

CS350: Data Structures

Graphs

James Moscola

Department of Physical Sciences

York College of Pennsylvania



Graph Data Structure

- A **graph** is a data structure consisting of a set of **vertices** connected by **edges**

$$G = (V, E)$$

where V is the set of vertices and E is the set of edges

- The following can all be considered special cases of a graph data structure
 - Linked lists
 - Trees
 - Skip lists

Graph Data Structure

- **Graphs have many uses:**
 - Representing the control-flow of a program
 - Underlying data structure for mapping/navigation systems
 - Each vertex represents an intersection
 - Each edge represents a road between intersections
- **A node and/or an edge in a graph may have some additional information associated with it**
 - For the mapping system:
 - Each vertex may have a GPS coordinate associated with it
 - Each edge may have a distance and/or a speed-limit associated with it
 - The value or values associated with an edge are sometimes referred to as the **weight** of the edge

Graph Data Structure

- A graph can be defined as follows:

$$G = (V, E)$$

where V is the set of vertices and E is the set of edges

- Each edge is a pair (v, w) where $v, w \in V$

- In a **directed graph**, the ordering of an edge pair matters

- $(v, w) \neq (w, v)$

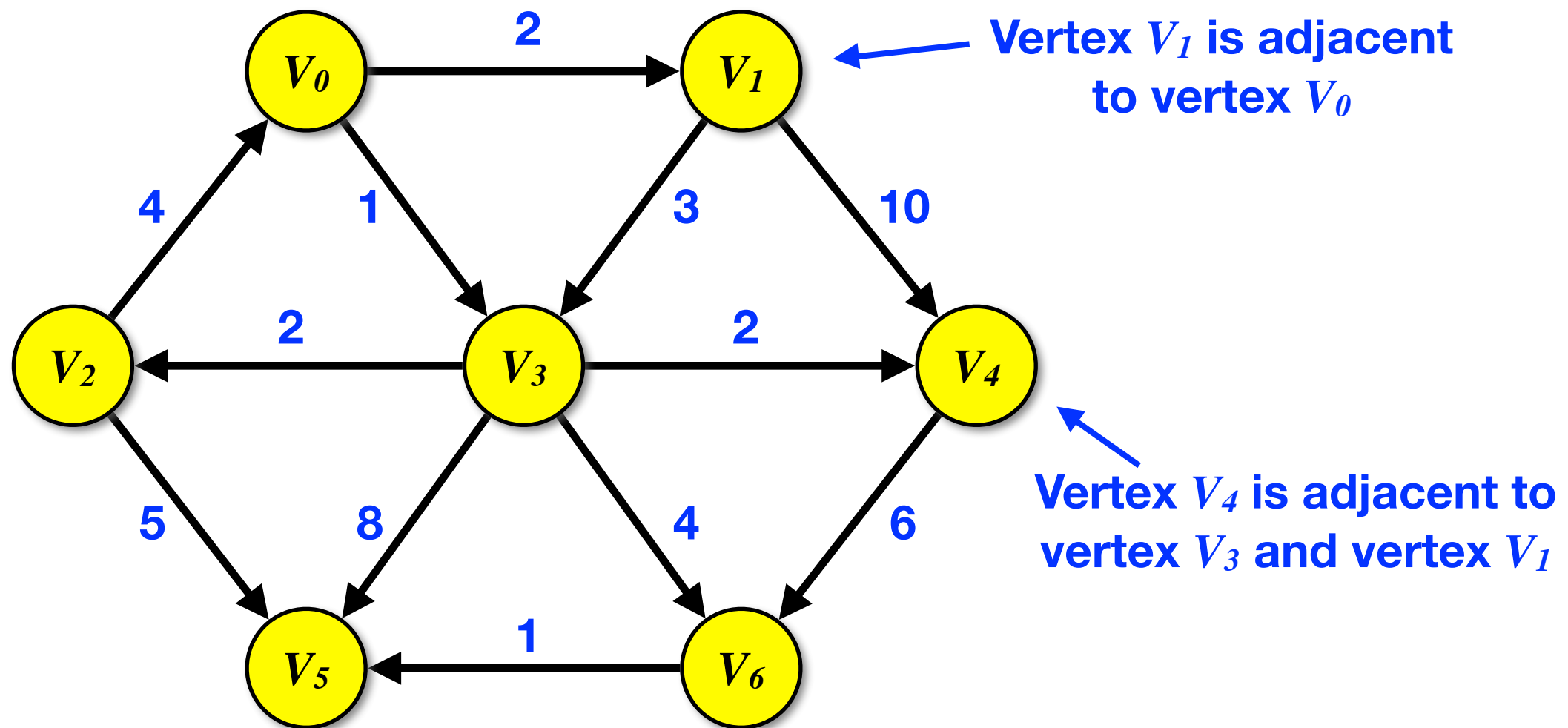


- In an **undirected graph**, the ordering of an edge pair does not matter

- $(v, w) = (w, v)$

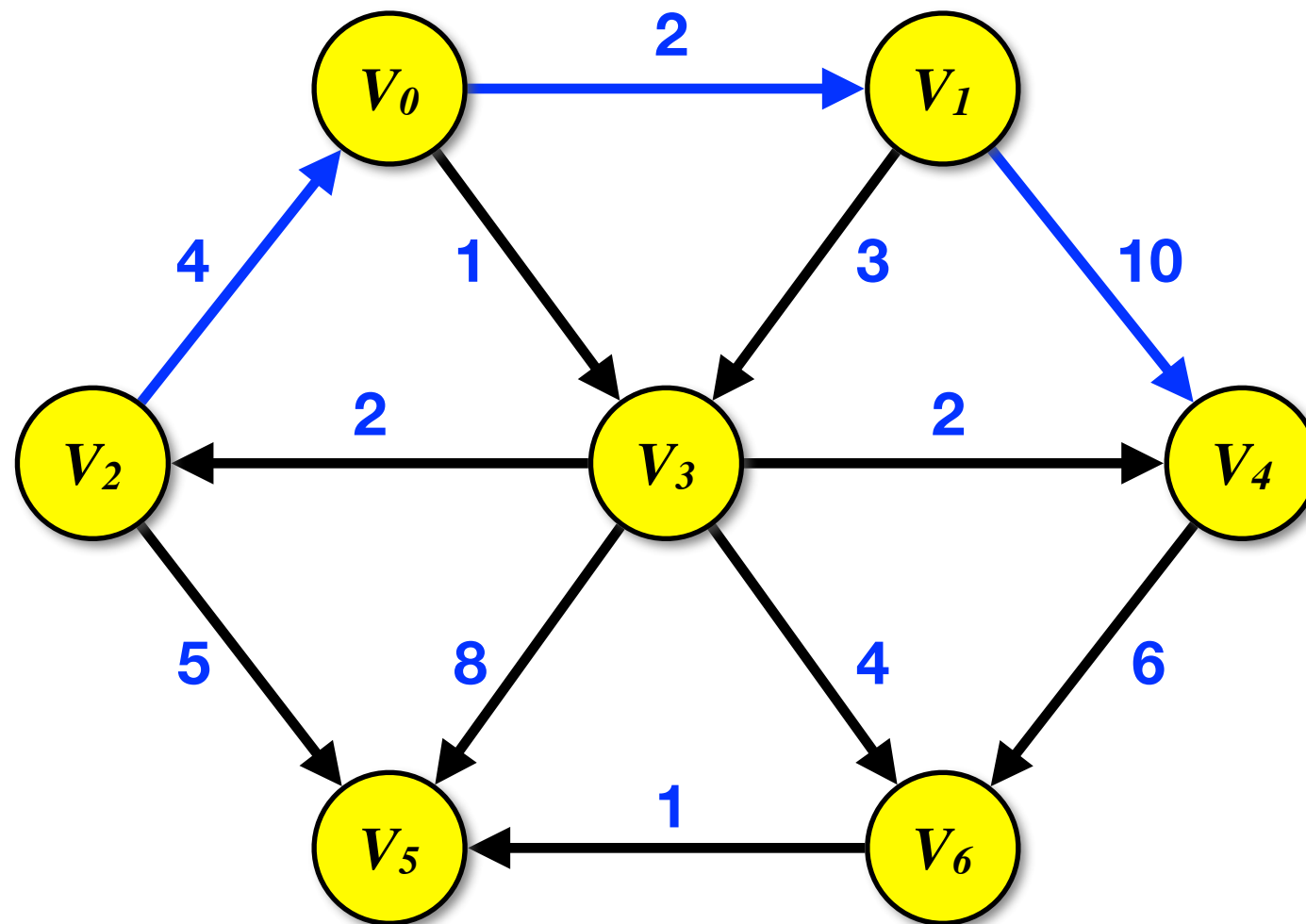


Example of a Directed Graph with Weights



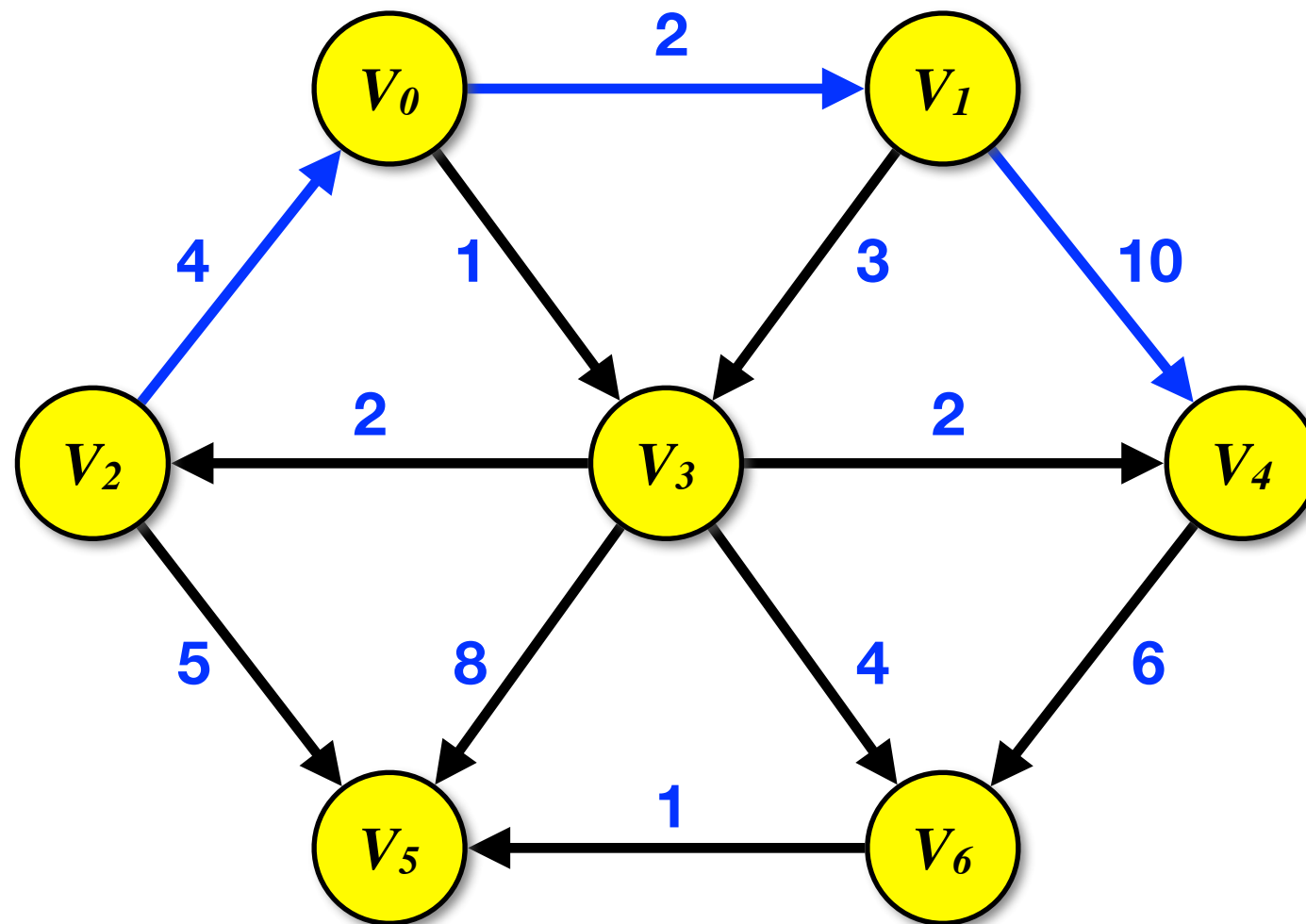
$$E = \left\{ (V_0, V_1, 2), (V_0, V_3, 1), (V_1, V_3, 3), (V_1, V_4, 10), \right. \\ \left. (V_3, V_4, 2), (V_3, V_6, 4), (V_3, V_5, 8), (V_3, V_2, 2), \right. \\ \left. (V_2, V_0, 4), (V_2, V_5, 5), (V_4, V_6, 6), (V_6, V_5, 1) \right\}$$

Example of a Directed Graph with Weights



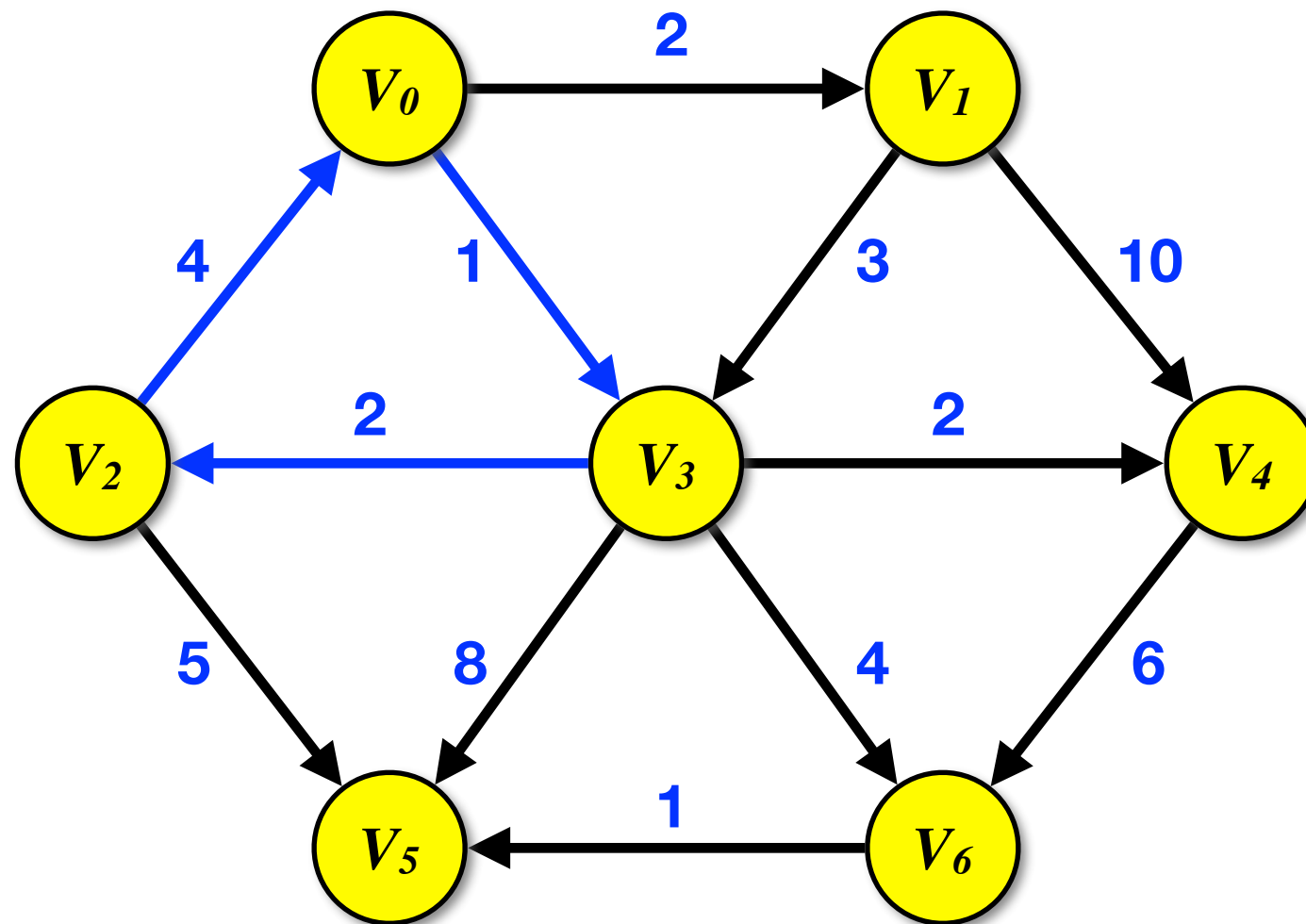
- The **path length** between two vertices is the number of edges that must be traversed to get from one vertex to the other
 - Multiple paths may exist between two vertices
 - Example: path length from V_2 to V_4 is 3 -- ($V_2 \rightarrow V_0 \rightarrow V_1 \rightarrow V_4$)

Example of a Directed Graph with Weights



- The **weighted path length** between two vertices is the sum of the weights of the edges along the path
 - Multiple paths may exist between two vertices
 - Example: weighted path length from V_2 to V_4 is 16 (or 7, or 11, ...)

Example of a Directed Graph with Weights

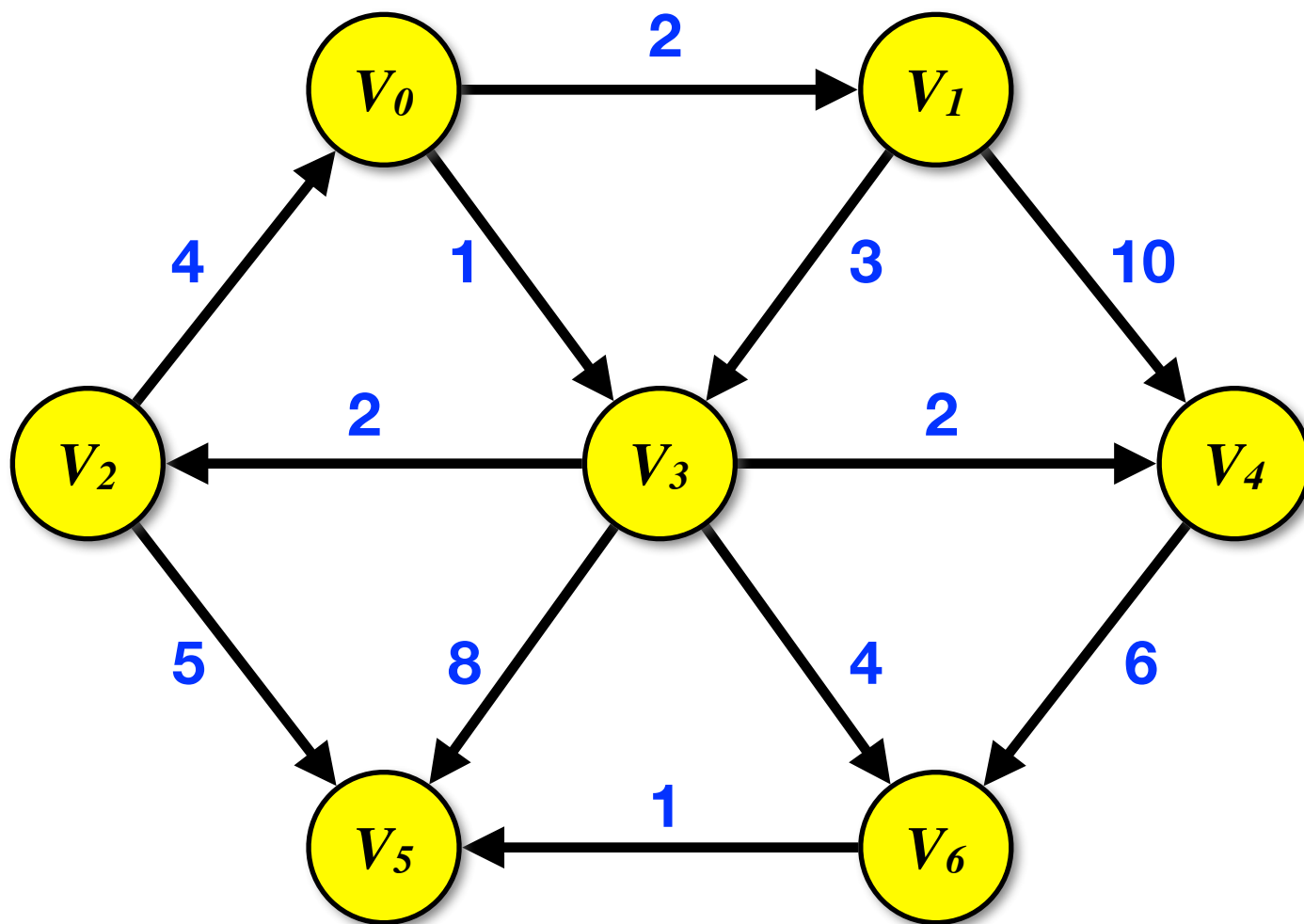


- A **cycle** in a directed graph is a path that begins and ends at the same vertex and contains at least one edge
 - A directed graph may have zero, one, or more cycles
 - A graph with cycles is said to be **cyclic**, whereas a graph with no cycles is said to be **acyclic**

Graph Representations

- **There are two common representations for graphs, the first is an adjacency matrix**
 - Utilizes a 2-dimensional matrix, where each vertex is represented by both a row and a column
 - Each location in the matrix, $\text{mat}[v][w]$, represents the weight of a directed edge between vertex v and vertex w
 - If no edge exists between a vertex v and a vertex w , then set the edge weight in the matrix to **INFINITY**
 - Constant time operation to find info about any edge
 - Requires $O(|V|^2)$ space to store matrix
 - Can be wasteful if representing a **sparse graph**, that is, a graph without many edge

Adjacency Matrix Representation



	V	V	V	V	V	V	V
V	∞	2	∞	1	∞	∞	∞
V	∞	∞	∞	3	10	∞	∞
V	4	∞	∞	∞	∞	5	∞
V	∞	∞	2	∞	2	8	4
V	∞	∞	∞	∞	∞	∞	6
V	∞	∞	∞	∞	∞	∞	∞
V	∞	∞	∞	∞	∞	1	∞

Graph Representations

- **Another graph representation is the adjacency list**
 - The graph is represented as a list of edges
 - For each vertex, keep a list of all adjacent vertices
 - Each edge appears in an adjacency list, thus the space required is $O(|E|)$
 - Better suited for sparse graphs
 - May take additional time to search lists to see if an edge exists between two vertices

Adjacency List Representation

