# Learning Objectives

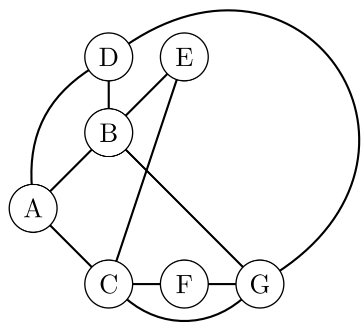
By the end of this worksheet, you will:

- Apply basic graph definitions to answer questions about properties of graphs.

---

1. **Terminology review.** One of the tricky things about learning graphs is that there's a lot of terminology to understand. This exercise will give you the opportunity to practice reading and using this terminology.

   Consider the graph below.




   (a) How many vertices does this graph have?

   > **Solution**
   >
   > There are seven vertices.

   (b) How many edges does this graph have?

   > **Solution**
   >
   > There are eleven edges.

   (c) List all the vertices that are adjacent to vertex G.

   > **Solution**
   >
   > B, C, D, F.

   (d) What is the distance between A and G? Is there more than one shortest path between A and G?

   > **Solution**
   >
   > The distance is 2. There are actually three shortest paths between A and G: [A, B, G], [A, C, G], [A, D, G].

   (e) Find a path that goes through all vertices of the graph. (Remember that a path cannot have any duplicate vertices.)

   > **Solution**
   >
   > Many different options, e.g., [A, B, D, G, F, C, E].

2. **Vertex degree.** Consider the following definition.

**Definition 1** (degree)**.** Let $G = (V, E)$ be a graph, and $v$ be a vertex in $V$. The **degree** of $v$, denoted $d(v)$, is the number of neighbours of $v$.

Answer the following questions about this definition.

(a) In the graph on the previous page, what is the degree of vertex D?

**Solution**

3. (The neighbours of D are A, B, and G.)

(b) In the graph on the previous page, which vertex/vertices have the largest degree?

**Solution**

B, C, and G all have degree 4.

(c) Let $G = (V, E)$ be a graph, and assume that for all $v \in V$, $d(v) \leq 5$. Find and prove a good upper bound (exact, not asymptotic) on the total number of edges, $|E|$, in terms of the number of vertices, $|V|$.

Formally, you can think of this as proving the following statement (after filling in the blank):

$$\forall G = (V, E),\ \big(\forall v \in V,\ d(v) \leq 5\big) \Rightarrow |E| \leq \underline{\hspace{3cm}}$$

[Note: once you have the right number in mind, the proof isn't computationally complex. Think about trying to count the number of edges a particular vertex can be an endpoint for.]

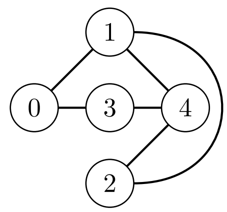
**Solution**

*Proof.* Let $G = (V, E)$ be a graph, and assume that for all $v \in V$, $d(v) \leq 5$. We'll prove that $|E| \leq 5|V|$. Our assumption tells us that every vertex is an endpoint for at most 5 edges. So then in total, all of the vertices can be an endpoint for at most $5|V|$ edges. Since every edge must have an endpoint, this means that the number of edges is at most $5|V|$. $\square$

*Note*: in fact, every edge must have exactly *two* distinct endpoints. So since there are at most $5|V|$ possible endpoint spots, this means that the number of edges is at most $\frac{5}{2}|V|$.

3. **Adjacency matrix.** Suppose we have a graph $G = (V, E)$, where the vertex set is $\{0, 1, \ldots, n-1\}$ for some $n \in \mathbb{N}$. One of the most common ways to represent such a graph in a computer program is by using a two-dimensional array $A$, where $A[i][j]$ is set to 1 if vertices $i$ and $j$ are adjacent, and 0 if they aren't. This graph representation is known as the *adjacency matrix*.

   (a) Consider the graph below.




   Write down the adjacency matrix of this graph. The first row (which shows the adjacency relationships of vertex 0) has been done for you:

   **Solution**

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

   (b) Write an algorithm which does the following:

   - Given as input an adjacency matrix $A$ representing a graph, and a number $i$ representing a vertex in the graph,
   - the algorithm returns the degree of the given vertex in the given graph.

   **Solution**

```python
def find_degree(A: List[List[int]], i: int) -> int:
    """Note: we represent an adjacency matrix as a list of lists."""
    d = 0
    for j in A[i]:
        d = d + j       # Note: j is either 0 or 1.
    return d
```