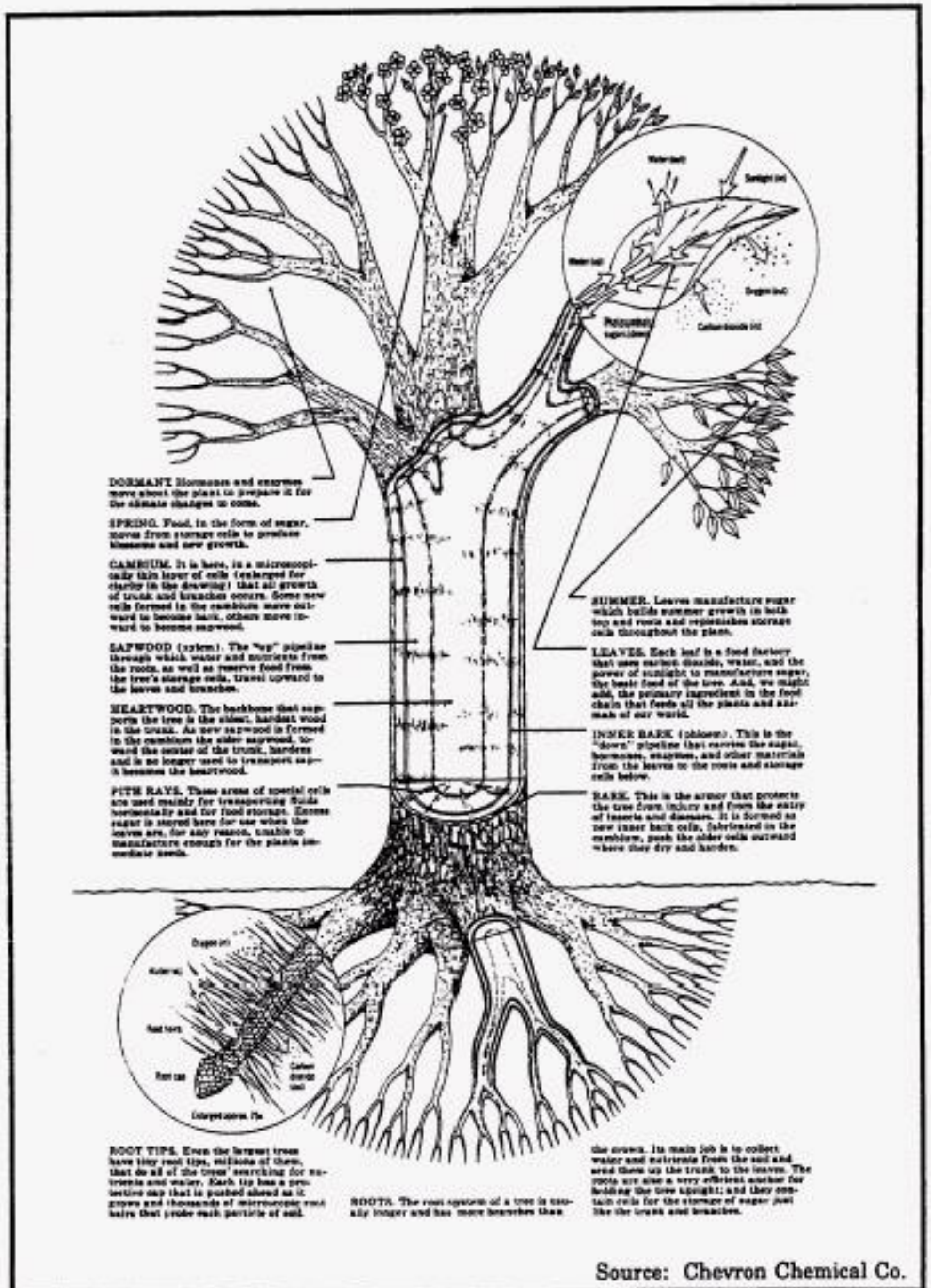
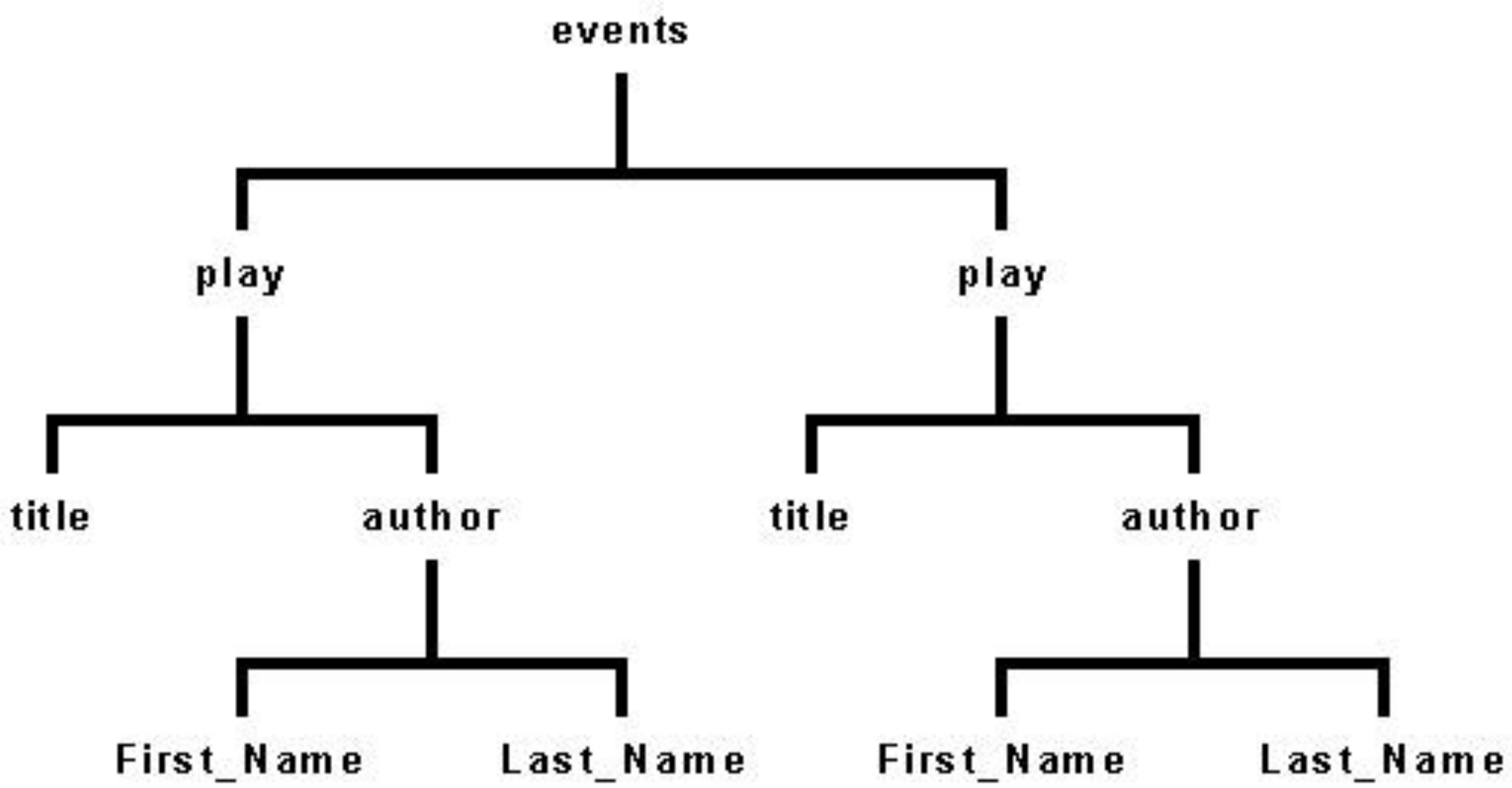
Tree



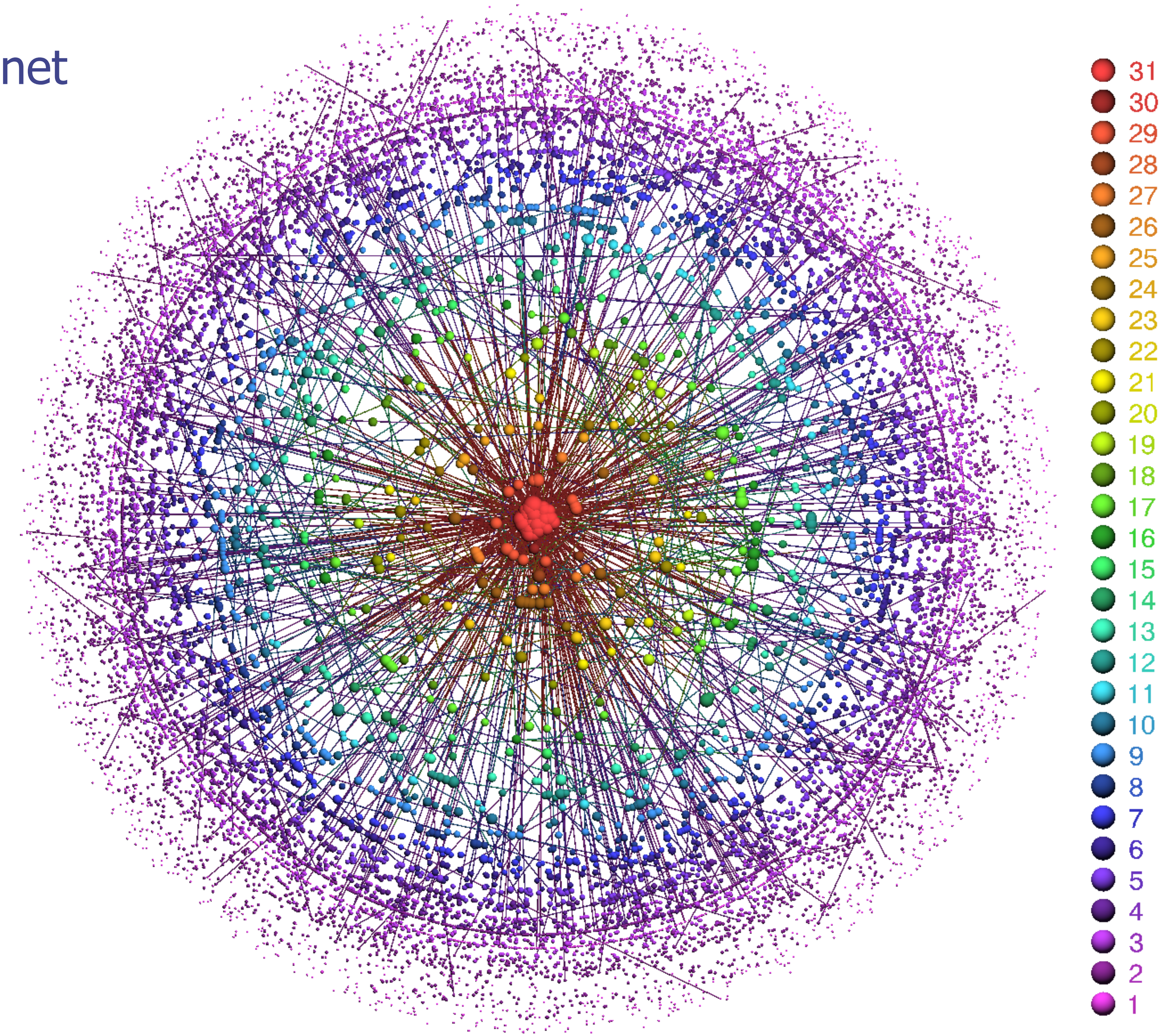
DORMANT.
SPRING.
CAMBIUM.
SAPWOOD (xylem).
HEARTWOOD.
PITH RAYS.
SUMMER.
LEAVES.
INNER BARK (phloem).
BARK.
ROOT TIPS.
ROOTS.
Source: Chevron Chemical Co.

Tree

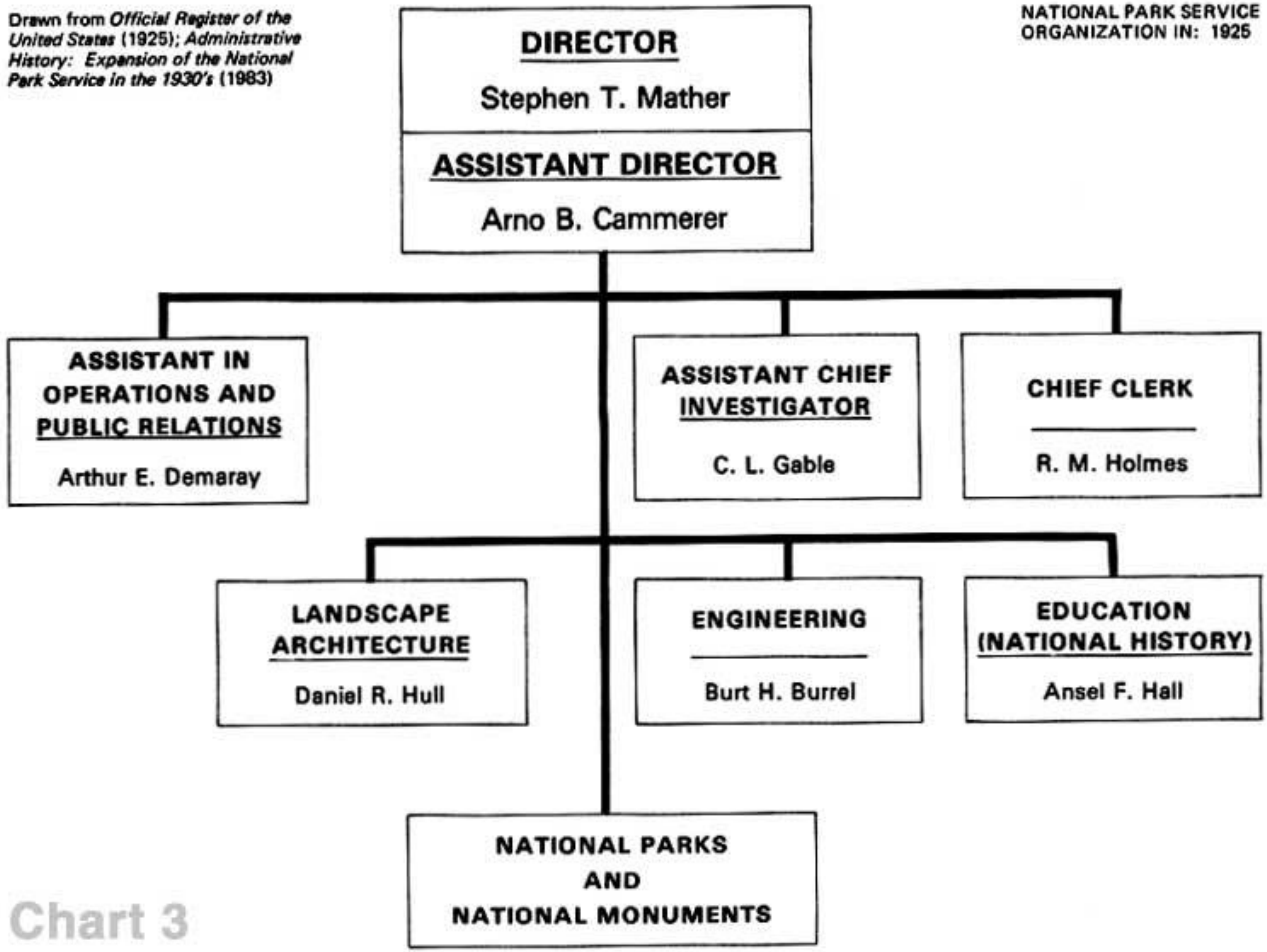


events
play
play
title
author
title
author
First_Name
Last_Name
First_Name
Last_Name

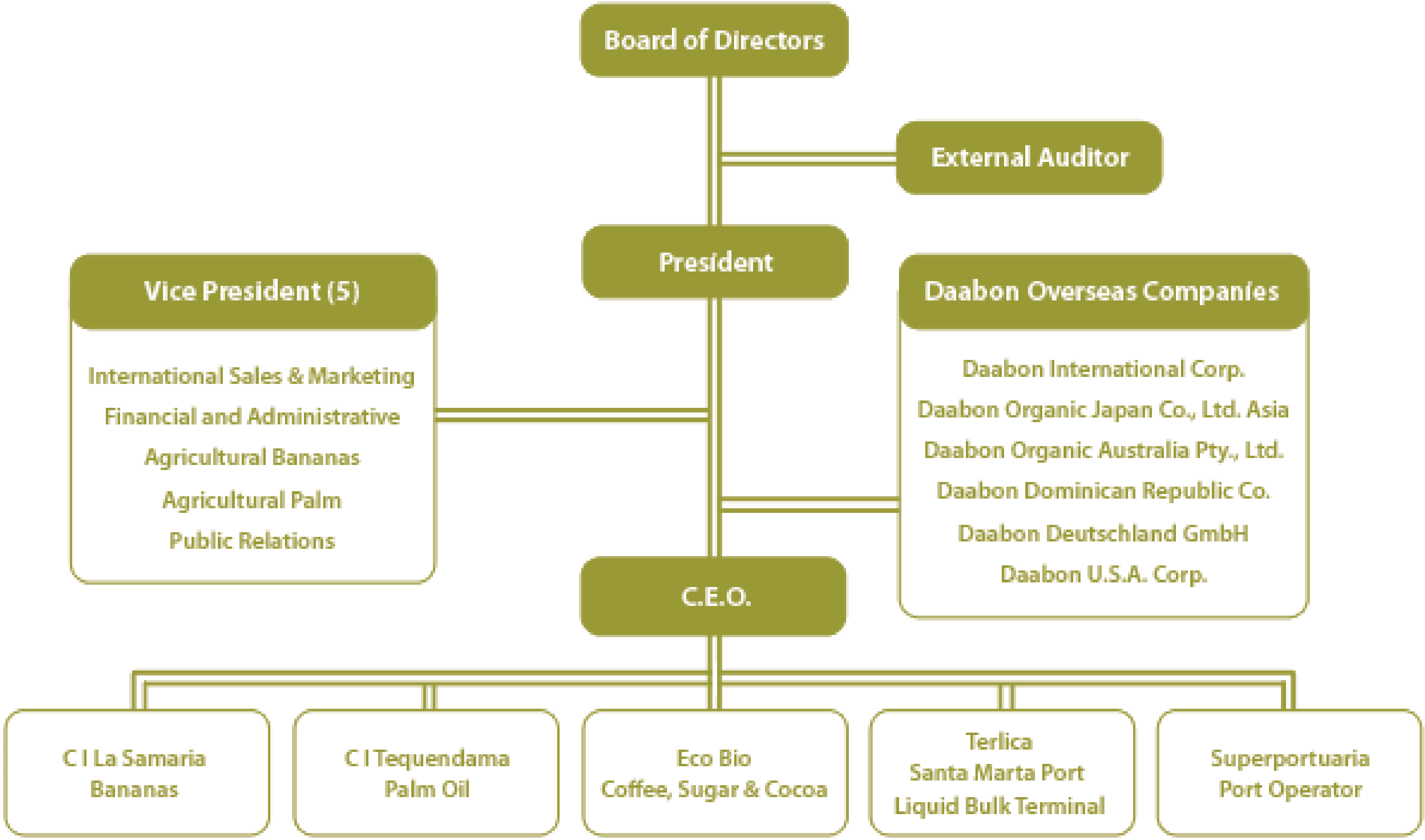
Internet


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

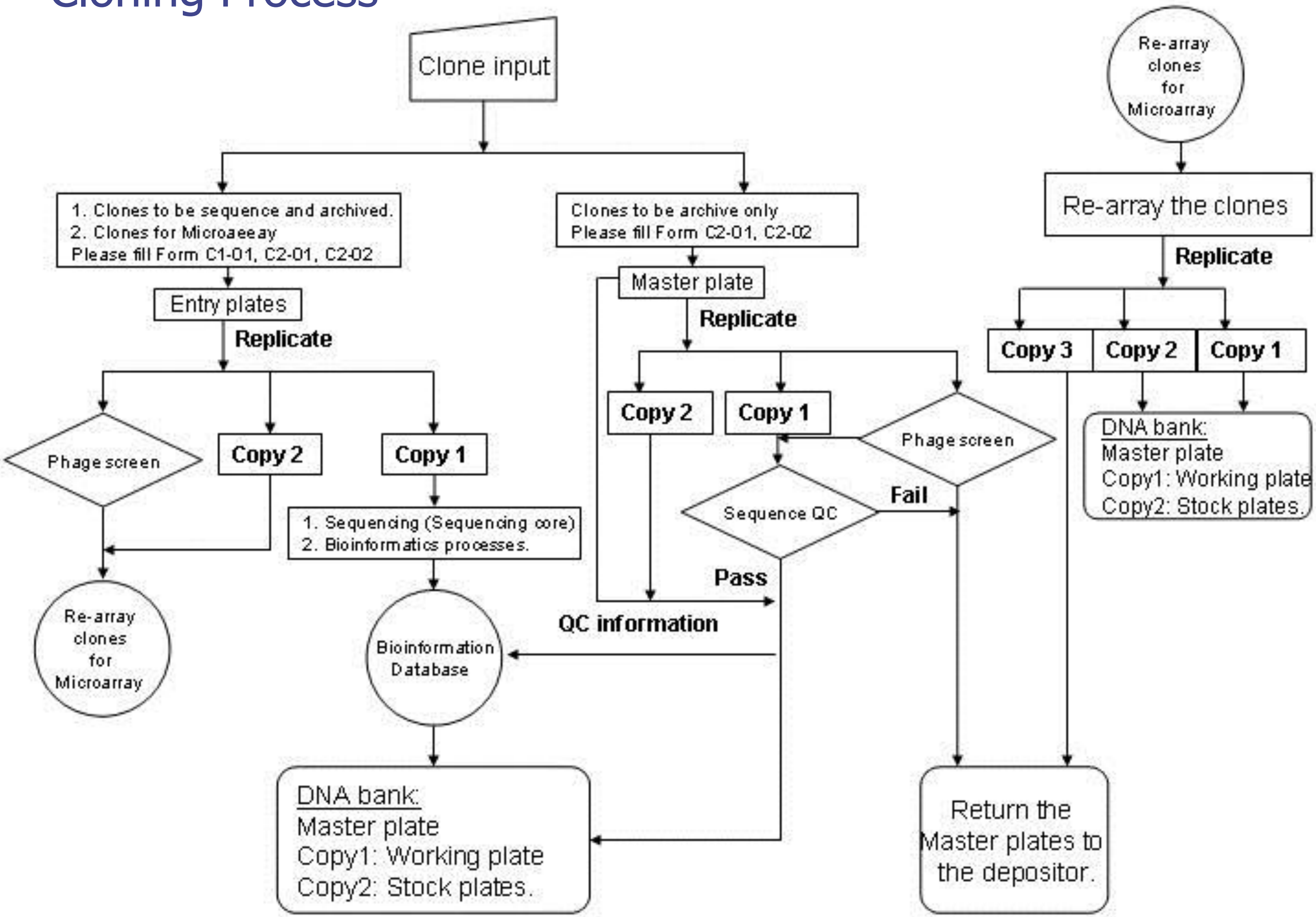
Organizational chart



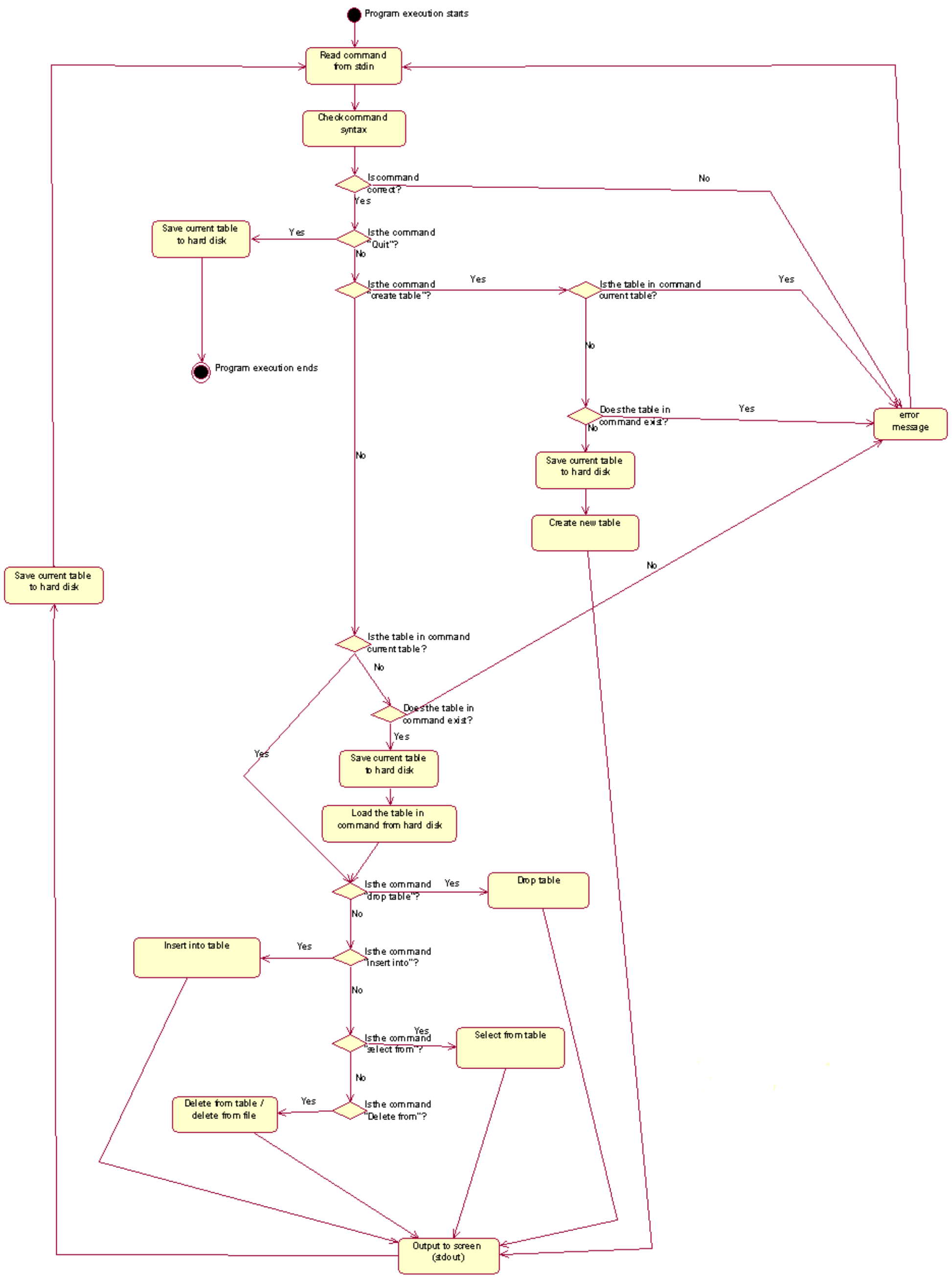
Organizational Chart



Cloning Process

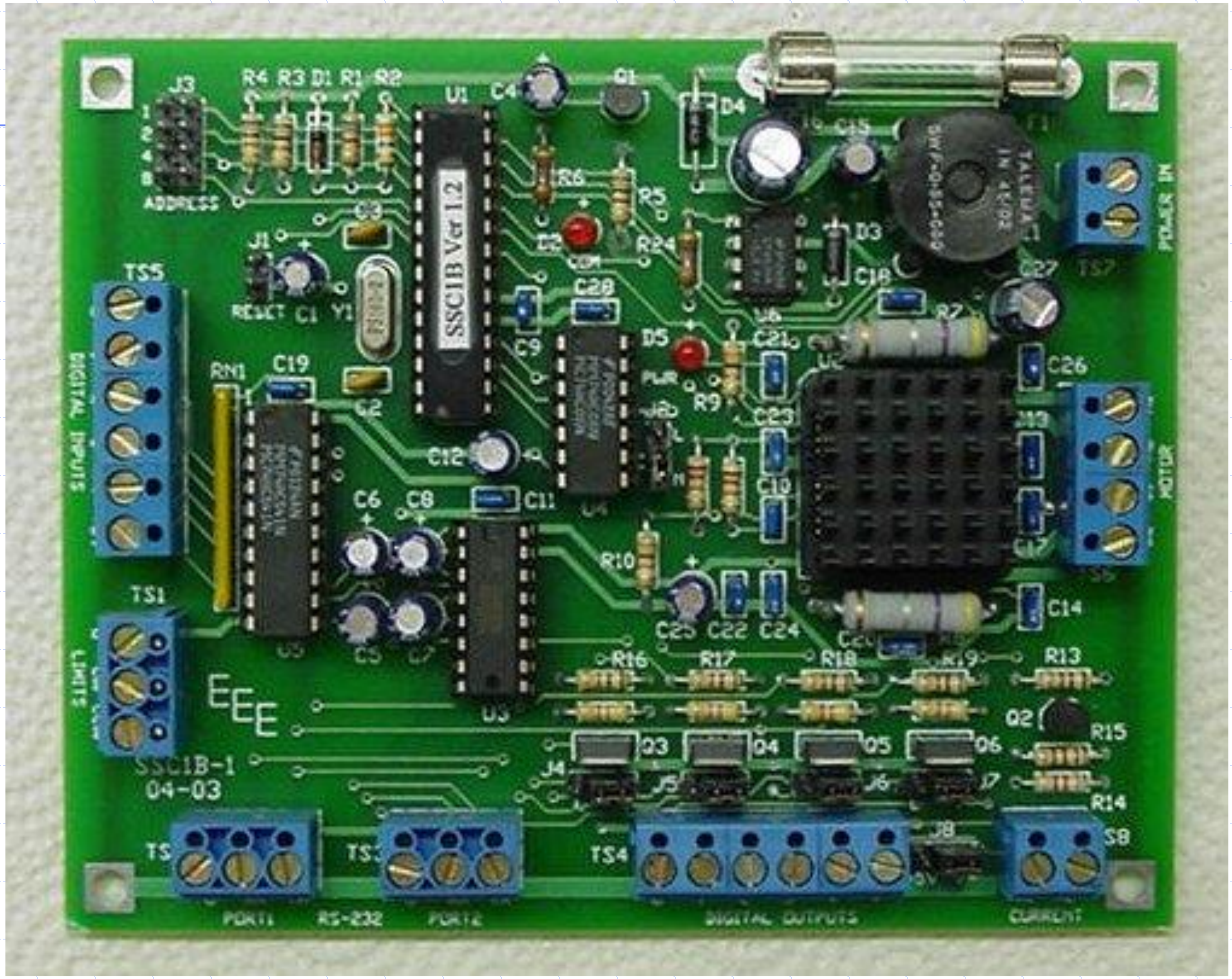


Program Flow Chart

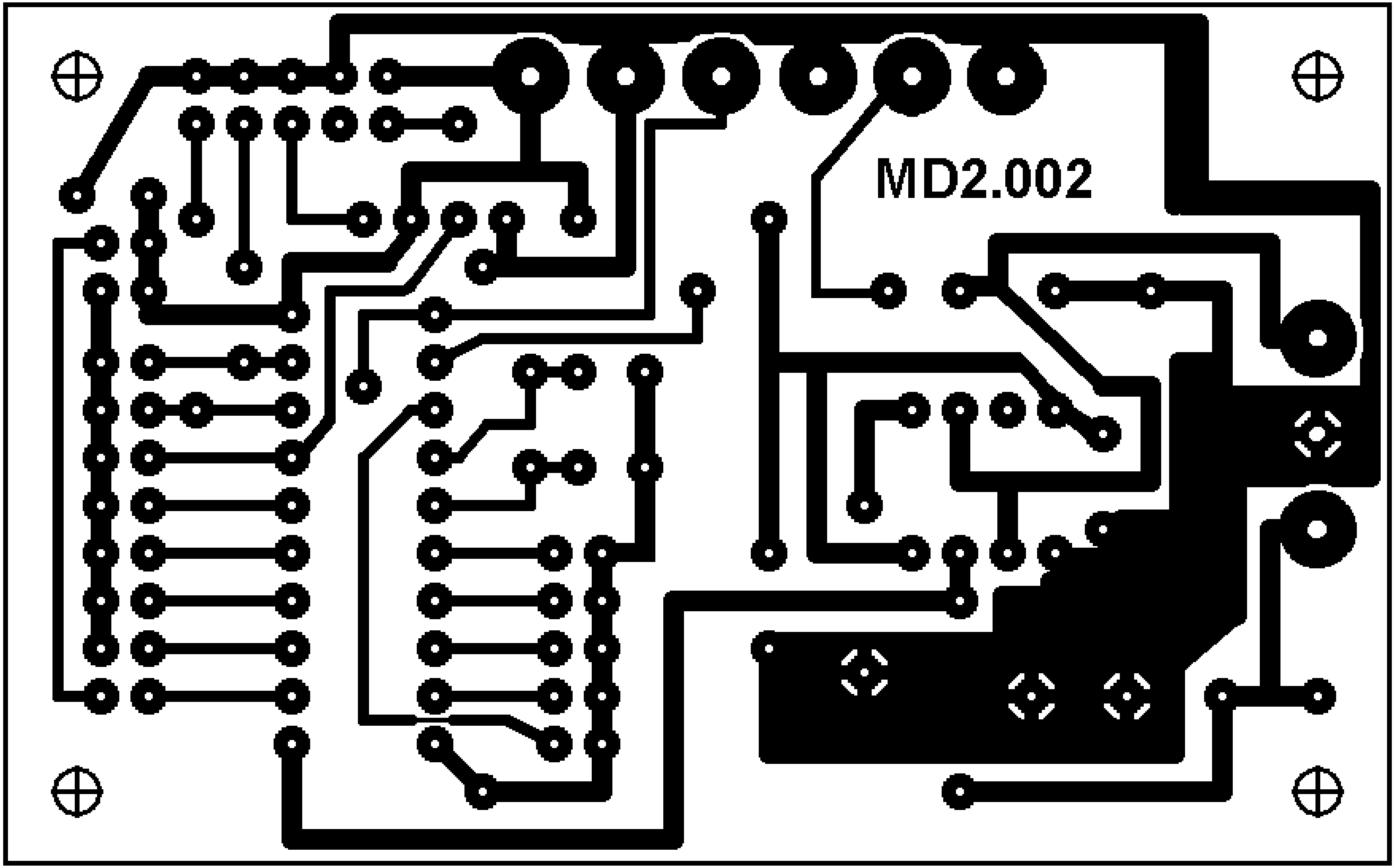


Flight Routes

Electronic Circuit Board



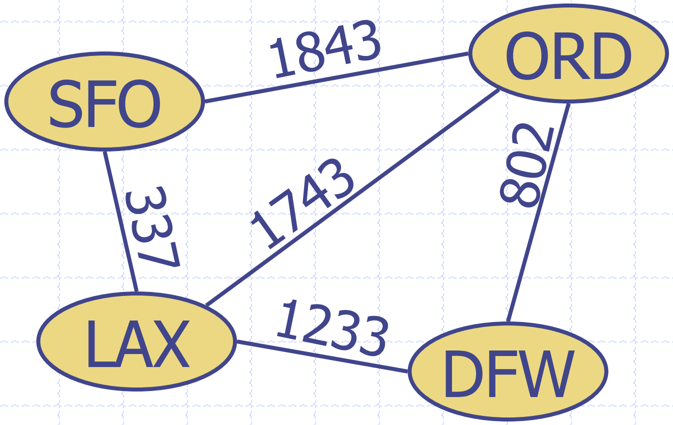
Electronic Circuit Board



What do these have in common?

- They are graphs…

Graphs

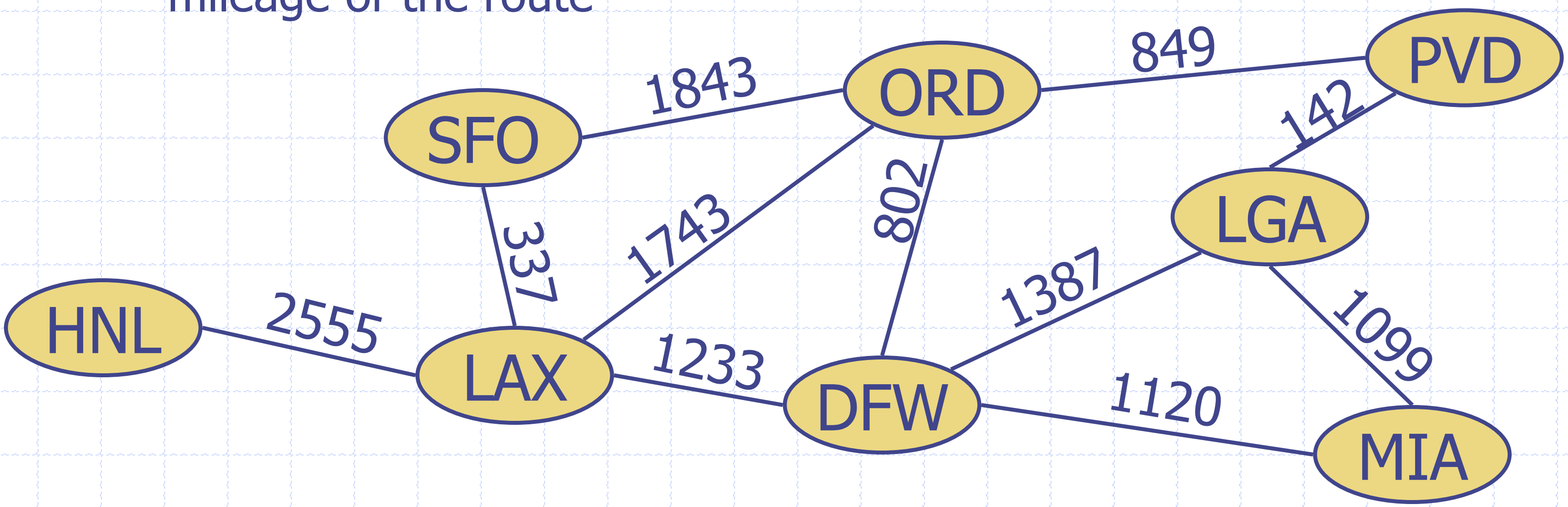


Outline and Reading

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

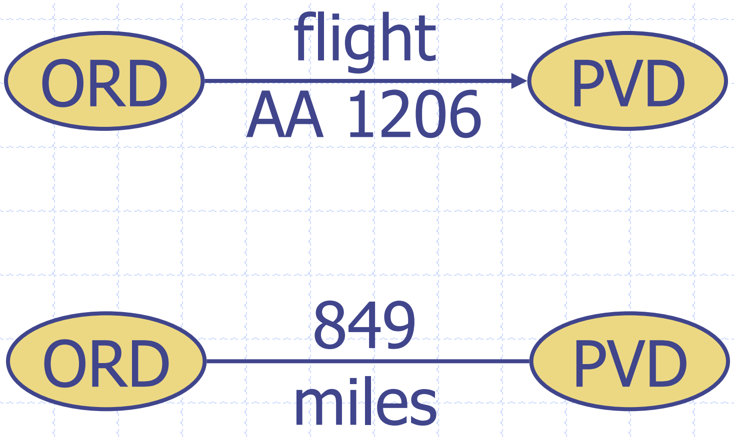
Graph

- 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



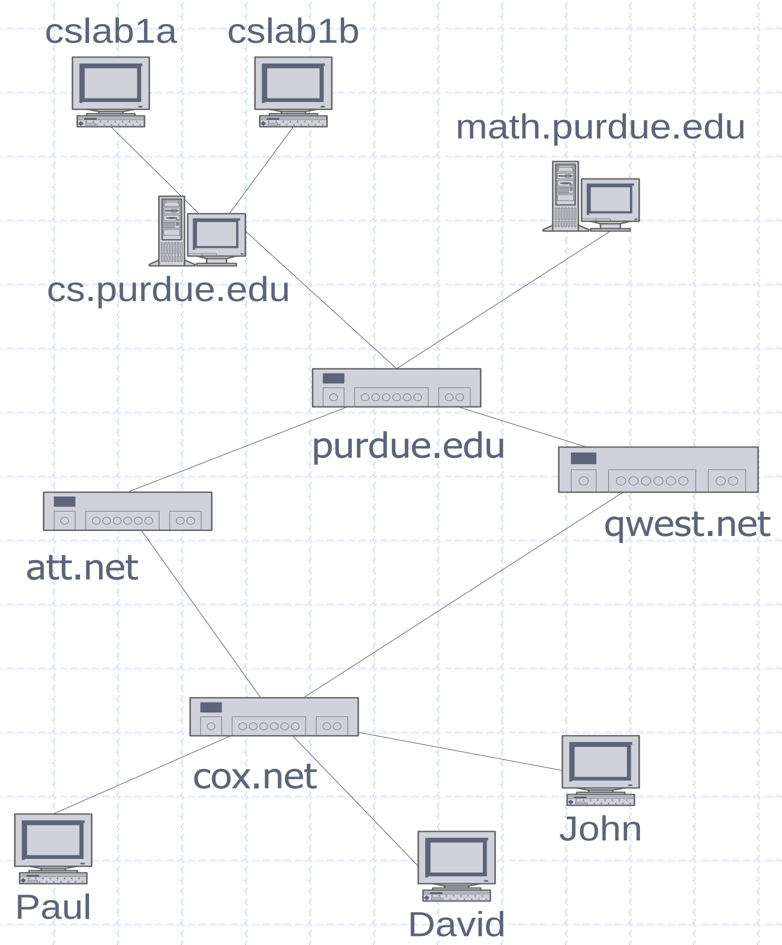
Edge Types

- 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



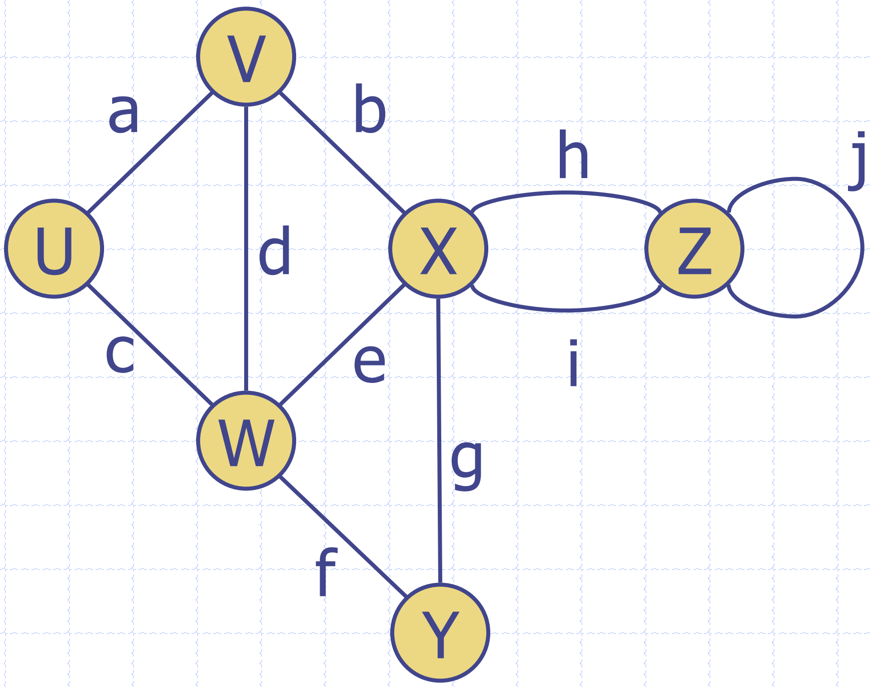
Applications

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



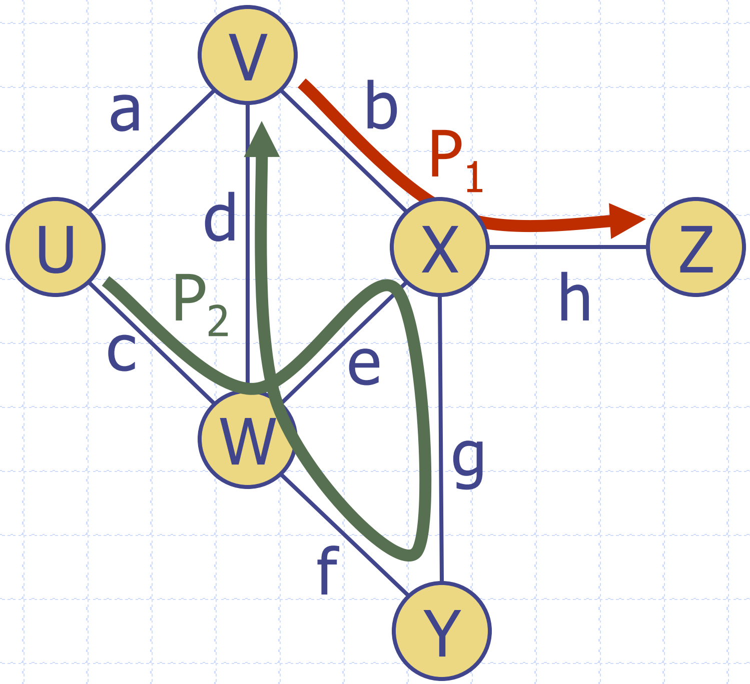
Terminology

- 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*



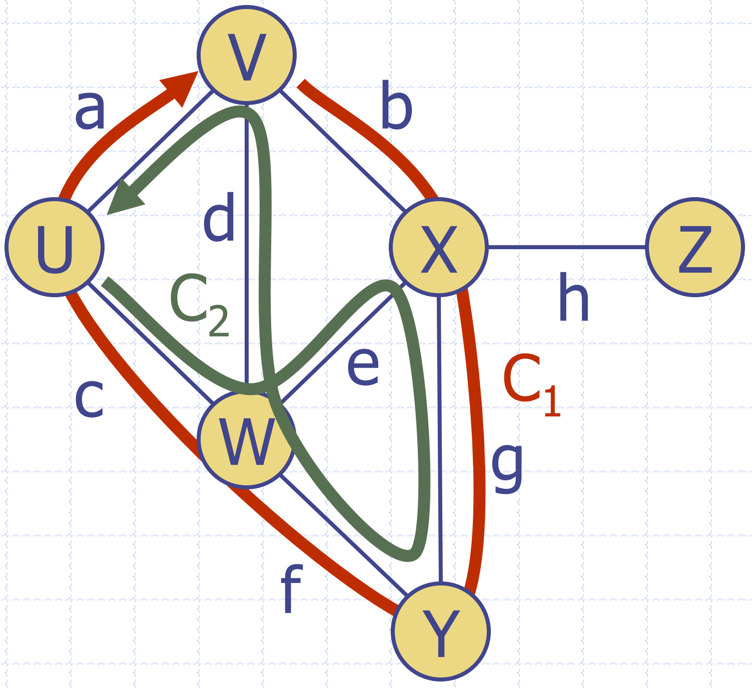
Terminology (cont.)

- 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

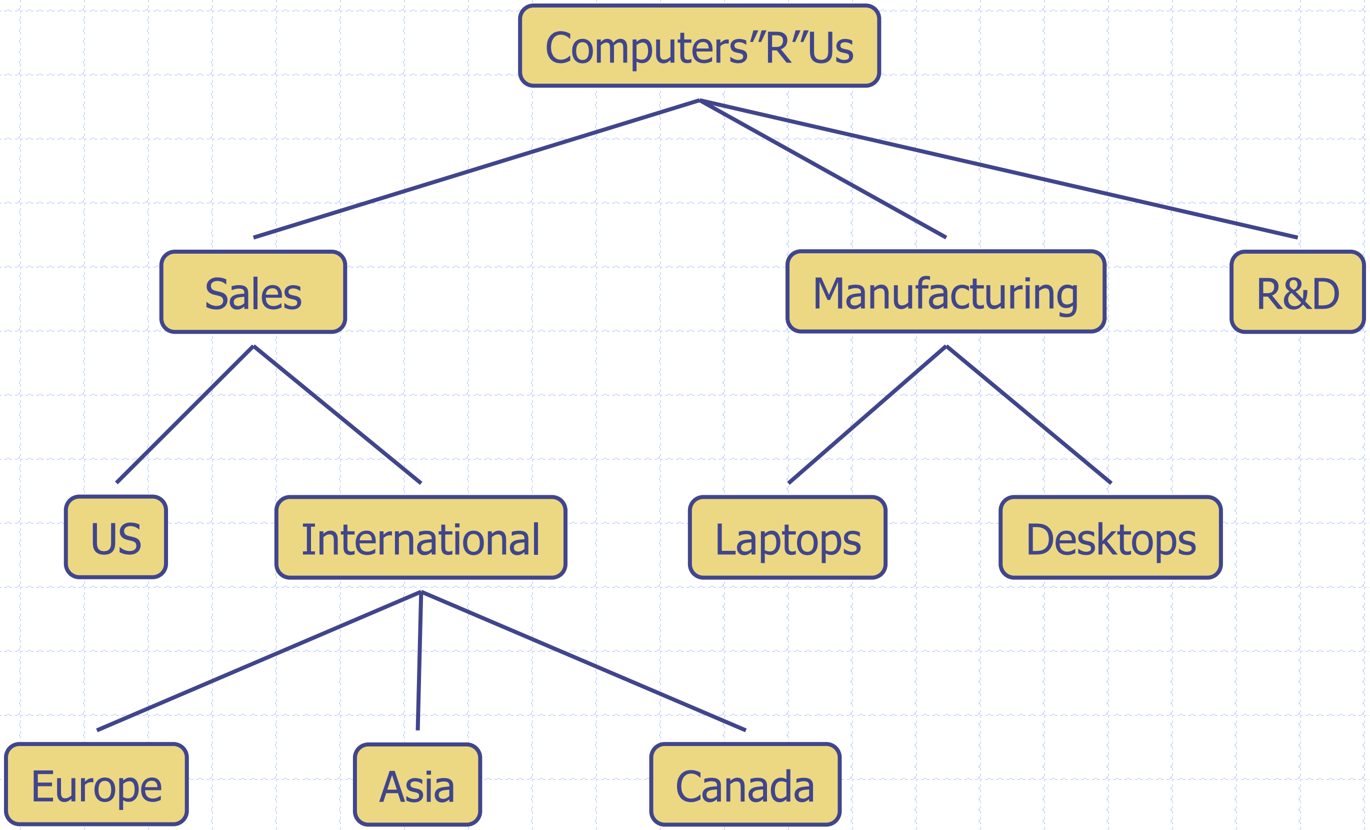


Terminology (cont.)

- 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
  - $C_1=(V,b,X,g,Y,f,W,c,U,a,\hookleftarrow)$ is a simple cycle
  - $C_2=(U,c,W,e,X,g,Y,f,W,d,V,a,\hookleftarrow)$ 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

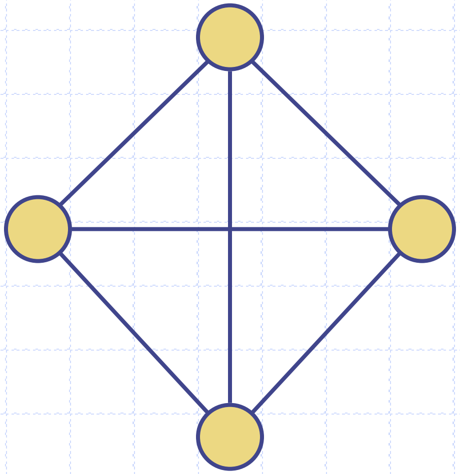
In an undirected graph with no self-loops and no multiple edges

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

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

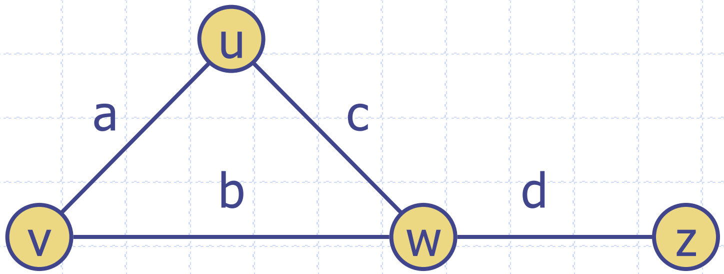
- Vertices and edges
  - are positions
  - store elements
- Accessor methods
  - aVertex()
  - incidentEdges(v)
  - endVertices(e)
  - isDirected(e)
  - origin(e)
  - destination(e)
  - opposite(v, e)
  - areAdjacent(v, w)
- Update methods
  - insertVertex(o)
  - insertEdge(v, w, o)
  - insertDirectedEdge(v,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

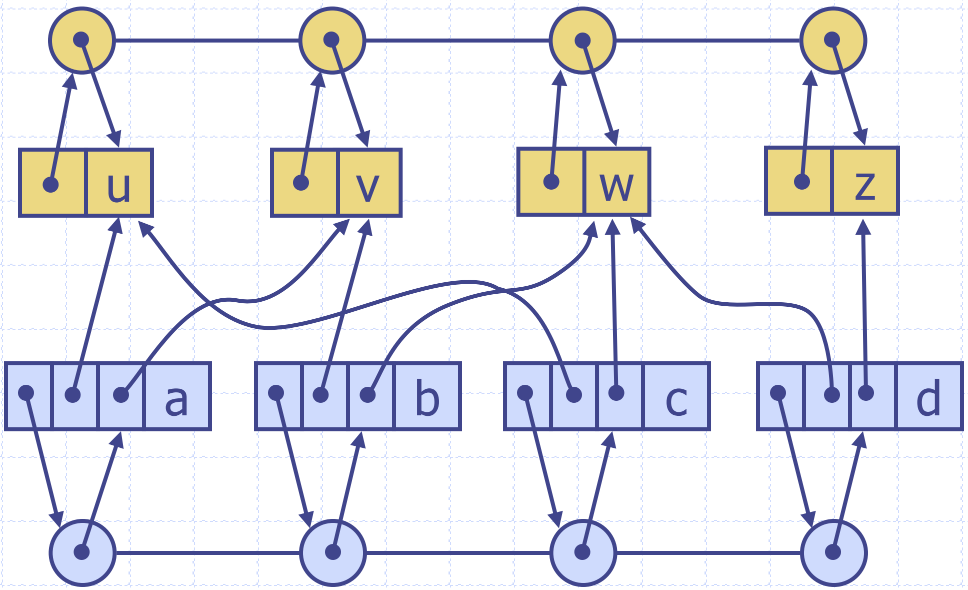
Data Representation

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

Edge List Structure

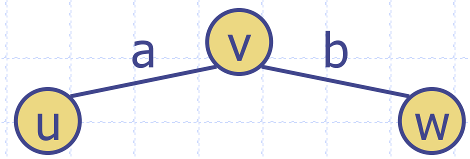
- Vertex object
  - element
  - reference to position in vertex sequence
- Edge object
  - element
  - origin vertex object
  - destination vertex object
  - reference to position in edge sequence
- Vertex sequence
  - sequence of vertex objects
- Edge sequence
  - sequence of edge objects

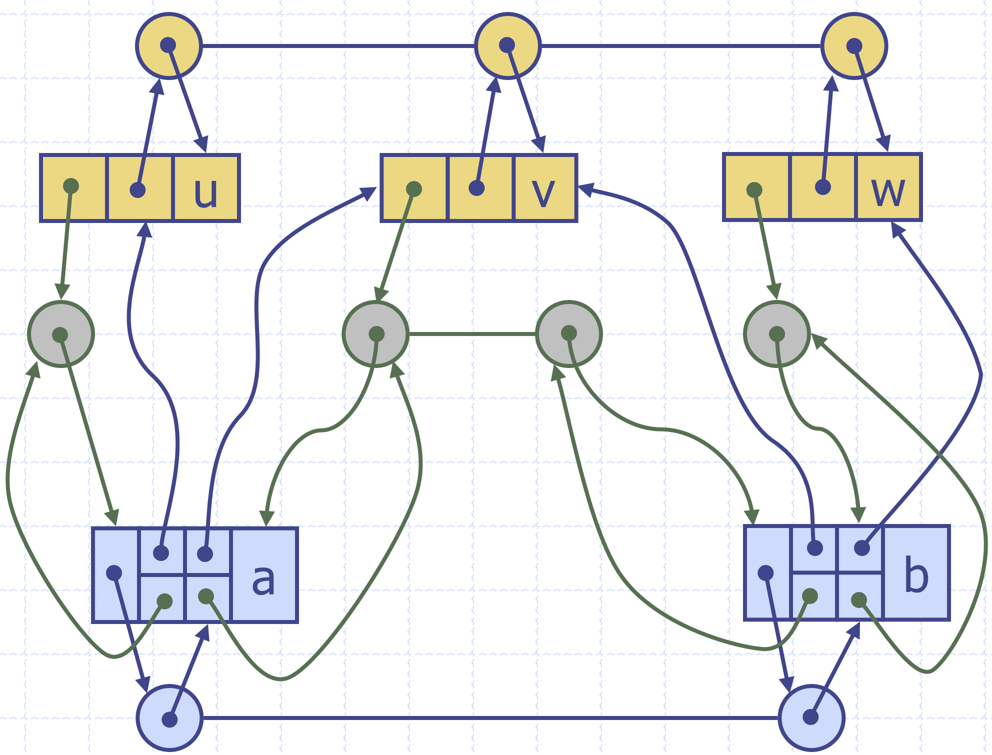




Or, Adjacency List Structure

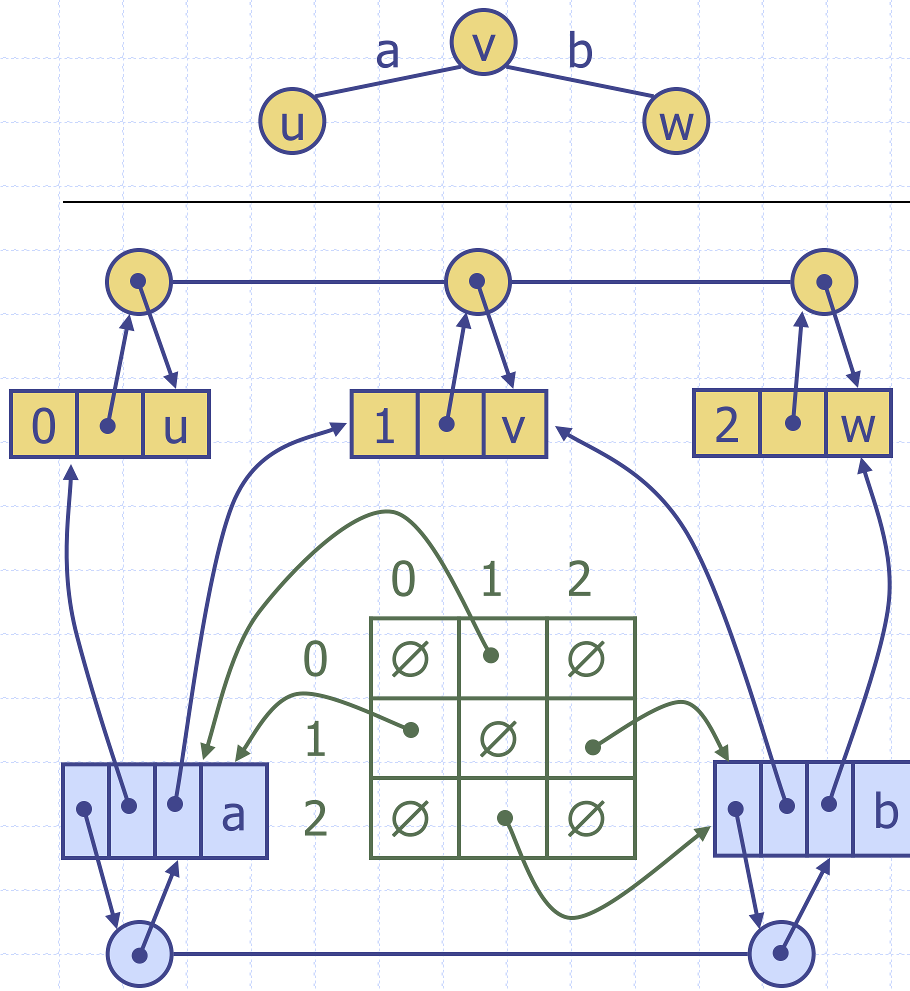
- Edge list structure
- Incidence sequence for each vertex
  - sequence of references to edge objects of incident edges
- Augmented edge objects
  - references to associated positions in incidence sequences of end vertices





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