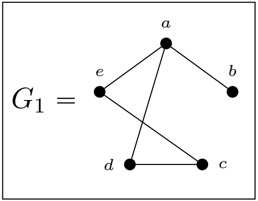
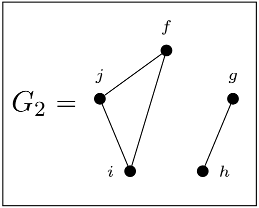
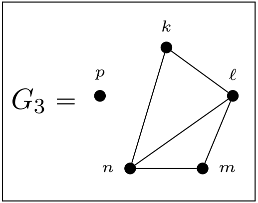
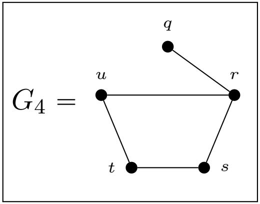
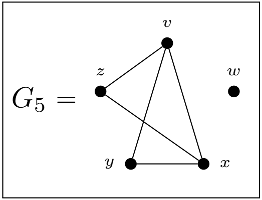
Warmup: Find which pairs of the following graphs are isomorphic. For any two graphs that are isomorphic, give an isomorphism.

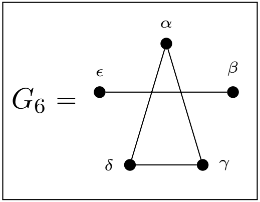












New graphs from old

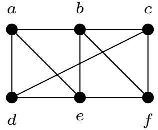
Let $G = (V, E)$ be a simple graph.

New graphs from old

Let $G = (V, E)$ be a simple graph. A subgraph of a graph $G = (V, E)$ is a graph $H = (W, F)$ such that $W \subseteq V$ and $F \subseteq E$.

New graphs from old

Let $G = (V, E)$ be a simple graph. A subgraph of a graph $G = (V, E)$ is a graph $H = (W, F)$ such that $W \subseteq V$ and $F \subseteq E$. For example, let

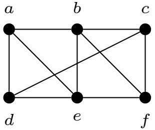
$G =$ 

Some subgraphs include:

New graphs from old

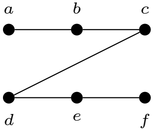
Let $G = (V, E)$ be a simple graph. A subgraph of a graph $G = (V, E)$ is a graph $H = (W, F)$ such that $W \subseteq V$ and $F \subseteq E$. For example, let

$G =$



Some subgraphs include:

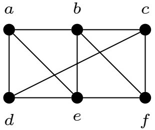
$H_1 =$



New graphs from old

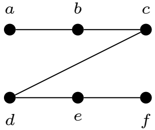
Let $G = (V, E)$ be a simple graph. A subgraph of a graph $G = (V, E)$ is a graph $H = (W, F)$ such that $W \subseteq V$ and $F \subseteq E$. For example, let

$G =$

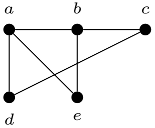


Some subgraphs include:

$H_1 =$

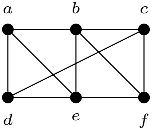


$H_2 =$

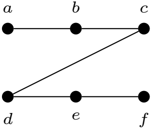


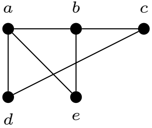
New graphs from old

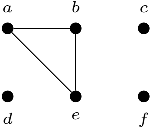
Let $G = (V, E)$ be a simple graph. A subgraph of a graph $G = (V, E)$ is a graph $H = (W, F)$ such that $W \subseteq V$ and $F \subseteq E$. For example, let

$G =$ 

Some subgraphs include:

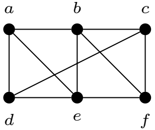
$H_1 =$ 

$H_2 =$ 

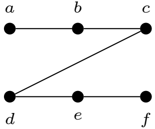
$H_3 =$ 

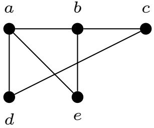
New graphs from old

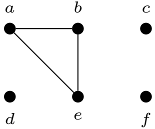
Let $G = (V, E)$ be a simple graph. A subgraph of a graph $G = (V, E)$ is a graph $H = (W, F)$ such that $W \subseteq V$ and $F \subseteq E$. For example, let

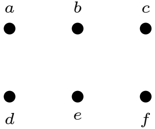
$G =$ 

Some subgraphs include:

$H_1 =$ 

$H_2 =$ 

$H_3 =$ 

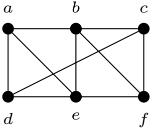
$H_4 =$ 

New graphs from old

Let $G = (V, E)$ be a simple graph. A subgraph of a graph $G = (V, E)$ is a graph $H = (W, F)$ such that $W \subseteq V$ and $F \subseteq E$.

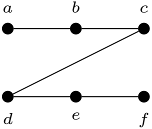
For example, let

$G =$

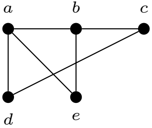


Some subgraphs include:

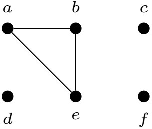
$H_1 =$



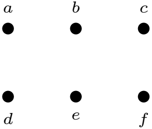
$H_2 =$



$H_3 =$



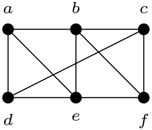
$H_4 =$



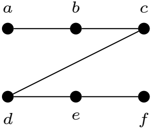
$H_5 =$

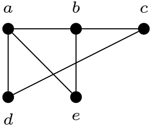
New graphs from old

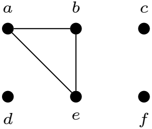
Let $G = (V, E)$ be a simple graph. A subgraph of a graph $G = (V, E)$ is a graph $H = (W, F)$ such that $W \subseteq V$ and $F \subseteq E$. For example, let

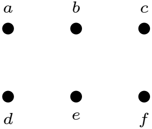
$G =$ 

Some subgraphs include:

$H_1 =$ 

$H_2 =$ 

$H_3 =$ 

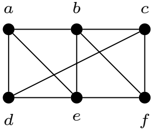
$H_4 =$ 

$H_5 =$ 

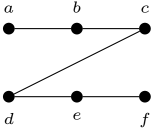
$H_6 = \varnothing$

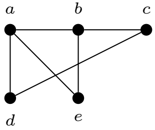
New graphs from old

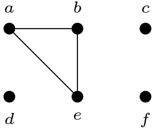
Let $G = (V, E)$ be a simple graph. A subgraph of a graph $G = (V, E)$ is a graph $H = (W, F)$ such that $W \subseteq V$ and $F \subseteq E$. For example, let

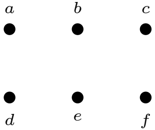
$G =$ 

Some subgraphs include:

$H_1 =$ 

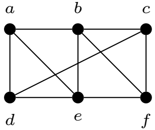
$H_2 =$ 

$H_3 =$ 

$H_4 =$ 

$H_5 =$ 

$H_6 = \varnothing$

$H_7 =$ 

New graphs from old

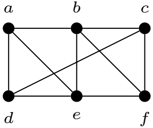
Let $G = (V, E)$ be a simple graph.

For $v \in V$, the graph $G - v$ is the graph made by deleting v and any edge incident to v.

New graphs from old

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

For $v \in V$, the graph $G - v$ is the graph made by deleting v and any edge incident to v. For example, if

$G =$ 

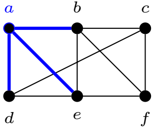
then

$$G - a$$

New graphs from old

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

For $v \in V$, the graph $G - v$ is the graph made by deleting v and any edge incident to v. For example, if

$G =$ 

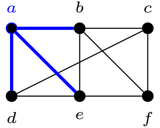
then

$$G - a$$

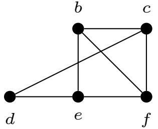
New graphs from old

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

For $v \in V$, the graph $G - v$ is the graph made by deleting v and any edge incident to v. For example, if

$G =$ 

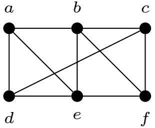
then

$G - a =$ 

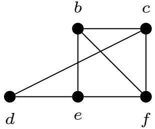
New graphs from old

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

For $v \in V$, the graph $G - v$ is the graph made by deleting v and any edge incident to v. For example, if

$G =$ 

then

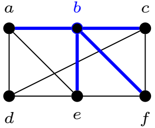
$G - a =$ 

$G - b$

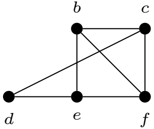
New graphs from old

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

For $v \in V$, the graph $G - v$ is the graph made by deleting v and any edge incident to v. For example, if

$G =$ 

then

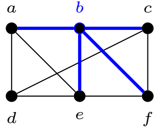
$G - a =$ 

$G - b$

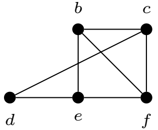
New graphs from old

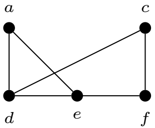
Let $G = (V, E)$ be a simple graph.

For $v \in V$, the graph $G - v$ is the graph made by deleting v and any edge incident to v. For example, if

$G =$ 

then

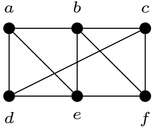
$G - a =$ 

$G - b =$ 

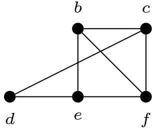
New graphs from old

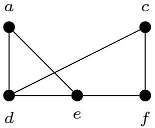
Let $G = (V, E)$ be a simple graph.

For $v \in V$, the graph $G - v$ is the graph made by deleting v and any edge incident to v. For example, if

$G =$ 

then

$G - a =$ 

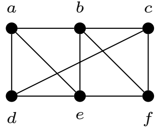
$G - b =$ 

Let $W \subseteq V$. The subgraph induced by W , denoted $G[W]$, is the subgraph made by deleting everything not in W.

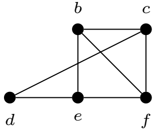
New graphs from old

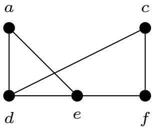
Let $G = (V, E)$ be a simple graph.

For $v \in V$, the graph $G - v$ is the graph made by deleting v and any edge incident to v. For example, if

$G =$ 

then

$G - a =$ 

$G - b =$ 

Let $W \subseteq V$. The subgraph induced by W , denoted $G[W]$, is the subgraph made by deleting everything not in W. For example, $G[\{b, c, d, e, f\}] = G - a$.

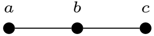
Counting subgraphs

Counting subgraphs

Step 1: Break into cases based on the vertex set.

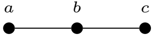
Counting subgraphs

Step 1: Break into cases based on the vertex set.

For example, let $G =$ 

Counting subgraphs

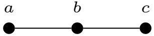
Step 1: Break into cases based on the vertex set.

For example, let $G =$ 

G hase vertex set $V = V_G = \{a, b, c\}$.

Counting subgraphs

Step 1: Break into cases based on the vertex set.

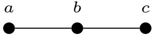
For example, let $G =$ 

G hase vertex set $V = V_G = \{a, b, c\}$.

If $H \subseteq G$, then $V_H \subseteq \{a, b, c\}$.

Counting subgraphs

Step 1: Break into cases based on the vertex set.

For example, let $G =$ 

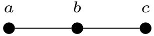
G hase vertex set $V = V_G = \{a, b, c\}$.

If $H \subseteq G$, then $V_H \subseteq \{a, b, c\}$. Possibilities:

$$\emptyset, \quad \{a\}, \quad \{b\}, \quad \{c\}, \quad \{a, b\},$$
$$\{a, c\}, \quad \{b, c\}, \quad \text{or} \quad \{a, b, c\}.$$

Counting subgraphs

Step 1: Break into cases based on the vertex set.

For example, let $G =$ 

G hase vertex set $V = V_G = \{a, b, c\}$.

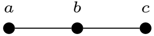
If $H \subseteq G$, then $V_H \subseteq \{a, b, c\}$. Possibilities:

$$\emptyset, \quad \{a\}, \quad \{b\}, \quad \{c\}, \quad \{a, b\},$$
$$\{a, c\}, \quad \{b, c\}, \quad \text{or} \quad \{a, b, c\}.$$

Step 2: Draw the induced graphs for each vertex subset.

Counting subgraphs

Step 1: Break into cases based on the vertex set.

For example, let $G =$ 

G hase vertex set $V = V_G = \{a, b, c\}$.

If $H \subseteq G$, then $V_H \subseteq \{a, b, c\}$. Possibilities:

$$\emptyset, \quad \{a\}, \quad \{b\}, \quad \{c\}, \quad \{a, b\},$$
$$\{a, c\}, \quad \{b, c\}, \quad \text{or} \quad \{a, b, c\}.$$

Step 2: Draw the induced graphs for each vertex subset.

In our example,

$$G[\emptyset] = \emptyset,$$

Counting subgraphs

Step 1: Break into cases based on the vertex set.

For example, let $G = \overset{a}{\bullet}\!\!-\!\!\!-\!\!\!-\!\!\overset{b}{\bullet}\!\!-\!\!\!-\!\!\!-\!\!\overset{c}{\bullet}$

G hase vertex set $V = V_G = \{a, b, c\}$.

If $H \subseteq G$, then $V_H \subseteq \{a, b, c\}$. Possibilities:

$$\emptyset, \quad \{a\}, \quad \{b\}, \quad \{c\}, \quad \{a, b\},$$
$$\{a, c\}, \quad \{b, c\}, \quad \text{or} \quad \{a, b, c\}.$$

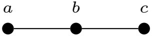
Step 2: Draw the induced graphs for each vertex subset.

In our example,

$$G[\emptyset] = \emptyset, \quad G[\{a\}] = \overset{a}{\bullet}, \quad G[\{b\}] = \overset{b}{\bullet}, \quad G[\{c\}] = \overset{c}{\bullet},$$

Counting subgraphs

Step 1: Break into cases based on the vertex set.

For example, let $G =$ 

G hase vertex set $V = V_G = \{a, b, c\}$.

If $H \subseteq G$, then $V_H \subseteq \{a, b, c\}$. Possibilities:

$$\emptyset, \quad \{a\}, \quad \{b\}, \quad \{c\}, \quad \{a, b\},$$
$$\{a, c\}, \quad \{b, c\}, \quad \text{or} \quad \{a, b, c\}.$$

Step 2: Draw the induced graphs for each vertex subset.

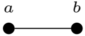
In our example,

$G[\emptyset] = \emptyset$, $G[\{a\}] =$ 

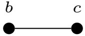
, $G[\{b\}] =$ 

, $G[\{c\}] =$ 

,

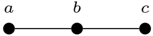
$G[\{a, b\}] =$ 

, $G[\{a, c\}] =$ 

, $G[\{b, c\}] =$ 

Counting subgraphs

Step 1: Break into cases based on the vertex set.

For example, let $G =$ 

G hase vertex set $V = V_G = \{a, b, c\}$.

If $H \subseteq G$, then $V_H \subseteq \{a, b, c\}$. Possibilities:

$$\emptyset, \quad \{a\}, \quad \{b\}, \quad \{c\}, \quad \{a, b\},$$
$$\{a, c\}, \quad \{b, c\}, \quad \text{or} \quad \{a, b, c\}.$$

Step 2: Draw the induced graphs for each vertex subset.

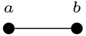
In our example,

$G[\emptyset] = \emptyset$, $G[\{a\}] =$ 

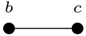
, $G[\{b\}] =$ 

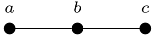
, $G[\{c\}] =$ 

,

$G[\{a, b\}] =$ 

, $G[\{a, c\}] =$ 

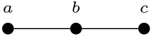
, $G[\{b, c\}] =$ 

$G[\{a, b, c\}] = G =$ 

.

Counting subgraphs

Step 1: Break into cases based on the vertex set.

For example, let $G =$ 

G hase vertex set $V = V_G = \{a, b, c\}$.

If $H \subseteq G$, then $V_H \subseteq \{a, b, c\}$. Possibilities:

$$\varnothing, \quad \{a\}, \quad \{b\}, \quad \{c\}, \quad \{a,b\},$$
$$\{a,c\}, \quad \{b,c\}, \quad \text{or} \quad \{a,b,c\}.$$

Step 2: Draw the induced graphs for each vertex subset.

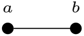
In our example,

$G[\varnothing] = \varnothing$, $G[\{a\}] =$ 

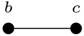
, $G[\{b\}] =$ 

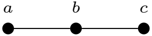
, $G[\{c\}] =$ 

,

$G[\{a,b\}] =$ 

, $G[\{a,c\}] =$ 

, $G[\{b,c\}] =$ 

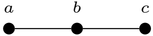
$G[\{a,b,c\}] = G =$ 

.

Step 3: Count the number of subgraphs of the induced graphs that have the same vertex set, but possibly fewer edges.

Counting subgraphs

Step 1: Break into cases based on the vertex set.

For example, let $G =$ 

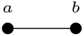
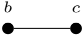
G hase vertex set $V = V_G = \{a, b, c\}$.
If $H \subseteq G$, then $V_H \subseteq \{a, b, c\}$. Possibilities:

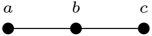
$$\varnothing, \quad \{a\}, \quad \{b\}, \quad \{c\}, \quad \{a, b\},$$
$$\{a, c\}, \quad \{b, c\}, \quad \text{or} \quad \{a, b, c\}.$$

Step 2: Draw the induced graphs for each vertex subset.

In our example,

$G[\varnothing] = \varnothing$, $G[\{a\}] =$ , $G[\{b\}] =$ , $G[\{c\}] =$ ,

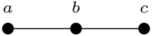
$G[\{a, b\}] =$ , $G[\{a, c\}] =$ , $G[\{b, c\}] =$ 

$G[\{a, b, c\}] = G =$ .

Step 3: Count the number of subgraphs of the induced graphs that have the same vertex set, but possibly fewer edges. This reduces to looking at each edge and deciding to keep it or lose it.

Counting subgraphs

Step 1: Break into cases based on the vertex set.

For example, let $G =$ 

G hase vertex set $V = V_G = \{a, b, c\}$.

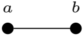
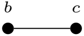
If $H \subseteq G$, then $V_H \subseteq \{a, b, c\}$. Possibilities:

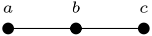
$$\emptyset, \quad \{a\}, \quad \{b\}, \quad \{c\}, \quad \{a,b\},$$
$$\{a,c\}, \quad \{b,c\}, \quad \text{or} \quad \{a,b,c\}.$$

Step 2: Draw the induced graphs for each vertex subset.

In our example,

$G[\emptyset] = \emptyset$, $G[\{a\}] =$ , $G[\{b\}] =$ , $G[\{c\}] =$ ,

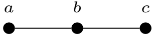
$G[\{a,b\}] =$ , $G[\{a,c\}] =$ , $G[\{b,c\}] =$ 

$G[\{a,b,c\}] = G =$ .

Step 3: Count the number of subgraphs of the induced graphs that have the same vertex set, but possibly fewer edges. This reduces to looking at each edge and deciding to keep it or lose it. So there are $2^{\#\text{edges}}$ such subgraphs.

Counting subgraphs

Step 1: Break into cases based on the vertex set.

For example, let $G =$ 

G hase vertex set $V = V_G = \{a, b, c\}$.

If $H \subseteq G$, then $V_H \subseteq \{a, b, c\}$. Possibilities:

$$\varnothing, \quad \{a\}, \quad \{b\}, \quad \{c\}, \quad \{a,b\},$$
$$\{a,c\}, \quad \{b,c\}, \quad \text{or} \quad \{a,b,c\}.$$

Step 2: Draw the induced graphs for each vertex subset.

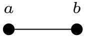
In our example,

$G[\varnothing] = \varnothing$, $G[\{a\}] =$ 

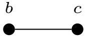
, $G[\{b\}] =$ 

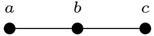
, $G[\{c\}] =$ 

,

$G[\{a,b\}] =$ 

, $G[\{a,c\}] =$ 

, $G[\{b,c\}] =$ 

$G[\{a,b,c\}] = G =$ 

.

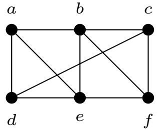
Step 3: Count the number of subgraphs of the induced graphs that have the same vertex set, but possibly fewer edges. This reduces to looking at each edge and deciding to keep it or lose it. So there are $2^{\#\text{edges}}$ such subgraphs. $1 + 1 + 1 + 1 + 2 + 1 + 2 + 2^2 = \boxed{13}$

Edge operations.

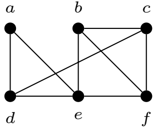
Subtraction: Let $\epsilon \in E$. Then $G - \epsilon$ is the subgraph of G with vertex set V and edge set $E - \{\epsilon\}$.

Edge operations.

Subtraction: Let $\epsilon \in E$. Then $G - \epsilon$ is the subgraph of G with vertex set V and edge set $E - \{\epsilon\}$.
For example, if

$G =$ 

then

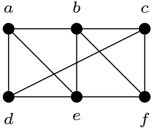
$G - (a{-}b) =$ 

Edge operations.

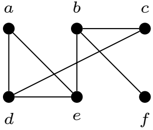
If $F \subseteq E$, the graph G is the subgraph with vertex set V and edge set $E - F$.

Edge operations.

If $F \subseteq E$, the graph G is the subgraph with vertex set V and edge set $E - F$.
For example, if

$G =$ 

then

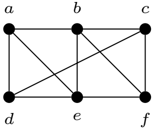
$G - \{a{-}b, c{-}f, e{-}f\} =$ 

Edge operations.

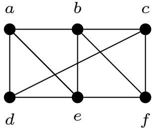
Addition: For an edge ϵ on the vertex set V but not in E, $G + \epsilon$ is the graph containing G satisfying $(G + \epsilon) - \epsilon = G$.

Edge operations.

Addition: For an edge ϵ on the vertex set V but not in E, $G + \epsilon$ is the graph containing G satisfying $(G + \epsilon) - \epsilon = G$. For example, if

$G =$ 

then

$G + (a{-}e) =$ 

Edge operations.

Let $\epsilon \in E$. There are two kinds of "contraction" of ϵ: contraction of graphs (allowing for multiple edges and loops) and contraction of simple graphs (not allowing for multiple edges and loops).

Edge operations.

Let $\epsilon \in E$. There are two kinds of "contraction" of ϵ: contraction of graphs (allowing for multiple edges and loops) and contraction of simple graphs (not allowing for multiple edges and loops).

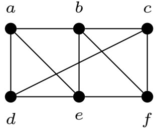
Contraction of an edge in graphs: If we're considering all graphs, then the graph G/ϵ is the graph obtained by contracting the edge ϵ, which means merging the vertices that are incident to ϵ.

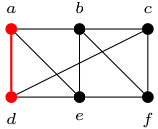
Edge operations.

Let $\epsilon \in E$. There are two kinds of "contraction" of ϵ: contraction of graphs (allowing for multiple edges and loops) and contraction of simple graphs (not allowing for multiple edges and loops).

Contraction of an edge in graphs: If we're considering all graphs, then the graph G/ϵ is the graph obtained by contracting the edge ϵ, which means merging the vertices that are incident to ϵ.

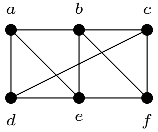
For example, if

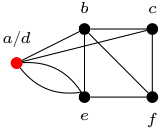
$G =$  and $\epsilon = a{-}d$, then

$G/\epsilon =$ 

Edge operations.

Let $\epsilon \in E$. There are two kinds of “contraction” of ϵ: contraction of graphs (allowing for multiple edges and loops) and contraction of simple graphs (not allowing for multiple edges and loops).
Contraction of an edge in graphs: If we're considering all graphs, then the graph G/ϵ is the graph obtained by contracting the edge ϵ, which means merging the vertices that are incident to ϵ.
For example, if

$G =$  and $\epsilon = a{-}d$, then

$G/\epsilon =$ 

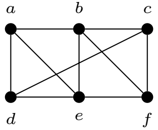
Edge operations.

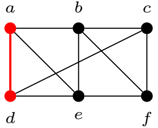
Let $\epsilon \in E$. There are two kinds of "contraction" of ϵ: contraction of graphs (allowing for multiple edges and loops) and contraction of simple graphs (not allowing for multiple edges and loops).

Contraction of an edge in simple graphs: If we're considering only simple graphs, then the graph G/ϵ is the graph obtained by contracting the edge ϵ, and then deleting any loops or multiple edges.

Edge operations.

Let $\epsilon \in E$. There are two kinds of "contraction" of ϵ: contraction of graphs (allowing for multiple edges and loops) and contraction of simple graphs (not allowing for multiple edges and loops).
Contraction of an edge in simple graphs: If we're considering only simple graphs, then the graph G/ϵ is the graph obtained by contracting the edge ϵ, and then deleting any loops or multiple edges.
For example, if

$G =$  and $\epsilon = a - d$, then

$(G/\epsilon)_{\text{simple}} =$ 

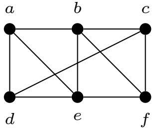
Edge operations.

Let $\epsilon \in E$. There are two kinds of "contraction" of ϵ: contraction of graphs (allowing for multiple edges and loops) and contraction of simple graphs (not allowing for multiple edges and loops).

Contraction of an edge in simple graphs: If we're considering only simple graphs, then the graph G/ϵ is the graph obtained by contracting the edge ϵ, and then deleting any loops or multiple edges.

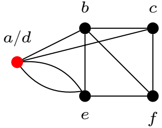
For example, if

$G =$



and $\epsilon = a{-}d$, then

$(G/\epsilon)_{\text{simple}} =$

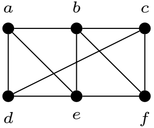


Edge operations.

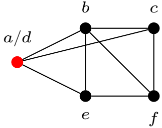
Let $\epsilon \in E$. There are two kinds of "contraction" of ϵ: contraction of graphs (allowing for multiple edges and loops) and contraction of simple graphs (not allowing for multiple edges and loops).

Contraction of an edge in simple graphs: If we're considering only simple graphs, then the graph G/ϵ is the graph obtained by contracting the edge ϵ, and then deleting any loops or multiple edges.

For example, if

$G =$ 

and $\epsilon = a - d$, then

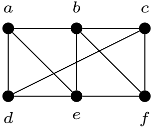
$(G/\epsilon)_{\text{simple}} =$ 

Edge operations.

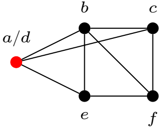
Let $\epsilon \in E$. There are two kinds of "contraction" of ϵ: contraction of graphs (allowing for multiple edges and loops) and contraction of simple graphs (not allowing for multiple edges and loops).

Contraction of an edge in simple graphs: If we're considering only simple graphs, then the graph G/ϵ is the graph obtained by contracting the edge ϵ, and then deleting any loops or multiple edges.

For example, if

$G =$ 

and $\epsilon = a{-}d$, then

$(G/\epsilon)_{\text{simple}} =$ 

Note that G/e and $(G/\epsilon)_{\text{simple}}$ are not in general subgraphs of G.

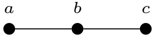
Unions

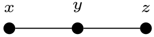
The union of two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is

$$G_1 \cup G_2 = (V_1 \cup V_2, E_1 \cup E_2).$$

Examples:

If

$G_1 =$ 

and $G_2 =$ 

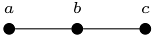
Unions

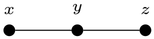
The union of two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is

$$G_1 \cup G_2 = (V_1 \cup V_2, E_1 \cup E_2).$$

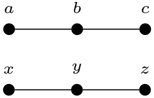
Examples:

If

$G_1 =$ 

and $G_2 =$ 

then

$G_1 \cup G_2 =$ 

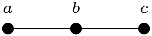
Unions

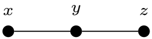
The union of two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is

$$G_1 \cup G_2 = (V_1 \cup V_2, E_1 \cup E_2).$$

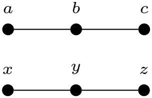
Examples:

If

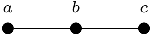
$G_1 =$ 

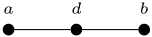
and $G_2 =$ 

then

$G_1 \cup G_2 =$ 

If

$G_1 =$ 

and $G_2 =$ 

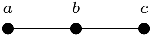
Unions

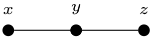
The union of two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is

$$G_1 \cup G_2 = (V_1 \cup V_2, E_1 \cup E_2).$$

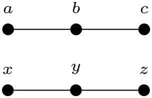
Examples:

If

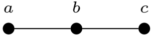
$G_1 =$ 

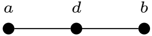
and $G_2 =$ 

then

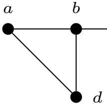
$G_1 \cup G_2 =$ 

If

$G_1 =$ 

and $G_2 =$ 

then

$G_1 \cup G_2 =$ 

Complements

Consider G as a subgraph of $K[V]$, the complete graph on the vertex set V. The complement of the graph G is

$$\bar{G} = (V, E_{K[V]} - E).$$

In other words, G and $\bar{G}$ have the same vertex set, but u and v are adjacent in $\bar{G}$ if and only if u and v are not adjacent in G.

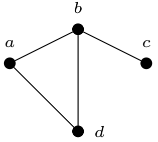
Complements

Consider G as a subgraph of $K[V]$, the complete graph on the vertex set V. The complement of the graph G is

$$\bar{G} = (V, E_{K[V]} - E).$$

In other words, G and $\bar{G}$ have the same vertex set, but u and v are adjacent in $\bar{G}$ if and only if u and v are not adjacent in G.

Example: Let

$G =$ 

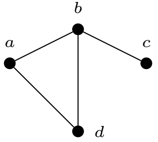
Complements

Consider G as a subgraph of $K[V]$, the complete graph on the vertex set V. The complement of the graph G is

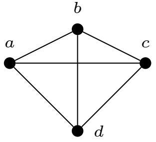
$$\bar{G} = (V, E_{K[V]} - E).$$

In other words, G and $\bar{G}$ have the same vertex set, but u and v are adjacent in $\bar{G}$ if and only if u and v are not adjacent in G.

Example: Let

$G =$ 

Then

$K[V] =$ 

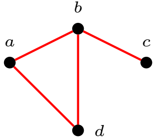
Complements

Consider G as a subgraph of $K[V]$, the complete graph on the vertex set V. The complement of the graph G is

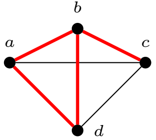
$$\bar{G} = (V, E_{K[V]} - E).$$

In other words, G and $\bar{G}$ have the same vertex set, but u and v are adjacent in $\bar{G}$ if and only if u and v are not adjacent in G.

Example: Let

$G =$ 

Then

$K[V] =$ 

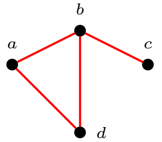
Complements

Consider G as a subgraph of $K[V]$, the complete graph on the vertex set V. The complement of the graph G is

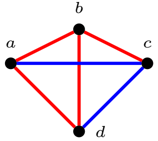
$$\bar{G} = (V, E_{K[V]} - E).$$

In other words, G and $\bar{G}$ have the same vertex set, but u and v are adjacent in $\bar{G}$ if and only if u and v are not adjacent in G.

Example: Let

$G =$ 

Then

$K[V] =$ 

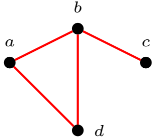
Complements

Consider G as a subgraph of $K[V]$, the complete graph on the vertex set V. The complement of the graph G is

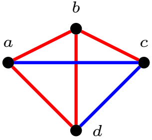
$$\bar{G} = (V, E_{K[V]} - E).$$

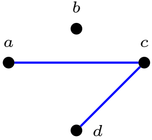
In other words, G and $\bar{G}$ have the same vertex set, but u and v are adjacent in $\bar{G}$ if and only if u and v are not adjacent in G.

Example: Let

$G =$ 

Then

$K[V] =$ 

so $\overline{G} =$ 

Connectedness

Let $G = (V, E)$ be a graph. A walk is an alternating sequence of vertices and edges

$$w = (v_0, e_1, v_1, e_2, \cdots, e_n, v_n)$$

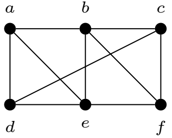
such that e_i has endpoints v_{i-1} and v_i. We say w has length n.

Connectedness

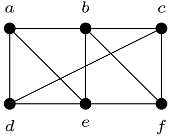
Let $G = (V, E)$ be a graph. A walk is an alternating sequence of vertices and edges

$$w = (v_0, e_1, v_1, e_2, \cdots, e_n, v_n)$$

such that e_i has endpoints v_{i-1} and v_i. We say w has length n. For example, if

$G =$  ,

the walk
looks like

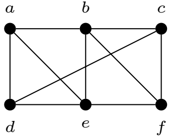


Connectedness

Let $G = (V, E)$ be a graph. A walk is an alternating sequence of vertices and edges

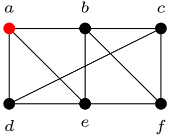
$$w = (v_0, e_1, v_1, e_2, \cdots, e_n, v_n)$$

such that e_i has endpoints v_{i-1} and v_i. We say w has length n. For example, if

$G =$  ,

the walk $(a$

looks like

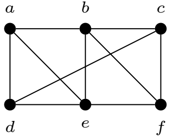


Connectedness

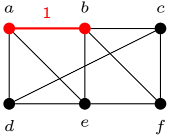
Let $G = (V, E)$ be a graph. A walk is an alternating sequence of vertices and edges

$$w = (v_0, e_1, v_1, e_2, \cdots, e_n, v_n)$$

such that e_i has endpoints v_{i-1} and v_i. We say w has length n. For example, if

$G =$  ,

the walk $(a, a{-}b, b$
looks like

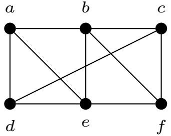


Connectedness

Let $G = (V, E)$ be a graph. A walk is an alternating sequence of vertices and edges

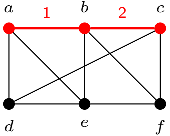
$$w = (v_0, e_1, v_1, e_2, \cdots, e_n, v_n)$$

such that e_i has endpoints v_{i-1} and v_i. We say w has length n. For example, if

$G =$ 

,

the walk $(a, a{-}b, b, b{-}c, c$
looks like

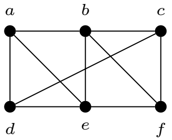


Connectedness

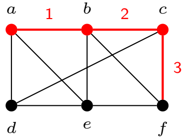
Let $G = (V, E)$ be a graph. A walk is an alternating sequence of vertices and edges

$$w = (v_0, e_1, v_1, e_2, \cdots, e_n, v_n)$$

such that e_i has endpoints v_{i-1} and v_i. We say w has length n. For example, if

$G =$  ,

the walk $(a, a{-}b, b, b{-}c, c, c{-}f, f$
looks like

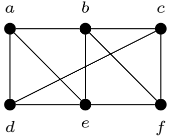


Connectedness

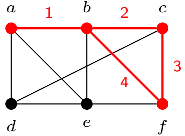
Let $G = (V, E)$ be a graph. A walk is an alternating sequence of vertices and edges

$$w = (v_0, e_1, v_1, e_2, \cdots, e_n, v_n)$$

such that e_i has endpoints v_{i-1} and v_i. We say w has length n. For example, if

$G =$ ,

the walk $(a, a{-}b, b, b{-}c, c, c{-}f, f, f{-}b, b$
looks like

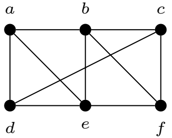


Connectedness

Let $G = (V, E)$ be a graph. A walk is an alternating sequence of vertices and edges

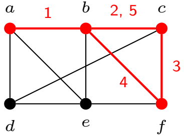
$$w = (v_0, e_1, v_1, e_2, \cdots, e_n, v_n)$$

such that e_i has endpoints v_{i-1} and v_i. We say w has length n. For example, if

$G =$ 

,

the walk $(a, a-b, b, b-c, c, c-f, f, f-b, b, b-c, c$
looks like

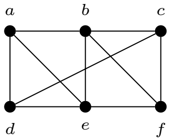


Connectedness

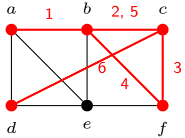
Let $G = (V, E)$ be a graph. A walk is an alternating sequence of vertices and edges

$$w = (v_0, e_1, v_1, e_2, \cdots, e_n, v_n)$$

such that e_i has endpoints v_{i-1} and v_i. We say w has length n. For example, if

$G =$  ,

the walk $(a, a{-}b, b, b{-}c, c, c{-}f, f, f{-}b, b, b{-}c, c, c{-}d, d)$ looks like



Special kinds of walks:

1. A closed walk or circuit is a walk where $v_0 = v_n$.

Special kinds of walks:

1. A closed walk or circuit is a walk where $v_0 = v_n$.
2. A path is a walk so that no vertices (and therefore edges) are repeated.

Special kinds of walks:

1. A closed walk or circuit is a walk where $v_0 = v_n$.
2. A path is a walk so that no vertices (and therefore edges) are repeated.
3. A cycle is a walk where $v_0 = v_n$ but no other vertices are repeated.

Special kinds of walks:

1. A closed walk or circuit is a walk where $v_0 = v_n$.
2. A path is a walk so that no vertices (and therefore edges) are repeated.
3. A cycle is a walk where $v_0 = v_n$ but no other vertices are repeated.
4. A trail is a walk where no edges are repeated.

Special kinds of walks:

1. A closed walk or circuit is a walk where $v_0 = v_n$.
2. A path is a walk so that no vertices (and therefore edges) are repeated.
3. A cycle is a walk where $v_0 = v_n$ but no other vertices are repeated.
4. A trail is a walk where no edges are repeated.

Note: See the remark on p. 679 of the book to reconcile the difference between the terminology we're using and the terminology in the book!!

Special kinds of walks:

1. A closed walk or circuit is a walk where $v_0 = v_n$.
2. A path is a walk so that no vertices (and therefore edges) are repeated.
3. A cycle is a walk where $v_0 = v_n$ but no other vertices are repeated.
4. A trail is a walk where no edges are repeated.

Note: See the remark on p. 679 of the book to reconcile the difference between the terminology we're using and the terminology in the book!!

In a simple graph, the sequence of vertices determines the walk, since there's at most one edge between any two vertices.

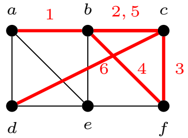
Special kinds of walks:

1. A closed walk or circuit is a walk where $v_0 = v_n$.
2. A path is a walk so that no vertices (and therefore edges) are repeated.
3. A cycle is a walk where $v_0 = v_n$ but no other vertices are repeated.
4. A trail is a walk where no edges are repeated.

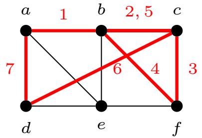
Note: See the remark on p. 679 of the book to reconcile the difference between the terminology we're using and the terminology in the book!!

In a simple graph, the sequence of vertices determines the walk, since there's at most one edge between any two vertices.

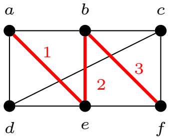
Walk:
a, b, c, f, b, c, d



Circuit:
a, b, c, f, b, c, d, a



Path:
a, e, b, f



Special kinds of walks:

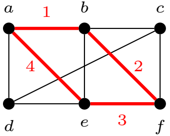
1. A closed walk or circuit is a walk where $v_0 = v_n$.
2. A path is a walk so that no vertices (and therefore edges) are repeated.
3. A cycle is a walk where $v_0 = v_n$ but no other vertices are repeated.
4. A trail is a walk where no edges are repeated.

Note: See the remark on p. 679 of the book to reconcile the difference between the terminology we're using and the terminology in the book!!

In a simple graph, the sequence of vertices determines the walk, since there's at most one edge between any two vertices.

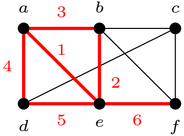
Cycle:

a, b, f, e, a



Trail:

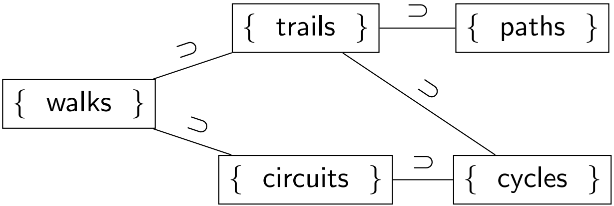
a, e, b, a, d, e, f



Special kinds of walks:

1. A closed walk or circuit is a walk where $v_0 = v_n$.
2. A path is a walk so that no vertices (and therefore edges) are repeated.
3. A cycle is a walk where $v_0 = v_n$ but no other vertices are repeated.
4. A trail is a walk where no edges are repeated.

Note:



Special kinds of walks:

1. A closed walk or circuit is a walk where $v_0 = v_n$.
2. A path is a walk so that no vertices (and therefore edges) are repeated.
3. A cycle is a walk where $v_0 = v_n$ but no other vertices are repeated.
4. A trail is a walk where no edges are repeated.

Note:

$$
\begin{tikzcd}
 & \boxed{\{\ \text{trails}\ \}} \arrow[r, dash, "\supseteq"] \arrow[ddr, dash, "\supseteq"] & \boxed{\{\ \text{paths}\ \}} \\
\boxed{\{\ \text{walks}\ \}} \arrow[ur, dash, "\supseteq"] \arrow[dr, dash, "\supseteq"'] & & \\
 & \boxed{\{\ \text{circuits}\ \}} \arrow[r, dash, "\supseteq"] & \boxed{\{\ \text{cycles}\ \}}
\end{tikzcd}
$$

A maximal path is a path that cannot be extended on either end to be a longer path.

Special kinds of walks:

1. A closed walk or circuit is a walk where $v_0 = v_n$.
2. A path is a walk so that no vertices (and therefore edges) are repeated.
3. A cycle is a walk where $v_0 = v_n$ but no other vertices are repeated.
4. A trail is a walk where no edges are repeated.

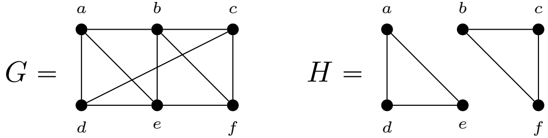
Note:

$$
\begin{tikzcd}
& \boxed{\{\text{ trails }\}} \arrow[r, dash, "\supset"] \arrow[ddr, dash, "\supset"] & \boxed{\{\text{ paths }\}} \arrow[r, dash, "\supset"] & \boxed{\{\text{ max'l paths }\}} \\
\boxed{\{\text{ walks }\}} \arrow[ur, dash, "\supset"] \arrow[dr, dash, "\supset"'] & & & \\
& \boxed{\{\text{ circuits }\}} \arrow[r, dash, "\supset"] & \boxed{\{\text{ cycles }\}} &
\end{tikzcd}
$$

A maximal path is a path that cannot be extended on either end to be a longer path.

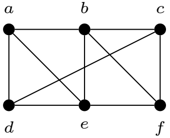
A graph is connected if for every pair of vertices u and v, there is a walk from u to v.

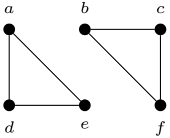
A graph is connected if for every pair of vertices u and v, there is a walk from u to v. For example:



G is connected; H is not.

A graph is connected if for every pair of vertices u and v, there is a walk from u to v. For example:

$G =$


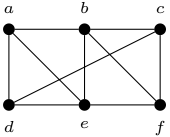
$H =$


G is connected; H is not.

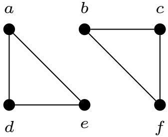
A connected component of a graph is a maximally connected subgraph of G (H above has two connected components).

A graph is connected if for every pair of vertices u and v, there is a walk from u to v. For example:

$G =$

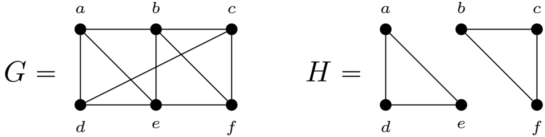


$H =$



G is connected; H is not.
A connected component of a graph is a maximally connected subgraph of G (H above has two connected components). Note that every connected component (or union of connected components) is an induced subgraph.

A graph is connected if for every pair of vertices u and v, there is a walk from u to v. For example:



G is connected; H is not.

A connected component of a graph is a maximally connected subgraph of G (H above has two connected components). Note that every connected component (or union of connected components) is an induced subgraph. Further, if W is the set of vertices in some connected component, then the induced subgraph H by W has the property that the degrees of the vertices in H are the same as the degrees of the corresponding vertices in G.

We say two vertices are connected if there is a walk between them.

Connected is an equivalence relation on vertices:

We say two vertices are connected if there is a walk between them.

Connected is an equivalence relation on vertices:

Reflexive:

We say two vertices are connected if there is a walk between them.

Connected is an equivalence relation on vertices:

Reflexive: The walk from v to itself is the walk of length 0: $v_0 = v = v_n$.

We say two vertices are connected if there is a walk between them.

Connected is an equivalence relation on vertices:

Reflexive: The walk from v to itself is the walk of length 0: $v_0 = v = v_n$.

Symmetric:

We say two vertices are connected if there is a walk between them.

Connected is an equivalence relation on vertices:

- Reflexive: The walk from v to itself is the walk of length 0: $v_0 = v = v_n$.
- Symmetric: If there is a walk from u to v, then the walk from v to u is the reverse sequence.

We say two vertices are connected if there is a walk between them.

Connected is an equivalence relation on vertices:

Reflexive: The walk from v to itself is the walk of length 0: $v_0 = v = v_n$.

Symmetric: If there is a walk from u to v, then the walk from v to u is the reverse sequence.

Transitive:

We say two vertices are connected if there is a walk between them.

Connected is an equivalence relation on vertices:

- Reflexive: The walk from v to itself is the walk of length 0: $v_0 = v = v_n$.
- Symmetric: If there is a walk from u to v, then the walk from v to u is the reverse sequence.
- Transitive: If there is a walk from a to b $w_a = a, e_1, v_1, \ldots, e_n, b$, and a walk from b to c, $w_b = b, e'_1, v'_1, \ldots, e'_n, c$,

We say two vertices are connected if there is a walk between them.

Connected is an equivalence relation on vertices:

Reflexive: The walk from v to itself is the walk of length 0: $v_0 = v = v_n$.

Symmetric: If there is a walk from u to v, then the walk from v to u is the reverse sequence.

Transitive: If there is a walk from a to b $w_a = a, e_1, v_1, \ldots, e_n, b$, and a walk from b to c, $w_b = b, e'_1, v'_1, \ldots, e'_n, c$, then

$$w = a, e_1, v_1, \ldots, e_n, b, e'_1, v'_1, \ldots, e'_n, c$$

is a walk from a to c.

We say two vertices are connected if there is a walk between them.

Connected is an equivalence relation on vertices:

Reflexive: The walk from v to itself is the walk of length 0: $v_0 = v = v_n$.

Symmetric: If there is a walk from u to v, then the walk from v to u is the reverse sequence.

Transitive: If there is a walk from a to b $w_a = a, e_1, v_1, \ldots, e_n, b$, and a walk from b to c, $w_b = b, e_1', v_1', \ldots, e_n', c$, then

$$w = a, e_1, v_1, \ldots, e_n, b, e_1', v_1', \ldots, e_n', c$$

is a walk from a to c.

The equivalence class is the connected component.

Graph invariants

Recall, a graph invariant is a statistic about a graph that is preserved under isomorphisms (relabeling of the vertices). Namely, if you don't need the labels to calculate the statistic, then it's probably a graph invariant.

Graph invariants

Recall, a graph invariant is a statistic about a graph that is preserved under isomorphisms (relabeling of the vertices). Namely, if you don't need the labels to calculate the statistic, then it's probably a graph invariant.

1. $|V|$, $|E|$

Graph invariants

Recall, a graph invariant is a statistic about a graph that is preserved under isomorphisms (relabeling of the vertices). Namely, if you don't need the labels to calculate the statistic, then it's probably a graph invariant.

1. $|V|$, $|E|$
2. Degree sequence

Graph invariants

Recall, a graph invariant is a statistic about a graph that is preserved under isomorphisms (relabeling of the vertices). Namely, if you don't need the labels to calculate the statistic, then it's probably a graph invariant.

1. $|V|$, $|E|$
2. Degree sequence

    Also: Minimum degree, maximum degree

Graph invariants

Recall, a graph invariant is a statistic about a graph that is preserved under isomorphisms (relabeling of the vertices). Namely, if you don't need the labels to calculate the statistic, then it's probably a graph invariant.

1. $|V|$, $|E|$
2. Degree sequence

   Also: Minimum degree, maximum degree, vertex of degree d_1 adjacent to vertex of degree d_2, ...

Graph invariants

Recall, a graph invariant is a statistic about a graph that is preserved under isomorphisms (relabeling of the vertices). Namely, if you don't need the labels to calculate the statistic, then it's probably a graph invariant.

1. $|V|$, $|E|$
2. Degree sequence

   Also: Minimum degree, maximum degree, vertex of degree d_1 adjacent to vertex of degree d_2, ...
3. Bipartite or not

Graph invariants

Recall, a graph invariant is a statistic about a graph that is preserved under isomorphisms (relabeling of the vertices). Namely, if you don't need the labels to calculate the statistic, then it's probably a graph invariant.

1. $|V|$, $|E|$
2. Degree sequence

   Also: Minimum degree, maximum degree, vertex of degree d_1 adjacent to vertex of degree d_2, ...
3. Bipartite or not

   If any subgraph is not bipartite, then G is not bipartite.

Graph invariants

Recall, a graph invariant is a statistic about a graph that is preserved under isomorphisms (relabeling of the vertices). Namely, if you don't need the labels to calculate the statistic, then it's probably a graph invariant.

1. $|V|$, $|E|$
2. Degree sequence

   Also: Minimum degree, maximum degree, vertex of degree d_1 adjacent to vertex of degree d_2, ...
3. Bipartite or not

   If any subgraph is not bipartite, then G is not bipartite. A graph is bipartite if and only if it has no odd cycles as subgraphs.

Graph invariants

Recall, a graph invariant is a statistic about a graph that is preserved under isomorphisms (relabeling of the vertices). Namely, if you don't need the labels to calculate the statistic, then it's probably a graph invariant.

1. $|V|$, $|E|$
2. Degree sequence

   Also: Minimum degree, maximum degree, vertex of degree d_1 adjacent to vertex of degree d_2, ...
3. Bipartite or not

   If any subgraph is not bipartite, then G is not bipartite. A graph is bipartite if and only if it has no odd cycles as subgraphs.
4. Connected or not

Graph invariants

Recall, a graph invariant is a statistic about a graph that is preserved under isomorphisms (relabeling of the vertices). Namely, if you don't need the labels to calculate the statistic, then it's probably a graph invariant.

1. $|V|$, $|E|$
2. Degree sequence

   Also: Minimum degree, maximum degree, vertex of degree d_1 adjacent to vertex of degree d_2, ...
3. Bipartite or not

   If any subgraph is not bipartite, then G is not bipartite. A graph is bipartite if and only if it has no odd cycles as subgraphs.
4. Connected or not
5. Paths or cycles of particular lengths

   Also: longest path or cycle length

Graph invariants

Recall, a graph invariant is a statistic about a graph that is preserved under isomorphisms (relabeling of the vertices). Namely, if you don't need the labels to calculate the statistic, then it's probably a graph invariant.

1. $|V|$, $|E|$
2. Degree sequence

   Also: Minimum degree, maximum degree, vertex of degree d_1 adjacent to vertex of degree d_2, ...
3. Bipartite or not

   If any subgraph is not bipartite, then G is not bipartite. A graph is bipartite if and only if it has no odd cycles as subgraphs.
4. Connected or not
5. Paths or cycles of particular lengths

   Also: longest path or cycle length, maximal paths of certain lengths, ...