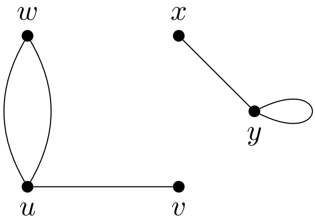
A graph is a set of objects, or vertices, together with a (multi)set of edges that connect pairs of vertices. (Think driving routes between cities, or social connections between people.)

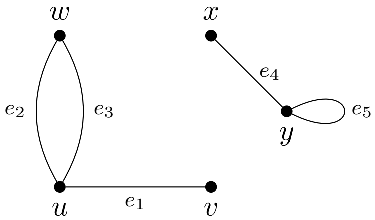
A graph is a set of objects, or vertices, together with a (multi)set of edges that connect pairs of vertices. (Think driving routes between cities, or social connections between people.)

Example:



A graph is a set of objects, or vertices, together with a (multi)set of edges that connect pairs of vertices. (Think driving routes between cities, or social connections between people.)

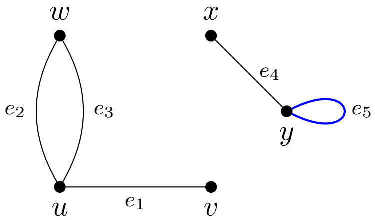
Example:



Here, the vertices are $V = \{u, v, w, x, y\}$, and the edges are
$E = \{e_1 = u{-}v,\ e_2 = u{-}w,\ e_3 = u{-}w,\ e_4 = x{-}y,\ e_5 = y{-}y\}$.

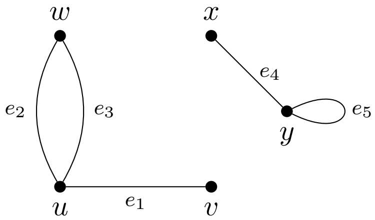
A graph is a set of objects, or vertices, together with a (multi)set of edges that connect pairs of vertices. (Think driving routes between cities, or social connections between people.)

Example:



Here, the vertices are $V = \{u, v, w, x, y\}$, and the edges are $E = \{e_1 = u{-}v,\ e_2 = u{-}w,\ e_3 = u{-}w,\ e_4 = x{-}y,\ e_5 = y{-}y\}$. An edge that connects a vertex to itself (like e_5) is called a loop.

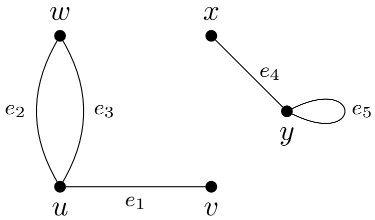
A graph is a set of objects, or vertices, together with a (multi)set of edges that connect pairs of vertices. (Think driving routes between cities, or social connections between people.)

Example: 

Here, the vertices are $V = \{u, v, w, x, y\}$, and the edges are $E = \{e_1 = u{-}v,\ e_2 = u{-}w,\ e_3 = u{-}w,\ e_4 = x{-}y,\ e_5 = y{-}y\}$.
An edge that connects a vertex to itself (like e_5) is called a loop.
We say a vertex a is adjacent to a vertex b if there is an edge connecting a and b.

A graph is a set of objects, or vertices, together with a (multi)set of edges that connect pairs of vertices. (Think driving routes between cities, or social connections between people.)

Example:



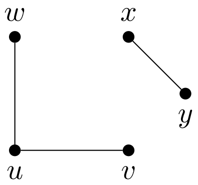
Here, the vertices are $V = \{u, v, w, x, y\}$, and the edges are $E = \{e_1 = u{-}v,\ e_2 = u{-}w,\ e_3 = u{-}w,\ e_4 = x{-}y,\ e_5 = y{-}y\}$. An edge that connects a vertex to itself (like e_5) is called a loop. We say a vertex a is adjacent to a vertex b if there is an edge connecting a and b. (Notice that for a generic graph, "adjacency" is a symmetric relation, but is not reflexive nor is it transitive.)

Classes of graphs:

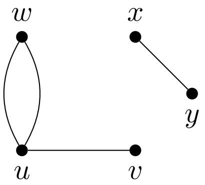
A graph is simple if there are no loops and every pair of vertices has at most one edge between them.

Classes of graphs:

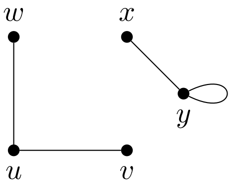
A graph is simple if there are no loops and every pair of vertices has at most one edge between them.



Simple!



NOT simple!



NOT simple!

Classes of graphs:

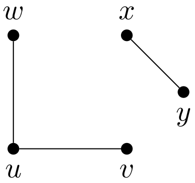
A graph is simple if there are no loops and every pair of vertices has at most one edge between them.

A graph is a multigraph if there are no loops, but there could be multiple edges between two vertices.

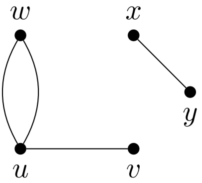
Classes of graphs:

A graph is simple if there are no loops and every pair of vertices has at most one edge between them.

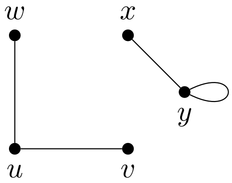
A graph is a multigraph if there are no loops, but there could be multiple edges between two vertices.



Multigraph!



Multigraph!



NOT Multigraph!

Classes of graphs:

A graph is simple if there are no loops and every pair of vertices has at most one edge between them.

A graph is a multigraph if there are no loops, but there could be multiple edges between two vertices.

A graph is a pseudograph if there could be loops or multiple edges. (This is just what we call a graph.)

Classes of graphs:

A graph is simple if there are no loops and every pair of vertices has at most one edge between them.

A graph is a multigraph if there are no loops, but there could be multiple edges between two vertices.

A graph is a pseudograph if there could be loops or multiple edges. (This is just what <u>we</u> call a graph.)

So

$$\{ \text{ pseudographs/graphs } \} \supsetneq \{ \text{ multigraphs } \} \supsetneq \{ \text{ simple graphs } \}.$$

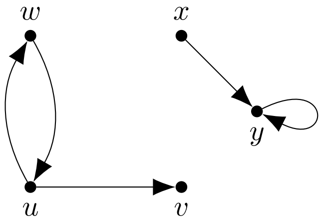
(Note: The $\supsetneq$ symbol is used here because, for example, every simple graph is a multigraph, but there are multigraphs that are not simple.)

Directed graphs

A directed graph (also called a digraph or a quiver) is a graph, together with a choice of direction for each edge. (Think flights from one city to the other, or a flow chart.)

Directed graphs

A directed graph (also called a digraph or a quiver) is a graph, together with a choice of direction for each edge. (Think flights from one city to the other, or a flow chart.) For example,



Directed graphs

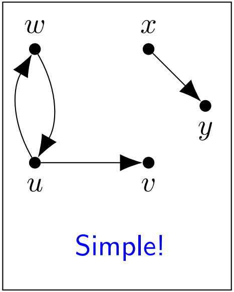
A directed graph (also called a digraph or a quiver) is a graph, together with a choice of direction for each edge. (Think flights from one city to the other, or a flow chart.)

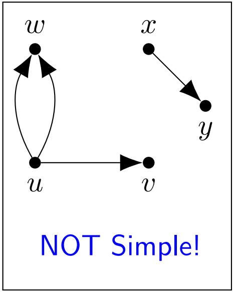
A directed graph is simple if there are no loops and every pair of vertices has at most one edge in each direction between them.

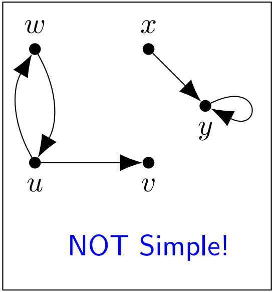
Directed graphs

A directed graph (also called a digraph or a quiver) is a graph, together with a choice of direction for each edge. (Think flights from one city to the other, or a flow chart.)

A directed graph is simple if there are no loops and every pair of vertices has at most one edge in each direction between them.







Directed graphs

A directed graph (also called a digraph or a quiver) is a graph, together with a choice of direction for each edge. (Think flights from one city to the other, or a flow chart.)

A directed graph is simple if there are no loops and every pair of vertices has at most one edge in each direction between them.

A directed graph is a directed multigraph if there could be loops or multiple edges. (This is just what we call a directed graph)

Directed graphs

A directed graph (also called a digraph or a quiver) is a graph, together with a choice of direction for each edge. (Think flights from one city to the other, or a flow chart.)

A directed graph is simple if there are no loops and every pair of vertices has at most one edge in each direction between them.

A directed graph is a directed multigraph if there could be loops or multiple edges. (This is just what we call a directed graph)

So

$$\{ \text{ directed (multi)graphs } \} \supsetneq \{ \text{ directed simple graphs } \}.$$

Directed graphs

A directed graph (also called a digraph or a quiver) is a graph, together with a choice of direction for each edge. (Think flights from one city to the other, or a flow chart.)

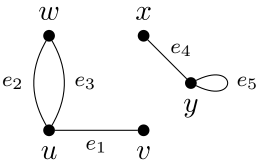
A directed graph is simple if there are no loops and every pair of vertices has at most one edge in each direction between them.

A directed graph is a directed multigraph if there could be loops or multiple edges. (This is just what we call a directed graph)

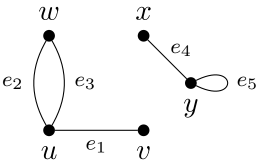
So

$$\{\text{ directed (multi)graphs }\} \supsetneq \{\text{ directed simple graphs }\}.$$

The book also talks about mixed graphs, where some of the edges are directed and some aren't. We usually take care of this by modeling the non-directed edges with *two directed edges*, one in each direction.

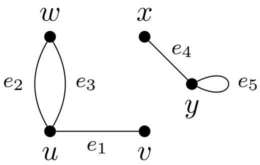
$G =$ 

We say a vertex a is adjacent to a vertex b if there is an edge connecting a and b.

$G =$ 

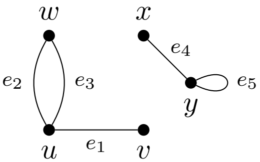
We say a vertex a is adjacent to a vertex b if there is an edge connecting a and b. For example, in G,

u is adjacent to

$G =$ 

We say a vertex a is adjacent to a vertex b if there is an edge connecting a and b. For example, in G,

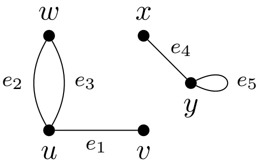
u is adjacent to w and v;

$G =$ 

We say a vertex a is adjacent to a vertex b if there is an edge connecting a and b. For example, in G,

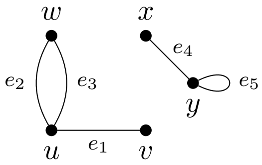
u is adjacent to w and v; v is adjacent to

$G =$



We say a vertex a is adjacent to a vertex b if there is an edge connecting a and b. For example, in G,

u is adjacent to w and v; v is adjacent to u;

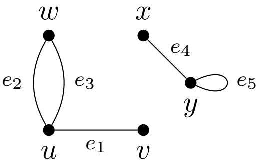
$G =$ 

We say a vertex a is adjacent to a vertex b if there is an edge connecting a and b. For example, in G,

u is adjacent to w and v; v is adjacent to u;

y is adjacent to

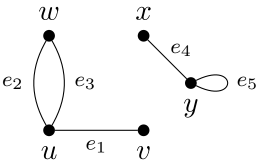
$G =$



We say a vertex a is adjacent to a vertex b if there is an edge connecting a and b. For example, in G,

u is adjacent to w and v; v is adjacent to u;

y is adjacent to x and y.

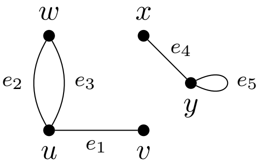
$G =$ 

We say a vertex a is adjacent to a vertex b if there is an edge connecting a and b. For example, in G,

u is adjacent to w and v; v is adjacent to u;

y is adjacent to x and y.

We say that an edge is incident to a vertex if the edge connects to the vertex.

$G =$ 

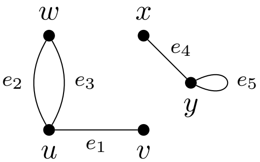
We say a vertex a is adjacent to a vertex b if there is an edge connecting a and b. For example, in G,

u is adjacent to w and v; v is adjacent to u;

y is adjacent to x and y.

We say that an edge is incident to a vertex if the edge connects to the vertex. For example, in G,

e_1 is incident to

$G =$ 

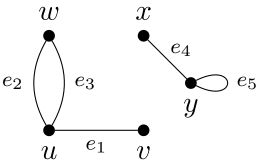
We say a vertex a is adjacent to a vertex b if there is an edge connecting a and b. For example, in G,

u is adjacent to w and v; v is adjacent to u;

y is adjacent to x and y.

We say that an edge is incident to a vertex if the edge connects to the vertex. For example, in G,

e_1 is incident to u and v;

$G =$ 

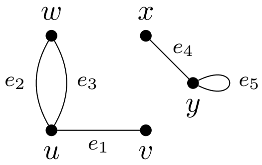
We say a vertex a is adjacent to a vertex b if there is an edge connecting a and b. For example, in G,

u is adjacent to w and v; v is adjacent to u;

y is adjacent to x and y.

We say that an edge is incident to a vertex if the edge connects to the vertex. For example, in G,

e_1 is incident to u and v; e_5 is incident to

$G =$ 

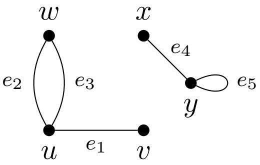
We say a vertex a is adjacent to a vertex b if there is an edge connecting a and b. For example, in G,

u is adjacent to w and v; v is adjacent to u;
y is adjacent to x and y.

We say that an edge is incident to a vertex if the edge connects to the vertex. For example, in G,

e_1 is incident to u and v; e_5 is incident to y.

$G =$



We say a vertex a is adjacent to a vertex b if there is an edge connecting a and b. For example, in G,

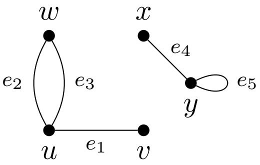
u is adjacent to w and v; v is adjacent to u;

y is adjacent to x and y.

We say that an edge is incident to a vertex if the edge connects to the vertex. For example, in G,

e_1 is incident to u and v; e_5 is incident to y.

If two vertices u and v are adjacent, we we say that they are neighbors, and that u is in the neighborhood $N(v)$ of v (and vice-versa).

$G =$ 

$N(u) = \{v, w\}$
$N(v) = \{u\}$
$N(w) = \{u\}$
$N(x) = \{y\}$
$N(y) = \{x, y\}$

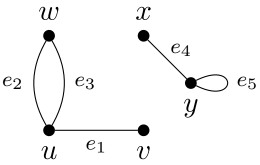
We say a vertex a is adjacent to a vertex b if there is an edge connecting a and b. For example, in G,

u is adjacent to w and v; v is adjacent to u;
y is adjacent to x and y.

We say that an edge is incident to a vertex if the edge connects to the vertex. For example, in G,

e_1 is incident to u and v; e_5 is incident to y.

If two vertices u and v are adjacent, we we say that they are neighbors, and that u is in the neighborhood $N(v)$ of v (and vice-versa).

$G =$ 

$N(u) = \{v, w\}$
$N(v) = \{u\}$
$N(w) = \{u\}$
$N(x) = \{y\}$
$N(y) = \{x, y\}$

We say a vertex a is adjacent to a vertex b if there is an edge connecting a and b. For example, in G,

u is adjacent to w and v; v is adjacent to u;

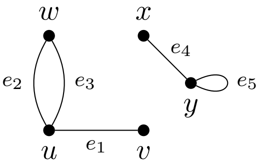
y is adjacent to x and y.

We say that an edge is incident to a vertex if the edge connects to the vertex. For example, in G,

e_1 is incident to u and v; e_5 is incident to y.

If two vertices u and v are adjacent, we we say that they are neighbors, and that u is in the neighborhood $N(v)$ of v (and vice-versa). If $A \subseteq V$, then

$$N(A) = \bigcup_{v \in A} N(v).$$

$G =$ 

$$N(u) = \{v, w\}$$
$$N(v) = \{u\}$$
$$N(w) = \{u\}$$
$$N(x) = \{y\}$$
$$N(y) = \{x, y\}$$

$$N(\{u, v\}) = \{u, v, w\}$$
$$N(\{u, y\}) = \{v, w, x, y\}$$

We say a vertex a is adjacent to a vertex b if there is an edge connecting a and b. For example, in G,

u is adjacent to w and v; v is adjacent to u;
y is adjacent to x and y.

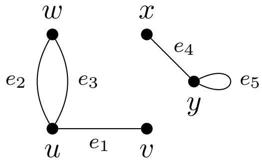
We say that an edge is incident to a vertex if the edge connects to the vertex. For example, in G,

e_1 is incident to u and v; e_5 is incident to y.

If two vertices u and v are adjacent, we we say that they are neighbors, and that u is in the neighborhood $N(v)$ of v (and vice-versa). If $A \subseteq V$, then

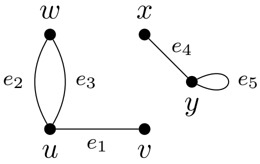
$$N(A) = \bigcup_{v \in A} N(v).$$

$G =$



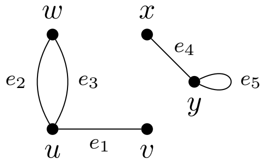
$$
\begin{aligned}
N(u) &= \{v, w\} \\
N(v) &= \{u\} \\
N(w) &= \{u\} \\
N(x) &= \{y\} \\
N(y) &= \{x, y\}
\end{aligned}
$$

The degree $\deg(v)$ of a vertex v is the number of edge ends attached to v.

$G =$ 

$$
\begin{array}{ll}
N(u) = \{v, w\} & \deg(u) = 3 \\
N(v) = \{u\} & \deg(v) = 1 \\
N(w) = \{u\} & \deg(w) = 2 \\
N(x) = \{y\} & \deg(x) = 1 \\
N(y) = \{x, y\} & \deg(y) = 3
\end{array}
$$

The degree $\deg(v)$ of a vertex v is the number of edge ends attached to v.

$G =$ 

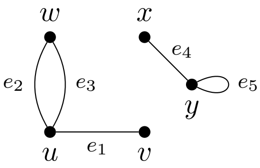
$$
\begin{aligned}
N(u) &= \{v, w\} & \deg(u) &= 3\\
N(v) &= \{u\} & \deg(v) &= 1\\
N(w) &= \{u\} & \deg(w) &= 2\\
N(x) &= \{y\} & \deg(x) &= 1\\
N(y) &= \{x, y\} & \deg(y) &= 3
\end{aligned}
$$

The degree $\deg(v)$ of a vertex v is the number of edge ends attached to v.

Fact: $\deg(v) \geqslant |N(v)|$; and a graph is simple if and only if

$$\deg(v) = |N(v)| \text{ for all } v \in V.$$

$G =$



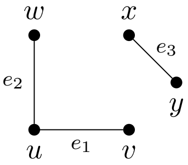
$$\begin{aligned} N(u) &= \{v, w\} & \deg(u) &= 3 \\ N(v) &= \{u\} & \deg(v) &= 1 \\ N(w) &= \{u\} & \deg(w) &= 2 \\ N(x) &= \{y\} & \deg(x) &= 1 \\ N(y) &= \{x, y\} & \deg(y) &= 3 \end{aligned}$$

The degree $\deg(v)$ of a vertex v is the number of edge ends attached to v.

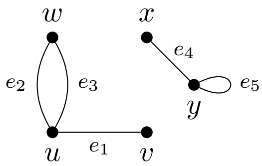
Fact: $\deg(v) \geqslant |N(v)|$; and a graph is simple if and only if

$$\deg(v) = |N(v)| \text{ for all } v \in V.$$

$H =$

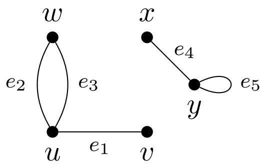


$$\begin{aligned} N(u) &= \{v, w\} & \deg(u) &= 2 \\ N(v) &= \{u\} & \deg(v) &= 1 \\ N(w) &= \{u\} & \deg(w) &= 1 \\ N(x) &= \{y\} & \deg(x) &= 1 \\ N(y) &= \{x\} & \deg(y) &= 1 \end{aligned}$$

$G =$ 

$$
\begin{array}{ll}
N(u) = \{v, w\} & \deg(u) = 3 \\
N(v) = \{u\} & \deg(v) = 1 \\
N(w) = \{u\} & \deg(w) = 2 \\
N(x) = \{y\} & \deg(x) = 1 \\
N(y) = \{x, y\} & \deg(y) = 3
\end{array}
$$

The degree $\deg(v)$ of a vertex v is the number of edge ends attached to v. We call a graph regular if all the vertices have the same degree.

$G =$ 

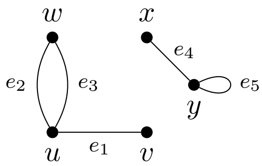
$N(u) = \{v, w\}$	$\deg(u) = 3$
$N(v) = \{u\}$	$\deg(v) = 1$
$N(w) = \{u\}$	$\deg(w) = 2$
$N(x) = \{y\}$	$\deg(x) = 1$
$N(y) = \{x, y\}$	$\deg(y) = 3$

The degree $\deg(v)$ of a vertex v is the number of edge ends attached to v. We call a graph regular if all the vertices have the same degree.

Theorem (The handshake theorem)

In a graph $G = (V, E)$*,*

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

$G =$ 

$$\begin{array}{ll} N(u) = \{v, w\} & \deg(u) = 3 \\ N(v) = \{u\} & \deg(v) = 1 \\ N(w) = \{u\} & \deg(w) = 2 \\ N(x) = \{y\} & \deg(x) = 1 \\ N(y) = \{x, y\} & \deg(y) = 3 \end{array}$$

The degree $\deg(v)$ of a vertex v is the number of edge ends attached to v. We call a graph regular if all the vertices have the same degree.

Theorem (The handshake theorem)

In a graph $G = (V, E)$,

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

Corollary

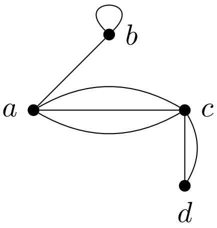
In any graph, there are an even number of odd vertices.

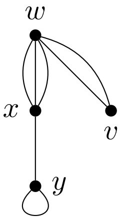
Graph isomorphisms

We say two graphs G and G' are isomorphic if there is a relabeling of the vertices of G that transforms it into G'. In other words, there is a bijection

$$f : V \to V'$$

such that the induced map on E is a bijection $f : E \to E'$.
For example,

$G =$ 

and $G' =$ 

are isomorphic via the map (doesn't depend on the drawing)

$$a \mapsto x, \quad b \mapsto y, \quad c \mapsto x, \quad d \mapsto v.$$

Recall: an equivalence relation on a set $\mathcal{A}$ is a pairing $\sim$ that is reflexive ($a \sim a$), symmetric ($a \sim b$ iff $b \sim a$), and transitive ($a \sim b$ and $b \sim c$ implies $a \sim c$).

Recall: an equivalence relation on a set $\mathcal{A}$ is a pairing $\sim$ that is reflexive ($a \sim a$), symmetric ($a \sim b$ iff $b \sim a$), and transitive ($a \sim b$ and $b \sim c$ implies $a \sim c$). Given an equivalence relation, an equivalence class is a maximal set of things that are pairwise equivalent.

Recall: an equivalence relation on a set $\mathcal{A}$ is a pairing $\sim$ that is reflexive ($a \sim a$), symmetric ($a \sim b$ iff $b \sim a$), and transitive ($a \sim b$ and $b \sim c$ implies $a \sim c$). Given an equivalence relation, an equivalence class is a maximal set of things that are pairwise equivalent. Here, if $\mathcal{G}$ is the set of all graphs, then

$$G \sim H \quad \text{whenever} \quad G \text{ is isomorphic to } H$$

is an equivalence relation.

Recall: an equivalence relation on a set $\mathcal{A}$ is a pairing $\sim$ that is reflexive ($a \sim a$), symmetric ($a \sim b$ iff $b \sim a$), and transitive ($a \sim b$ and $b \sim c$ implies $a \sim c$). Given an equivalence relation, an equivalence class is a maximal set of things that are pairwise equivalent. Here, if $\mathcal{G}$ is the set of all graphs, then

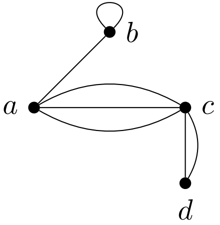
$$G \sim H \quad \text{whenever} \quad G \text{ is isomorphic to } H$$

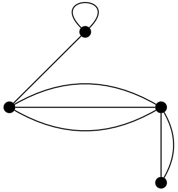
is an equivalence relation. For an equivalence class of graphs, we draw the associated unlabeled graph.

Recall: an equivalence relation on a set $\mathcal{A}$ is a pairing $\sim$ that is reflexive ($a \sim a$), symmetric ($a \sim b$ iff $b \sim a$), and transitive ($a \sim b$ and $b \sim c$ implies $a \sim c$). Given an equivalence relation, an equivalence class is a maximal set of things that are pairwise equivalent. Here, if $\mathcal{G}$ is the set of all graphs, then

$$G \sim H \quad \text{whenever} \quad G \text{ is isomorphic to } H$$

is an equivalence relation. For an equivalence class of graphs, we draw the associated unlabeled graph. For example, the equivalence class of graphs corresponding to

$G =$ 

is  .

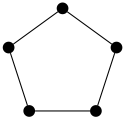
Special graphs

Cycles. A cycle C_n is the equivalence class of simple graphs on n vertices $\{v_1, v_2, \ldots, v_n\}$ so that v_i is adjacent to $v_{i\pm1}$ (v_1 is adjacent to v_n).

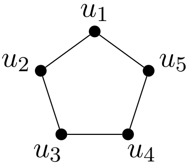
Special graphs

Cycles. A cycle C_n is the equivalence class of simple graphs on n vertices $\{v_1, v_2, \ldots, v_n\}$ so that v_i is adjacent to $v_{i\pm 1}$ (v_1 is adjacent to v_n).

equivalence class C_5



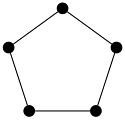
one graph in the class C_5



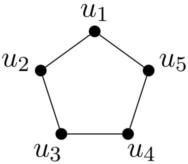
Special graphs

Cycles. A cycle C_n is the equivalence class of simple graphs on n vertices $\{v_1, v_2, \ldots, v_n\}$ so that v_i is adjacent to $v_{i\pm1}$ (v_1 is adjacent to v_n).

equivalence class C_5



one graph in the class C_5

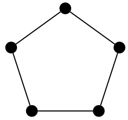


Wheels. A wheel W_n is the cycle C_n together with an additional vertex that is adjacent to every other vertex.

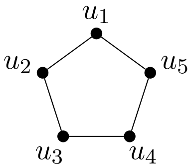
Special graphs

Cycles. A cycle C_n is the equivalence class of simple graphs on n vertices $\{v_1, v_2, \ldots, v_n\}$ so that v_i is adjacent to $v_{i\pm1}$ (v_1 is adjacent to v_n).

equivalence class C_5

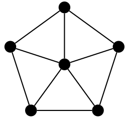


one graph in the class C_5

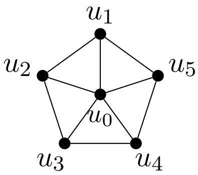


Wheels. A wheel W_n is the cycle C_n together with an additional vertex that is adjacent to every other vertex.

equivalence class W_5



one graph in the class W_5



Special graphs

Complete graphs. The complete graph on n vertices, denoted K_n, is the equivalence class of simple graphs on n vertices so that $N(v) = V - \{v\}$ for all all $v \in V$.

Special graphs

Complete graphs. The complete graph on n vertices, denoted K_n, is the equivalence class of simple graphs on n vertices so that $N(v) = V - \{v\}$ for all all $v \in V$. For example,

$K_1 = \bullet$

Special graphs

Complete graphs. The complete graph on n vertices, denoted K_n, is the equivalence class of simple graphs on n vertices so that $N(v) = V - \{v\}$ for all all $v \in V$. For example,

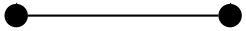
$K_1 =$ 

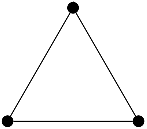
$K_2 =$ 

Special graphs

Complete graphs. The complete graph on n vertices, denoted K_n, is the equivalence class of simple graphs on n vertices so that $N(v) = V - \{v\}$ for all all $v \in V$. For example,

$K_1 =$ 

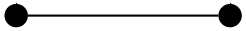
$K_2 =$ 

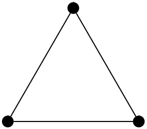
$K_3 =$ 

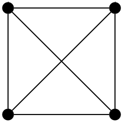
Special graphs

Complete graphs. The complete graph on n vertices, denoted K_n, is the equivalence class of simple graphs on n vertices so that $N(v) = V - \{v\}$ for all all $v \in V$. For example,

$K_1 =$ 

$K_2 =$ 

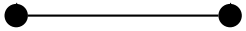
$K_3 =$ 

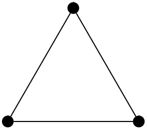
$K_4 =$ 

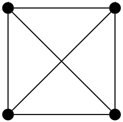
Special graphs

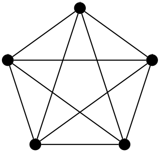
Complete graphs. The complete graph on n vertices, denoted K_n, is the equivalence class of simple graphs on n vertices so that $N(v) = V - \{v\}$ for all all $v \in V$. For example,

$K_1 =$ 

$K_2 =$ 

$K_3 =$ 

$K_4 =$ 

$K_5 =$ 

Bipartite graphs. A graph is bipartite if V can be partitioned into two nonempty subsets V_1 and V_2 so that no vertex in V_i is adjacent to any other vertex in V_i for $i = 1$ or 2.

Bipartite graphs. A graph is bipartite if V can be partitioned into two nonempty subsets V_1 and V_2 so that no vertex in V_i is adjacent to any other vertex in V_i for $i = 1$ or 2.
In particular, for any $m \geqslant n \geqslant 1$, the complete bipartite graph $K_{n,m}$ is the class of simple graphs corresponding to the graph with vertices $V = V_1 \cup V_2$, where

$$V_1 = \{v_1, \ldots, v_n\} \quad V_2 = \{u_1, \cdots u_m\}$$

$$N(v_i) = V_2 \qquad \text{and} \qquad N(u_i) = V_1$$

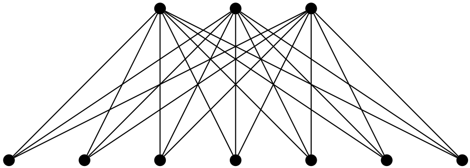
for all i.

Bipartite graphs. A graph is bipartite if V can be partitioned into two nonempty subsets V_1 and V_2 so that no vertex in V_i is adjacent to any other vertex in V_i for $i = 1$ or 2.
In particular, for any $m \geqslant n \geqslant 1$, the complete bipartite graph $K_{n,m}$ is the class of simple graphs corresponding to the graph with vertices $V = V_1 \cup V_2$, where

$$V_1 = \{v_1, \ldots, v_n\} \quad V_2 = \{u_1, \cdots u_m\}$$

$$N(v_i) = V_2 \qquad \text{and} \qquad N(u_i) = V_1$$

for all i. For example,

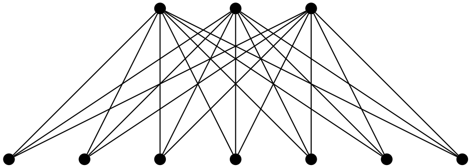
$K_{7,3} =$ 

Bipartite graphs. A graph is bipartite if V can be partitioned into two nonempty subsets V_1 and V_2 so that no vertex in V_i is adjacent to any other vertex in V_i for $i = 1$ or 2.
In particular, for any $m \geqslant n \geqslant 1$, the complete bipartite graph $K_{n,m}$ is the class of simple graphs corresponding to the graph with vertices $V = V_1 \cup V_2$, where

$$V_1 = \{v_1, \ldots, v_n\} \quad V_2 = \{u_1, \cdots u_m\}$$

$$N(v_i) = V_2 \qquad \text{and} \qquad N(u_i) = V_1$$

for all i. For example,

$K_{7,3} =$ 

One way to show that a graph is bipartite is to "color" the vertices two different colors, so that no two vertices of the same color are adjacent.

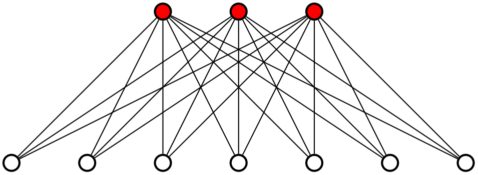
Bipartite graphs. A graph is bipartite if V can be partitioned into two nonempty subsets V_1 and V_2 so that no vertex in V_i is adjacent to any other vertex in V_i for $i = 1$ or 2.
In particular, for any $m \geqslant n \geqslant 1$, the complete bipartite graph $K_{n,m}$ is the class of simple graphs corresponding to the graph with vertices $V = V_1 \cup V_2$, where

$$V_1 = \{v_1, \ldots, v_n\} \quad V_2 = \{u_1, \cdots u_m\}$$

$$N(v_i) = V_2 \qquad \text{and} \qquad N(u_i) = V_1$$

for all i. For example,

$K_{7,3} =$



One way to show that a graph is bipartite is to "color" the vertices two different colors, so that no two vertices of the same color are adjacent.

Bipartite graphs. A graph is bipartite if V can be partitioned into two nonempty subsets V_1 and V_2 so that no vertex in V_i is adjacent to any other vertex in V_i for $i = 1$ or 2.

One way to show that a graph is bipartite is to "color" the vertices two different colors, so that no two vertices of the same color are adjacent.

Hypercubes. Let Q_n be the graph with vertex set

$$V = \{ \text{ bit strings (1's and 0's) of length } n \}$$

and edge set

$$E = \{u{-}v \mid u \text{ and } v \text{ differ in exactly one bit } \}.$$

Bipartite graphs. A graph is bipartite if V can be partitioned into two nonempty subsets V_1 and V_2 so that no vertex in V_i is adjacent to any other vertex in V_i for $i = 1$ or 2.

One way to show that a graph is bipartite is to "color" the vertices two different colors, so that no two vertices of the same color are adjacent.

Hypercubes. Let Q_n be the graph with vertex set

$$V = \{ \text{ bit strings (1's and 0's) of length } n \; \}$$

and edge set

$$E = \{u{-}v \mid u \text{ and } v \text{ differ in exactly one bit } \}.$$

Q_1

Bipartite graphs. A graph is bipartite if V can be partitioned into two nonempty subsets V_1 and V_2 so that no vertex in V_i is adjacent to any other vertex in V_i for $i = 1$ or 2.

One way to show that a graph is bipartite is to "color" the vertices two different colors, so that no two vertices of the same color are adjacent.

Hypercubes. Let Q_n be the graph with vertex set

$$V = \{ \text{ bit strings (1's and 0's) of length } n \ \}$$

and edge set

$$E = \{u{-}v \mid u \text{ and } v \text{ differ in exactly one bit } \}.$$

$Q_1 =$ 0   1

Bipartite graphs. A graph is bipartite if V can be partitioned into two nonempty subsets V_1 and V_2 so that no vertex in V_i is adjacent to any other vertex in V_i for $i = 1$ or 2.

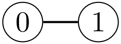
One way to show that a graph is bipartite is to "color" the vertices two different colors, so that no two vertices of the same color are adjacent.

Hypercubes. Let Q_n be the graph with vertex set

$$V = \{ \text{ bit strings (1's and 0's) of length } n \ \}$$

and edge set

$$E = \{u{-}v \mid u \text{ and } v \text{ differ in exactly one bit } \}.$$

$Q_1 =$ 

Bipartite graphs. A graph is bipartite if V can be partitioned into two nonempty subsets V_1 and V_2 so that no vertex in V_i is adjacent to any other vertex in V_i for $i = 1$ or 2.

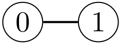
One way to show that a graph is bipartite is to "color" the vertices two different colors, so that no two vertices of the same color are adjacent.

Hypercubes. Let Q_n be the graph with vertex set

$$V = \{ \text{ bit strings (1's and 0's) of length } n \}$$

and edge set

$$E = \{u{-}v \mid u \text{ and } v \text{ differ in exactly one bit } \}.$$

$Q_1 =$ 

Q_2

Bipartite graphs. A graph is bipartite if V can be partitioned into two nonempty subsets V_1 and V_2 so that no vertex in V_i is adjacent to any other vertex in V_i for $i = 1$ or 2.

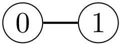
One way to show that a graph is bipartite is to "color" the vertices two different colors, so that no two vertices of the same color are adjacent.

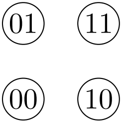
Hypercubes. Let Q_n be the graph with vertex set

$$V = \{ \text{ bit strings (1's and 0's) of length } n \ \}$$

and edge set

$$E = \{u{-}v \mid u \text{ and } v \text{ differ in exactly one bit } \}.$$

$Q_1 =$ 

$Q_2 =$ 

Bipartite graphs. A graph is bipartite if V can be partitioned into two nonempty subsets V_1 and V_2 so that no vertex in V_i is adjacent to any other vertex in V_i for $i = 1$ or 2.

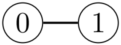
One way to show that a graph is bipartite is to "color" the vertices two different colors, so that no two vertices of the same color are adjacent.

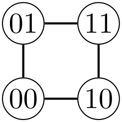
Hypercubes. Let Q_n be the graph with vertex set

$$V = \{ \text{ bit strings (1's and 0's) of length } n \ \}$$

and edge set

$$E = \{u - v \mid u \text{ and } v \text{ differ in exactly one bit } \}.$$

$Q_1 =$ 

$Q_2 =$ 

Bipartite graphs. A graph is bipartite if V can be partitioned into two nonempty subsets V_1 and V_2 so that no vertex in V_i is adjacent to any other vertex in V_i for $i = 1$ or 2.

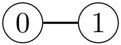
One way to show that a graph is bipartite is to "color" the vertices two different colors, so that no two vertices of the same color are adjacent.

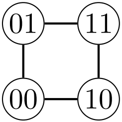
Hypercubes. Let Q_n be the graph with vertex set

$$V = \{ \text{ bit strings (1's and 0's) of length } n \}$$

and edge set

$$E = \{u{-}v \mid u \text{ and } v \text{ differ in exactly one bit } \}.$$

$Q_1 =$ 

$Q_2 =$ 

Q_3

Bipartite graphs. A graph is bipartite if V can be partitioned into two nonempty subsets V_1 and V_2 so that no vertex in V_i is adjacent to any other vertex in V_i for $i = 1$ or 2.

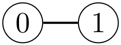
One way to show that a graph is bipartite is to “color” the vertices two different colors, so that no two vertices of the same color are adjacent.

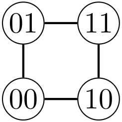
Hypercubes. Let Q_n be the graph with vertex set

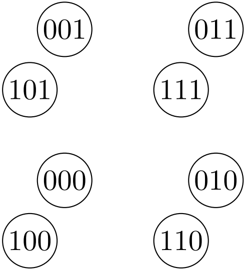
$$V = \{ \text{ bit strings (1's and 0's) of length } n \ \}$$

and edge set

$$E = \{ u{-}v \mid u \text{ and } v \text{ differ in exactly one bit } \}.$$

$Q_1 =$ 

$Q_2 =$ 

$Q_3 =$ 

Bipartite graphs. A graph is bipartite if V can be partitioned into two nonempty subsets V_1 and V_2 so that no vertex in V_i is adjacent to any other vertex in V_i for $i = 1$ or 2.

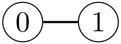
One way to show that a graph is bipartite is to "color" the vertices two different colors, so that no two vertices of the same color are adjacent.

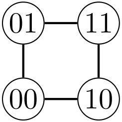
Hypercubes. Let Q_n be the graph with vertex set

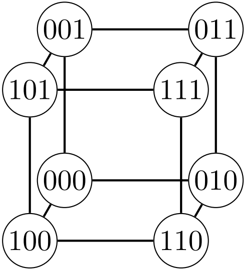
$$V = \{ \text{ bit strings (1's and 0's) of length } n \ \}$$

and edge set

$$E = \{u{-}v \mid u \text{ and } v \text{ differ in exactly one bit } \}.$$

$Q_1 =$ 

$Q_2 =$ 

$Q_3 =$ 

Bipartite graphs. A graph is bipartite if V can be partitioned into two nonempty subsets V_1 and V_2 so that no vertex in V_i is adjacent to any other vertex in V_i for $i = 1$ or 2.

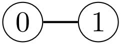
One way to show that a graph is bipartite is to "color" the vertices two different colors, so that no two vertices of the same color are adjacent.

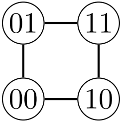
Hypercubes. Let Q_n be the graph with vertex set

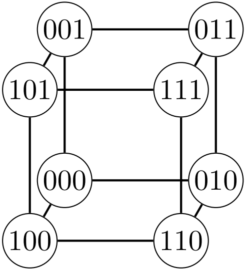
$$V = \{ \text{ bit strings (1's and 0's) of length } n \text{ } \}$$

and edge set

$$E = \{u{-}v \mid u \text{ and } v \text{ differ in exactly one bit } \}.$$

$Q_1 =$ 

$Q_2 =$ 

$Q_3 =$ 

Color vertices with an even number of 0's red.

Bipartite graphs. A graph is bipartite if V can be partitioned into two nonempty subsets V_1 and V_2 so that no vertex in V_i is adjacent to any other vertex in V_i for $i = 1$ or 2.

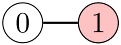
One way to show that a graph is bipartite is to "color" the vertices two different colors, so that no two vertices of the same color are adjacent.

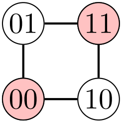
Hypercubes. Let Q_n be the graph with vertex set

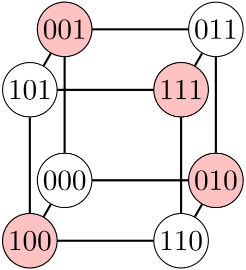
$$V = \{ \text{ bit strings (1's and 0's) of length } n \ \}$$

and edge set

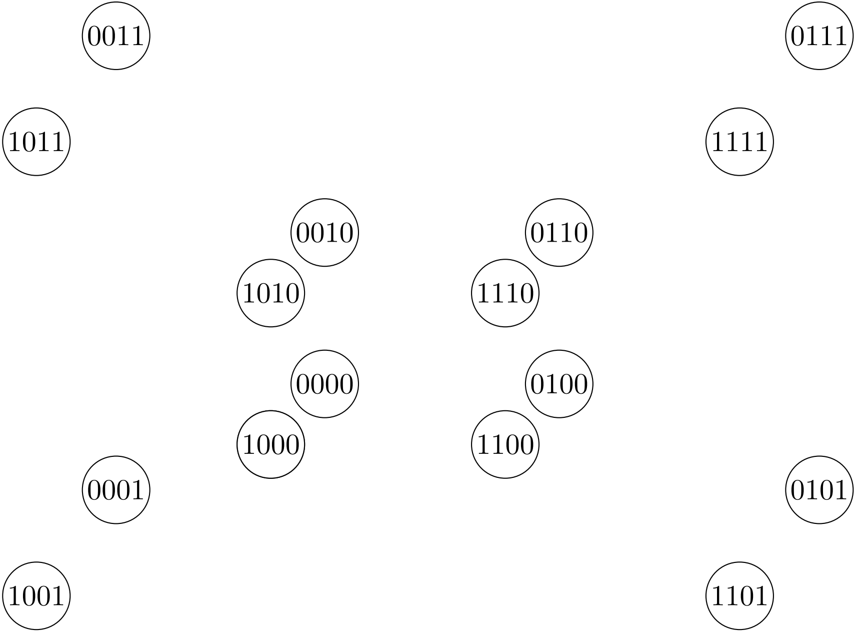
$$E = \{u{-}v \mid u \text{ and } v \text{ differ in exactly one bit } \}.$$

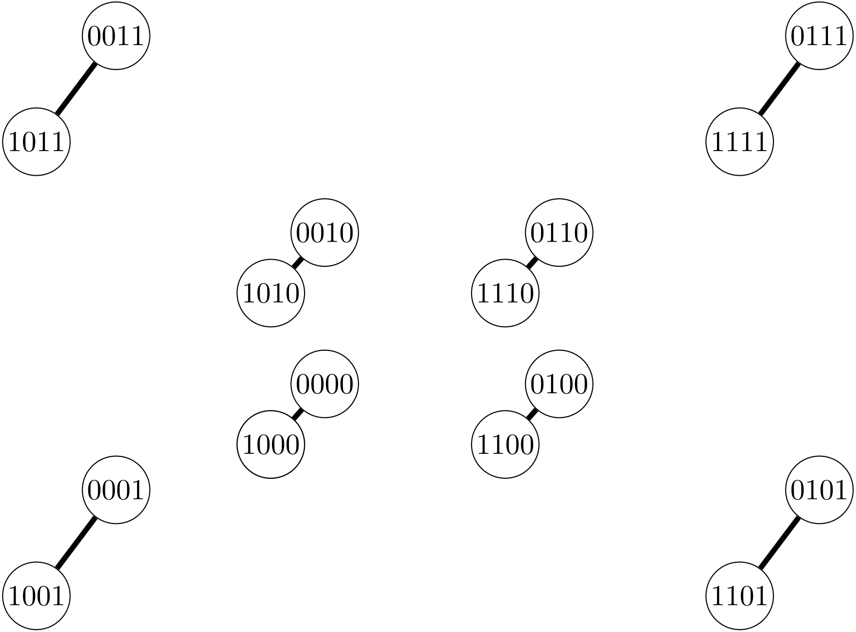
$Q_1 =$ 

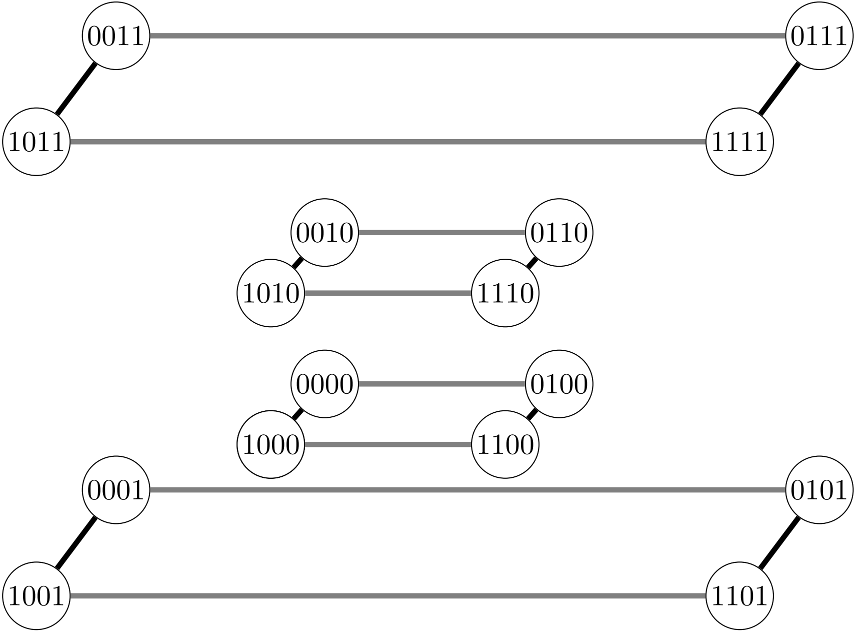
$Q_2 =$ 

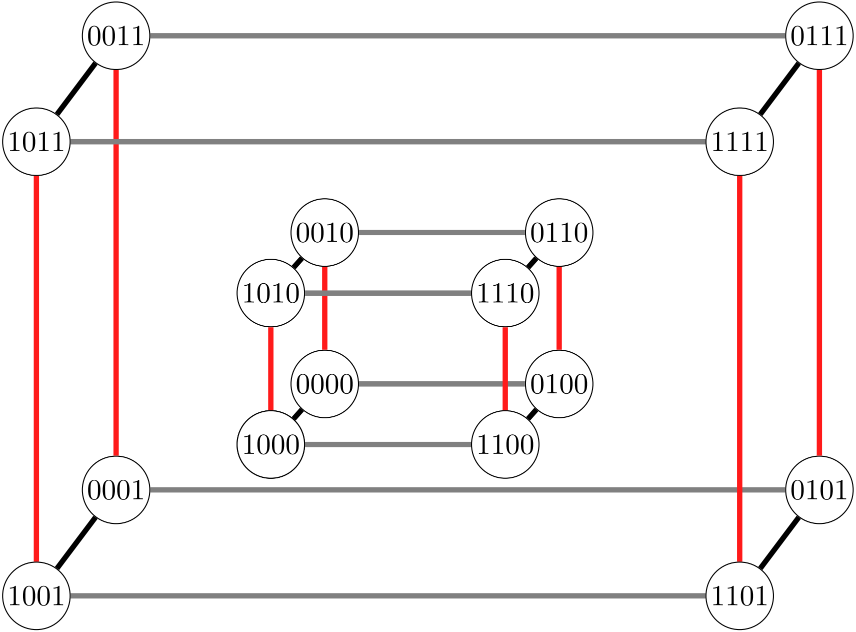
$Q_3 =$ 

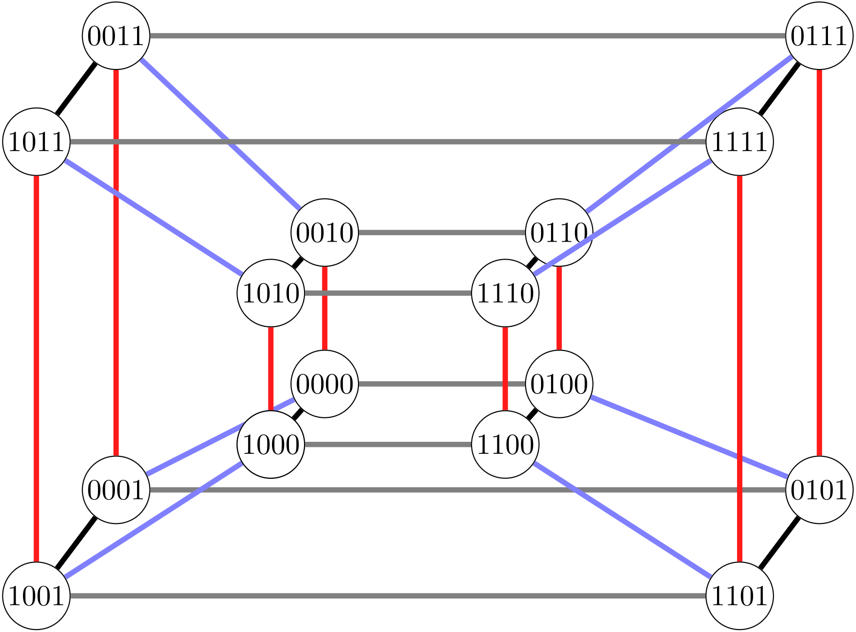
Color vertices with an even number of 0's red.


0011
0111
1011
1111
0010
0110
1010
1110
0000
0100
1000
1100
0001
0101
1001
1101


0011
1011
0111
1111
0010
1010
0110
1110
0000
1000
0100
1100
0001
1001
0101
1101


0011
0111
1011
1111
0010
0110
1010
1110
0000
0100
1000
1100
0001
0101
1001
1101


0011
0111
1011
1111
0010
0110
1010
1110
0000
0100
1000
1100
0001
0101
1001
1101


0011
0111
1011
1111
0010
0110
1010
1110
0000
0100
1000
1100
0001
0101
1001
1101

Graph invariants

To prove that two graphs are isomorphic, you need to find an isomorphism.

Graph invariants

To prove that two graphs are isomorphic, you need to find an isomorphism. To show that they're not isomorphic, you have to show that no isomorphism exists, which can be harder!

Graph invariants

To prove that two graphs are isomorphic, you need to find an isomorphism. To show that they're not isomorphic, you have to show that no isomorphism exists, which can be harder! So we look for properties of the graphs that are preserved by isomorphisms. These are called (graph) invariants.

Graph invariants

To prove that two graphs are isomorphic, you need to find an isomorphism. To show that they're not isomorphic, you have to show that no isomorphism exists, which can be harder! So we look for properties of the graphs that are preserved by isomorphisms. These are called (graph) invariants.

Example: The number of vertices in a graph is an invariant.
(If G is isomorphic to H, then there is a bijection between their vertex sets, so those vertex sets must have the same size.

Graph invariants

To prove that two graphs are isomorphic, you need to find an isomorphism. To show that they're not isomorphic, you have to show that no isomorphism exists, which can be harder! So we look for properties of the graphs that are preserved by isomorphisms. These are called (graph) invariants.

Example: The number of vertices in a graph is an invariant. (If G is isomorphic to H, then there is a bijection between their vertex sets, so those vertex sets must have the same size. Conversely, if G and H have a different number of vertices, then no such bijection exits.)

Graph invariants

To prove that two graphs are isomorphic, you need to find an isomorphism. To show that they're not isomorphic, you have to show that no isomorphism exists, which can be harder! So we look for properties of the graphs that are preserved by isomorphisms. These are called (graph) invariants.

Example: The number of vertices in a graph is an invariant. (If G is isomorphic to H, then there is a bijection between their vertex sets, so those vertex sets must have the same size. Conversely, if G and H have a different number of vertices, then no such bijection exits.)

For example, C_5 and C_6 are different isomorphism classes.

Graph invariants

To prove that two graphs are isomorphic, you need to find an isomorphism. To show that they're not isomorphic, you have to show that no isomorphism exists, which can be harder! So we look for properties of the graphs that are preserved by isomorphisms. These are called (graph) invariants.

Example: The number of vertices in a graph is an invariant. (If G is isomorphic to H, then there is a bijection between their vertex sets, so those vertex sets must have the same size. Conversely, if G and H have a different number of vertices, then no such bijection exits.)

For example, C_5 and C_6 are different isomorphism classes.

Similarly, the number of edges in a graph is an invariant.

Graph invariants

To prove that two graphs are isomorphic, you need to find an isomorphism. To show that they're not isomorphic, you have to show that no isomorphism exists, which can be harder! So we look for properties of the graphs that are preserved by isomorphisms. These are called (graph) invariants.

Example: The number of vertices in a graph is an invariant. (If G is isomorphic to H, then there is a bijection between their vertex sets, so those vertex sets must have the same size. Conversely, if G and H have a different number of vertices, then no such bijection exits.)

For example, C_5 and C_6 are different isomorphism classes.

Similarly, the number of edges in a graph is an invariant.

For example, C_5 and K_5 are different isomorphism classes.

Graph invariants

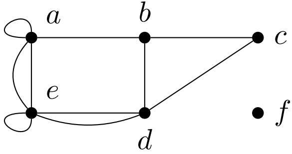
To prove that two graphs are isomorphic, you need to find an isomorphism. To show that they're not isomorphic, you have to show that no isomorphism exists, which can be harder! So we look for properties of the graphs that are preserved by isomorphisms. These are called (graph) invariants.

Example: The degree sequence of a graph is the list of degrees of vertices in the graph, given in decreasing order.

Graph invariants

To prove that two graphs are isomorphic, you need to find an isomorphism. To show that they're not isomorphic, you have to show that no isomorphism exists, which can be harder! So we look for properties of the graphs that are preserved by isomorphisms. These are called (graph) invariants.

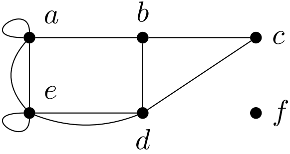
Example: The degree sequence of a graph is the list of degrees of vertices in the graph, given in decreasing order. For example, the degree sequence of

$G =$  is

Graph invariants

To prove that two graphs are isomorphic, you need to find an isomorphism. To show that they're not isomorphic, you have to show that no isomorphism exists, which can be harder! So we look for properties of the graphs that are preserved by isomorphisms. These are called (graph) invariants.

Example: The degree sequence of a graph is the list of degrees of vertices in the graph, given in decreasing order. For example, the degree sequence of

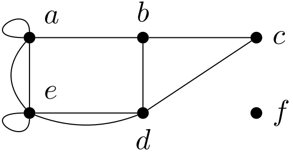
$G =$ 

is 6, 5, 4, 3, 2, 0.

Graph invariants

To prove that two graphs are isomorphic, you need to find an isomorphism. To show that they're not isomorphic, you have to show that no isomorphism exists, which can be harder! So we look for properties of the graphs that are preserved by isomorphisms. These are called (graph) invariants.

Example: The degree sequence of a graph is the list of degrees of vertices in the graph, given in decreasing order. For example, the degree sequence of

$G =$ 

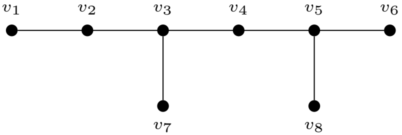
is 6, 5, 4, 3, 2, 0.

(Again, if the degree sequences of G and H differ, then $G \ncong H$. But if the degree sequences match, the *might* be isomorphic, but they *might not be*.)

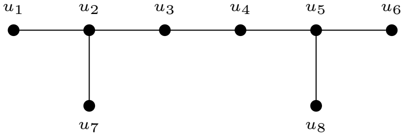
Graph invariants

For example, consider the graphs

$G =$

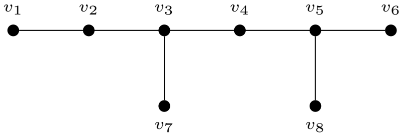


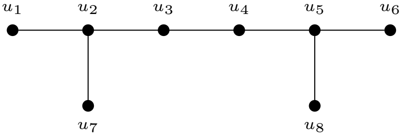
$H =$



Graph invariants

For example, consider the graphs

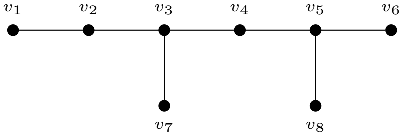
$G =$ 

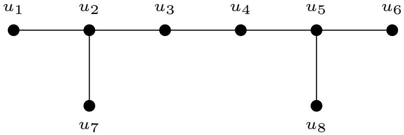
$H =$ 

Both of these graphs have the degree sequence $3, 3, 2, 2, 1, 1, 1, 1$.

Graph invariants

For example, consider the graphs

$G =$ 

$H =$ 

Both of these graphs have the degree sequence $3, 3, 2, 2, 1, 1, 1, 1$. But in G, there's a vertex of degree 1 adjacent to a vertex of degree 2, where as no vertex of degree 1 is adjacent to a vertex of degree 2 in H. So $G \ncong H$.