15.1 Introduction to Graphs

In this chapter, we're going to learn about the last major data structure for this course: graphs. Picture a social network: there's you and thousands or millions of other users, each with your own accounts and data. The concept that makes social network *social* is the presence of links—relationships—between users, whether that's as friends, followers, subscribers, clients, partners, and more. Or picture a geographic network, like cities and the flight paths connecting them.

Airport map from http://www.martingrandjean.ch/connected-world-air-traffic-network/.

Though social media apps and airports play very different roles in our lives, both rely on being able to represent data in the form of a *network*, consisting of entities and the relationships between them. Over the course of this chapter, we'll study the formal data type that computer scientists use to represent networks, learning both theoretical properties of this data type and one implementation of this data type in Python.

Graphs

Let us start with some basic definitions.

Definition. A **graph** is a pair of sets (V, E), which are defined as follows:

- V is a set of objects. Each element of V is called a **vertex** of the graph, and V itself is called the set of **vertices** of the graph.

- E is a set of pairs of objects from V, where each pair $\{v_1, v_2\}$ is a set consisting of two distinct vertices—i.e., $v_1, v_2 \in V$ and $v_1 \neq v_2$ —and is called an **edge** of the graph.

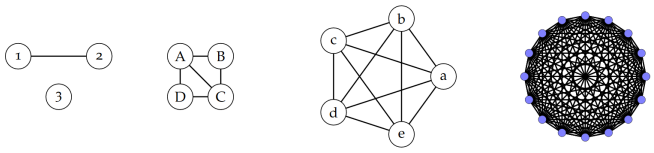
  Order does not matter in the pairs, and so $\{v_1, v_2\}$ and $\{v_2, v_1\}$ represent the same edge.[1]

[1] In future courses, you'll study a variants of graphs called *directed graphs*, where vertex order in an edge does matter.

The conventional notation to introduce a graph is to write $G = (V, E)$, where G is the graph itself, V is its vertex set, and E is its edge set.

Intuitively, the set of vertices of a graph represents a collection of objects, and the set of edges of a graph represent the relationships between those objects. For example, if we wanted to use the terminology of graphs to describe Facebook, we could say that each Facebook user is a *vertex*, while each friendship between two Facebook users is an *edge* between the corresponding vertices.

We often draw graphs using dots to represent vertices, and line segments to represent edges. We have drawn some examples of graphs below.

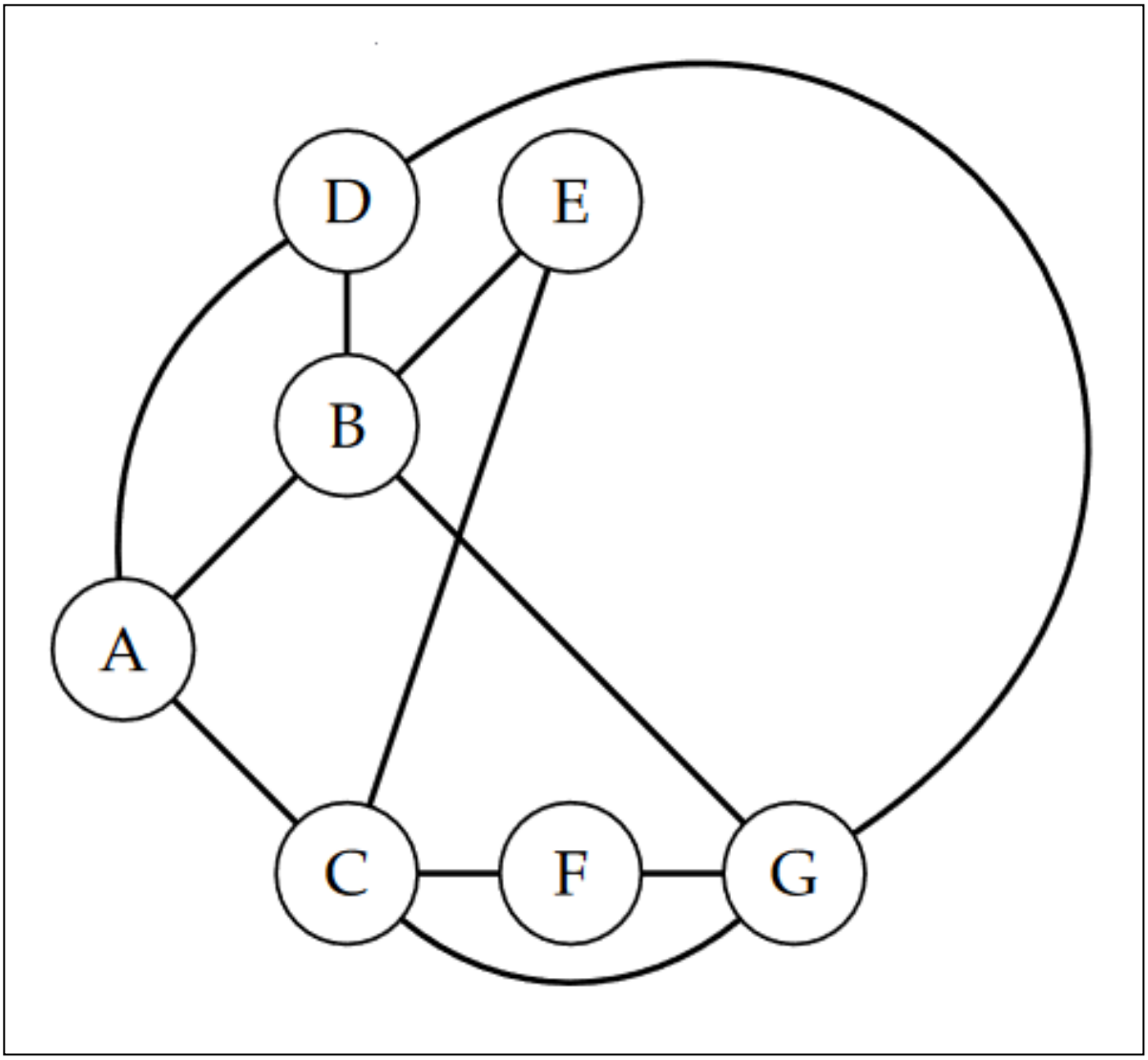


Let's consider an example to practice the above terminology. Consider the graph on the right. This graph consists of *seven* vertices (the circles with letters A through G) and *eleven* edges.

You'll notice that graph diagrams look similar to the tree diagrams. And indeed there are many similarities between graphs and trees, as we'll explore later in this chapter. Right now, you can view graphs as a generalization of trees: rather than enforcing a strict parent-child hierarchy on the data, graphs support any vertex being joined by an edge to any other vertex.

Our next set of definitions introduces one of the key properties of a vertex in a graph: how many edges that vertex is a part of.

Definition. Let $G = (V, E)$, and let $v_1, v_2 \in V$. We say that v_1 and v_2 are **adjacent** if and only if there exists an edge between them, i.e., $\{v_1, v_2\} \in E$. Equivalently, we can also say that v_1 and v_2 are **neighbours**.[2]



[2] Remember that order doesn't matter in the edge pairs, so this is a *symmetric* relationship.

Definition. Let $G = (V, E)$, and let $v \in V$. We say that the **degree** of v, denoted $d(v)$, is its number of neighbours, or equivalently, how many edges v is a part of.

For example, in our example graph above, we could say that the vertices A and B are adjacent, but A and G are not. The degree of vertex A is 3, since it has three neighbours: B, C, and D.

Paths and connectedness

Often when we use graphs in modelling the real world, it is not sufficient to capture just a single relationship between entities. Our goal now is to use individual edges, which represent some sort of relationship between vertices, to build up extended, indirect connections between vertices. In a social network, for example, we want to be able to go from friends to "friends of friends," and even "friends of friends of friends of friends." In a

graph representing roads between cities, we want to be able to go from "a route between cities using one road" to "a route between cities using k roads." We use the following definitions to make precise these notions of "indirect" relationships.

Definition. Let $G = (V, E)$ and let $u, u' \in V$. A **path** between[3] u and u' is a sequence of *distinct* vertices $v_0, v_1, v_2, \ldots, v_k \in V$ which satisfy the following properties:

[3] Like edges, paths are directionless; a path from u to u' is also a path from u' to u.

- $v_0 = u$ and $v_k = u'$. (The endpoints of the path are u and u'.)
- Each consecutive pair of vertices are adjacent. (So v_0 and v_1 are adjacent, and so are v_1 and v_2, v_2 and v_3, etc.)

We allow k to be zero; this path would be just a single vertex v_0.

The **length** of a path is one less than the number of vertices in the sequence (so the above sequence would have length k); more intuitively, the length of the path is the number of *edges* which are used by this sequence.

We say that u and u' are **connected** when there exists a path between u and u'.[4] Because we allow zero-length paths, a vertex is always connected to itself.

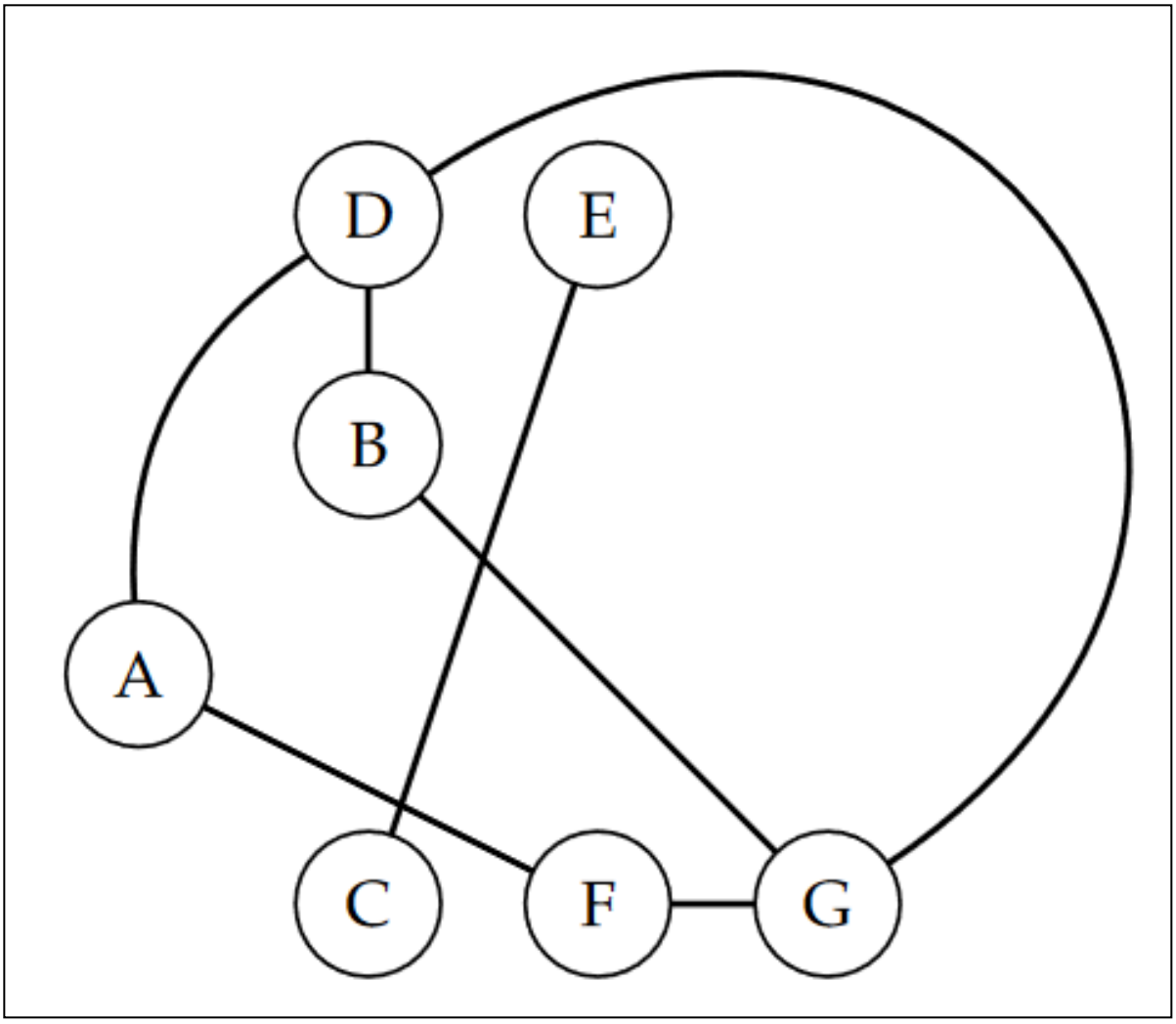
[4] This definition is existentially-quantified; there could be more than one path between u and u'.

We say that graph G is **connected** when for all pairs of vertices $u, v \in V$, u and v are connected.

Being connected is a fundamental property of graphs. Imagine, for example, a geographical representation where each graph vertex is a city, and each edge a road between two cities. If this graph is not connected, then there is at least one pair of cities for which it is not possible to get from one to the other by road.

An example

Consider the graph on the right. Let's practice using the above terminology by answering



the following questions.

1. Are the vertices A and B adjacent?

   Answer: No, because there is no edge between them.

2. Are the vertices A and B connected?

   Answer: Yes, there is a path between them. For example, A, F, G, B.[5]

   [5] Note that there is more than one path between A and B. The definition of connectedness between two vertices is *existentially-quantified*, so we only are asking whether there exists a path between them, not how many such paths there are.

3. What is the length of the shortest path between vertices B and F?

   Answer: There is a path of length two between B and F: B, G, F. How do we know this is the shortest one? The only path of length one that could be between B and F is simply the sequence B, F; but this is *not* a path because B and F are not adjacent.

And finally, let's do a short proof involving our definition of connectedness.

Example. Prove that this example graph is not connected.

Translation. Let us first translate the statement "this graph is not connected." We'll let $G = (V, E)$ refer to this graph (and corresponding vertex and edge sets). So we can write this statement as "G is not connected," but that's not very illuminating. Let us unpack the definition of connected for graphs, which requires every pair of vertices in the graph to be connected:[6]

[6] This is both a review of logical manipulation rules and of practicing unpacking definitions!

$$
\begin{aligned}
& G \text{ is not connected} \\
\iff\ & \neg(G \text{ is connected}) \\
\iff\ & \neg(\forall u, v \in V,\ u \text{ and } v \text{ are connected}) \\
\iff\ & \exists u, v \in V,\ u \text{ and } v \text{ are not connected} \\
\iff\ & \exists u, v \in V,\ \text{there is no path between } u \text{ and } v
\end{aligned}
$$

We actually went a step further and unpacked the definition of connected for vertex pairs as well. Hopefully this makes it clear what it is we need to show: that there exist two vertices in the graph which do not have a path between them.

Proof. Let $G = (V, E)$ be the above graph. Let u and v be the vertices labelled E and B, respectively. We will show that there does not exist a path between u and v.

Suppose for a contradiction that there exists a path $v_0, v_1, \ldots, v_k$ between u and v, where $v_0 = E$. Since v_0 and v_1 must be adjacent, and C is the only vertex adjacent to E, we know that $v_1 = C$. Since we know $v_k = B$, the path cannot be over yet; i.e., $k \geq 2$.

So what about v_2? By the definition of *path*, we know that v_2 must be adjacent to C, and must be distinct from E and C. But the only vertex that's adjacent to C is E, and so v_2 cannot exist, which gives us our contradiction. ∎