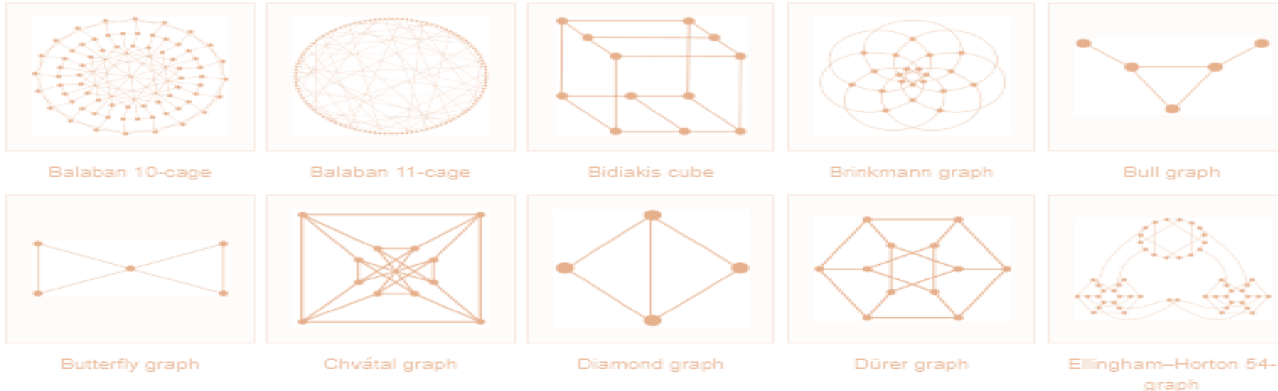
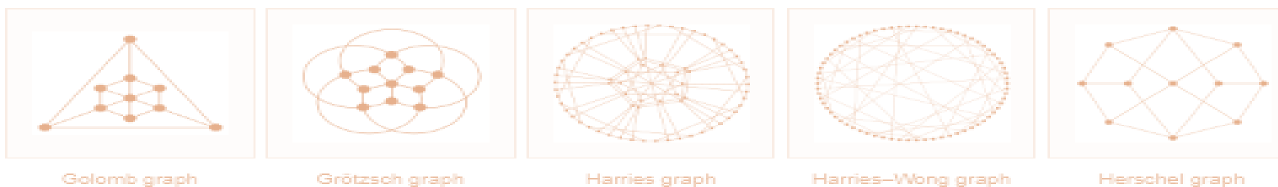
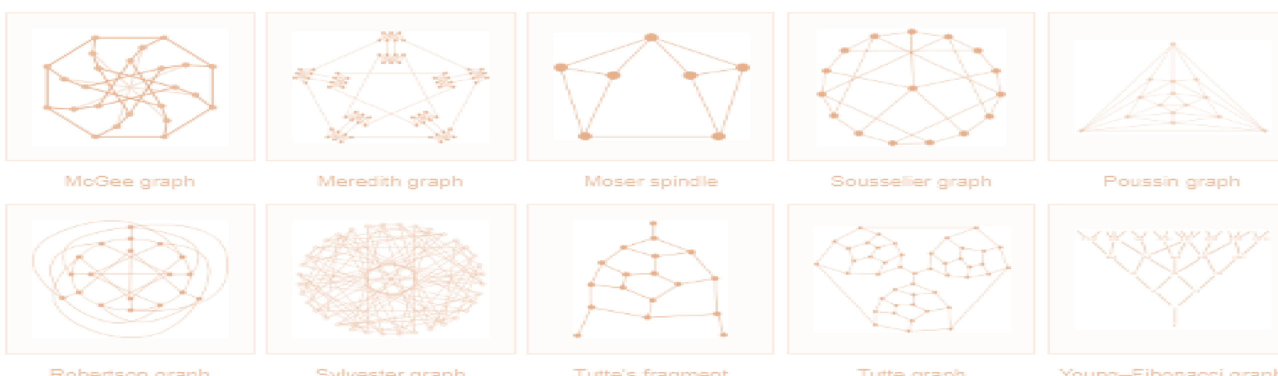


Graph Theory

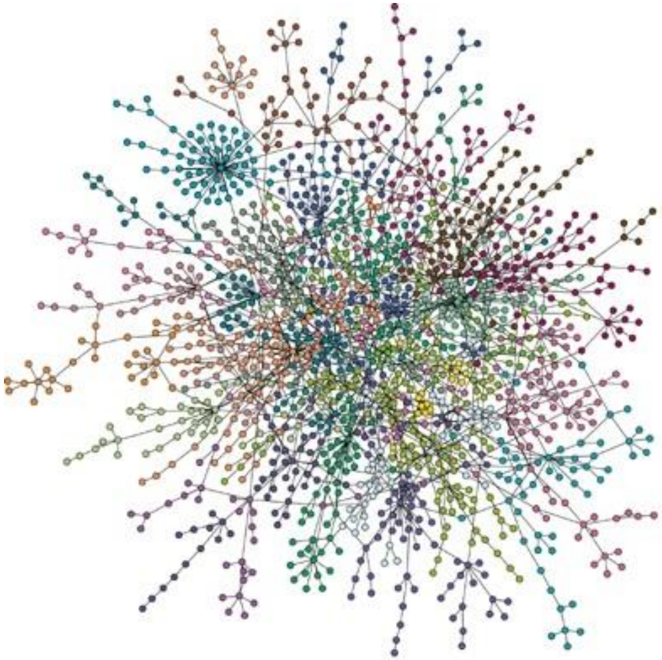


Introduction to Graph Theory (A)

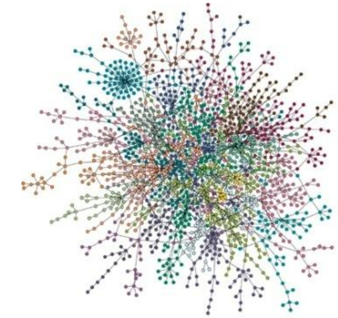


About the Course

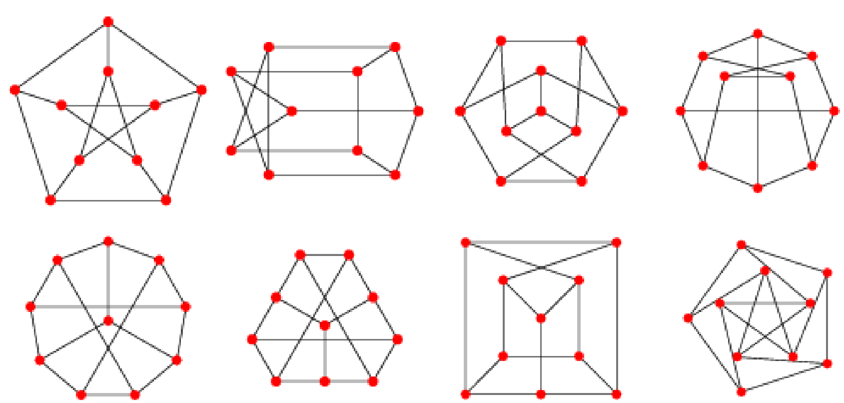
- Schedule:
  - Wednesday: 14:00 – 17:00 (Lecture)
  - Wednesday: 17:00 – XY:00 (Lab)
- Office:
  - … Anytime
- Mail:
  - ipolenak@cs.uoi.gr
  - ipolenak@uoi.gr
- Course Page
  - classweb
  - www.cs.uoi.gr/~ipolenak
- Grading and Examination
  - Implementation (C/Java preferred)
  - One exercise per week



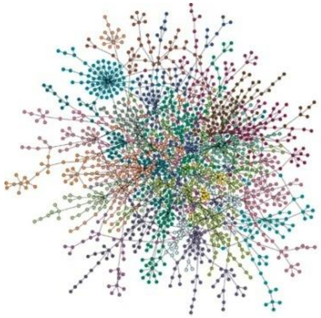
Basics…



- **What is a "graph"**
  - A graph G is a structure amounting to a set of objects in which, some pairs of the objects are in some sense related.
  - The objects → "mathematical abstractions" also called <u>vertices</u> (or, nodes/points).
  - The pairs of vertices → "relation between objects" also called <u>edges</u> (or, arc or line).
  - A graph is depicted in diagrammatic form as:
    - a set of dots for the vertices,
    - joined by lines or curves for the edges.

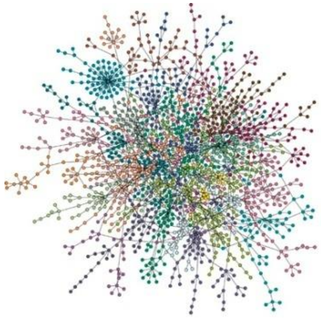


Basics…



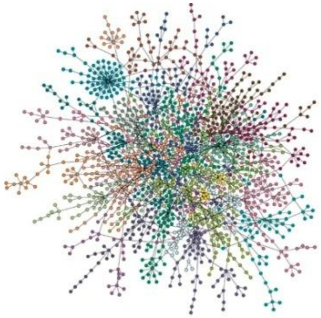
- **What is a "graph"**
  - Term: A generalization of the simple "dot-edge" concept .
  - Representation: Graph $G = (V, E)$ consists set of vertices denoted by V, or by $V(G)$ and set of edges E, or $E(G)$.
  - Edge Orientation/Direction: Represents the order between the vertices it connects.
  - Edge Weight: Represents a "utilization cost" between the vertices it connects.
  - Loop: A loop is an edge whose endpoints are equal i.e., an edge joining a vertex to it self is called a loop.
  - Multiple Edges: Two or more edges joining the same pair of vertices.

Basics…

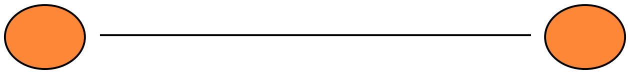


- **Anatomy of a Graph – Simple Graph (Undirected)**
  - The edges are represented as $\{u, v\}$.
    Disregards any sense of direction and treats both end vertices interchangeably.
  - A simple graph (undirected graph) is an ordered pair $G = (V, E)$ comprising:
    - V a set of vertices (also called nodes or points);
    - $E \subseteq \{\{x, y\} \mid (x, y) \in V^2 \wedge x \neq y\}$ a set of edges (also called links or lines), which are unordered pairs of vertices (i.e., an edge is associated with two distinct vertices).

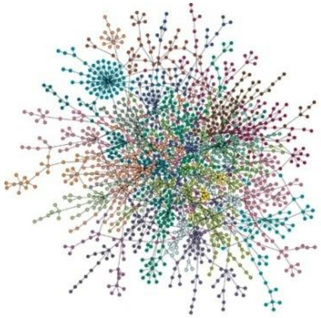
Basics…



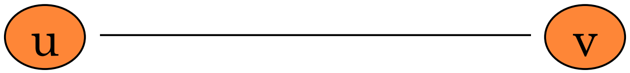
- **Anatomy of a Graph – Simple Graph (Undirected)**
  - The edges are represented as $\{u, v\}$.
    Disregards any sense of direction and treats both end vertices interchangeably.
  - A simple graph (undirected graph) is an ordered pair $G = (V, E)$ comprising:
    - V a set of vertices (also called nodes or points);
    - $E \subseteq \{\{x, y\} \mid (x, y) \in V^2 \wedge x \neq y\}$ a set of edges (also called links or lines), which are unordered pairs of vertices (i.e., an edge is associated with two distinct vertices).



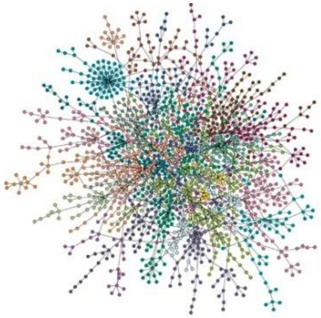
Basics…



- **Anatomy of a Graph – Simple Graph (Undirected)**
  - The edges are represented as $\{u, v\}$.
    Disregards any sense of direction and treats both end vertices interchangeably.
  - A simple graph (undirected graph) is an ordered pair $G = (V, E)$ comprising:
    - V a set of vertices (also called nodes or points);
    - $E \subseteq \{\{x, y\} \mid (x, y) \in V^2 \land x \neq y\}$ a set of edges (also called links or lines), which are unordered pairs of vertices (i.e., an edge is associated with two distinct vertices).

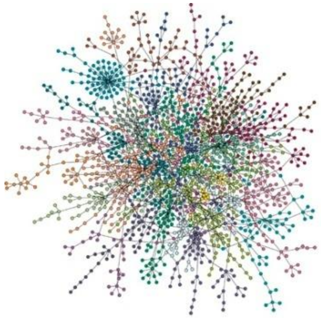


Basics…



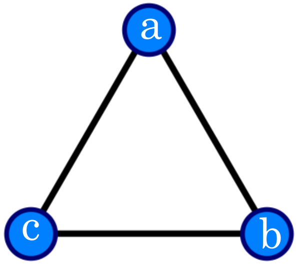
- **Anatomy of a Graph – Simple Graph (Undirected)**
  - The edges are represented as $\{u, v\}$.
    Disregards any sense of direction and treats both end vertices interchangeably.
  - A simple graph (undirected graph) is an ordered pair $G = (V, E)$ comprising:
    - V a set of vertices (also called nodes or points);
    - $E \subseteq \{\{x, y\} \mid (x, y) \in V^2 \wedge x \neq y\}$ a set of edges (also called links or lines), which are unordered pairs of vertices (i.e., an edge is associated with two distinct vertices).
  - Simple (Undirected) Graph: consists of V, a nonempty set of vertices, and E, a set of unordered pairs of distinct elements of V called edges (undirected)

Basics…

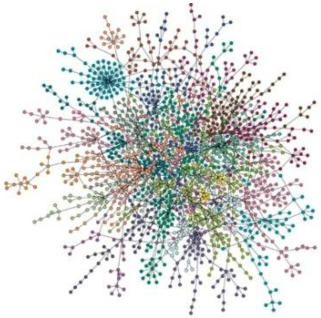


Anatomy of a Graph – Simple Graph (Undirected)

- The edges are represented as $\{u, v\}$.
  Disregards any sense of direction and treats both end vertices interchangeably.
- A simple graph (undirected graph) is an ordered pair $G = (V, E)$ comprising:
  - V a set of vertices (also called nodes or points);
  - $E \subseteq \{\{x, y\} \mid (x, y) \in V^2 \wedge x \neq y\}$ a set of edges (also called links or lines), which are unordered pairs of vertices (i.e., an edge is associated with two distinct vertices).
- Simple (Undirected) Graph: consists of V, a nonempty set of vertices, and E, a set of unordered pairs of distinct elements of V called edges (undirected)
- Representation Example: $G(V, E), V = \{a, b, c\}, E = \{\{a, b\}, \{b, c\}, \{c, a\}\}$



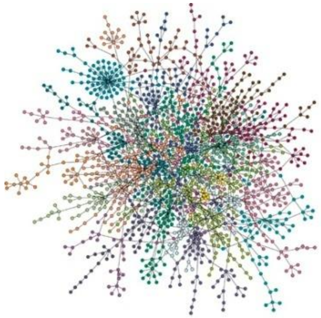
Basics…



Artifacts on Undirected Graphs

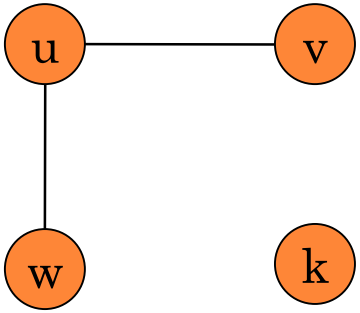
- Vertices u and v are adjacent if $\{u, v\}$ is an edge, e is called incident with u and v.
- Vertices u and v are called endpoints of $\{u, v\}$
- Degree of Vertex ($\deg(v)$): the number of edges incident on a vertex.
  - A loop contributes twice to the degree (why?).
- Pendant Vertex: $\deg(v) = 1$
- Isolated Vertex: $\deg(k) = 0$

Basics…

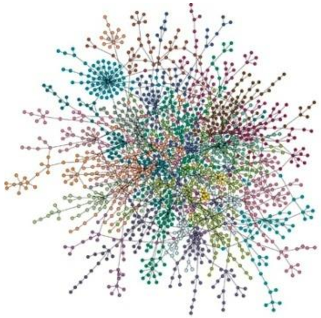


Artifacts on Undirected Graphs

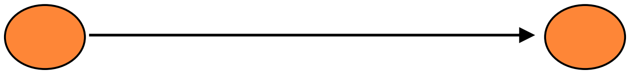
- Vertices u and v are adjacent if $\{u, v\}$ is an edge, e is called incident with u and v.
- Vertices u and v are called endpoints of $\{u, v\}$
- Degree of Vertex ($\deg(v)$): the number of edges incident on a vertex.
    - A loop contributes twice to the degree (why?).
- Pendant Vertex: $\deg(v) = 1$
- Isolated Vertex: $\deg(k) = 0$
- Representation Example:
    - For $V = \{u, v, w\}, E = \{\{u, w\}, \{u, v\}\}$,
    - $\deg(u) = 2, \deg(v) = 1, \deg(w) = 1, \deg(k) = 0,$
    - w and v are pendant , k is isolated



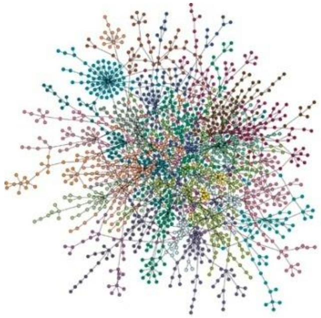
BASICS…



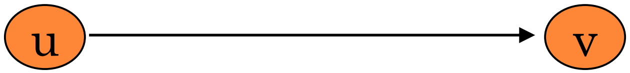
- **Anatomy of a Graph – Directed Graph (Digraph)**
  - The edges are represented as (u, v) starting from vertex u, finishing to vertex v
  - A directed graph or digraph is a graph in which edges have orientations, i.e., an ordered pair $G = (V, E)$ comprising:
    - V a set of vertices (also called nodes or points);
    - $E \subseteq \{(x, y) \mid (x, y) \in V^2 \wedge x \neq y\}$ a set of edges (directed edges) which are ordered pairs of distinct vertices



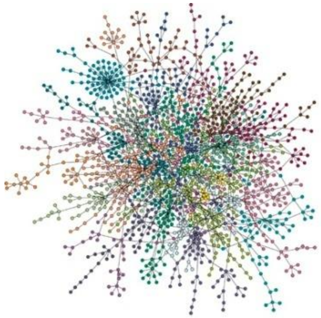
BASICS…



- **Anatomy of a Graph – Directed Graph (Digraph)**
  - The edges are represented as (u, v) starting from vertex u, finishing to vertex v
  - A directed graph or digraph is a graph in which edges have orientations, i.e., an ordered pair $G = (V, E)$ comprising:
    - V a set of vertices (also called nodes or points);
    - $E \subseteq \{(x, y) \mid (x, y) \in V\,2 \wedge x \neq y\}$ a set of edges (directed edges) which are ordered pairs of distinct vertices

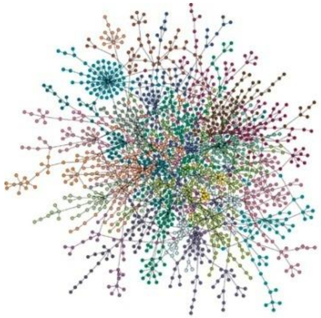


Basics…



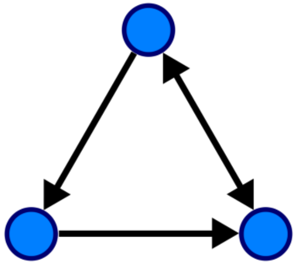
- **Anatomy of a Graph – Directed Graph (Digraph)**
  - The edges are represented as (u, v) starting from vertex u, finishing to vertex v
  - A directed graph or digraph is a graph in which edges have orientations, i.e., an ordered pair $G = (V, E)$ comprising:
    - V a set of vertices (also called nodes or points);
    - $E \subseteq \{(x, y) \mid (x, y) \in V\,2 \wedge x \neq y\}$ a set of edges (directed edges) which are ordered pairs of distinct vertices
  - A Directed Graph: $G(V, E)$, set of vertices V, and set of Edges E, that are ordered pair of elements of V (directed edges)

BASICS…

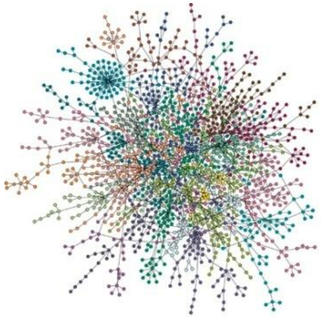


Anatomy of a Graph – Directed Graph (Digraph)

- The edges are represented as (u, v) starting from vertex u, finishing to vertex v
- A directed graph or digraph is a graph in which edges have orientations, i.e., an ordered pair $G = (V, E)$ comprising:
  - V a set of vertices (also called nodes or points);
  - $E \subseteq \{(x, y) \mid (x, y) \in V\,2 \wedge x \neq y\}$ a set of edges (directed edges) which are ordered pairs of distinct vertices
- A Directed Graph: $G(V, E)$, set of vertices V, and set of Edges E, that are ordered pair of elements of V (directed edges)
- Representation Example: $G(V, E), V = \{u, v, w\}$,
  $E = \{(a, c), (c, b), (b, a) , (a, b)\}$

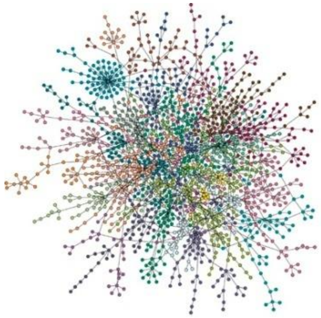


BASICS…



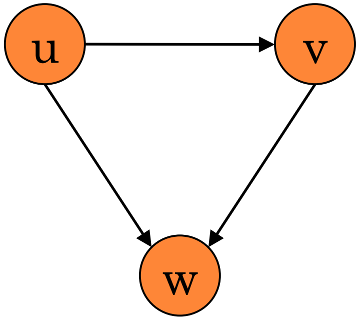
- **Artifacts on Directed Graphs**
  - For the edge (u, v), u is adjacent to v OR v is adjacent from u,
    - u → Initial vertex,
    - v → Terminal vertex
  - In-degree ($deg^-(u)$): number of edges for which u is terminal vertex
  - Out-degree ($deg^+(u)$): number of edges for which u is initial vertex
    - A loop contributes 1 to both in-degree and out-degree (why?)

Basics…



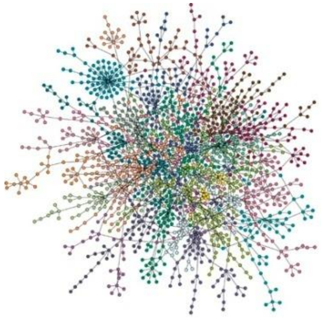
Artifacts on Directed Graphs

- For the edge (u, v), u is <u>adjacent</u> to v OR v is <u>adjacent</u> from u,
  - u → <u>Initial</u> vertex,
  - v → <u>Terminal</u> vertex
- <u>In-degree</u> ($deg^-(u)$): number of edges for which u is terminal vertex
- <u>Out-degree</u> ($deg^+(u)$): number of edges for which u is initial vertex
  - A loop contributes 1 to both in-degree and out-degree (why?)
- Representation Example:
  - For V = {u, v, w} , E = {(u, w), (v, w), (u, v)},
  - deg^- (u) = 0, deg^- (v) = 1, deg^- (w) = 2
  - deg^+ (u) = 2, deg^+ (v) = 1, deg^+ (u) = 0



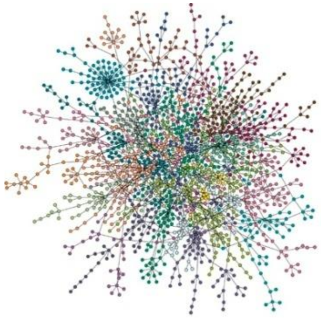
Basics…

- **Overview on Simple Graphs and Digraphs**



Type	Edges	Multiple Edges?	Loops?
Simple Graph	undirected	No	No
Multigraph	undirected	Yes	No
Pseudograph	undirected	Yes	Yes
Digraph	directed	No	Yes
Directed Multigraph	directed	Yes	Yes

Basics…



Overview on Simple Graphs and Digraphs

- **Undirected Graph**

  Remark: Edges have no orientation ⟶ $\{x, y\} \in E(G) \Leftrightarrow \{y, x\} \in E(G)$.

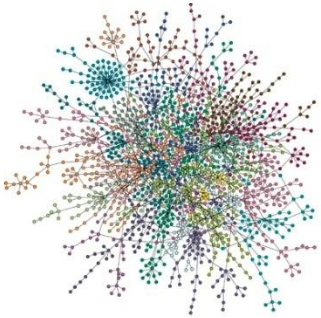
- **Directed Graph**

  Remark: Edges have orientation ⟶ $\{x, y\} \in E(G) \not\Leftrightarrow \{y, x\} \in E(G)$.

- **Weighted Graph (undirected or directed)**
  - A graph in which a number (weight) is assigned to each edge.
  - Such weights might represent for example costs, lengths or capacities, or…
  - Weighted Graphs arise in many contexts, e.g. in shortest path problems.

G=(V,E)	Undirected Edges	Directed Edges
Unweighted Edges		
Weighted Edges		

Basics…



Overview on Simple Graphs and Digraphs

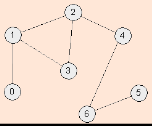
- **Undirected Graph**

  Remark: Edges have no orientation $\rightarrow \{x, y\} \in E(G) \Leftrightarrow \{y, x\} \in E(G)$.

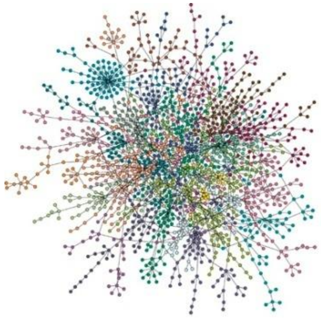
- **Directed Graph**

  Remark: Edges have orientation $\rightarrow \{x, y\} \in E(G) \not\Leftrightarrow \{y, x\} \in E(G)$.

- **Weighted Graph (undirected or directed)**
  - A graph in which a number (weight) is assigned to each edge.
  - Such weights might represent for example costs, lengths or capacities, or…
  - Weighted Graphs arise in many contexts, e.g. in shortest path problems.

G=(V,E)	Undirected Edges	Directed Edges
Unweighted Edges		
Weighted Edges		

Basics…



Overview on Simple Graphs and Digraphs

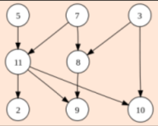
- **Undirected Graph**

  Remark: Edges have no orientation $\longrightarrow \{x, y\} \in E(G) \Leftrightarrow \{y, x\} \in E(G)$.

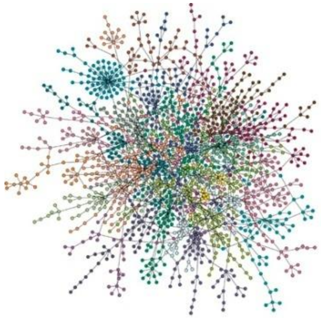
- **Directed Graph**

  Remark: Edges have orientation $\longrightarrow \{x, y\} \in E(G) \not\Leftrightarrow \{y, x\} \in E(G)$.

- **Weighted Graph (undirected or directed)**
  - A graph in which a number (weight) is assigned to each edge.
  - Such weights might represent for example costs, lengths or capacities, or…
  - Weighted Graphs arise in many contexts, e.g. in shortest path problems.

G=(V,E)	Undirected Edges	Directed Edges
Unweighted Edges		
Weighted Edges		

BASICS…



Overview on Simple Graphs and Digraphs

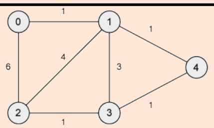
- **Undirected Graph**

  Remark: Edges have no orientation ⟶ $\{x, y\} \in E(G) \Leftrightarrow \{y, x\} \in E(G)$.

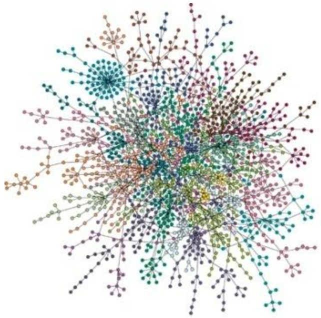
- **Directed Graph**

  Remark: Edges have orientation ⟶ $\{x, y\} \in E(G) \not\Leftrightarrow \{y, x\} \in E(G)$.

- **Weighted Graph (undirected or directed)**
  - A graph in which a number (weight) is assigned to each edge.
  - Such weights might represent for example costs, lengths or capacities, or…
  - Weighted Graphs arise in many contexts, e.g. in shortest path problems.

G=(V,E)	Undirected Edges	Directed Edges
Unweighted Edges		
Weighted Edges		

Basics…



Overview on Simple Graphs and Digraphs

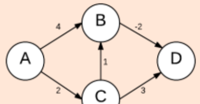
- **Undirected Graph**

  Remark: Edges have no orientation $\longrightarrow \{x, y\} \in E(G) \Leftrightarrow \{y, x\} \in E(G)$.

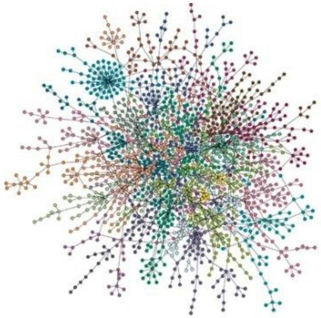
- **Directed Graph**

  Remark: Edges have orientation $\longrightarrow \{x, y\} \in E(G) \not\Leftrightarrow \{y, x\} \in E(G)$.

- **Weighted Graph (undirected or directed)**
  - A graph in which a number (weight) is assigned to each edge.
  - Such weights might represent for example costs, lengths or capacities, or…
  - Weighted Graphs arise in many contexts, e.g. in shortest path problems.

G=(V,E)	Undirected Edges	Directed Edges
Unweighted Edges		
Weighted Edges		

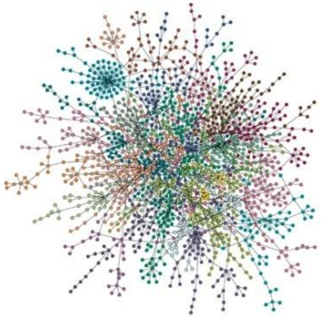
BASICS…



Graph Artifacts – Clique

- A clique is a subset of vertices of an undirected graph such that every two distinct vertices in the clique are adjacent; that is, its induced subgraph is complete.
- A **clique**, C, in an undirected graph $G = (V, E)$ is a subset of the vertices, $C \subseteq V$, such that every two distinct vertices are adjacent, or , equivalently, the induced subgraph of G induced by C is a complete graph.

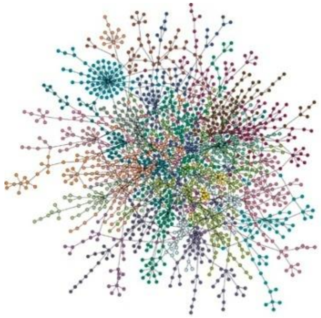
Basics…



Graph Artifacts – Clique

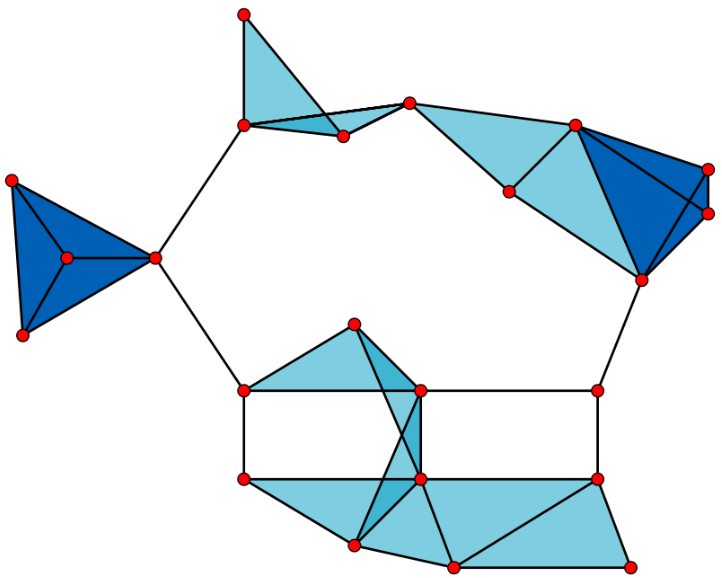
- A clique is a subset of vertices of an undirected graph such that every two distinct vertices in the clique are adjacent; that is, its induced subgraph is complete.
- A **clique**, C, in an undirected graph $G = (V, E)$ is a subset of the vertices, $C \subseteq V$, such that every two distinct vertices are adjacent, or , equivalently, the induced subgraph of G induced by C is a complete graph.

BASICS…

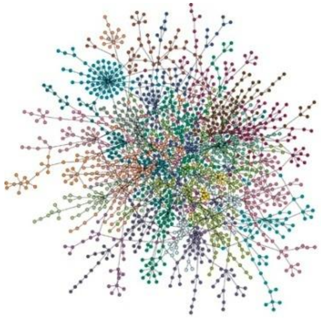


Graph Artifacts – Clique

- A clique is a subset of vertices of an undirected graph such that every two distinct vertices in the clique are adjacent; that is, its induced subgraph is complete.
- A **clique**, C, in an undirected graph $G = (V, E)$ is a subset of the vertices, $C \subseteq V$, such that every two distinct vertices are adjacent, or , equivalently, the induced subgraph of G induced by C is a complete graph.

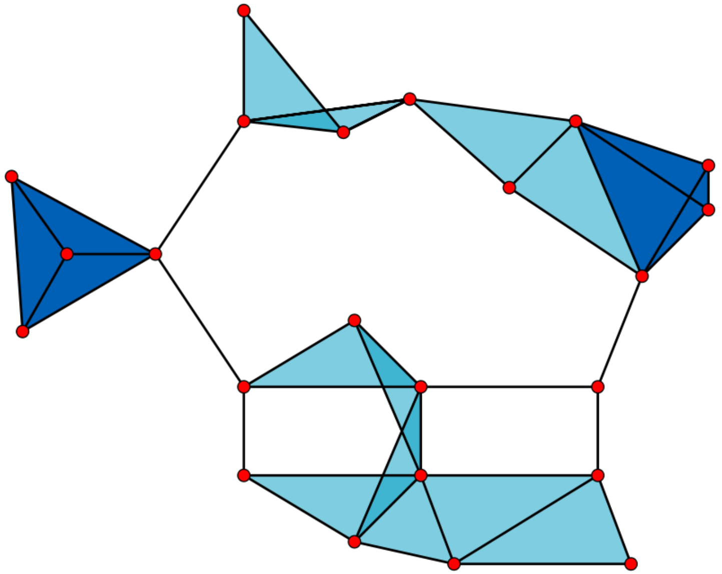


Basics…



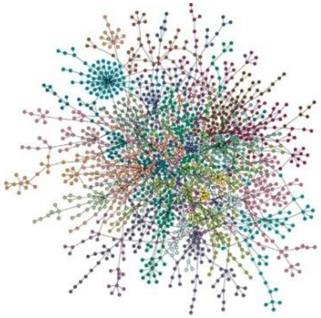
Graph Artifacts – Clique

- A clique is a subset of vertices of an undirected graph such that every two distinct vertices in the clique are adjacent; that is, its induced subgraph is complete.
- A **clique**, C, in an undirected graph $G = (V, E)$ is a subset of the vertices, $C \subseteq V$, such that every two distinct vertices are adjacent, or , equivalently, the induced subgraph of G induced by C is a complete graph.



- 23 × 1-vertex cliques
- 42 × 2-vertex cliques
- 19 × 3-vertex cliques
- 2 × 4-vertex cliques

Basics…

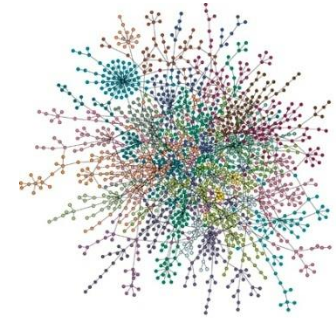


Graph Artifacts – Clique

- A clique is a subset of vertices of an undirected graph such that every two distinct vertices in the clique are adjacent; that is, its induced subgraph is complete.
- A **clique**, C, in an undirected graph $G = (V, E)$ is a subset of the vertices, $C \subseteq V$, such that every two distinct vertices are adjacent, or , equivalently, the induced subgraph of G induced by C is a complete graph.

- A **maximal clique** is a clique that cannot be extended by including one more adjacent vertex, that is, a clique which does not exist exclusively within the vertex set of a larger clique.
- A **maximum clique** of a graph, G, is a clique, such that there is no clique with more vertices.
  - The **clique number $\boldsymbol{\omega(G)}$** of a graph G is the number of vertices in a **maximum clique** in G.

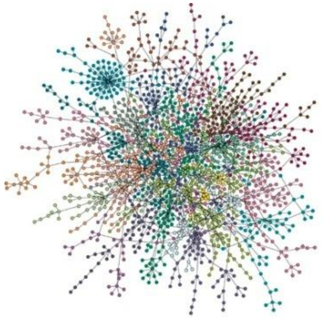
Basics…



- **Graph Artifacts – Clique**
  - A clique is a subset of vertices of an undirected graph such that every two distinct vertices in the clique are adjacent; that is, its induced subgraph is complete.
  - A **clique**, C, in an undirected graph $G = (V, E)$ is a subset of the vertices, $C \subseteq V$, such that every two distinct vertices are adjacent, or , equivalently, the induced subgraph of G induced by C is a complete graph.

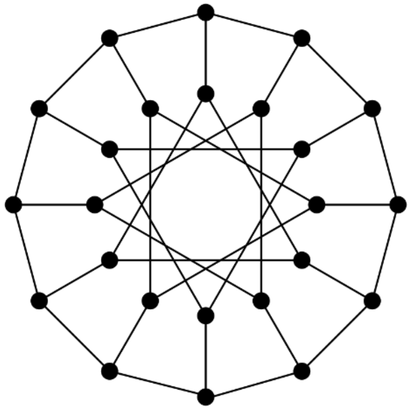
  - The **intersection number** of G is the smallest number of cliques that together cover all edges of G.
  - The **clique cover number** of G is the smallest number of cliques of G whose union covers the set of vertices V of the graph.
  - A **maximum clique transversal** of a graph is a subset of vertices with the property that each maximum clique of the graph contains at least one vertex in the subset.

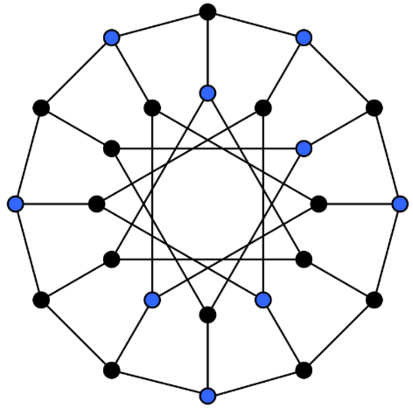
Basics…



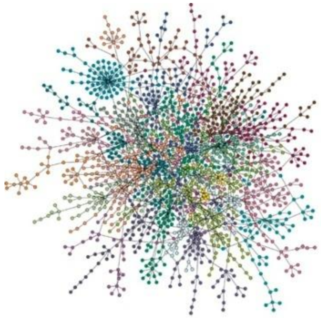
Graph Artifacts – Independent Set

- An **independent set**, **stable set**, coclique or anticlique is a set of vertices in a graph, no two of which are adjacent.
- An **independent set** it is a set S of vertices such that for every two vertices in S, there is no edge connecting the two.
- A **maximal independent set** is either an independent set such that adding any other vertex to the set forces the set to contain an edge or the set of all vertices of the empty graph.
- A **maximum independent set** is an independent set of largest possible size for a given graph G. This size is called the **independence number of G $\alpha(G)$**.

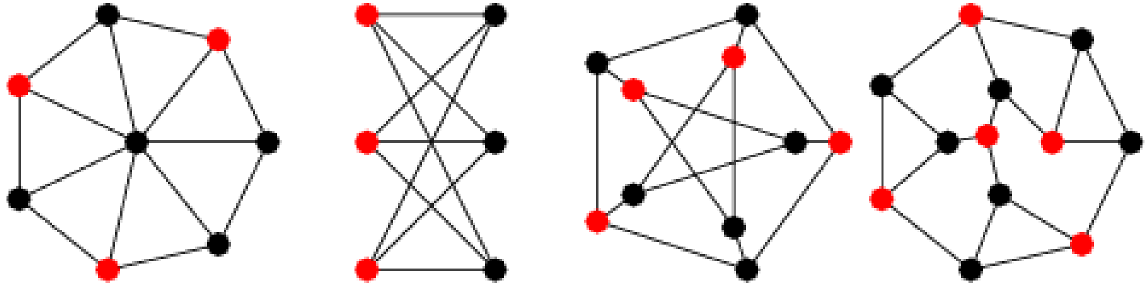




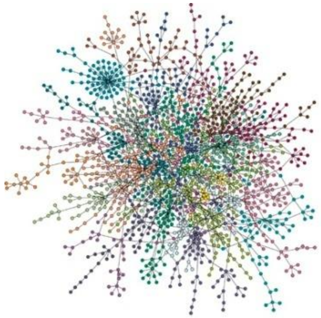
Basics…



- **Graph Artifacts – Independent Set**
  - An **independent set**, **stable set**, coclique or anticlique is a set of vertices in a graph, no two of which are adjacent.
  - An **independent set** it is a set S of vertices such that for every two vertices in S, there is no edge connecting the two.
  - A **maximal independent set** is either an independent set such that adding any other vertex to the set forces the set to contain an edge or the set of all vertices of the empty graph.
  - A **maximum independent set** is an independent set of largest possible size for a given graph G. This size is called the **independence number of G $\alpha(G)$**.

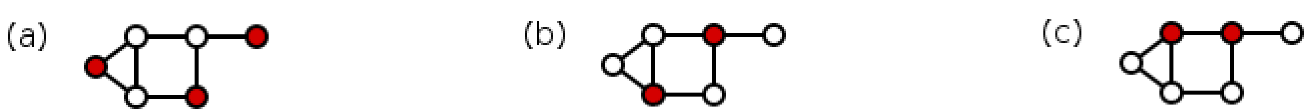


Basics…



Graph Artifacts – Independent Set

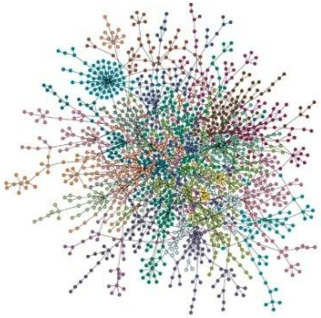
- An **independent set**, **stable set**, coclique or anticlique is a set of vertices in a graph, no two of which are adjacent.
- An **independent set** it is a set S of vertices such that for every two vertices in S, there is no edge connecting the two.
- A **maximal independent set** is either an independent set such that adding any other vertex to the set forces the set to contain an edge or the set of all vertices of the empty graph.
- A **maximum independent set** is an independent set of largest possible size for a given graph G. This size is called the **independence number of G $\alpha(G)$**.
- A **dominating set** for a graph $G = (V, E)$ is a subset D of V such that every vertex not in D is adjacent to at least one member of D.
- The **domination number $\gamma(G)$** is the number of vertices in a smallest dominating set for G.

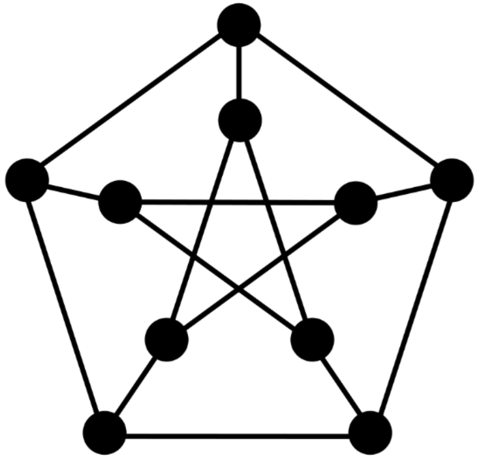


BASICS…

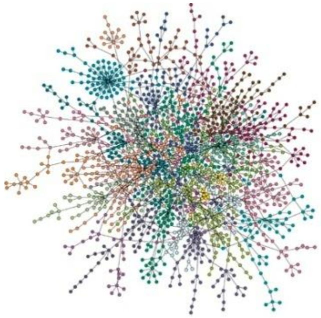
Graph Artifacts – Coloring

- Graph coloring is a special case of graph labeling; it is an assignment colors to elements of a graph subject to certain constraints.
- A coloring of a graph is almost always a **proper** vertex coloring, namely a labeling of the graph's vertices with colors such that no two vertices sharing the same edge have the same color.
- A coloring using at most k colors is called a **(proper) *k*-coloring**.



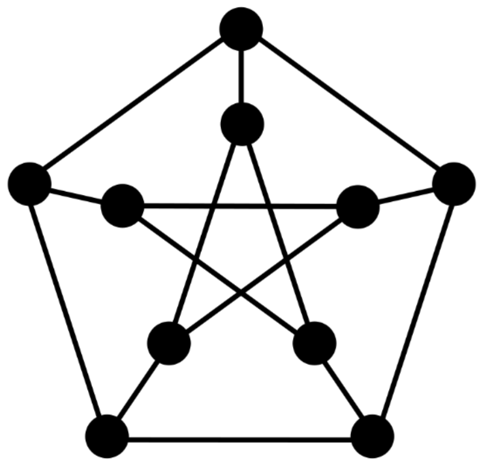


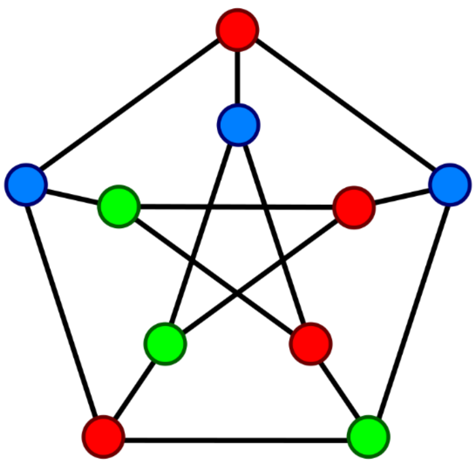
BASICS…



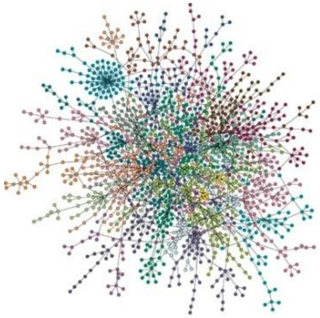
Graph Artifacts – Coloring

- Graph coloring is a special case of graph labeling; it is an assignment colors to elements of a graph subject to certain constraints.
- A coloring of a graph is almost always a **proper** vertex coloring, namely a labeling of the graph's vertices with colors such that no two vertices sharing the same edge have the same color.
- A coloring using at most k colors is called a **(proper) *k*-coloring**.





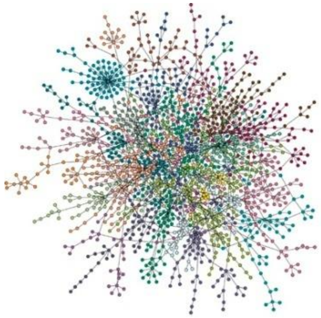
BASICS…



Graph Artifacts – Coloring

- Graph coloring is a special case of graph labeling; it is an assignment colors to elements of a graph subject to certain constraints.
- A coloring of a graph is almost always a **proper vertex coloring**, namely a labeling of the graph's vertices with colors such that no two vertices sharing the same edge have the same color.
- A coloring using at most k colors is called a **(proper) k-coloring**.
- The smallest number of colors needed to color a graph G is called its **chromatic number** $\chi(G)$

Basics…

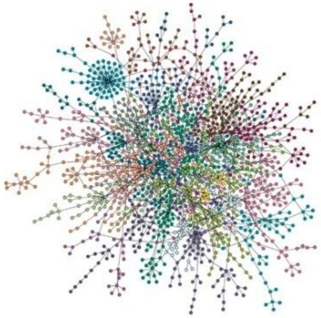


Graph Artifacts – Coloring

- Graph coloring is a special case of graph labeling; it is an assignment colors to elements of a graph subject to certain constraints.
- A coloring of a graph is almost always a **proper vertex coloring**, namely a labeling of the graph's vertices with colors such that no two vertices sharing the same edge have the same color.
- A coloring using at most k colors is called a **(proper) k-coloring**.
- The smallest number of colors needed to color a graph G is called its **chromatic number $\chi(G)$**

- A graph that can be assigned a (proper) k-coloring is **k-colorable**, and it is k-chromatic if its chromatic number is exactly k
- A subset of vertices assigned to the same color is called a color class, every such class forms an

BASICS…

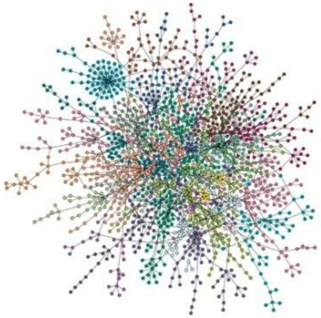


Graph Artifacts – Coloring

- Graph coloring is a special case of graph labeling; it is an assignment colors to elements of a graph subject to certain constraints.
- A coloring of a graph is almost always a **proper vertex coloring**, namely a labeling of the graph's vertices with colors such that no two vertices sharing the same edge have the same color.
- A coloring using at most k colors is called a **(proper) k-coloring**.
- The smallest number of colors needed to color a graph G is called its **chromatic number $\chi(G)$**

- A graph that can be assigned a (proper) k-coloring is **k-colorable**, and it is k-chromatic if its chromatic number is exactly k
- A subset of vertices assigned to the same color is called a color class, every such class forms an independent set.
- a k-coloring is the same as a partition of the vertex set into k independent sets, and the terms k-partite and k-colorable have the same meaning

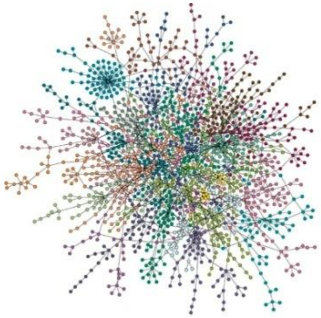
BASICS…



- **Graph Artifacts – Coloring**
  - Graph coloring is a special case of graph labeling; it is an assignment colors to elements of a graph subject to certain constraints.
  - A coloring of a graph is almost always a **proper vertex coloring**, namely a labeling of the graph's vertices with colors such that no two vertices sharing the same edge have the same color.
  - A coloring using at most k colors is called a **(proper) k-coloring**.
  - The smallest number of colors needed to color a graph G is called its **chromatic number $\chi(G)$**

  - Assigning distinct colors to distinct vertices always yields a proper coloring,
    - $1 \leq \chi(\mathrm{G}) \leq n$

Basics…

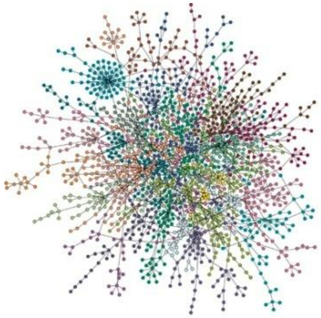


Graph Artifacts – Coloring

- Graph coloring is a special case of graph labeling; it is an assignment colors to elements of a graph subject to certain constraints.
- A coloring of a graph is almost always a **proper vertex coloring**, namely a labeling of the graph's vertices with colors such that no two vertices sharing the same edge have the same color.
- A coloring using at most k colors is called a **(proper) k-coloring**.
- The smallest number of colors needed to color a graph G is called its **chromatic number $\chi(G)$**

- The only graphs that can be 1-colored are edgeless graphs. A complete graph K_n of n vertices requires $\chi(K_n) = n$ colors.
- In an optimal coloring there must be at least one of the graph's m edges between every pair of color classes,
  - $\chi(G) \cdot (\chi(G) - 1) \leq \chi(\mathrm{G}) \leq 2\mathrm{m}$

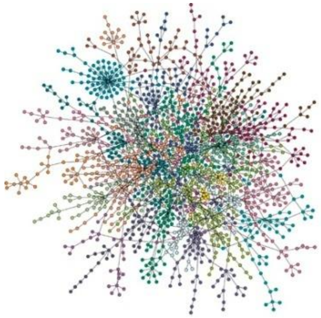
BASICS…



- **Graph Artifacts – Coloring**
  - Graph coloring is a special case of graph labeling; it is an assignment colors to elements of a graph subject to certain constraints.
  - A coloring of a graph is almost always a **proper vertex coloring**, namely a labeling of the graph's vertices with colors such that no two vertices sharing the same edge have the same color.
  - A coloring using at most k colors is called a **(proper) k-coloring**.
  - The smallest number of colors needed to color a graph G is called its **chromatic number $\chi(G)$**

  - If G contains a clique of size k, then at least k colors are needed to color that clique; in other words, the chromatic number is …

Basics…

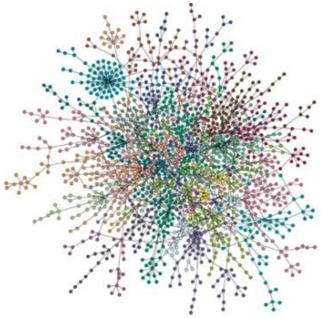


Graph Artifacts – Coloring

- Graph coloring is a special case of graph labeling; it is an assignment colors to elements of a graph subject to certain constraints.
- A coloring of a graph is almost always a **proper vertex coloring**, namely a labeling of the graph's vertices with colors such that no two vertices sharing the same edge have the same color.
- A coloring using at most k colors is called a **(proper) $\boldsymbol{k}$-coloring**.
- The smallest number of colors needed to color a graph G is called its **chromatic number** $\chi(G)$

- If G contains a clique of size k, then at least k colors are needed to color that clique; in other words, the chromatic number is at least the clique number
  - $\chi(G) \geq \omega(\mathrm{G})$

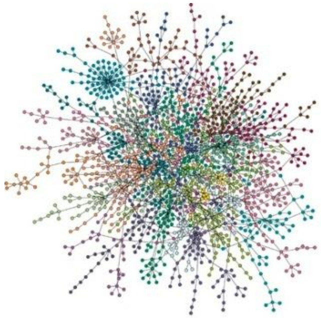
Basics…



- **Graph Artifacts – Isomorphism**
  - An isomorphism of graphs G and H is a bijection between the vertex sets of G and H such that any two vertices u and v of G are adjacent in G if and only if $f(u)$ and $f(v)$ are adjacent in H:

$$f\colon V(g) \rightarrow V(H)$$

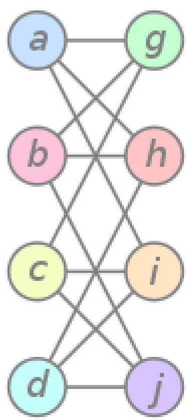
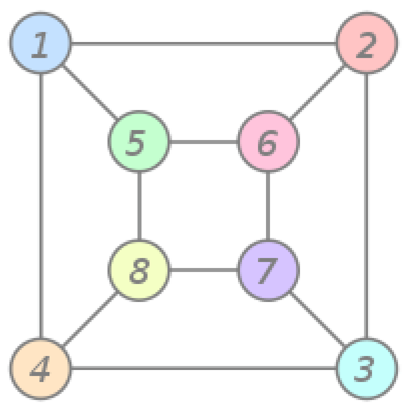
BASICS…



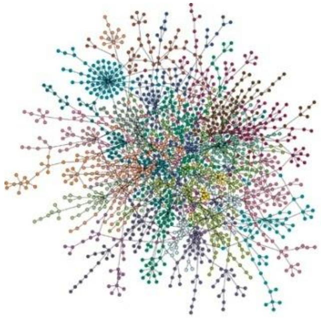
Graph Artifacts – Isomorphism

- An isomorphism of graphs G and H is a bijection between the vertex sets of G and H such that any two vertices u and v of G are adjacent in G if and only if $f(u)$ and $f(v)$ are adjacent in H:

$$f\colon V(g) \rightarrow V(H)$$

Graph G	Graph H	An isomorphism between G and H
		$f(a) = 1$ $f(b) = 6$ $f(c) = 8$ $f(d) = 3$ $f(g) = 5$ $f(h) = 2$ $f(i) = 4$ $f(j) = 7$

BASICS…



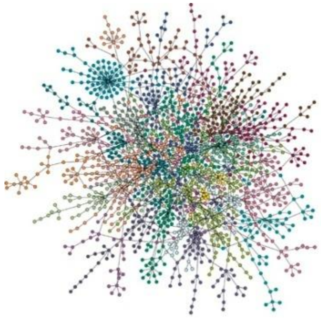
Graph Artifacts – Isomorphism

- An isomorphism of graphs G and H is a bijection between the vertex sets of G and H such that any two vertices u and v of G are adjacent in G if and only if $f(u)$ and $f(v)$ are adjacent in H:

$$f : V(g) \rightarrow V(H)$$

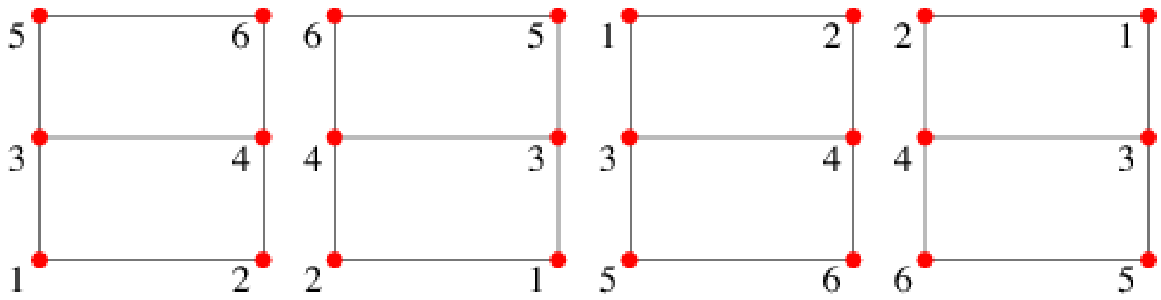
- This kind of bijection is commonly described as "edge-preserving bijection", in accordance with the general notion of isomorphism being a structure-preserving bijection
- If an isomorphism exists between two graphs, then the graphs are called **isomorphic** and denoted as $G \simeq H$.
- In the case when the bijection is a mapping of a graph onto itself, i.e., when G and H are one and the same graph, the bijection is called an **automorphism of G**.

Basics…

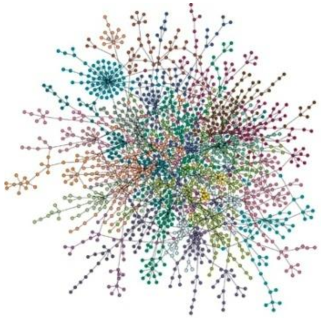


Graph Artifacts – Automorphism

- An automorphism of a graph is a form of symmetry in which the graph is mapped onto itself while preserving the edge–vertex connectivity.
- An **Automorphism** of a graph $G = (V, E)$ is a permutation σ of the vertex set V, such that the pair of vertices (u, v) form an edge if and only if the pair $(\sigma(u), \sigma(v))$ also form an edge (i.e., it is a graph isomorphism from G to itself).

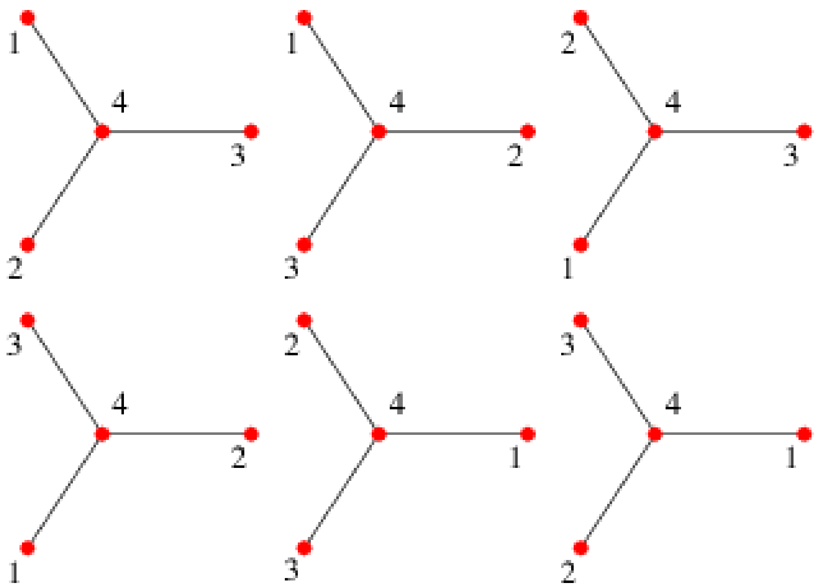


Basics…

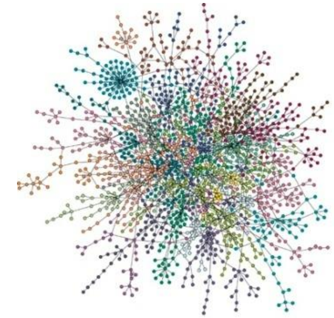


Graph Artifacts – Automorphism

- An automorphism of a graph is a form of symmetry in which the graph is mapped onto itself while preserving the edge–vertex connectivity.
- An **Automorphism** of a graph $G = (V, E)$ is a permutation σ of the vertex set V, such that the pair of vertices (u, v) form an edge if and only if the pair $(\sigma(u), \sigma(v))$ also form an edge (i.e., it is a graph isomorphism from G to itself).



Basics…

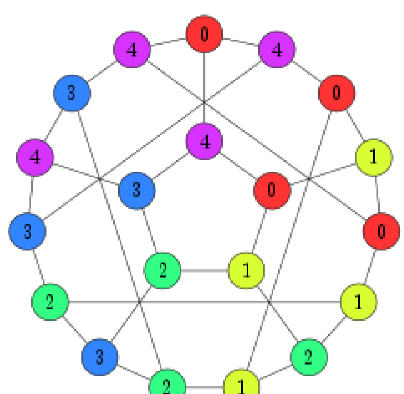


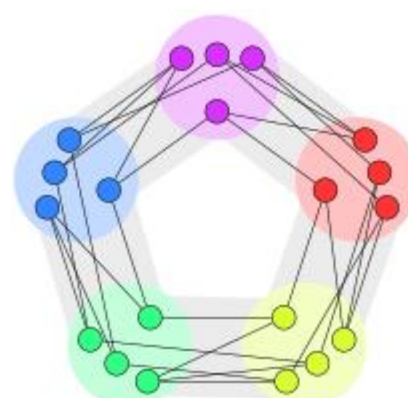
Graph Artifacts – Homomorphism

- A graph homomorphism is a mapping between two graphs that respects their structure.
- **Homomorphism** is a function between the vertex sets of two graphs that maps adjacent vertices to adjacent vertices
- A graph homomorphism f from a graph $G = (V(G), E(G))$ to a graph $H = (V(H), E(H))$, written $f : G \rightarrow H$ is a function from $V(G)$ to $V(H)$ that maps endpoints of each edge in G to endpoints of an edge in H.

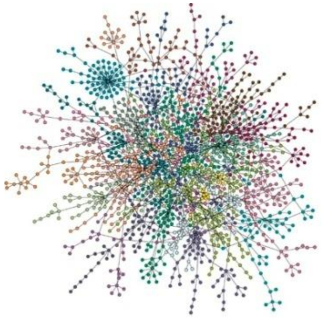
$$\{u, v\} \in E(G) \rightarrow \{f(u), f(v)\} \in E(H), \forall\, u, v \in V(G).$$

- If there exists any homomorphism from G to H, then G is said to be homomorphic to H or H −colorable



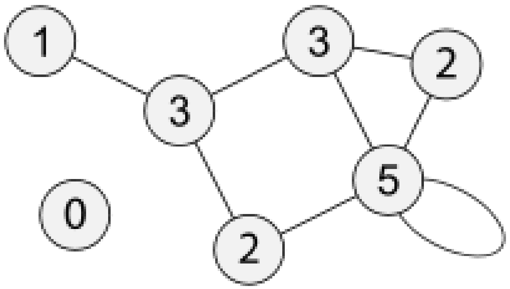


Basics…



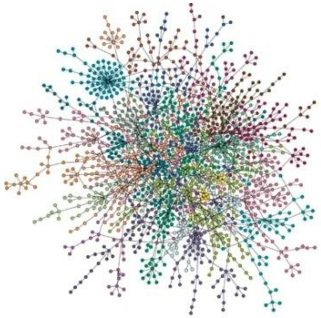
Graph Artifacts – Degree Sequence

- The **degree sequence** S of an undirected graph is the non-increasing sequence of its vertex degrees
- The degree sequence is a graph **invariant** so isomorphic graphs have the same degree sequence.

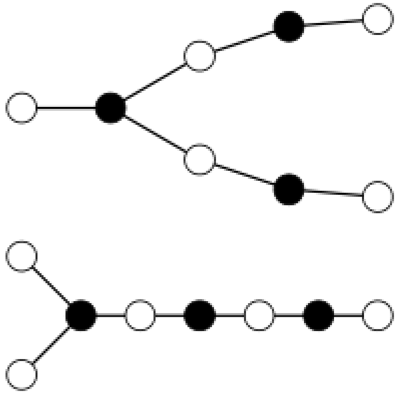


(5, 3, 3, 2, 2, 1, 0)

BASICS…

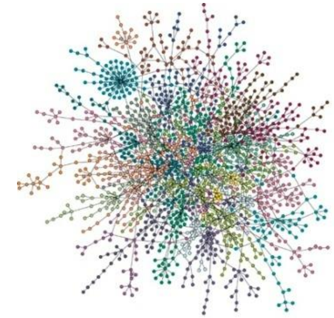


- **Graph Artifacts – Degree Sequence**
  - The **degree sequence** S of an undirected graph is the non-increasing sequence of its vertex degrees
  - The degree sequence is a graph **invariant** so isomorphic graphs have the same degree sequence.
  - The degree sequence does not, in general, uniquely identify a graph; in some cases, non-isomorphic graphs have the same degree sequence.



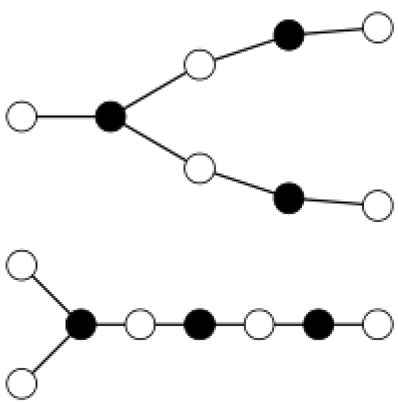
(3, 2, 2, 2, 2, 1, 1, 1)

Basics…



Graph Artifacts – Degree Sequence

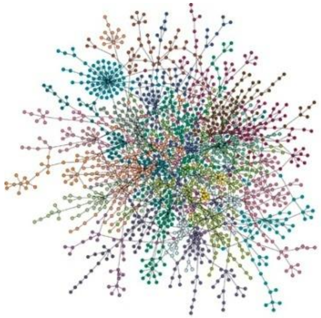
- The **degree sequence** S of an undirected graph is the non-increasing sequence of its vertex degrees
- The degree sequence is a graph **invariant** so isomorphic graphs have the same degree sequence.
- The degree sequence does not, in general, uniquely identify a graph; in some cases, non-isomorphic graphs have the same degree sequence.



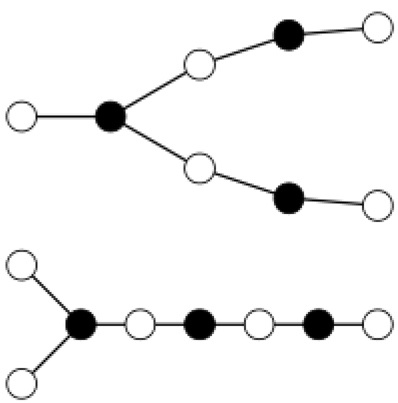
(3, 2, 2, 2, 2, 1, 1, 1)

(2,2,2,2,2,2) ???

Basics…

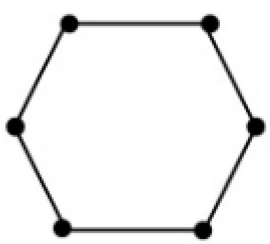


- **Graph Artifacts – Degree Sequence**
  - The **degree sequence** S of an undirected graph is the non-increasing sequence of its vertex degrees
  - The degree sequence is a graph **invariant** so isomorphic graphs have the same degree sequence.
  - The degree sequence does not, in general, uniquely identify a graph; in some cases, non-isomorphic graphs have the same degree sequence.



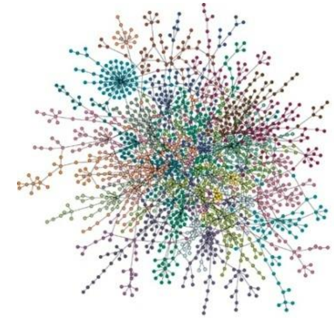
(3, 2, 2, 2, 2, 1, 1, 1)

(2,2,2,2,2,2) ???



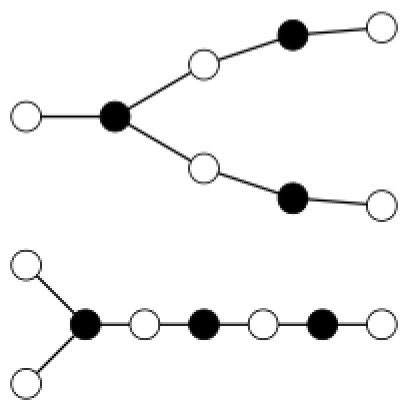
C_6

Basics…



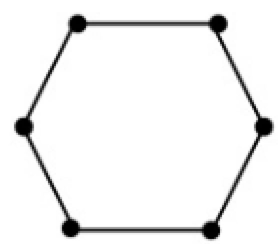
Graph Artifacts – Degree Sequence

- The **degree sequence** *S* of an undirected graph is the non-increasing sequence of its vertex degrees
- The degree sequence is a graph **invariant** so isomorphic graphs have the same degree sequence.
- The degree sequence does not, in general, uniquely identify a graph; in some cases, non-isomorphic graphs have the same degree sequence.

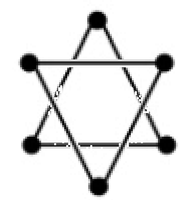


(3, 2, 2, 2, 2, 1, 1, 1)

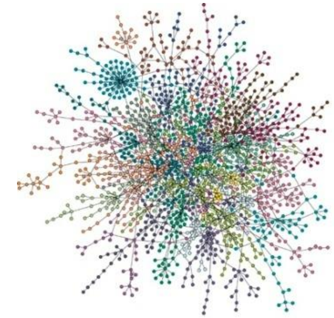
(2,2,2,2,2,2) ???



C_6

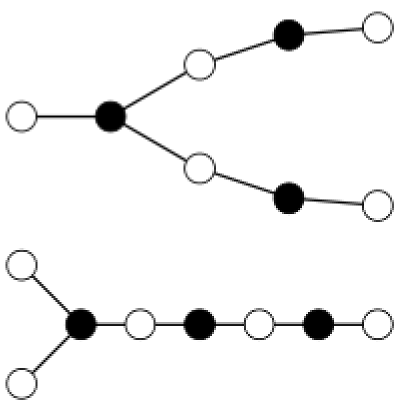


Basics…



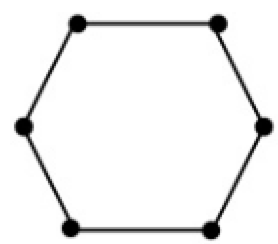
Graph Artifacts – Degree Sequence

- The **degree sequence** *S* of an undirected graph is the non-increasing sequence of its vertex degrees
- The degree sequence is a graph **invariant** so isomorphic graphs have the same degree sequence.
- The degree sequence does not, in general, uniquely identify a graph; in some cases, non-isomorphic graphs have the same degree sequence.

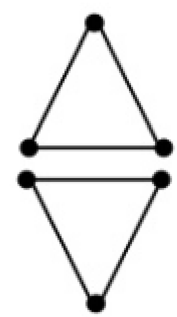


(3, 2, 2, 2, 2, 1, 1, 1)

(2,2,2,2,2,2) ???

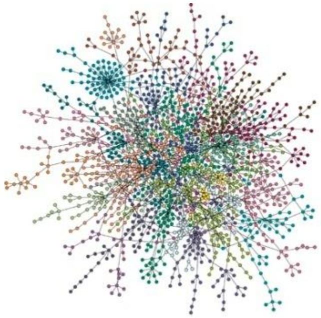


C_6



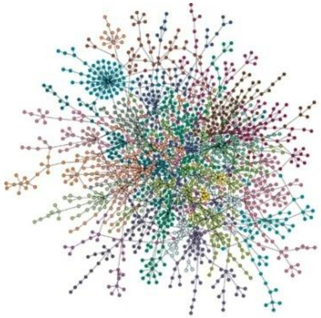
$2C_3$

Basics…



- **Graph Artifacts – Degree Sequence**
  - The **degree sequence** S of an undirected graph is the non-increasing sequence of its vertex degrees
  - The **degree sequence problem** is the problem of finding some or all graphs with the degree sequence S being a given non-increasing sequence of positive integers.
  - Trailing zeroes may be ignored since they are trivially realized by adding an appropriate number of isolated vertices to the graph.
  - A sequence S for which the degree sequence problem has a solution, is called a **graphic sequence**.
    - Any sequence with an odd sum, cannot be **realized** as the degree sequence of a graph (**realization**).
    - If a sequence has an even sum, it is the degree sequence of a multigraph.
    - The construction of such a graph is straightforward: connect vertices with odd degrees in pairs by a matching, and fill out the remaining even degree counts by self-loops.

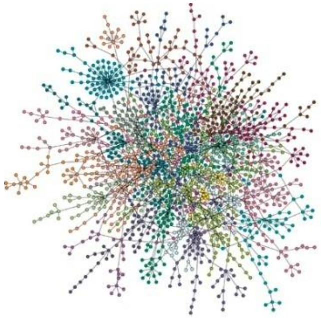
Basics…



Graph Artifacts – Degree Sequence

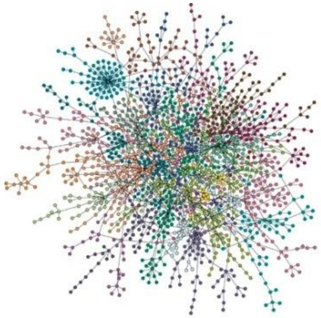
- The **degree sequence** S of an undirected graph is the non-increasing sequence of its vertex degrees
- The **degree sequence problem** is the problem of finding some or all graphs with the degree sequence S being a given non-increasing sequence of positive integers.
- Trailing zeroes may be ignored since they are trivially realized by adding an appropriate number of isolated vertices to the graph.
- A sequence S for which the degree sequence problem has a solution, is called a **graphic sequence**.
  - The question of whether a given degree sequence S can be **realized** by a simple graph is more challenging.
  - **graph enumeration**
    - finding the number of graphs with a given degree sequence S

Basics…



- **Graph Artifacts – Degree Sequence**
  - The **degree sequence** S of an undirected graph is the non-increasing sequence of its vertex degrees
  - The **degree sequence problem** is the problem of finding some or all graphs with the degree sequence S being a given non-increasing sequence of positive integers.
  - Trailing zeroes may be ignored since they are trivially realized by adding an appropriate number of isolated vertices to the graph.
  - A sequence S for which the degree sequence problem has a solution, is called a **graphic sequence**.
    - The question of whether a given degree sequence S can be **realized** by a simple graph is more challenging.
    - **graph realization problem**
      - Erdős–Gallai theorem / Havel–Hakimi algorithm.

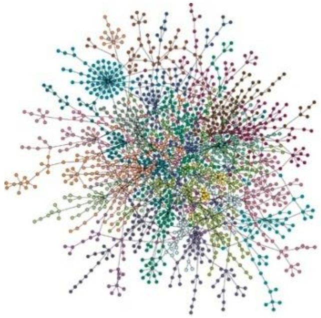
Basics…



Graph Artifacts – Degree Sequence

- The **degree sequence** *S* of an undirected graph is the non-increasing sequence of its vertex degrees
- The **degree sequence problem** is the problem of finding some or all graphs with the degree sequence *S* being a given non-increasing sequence of positive integers.
- Trailing zeroes may be ignored since they are trivially realized by adding an appropriate number of isolated vertices to the graph.
- A sequence *S* for which the degree sequence problem has a solution, is called a **graphic sequence**.
  - The question of whether a given degree sequence *S* can be **realized** by a simple graph is more challenging.
  - **graph realization problem**
    - Erdős–Gallai theorem / **Havel–Hakimi algorithm**.

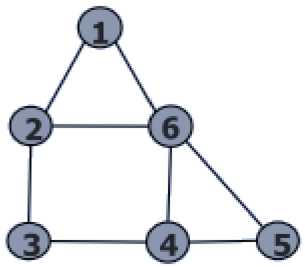
Basics…



- **Graph Artifacts – Degree Sequence**
  - **Havel–Hakimi Theorem**.

A degree sequence $S = d_1, d_2, \dots, d_n$ is graphic, ***iff*** the degree sequence S_1

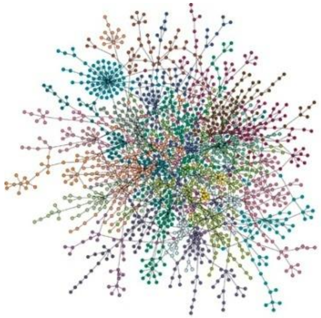
$S_1 = \underbrace{d_2 - 1, d_3 - 1, \dots, d_{d_1+1} - 1}_{d_1}, d_{d_1+2}, \dots, d_n$ is graphic …



$S = (4, \underbrace{3, 3, 2, 2, 2}_{2,\, 2,\, 1,\, 1,\, 2})$

$S = (4, 3, 3, 2, 2, 2)$ ***iff*** $S_1 = (2, 2, 2, 1, 1)$

BASICS…



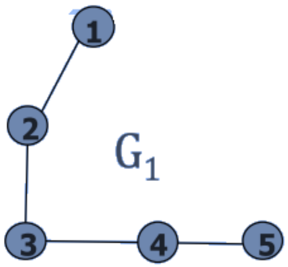
- **Graph Artifacts – Degree Sequence**
  - **Havel–Hakimi Theorem (Proof - $\Longleftarrow$)**

Let $S_1 = d_2 - 1, d_3 - 1, \dots, d_{d_1+1} - 1, d_{d_1+2}, \dots, d_n$ is graphic …
and let G_1 its realization.

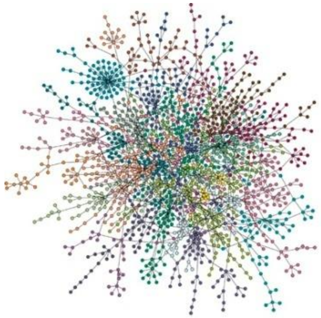
Additionally, let $x_2, x_3, \dots, x_n$ the labels of the n-1 vertices of the graph G_1:

$$d(x_i) = \begin{cases} d_i - 1 & 2 \le i \le d_1 + 1 \\ d_i & d_1 + 2 \le i \le n \end{cases}$$

$S_1 = (2, 2, 1, 1, 2)$



BASICS…



- **Graph Artifacts – Degree Sequence**
  - **Havel–Hakimi Theorem (Proof - $\Leftarrow$)**

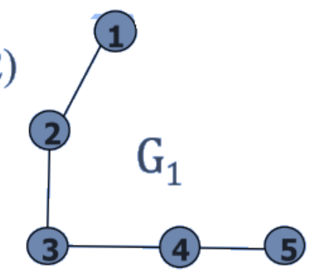
Let $S_1 = d_2 - 1, d_3 - 1, \ldots, d_{d_1+1} - 1, d_{d_1+2}, \ldots, d_n$ is graphic …
and let G_1 its realization.

Additionally, let $x_2, x_3, \ldots, x_n$ the labels of the n-1 vertices of the graph G_1:

2, 4, 3, 5, 1

$$d(x_i) = \begin{cases} d_i - 1 & 2 \leq i \leq d_1 + 1 \\ d_i & d_1 + 2 \leq i \leq n \end{cases}$$

$S_1 = (2, 2, 1, 1, 2)$



Basics…

- **Graph Artifacts – Degree Sequence**
  - **Havel–Hakimi Theorem (Proof - $\Leftarrow$)**

Let $S_1 = d_2 - 1, d_3 - 1, \dots, d_{d_1+1} - 1, d_{d_1+2}, \dots, d_n$ is graphic … and let G_1 its realization.

Additionally, let $x_2, x_3, \dots, x_n$ the labels of the n-1 vertices of the graph G_1:

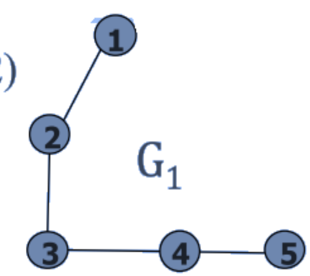
2, 4, 3, 5, 1

$$d(x_i) = \begin{cases} d_i - 1 & 2 \le i \le d_1 + 1 \\ d_i & d_1 + 2 \le i \le n \end{cases}$$

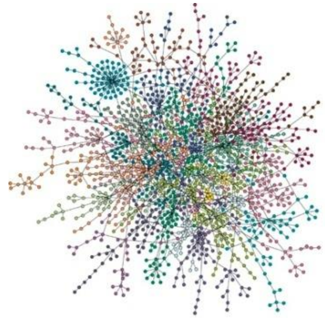
From G_1 we can construct a new graph G as follows:

- insert a new vertex x_1
- with edges $(x_1, x_i)\ \forall\ 2 \le i \le d_i + 1$

$S_1 = (2, 2, 1, 1, 2)$



Basics…



Graph Artifacts – Degree Sequence

- **Havel–Hakimi Theorem (Proof - $\Leftarrow$)**

Let $S_1 = d_2 - 1, d_3 - 1, \dots, d_{d_1+1} - 1, d_{d_1+2}, \dots, d_n$ is graphic …
and let G_1 its realization.

Additionally, let $x_2, x_3, \dots, x_n$ the labels of the n-1 vertices of the graph G_1:

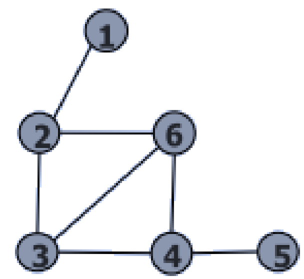
2, 4, 3, 5, 1

$$d(x_i) = \begin{cases} d_i - 1 & 2 \leq i \leq d_1 + 1 \\ d_i & d_1 + 2 \leq i \leq n \end{cases}$$

From G_1 we can construct a new graph G as follows:

- insert a new vertex x_1
- with edges $(x_1, x_i)\ \forall\ 2 \leq i \leq d_i + 1$

The new graph G has S as degree sequence,
and hence the S is graphic!



2, 4, 3, 5, 1

Basics…

- **Graph Artifacts – Degree Sequence**
  - **Havel–Hakimi Theorem (Proof - $\Longleftarrow$)**

Let $S_1 = d_2 - 1, d_3 - 1, \dots, d_{d_1+1} - 1, d_{d_1+2}, \dots, d_n$ is graphic …
and let G_1 its realization.

Additionally, let $x_2, x_3, \dots, x_n$ the labels of the n-1 vertices of the graph G_1:

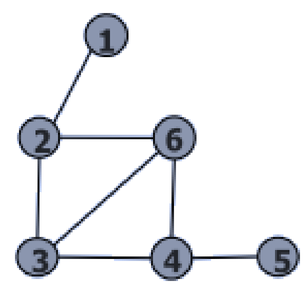
2, 4, 3, 5, 1

$$d(x_i) = \begin{cases} d_i - 1 & 2 \le i \le d_1 + 1 \\ d_i & d_1 + 2 \le i \le n \end{cases}$$

From G_1 we can construct a new graph G as follows:

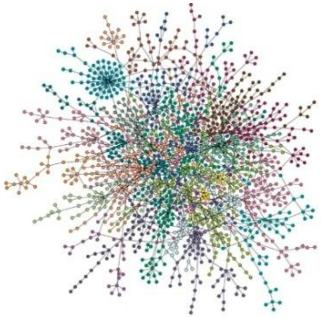
- insert a new vertex x_1
- with edges $(x_1, x_i)\ \forall\ 2 \le i \le d_i + 1$

The new graph G has S as degree sequence,
and hence the S is graphic!



2, 4, 3, 5, 1

BASICS…



- **Graph Artifacts – Degree Sequence**
  - **Havel–Hakimi Theorem (Proof - $\Longrightarrow$)**

Let $S = d_1, d_2, \ldots, d_n$ is a graphic degree sequence…
we will show that S_1 is also a graphic degree sequence.

BASICS…

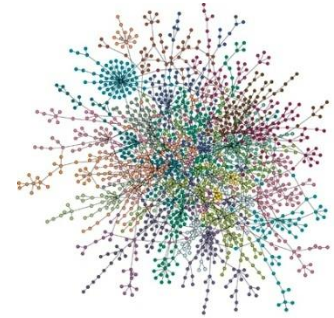
- **Graph Artifacts – Degree Sequence**
  - **Havel–Hakimi Theorem (Proof - $\Longrightarrow$)**

Let $S = d_1, d_2, \dots, d_n$ is a graphic degree sequence…
we will show that S_1 is also a graphic degree sequence.

Since S is graphic $\Rightarrow$ $\exists$ more than one realizations of S

$\Rightarrow$ $\exists$ $\mathrm{G}_1, \mathrm{G}_2, \dots G_k, k \geq 1$ of order n having S as their the degree sequence.

BASICS…



- **Graph Artifacts – Degree Sequence**
  - **Havel–Hakimi Theorem (Proof - $\Longrightarrow$)**

Let $S = d_1, d_2, \dots, d_n$ is a graphic degree sequence…
we will show that S_1 is also a graphic degree sequence.

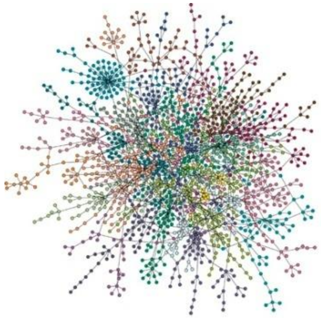
Since S is graphic $\Rightarrow$ $\exists$ more than one realizations of S

$\Rightarrow$ $\exists$ $\mathrm{G}_1, \mathrm{G}_2, \dots G_k, k \geq 1$ of order n having S as their the degree sequence.

From these graphs, let G a graph with a speceific property:

- $V(G) = \{x_1, x_2, \dots, x_n\}$, where $d(x_i) = d_i, \;\; 1 \leq \; i \leq \; n$
- The sum of the degrees of the vertices adjacent to x_1 is maximum

BASICS…



Graph Artifacts – Degree Sequence

- **Havel–Hakimi Theorem (Proof - $\Longrightarrow$)**

Let $S = d_1, d_2, \dots, d_n$ is a graphic degree sequence…
we will show that S_1 is also a graphic degree sequence.

Since S is graphic $\Rightarrow$ $\exists$ more than one realizations of S

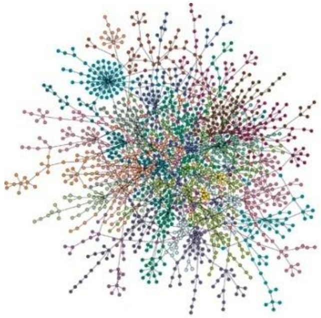
$\Rightarrow$ $\exists$ $\mathrm{G}_1, \mathrm{G}_2, \dots G_k, k \geq 1$ of order n having S as their the degree sequence.

From these graphs, let G a graph with a speceific property:

- $V(G) = \{x_1, x_2, \dots, x_n\}$, where $d(x_i) = d_i,\ \ 1 \leq i \leq n$
- The sum of the degrees of the vertices adjacent to x_1 is maximum

❖ **<u>CLAIM (A)</u>** : In G, vertex x_1 is adjacent to d_1 vertices $x_2, x_3, \dots, x_{d_1+1}$ with degrees $d_2, d_3, \dots, d_{d_1+1}$ with the maximum degrees.

Basics…



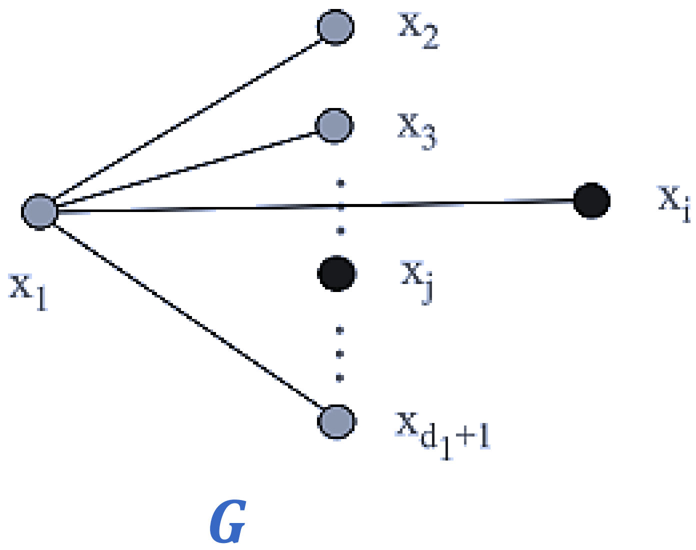
- **Graph Artifacts – Degree Sequence**
  - **Havel–Hakimi Theorem (Proof - $\Longrightarrow$)**

    Assume the opposite: Let x_1 not adjacent to all d_1 vertices with the maximum degrees $d_2, d_3, \dots, d_n$ in graph G.

BASICS…

- **Graph Artifacts – Degree Sequence**
  - **Havel–Hakimi Theorem (Proof - $\Longrightarrow$)**

Assume the opposite: Let x_1 not adjacent to all d_1 vertices with the maximum degrees $d_2, d_3, \ldots, d_n$ in graph G.

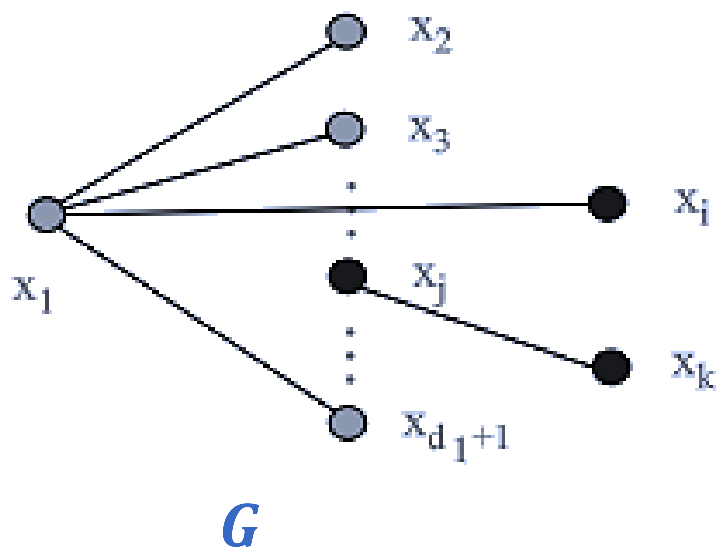


Since $d(x_1) = d_1, \exists\ x_j, x_i \in V(G)$: $d(x_j) > d(x_i)$ (from property of S), and $(x_1, x_j) \notin E(G), (x_1, x_i) \in E(G)$

Basics…

- **Graph Artifacts – Degree Sequence**
  - **Havel–Hakimi Theorem (Proof - $\Longrightarrow$)**

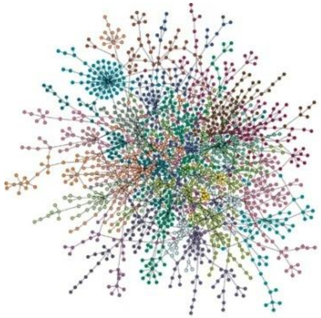
Assume the opposite: Let x_1 not adjacent to all d_1 vertices with the maximum degrees $d_2, d_3, \ldots, d_n$ in graph G.



Since $d(x_j) > d(x_i), \exists\, x_k \in V(G)$:

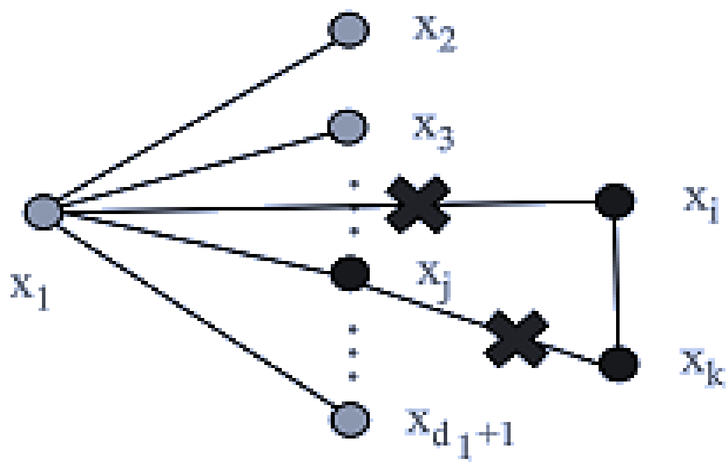
$$(x_j, x_k) \in E(G)$$
$$(x_i, x_k) \notin E(G)$$

BASICS…



- **Graph Artifacts – Degree Sequence**
  - **Havel–Hakimi Theorem (Proof - $\Longrightarrow$)**

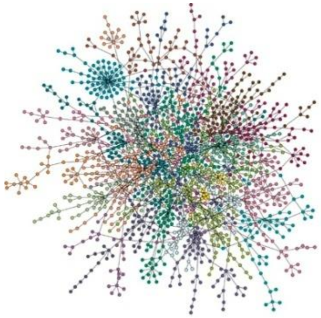
Assume the opposite: Let x_1 not adjacent to all d_1 vertices with the maximum degrees $d_2, d_3, \dots, d_n$ in graph G.



G

Swap edges in $E(G)$:
delete (x_1, x_i) , (x_j, x_k)
insert $(x_1, x_j), (x_i, x_k)$

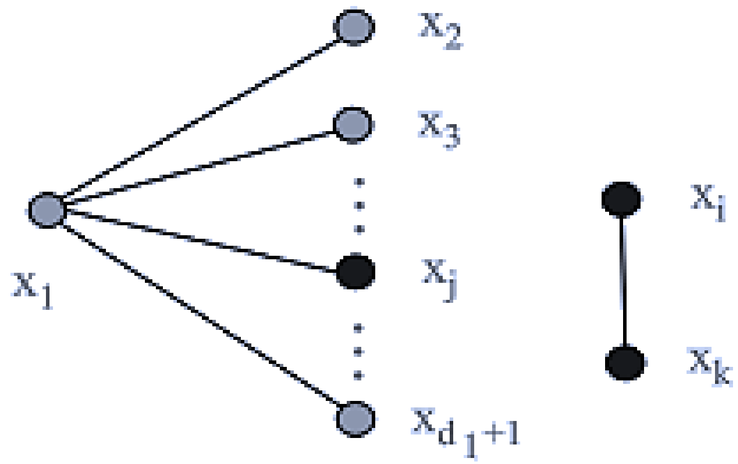
Basics…



Graph Artifacts – Degree Sequence

- **Havel–Hakimi Theorem (Proof - $\Longrightarrow$)**

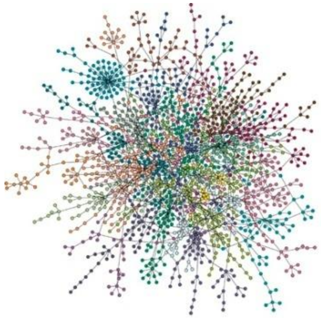
Assume the opposite: Let x_1 not adjacent to all d_1 vertices with the maximum degrees $d_2, d_3, \dots, d_n$ in graph G.



$\boldsymbol{G'}$

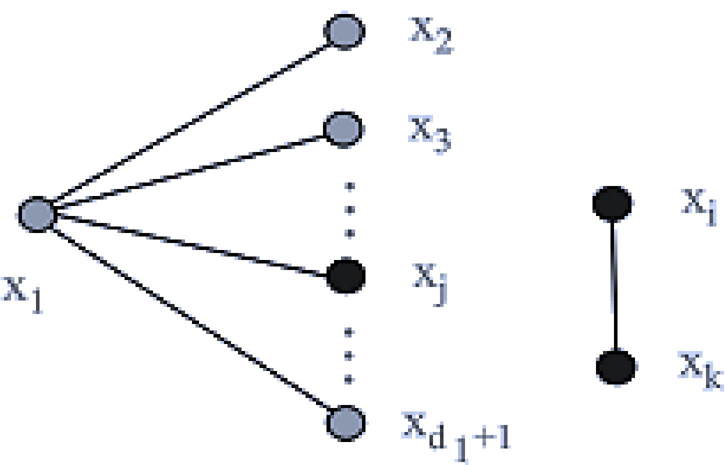
G' has S as degree sequence (like G). In G' the degree sum of the neighbors of x_1 is greater than the one in G $\Longrightarrow$ …

Basics…



- **Graph Artifacts – Degree Sequence**
  - **Havel–Hakimi Theorem (Proof - ⟹)**

Assume the opposite: Let x_1 not adjacent to all d_1 vertices with the maximum degrees $d_2, d_3, \dots, d_n$ in graph G.



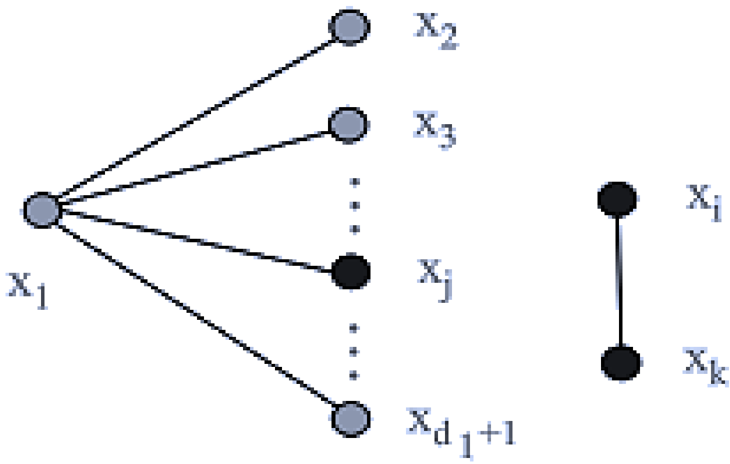
G'

Contradiction in the choice of graph G …
… It holds the **CLAIM (A)**

Basics…

- **Graph Artifacts – Degree Sequence**
  - **Havel–Hakimi Theorem (Proof - $\Longrightarrow$)**

Assume the opposite: Let x_1 not adjacent to all d_1 vertices with the maximum degrees $d_2, d_3, \dots, d_n$ in graph G.

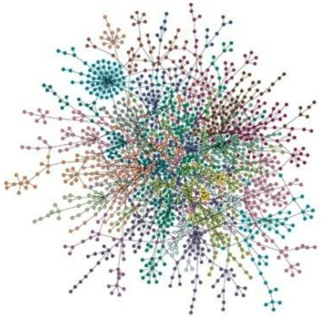


G'

Contradiction in the choice of graph G …
… It holds the **CLAIM (A)**

<u>CLAIM (A)</u>: In G, vertex x_1 is adjacent to $d_1$vertices $x_2, x_3, \dots, x_{d_1+1}$ with degrees $d_2, d_3, \dots, d_{d_1+1}$ with the maximum degrees.

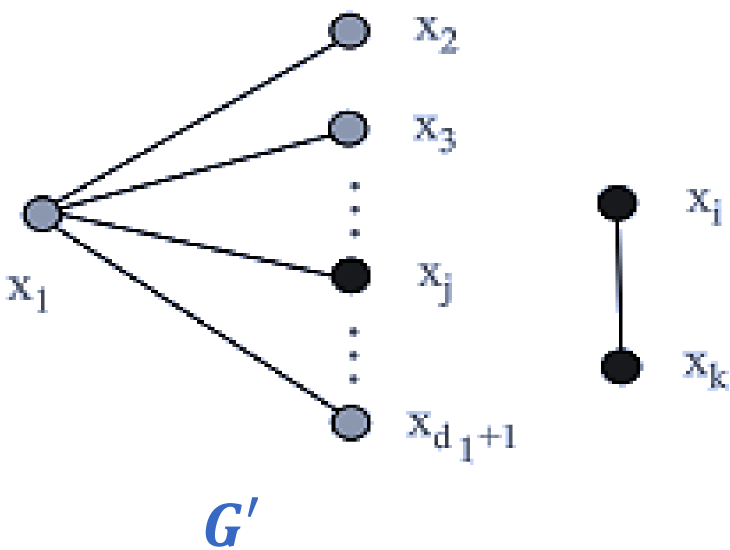
BASICS…



Graph Artifacts – Degree Sequence

- **Havel–Hakimi Theorem (Proof - ⟹)**

Assume the opposite: Let x_1 not adjacent to all d_1 vertices with the maximum degrees $d_2, d_3, \dots, d_n$ in graph G.



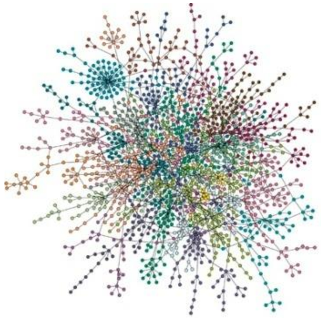
Contradiction in the choice of graph G …

… It holds the **CLAIM (A)**

CLAIM (A): In G, vertex x_1 is adjacent to d_1 vertices $x_2, x_3, \dots, x_{d_1+1}$ with degrees $d_2, d_3, \dots, d_{d_1+1}$ with the maximum degrees.

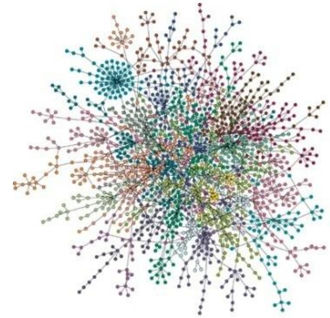
- The graph $G - x_1$ has S_1 as degree sequence and hence S_1 is graphic!

BASICS…



- **Graph Artifacts – Degree Sequence**
  - The **degree sequence** S of an undirected graph is the non-increasing sequence of its vertex degrees
  - Properties of **degree sequence**:
    1. Contain non-negative values.
    2. Values should be less than n.
    3. The cardinality of the odd values to be even.

Basics…



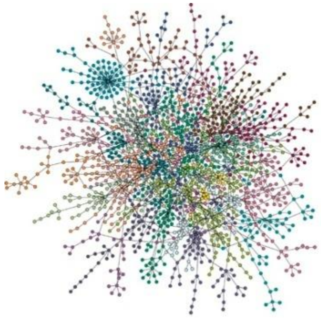
- **Graph Artifacts – Degree Sequence**
  - The **degree sequence** S of an undirected graph is the non-increasing sequence of its vertex degrees
  - **Graphic Sequence Recognition Algorithm**

    Input: A sequence S of n non-negative integers

    Output: Answer (YES/NO) if the sequence is graphic.

    1. If $\exists\, d \in S : d > n - 1 \rightarrow NO$
    2. If the sequence is formed by zeroes $\rightarrow YES$
    3. If the sequence contains at least one negative value $\rightarrow NO$
    4. If necessary, rearrange the sequence as to be non-increasing
    5. Delete the first element d_1 and decrease by one the elements after d_1.
    6. Go to step 2

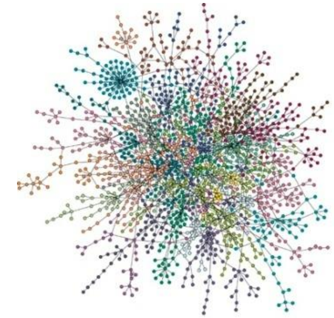
BASICS…



- **Graph Artifacts – Degree Sequence**
  - The **degree sequence** S of an undirected graph is the non-increasing sequence of its vertex degrees
  - **Graphic Sequence Recognition Algorithm**

    Input: A sequence S of n non-negative integers
    Output: (YES/NO) if the sequence is graphic.
    1. If $\exists\, d \in S : d > n - 1 \rightarrow NO$
    2. If the sequence is formed by zeroes $\rightarrow YES$
    3. If the sequence contains at least one negative value $\rightarrow NO$
    4. If necessary, rearrange the sequence as to be non-increasing
    5. Delete the first element d_1 and decrease by one the $|d_1|$ elements.
    6. Go to step 2

BASICS…



Graph Artifacts – Degree Sequence

- The **degree sequence** S of an undirected graph is the non-increasing sequence of its vertex degrees

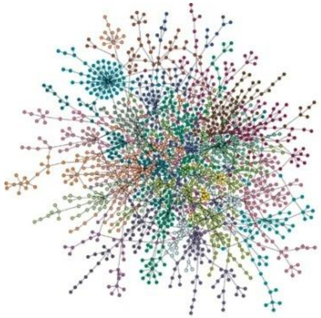
- **Graphic Sequence Recognition Algorithm**

Input: A sequence S of n non-negative integers
Output: (YES/NO) if the sequence is graphic.

1. If $\exists\, d \in S : d > n - 1 \rightarrow NO$
2. If the sequence is formed by zeroes $\rightarrow YES$
3. If the sequence contains at least one negative value $\rightarrow NO$
4. If necessary, rearrange the sequence as to be non-increasing
5. Delete the first element d_1 and decrease by one the $|d_1|$ elements.
6. Go to step 2

7,6,5,5,5,4,4,2

BASICS…



Graph Artifacts – Degree Sequence

- The **degree sequence** S of an undirected graph is the non-increasing sequence of its vertex degrees

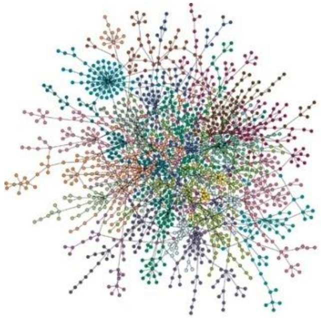
- **Graphic Sequence Recognition Algorithm**

Input: A sequence S of n non-negative integers
Output: (YES/NO) if the sequence is graphic.

1. If $\exists\, d \in S : d > n - 1 \rightarrow NO$
2. If the sequence is formed by zeroes $\rightarrow YES$
3. If the sequence contains at least one negative value $\rightarrow NO$
4. If necessary, rearrange the sequence as to be non-increasing
5. Delete the first element d_1 and decrease by one the $|d_1|$ elements.
6. Go to step 2

7,6,5,5,5,4,4,2

BASICS…



Graph Artifacts – Degree Sequence

- The **degree sequence** S of an undirected graph is the non-increasing sequence of its vertex degrees

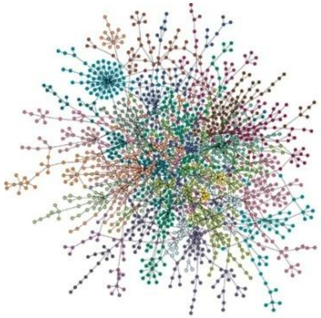
- **Graphic Sequence Recognition Algorithm**

  Input: A sequence S of n non-negative integers
  Output: (YES/NO) if the sequence is graphic.

  1. If $\exists\, d \in S : d > n - 1 \rightarrow NO$
  2. If the sequence is formed by zeroes $\rightarrow YES$
  3. If the sequence contains at least one negative value $\rightarrow NO$
  4. If necessary, rearrange the sequence as to be non-increasing
  5. Delete the first element d_1 and decrease by one the $|d_1|$ elements.
  6. Go to step 2

7,6,5,5,5,4,4,2

Basics…



Graph Artifacts – Degree Sequence

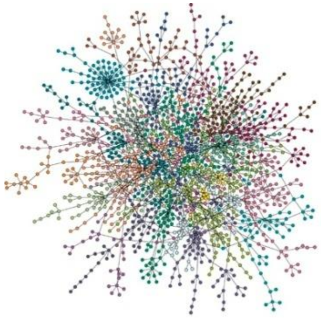
- The **degree sequence** S of an undirected graph is the non-increasing sequence of its vertex degrees
- **Graphic Sequence Recognition Algorithm**

Input: A sequence S of n non-negative integers
Output: (YES/NO) if the sequence is graphic.

1. If $\exists\, d \in S : d > n - 1 \rightarrow NO$
2. If the sequence is formed by zeroes $\rightarrow YES$
3. If the sequence contains at least one negative value $\rightarrow NO$
4. If necessary, rearrange the sequence as to be non-increasing
5. Delete the first element d_1 and decrease by one the $|d_1|$ elements.
6. Go to step 2

7,6,5,5,5,4,4,2

BASICS…



Graph Artifacts – Degree Sequence

- The **degree sequence** S of an undirected graph is the non-increasing sequence of its vertex degrees

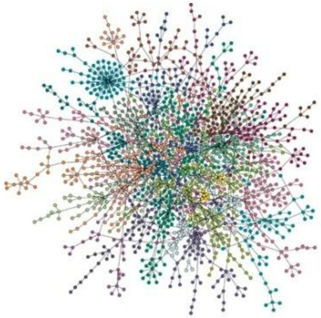
- **Graphic Sequence Recognition Algorithm**

Input: A sequence S of n non-negative integers
Output: (YES/NO) if the sequence is graphic.

1. If $\exists\, d \in S : d > n - 1 \rightarrow NO$
2. If the sequence is formed by zeroes $\rightarrow YES$
3. If the sequence contains at least one negative value $\rightarrow NO$
4. If necessary, rearrange the sequence as to be non-increasing
5. Delete the first element d_1 and decrease by one the $|d_1|$ elements.
6. Go to step 2

7,6,5,5,5,4,4,2

BASICS…



Graph Artifacts – Degree Sequence

- The **degree sequence** S of an undirected graph is the non-increasing sequence of its vertex degrees

- **Graphic Sequence Recognition Algorithm**

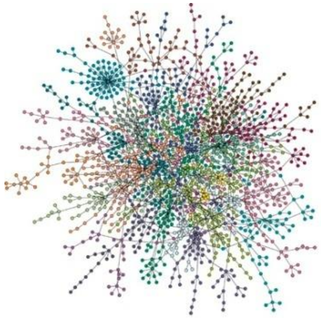
Input: A sequence S of n non-negative integers
Output: (YES/NO) if the sequence is graphic.

1. If $\exists\, d \in S : d > n - 1 \rightarrow NO$
2. If the sequence is formed by zeroes $\rightarrow YES$
3. If the sequence contains at least one negative value $\rightarrow NO$
4. If necessary, rearrange the sequence as to be non-increasing
5. **Delete the first element d_1 and decrease by one the $|d_1|$ elements.**
6. Go to step 2

7,6,5,5,5,4,4,2

R(7), S(|7|) → **5,4,4,4,3,3,1**

Basics…



Graph Artifacts – Degree Sequence

- The **degree sequence** S of an undirected graph is the non-increasing sequence of its vertex degrees

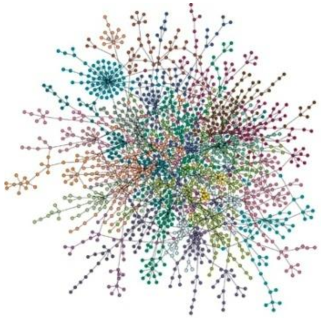
- **Graphic Sequence Recognition Algorithm**

Input: A sequence S of n non-negative integers
Output: (YES/NO) if the sequence is graphic.

1. If $\exists\, d \in S : d > n - 1 \rightarrow NO$
2. If the sequence is formed by zeroes $\rightarrow YES$
3. If the sequence contains at least one negative value $\rightarrow NO$
4. If necessary, rearrange the sequence as to be non-increasing
5. Delete the first element d_1 and decrease by one the $|d_1|$ elements.
6. Go to step 2

7,6,5,5,5,4,4,2
R(7), S(|7|) → **5,4,4,4,3,3,1**

Basics…



Graph Artifacts – Degree Sequence

- The **degree sequence** S of an undirected graph is the non-increasing sequence of its vertex degrees

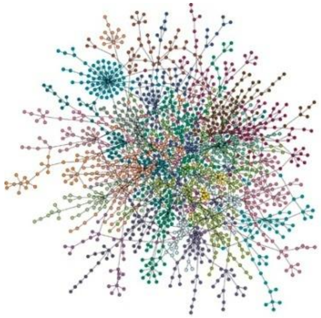
- **Graphic Sequence Recognition Algorithm**

Input: A sequence S of n non-negative integers
Output: (YES/NO) if the sequence is graphic.

1. If $\exists\, d \in S : d > n - 1 \rightarrow NO$
2. If the sequence is formed by zeroes $\rightarrow YES$
3. If the sequence contains at least one negative value $\rightarrow NO$
4. If necessary, rearrange the sequence as to be non-increasing
5. Delete the first element d_1 and decrease by one the $|d_1|$ elements.
6. Go to step 2

7,6,5,5,5,4,4,2
R(7), S(|7|) → **5,4,4,4,3,3,1**

BASICS…



Graph Artifacts – Degree Sequence

- The **degree sequence** S of an undirected graph is the non-increasing sequence of its vertex degrees

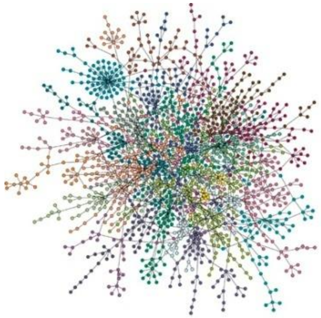
- **Graphic Sequence Recognition Algorithm**

Input: A sequence S of n non-negative integers
Output: (YES/NO) if the sequence is graphic.

1. If $\exists\, d \in S : d > n-1 \rightarrow NO$
2. If the sequence is formed by zeroes $\rightarrow YES$
3. If the sequence contains at least one negative value $\rightarrow NO$
4. If necessary, rearrange the sequence as to be non-increasing
5. Delete the first element d_1 and decrease by one the $|d_1|$ elements.
6. Go to step 2

7,6,5,5,5,4,4,2
R(7), S(|7|) → **5,4,4,4,3,3,1**

Basics…



Graph Artifacts – Degree Sequence

- The **degree sequence** S of an undirected graph is the non-increasing sequence of its vertex degrees

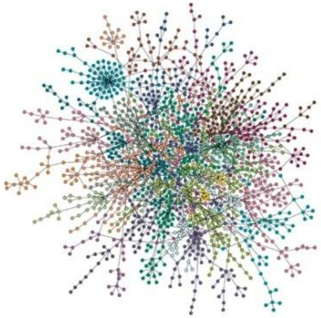
- **Graphic Sequence Recognition Algorithm**

Input: A sequence S of n non-negative integers
Output: (YES/NO) if the sequence is graphic.

1. If $\exists\, d \in S : d > n - 1 \rightarrow NO$
2. If the sequence is formed by zeroes $\rightarrow YES$
3. If the sequence contains at least one negative value $\rightarrow NO$
4. If necessary, rearrange the sequence as to be non-increasing
5. Delete the first element d_1 and decrease by one the $|d_1|$ elements.
6. Go to step 2

	7,6,5,5,5,4,4,2
R(7), S(|7|) →	**5,4,4,4,3,3,1**

Basics…



Graph Artifacts – Degree Sequence

- The **degree sequence** S of an undirected graph is the non-increasing sequence of its vertex degrees

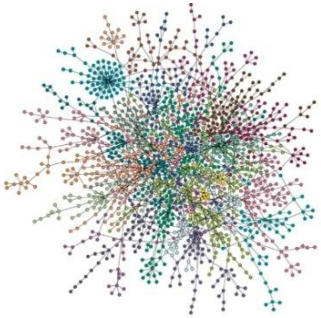
- **Graphic Sequence Recognition Algorithm**

Input: A sequence S of n non-negative integers
Output: (YES/NO) if the sequence is graphic.

1. If $\exists\, d \in S : d > n - 1 \rightarrow NO$
2. If the sequence is formed by zeroes $\rightarrow YES$
3. If the sequence contains at least one negative value $\rightarrow NO$
4. If necessary, rearrange the sequence as to be non-increasing
5. **Delete the first element d_1 and decrease by one the $|d_1|$ elements.**
6. Go to step 2

	7,6,5,5,5,4,4,2
R(7), S(|7|) →	5,4,4,4,3,3,1
R(5), S(|5|) →	3,3,3,2,2,1

Basics…



Graph Artifacts – Degree Sequence

- The **degree sequence** S of an undirected graph is the non-increasing sequence of its vertex degrees

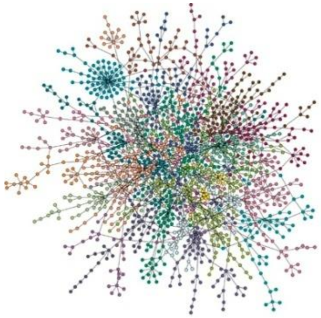
- **Graphic Sequence Recognition Algorithm**

Input: A sequence S of n non-negative integers
Output: (YES/NO) if the sequence is graphic.

1. If $\exists\, d \in S : d > n - 1 \rightarrow NO$
2. If the sequence is formed by zeroes $\rightarrow YES$
3. If the sequence contains at least one negative value $\rightarrow NO$
4. If necessary, rearrange the sequence as to be non-increasing
5. Delete the first element d_1 and decrease by one the $|d_1|$ elements.
6. Go to step 2

	7,6,5,5,5,4,4,2
R(7), S(|7|) →	**5,4,4,4,3,3,1**
R(5), S(|5|) →	**3,3,3,2,2,1**

BASICS…



Graph Artifacts – Degree Sequence

- The **degree sequence** S of an undirected graph is the non-increasing sequence of its vertex degrees

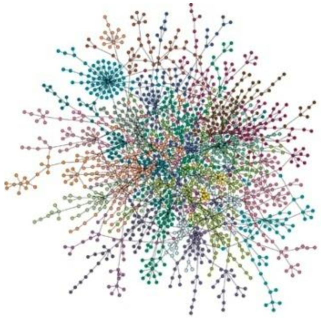
- **Graphic Sequence Recognition Algorithm**

Input: A sequence S of n non-negative integers
Output: (YES/NO) if the sequence is graphic.

1. If $\exists\, d \in S : d > n - 1 \rightarrow NO$
2. If the sequence is formed by zeroes $\rightarrow YES$
3. If the sequence contains at least one negative value $\rightarrow NO$
4. If necessary, rearrange the sequence as to be non-increasing
5. Delete the first element d_1 and decrease by one the $|d_1|$ elements.
6. Go to step 2

7,6,5,5,5,4,4,2
R(7), S(|7|) → **5,4,4,4,3,3,1**
R(5), S(|5|) → **3,3,3,2,2,1**

BASICS…



Graph Artifacts – Degree Sequence

- The **degree sequence** S of an undirected graph is the non-increasing sequence of its vertex degrees

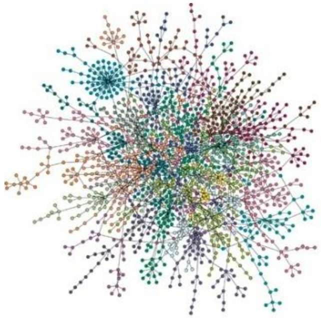
- **Graphic Sequence Recognition Algorithm**

Input: A sequence S of n non-negative integers
Output: (YES/NO) if the sequence is graphic.

1. If $\exists\, d \in S : d > n - 1 \rightarrow NO$
2. If the sequence is formed by zeroes $\rightarrow YES$
3. If the sequence contains at least one negative value $\rightarrow NO$
4. If necessary, rearrange the sequence as to be non-increasing
5. Delete the first element d_1 and decrease by one the $|d_1|$ elements.
6. Go to step 2

7,6,5,5,5,4,4,2
R(7), S(|7|) → **5,4,4,4,3,3,1**
R(5), S(|5|) → **3,3,3,2,2,1**

BASICS…



Graph Artifacts – Degree Sequence

- The **degree sequence** S of an undirected graph is the non-increasing sequence of its vertex degrees

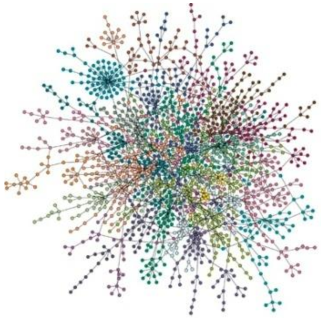
- **Graphic Sequence Recognition Algorithm**

  Input: A sequence S of n non-negative integers
  Output: (YES/NO) if the sequence is graphic.

  1. If $\exists\, d \in S : d > n - 1 \rightarrow NO$
  2. If the sequence is formed by zeroes $\rightarrow YES$
  3. If the sequence contains at least one negative value $\rightarrow NO$
  4. If necessary, rearrange the sequence as to be non-increasing
  5. Delete the first element d_1 and decrease by one the $|d_1|$ elements.
  6. Go to step 2

	7,6,5,5,5,4,4,2
R(7), S(|7|) →	5,4,4,4,3,3,1
R(5), S(|5|) →	3,3,3,2,2,1

BASICS…



Graph Artifacts – Degree Sequence

- The **degree sequence** S of an undirected graph is the non-increasing sequence of its vertex degrees

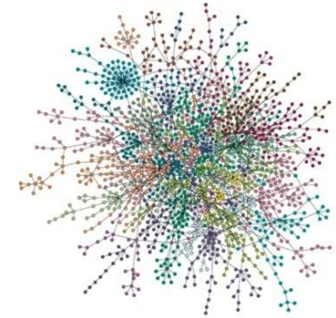
- **Graphic Sequence Recognition Algorithm**

Input: A sequence S of n non-negative integers
Output: (YES/NO) if the sequence is graphic.

1. If $\exists\, d \in S : d > n - 1 \rightarrow NO$
2. If the sequence is formed by zeroes $\rightarrow YES$
3. If the sequence contains at least one negative value $\rightarrow NO$
4. If necessary, rearrange the sequence as to be non-increasing
5. **Delete the first element d_1 and decrease by one the $|d_1|$ elements.**
6. Go to step 2

	7,6,5,5,5,4,4,2
R(7), S(|7|) →	5,4,4,4,3,3,1
R(5), S(|5|) →	3,3,3,2,2,1
R(3), S(|3|) →	2,2,1,2,1

Basics…



Graph Artifacts – Degree Sequence

- The **degree sequence** S of an undirected graph is the non-increasing sequence of its vertex degrees

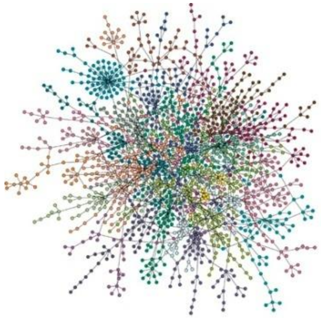
- **Graphic Sequence Recognition Algorithm**

Input: A sequence S of n non-negative integers
Output: (YES/NO) if the sequence is graphic.

1. If $\exists\, d \in S : d > n - 1 \rightarrow NO$
2. If the sequence is formed by zeroes $\rightarrow YES$
3. If the sequence contains at least one negative value $\rightarrow NO$
4. If necessary, rearrange the sequence as to be non-increasing
5. Delete the first element d_1 and decrease by one the $|d_1|$ elements.
6. Go to step 2

	7,6,5,5,5,4,4,2
R(7), S(|7|) →	5,4,4,4,3,3,1
R(5), S(|5|) →	3,3,3,2,2,1
R(3), S(|3|) →	2,2,1,2,1

BASICS…



Graph Artifacts – Degree Sequence

- The **degree sequence** S of an undirected graph is the non-increasing sequence of its vertex degrees

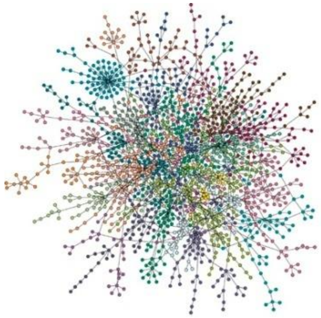
- **Graphic Sequence Recognition Algorithm**

Input: A sequence S of n non-negative integers
Output: (YES/NO) if the sequence is graphic.

1. If $\exists\, d \in S : d > n - 1 \rightarrow NO$
2. If the sequence is formed by zeroes $\rightarrow YES$
3. If the sequence contains at least one negative value $\rightarrow NO$
4. If necessary, rearrange the sequence as to be non-increasing
5. Delete the first element d_1 and decrease by one the $|d_1|$ elements.
6. Go to step 2

	7,6,5,5,5,4,4,2
R(7), S(|7|) →	**5,4,4,4,3,3,1**
R(5), S(|5|) →	**3,3,3,2,2,1**
R(3), S(|3|) →	**2,2,1,2,1**

Basics…



Graph Artifacts – Degree Sequence

- The **degree sequence** S of an undirected graph is the non-increasing sequence of its vertex degrees

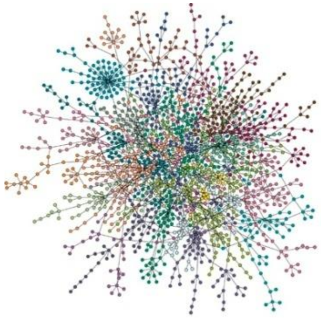
- **Graphic Sequence Recognition Algorithm**

Input: A sequence S of n non-negative integers
Output: (YES/NO) if the sequence is graphic.

1. If $\exists\, d \in S : d > n - 1 \rightarrow NO$
2. If the sequence is formed by zeroes $\rightarrow YES$
3. If the sequence contains at least one negative value $\rightarrow NO$
4. If necessary, rearrange the sequence as to be non-increasing
5. Delete the first element d_1 and decrease by one the $|d_1|$ elements.
6. Go to step 2

	7,6,5,5,5,4,4,2
R(7), S(|7|) →	**5,4,4,4,3,3,1**
R(5), S(|5|) →	**3,3,3,2,2,1**
R(3), S(|3|) →	**2,2,1,2,1**

Basics…



Graph Artifacts – Degree Sequence

- The **degree sequence** S of an undirected graph is the non-increasing sequence of its vertex degrees

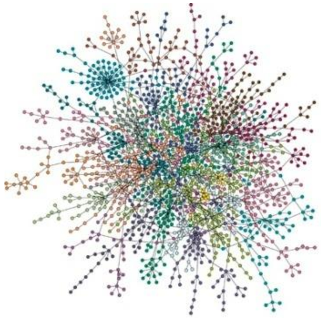
- **Graphic Sequence Recognition Algorithm**

Input: A sequence S of n non-negative integers
Output: (YES/NO) if the sequence is graphic.

1. If $\exists\, d \in S : d > n - 1 \rightarrow NO$
2. If the sequence is formed by zeroes $\rightarrow YES$
3. If the sequence contains at least one negative value $\rightarrow NO$
4. **If necessary, rearrange the sequence as to be non-increasing**
5. Delete the first element d_1 and decrease by one the $|d_1|$ elements.
6. Go to step 2

	7,6,5,5,5,4,4,2
R(7), S(|7|) →	**5,4,4,4,3,3,1**
R(5), S(|5|) →	**3,3,3,2,2,1**
R(3), S(|3|) →	**2,2,1,2,1**
Order (S) →	**2,2,2,1,1**

Basics…



Graph Artifacts – Degree Sequence

- The **degree sequence** S of an undirected graph is the non-increasing sequence of its vertex degrees

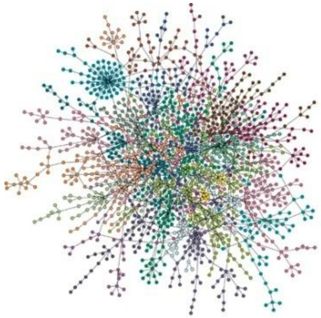
- **Graphic Sequence Recognition Algorithm**

Input: A sequence S of n non-negative integers
Output: (YES/NO) if the sequence is graphic.

1. If $\exists\, d \in S : d > n - 1 \rightarrow NO$
2. If the sequence is formed by zeroes $\rightarrow YES$
3. If the sequence contains at least one negative value $\rightarrow NO$
4. If necessary, rearrange the sequence as to be non-increasing
5. **Delete the first element d_1 and decrease by one the $|d_1|$ elements.**
6. Go to step 2

	7,6,5,5,5,4,4,2
R(7), S(|7|) →	5,4,4,4,3,3,1
R(5), S(|5|) →	3,3,3,2,2,1
R(3), S(|3|) →	2,2,1,2,1
Order (S) →	2,2,2,1,1
R(2), S(|2|) →	1,1,1,1

Basics…



Graph Artifacts – Degree Sequence

- The **degree sequence** S of an undirected graph is the non-increasing sequence of its vertex degrees

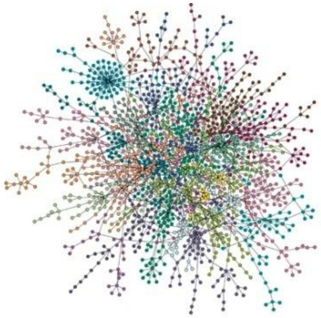
- **Graphic Sequence Recognition Algorithm**

Input: A sequence S of n non-negative integers
Output: (YES/NO) if the sequence is graphic.

1. If $\exists\, d \in S : d > n - 1 \rightarrow NO$
2. If the sequence is formed by zeroes $\rightarrow YES$
3. If the sequence contains at least one negative value $\rightarrow NO$
4. If necessary, rearrange the sequence as to be non-increasing
5. Delete the first element d_1 and decrease by one the $|d_1|$ elements.
6. Go to step 2

	7,6,5,5,5,4,4,2
R(7), S(|7|) →	**5,4,4,4,3,3,1**
R(5), S(|5|) →	**3,3,3,2,2,1**
R(3), S(|3|) →	**2,2,1,2,1**
Order (S) →	**2,2,2,1,1**
R(2), S(|2|) →	**1,1,1,1**

BASICS…



Graph Artifacts – Degree Sequence

- The **degree sequence** S of an undirected graph is the non-increasing sequence of its vertex degrees

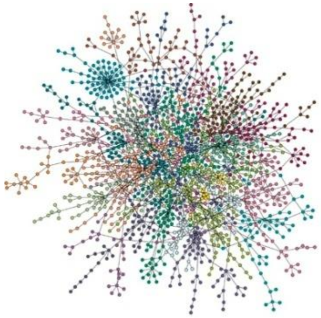
- **Graphic Sequence Recognition Algorithm**

Input: A sequence S of n non-negative integers
Output: (YES/NO) if the sequence is graphic.

1. If $\exists\, d \in S : d > n - 1 \rightarrow NO$
2. If the sequence is formed by zeroes $\rightarrow YES$
3. If the sequence contains at least one negative value $\rightarrow NO$
4. If necessary, rearrange the sequence as to be non-increasing
5. Delete the first element d_1 and decrease by one the $|d_1|$ elements.
6. Go to step 2

	7,6,5,5,5,4,4,2
R(7), S(|7|) →	5,4,4,4,3,3,1
R(5), S(|5|) →	3,3,3,2,2,1
R(3), S(|3|) →	2,2,1,2,1
Order (S) →	2,2,2,1,1
R(2), S(|2|) →	1,1,1,1

Basics…



Graph Artifacts – Degree Sequence

- The **degree sequence** S of an undirected graph is the non-increasing sequence of its vertex degrees

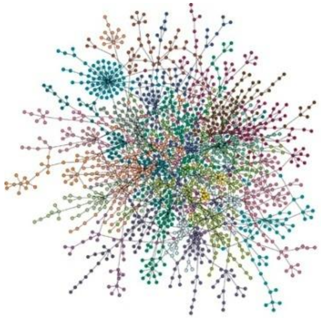
- **Graphic Sequence Recognition Algorithm**

Input: A sequence S of n non-negative integers
Output: (YES/NO) if the sequence is graphic.

1. If $\exists\, d \in S : d > n - 1 \rightarrow NO$
2. If the sequence is formed by zeroes $\rightarrow YES$
3. If the sequence contains at least one negative value $\rightarrow NO$
4. If necessary, rearrange the sequence as to be non-increasing
5. Delete the first element d_1 and decrease by one the $|d_1|$ elements.
6. Go to step 2

	7,6,5,5,5,4,4,2
R(7), S(|7|) →	5,4,4,4,3,3,1
R(5), S(|5|) →	3,3,3,2,2,1
R(3), S(|3|) →	2,2,1,2,1
Order (S) →	2,2,2,1,1
R(2), S(|2|) →	1,1,1,1

Basics…



Graph Artifacts – Degree Sequence

- The **degree sequence** S of an undirected graph is the non-increasing sequence of its vertex degrees

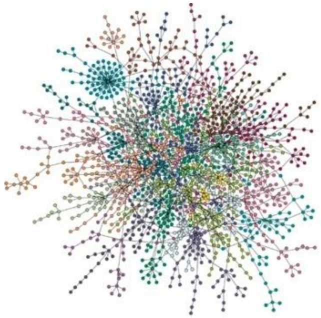
- **Graphic Sequence Recognition Algorithm**

Input: A sequence S of n non-negative integers
Output: (YES/NO) if the sequence is graphic.

1. If $\exists\, d \in S : d > n - 1 \rightarrow NO$
2. If the sequence is formed by zeroes $\rightarrow YES$
3. If the sequence contains at least one negative value $\rightarrow NO$
4. If necessary, rearrange the sequence as to be non-increasing
5. Delete the first element d_1 and decrease by one the $|d_1|$ elements.
6. Go to step 2

	7,6,5,5,5,4,4,2
R(7), S(|7|) →	**5,4,4,4,3,3,1**
R(5), S(|5|) →	**3,3,3,2,2,1**
R(3), S(|3|) →	**2,2,1,2,1**
Order (S) →	**2,2,2,1,1**
R(2), S(|2|) →	**1,1,1,1**

Basics…



Graph Artifacts – Degree Sequence

- The **degree sequence** S of an undirected graph is the non-increasing sequence of its vertex degrees

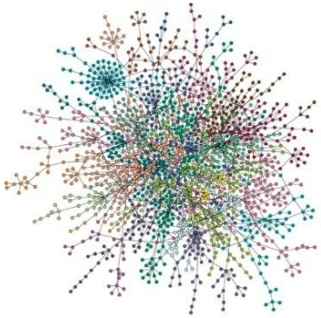
- **Graphic Sequence Recognition Algorithm**

Input: A sequence S of n non-negative integers
Output: (YES/NO) if the sequence is graphic.

1. If $\exists\, d \in S : d > n - 1 \rightarrow NO$
2. If the sequence is formed by zeroes $\rightarrow YES$
3. If the sequence contains at least one negative value $\rightarrow NO$
4. If necessary, rearrange the sequence as to be non-increasing
5. **Delete the first element d_1 and decrease by one the $|d_1|$ elements.**
6. Go to step 2

	7,6,5,5,5,4,4,2
R(7), S(|7|) →	5,4,4,4,3,3,1
R(5), S(|5|) →	3,3,3,2,2,1
R(3), S(|3|) →	2,2,1,2,1
Order (S) →	2,2,2,1,1
R(2), S(|2|) →	1,1,1,1
R(1), S(|1|) →	0,1,1

BASICS…



Graph Artifacts – Degree Sequence

- The **degree sequence** S of an undirected graph is the non-increasing sequence of its vertex degrees

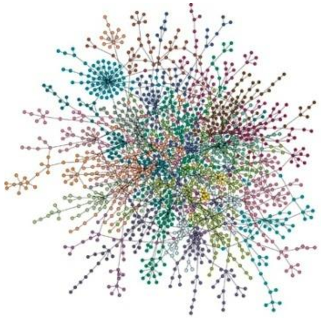
- **Graphic Sequence Recognition Algorithm**

Input: A sequence S of n non-negative integers
Output: (YES/NO) if the sequence is graphic.

1. If $\exists\, d \in S : d > n - 1 \rightarrow NO$
2. If the sequence is formed by zeroes $\rightarrow YES$
3. If the sequence contains at least one negative value $\rightarrow NO$
4. If necessary, rearrange the sequence as to be non-increasing
5. Delete the first element d_1 and decrease by one the $|d_1|$ elements.
6. Go to step 2

	7,6,5,5,5,4,4,2
R(7), S(|7|) →	**5,4,4,4,3,3,1**
R(5), S(|5|) →	**3,3,3,2,2,1**
R(3), S(|3|) →	**2,2,1,2,1**
Order (S) →	**2,2,2,1,1**
R(2), S(|2|) →	**1,1,1,1**
R(1), S(|1|) →	**0,1,1**

Basics…



Graph Artifacts – Degree Sequence

- The **degree sequence** S of an undirected graph is the non-increasing sequence of its vertex degrees

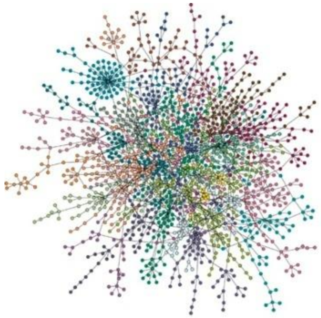
- **Graphic Sequence Recognition Algorithm**

Input: A sequence S of n non-negative integers
Output: (YES/NO) if the sequence is graphic.

1. If $\exists\, d \in S : d > n - 1 \rightarrow NO$
2. If the sequence is formed by zeroes $\rightarrow YES$
3. If the sequence contains at least one negative value $\rightarrow NO$
4. If necessary, rearrange the sequence as to be non-increasing
5. Delete the first element d_1 and decrease by one the $|d_1|$ elements.
6. Go to step 2

	7,6,5,5,5,4,4,2
R(7), S(|7|) →	**5,4,4,4,3,3,1**
R(5), S(|5|) →	**3,3,3,2,2,1**
R(3), S(|3|) →	**2,2,1,2,1**
Order (S) →	**2,2,2,1,1**
R(2), S(|2|) →	**1,1,1,1**
R(1), S(|1|) →	**0,1,1**

Basics…



Graph Artifacts – Degree Sequence

- The **degree sequence** S of an undirected graph is the non-increasing sequence of its vertex degrees

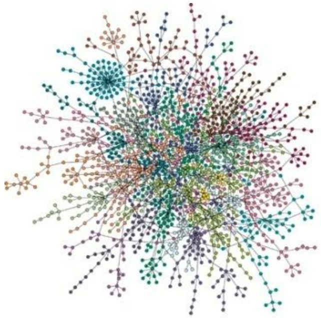
- **Graphic Sequence Recognition Algorithm**

Input: A sequence S of n non-negative integers
Output: (YES/NO) if the sequence is graphic.

1. If $\exists\, d \in S : d > n - 1 \rightarrow NO$
2. If the sequence is formed by zeroes $\rightarrow YES$
3. If the sequence contains at least one negative value $\rightarrow NO$
4. If necessary, rearrange the sequence as to be non-increasing
5. Delete the first element d_1 and decrease by one the $|d_1|$ elements.
6. Go to step 2

	7,6,5,5,5,4,4,2
R(7), S(|7|) →	**5,4,4,4,3,3,1**
R(5), S(|5|) →	**3,3,3,2,2,1**
R(3), S(|3|) →	**2,2,1,2,1**
Order (S) →	**2,2,2,1,1**
R(2), S(|2|) →	**1,1,1,1**
R(1), S(|1|) →	**0,1,1**

BASICS…



Graph Artifacts – Degree Sequence

- The **degree sequence** S of an undirected graph is the non-increasing sequence of its vertex degrees

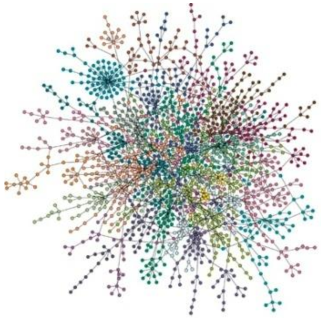
- **Graphic Sequence Recognition Algorithm**

Input: A sequence S of n non-negative integers
Output: (YES/NO) if the sequence is graphic.

1. If $\exists\, d \in S : d > n-1 \rightarrow NO$
2. If the sequence is formed by zeroes $\rightarrow YES$
3. If the sequence contains at least one negative value $\rightarrow NO$
4. **If necessary, rearrange the sequence as to be non-increasing**
5. Delete the first element d_1 and decrease by one the $|d_1|$ elements.
6. Go to step 2

	7,6,5,5,5,4,4,2
R(7), S(|7|) →	5,4,4,4,3,3,1
R(5), S(|5|) →	3,3,3,2,2,1
R(3), S(|3|) →	2,2,1,2,1
Order (S) →	2,2,2,1,1
R(2), S(|2|) →	1,1,1,1
R(1), S(|1|) →	0,1,1
Order (S) →	1,1,0

BASICS…



Graph Artifacts – Degree Sequence

- The **degree sequence** S of an undirected graph is the non-increasing sequence of its vertex degrees

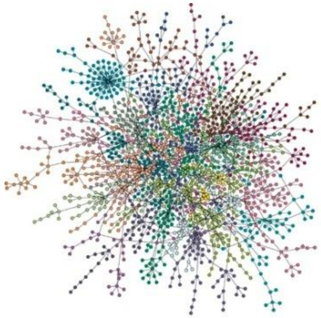
- **Graphic Sequence Recognition Algorithm**

Input: A sequence S of n non-negative integers
Output: (YES/NO) if the sequence is graphic.

1. If $\exists\, d \in S : d > n - 1 \rightarrow NO$
2. If the sequence is formed by zeroes $\rightarrow YES$
3. If the sequence contains at least one negative value $\rightarrow NO$
4. If necessary, rearrange the sequence as to be non-increasing
5. **Delete the first element d_1 and decrease by one the $|d_1|$ elements.**
6. Go to step 2

	7,6,5,5,5,4,4,2
R(7), S(|7|) →	5,4,4,4,3,3,1
R(5), S(|5|) →	3,3,3,2,2,1
R(3), S(|3|) →	2,2,1,2,1
Order (S) →	2,2,2,1,1
R(2), S(|2|) →	1,1,1,1
R(1), S(|1|) →	0,1,1
Order (S) →	1,1,0
R(1), S(|1|) →	0,0

Basics…



Graph Artifacts – Degree Sequence

- The **degree sequence** S of an undirected graph is the non-increasing sequence of its vertex degrees

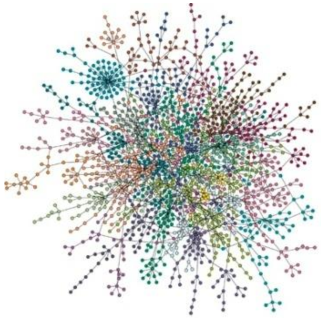
- **Graphic Sequence Recognition Algorithm**

Input: A sequence S of n non-negative integers
Output: (YES/NO) if the sequence is graphic.

1. If $\exists\, d \in S : d > n - 1 \rightarrow NO$
2. If the sequence is formed by zeroes $\rightarrow YES$
3. If the sequence contains at least one negative value $\rightarrow NO$
4. If necessary, rearrange the sequence as to be non-increasing
5. Delete the first element d_1 and decrease by one the $|d_1|$ elements.
6. Go to step 2

	7,6,5,5,5,4,4,2
R(7), S(|7|) →	5,4,4,4,3,3,1
R(5), S(|5|) →	3,3,3,2,2,1
R(3), S(|3|) →	2,2,1,2,1
Order (S) →	2,2,2,1,1
R(2), S(|2|) →	1,1,1,1
R(1), S(|1|) →	0,1,1
Order (S) →	1,1,0
R(1), S(|1|) →	0,0

Basics…



Graph Artifacts – Degree Sequence

- The **degree sequence** S of an undirected graph is the non-increasing sequence of its vertex degrees

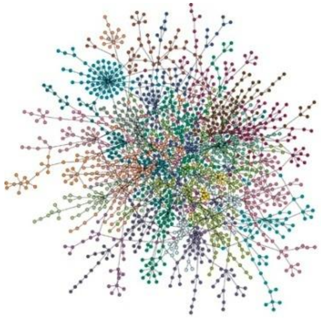
- **Graphic Sequence Recognition Algorithm**

  Input: A sequence S of n non-negative integers
  Output: (YES/NO) if the sequence is graphic.

  1. If $\exists\, d \in S : d > n - 1 \rightarrow NO$
  2. **If the sequence is formed by zeroes**
     $\rightarrow \boldsymbol{YES}$
     $\rightarrow$ **EXIT**
  3. If the sequence contains at least one negative value $\rightarrow NO$
  4. If necessary, rearrange the sequence as to be non-increasing
  5. Delete the first element d_1 and decrease by one the $|d_1|$ elements.
  6. Go to step 2

	7,6,5,5,5,4,4,2
R(7), S(|7|) →	5,4,4,4,3,3,1
R(5), S(|5|) →	3,3,3,2,2,1
R(3), S(|3|) →	2,2,1,2,1
Order (S) →	2,2,2,1,1
R(2), S(|2|) →	1,1,1,1
R(1), S(|1|) →	0,1,1
Order (S) →	1,1,0
R(1), S(|1|) →	0,0

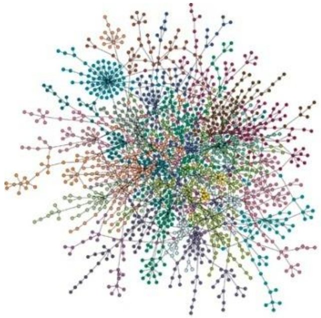
BASICS…



- **Graph Artifacts – Connectivity (Path)**
  - A **path in a simple graph** is a finite or infinite sequence of edges which joins a sequence of vertices which, by most definitions, are all distinct (and since the vertices are distinct, so are the edges).
  - A **walk** is a finite or infinite sequence of edges which joins a sequence of vertices.

    Let $G = (V, E, \phi)$ be a graph. A finite walk is a sequence of edges $(e_1, e_2, \ldots, e_{n-1})$ for which there is a sequence of vertices $(v_1, v_2, \ldots, v_n)$ such that $\phi(e_i) = \{v_i, v_{i+1}\}\ for\ i = 1, 2, \ldots, n-1$. $(v_1, v_2, \ldots, v_n)$ is the vertex sequence of the walk.

BASICS…

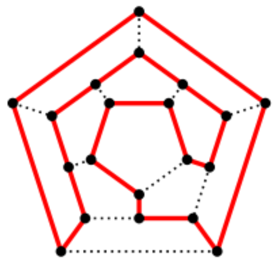


Graph Artifacts – Connectivity (Path)

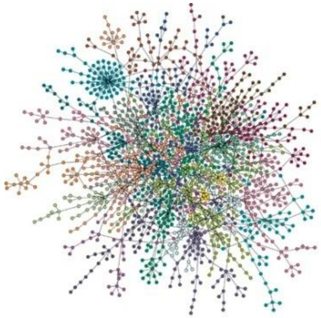
- A **path in a simple graph** is a finite or infinite sequence of edges which joins a sequence of vertices which, by most definitions, are all distinct (and since the vertices are distinct, so are the edges).
- A **walk** is a finite or infinite sequence of edges which joins a sequence of vertices.

  Let $G = (V, E, \phi)$ be a graph. A finite walk is a sequence of edges $(e_1, e_2, \dots, e_{n-1})$ for which there is a sequence of vertices $(v_1, v_2, \dots, v_n)$ such that $\phi(e_i) = \{vi, v_{i+1}\}\, for\ i = 1, 2, \dots, n-1$. $(v_1, v_2, \dots, v_n)$ is the vertex sequence of the walk.
- A **trail** is a walk in which all **edges are distinct**.

A Hamiltonian path (or traceable path) is a path in an undirected or directed graph that visits each vertex exactly once



BASICS…



Graph Artifacts – Connectivity (Path)

- A **path in a simple graph** is a finite or infinite sequence of edges which joins a sequence of vertices which, by most definitions, are all distinct (and since the vertices are distinct, so are the edges).

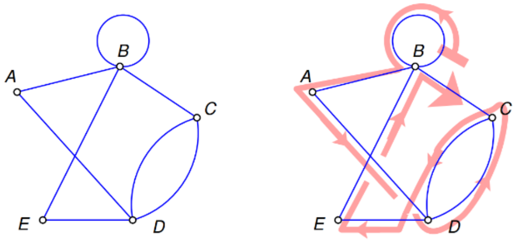
- A **walk** is a finite or infinite sequence of edges which joins a sequence of vertices.

  Let $G = (V, E, \phi)$ be a graph. A finite walk is a sequence of edges $(e_1, e_2, \ldots, e_{n-1})$ for which there is a sequence of vertices $(v_1, v_2, \ldots, v_n)$ such that $\phi(e_i) = \{vi, v_{i+1}\}\, for\ i = 1, 2, \ldots, n - 1$. $(v_1, v_2, \ldots, v_n)$ is the vertex sequence of the walk.

- A trail is a walk in which all edges are distinct.

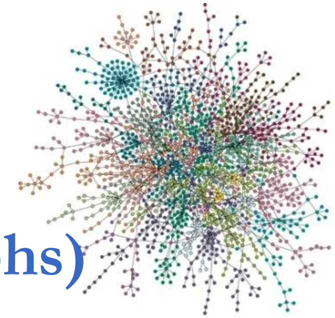
- A **path** is a trail in which all **vertices are distinct**.

An Eulerian trail (or Eulerian path) is a trail in a finite graph that visits every edge exactly once (allowing for revisiting vertices).



An Euler path: BBADCDEBC

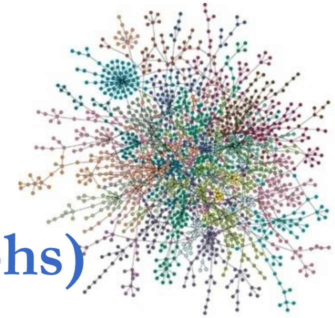
Basics…



- **Graph Artifacts – Connectivity (Connected Graphs)**
  - An undirected graph is **connected** when it has at least one vertex and **there is a path between every pair of vertices**.
  - A graph is connected when it has **exactly one connected component**.
  - In a connected graph, there are **no unreachable vertices**.
  - A graph with just one vertex is connected.
  - An edgeless graph with two or more vertices is disconnected.

  - In an undirected graph, a pair of vertices $\{x, y\} \in V(G)$ is called connected if a path leads from x to y.
  - An undirected graph G is **connected** $iff \; \forall \{x, y\} \in V(G)$: the graph G is connected.
  - A undirected graph G that is not connected is disconnected.
  - An undirected graph G is said to be **disconnected** if there exist two nodes in G such that no path in G has those nodes as endpoints.

Basics…

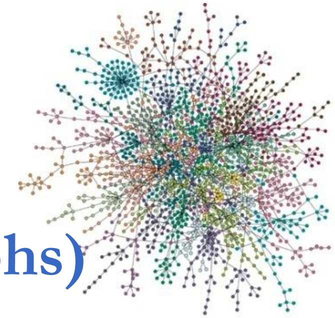


Graph Artifacts – Connectivity (Connected Graphs)

- An undirected graph is **connected** when it has at least one vertex and **there is a path between every pair of vertices**.
- A graph is connected when it has **exactly one connected component**.
- In a connected graph, there are **no unreachable vertices**.
- A graph with just one vertex is connected.
- An edgeless graph with two or more vertices is disconnected.

- In a **directed graph**, a pair of vertices $(x, y) \in V(G)$ is called **strongly connected** if a directed path leads from x to y. If (by making the graph undirected) an undirected path leads from x to y, the pair is **weakly connected** .
- A directed graph is **strongly connected** $iff\ \forall\ \{x, y\} \in V(G)$ the graph G is strongly connected
- A directed graph is **weakly connected** $iff\ \forall\ \{x, y\} \in V(G)$ the graph G is weakly connected

Basics…

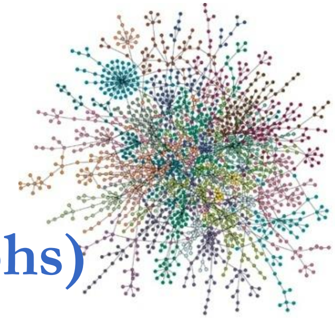


Graph Artifacts – Connectivity (Connected Graphs)

- An undirected graph is **connected** when it has at least one vertex and **there is a path between every pair of vertices**.
- A graph is connected when it has **exactly one connected component**.
- In a connected graph, there are **no unreachable vertices**.
- A graph with just one vertex is connected.
- An edgeless graph with two or more vertices is disconnected.

- A **directed graph** is called **weakly connected** if replacing all of its directed edges with undirected edges produces a connected (undirected) graph.
- It is **unilaterally connected** or unilateral if it contains a directed path from u to v or a directed path from v to u for every pair of vertices u, v.
- It is **strongly connected**, or simply strong if it contains a directed path from u to v and a directed path from v to u for every pair of vertices u, v.
- The **strong components** are the maximal strongly connected subgraphs.

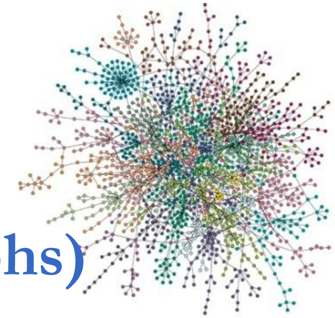
Basics…



Graph Artifacts – Connectivity (Connected Graphs)

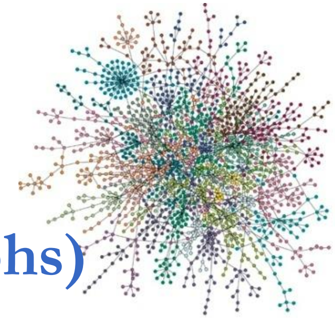
- A **connected component** is a maximal connected subgraph of G. Each vertex belongs to exactly one connected component, as does each edge.
- A **cut**, **vertex cut**, or **separating set** of a connected graph G is a set of vertices whose removal renders G disconnected.
- The **connectivity** or vertex connectivity $\boldsymbol{\kappa(G)}$ (where G is not a complete graph) is the size of a minimal vertex cut.
- A graph is called **k-connected** if its vertex connectivity is k or greater.
- Any graph G is said to be k-connected if it contains at least $k+1$ vertices, but does not contain a set of $k-1$ vertices whose removal disconnects the graph; and $\kappa(G)$ is defined as the largest k such that G is k-connected.
- In particular, a complete graph with n vertices, denoted K_n, has no vertex cuts at all, but $\kappa(K_n) = n - 1$.
- A vertex cut for two vertices u and v is a set of vertices whose removal from the graph disconnects u and v.

Basics…



- **Graph Artifacts – Connectivity (Connected Graphs)**
  - A **connected component** is a maximal connected subgraph of G. Each vertex belongs to exactly one connected component, as does each edge.
  - A **cut**, **vertex cut**, or **separating set** of a connected graph G is a set of vertices whose removal renders G disconnected.
  - The **connectivity** or vertex connectivity $\boldsymbol{\kappa(G)}$ (where G is not a complete graph) is the size of a minimal vertex cut
  - A graph is called **k-connected** if its vertex connectivity is k or greater.
  - The **local connectivity** $\kappa(u, v)$ is the size of a smallest vertex cut separating u and v.
  - Local connectivity is symmetric for undirected graphs; that is, $\kappa(u, v) = \kappa(v, u)$.
  - Except for complete graphs, $\kappa(G)$ equals the minimum of $\kappa(u, v)$ over all nonadjacent pairs of vertices u, v.

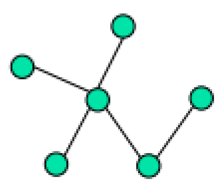
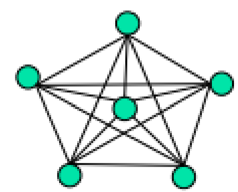
BASICS…



- **Graph Artifacts – Connectivity (Connected Graphs)**
  - **Theorem**: Let G a regular graph of n vertices and m edges, and let $k \geq 1$ the cardinality of its connected components.

    It holds that:

    $$\boldsymbol{n - k \;\leq\; m \;\leq\; (n-k)(n-k+1)/2}$$

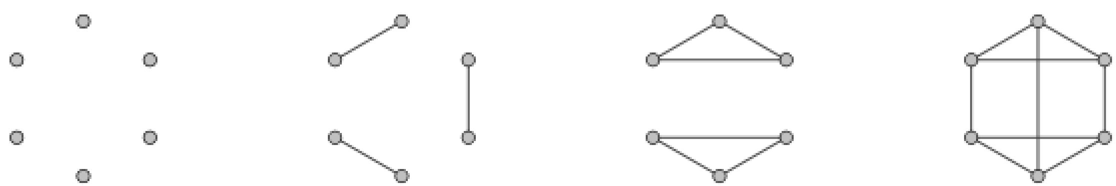
    - $k = 1$ (G connected): $n \;\leq\; m \;\leq\; n(n-1)\,/\,2$
    -  $\leq \; m \; \leq$ 

  - **Corollary**: Every regular graph of n vertices and at least $((n-1)(n-2))/2$ edges is connected.

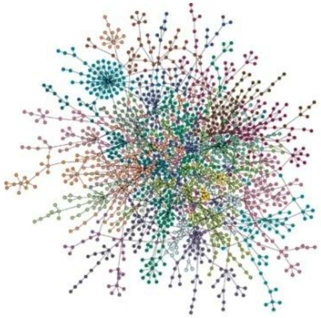
BASICS…

Graph Types – Regular Graphs

- A **regular graph** is a graph where each vertex has the same degree.
- A regular directed graph must also satisfy the stronger condition that the indegree and outdegree of each vertex are equal to each other.
- A regular graph with vertices of degree k is called a k-regular graph or regular graph of degree k.
- Also, from the handshaking lemma, a regular graph of odd degree will contain an even number of vertices.



Basics…



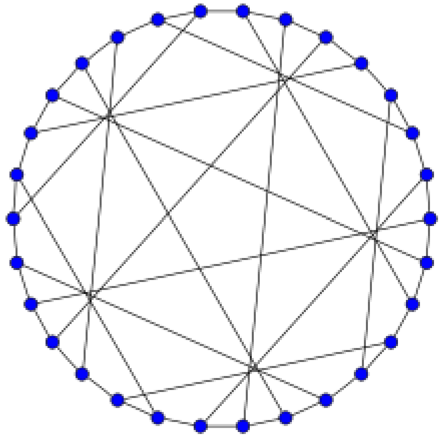
Graph Types – Path Graphs

- A **path graph** or linear graph is a graph whose vertices can be listed in the order $v_1, v_2, \ldots, v_n$ such that the edges are $\{v_i, vi_{+1}\}$ where $i = 1, 2, \ldots, n - 1$.
- Equivalently, a path with at least two vertices is connected and has two terminal vertices (vertices that have degree 1), while all others (if any) have degree 2.
- A regular graph with vertices of degree k is called a k-regular graph or regular graph of degree k.

Basics…

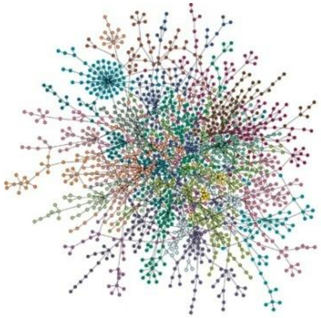
Graph Types – (r,g)-Cages Graphs

- An **(r, g)-graph** is defined to be a graph in which each vertex has exactly r neighbors, and in which the shortest cycle has length exactly g.
- It is known that an (r, g)-graph exists for any combination of $r \geq 2$ and $g \geq 3$. An (r, g)-cage is an (r, g)-graph with the fewest possible number of vertices, among all (r, g)-graphs.



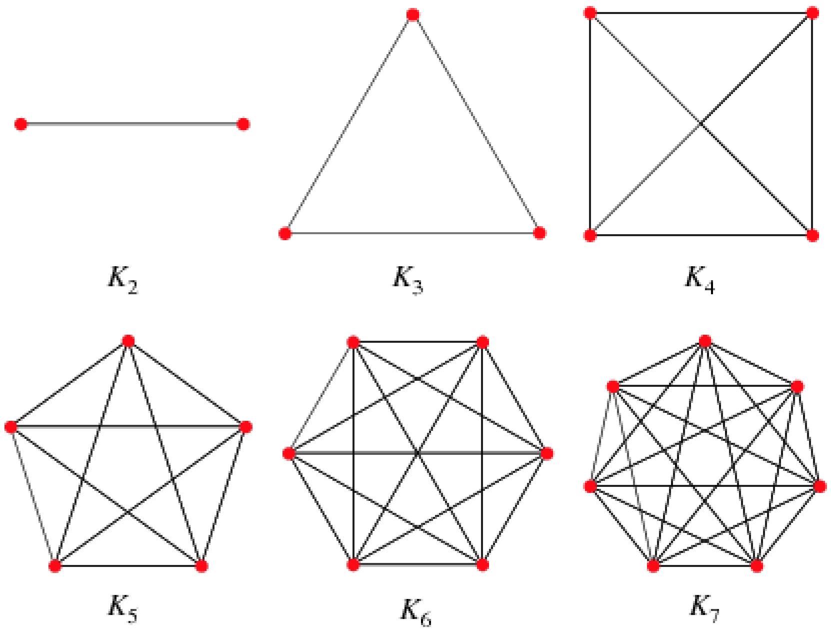
(3,8)-cage

Basics…

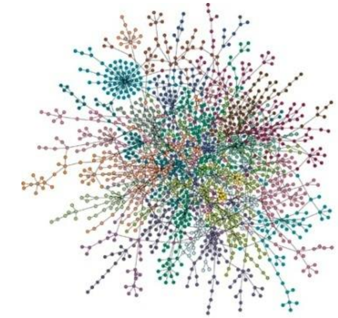


Graph Types – Complete Graphs

- The simple graph that contains exactly one edge between each pair of distinct vertices.
- A Complete Graph is denoted by K_n.

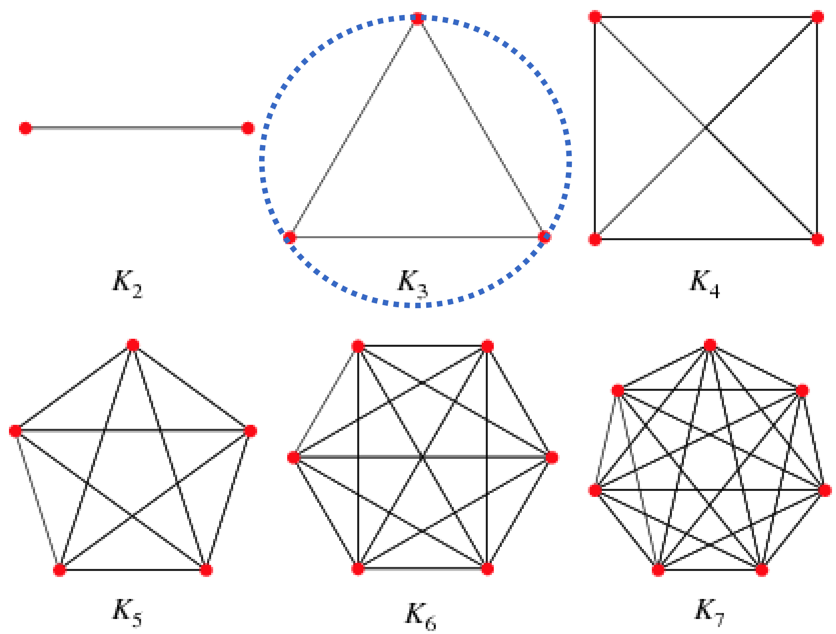


BASICS…

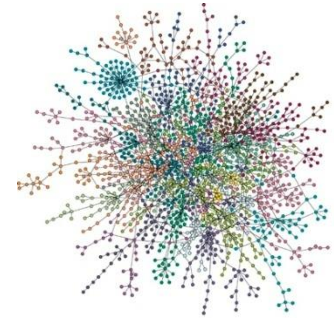


Graph Types – Complete Graphs

- The simple graph that contains exactly one edge between each pair of distinct vertices.
- A Complete Graph is denoted by K_n.

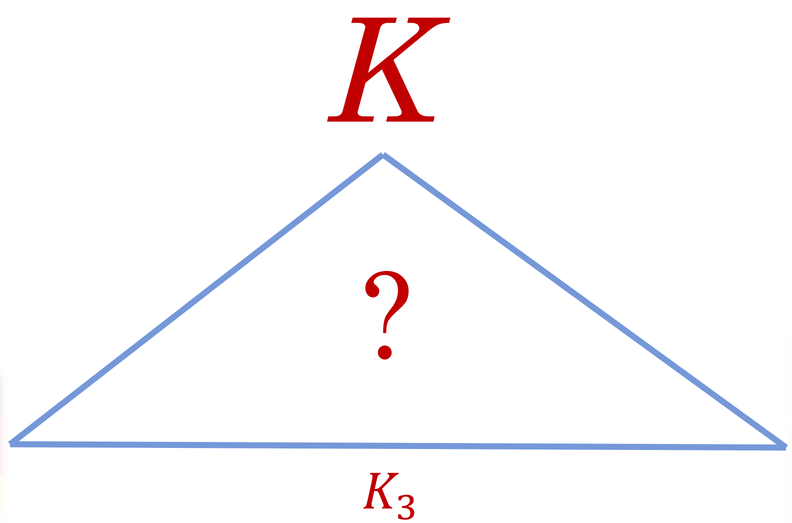


BASICS…



Graph Types – Complete Graphs

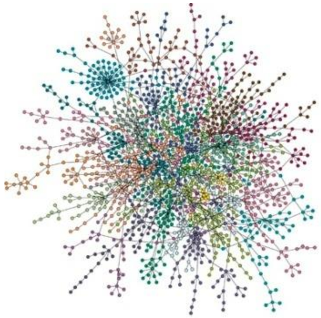
- The simple graph that contains exactly one edge between each pair of distinct vertices.
- A Complete Graph is denoted by K_n.



Vollständiger Graph

Kazimierz Kuratowski

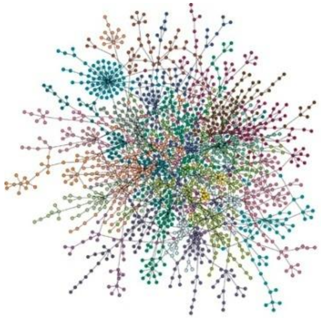
Basics…



Graph Types – Complete Graphs

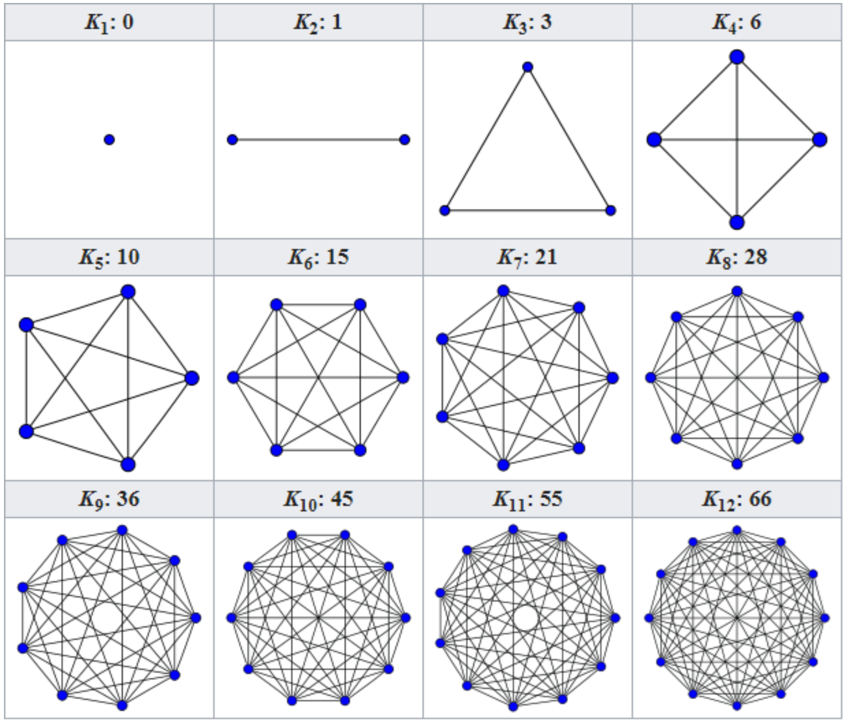
- The simple graph that contains exactly one edge between each pair of distinct vertices.
- A Complete Graph is denoted by $K_{n.}$
- The maximum number of edges is $\frac{n \cdot (n-1)}{2}$.

Basics…

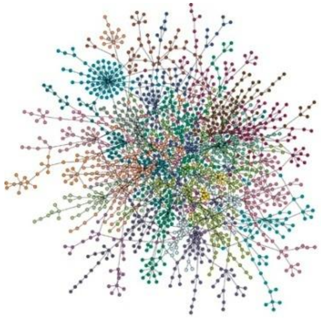


Graph Types – Complete Graphs

- The simple graph that contains exactly one edge between each pair of distinct vertices.
- A Complete Graph is denoted by K_n.
- The maximum number of edges is $\frac{n \cdot (n-1)}{2}$.



Basics…



Graph Types – 1st Euler's Theorem

- **Euler's Theorem:** the sum of the degrees of all vertices is equal to twice the number of edges.

  Proof:

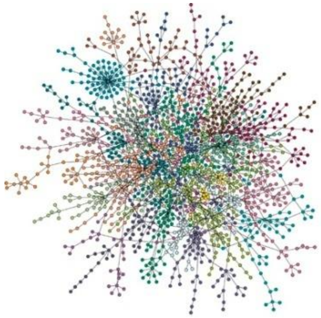
  - It holds that in the sum of the degrees, each edge contributes twice , one for each endpoint (adjacent vertices).

- **Corollary**: The number of vertices of odd degree, is even number.

  Proof:

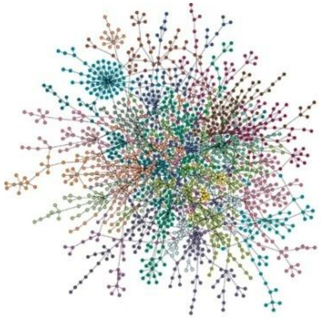
  According to Euler's Theorem, → The sum of the degrees of a graph's vertices is even number → The graph may not include odd number of vertices of odd degree.

BASICS…



- **Graph Types – Complete Graphs**
  - In a party of people some of whom shake hands, an even number of people must have shaken an odd number of other people's hands.
  - Every finite undirected graph has an even number of vertices with odd degree…

HANDSHAKE LEMMA

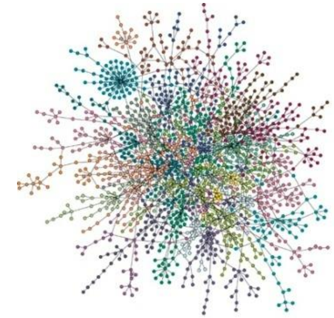


Basics…

- **Graph Types – Complete Graphs**
  - In a party of people some of whom shake hands, an even number of people must have shaken an odd number of other people's hands.
  - Every finite undirected graph has an even number of vertices with odd degree…

$$\sum_{v \in V(G)} \deg(v) = 2|E|$$

HANDSHAKE LEMMA



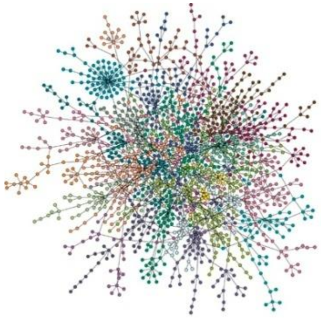
Basics…

- **Graph Types – Complete Graphs**
  - In a party of people some of whom shake hands, an even number of people must have shaken an odd number of other people's hands.
  - Every finite undirected graph has an even number of vertices with odd degree…

$$\sum_{v \in V(G)} \deg(v) = 2|E|$$

HANDSHAKE LEMMA

BASICS…



Graph Types – Complete Graphs

- In a party of people some of whom shake hands, an even number of people must have shaken an odd number of other people's hands.
- Every finite undirected graph has an even number of vertices with odd degree, i.e., $\sum_{v\in V(G)} \deg(v) = 2|E|$, where each vertex contributes one on each incident node

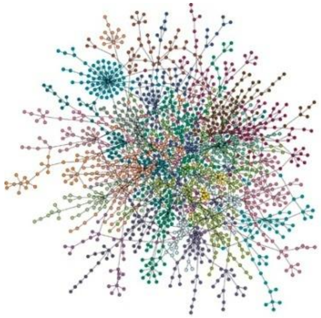
Theorem 1

- For a given graph G=(V,E), the cardinality of the vertices of odd-degree, is even number.

Theorem 2

1) From Th.1 $\rightarrow \sum \mathrm{d}(v)$ is even
2) $\sum \mathrm{d}(v)$ = |even-degree vertices| + |odd-degree vertices|
3) |even-degree vertices| is even → |odd-degree vertices| is even

HANDSHAKE LEMMA

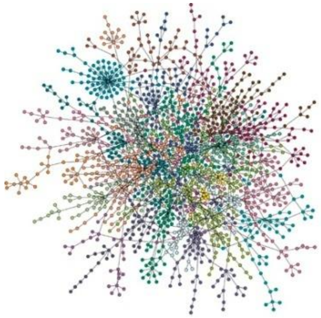


Basics…

- **Graph Types – Elementary Problems**
  - APPLICATION!

    Find the number of edges of the graphs K_n and $K_{n,m}$

BASICS…



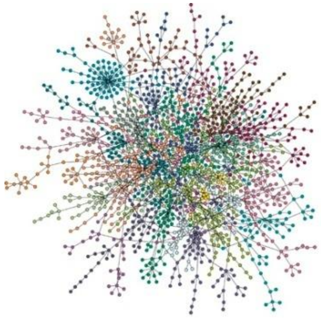
- **Graph Types – Elementary Problems**
  - APPLICATION!

    Find the number of edges of the graphs K_n and $K_{n,m}$

    … A complete graph (denoted with K_n) has $\binom{n}{k}$ edges and is regular graph with vertices of degree $n-1$.

    … $\frac{n!}{2!(n-2)!} = \ldots$

Basics…



Graph Types – Elementary Problems

- <u>APPLICATION</u>!

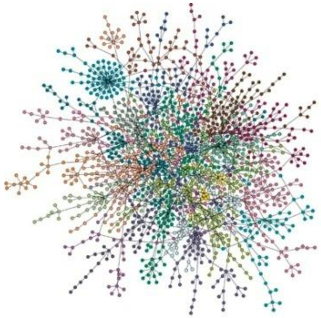
  Find the number of edges of the graphs K_n and $K_{n,m}$

  … A complete graph (denoted with K_n) has $\binom{n}{k}$ edges and is regular graph with vertices of degree $n-1$.

  … $\frac{n!}{2!(n-2)!} = \ldots$

  - According to 1st Euler's Theorem: Since the vertices of K_n graph have degree n-1, the edges are $\frac{n(n-1)}{2}$.
  - In a $K_{n,m}$ graph, there are n vertices of degree m and m vertices of degree n, and hence, the number of vertices is $n \cdot m$

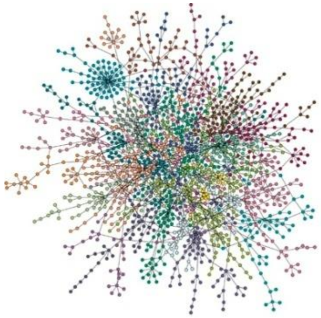
Basics…



- **Graph Types – Elementary Problems**
  - APPLICATION!

    Prove that in each graph there exist at least two vertices with the same degree.

BASICS…

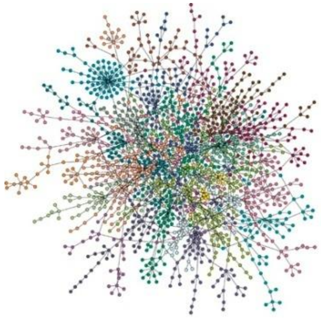


- **Graph Types – Elementary Problems**
  - APPLICATION!

    Prove that in each graph there exist at least two vertices with the same degree.

n vertices → n-1 degrees

Basics…



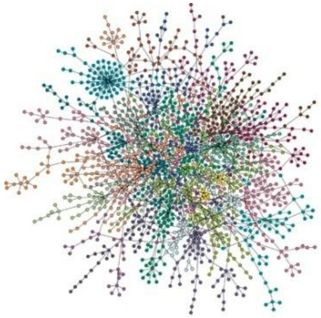
- **Graph Types – Elementary Problems**
  - APPLICATION!

    Prove that in each graph there exist at least two vertices with the same degree.

    Let n be the number of vertices. We distinct two cases:

    1) The graph has at least one isolated vertex. The possible degrees are: $0, \ldots, n-2$ and n vertices must have one of the n-1 possible degrees $(0, \ldots, n-2)$ → at least two vertices have the same degree.

BASICS…



Graph Types – Elementary Problems

- APPLICATION!

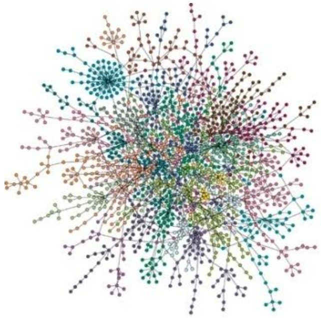
  Prove that in each graph there exist at least two vertices with the same degree.

  Let n be the number of vertices. We distinct two cases:

  1) The graph has at least one isolated vertex. The possible degrees are: $0, \ldots, n-2$ and n vertices must have one of the n-1 possible degrees $(0, \ldots, n-2)$ → at least two vertices have the same degree.

  2) The graph has no isolated vertices. Then the possible degrees are $n-1$ (i.e., $1, \ldots, n-1$) → at least two vertices have the same degree.

Basics…



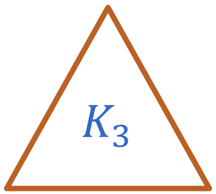
- **Graph Types – Elementary Problems**
  - APPLICATION!

    Prove that in a group of at least 6 persons there exist at least 3 persons that by two they know each other, or there exist at least three persons that by two the do not know each other.

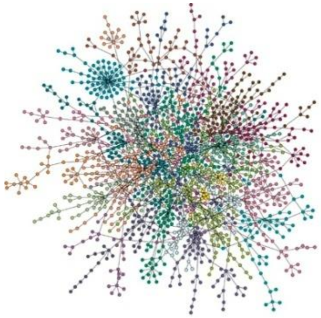
BASICS…

- **Graph Types – Elementary Problems**
  - APPLICATION!

    Prove that in a group of at least 6 persons there exist at least 3 persons that by two they know each other, or there exist at least three persons that by two the do not know each other.



Basics…

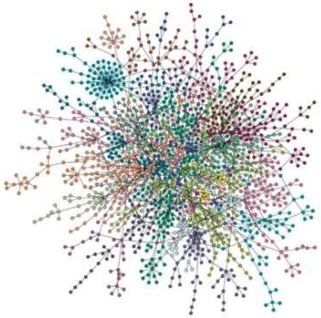


Graph Types – Elementary Problems

- APPLICATION!

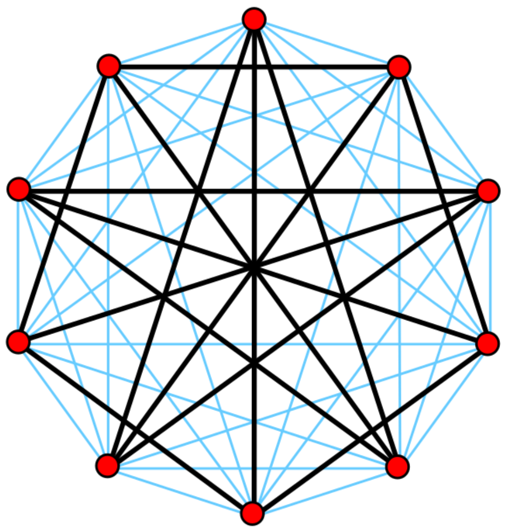
  Prove that in a group of at least 6 persons there exist at least 3 persons that by two they know each other, or there exist at least three persons that by two the do not know each other.

- $Let\ G = (V, E)$, $|V(G)| = 6$, and let $v \in V(G)$
  - Partition $\{V(G) - v\}$ to V_1, V_2: $V_1 \cap V_2 = \emptyset$, with elements adjacent to v
  - V_1 or V_2 contains at least 3 elements (let $|V_1| \geq 3$)
    [if $|V_2| \geq 3$ work in the complement of the graph]
  - If ≥ 2 vertices, let $u, w \in V_1$ are adjacent $\rightarrow u, v, w\ is\ K_3$

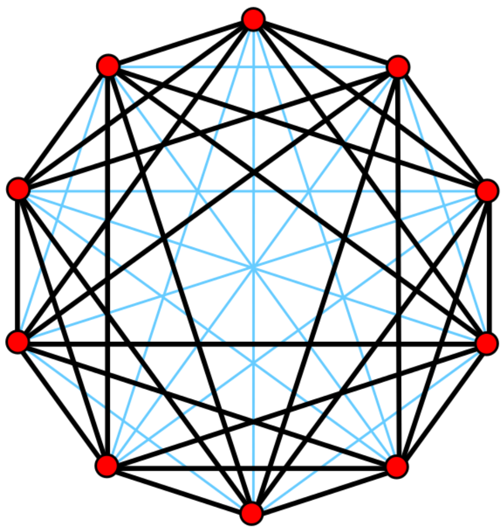


Graph Types – Complement

- The **Complement** or **inverse** of a graph G is a graph H on the same vertices such that two distinct vertices of H are adjacent if and only if they are not adjacent in G.
- In order to generate the complement of a graph, one fills in all the missing edges required to form a complete graph, and removes all the edges that were previously there.

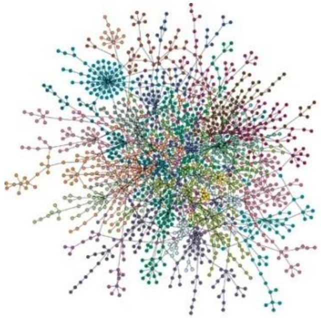


G



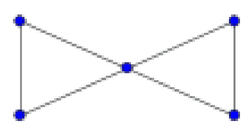
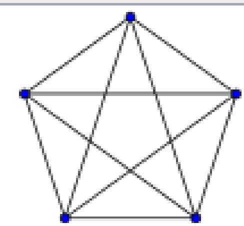
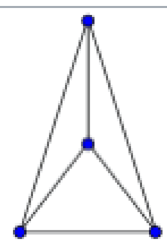
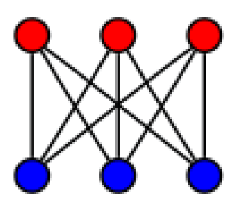
H

Basics…

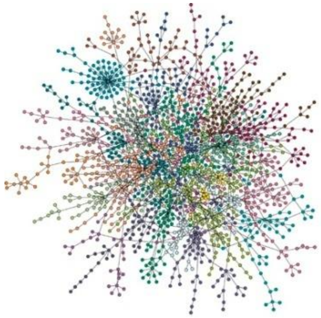


Graph Types – Planar

- A **planar graph** is a graph that can be embedded in the plane, i.e., it can be drawn on the plane in such a way that its edges intersect only at their endpoints.
- A planar graph can be drawn in such a way that no edges cross each:

Example graphs	
Planar	**Nonplanar**
 Butterfly graph	 Complete graph K_5
 Complete graph K_4	 Utility graph $K_{3,3}$

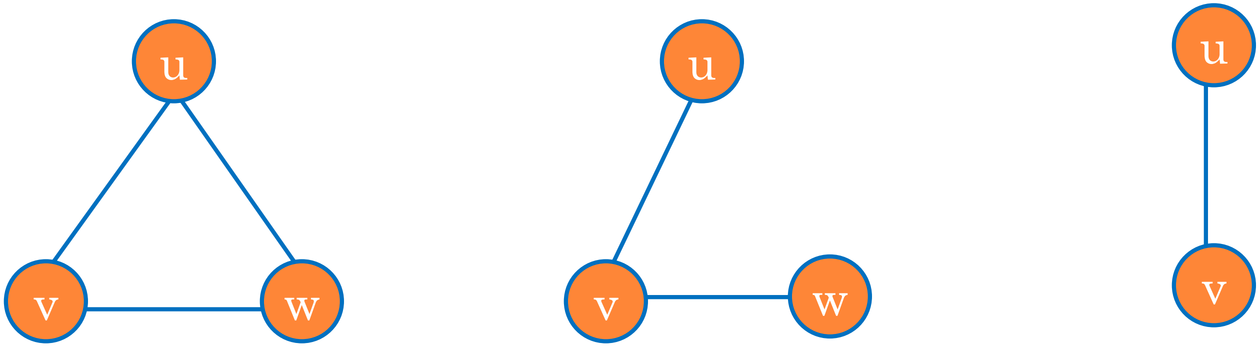
BASICS…



Graph Types – Subgraphs (subsets)

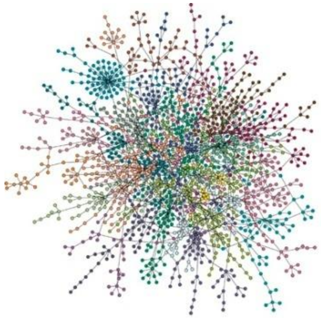
- A subgraph of a graph $G = (V, E)$ is a graph $H = (V', E')$ where V' is a subset of V and E' is a subset of E
- Application example: solving sub-problems within a graph.
- Representation example:

$$V = \{u, v, w\},$$
$$E = \{\{u, w\}, \{w, v\}, \{v, u\}\},$$
$$H_1, H_2$$



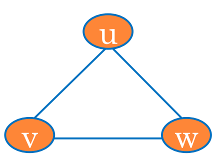
G $\quad$ H_1 $\quad$ H_2

BASICS…

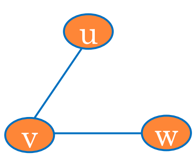


Graph Types – Subgraphs (induced subgraph)

- Let $G = (V, E)$ be any graph, and let $S \subset V$ be any subset of vertices of G.
- Then the induced subgraph $G[S]$ is the graph whose vertex set is S and whose edge set consists of all of the edges in E that have both endpoints in S.
- A subgraph H of G is called **INDUCED**, if for any two vertices u, v in H, u and v are adjacent in H if and only if they are adjacent in G(H has the same edges as G between the vertices in H).
- A general subgraph can have less edges between the same vertices than the original one.
- So, an induced subgraph can be constructed by deleting vertices alongside with their incident edges, but no more edges (If additional edges are deleted, the subgraph is not induced.)

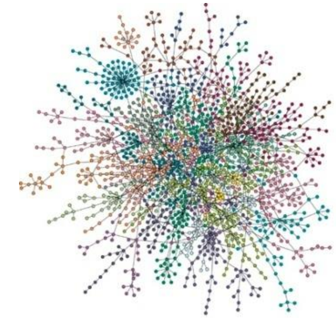


G



Subgraph of G

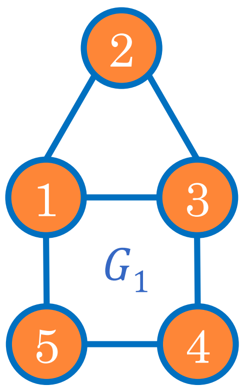
Induced Subgraph of G

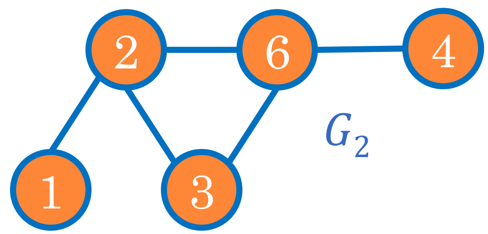


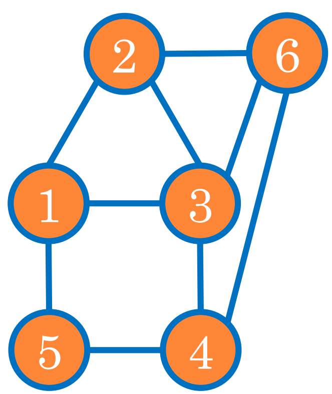
Basics…

Graph Types – Combined Graphs (union)

- Construct a graph G by the union of the graphs G_1, G_2
- $G = G_1 \cup G_2$ *wherein* $E = E_1 \cup E_2$ *and* $V = V_1 \cup V_2$





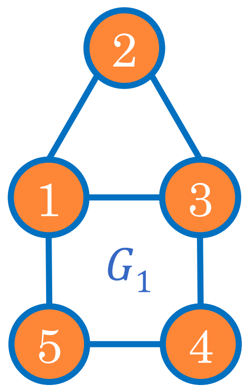


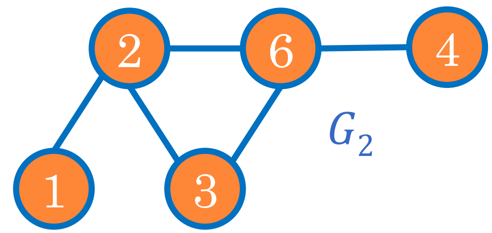
$G = G_1 \cup G_2$

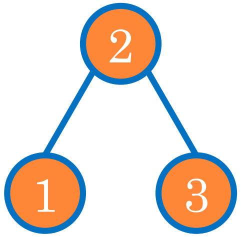
Basics…

Graph Types – Combined Graphs (intersection)

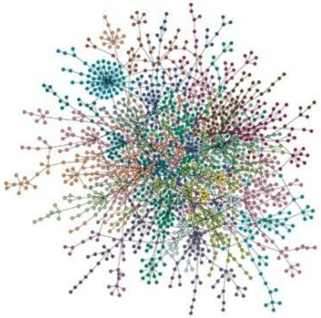
- Construct a graph G by the intersection of the graphs G_1, G_2
- $G = G_1 \cap G_2$ *wherein* $E = E_1 \cap E_2$ *and* $V = V_1 \cap V_2$



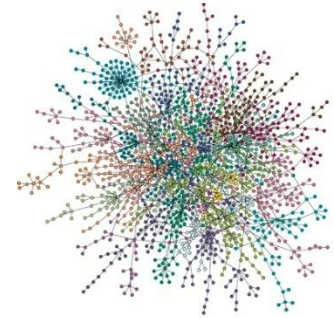




$G = G_1 \cap G_2$

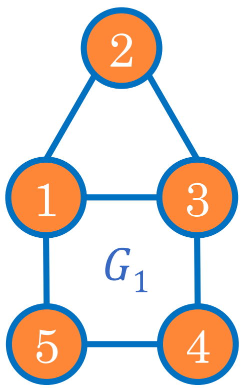


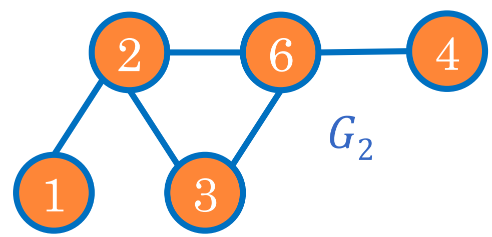
Basics…

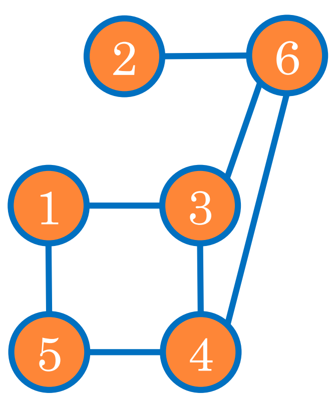


Graph Types – Combined Graphs (ring-sum)

- Construct a graph G by the ring-sum of the graphs G_1, G_2
- $G = G_1 \oplus G_2$ *wherein* $E = E_1 \oplus E_2$ *and* $V = V_1 \oplus V_2$

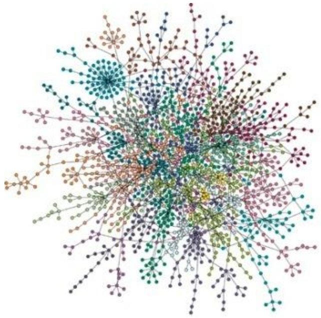




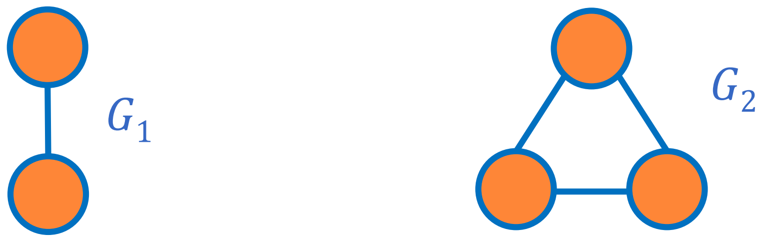


$G = G_1 \oplus G_2$

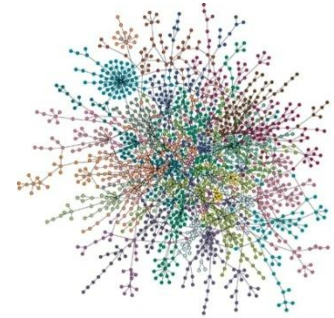
Basics…



- **Graph Types – Combined Graphs (join / sum)**
  - Construct a graph G by the join of the graphs G_1, G_2
  - $G = (V, E) = \mathrm{G}_1 + \mathrm{G}_2 \ wherein \ E = E_1 \cup \mathrm{E}_2 \cup \mathrm{E}_3 \ and \ V = V_1 \cup V_2$



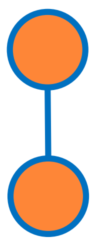
Basics…

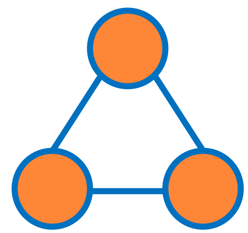


Graph Types – Combined Graphs (join / sum)

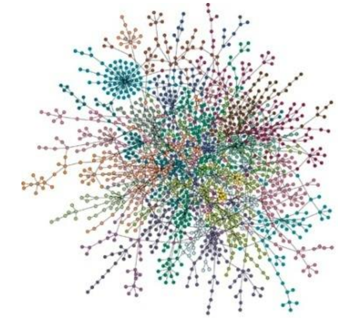
- Construct a graph G by the join of the graphs G_1, G_2
- $G = (V, E) = \mathrm{G}_1 + \mathrm{G}_2$ *wherein* $E = E_1 \cup \mathrm{E}_2 \cup \mathrm{E}_3$ *and* $V = V_1 \cup V_2$

E_3 contains the vertices with one side in V_1 and the other side in V_2





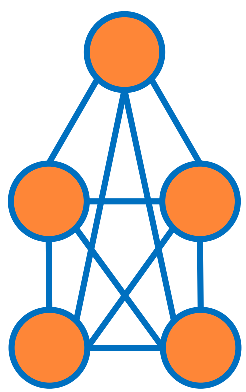
BASICS…



Graph Types – Combined Graphs (join / sum)

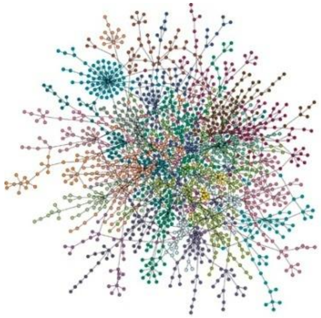
- Construct a graph G by the join of the graphs G_1, G_2
- $G = (V, E) = \mathrm{G}_1 + \mathrm{G}_2$ *wherein* $E = E_1 \cup \mathrm{E}_2 \cup \mathrm{E}_3$ *and* $V = V_1 \cup V_2$

E_3 contains the vertices with one side in V_1 and the other side in V_2



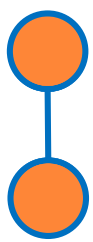
$G = G_1 + \mathrm{G}_2$

BASICS…



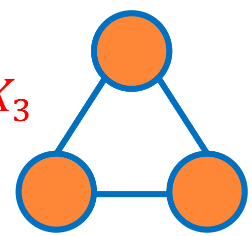
- **Graph Types – Combined Graphs (join / sum)**
  - Construct a graph G by the join of the graphs G_1, G_2
  - $G = (V,E) = G_1 + G_2$ *wherein* $E = E_1 \cup E_2 \cup E_3$ *and* $V = V_1 \cup V_2$

K_2



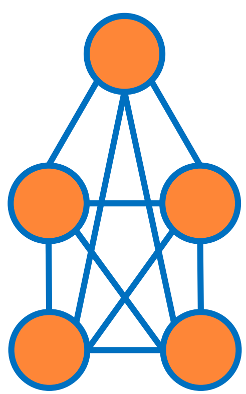
G_1

K_3



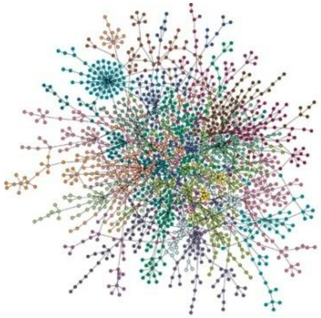
G_2

K_5



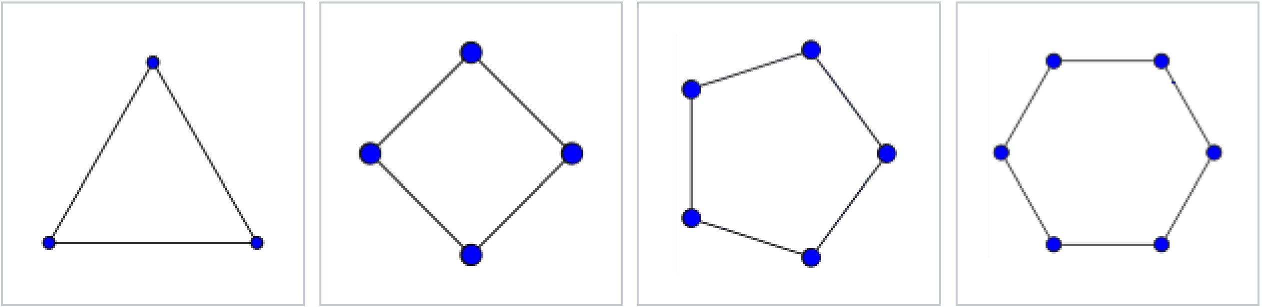
$G = G_1 + G_2$

Basics…



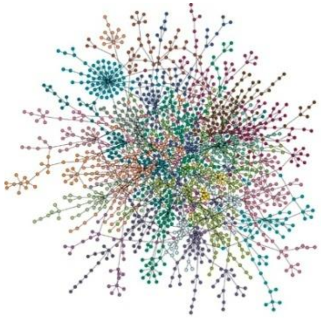
Graph Types – Cycle Graphs

- A cycle graph (or, circular graph) is a graph that consists of a single cycle, or in other words, some number of vertices connected in a closed chain.
- The number of vertices in Cn equals the number of edges, and every vertex has degree 2; that is, every vertex has exactly two incident edges.
- The cycle graph with n vertices is denoted by $C_n, n \geq 3$



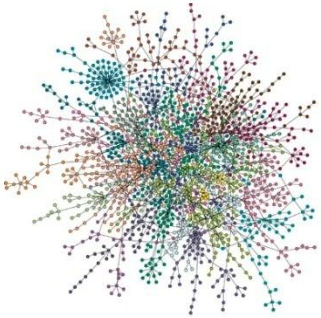
C_3 C_4 C_5 C_6

Basics…



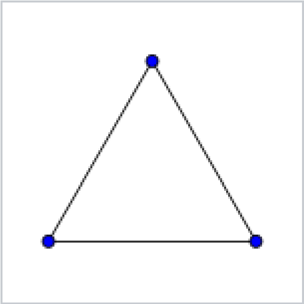
- **Graph Types – Wheel Graphs**
  - The simple graph, obtained by adding an additional vertex to C_n and connecting all vertices to this new vertex by new edges.
  - A Wheel Graph is denoted by $W_n, n \geq 3$

Basics…

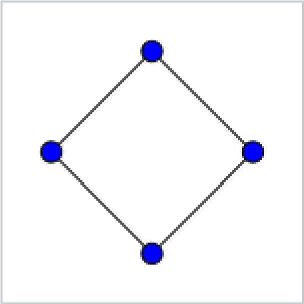


Graph Types – Wheel Graphs

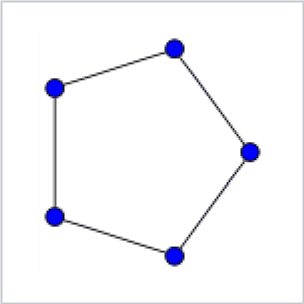
- The simple graph, obtained by adding an additional vertex to C_n and connecting all vertices to this new vertex by new edges.
- A Wheel Graph is denoted by $W_n, n \geq 3$
- W_n is a graph on n vertices constructed by connecting a single vertex to every vertex in an $(n - 1)$-cycle



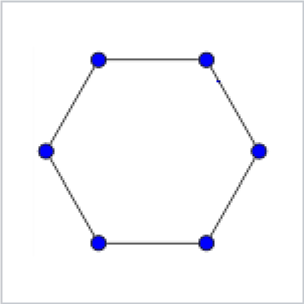
C_3



C_4



C_5

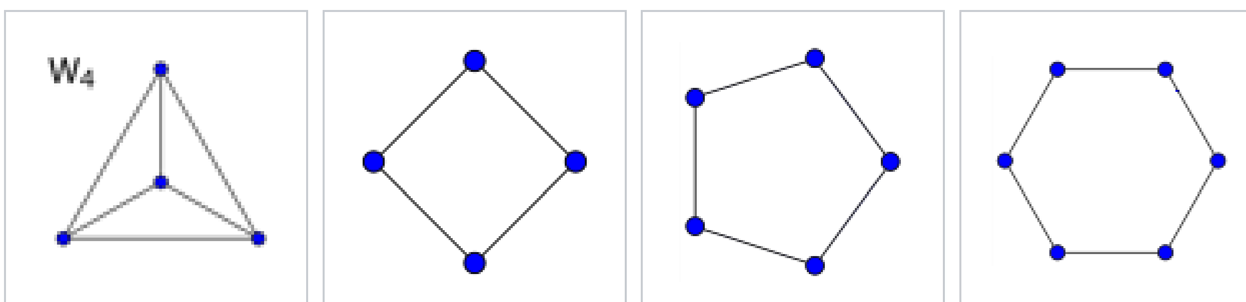


C_6

BASICS…

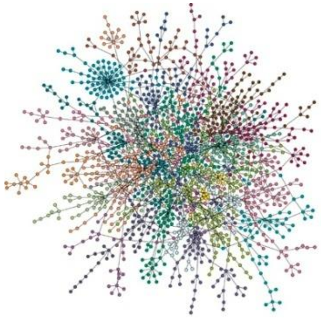
Graph Types – Wheel Graphs

- The simple graph, obtained by adding an additional vertex to C_n and connecting all vertices to this new vertex by new edges.
- A Wheel Graph is denoted by $W_n, n \geq 3$
- W_n is a graph on n vertices constructed by connecting a single vertex to every vertex in an $(n - 1)$-cycle



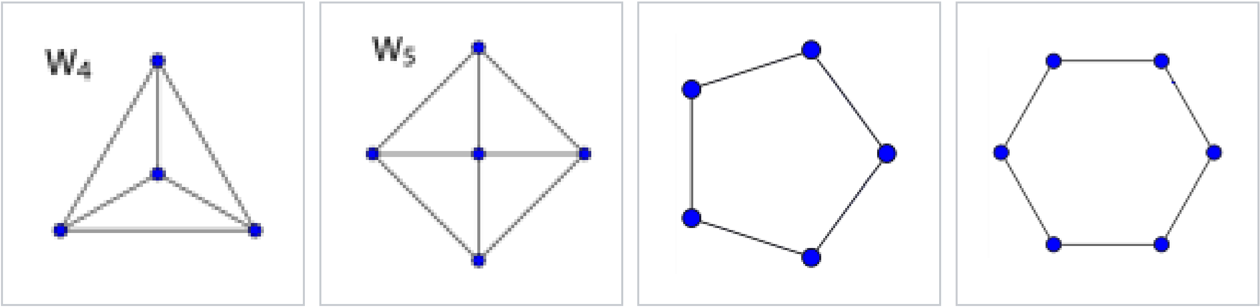
C_4 C_5 C_6

BASICS…



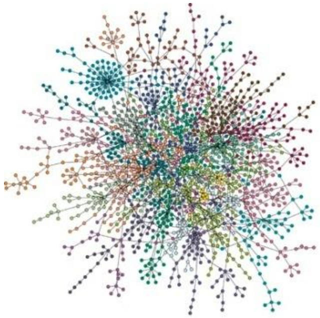
Graph Types – Wheel Graphs

- The simple graph, obtained by adding an additional vertex to C_n and connecting all vertices to this new vertex by new edges.
- A Wheel Graph is denoted by $W_n, n \geq 3$
- W_n is a graph on n vertices constructed by connecting a single vertex to every vertex in an $(n - 1)$-cycle



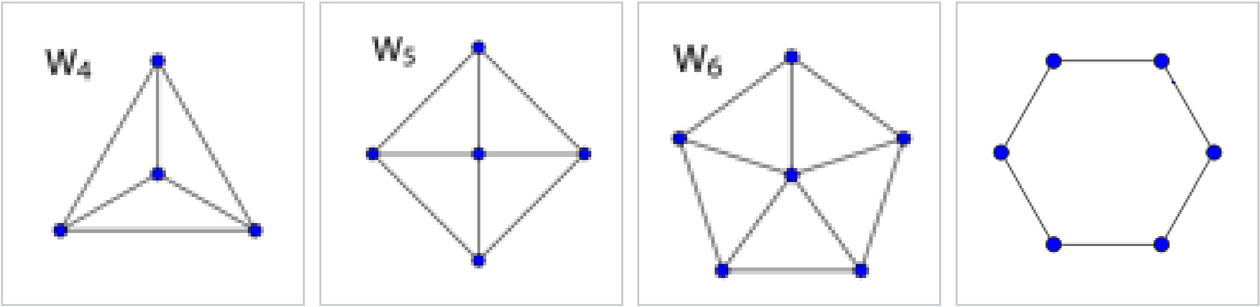
C_5 C_6

Basics…



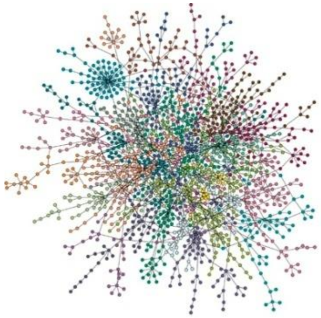
Graph Types – Wheel Graphs

- The simple graph, obtained by adding an additional vertex to C_n and connecting all vertices to this new vertex by new edges.
- A Wheel Graph is denoted by $W_n, n \geq 3$
- W_n is a graph on n vertices constructed by connecting a single vertex to every vertex in an $(n-1)$-cycle



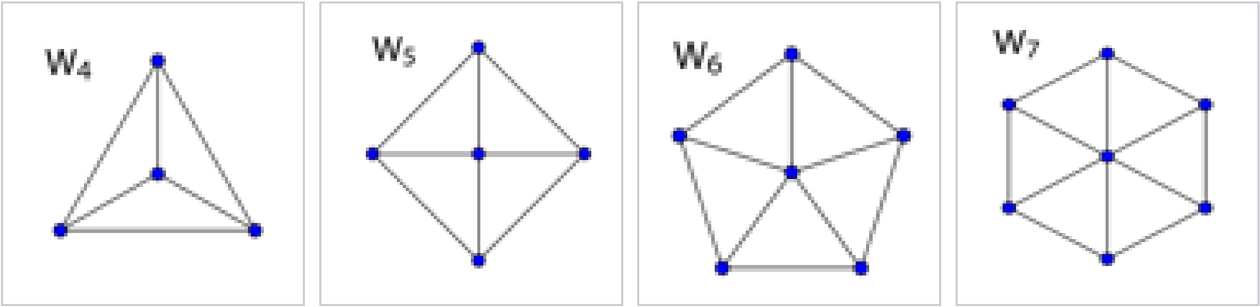
C_6

Basics…

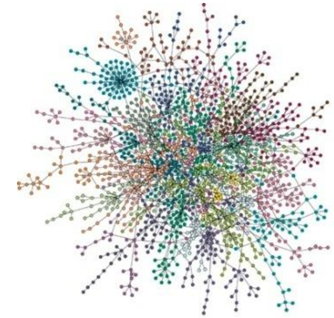


Graph Types – Wheel Graphs

- The simple graph, obtained by adding an additional vertex to C_n and connecting all vertices to this new vertex by new edges.
- A Wheel Graph is denoted by $W_n, n \geq 3$
- W_n is a graph on n vertices constructed by connecting a single vertex to every vertex in an $(n - 1)$-cycle

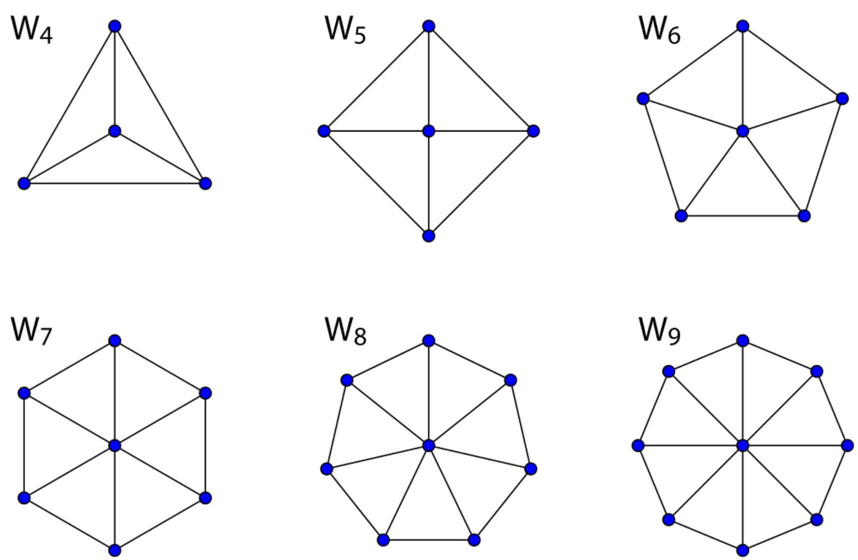


BASICS…



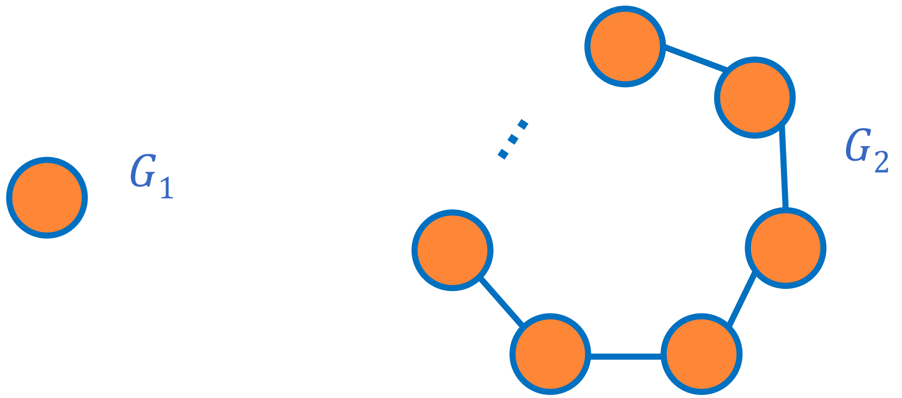
Graph Types – Wheel Graphs

- The simple graph, obtained by adding an additional vertex to C_n and connecting all vertices to this new vertex by new edges.
- A Wheel Graph is denoted by $W_n, n \geq 3$
- W_n is a graph on n vertices constructed by connecting a single vertex to every vertex in an $(n - 1)$-cycle

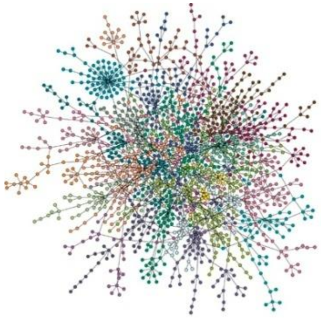


Basics…

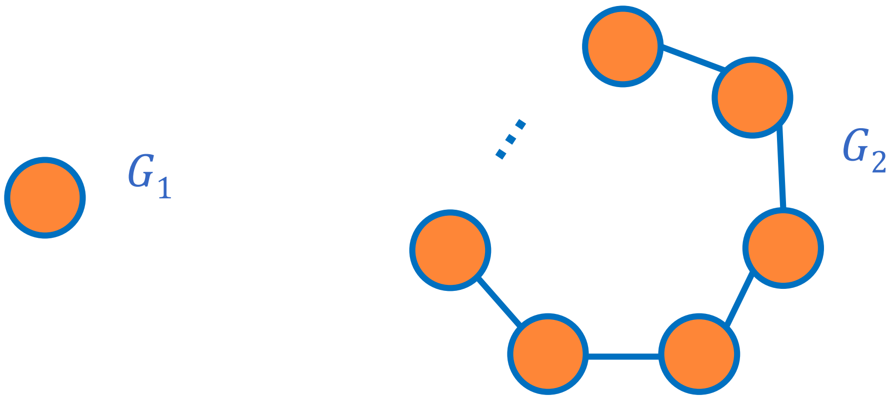
- **Graph Types – Combined Graphs (join / sum)**
  - Construct a graph G by the join of the graphs G_1, G_2
  - $G = (V, E) = \mathrm{G}_1 + \mathrm{G}_2 \ wherein \ E = E_1 \cup \mathrm{E}_2 \cup \mathrm{E}_3 \ and \ V = V_1 \cup V_2$

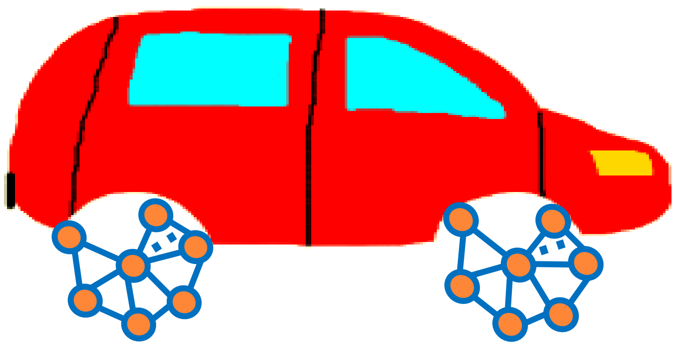


Basics…

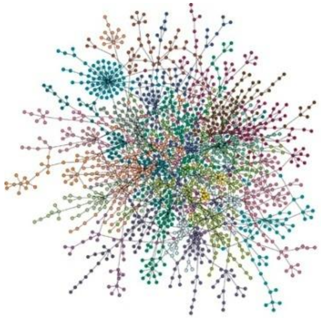


- **Graph Types – Combined Graphs (join / sum)**
  - Construct a graph G by the join of the graphs G_1, G_2
  - $G = (V, E) = \mathrm{G}_1 + \mathrm{G}_2 \; wherein \; E = E_1 \cup \mathrm{E}_2 \cup \mathrm{E}_3 \; and \; V = V_1 \cup V_2$





Basics…



Graph Types – Combined Graphs (join / sum)

- Construct a graph G by the join of the graphs G_1, G_2
- $G = (V, E) = \mathrm{G}_1 + \mathrm{G}_2 \; wherein \; E = E_1 \cup \mathrm{E}_2 \cup \mathrm{E}_3 \; and \; V = V_1 \cup V_2$