CPSC 131
Data Structures Concepts

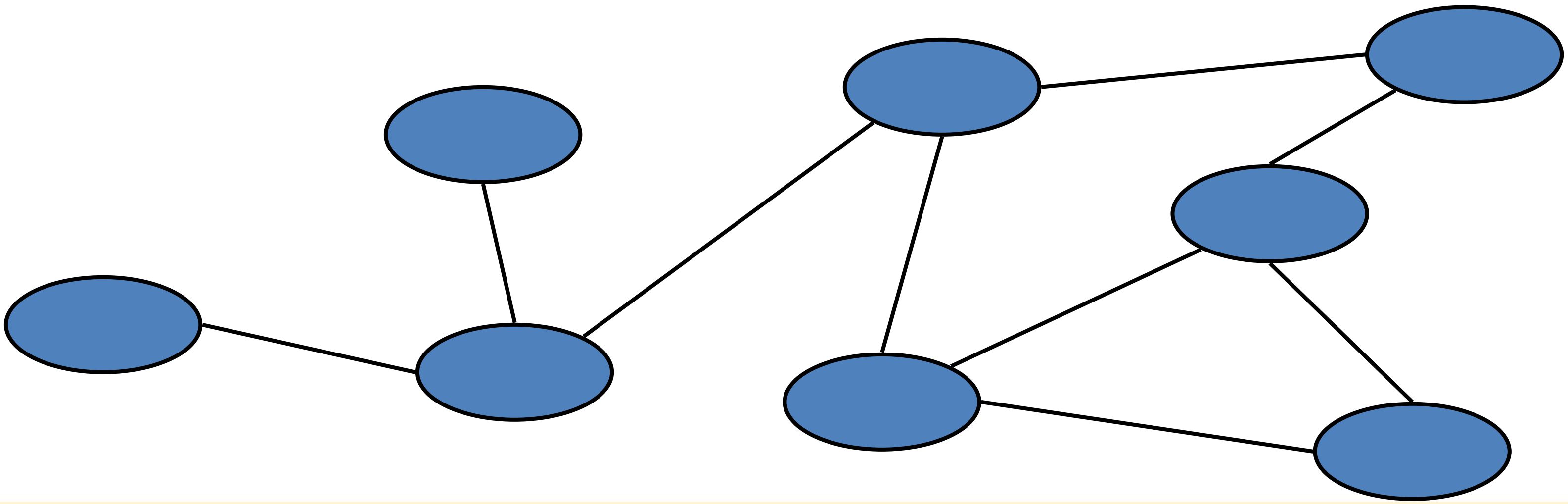
Dr. Anand Panangadan
apanangadan@fullerton.edu

Goals

- Graphs
  - Terminology
  - Applications
  - Abstract Data Structure
  - Two possible implementations
    1. Adjacency list
    2. Adjacency matrix

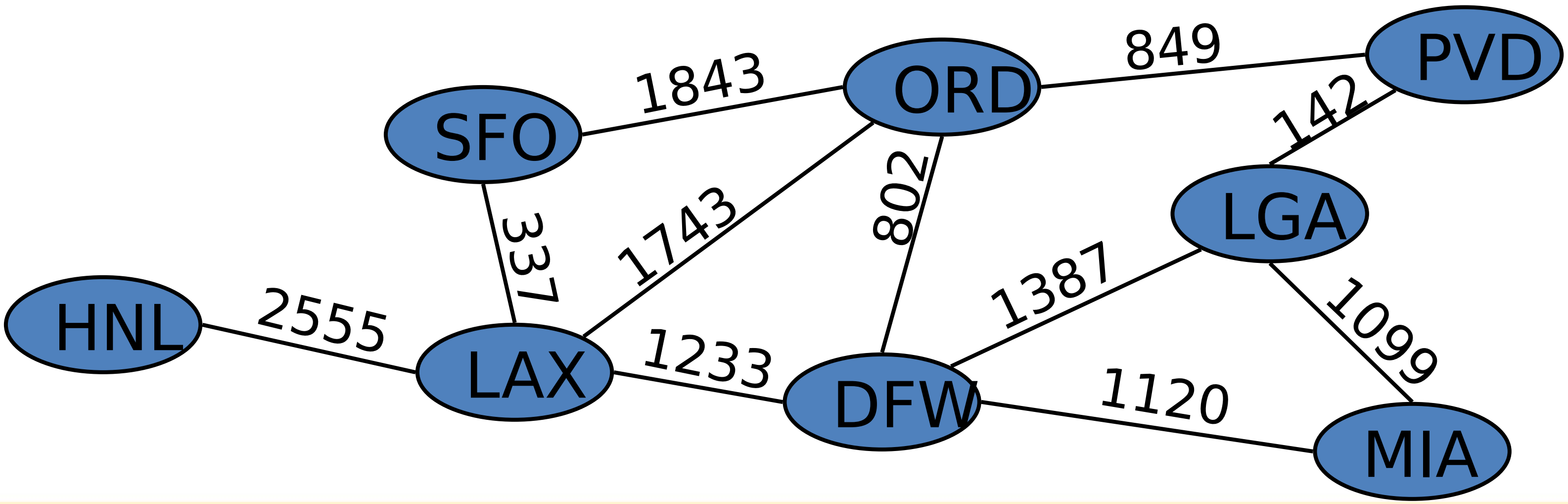
Graphs

- 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



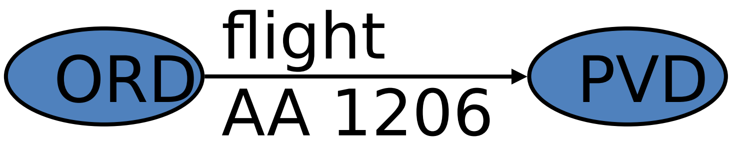
Graphs

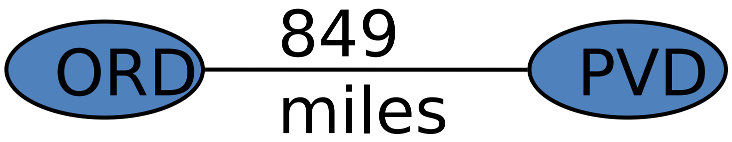
- Example:
  - A vertex: an airport (three-letter airport code)
  - An edge: a flight route between two airports (mileage of the route)



Edge Types

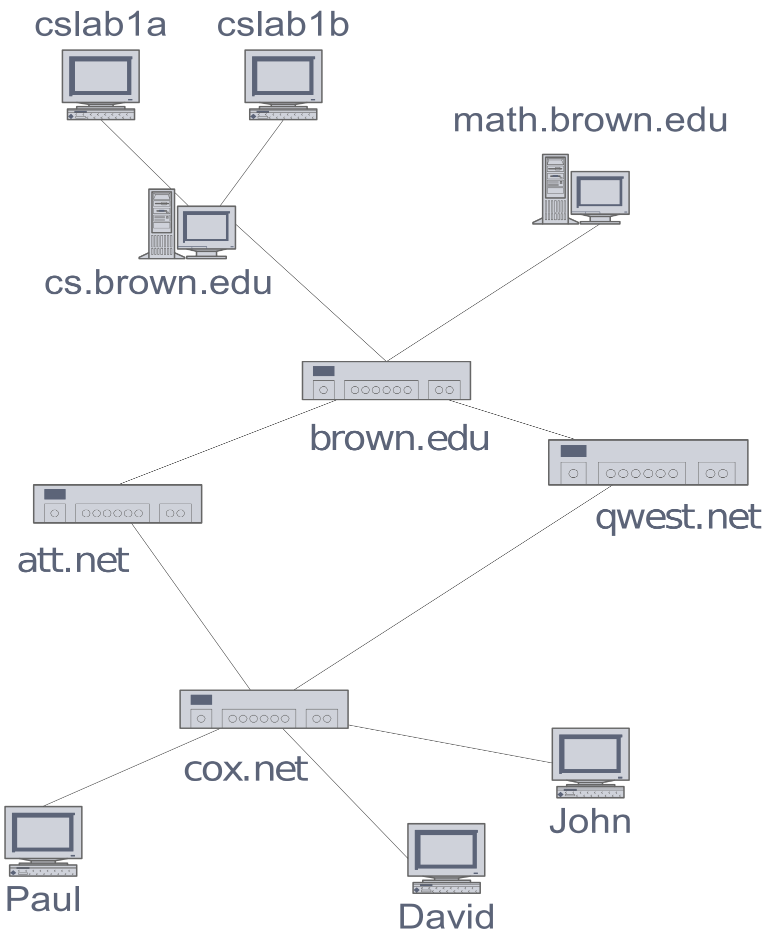
- Directed edge
  - ordered pair of vertices (u,v)
  - first vertex u is the origin
  - second vertex v is the destination
  - e.g., a flight
- Undirected edge
  - unordered pair of vertices (u,v)
  - e.g., a flight route
- Directed graph
  - all the edges are directed
  - e.g., route network
- Undirected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



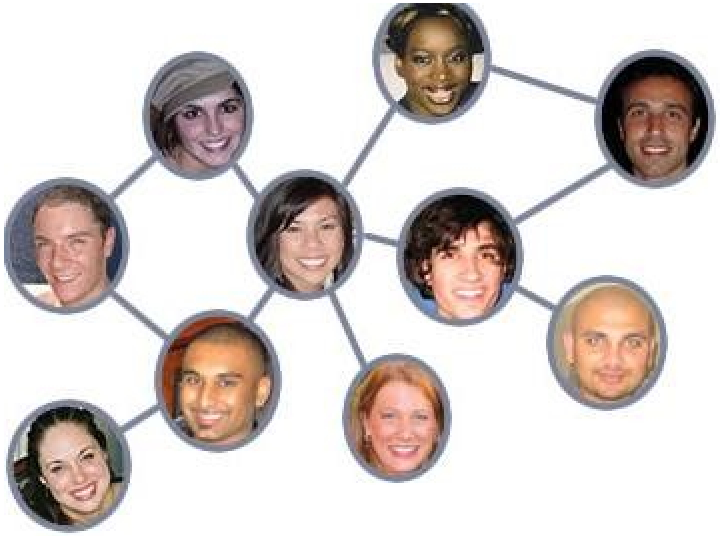


Applications

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

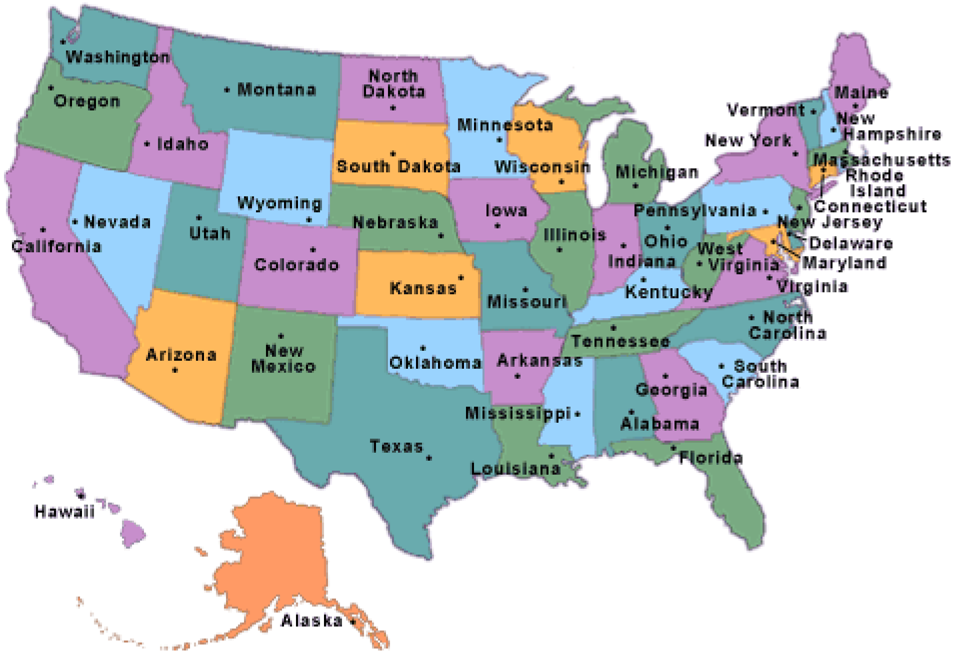


Social network



What are:
- Vertices
- Edges
- Vertex element/label
- Edge element/label

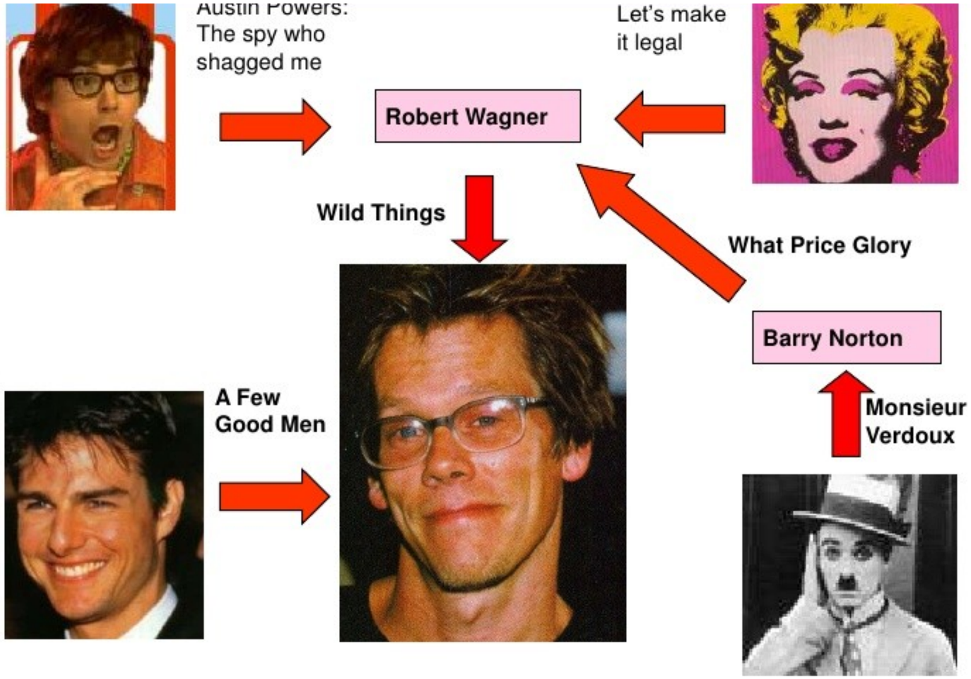
State map



What are:

- Vertices
- Edges
- Vertex element/label
- Edge element/label

Actor collaboration network

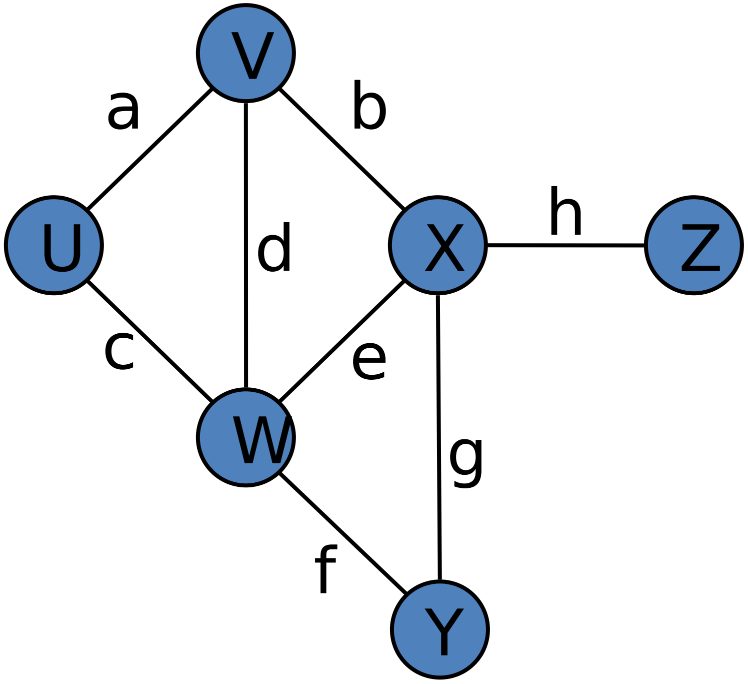


What are:

- Vertices
- Edges
- Vertex element/label
- Edge element/label

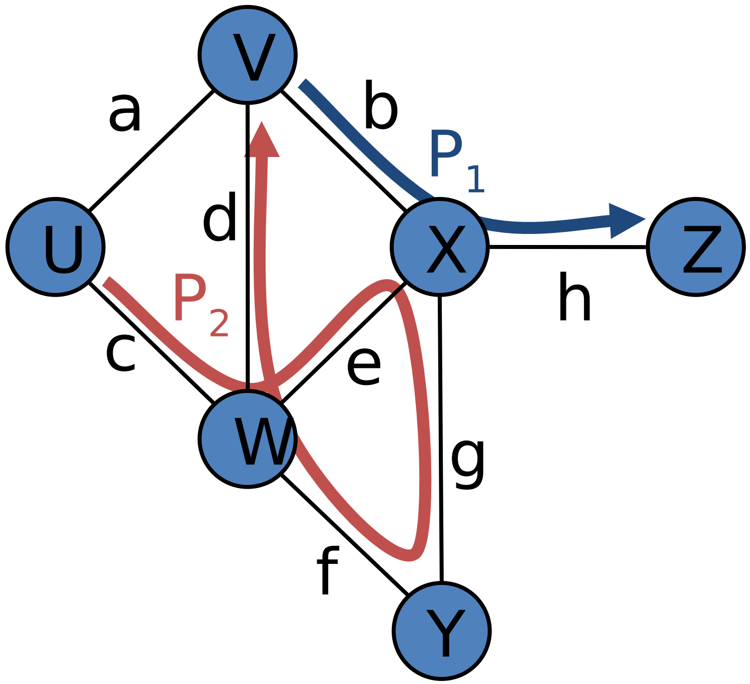
Terminology

- End vertices (or endpoints) of an edge
  - Endpoints of a?
  - U and V
- Edges incident on a vertex
  - Incident on V?
  - a, d, and b
- Adjacent vertices
  - U and V?
  - U and V are adjacent
  - U and X?
- Degree of a vertex
  - Degree of X?
  - X has degree 4



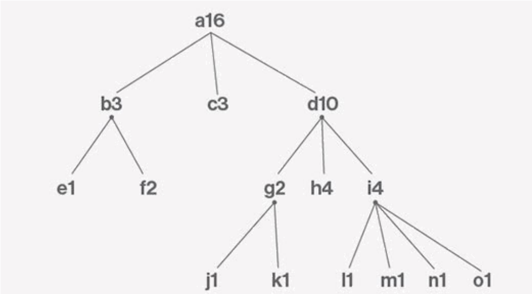
Terminology (cont.)

- 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

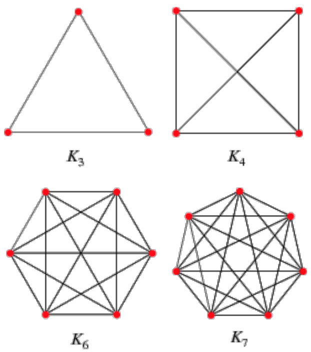


Terminology (cont.)

- **Tree**: an undirected connected graph that
  - is acyclic (no cycles)
  - OR would become disconnected if an edge is removed
  - OR any two vertices are connected by a unique simple path.



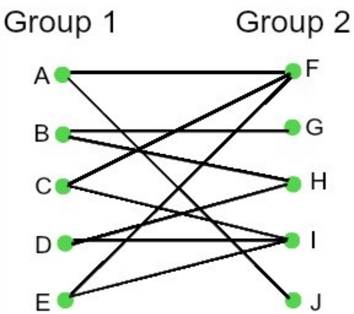
Tree



Complete

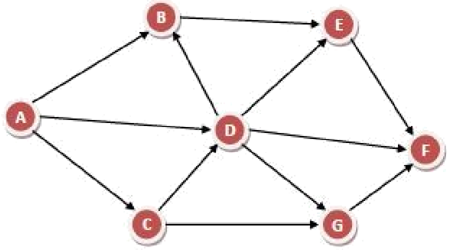
- **Complete** Graph: each pair of graph vertices is connected by an edge.

- **Bipartite** Graph: a graph whose vertices can be divided into two disjoint groups such that no two vertices in the same group share an edge



Bipartite

- **Directed Acyclic** Graph: a directed graph with no cycles



DAG

Graph Representations

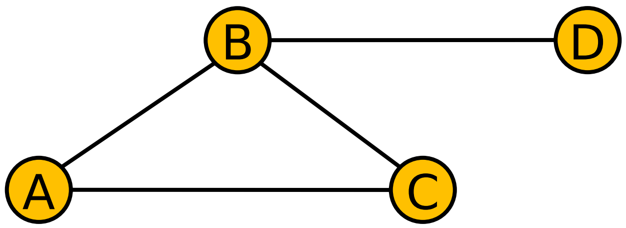
- How to represent a graph in a computer program?
- Efficiently:
  – Check if two vertices are adjacent?
  – List all adjacent vertices of a vertex
  – Add/remove a vertex to/from the graph
  – Add/remove an edge to/from the graph

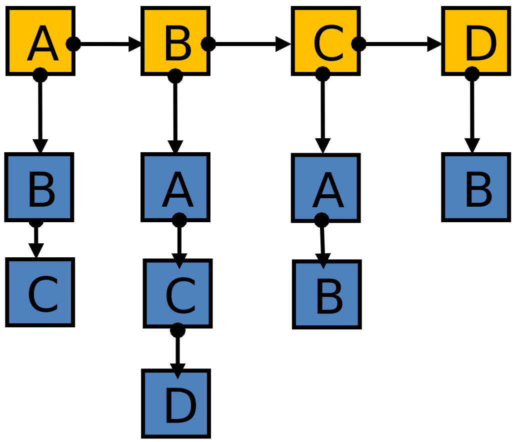
Two representations of a Graph

- Adjacency List
- Adjacency Matrix

Adjacency List

- Separate list of incident edges for each vertex





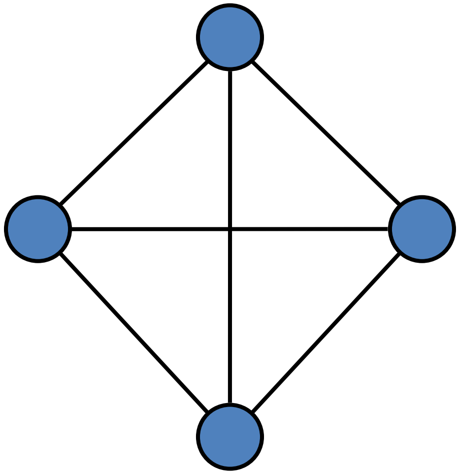
Property 1

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

Proof: each edge is counted twice

Notation

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



Example

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

Property 2

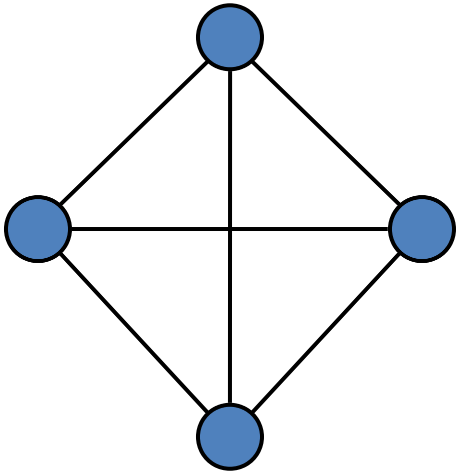
In an undirected graph with no self-loops and no multiple edges

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

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

Notation

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



Example

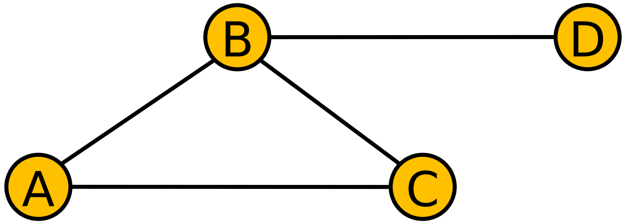
- $n = 4$
- $m = 6$
- $\deg(v) = 3$

Performance of Adjacency List structure

▪ n vertices, m edges ▪ no parallel edges, no self-loops	Adjacency List
Space	$O(n + m)$
`v.incidentEdges()`	$O(\deg(v))$
`v.isAdjacentTo (w)`	$O(\min(\deg(v), \deg(w)))$
`insertVertex(o)`	$O(1)$
`insertEdge(v, w, o)`	$O(1)$
`eraseVertex(v)`	$O(\deg(v))$
`eraseEdge(e)`	$O(1)$
`vertices()`	$O(n)$
`edges()`	$O(m)$
`e.isIncidentOn(v)`	$O(1)$
`e.endVertices()`	$O(1)$
`e.opposite(v)`	$O(1)$

Adjacency Matrix

- 2D-array
  - True when cell myarray[i][j] represents an edge
  - False for nonadjacent vertices



	A	B	C	D
A		●	●	
B	●		●	●
C	●	●		
D		●		

Comparative performance

▪ n vertices, m edges	Adjacency List	Adj. Matrix
Space	$O(n + m)$	$O(n^2)$
`List adjacent vertices of` ***v***	O(n)	O(n)
v`.isAdjacentTo (`***w***`)`	O(n)	O(1)
`insertVertex(`***v***`)`	O(1)	O(n^2)
`insertEdge(`***v***`, `***w***`)`	O(1)	O(1)