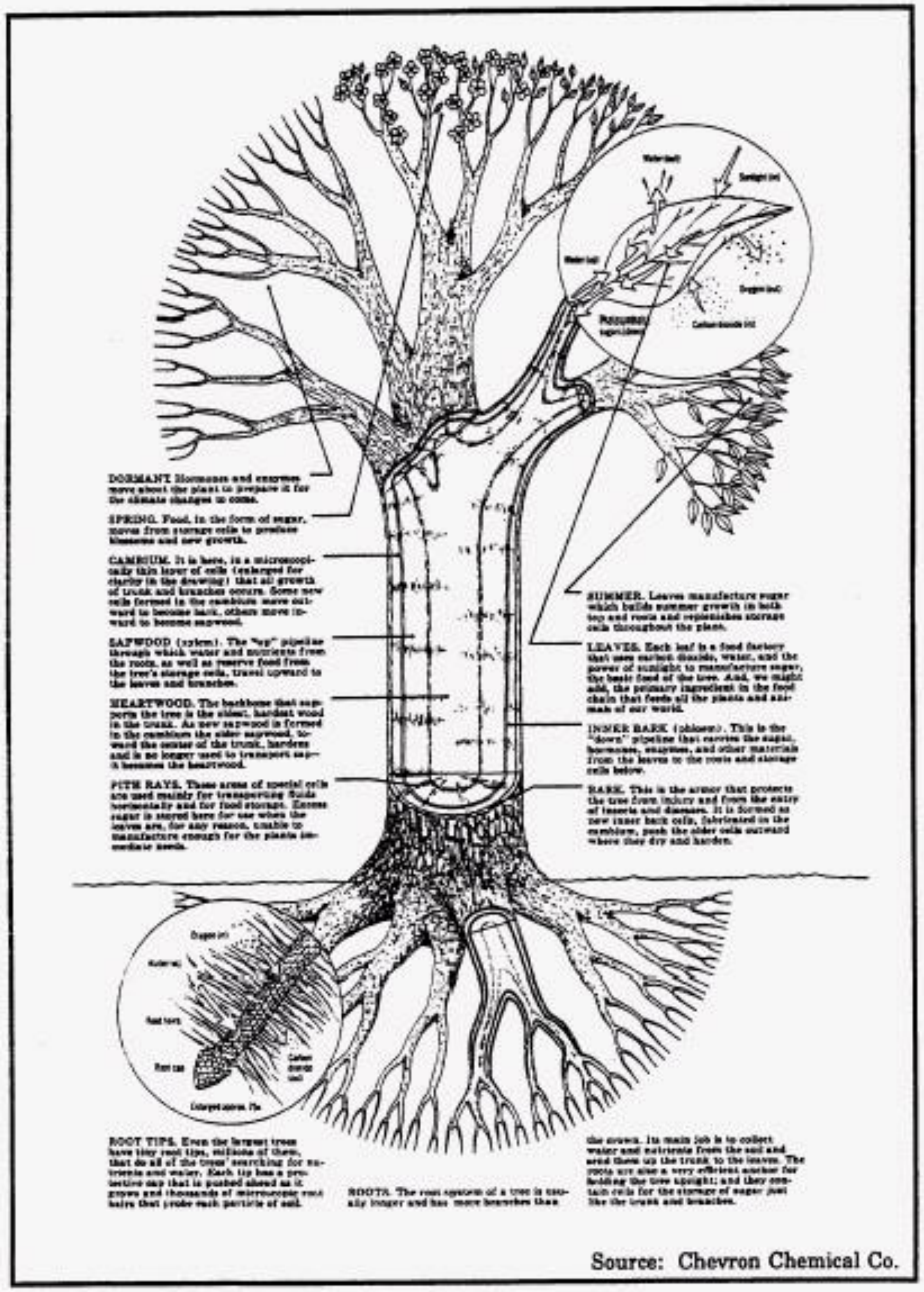
Tree



DORMANT. Hormones and enzymes move about the plant to prepare it for the climate changes to come.

SPRING. Food, in the form of sugar, moves from storage cells to produce blossoms and new growth.

CAMBIUM. It is here, in a microscopically thin layer of cells (enlarged for clarity in the drawing) that all growth of trunk and branches occurs. Some new cells formed in the cambium move outward to become bark, others move inward to become sapwood.

SAPWOOD (xylem). The "up" pipeline through which water and nutrients from the roots, as well as reserve food from the tree's storage cells, travel upward to the leaves and branches.

HEARTWOOD. The backbone that supports the tree is the oldest, hardest wood in the trunk. As new sapwood is formed in the cambium the older sapwood, toward the center of the trunk, hardens and is no longer used to transport sap—it becomes the heartwood.

PITH RAYS. These areas of special cells are used mainly for transporting fluids horizontally and for food storage. Excess sugar is stored here for use when the leaves are, for any reason, unable to manufacture enough for the plants immediate needs.

SUMMER. Leaves manufacture sugar which builds summer growth in both top and roots and replenishes storage cells throughout the plant.

LEAVES. Each leaf is a food factory that uses carbon dioxide, water, and the power of sunlight to manufacture sugar, the basic food of the tree. And, we might add, the primary ingredient in the food chain that feeds all the plants and animals of our world.

INNER BARK (phloem). This is the "down" pipeline that carries the sugar, hormones, enzymes, and other materials from the leaves to the roots and storage cells below.

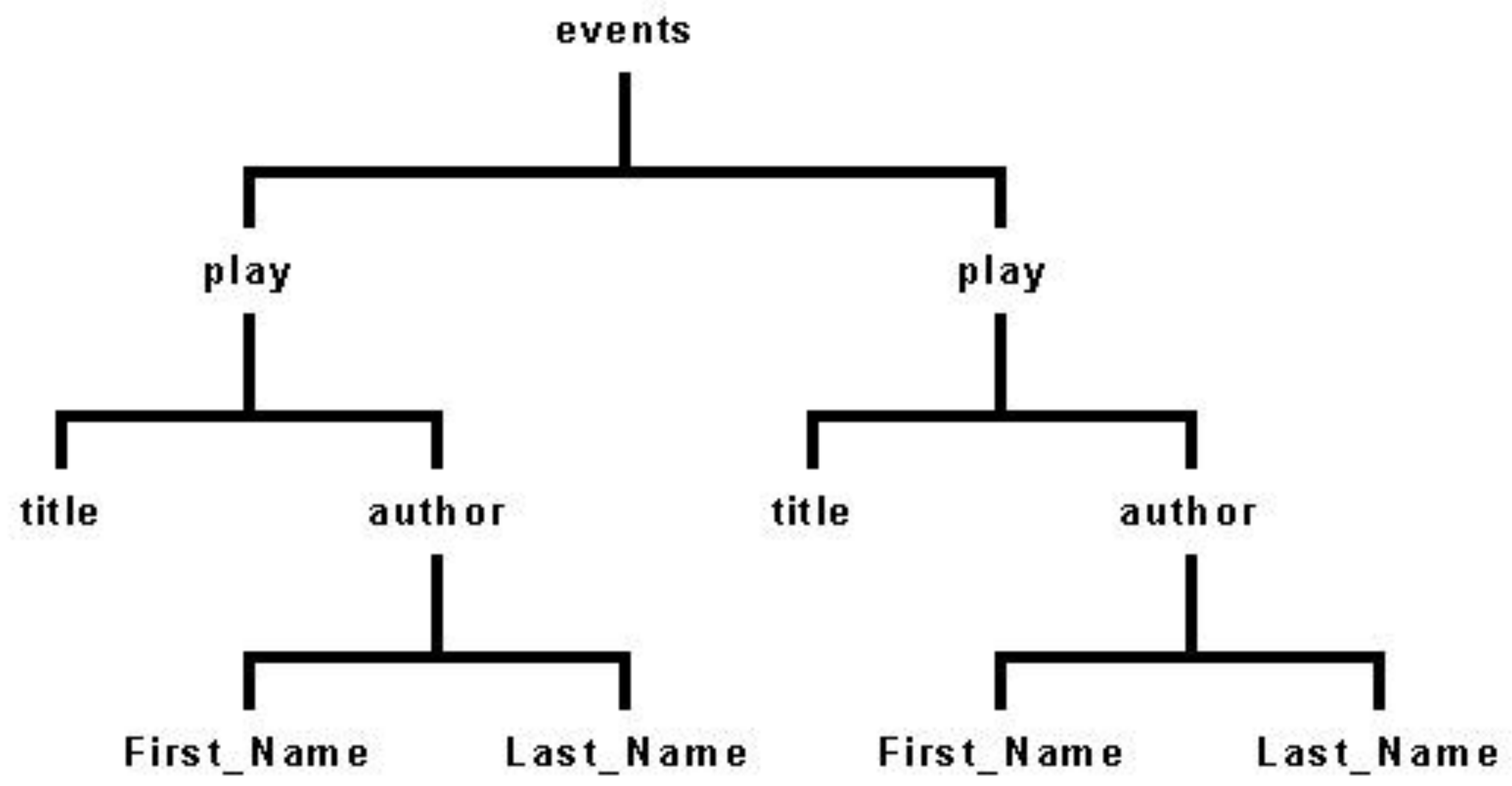
BARK. This is the armor that protects the tree from injury and from the entry of insects and diseases. It is formed as new inner bark cells, fabricated in the cambium, push the older cells outward where they dry and harden.

ROOT TIPS. Even the largest trees have tiny root tips, millions of them, that do all of the trees' searching for nutrients and water. Each tip has a protective cap that is pushed ahead as it grows and thousands of microscopic root hairs that probe each particle of soil.

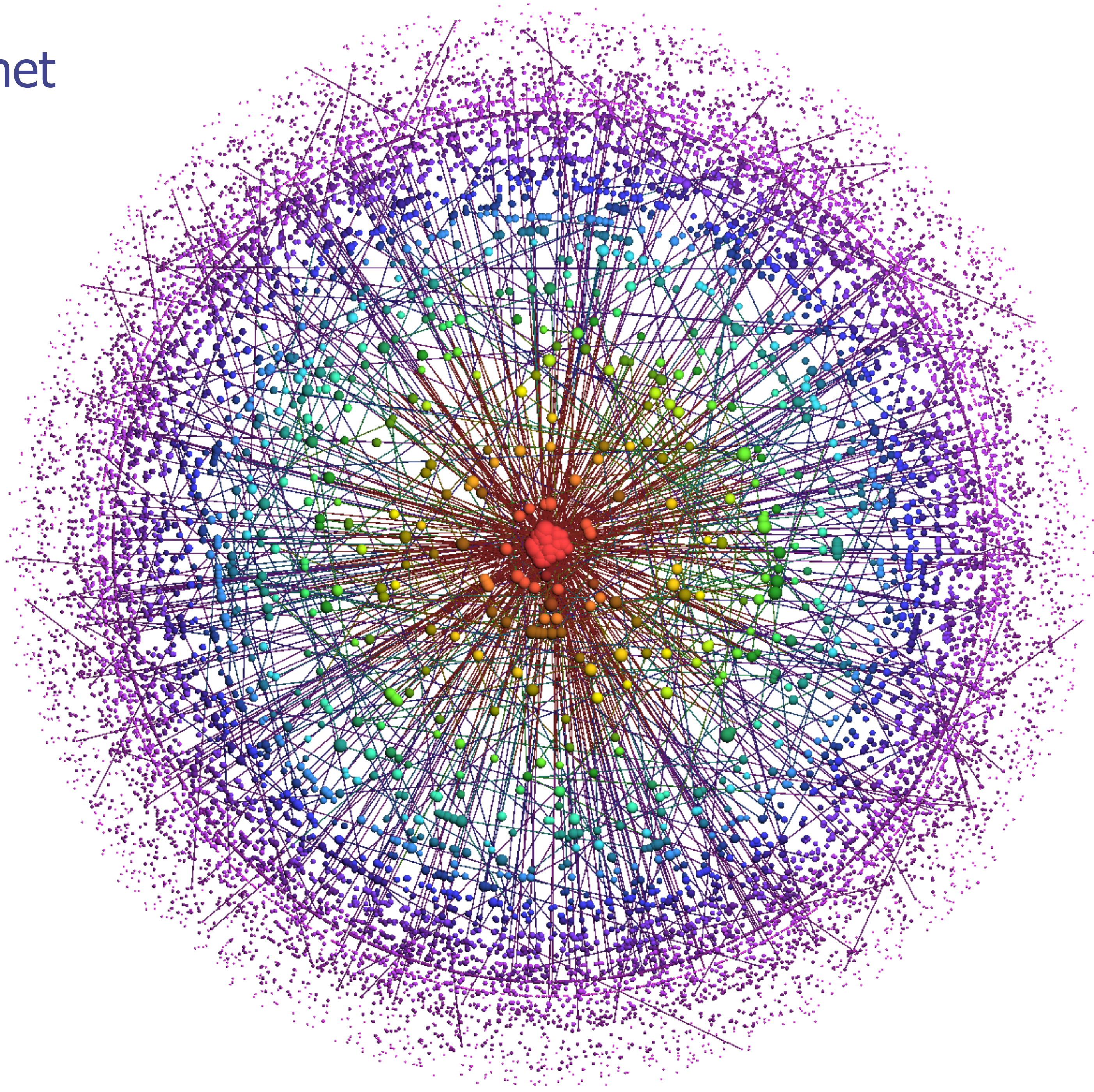
ROOTS. The root system of a tree is usually longer and has more branches than the crown. Its main job is to collect water and nutrients from the soil and send them up the trunk to the leaves. The roots are also a very efficient anchor for holding the tree upright; and they contain cells for the storage of sugar just like the trunk and branches.

Source: Chevron Chemical Co.

Tree

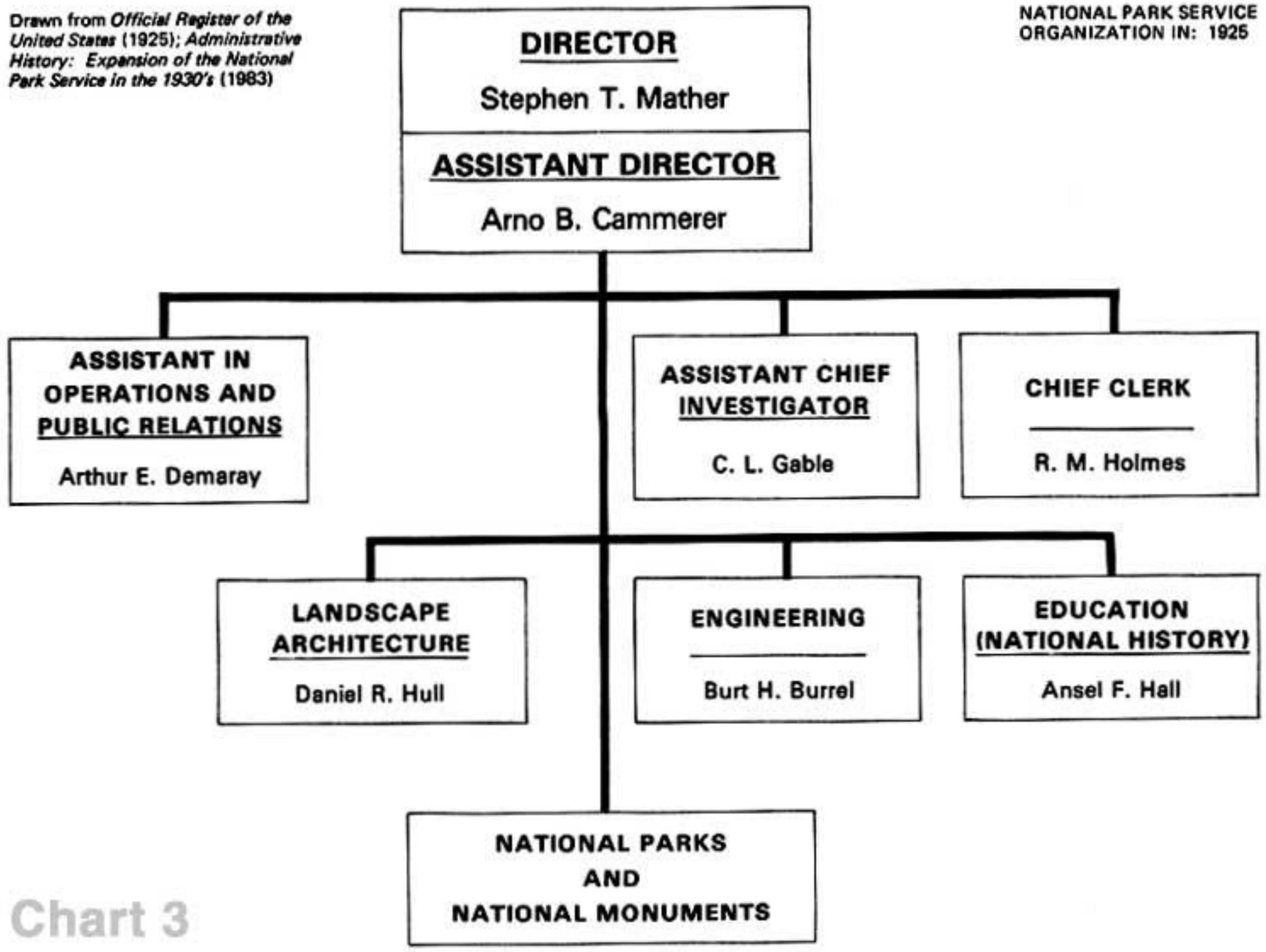

events
play
play
title
author
title
author
First_Name
Last_Name
First_Name
Last_Name

Internet

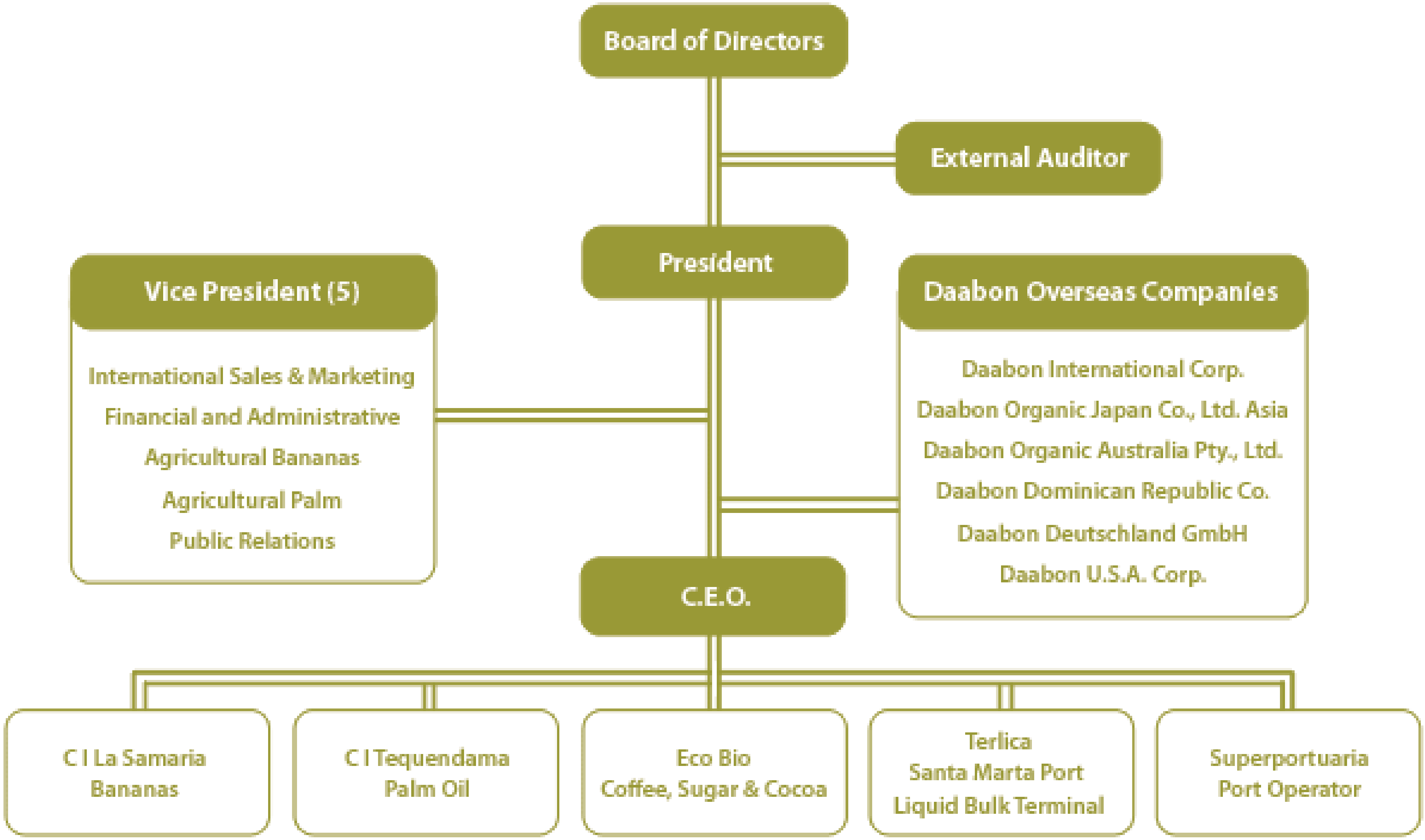


31
30
29
28
27
26
25
24
23
22
21
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1

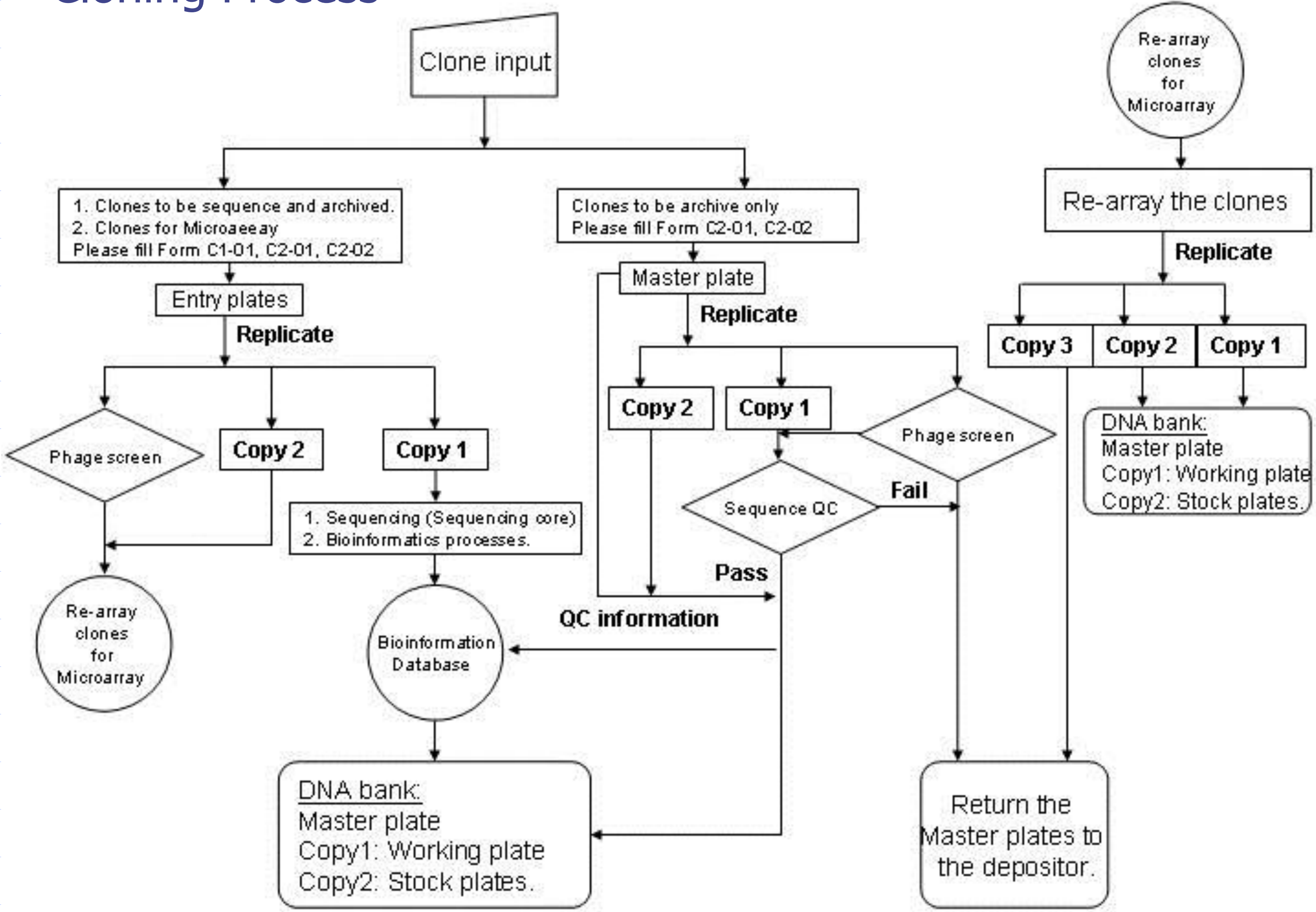
Organizational chart


Drawn from *Official Register of the United States* (1925); *Administrative History: Expansion of the National Park Service in the 1930's* (1983)
NATIONAL PARK SERVICE ORGANIZATION IN: 1925
DIRECTOR
Stephen T. Mather
ASSISTANT DIRECTOR
Arno B. Cammerer
ASSISTANT IN OPERATIONS AND PUBLIC RELATIONS
Arthur E. Demaray
ASSISTANT CHIEF INVESTIGATOR
C. L. Gable
CHIEF CLERK
R. M. Holmes
LANDSCAPE ARCHITECTURE
Daniel R. Hull
ENGINEERING
Burt H. Burrel
EDUCATION (NATIONAL HISTORY)
Ansel F. Hall
NATIONAL PARKS AND NATIONAL MONUMENTS
Chart 3

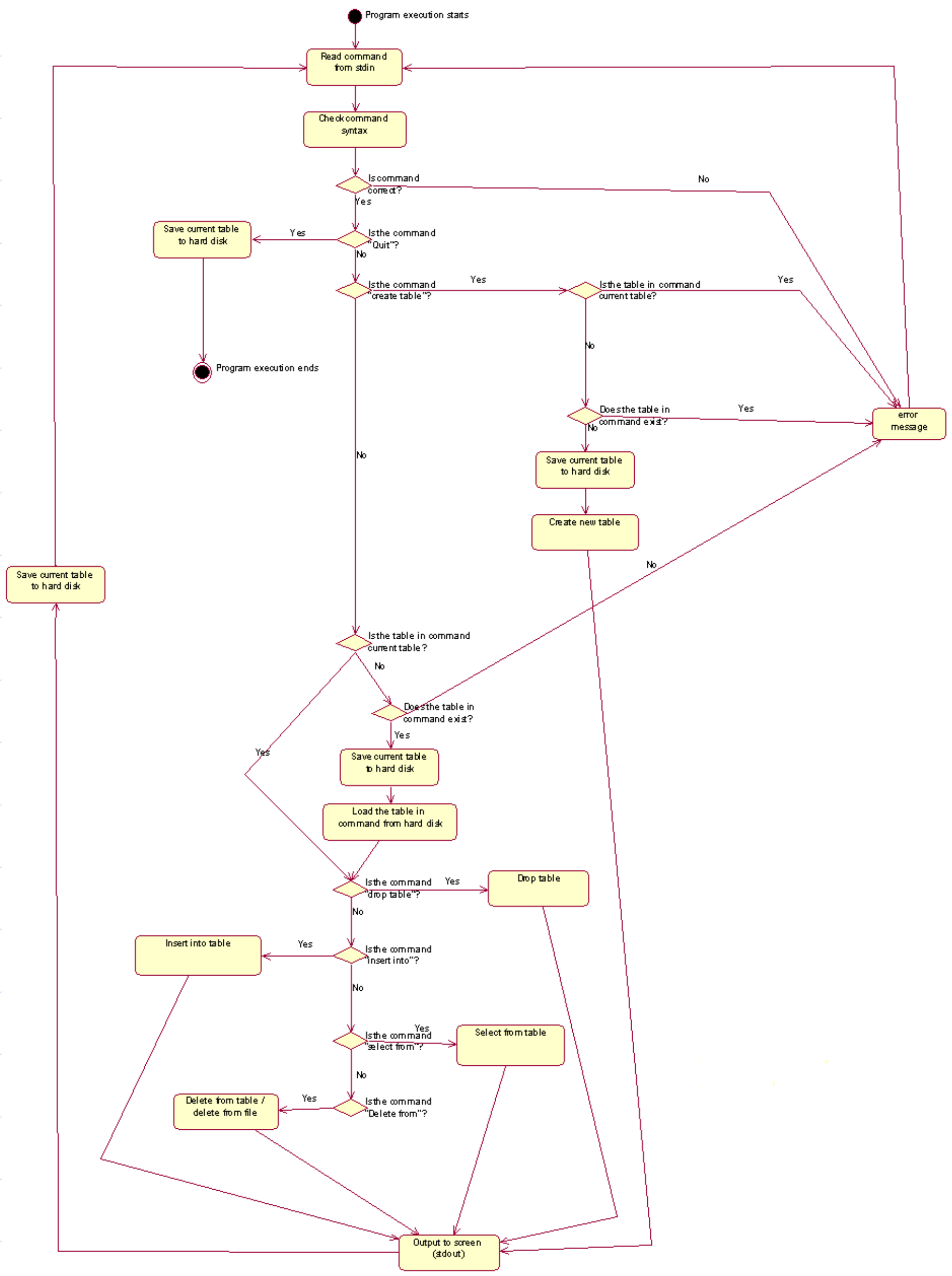
Organizational Chart


Board of Directors
External Auditor
President
Vice President (5)
International Sales & Marketing
Financial and Administrative
Agricultural Bananas
Agricultural Palm
Public Relations
Daabon Overseas Companies
Daabon International Corp.
Daabon Organic Japan Co., Ltd. Asia
Daabon Organic Australia Pty., Ltd.
Daabon Dominican Republic Co.
Daabon Deutschland GmbH
Daabon U.S.A. Corp.
C.E.O.
C I La Samaria Bananas
C I Tequendama Palm Oil
Eco Bio Coffee, Sugar & Cocoa
Terlica Santa Marta Port Liquid Bulk Terminal
Superportuaria Port Operator

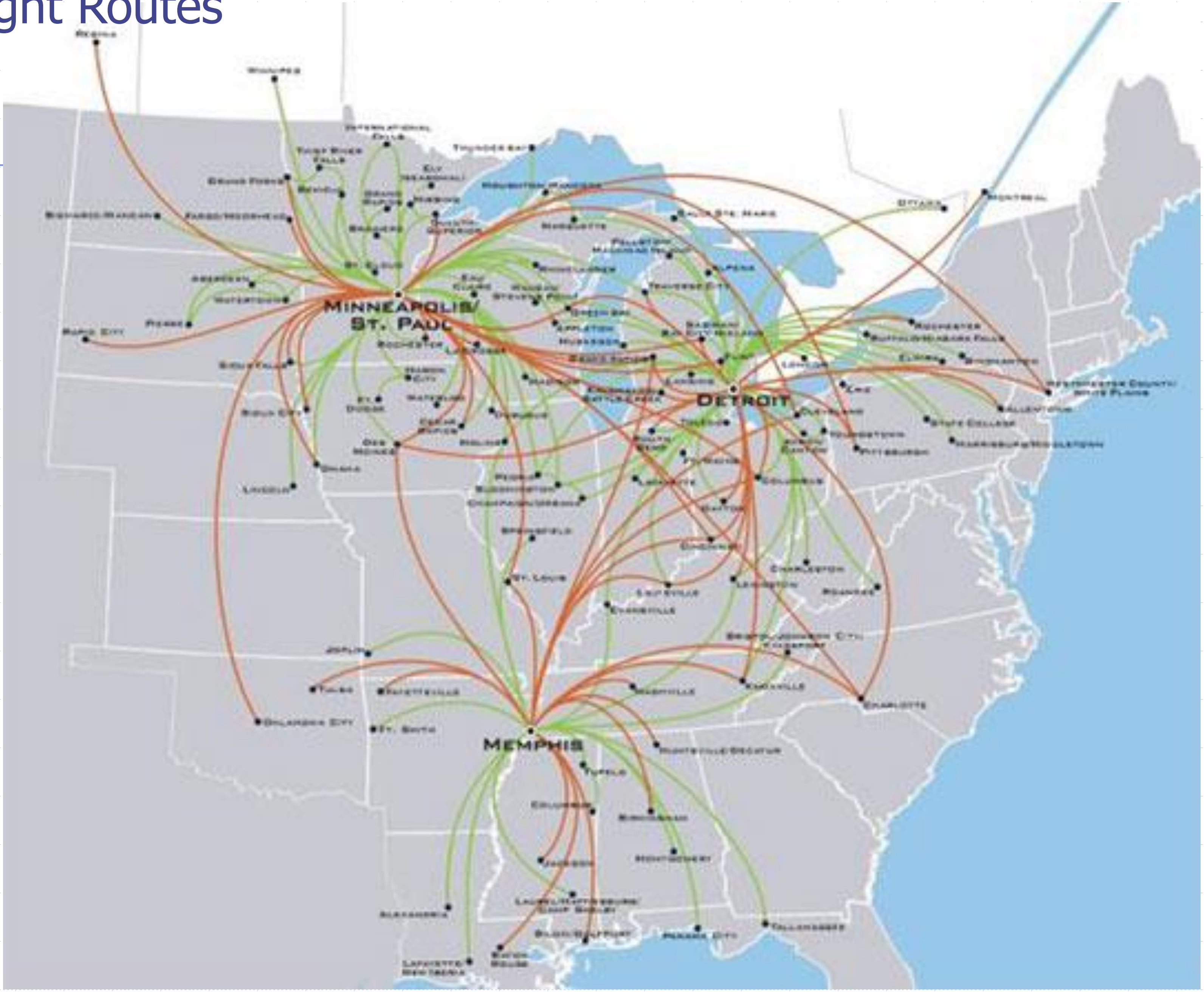
Cloning Process



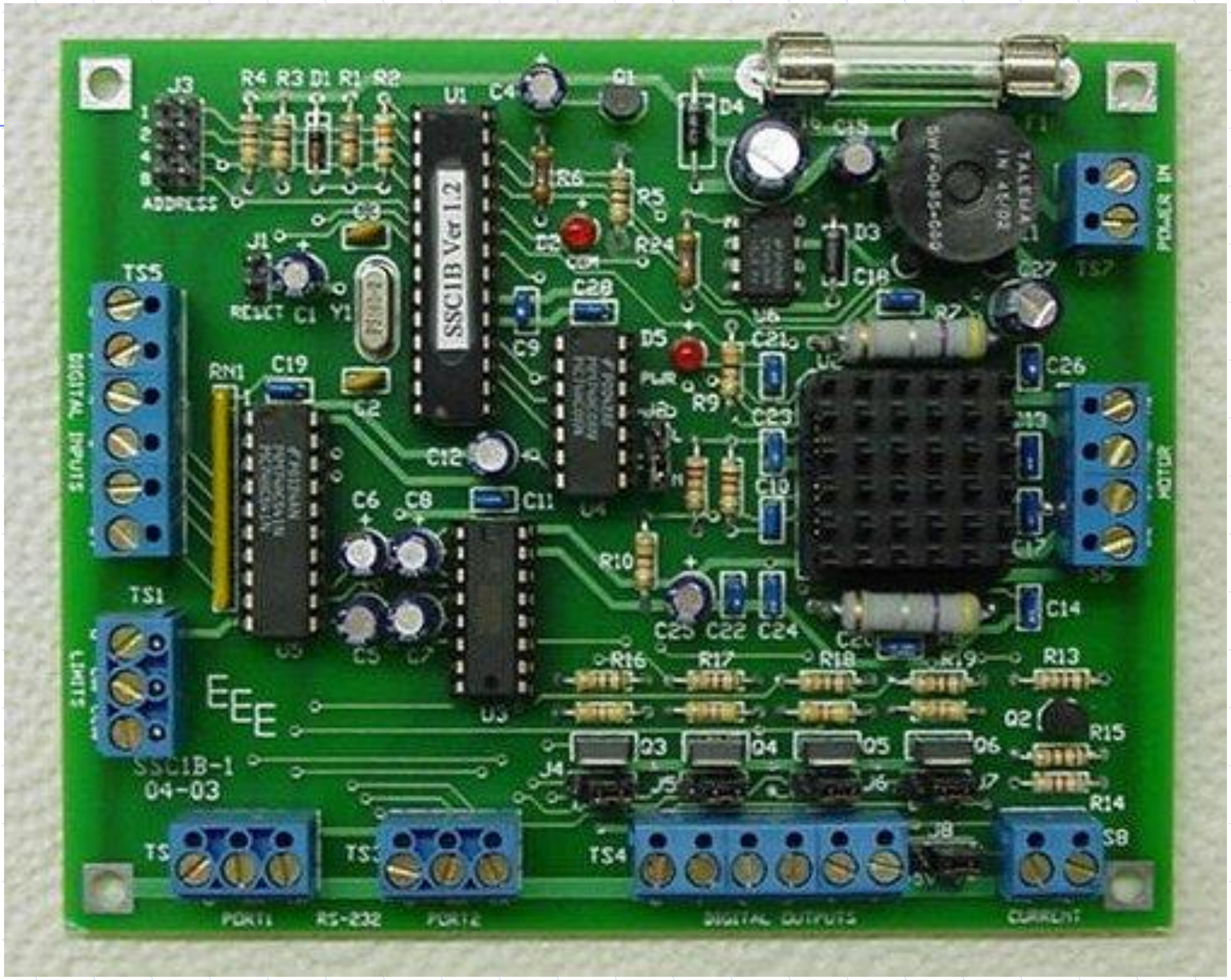
Program Flow Chart


Program execution starts
Read command from stdin
Check command syntax
Is command correct?
No
Yes
Is the command "Quit"?
Yes
Save current table to hard disk
Program execution ends
No
Is the command "create table"?
Yes
Is the table in command current table?
Yes
No
Does the table in command exist?
Yes
error message
No
Save current table to hard disk
Create new table
No
No
Is the table in command current table?
No
Does the table in command exist?
Yes
Yes
Save current table to hard disk
Load the table in command from hard disk
Is the command "drop table"?
Yes
Drop table
No
Is the command "Insert into"?
Yes
Insert into table
No
Is the command "select from"?
Yes
Select from table
No
Is the command "Delete from"?
Yes
Delete from table / delete from file
Output to screen (stdout)
Save current table to hard disk

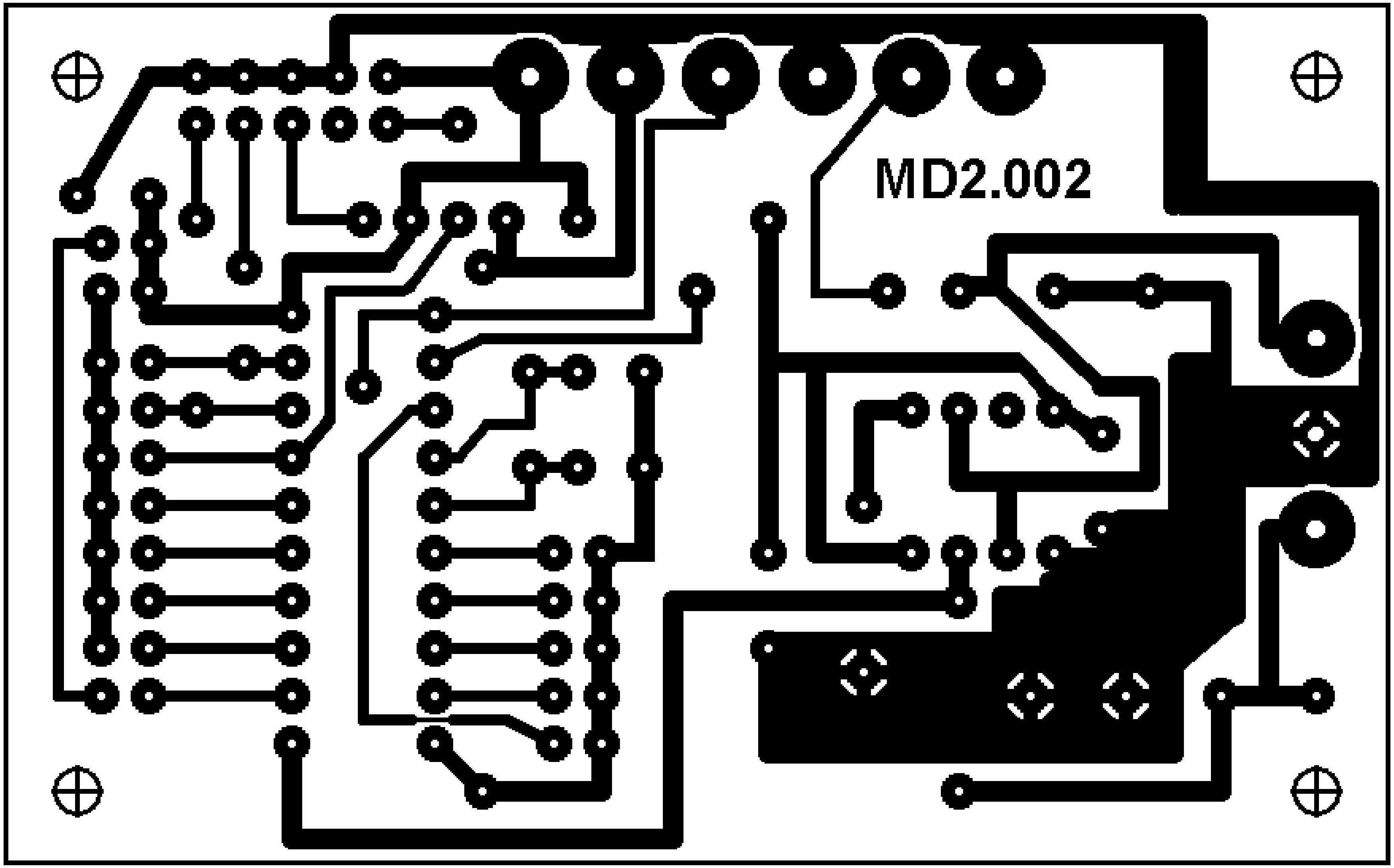
Flight Routes



Electronic Circuit Board



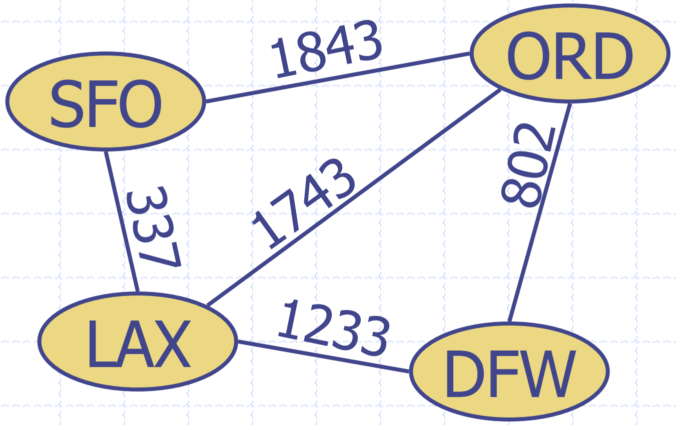
Electronic Circuit Board



What do these have in common?

- They are graphs…

Graphs

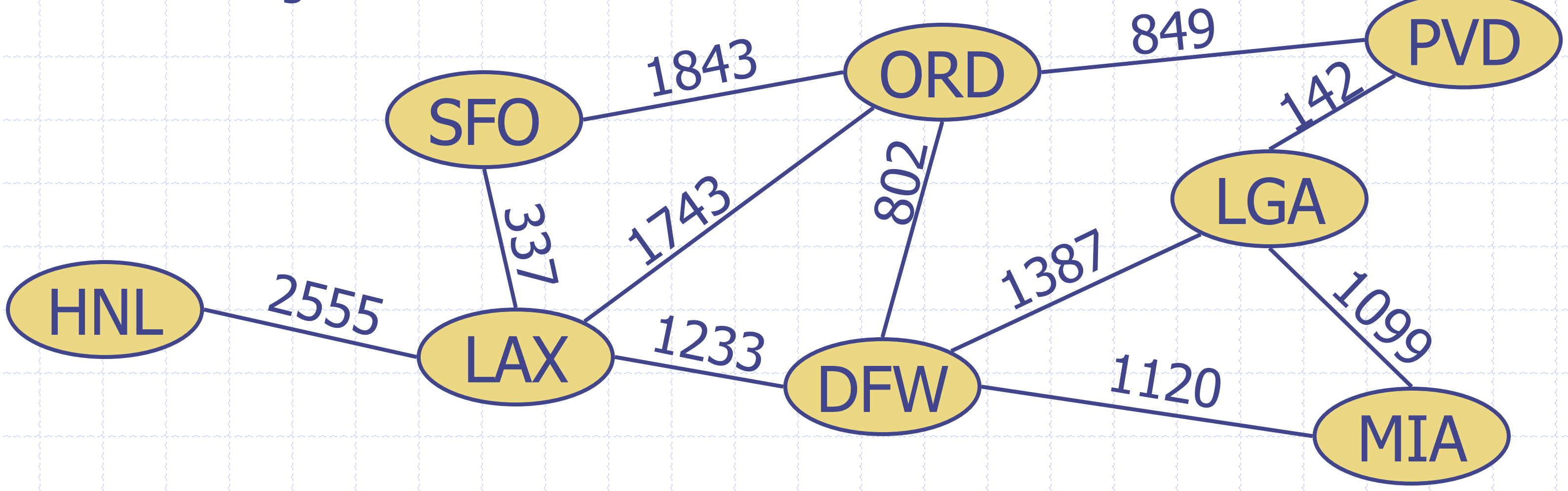


Outline and Reading

- Graphs
  - Definition
  - Applications
  - Terminology
  - Properties
  - ADT
- Data structures for graphs
  - Edge list structure
  - Adjacency list structure
  - Adjacency matrix structure

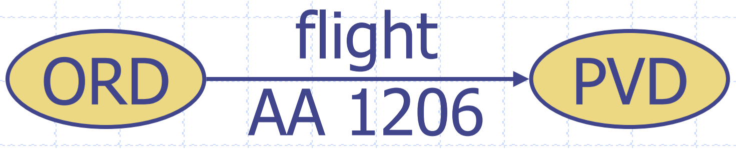
Graph

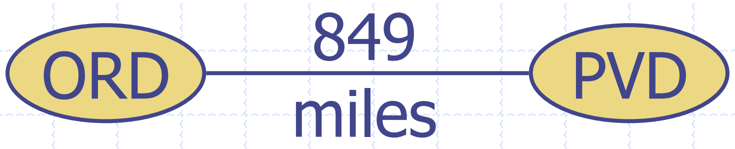
- A graph is a pair (V, E), where
  - V is a set of nodes, called vertices
  - E is a collection of pairs of vertices, called edges
  - Vertices and edges are positions and store elements
- Example:
  - A vertex represents an airport and stores the three-letter airport code
  - An edge represents a flight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



Edge Types

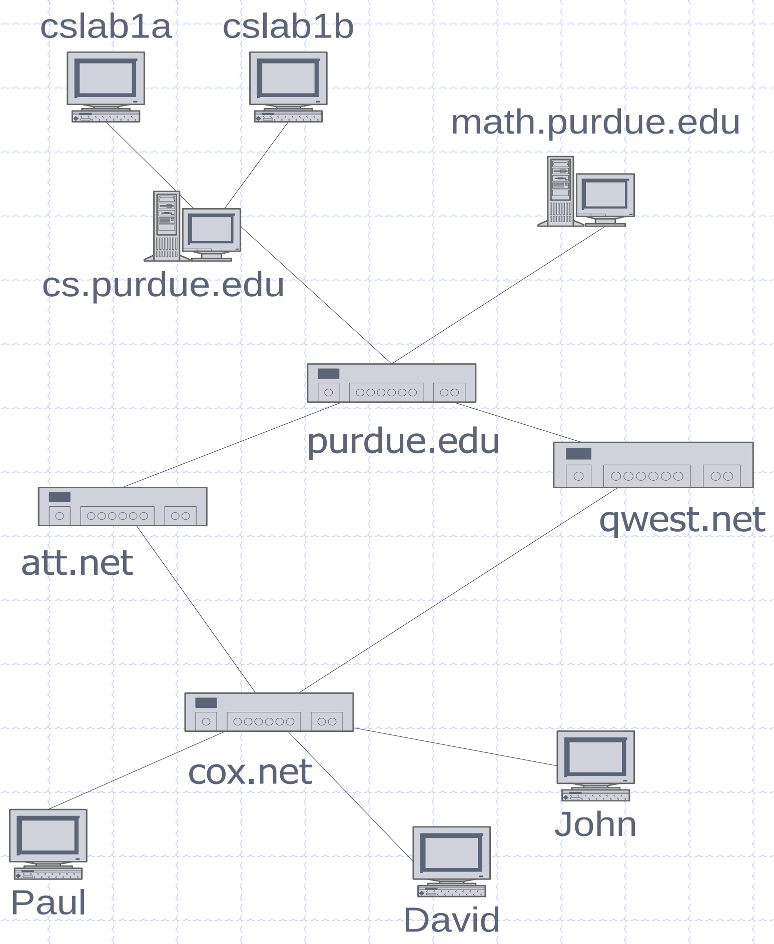
- Directed edge
  - ordered pair of vertices (u,v)
  - first vertex u is the origin
  - second vertex v is the destination
  - e.g., a flight
- Undirected edge
  - unordered pair of vertices (u,v)
  - e.g., a flight route
- Directed graph
  - all the edges are directed
  - e.g., route network
- Undirected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





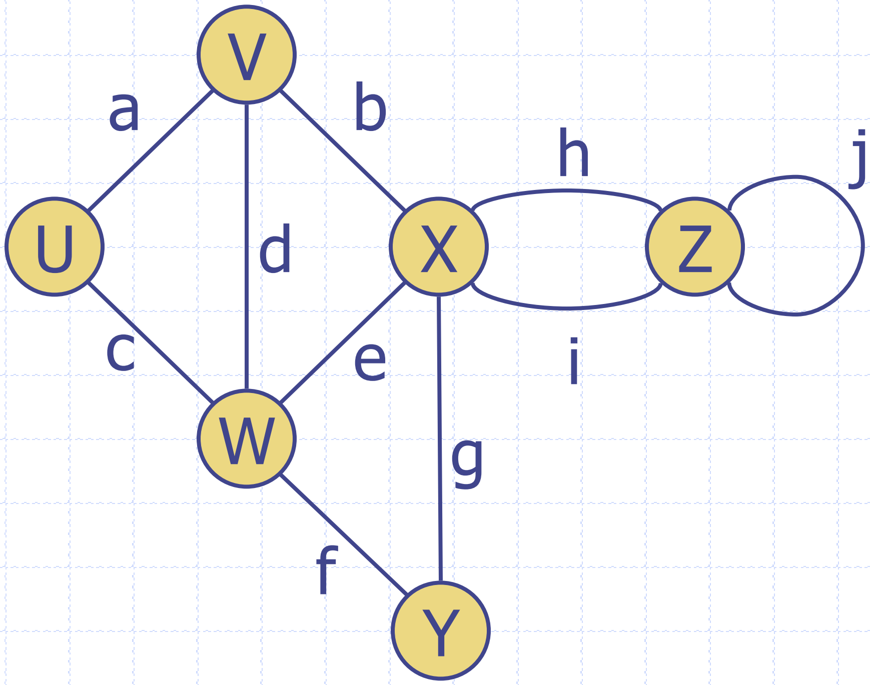
Applications

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



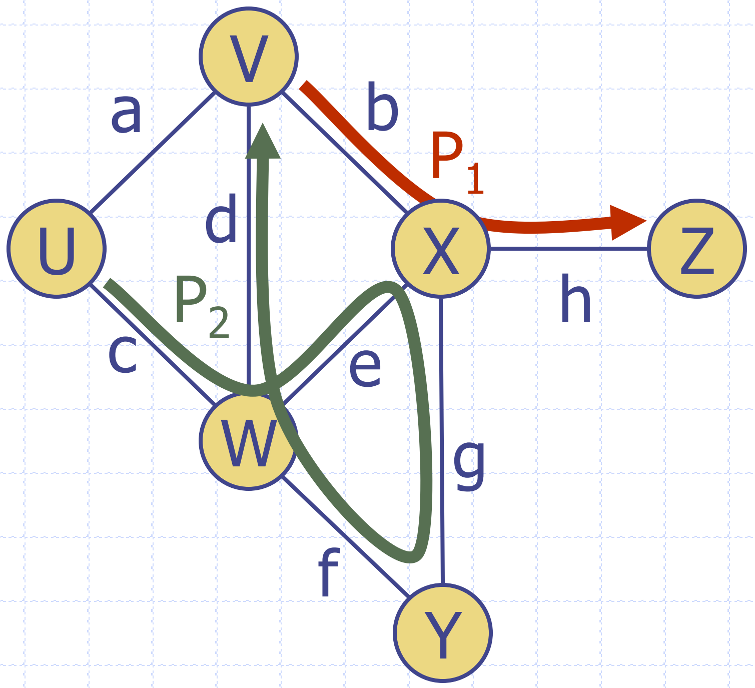
Terminology

- End vertices (or endpoints) of an edge
  - U and V are the *endpoints* of an edge
- 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*



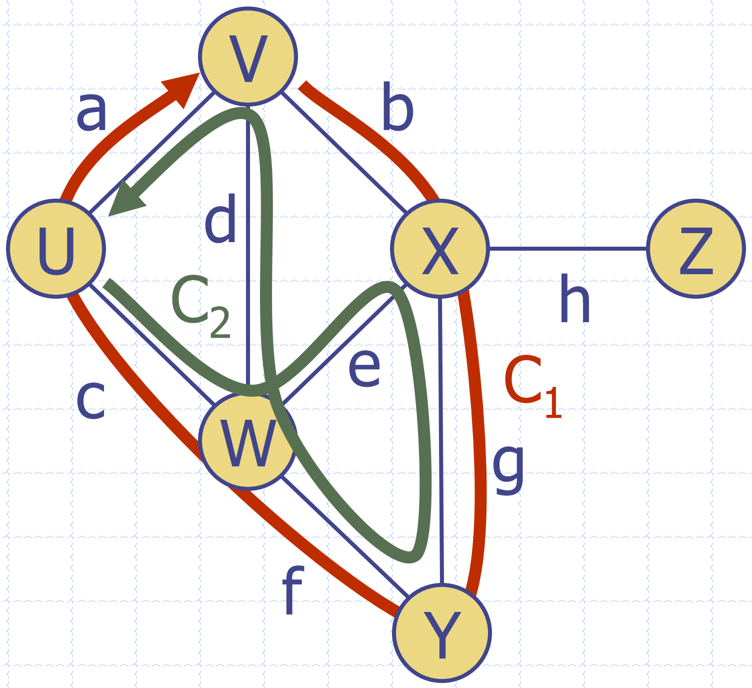
Terminology (cont.)

- 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

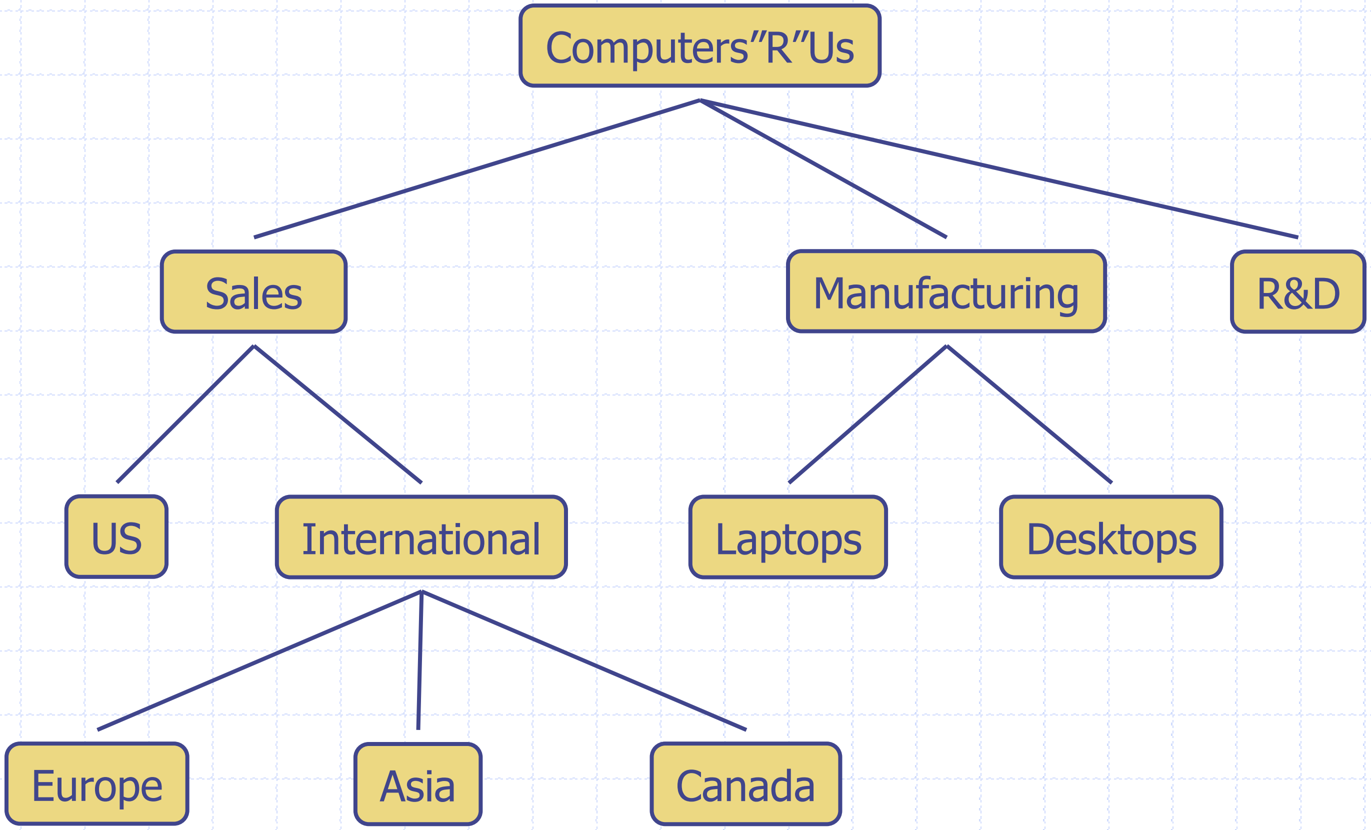


Terminology (cont.)

- 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



Trees: Revisited



Trees: Revisited

- A tree is a graph
- Using tree terminology,
  - What are the vertices of a tree called?
    - Nodes
  - What are the edges of a tree called?
    - Edges (or pointers?)
  - Is a tree "directed" or "undirected"
    - You could say:
      - **Directed:** have parent->child pointers
      - **Undirected:** have parent->child and child->parent pointers

Trees: Revisited

- Using tree terminology,
  - What is the degree of the vertices of a binary tree?
    - 3
  - What is an example path through a tree?
    - A pre/in/post-order traversal
  - What is an example cycle through a tree?
    - Trick question:
      - if you consider the tree a “directed graph”, then no cycles
      - Otherwise, a pre/in/post-order traversal yields a cycle

Properties

Property 1: Degree sum

$\Sigma_n \deg(v_n) = 2m$ **Why?**

Proof: each edge is counted twice

Property 2: Edge count bound

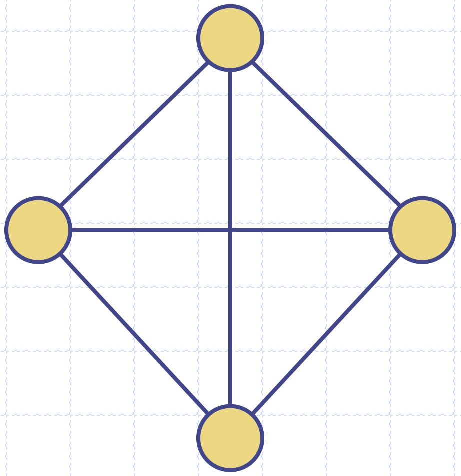
In an undirected graph with no self-loops and no multiple edges

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

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

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

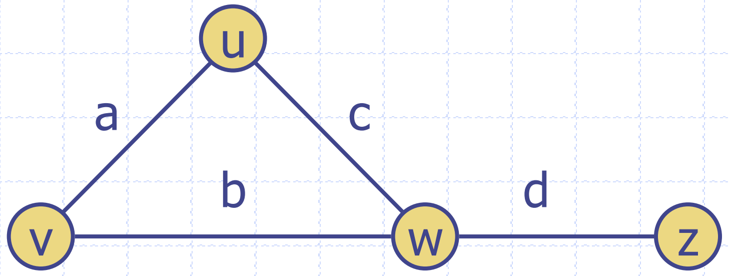
- Vertices and edges
  - are positions
  - store elements
- Accessor methods
  - aVertex()
  - incidentEdges(v)
  - endVertices(e)
  - isDirected(e)
  - origin(e)
  - destination(e)
  - opposite(v, e)
  - areAdjacent(v, w)
- Update methods
  - insertVertex(o)
  - insertEdge(v, w, o)
  - insertDirectedEdge(v, w, o)
  - removeVertex(v)
  - removeEdge(e)
- Generic methods
  - numVertices()
  - numEdges()
  - vertices()
  - edges()

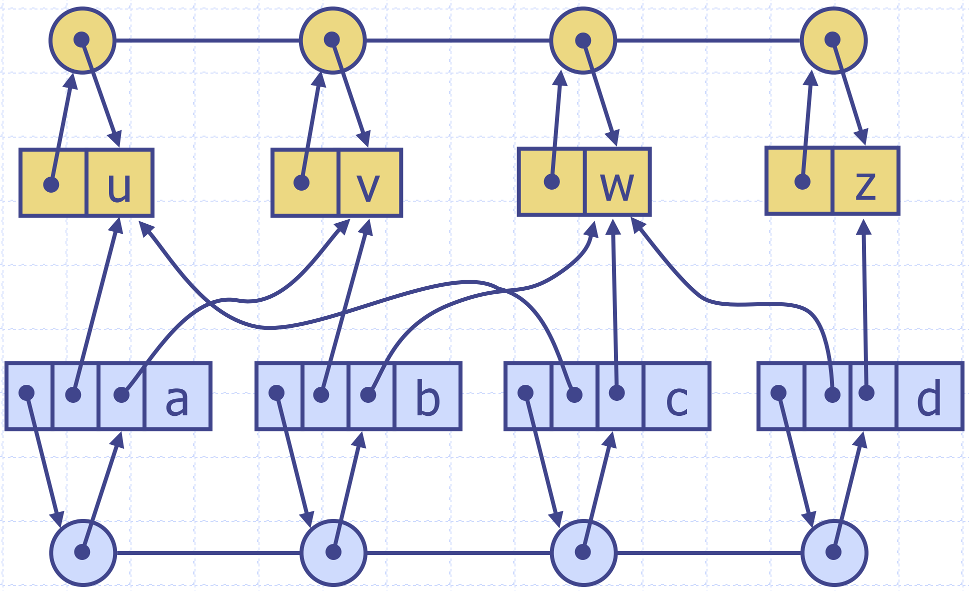
Data Representation

- How do you store a graph’s vertices, edges and connectivity?

Edge List Structure

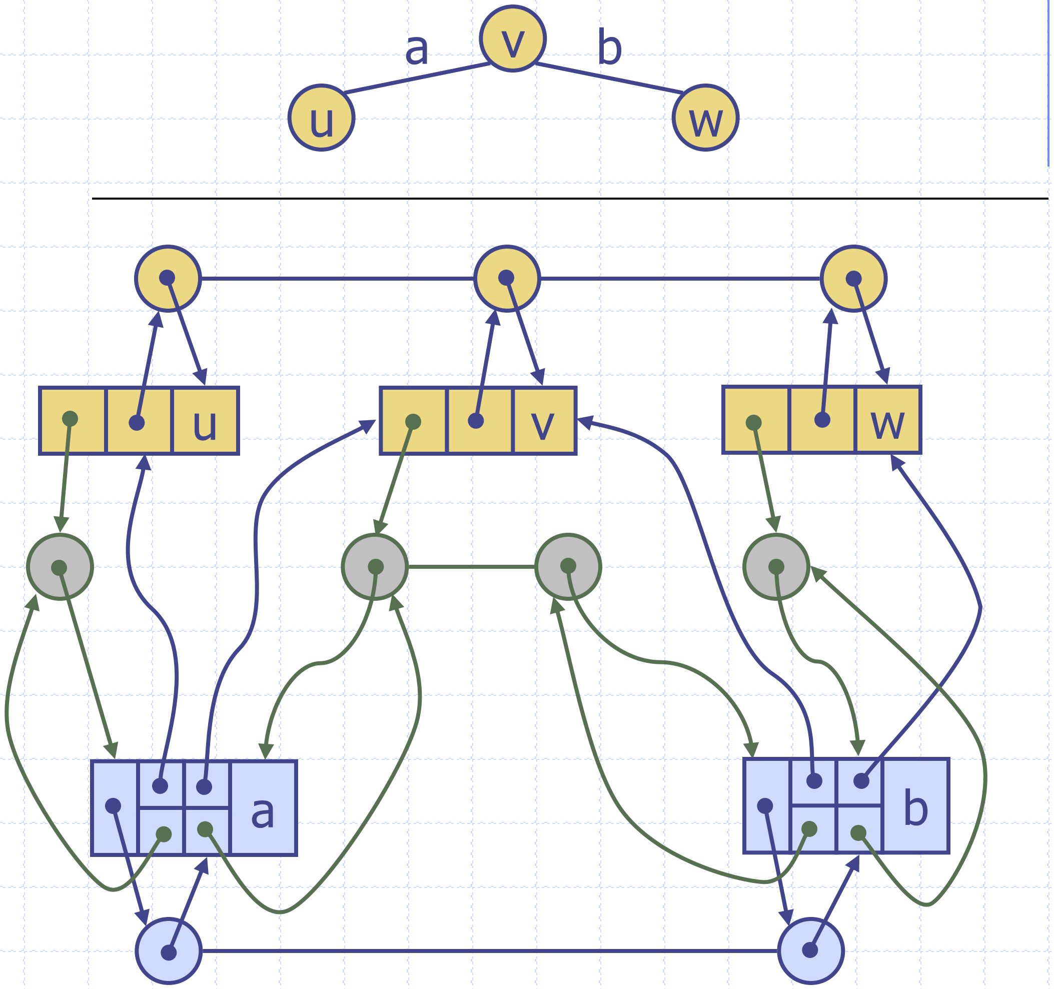
- Vertex object
  - element
  - reference to position in vertex sequence
- Edge object
  - element
  - origin vertex object
  - destination vertex object
  - reference to position in edge sequence
- Vertex sequence
  - sequence of vertex objects
- Edge sequence
  - sequence of edge objects





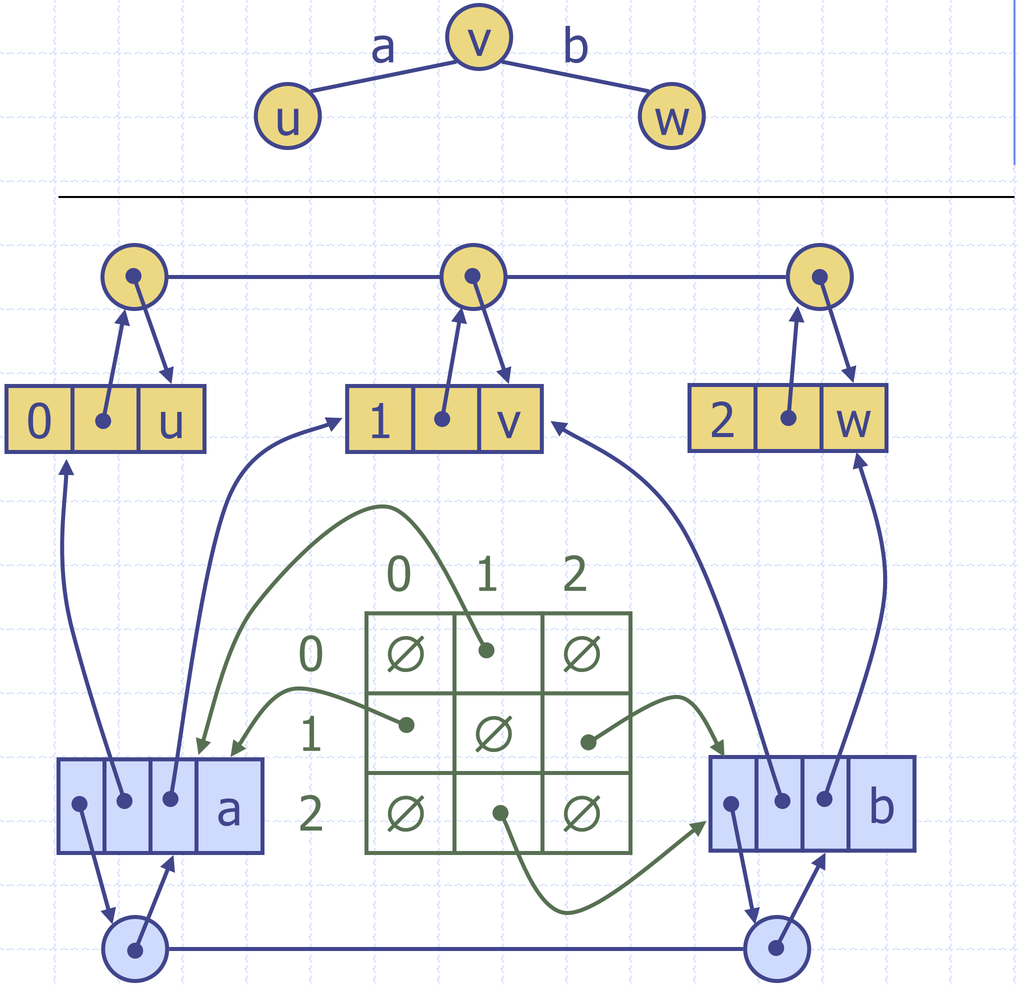
Or, Adjacency List Structure

- 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



Or, 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



Asymptotic Performance

◆ n vertices, m edges ◆ no parallel edges ◆ no self-loops ◆ Bounds are "big-Oh"	Edge List	
Space	$n + m$	
incidentEdges(v)	m	
areAdjacent (v, w)	m	
insertVertex(o)	1	
insertEdge(v, w, o)	1	
removeVertex(v)	m	
removeEdge(e)	1	

Asymptotic Performance

◆ n vertices, m edges ◆ no parallel edges ◆ no self-loops ◆ Bounds are "big-Oh"	Edge List	Adjacency List
Space	$n + m$	$n + m$
incidentEdges(v)	m	$\deg(v)$
areAdjacent (v, w)	m	$\min(\deg(v), \deg(w))$
insertVertex(o)	1	1
insertEdge(v, w, o)	1	1
removeVertex(v)	m	$\deg(v)$
removeEdge(e)	1	1

Asymptotic Performance

◆ n vertices, m edges ◆ no parallel edges ◆ no self-loops ◆ Bounds are “big-Oh”	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