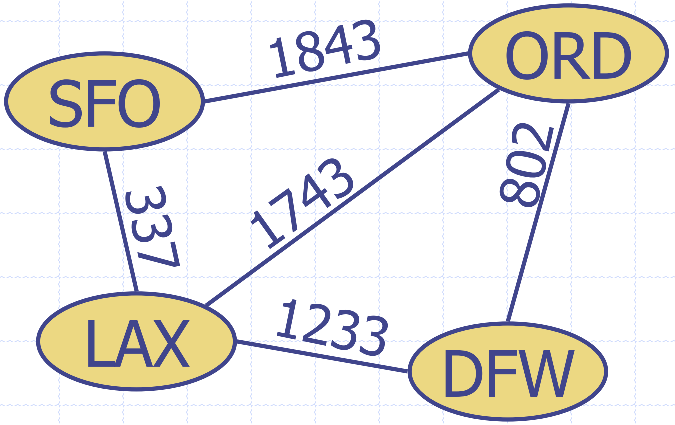
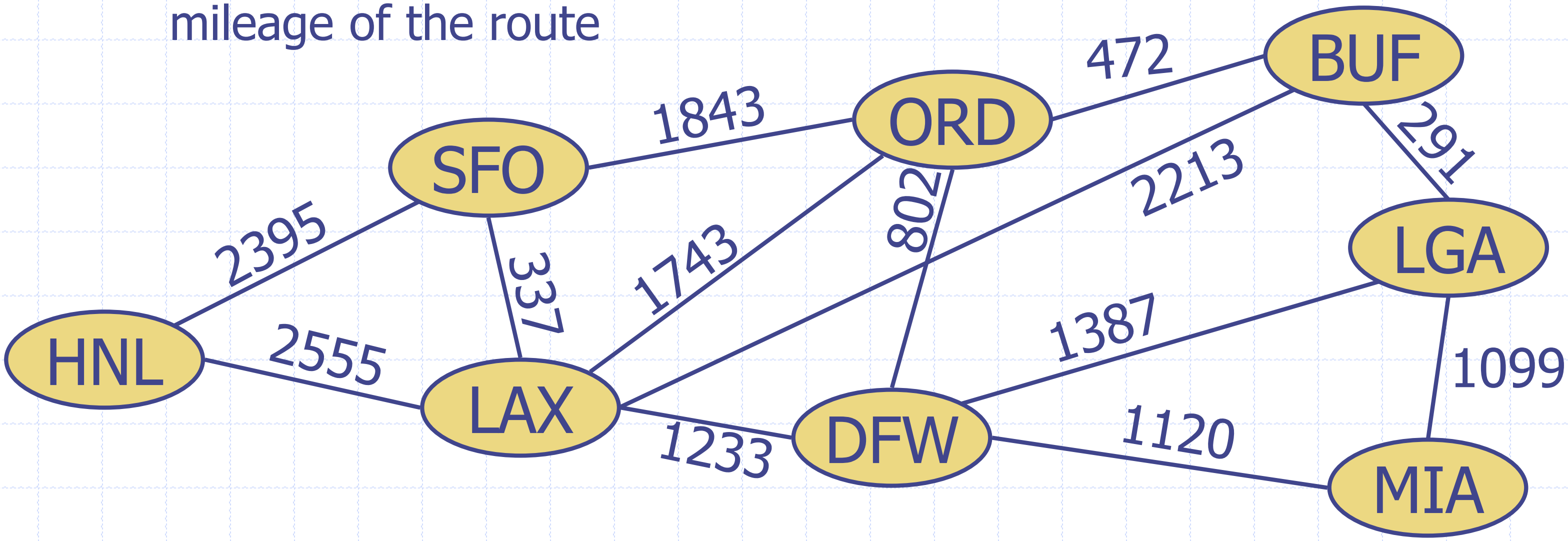
Graphs



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
  - 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

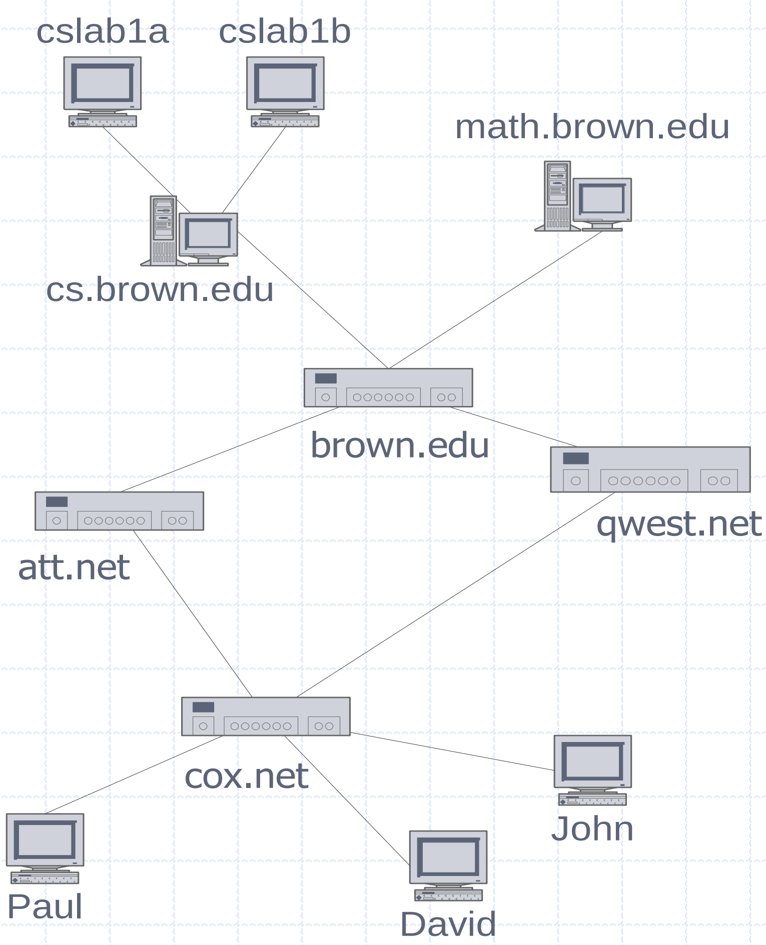


Edge Types

- 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., route network
- Undirected graph
  - all the edges are undirected
  - e.g., flight network

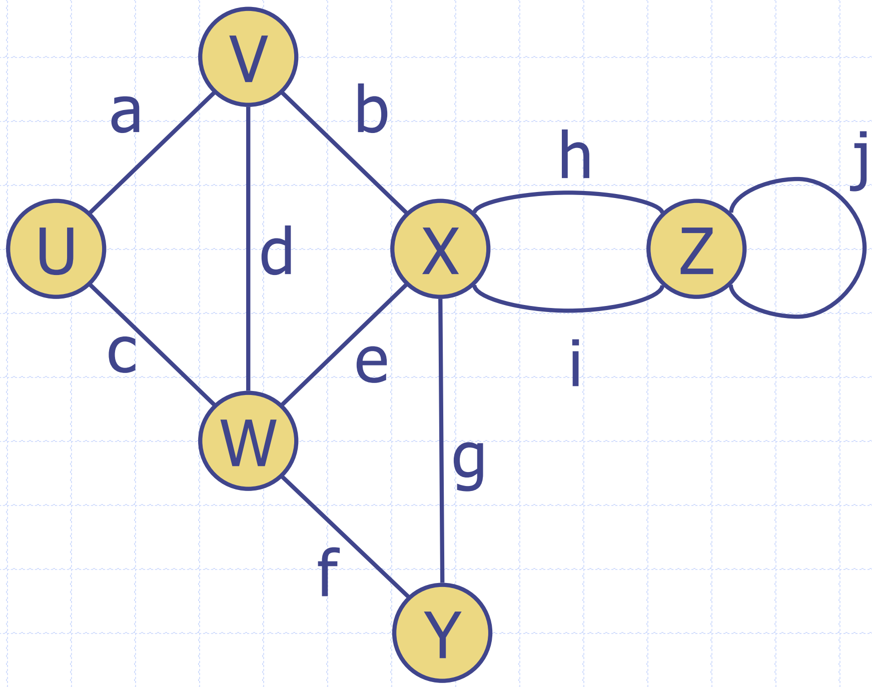
Applications

- Electronic circuits
  - Printed circuit board
  - Integrated circuit
- Transportation networks
  - Highway network
  - Flight network
- Computer networks
  - Local area network
  - Internet
  - Web
- Databases
  - Entity-relationship diagram



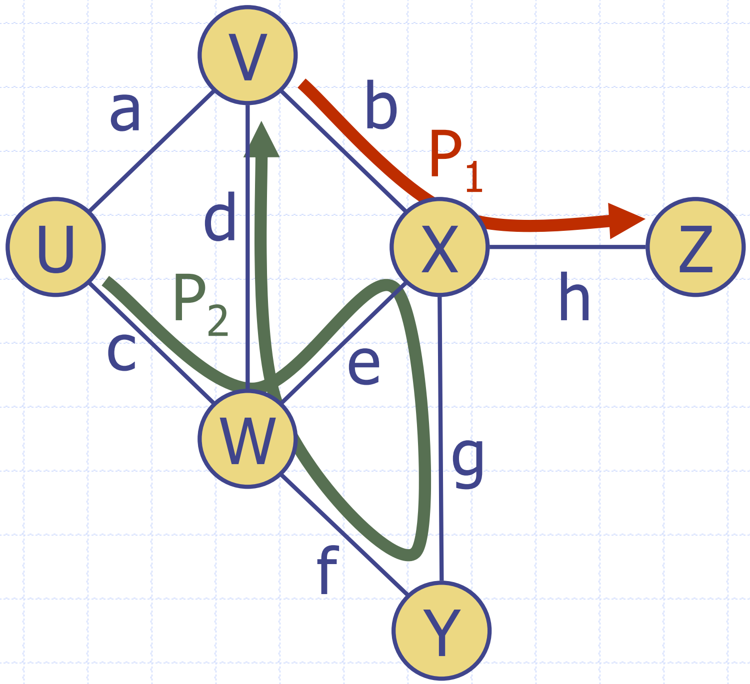
Terminology

- 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



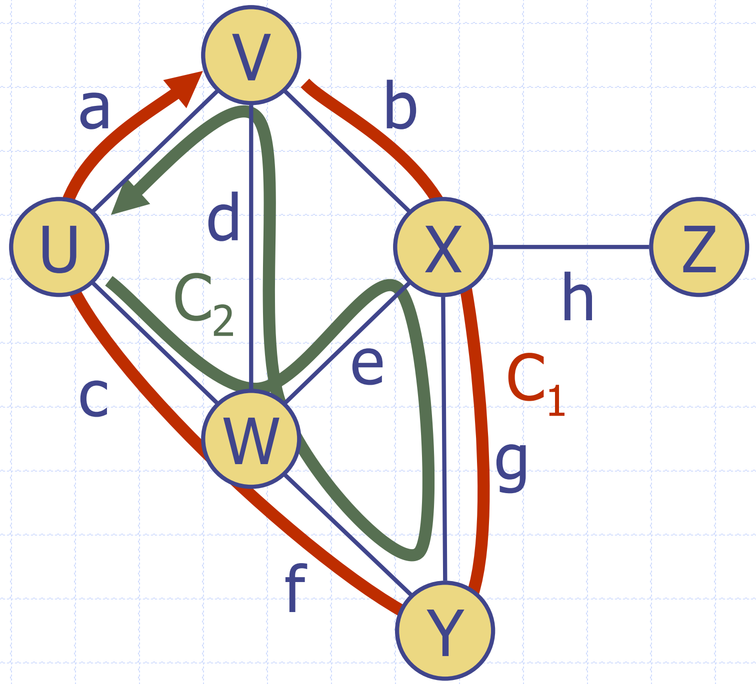
Terminology (cont.)

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



Properties

Property 1

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

Proof: each edge is counted twice

Property 2

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

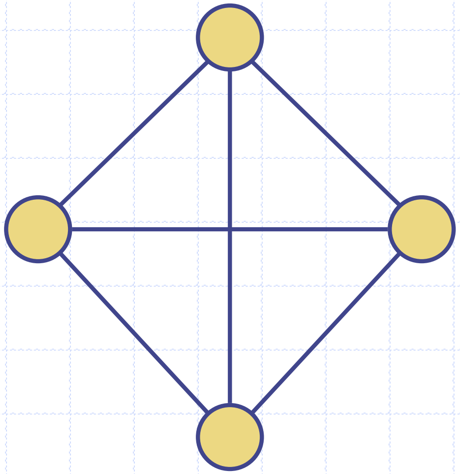
$m \leq n\,(n-1)/2$

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

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

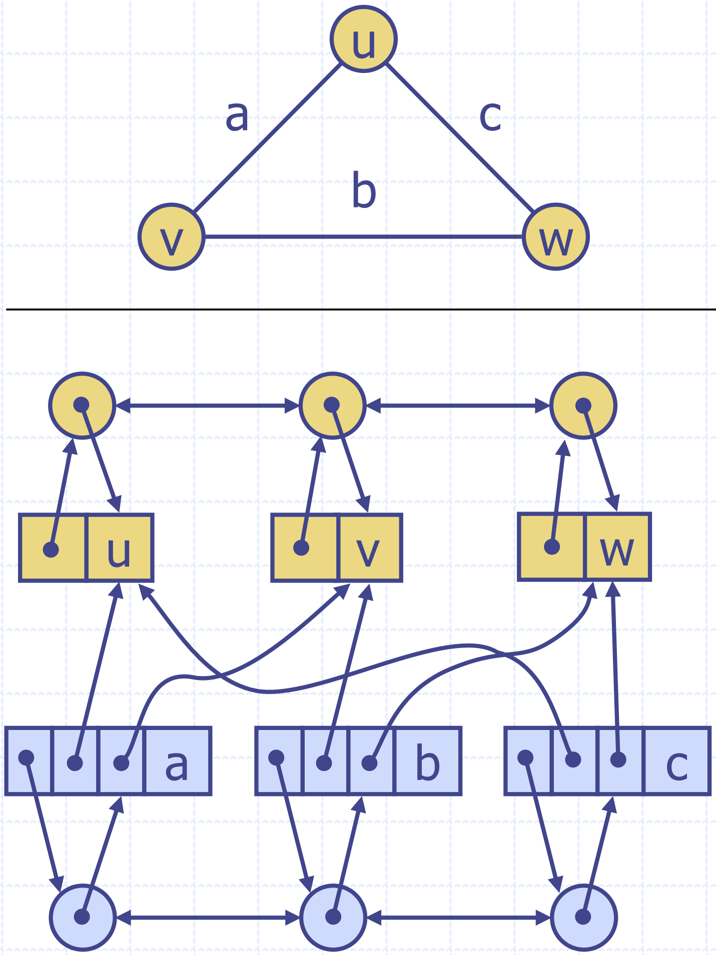
© 2010 Goodrich, Tamassia

Main Methods of the Graph ADT

- Vertices and edges
  - are positions
  - store elements
  - are objects themselves
- Accessor methods
  - e.endVertices(): a list of the two endvertices of 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

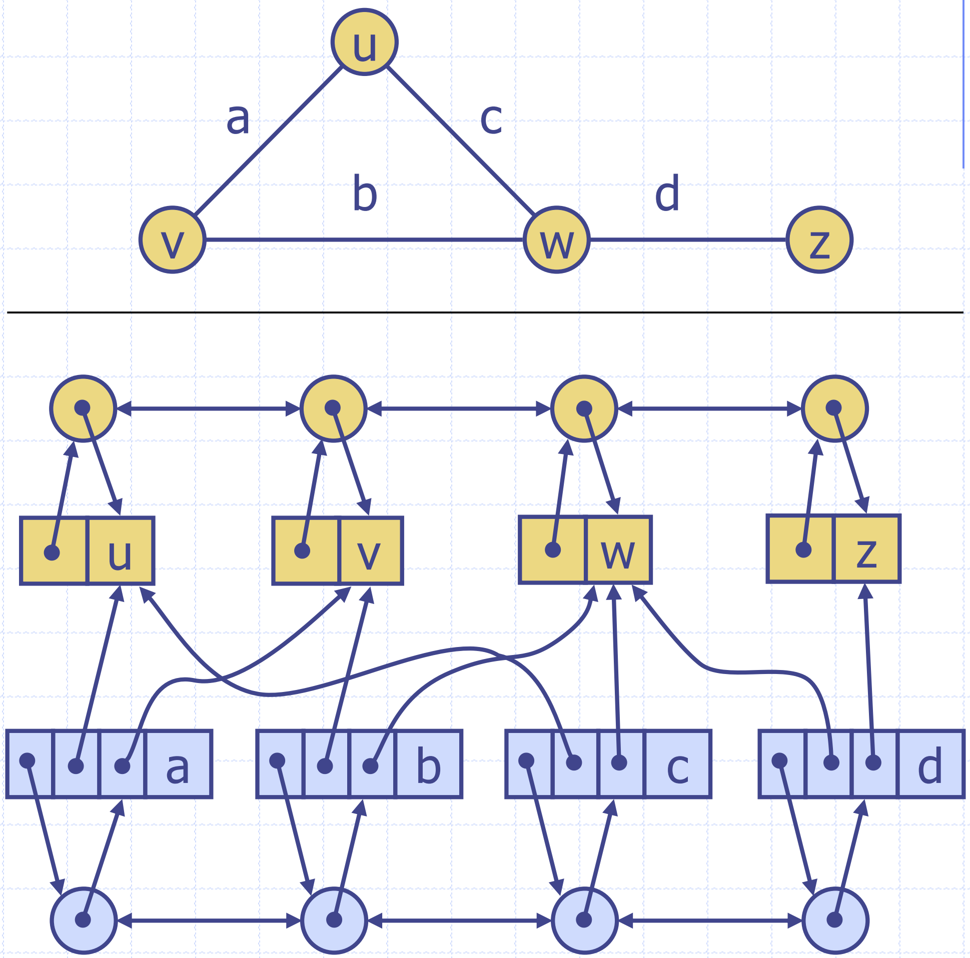
Edge List Structure

- 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



Edge List Structure

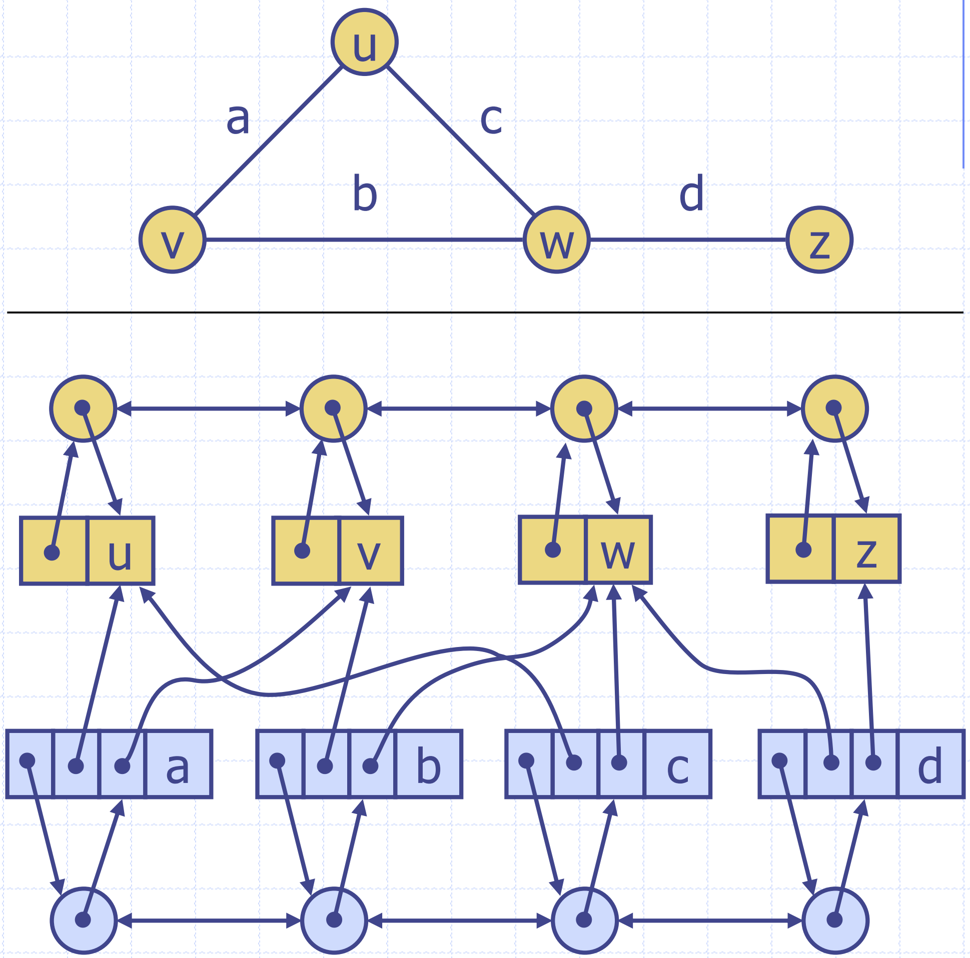
- Adding vertices:
  - Input: vertex data.
  - Create vertex object.
  - Add node to vertex list head/tail.
  - Update links.
  - O(1) runtime.
- Adding edges:
  - Input: two vertex objects, edge data.
  - Create edge object with pointers to vertices.
  - Add node to edge list.
  - Update links.
  - O(1) runtime.



© 2010 Goodrich, Tamassia

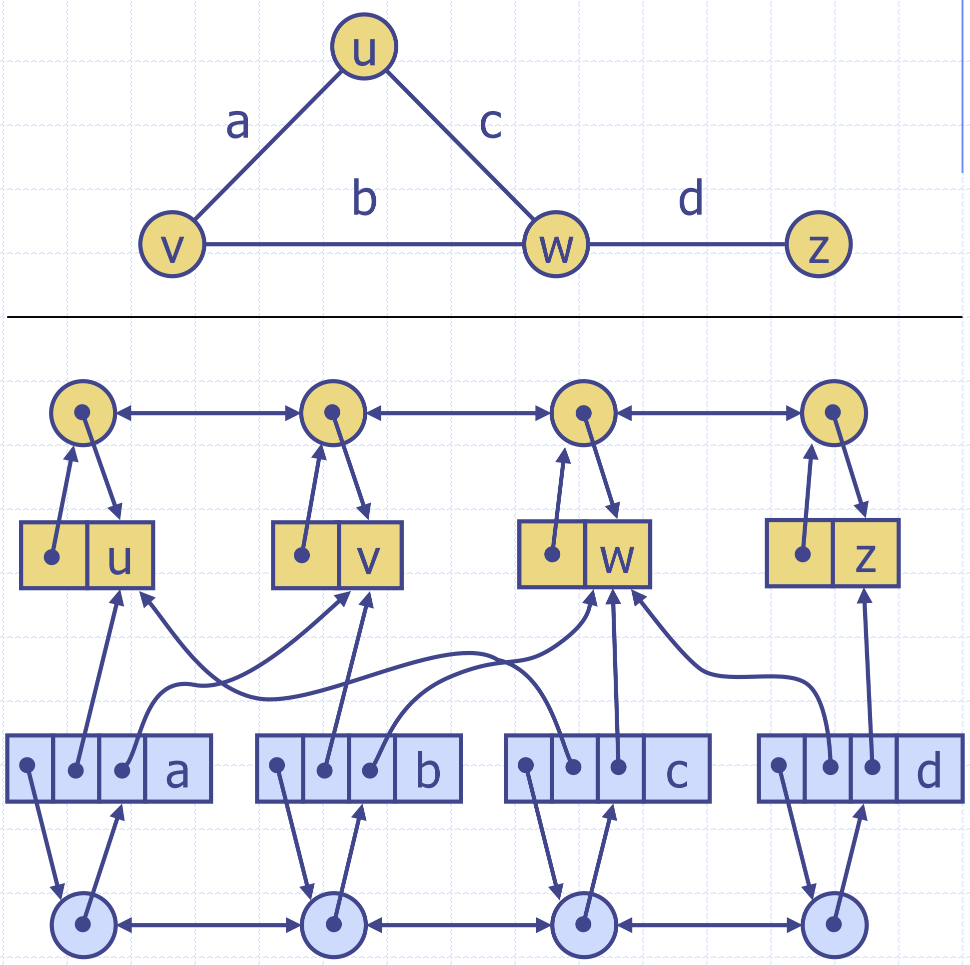
Edge List Structure

- Determine if two vertices are adjacent:
  - Input: two vertex objects.
  - From beginning of edge sequence, look for edge.
  - O(m) runtime.
- Find all adjacent edges:
  - Input: vertex object.
  - From beginning of edge sequence, look at each edge for vertex.
  - O(m) runtime.



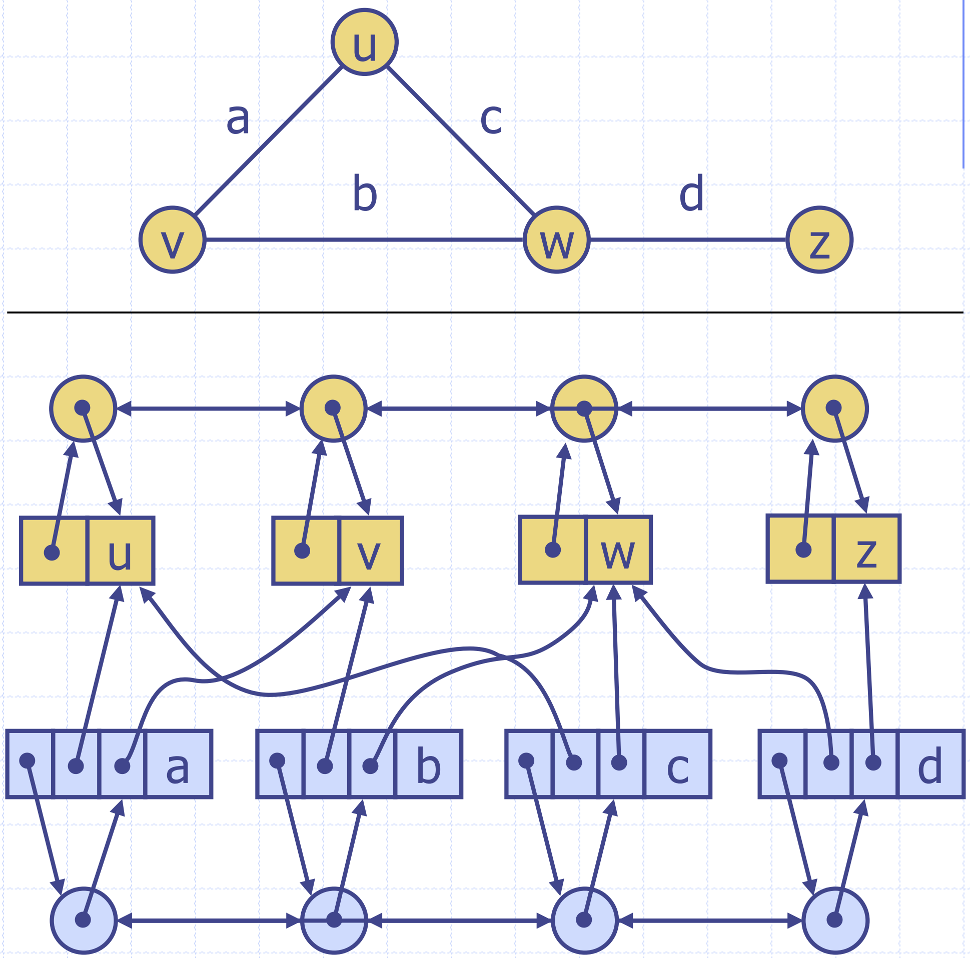
Edge List Structure

- Remove an edge:
  - Input: edge object.
  - Remove edge from edge sequence.
  - O(1) runtime.
- E.g., remove edge a.



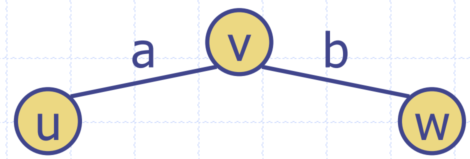
Edge List Structure

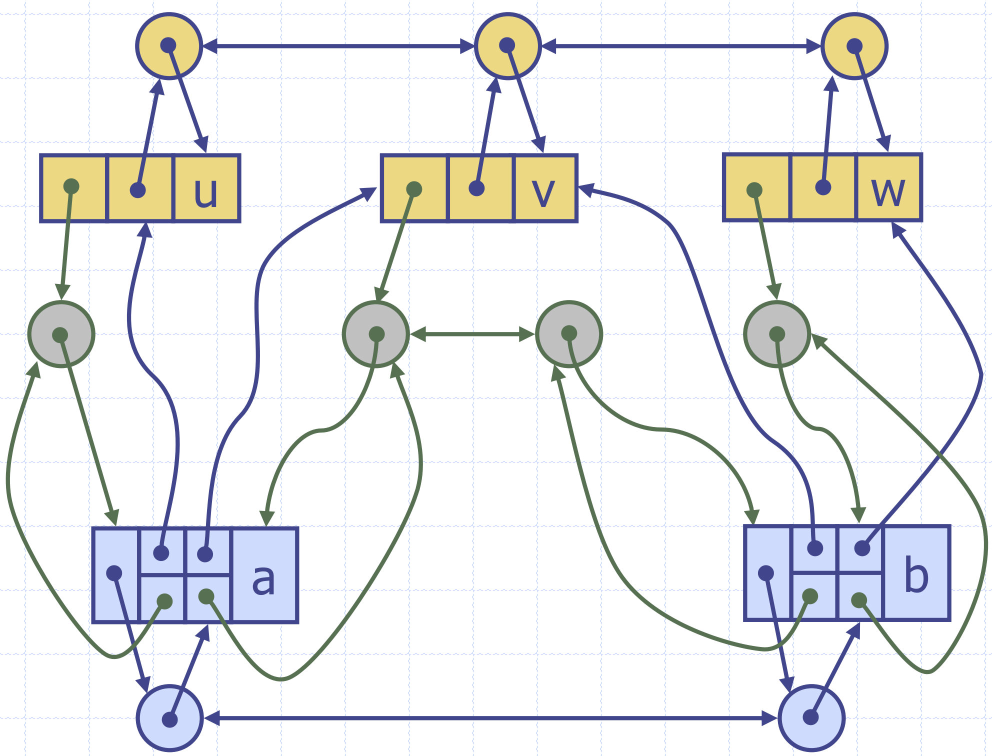
- Remove a vertex:
  - Input: vertex object.
  - From beginning of edge sequence, find edges containing vertex and remove edge.
  - Remove vertex from vertex sequence.
  - O(m)+O(1) runtime.
- E.g., remove vertex w.



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

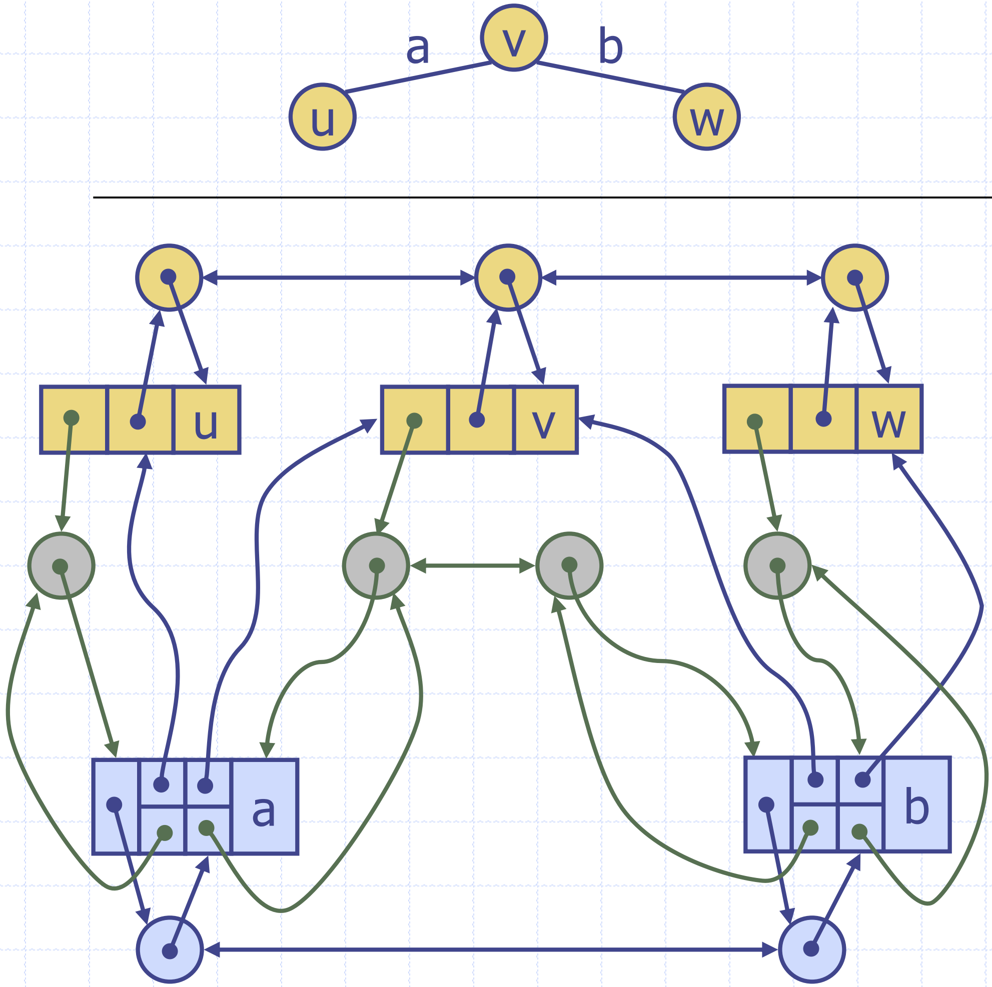




© 2010 Goodrich, Tamassia

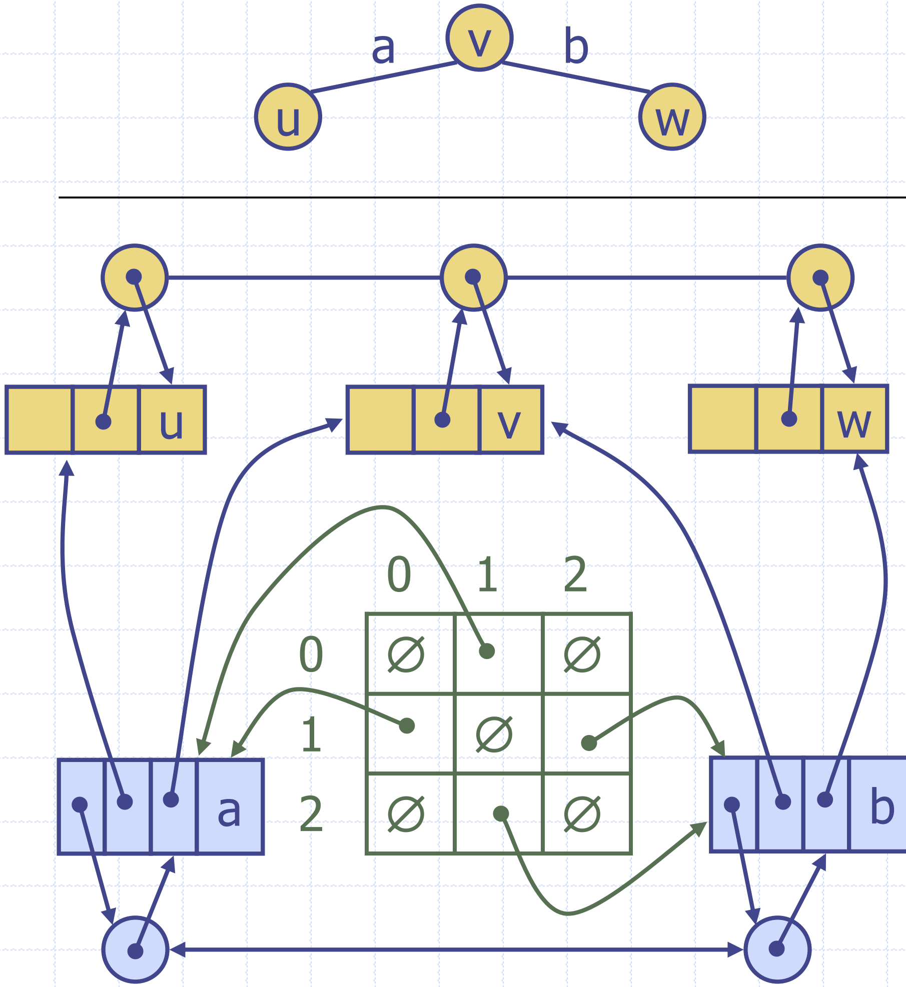
Adjacency List Structure

- Add vertex/edge:
  - Still O(1) runtime.
- Determine if two vertices are adjacent:
  - Traverse corresponding edge lists.
  - min(deg(u),deg(v)) runtime.
- Find all adjacent edges:
  - O(1) runtime.
- Remove edge:
  - O(1) runtime.
- Remove vertex:
  - Traverse corresponding edge list.
  - O(deg(v)) runtime.



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 nonadjacent vertices
- The "old fashioned" version just has 0 for no edge and 1 for edge



Performance

- n vertices, m edges - no parallel edges - no self-loops	Edge List	Adjacency List	Adjacency Matrix
Space	$n + m$	$n + m$	n^2
v.incidentEdges()	m	$\deg(v)$	n
u.isAdjacentTo (v)	m	$\min(\deg(v), \deg(w))$	1
insertVertex(o)	1	1	n^2
insertEdge(v, w, o)	1	1	1
eraseVertex(v)	m	$\deg(v)$	n^2
eraseEdge(e)	1	1	1

© 2010 Goodrich, Tamassia