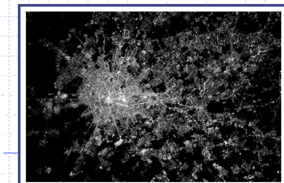
Presentation for use with the textbook, Algorithm Design and Applications, by M. T. Goodrich and R. Tamassia, Wiley, 2015

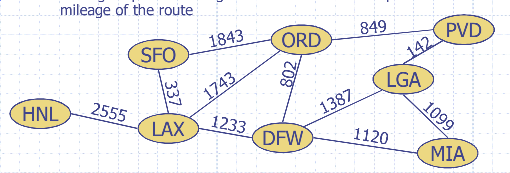
Graph Terminology and Representations



The metropolitan area of Milan, Italy at night. Astronaut photograph ISS026-E-28829, 2011. U.S. government image. NASA-JSC.

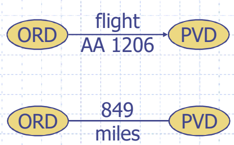
Graphs

- 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



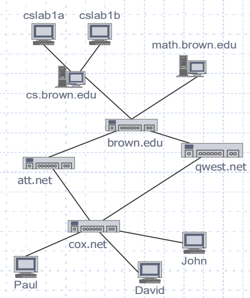
Edge Types

- 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



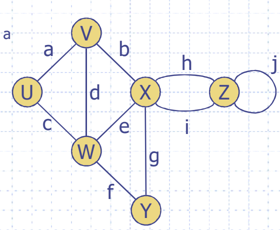
Applications

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



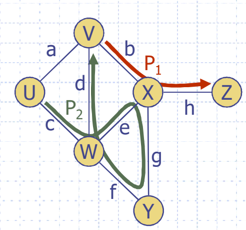
Terminology

- 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



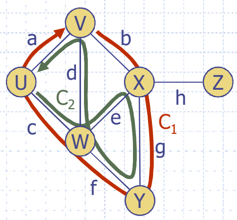
Terminology (cont.)

- 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,↵) is a simple cycle
  - C_2=(U,c,W,e,X,g,Y,f,W,d,V,a,↵) 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

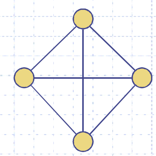
$m \leq n\,(n-1)/2$

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

Vertices and Edges

- A **graph** is a collection of **vertices** and **edges**.
- A **Vertex** is can be an abstract unlabeled object or it can be labeled (e.g., with an integer number or an airport code) or it can store other objects
- An **Edge** can likewise be an abstract unlabeled object or it can be labeled (e.g., a flight number, travel distance, cost), or it can also store other objects.

Graph Operations

- Return the number, n, of vertices in G.
- Return the number, m, of edges in G.
- Return a set or list containing all n vertices in G.
- Return a set or list containing all m edges in G.
- Return some vertex, v, in G.
- Return the degree, deg(v), of a given vertex, v, in G.
- Return a set or list containing all the edges incident upon a given vertex, v, in G.
- Return a set or list containing all the vertices adjacent to a given vertex, v, in G.
- Return the two end vertices of an edge, e, in G; if e is directed, indicate which vertex is the origin of e and which is the destination of e.
- Return whether two given vertices, v and w, are adjacent in G.

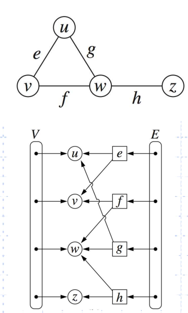
Graph Operations, Continued

- Indicate whether a given edge, e, is directed in G.
- Return the in-degree of v, inDegree(v).
- Return a set or list containing all the incoming (or outgoing) edges incident upon a given vertex, v, in G.
- Return a set or list containing all the vertices adjacent to a given vertex, v, along incoming (or outgoing) edges in G.

- Insert a new directed (or undirected) edge, e, between two given vertices, v and w, in G.
- Insert a new (isolated) vertex, v, in G.
- Remove a given edge, e, from G.
- Remove a given vertex, v, and all its incident edges from G.

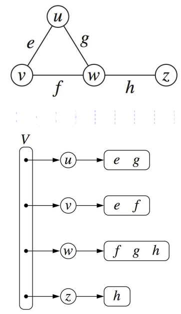
Edge List Structure

- 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



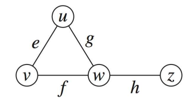
Adjacency List Structure

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



		0	1	2	3
u →	0		e	g	
v →	1	e		f	
w →	2	g	f		h
z →	3			h	

Performance

(All bounds are big-oh running times, except for "Space")

▪ 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