Graphs and Trees

- Graphs Definition and Applications
- Basic terminology: degree, incidence, adjacency, neighbors
- Graph representation
  - Adjacency matrix and Adjacency list
- Graph connectivity
  - Path and Cycle
  - Connected graphs and connected components
  - Strongly connected (di)graphs and strongly connected components
- Trees and Forest

Imdad ullah Khan

Graphs

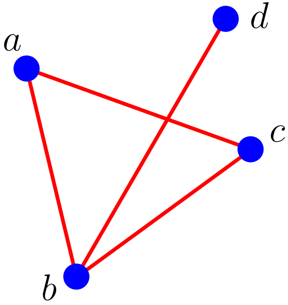
Formally: A graph is

1. a set of vertices V
2. a set of edges E, a collection of 2-sets of V

$G = (V, E)$

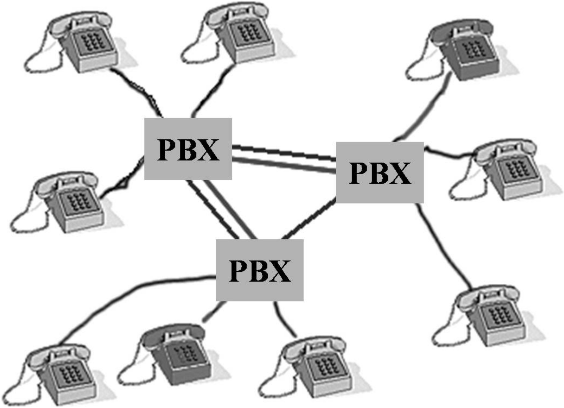
$V = \{a, b, c, d\}$

$E = \{(a, b), (a, c), (b, c), (b, d)\}$



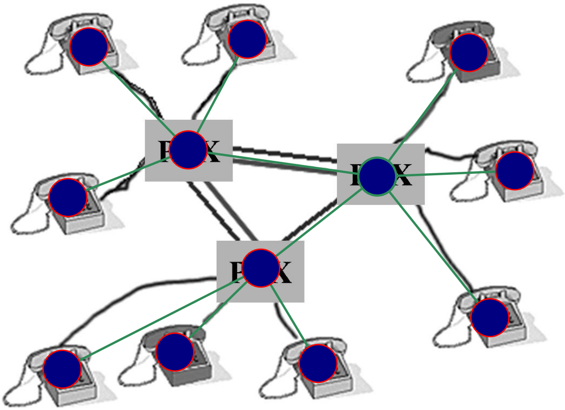
Graphs are everywhere: Communication Networks

Graphs model communication networks



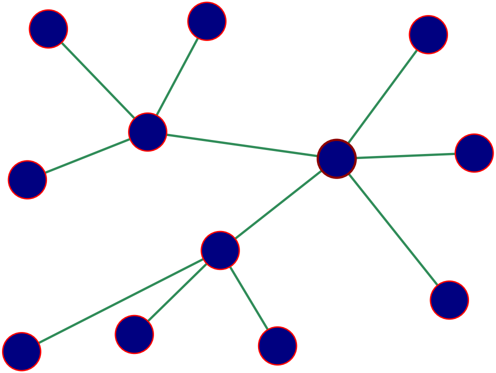
Graphs are everywhere: Communication Networks

Graphs model communication networks



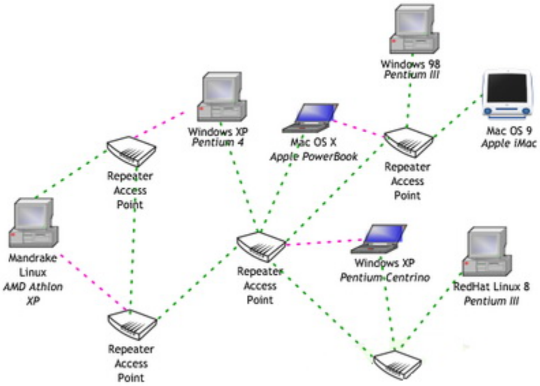
Graphs are everywhere: Communication Networks

Graphs model communication networks



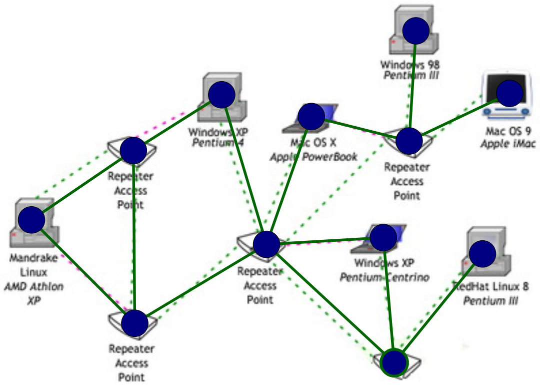
Graphs are everywhere: The Internet

Graphs models the Internet



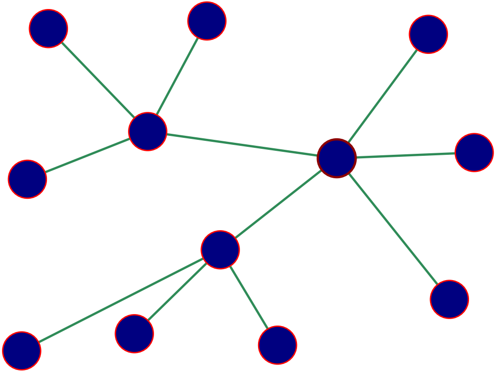
Graphs are everywhere: The Internet

Graphs models the Internet



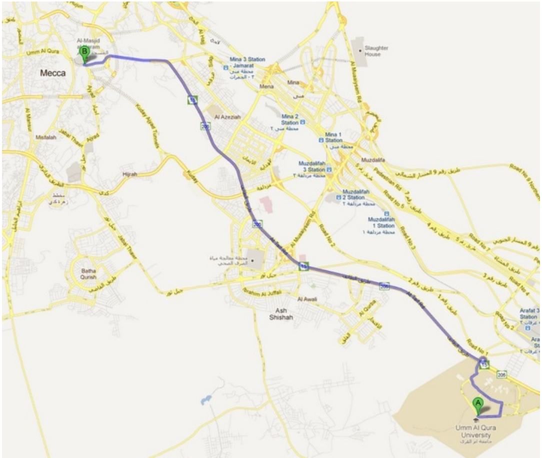
Graphs are everywhere: The Internet

Graphs models the Internet



Graphs are everywhere: The Highway Network

Graphs model highway networks

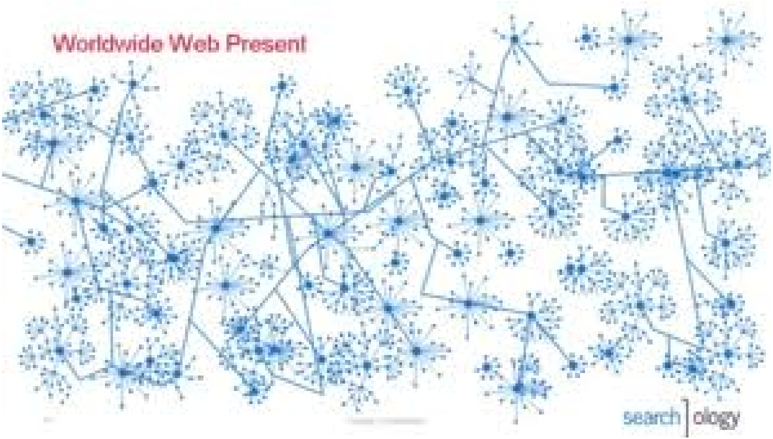


Graphs are everywhere: The Highway Network

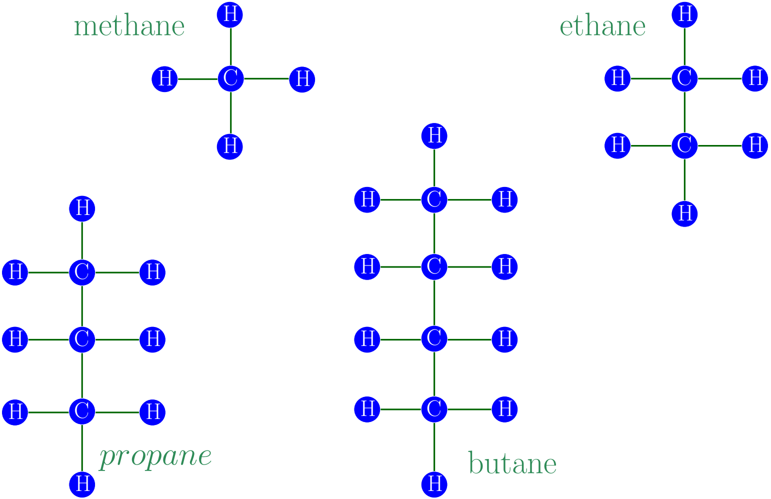
Graphs model highway networks

Graphs are everywhere: Web Graph and Social Networks

The Web Graph and Social Networks

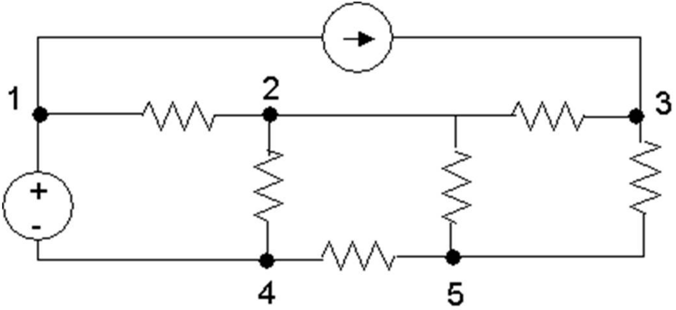


Graphs are everywhere: Scientific Networks



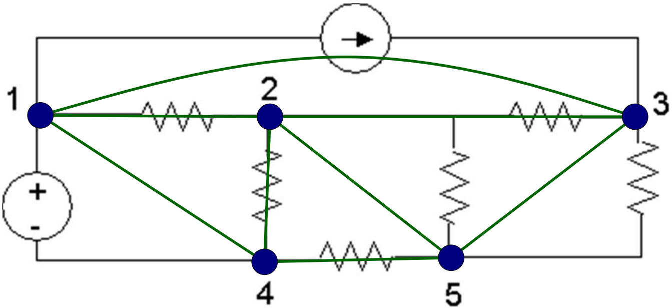
Graphs are everywhere: Circuits

What are node voltages of the following circuit?



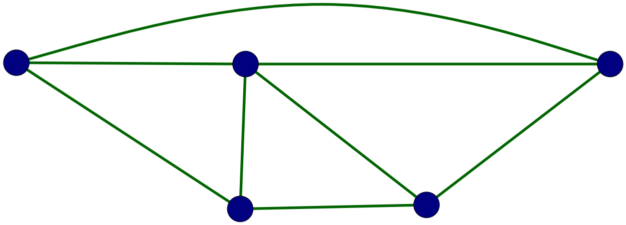
Graphs are everywhere: Circuits

What are node voltages of the following circuit?



Graphs are everywhere: Circuits

What are node voltages of the following circuit?



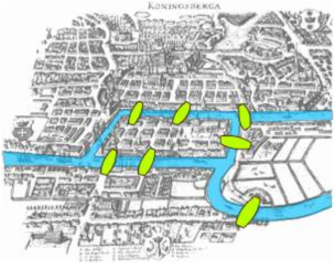
Graph Analysis

Depending on the domain of graphs and applications the area is also called

Network Analysis, Link Analysis, Social Network Analysis

Modeling, formulating and solving problems with graphs

Use tools from graph theory, linear algebra, algebraic graph theory, and algorithms for data analysis problems modeled with graphs



One of the earliest graph analysis: Euler argued that there is no way to tour the city of Konigsberg (now Kaliningrad) crossing each of the 7 bridges exactly once

Graph Analysis

Rather than individual data points or the global structure of the datasets

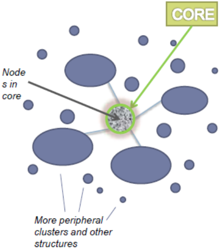
Graph Analysis focuses on pairwise interaction between objects

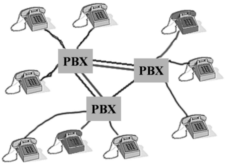
Allows to examine how pairwise interaction of entities in a network determine the behavior or function of an individual entity, groups of entities or the whole system

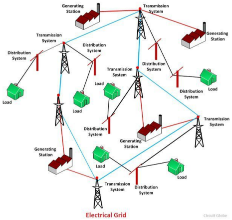
Graphs are everywhere: Six major classes of networks

Technological Networks

- Internet (Autonomous Systems connected with BGP connections)
- Telecom Network (telephone devices connected with wires or wireless)
- Power Grid (generating stations/users and transmission line)



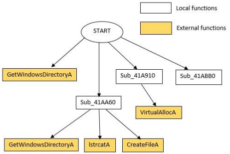


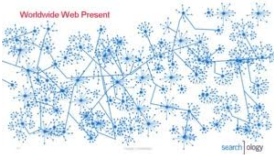


Graphs are everywhere: Six major classes of networks

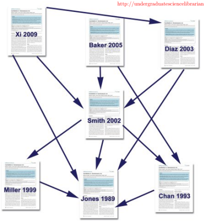
Information Networks

- Software (functions connected with function calls)
- The Web Graph (webpages and hyperlinks)
- Documents (Research papers and citations)



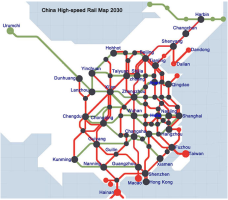


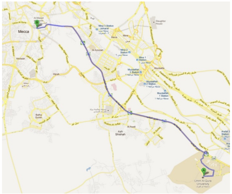
http://undergraduatesciencelibrarian.org/



Transportation Networks

- Railway System (train stations and railroad tracks)
- Highway network (Intersections and road segments)
- Air Transportation (Airports and non-stop flight)





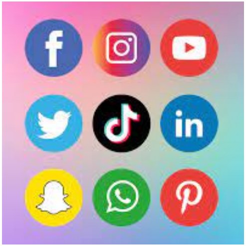
Airport code	Location
BER	Berlin-Brandenburg
BRE	Bremen
BWE	Braunschweig
CGN	Cologne
DRS	Dresden
DTM	Dortmund
DUS	Dusseldorf
FDH	Friedrichshafen
FKB	Karlsruhe
FMM	Memmingen
FMO	Muenster
FRA	Frankfurt
GWT	Westerland
HAJ	Hanover
HAM	Hamburg
HHN	Frankfurt Hahn
LEJ	Leipzig/Halle
MUC	Munich
NUE	Nuremberg
PAD	Paderborn
RLG	Rostock-Laage
SCN	Saarbruecken
STR	Stuttgart
SXF	Berlin-Schönefeld
TXL	Berlin-Tegel

Graphs are everywhere: Six major classes of networks

Social Networks

- Social Network (people and friendship/acquaintance/coworker relation)
- Online Social Network (people and friendship or following relation)

Social Network



Online Social Nework

Graphs are everywhere: Six major classes of networks

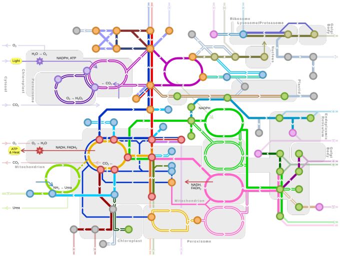
Biological Networks

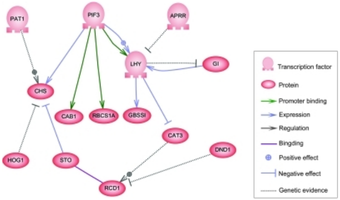
Represents interactions between biological units

ecological, evolutionary, physiological, metabolic, gene regulatory network

Most genes and proteins play a role through interactions with other proteins, genes, and biomolecules

Analyzed to understand the origin and function of cellular components, treatments for diseases, determine comorbidies and risk factors





Graphs are everywhere: Six major classes of networks

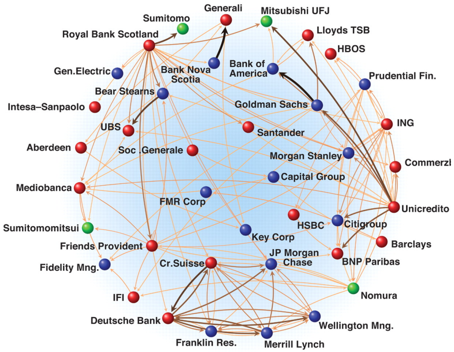
Economic Networks

Business, companies, governments interacting via credit and investment, trade relations, supply chain

REVIEW article
Front. Appl. Math. Stat., 28 August 2018 | https://doi.org/10.3389/fams.2018.00037

Understanding the World Economy in Terms of Networks: A Survey of Data-Based Network Science Approaches on Economic Networks

Frank Emmert-Streib[1,2*], Shailesh Tripathi[3], Olli Yli-Harja[2,4] and Matthias Dehmer[3,5,6]



F. Schweitzer et.al. (2009) Economic Networks: The New Challeng

Graphs are everywhere

Graph	Vertices	Edges	Flow
Communications	Telephones exchanges, computers, satellites	Cables, fiber optics, microwave relays	Voice, video, packets
Circuits	Gates, registers, processors	Wires	Current
Mechanical	Joints	Rods, beams, springs	Heat, energy
Hydraulic	Reservoirs, pumping stations, lakes	Pipelines	Fluid, oil
Financial	Stocks, currency	Transactions	Money
Transportation	Airports, rail yards, street intersections	Highways, railbeds, airway routes	Freight, vehicles, passengers

Types of Graphs: Undirected Graph

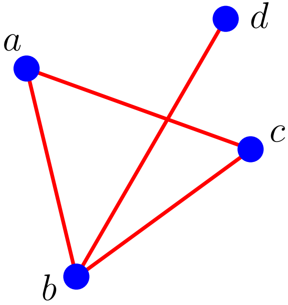
A graph is

1. a set of vertices V
2. a set of edges E, subset of 2-sets of V

$G = (V, E)$

$V = \{a, b, c, d\}$

$E = \{(a, b), (a, c), (b, c), (b, d)\}$



Types of Graphs: Undirected Graph

Let $G = (V, E)$ be a simple graph

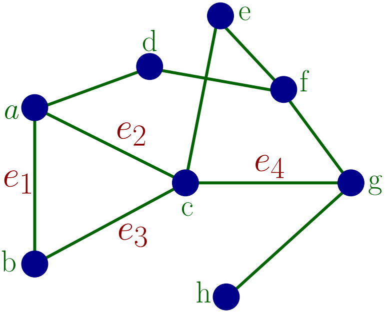
$|V| = n$, is the order of G

$|E|$ is the size of G

What is maximum possible size of G?

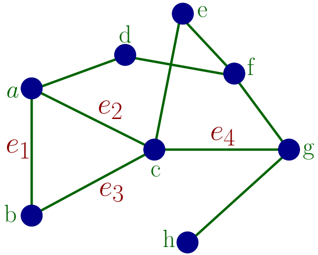
$$\binom{n}{2}$$

Graph Terminology: Incidence



- $e_1 = (a, b)$: a and b are endpoints of e_1
- a and b are adjacent
- e_1 is incident to a and b

Graph Terminology: Degree



Degree of a vertex is the number of edges incident on it

The number of vertices adjacent to it

Denoted by $deg(v)$ or $d(v)$

$$0 \leq deg(v) \leq n-1$$

Types of Graphs: Directed Graphs (digraphs)

Applications require different types of graphs

Directed Graphs (digraphs)

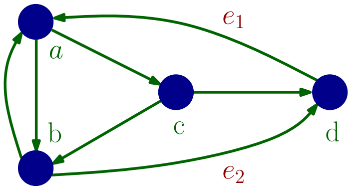
$G = (V, E)$

V is set of vertices

E is set of edges (ordered pairs)

$E \subseteq V \times V$ (ordered pairs)

irreflexive relation on V



Types of Graphs: Directed Graph

Let $G = (V, E)$ be a digraph

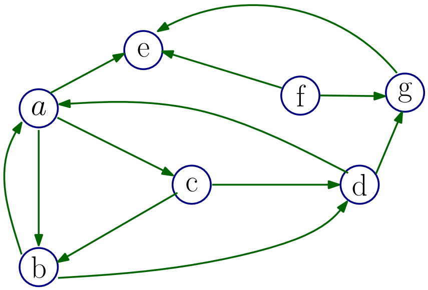
$|V| = n$, is the order of G

$|E|$ is the size of G

What is maximum possible size of G?

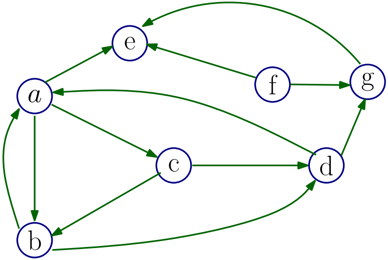
$$n(n-1)$$

Graph Terminology: Degree



- $e_1 = (d, a) \neq (a, d)$
- d is source of e_1 and a is target of e_1

Graph Terminology: Degree



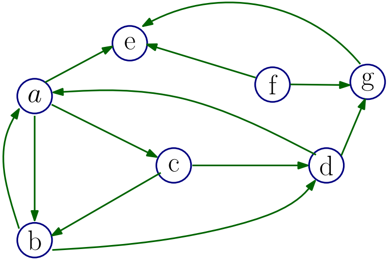
in-degree of a vertex is number of (directed) edges incoming into it

The number of edges with it as a target

Denoted by $deg^-(v)$ or $d^-(v)$

$$0 \leq deg^-(v) \leq n-1$$

Graph Terminology: Degree



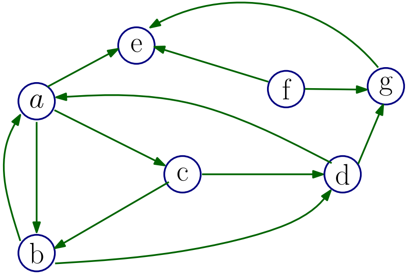
out-degree of a vertex is number of (directed) edges outgoing form it

The number of edges with it as a source

Denoted by $deg^+(v)$ or $d^+(v)$

$$0 \leq deg^+(v) \leq n-1$$

Graph Terminology: Degree



- $deg^+(a) = 3$ $\qquad$ $deg^-(a) = 2$
- $deg^+(c) = ?$ $\qquad$ $deg^-(c) = ?$
- $deg^+(g) = ?$ $\qquad$ $deg^-(g) = ?$

Graph Terminology: Neighborhood

Neighborhood of v is the set of vertices adjacent to v

$$N(v) = \left\{u : (v, u) \in E\right\}$$

- Neighborhoods in digraph
- *in-neighborhood*
- *out-neighborhood*

Handshaking Lemma

Theorem (The Handshaking Lemma)

The sum of the degrees of vertices in a graph is even

Theorem (The Handshaking Lemma (A more precise version))

$$\sum_{v} deg(v) \; = \; 2|E|$$

Theorem (The Handshaking Lemma (Another version))

The number of odd degree vertices in a graph is even

Theorem (The Handshaking Lemma (Diagraph version))

$$\sum_{v} deg^{+}(v) \; = \; \sum_{v} deg^{-}(v) \; = \; |E|$$

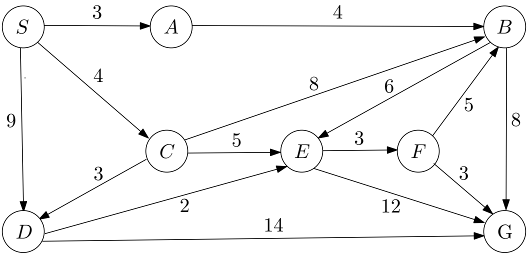
Types of Graphs: Weighted Graphs (digraphs)

Some applications work on graphs with weights on edges

Weighted Graphs (digraphs) : $G = (V, E, w)$

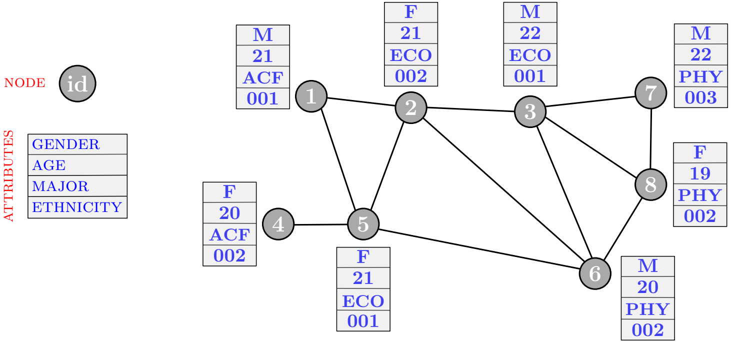
- V : Set of vertices
- E : Set of edges (directed edges)
- w : cost/weight function: $w : E \to \mathbb{R}$

▷ weights could be lengths, airfare, toll, energy



Types of Graphs: Attributed Graphs

- Each element (vertex/edge) has associated properties
- It can be directed/undirected



Graph Representation: Adjacency Matrix

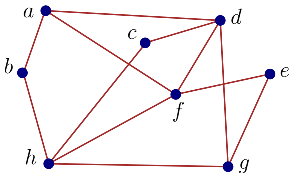
Undirected Simple Graphs

$G = (V, E)$

V is set of vertices

E is set of edges

(unordered pairs (2-subsets) of V)



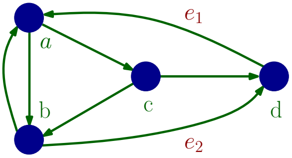
Directed Graphs (digraphs)

$G = (V, E)$

V is set of vertices

E is set of edges

(ordered pairs of V)

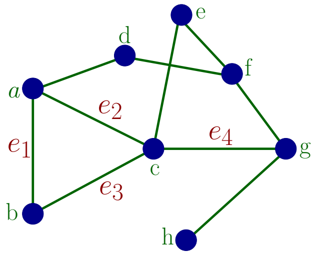


Undirected Graph Representation: Adjacency Matrix

We represent undirected $G = (V, E)$ with an **adjacency matrix** A_G

- Fix an arbitrary ordering of V
- One row for each vertex in V
- One column for each vertex in V

$$A_{ij} = \begin{cases} 1 & if (v_i, v_j) \in E \\ 0 & if (v_i, v_j) \notin E \end{cases}$$



$A_G =$

	a	b	c	d	e	f	g	h
a	0	1	1	1	0	0	0	0
b	1	0	1	0	0	0	0	0
c	1	1	0	0	1	0	1	0
d	1	0	0	0	0	1	0	0
e	0	0	1	0	0	1	0	0
f	0	0	0	1	1	0	1	0
g	0	0	1	0	0	1	0	1
h	0	0	0	0	0	0	1	0

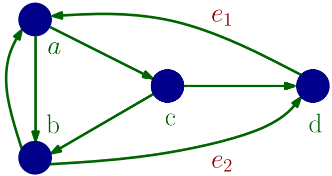
Directed Graph Representation: Adjacency Matrix

Digraph $G = (V, E)$ is a relation on V

We represent G with an **adjacency matrix** A_G

- Fix an arbitrary ordering of V
- One row for each vertex in V
- One column for each vertex in V

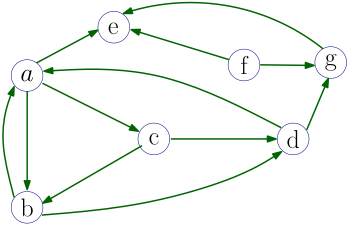
$$A_{ij} = \begin{cases} 1 & if (v_i, v_j) \in E \\ 0 & if (v_i, v_j) \notin E \end{cases}$$

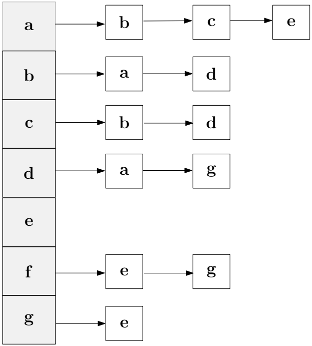


$$A = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

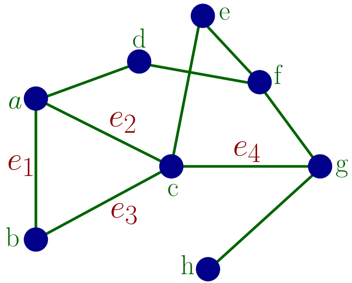
Graph Representation: Adjacency List

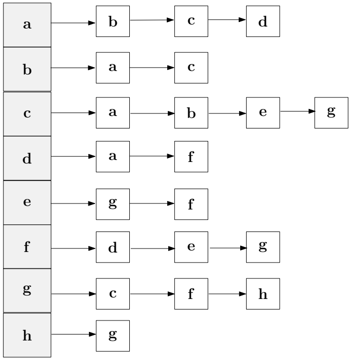
Represent digraph by listing neighbors of each vertex



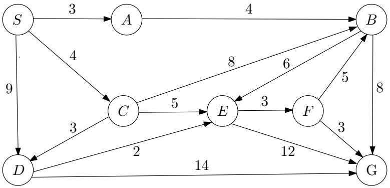


Represent undirected graph by listing neighbors of each vertex





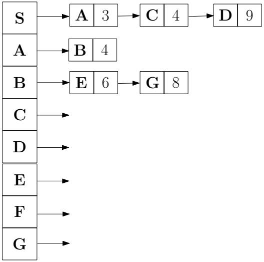
Weighted Graph Representation



Weighted Adjacency Matrix

	S	A	B	C	D	E	F	G
S	0	3	0	4	9	0	0	0
A	0	0	4	0	0	0	0	0
B	0	0	0	0	0	6	0	8
C	⋮							
D								
E								
F								
G								

Weighted Adjacency Lists



Graph Representation: Tradeoff

$G = (V, E), \quad |V| = n, \quad |E| = m$

- Adjacency matrix representation
  - requires n^2 bits
  - Edge query $\left[(a, b) \in E?\right]$ requires one memory lookup
- Adjacency list representation
  - requires $2m$ integers (vertex ids) $\sim 2m \log n$ bits
  - Edge query $\left[(a, b) \in E?\right]$ requires list traversal

Usually real-world graphs are very **sparse** $m \;=\; C \cdot n \log n$

$\triangleright$ Adjacency lists are preferred

For **dense** graphs adjacency matrix is better

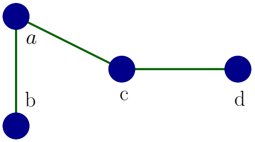
Graph Complement

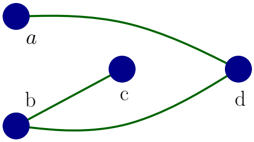
Graph Complement

$G = (V, E) \;\rightarrow\; \overline{G} = (V, \overline{E})$

$$(u, v) \in \overline{E} \;\; \mathit{iff} \;\; (u, v) \notin E$$

- Vertex set is the same
- Each edge become non-edge and each non-edge becomes edge (except self-loops)





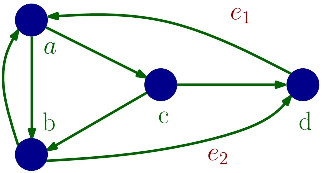
Graph Transpose

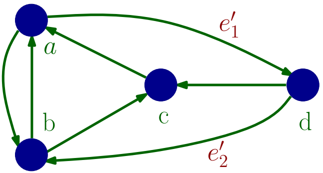
Graph Transpose

$G = (V, E) \;\rightarrow\; G^T = (V, E')$

$$(u, v) \in E' \;\; iff \;\; (v, u) \in E$$

- Vertex set is the same
- Direction/orientation of edges are reversed

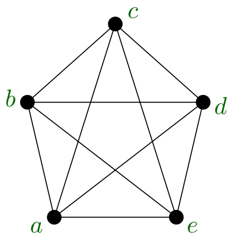




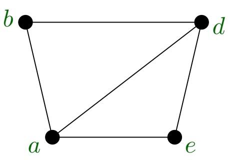
Subgraph

$H = (V', E')$ is a subgraph of $G = (V, E)$, if $V' \subseteq V$ and $E' \subseteq E$.

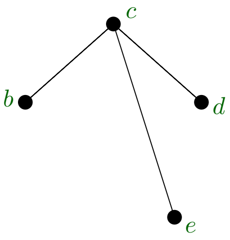
Denoted as $H \subseteq G$



G



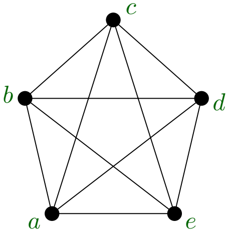
$H_1 \subseteq G$



$H_2 \subseteq G$

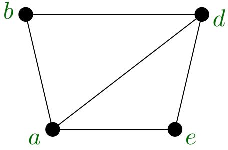
Induced Subgraph

$H = (V', E')$ is an induced subgraph of $G = (V, E)$, if $V' \subseteq V$ and $E' = E|_{V'}$ (all edges in E with both endpoints in V')



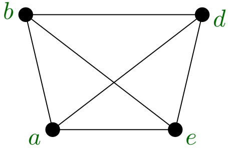
G

Not induced subgraph



$H_1 \subseteq G$

Induced subgraph



$H_2 \subseteq G$

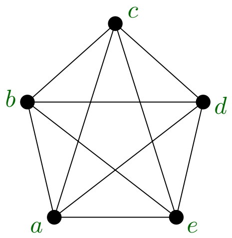
An induced subgraph is completely determined by V'

Spanning Subgraph

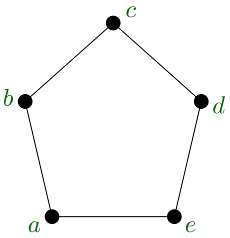
$H = (V', E')$ is a spanning subgraph of $G = (V, E)$

- if $V' = V$ and
- $E' \subseteq E$

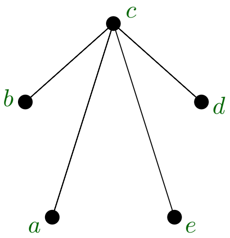
Denoted as $H \subseteq G$



G



$H_1 \subseteq G$



$H_2 \subseteq G$

Graph Connectivity

A path in a digraph is a sequence of vertices

$$v_1, v_2, \ldots, v_k$$

such that $(v_i, v_{i+1}) \in E$ for $1 \leq i \leq k-1$ without repeating any vertex

Length of the path is the number of edges

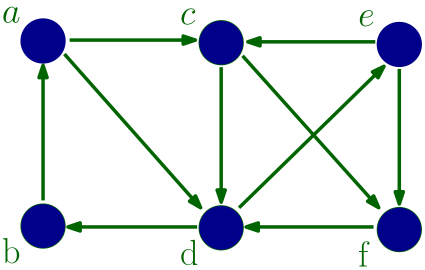
Graph Connectivity

A path in a digraph is a sequence of vertices

$$v_1, v_2, \dots, v_k$$

such that $(v_i, v_{i+1}) \in E$ for $1 \leq i \leq k-1$ without repeating any vertex

Length of the path is the number of edges



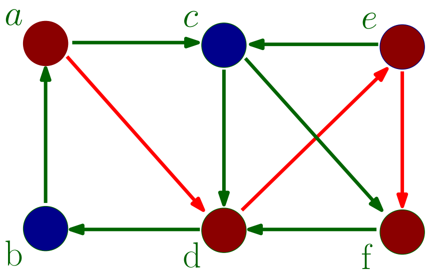
Graph Connectivity

A path in a digraph is a sequence of vertices

$$v_1, v_2, \ldots, v_k$$

such that $(v_i, v_{i+1}) \in E$ for $1 \le i \le k-1$ without repeating any vertex

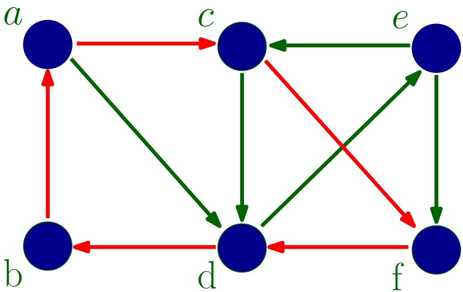
Length of the path is the number of edges



a, d, e, f

Graph Connectivity

A cycle is a path that starts and ends at the same vertex

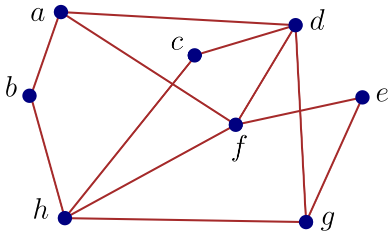


$$a, c, f, d, b, a$$

Graph Connectivity

In an undirected graph a pair of vertices u **and** v **are connected** if there is a path between u and v

- Do not mix-up this notion with that of u and v being adjacent
- If u and v are adjacent, then u is connected to v
- The converse is not necessarily true



Are c and d adjacent?

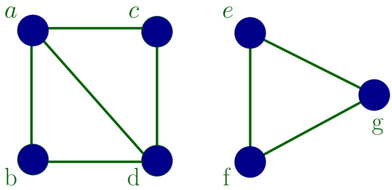
Are c and d connected?

Are a and g adjacent?

Are a and g connected?

Graph Connectivity

In an undirected graph a pair of vertices u **and** v **are connected** if there is a path between u and v



Are c and d adjacent?

Are c and d connected?

Are b and c adjacent?

Are b and c connected?

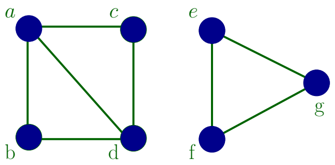
Are a and e adjacent?

Are a and e connected?

Are f and g connected?

Graph Connectivity

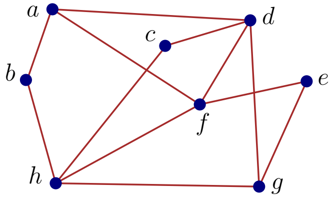
An **undirected graph is connected** if all pairs of distinct vertices are connected



Are b and c connected?

Is every pair connected?

Is the graph connected?



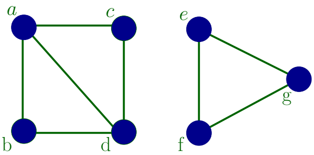
Are a and g connected?

Is every pair connected?

Is the graph connected?

Graph Connectivity

A **connected component** of G is a maximal connected subgraph (every possible connected vertex is included)



Is the graph connected?

Is the subgraph induced by $\{e, f, g\}$ connected?

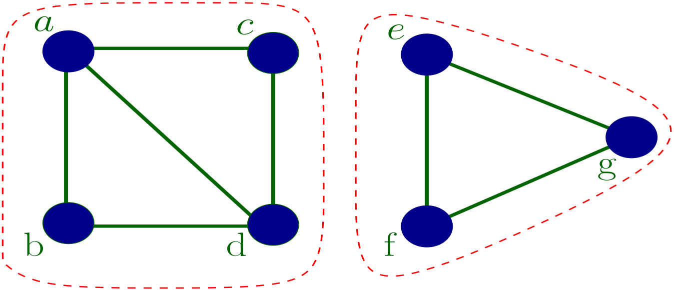
Is the subgraph induced by $\{e, f, g\}$ a connected component?

Is the subgraph induced by $\{a, b, c\}$ connected?

Is the subgraph induced by $\{a, b, c\}$ a connected component?

Graph Connectivity

A **connected component** of G is a maximal connected subgraph (every possible connected vertex is included)



Connected components of G

Undirected Graph Connectivity

In an undirected graph u **and** v **are connected** if there is a path between u and v

An **undirected graph is connected** if all pairs of distinct vertices are connected

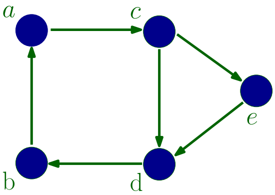
- i.e. if there is a path between every pair of distinct vertices

A **connected component** of G is a maximal connected subgraph (every possible connected vertex is included)

- A subset of vertices in which all pairs are connected and no other vertex can be added

Digraph Connectivity

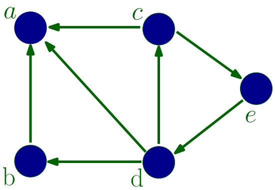
In a digraph u and v are **strongly connected**, if there is a path from u to v AND a path from v to u



Is there a path from a to e?

Is there a path from e to a?

Are a and e strongly connected?



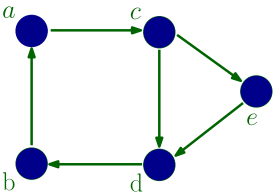
Is there a path from a to e?

Is there a path from e to a?

Are a and e strongly connected?

Digraph Connectivity

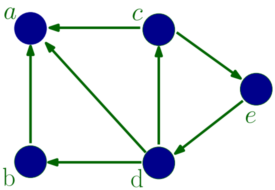
A **digraph is strongly connected**, if every pair of distinct vertices are strongly connected



Are c and d strongly connected?

Are all pairs strongly connected?

Is the graph strongly connected?



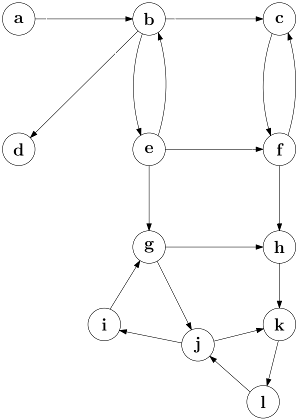
Are c and d strongly connected?

Are all pairs strongly connected?

Is the graph strongly connected?

Digraph Connectivity

A **strongly connected component** in a digraph is a maximal strongly connected subgraph (every possible strongly connected vertex is included)



Are a and b strongly connected?

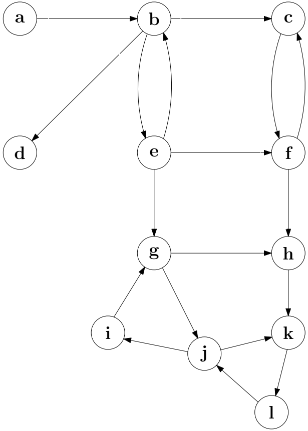
Are b and e strongly connected?

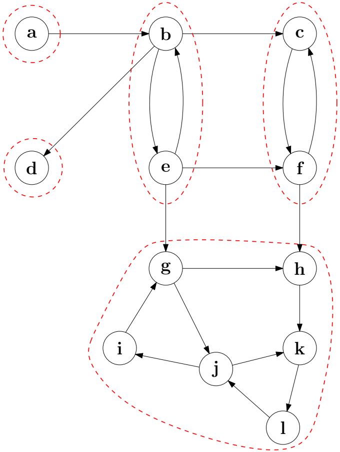
Is the subgraph induced by $\{j, k, l\}$ strongly connected?

Is the subgraph induced by $\{j, k, l\}$ a strongly connected component?

Digraph Connectivity

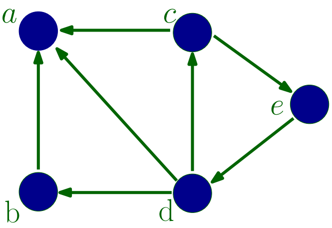
A **strongly connected component** in a digraph is a maximal strongly connected subgraph (every possible strongly connected vertex is included)

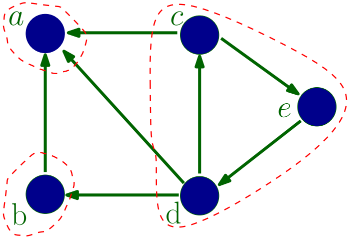




Digraph Connectivity

A **strongly connected component** in a digraph is a maximal strongly connected subgraph (every possible strongly connected vertex is included)





Directed Graph Connectivity

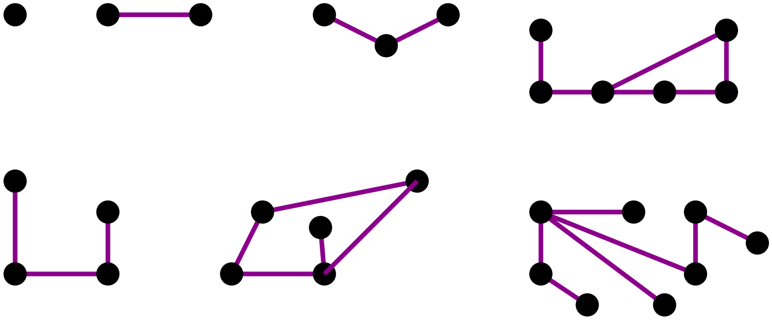
In a digraph u and v are **strongly connected**, if there is a path from u to v AND a path from v to u

A **digraph is strongly connected**, if every pair of distinct vertices are strongly connected

A **strongly connected component** in a digraph is a maximal strongly connected subgraph (every possible strongly connected vertex is included)

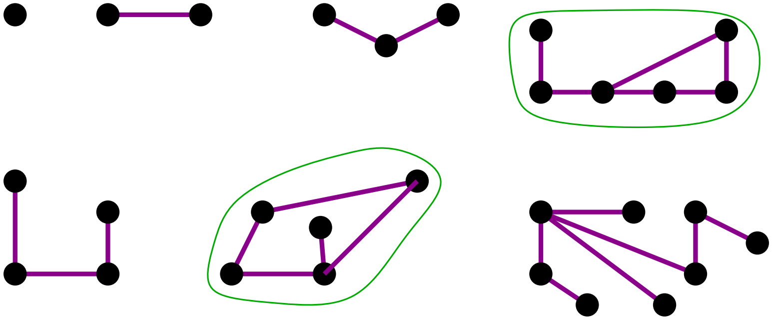
A **tree** is a connected graph with no cycles

Which ones of the following are trees?



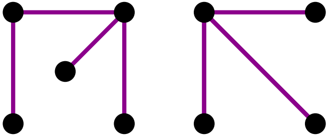
A tree is a connected graph with no cycles

Which ones of the following are trees?



Special Graphs : Forest

A forest is a ~~connected~~ graph with no cycles



Each connected component of a forest is a tree

Characterizations of Trees

A tree is a connected graph with no cycles

Theorem

A graph is a tree if and only if there is a unique path between any two vertices

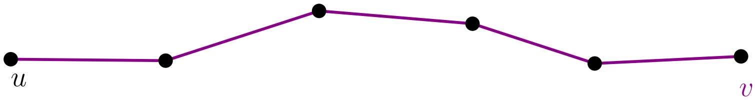
Characterizations of Trees

Theorem

A graph is a tree if and only if there is a unique path between any two vertices

Proof: tree $\implies$ unique path

Let u and v have two "different" paths b/w them



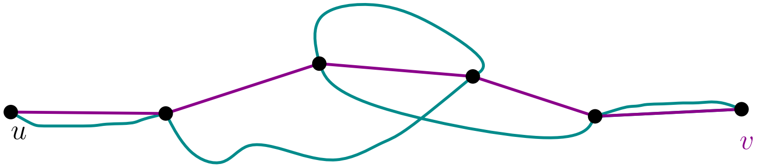
Characterizations of Trees

Theorem

A graph is a tree if and only if there is a unique path between any two vertices

Proof: tree $\implies$ unique path

Let u and v have two "different" paths b/w them



This creates a cycle

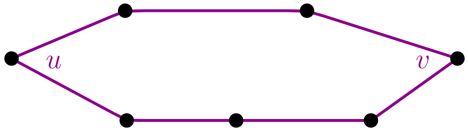
Characterizations of Trees

Theorem

A graph is a tree if and only if there is a unique path between any two vertices

Proof: unique path $\implies$ tree

The graph is connected, as (unique) paths exist. It has no cycles. If cycle exists, then



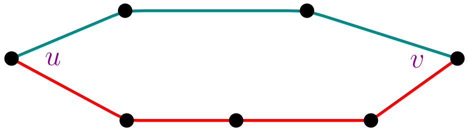
Characterizations of Trees

Theorem

A graph is a tree if and only if there is a unique path between any two vertices

Proof: unique path $\implies$ tree

The graph is connected, as (unique) paths exist. It has no cycles. If cycle exists, then



we get at least two paths

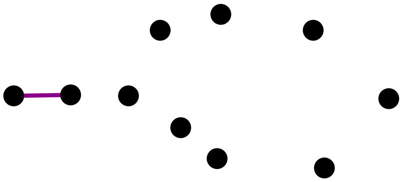
Characterizations of Trees

A leaf in a graph is a vertex with degree 1

Theorem

Any tree has at least one leaf

Proof If no leaf, then every vertex has degree ≥ 2
Start a walk from any vertex. In each step take an unvisited edge.



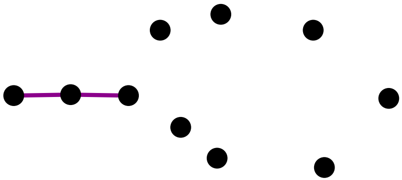
A leaf in a graph is a vertex with degree 1

Theorem

Any tree has at least one leaf

Proof If no leaf, then every vertex has degree ≥ 2

Start a walk from any vertex. In each step take an unnvisited edge.



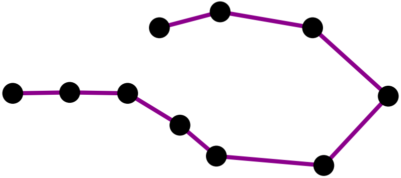
Characterizations of Trees

A leaf in a graph is a vertex with degree 1

Theorem

Any tree has at least one leaf

Proof If there is no leaf, then $\forall\ v \in V\ \ deg(v) \geq 2$
Start a walk from any vertex. In each step take an unvisited edge



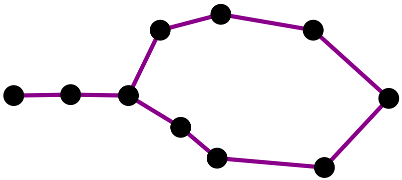
Characterizations of Trees

A leaf in a graph is a vertex with degree 1

Theorem

Any tree has at least one leaf

Proof If no leaf, then every vertex has degree ≥ 2
Start a walk from any vertex. In each step take an unvisited edge.



cannot get stuck unless a cycle exists

Characterizations of Trees

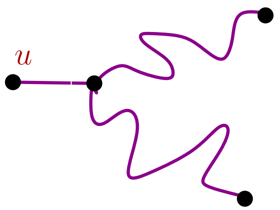
A leaf in a graph is a vertex with degree 1

Any tree has at least one leaf

Theorem

Any tree on n vertices has $n-1$ edges

Inductive Proof: Remove a leaf u



A leaf in a graph is a vertex with degree 1

Any tree has at least one leaf

Theorem

Any tree on n vertices has $n-1$ edges

Inductive Proof: Remove a leaf u

$T-u$ is a tree

$T-u$ has $n-2$ edges

So, T has $n-1$ edges