Graphs & Trees

CISC 2210 — CUNY Brooklyn College
Lecture Notes

Graphs & Trees: Intro

In Topic 4, we talked about the **relation** data structure, which is a set of ordered pairs of the form (s, t), where $s \in S$ and $t \in T$. When $S = T$, we called it a relation on S.

Example: the relation $\mathrm{GreaterThan}$ on $S = \{1, 2, 3, 4\}$ consists of the pairs $\{(2, 1), (3, 2), (3, 1), (4, 3), (4, 2), (4, 1)\}$.

Besides listing the elements of a relation, drawing a mapping diagram, or plotting the relation on the coordinate plane, we have two more ways to visualize, and hence learn more, about a relation: (1) an **adjacency matrix** and (2) a **directed graph**.

The adjacency matrix for the relation $\mathrm{GreaterThan}$ above, for example, will be: $M = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{bmatrix}$.

See the idea? The number of rows (and columns) of the adjacency matrix is equal to $|S|$ (the size of the set S). Since $|S| = |\{1, 2, 3, 4\}| = 4$, this means that the adjacency matrix would consist of 4 rows (and 4 columns).

The way we fill out the matrix is based on the following 2 rules:

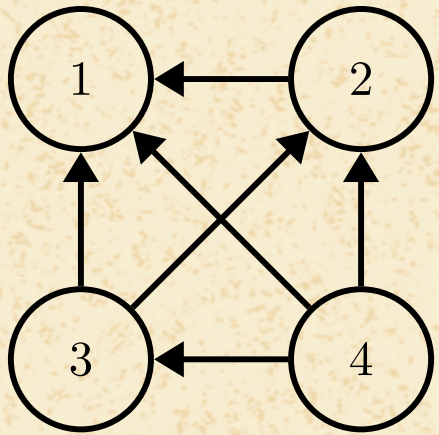
1. If $(i, j) \in \mathrm{GreaterThan}$ (that is, if $i > j$), then $M_{i,j} = 1$.
2. Otherwise, if $(i, j) \notin \mathrm{GreaterThan}$, then $M_{i,j} = 0$.

In other words, an adjacency matrix will tell us which pairs are 'lit up' and exist in the relation, and which pairs aren't.

Fun fact: the trend of the 1s and 0s in the adjacency matrix might show us some important patterns in the underlying relation. In the case of M on the previous slide, for instance, we clearly see how all the 1s are grouped below the main diagonal, which means that the relation is applied only to the pairs where $i > j$, which is the whole idea of $\mathrm{GreaterThan}$!

Graphs & Trees: Intro

Similarly, we can draw a **directed graph** to represent this $\mathrm{GreaterThan}$ relation on S as follows:



A di-graph for the relation GreaterThan on S. Miriam Briskman, CC BY-NC 4.0.

Here, we have a node/bubble for every item in the set S, and an arrow $i \to j$ for every $i > j$.

```
% Code for the di-graph for the relation GreaterThan on S on the previous slide:
\documentclass[border=1pt]{standalone}

% The graphics library we use: TikZ
\usepackage{tikz}
% Allow more arrow feature customization:
\usetikzlibrary{arrows.meta}

% Set the styles of the vertices and the edges in the digraph:
\tikzset{vertex/.style={draw, very thick, circle, minimum size=1cm},
edge/.style={very thick, -Triangle}}
% '-Triangle' means that the arrowhead has a shape of a triangle (there are many other arrowhead
styles in TikZ - check them out! https://tikz.dev/tikz-arrows#sec-16.5)
```

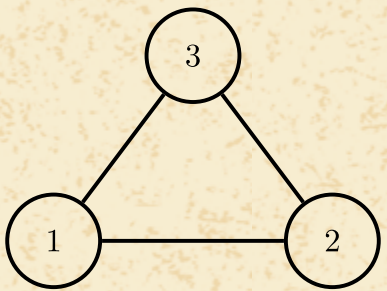
Graphs & Trees: Intro

So, what exactly is a directed graph (or a graph, in general?)

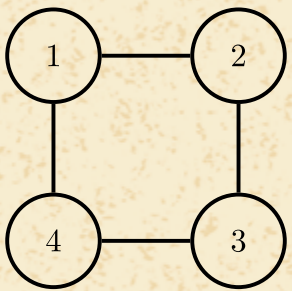
A **graph** (also called an **undirected graph**) is an ordered pair $G = (V, E)$ in which:

- The set V is a non-empty set of **vertices** (singular: **vertex**), also called **nodes**, and
- The set E is a set of **undirected edges** of the form $\{v_1, v_2\}$, where $v_1, v_2 \in V$ (each edge connects two vertices.)

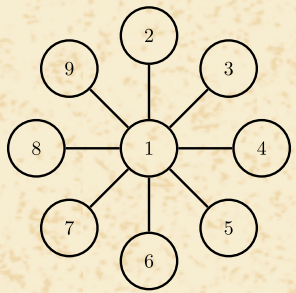
Note a graph's edge is a two-way street: having $\{v_1, v_2\} \in E$ is the same as having both $v_1 \to v_2$ **and** $v_2 \to v_1$.



A graph shaped as a triangle. Miriam Briskman, CC BY-NC 4.0.



A graph shaped as a square. Miriam Briskman, CC BY-NC 4.0.



A star-shaped graph. Miriam Briskman, CC BY-NC 4.0.

Graphs & Trees: Intro


```
% Code for the triangular
graph:
\documentclass[border=1pt]
{standalone}

% The graphics library we use:
TikZ
\usepackage{tikz}

% Set the styles of the
vertices and the edges in the
graph:
\tikzset{vertex/.style={draw,
very thick, circle, minimum
```

```
% Code for the square graph:
\documentclass[border=1pt]
{standalone}

% The graphics library we use:
TikZ
\usepackage{tikz}

% Set the styles of the
vertices and the edges in the
graph:
\tikzset{vertex/.style={draw,
very thick, circle, minimum
size=1cm},
```

```
% Code for the star-shaped
graph:
\documentclass[border=1pt]
{standalone}

% The graphics library we use:
TikZ
\usepackage{tikz}

% Set the styles of the
vertices and the edges in the
graph:
\tikzset{vertex/.style={draw,
very thick, circle, minimum
```

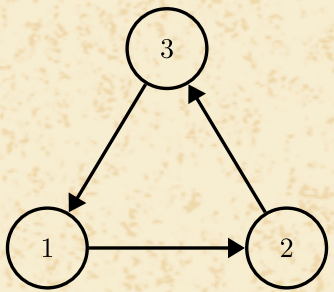
Graphs & Trees: Intro

A **directed graph** (or **di-graph** / **digraph**) is an ordered pair $G = (V, E)$ in which:

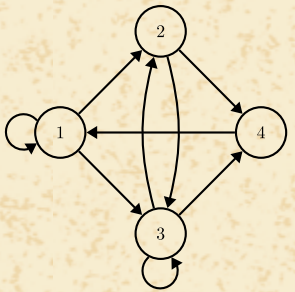
- The set V is a non-empty set of **vertices** / **nodes**, and
- The set E is a set of **directed edges** (also called **arcs** or **arrows**) of the form (v_1, v_2), where $v_1, v_2 \in V$ (each edge connects two vertices.)

Note a digraph's edge is a one-way street: having $(v_1, v_2) \in E$ is the same as having **only** $v_1 \to v_2$ (but it doesn't tell anything about $v_2 \to v_1$.) In other words, the direction of the arrow matters in a directed graph.

A digraph shaped as a chain. Miriam Briskman, CC BY-NC 4.0.



A digraph shaped as a cycle. Miriam Briskman, CC BY-NC 4.0.



A digraph with multiple edges and self-loops. Miriam Briskman, CC BY-NC 4.0.

```
% Code for the chain digraph:
\documentclass[border=1pt]
{standalone}

% The graphics library we use:
TikZ
\usepackage{tikz}

\usetikzlibrary{arrows.meta}

% Set the styles of the
vertices and the edges in the
digraph:
\tikzset{vertex/.style={draw,
```

```
% Code for the cycle-shaped
digraph:
\documentclass[border=1pt]
{standalone}

% The graphics library we use:
TikZ
\usepackage{tikz}

\usetikzlibrary{arrows.meta}

% Set the styles of the
vertices and the edges in the
digraph:
```

```
% Code for the multi-edge
digraph:
\documentclass[border=1pt]
{standalone}

% The graphics library we use:
TikZ
\usepackage{tikz}

% The 'bending' library lets us
create perfectly circular self-
loops:
\usetikzlibrary{arrows.meta,
bending}
```

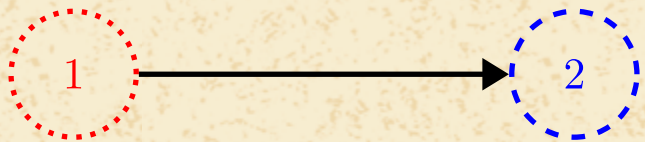
Any questions so far?

Graph Properties / Features

Let's look at features that graphs or digraphs have before viewing some special types of graphs.

1. In a directed edge (v_1, v_2):
   - The node v_1 is called the **initial vertex** (or **source** vertex), and
   - The node v_2 is called the **terminal vertex** (or **destination** vertex).

   The direction of the edge goes from the initial vertex to the terminal vertex.



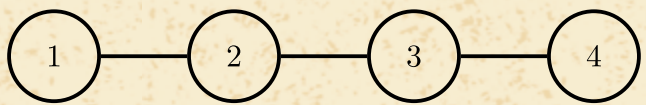
The directed edge is $(1, 2)$. Vertex 1 is the initial vertex, and vertex 2 is the terminal vertex. Miriam Briskman, CC BY-NC 4.0.

```
\documentclass[border=1pt]{standalone}
\usepackage{tikz}
\usetikzlibrary{arrows.meta}

\tikzset{
vertex/.style={draw, circle, very thick,
minimum size=1cm},
```

Graph Properties / Features

2. A **path** in a graph is a sequence of vertices $v_1, v_2, \ldots, v_n$ such that consecutive vertices are connected by edges. In a digraph, a path is denoted as $v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_n$
   `$v_1 \rightarrow v_2 \rightarrow \dots \rightarrow v_n$`.

   A path represents movement through the graph from one vertex to another.

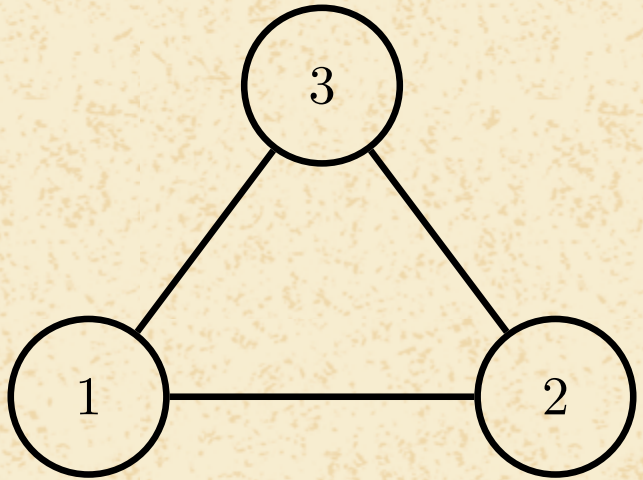


The sequence $1, 2, 3, 4$ is a path because each consecutive pair of vertices is connected by an edge. Miriam Briskman, CC BY-NC 4.0.

```
\documentclass[border=1pt]{standalone}
\usepackage{tikz}

\tikzset{
vertex/.style={draw, circle, very thick,
minimum size=1cm},
edge/.style={very thick}
```

Graph Properties / Features

3. A **closed path** is a path that begins and ends at the same vertex.



The sequence $1, 2, 3, 1$ is a closed path because it starts and ends at vertex 1. Miriam Briskman, CC BY-NC 4.0.

```
\documentclass[border=1pt]{standalone}
\usepackage{tikz}

\tikzset{
vertex/.style={draw, circle, very thick,
minimum size=1cm},
edge/.style={very thick}
```

Graph Properties / Features

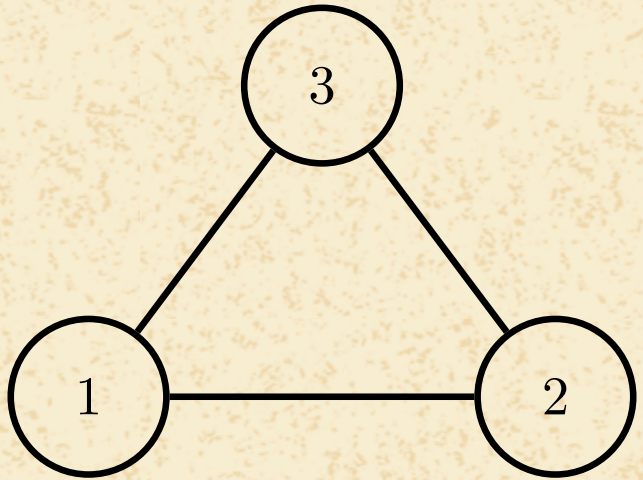
4. The **length** of a path is the number of edges (not the number of vertices) used in the path.

The path $1, 2, 3$ contains two edges: $\{1, 2\}$ and $\{2, 3\}$. Therefore, its length is 2. Miriam Briskman, CC BY-NC 4.0.

```
\documentclass[border=1pt]{standalone}
\usepackage{tikz}

\tikzset{
vertex/.style={draw, circle, very thick,
minimum size=1cm},
edge/.style={very thick}
```

Graph Properties / Features

5. A **cycle** is a closed path in which no edge is repeated and no vertex is repeated except the first and last vertex.



The sequence $1, 2, 3, 1$ forms a cycle because it starts and ends at the same vertex without repeating other vertices or edges. The sequence $1, 2, 3, 1, 2, 3, 1$ is a closed path but not a cycle. Miriam Briskman, CC BY-NC 4.0.

```
\documentclass[border=1pt]{standalone}
\usepackage{tikz}

\tikzset{
vertex/.style={draw, circle, very thick,
minimum size=1cm},
edge/.style={very thick}
```

Graph Properties / Features

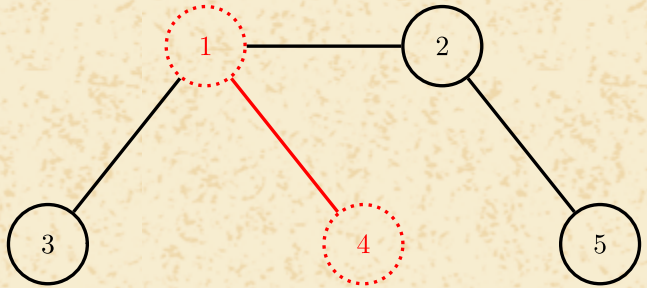
6. Two edges are called **adjacent edges** if they share a common endpoint (= common vertex).

The edges $\{1, 2\}$ and $\{2, 3\}$ are adjacent because they both contain vertex 2. Miriam Briskman, CC BY-NC 4.0.

```
\documentclass[border=1pt]{standalone}
\usepackage{tikz}

\tikzset{
vertex/.style={draw, circle, very thick,
minimum size=1cm},
edge/.style={very thick}
```

Graph Properties / Features

7. The two vertices connected by an edge are called the **endpoints** of the edge.

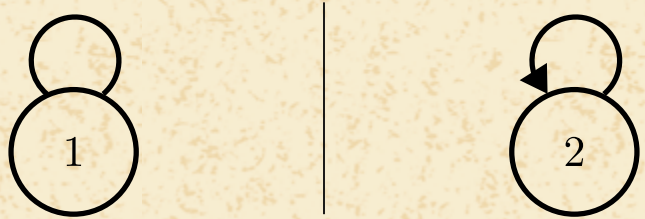


The endpoints of the edge $\{1, 4\}$ are vertices 1 and 4. Miriam Briskman, CC BY-NC 4.0.

```
\documentclass[border=1pt]{standalone}
\usepackage{tikz}

\tikzset{
vertex/.style={draw, circle, very thick,
minimum size=1cm},
edge/.style={very thick}
```

Graph Properties / Features

8. A **loop** (or **self-loop**) is an edge that connects a vertex to itself.



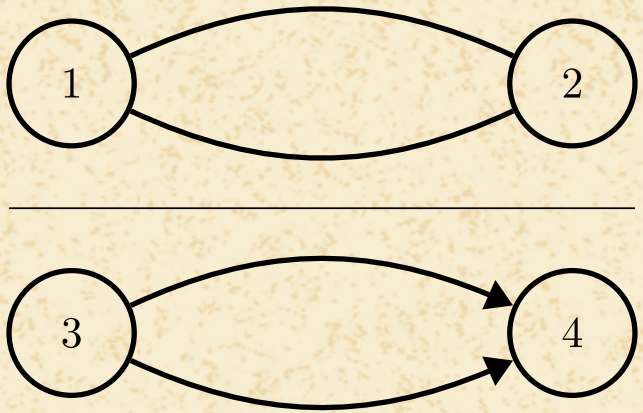
The edge begins and ends at the same vertex (1 on the graph (left), and 2 on the digraph), so it is a self-loop. Miriam Briskman, CC BY-NC 4.0.

```
\documentclass[border=1pt]{standalone}
\usepackage{tikz}
\usetikzlibrary{arrows.meta, bending}

\tikzset{
vertex/.style={draw, circle, very thick,
minimum size=1cm},
```

Graph Properties / Features

9. **Parallel edges** (or **multiple edges**) are distinct edges that connect the same pair of vertices.

In a digraph, parallel edges would be two distinct edges of the form (v_1, v_2) from a vertex v_1 to a vertex v_2.



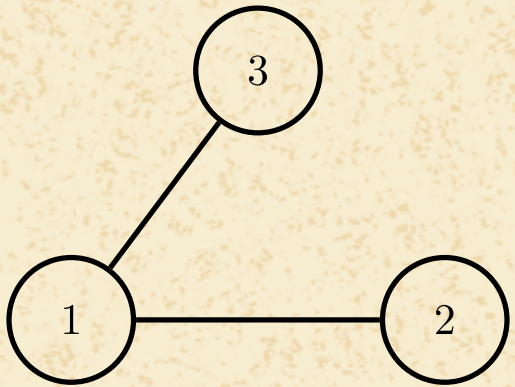
There are two distinct edges connecting vertices 1 and 2 (and, in the digraph, 3 and 4,) so these are parallel edges. Miriam Briskman, CC BY-NC 4.0.

```
\documentclass[border=1pt]{standalone}
\usepackage{tikz}
\usetikzlibrary{arrows.meta, bending}

\tikzset{
vertex/.style={draw, circle, very thick,
minimum size=1cm},
```

Graph Properties / Features

10. The **degree** of a vertex in an undirected graph is the number of edges that are attached to this vertex.

    Note that a loop contributes 2 to the degree of a vertex because the edge both begins and ends at the same vertex.



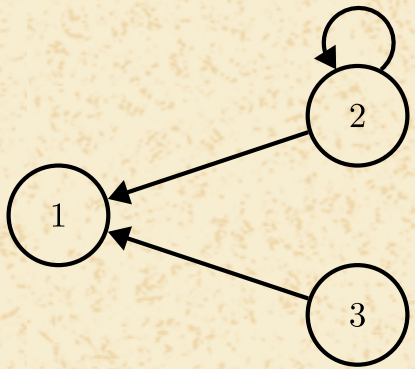
Vertex 1 has the degree of 2 (since two edges are connected to it), while each of vertices 2 and 3 has the degree of 1. Miriam Briskman, CC BY-NC 4.0.

```
\documentclass[border=1pt]{standalone}
\usepackage{tikz}

\tikzset{
vertex/.style={draw, circle, very thick,
minimum size=1cm},
edge/.style={very thick}
```

We denote the degree of a vertex v as $\deg(v)$ `$\text{deg}(v)$`.

Graph Properties / Features

11. In a directed graph:
    - The **in-degree** of a vertex in a directed graph, denoted $\text{indeg}(v)$ `$\text{indeg}(v)$`, is the number of directed edges entering the vertex, and
    - The **out-degree** of a vertex in a directed graph, denoted $\text{outdeg}(v)$ `$\text{outdeg}(v)$`, is the number of directed edges leaving the vertex.

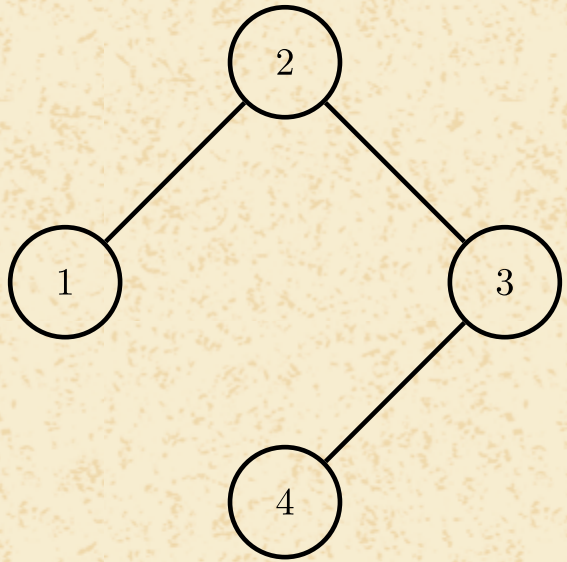


In this digraph, we have: $\text{indeg}(1) = 2$ and $\text{outdeg}(1) = 0$; $\text{indeg}(2) = 1$ and $\text{outdeg}(2) = 2$; and $\text{indeg}(3) = 0$ and $\text{outdeg}(3) = 1$. Miriam Briskman, CC BY-NC 4.0.

```
\documentclass[border=1pt]{standalone}
\usepackage{tikz}
\usetikzlibrary{arrows.meta, bending}

\tikzset{
vertex/.style={draw, circle, very thick,
minimum size=1cm},
```

Graph Properties / Features

12. The **degree sequence** of a graph is the list of vertex degrees written in non-increasing order.



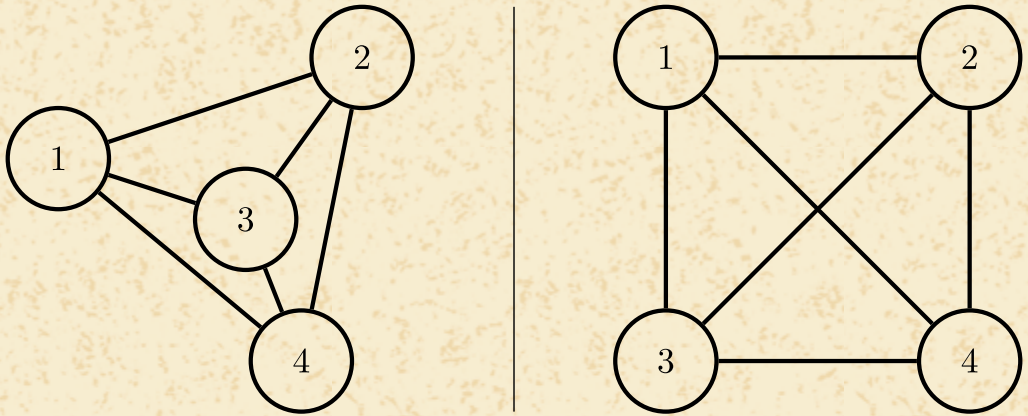
The vertex degrees are $\deg(1) = 1$, $\deg(2) = 2$, $\deg(3) = 2$, and $\deg(4) = 1$, so the degree sequence is $(2, 2, 1, 1)$. Miriam Briskman, CC BY-NC 4.0.

```
\documentclass[border=1pt]{standalone}
\usepackage{tikz}

\tikzset{
vertex/.style={draw, circle, very thick,
minimum size=1cm},
edge/.style={very thick}
```

Graph Properties / Features

13. Two graphs are **isomorphic** if there exists a one-to-one correspondence between their vertices that preserves adjacency.

    In other words, two isomorphic graphs have the same structure even if their drawings or vertex labels are different.

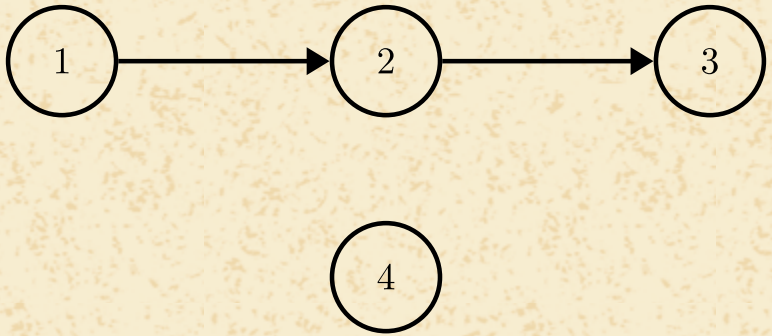


Two isomorphic graphs: just move the vertices around to turn the left graph into the right one. Miriam Briskman, CC BY-NC 4.0.

```
\documentclass[border=1pt]{standalone}
\usepackage{tikz}

\tikzset{
vertex/.style={draw, circle, very thick,
minimum size=1cm},
edge/.style={very thick}
```

Graph Properties / Features

14. In a directed graph, a vertex v_2 is **reachable** from a vertex v_1 if there exists a directed path from v_1 to v_2.



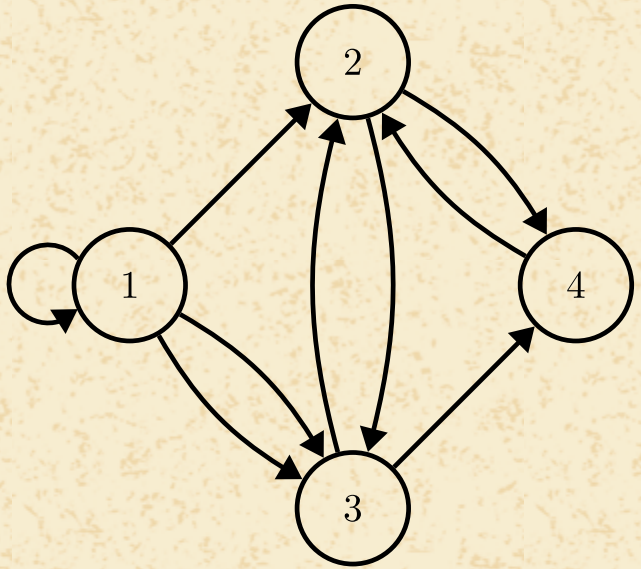
Vertex 3 is reachable from vertex 1, but vertex 4 isn't reachable from any node. Miriam Briskman, CC BY-NC 4.0.

```
\documentclass[border=1pt]{standalone}
\usepackage{tikz}
\usetikzlibrary{arrows.meta}

\tikzset{
vertex/.style={draw, circle, very thick,
minimum size=1cm},
```

Graph Properties / Features

Fun Class Activity:

1. Consider the following digraph D:
    1. List all the edges whose initial vertex is v_1.
    2. List all the edges whose terminal vertex is v_1.
    3. What is the in-degree and out-degree of v_1?
    4. List two different paths from v_1 to v_4.
    5. Does D have:
        1. A cycle?
        2. A loop?
        3. Parallel edges?
    6. Is v_1 reachable from v_4? Explain.



The digraph D. Miriam Briskman, CC BY-NC 4.0.

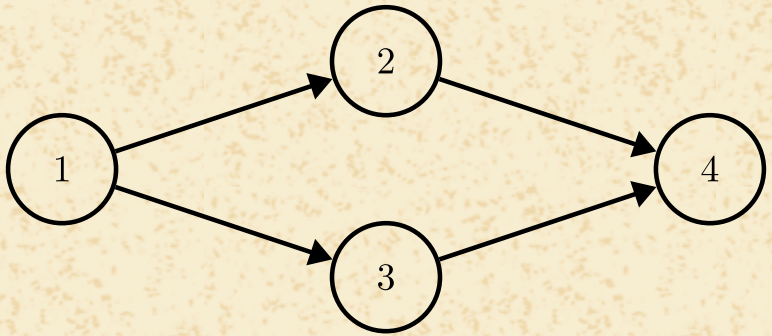
Graph Properties / Features

Any questions so far?

Special Graphs

1. A **Directed Acyclic Graph** (**DAG**) is a directed graph that contains no directed cycles.

   In a DAG, it is impossible to start at a vertex and follow directed edges back to the same vertex.

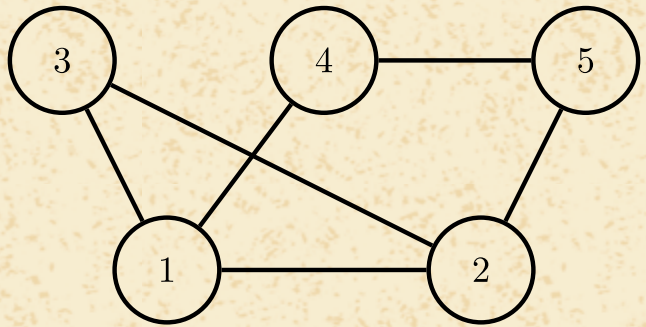


This digraph is acyclic because there is no directed path that returns to its starting vertex. Miriam Briskman, CC BY-NC 4.0.

```
\documentclass[border=1pt]{standalone}
\usepackage{tikz}
\usetikzlibrary{arrows.meta}

\tikzset{
vertex/.style={draw, circle, very thick,
minimum size=1cm},
```

Special Graphs

2. A **simple graph** is a graph that contains:
   - no loops, and
   - no parallel edges.



There are neither self-loops or parallel edges in this graph. Miriam Briskman, CC BY-NC 4.0.

```
\documentclass[border=1pt]{standalone}
\usepackage{tikz}

\tikzset{
vertex/.style={draw, circle, very thick,
minimum size=1cm},
edge/.style={very thick}
```

Special Graphs

3. A **multigraph** is a graph that may contain parallel edges or loops.

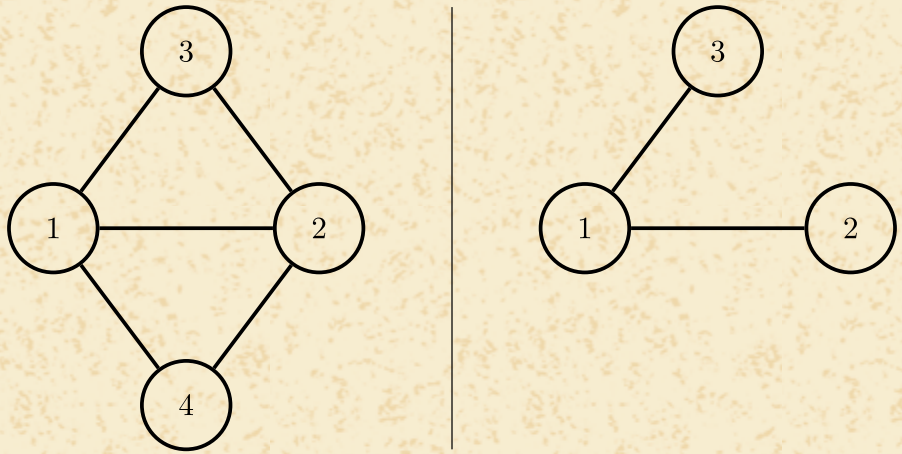
This graph is a multigraph because it contains multiple edges between the same pair of vertices. Miriam Briskman, CC BY-NC 4.0.

```
\documentclass[border=1pt]{standalone}
\usepackage{tikz}

\tikzset{
vertex/.style={draw, circle, very thick,
minimum size=1cm},
edge/.style={very thick}
```

Special Graphs

4. A **subgraph** of a graph G is a graph formed using a subset of the vertices and a subset of the edges of G.

   Meaning, to generate a subgraph of G, you would delete at least one of G's edges or vertices.

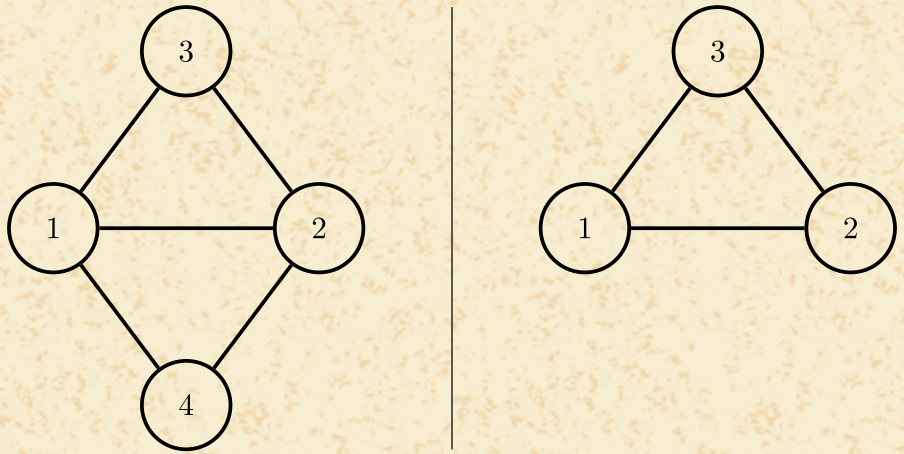


Removing the node 4, the edges connected to it, and one more edge $\{2, 3\}$, create a subgraph (right) of G. Miriam Briskman, CC BY-NC 4.0.

```
\documentclass[border=1pt]{standalone}
\usepackage{tikz}

\tikzset{
vertex/.style={draw, circle, very thick,
minimum size=1cm},
edge/.style={very thick}
```

Special Graphs

5. An **induced subgraph** is formed from a subset of vertices together with **all** edges connecting those vertices in the original graph.



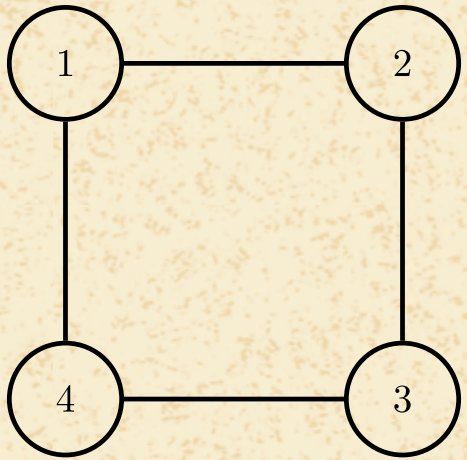
Removing vertex 4 and only the edges that were connected to this vertex 4 (but no other edges) creates an induced subgraph of G. Miriam Briskman, CC BY-NC 4.0.

```
\documentclass[border=1pt]{standalone}
\usepackage{tikz}

\tikzset{
vertex/.style={draw, circle, very thick,
minimum size=1cm},
edge/.style={very thick}
```

Special Graphs

6. A graph is called **regular** if all vertices have the same degrees.

   If every vertex has degree k, the graph is called a k**-regular graph**.



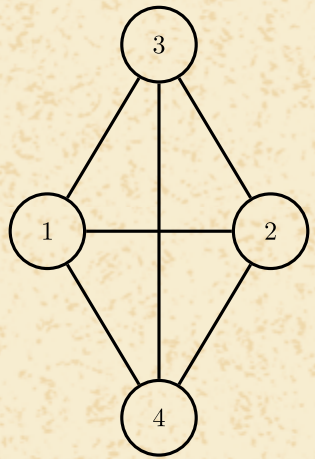
The degree of every node in this graph is 2: this is a 2-regular graph. Miriam Briskman, CC BY-NC 4.0.

```
\documentclass[border=1pt]{standalone}
\usepackage{tikz}

\tikzset{
vertex/.style={draw, circle, very thick,
minimum size=1cm},
edge/.style={very thick}
```

Special Graphs

7. A **complete graph** is a simple graph in which every pair of distinct vertices is connected by an edge.

   The complete graph on n vertices is denoted by K_n.



Every pair of vertices is connected by an edge, so this graph is the complete graph K_4. Miriam Briskman, CC BY-NC 4.0.

```
\documentclass[border=1pt]{standalone}
\usepackage{tikz}

\tikzset{
vertex/.style={draw, circle, very thick,
minimum size=1cm},
edge/.style={very thick}
```

Special Graphs

8. A graph is **connected** if there exists a path between every pair of vertices.

   A **connected component** is a maximal connected subgraph.

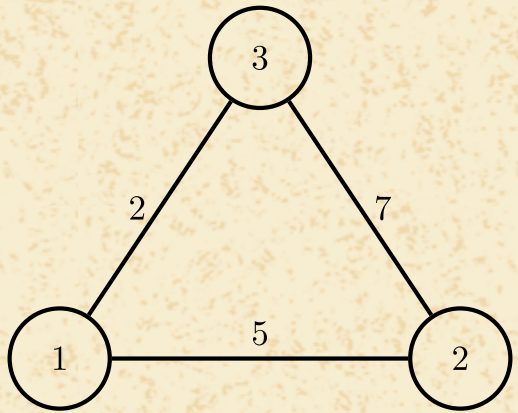
This graph is not connected because there is no path between vertices in the left pair and vertices in the right pair. The graph, however, contains two connected components: $\{1, 2\}$ and $\{3, 4\}$. Miriam Briskman, CC BY-NC 4.0.

```
\documentclass[border=1pt]{standalone}
\usepackage{tikz}

\tikzset{
vertex/.style={draw, circle, very thick,
minimum size=1cm},
edge/.style={very thick}
```

Special Graphs

9. A **weighted graph** is a graph in which edges are assigned numerical values called **weights**.

Weights may represent distances, costs, times, or capacities.



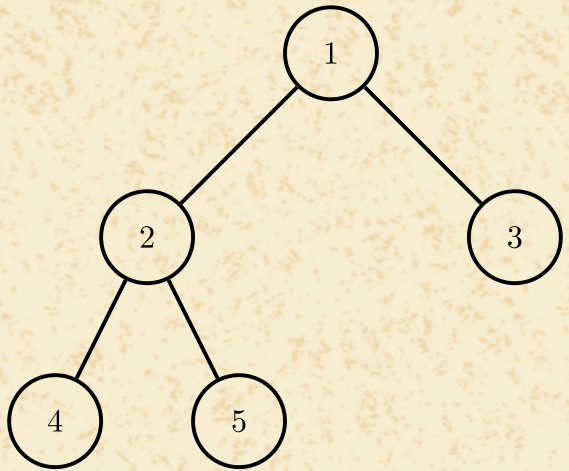
Each edge contains a numerical weight. For example, the edge between 1 and 2 has the weight of 5. Miriam Briskman, CC BY-NC 4.0.

```
\documentclass[border=1pt]{standalone}
\usepackage{tikz}

\tikzset{
vertex/.style={draw, circle, very thick,
minimum size=1cm},
edge/.style={very thick}
```

Special Graphs

10. A **tree** is a connected graph that contains no cycles.

    Equivalently, a tree is a connected acyclic graph.



This graph is a tree because it is connected and contains no cycles. Miriam Briskman, CC BY-NC 4.0.

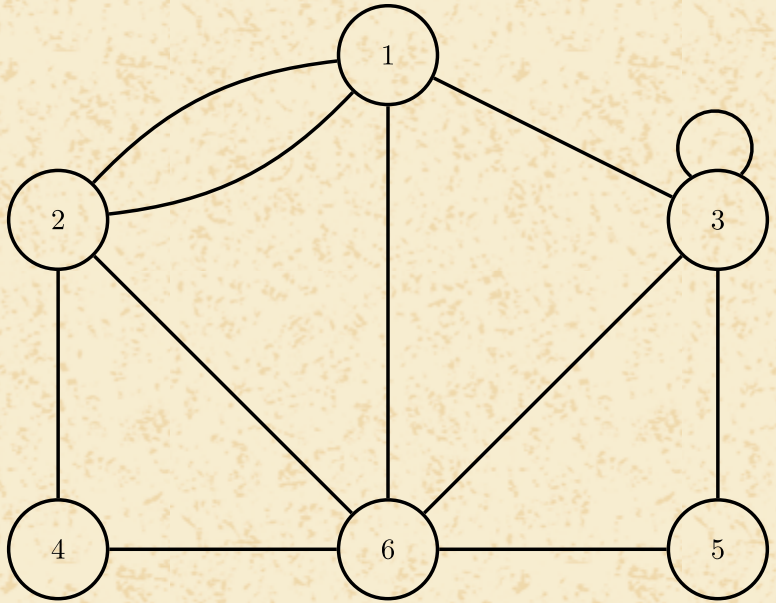
```
\documentclass[border=1pt]{standalone}
\usepackage{tikz}

\tikzset{
vertex/.style={draw, circle, very thick,
minimum size=1cm},
edge/.style={very thick}
```

Special Graphs

Fun Class Activity:

1. Consider the following graph G.
   Is G:
   1. A simple graph?
   2. A multigraph?
   3. A regular graph?
   4. A complete graph?
   5. A connected graph?
   6. A weighted graph?

   Explain your answers.



The graph G. Miriam Briskman, CC BY-NC 4.0.

Special Graphs

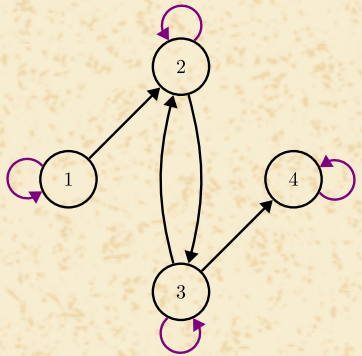
Any questions so far?

Relation Properties via Digraphs

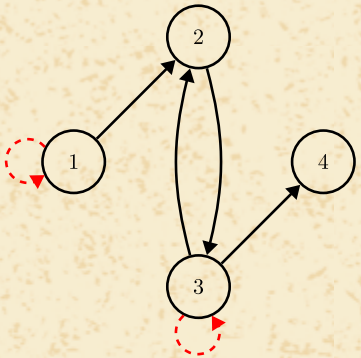
Since, as we learned, relations can be depicted using diagraphs (or graphs,) we can find what properties (reflexive, anti-reflexive, symmetric, anti-symmetric, and transitive) a relation has by looking at its digraph.

1. **Definition.** [Reflexive relations] A relation R on a set S is called **reflexive** if, for every $x \in S$, the pair (x, x) is in R.

   Correspondingly, if <u>every</u> vertex in the digraph D of relation R has a self-loop, then R is reflexive.



A digraph with vertices each having a self-loop: reflexive. Miriam Briskman, CC BY-NC 4.0.

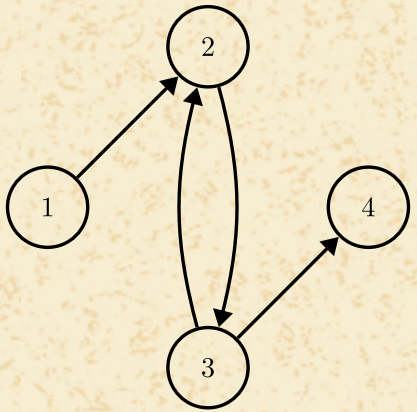


A digraph with at least one vertex without a self-loop: not reflexive. Miriam Briskman, CC BY-NC 4.0.

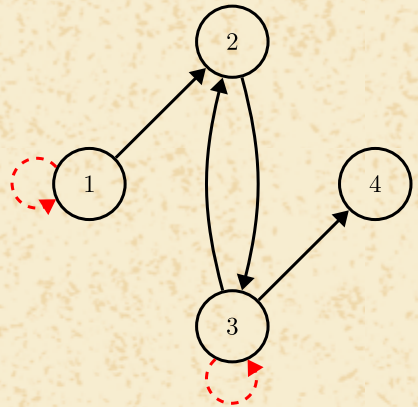
Relation Properties via Digraphs

2. **Definition.** [Anti-reflexive relations] A relation R on a set S is called **anti-reflexive** if, for every $x \in S$, the pair (x, x) is NOT in R.

   Correspondingly, if none of the vertices in the digraph D of relation R has a self-loop, then R is anti-reflexive.



A digraph without any self-loops: anti-reflexive. Miriam Briskman, CC BY-NC 4.0.

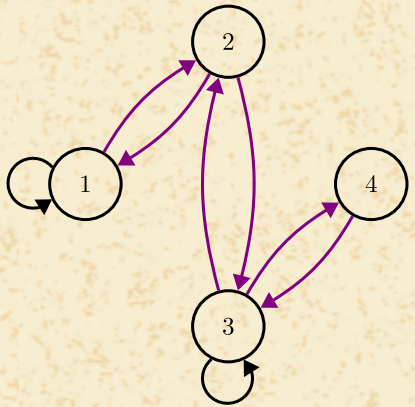


A digraph with at least one vertex with a self-loop: not anti-reflexive. Miriam Briskman, CC BY-NC 4.0.

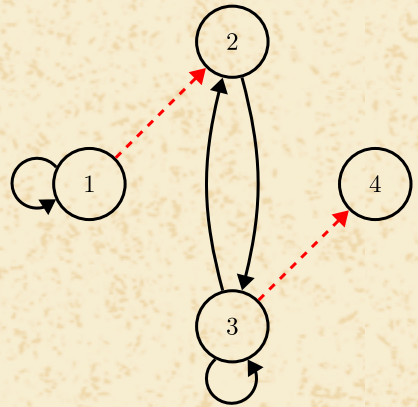
Relation Properties via Digraphs

3. **Definition.** [Symmetric relations] A relation R on a set S is called **symmetric** if, for every pair (x, y) in R (where $x, y \in S$,) the pair (y, x) is also in R.

   Correspondingly, if for every edge (v_1, v_2) in the digraph D of relation R there exists another edge in the opposite direction, (v_2, v_1), then R is symmetric.



A digraph with edge (v_1, v_2) implying that we also have the edge (v_2, v_1): symmetric. Miriam Briskman, CC BY-NC 4.0.

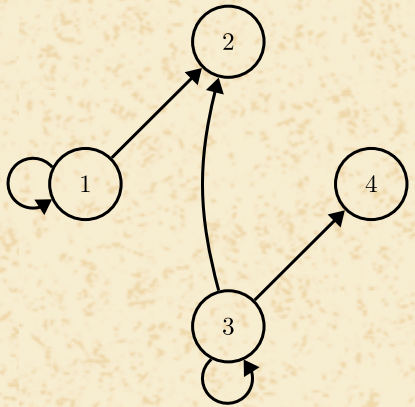


A digraph with at least one edge (v_1, v_2) without the edge (v_2, v_1): not symmetric. Miriam Briskman, CC BY-NC 4.0.

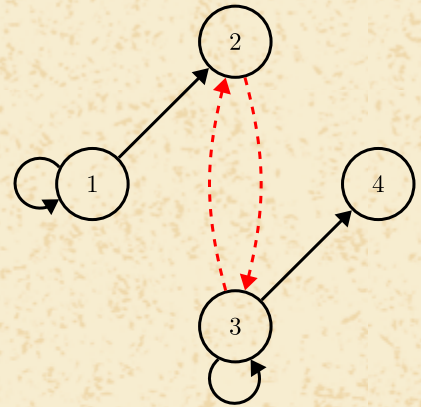
Relation Properties via Digraphs

4. **Definition.** [Anti-symmetric relations] A relation R on a set S is called **anti-symmetric** if the following is true: if the pairs (x, y) and (y, x) are both in R, then $x = y$.

   Correspondingly, if, for every two edges (v_1, v_2) and (v_2, v_1) that exist in the digraph D of relation R, the values of the nodes $v_1 = v_2$ are equal, then R is anti-symmetric.



A digraph with edges (v_1, v_2) and (v_2, v_1) implying $v_1 = v_2$: anti-symmetric. Miriam Briskman, CC BY-NC 4.0.

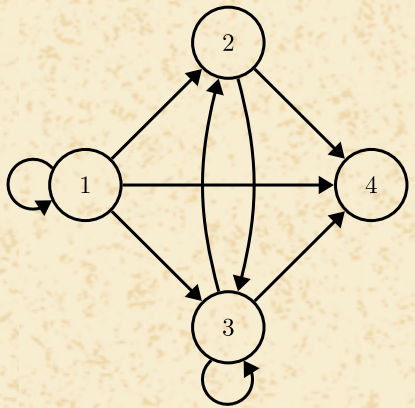


A digraph with at least one pair of edges (v_1, v_2) and (v_2, v_1) but $v_1 \neq v_2$: not anti-symmetric. Miriam Briskman, CC BY-NC 4.0.

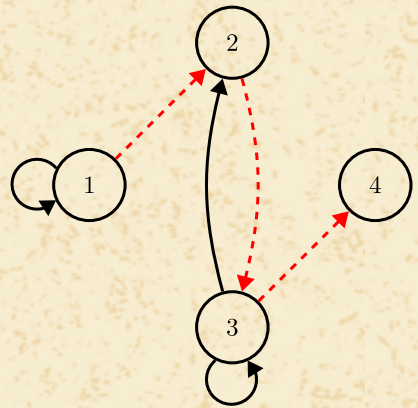
Relation Properties via Digraphs

5. **Definition.** [Transitive relations] A relation R on a set S is called **transitive** if the following is true: if the pairs (x, y) and (y, z) are both in R, then the pair (x, z) is also in R.

   Correspondingly, if, for every two edges (v_1, v_2) and (v_2, v_3) that exist in the digraph D of relation R, the edge (v_1, v_3) also exists in D, then R is transitive.



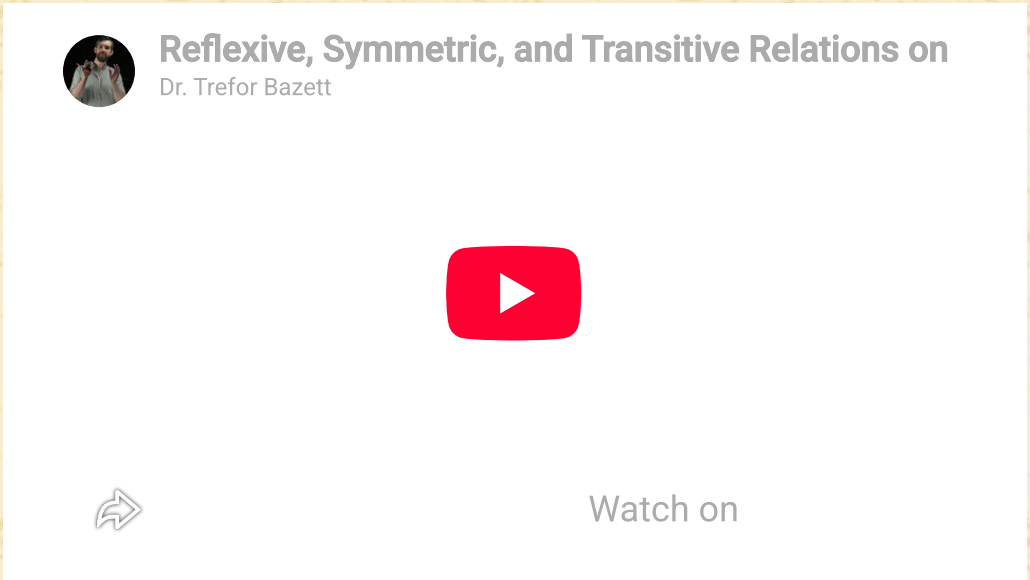
A digraph with edges (v_1, v_2) and (v_2, v_3) implying the existence of (v_1, v_3): transitive. Miriam Briskman, CC BY-NC 4.0.



A digraph with at least one pair of edges (v_1, v_2) and (v_2, v_3) but no edge (v_1, v_3): not transitive. Miriam Briskman, CC BY-NC 4.0.

Relation Properties via Digraphs

The video at https://www.youtube.com/watch?v=q0xN_N7l_Kw briefly explains and exemplifies the properties reflexivity, symmetry, and transitivity of a relation using its digraph:



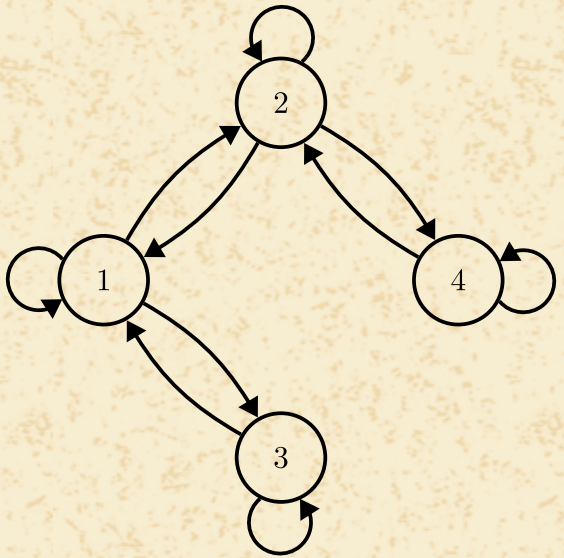
https://www.youtube.com/watch?v=q0xN_N7l_Kw

Relation Properties via Digraphs

Fun Class Activity:

1. Consider the following digraph D.
   Is D:
   1. Reflexive?
   2. Anti-reflexive?
   3. Symmetric?
   4. Anti-Symmetric?
   5. Transitive?

   Explain your answers.



The digraph D. Miriam Briskman, CC BY-NC 4.0.

Any questions so far?

Checking for Graph Isomorphism

As we learned earlier, two graphs are **isomorphic** if they have exactly the same structure, even if they are drawn differently or their vertices have different labels.

Two graphs are considered identical from a graph-theoretic point of view if we can rename the vertices of one graph so that all adjacency relationships are preserved.

Formally:

Definition. [Isomorphism] Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two graphs. An **isomorphism** from G_1 to G_2 is a bijective function $f : V_1 \to V_2$ such that, for every pair of vertices $u, v \in V_1$, we have $(u, v) \in E_1$ if and only if $(f(u), f(v)) \in E_2$.

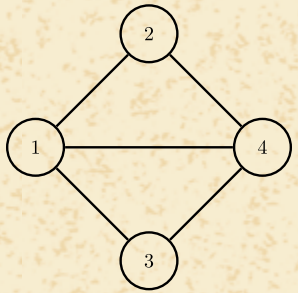
If such a function exists, we write $G_1 \cong G_2$ `$G_1 \cong G_2$`, pronounced "G_1 is isomorphic to G_2".

Checking for Graph Isomorphism

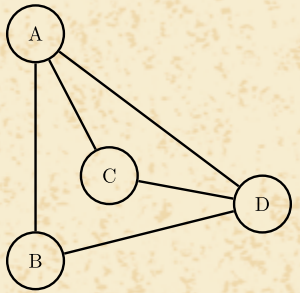
In simple words, the definition of Isomorphism means that:

- Every vertex in G_1 is matched with exactly one vertex in G_2.
- Every edge in G_1 must correspond to an edge in G_2.
- Every non-edge in G_1 must correspond to a non-edge in G_2.
- The structure must remain unchanged.

We will now show that the following two graphs are isomorphic:



The graph G_1. Miriam Briskman, CC BY-NC 4.0.



The graph G_2. Miriam Briskman, CC BY-NC 4.0.

Checking for Graph Isomorphism

To verify their isomorphism, define the bijective function: $f(1) = A$, $f(2) = C$, $f(3) = B$, and $f(4) = D$.

Using this function, note how the following edges correspond:

- $\{1, 2\} \Leftrightarrow \{A, C\}$
- $\{2, 4\} \Leftrightarrow \{C, D\}$
- $\{1, 3\} \Leftrightarrow \{A, B\}$
- $\{3, 4\} \Leftrightarrow \{B, D\}$
- $\{4, 1\} \Leftrightarrow \{D, A\}$

Since the structure is preserved, we have $G_1 \cong G_2$.

Checking for Graph Isomorphism

Before trying to verify isomorphism, we might want to try ruling out isomorphism first. If any of the following properties differ, the graphs cannot be isomorphic:

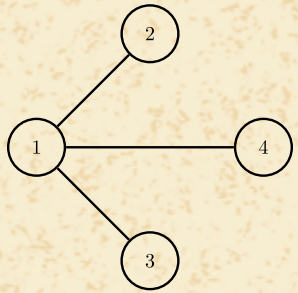
1. Number of vertices
2. Number of edges
3. Degree sequence
4. Number of connected components
5. Number of self-loops
6. Presence of cycles

If all these properties match, the graphs *may* be isomorphic.

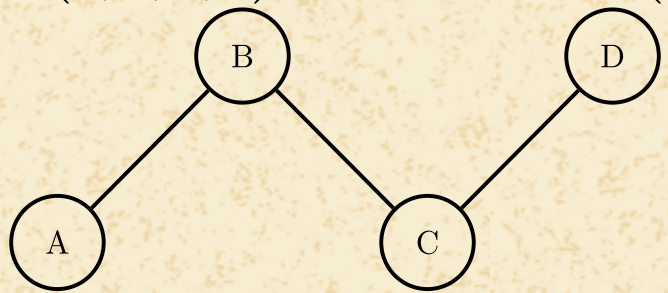
Let's walk through this list and show some examples of graphs that aren't isomorphic.

Checking for Graph Isomorphism

1. **Different Number of Vertices**. If two graphs have different numbers of vertices, they cannot be isomorphic.
   Example: One graph has 4 vertices and the other has 5 vertices.
2. **Different Number of Edges**.
   Example: Graph G_1 has 6 directed edges, while G_2 has 7 directed edges.
3. **Different Degree Sequences**. [Recall that the degree sequence of a graph is the list of vertex degrees arranged in sorted, non-increasing order.]
   Example: The following graphs feature different degree sequences; one has $(3, 1, 1, 1)$ and the other has $(2, 2, 1, 1)$.



The graph G_1 with the degree sequence $(3, 1, 1, 1)$. Miriam Briskman, CC BY-NC 4.0.



The graph G_2 with the degree sequence $(2, 2, 1, 1)$. Miriam Briskman, CC BY-NC 4.0.

Checking for Graph Isomorphism

4. **Different Number of Connected Components**.
   Example: One graph is built of a single connected component (all verices are reachable from each other) and the other has 2 connected components (at least one vertex can't be reached from another vertex).
5. **Different Number of Self-Loops**.
   Example: In graph G_1, 3 vertices have loops, while in G_2, only 2 vertices have loops.
6. **Different Cycle Structures**.
   Example: G_1 has a 5-vertex loop, while G_2 has only 4-vertex loops or shorter than 4-vertex.

Since isomorphic graphs are ones in which the vertices were re-arranged in space (without vertices, edges, or structure being deleted or modified,) we can demonstrate how to 'shift' one graph into another by moving its vertices.

One convenient online app for constructing graphs and beign able to move them in space is Graph Editor by CS Academy. Feel free to use it when working with graphs!

Checking for Graph Isomorphism

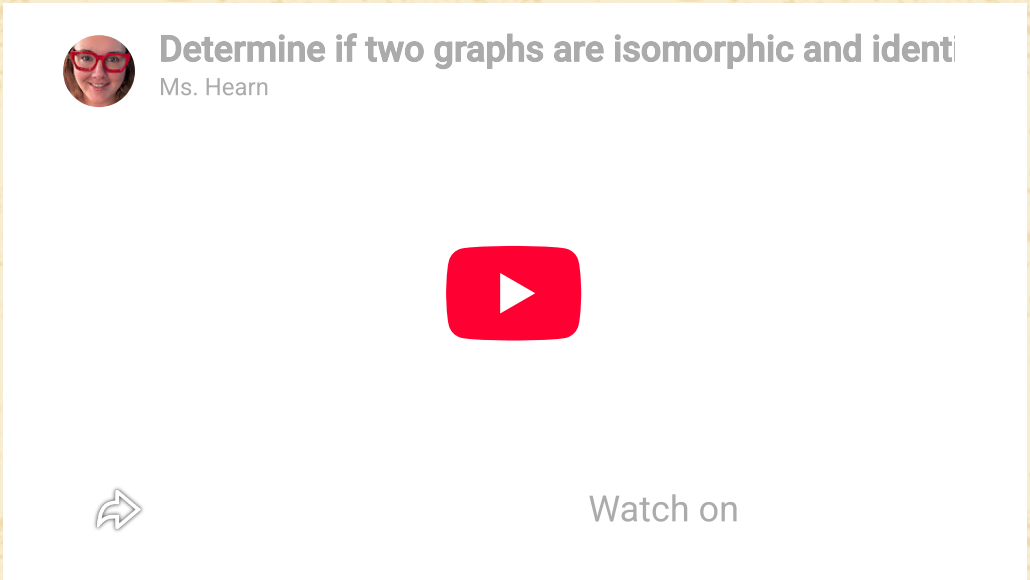
Once you found that two graphs are isomorphic, you can construct the adjacency matrix for both graphs. That is, once you know which nodes in G_1 and in G_2 match, the adjacency matrix for G_1 will be **identical** to the one for G_2 (using the same matching order of vertices.)

For example, the adjacency matrices for graphs G_1 and G_2 from slide 48 are respectively:

$$\text{adj}(G_1) = \begin{bmatrix} & \mathbf{1} & \mathbf{2} & \mathbf{3} & \mathbf{4} \\ \mathbf{1} & 0 & 1 & 1 & 1 \\ \mathbf{2} & 1 & 0 & 0 & 0 \\ \mathbf{3} & 1 & 0 & 0 & 1 \\ \mathbf{4} & 1 & 0 & 1 & 0 \end{bmatrix} \quad \text{and} \quad \text{adj}(G_2) = \begin{bmatrix} & \boldsymbol{A} & \boldsymbol{C} & \boldsymbol{B} & \boldsymbol{D} \\ \boldsymbol{A} & 0 & 1 & 1 & 1 \\ \boldsymbol{C} & 1 & 0 & 0 & 0 \\ \boldsymbol{B} & 1 & 0 & 0 & 1 \\ \boldsymbol{D} & 1 & 0 & 1 & 0 \end{bmatrix}$$

Checking for Graph Isomorphism

The video at https://www.youtube.com/watch?v=RoDR40UG--s shows how we can identify and verify the isomorphism of two given graphs:



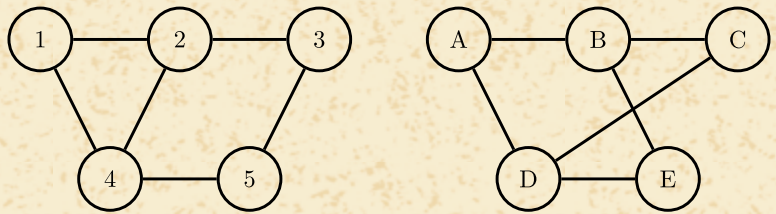
https://www.youtube.com/watch?v=RoDR40UG--s

Checking for Graph Isomorphism

Fun Class Activity:

Consider the following two graphs G_1 and G_2.
Are G_1 and G_2 isomorphic?

- If no, explain why (you could bring one differing feature that prevents isomorphism from those features we've looked into). Be specific by referring to the given graphs' data.
- If yes:
  1. List the matching vertices of the graphs, and
  2. Construct their common adjacency matrix.



Exercise graphs. Miriam Briskman, CC BY-NC 4.0.

Checking for Graph Isomorphism

Any questions?

Graphs & Trees: Sources

Aspnes, James. (2022). Chapter 10. *Notes on Discrete Mathematics*. License: CC BY-SA: Attribution-ShareAlike.

"CS202: Discrete Structures | Units 7 and 8." *Saylor Academy*. License: CC BY: Attribution.

Davis, Stephen. (2019). Chapter 5, Sections 1 and 2. *A Cool Brisk Walk Through Discrete Mathematics*, version 2.2. License: CC BY-SA: Attribution-ShareAlike.

Doerr, Al and Levasseur, Ken. (2024). Chapters 9 and 10. *Applied Discrete Structures*, https://discretemath.org/. License: CC BY-NC-SA: Attribution-NonCommercial-ShareAlike.

Jamaloodee, Mohamed, et al. Chapter 12. *Discrete Math*. Georgia Gwinnett College. License: CC BY-NC: Attribution-NonCommercial.

Levin, Oscar. (2023). Chapters 4.2 and 4. *Discrete Mathematics: An Open Introduction*, https://discrete.openmathbooks.org/dmoi3/, 3rd edition. License: CC BY-SA: Attribution-ShareAlike.