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

\[ G_1 = \]

\[ G_2 = \]

\[ G_3 = \]

\[ G_4 = \]

\[ G_5 = \]

\[ G_6 = \]
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 $F \subseteq E$. Some subgraphs include:

- $H_1 = (V, E)$
- $H_2 = (V, E)$
- $H_3 = (V, E)$
- $H_4 = (V, E)$
- $H_5 = (V, E)$
- $H_6 = (V, E)$
- $H_7 = (V, E)$
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

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

$$G = \begin{array}{c}
\begin{tikzpicture}
    \node (a) at (0,0) {a};
    \node (b) at (1,0) {b};
    \node (c) at (2,0) {c};
    \node (d) at (0,-1) {d};
    \node (e) at (1,-1) {e};
    \node (f) at (2,-1) {f};
    \draw (a) -- (b);
    \draw (a) -- (c);
    \draw (a) -- (d);
    \draw (a) -- (e);
    \draw (a) -- (f);
    \draw (b) -- (c);
    \draw (b) -- (d);
    \draw (b) -- (e);
    \draw (b) -- (f);
    \draw (c) -- (d);
    \draw (c) -- (e);
    \draw (c) -- (f);
    \draw (d) -- (e);
    \draw (d) -- (f);
    \draw (e) -- (f);
\end{tikzpicture}
\end{array}$$

Some subgraphs include:

$$H_1 = \begin{array}{c}
\begin{tikzpicture}
    \node (a) at (0,0) {a};
    \node (b) at (1,0) {b};
    \node (c) at (2,0) {c};
    \node (d) at (0,-1) {d};
    \node (e) at (1,-1) {e};
    \node (f) at (2,-1) {f};
    \draw (a) -- (b);
    \draw (a) -- (c);
    \draw (a) -- (d);
    \draw (a) -- (e);
    \draw (a) -- (f);
    \draw (b) -- (c);
    \draw (b) -- (d);
    \draw (b) -- (e);
    \draw (b) -- (f);
    \draw (c) -- (d);
    \draw (c) -- (e);
    \draw (c) -- (f);
    \draw (d) -- (e);
    \draw (d) -- (f);
    \draw (e) -- (f);
\end{tikzpicture}
\end{array}$$
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

![Graph G with nodes a, b, c, d, e, f and edges between a and b, a and c, b and c, d and e, d and f, e and f]

Some subgraphs include:

$H_1 = \begin{array}{ccc}
    a & b & c \\
    d & e & f
\end{array}$

$H_2 = \begin{array}{ccc}
    a & b & c \\
    d & e
\end{array}$
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

$$G = \begin{array}{c}
a & b & c \\
d & e & f \\
\end{array}$$

Some subgraphs include:

$$H_1 = \begin{array}{c}
a & b & c \\
d & e & f \\
\end{array}$$

$$H_2 = \begin{array}{c}
a & b & c \\
d & e & f \\
\end{array}$$

$$H_3 = \begin{array}{c}
a & b & c \\
d & e & f \\
\end{array}$$
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

$$G = \begin{array}{c}
\text{a} & \text{b} & \text{c} \\
\text{d} & \text{e} & \text{f} \\
\end{array}$$

Some subgraphs include:

$$H_1 = \begin{array}{c}
\text{a} & \text{b} & \text{c} \\
\text{d} & \text{e} & \text{f} \\
\end{array}$$

$$H_2 = \begin{array}{c}
\text{a} & \text{b} & \text{c} \\
\text{d} & \text{e} & \text{f} \\
\end{array}$$

$$H_3 = \begin{array}{c}
\text{a} & \text{b} & \text{c} \\
\text{d} & \text{e} & \text{f} \\
\end{array}$$

$$H_4 = \begin{array}{c}
\text{a} & \text{b} & \text{c} \\
\text{d} & \text{e} & \text{f} \\
\end{array}$$

$$H_5 = \begin{array}{c}
\text{b} \\
\text{d} & \text{e} & \text{f} \\
\end{array}$$
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

$$G = \begin{array}{c}
\begin{array}{ccc}
  a & b & c \\
  d & e & f \\
\end{array}
\end{array}$$

Some subgraphs include:

- $H_1 = \begin{array}{c}
\begin{array}{ccc}
  a & b & c \\
  d & e & f \\
\end{array}
\end{array}$
- $H_2 = \begin{array}{c}
\begin{array}{ccc}
  a & b & c \\
  d & e & f \\
\end{array}
\end{array}$
- $H_3 = \begin{array}{c}
\begin{array}{ccc}
  a & b & c \\
  d & e & f \\
\end{array}
\end{array}$
- $H_4 = \begin{array}{c}
\begin{array}{ccc}
  a & b & c \\
  d & e & f \\
\end{array}
\end{array}$
- $H_5 = b$
- $H_6 = \emptyset$
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

$$G = \begin{array}{c}
\begin{tikzpicture}
\node (a) at (0,0) {a};
\node (b) at (1,0) {b};
\node (c) at (2,0) {c};
\node (d) at (0,-1) {d};
\node (e) at (1,-1) {e};
\node (f) at (2,-1) {f};
\draw (a) -- (b);
\draw (b) -- (c);
\draw (c) -- (d);
\draw (d) -- (e);
\draw (e) -- (f);
\draw (f) -- (a);
\end{tikzpicture}
\end{array}$$

Some subgraphs include:

- $H_1 = \begin{array}{c}
\begin{tikzpicture}
\node (a) at (0,0) {a};
\node (b) at (1,0) {b};
\node (c) at (2,0) {c};
\node (d) at (0,-1) {d};
\node (e) at (1,-1) {e};
\node (f) at (2,-1) {f};
\draw (a) -- (b);
\draw (b) -- (c);
\end{tikzpicture}
\end{array}$
- $H_2 = \begin{array}{c}
\begin{tikzpicture}
\node (a) at (0,0) {a};
\node (b) at (1,0) {b};
\node (c) at (2,0) {c};
\node (d) at (0,-1) {d};
\node (e) at (1,-1) {e};
\draw (a) -- (b);
\draw (b) -- (c);
\end{tikzpicture}
\end{array}$
- $H_3 = \begin{array}{c}
\begin{tikzpicture}
\node (a) at (0,0) {a};
\node (b) at (1,0) {b};
\node (c) at (2,0) {c};
\node (d) at (0,-1) {d};
\node (e) at (1,-1) {e};
\draw (a) -- (b);
\draw (b) -- (c);
\end{tikzpicture}
\end{array}$
- $H_4 = \begin{array}{c}
\begin{tikzpicture}
\node (a) at (0,0) {a};
\node (b) at (1,0) {b};
\node (c) at (2,0) {c};
\node (d) at (0,-1) {d};
\node (e) at (1,-1) {e};
\end{tikzpicture}
\end{array}$
- $H_5 = \begin{array}{c}
\begin{tikzpicture}
\node (a) at (0,0) {a};
\node (b) at (1,0) {b};
\node (c) at (2,0) {c};
\node (d) at (0,-1) {d};
\node (e) at (1,-1) {e};
\end{tikzpicture}
\end{array}$
- $H_6 = \emptyset$
- $H_7 = \begin{array}{c}
\begin{tikzpicture}
\node (a) at (0,0) {a};
\node (b) at (1,0) {b};
\node (c) at (2,0) {c};
\node (d) at (0,-1) {d};
\node (e) at (1,-1) {e};
\node (f) at (2,-1) {f};
\draw (a) -- (b);
\draw (b) -- (c);
\draw (c) -- (d);
\draw (d) -- (e);
\draw (e) -- (f);
\draw (f) -- (a);
\end{tikzpicture}
\end{array}
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

![Graph Image]

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

![Graph diagram]

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

$$G = \begin{array}{c}
\begin{tikzpicture}
  \node (a) at (0,0) {$a$};
  \node (b) at (1,0) {$b$};
  \node (c) at (2,0) {$c$};
  \node (d) at (0,-1) {$d$};
  \node (e) at (1,-1) {$e$};
  \node (f) at (2,-1) {$f$};
  \draw (a) -- (b);
  \draw (a) -- (c);
  \draw (a) -- (d);
  \draw (b) -- (c);
  \draw (b) -- (e);
  \draw (c) -- (f);
  \draw (d) -- (e);
  \draw (d) -- (f);
  \draw (e) -- (f);
\end{tikzpicture}
\end{array}$$

then

$$G - a = \begin{array}{c}
\begin{tikzpicture}
  \node (b) at (0,0) {$b$};
  \node (c) at (1,0) {$c$};
  \node (d) at (0,-1) {$d$};
  \node (e) at (1,-1) {$e$};
  \node (f) at (2,-1) {$f$};
  \draw (b) -- (c);
  \draw (b) -- (d);
  \draw (b) -- (e);
  \draw (c) -- (f);
  \draw (d) -- (e);
  \draw (d) -- (f);
  \draw (e) -- (f);
\end{tikzpicture}
\end{array}$$
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

For example, if

$G = \begin{array}{ccc}
a & b & c \\
d & e & f \\
\end{array}$

then

$G - a = \begin{array}{ccc}
& b & c \\
d & e & f \\
\end{array}$

$G - b$
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

$G = \begin{array}{ccc}
  a & b & c \\
  d & e & f \\
\end{array}$

then

$G - a = \begin{array}{ccc}
  & b & c \\
  d & e & f \\
\end{array}$

$G - b$
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

\[
G = \begin{array}{c}
\text{a} & \text{b} & \text{c} \\
\text{d} & \text{e} & \text{f}
\end{array}
\]

then

\[
G - a = \begin{array}{c}
\text{b} & \text{c} \\
\text{d} & \text{e} & \text{f}
\end{array}
\]

\[
G - b = \begin{array}{c}
\text{a} & \text{c} \\
\text{d} & \text{e} & \text{f}
\end{array}
\]
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

\[
G = 
\begin{array}{ccc}
    a & b & c \\
    d & e & f \\
\end{array}
\]

then

\[
G - a = 
\begin{array}{ccc}
    b & c \\
    d & e & f \\
\end{array}
\quad G - b = 
\begin{array}{ccc}
    a & c \\
    d & e & f \\
\end{array}
\]

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

$$G = \begin{array}{c}
  a & b & c \\
  d & e & f \\
\end{array}$$

then

$$G - a = \begin{array}{c}
  b & c \\
  d & e & f \\
\end{array} \quad \begin{array}{c}
  a & c \\
  d & e & f \\
\end{array}$$

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

Step 1: Break into cases based on the vertex set. For example, let $G$ have vertex set $V = \{a, b, c\}$. If $H \subseteq G$, then $V_H \subseteq V$. Possibilities: $H = V$, $V_H = \{a\}$, $V_H = \{b\}$, $V_H = \{c\}$, $V_H = \{a, b\}$, $V_H = \{a, c\}$, $V_H = \{b, c\}$, or $V_H = \{a, b, c\}$.

Step 2: Draw the induced graphs for each vertex subset. In our example, $G_{\{H\}} = H$, $G_{\{a\}}$, $G_{\{b\}}$, $G_{\{c\}}$, $G_{\{a, b\}}$, $G_{\{a, c\}}$, $G_{\{b, c\}}$, or $G_{\{a, b, c\}}$.

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.
Counting subgraphs

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

For example, let $G = \binom{\text{a} \quad \text{b} \quad \text{c}}{\text{a}}$. If $H \subseteq G$, then $V(H) \subseteq \{a, b, c\}$.

Possibilities: $H = \emptyset$, $t(a,u)$, $t(b,u)$, $t(c,u)$, $t(a,b,u)$, $t(a,c,u)$, $t(b,c,u)$, or $t(a,b,c,u)$.

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

In our example, $G \binom{\text{H}}{\text{a}}$, $G \binom{\text{rt}}{\text{a}}$, $G \binom{\text{rt}}{\text{b}}$, $G \binom{\text{rt}}{\text{c}}$, $G \binom{\text{rt}}{\text{a,b}}$, $G \binom{\text{rt}}{\text{a,c}}$, $G \binom{\text{rt}}{\text{b,c}}$, or $G \binom{\text{rt}}{\text{a,b,c}}$.

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 $G = \begin{array}{c} a \end{array} - \begin{array}{c} b \end{array} - \begin{array}{c} c \end{array}$

$G$ has vertex set $V = V_G = \{a, b, c\}$. 
Counting subgraphs

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

For example, let $G = \begin{array}{c}
\begin{array}{c}
\bullet \\
\bullet \\
\bullet
\end{array}
\end{array}$

$G$ has 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 $G = \cdot \cdot \cdot$  
$G$ has 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 $G = \begin{tikzpicture}[baseline = 0]
\node[draw, circle, fill=black] (a) at (0,0) {};
\node[draw, circle, fill=black] (b) at (1,0) {};
\node[draw, circle, fill=black] (c) at (2,0) {};
\draw (a) -- (b) -- (c);
\end{tikzpicture}$

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

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

- $\emptyset$, $\{a\}$, $\{b\}$, $\{c\}$, $\{a, b\}$,
- $\{a, c\}$, $\{b, c\}$, or $\{a, b, c\}$.

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 $G = a \quad b \quad c$

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

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

Step 2: Draw the induced graphs for each vertex subset.
In our example,

$G[\emptyset] = \emptyset$, 

$G[\{a\}]$, 

$G[\{b\}]$, 

$G[\{c\}]$, 

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

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

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

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

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

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

For example, let \( G = \begin{array}{c} a \quad b \quad c \end{array} \)

\( G \) has vertex set \( V = V_G = \{a, b, c\} \).

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

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

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

In our example,

\[ G[\emptyset] = \emptyset, \quad G[\{a\}] = \begin{array}{c} a \end{array}, \quad G[\{b\}] = \begin{array}{c} b \end{array}, \quad G[\{c\}] = \begin{array}{c} c \end{array}, \quad \]
Counting subgraphs

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

For example, let $G = \begin{array}{ccc} a & b & c \\ \end{array}$

$G$ has 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,

$G[\varnothing] = \varnothing, \quad G[\{a\}] = \begin{array}{c} a \\ \end{array}, \quad G[\{b\}] = \begin{array}{c} b \\ \end{array}, \quad G[\{c\}] = \begin{array}{c} c \\ \end{array},$

$G[\{a, b\}] = \begin{array}{ccc} a & b \\ \end{array}, \quad G[\{a, c\}] = \begin{array}{ccc} a & c \\ \end{array}, \quad G[\{b, c\}] = \begin{array}{ccc} b & c \\ \end{array}$
Counting subgraphs

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

For example, let \(G = \begin{array}{ccc} a & b & c \end{array}\)

\(G\) has vertex set \(V = V_G = \{a, b, c\}\).

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

\(\emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}\) or \(\{a, b, c\}\).

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

In our example,

\[
\begin{align*}
G[\emptyset] &= \emptyset, & G[\{a\}] &= a, & G[\{b\}] &= b, & G[\{c\}] &= c, \\
G[\{a, b\}] &= a \, b, & G[\{a, c\}] &= a \, c, & G[\{b, c\}] &= b \, c \\
G[\{a, b, c\}] &= G = a \, b \, c.
\end{align*}
\]
Counting subgraphs

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

For example, let \( G = \bullet - \bullet - \bullet \).

\( G \) has vertex set \( V = V_G = \{a, b, c\} \).

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

\( \emptyset, \ \{a\}, \ \{b\}, \ \{c\}, \ \{a, b\}, \ \{a, c\}, \ \{b, c\}, \ or \ \{a, b, c\} \).

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

In our example,

\[
\begin{align*}
G[\emptyset] &= \emptyset, & G[\{a\}] &= a, & G[\{b\}] &= b, & G[\{c\}] &= c, \\
G[\{a, b\}] &= a - b, & G[\{a, c\}] &= a - c, & G[\{b, c\}] &= b - c, \\
G[\{a, b, c\}] &= G = a - b - c.
\end{align*}
\]

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 $G = a \rightarrow b \rightarrow c$.

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

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

- $\emptyset$, $\{a\}$, $\{b\}$, $\{c\}$, $\{a, b\}$,
- $\{a, c\}$, $\{b, c\}$, or $\{a, b, c\}$.

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

In our example,

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

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

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

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 $G = \begin{array}{c}
a & b & c \\
\end{array}$
$G$ has vertex set $V = V_G = \{a, b, c\}$.
If $H \subseteq G$, then $V_H \subseteq \{a, b, c\}$. Possibilities:
- $\emptyset$, $\{a\}$, $\{b\}$, $\{c\}$, $\{a, b\}$,
- $\{a, c\}$, $\{b, c\}$, or $\{a, b, c\}$.

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

In our example,

- $G[\emptyset] = \emptyset$, $G[\{a\}] = \begin{array}{c}a \\
\end{array}$, $G[\{b\}] = \begin{array}{c}b \\
\end{array}$, $G[\{c\}] = \begin{array}{c}c \\
\end{array}$,
- $G[\{a, b\}] = \begin{array}{c}a & b \\
\end{array}$, $G[\{a, c\}] = \begin{array}{c}a & c \\
\end{array}$, $G[\{b, c\}] = \begin{array}{c}b & c \\
\end{array}$
- $G[\{a, b, c\}] = G = \begin{array}{c}a & b & c \\
\end{array}$.

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 $G = \begin{array}{ccc} a & b & c \\ \end{array}$

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

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

- $\emptyset$
- $\{a\}$, $\{b\}$, $\{c\}$, $\{a, b\}$,
- $\{a, c\}$, $\{b, c\}$, or $\{a, b, c\}$.

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

In our example,

- $G[\emptyset] = \emptyset$, $G[\{a\}] = \begin{array}{c} a \\ \end{array}$, $G[\{b\}] = \begin{array}{c} b \\ \end{array}$, $G[\{c\}] = \begin{array}{c} c \\ \end{array}$,
- $G[\{a, b\}] = \begin{array}{ccc} \bullet & \cdots & \bullet \\ a & \cdots & b \\ \end{array}$, $G[\{a, c\}] = \begin{array}{ccc} \bullet & \cdots & \bullet \\ a & \cdots & c \\ \end{array}$, $G[\{b, c\}] = \begin{array}{ccc} \bullet & \cdots & \bullet \\ b & \cdots & c \\ \end{array}$
- $G[\{a, b, c\}] = G = \begin{array}{ccc} a & \cdots & c \\ b & \cdots & b \end{array}$.

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. $1 + 1 + 1 + 1 + 2 + 1 + 2 + 2^2 = 13$
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

\[
G = \begin{array}{ccc}
  a & b & c \\
  d & e & f
\end{array}
\]

then

\[
G - (a - b) = \begin{array}{ccc}
  a & b & c \\
  d & e & f
\end{array}
\]
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

$$G = \begin{array}{c}
\begin{array}{ccc}
a & b & c \\
\bullet & \bullet & \bullet \\
d & e & f
\end{array}
\end{array}$$

then

$$G - \{a-b, c-f, e-f\} = \begin{array}{c}
\begin{array}{ccc}
a & b & c \\
\bullet & \bullet & \bullet \\
d & e & f
\end{array}
\end{array}$$
Edge operations.

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

$$G = \begin{array}{c}
\begin{array}{ccc}
    a & b & c \\
    d & e & f
\end{array}
\end{array}$$

then

$$G + (a-e) = \begin{array}{c}
\begin{array}{ccc}
    a & b & c \\
    d & e & f
\end{array}
\end{array}$$
Edge operations.

Let $\epsilon \in E$. There are two kinds of “contraction” of $\epsilon$: 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 $\epsilon$: 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 $G/\epsilon$ is the graph obtained by *contracting* the edge $\epsilon$, which means merging the vertices that are incident to $\epsilon$. 
Edge operations.

Let $\epsilon \in E$. There are two kinds of “contraction” of $\epsilon$: 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 $G/\epsilon$ is the graph obtained by contracting the edge $\epsilon$, which means merging the vertices that are incident to $\epsilon$.

For example, if $G = \begin{graph}
\node a / b / c / d / e / f;
\node / a; / b; / c; / d; / e; / f;
\edge a b / b c / c a / a d / d e / e f / f a / b c / c a / a d / d e / e f;\end{graph}$ and $\epsilon = a - d$, then $G/\epsilon = \begin{graph}
\node a / b / c / d / e / f;
\node / a; / b; / c; / d; / e; / f;
\edge a e / a f / b c / c a / c f / d f / e f / e d;\end{graph}$
Edge operations.

Let $\epsilon \in E$. There are two kinds of “contraction” of $\epsilon$: 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 $G/\epsilon$ is the graph obtained by contracting the edge $\epsilon$, which means merging the vertices that are incident to $\epsilon$.

For example, if $G = \begin{array}{cccc}
  a & b & c \\
  d & e & f
\end{array}$ and $\epsilon = a-d$, then $G/\epsilon = \begin{array}{cccc}
  a/d & b & c \\
  e & f
\end{array}$.
Edge operations.

Let $\epsilon \in E$. There are two kinds of “contraction” of $\epsilon$: 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 $G/\epsilon$ is the graph obtained by **contracting** the edge $\epsilon$, and then deleting any loops or multiple edges.
Edge operations.

Let $\epsilon \in E$. There are two kinds of “contraction” of $\epsilon$: 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 $G/\epsilon$ is the graph obtained by contracting the edge $\epsilon$, and then deleting any loops or multiple edges.

For example, if

\[ G = \]

\[
\begin{array}{ccc}
  a & b & c \\
  d & e & f \\
\end{array}
\]

and $\epsilon = a-d$, then

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

\[
\begin{array}{ccc}
  a & b & c \\
  d & e & f \\
\end{array}
\]

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


Edge operations.

Let $\epsilon \in E$. There are two kinds of “contraction” of $\epsilon$: 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 $G/\epsilon$ is the graph obtained by contracting the edge $\epsilon$, and then deleting any loops or multiple edges.

For example, if

$$G = \begin{array}{ccc}
\text{a} & \text{b} & \text{c} \\
\text{d} & \text{e} & \text{f}
\end{array}$$

and $\epsilon = a - d$, then

$$(G/\epsilon)_\text{simple} = \begin{array}{ccc}
\text{b} & \text{c} \\
\text{e} & \text{f}
\end{array}$$
Edge operations.

Let $\epsilon \in E$. There are two kinds of “contraction” of $\epsilon$: 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 $G/\epsilon$ is the graph obtained by contracting the edge $\epsilon$, and then deleting any loops or multiple edges.

For example, if

$$G = \begin{array}{ccc} a & b & c \\ d & e & f \end{array}$$

and $\epsilon = a - d$, then

$$\left( G/\epsilon \right)_{\text{simple}} = \begin{array}{ccc} b & c \\ e & f \end{array}$$
Edge operations.

Let $\epsilon \in E$. There are two kinds of “contraction” of $\epsilon$: 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 $G/\epsilon$ is the graph obtained by contracting the edge $\epsilon$, and then deleting any loops or multiple edges. 

For example, if

$$G = \begin{array}{ccc}
\begin{array}{ccc}
a & b & c \\
d & e & f
\end{array}
\end{array}$$

and $\epsilon = a-d$, then

$$(G/\epsilon)_{\text{simple}} = \begin{array}{ccc}
\begin{array}{ccc}
a/d & b & c \\
e & f
\end{array}
\end{array}$$

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

\[
G_1 = a \quad b \quad c \quad \text{and} \quad G_2 = x \quad y \quad z
\]
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

$$G_1 = a \quad b \quad c$$

and

$$G_2 = x \quad y \quad z$$

then

$$G_1 \cup G_2 = \begin{array}{c}
\quad a \quad b \quad c \\
\bullet - \bullet - \bullet \\
\quad x \quad y \quad z \\
\bullet - \bullet - \bullet
\end{array}$$
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

$G_1 = \begin{array}{ccc} & a & b & c \end{array}$ and $G_2 = \begin{array}{ccc} & x & y & z \end{array}$

then

$G_1 \cup G_2 = \begin{array}{ccc} a & b & c \\ x & y & z \end{array}$

If

$G_1 = \begin{array}{ccc} & a & b & c \end{array}$ and $G_2 = \begin{array}{ccc} & a & d & b \end{array}$

then

$G_1 \cup G_2 = \begin{array}{ccc} a & b & c \\ a & d & b \end{array}$
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

\[
G_1 = \begin{array}{ccc}
  a & b & c \\
\end{array}
\quad \text{and} \quad G_2 = \begin{array}{ccc}
  x & y & z \\
\end{array}
\]

then

\[
G_1 \cup G_2 = \begin{array}{ccc}
  a & b & c \\
  x & y & z \\
\end{array}
\]

If

\[
G_1 = \begin{array}{ccc}
  a & b & c \\
\end{array}
\quad \text{and} \quad G_2 = \begin{array}{ccc}
  a & d & b \\
\end{array}
\]

then

\[
G_1 \cup G_2 = \begin{array}{ccc}
  a & b & c \\
  & & \\
  & & \\
  d & & \\
\end{array}
\]
Complements

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

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

In other words, $G$ and $\overline{G}$ have the same vertex set, but $u$ and $v$ are adjacent in $\overline{G}$ 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

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

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

Example: Let

$$G =$$

![Diagram of G]

Then $K[V] = a \, b \, c \, d$ so $G = a \, b \, c \, d$. 
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

$$
G = \begin{align*}
&\begin{array}{c}
\text{a} \\
\text{b} \\
\text{c} \\
\text{d}
\end{array} \\
&\begin{array}{c}
\text{a} \\
\text{b} \\
\text{c} \\
\text{d}
\end{array}
\end{align*}
$$

Then

$$
K[V] = \begin{align*}
&\begin{array}{c}
\text{a} \\
\text{b} \\
\text{c} \\
\text{d}
\end{array} \\
&\begin{array}{c}
\text{a} \\
\text{b} \\
\text{c} \\
\text{d}
\end{array}
\end{align*}
$$
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

$$G = \begin{align*}
  &a &b &c \\
  &\downarrow &\downarrow &\downarrow \\
  a & &b &c \\
  &\downarrow &\downarrow &\downarrow \\
  &d & &d
\end{align*}$$

Then

$$K[V] = \begin{align*}
  &a &b &c \\
  &\downarrow &\downarrow &\downarrow \\
  a & &b &c \\
  &\downarrow &\downarrow &\downarrow \\
  &d & &d
\end{align*}$$
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

$$G = \begin{align*}
\begin{array}{c}
 a \\
 b \\
 c \\
 d
\end{array}
\end{align*}$$

Then

$$K[V] = \begin{align*}
\begin{array}{c}
 a \\
 b \\
 c \\
 d
\end{array}
\end{align*}$$
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

$G = \begin{align*}
  &a &b \\
  a & &c \\
  b &c &d \\
  a &b &c &d
\end{align*}$

Then

$K[V] = \begin{align*}
  &a &b &c \\
  a & & & & \\
  b & & & & \\
  c & & & & \\
  d & & & & \\
  a &b &c &d
\end{align*}$

so

$$\bar{G} = \begin{align*}
  &a &b \\
  a & & \\
  b & & \\
  a &b &c &d
\end{align*}.$$
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

![Diagram of graph G]

the walk looks like

![Diagram of another graph]
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

the walk $(a$

looks like
Connectedness

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

\[
    w = (v_0, e_1, v_1, e_2, \ldots, e_n, v_n)
\]

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

\[
    G = \begin{array}{c}
        a & b & c \\
        d & e & f
    \end{array}
\]

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

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

```
a b c
d e f
```

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

the walk $(a, a\rightarrow b, b\rightarrow c, c, c\rightarrow f, f, f\rightarrow b, b)$ 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

the walk $(a, a\rightarrow b, b\rightarrow c, c, c\rightarrow f, f, f\rightarrow b, b, b\rightarrow c, c)$ 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

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 $v_0 = v_n$. 

Note: See the remark on p. 679 of the book to reconcile 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.

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.
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 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!!
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:

- 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:

\[ \{ \text{walks} \} \rightarrow \{ \text{circuits} \} \rightarrow \{ \text{trails} \} \rightarrow \{ \text{paths} \} \rightarrow \{ \text{max'l paths} \} \rightarrow \{ \text{cycles} \} \]

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:

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

\[
\{\text{walks}\} \rightarrow \{\text{trails}\} \rightarrow \{\text{paths}\} \rightarrow \{\text{max'l paths}\}
\]

\[
\{\text{walks}\} \rightarrow \{\text{circuits}\} \rightarrow \{\text{cycles}\}
\]
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:

\[
G = \begin{array}{c}
\text{a} \\
\text{b} \\
\text{c} \\
\text{d} \\
\text{e} \\
\text{f}
\end{array}
\begin{array}{c}
\text{a} \\
\text{b} \\
\text{c} \\
\text{d} \\
\text{e} \\
\text{f}
\end{array}
\]

\[H = \]

\( G \) 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:

\[
G = \begin{array}{c}
\begin{array}{ccc}
  a & b & c \\
  d & e & f \\
\end{array}
\end{array}
\quad \quad \quad H = \begin{array}{c}
\begin{array}{ccc}
  a & b & c \\
  d & e & f \\
\end{array}
\end{array}
\]

\( G \) 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:

$$G = \begin{array}{c}
\text{a} \\
\text{d}
\end{array} \quad \begin{array}{c}
\text{b} \\
\text{e}
\end{array} \quad \begin{array}{c}
\text{c} \\
\text{f}
\end{array}$$

$$H = \begin{array}{c}
\text{a} \\
\text{d}
\end{array} \quad \begin{array}{c}
\text{b} \\
\text{e}
\end{array} \quad \begin{array}{c}
\text{c} \\
\text{f}
\end{array}$$

$G$ 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:

$$G = \begin{array}{c}
\text{a} & \text{b} & \text{c} \\
\text{d} & \text{e} & \text{f}
\end{array}$$

$$H = \begin{array}{c}
\text{a} & \text{b} & \text{c} \\
\text{d} & \text{e} & \text{f}
\end{array}$$

$G$ 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:

- Reflexive: The walk from \( v \) to itself is the walk of length 0: \( v \rightarrow v \).
- 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 \( b \) and a walk from \( b \) to \( c \), then the concatenation of these two walks is a walk from \( a \) to \( c \).

The equivalence class is the connected component.
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 $c$. 

The equivalence class is the connected component.
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.
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, . . .