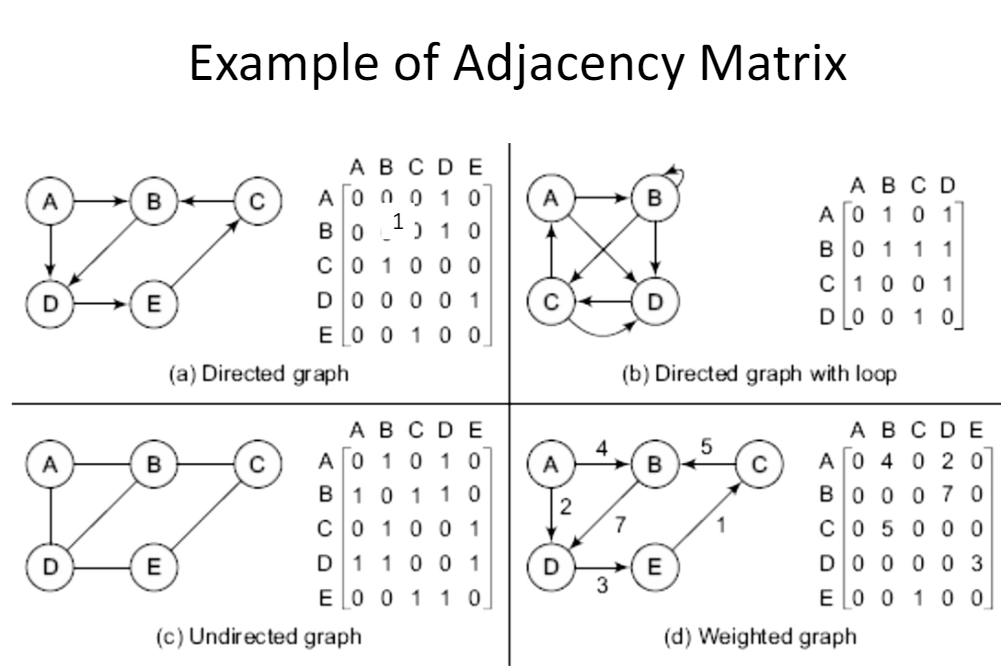
**Adjacency Matrix**

From adjacency matrix A, aij = 1 means that there exists a path of length 1 from vi to vj.

Consider the matric multiplication of A with itself. Any entry a2ij = 1 if aik = a kj = 1. That is, if there is an edge (vi,vk) and (vk,vj). This implies that there is a path from vi, to vj of length 2.

Similarly, every entry in the ith row and jth column of A cubed gives the number of paths of length 3 from node vi to vj.



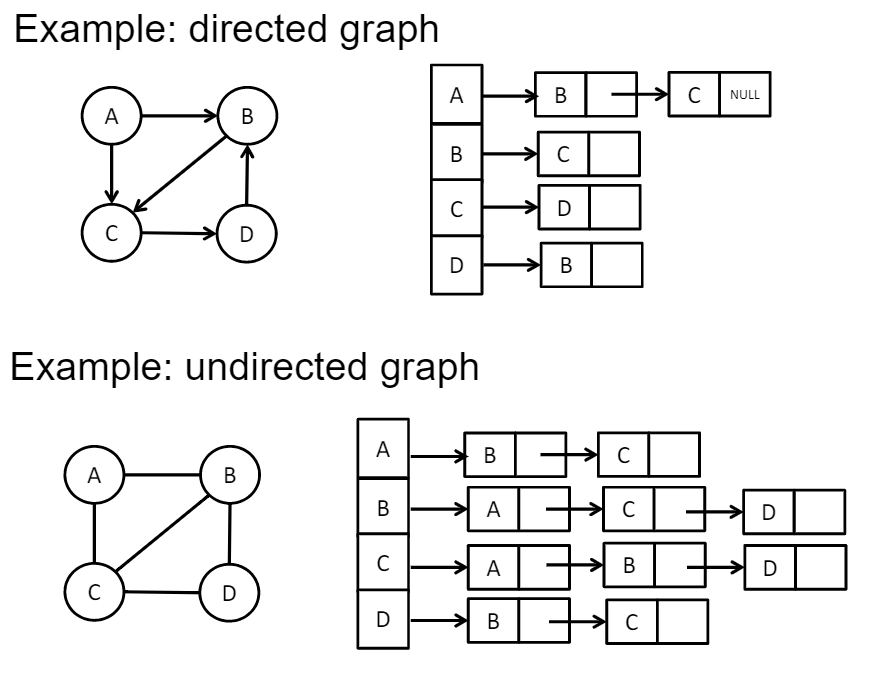
**Adjacency list**

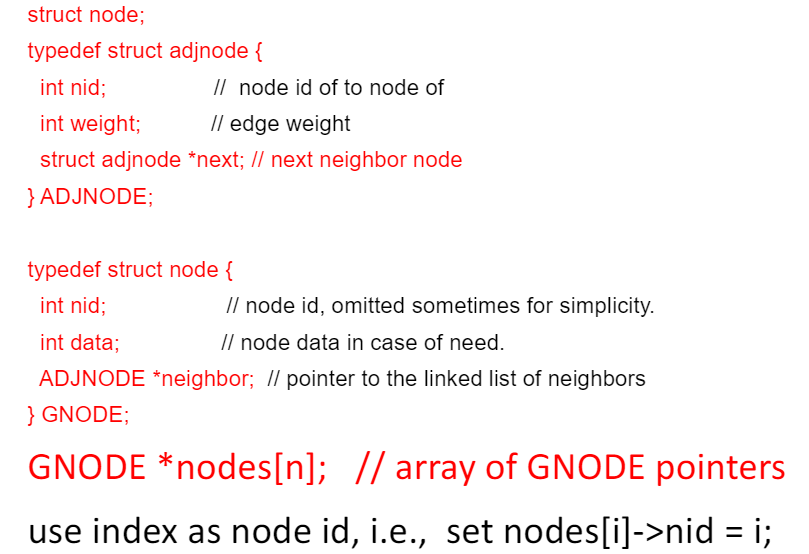
The adjacency list represents a graph by the list of nodes, and a list of its neighbor nodes for each node.

The list of nodes can be implemented by an array, or a linked list of node structure:

{node id, name, data, pointer to linked list of adjnode structure}

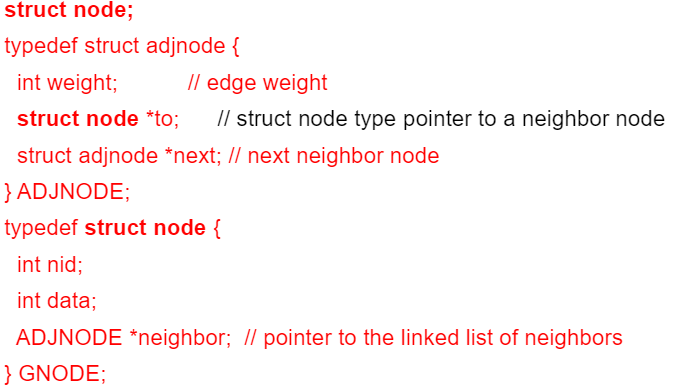
Adjnode structure: {node id, weight, pointer to next adjnode}

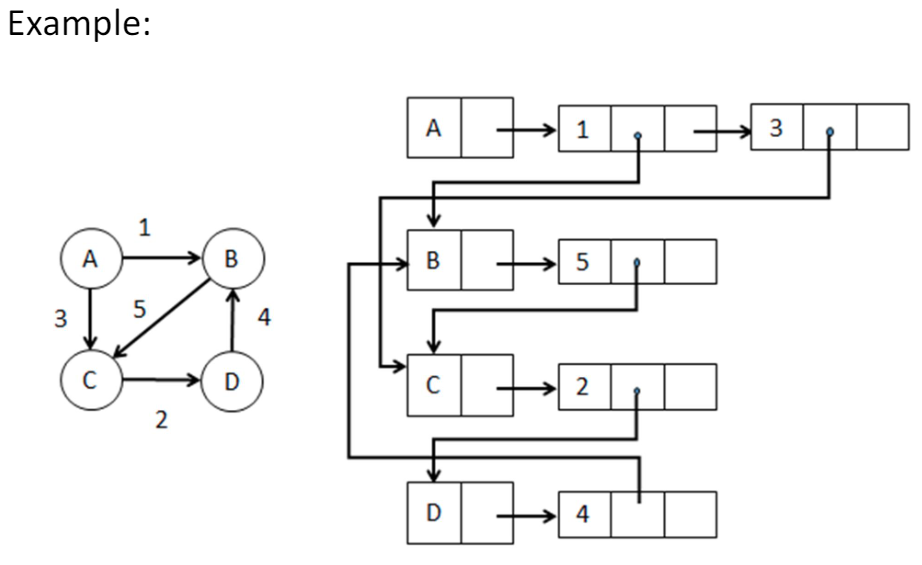




**Linked node representation**

The linked node representation is similar to the adjacency list. However, it uses pointer to represent neighbor nodes instead of neighbor node id (index). The node structure linked node representation is defined as follows:





**Edge List**

The edge list representation is simply a list of edges. An edge can be represented by an edge structure

{ from node id, to node id, weight}

The list of edges can be implemented by array, linked list, or tree data structures



**Comparisons**

* Adjacency matrix
  + Support directed, undirected, weighted graphs
  + Efficient for accessing edges and most graph algorithms
  + Good for dense and small graphs
  + Less space efficiency for large sparse graphs
* Adjacency list
  + Support directed, undirected and weighted graphs
  + Efficient for accessing nodes and neighbors and many graph algorithms
  + Space efficient for large sparse graphs
* Edge list
  + Support directed, undirected, weighted graphs
  + Efficient for edge-oriented algorithms, e.g. Find the weight of a graph
  + Less efficient for most graph algorithms, e.g. Finding neighbors

**Others**

* Represent graph by sets: use a set to represent vertex set, and another set for edge set
* Vertex set and edge set can be stored distributed, i.e. vertex and edge sets are portioned and stored in different computers
* Distributed algorithms or parallel algorithms are commonly used to work on distributed graph representations of super large size.
* Dynamic representation: vertices and edges can be dynamically added when necessary