

## 1.1 Real-world Graphs

A graph is a mathematic structure of discrete entities (**vertices**) and the interactions between them (**edges**). Pretty much anything that exists can be described in the notation of graphs. Or, as I like to say, **everything is a graph**.

Real-world graphs are graphs that arise in biology, technology, social science, etc. Commonly studied graphs include road networks, social networks, protein interaction networks, the brain, the Internet, among many many others. However, a lot of the graphs that we'll consider in this class don't fully represent these real-world properties. This is generally done for ease of proofs and understanding.

## 1.2 Basic Definitions

Below are some basic (and more formalized) definitions and terminology that will be used throughout the course. Additional terms and definitions will be introduced as we come across them.

A **graph**  $G$  is a tuple consisting of a set  $V(G)$  of elements called **vertices**, a set  $E(G)$  of pair of vertices called **edges**, and the **endpoint** relations of edges that associate each edge with two vertices. We consider **undirected graphs** for now, in which each edge is a non-directional pairwise relation.

If  $e = (v, u)$  is an edge in  $G$ , then

- $e$  **joins**  $u$  and  $v$ ;  $e$  is **incident** with  $u$  and  $v$ ;
- $u$  and  $v$  are **incident** with  $e$ ;
- $u$  and  $v$  are **endpoints** of  $e$ ;
- $u$  and  $v$  are **adjacent** to each other;
- $u$  and  $v$  are in each others' **neighborhood**.

The **degree** of  $v$  is the number of  $u \subseteq V(G)$  that are adjacent to  $v$ .

The **order** of a graph  $G(V, E)$  is  $|V|$ ; the **size** of  $G(V, E)$  is  $|E|$ . If both  $|V|$  and  $|E|$  are finite,  $G$  is called **finite**. A graph of order  $p$  and size  $q$  is called a  $(p, q)$ -graph.

**Multiple edges** or **multi-edges** are edges which have the same pair of endpoints. **Loops** are edges in which the endpoints are the same vertex.

A **simple graph** has no multiple edges or loops

A **null graph** is a graph with  $V = E = \emptyset$

A **trivial graph** is a graph with  $E = \emptyset$  and  $|V| = 1$

An **empty graph** is a graph with  $E = \emptyset$  and  $|V| \geq 1$

A **path** is a simple graph whose vertices can be listed such that any two vertices are adjacent iff they are consecutive in the list. A **cycle** is a simple graph with an equal number of vertices and edges whose vertices can be placed around a circle and two vertices are adjacent iff they appear consecutively along the circle. A **tree** is a simple connected graph with no cycles.

A **bipartite graph** is a graph which is the union of two disjoint independent sets.

A **complete graph**, or **clique**, is a graph where any two vertices in the graph are adjacent. We denote a clique of size  $n$  by  $K_n$ .

A **complete bipartite graph** or **biclique** with independent sets of sizes  $n$  and  $m$  we denote as  $K_{n,m}$

A **subgraph** of a graph  $G$  is a graph  $H$  that is entirely **contained** in  $G$  ( $H \subseteq G$ ), or that  $V(H) \subseteq V(G)$  and  $E(H) \subseteq E(G)$  with all endpoint assignment being the same.

A bulk of this class will be diving deeper into proofs, algorithms, and interesting properties for each of the above various graph types (and more!). Some of the terminology used we haven't explicitly defined yet, but we will in due time as necessary.

## 1.3 Graph Representation

There are multiple ways to represent a graph. Below are a few examples.

An **adjacency matrix**  $A(G)$  is an  $n \times n$  (where  $n = |V|$ ) matrix where a (positive) nonzero value in each  $a_{i,j}$  indicates that many edges with endpoints from  $v_i$  to  $v_j$ . The sum of nonzeros in a row  $i$  is equal to the degree of  $v_i$ . For a simple graph with no loops, the diagonal will be zeros and the only nonzero appearing will be 1. For undirected graphs, the adjacency matrix will be symmetric.

An **incidence matrix**  $M(G)$  is an  $n \times m$  (where  $n = |V|$  and  $m = |E|$ ) in which a value of 1 in  $m_{i,j}$  indicates that  $v_i$  is incident on edge  $e_j$ . Again the sum of nonzeros in a row  $i$  is the degree of  $v_i$ .

For memory efficiency with most real-world sparse graphs, adjacency/incidence matrices are rarely used in computation. One of the easiest graph representations commonly utilized is to simply store for each vertex its degree and adjacencies (adjacency format). Another common representation is the **compressed sparse row** (CSR) format. It uses two arrays: the first array  $L$  of length  $2|E|$  lists in order adjacencies of  $v_1$  then  $v_2$  then

...  $v_n$ . The second array  $O$  of length  $|V| + 1$  lists offsets for each  $v_i$  to where their adjacencies begin in the first array. The degree for any  $v_i$  can be calculated as  $O[i+1] - O[i]$ . For iterative computations, a CSR format can have a slight locality/cache benefit over the degree-adjacency representation.