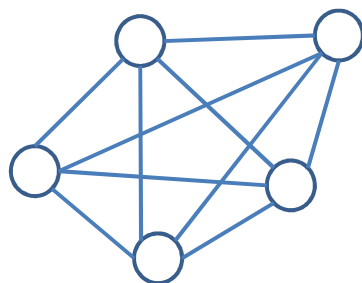
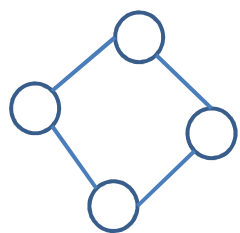


Image source:  
[http://www.sagemath.org/doc/thematic\\_tutorials/linear\\_programming.html/](http://www.sagemath.org/doc/thematic_tutorials/linear_programming.html/)

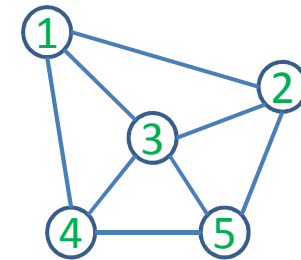
# Basic Concepts

- The order of the graph  $G$ ,  $n = |V|$
- The size of the graph  $G$ ,  $m = |E|$
- A graph is planar if it can be drawn in a plane without any of the edges crossing
- The density of the graph  $G$ ,  $\frac{m}{\binom{n}{2}}$
- A graph of density 1 is called complete



# Basic Concepts (II)

- The adjacency matrix  $A_G$  of a given graph  $G = (V, E)$  of order  $n$  is an  $n \times n$  matrix  $A_G = (a_{u,v})$  where  $a_{u,v} = \begin{cases} 1, & \text{if } \{u, v\} \in E \\ 0, & \text{otherwise} \end{cases}$
- The number of edges incident on a given vertex  $v$  is the degree of  $v$ , denoted by  $\deg(v)$ .
- A graph is regular if all of the vertices have the same degree



- Degree matrix  $D = \begin{bmatrix} \deg(v_0) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \deg(v_{n-1}) \end{bmatrix}$



# Basic Concepts (III)

- A partition of the vertices  $V$  of a graph  $G = (V, E)$  into two nonempty sets  $S$  and  $V \setminus S$  is called a cut and is denoted by  $(S, V \setminus S)$ .
  - A cut is uniquely identified by defining a set  $S$ ; therefore, *any subset of  $V$  can be called a cut.*
    - Typically  $|S| \leq \frac{n}{2}$
  - The **cut size** is the number of edges that connect vertices in  $S$  to vertices in  $V \setminus S$ .
  - If the edges have weights, the cut size is re-defined as the sum of the weights of the edges crossing the cut.
  - The sum of degrees in a cut  $S$  is defined as

$$\deg(S) = \sum_{v \in S} \deg(v)$$

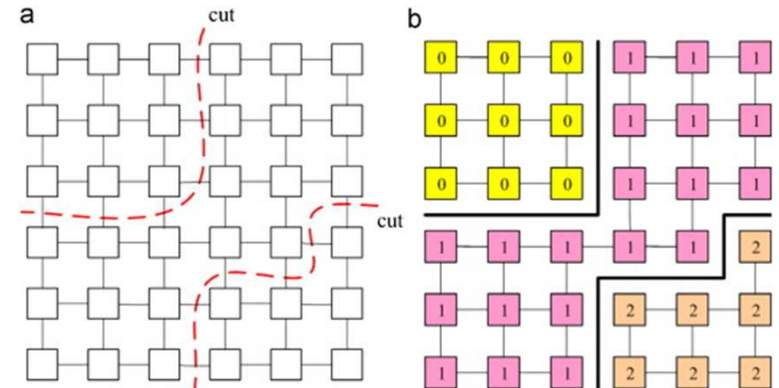
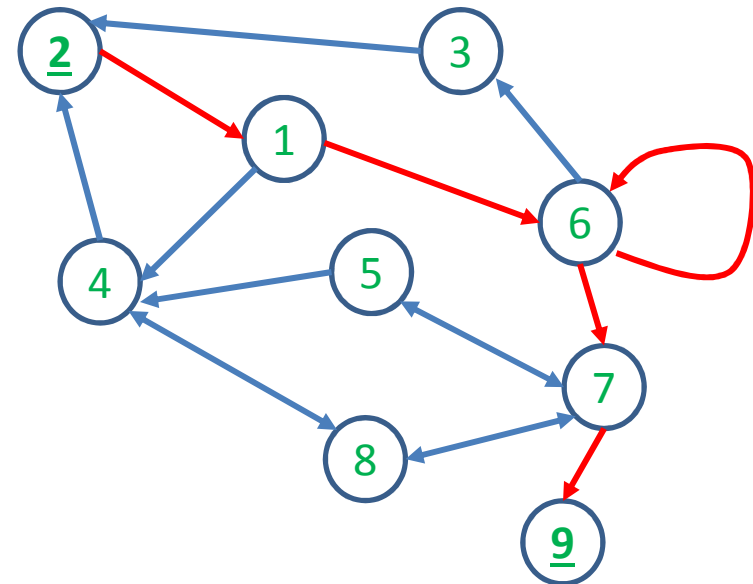
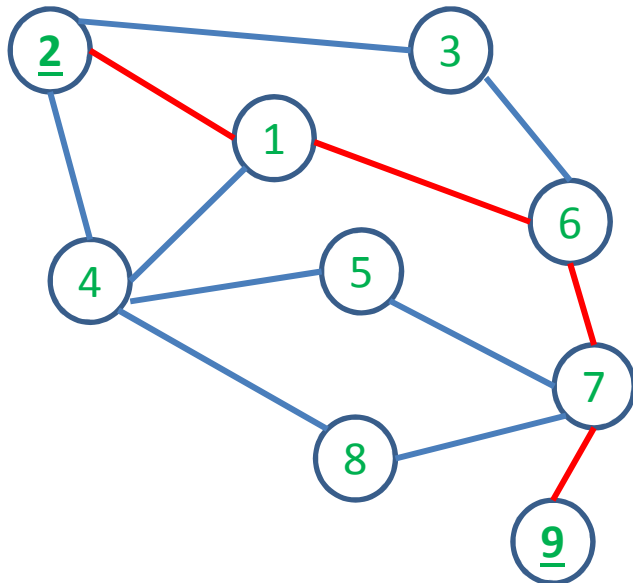


Image source:

<http://www.sciencedirect.com/science/article/pii/S0031320312004219>

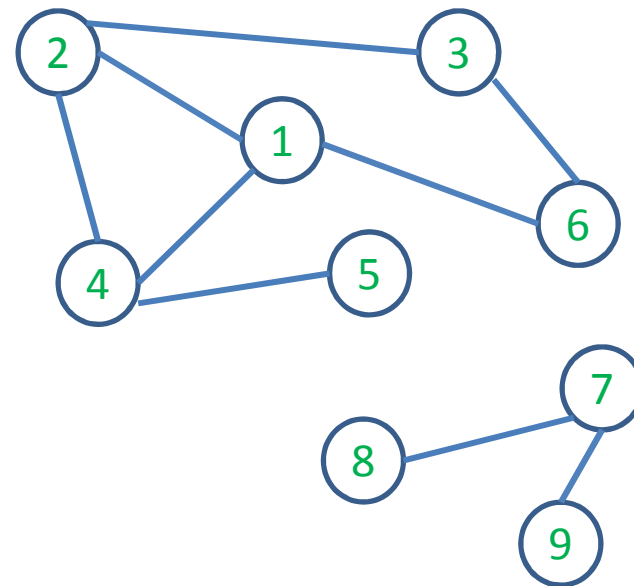
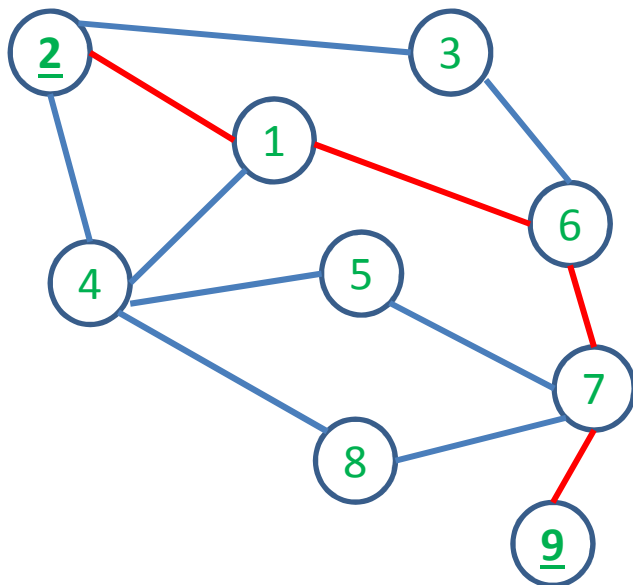
# Basic Concepts (IV)

- A path from  $v$  to  $u$  in a graph  $G = (V, E)$  is a sequence of edges in  $E$  starting at vertex  $v_0 = v$  and ending at vertex  $v_{k+1} = u$ .
- The path is simple if no vertex is repeated



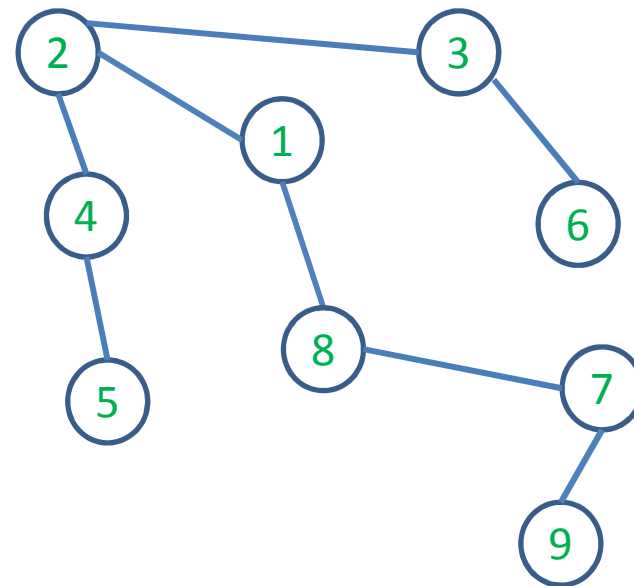
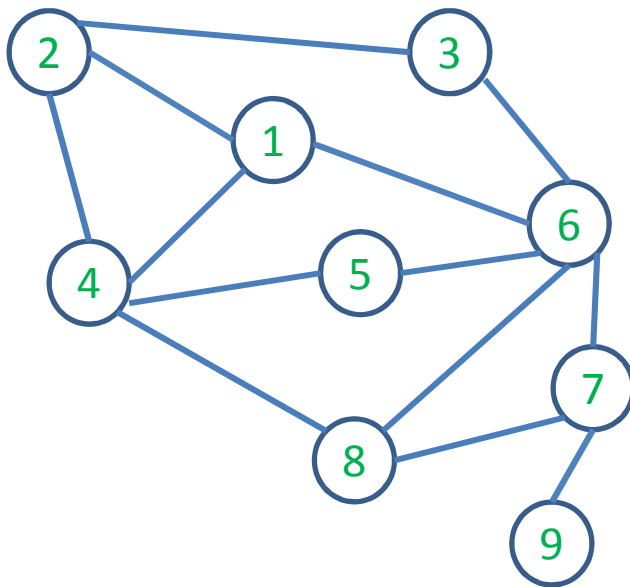
# Basic Concepts (IV)

- The length of the path is the number of edges on it
- The distance between is the shortest path connecting them
- A graph is connected if there exist paths between all pairs of vertices; otherwise, it is disconnected.
- The *minimum* number of edges that would need to be removed from G in order to make the graph disconnected is the edge-connectivity of the graph.



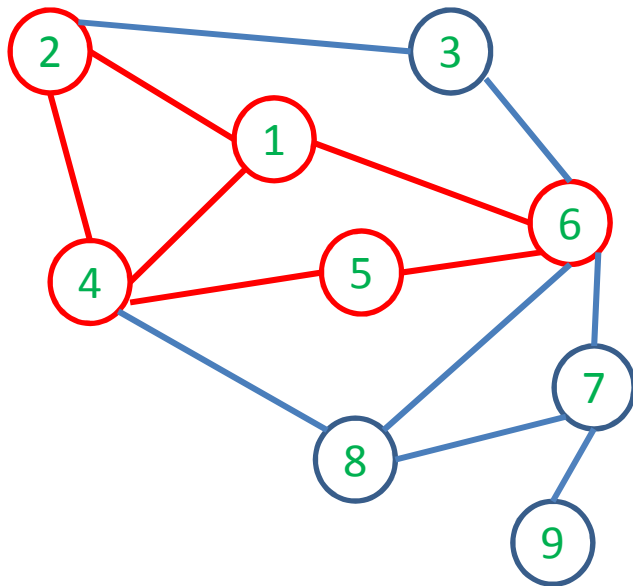
# Basic Concepts (V)

- A cycle is a simple path that begins and ends at the same vertex.
- A graph that contains no cycle is acyclic and is also called forest.
- A connected forest is called a tree.



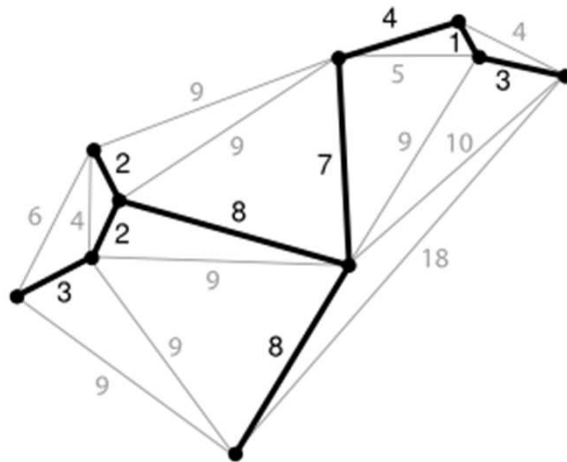
# Basic Concepts (VI)

- A subgraph  $G_S = (S, E_S)$  of  $G = (V, E)$  is composed of a set of vertices  $S \subseteq V$  and a set of edges  $E_S \subseteq E$ .  $G$  is then a supergraph of  $G_S$ .



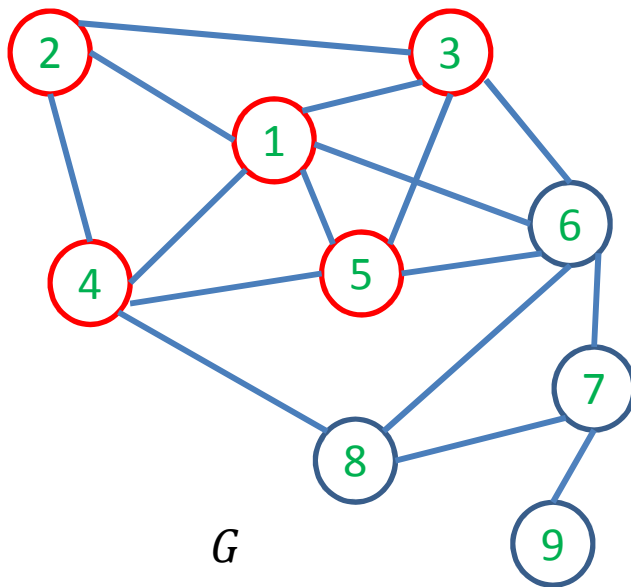
# Basic Concepts (VI)

- A connected acyclic subgraph that includes all vertices in  $V$  is called a spanning tree of  $G$ .
  - A spanning tree has exactly  $n - 1$  edges
  - If the edges have weights, the spanning tree with smallest total weights is called the minimum spanning tree (there may exist several of them)



# Basic Concepts (VII)

- An induced subgraph of a graph  $G = (V, E)$  is the graph with the vertex set  $S \subseteq V$  with an edge set  $E(S)$  that includes all such edges  $\{v, u\}$  in  $E$  with both of the vertices  $v$  and  $u$  included in the set  $S$
- The subgraph induced by the vertex subset  $S$  is denoted by  $G(S)$ .

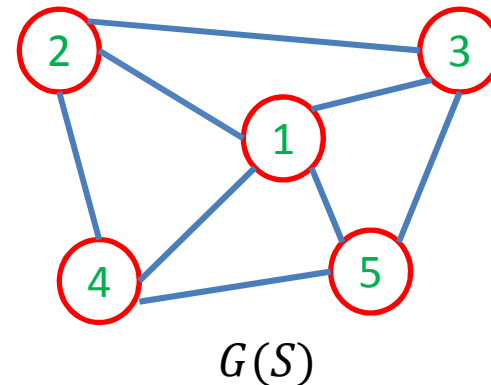
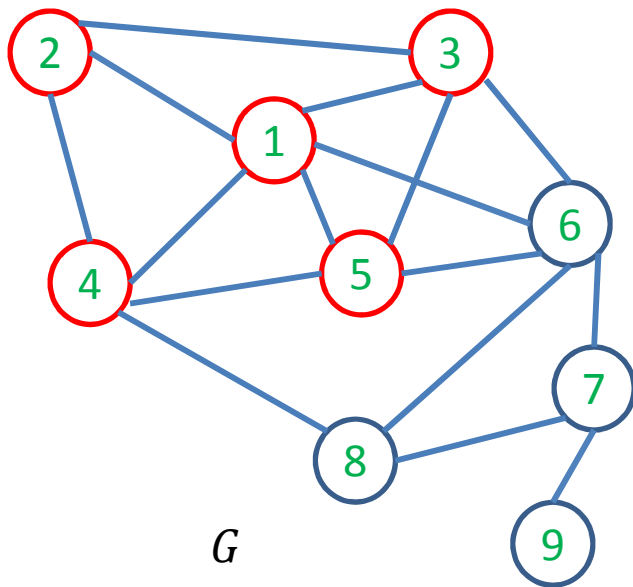


?

$G(S)$

# Basic Concepts (VII)

- An induced subgraph of a graph  $G = (V, E)$  is the graph with the vertex set  $S \subseteq V$  with an edge set  $E(S)$  that includes all such edges  $\{v, u\}$  in  $E$  with both of the vertices  $v$  and  $u$  included in the set  $S$
- The subgraph induced by the vertex subset  $S$  is denoted by  $G(S)$ .

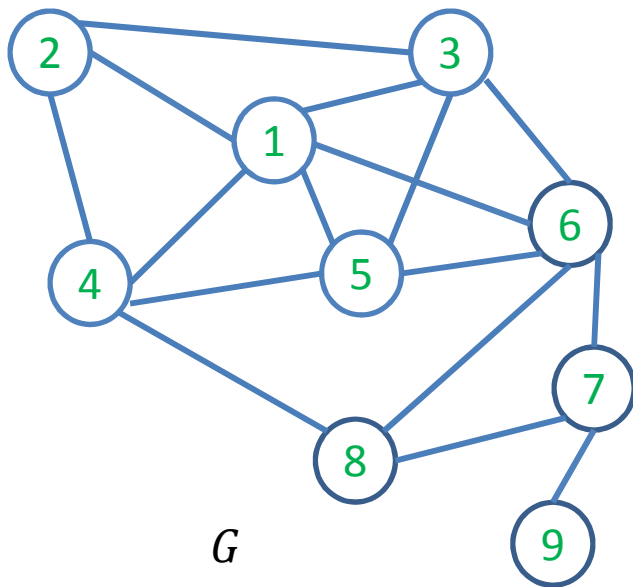




# Basic Concepts (VII)

- An induced subgraph that is a complete graph is called a clique.
- The local density of an induced subgraph in  $G = (V, E)$  is defined as

$$\delta(G(S)) = \frac{|E(S)|}{\binom{|S|}{2}}$$

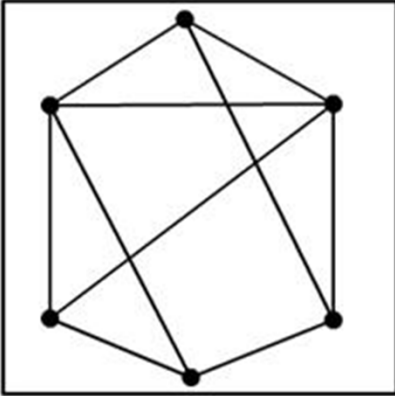


# Basic Concepts (VIII)

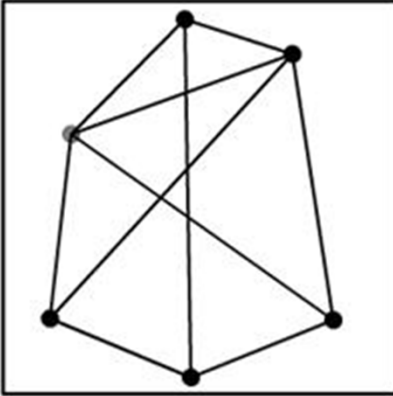
- Two graphs  $G_i$  and  $G_j$  are isomorphic if there exists a bijective (one-to-one) mapping  $f: V_i \rightarrow V_j$  (called an isomorphism) such that  $\{v, w\} \in E_i$  if and only if  $\{f(v), f(w)\} \in E_j$

Isomorphic Graphs

Graph G



Graph H

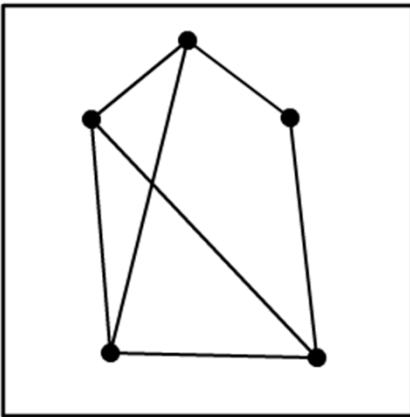


Drag the nodes of H to make it look like G, if possible.

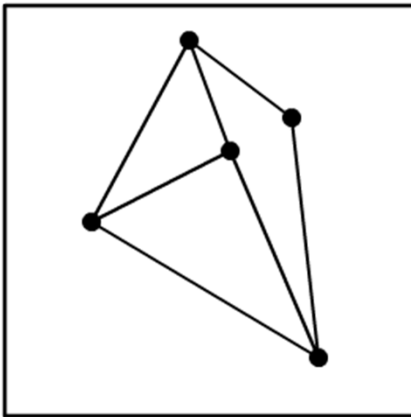
Are G and H isomorphic?

Isomorphic Graphs

Graph G



Graph H



Drag the nodes of H to make it look like G, if possible.

Are G and H isomorphic?

Image source:

<http://www.flashandmath.com/mathlets/discrete/graphtheory/graph4.html>

# Basic Concepts (IX)

- The **spectrum** of a graph  $G = (V, E)$  is defined as the list of eigenvalues (together with their multiplicities) of its adjacency matrix  $A_G$ .
- Laplacian matrix  $L = D - A_G$
- The normalized Laplacian is defined as

$$\mathcal{L} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}} = I - D^{-\frac{1}{2}} A_G D^{-\frac{1}{2}}$$

$D$  is the degree matrix

| Labeled graph | Degree matrix  | Adjacency matrix   | Laplacian matrix   |
|---------------|--|--|--|
|               | $\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$ | $\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$ | $\begin{pmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix}$ |

# Basic Concepts (X)

- The entry of the normalized Laplacian matrix

$$\mathcal{L}_{uv} = \begin{cases} 1, & \text{if } u = v \text{ and } \deg(v) > 0, \\ \frac{1}{\sqrt{\deg(u) \cdot \deg(v)}}, & \text{if } u \in \Gamma(v), \\ 0, & \text{otherwise.} \end{cases}$$

- Using normalized Laplacian is convenient as the eigenvalues of it all fall within the interval  $[0,2]$ .
- This makes the comparison of the spectra of two graphs easier.
  - Graphs that have the same spectrum are called cospectral.
- The smallest eigenvalue is always 0, its corresponding eigenvector is simply a vector with each element being the square-root of the degree of the corresponding vertex.

# Example Tasks of Graph Processing

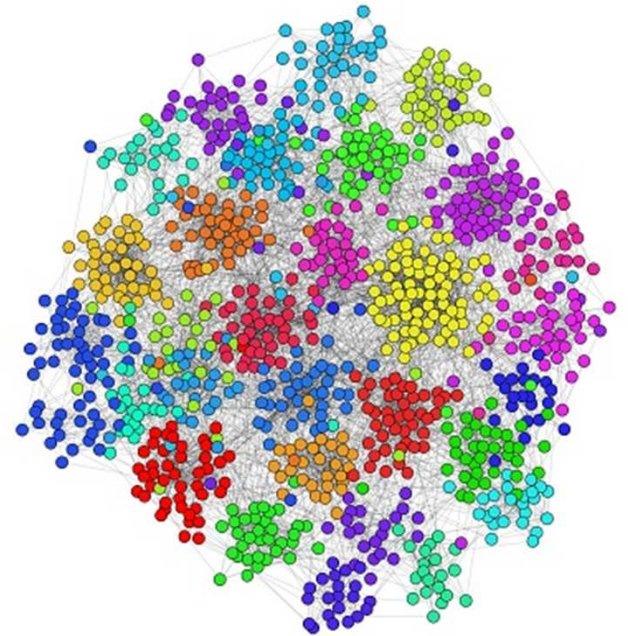
- **Conventional graph processing** (graph model, laws, graph dynamics, visualization, summarization, graph clustering, link analysis, social network analysis, ...)
- **Graph Pattern Mining**
  - Frequent graph patterns
  - Pattern summarization
  - Optimal graph patterns
  - Graph patterns with constraints
  - Approximate graph patterns
- **Graph Classification**
  - Pattern-based approach
  - Decision tree
  - Decision stumps
- **Graph Compression**

# Example Tasks of Graph Processing

- **Conventional graph processing** (graph model, laws, graph dynamics, visualization, summarization, **graph clustering**, social network analysis, link analysis, ...)
- **Graph Pattern Mining**
  - Frequent graph patterns
  - Pattern summarization
  - Optimal graph patterns
  - Graph patterns with constraints
  - Approximate graph patterns
- **Graph Classification**
  - Pattern-based approach
  - Decision tree
  - Decision stumps
- **Graph Compression**

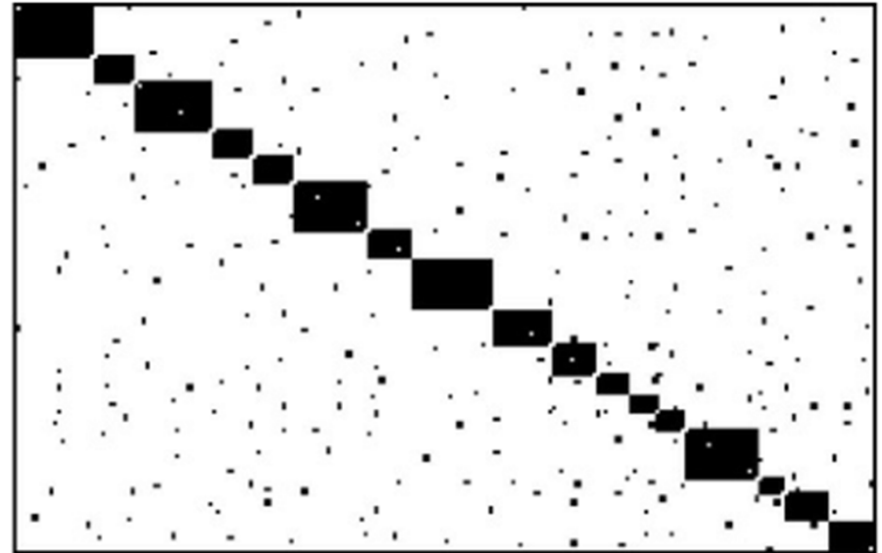
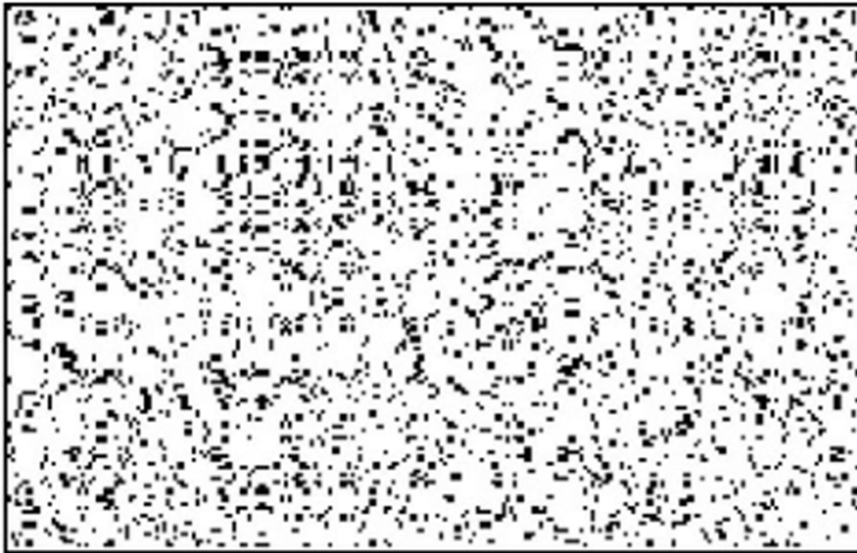
# Graph Clustering

- Formally, given a data set, the goal of clustering is to divide the data set into clusters such that the elements assigned to a particular cluster **are similar or connected** in some predefined sense.
- A clustering algorithm outputs a clustering for any input graph.
- If the structure of the graph is completely uniform, with the edges evenly distributed over the set of vertices, the clustering computed by any algorithm will be rather arbitrary.
- Quality measures – and if feasible, **visualizations** – will help to determine whether there are significant clusters present in the graph and whether a given clustering reveals them or not.



<http://khorshid.ut.ac.ir/~s.tabrizi/>

# An Example



The adjacency matrix of a 210-vertex graph with 1505 edges composed of 17 dense clusters. On the left, the vertices are ordered randomly and the graph structure can hardly be observed. On the right, the vertex ordering is by cluster and the 17-cluster structure is evident. Each black dot corresponds to an element of the adjacency matrix that has the value one, the white areas correspond to elements with the value zero.

Matrix diagonalization in itself is an important application of clustering algorithms.

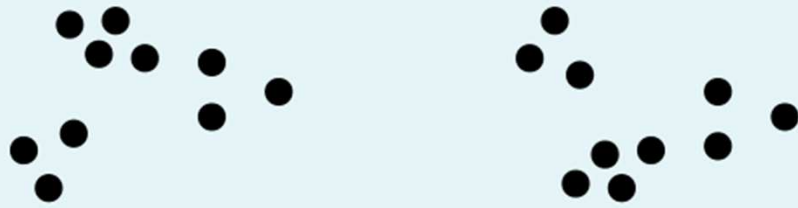
Such computations in turn enable efficient algorithms for graph partitioning, as the graph partitioning problem can be written in the form of a set of linear equations



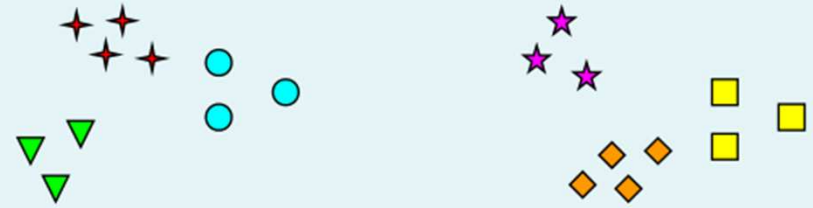
# Graph Clustering

- **Graph clustering** is the task of *grouping the vertices* of the graph into clusters taking into consideration the edge structure of the graph in such a way that there should be many edges within each cluster and relatively few between the clusters.
- *Unfortunately, no single definition of a cluster in graphs is universally accepted.*

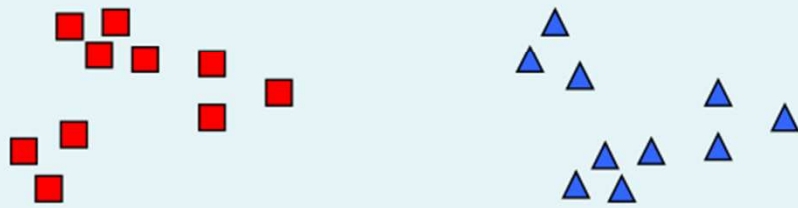
# Notion of a cluster can be ambiguous



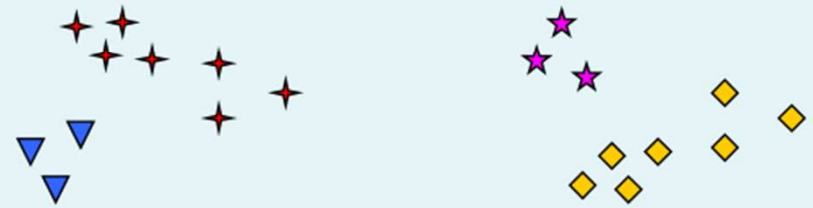
How many clusters?



Six Clusters



Two Clusters



Four Clusters

# Generation **Models** for Clustered Graphs

- Uniform random graphs (Gilbert model)
  - With  $n$  vertices, each of the  $\binom{n}{2}$  possible edges is included in the graph with probability  $p$
  - No dense clusters

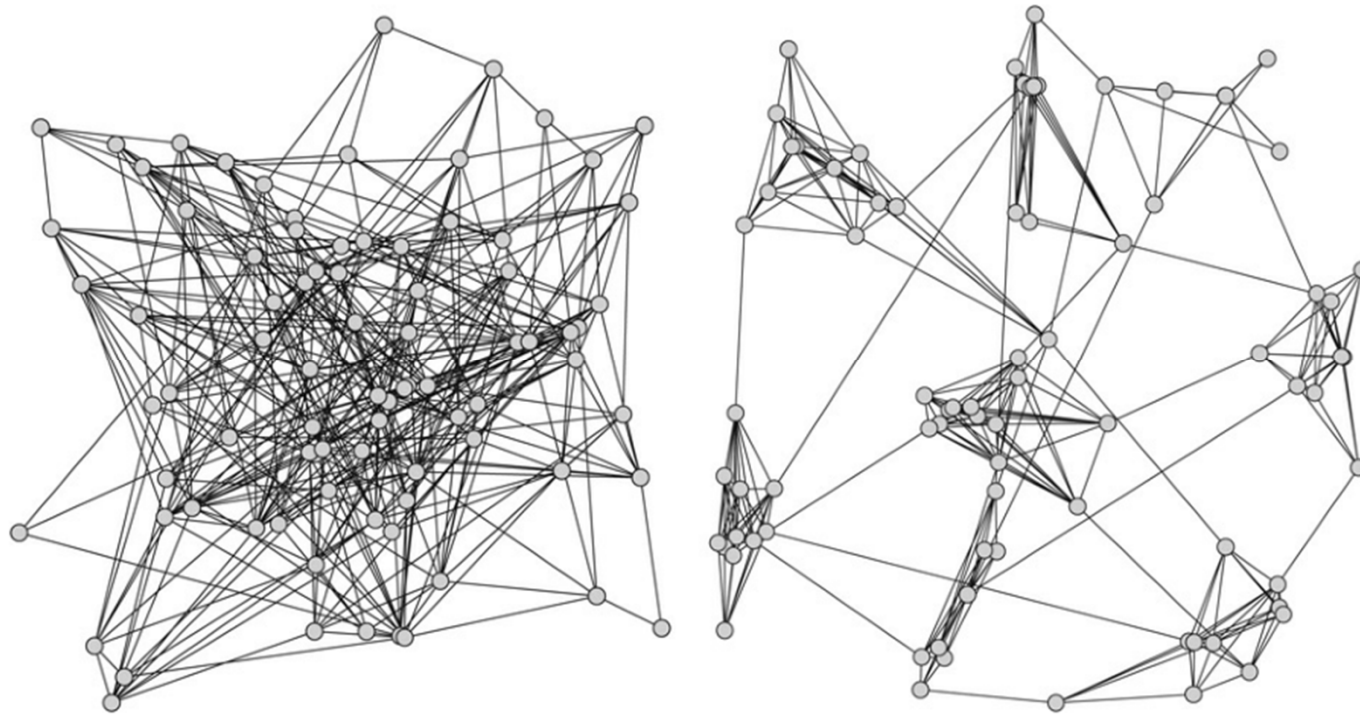
# Generation **Models** for Clustered Graphs

- Uniform random graphs (Gilbert model)
  - With  $n$  vertices, each of the  $\binom{n}{2}$  possible edges is included in the graph with probability  $p$
  - No dense clusters
- Planted  $l$ -partition model
  - A graph is generated with  $n = l \cdot k$  vertices that are partitioned into  $l$  groups each with  $k$  vertices. Two probability parameters  $p$  and  $q < p$  are used to construct the edge set: each pair of vertices that are in the same group share an edge with the higher probability  $p$ , whereas each pair of vertices in different groups shares an edge with the lower probability  $r$ .

# Generation Models for Clustered Graphs

- *Relaxed caveman graph*
  - In social sciences to capture the clustering properties of social network.
  - Each “**cave**” is a small complete graph.
  - Caves are linked by moving one of the edges in each cave to point to another cave.
  - With clear cluster structure, possibly with hierarchy

# An Example



Two graphs both of which have 84 vertices and 358 edges. The graph on the left is a uniform random graph of the  $\mathcal{G}_{n,m}$  model and the graph on the right has a relaxed caveman structure. Both graphs were drawn with spring-force visualization.

# Desirable Cluster Properties

- A cluster should be at least a connected subgraph. Preferably more paths (dense) within the subgraph.
- We classify the edges incident on  $v \in \mathcal{C}$  into two groups
  - Internal edges that connect  $v$  to other vertices also in  $\mathcal{C}$ .

$$\deg_{int}(v, \mathcal{C}) = |\Gamma(v) \cap \mathcal{C}|$$

- External edges that connect  $v$  to vertices outside of  $\mathcal{C}$

$$\deg_{ext}(v, \mathcal{C}) = |\Gamma(v) \cap (V \setminus \mathcal{C})|$$

$$\deg(v) = \deg_{int}(v, \mathcal{C}) + \deg_{ext}(v, \mathcal{C})$$

- Therefore,  $\deg_{int}(v, \mathcal{C})=0$  implies  $\mathcal{C}$  is a good cluster.

- One measure that helps to evaluate the sparsity of connections from the cluster to the rest of the graph is the cut size  $c(\mathcal{C}, V \setminus \mathcal{C})$ . The smaller the cut size, the better “isolated” the cluster.
- Determining when a cluster is dense is naturally achieved by computing the graph density.



- Local density of cluster  $\mathcal{C}$ , or intra-cluster density

$$\delta_{int}(\mathcal{C}) = \frac{|\{u, v\} \mid u \in \mathcal{C}, v \in \mathcal{C}\}|}{|\mathcal{C}|(|\mathcal{C}| - 1)}$$

- The inter-cluster density of a given clustering of a graph  $G$  into  $k$  clusters is the average of the intra-cluster densities of the included clusters

$$\delta_{int}(G|\mathcal{C}_1, \dots, \mathcal{C}_k) = \frac{1}{k} \sum_{i=1}^k \delta_{int}(\mathcal{C}_i)$$

The external density is

$$\delta_{int}(G|\mathcal{C}_1, \dots, \mathcal{C}_k) = \frac{|\{u, v\} \mid u \in \mathcal{C}_i, v \in \mathcal{C}_j, i \neq j\}|}{n(n-1) - \sum_{l=1}^k (|\mathcal{C}_l|(|\mathcal{C}_l| - 1))}$$

# Challenges of Clustering

- Globally speaking, the internal density of a good clustering should be notably higher than the density of the graph  $G$  and the inter-cluster density of the clustering should be lower.
- Therefore, the *loosest possible definition* of a graph cluster is that of a connected component, and the *strictest definition* is that each cluster should be a maximal *clique*. Connected components are easily computed in  $O(n + m)$ -time with a breadth-first search, whereas maximal clique detection is NP-complete.
- In most occasions, the semantically useful clusters lie somewhere in between these two extremes.
- When the input graph is very large, such as the Internet at router level or the Web graph, it is highly infeasible to rely on algorithms with exponential running time, as even linear-time computation gets tedious.

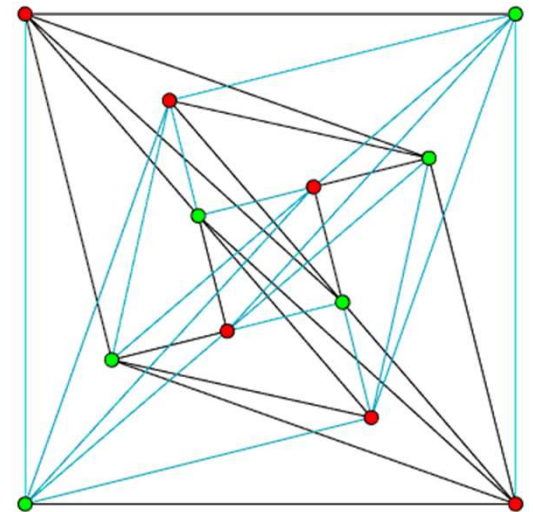
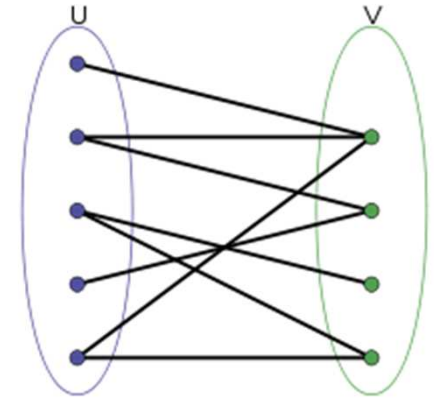
# Challenges of Clustering (II)

- Deterministic or probabilistic.
  - It is not always clear whether each vertex should be assigned fully to a cluster or could it instead have different “levels of membership” in several clusters?
  - In document clustering, such a situation is easily imaginable: a document can be mainly about fishing, for example, but also addresses sailing-related issues, and hence could be clustered into “fishing” with 0.9 membership, for example, and to “sailing” with a level of 0.3. Another solution would be creating a super-cluster to include all documents related to fishing and sailing, but the downside is that there can be documents on fishing that have no relation to sailing whatsoever.

# Representations of Clusters for Different Classes of Graphs

- Bipartite graphs

- A bipartite graph is a graph where the vertex set  $V$  can be split in two sets  $A$  and  $B$  such that all edges lie between those two sets: if  $\{v, w\} \in E$ , either  $v \in A$  and  $w \in B$  or  $v \in B$  and  $w \in A$ .
- Such graphs are natural for many application areas where the vertices represent two distinct classes of objects, such as customers and products; an edge could signify for example that a certain customer has bought a certain product. Possible clustering tasks could be grouping the customers by the types of products they purchase or grouping products purchased by the same people — the motivation could be targeted marketing, for instance. Carrasco et al. study a graph of advertisers and keywords used in advertisements to identify submarkets by clustering.



- Directed graphs

- Web graphs are directed graphs formed by web pages as vertices and hyperlinks as edges. A clustering of a higher-level web graph formed by all Chilean domains was presented by Virtanen. *Clustering of web pages can help identify topics and group similar pages.* This opens applications in search-engine technology; building artificial clusters is known to be a popular trick among websites of adult content to try to fool the PageRank algorithm used by Google to rate the quality of websites.

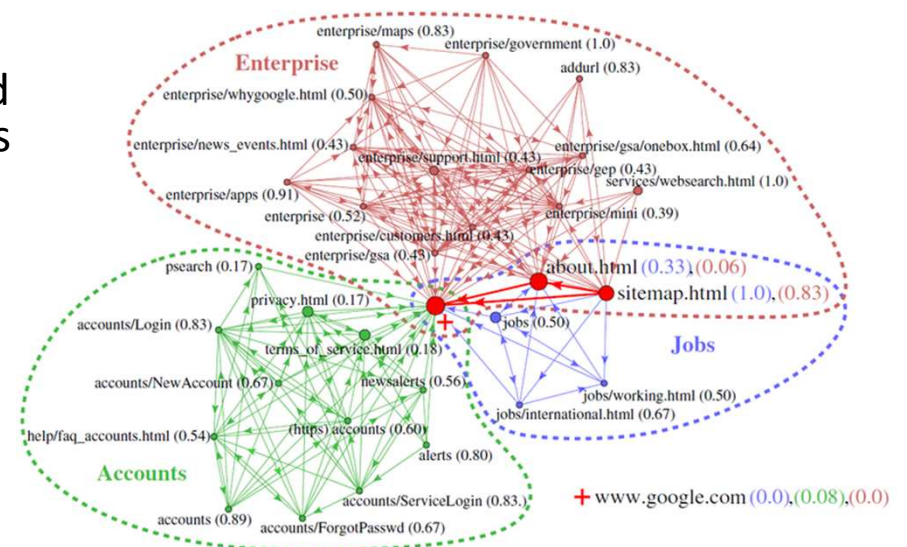


Image: <http://cfinder.org/>

# Measures for Identifying Clusters

- There are two main approaches for identifying a good cluster: one may either compute some values for the vertices and then classify the vertices into clusters based on the values obtained, or compute a fitness measure over the set of possible clusters and then choose among the set of cluster candidates that optimize the measure used.

- Vertex similarity metric
  - Graph distance
  - Adjacent-based measures
  - Connectivity measures
  
- If a similarity measure has been defined for the vertices, the cluster should contain vertices with close-by values

- Cluster fitness measures

- Density measures

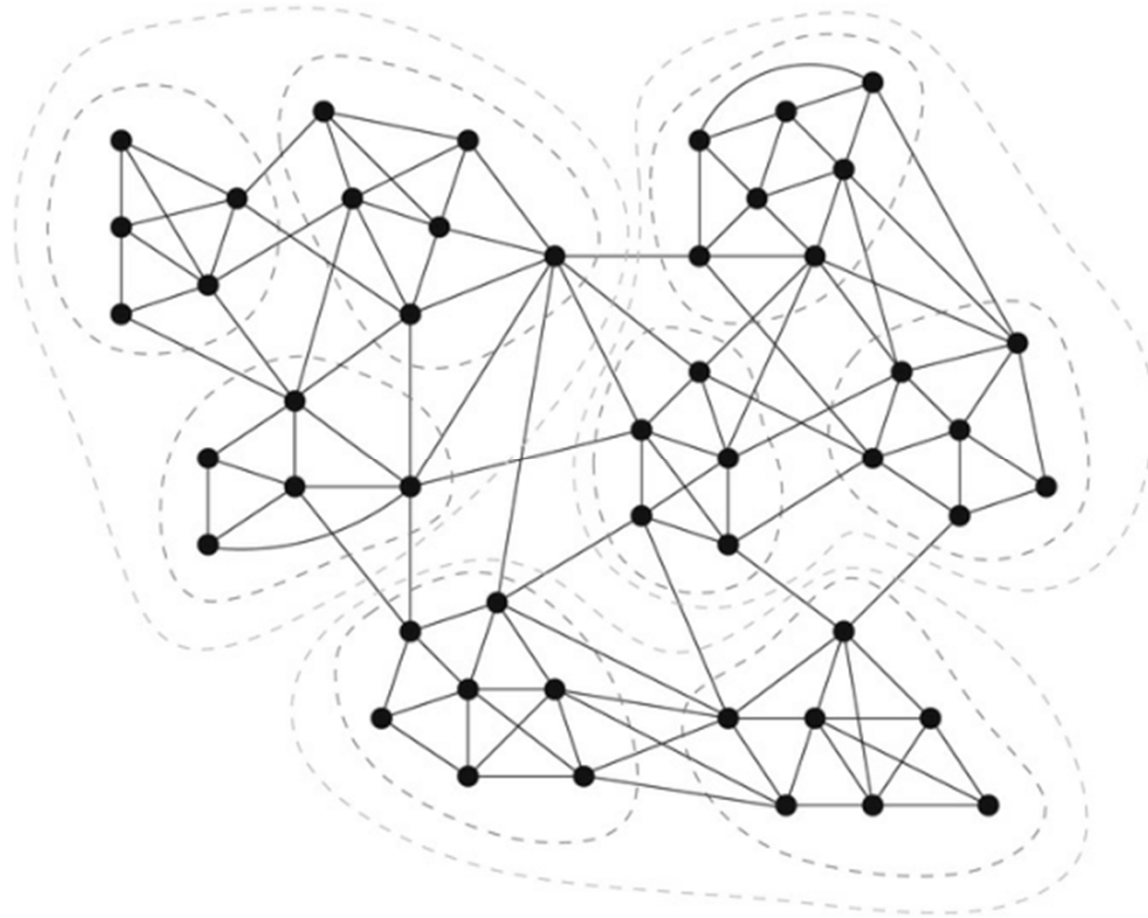
- $\delta_{int}(\mathcal{C}) = \frac{|\{u,v\} \mid u \in \mathcal{C}, v \in \mathcal{C}\}|}{|\mathcal{C}|(|\mathcal{C}|-1)}$

- Cut-based measures

- $\delta_{int}(G \mid \mathcal{C}_1, \dots, \mathcal{C}_k) = \frac{|\{u,v\} \mid u \in \mathcal{C}_i, v \in \mathcal{C}_j, i \neq j\}|}{n(n-1) - \sum_{l=1}^k (|\mathcal{C}_l|(|\mathcal{C}_l|-1))}$



# Hierarchical Clustering



# Example Tasks of Graph Processing

- **Conventional graph processing** (graph model, laws, graph dynamics, visualization, summarization, graph clustering, link analysis, social network analysis, ...)
- **Graph Pattern Mining**
  - Frequent graph patterns
  - Pattern summarization
  - Optimal graph patterns
  - Graph patterns with constraints
  - Approximate graph patterns
- **Graph Classification**
  - Pattern-based approach
  - Decision tree
  - Decision stumps
- **Graph Compression**

# Applications of Graph Patterns

- Mining biochemical structures
- Finding biological conserved sub-networks
- Finding functional modules
- Program control flow analysis
- Intrusion network analysis
- Mining communication networks
- Anomaly detection
- Mining XML structures
- Building blocks for graph classification, clustering, compression,
- Comparison, correlation analysis, and indexing

# Frequent Graph Pattern

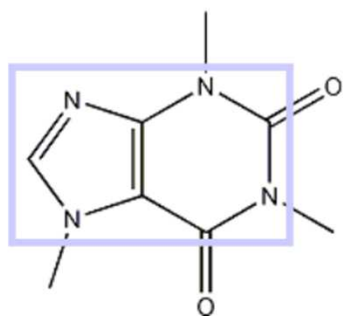
Given a graph dataset  $D$ , find subgraphs  $g$ , such that

$$freq(g) \geq \theta$$

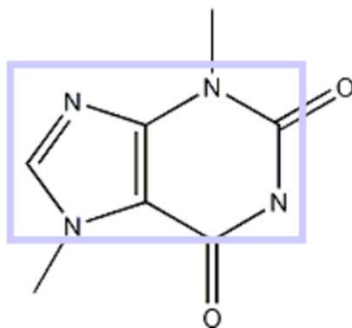
where  $freq(g)$  is the percentage of graphs in  $D$  that contain  $g$ .

# Frequent Subgraphs

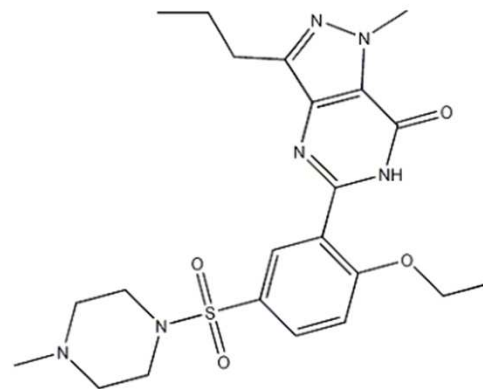
## CHEMICAL COMPOUNDS



(a) caffeine



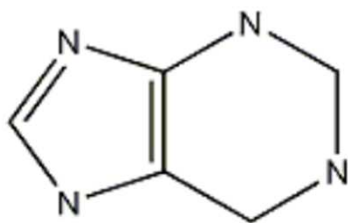
(b) diurobromine



(c) viagra

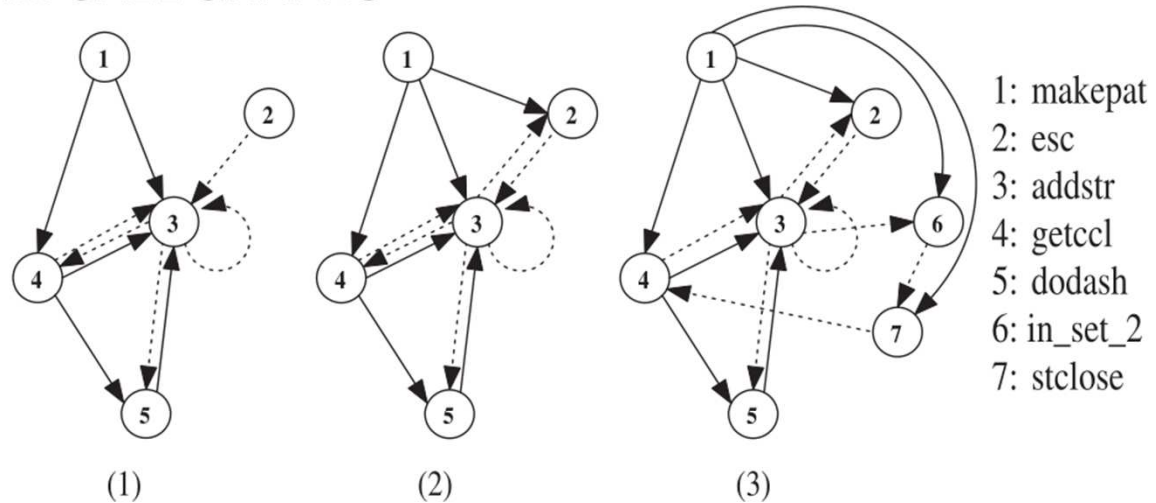
...

## FREQUENT SUBGRAPH

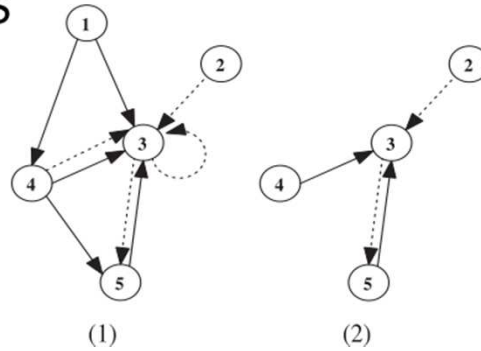


# Frequent Subgraphs

## PROGRAM CALL GRAPHS



## FREQUENT SUBGRAPHS (MIN SUPPORT IS 2)

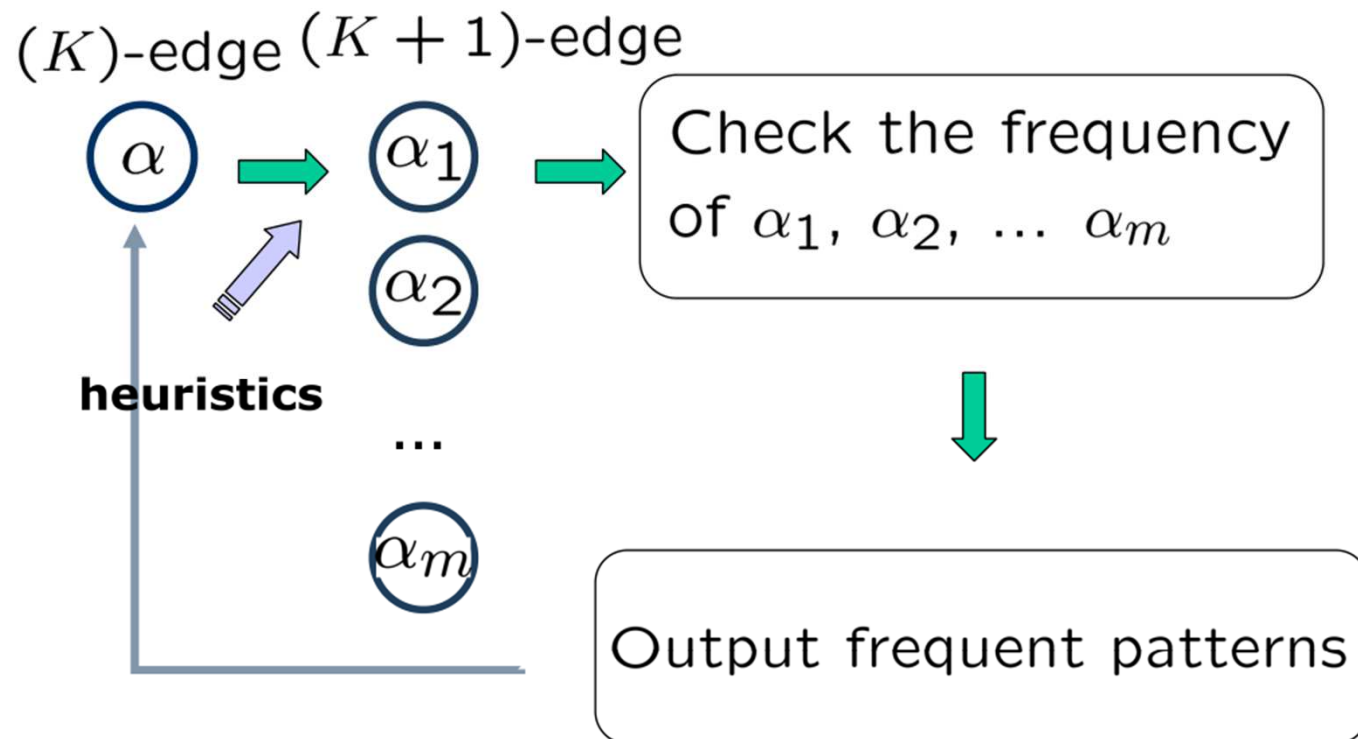


# Graph Mining Algorithms

- Inductive Logic Programming (WARMR, King et al. 2001)
  - Graphs are represented by Datalog facts
- Graph Based Approaches
  - Apriori-based approach
  - AGM/AcGM: Inokuchi, et al. (PKDD'00)
  - FSG: Kuramochi and Karypis (ICDM'01)
  - PATH#: Vanetik and Gudes (ICDM'02, ICDM'04)
  - FFSM: Huan, et al. (ICDM'03) and SPIN: Huan et al. (KDD'04)
  - FTOSM: Horvath et al. (KDD'06)
- Pattern growth approach
  - Subdue: Holder et al. (KDD'94)
  - MoFa: Borgelt and Berthold (ICDM'02)
  - gSpan: Yan and Han (ICDM'02)
  - Gaston: Nijssen and Kok (KDD'04)
  - CMTreeMiner: Chi et al. (TKDE'05)
  - LEAP: Yan et al. (SIGMOD'08)

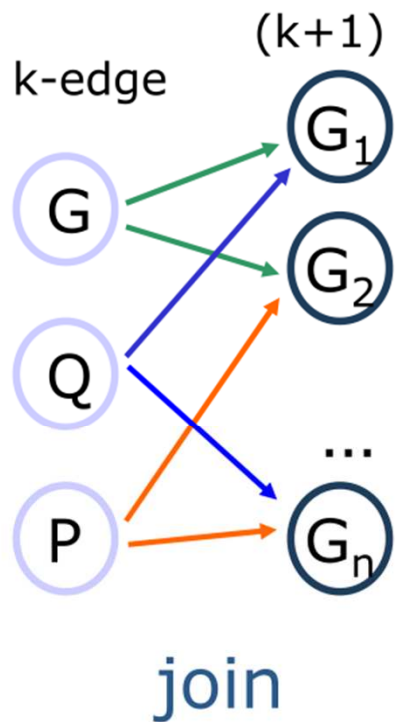
# Aprior Property

- If a graph is frequent, all of its subgraphs are frequent.



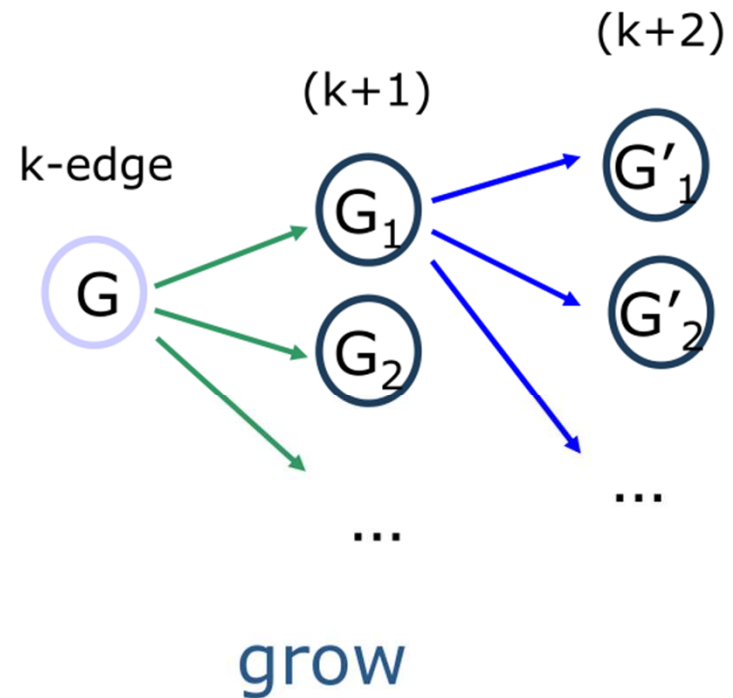


# Generation of Candidate Patterns



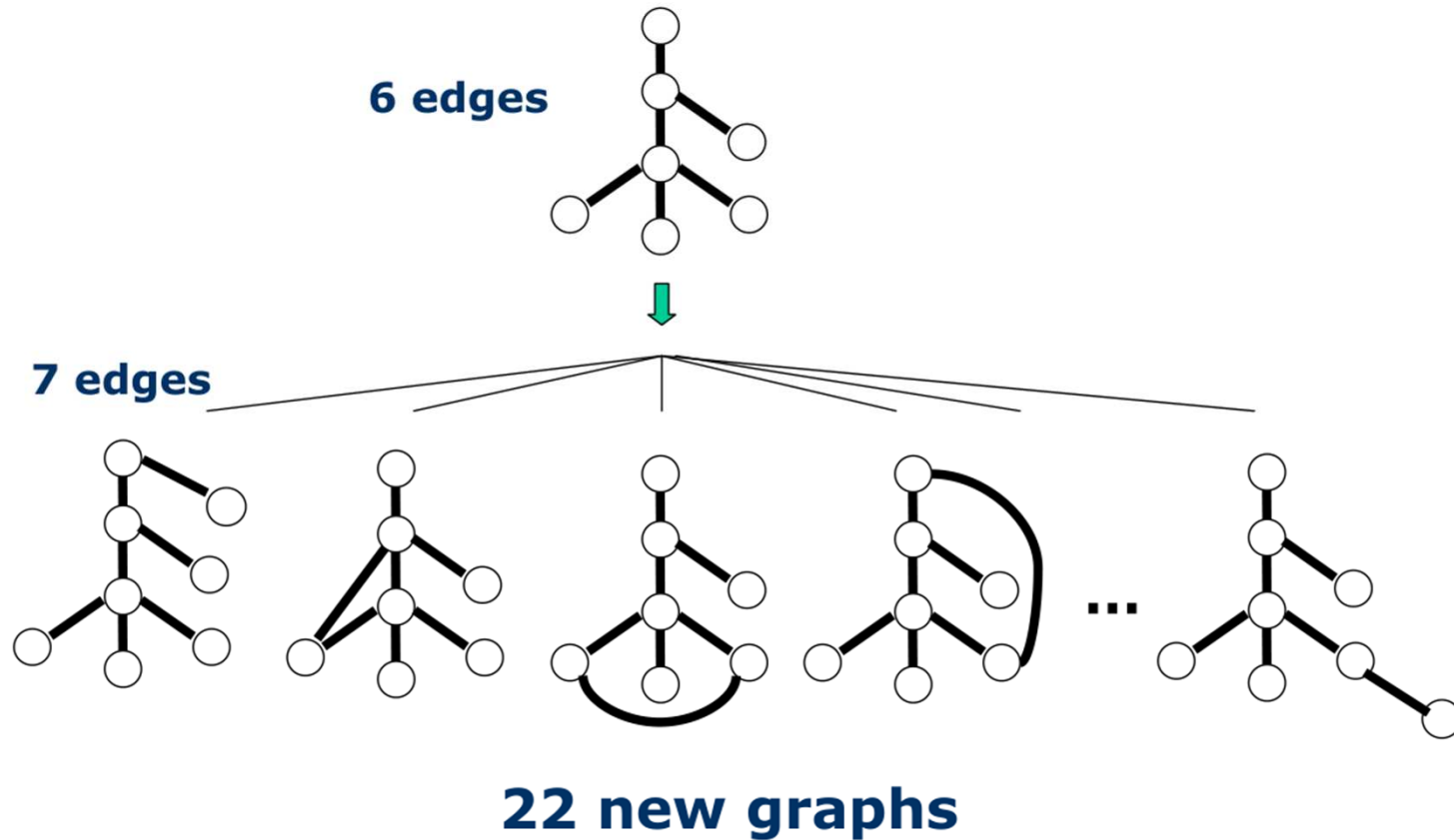
Apriori-Based Approach

VS.



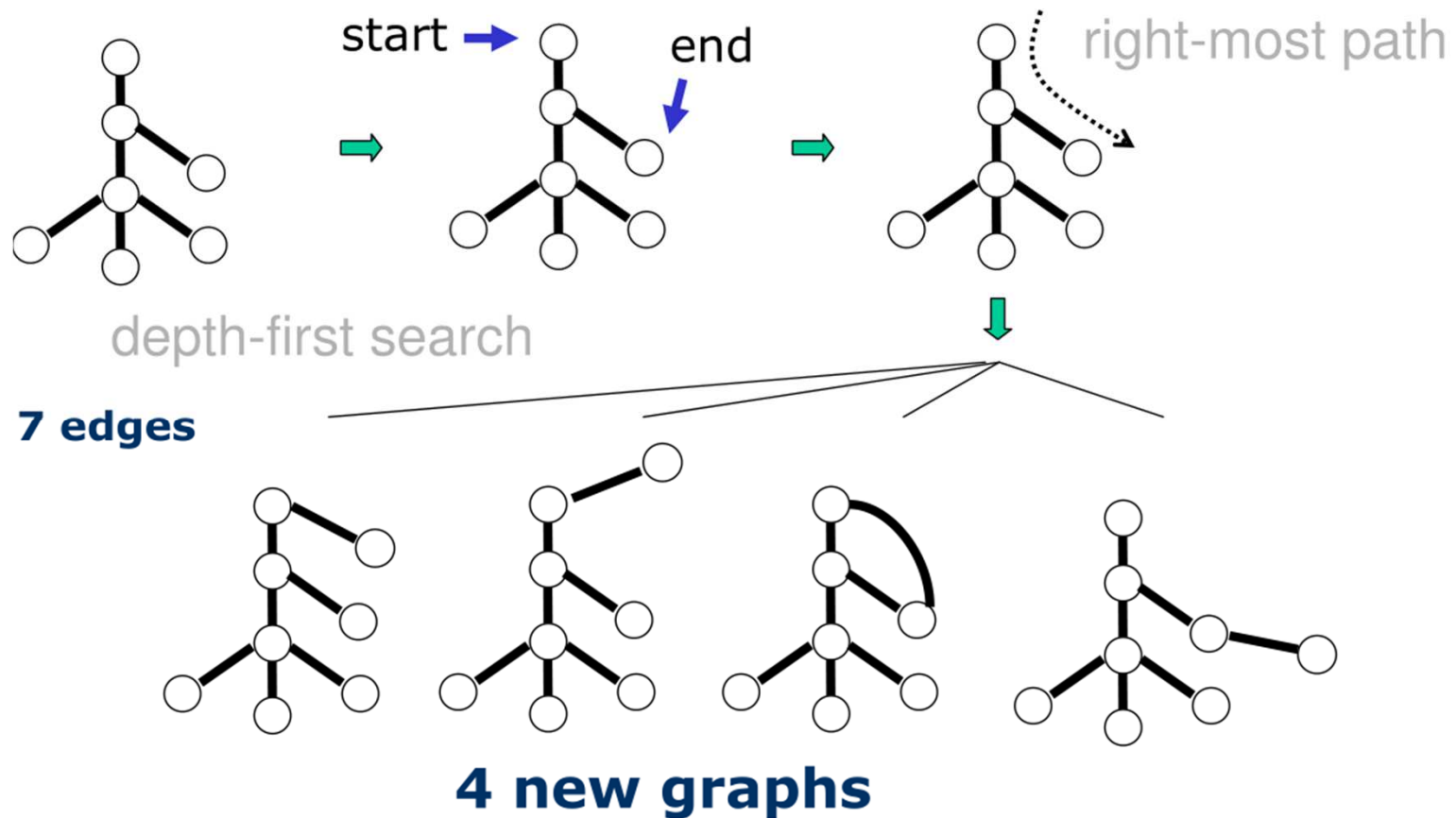
Pattern-Growth Approach

# Discovery Order: Free Extension



# Discovery Order: Right-Most Extension

(Yan and Han ICDM'02)



# Duplicates Elimination

Existing patterns  $g_1, g_2, \dots, g_m$

Newly discovered pattern  $g$

- Option 1
  - Check graph isomorphism of  $g$  with each graph (slow)
- Option 2
  - Transform each graph to a canonical label, create a hash value for this canonical label, and check if there is a match with  $g$  (faster)
- Option 3
  - Build a canonical order and generate graph patterns in that order (fastest)

# Graph Pattern **Explosion** Problem

- If a graph is frequent, all of its subgraphs are frequent – the **Apriori property**
- An  $n$ -edge frequent graph may have  $2^n$  subgraphs!
- In the AIDS antiviral screen dataset with **400+** compounds, at the support level 5%, there are > 1M frequent graph patterns
- **Conclusions:** Many enumeration algorithms are available AGM, FSG, gSpan, Path-Join, MoFa, FFISM, SPIN, Gaston, and so on, but two significant problems exist

Problem 1: Exponential Pattern Set

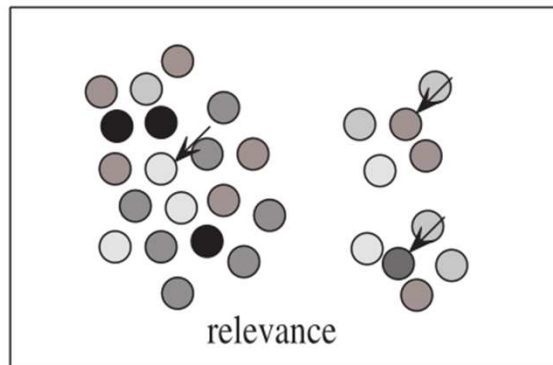
Problem 2: Threshold Setting

# Pattern Summarization

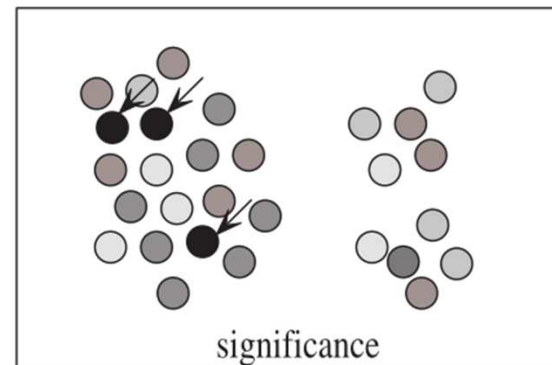
(Xin et al., KDD'06, Chen et al. CIKM'08)

- Too many patterns may not lead to more explicit knowledge
- It can confuse users as well as further discovery (e.g., clustering, classification, indexing, etc.)
- A small set of “**representative**” patterns that preserve most of the information

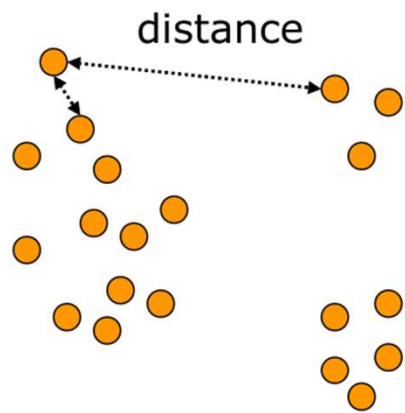
Clustering



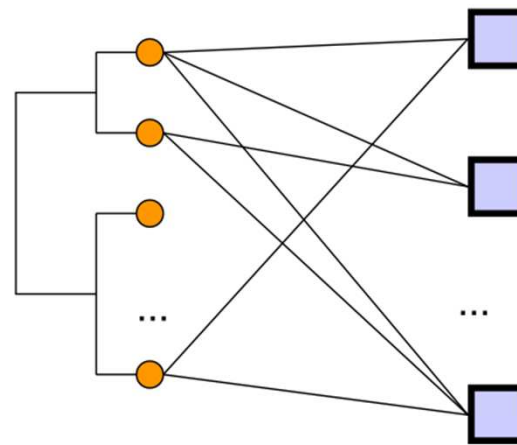
Top-K



# Pattern Distance



patterns



patterns

data

measure 1: pattern based

- pattern containment
- pattern similarity

measure 2: data based

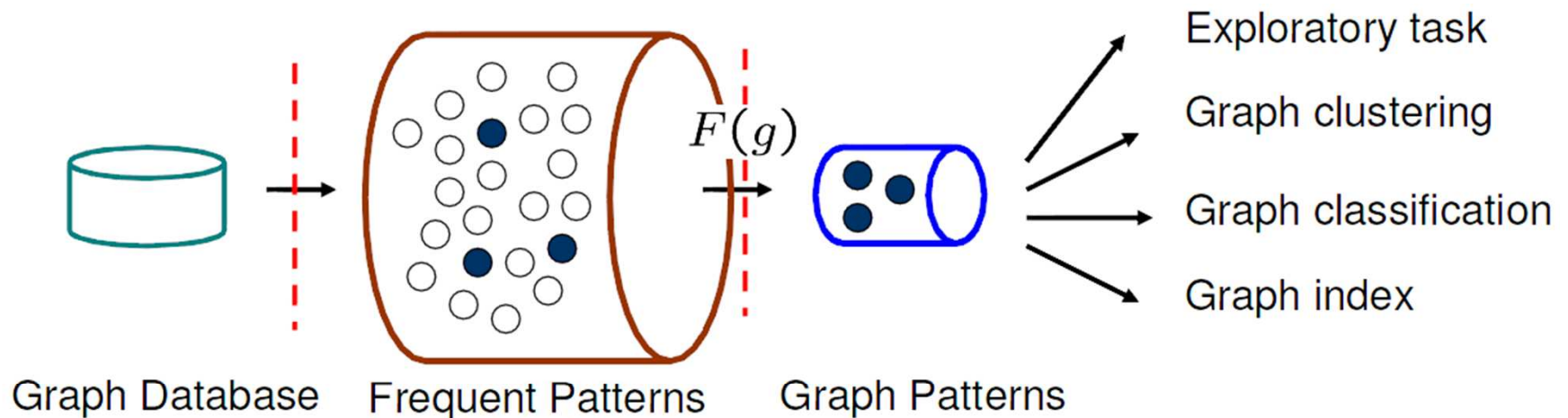
- data similarity

# Closed and Maximal Graph Pattern

- Closed Frequent Graph
  - A frequent graph  $G$  is closed if there exists no supergraph of  $G$  that carries the same support as  $G$
  - If some of  $G$ 's subgraphs have the same support, it is unnecessary to output these subgraphs (**nonclosed graphs**)
  - Lossless compression: still ensures that the mining result is complete
- Maximal Frequent Graph
  - A frequent graph  $G$  is maximal if there exists no supergraph of  $G$  that is frequent



# Frequent Pattern Based Mining



1. Bottleneck : millions, even billions of patterns
2. No guarantee of quality

**Many other recent techniques and applications ....**

# Acknowledgment

- Thanks material from
  - Survey of graph clustering by Satu Elisa Schaeffer
  - Survey of graph cut
  - Graph mining tutorial by Karsten Borgwardt and Xifeng Yan