BIL694-Lecture 1: Introduction to Graphs

Lecturer: Lale Özkahya

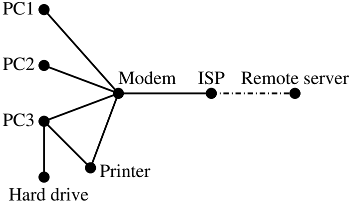
Resources for the presentation:
http://www.math.ucsd.edu/ gptesler/184a/calendar.html
http://www.inf.ed.ac.uk/teaching/courses/dmmr/

Outline

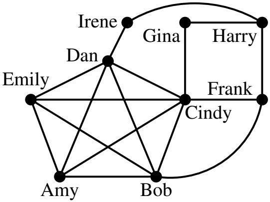
Outline

Graphs

Computer network



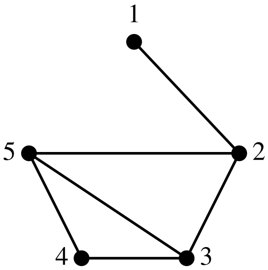
Friends



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



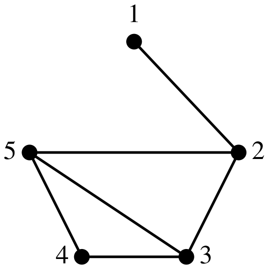
- The dots are called *vertices* or *nodes* (singular: vertex, node)

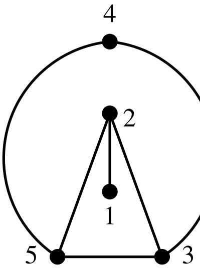
$$V = \text{set of vertices} = \{1, 2, 3, 4, 5\}$$

- 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 $\{2, 5\} = \{5, 2\}$.

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

Simple graphs

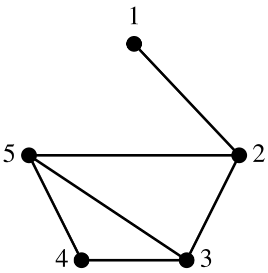




A *simple graph* is $G = (V, E)$:

- 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

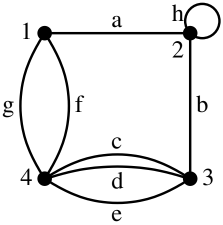
The sum of degrees of all vertices is twice the number of edges:

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

Proof.

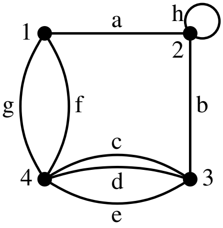
- Let $S = \{\, (v,e) \;:\; v \in V,\;\; e \in E,\;\; \text{vertex } v \text{ is in edge } e \,\}$
- **Count $|S|$ by vertices:** Each vertex v is contained in $d(v)$ edges, so
$$|S| = \sum_{v\in V} d(v).$$
- **Count $|S|$ by edges:** Each edge has two vertices, so
$$|S| = \sum_{e\in E} 2 = 2\,|E|\;.$$

Multigraphs



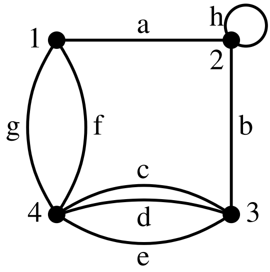
- Some networks have *multiple edges* between two vertices. Notation $\{3, 4\}$ is ambiguous, so write labels on the edges: c, d, e.
- There can be an edge from a vertex to itself, called a *loop* (such as h above). A loop has one vertex, so $\{2, 2\} = \{2\}$.
- A simple graph does not have multiple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



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

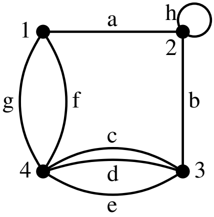
$$\phi(a) = \{1, 2\}$$
$$\phi(b) = \{2, 3\}$$
$$\phi(c) = \phi(d) = \phi(e) = \{3, 4\}$$
$$\phi(f) = \phi(g) = \{1, 4\}$$
$$\phi(h) = \{2\}$$

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 $u, v \in V$.



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

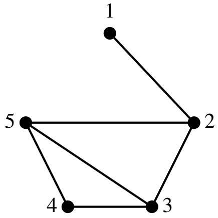
- $a_{uv} = a_{vu}$ for all vertices u, v. Thus, A is a *symmetric matrix* ($A = A^T$).
- The sum of entries in row u is the degree of u.
- **Technicality:** A loop on vertex v counts as
  - 1 edge in E,
  - degree 2 in $d(v)$ and in a_{vv} (it touches vertex v twice),

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

Adjacency matrix of a simple graph

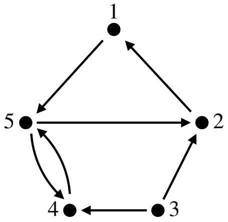
In a simple graph:

- 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).



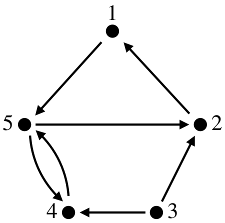
$$A = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

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)

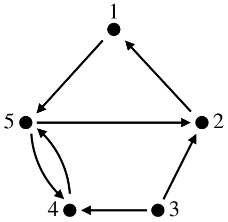


$$V = \{1, 2, 3, 4, 5\}$$

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

- 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

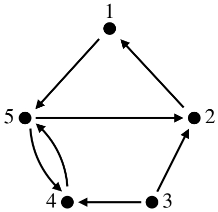


- For a vertex v, the *indegree* is the number of edges going into v, and the *outdegree* is the number of edges going out from v.

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 $|E|$.

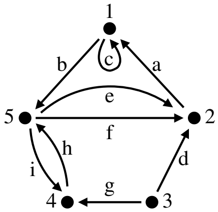
Adjacency matrix of a directed graph



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

- 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



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

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

- 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}.

Outline

Isomorphism of Graphs

Definition: Two (undirected) graphs *G1* = (*V1*, *E*1) and *G*2 = (*V*2, *E*2) are *isomorphic* if there is a bijection, $f: V_1 \rightarrow V_2$, with the property that for all vertices $a, b \in V_1$

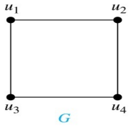
$$\{a, b\} \in E_1 \quad \text{if and only if} \quad \{f(a), f(b)\} \in E_2$$

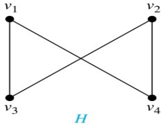
Such a function *f* is called an *isomorphism*.

Intuitively, isomorphic graphs are “THE SAME”, except for “renamed” vertices.

Isomorphism of Graphs (*cont.*)

Example: Show that the graphs $G = (V, E)$ and $H = (W, F)$ are isomorphic.





Solution: The function f with $f(u1) = v1$, $f(u2) = v4$, $f(u3) = v3$, and $f(u4) = v2$ is a one-to-one correspondence between V and W.

Isomorphism of Graphs (*cont.*)

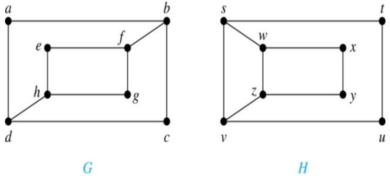
It is difficult to determine whether two graphs are isomorphic by brute force: there are $n!$ bijections between vertices of two n-vertex graphs.

Often, we can show two graphs are not isomorphic by finding a property that only one of the two graphs has. Such a property is called *graph invariant*:

- e.g., number of vertices of given degree, the degree sequence (list of the degrees), ...

Isomorphism of Graphs (*cont.*)

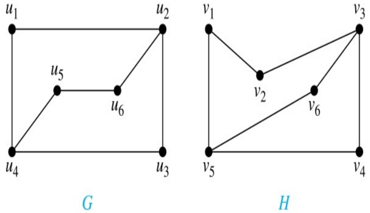
Example: Are these graphs are isomorphic?



Solution: No! Since *deg*(*a*) = 2 in *G*, *a* must correspond to *t*, *u*, *x*, or *y*, since these are the vertices of degree 2 in H. But each of these vertices is adjacent to another vertex of degree 2 in *H*, which is not true for *a* in *G*. So, G and H can not be isomorphic.

Isomorphism of Graphs (*cont.*)

Example: Determine whether these two graphs are isomorphic.



Solution: The function *f* is defined by: $f(u1) = v6$, $f(u2) = v3$, $f(u3) = v4$, $f(u4) = v5$, $f(u5) = v1$, and $f(u6) = v2$ is a bijection.

Algorithms for Graph Isomorphism

- The best algorithms known for determining whether two graphs are isomorphic have exponential worst-case time complexity (in the number of vertices of the graphs).
- However, there are algorithms with good time complexity in many practical cases.
- See, e.g., a publicly available software called NAUTY for graph isomorphism.

Applications of Graph Isomorphism

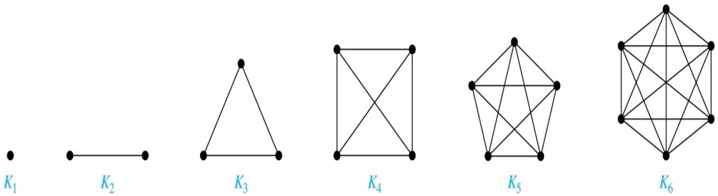
The question whether graphs are isomorphic plays an important role in applications of graph theory. For example:

Chemists use molecular graphs to model chemical compounds. Vertices represent atoms and edges represent chemical bonds. When a new compound is synthesized, a database of molecular graphs is checked to determine whether the new compound is isomorphic to the graph of an already known one.

Outline

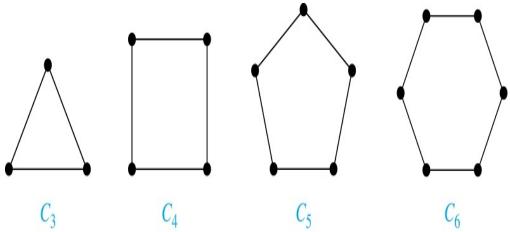
Special Types of Graphs: Complete Graphs

A *complete graph on n vertices*, denoted by K_n, is the simple graph that contains exactly one edge between each pair of distinct vertices.



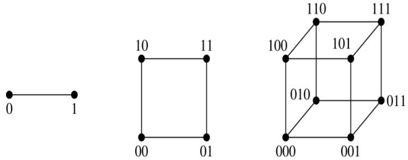
Special Types of Graphs: Cycles

A *cycle* C_n for *n* ≥ 3 consists of *n* vertices *v*1, *v*2 ,⋯ , *v*n, and edges {*v*1, *v*2}, {*v*2, *v*3} ,⋯ , {*vn*-1, *vn*}, {*vn*, *v*1}.



Special Types of Simple Graphs: *n*-Cubes

An *n-dimensional hypercube*, or *n-cube*, is a graph with 2^n vertices representing all bit strings of length *n*, where there is an edge between two vertices if and only if they differ in exactly one bit position.



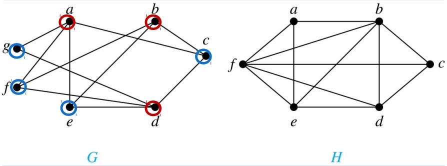
Q_1 Q_2 Q_3

Bipartite Graphs

Definition:

An equivalent definition of a bipartite graph is one where it is possible to color the vertices either red or blue so that no two adjacent vertices are the same color.

G is bipartite

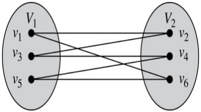


H is not bipartite: if we color *a* red, then its neighbors *f* and *b* must be blue. But f and b are adjacent.

Bipartite Graphs (*continued*)

Example: Show that C_6 is bipartite.

Solution: Partition the vertex set into *V*1 = {*v*1, *v*3, *v*5} and *V*2 = {*v*2, *v*4, *v*6}:



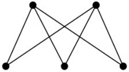
Example: Show that *C_3* is not bipartite.

Solution: If we partition vertices of *C*_3 into two nonempty sets, one set must contain two vertices. But every vertex is connected to every other. So, the two vertices in the same partition are connected. Hence, *C*_3 is not bipartite.

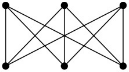
Complete Bipartite Graphs

Definition: A *complete bipartite graph* is a graph that has its vertex set partitioned $K_{m,n}$ into two subsets *V*_1 of size *m* and *V*_2 of size *n* such that there is an edge from every vertex in *V*_1 to every vertex in *V*_2.

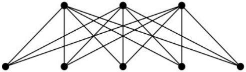
Examples:



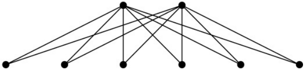
$K_{2,3}$



$K_{3,3}$



$K_{3,5}$

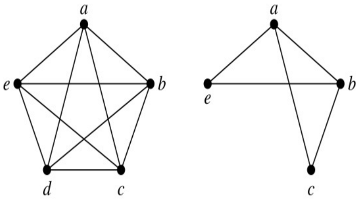


$K_{2,6}$

Subgraphs

Definition: A *subgraph of a graph* *G* = (*V*,*E*) is a graph (*W*,*F*), where $W \subseteq V$ and $F \subseteq E$. A subgraph *H* of *G* is a proper subgraph of *G* if *H* ≠ *G*.

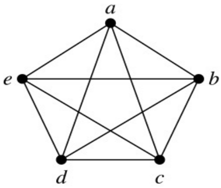
Example: here is K_5 and one of its (proper) subgraphs:

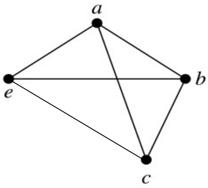


Induced Subgraphs

Definition: Let *G* = (*V*, *E*) be a graph. The *subgraph induced* by a subset *W* of the vertex set *V* is the graph H= (*W*,*F*), whose edge set *F* contains an edge in *E* if and only if both endpoints are in *W*.

Example: Here is *K*_5 and its induced subgraph induced by *W* = {*a*,*b*,*c*,*e*}.





Bipartite Graphs

Theorem (König, 1936)

A graph is bipartite if and only if it has no odd cycle.

Proof

Necessity: If G is bipartite, clearly every cycle has even length.

Sufficiency:

- Assume that G has no odd cycle, show that G is bipartite.
- Let $u \in V(G)$ and for each $v \in V(G)$, let $f(v)$ be the minimum length of a u, v-path (finite, assuming G is connnected).
- $X := \{v \in V(G) : f(v) \text{is even}\}$, $\qquad Y := \{v \in V(G) : f(v) \text{is odd}\}$.

Bipartite Graphs

Theorem (König, 1936)

A graph is bipartite if and only if it has no odd cycle.

Proof

Necessity: If G is bipartite, clearly every cycle has even length.

Sufficiency:

- Assume that G has no odd cycle, show that G is bipartite.
- Let $u \in V(G)$ and for each $v \in V(G)$, let $f(v)$ be the minimum length of a u, v-path (finite, assuming G is connnected).
- $X := \{v \in V(G) : f(v) \text{is even}\}, \qquad Y := \{v \in V(G) : f(v) \text{is odd}\}.$
- Any edge in $vw \in G[X]$, together with the shortest u, v- and u, w-paths create an odd walk, call it W. (Same if vw is an edge in $G[Y]$)

Bipartite Graphs

Theorem (König, 1936)

A graph is bipartite if and only if it has no odd cycle.

Proof

Necessity: If G is bipartite, clearly every cycle has even length.

Sufficiency:

- Assume that G has no odd cycle, show that G is bipartite.
- Let $u \in V(G)$ and for each $v \in V(G)$, let $f(v)$ be the minimum length of a u, v-path (finite, assuming G is connnected).
- $X := \{v \in V(G) : f(v) \text{is even}\}, \qquad Y := \{v \in V(G) : f(v) \text{is odd}\}.$
- Any edge in $vw \in G[X]$, together with the shortest u, v- and u, w-paths create an odd walk, call it W. (Same if vw is an edge in $G[Y]$)
- And any odd walk contains an odd cycle (by Lemma 1.2.15 in the book). Contradiction!

Bipartite Graphs

Theorem (König, 1936)

A graph is bipartite if and only if it has no odd cycle.

Proof

Necessity: If G is bipartite, clearly every cycle has even length.

Sufficiency:

- Assume that G has no odd cycle, show that G is bipartite.
- Let $u \in V(G)$ and for each $v \in V(G)$, let $f(v)$ be the minimum length of a u, v-path (finite, assuming G is connnected).
- $X := \{v \in V(G) : f(v) \text{is even}\}$, $Y := \{v \in V(G) : f(v) \text{is odd}\}$.
- Any edge in $vw \in G[X]$, together with the shortest u, v- and u, w-paths create an odd walk, call it W. (Same if vw is an edge in $G[Y]$)
- And any odd walk contains an odd cycle (by Lemma 1.2.15 in the book). Contradiction!
- Therefore, X and Y are independent sets.

Bipartite Graphs

Theorem (König, 1936)

A graph is bipartite if and only if it has no odd cycle.

Proof

Necessity: If G is bipartite, clearly every cycle has even length.

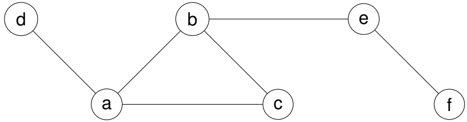
Sufficiency:

- Assume that G has no odd cycle, show that G is bipartite.
- Let $u \in V(G)$ and for each $v \in V(G)$, let $f(v)$ be the minimum length of a u, v-path (finite, assuming G is connnected).
- $X := \{v \in V(G) : f(v) \text{is even}\}$, $\quad Y := \{v \in V(G) : f(v) \text{is odd}\}$.
- Any edge in $vw \in G[X]$, together with the shortest u, v- and u, w-paths create an odd walk, call it W. (Same if vw is an edge in $G[Y]$)
- And any odd walk contains an odd cycle (by Lemma 1.2.15 in the book). Contradiction!
- Therefore, X and Y are independent sets.
- Same ideas can be applied for each component, if G had more than one component.

Outline

Connectness in undirected graphs

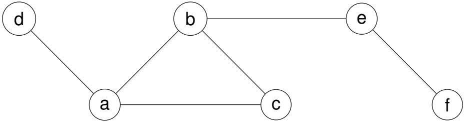
Definition: An undirected graph $G = (V, E)$ is called connected, if there is a path between every pair of distinct vertices.
It is called disconnnected otherwise.



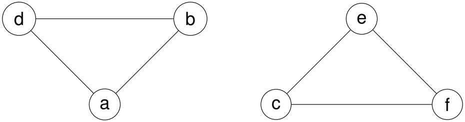
This graph is connected

Connectness in undirected graphs

Definition: An undirected graph $G = (V, E)$ is called connected, if there is a path between every pair of distinct vertices.
It is called disconnnected otherwise.

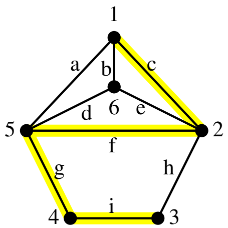


This graph is connected



This graph is **not** connected

Walks



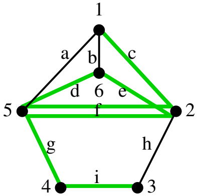
- Trace along edges from vertex x to y, without lifting your pen.
- The walk in yellow is represented as a sequence of edges c, f, g, i, or a sequence of vertices $1, 2, 5, 4, 3$.
- A *walk* from vertex x to y is a sequence of edges, each connected to the next by a vertex:

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

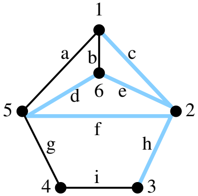
  In a directed graph, edge directions must be respected:

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

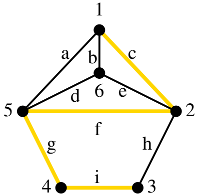
Walks



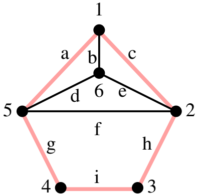
Walk



Trail



Path



Cycle

- 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).
- A *cycle* is a closed path.

Different Definitions of Trees

A tree is a simple connected graph that contains no cycle.

Theorem

The following statements are equivalent for a graph T:

1. *T is a tree;*
2. *Any two vertices of T are connected by a unique path in T;*
3. *T is minimally connected, that is, T is connected but $T - e$ is disconnected for every edge e in T;*
4. *T is maximally acylic, that is T contains no cycle, but $T + xy$ does, for any two non-adjacent vertices x, y in T.*

Observations on Trees

Corollary: Every tree has at least two vertices with degree 1.

Observations on Trees

Corollary: Every tree has at least two vertices with degree 1.
Proof: Take any longest simple path $x_0, \ldots, x_m$ in T. Both x_0 and x_m must have degree 1: otherwise there is s a longer path in T.

Corollary: The vertices of a tree can always be enumerated, say as $v_1, \ldots, v_n$ so that every v_i with $i \geq 2$ has a unique neighbor in $\{v_1, \ldots, v_{i-1}\}$.

Observations on Trees

Corollary: Every tree has at least two vertices with degree 1.
Proof: Take any longest simple path $x_0, \ldots, x_m$ in T. Both x_0 and x_m must have degree 1: otherwise there is s a longer path in T.

Corollary: The vertices of a tree can always be enumerated, say as $v_1, \ldots, v_n$ so that every v_i with $i \geq 2$ has a unique neighbor in $\{v_1, \ldots, v_{i-1}\}$.

Corollary: A connected graph with n vertices is a tree if and only if it has $n - 1$ edges.
Proof: Exercise, use induction on n

Observations on Trees

Corollary: Every tree has at least two vertices with degree 1.
Proof: Take any longest simple path $x_0, \ldots, x_m$ in T. Both x_0 and x_m must have degree 1: otherwise there is s a longer path in T.

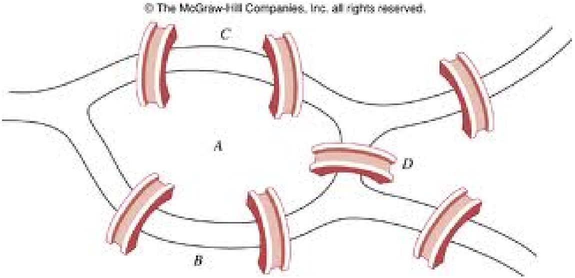
Corollary: The vertices of a tree can always be enumerated, say as $v_1, \ldots, v_n$ so that every v_i with $i \geq 2$ has a unique neighbor in $\{v_1, \ldots, v_{i-1}\}$.

Corollary: A connected graph with n vertices is a tree if and only if it has $n-1$ edges.
Proof: Exercise, use induction on n

Corollary: If T is a tree and G is any graph with $\delta(G) \geq |T| - 1$, then $T \subset G$, that is G has a subgraph isomorphic to T.
Proof Construct T by using a greedy algorithm.

The Königsberg Bridge Problem

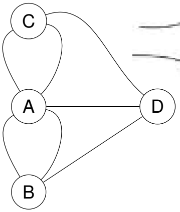
Leonard Euler (1707-1783) was asked to solve the following:

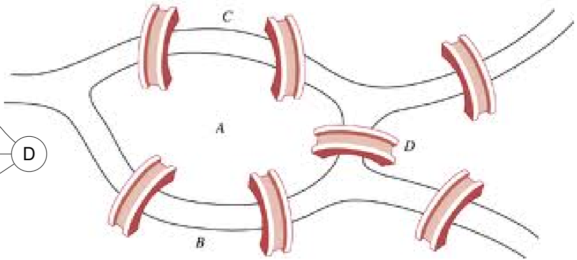


Question: Can you start a walk somewhere in Königsberg, walk across each of the 7 bridges exactly once, and end up back where you started from?

The Königsberg Bridge Problem

Leonard Euler (1707-1783) was asked to solve the following:





Question: Can you start a walk somewhere in Königsberg, walk across each of the 7 bridges exactly once, and end up back where you started from?

Euler (in 1736) used “graph theory” to answer this question.

Characterization of Eulerian Graphs

Theorem

A connnected graph G is Eulerian if and only if all of its vertices have even degree.

Proof

Necessity: Add direction to each edge as it is visited along the Eulerian circuit. Since each vertex has indegree equal to its outdegree, the degree (undirected) of each vertex is even.

Characterization of Eulerian Graphs

Theorem

A connnected graph G is Eulerian if and only if all of its vertices have even degree.

Proof

Necessity: Add direction to each edge as it is visited along the Eulerian circuit. Since each vertex has indegree equal to its outdegree, the degree (undirected) of each vertex is even.

Sufficiency: Proof by induction on the number of edges, m.

- Basis step: $m = 0$, true.

Characterization of Eulerian Graphs

Theorem

A connnected graph G is Eulerian if and only if all of its vertices have even degree.

Proof

Necessity: Add direction to each edge as it is visited along the Eulerian circuit. Since each vertex has indegree equal to its outdegree, the degree (undirected) of each vertex is even.

Sufficiency: Proof by induction on the number of edges, m.

- Basis step: $m = 0$, true.
- Key Lemma: (Lemma 1.2.25): If every vertex of a graph G has degree at least 2, then G contains a cycle.
- Since every vertex in G has even degree, every vertex has degree at least 2 in every component of G. By the key lemma, there is a cycle in each component of G.

Characterization of Eulerian Graphs

Theorem

A connnected graph G is Eulerian if and only if all of its vertices have even degree.

Proof

Necessity: Add direction to each edge as it is visited along the Eulerian circuit. Since each vertex has indegree equal to its outdegree, the degree (undirected) of each vertex is even.

Sufficiency: Proof by induction on the number of edges, m.

- Basis step: $m = 0$, true.
- Key Lemma: (Lemma 1.2.25): If every vertex of a graph G has degree at least 2, then G contains a cycle.
- Since every vertex in G has even degree, every vertex has degree at least 2 in every component of G. By the key lemma, there is a cycle in each component of G.
- So, pick a cycle in one component, call it C. Let $G' := G - C$.

Characterization of Eulerian Graphs

Theorem

A connnected graph G is Eulerian if and only if all of its vertices have even degree.

Proof

Necessity: Add direction to each edge as it is visited along the Eulerian circuit. Since each vertex has indegree equal to its outdegree, the degree (undirected) of each vertex is even.

Sufficiency: Proof by induction on the number of edges, m.

- Basis step: $m = 0$, true.
- Key Lemma: (Lemma 1.2.25): If every vertex of a graph G has degree at least 2, then G contains a cycle.
- Since every vertex in G has even degree, every vertex has degree at least 2 in every component of G. By the key lemma, there is a cycle in each component of G.
- So, pick a cycle in one component, call it C. Let $G' := G - C$.
- By inductive hypothesis, every component of G' has an Euler circuit D.
- D, together with C is an Euler circuit of G.

Conclusions

Corollary

Every even graph (all degree even) decomposes into cycles.

Conclusions

Corollary

Every even graph (all degree even) decomposes into cycles.

Proposition

If for a graph G, $\delta(G) \geq k$, then G contains a P_{k+1}. If $k \geq 2$, then G also contains a cycle of length at least $k+1$.