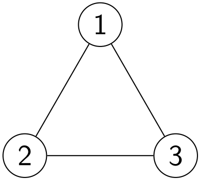
# CMPSC 465: LECTURE X

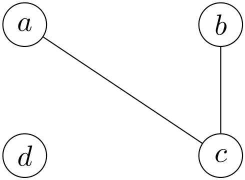
## Graphs and graph algorithms

Ke Chen

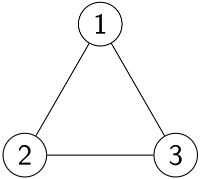
September 22, 2025

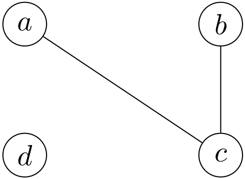
# Graphs



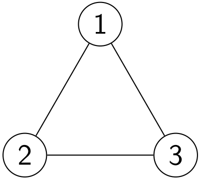
# Graphs

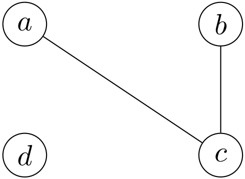



- A graph is defined by a set of vertices $V$ and a set of edges $E$.

# Graphs



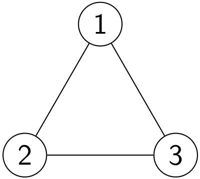



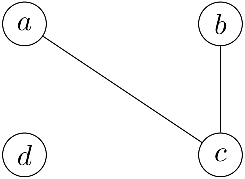

- A graph is defined by a set of vertices $V$ and a set of edges $E$.
- Each edge consists of a pair of vertices.

# Graphs


- A graph is defined by a set of vertices $V$ and a set of edges $E$.
- Each edge consists of a pair of vertices.
- Vertices are also called nodes.

# Graphs



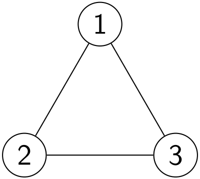




$$V = \{1, 2, 3\}$$

$$E = \{\{1, 2\}, \{2, 3\}, \{3, 1\}\}$$

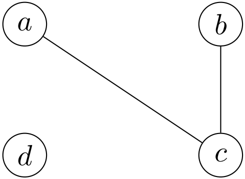
- A graph is defined by a set of vertices $V$ and a set of edges $E$.
- Each edge consists of a pair of vertices.
- Vertices are also called nodes.

# Graphs




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

$$E = \{\{1, 2\}, \{2, 3\}, \{3, 1\}\}$$

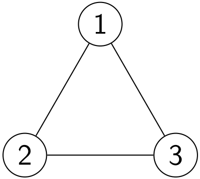


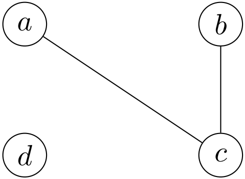

$$V = \{a, b, c, d\}$$

$$E = \{\{a, c\}, \{c, b\}\}$$

- A graph is defined by a set of vertices $V$ and a set of edges $E$.
- Each edge consists of a pair of vertices.
- Vertices are also called nodes.

## Graphs




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

$$E = \{\{1, 2\}, \{2, 3\}, \{3, 1\}\}$$

$$V = \{a, b, c, d\}$$

$$E = \{\{a, c\}, \{c, b\}\}$$

- A graph is defined by a set of vertices $V$ and a set of edges $E$.
- Each edge consists of a pair of vertices.
- Vertices are also called nodes.
- We write $G = (V, E)$ for a graph with vertex set $V$ and edge set $E$.

# Why graphs?

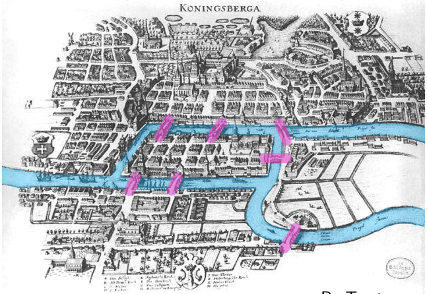
By Jakob Emanuel Handmann

# Why graphs?

Leonhard Euler laid the foundations of graph theory in 1736 while studying the problem of the Seven Bridges of Königsberg.

# Why graphs?

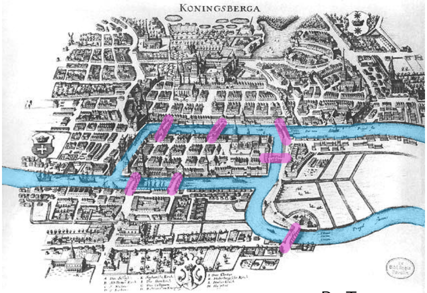
Leonhard Euler laid the foundations of graph theory in 1736 while studying the problem of the Seven Bridges of Königsberg.



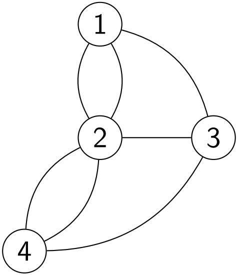

By Twotwos

# Why graphs?

Leonhard Euler laid the foundations of graph theory in 1736 while studying the problem of the Seven Bridges of Königsberg.

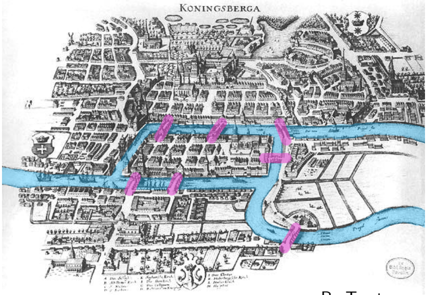



By Twotwos

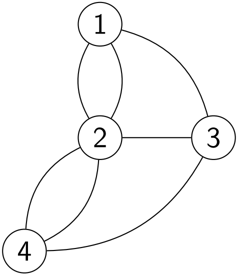


# Why graphs?

Leonhard Euler laid the foundations of graph theory in 1736 while studying the problem of the Seven Bridges of Königsberg.




By Twotwos




Since then graphs have been used to model a variety of things: maps, relationships, constraints, networks, ...

# Undirected vs Directed

Often, graphs are used to model relationships that are not symmetric, such as one-way roads or hyperlinks.

# Undirected vs Directed

Often, graphs are used to model relationships that are not symmetric, such as one-way roads or hyperlinks.

**Idea** Make edges directed.

# Undirected vs Directed

Often, graphs are used to model relationships that are not symmetric, such as one-way roads or hyperlinks.

Idea Make edges directed.

Notation

- set notation for undirected edges: $\{x, y\}$.
- ordered pair for directed edges: $(x, y)$ (from $x$ to $y$)

# Undirected vs Directed

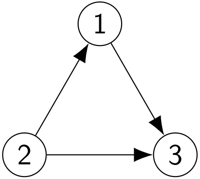
Often, graphs are used to model relationships that are not symmetric, such as one-way roads or hyperlinks.

Idea Make edges directed.

Notation

- set notation for undirected edges: $\{x, y\}$.
- ordered pair for directed edges: $(x, y)$ (from $x$ to $y$)

Example




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

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

# Graph representations

For a graph $G = (V, E)$, there are two standard data structures:

- Adjacency matrix
- Adjacency list

# Graph representations

For a graph $G = (V, E)$, there are two standard data structures:

- Adjacency matrix
- Adjacency list

We create an $|V| \times |V|$ matrix $A$ where

$$A_{ij} = \begin{cases} 1 & \text{if there is an edge from } i \text{ to } j \\ 0 & \text{otherwise} \end{cases}.$$

# Graph representations

For a graph $G = (V, E)$, there are two standard data structures:

- Adjacency matrix
- Adjacency list

We create an $|V| \times |V|$ matrix $A$ where

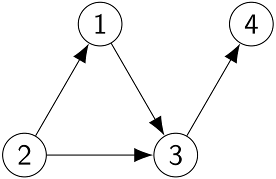
$$A_{ij} = \begin{cases} 1 & \text{if there is an edge from } i \text{ to } j \\ 0 & \text{otherwise} \end{cases}.$$

- $A$ is called the adjacency matrix of $G$.
- If $G$ is undirected, $A$ is symmetric ($A_{ij} = A_{ji}$).
- Diagonal entries are often set to $0$, since we typically work with simple graphs (no self-loop, no multi-edges).

# Graph representations

For a graph $G = (V, E)$, there are two standard data structures:

- Adjacency matrix
- Adjacency list

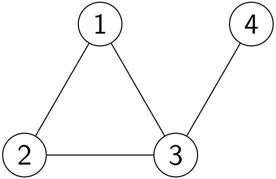



$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

# Graph representations

For a graph $G = (V, E)$, there are two standard data structures:

- Adjacency matrix
- Adjacency list




$$\begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

# Graph representations

For a graph $G = (V, E)$, there are two standard data structures:

- Adjacency matrix
- Adjacency list

We create $|V|$ linked lists, one for each vertex.

The linked list for vertex $v$ contains all the neighbors of $v$, namely, all nodes $u$ that $v$ can reach in one hop.

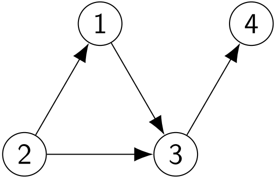
# Graph representations

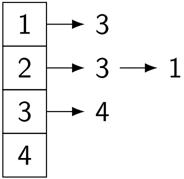
For a graph $G = (V, E)$, there are two standard data structures:

- Adjacency matrix
- Adjacency list

We create $|V|$ linked lists, one for each vertex.

The linked list for vertex $v$ contains all the neighbors of $v$, namely, all nodes $u$ that $v$ can reach in one hop.



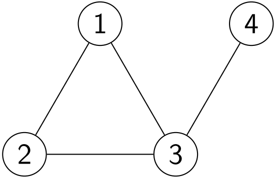
# Graph representations

For a graph $G = (V, E)$, there are two standard data structures:

- Adjacency matrix
- Adjacency list

We create $|V|$ linked lists, one for each vertex.

The linked list for vertex $v$ contains all the neighbors of $v$, namely, all nodes $u$ that $v$ can reach in one hop.




1 → 3 → 2
2 → 3 → 1
3 → 4 → 1 → 2
4 → 3

# Graph representations

For a graph $G = (V, E)$, there are two standard data structures:

- Adjacency matrix
- Adjacency list

| | Adjacency matrix | Adjacency list |
|---|---|---|
| Memory | | |
| Edge Query | | |
| Find all neighbors | | |

# Graph representations

For a graph $G = (V, E)$, there are two standard data structures:

- Adjacency matrix
- Adjacency list

| | Adjacency matrix | Adjacency list |
|---|---|---|
| Memory | $O\left(\lvert V\rvert^2\right)$ | |
| Edge Query | | |
| Find all neighbors | | |

# Graph representations

For a graph $G = (V, E)$, there are two standard data structures:

- Adjacency matrix
- Adjacency list

| | Adjacency matrix | Adjacency list |
|---|---|---|
| Memory | $O\left(\|V\|^2\right)$ | $O\left(\|E\|\right)$ |
| Edge Query | | |
| Find all neighbors | | |

# Graph representations

For a graph $G = (V, E)$, there are two standard data structures:

- Adjacency matrix
- Adjacency list

| | Adjacency matrix | Adjacency list |
|---|---|---|
| Memory | $O\left(\lvert V\rvert^2\right)$ | $O\left(\lvert E\rvert\right)$ |
| Edge Query | $O\left(1\right)$ | |
| Find all neighbors | | |

# Graph representations

For a graph $G = (V, E)$, there are two standard data structures:

- Adjacency matrix
- Adjacency list

| | Adjacency matrix | Adjacency list |
|---|---|---|
| Memory | $O\left(\lvert V\rvert^2\right)$ | $O\left(\lvert E\rvert\right)$ |
| Edge Query | $O\left(1\right)$ | $O\left(\lvert V\rvert\right)$ |
| Find all neighbors | | |

# Graph representations

For a graph $G = (V, E)$, there are two standard data structures:

- Adjacency matrix
- Adjacency list

| | Adjacency matrix | Adjacency list |
|---|---|---|
| Memory | $O\left(\lvert V\rvert^2\right)$ | $O\left(\lvert E\rvert\right)$ |
| Edge Query | $O\left(1\right)$ | $O\left(\lvert V\rvert\right)$ |
| Find all neighbors | $O\left(\lvert V\rvert\right)$ | |

# Graph representations

For a graph $G = (V, E)$, there are two standard data structures:

- Adjacency matrix
- Adjacency list

| | Adjacency matrix | Adjacency list |
|---|---|---|
| Memory | $O\left(\lvert V\rvert^2\right)$ | $O\left(\lvert E\rvert\right)$ |
| Edge Query | $O\left(1\right)$ | $O\left(\lvert V\rvert\right)$ |
| Find all neighbors | $O\left(\lvert V\rvert\right)$ | $O\left(\lvert V\rvert\right)$ |

# Connected component: the undirected case

Let $G = (V, E)$ be an undirected graph.

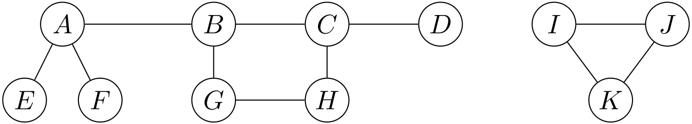
Question Is vertex $v \in V$ connected to vertex $w \in V$?

# Connected component: the undirected case

Let $G = (V, E)$ be an undirected graph.

Question Is vertex $v \in V$ connected to vertex $w \in V$?

Example

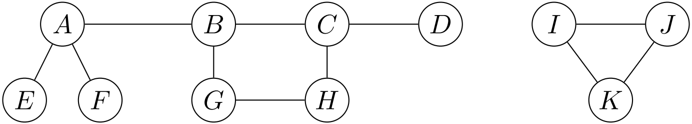


# Connected component: the undirected case

Let $G = (V, E)$ be an undirected graph.

Question Is vertex $v \in V$ connected to vertex $w \in V$?

Example



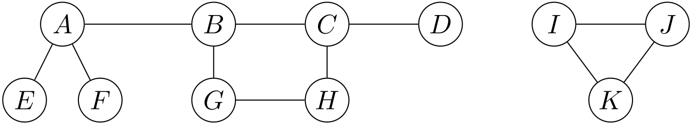

- $A$ is connected to $G$.
- $A$ is connected to $E$.
- $A$ is not connected to $I$, $J$, or $K$.

# Connected component: the undirected case

Let $G = (V, E)$ be an undirected graph.

Question Is vertex $v \in V$ connected to vertex $w \in V$?

Example



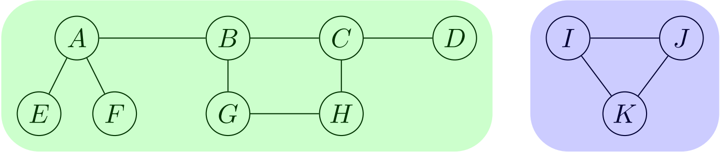

Definition A connected component is a maximal set of connected vertices.

# Connected component: the undirected case

Let $G = (V, E)$ be an undirected graph.

Question Is vertex $v \in V$ connected to vertex $w \in V$?

Example




Definition A connected component is a maximal set of connected vertices.

# Connected component: the undirected case

- There are two standard algorithms for finding the connected components of a graph:
  - Depth First Search (DFS)
  - Breadth First Search (BFS)

# Connected component: the undirected case

- There are two standard algorithms for finding the connected components of a graph:
  - Depth First Search (DFS)
  - Breadth First Search (BFS)
- Once we know the connected components, we also know whether $v$ is connected to $w$ for any pair of vertices $v$ and $w$.

# Connected component: the undirected case

- There are two standard algorithms for finding the connected components of a graph:
  - Depth First Search (DFS)
  - Breadth First Search (BFS)
- Once we know the connected components, we also know whether $v$ is connected to $w$ for any pair of vertices $v$ and $w$.
- DFS and BFS are very powerful algorithms, with interesting properties and applications.