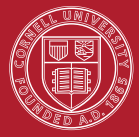


Cornell University

Lecture 2 - Graph topologies and characteristics

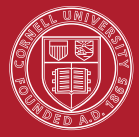
Graph-Based Data Science For Networked Systems
ECE 5260 – ORIE 5735

Anna Scaglione

Cornell Tech

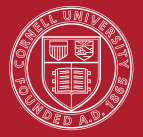
January 26, 2026

Content



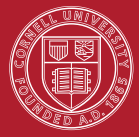
Cornell University

Outline



Cornell University

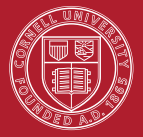
Review and overview



Cornell University

- We introduced basic definitions that pertain graphs as well as graph structures.
- We rapidly surveyed a variety of models and examples that leverage graph based data science.
  - **Relational** interactions: social, economic, knowledge networks, biological.
  - **Data modeling networks** for deterministic multi-dimensional data or random observations.
  - **Networked** infrastructures: the internet electric power, transportation, gas, water, etc.
- Today we focus again on the study of graph structures and specific types of graphs

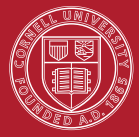
Review and overview



Cornell University

- We introduced basic definitions that pertain graphs as well as graph structures.
- We rapidly surveyed a variety of models and examples that leverage graph based data science.
  - **Relational** interactions: social, economic, knowledge networks, biological.
  - **Data modeling networks** for deterministic multi-dimensional data or random observations.
  - **Networked** infrastructures: the internet electric power, transportation, gas, water, etc.
- Today we focus again on the study of graph structures and specific types of graphs

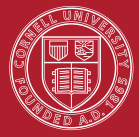
Review and overview



Cornell University

- We introduced basic definitions that pertain graphs as well as graph structures.
- We rapidly surveyed a variety of models and examples that leverage graph based data science.
  - **Relational** interactions: social, economic, knowledge networks, biological.
  - **Data modeling networks** for deterministic multi-dimensional data or random observations.
  - **Networked** infrastructures: the internet electric power, transportation, gas, water, etc.
- Today we focus again on the study of graph structures and specific types of graphs

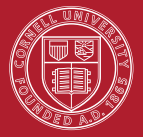
Review and overview



Cornell University

- We introduced basic definitions that pertain graphs as well as graph structures.
- We rapidly surveyed a variety of models and examples that leverage graph based data science.
  - **Relational** interactions: social, economic, knowledge networks, biological.
  - **Data modeling networks** for deterministic multi-dimensional data or random observations.
  - **Networked** infrastructures: the internet electric power, transportation, gas, water, etc.
- Today we focus again on the study of graph structures and specific types of graphs

Outline



Cornell University

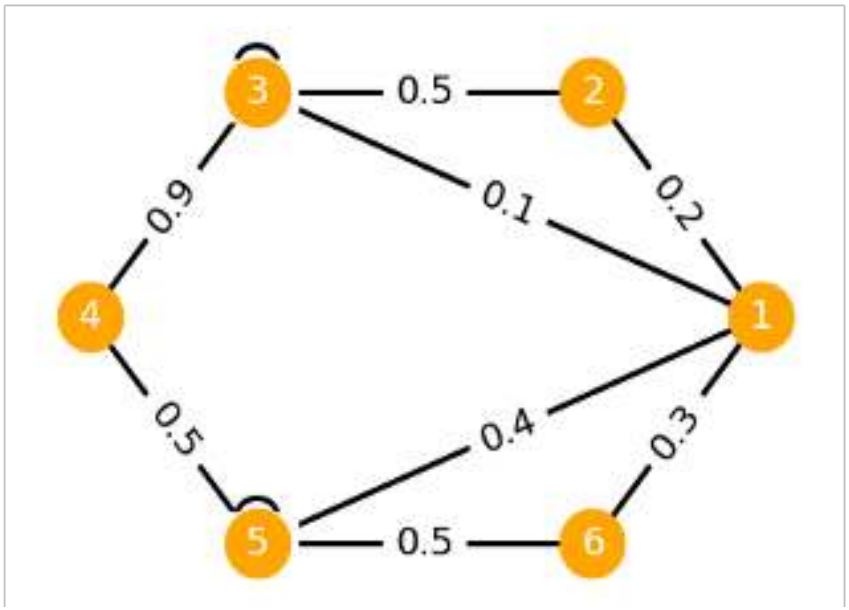
Definition of a graph

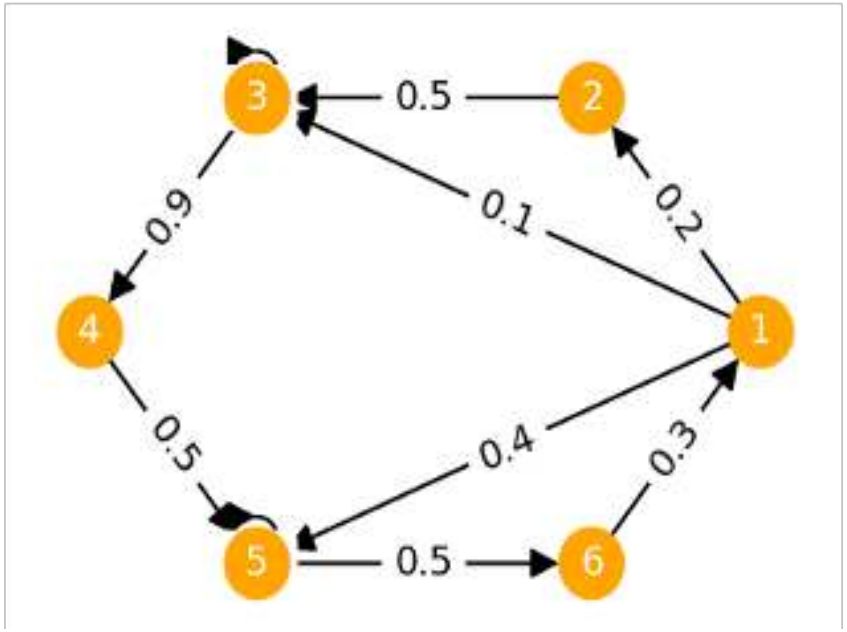
$$\mathcal{V} = \{1, 2, 3, 4, 5, 6\}$$

$$\mathcal{E} = \{(1,2), (1,5), (1,3), (1,6), (2,3), (3,3), (3,4), (4,5), (5,5), (5,6)\}$$

$$\boldsymbol{w} = [w_{1,2}, w_{1,5}, w_{1,3}, w_{1,6}, w_{2,3}, w_{3,3}, w_{3,4}, w_{4,5}, w_{5,5}, w_{5,6}]$$

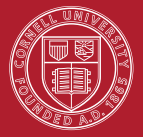
$$= [0.2, 0.4, 0.1, 0.3, 0.5, 0.1, 0.9, 0.5, 0.5, 0.5]$$





- We need to define the nodes, edge sets, possibly edge weights. For multigraphs we would have multiple sets of edges.

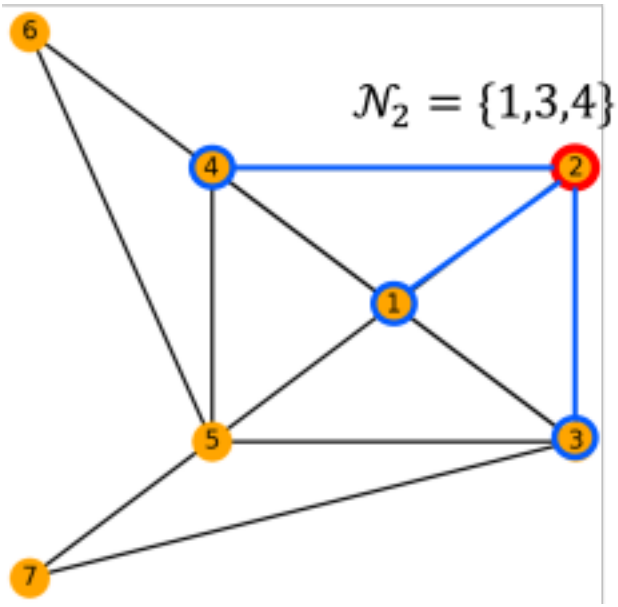
Outline

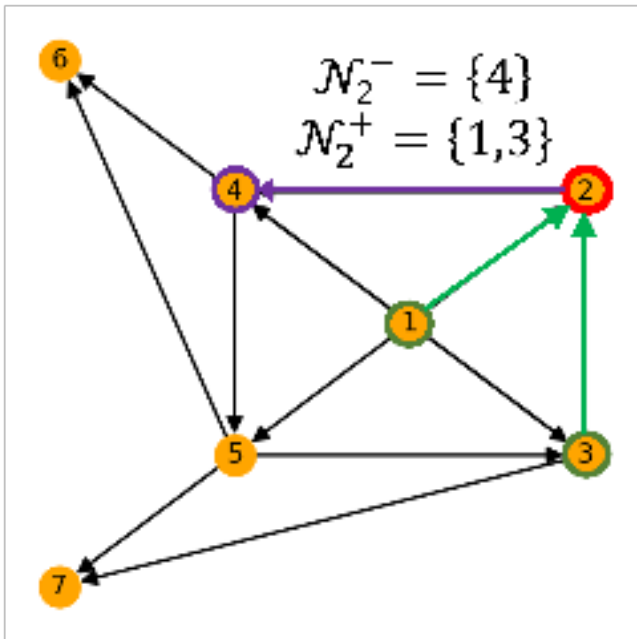


Cornell University

Neighborhoods

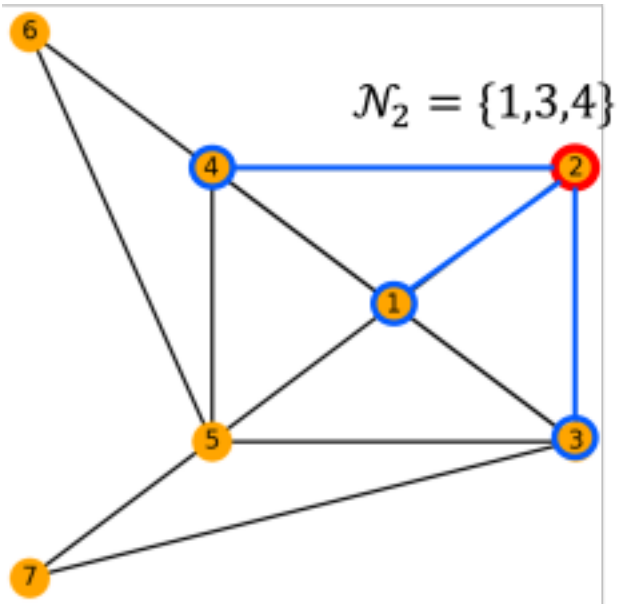
- $\mathcal{N}_v \subset \mathcal{V}$ contains all nodes connected to node v and the node degree $d_v = |\mathcal{N}_v|$
- $\mathcal{N}_v^+ \subset \mathcal{V}$ contains all nodes in edges that enter node, $\mathcal{N}_v^- \subset \mathcal{V}$ contains all nodes in edges that exit node v. In-degree and out-degree are $d_v^+ = |\mathcal{N}_v^+|$ and $d_v^- = |\mathcal{N}_v^+|$ respectively.

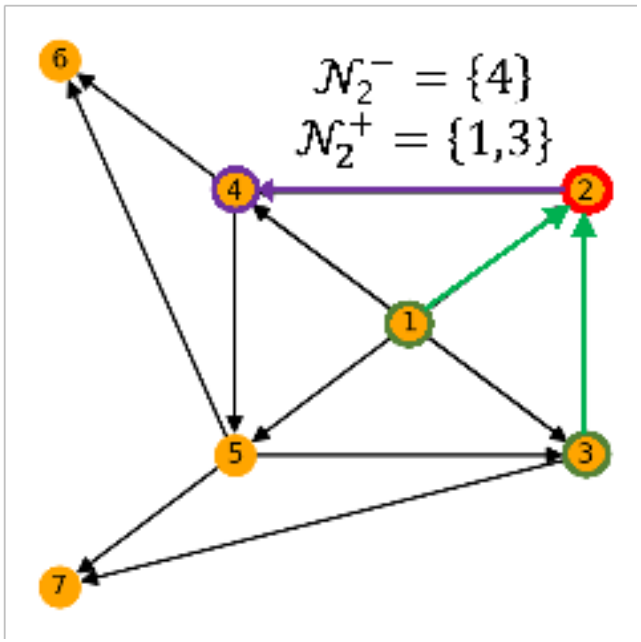




Neighborhoods

- $\mathcal{N}_v \subset \mathcal{V}$ contains all nodes connected to node v and the node degree $d_v = |\mathcal{N}_v|$
- $\mathcal{N}_v^+ \subset \mathcal{V}$ contains all nodes in edges that enter node, $\mathcal{N}_v^- \subset \mathcal{V}$ contains all nodes in edges that exit node v. In-degree and out-degree are $d_v^+ = |\mathcal{N}_v^+|$ and $d_v^- = |\mathcal{N}_v^+|$ respectively.





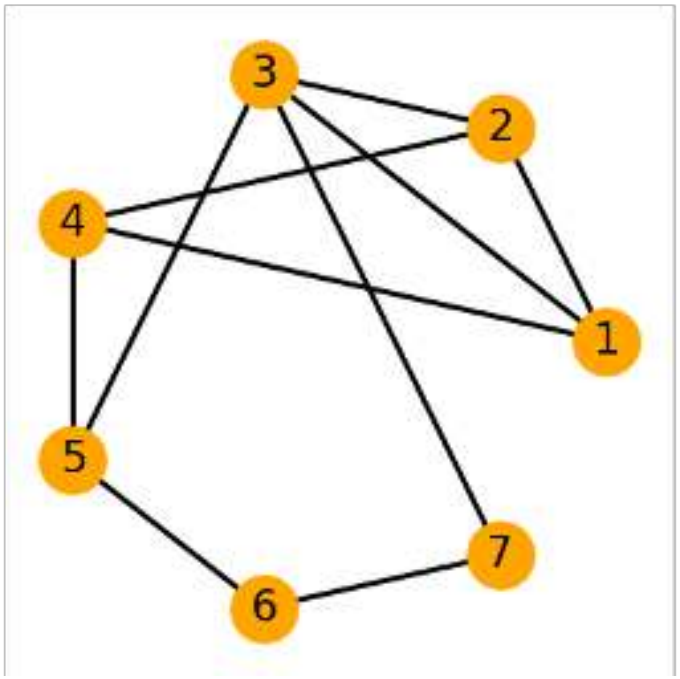
Degree sequence
Graphs topological characteristics

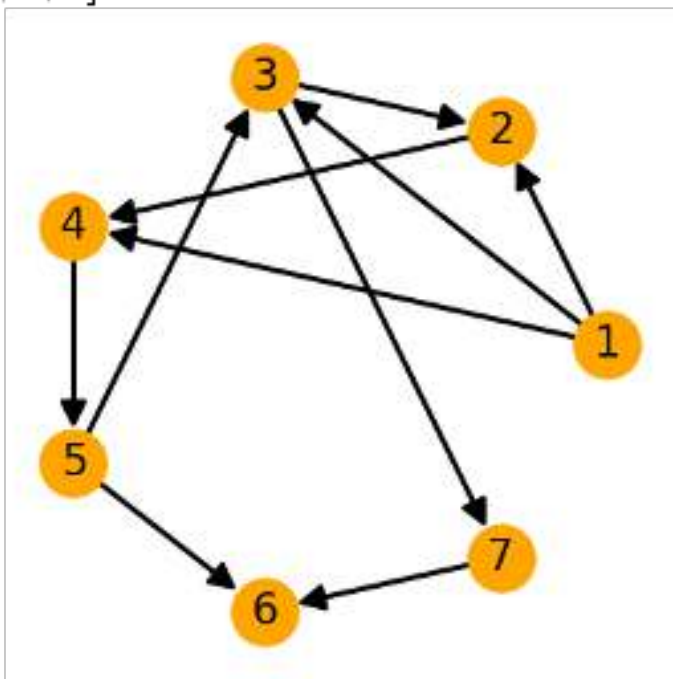
The degree sequence is the vector $\boldsymbol{d} = [d_1, \ldots, d_{|\mathcal{V}|}]$ of nodal degrees. For directed graphs we have the sequence of in degrees and out degrees, $\boldsymbol{d}^+$ nad $\boldsymbol{d}^-$.

$$\boldsymbol{d} = [d_1, \ldots, d_7] = [3, 3, 4, 3, 3, 2, 2]$$

$$\boldsymbol{d}^+ = [d_1^+, \ldots, d_7^+] = [0, 2, 2, 2, 1, 2, 1]$$

$$\boldsymbol{d}^- = [d_1^-, \ldots, d_7^-] = [3, 1, 2, 1, 2, 0, 1]$$





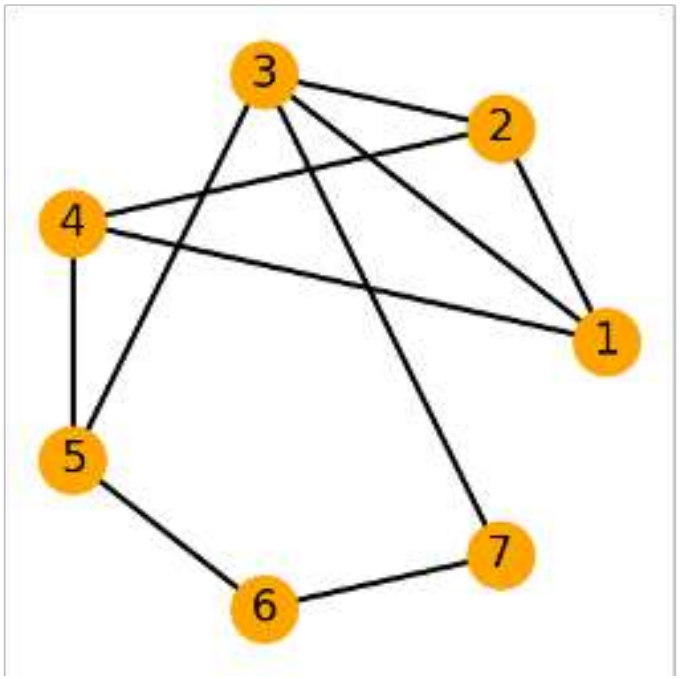
The average degree

Graphs topological characteristics

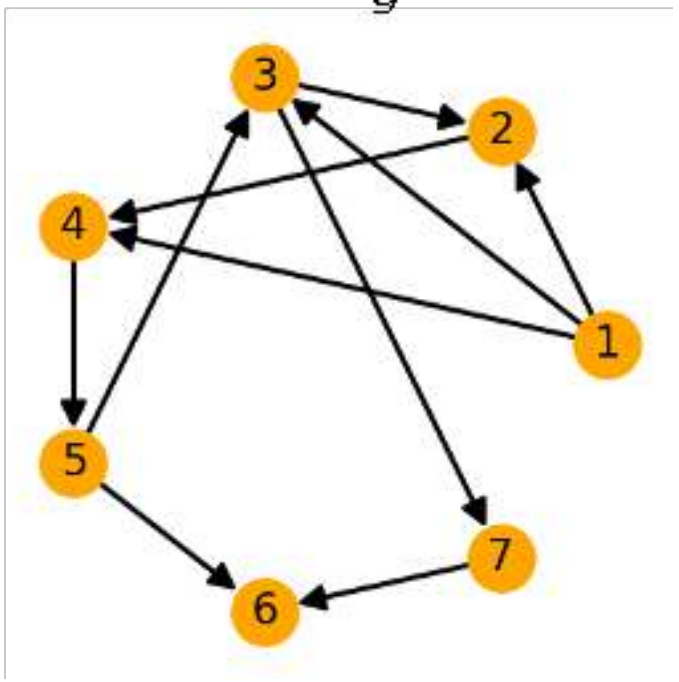
The average degree is:

$$\overline{d}_{\mathcal{G}} = \frac{1}{|\mathcal{V}|}\sum_{v=1}^{|\mathcal{V}|} d_v, \quad \overline{d}^{+}_{\mathcal{G}} = \frac{1}{|\mathcal{V}|}\sum_{v=1}^{|\mathcal{V}|} d^{+}_v, \quad \overline{d}^{-}_{\mathcal{G}} = \frac{1}{|\mathcal{V}|}\sum_{v=1}^{|\mathcal{V}|} d^{-}_v$$

$d_{\mathcal{G}} = 2.8571$



$\overline{d}^{+}_{\mathcal{G}} = 1.4285 \quad \overline{d}^{-}_{\mathcal{G}} = 1.4285$

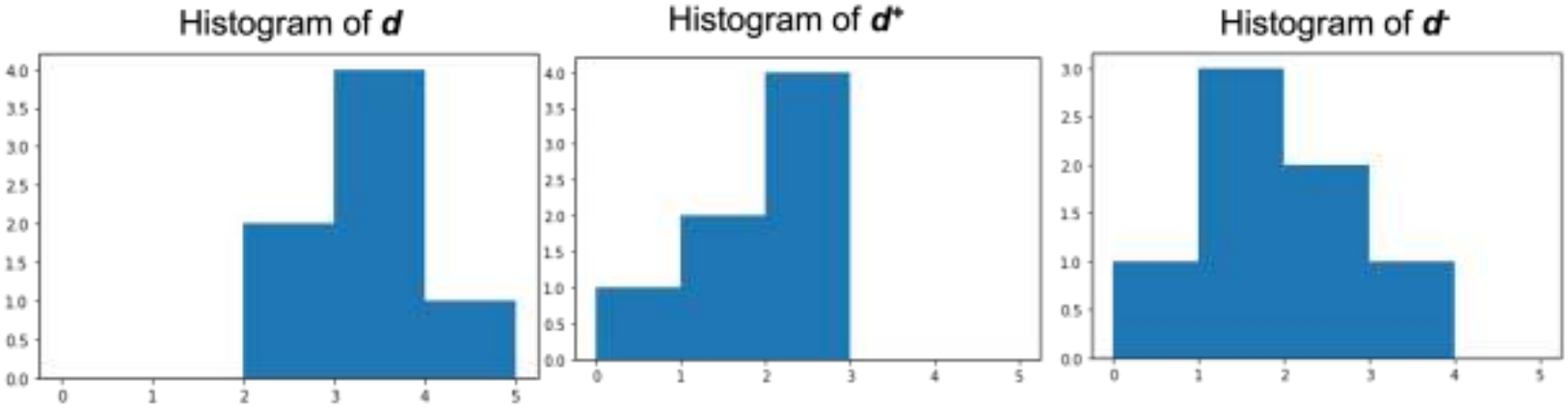


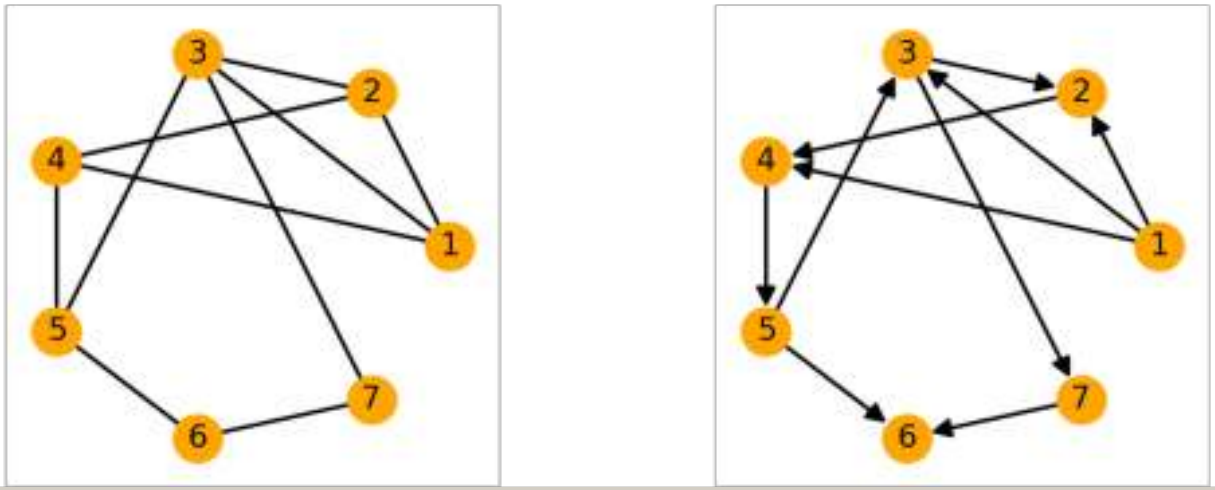
The empirical degree distribution

Graphs topological characteristics

The empirical degree distribution is the normalized histogram n_k:

$$P_{\text{degree}}(k) := \frac{n_k}{|\mathcal{V}|}, \quad n_k := \text{number of nodes with degree} = k$$





Network density

Graphs topological characteristics

Cornell University

- Let $|\mathcal{V}| = N$. If we don't consider self-loops the number of edges of an undirected graph is:

$$|\mathcal{E}| \leq \binom{N}{2} = \frac{N(N-1)}{2}$$

and for a directed graph is:

$$|\mathcal{E}| \leq N(N-1)$$

- Equality holds for a complete graph.

Network density

An undirected graph density is $\delta = \frac{2|\mathcal{E}|}{N(N-1)}$ and that of a directed graph is $\delta' = \frac{|\mathcal{E}|}{N(N-1)}$.

Network density

Graphs topological characteristics

- Let $|\mathcal{V}| = N$. If we don't consider self-loops the number of edges of an undirected graph is:

$$|\mathcal{E}| \leq \binom{N}{2} = \frac{N(N-1)}{2}$$

and for a directed graph is:

$$|\mathcal{E}| \leq N(N-1)$$

- Equality holds for a complete graph.

Network density

An undirected graph density is $\delta = \frac{2|\mathcal{E}|}{N(N-1)}$ and that of a directed graph is $\delta' = \frac{|\mathcal{E}|}{N(N-1)}$.

Network density

Graphs topological characteristics

- Let $|\mathcal{V}| = N$. If we don't consider self-loops the number of edges of an undirected graph is:

$$|\mathcal{E}| \leq \binom{N}{2} = \frac{N(N-1)}{2}$$

and for a directed graph is:

$$|\mathcal{E}| \leq N(N-1)$$

- Equality holds for a complete graph.

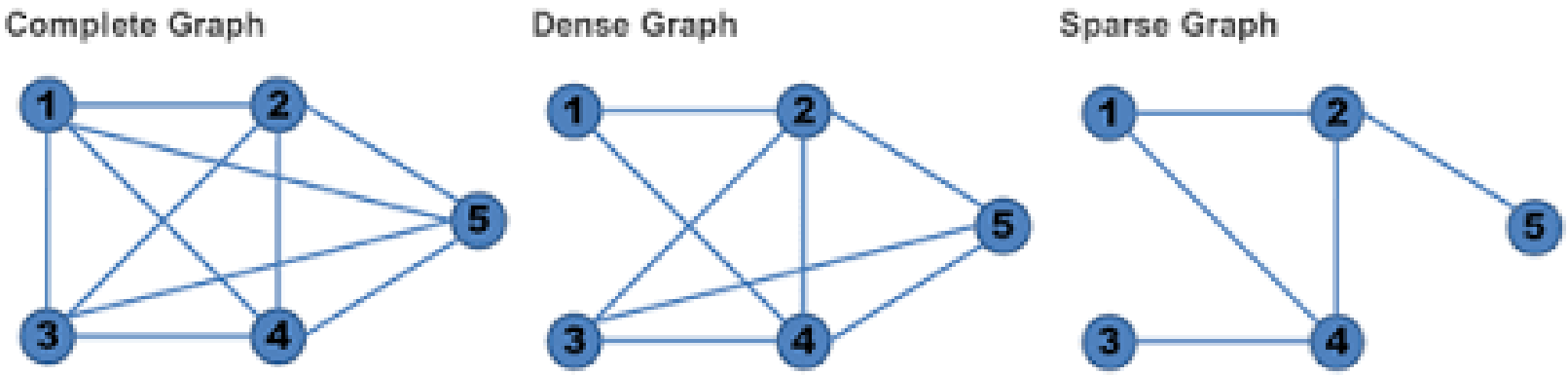
Network density

An undirected graph density is $\delta = \frac{2|\mathcal{E}|}{N(N-1)}$ and that of a directed graph is $\delta' = \frac{|\mathcal{E}|}{N(N-1)}$.

Sparse and dense networks

Graphs topological characteristics

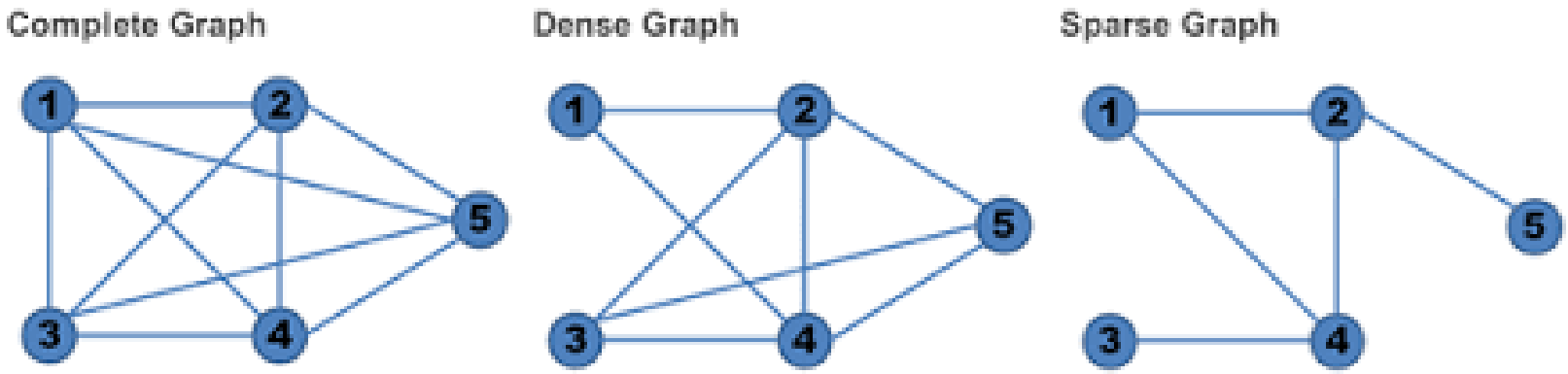
- Many networks are relatively sparse.
- A tree is a sparse connected graph, and $|\mathcal{E}| = N - 1$.
- In general, a graph is considered sparse if the number of edges is $O(N)$ and dense if it is $O(N^2)$.



Sparse and dense networks

Graphs topological characteristics

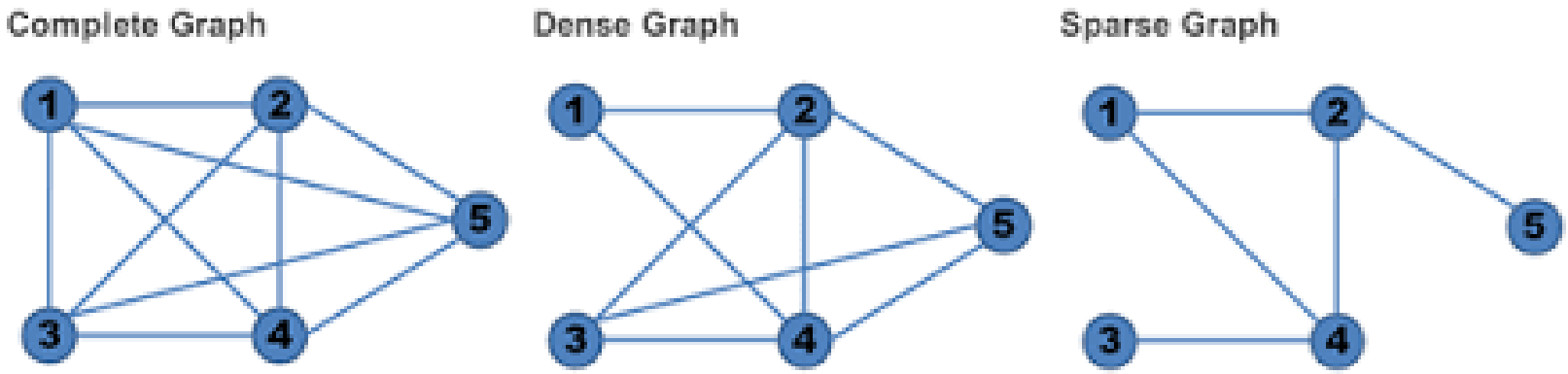
- Many networks are relatively sparse.
- A tree is a sparse connected graph, and $|\mathcal{E}| = N - 1$.
- In general, a graph is considered sparse if the number of edges is $O(N)$ and dense if it is $O(N^2)$.



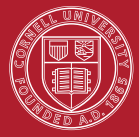
Sparse and dense networks

Graphs topological characteristics

- Many networks are relatively sparse.
- A tree is a sparse connected graph, and $|\mathcal{E}| = N - 1$.
- In general, a graph is considered sparse if the number of edges is $O(N)$ and dense if it is $O(N^2)$.



Outline

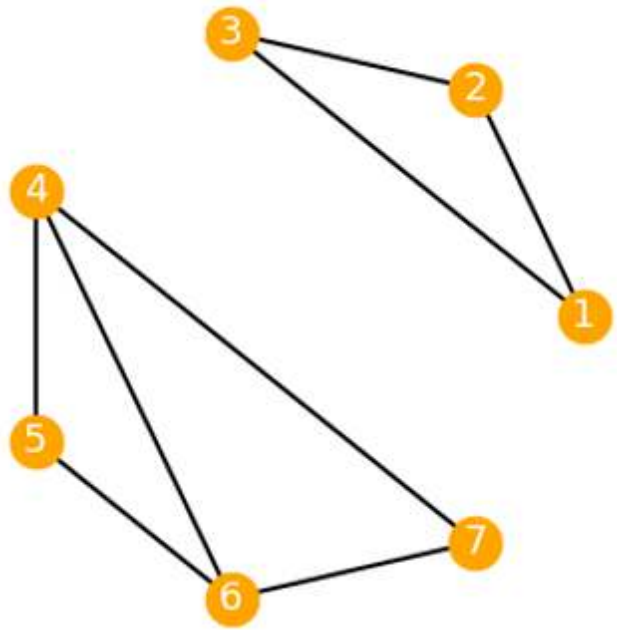


Cornell University

Disconnected graphs

- If a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is disconnected, we can divide the nodes into subsets of nodes that are strongly connected, forming a set of graphs $\mathcal{G}_k, k = 1, \ldots, K$ that are called the components of $\mathcal{G}$.
- They are such that $\mathcal{V}_k \cap \mathcal{V}_{k'} = \emptyset$, $\mathcal{E}_k \cap \mathcal{E}_{k'} = \emptyset$ and:

$$\mathcal{V} = \bigcup_{k=1}^{K} \mathcal{V}_k, \ \mathcal{E} = \bigcup_{k=1}^{K} \mathcal{E}_k$$



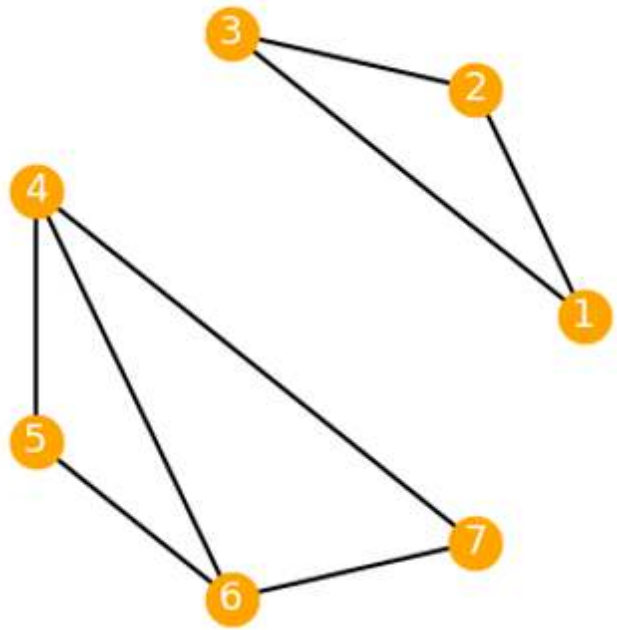
A graph with two components.

- Note that it is possible that some of these components are isolated nodes.
- A strongly connected graph is a giant component

Disconnected graphs

- If a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is disconnected, we can divide the nodes into subsets of nodes that are strongly connected, forming a set of graphs $\mathcal{G}_k, k = 1, \ldots, K$ that are called the components of $\mathcal{G}$.
- They are such that $\mathcal{V}_k \cap \mathcal{V}_{k'} = \emptyset$, $\mathcal{E}_k \cap \mathcal{E}_{k'} = \emptyset$ and:

$$\mathcal{V} = \bigcup_{k=1}^{K} \mathcal{V}_k, \ \mathcal{E} = \bigcup_{k=1}^{K} \mathcal{E}_k$$



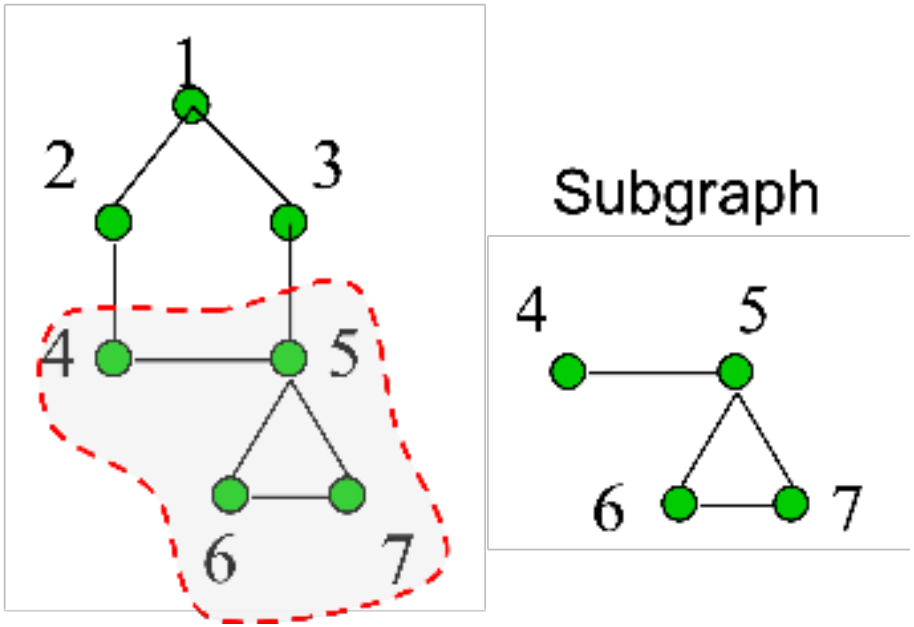
A graph with two components.

- Note that it is possible that some of these components are isolated nodes.
- A strongly connected graph is a giant component

Subgraphs

Subgraph

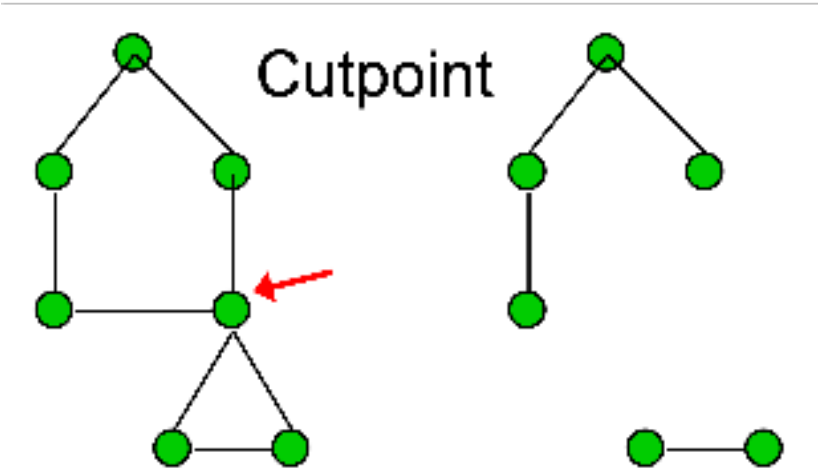
A subgraph of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a graph whose nodes are a subset $\mathcal{V}_s \subset \mathcal{V}$ and whose edges $\mathcal{E}_s \subset \mathcal{E}$ are all edges connecting pairs of vertices in $\mathcal{V}_s$.



Cutpoints

Cutpoint

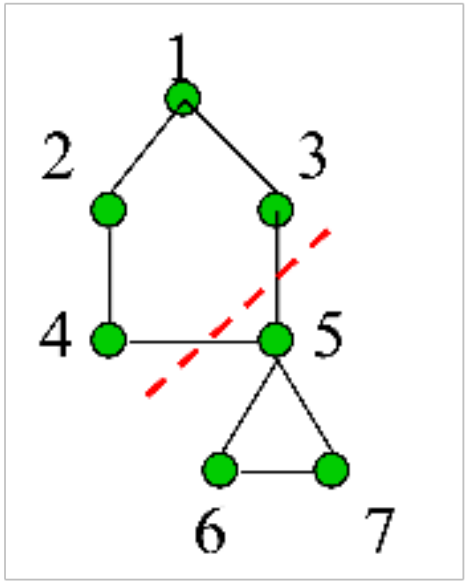
If removing a vertex from the graph and its edges to the rest of the network increases the number of **graph components** then such vertex is a cutpoint.



Network cuts

Network cut

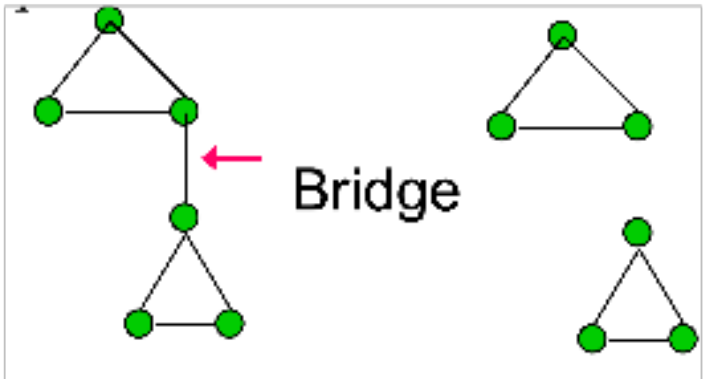
A network cut is a subset of edges $\mathcal{E}_{i,j} \subset \mathcal{E}$ such that removing those edges disconnects the network in two components with nodes respectively in $\mathcal{V}_i$ and $\mathcal{V}_j$ that form a partition of $\mathcal{V}$, i.e. $\mathcal{V}_i \cap \mathcal{V}_j = \emptyset$ and $\mathcal{V}_i \cup \mathcal{V}_j = \mathcal{V}$.



The cut is {(3,5),(4,5)}

Bridge

A bridge is a network cut that consists only of one edge.



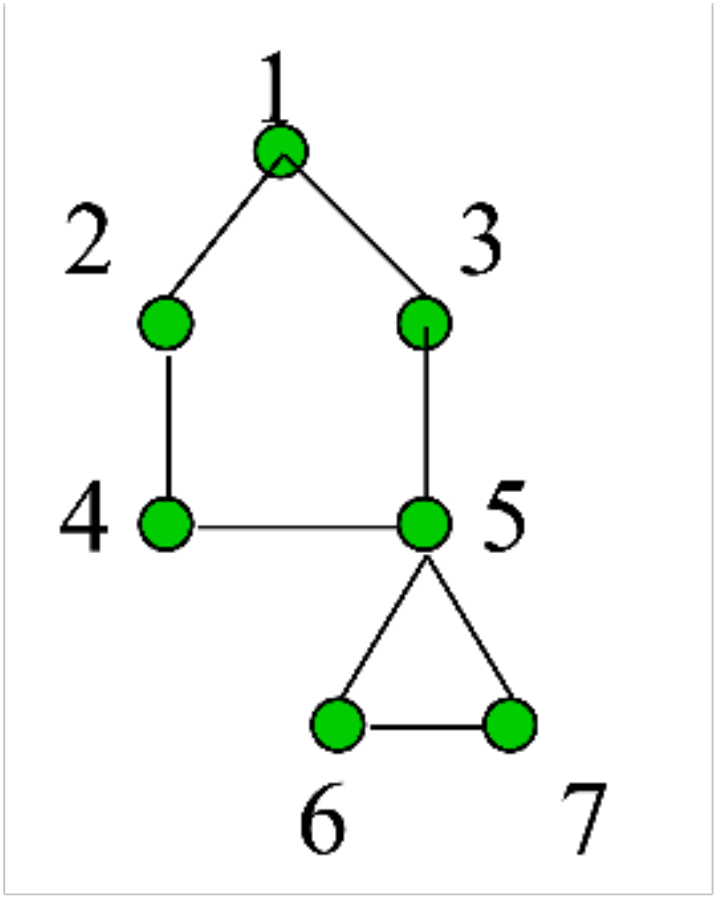
Connectivity

Vertex/Node Connectivity

The network node connectivity $\kappa(\mathcal{G})$ is the minimum number of nodes $\mathcal{V}$ whose removal disconnects the graph. A graph is k-connected if $\kappa(G) = k$.

Edge Connectivity

The network connectivity is the size of its minimum edge cut $\lambda(\mathcal{G}) := \min_{ij} \mathcal{E}_{i,j} \subset \mathcal{E}$ where the minization is over all possible partitions of the nodes in two sets $\mathcal{V}_i \cap \mathcal{V}_j = \emptyset$.



$$\kappa(\mathcal{G}) = 1, \quad \lambda(\mathcal{G}) = 2$$

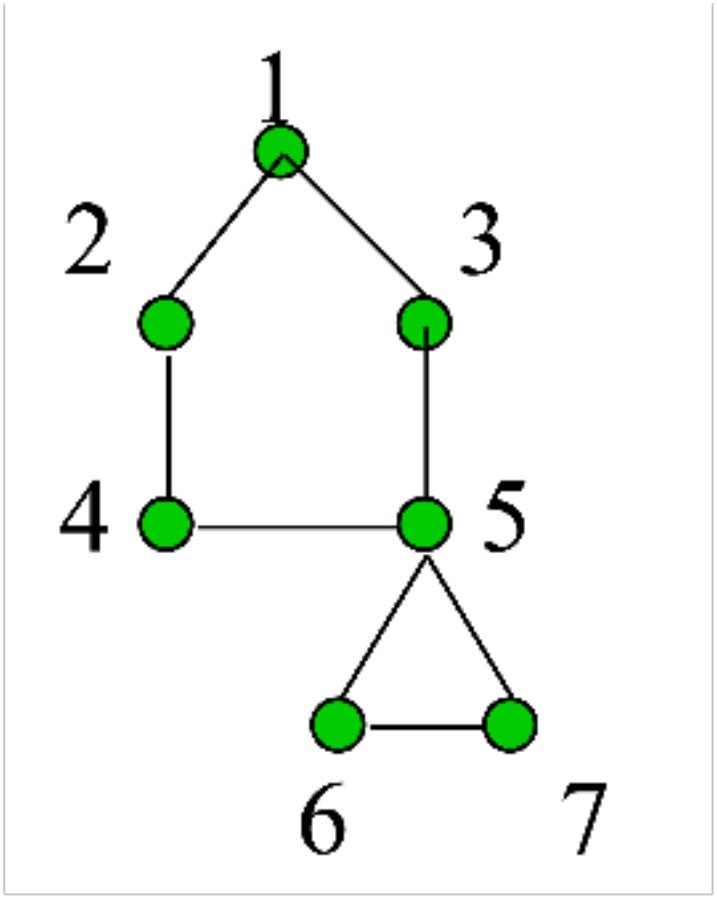
Connectivity

Vertex/Node Connectivity

The network node connectivity $\kappa(\mathcal{G})$ is the minimum number of nodes $\mathcal{V}$ whose removal disconnects the graph. A graph is k-connected if $\kappa(G) = k$.

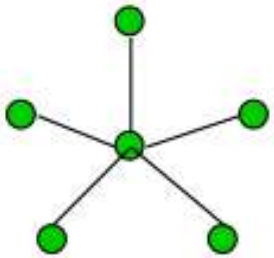
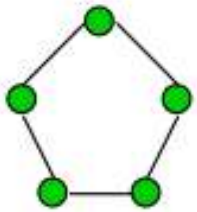
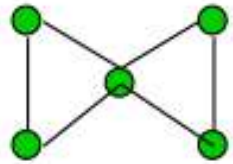
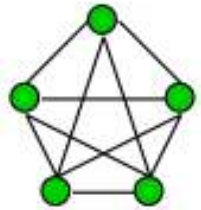
Edge Connectivity

The network connectivity is the size of its minimum edge cut $\lambda(\mathcal{G}) := \min_{ij} \mathcal{E}_{i,j} \subset \mathcal{E}$ where the minization is over all possible partitions of the nodes in two sets $\mathcal{V}_i \cap \mathcal{V}_j = \emptyset$.



$$\kappa(\mathcal{G}) = 1, \quad \lambda(\mathcal{G}) = 2$$

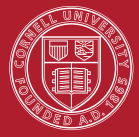
Connectivity and edge connectivity

Graph				
Edge-Connectivity	1	2	2	4
Connectivity	1	2	1	4

Whitney's theorem

For any graph $\mathcal{G}$, $\kappa(\mathcal{G}) \leq \lambda(\mathcal{G}) \leq \delta(\kappa(\mathcal{G}))$, where $\delta(\kappa(\mathcal{G})) := \min_{v \in \mathcal{N}} d_v$ is the minimum degree in the graph.

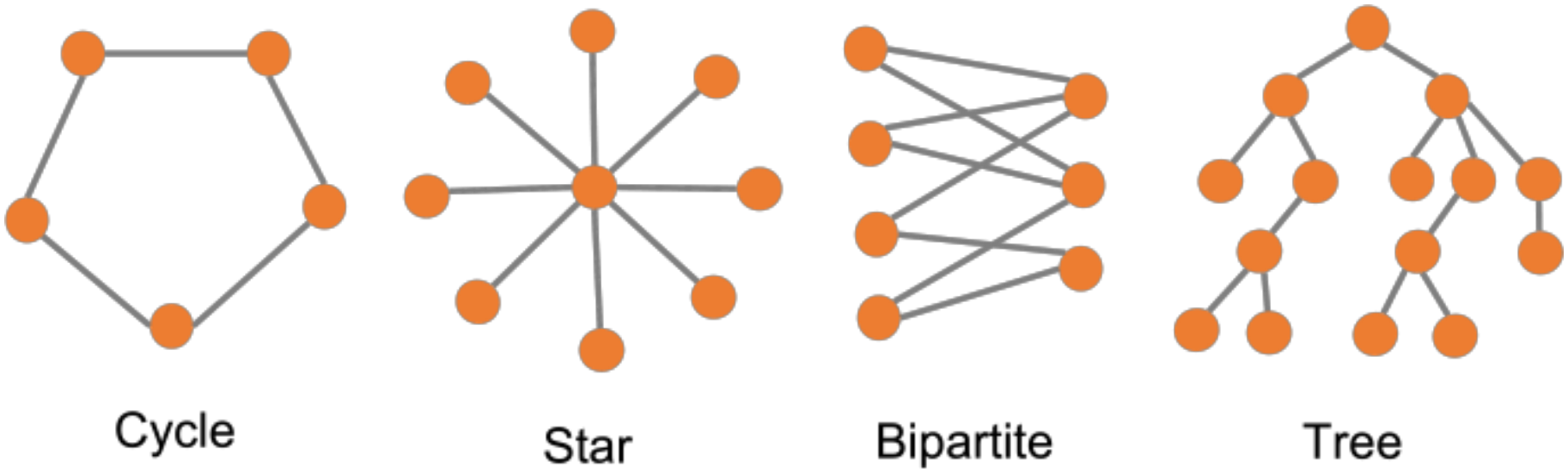
Outline



Cornell University

Specific families of graph topologies

Graphs



All these graphs can be directed or undirected (as shown).

Graphs without cycles

- We are going to talk more in the next lecture about the definitions of **path** in a graph and **cycles**.
- They are both quite intuitive: a path as a sequence of edges that are not repeated and that allow you to go from one node v to another node u in the graph through a series of intermediate edges (*hops*).
- A cycle is like a path but connects node v with itself.

Directed Acyclic Graph (DAG)

A directed graph with no cycles is a DAG.

- DAGs applications: biology (evolution, family trees, epidemiology), information science (citation networks), computation (scheduling tasks), learning (graphical models).

Graphs without cycles

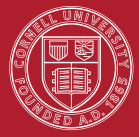
- We are going to talk more in the next lecture about the definitions of **path** in a graph and **cycles**.
- They are both quite intuitive: a path as a sequence of edges that are not repeated and that allow you to go from one node v to another node u in the graph through a series of intermediate edges (*hops*).
- A cycle is like a path but connects node v with itself.

Directed Acyclic Graph (DAG)

A directed graph with no cycles is a DAG.

- DAGs applications: biology (evolution, family trees, epidemiology), information science (citation networks), computation (scheduling tasks), learning (graphical models).

Graphs without cycles



Cornell University

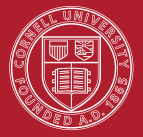
- We are going to talk more in the next lecture about the definitions of **path** in a graph and **cycles**.
- They are both quite intuitive: a path as a sequence of edges that are not repeated and that allow you to go from one node v to another node u in the graph through a series of intermediate edges (*hops*).
- A cycle is like a path but connects node v with itself.

Directed Acyclic Graph (DAG)

A directed graph with no cycles is a DAG.

- DAGs applications: biology (evolution, family trees, epidemiology), information science (citation networks), computation (scheduling tasks), learning (graphical models).

Graphs without cycles (cont.)



Cornell University

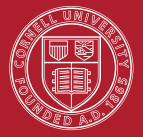
Tree

A **tree** is an undirected graph in which any two vertices are connected by exactly one path, or equivalently a connected acyclic undirected graph.

Polytree

A **polytree** is a DAG whose underling undirected topology is a tree.

Graphs without cycles (cont.)



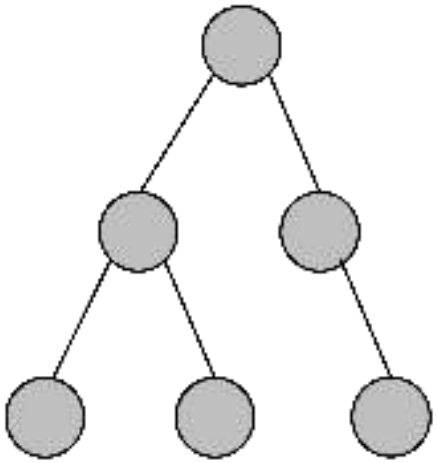
Tree

A **tree** is an undirected graph in which any two vertices are connected by exactly one path, or equivalently a connected acyclic undirected graph.

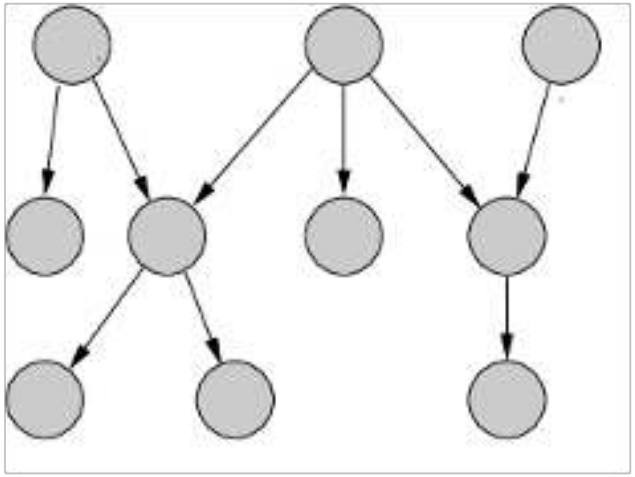
Polytree

A **polytree** is a DAG whose underling undirected topology is a tree.

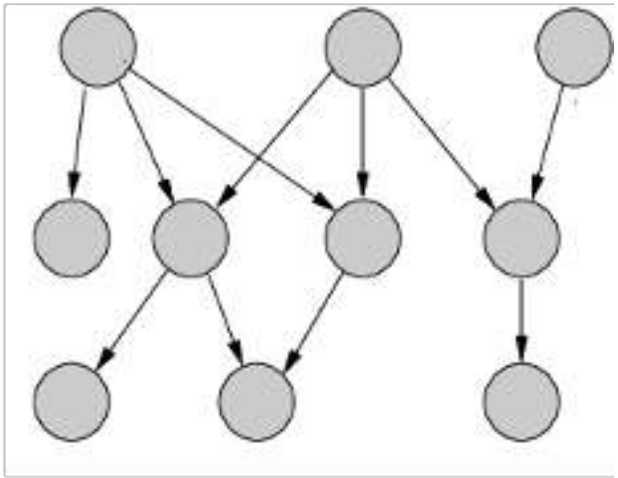
Graphs without cycles: examples



A Tree

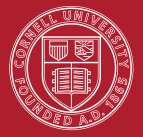


A Polytree (also a DAG)



A DAG that is not a Polytree

Graphs without cycles



Cornell University

Forest

A **forest** is an undirected graph in which any two vertices are connected by at most one path. They are a disjoint union of trees.

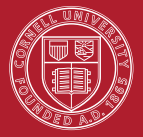
Polyforest

A **polyforest** (or directed forest or oriented forest) is a directed acyclic graph whose underlying undirected graph is a forest.

Leaf

The **leafs** of a tree (polytree, forest, polyforest) are nodes with degree 1. Every other vertex v with $d_v > 1$ is internal.

Graphs without cycles



Forest

A **forest** is an undirected graph in which any two vertices are connected by at most one path. They are a disjoint union of trees.

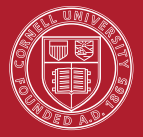
Polyforest

A **polyforest** (or directed forest or oriented forest) is a directed acyclic graph whose underlying undirected graph is a forest.

Leaf

The **leafs** of a tree (polytree, forest, polyforest) are nodes with degree 1. Every other vertex v with $d_v > 1$ is internal.

Graphs without cycles



Forest

A **forest** is an undirected graph in which any two vertices are connected by at most one path. They are a disjoint union of trees.

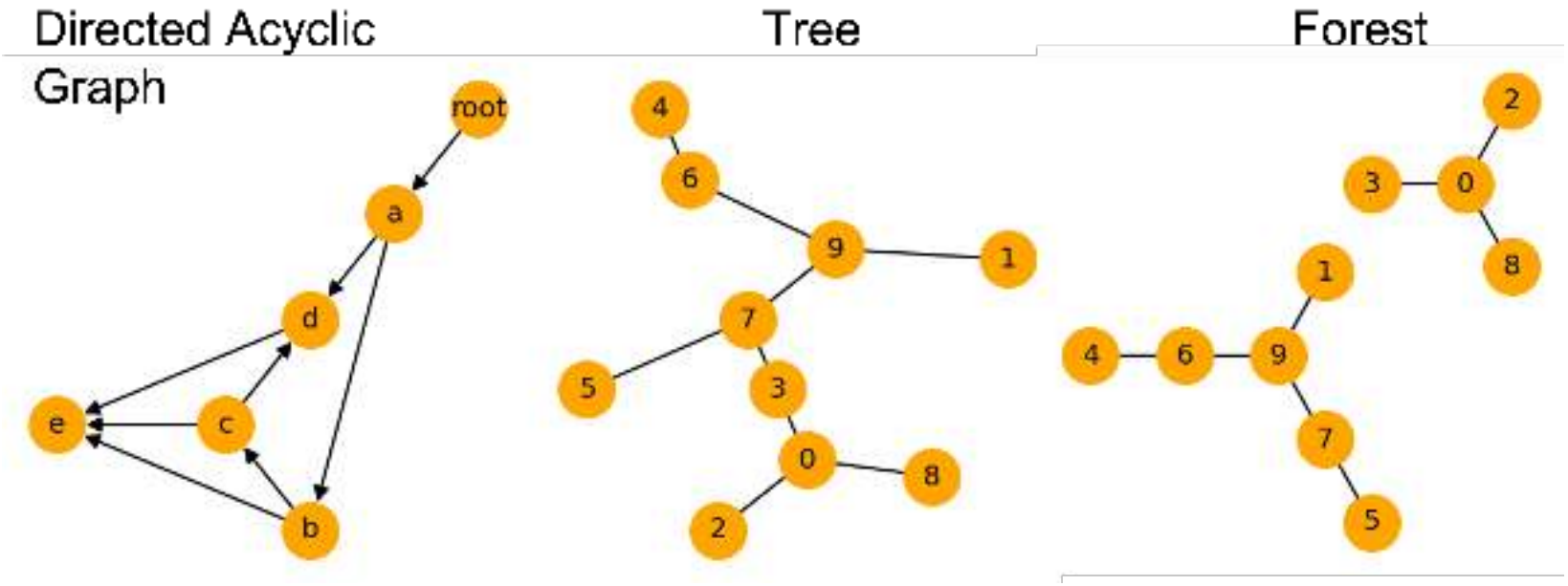
Polyforest

A **polyforest** (or directed forest or oriented forest) is a directed acyclic graph whose underlying undirected graph is a forest.

Leaf

The **leafs** of a tree (polytree, forest, polyforest) are nodes with degree 1. Every other vertex v with $d_v > 1$ is internal.

Examples

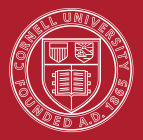


- To visualize DAGs is useful to apply what is called topological sorting of its nodes. The Networkx function is NX.TOPOLOGICAL_SORT(GRAPH).

Topological sorting

A linear ordering of its vertices such that for every directed edge from vertex v to vertex u, v comes before u in the. ordering.

Partial ordering for DAG

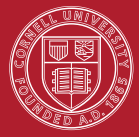


- Node v of a DAG is *rechable* from node u if there is a path connecting them.
- For DAG the topological ordering is associated to what is a **partial ordering** of the nodes that is based on reachability:

$$u \leq v \quad \Leftrightarrow \quad \text{if } v \text{ is reachable from } u$$

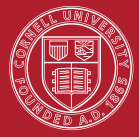
- The partial ordering is not *one to one*, in the sense that multiple DAGs can lead to the same partial ordering of a set of nodes.
- A **topological ordering** of nodes into a sequence is such that for every edge the start node of the edge occurs earlier than the end node of the edge. It is possible only for DAGs

Properties of trees



- Trees are such that $|\mathcal{E}| = |\mathcal{V}| - 1$. The condition $|\mathcal{E}| \geq |\mathcal{V}|$ is the simplest check to determine if an undirected graph has cycles!
- For a forest, we can count the number of trees n_{tree} since $|\mathcal{E}| = |\mathcal{V}| - n_{\text{tree}}$
- In a rooted tree, one internal node is designated as the root. The height or depth of a rooted tree, is the maximum path length from root to leaf.

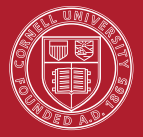
Properties of trees



Cornell University

- Trees are such that $|\mathcal{E}| = |\mathcal{V}| - 1$. The condition $|\mathcal{E}| \geq |\mathcal{V}|$ is the simplest check to determine if an undirected graph has cycles!
- For a forest, we can count the number of trees n_{tree} since $|\mathcal{E}| = |\mathcal{V}| - n_{\text{tree}}$
- In a rooted tree, one internal node is designated as the root. The height or depth of a rooted tree, is the maximum path length from root to leaf.

Properties of trees



Cornell University

- Trees are such that $|\mathcal{E}| = |\mathcal{V}| - 1$. The condition $|\mathcal{E}| \geq |\mathcal{V}|$ is the simplest check to determine if an undirected graph has cycles!
- For a forest, we can count the number of trees n_{tree} since $|\mathcal{E}| = |\mathcal{V}| - n_{\text{tree}}$
- In a rooted tree, one internal node is designated as the root. The height or depth of a rooted tree, is the maximum path length from root to leaf.

Combinatorial studies: Ex. Number of trees

- For most families of graphs someone has attempted to calculate how many possible different graphs of that type exist. These are typically some special combination.
- They can be useful when reasoning on random graphs.

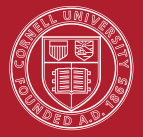
Number of possible trees

The number of trees whose nodes have a specific tuple of degrees $\boldsymbol{d} = (d_1, \ldots, d_{|\mathcal{V}|})$ is the multinomial coefficient:

$$\kappa(\boldsymbol{d}) = \binom{n-2}{d_1 - 1, d_2 - 1, \ldots, d_{|\mathcal{V}|} - 1}$$

- Modeling trees as the outcome of a random experiment, then $1/\kappa(\boldsymbol{d})$ can be considered as the *probability* of a certain tree with degrees $\boldsymbol{d}$.

Bipartitie Graphs



Cornell University

Bipartitie Graph

A Bipartitie Graphs (a.k.a. Bigraph) is a graph whose vertices can be divided into two disjoint and independent sets $\mathcal{A}$ and $\mathcal{B}$, called the *parts of the graph* $\mathcal{G}$, so that every edge $e \in \mathcal{E}$ is $e = (a, b), a \in \mathcal{A}, \quad b \in \mathcal{B}$. A *balanced* Bipartitie Graph has $|\mathcal{A}| = |\mathcal{B}|$.

- A graph is bipartite if and only if it does not contain an odd cycle. Trees are bipartite graphs (...it has no cycles at all!)
- The algorithm Depth First Search (DFS) can be used to verify if the graph is bipartite.

Bipartitie Graphs

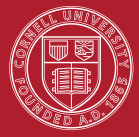
Cornell University

Bipartitie Graph

A Bipartitie Graphs (a.k.a. Bigraph) is a graph whose vertices can be divided into two disjoint and independent sets $\mathcal{A}$ and $\mathcal{B}$, called the *parts of the graph* $\mathcal{G}$, so that every edge $e \in \mathcal{E}$ is $e = (a, b), a \in \mathcal{A}, \quad b \in \mathcal{B}$. A *balanced* Bipartitie Graph has $|\mathcal{A}| = |\mathcal{B}|$.

- A graph is bipartite if and only if it does not contain an odd cycle. Trees are bipartite graphs (...it has no cycles at all!)
- The algorithm Depth First Search (DFS) can be used to verify if the graph is bipartite.

Bipartitie Graphs



Cornell University

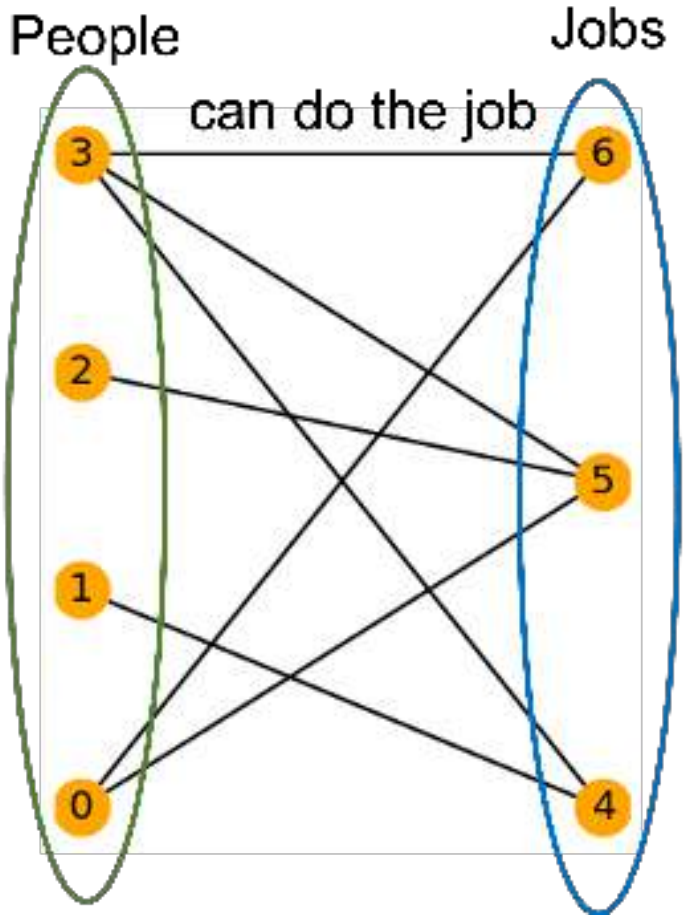
Bipartitie Graph

A Bipartitie Graphs (a.k.a. Bigraph) is a graph whose vertices can be divided into two disjoint and independent sets $\mathcal{A}$ and $\mathcal{B}$, called the *parts of the graph* $\mathcal{G}$, so that every edge $e \in \mathcal{E}$ is $e = (a, b), a \in \mathcal{A}, \quad b \in \mathcal{B}$. A *balanced* Bipartitie Graph has $|\mathcal{A}| = |\mathcal{B}|$.

- A graph is bipartite if and only if it does not contain an odd cycle. Trees are bipartite graphs (...it has no cycles at all!)
- The algorithm Depth First Search (DFS) can be used to verify if the graph is bipartite.

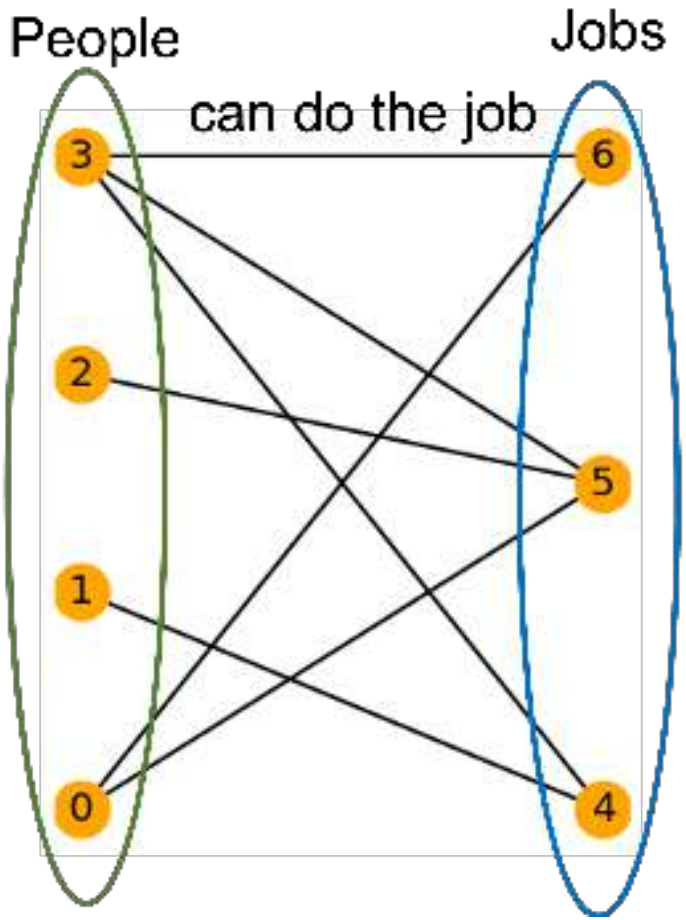
Bipartitie Graphs models

- Bipartite graphs are good models for *group membership*
- For example:suppose we have a set of people and set of jobs. Each person can do only some of the jobs.
- A bipartite graph that connects people with jobs is a natural representation.



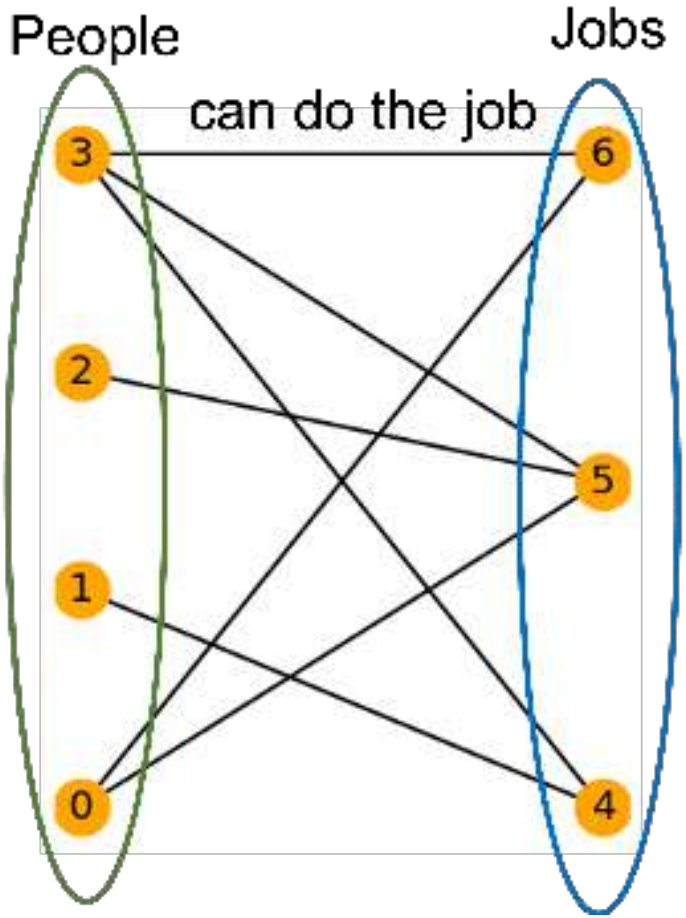
Bipartitie Graphs models

- Bipartite graphs are good models for *group membership*
- For example:suppose we have a set of people and set of jobs. Each person can do only some of the jobs.
- A bipartite graph that connects people with jobs is a natural representation.

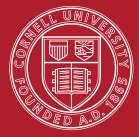


Bipartitie Graphs models

- Bipartite graphs are good models for *group membership*
- For example:suppose we have a set of people and set of jobs. Each person can do only some of the jobs.
- A bipartite graph that connects people with jobs is a natural representation.



Nodal degrees properties



Cornell University

Degrees and edges

For a general undirected graph:

$$\text{Handshaking theorem} \quad \sum_{v \in \mathcal{V}} d_v = 2|\mathcal{E}|$$

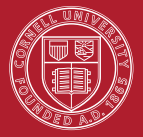
The proof is simple $\sum_{v \in \mathcal{V}} d_v = |\bigcup_{v \in \mathcal{V}} \mathcal{N}_v|$ in which each edge (u, v) is counted twice.

Degrees and edges of a bipartite graph

For a bigraph:

$$\sum_{u \in \mathcal{A}} d_u = \sum_{v \in \mathcal{B}} d_v = |\mathcal{E}|$$

Nodal degrees properties



Cornell University

Degrees and edges

For a general undirected graph:

$$\text{Handshaking theorem} \quad \sum_{v \in \mathcal{V}} d_v = 2|\mathcal{E}|$$

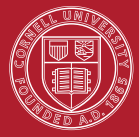
The proof is simple $\sum_{v \in \mathcal{V}} d_v = |\bigcup_{v \in \mathcal{V}} \mathcal{N}_v|$ in which each edge (u, v) is counted twice.

Degrees and edges of a bipartite graph

For a bigraph:

$$\sum_{u \in \mathcal{A}} d_u = \sum_{v \in \mathcal{B}} d_v = |\mathcal{E}|$$

Nodal degrees properties



Cornell University

Degrees and edges

For a general undirected graph:

$$\text{Handshaking theorem} \quad \sum_{v \in \mathcal{V}} d_v = 2|\mathcal{E}|$$

The proof is simple $\sum_{v \in \mathcal{V}} d_v = |\bigcup_{v \in \mathcal{V}} \mathcal{N}_v|$ in which each edge (u, v) is counted twice.

Degrees and edges of a bipartite graph

For a bigraph:

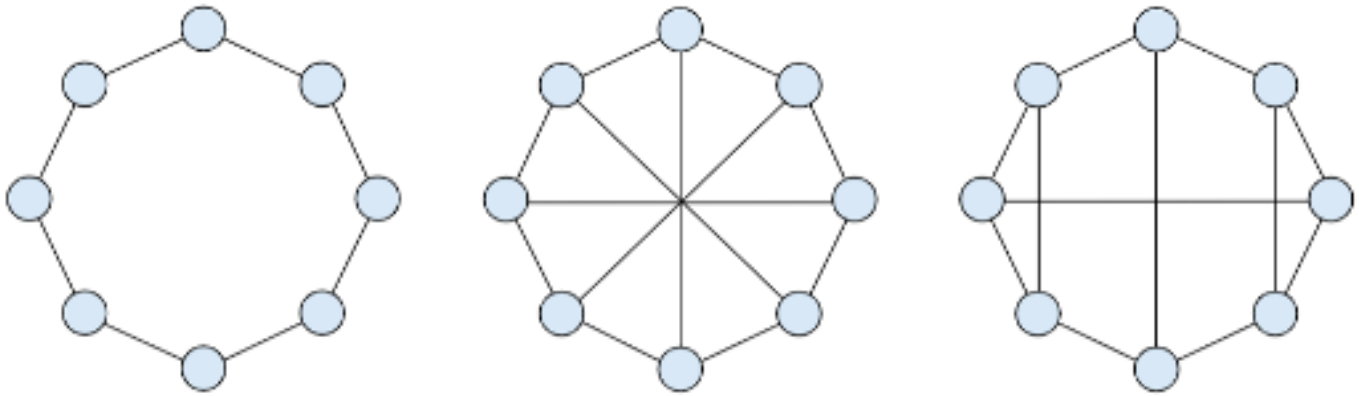
$$\sum_{u \in \mathcal{A}} d_u = \sum_{v \in \mathcal{B}} d_v = |\mathcal{E}|$$

Regular graphs

Cornell University

The d-regular graph

In a d-regular graph every node has the same degree d.



Regular tree

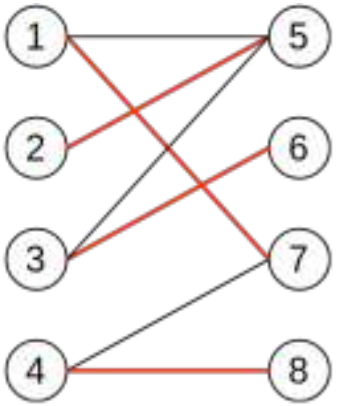
In a d-regular tree all internal nodes have the same degree.

Balanced graphs

Cornell University

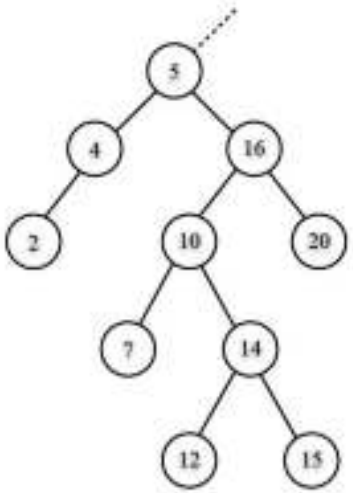
Balanced bipartitie graph

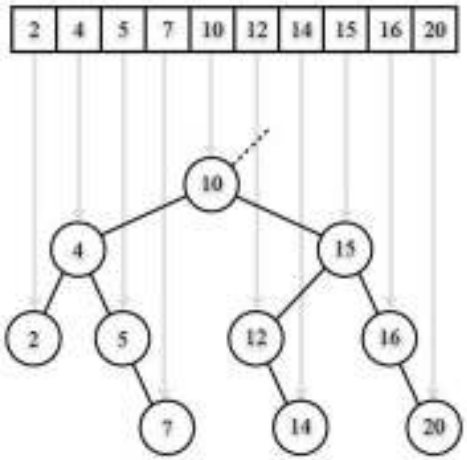
A *balanced* bipartitie graph has $|\mathcal{A}| = |\mathcal{B}|$.



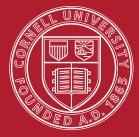
Balanced tree

In a height-balanced tree the left and right subtrees height of any node differs at most by one.



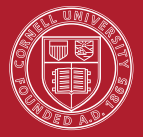


Outline



Cornell University

Embedding graphs on surfaces



Cornell University

Graph embeddings

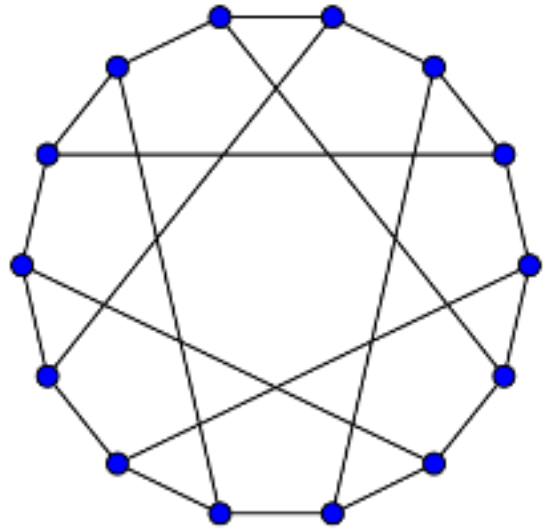
A graph $\mathcal{G}$ embedding is a representation of the graph nodes as points and its edges as arcs on a surface S such that:

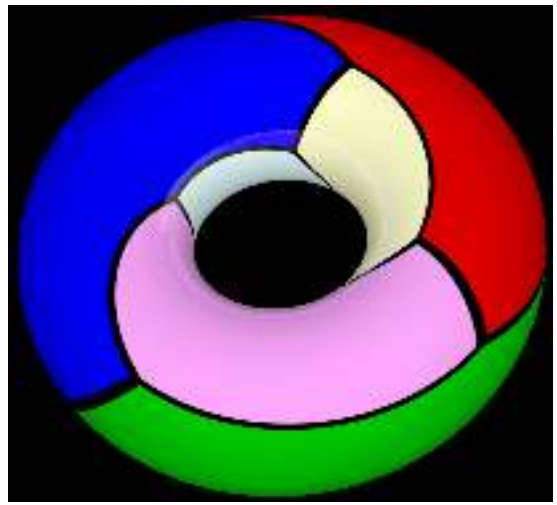
1. The verteces $v \in \mathcal{V}$ are distinct points $x_v \in \mathcal{S}$
2. Edges $(u, v) = e \in \mathcal{E}$ are arcs/curves on the surface $\mathcal{S}$ whose end-points are x_u and x_v.
3. The arcs cannot intersect with other arcs, or include points associated with other vertexes.

Note:

- An *embedding* asks whether the graph can be drawn on a surface with *no* edge crossings (except at common endpoints).

Embedding graphs on surfaces



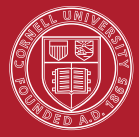


- Note that a *drawing* of a graph in the plane may have crossings *even if* the graph is planar (see right).
- If the embedding is possible he complement of the graph is the disjoint union of connected components called *faces* and are simply connected (can be deformed into a disk).
- On the right, the Heawood graph and the associated map embedded in the torus.

Key takeaway

Planarity is a property of the abstract graph, not of a particular picture.

What changes, what doesn't



Cornell University

Topology mindset (informal)

Topology studies properties preserved under continuous deformations: *stretching* and *bending* are allowed, but *tearing* and *gluing* are not.

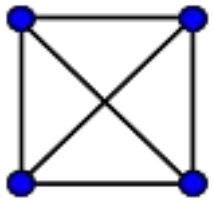
Combinatorial / graph invariants

– adjacency and degree sequence
– connected components
– existence of cycles (odd/even cycles)
– cuts, bridges, vertex/edge connectivity

Not invariants (depend on the drawing)

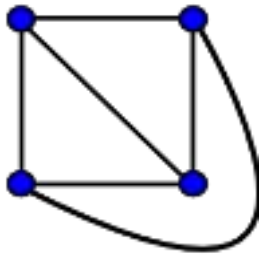
– lengths / angles of edges
– the apparent "shape" of the drawing
– whether edges cross *in that drawing*

Planar graphs



Planar graphs

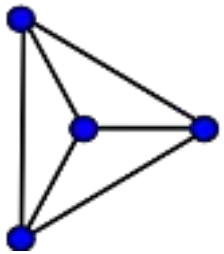
A planar graph is a graph that can be embedded in the plane drawn in a way that no edges cross each other.



Euler's theorem

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a connected planar graph, and let $\mathcal{F}$ be the set of faces of $\mathcal{G}$ in a planar embedding. Then,

$$|\mathcal{V}| - |\mathcal{E}| + |\mathcal{F}| = 2$$



Genus: how much "surface" a graph needs

- Some graphs are *not* planar: they cannot be embedded in the plane without crossings.
- If we allow a more complex surface (e.g., a torus), some non-planar graphs become embeddable.

Genus (informal definition)

The *genus* g of a surface is the number of "handles". The *genus of a graph* is the minimum g such that the graph can be embedded on a surface of genus g without crossings.

Genus Theorem

Euler characteristic (orientable surfaces)

If a graph $G = (V, E)$ is embedded on an orientable surface of genus g with face set F, then

$$|V| - |E| + |F| \;=\; 2 - 2g.$$

- Plane / sphere: $g = 0 \quad \Rightarrow \quad |V| - |E| + |F| = 2$ (your planar Euler formula)
- Torus: $g = 1 \quad \Rightarrow \quad |V| - |E| + |F| = 0$

Two notions of "embedding" in graph data science

(A) Graph $\to$ surface

– nodes $\mapsto$ points on a surface
– edges $\mapsto$ non-intersecting arcs
– yields faces, planarity, genus
– **Goal:** respect topology (no crossings)

(B) Graph $\to \mathbb{R}^d$

– nodes $\mapsto$ vectors in $\mathbb{R}^d$
– geometry reflects similarity/diffusion (Laplacian eigenmaps, node2vec, GNN embeddings...)
– **Goal:** *create* a geometry for learning

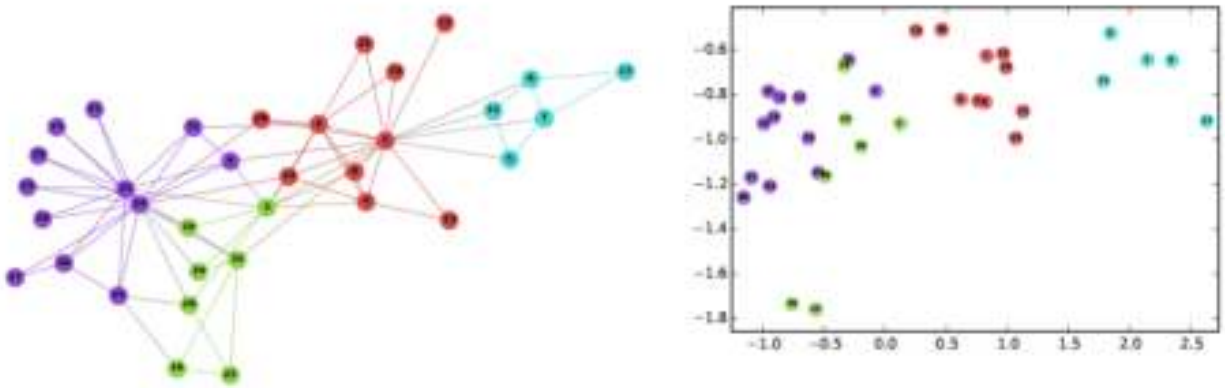
Punchline

One embedding respects topology; the other produces geometry.

- (A) In this lecture we connected graph theory with topology (faces, planarity, genus).

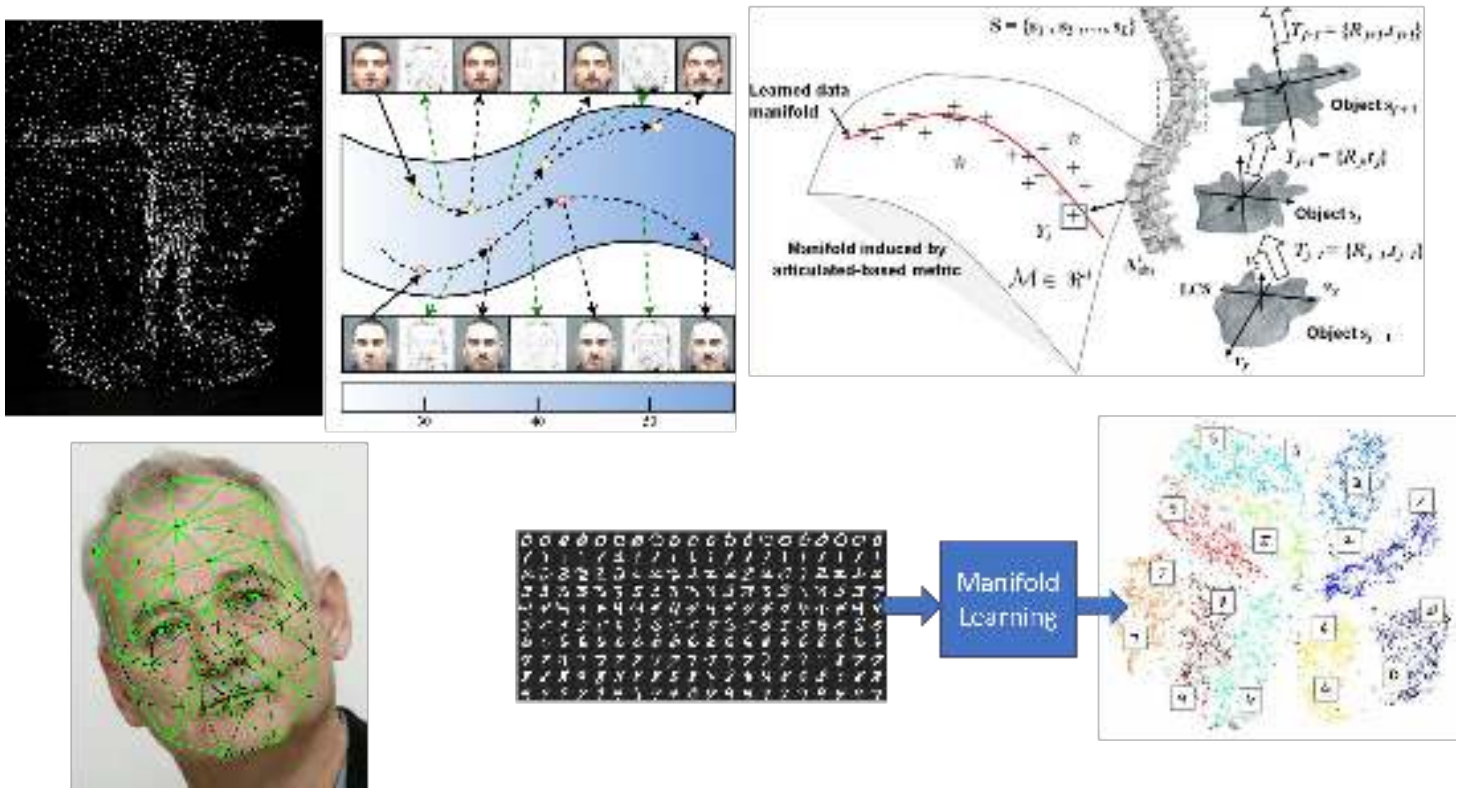
Example of graph embedding

- The graph is converted in a set of points in $\mathbb{R}^d$ that is not necessarily over a surface and that is an encoding of its nodes
- It will be discussed later in the course...

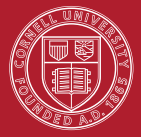


Continuous data topology converted into graphs

- This notion is used when graphs formed with data $\boldsymbol{x}_i \in \mathbb{R}^d$ $d > 1$ that are proxies for manifolds of dimension $d' < d$ containing the data. Think about point clouds for example.
- Meshes are graph representations of complex objects (e.g. faces).

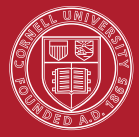


Outline



Cornell University

Concluding remarks



- In this lecture we introduced a number of definitions and type of graphs that will be useful as models for real networks.
- We will leverage these definitions in the study algebraic methods to characterize graphs.
- In this first phase, the data we are interested in analyzing are the graphs themselves and their properties.