

Data Structure

Graph

Graphs

- A data structure that consists of a set of nodes (*vertices*) and a set of edges that relate the nodes to each other
- The set of edges describes relationships among the vertices .
- A graph G is defined as follows:

$$G = (V, E)$$

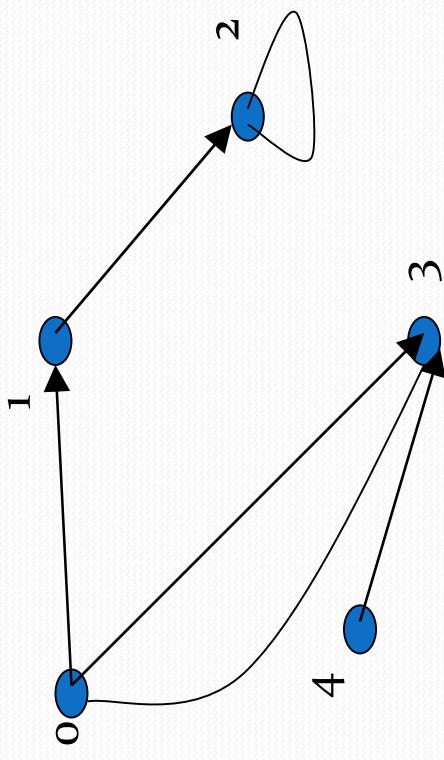
$V(G)$: a finite, nonempty set of vertices

$E(G)$: a set of edges (pairs of vertices)

Examples of Graphs

- $V = \{0, 1, 2, 3, 4\}$
- $E = \{(0,1), (1,2), (0,3), (3,0), (2,2), (4,3)\}$

When (x,y) is an edge,
we say that x is *adjacent to* y , and y
is *adjacent from* x .



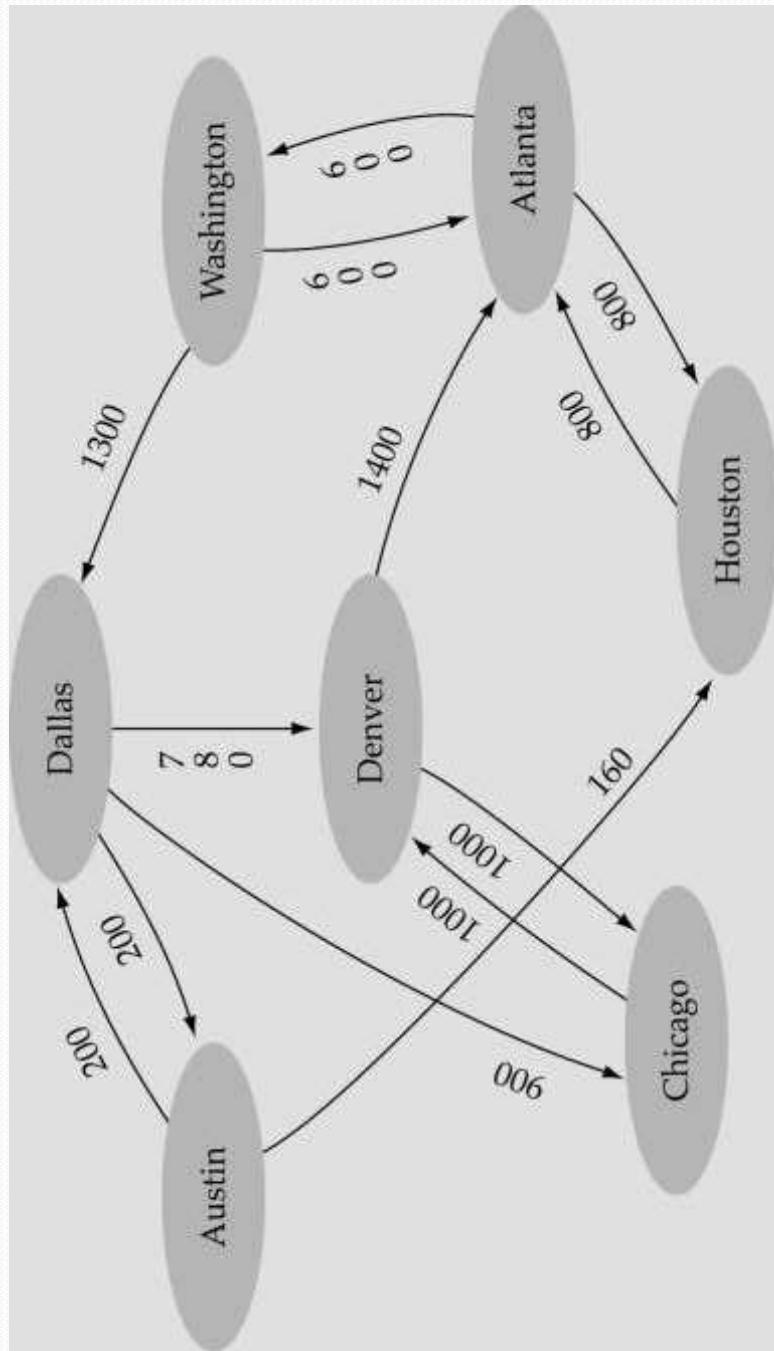
Directed vs. Undirected Graphs

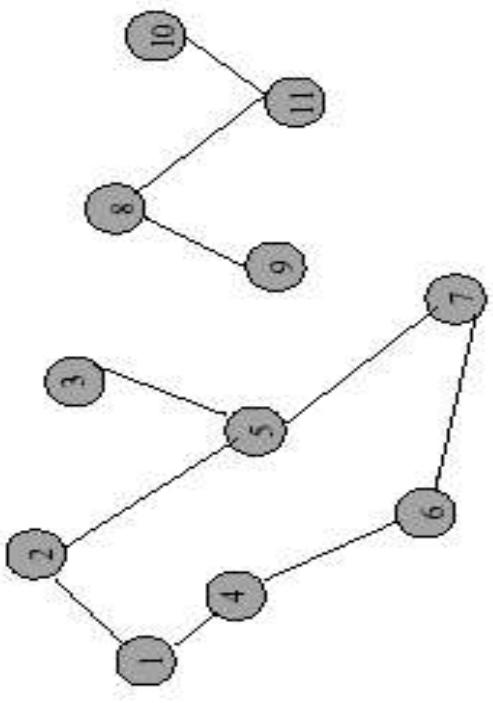
- Undirected edge has no orientation (no arrow head)
- Directed edge has an orientation (has an arrow head)
- Undirected graph – all edges are undirected
- Directed graph – all edges are directed



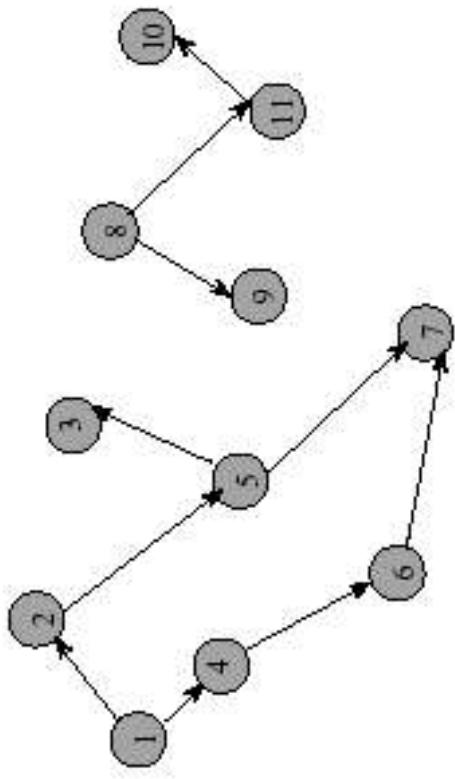
Weighted graph:

-a graph in which each edge carries a value





Undirected graph



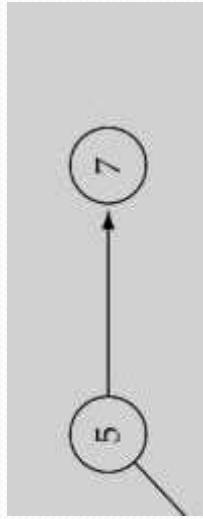
Directed graph

Directed Graph

- Directed edge (i, j) is **incident to vertex j** and **incident from vertex i**
- Vertex i is **adjacent to vertex j** , and vertex j is **adjacent from vertex i**

Graph terminology

- **Adjacent nodes:** two nodes are adjacent if they are connected by an edge

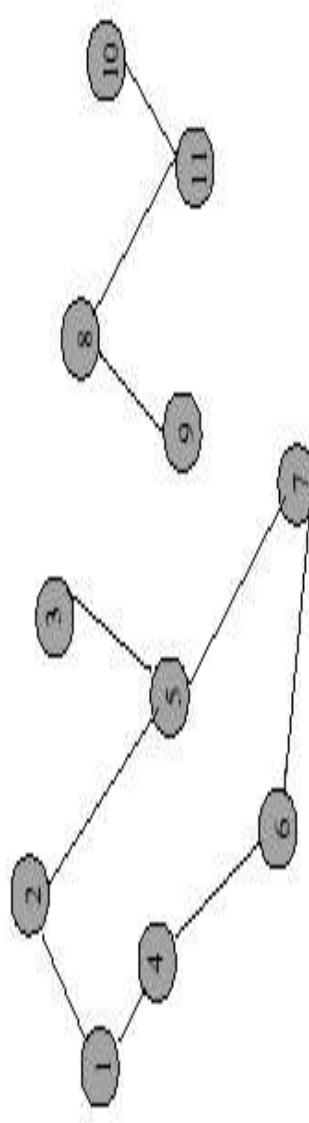


5 is adjacent to 7
7 is adjacent from

- **Path:** a sequence of vertices that connect two nodes in a graph
 - A **simple path** is a path in which all vertices, except possibly in the first and last, are different.
 - A **complete graph**: a graph in which every vertex is directly connected to every other vertex

Continued...

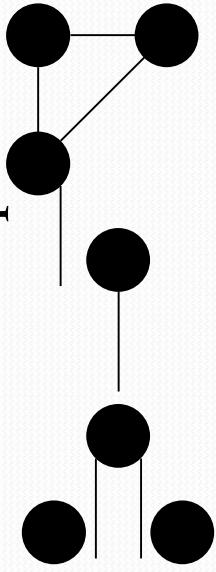
- A **cycle** is a simple path with the same start and end vertex.
- The **degree** of vertex i is the **no. of edges incident** on vertex i .



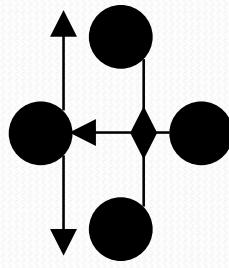
e.g., $\text{degree}(2) = 3$, $\text{degree}(5) = 3$, $\text{degree}(3) = 1$

Continued....

Undirected graphs are *connected* if there is a path between any two vertices



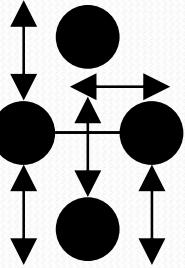
Directed graphs are *strongly connected* if there is a path from any one vertex to any other



Directed graphs are *weakly connected* if there is a path between any two vertices, ignoring direction



A *complete* graph has an edge between every pair of vertices



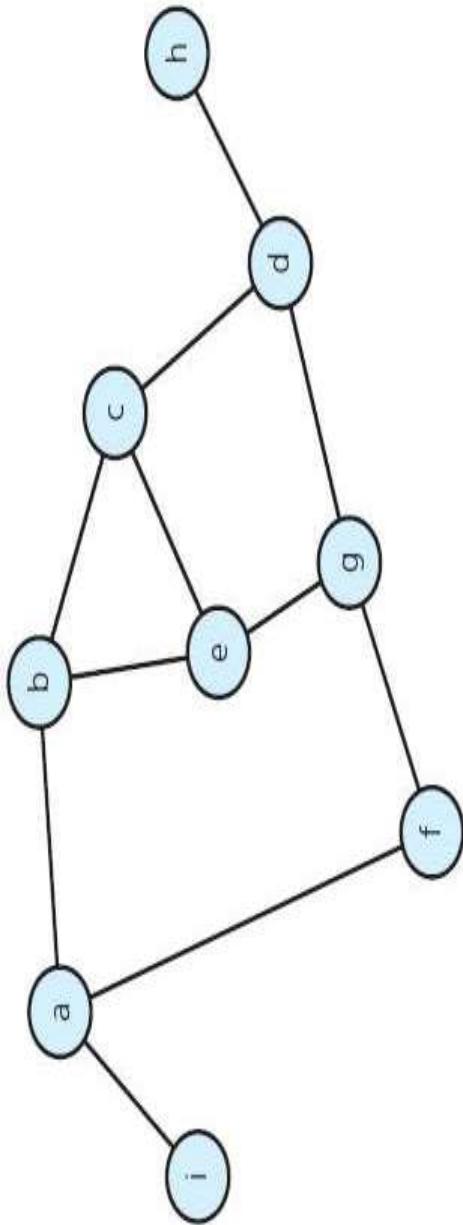
Continued...

- **Loops:** edges that connect a vertex to itself
- **Paths:** sequences of vertices p_0, p_1, \dots, p_m such that each adjacent pair of vertices are connected by an edge
 - A **simple path** is a path in which all vertices, except possibly in the first and last, are different.
- **Multiple Edges:** two nodes may be connected by >1 edge
 - **Simple Graphs:** have no loops and no multiple edges

Graph Properties

Number of Edges – Undirected Graph

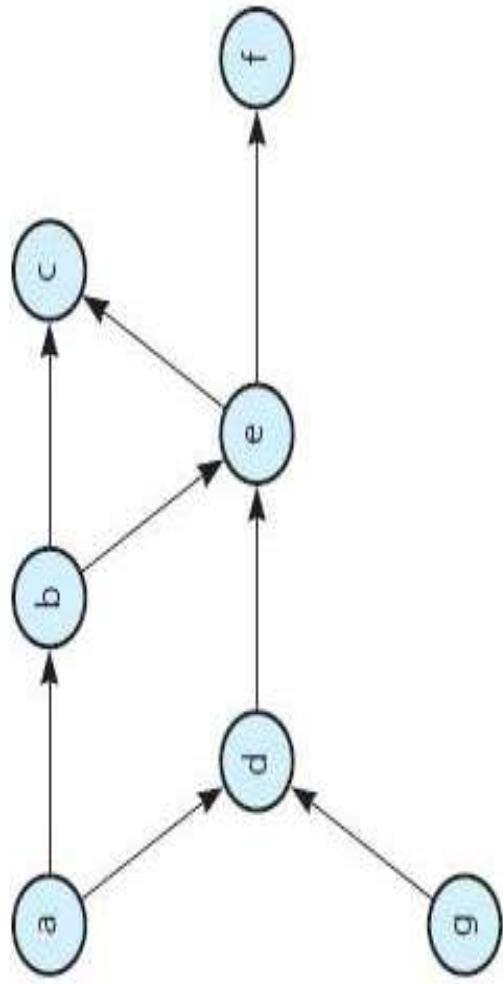
- The no. of possible pairs in an n vertex graph is $n^*(n-1)$
- Since edge (u,v) is the same as edge (v,u) , the number of edges in an undirected graph is $n^*(n-1)/2$.



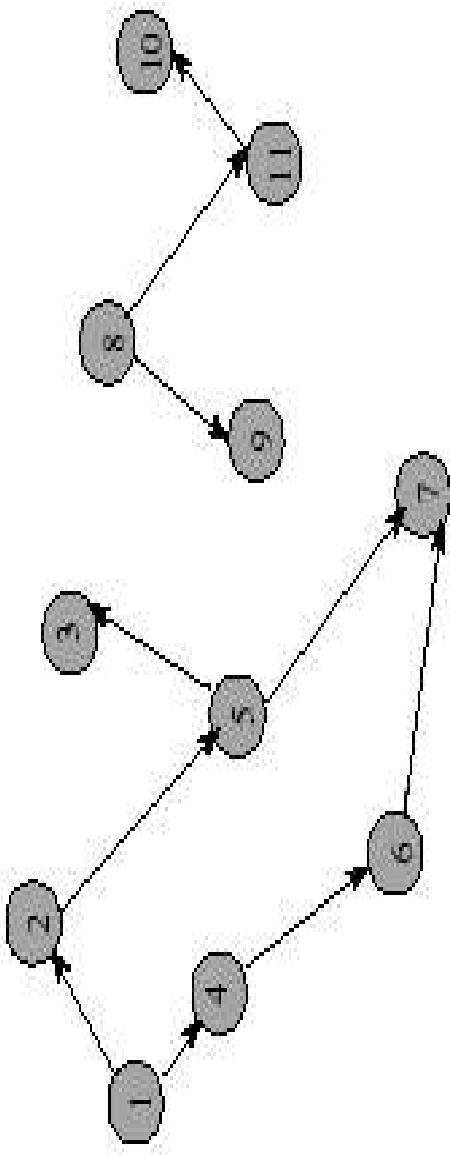
Graph

Number of Edges - Directed Graph

- The no. of possible pairs in an n vertex graph is $n^*(n-1)$
- Since edge (u,v) is not the same as edge (v,u) , the number of edges in a directed graph is $n^*(n-1)$
- Thus, the number of edges in a directed graph is $\leq n^*(n-1)$



Graph



- **In-degree** of vertex i is the **number of edges incident to i** (i.e., the number of incoming edges).
e.g., $\text{indegree}(2) = 1$, $\text{indegree}(8) = 0$
- **Out-degree** of vertex i is the **number of edges incident from i** (i.e., the number of outgoing edges).
e.g., $\text{outdegree}(2) = 1$, $\text{outdegree}(8) = 2$

Graph Representation

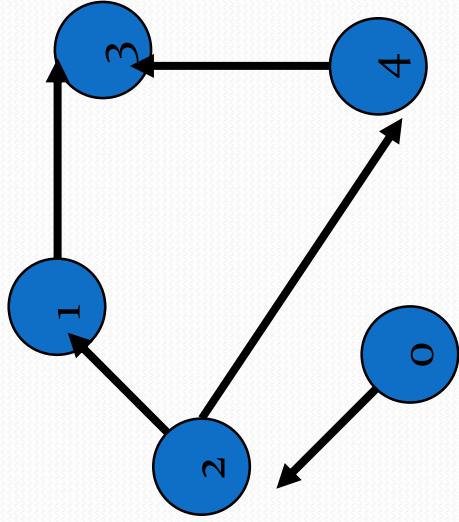
- For graphs to be computationally useful, they have to be conveniently represented in programs
- There are two computer representations of graphs:
 - Adjacency matrix representation
 - Adjacency lists representation

- *Adjacency Matrix*

- A square grid of boolean values
- If the graph contains N vertices, then the grid contains N rows and N columns
 - For two vertices numbered I and J , the element at row I and column J is true if there is an edge from I to J , otherwise false

Adjacency Matrix

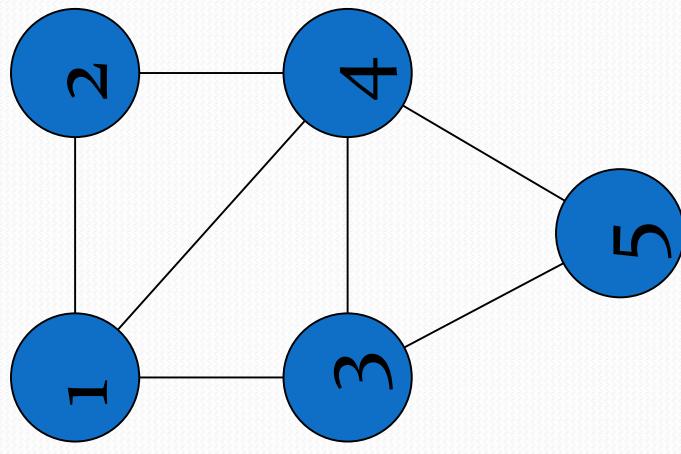
	0	1	2	3	4
0	false	true	false	false	false
1	false	false	true	true	false
2	true	false	false	true	false
3	false	false	false	false	true
4	false	false	true	false	false



Graph

Adjacency Matrix

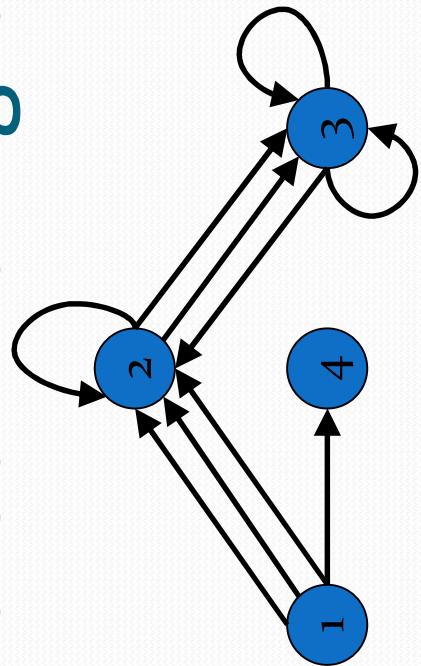
	1	2	3	4	5
1	0	1	1	1	0
2	1	0	0	1	0
3	1	0	0	1	1
4	1	1	1	0	1
5	0	0	1	1	0



Graph

Adjacency Matrix

-Directed Multigraphs



A:

$$\begin{pmatrix} 0 & 3 & 0 & 1 \\ 0 & 1 & 2 & 0 \\ 0 & 1 & 2 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

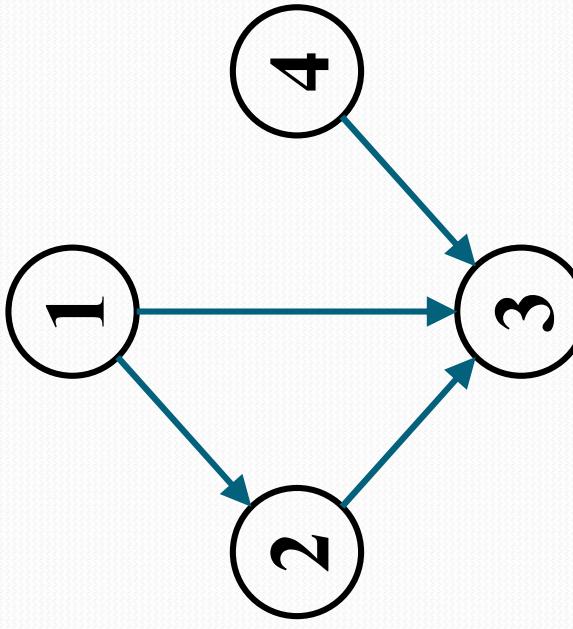
Adjacency Lists Representation

- A graph of n nodes is represented by a one-dimensional array L of linked lists, where
 - $L[i]$ is the linked list containing all the nodes adjacent from node i .
 - The nodes in the list $L[i]$ are in no particular order

Graphs: Adjacency List

- Adjacency list: for each vertex $v \in V$, store a list of vertices adjacent to v

- Example:



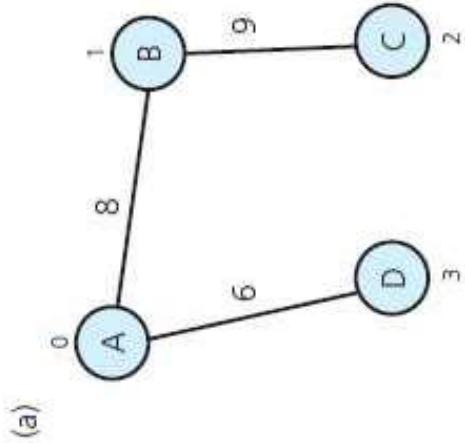
- $\text{Adj}[1] = \{2,3\}$
- $\text{Adj}[2] = \{3\}$
- $\text{Adj}[3] = \{\}$
- $\text{Adj}[4] = \{3\}$

- Variation: can also keep a list of edges coming *into* vertex

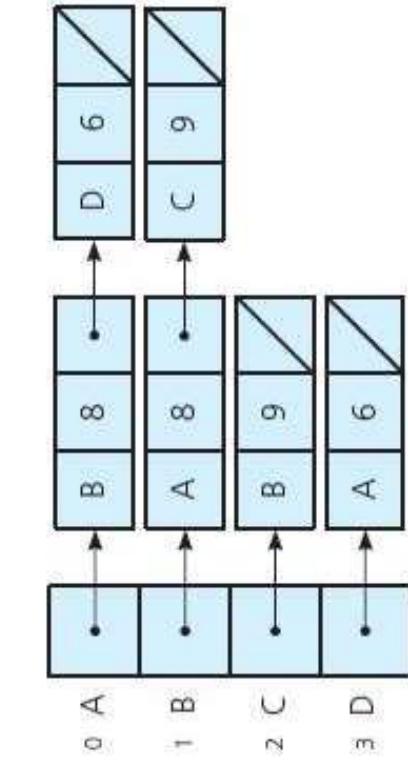
Graphs: Adjacency List

- How much storage is required?
 - The *degree* of a vertex $v = \#$ incident edges
 - Directed graphs have in-degree, out-degree
 - For directed graphs, # of items in adjacency lists is
$$\sum \text{out-degree}(v) = |E|$$
For undirected graphs, # items in adjacency lists is
$$\sum \text{degree}(v) = 2 |E|$$
 - So: Adjacency lists take $O(V+E)$ storage

Implementing Graphs



(b)



- (a) A weighted undirected graph and
- (b) its adjacency list

Thank you!!!