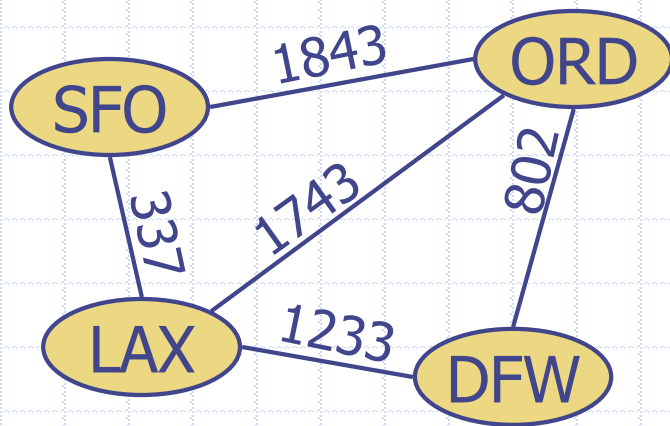
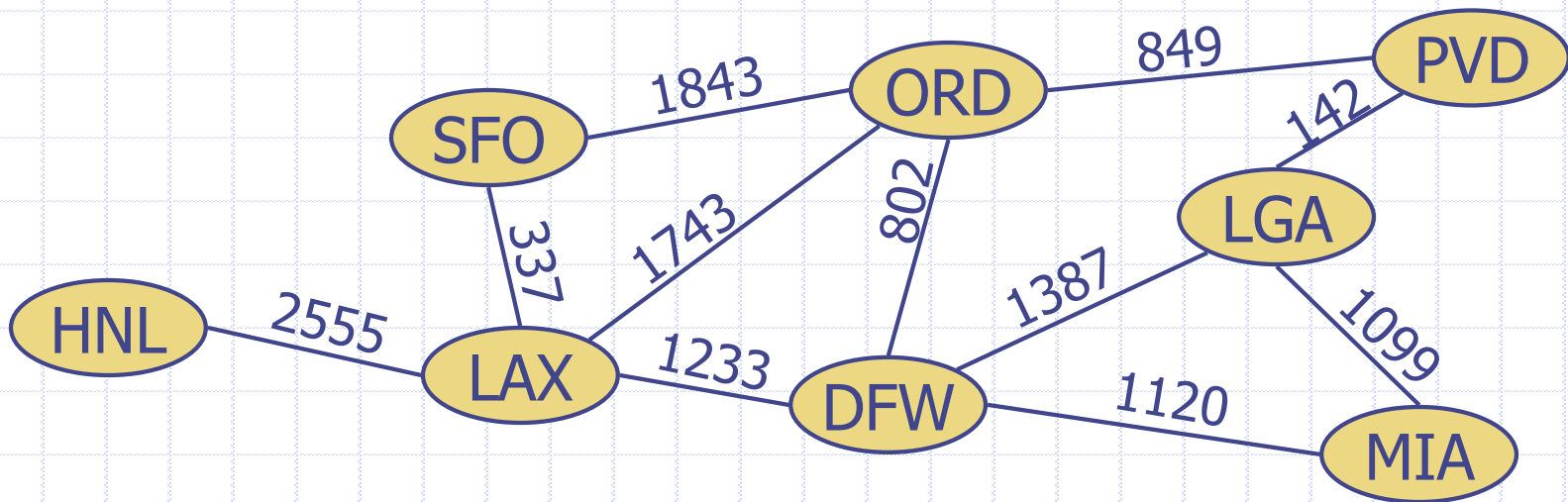


Graphs



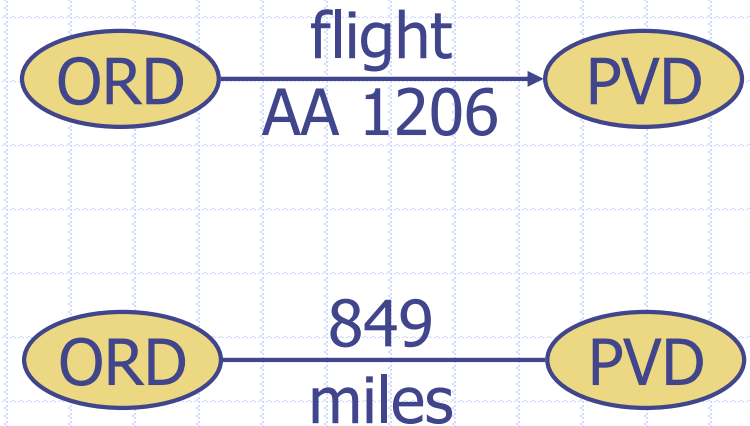
Graphs

- A graph is a pair (V, E) , where
 - V is a set of nodes, called **vertices**
 - E is a collection of pairs of vertices, called **edges**
- Example:
 - A vertex represents an airport and stores the three-letter airport code
 - An edge represents a flight route between two airports and stores the mileage of the route



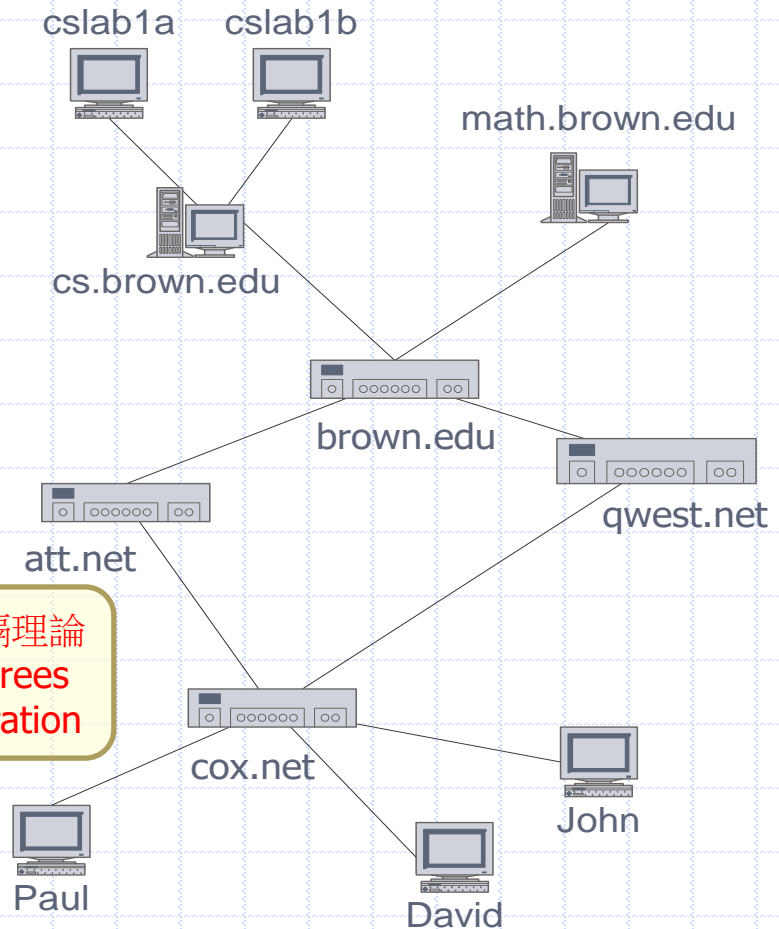
Edge Types

- Directed edge
 - ordered pair of vertices (u,v)
 - first vertex u is the origin or source
 - second vertex v is the destination
 - e.g., a flight number
- Undirected edge
 - unordered pair of vertices (u,v)
 - e.g., a flight mileage
- Directed graph (digraph)
 - all the edges are directed
- Undirected graph
 - all the edges are undirected



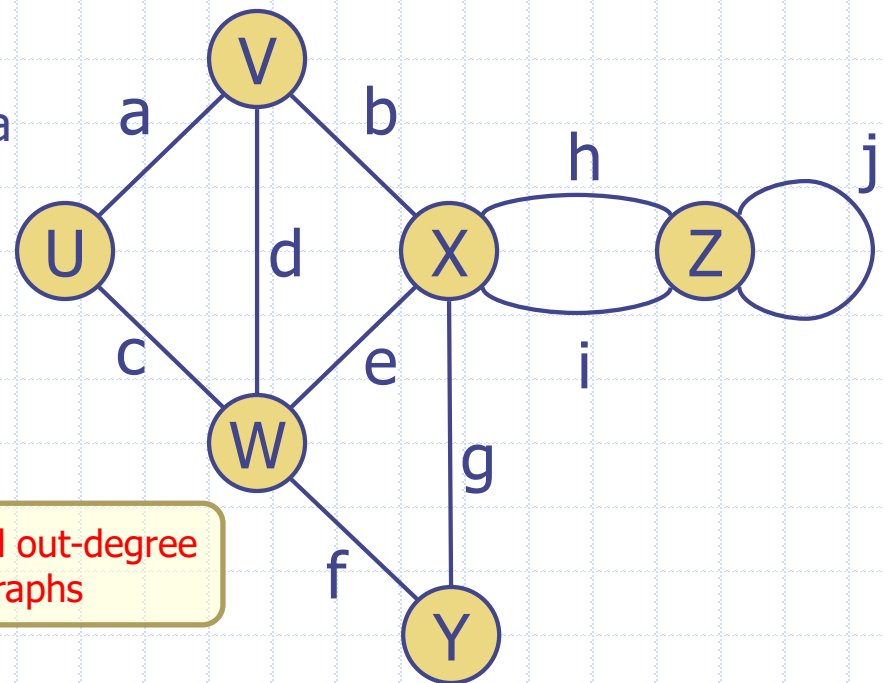
Applications

- ❑ Electronic circuits
 - Printed circuit board
 - Integrated circuit
- ❑ Transportation networks
 - Highway network
 - Flight network
- ❑ Computer networks
- ❑ Social networks
- ❑ Databases
 - Entity-relationship diagram



Terminology

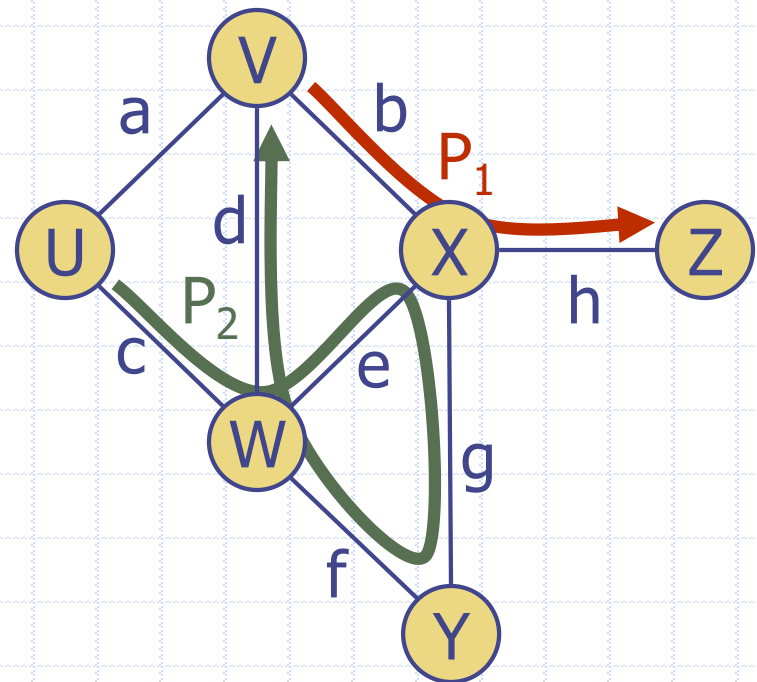
- End vertices (or endpoints) of an edge
 - U and V are the endpoints of a
- Edges incident on a vertex
 - a, d, and b are incident on V
- Adjacent vertices
 - U and V are adjacent
- Degree of a vertex
 - X has degree 5
- Parallel edges
 - h and i are parallel edges
- Self-loop
 - j is a self-loop



In-degree and out-degree
for digraphs

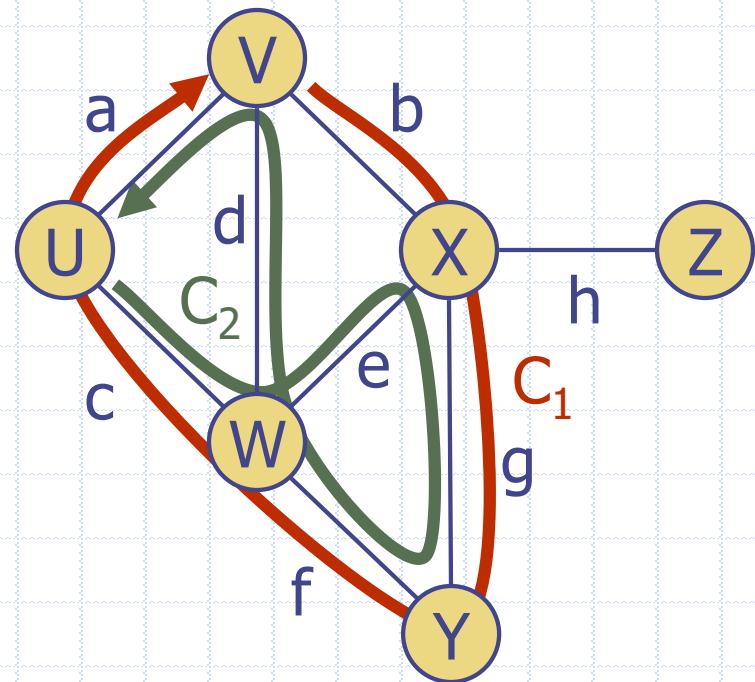
Terminology (cont.)

- Path
 - sequence of alternating vertices and edges
 - begins with a vertex
 - ends with a vertex
 - each edge is preceded and followed by its endpoints
- Simple path
 - path such that all its vertices and edges are distinct
- Examples
 - $P_1 = (V, b, X, h, Z)$ is a simple path
 - $P_2 = (U, c, W, e, X, g, Y, f, W, d, V)$ is a path that is not simple



Terminology (cont.)

- Cycle
 - circular sequence of alternating vertices and edges
 - each edge is preceded and followed by its endpoints
- Simple cycle
 - cycle such that all its vertices and edges are distinct
- Examples
 - $C_1 = (V, b, X, g, Y, f, W, c, U, a, \downarrow)$ is a simple cycle
 - $C_2 = (U, c, W, e, X, g, Y, f, W, d, V, a, \downarrow)$ is a cycle that is not simple



Properties

Property 1

$$\sum_v \deg(v) = 2m$$

Proof: each edge is counted twice

Property 2

In an undirected graph with no self-loops and no multiple edges

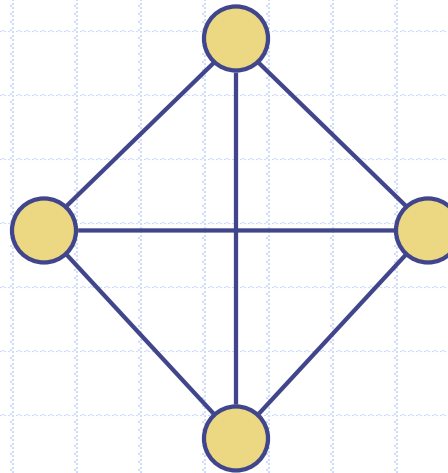
$$m \leq n(n-1)/2$$

Proof: each vertex has degree at most $(n-1)$

What is the bound for a directed graph?

Notation

n	number of vertices
m	number of edges
$\deg(v)$	degree of vertex v



Example

- $n = 4$
- $m = 6$
- $\deg(v) = 3$

Main Methods of the Graph ADT

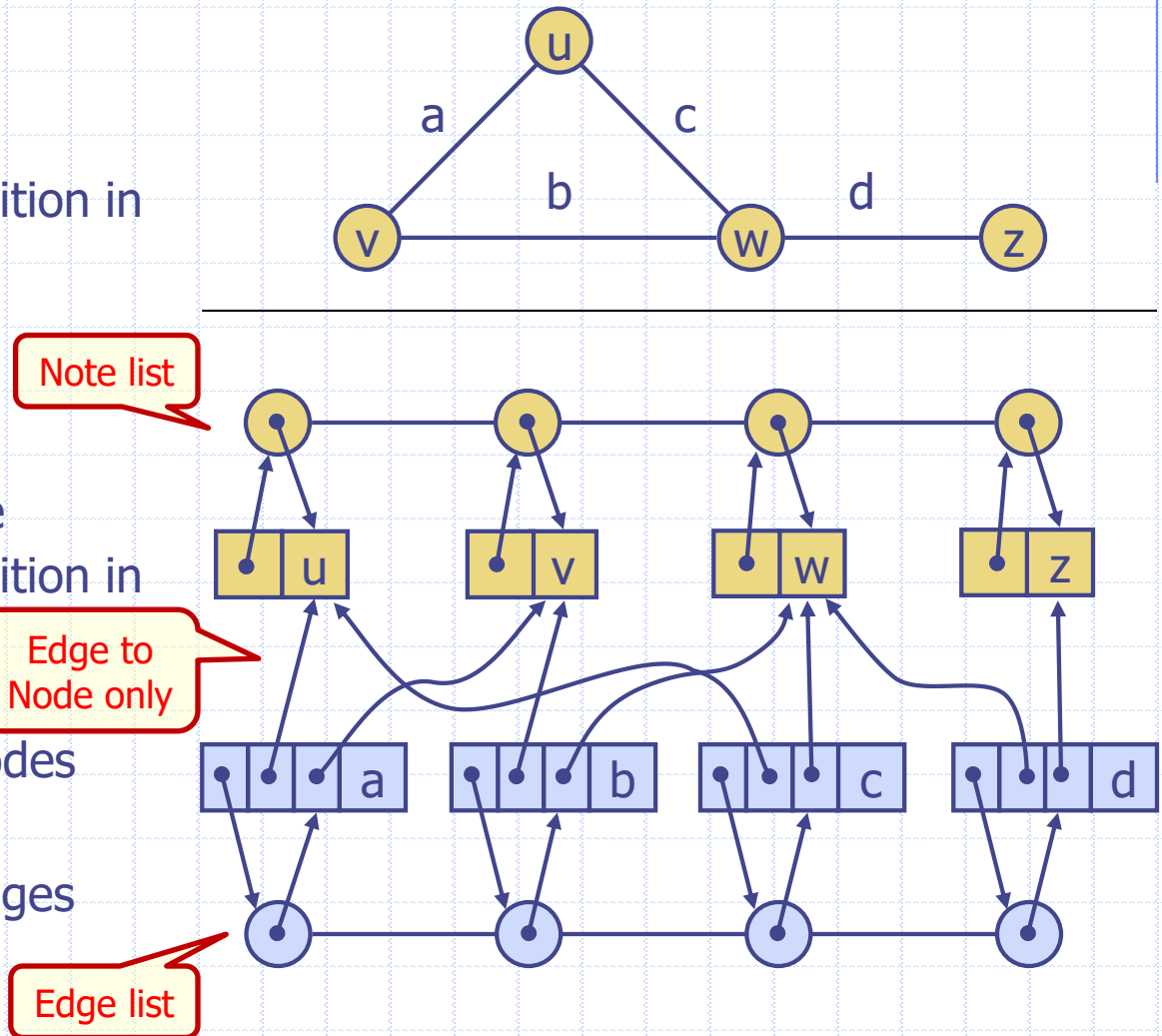
- Vertices and edges
 - are positions
 - store elements
- Accessor methods
 - `e.endVertices()`: a list of the two endvertices of `e`
 - `e.opposite(v)`: the vertex opposite of `v` on `e`
 - `u.isAdjacentTo(v)`: true iff `u` and `v` are adjacent
 - `*v`: reference to element associated with vertex `v`
 - `*e`: reference to element associated with edge `e`
- Update methods
 - `insertVertex(o)`: insert a vertex storing element `o`
 - `insertEdge(v, w, o)`: insert an edge `(v,w)` storing element `o`
 - `eraseVertex(v)`: remove vertex `v` (and its incident edges)
 - `eraseEdge(e)`: remove edge `e`
- Iterable collection methods
 - `incidentEdges(v)`: list of edges incident to `v`
 - `vertices()`: list of all vertices in the graph
 - `edges()`: list of all edges in the graph

Data Structures for Graphs

- Edge-list structure
- Adjacency-list structure
- Adjacency-matrix structure

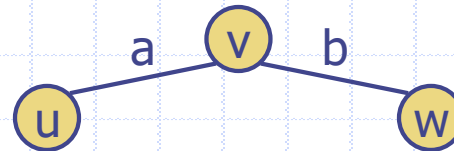
Edge-List Structure

- Node
 - element
 - reference to position in node list
- Edge
 - element
 - origin node
 - destination node
 - reference to position in edge list
- Node list
 - list of links to nodes
- Edge list
 - list of links to edges



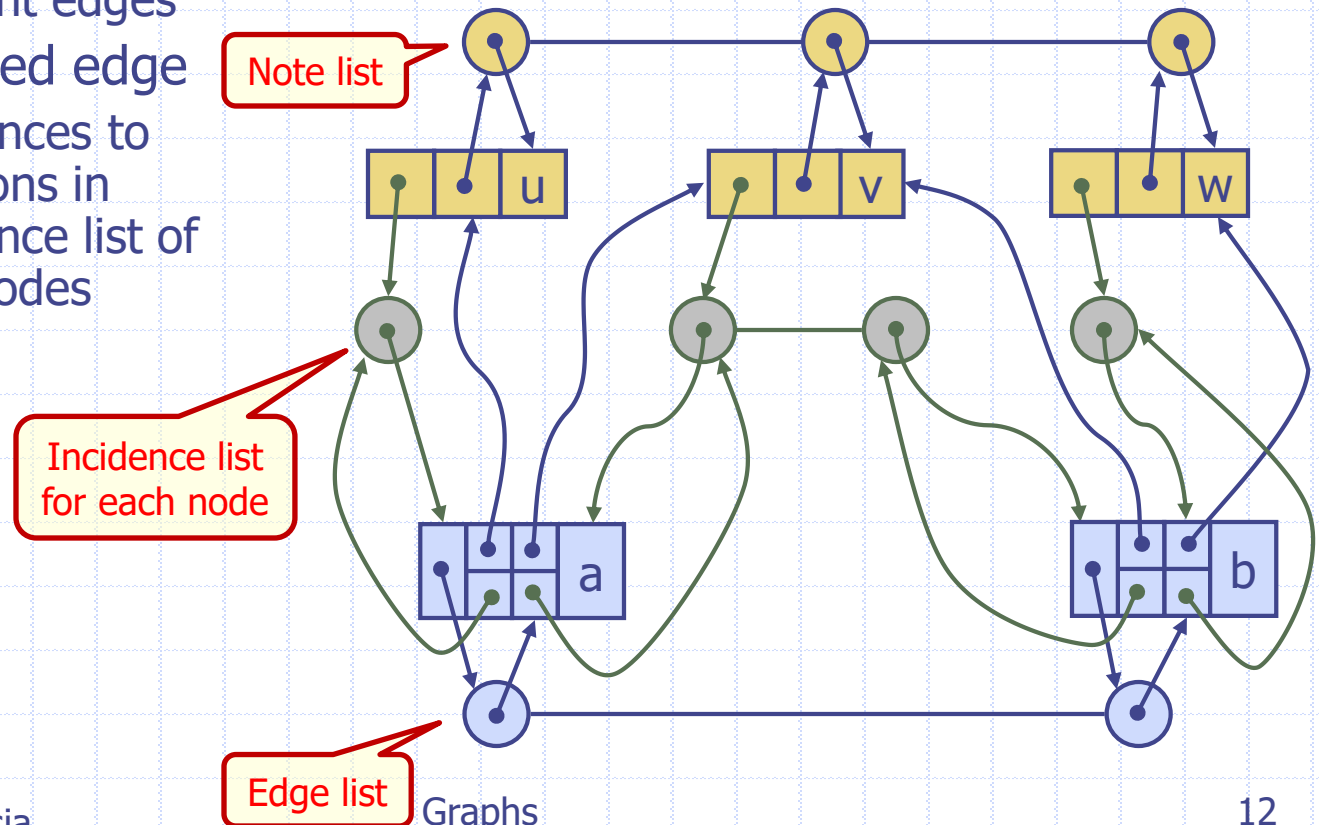
Adjacency-List Structure

- Incidence list for each node
 - list of links to incident edges
- Augmented edge
 - references to positions in incidence list of end nodes



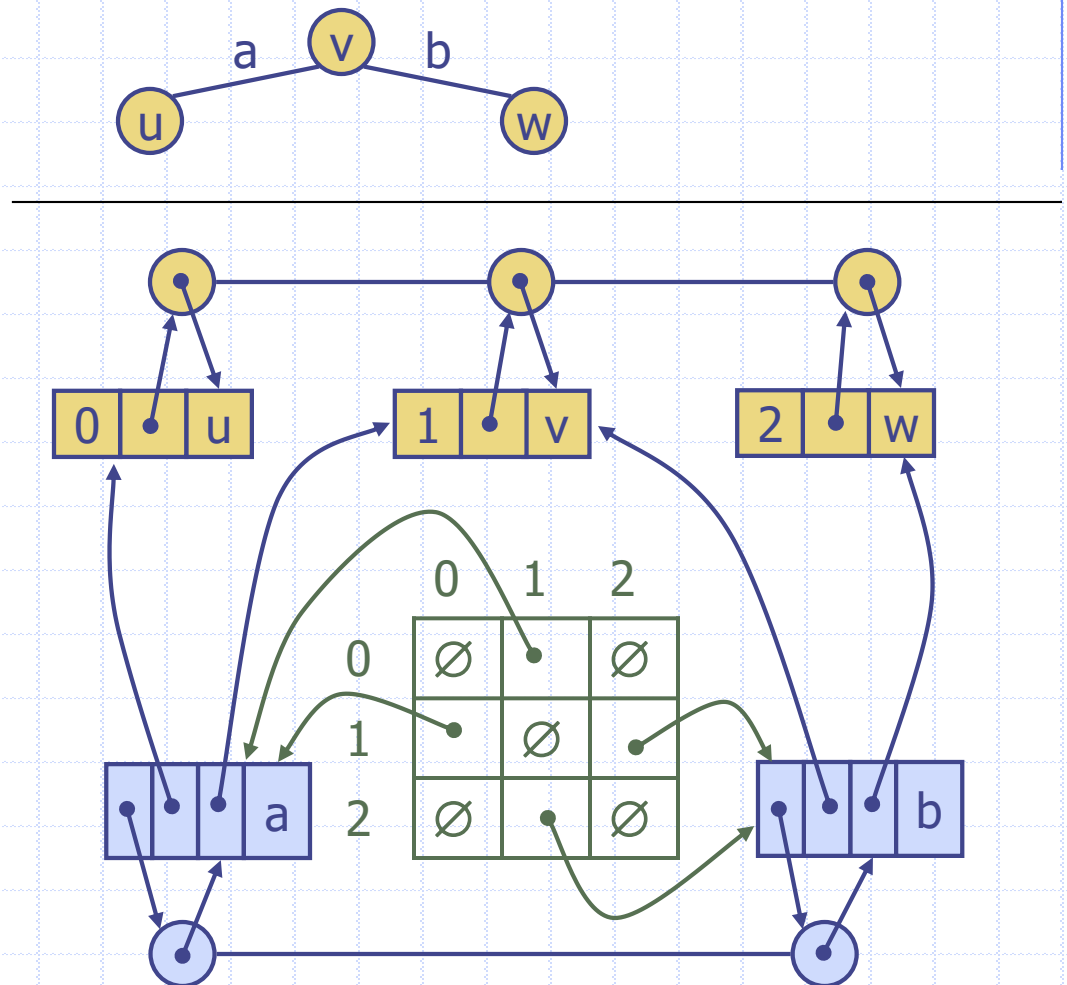
Adjacency list

```
u → v
v → u → w
w → v
```



Adjacency-Matrix Structure

- Augmented nodes
 - Integer index associated with node
- Adjacency matrix
 - Reference to edge for adjacent nodes
- The “old fashioned” matrix just has 0 for no edge and 1 for edge



Performance

We shall stick to this structure!

<ul style="list-style-type: none"> ▪ n vertices, m edges ▪ no parallel edges ▪ no self-loops 	Edge List	Adjacency List	Adjacency Matrix
Space	$n + m$	$n + m$	n^2
$v.\text{incidentEdges}()$	m	$\text{deg}(v)$	n
$u.\text{isAdjacentTo}(v)$	m	$\min(\text{deg}(u), \text{deg}(v))$	1
$\text{insertVertex}(o)$	1	1	n^2
$\text{insertEdge}(v, w, o)$	1	1	1
$\text{eraseVertex}(v)$	m	$\text{deg}(v)$	n^2
$\text{eraseEdge}(e)$	1	1	1

Assume V & E are stored in doubly linked lists

Create a new matrix