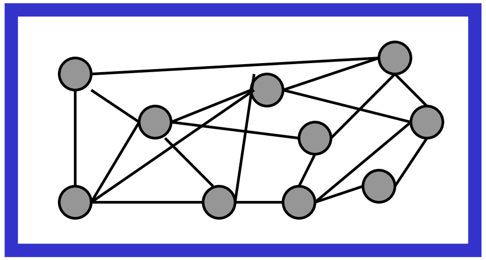
Chapter 9: Graph Theory

§9.1: Graphs and Graph Models

- Correspond to symmetric binary relations *R*.
- A *simple graph* *G*=(*V*,*E*) consists of:
  - a set *V* of *vertices* or *nodes* (*V* corresponds to the universe of the relation *R*),
  - a set *E* of *edges* / *arcs* / *links*: unordered pairs of [distinct?] elements $u,v \in V$, such that uRv.

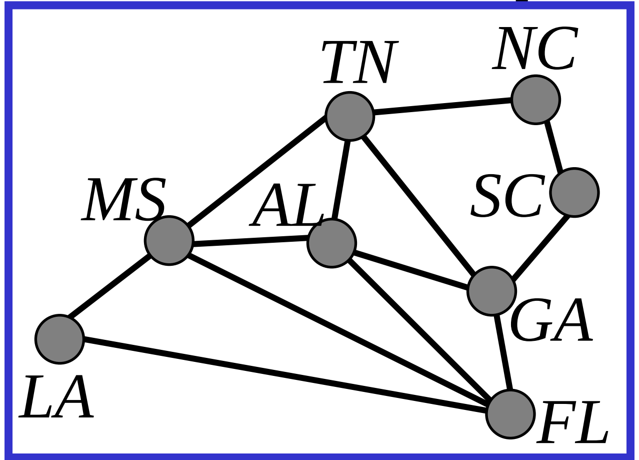


Visual Representation of a Simple Graph

Example of a *Simple Graph*

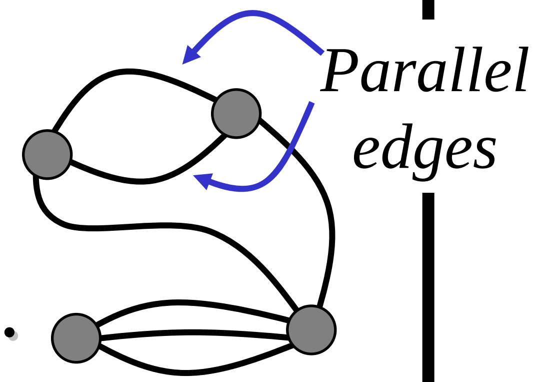
- Let V be the set of states in the far-southeastern U.S.:
  - V={FL, GA, AL, MS, LA, SC, TN, NC}
- Let E={{u,v}|u adjoins v}

  ={{FL,GA},{FL,AL},{FL,MS},
  {FL,LA},{GA,AL},{AL,MS},
  {MS,LA},{GA,SC},{GA,TN},
  {SC,NC},{NC,TN},{MS,TN},
  {MS,AL}}



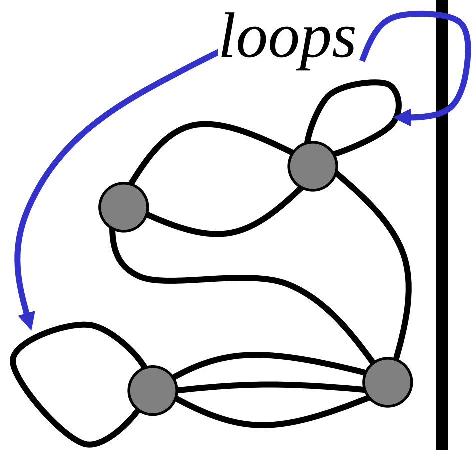
Multigraphs

- Like simple graphs, but there may be *more than one* edge connecting two given nodes.
- A *multigraph* $G=(V, E, f)$ consists of a set V of vertices, a set E of edges (as primitive objects), and a function $f:E\rightarrow\{\{u,v\}|u,v\in V \wedge u\neq v\}$.
- E.g., nodes are cities, edges are segments of major highways.



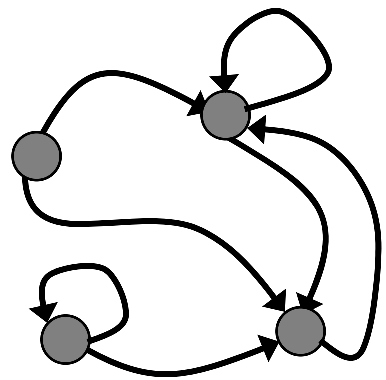
Pseudographs

- Like a multigraph, but edges connecting a node to itself are allowed.
- A *pseudograph* $G=(V, E, f)$ where $f:E\rightarrow\{\{u,v\}|u,v\in V\}$. Edge $e\in E$ is a *loop* if $f(e)=\{u,u\}=\{u\}$.
- *E.g.*, nodes are campsites in a state park, edges are hiking trails through the woods.



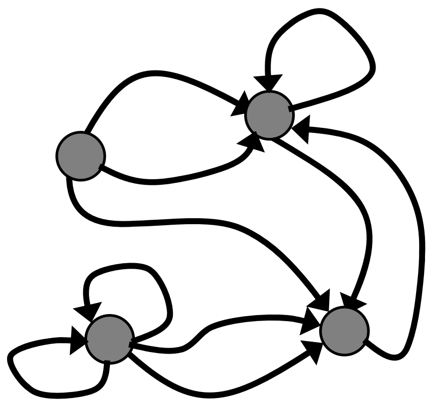
Directed Graphs

- Correspond to arbitrary binary relations *R*, which need not be symmetric.
- A ***directed graph*** (*V*,*E*) consists of a set of vertices *V* and a binary relation *E* on *V*.
- *E.g.*: *V* = people, $E=\{(x,y) \mid x \text{ loves } y\}$
- $e=(u,v)$,$u,v\in V$



Directed Multigraphs

- Like directed graphs, but there may be more than one arc from a node to another.
- A *directed multigraph* $G=(V, E, f)$ consists of a set V of vertices, a set E of edges, and a function $f{:}E\rightarrow V\times V$.
- E.g., V=web pages, E=hyperlinks. *The WWW is a directed multigraph...*



Types of Graphs: Summary

- Summary of the book's definitions.
- Keep in mind this terminology is not fully standardized...

Term	Edge type	Multiple edges ok?	Self-loops ok?
Simple graph	Undir.	No	No
Multigraph	Undir.	Yes	No
Pseudograph	Undir.	Yes	Yes
Directed graph	Directed	No	Yes
Directed multigraph	Directed	Yes	Yes

§9.2: Graph Terminology

- *Adjacent*, *connects*, *endpoints*, *degree*, *initial*, *terminal*, *in-degree*, *out-degree*, *complete*, *cycles*, *wheels*, *n-cubes*, *bipartite*, *subgraph*, *union*.

Adjacency

Let G be an undirected graph with edge set E. Let $e \in E$ be (or map to) the pair $\{u,v\}$. Then we say:

- u, v are *adjacent* / *neighbors* / *connected*.
- Edge e is *incident with* vertices u and v.
- Edge e *connects* u and v.
- Vertices u and v are *endpoints* of edge e.

Degree of a Vertex

- Let G be an undirected graph, $v \in V$ a vertex.
- The *degree* of v, $\deg(v)$, is its number of incident edges. (Except that any self-loops are counted twice.)
- A vertex with degree 0 is *isolated*.
- A vertex of degree 1 is *pendant*.

Handshaking Theorem

- Let G be an undirected (simple, multi-, or pseudo-) graph with vertex set V and edge set E. Then

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

- Corollary: Any undirected graph has an even number of vertices of odd degree.

Directed Adjacency

- Let *G* be a directed (possibly multi-) graph, and let *e* be an edge of *G* that is (or maps to) (*u*,*v*). Then we say:
  - *u* is *adjacent to v*, *v* is *adjacent from u*
  - *e comes from u*, *e goes to v*.
  - *e connects u to v*, *e goes from u to v*
  - the *initial vertex* of *e* is *u*
  - the *terminal vertex* of *e* is *v*

Directed Degree

- Let *G* be a directed graph, *v* a vertex of *G*.
  - The *in-degree* of *v*, $\deg^-(v)$, is the number of edges going to *v*.
  - The *out-degree* of *v*, $\deg^+(v)$, is the number of edges coming from *v*.
  - The *degree* of *v*, $\deg(v) \equiv \deg^-(v) + \deg^+(v)$, is the sum of *v*'s in-degree and out-degree.

Directed Handshaking Theorem

- Let G be a directed (possibly multi-) graph with vertex set V and edge set E. Then:

$$\sum_{v\in V} \deg^{-}(v) = \sum_{v\in V} \deg^{+}(v) = \frac{1}{2}\sum_{v\in V} \deg(v) = |E|$$

- Note that the degree of a node is unchanged by whether we consider its edges to be directed or undirected.

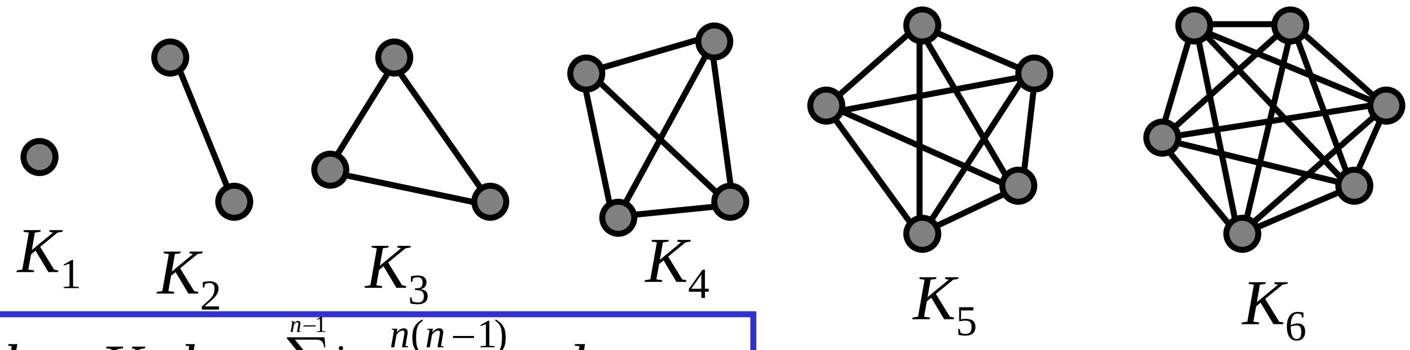
Special Graph Structures

Special cases of undirected graph structures:

- Complete graphs K_n
- Cycles C_n
- Wheels W_n
- n-Cubes Q_n
- Bipartite graphs
- Complete bipartite graphs $K_{m,n}$

Complete Graphs

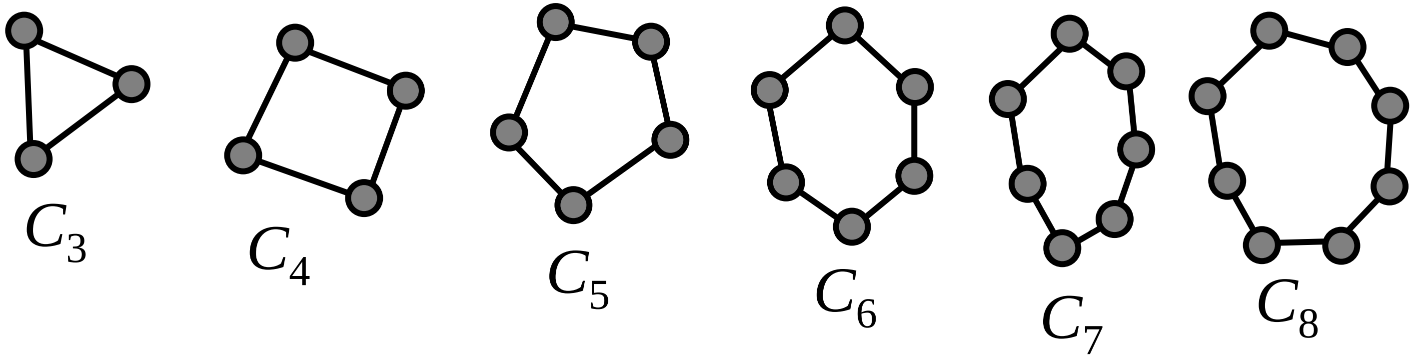
- For any $n \in \mathbf{N}$, a *complete graph* on n vertices, K_n, is a simple graph with n nodes in which every node is adjacent to every other node: $\forall u,v \in V$: $u \neq v \leftrightarrow \{u,v\} \in E$.



Note that K_n has $\sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}$ edges.

Cycles

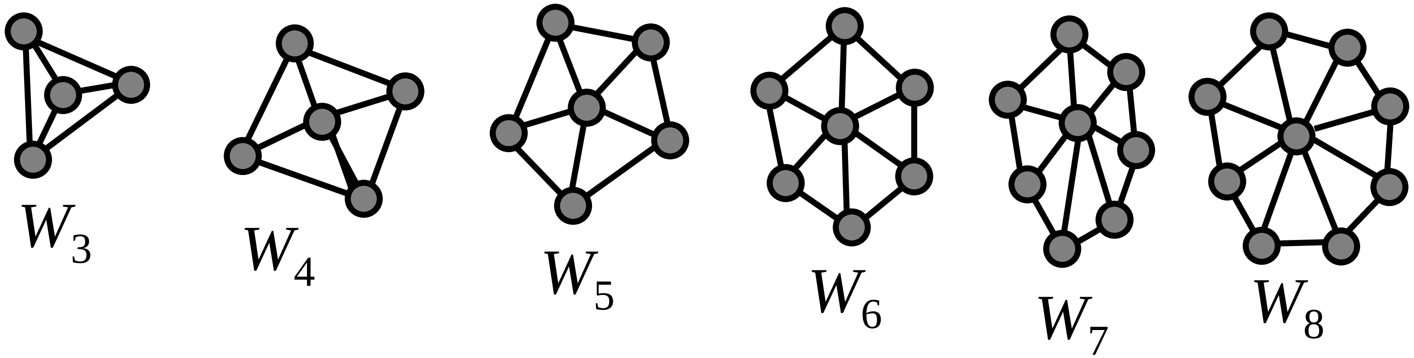
- For any $n \geq 3$, a *cycle* on n vertices, C_n, is a simple graph where $V=\{v_1,v_2,\ldots,v_n\}$ and $E=\{\{v_1,v_2\},\{v_2,v_3\},\ldots,\{v_{n-1},v_n\},\{v_n,v_1\}\}$.



How many edges are there in C_n?

Wheels

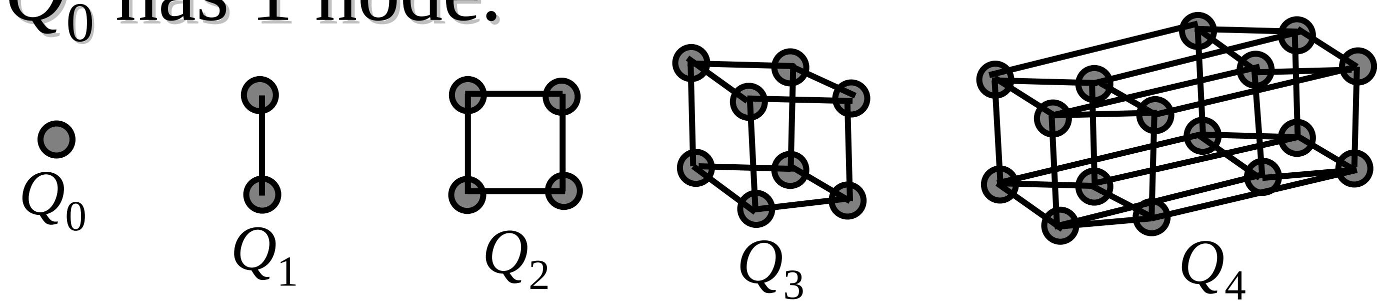
- For any $n \geq 3$, a *wheel* W_n, is a simple graph obtained by taking the cycle C_n and adding one extra vertex v_{hub} and n extra edges $\{\{v_{\text{hub}},v_1\}, \{v_{\text{hub}},v_2\},\ldots,\{v_{\text{hub}},v_n\}\}$.



How many edges are there in W_n?

n-cubes (hypercubes)

- For any $n \in \mathbf{N}$, the hypercube Q_n is a simple graph consisting of two copies of Q_{n-1} connected together at corresponding nodes. Q_0 has 1 node.



Number of vertices: 2^n. *Number of edges:Exercise to try!*

n-cubes (hypercubes)

- For any $n \in \mathbf{N}$, the hypercube Q_n can be defined recursively as follows:
  - $Q_0 = \{\{v_0\}, \varnothing\}$ (one node and no edges)
  - For any $n \in \mathbf{N}$, if $Q_n = (V, E)$, where $V = \{v_1, \ldots, v_a\}$ and $E = \{e_1, \ldots, e_b\}$, then $Q_{n+1} = (V \cup \{v_1', \ldots, v_a'\}, E \cup \{e_1', \ldots, e_b'\} \cup \{\{v_1, v_1'\}, \{v_2, v_2'\}, \ldots, \{v_a, v_a'\}\})$ where $v_1', \ldots, v_a'$ are new vertices, and where if $e_i = \{v_j, v_k\}$ then $e_i' = \{v_j', v_k'\}$.

Bipartite Graphs

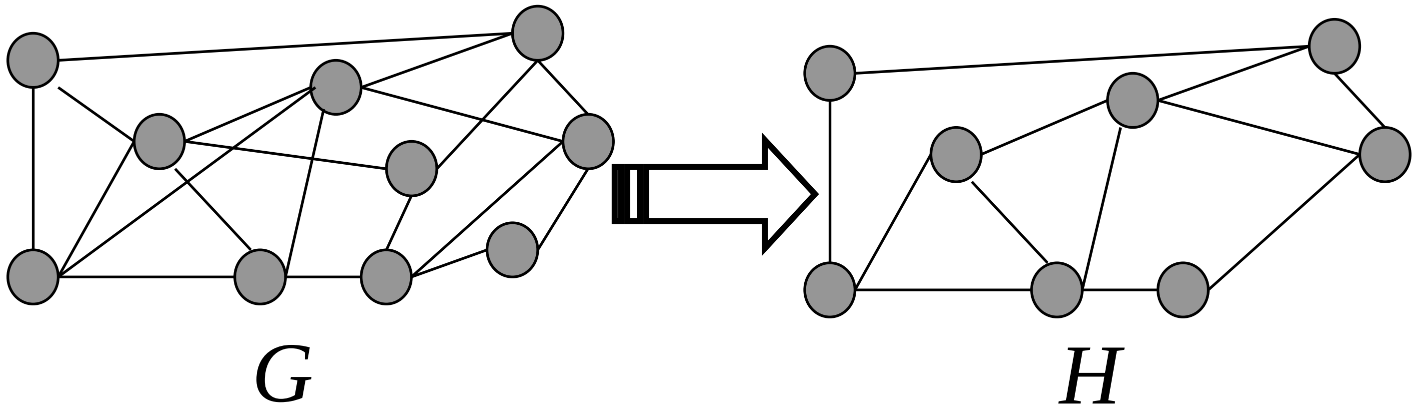
- A *simple graph* *G* is called ***bipartite*** if its vertex set *V* can be partitioned into two disjoint sets V_1 and V_2 such that every edge in the graph connects a vertex in V_1 and a vertex in V_2(so that no edge in *G* connects either two vertices in V_1 or two vertices in V_2)

Complete Bipartite Graphs

- The *complete bipartite graph* $K_{m,n}$ that has its vertex set partitioned into two subsets of m and n vertices, respectively. There is an edge between two vertices *iff* one vertex is in the first subset and the other is in the second subset.

Subgraphs

- A subgraph of a graph $G=(V,E)$ is a graph $H=(W,F)$ where $W \subseteq V$ and $F \subseteq E$.



Graph Unions

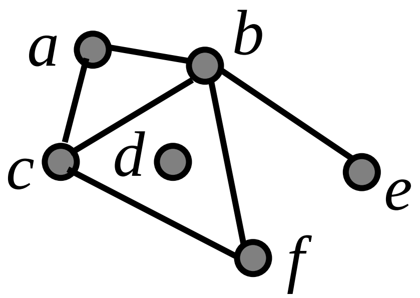
- The *union* $G_1 \cup G_2$ of two simple graphs $G_1=(V_1, E_1)$ and $G_2=(V_2,E_2)$ is the simple graph $(V_1 \cup V_2, E_1 \cup E_2)$.

§9.3: Graph Representations & Isomorphism

- Graph representations:
  - Adjacency lists.
  - Adjacency matrices.
  - Incidence matrices.
- Graph isomorphism:
  - Two graphs are isomorphic iff they are identical except for their node names.

Adjacency Lists

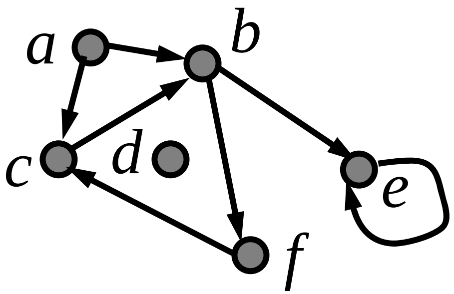
- A table with 1 row per vertex, listing its adjacent vertices.



Vertex	*Adjacent Vertices*
a	
b	
c	
d	
e	
f	

Directed Adjacency Lists

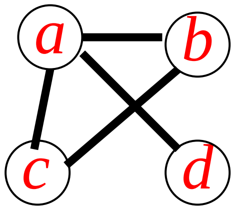
- 1 row per node, listing the terminal nodes of each edge incident from that node.



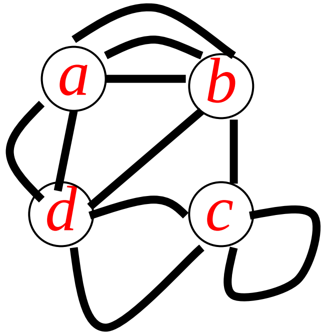
Initial Vertex	*Terminal Vertices*
a	
b	
c	
d	
e	
f	

Adjacency Matrices

- Matrix $\boldsymbol{A}=[a_{ij}]$, where a_{ij} is 1 if $\{v_i, v_j\}$ is an edge of G, 0 otherwise.



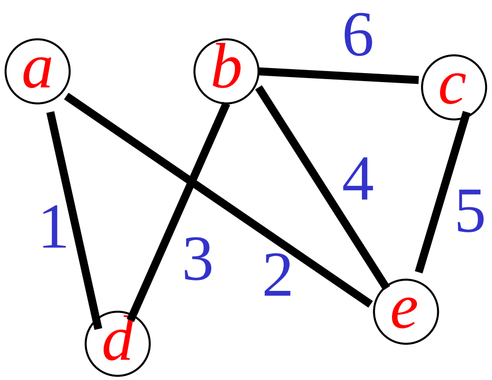
$$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$



$$\begin{bmatrix} 0 & 3 & 0 & 2 \\ 3 & 0 & 1 & 1 \\ 0 & 1 & 1 & 2 \\ 2 & 1 & 2 & 0 \end{bmatrix}$$

Incidence Matrices

- Matrix $\boldsymbol{A}=[a_{ij}]_{v\times e}$, where a_{ij} is 1 if e_j incident with v_i, 0 otherwise.



$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

Graph Isomorphism

- Formal definition:
  - Simple graphs $G_1=(V_1, E_1)$ and $G_2=(V_2, E_2)$ are *isomorphic* iff $\exists$ a bijection $f:V_1 \rightarrow V_2$ such that $\forall\ a,b \in V_1$, a and b are adjacent in G_1 iff $f(a)$ and $f(b)$ are adjacent in G_2.
  - f is the "renaming" function that makes the two graphs identical.
  - Definition can easily be extended to other types of graphs.

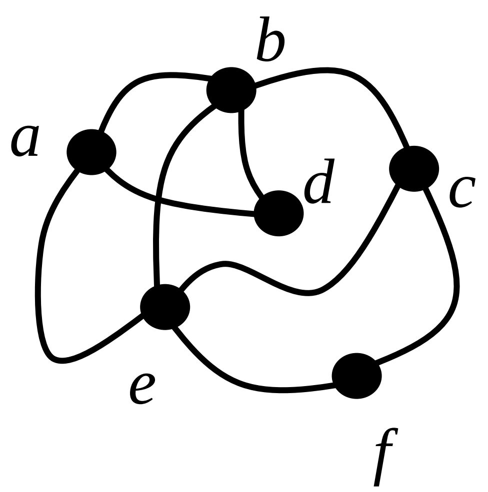
Graph Invariants under Isomorphism

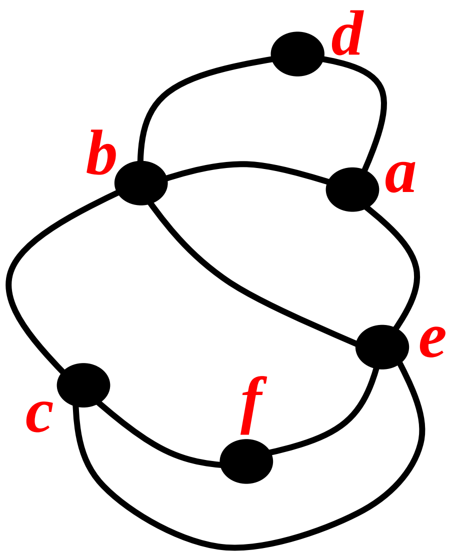
Necessary but not *sufficient* conditions for $G_1=(V_1, E_1)$ to be isomorphic to $G_2=(V_2, E_2)$:

- $|V1|=|V2|$, $|E1|=|E2|$.
- The number of vertices with degree *n* is the same in both graphs.
- For every proper subgraph *g* of one graph, there is a proper subgraph of the other graph that is isomorphic to *g*.

Isomorphism Example

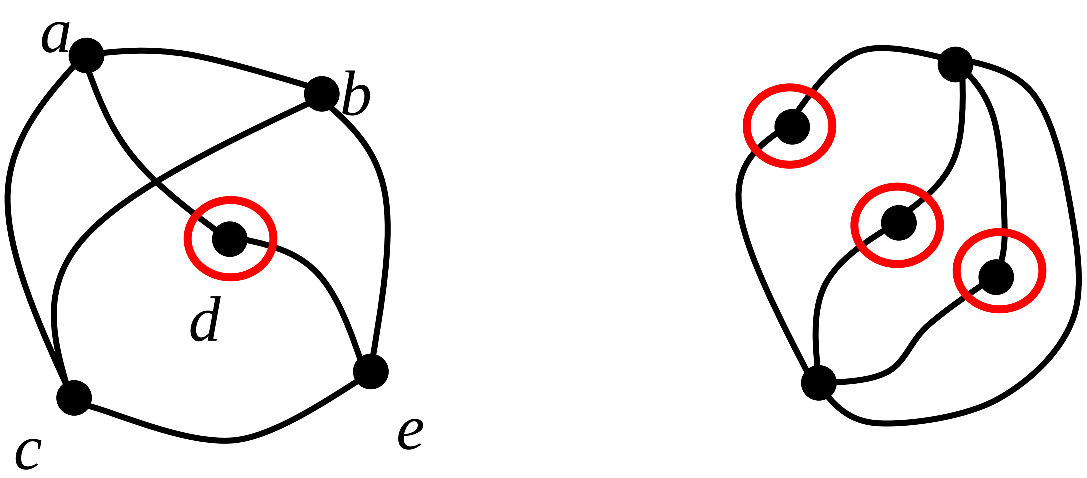
- If isomorphic, label the 2nd graph to show the isomorphism, else identify difference.





Are These Isomorphic?

- If isomorphic, label the 2nd graph to show the isomorphism, else identify difference.



** Same # of vertices*
** Same # of edges*
** Different # of verts of degree 2! (1 vs 3)*

§9.4: Connectivity

- In an undirected graph, a *path of length n from u to v* is a sequence of adjacent edges going from vertex *u* to vertex *v*.
- A path is a *circuit* if *u*=*v*.
- A path *traverses* the vertices along it.
- A path is *simple* if it contains no edge more than once.

Paths in Directed Graphs

- Same as in undirected graphs, but the path must go in the direction of the arrows.

Connectedness

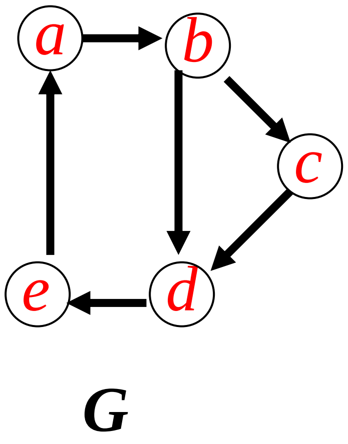
- An undirected graph is *connected* iff there is a path between every pair of distinct vertices in the graph.
- Theorem: There is a *simple* path between any pair of distinct vertices in a connected undirected graph.
- *Connected component*: connected subgraph
- A *cut vertex* or *cut edge* separates 1 connected component into 2 if removed.

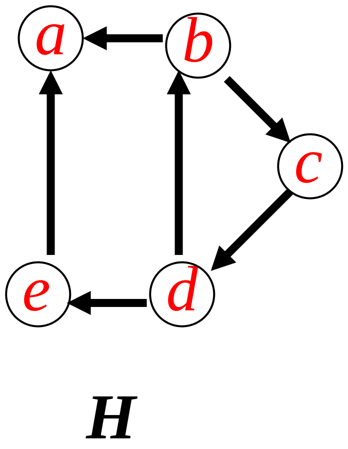
Directed Connectedness

- A directed graph is *strongly connected* iff there is a directed path from *a* to *b* and from *b* to *a* for any two vertices *a* and *b*.
- It is *weakly connected* iff the underlying *undirected* graph (*i.e.*, with edge directions removed) is connected.
- Note *strongly* implies *weakly* but not vice-versa.

Example

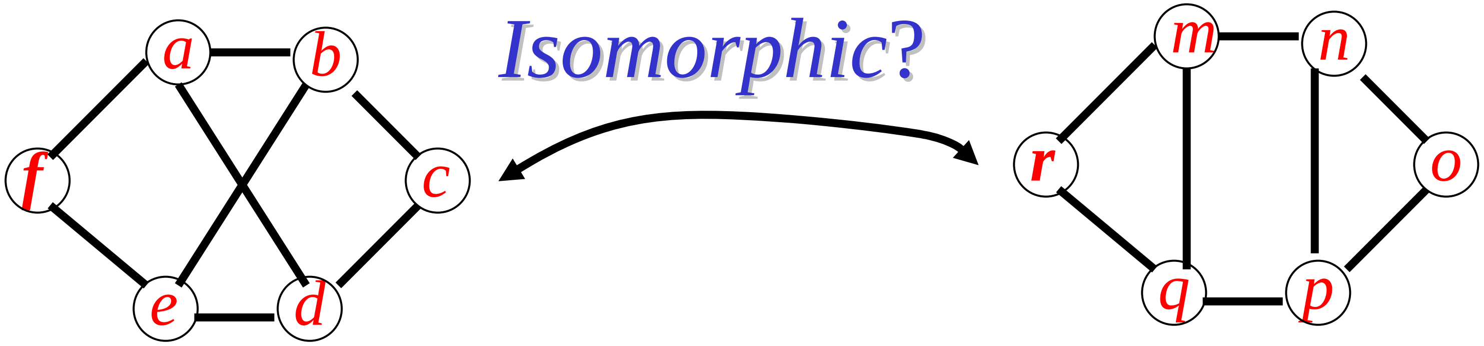
- Are the graphs *G* and *H* *strongly connected*? Are they *weakly connected*?





Paths & Isomorphism

- Note that connectedness, and the existence of a circuit or simple circuit of length *k* are graph invariants with respect to isomorphism.

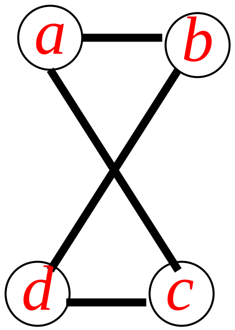


Counting Paths w Adjacency Matrices

- Let $\boldsymbol{A}$ be the adjacency matrix of graph G.
- The number of paths of length r from v_i to v_j is equal to $(\boldsymbol{A}^r)_{i,j}$. (The notation $(\boldsymbol{M})_{i,j}$ denotes $m_{i,j}$ where $[m_{i,j}] = \boldsymbol{M}$.)

Example

- How many paths of length 4 from *a* to *d*?



$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \qquad A^4 =$$

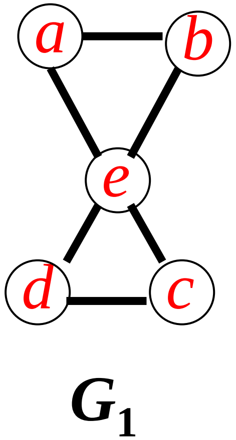
§9.5: Euler & Hamilton Paths

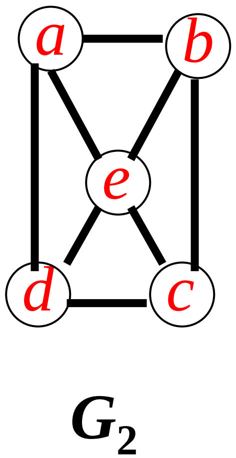
- An ***Euler circuit*** in a graph *G* is a simple circuit containing every edge of *G*.
- An *Euler path* in *G* is a simple path containing every edge of *G*.
- A ***Hamilton circuit*** is a circuit that traverses each vertex in *G* exactly once.
- A *Hamilton path* is a path that traverses each vertex in G exactly once.

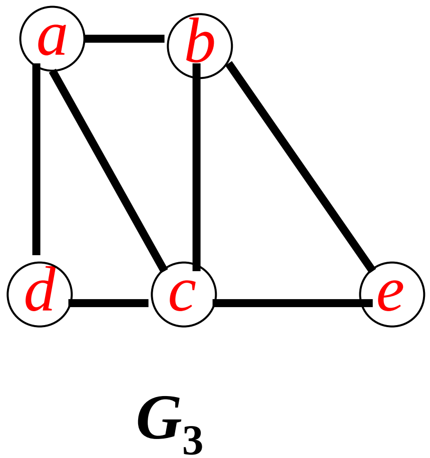
Example

- Which has an Euler circuit?

  Of those that do not, which has an Euler path ?

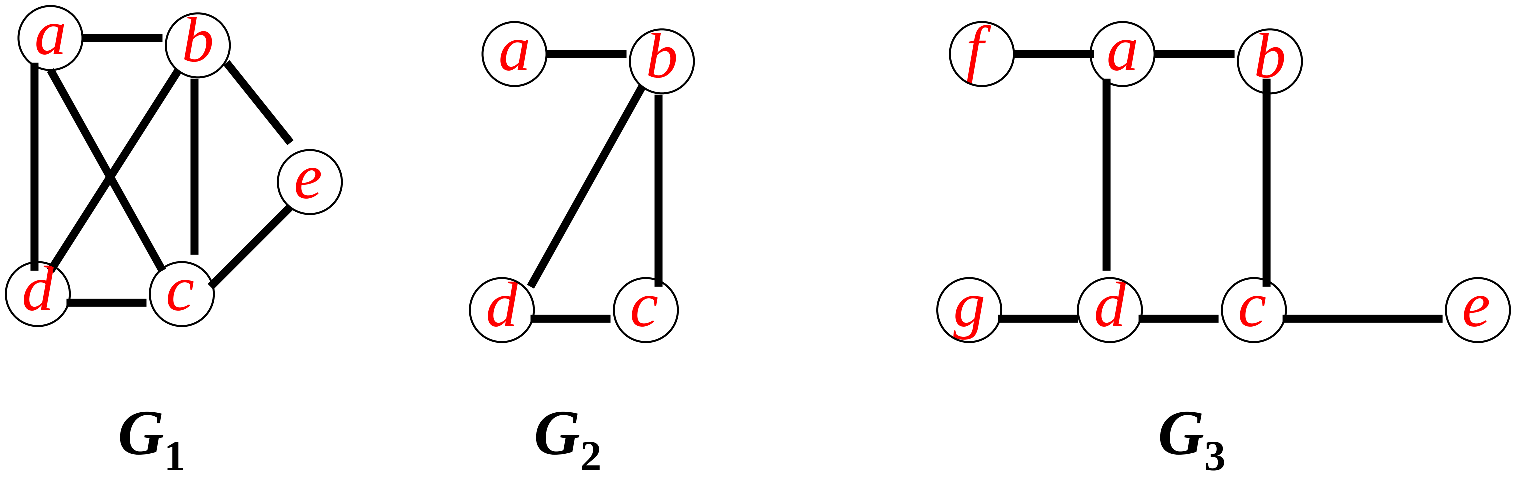






Example

- Which has an Hamilton circuit or, if not, a Hamilton path?



Some Useful Theorems

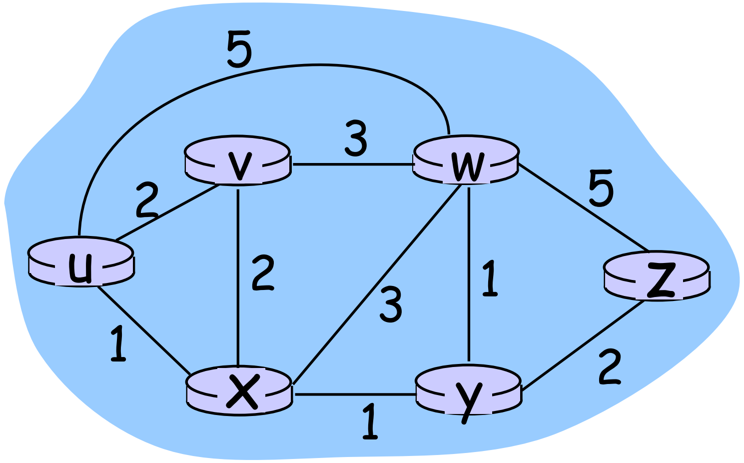
- A connected multigraph has an Euler circuit iff each vertex has even degree.
- A connected multigraph has an Euler path (but not an Euler circuit) iff it has exactly 2 vertices of odd degree.
- If (but not only if) G is connected, simple, has $n \geq 3$ vertices, and $\forall v \deg(v) \geq n/2$, then G has a Hamilton circuit.

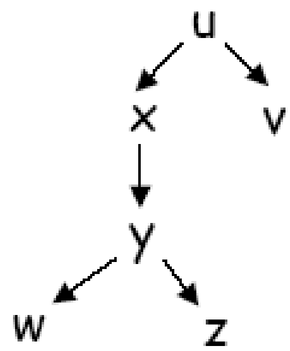
§9.6: Shortest Path Algorithm: Dijsktra's Algorithm

```
Initialization:
  N' = {u}
  for all nodes v
    if v adjacent to u
        then D(v) = c(u,v)
    else D(v) = ∞

Loop
   find w not in N' such that D(w) is a minimum
    add w to N'
    update D(v) for all v adjacent to w and not in N' :
       D(v) = min( D(v), D(w) + c(w,v) )
    /* new cost to v is either old cost to v or known
     shortest path cost to w plus cost from w to v */
until all nodes in N'
```

Dijkstra's Algorithm





Shortest path tree

Example

Find the shortest path from A to H.