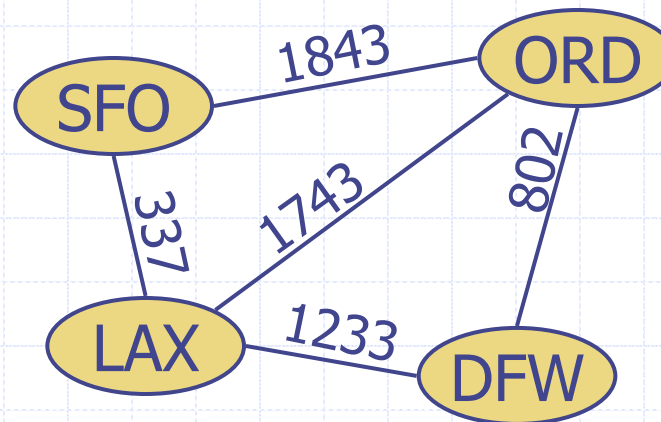


Graphs



Outline and Reading

◆ Graphs (§13.1)

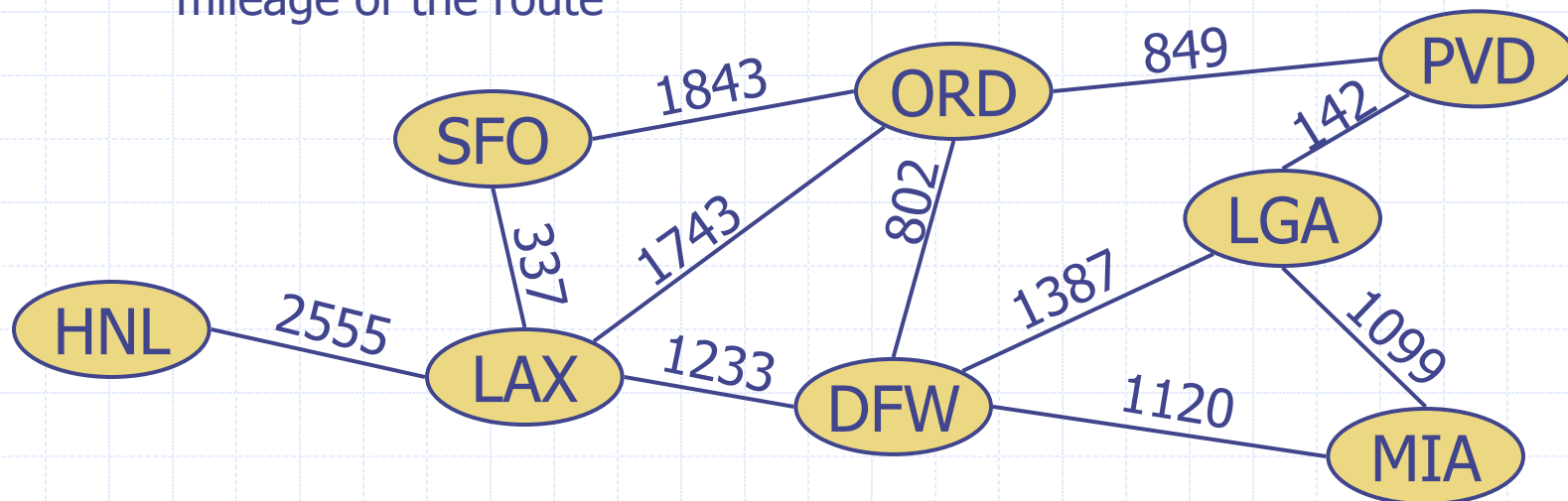
- Definition
- Applications
- Terminology
- Properties
- ADT

◆ Data structures for graphs (§13.2)

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

Graph

- ◆ 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**
 - Vertices and edges are positions and store elements
- ◆ 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



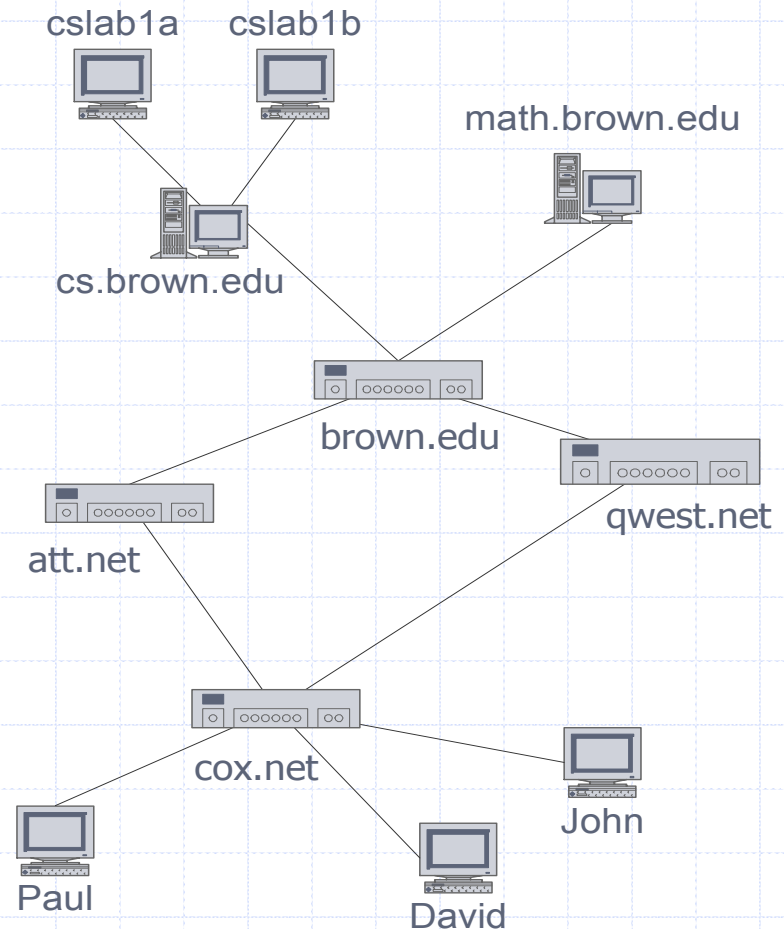
Edge Types

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



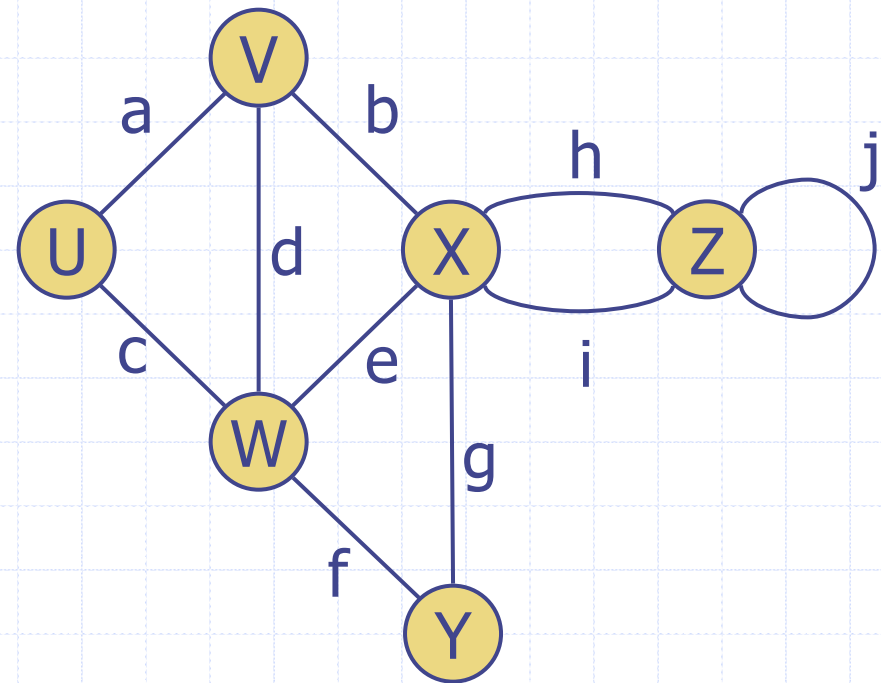
Applications

- ◆ Electronic circuits
 - Printed circuit board
 - Integrated circuit
- ◆ Transportation networks
 - Highway network
 - Flight network
- ◆ Computer networks
 - Local area network
 - Internet
 - Web
- ◆ Programming Languages
 - Inheritance 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 (multiple) edges
 - h and i are parallel edges
- ◆ Self-loop
 - j is a self-loop



Terminology (cont.)

◆ Path

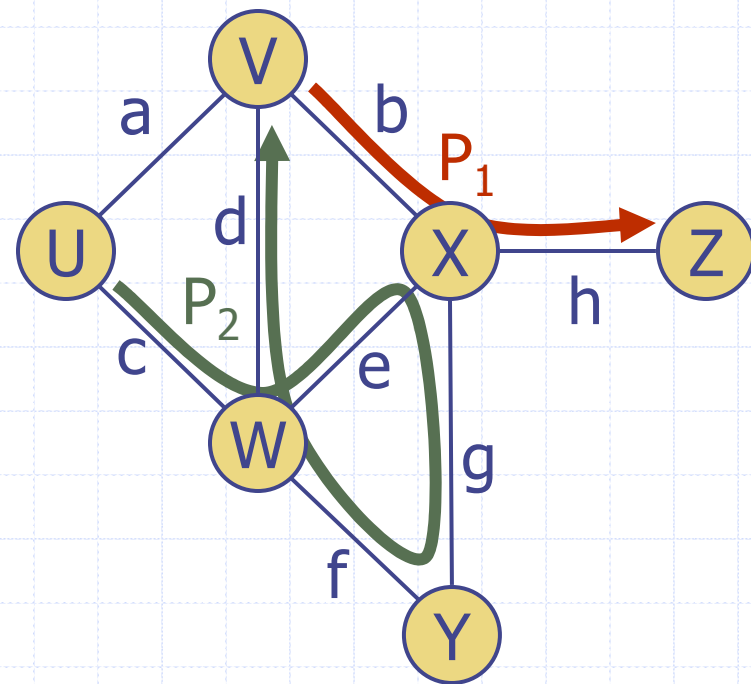
- sequence of alternating vertices and edges
- begins with a vertex
- ends with a vertex
- each edge is preceded and followed by its endpoints

◆ 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

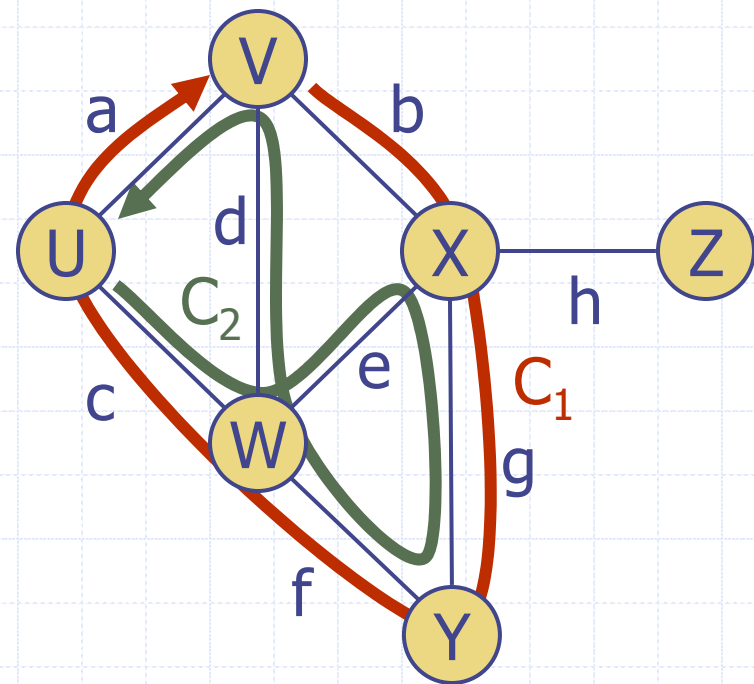
- circular sequence of alternating vertices and edges
- each edge is preceded and followed by its endpoints

◆ Simple cycle

- cycle such that all its vertices and edges are distinct

◆ Examples

- $C_1 = (V, b, X, g, Y, f, W, c, U, a, \rightarrow)$ is a simple cycle
- $C_2 = (U, c, W, e, X, g, Y, f, W, d, V, a, \rightarrow)$ is a cycle that is not simple



Properties

Property 1

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

Proof: each endpoint
is counted twice

Notation

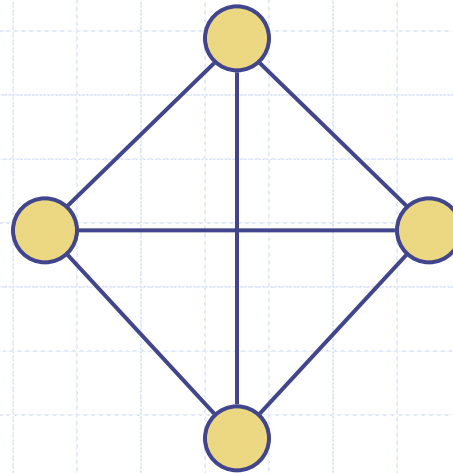
n	number of vertices
m	number of edges
$\deg(v)$	degree of vertex v

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
 $(n-1)$



Example

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

Main Methods of the Graph ADT

◆ Vertices and edges

- are positions
- store elements

◆ Accessor methods

- `aVertex()`
- `incidentEdges(v)`
- `endVertices(e)`
- `opposite(v, e)`
- `areAdjacent(v, w)`

◆ Update methods

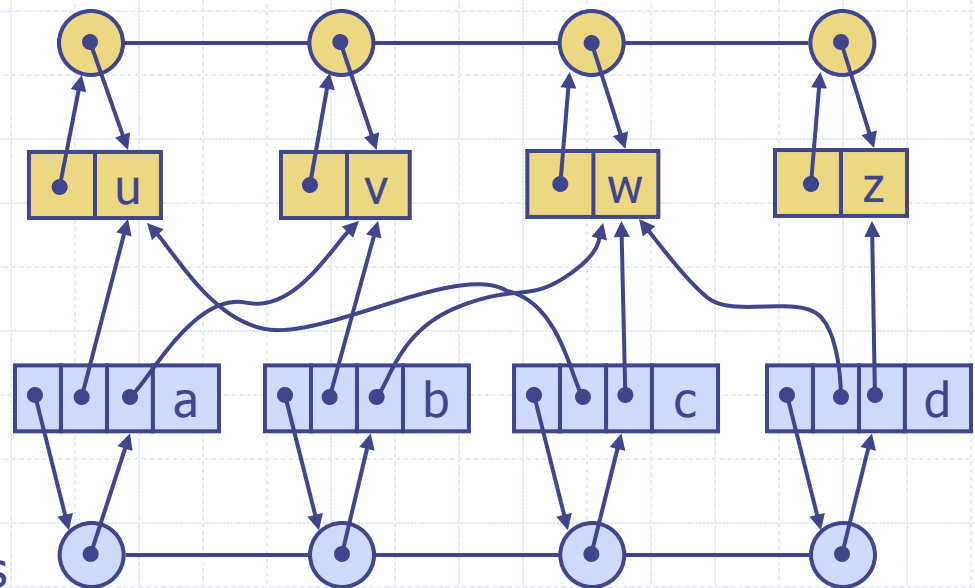
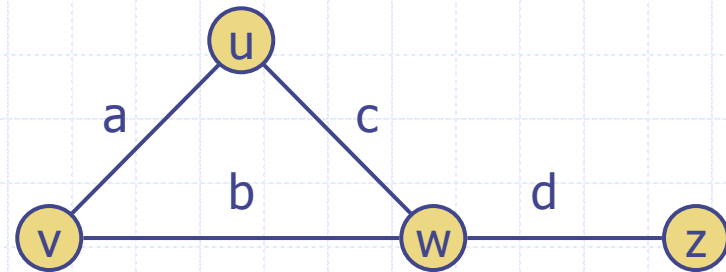
- `insertVertex(o)`
- `insertEdge(v, w, o)`
- `removeVertex(v)`
- `removeEdge(e)`
- `replace(v,x)`
- `replace(e,x)`

◆ Generic methods

- `numVertices()`
- `numEdges()`
- `vertices()`
- `edges()`

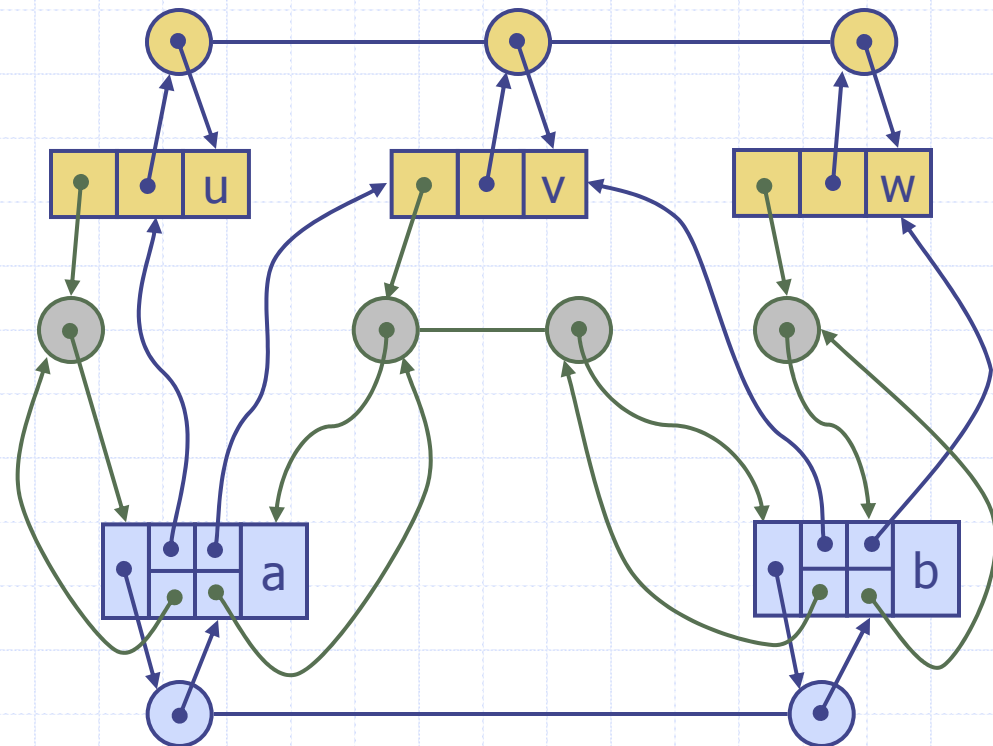
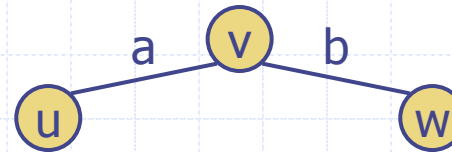
Edge List Structure

- ◆ Vertex object
 - element
 - reference to position in vertex sequence
- ◆ Edge object
 - element
 - origin vertex object
 - destination vertex object
 - reference to position in edge sequence
- ◆ Vertex sequence
 - sequence of vertex objects
- ◆ Edge sequence
 - sequence of edge objects



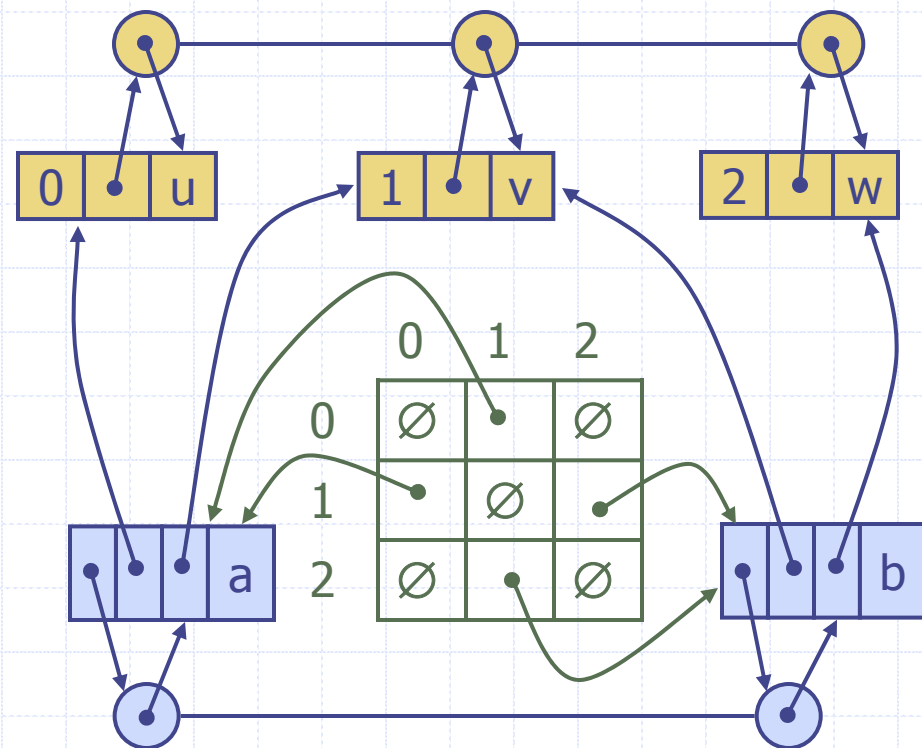
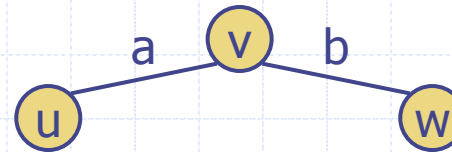
Adjacency List Structure

- ◆ Edge list structure
- ◆ Incidence sequence for each vertex
 - sequence of references to edge objects of incident edges
- ◆ Augmented edge objects
 - references to associated positions in incidence sequences of end vertices



Adjacency Matrix Structure

- ◆ Edge list structure
- ◆ Augmented vertex objects
 - Integer key (index) associated with vertex
- ◆ 2D-array adjacency array
 - Reference to edge object for adjacent vertices
 - Null for non adjacent vertices



Performance

<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
incidentEdges(v)	m	deg(v)	n
areAdjacent (v, w)	m	min(deg(v), deg(w))	1
insertVertex(o)	1	1	n^2
insertEdge(v, w, o)	1	1	1
removeVertex(v)	m	deg(v)	n^2
removeEdge(e)	1	1	1