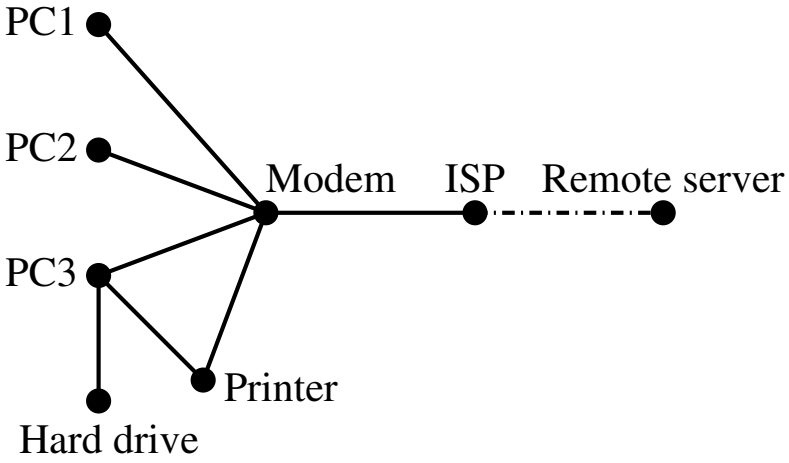
Chapter 9. Graph Theory

Prof. Tesler

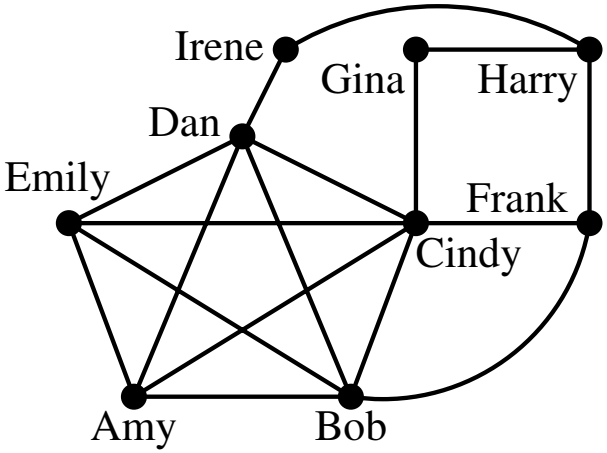
Math 184A
Winter 2019

Graphs

Computer network



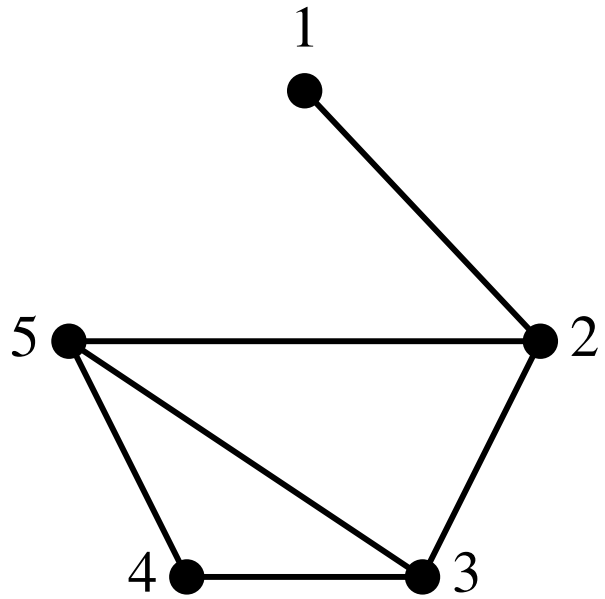
Friends



We have a network of items and connections between them. Examples:

- Telephone networks, computer networks
- Transportation networks (bus/subway/train/plane)
- Social networks
- Family trees, evolutionary trees
- Molecular graphs (atoms and chemical bonds)
- Various data structures in Computer Science

Graphs



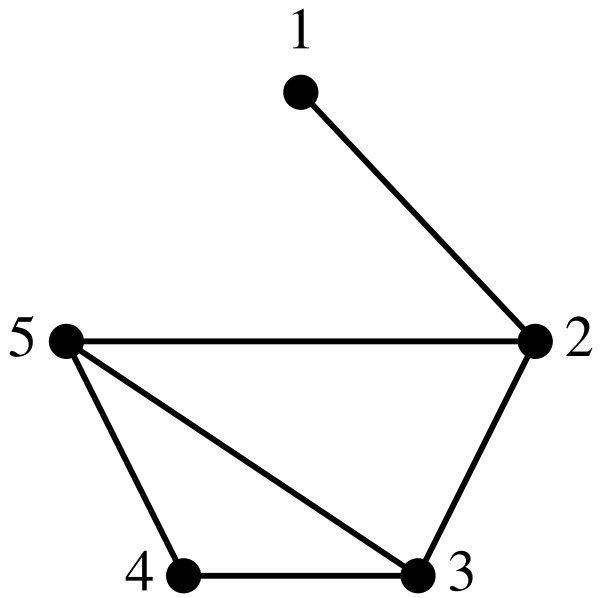
- The dots are called *vertices* or *nodes* (singular: vertex, node)

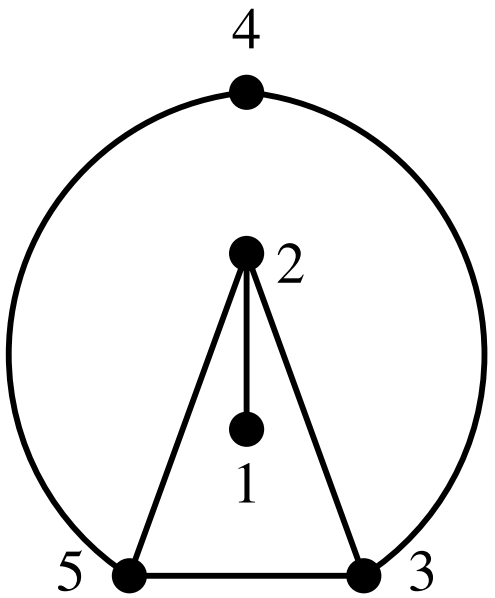
$$V = \text{set of vertices} = \{1, 2, 3, 4, 5\}$$

- The connections between vertices are called *edges*.
- Represent an edge as a set $\{i, j\}$ of two vertices.
  E.g., the edge between 2 and 5 is $\{2, 5\} = \{5, 2\}$.

$$E = \text{set of edges} = \{\{1, 2\}, \{2, 3\}, \{2, 5\}, \{3, 4\}, \{3, 5\}, \{4, 5\}\}$$

Simple graphs

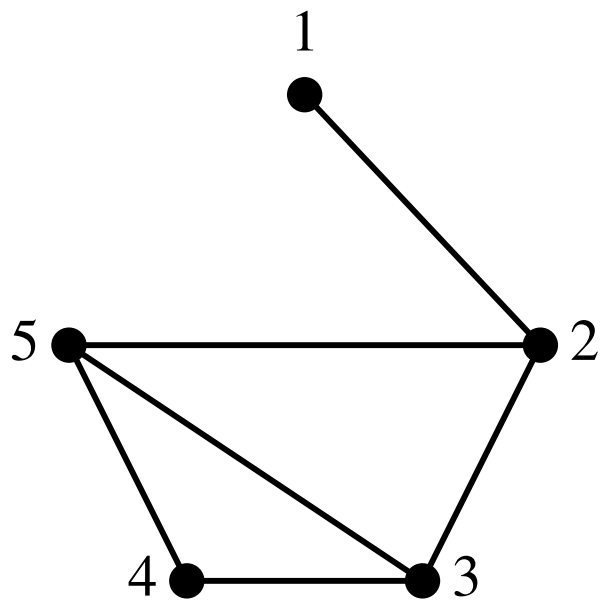




A *simple graph* is $G = (V, E)$:

- V is the set of vertices.
  It can be any set; $\{1, \ldots, n\}$ is just an example.
- E is the set of edges, of form $\{u, v\}$, where $u, v \in V$ and $u \neq v$.
  Every pair of vertices has either 0 or 1 edges between them.
- The drawings above represent the same abstract graph since they have the same V and E, even though the drawings look different.

Degrees



The *degree* of a vertex is the number of edges on it.

$$d(1) = 1 \qquad d(2) = 3 \qquad d(3) = 3 \qquad d(4) = 2 \qquad d(5) = 3$$

Sum of degrees $= 1 + 3 + 3 + 2 + 3 = 12$
Number of edges $= 6$

Sum of degrees

Theorem

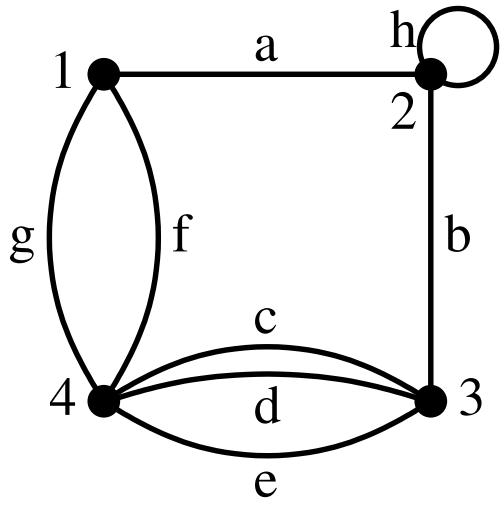
The sum of degrees of all vertices is twice the number of edges:

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

Proof.

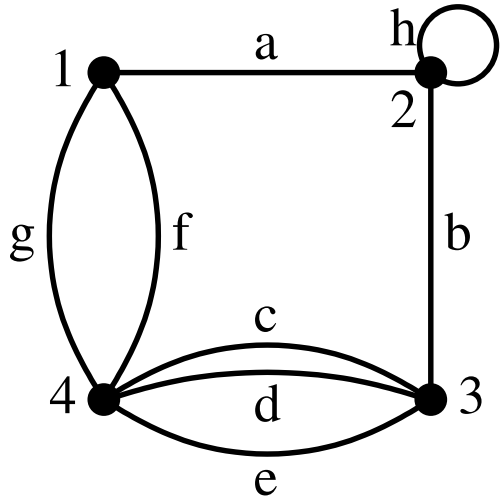
- Let $S = \{\,(v, e) \;:\; v \in V, \;\; e \in E, \;\; \text{vertex } v \text{ is in edge } e\,\}$
- **Count $|S|$ by vertices:** Each vertex v is contained in $d(v)$ edges, so
$$|S| = \sum_{v \in V} d(v).$$
- **Count $|S|$ by edges:** Each edge has two vertices, so
$$|S| = \sum_{e \in E} 2 = 2\,|E|\,.$$

Multigraphs



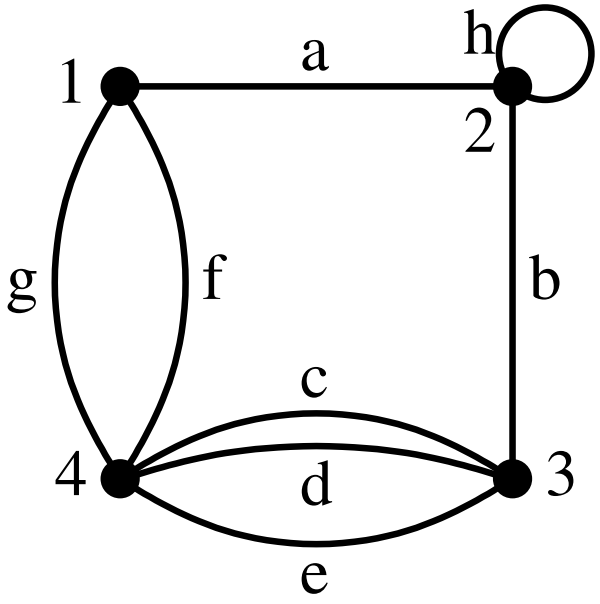
- Some networks have *multiple edges* between two vertices. Notation $\{3, 4\}$ is ambiguous, so write labels on the edges: c, d, e.
- There can be an edge from a vertex to itself, called a *loop* (such as h above). A loop has one vertex, so $\{2, 2\} = \{2\}$.
- A simple graph does not have multiple edges or loops.

Multigraphs



- Computer network with multiple connections between machines.
- Transportation network with multiple routes between stations.
- **But:** A graph of Facebook friends is a simple graph. It does not have multiple edges, since you're either friends or you're not. Also, you cannot be your own Facebook friend, so no loops.

Multigraphs



$$V = \{1, 2, 3, 4\}$$
$$E = \{a, b, c, d, e, f, g, h\}$$

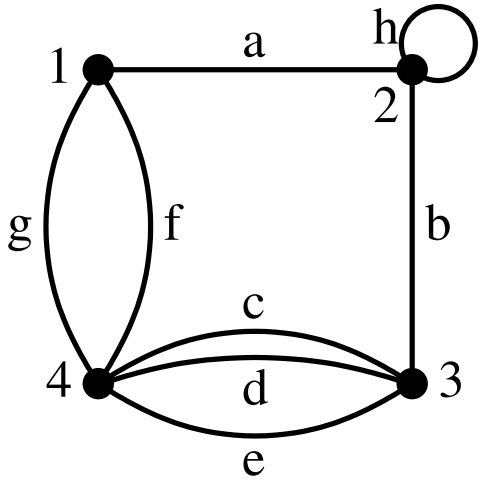
$$\begin{aligned} \phi(a) &= \{1, 2\} \\ \phi(b) &= \{2, 3\} \\ \phi(c) = \phi(d) = \phi(e) &= \{3, 4\} \\ \phi(f) = \phi(g) &= \{1, 4\} \\ \phi(h) &= \{2\} \end{aligned}$$

A *multigraph* is $G = (V, E, \phi)$, where:

- V is the set of vertices. It can be any set.
- E is the set of edge labels (with a unique label for each edge).
- $\phi : E \to \{\{u, v\} \ : \ u, v \in V\}$
  is a function from the edge labels to the pairs of vertices.
  $\phi(L) = \{u, v\}$ means the edge with label L connects u and v.

Adjacency matrix of a multigraph

- Let $n = |V|$
- The *adjacency matrix* of a multigraph is an $n \times n$ matrix $A = (a_{uv})$. Entry a_{uv} is the number of edges between vertices $u, v \in V$.



$$A = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 2 \\ 1 & 2 & 1 & 0 \\ 0 & 1 & 0 & 3 \\ 2 & 0 & 3 & 0 \end{bmatrix} \end{array}$$

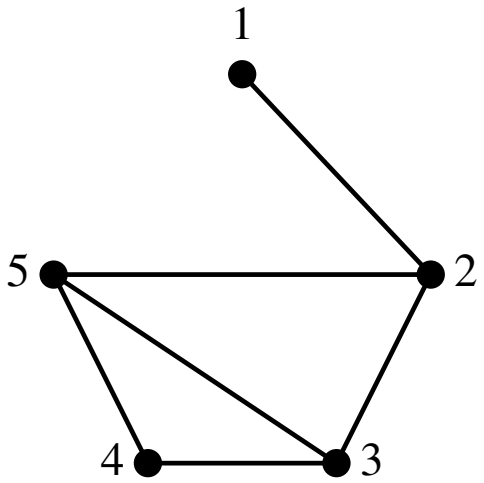
- $a_{uv} = a_{vu}$ for all vertices u, v. Thus, A is a *symmetric matrix* ($A = A^T$).
- The sum of entries in row u is the degree of u.
- **Technicality:** A loop on vertex v counts as
  - 1 edge in E,
  - degree 2 in $d(v)$ and in a_{vv} (it touches vertex v twice),

  With these rules, graphs with loops also satisfy $\sum_{v \in V} d(v) = 2\,|E|$.

Adjacency matrix of a simple graph

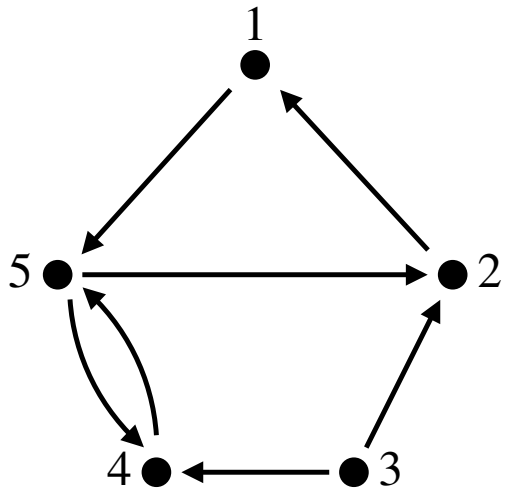
In a simple graph:

- All entries of the adjacency matrix are 0 or 1 (since there either is or is not an edge between each pair of vertices).
- The diagonal is all 0's (since there are no loops).



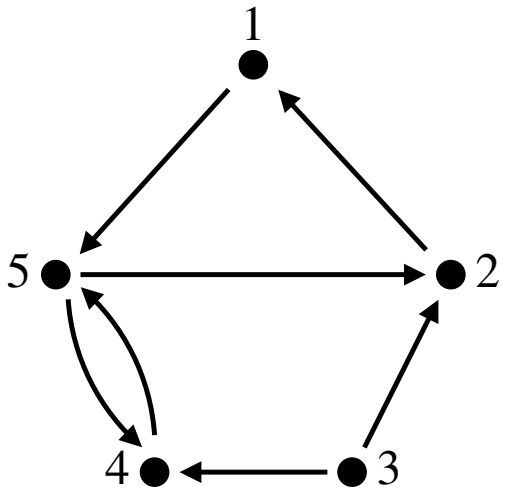
$$A = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

Directed graph (a.k.a. digraph)



- A *directed edge* is a connection with a direction.
- One-way transportation routes.
- Broadcast TV and satellite TV are one-way connections from the broadcaster to your antenna.
- Familiy tree: parent $\rightarrow$ child
- An unrequited Facebook friend request.

Directed graph (a.k.a. digraph)

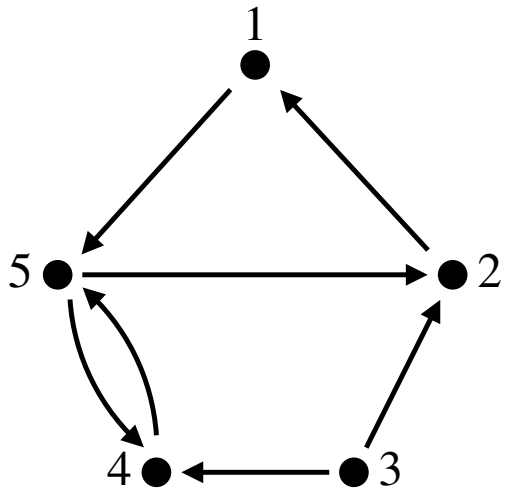


$$V = \{1, 2, 3, 4, 5\}$$

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

- Represent a directed edge $u \to v$ by an ordered pair (u, v).
  E.g., $3 \to 2$ is $(3, 2)$, but we do not have $2 \to 3$, which is $(2, 3)$.
- A directed graph is *simple* if each (u, v) occurs at most once, and there are no loops.
  - Represent it as $G = (V, E)$.
  - V is a set of vertices. It can be any set.
  - E is the set of edges. Each edge has form (u, v) with $u, v \in V$, $u \neq v$.
  - It is permissible to have both $(4, 5)$ and $(5, 4)$, since they are distinct.

Degrees in a directed graph

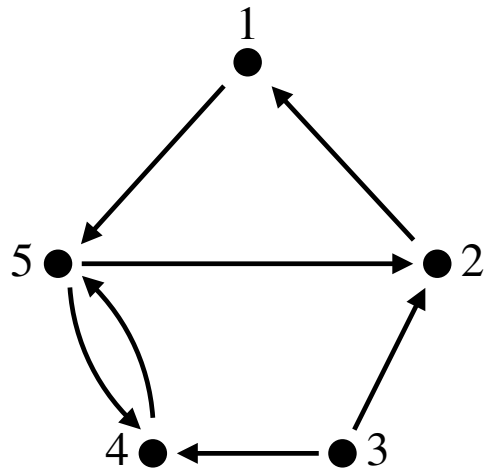


- For a vertex v, the *indegree* is the number of edges going into v, and the *outdegree* is the number of edges going out from v.

v	$\text{indegree}(v)$	$\text{outdegree}(v)$
1	1	1
2	2	1
3	0	2
4	2	1
5	2	2
Total	7	7

- The sum of indegrees is $|E|$ and the sum of outdegrees is $|E|$.

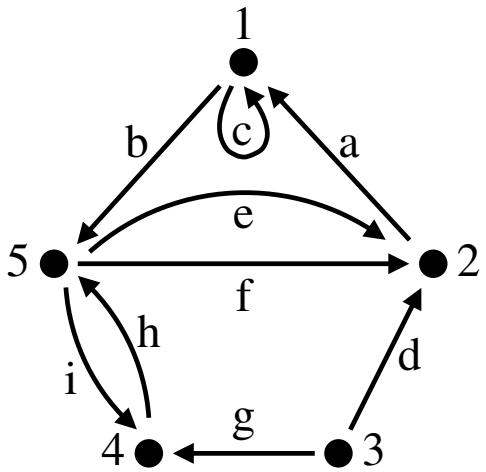
Adjacency matrix of a directed graph



$$A = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

- Let $n = |V|$
- The *adjacency matrix* of a directed graph is an $n \times n$ matrix $A = (a_{uv})$ with $u, v \in V$.
- Entry a_{uv} is the number of edges directed from u to v.
- a_{uv} and a_{vu} are not necessarily equal, so A is usually not symmetric.
- The sum of entries in row u is the outdegree of u.
  The sum of entries in column v is the indegree of v.

Directed multigraph



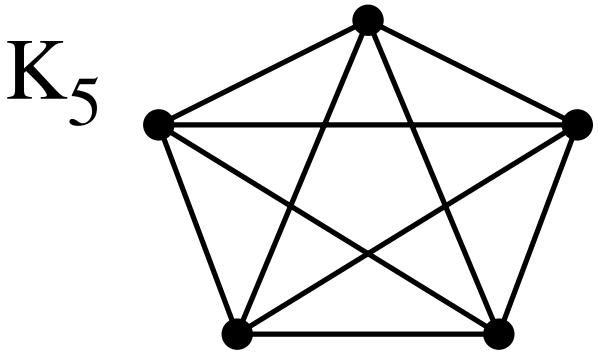
$$A = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 2 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

$V = \{1, \ldots, 5\}$
$E = \{a, \ldots, i\}$

$\phi(a) = (2,1)$ $\phi(b) = (1,5)$ $\phi(c) = (1,1)$
$\phi(d) = (3,2)$ $\phi(e) = (5,2)$ $\phi(f) = (5,2)$
$\phi(g) = (3,4)$ $\phi(h) = (4,5)$ $\phi(i) = (5,4)$

- A directed multigraph may have loops and multiple edges.
  - Represent it as $G = (V, E, \phi)$.
  - Name the edges with labels. Let E be the set of the labels.
  - $\phi(L) = (u, v)$ means the edge with label L goes from u to v.
- **Technicality:** A loop counts once in indegree, outdegree, and a_{vv}.

Complete graph K_n



- The *complete graph* on n vertices, denoted K_n, is a graph with n vertices and an edge for all pairs of distinct vertices.

- How many edges are in K_n?

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

How many simple graphs are there on n vertices?

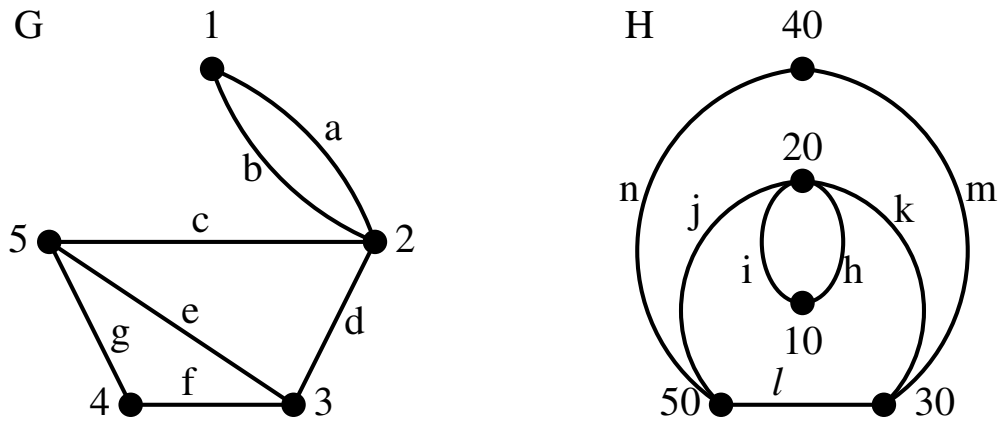
How many simple undirected graphs are there on n vertices?

- The edges are a subset of $\{\{u,v\} : u,v \in V, \quad u \neq v\}$, so $2^{\binom{n}{2}}$.
- For $n = 5$: $2^{5\cdot 4/2} = 2^{10} = 1024$

How many simple directed graphs are there on n vertices?

- The edges are a subset of $\{(u,v) : u,v \in V, \quad u \neq v\}$, so $2^{n(n-1)}$.
- For $n = 5$: $2^{5\cdot 4} = 2^{20} = 1048576$

Isomorphic graphs



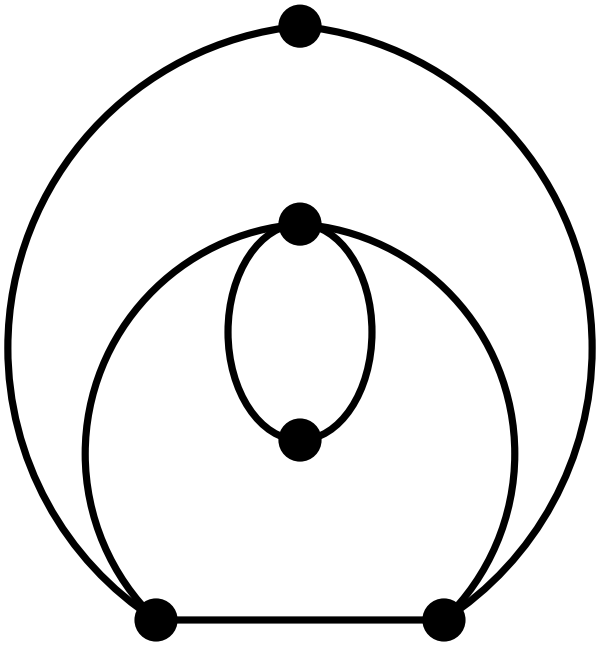
- Graphs G and H are *isomorphic* if there are bijections $\nu : V(G) \to V(H)$ and $\epsilon : E(G) \to E(H)$ that are compatible:
  - Undirected: Every edge $e = \{x, y\}$ in G has $\epsilon(e) = \{\nu(x), \nu(y)\}$ in H
  - Directed: Every edge $e = (x, y)$ in G has $\epsilon(e) = (\nu(x), \nu(y))$ in H
- The graphs are equivalent up to renaming the vertices and edges.

Vertices: $\nu(1) = 10 \quad \nu(2) = 20 \quad \nu(3) = 30 \quad \nu(4) = 40 \quad \nu(5) = 50$

Edges: $\epsilon(a) = h \quad \epsilon(b) = i \quad \epsilon(c) = j \quad \epsilon(d) = k \quad \epsilon(e) = l$
$\epsilon(f) = m \quad \epsilon(g) = n$

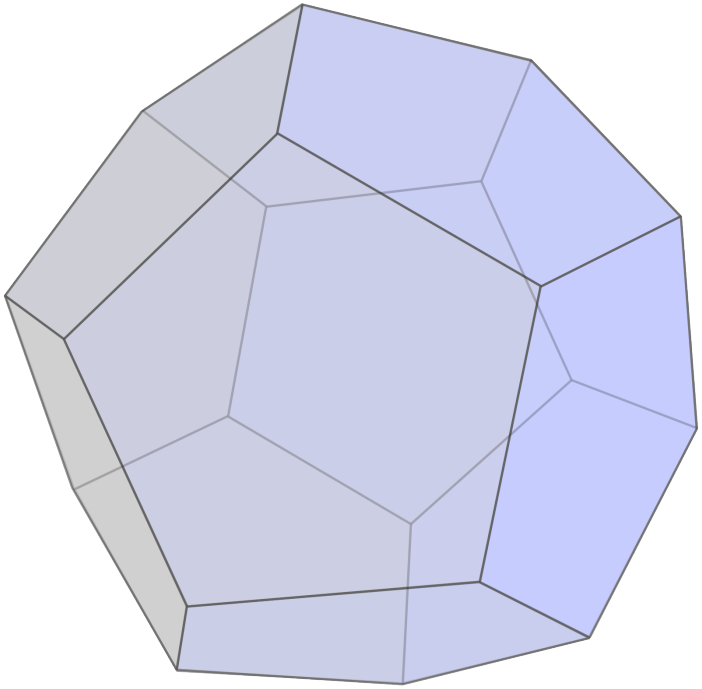
Compatibility: $a = \{1, 2\}$ and $\epsilon(a) = h = \{10, 20\} = \{\nu(1), \nu(2)\}$
. . . (Need to check all edges) . . .

Unlabeled graphs



- In an *unlabeled graph*, omit the labels on the vertices and edges.
- If labeled graphs are isomorphic, then removing the labels gives equivalent unlabeled graphs.
- This simplifies some problems by reducing the number of graphs (e.g., 1044 unlabeled simple graphs on 7 vertices vs. 2^{21} labeled).

Application: Polyhedra



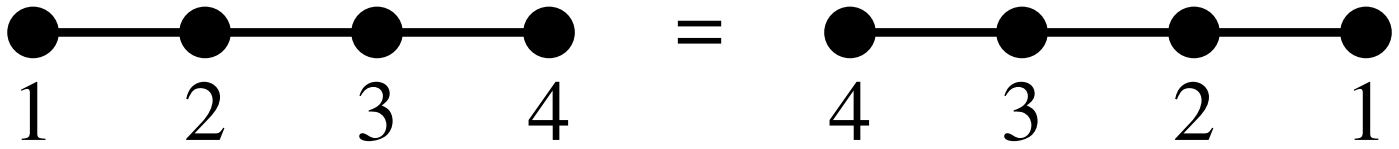
http://commons.wikimedia.org/wiki/File:Dodecahedron.svg

- A dodecahedron is a 3D shape with 20 vertices, 30 edges, and 12 pentagonal faces.
- Unlabeled graphs are used in studying other polyhedra (Ch. 12), polygons and tilings in 2D, and other geometric configurations. We can treat them as unlabeled, or pick one labeling if needed.

How many labelings of an unlabeled graph?

P_4 (*Path* on 4 vertices)

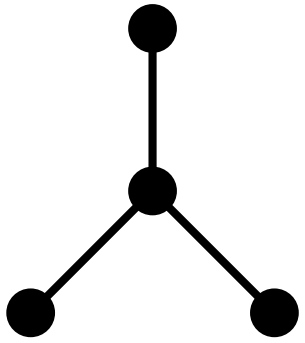
- Given an unlabeled graph on n vertices, how many ways can we label the vertices $1, \ldots, n$?
- Assign vertex labels $1, \ldots, n$ in one of $n!$ ways...
  ...but some of them are equal as abstract graphs (same vertices and edges)! Each abstract graph should only be counted once.



Both have $V = \{1, 2, 3, 4\}$, $E = \{\{1, 2\}, \{2, 3\}, \{3, 4\}\}$, so they're equal.

- For a simple path of $n > 1$ vertices ($n = 4$ above), reversing the order of the labels gives the same abstract graph, so $n!/2$ ways.

How many labelings of an unlabeled graph?



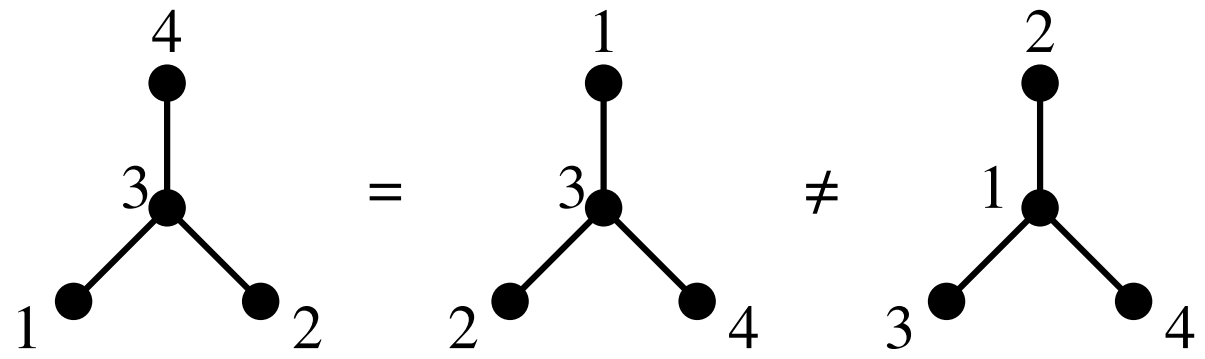
S_4: *Star* on 4 vertices

S_n : Star with one vertex at the center
and $n-1$ spokes around it.

How many distinct labelings does S_4 have, using vertex labels $1, 2, 3, 4$?

How many labelings of an unlabeled graph?

How many distinct labelings are there, using vertex labels $1, 2, 3, 4$?

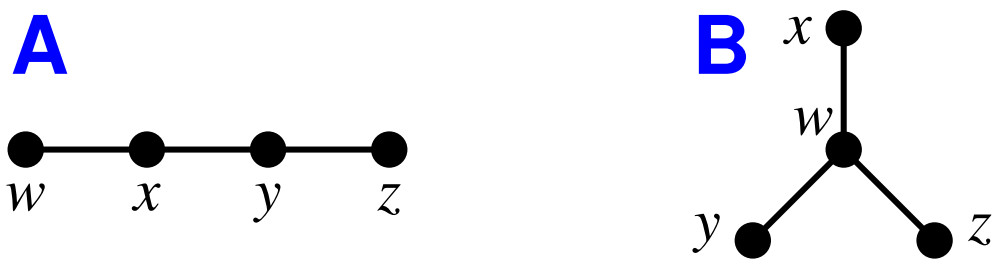


- Pick a number to go at the center.
- Place the other three numbers in any order around it.
- With 3 at center, we get edges $\{\{3,1\},\{3,2\},\{3,4\}\}$ regardless of the order of $1, 2, 4$ around the outside.
- So there are 4 labelings, corresponding to the number at the center.

How many labelings of an unlabeled graph?

General case

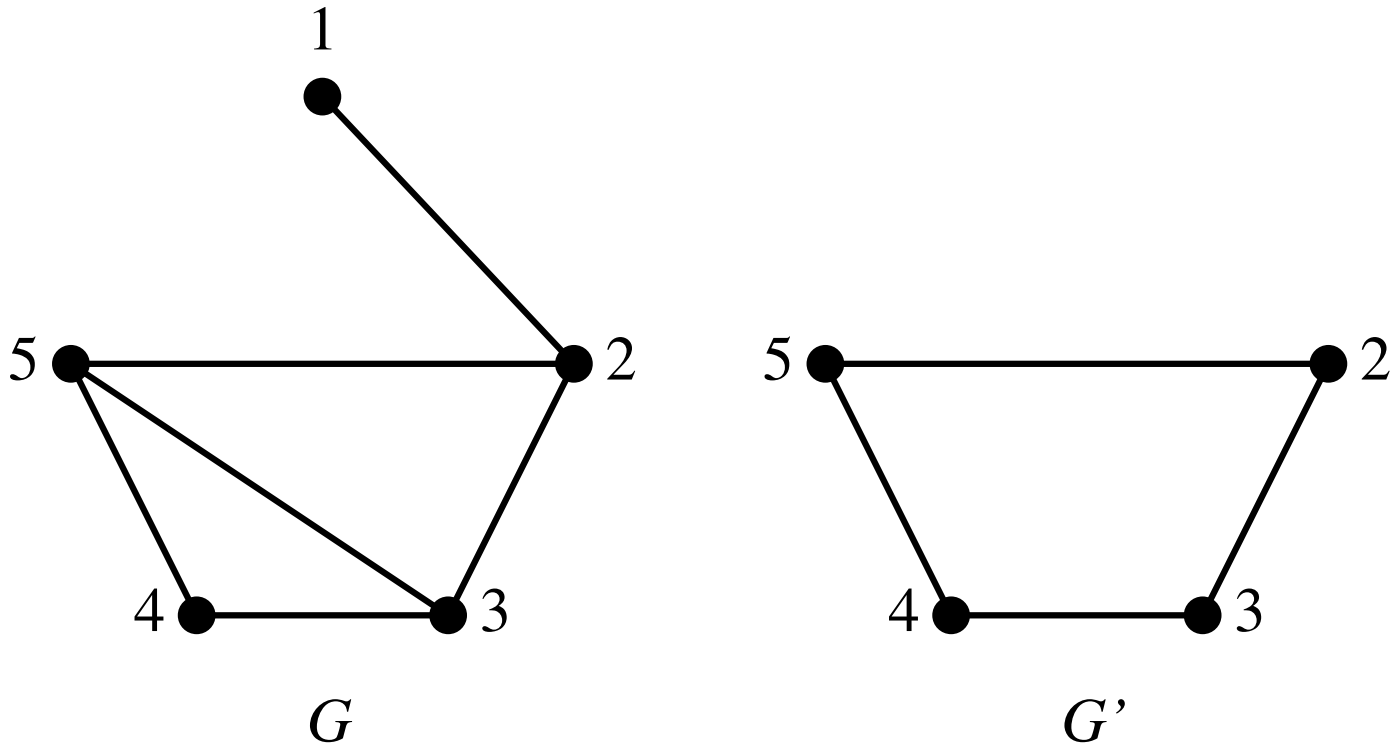
- Permuting positions of vertex labels in a graph is a group action.
- Labelings are equivalent when they give the same abstract graph.
- The labelings using $1, \ldots, n$ once each form a single orbit.
- Divide $n!$ by the number of permutations that stabilize a labeling.



A: The stabilizer of a labeling is to keep it the same or to reverse it.
Both give $E = \{\{w,x\},\{x,y\},\{y,z\}\}$.
For $n \geqslant 2$, path P_n has $n!/2$ labelings.

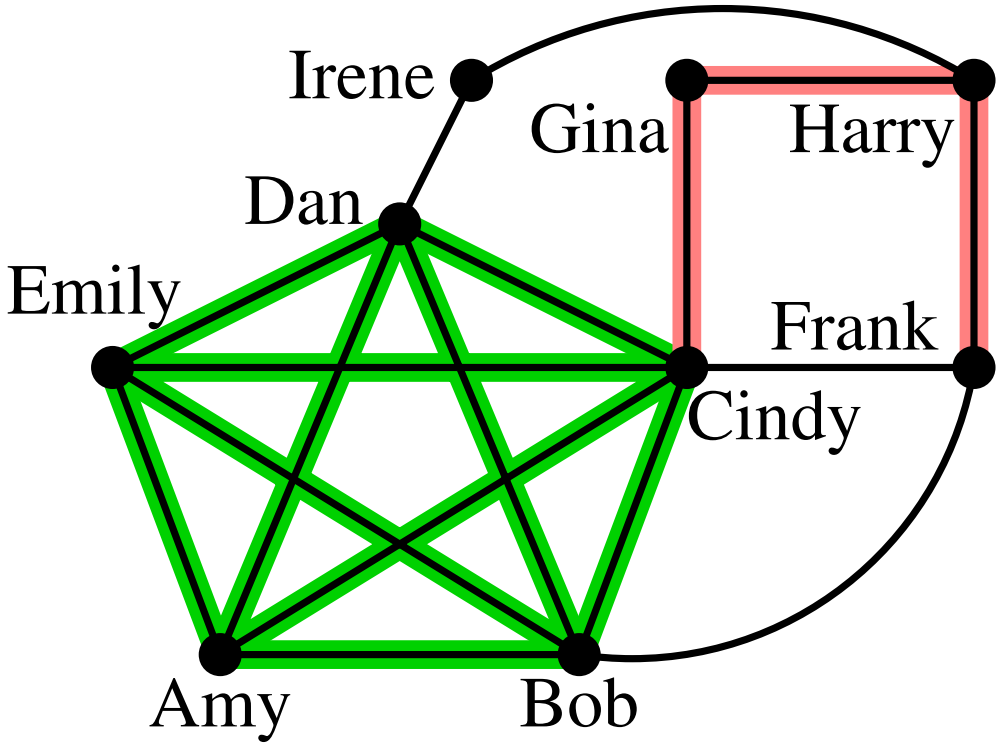
B: The stabilizer is to keep w the same and permute x, y, z arbitrarily.
These give $E = \{\{w,x\},\{w,y\},\{w,z\}\}$. Thus, $4!/3! = 4$ labelings.
For $n \geqslant 3$, star S_n has $n!/(n-1)! = n$ labelings.

Subgraphs



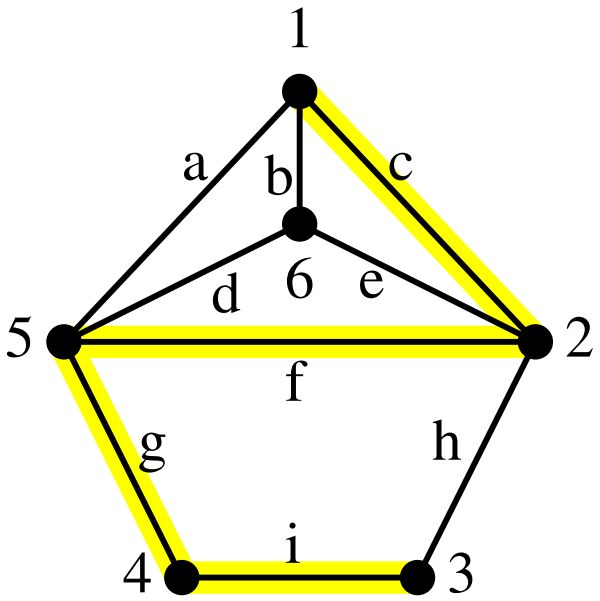
- Let $G = (V, E)$ be a graph.
- A *subgraph* is $G' = (V', E')$ where $V' \subseteq V$, $E' \subseteq E$, and the edges of E' only involve vertices of V'.
- If we remove a vertex v, we must remove all edges incident with it.
- We may also remove additional edges.
- For multigraphs and directed graphs, it's similar.

Subgraphs



- Green subgraph: Amy, Bob, Cindy, Dan, Emily and all their edges is a K_5.
- Pink subgraph: Cindy, Frank, Gina, Harry, and *some* of their edges.

Walks — Example: Transit map



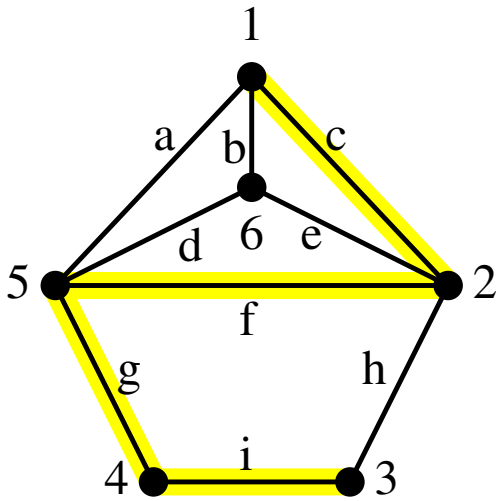
- In a bus network, you can get from station 1 to 3 via

  Route c from station 1 to station 2
  Route f from station 2 to station 5
  Route g from station 5 to station 4
  Route i from station 4 to station 3

- As a sequence of edges: c, f, g, i (4 edges)
  vertices: $1, 2, 5, 4, 3$ (5 vertices)

- The length of this is the number of edges, 4.

Walks



- Trace along edges from vertex x to y, without lifting your pen.
- A *walk* from vertex x to y is a sequence of edges, each connected to the next by a vertex:

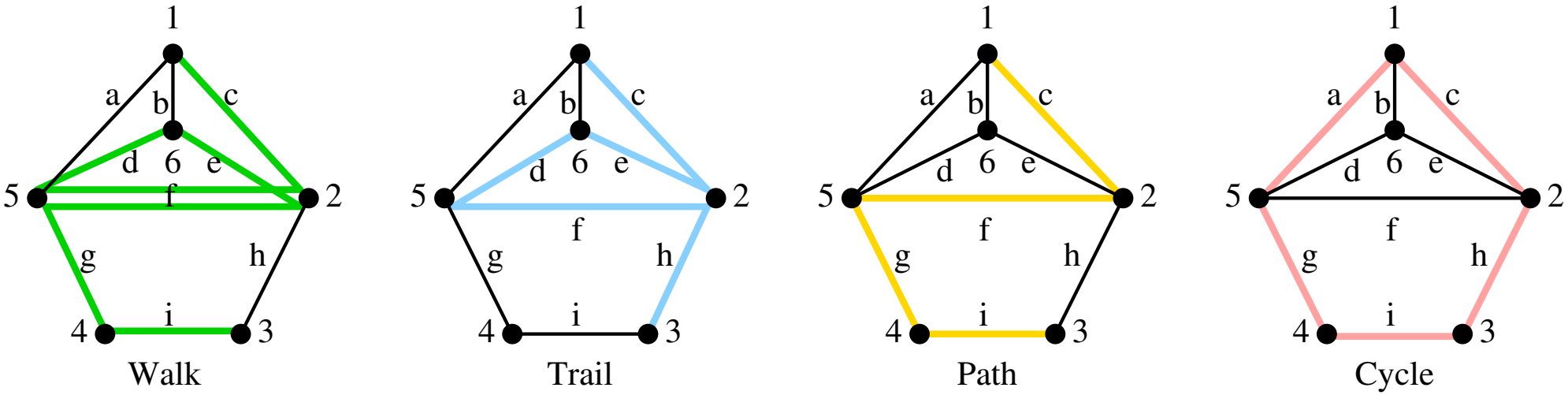
$$e_1 = \{x, v_1\} \quad e_2 = \{v_1, v_2\} \quad e_3 = \{v_2, v_3\} \quad \cdots \quad e_k = \{v_{k-1}, y\}$$

  In a directed graph, edge directions must be respected:

$$e_1 = (x, v_1) \quad e_2 = (v_1, v_2) \quad e_3 = (v_2, v_3) \quad \cdots \quad e_k = (v_{k-1}, y)$$

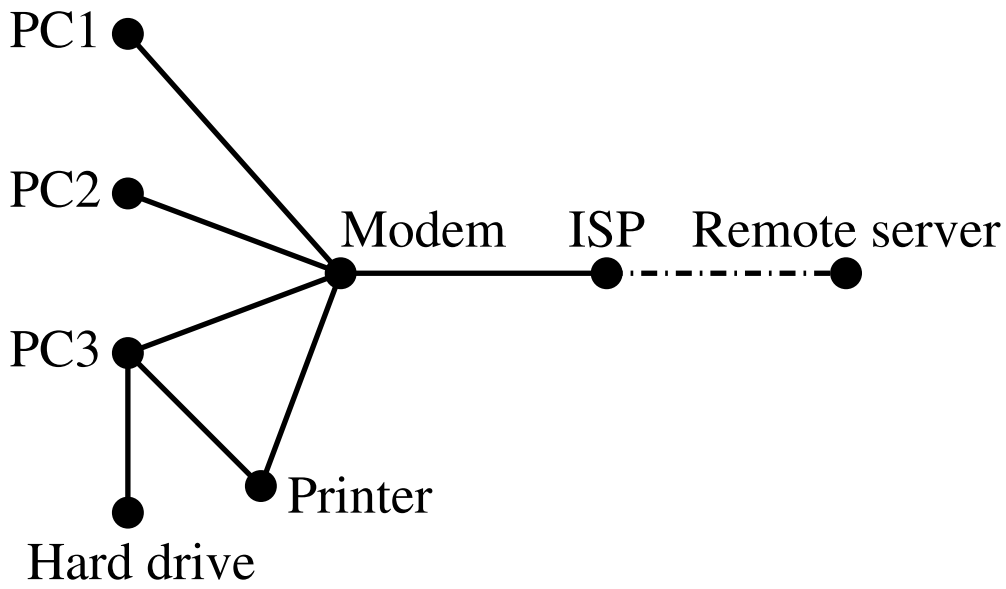
- The *length* of the walk is the number of edges, k (not the number of vertices, $k+1$).

Walks



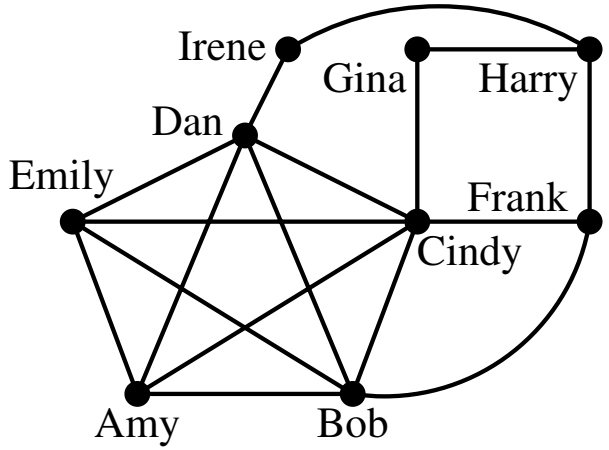
- In a *walk*, edges and vertices may be re-used.
- A *trail* is a walk with all edges distinct.
- A *path* is a walk with all vertices and edges distinct.
- A walk/trail/path is *open* if the start and end vertices are different, and *closed* if they are the same (this is allowed in a closed path, but no other vertices may be repeated).
- In our book, a *cycle* is a closed path.
  But some authors use *cycle* for a closed walk, so always check the definition an author is using.

Example: Paths in a computer network



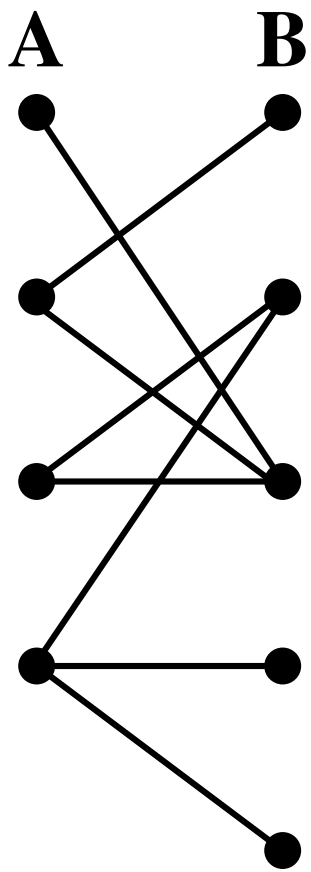
- For PC1 to print a page from a website, first it retrieves the page
$$\text{Remote server} \to \cdots \to \text{ISP} \to \text{Modem} \to \text{PC1}$$
and then it sends it to the printer
$$\text{PC1} \to \text{Modem} \to \text{Printer}$$
- PC3 can directly print on the printer w/o going through the Modem.
- For PC1 to read a file from the hard drive, it goes through a path
$$\text{Hard drive} \to \text{PC3} \to \text{Modem} \to \text{PC1}$$

Example: Graph of friends



- The *length* of a walk/trail/path is the number of edges in it.
  The *distance* between vertices is the length of the shortest path.
- **Amy's friends:** Bob, Cindy, Dan, Emily
  - Each is distance 1 from Amy.
- **Amy and Frank's mutual friends:** Bob, Cindy
  - They are the middle vertex on a path of length 2 from Amy to Frank.
- **Amy's friends of friends:** Frank, Gina, Irene
  - Each is distance 2 from Amy.
- **"Six degrees of separation":** A popular concept is that everyone is $\leqslant 6$ steps away from everyone in the world. (It's probably false.)

Bipartite graph (Ch. 11.2)



A *bipartite graph* is a graph in which:

- The set of vertices can be split as $V = A \cup B$, where $A \cap B = \emptyset$.
- Every edge has the form $\{a, b\}$ where $a \in A$ and $b \in B$.

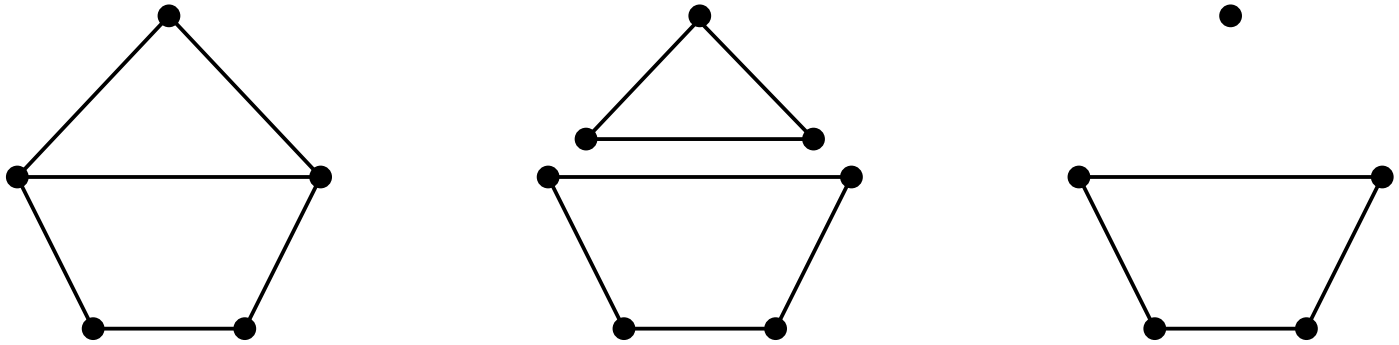
Note that there may be vertices $a \in A$, $b \in B$ that do not have an edge.

Cycle in a bipartite graph

A cycle in a bipartite graph must have even length:

- A cycle has consecutive vertices $v_0, v_1, \ldots, v_n$ with $v_0 = v_n$.
- For a cycle in a bipartite graph, the vertices alternate coming from
$$A, B, A, B, \ldots \qquad \text{or} \qquad B, A, B, A, \ldots.$$
- Since $v_0 = v_n$, they're both in A or both in B, so n is even.

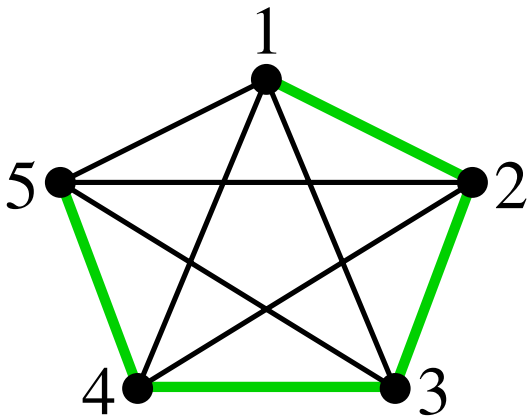
Connected graph



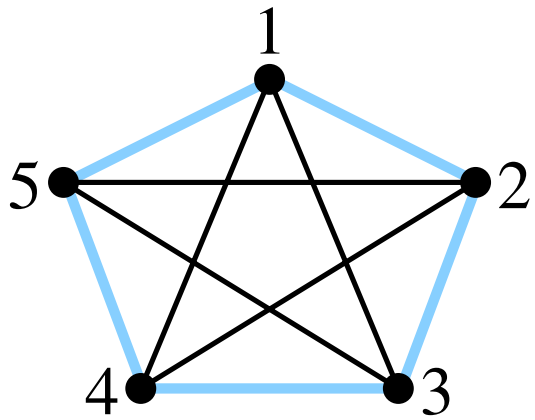
1 component | 2 components | 2 components

- An undirected graph is *connected* if for all vertices u, v, there is a path from u to v.
- The graph on the left is connected. The other two are not.
- A graph may be split into *connected components*.
  - Partition the graph into subgraphs. Vertices u, v are in the same connected component iff there is a path from u to v.
  - A vertex with no edges is in its own connected component.

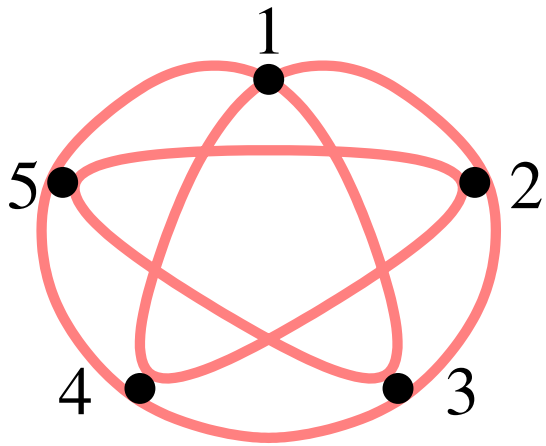
Walks using all vertices or all edges



Hamiltonian path
1,2,3,4,5



Hamiltonian cycle
1,2,3,4,5,1



Eulerian cycle
1,2,3,4,5,1,3,5,2,4,1

- A *Hamiltonian path* is a path that uses every vertex exactly once.
- An *Eulerian trail* is a trail that uses every edge exactly once.
  Note: Euler is pronounced “oiler.”
- When the starting point is the same as the ending point, these are called a *Hamiltonian cycle* and *Eulerian cycle*, respectively.
- A Hamiltonian cycle starts and ends at the same vertex, but all other vertices are used just once.
- These are used in many algorithms in Computer Science.

Dinner seating arrangements

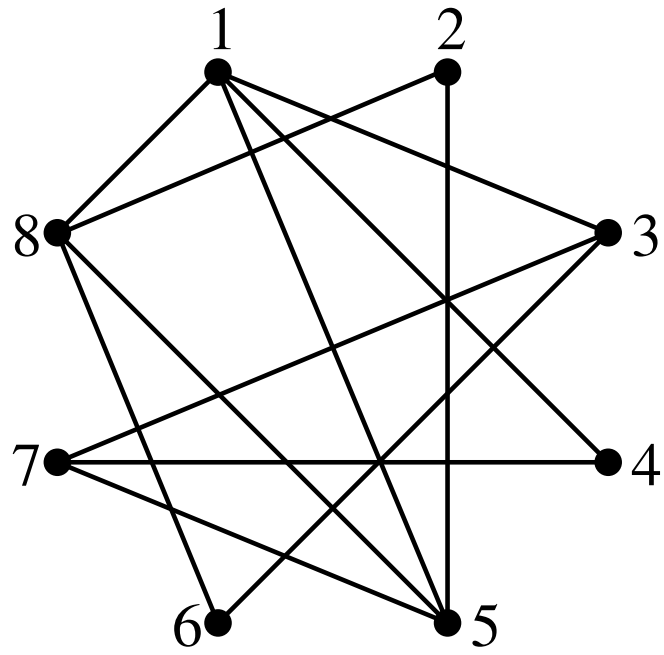
- Sit 8 people at a circular table so that no one knows their neighbors. ✓ indicates people who do not know each other and thus may be seated next to each other.

	1	2	3	4	5	6	7	8
1			✓	✓	✓			✓
2					✓			✓
3	✓					✓	✓	
4	✓					✓	✓	
5	✓	✓					✓	✓
6			✓	✓				✓
7			✓	✓	✓			
8	✓	✓			✓	✓		

- Form a graph on vertices $\{1, \ldots, 8\}$, with an edge $\{i, j\}$ if i and j may be seated next to each other.
  The table is essentially the adjacency matrix: ✓ $= 1$, blank $= 0$.

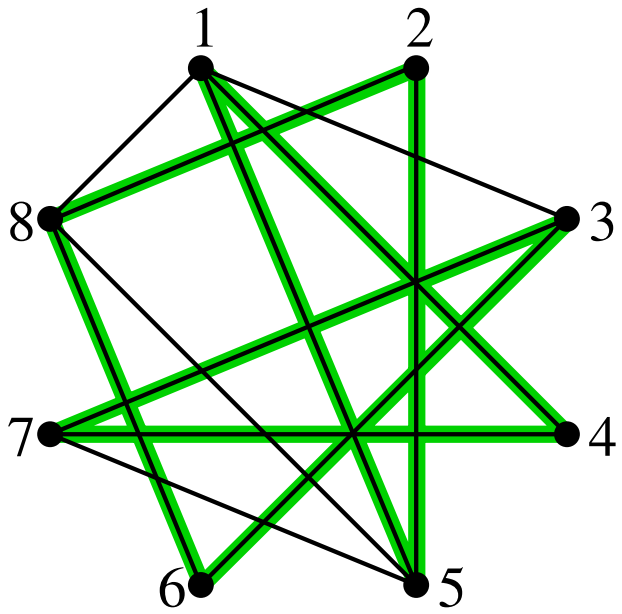
Dinner seating arrangements

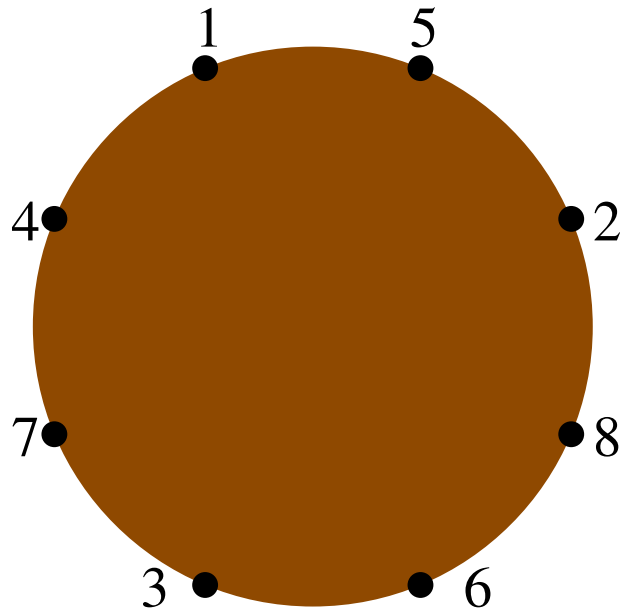
	1	2	3	4	5	6	7	8
1			✓	✓	✓			✓
2					✓			✓
3	✓					✓	✓	
4	✓					✓	✓	
5	✓	✓					✓	✓
6			✓	✓				✓
7			✓	✓	✓			
8	✓	✓			✓	✓		



- The solutions correspond to Hamiltonian cycles!
- Start with 1 and successively follow edges to vertices not yet used, until we return to 1 at the end.
- E.g., try 1, 8, 6, 3, 7, 4; but then we're stuck because we can't yet go back to 1!

Dinner seating arrangements



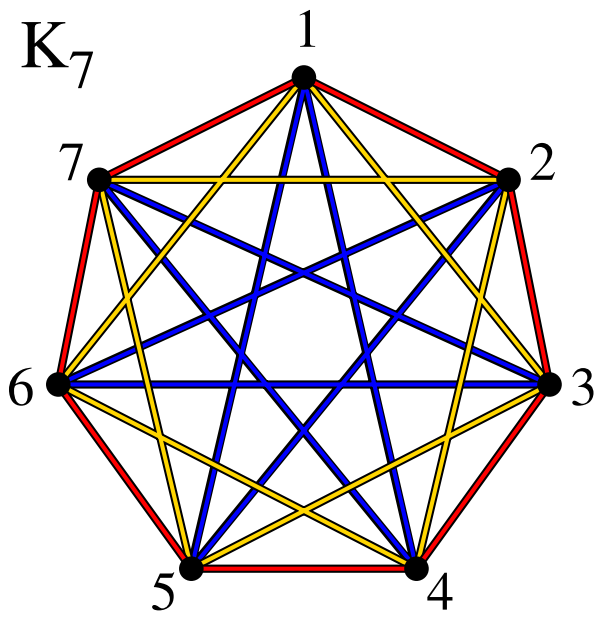


- One Hamiltonian cycle is $1, 5, 2, 8, 6, 3, 7, 4, 1$.
- If someone gives you a solution, it is easy to verify if it's correct. But testing all $n!$ possibilities is impractical unless n is small.
- There is no known *efficient* method, guaranteed to work in *all* graphs, to either find a Hamiltonian cycle or prove there isn't one. In Computer Science, it is classified as an NP-complete problem (CSE 101, or Chapter 20 of our book).

Dinner seating arrangements

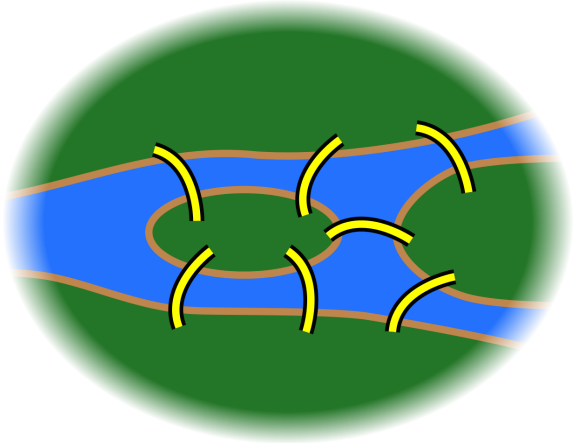
- A group of n people will have a series of dinners with everyone at the same circular table. How many dinners can be arranged so that no one repeats sitting next to the same person?
- After d such dinners, each person will have sat next to $2d$ people out of the $n-1$ people besides themself, so $2d \leqslant n-1$.
- An upper bound on the number of dinners possible is $\left\lfloor \frac{n-1}{2} \right\rfloor$.

Dinner seating arrangements



- If n is an odd prime, $(n-1)/2$ is achievable!
- For each $x = 1, 2, \ldots, (n-1)/2$, form a Hamiltonian cycle in K_n:
  $$1, \quad 1+x, \quad 1+2x, \quad \ldots, \quad 1+(n-1)x, \quad 1+nx \equiv 1 \pmod{n}$$
- There are no repeated numbers besides 1 at the start and end:
  If $1+ax \equiv 1+bx \bmod n$ then $(a-b)x \equiv 0 \bmod n$.
  Since n is prime, the only solution in this range is $a = b$.

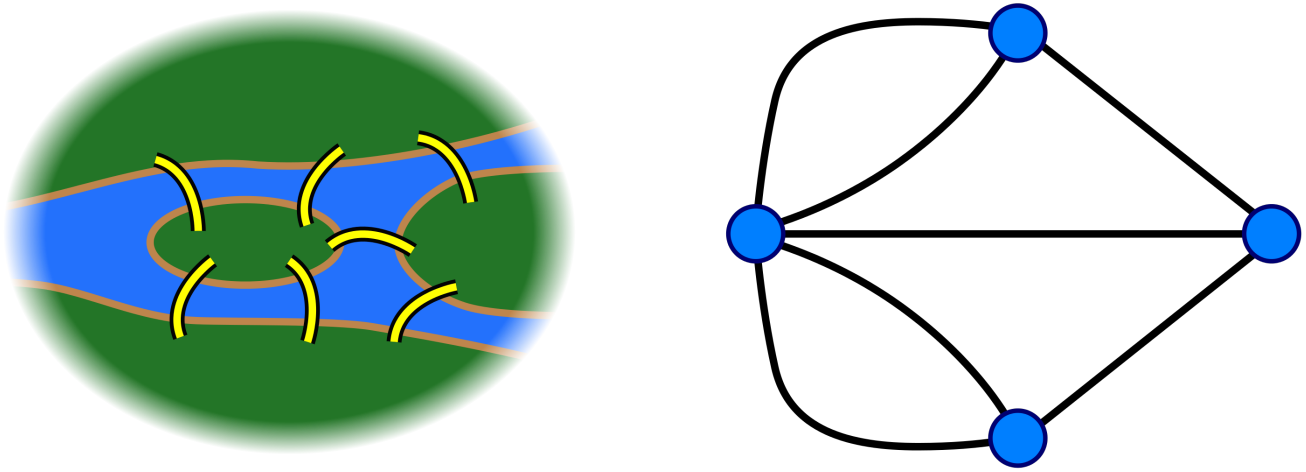
Eulerian Trails and the Seven Bridges of Königsberg



http://en.wikipedia.org/wiki/Seven_Bridges_of_K%C3%B6nigsberg

- In the 1700s, a river in Königsberg, Prussia, split the city into four land masses (including two islands), connected by seven bridges.
- Can you walk through town and cross every bridge exactly once? No backtracking, no partial bridge-crossings, etc.
- In 1735, Leonhard Euler proved it is impossible, and created the foundations of graph theory.

Seven Bridges of Königsberg

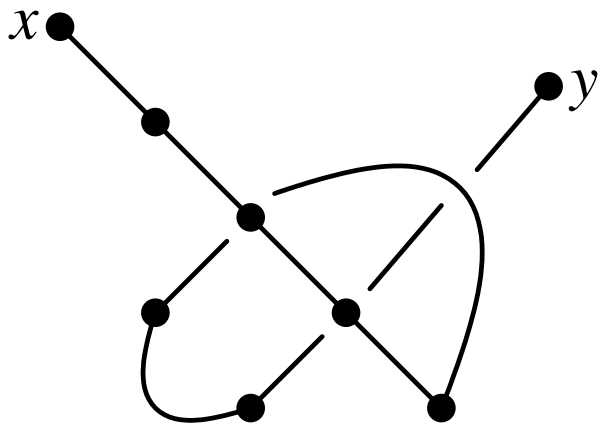


http://en.wikipedia.org/wiki/Seven_Bridges_of_K%C3%B6nigsberg

- Replace each land mass by a vertex and each bridge by an edge.
- For every vertex (except the first and last, if different) each time we enter on one edge, we exit on another. If we use up all the edges in this fashion, the degree is even.
- If the starting and ending vertices are different, they have odd degrees since the first/last edge do not have an in/out pair.
- Here, all vertices have odd degree, so such a walk is impossible.

Necessary conditions for Eulerian trails and cycles

- An *Eulerian trail* from vertex x to y ($x \neq y$) uses every edge once.
- If $x = y$: An *Eulerian cycle* is a cycle that uses every edge once.
- All the edges need to be in the same connected component.
- Each time we enter a vertex on one edge, we exit on a separate edge. So all vertices in this connected component must have even degree, except if $x \neq y$ then x, y have odd degrees.



Eulerian cycles

Theorem (Eulerian cycles)

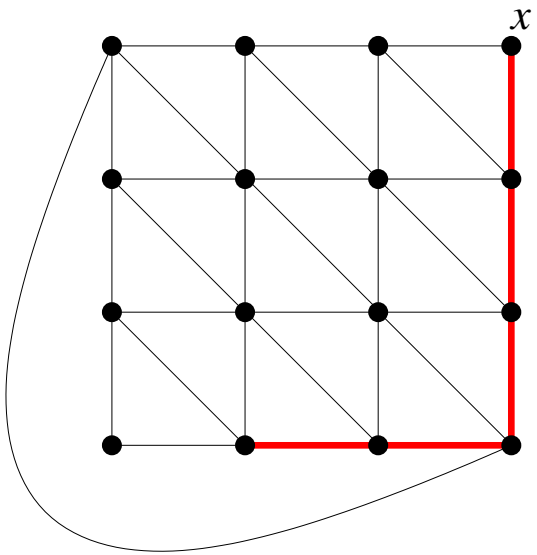
A connected graph has an Eulerian cycle if and only if all vertices have even degrees.

Proof, Step 1: Construct a cycle starting at any vertex x.

- We showed necessity; now we'll show sufficiency.
- Start at any vertex, $v_0 = x$.
- Pick any edge on x, say e_1, and follow it to the next vertex, v_1.
- Pick any unused edge on v_1, say e_2. Follow it to the next vertex, v_2.
- Continue alternately picking vertices & edges

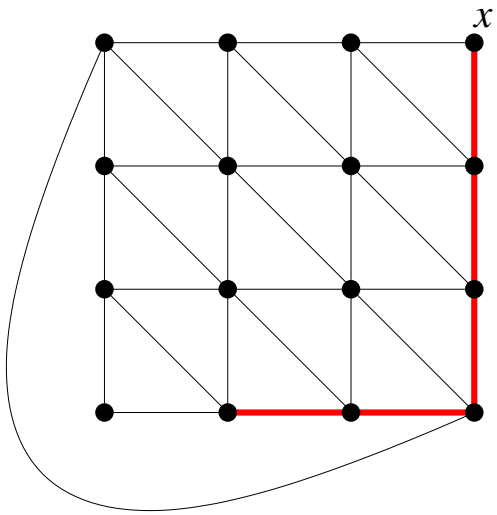
  $$v_0, e_1, v_1, e_2, v_2, \ldots, e_k, v_k,$$

  until forced to stop at a vertex v_k with no unused edge to follow.



Eulerian cycles, Proof, Step 1, continued

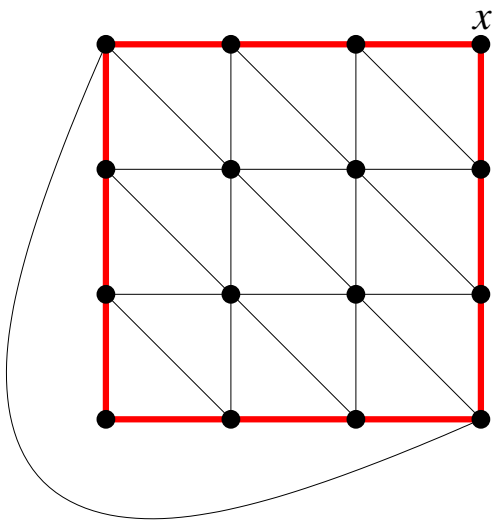
Construct a cycle starting at any vertex x



- So far we selected $v_0 = x$, e_1, v_1, e_2, v_2, $\ldots$, e_k, v_k.
- We used an even number of edges at every vertex on this trail (one edge in, another out), except the first and last vertices (x, v_k).
- If $v_k \neq x$, then we used an odd number of edges on v_k.
  But v_k has even degree, so there's an unused edge on it, and we did not have to stop!
- Thus, $v_k = x$, and that vertex uses an even number of edges too: first (e_1), last (e_k), and possibly in/out pairs in-between.

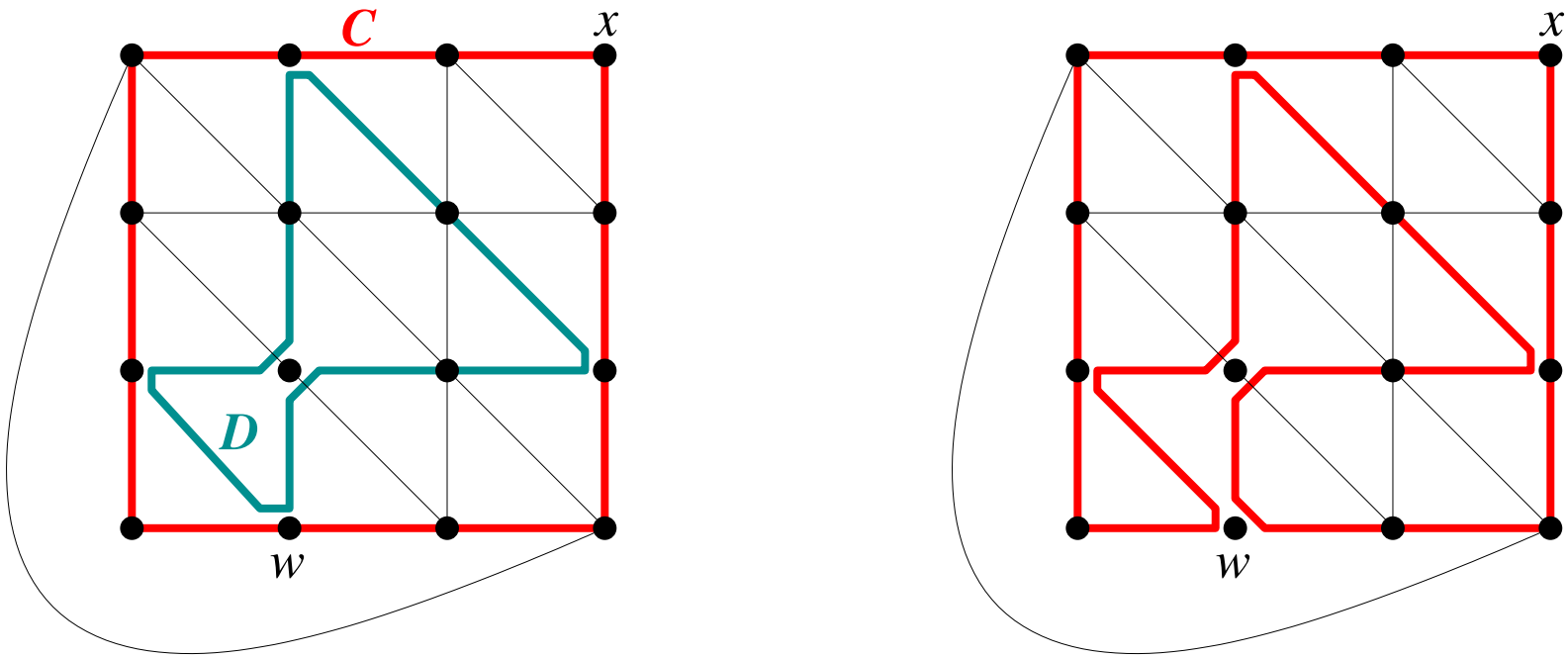
Eulerian cycles, Proof, Step 1, continued

Construct a cycle starting at any vertex x



We constructed a cycle based at x, but it may not use all edges of the graph.

Eulerian cycles, Proof, Step 2



- We have a cycle C, but an Eulerian cycle must use *all* edges.
- What if some edges are unused? Since the graph is connected, there is an unused edge that touches some vertex w on C.
- Form a new cycle D based at w, using the same algorithm as in Step 1. No edges in Step 1 may be re-used.
- Splice D into C at w to make C larger.
- Repeat until all edges are in C. Now it's an Eulerian cycle!

Eulerian trails

Theorem

A connected graph G has an Eulerian trail from x to y $(x \neq y)$ if and only if x, y have odd degrees while all other vertices have even degree.

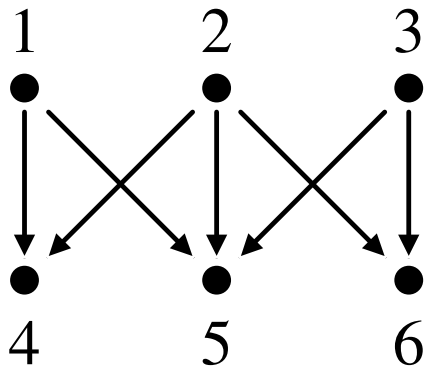
Proof.

- Form graph G' by adding an edge $\{x, y\}$ to G.
- Now all vertices have even degree, so there is an Eulerian cycle.
- Remove the new edge $\{x, y\}$ to form an Eulerian trail from x to y. □

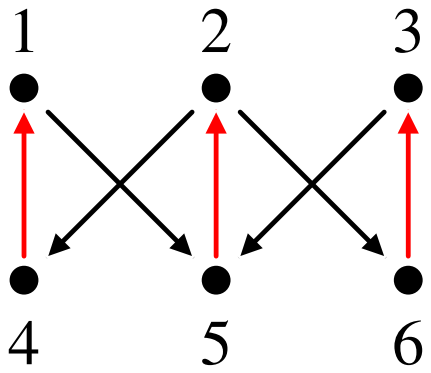
Generalization

For both Eulerian cycles and trails, we may replace *"connected graph"* by *"a graph whose vertices with nonzero degree form a single connected component"*. We may add isolated vertices since they have no edges, so no walks go through them.

Strongly connected components in a directed graph



Not strongly connected



Strongly connected

- A directed graph is *strongly connected* iff for every pair of vertices $x \neq y$, there is a directed path from x to y and also one from y to x.
- A vertex is *balanced* if its indegree and outdegree are equal.
  A graph is *balanced* if all vertices are balanced.
  - **Left graph:** None of the vertices are balanced.
  - **Right graph:** 1, 3, 4, 6 are balanced, and 2, 5 are unbalanced.
- Does either graph have an Eulerian trail or cycle?
  - **Left:** Can't go in then out of 1, so no. (Not balanced.)
  - **Right:** Start/end at the two unbalanced vertices: 2,4,1,5,2,6,3,5

Eulerian cycles in a directed graph

Theorem

- *A directed graph has an Eulerian cycle iff it is balanced and all vertices with degree $\neq 0$ form a strongly connected component.*
- *If a directed graph is balanced and the undirected version of the graph is connected, the directed graph is strongly connected.*
- *A directed graph has an Eulerian trail from x to y $(x \neq y)$ iff*
  - $\text{outdegree}(x) = \text{indegree}(x) + 1$
  - $\text{indegree}(y) = \text{outdegree}(y) + 1$
  - $\text{indegree}(v) = \text{outdegree}(v)$ *for all other vertices*
  - *The vertices with nonzero degree form a connected component in the undirected version of the graph.*

Proof (sketch)

For Eulerian cycles in undirected graphs, when we entered a vertex, even degrees ensured there was an edge on which to exit.
Balance ensures that for directed graphs.