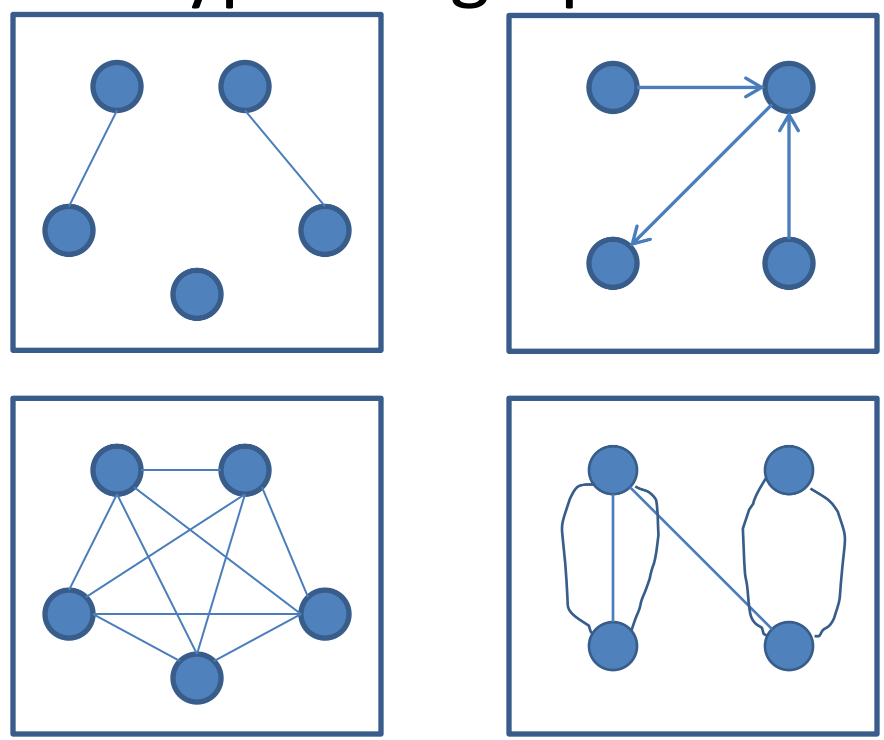
CSCI 340 Data Structure and Algorithm Analysis

Graphs Part I

Basic concepts

Graph

- Graph is a collection of nodes and the connections between them.
- There are many different types of graphs:
  - Simple graph
  - Directed graph
  - Multi-graph
  - Weighted graph
  - Complete graph
  - …

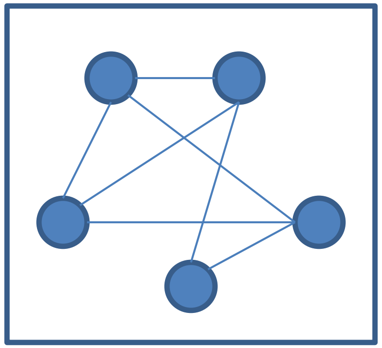


Simple Graph

- A simple graph $G=(V,E)$ consists of a nonempty set V of vertices and a possibly empty set E of edges. Each edge connects two vertices from V:

$$\{v_i,\ v_j\} = \{v_j,\ v_i\}$$

- $|V|$ denotes the number of vertices
- $|E|$ denotes the number of edges

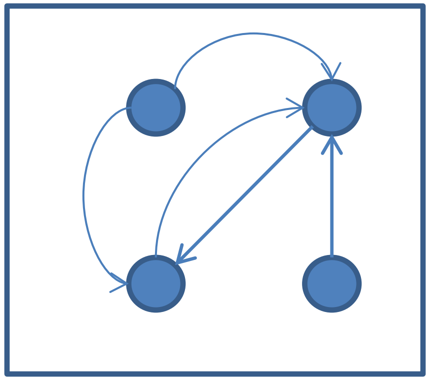


$|E| = 7$
$|V| = 5$

Directed Graph

- A directed graph (digraph) $G=(V,E)$ consists of a nonempty set V of vertices and a possibly empty set E of edges (or archs). Each edge connects two vertices from V:

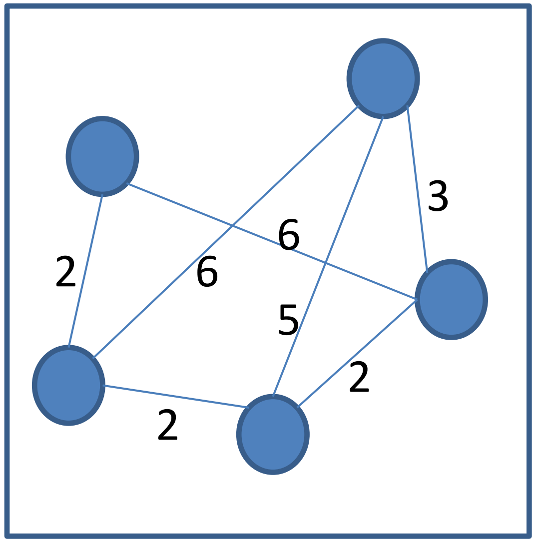
$$\{v_i,\ v_j\}\ != \{v_j,\ v_i\}$$



$|E| = 5$
$|V| = 4$

Weighted Graph

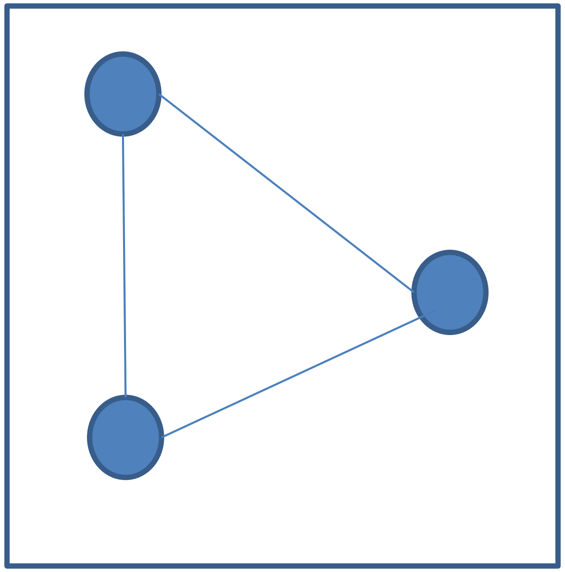
- A graph where edges have assigned numbers.
- The numbers could be distances values, lengths, costs, … etc.

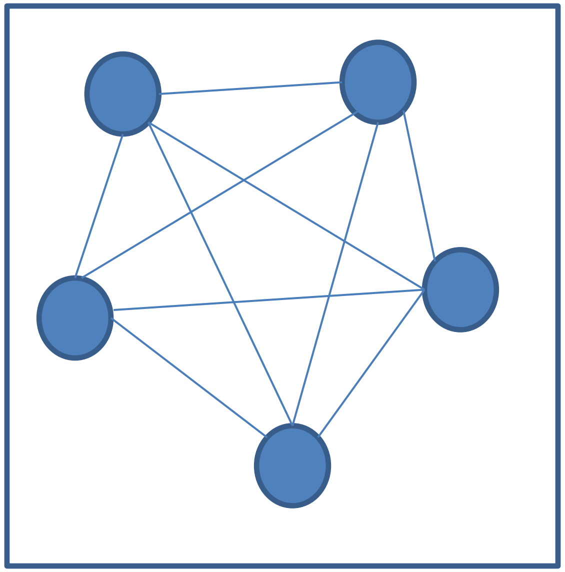


$|E| = 7$
$|V| = 5$

Complete Graph

- For each pair of vertices, there is exactly one edge connecting them.





|E| = 3
|V| = 3

|E| = 10
|V| = 5

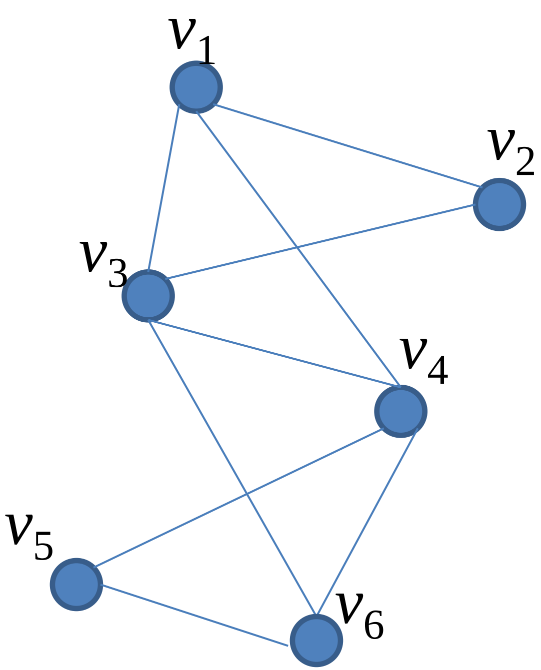
More terms

- **Subgraph** *G'* of *G=(V,E)* is a graph *(V',E')* such that $V' \subseteq V$ and $E' \subseteq E$.
- **Adjacent vertices**: two vertices v_i and v_j are adjacent if edge $(v_i, v_j) \subseteq E$.
  - Such edge is called incident with vertices v_i and v_j.
- The **degree** of a vertex *v*, deg(*v*), is the number of edges incident with *v*.
  - If deg(*v*) = 0, *v* is called isolated vertex.

More terms

- Path of $v_1, v_2, \ldots, v_n$ is a sequence of edges $(v_1, v_2), (v_2,v_3), \ldots, (v_{n-1}, v_n)$, denoted as path $v_1,v_2,v_3,\ldots,v_{n-1},v_n$.
- Circuit: There exists a path $v_1,v_2,\ldots,v_n$ where $v_1==v_n$ and no edge is repeated.
- Cycle: if all vertices in a circuit are different.

Path, Circuit, and cycle



Paths:

- (v1,v4), (v4,v6), (v6,v5)
- (v3, v6), (v6, v4), (v4, v1), (v1, v2)
- ...

Circuit:

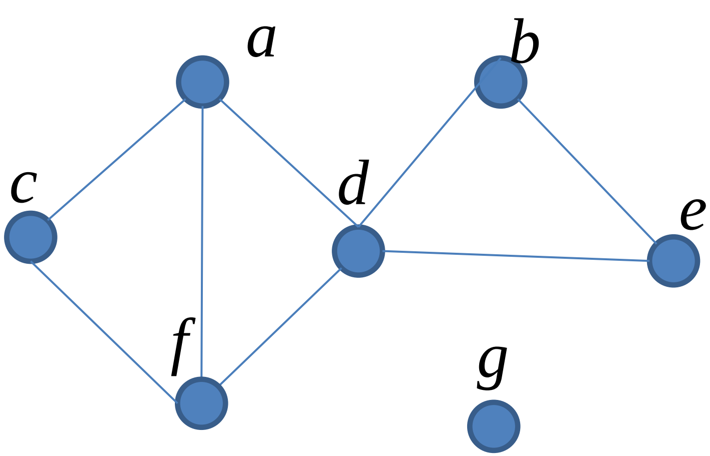
(v1, v3), (v3, v6), (v6, v4), (v4, v3), (v3, v2), (v2, v1)

Cycle:

(v1, v2), (v2, v3), (v3, v6), (v6, v5), (v5, v4), (v4, v1)

Graph representations

- Adjacency list
  - Vertices are stored as records or objects, and every vertex stores a list of adjacent vertices.
  - This data structure allows the storage of additional data on the vertices.



$a \rightarrow c \rightarrow d \rightarrow f$

$b \rightarrow d \rightarrow e$

$c \rightarrow a \rightarrow f$

$d \rightarrow a \rightarrow b \rightarrow e \rightarrow f$

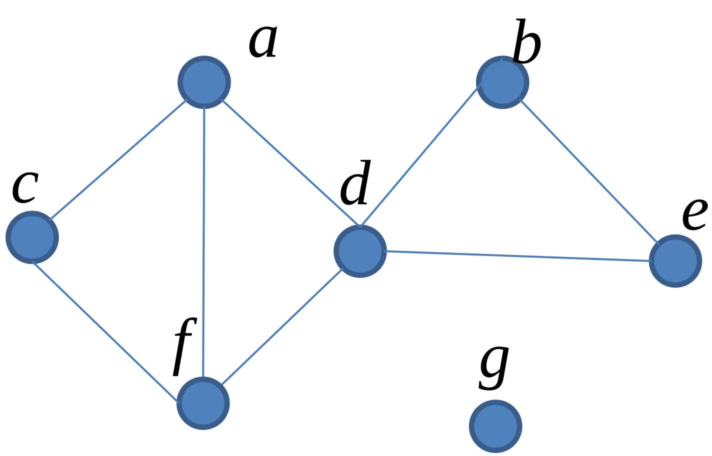
$e \rightarrow b \rightarrow d$

$f \rightarrow a \rightarrow c \rightarrow d$

g

Graph representations

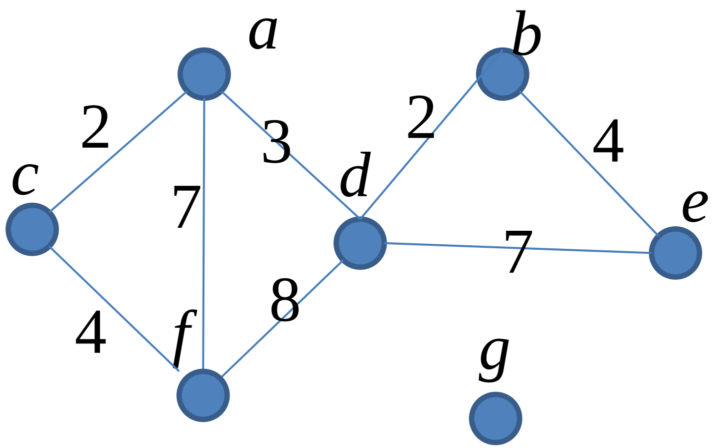
- Adjacency matrix
  - A two-dimension matrix
    - rows represent source vertices
    - columns represent destination vertices
  - Data on edges and vertices are stored externally



	a	b	c	d	e	f	g
a	0	0	1	1	0	1	0
b	0	0	0	1	1	0	0
c	1	0	0	0	0	1	0
d	1	1	0	0	1	1	0
e	0	1	0	1	0	0	0
f	1	0	1	1	0	0	0
g	0	0	0	0	0	0	0

Graph representations

- Adjacency matrix (cont.)
  - Symmetric for simple graphs
  - Non-symmetric for digraphs
  - In case of weighted graphs, values in matrix indicate weights of edges



	a	b	c	d	e	f	g
a	0	0	2	3	0	7	0
b	0	0	0	2	4	0	0
c	2	0	0	0	0	4	0
d	3	2	0	0	7	8	0
e	0	4	0	7	0	0	0
f	7	0	4	8	0	0	0
g	0	0	0	0	0	0	0

Typical operations in graph

Operation	Adjacency matrix	Adjacency list
Add a node	$O(\|V\|^2)$	$O(1)$
Remove a node	$O(\|V\|^2)$	$O(\|E\|)$
Add an edge	$O(1)$	$O(1)$
Remove an edge	$O(1)$	$O(\|V\|)$
Get neighbors of a node	$O(\|V\|)$	$O(\|V\|)$
Test an edge	$O(1)$	$O(\|V\|)$
Get/set edge	$O(1)$	$O(\|V\|)$
Storage	$O(\|V\|^2)$	$O(\|V\|+\|E\|)$