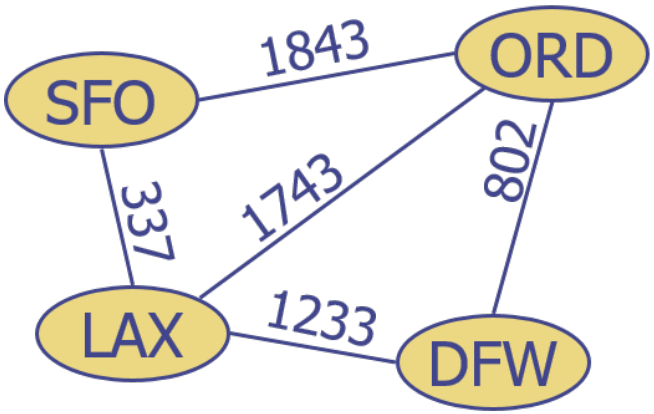
EECS 2011 M: Fundamentals of Data Structures

Suprakash Datta
Office: LAS 3043

Course page: `http://www.eecs.yorku.ca/course/2011M`
Also on Moodle

Graphs: Motivations and Basic Idea

- Tool for modeling many real applications
- Abstract model that throws away many non-essential aspects of a problem
- Nodes, connected by edges
- No geographical locations attached to node positions, no significance of edge lengths



Note: Some slides in this presentation have been adapted from the author's and Prof Elder's slides.

Graphs: Applications

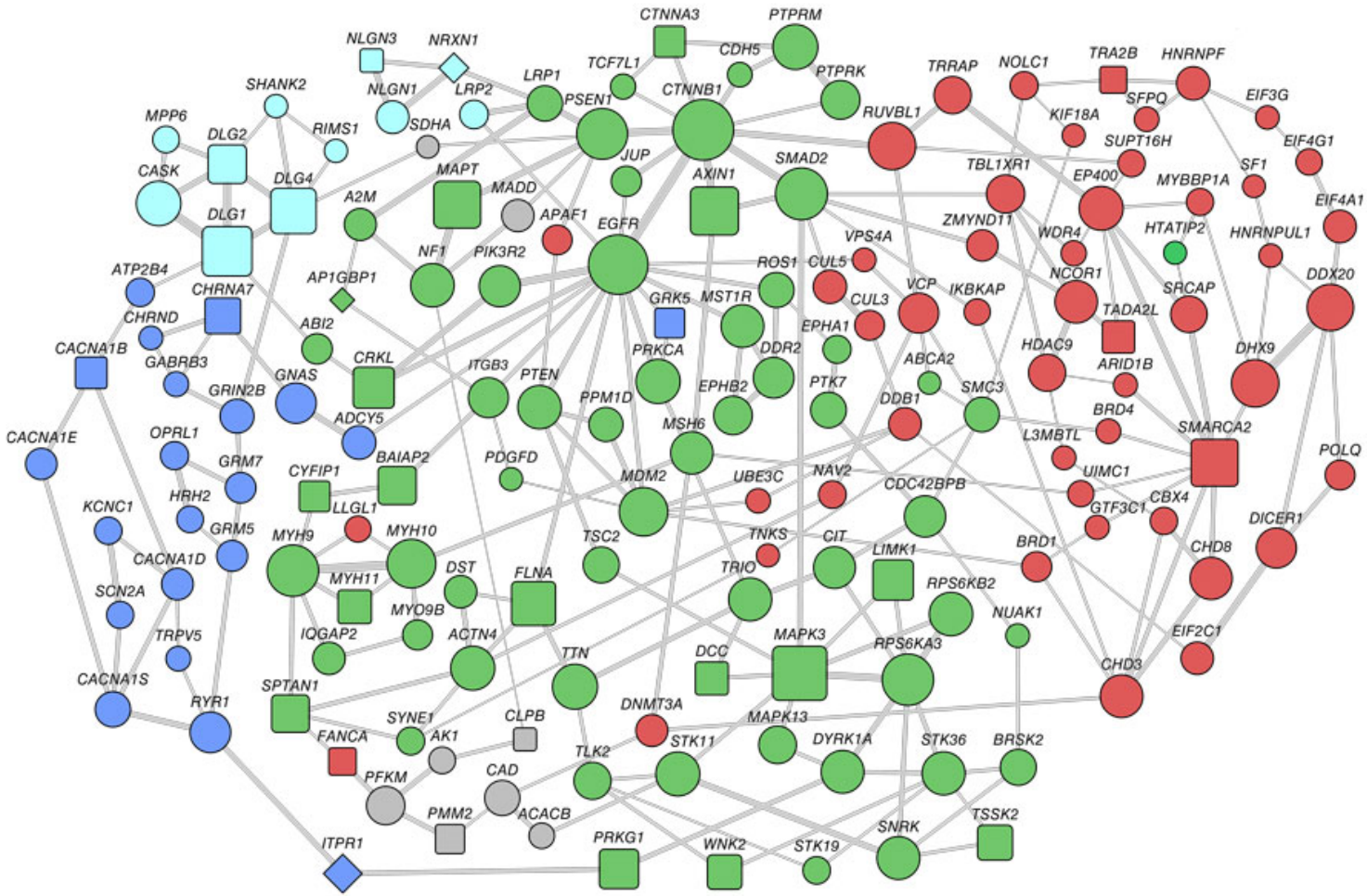
Many Applications, including:

- Road networks
- Subway/Train networks
- Airline networks
- Social Networks
- Power Grid
- Electronic Communication Networks
- Electrical Circuits
- Biological Networks
- Ecological Networks

Graphs: Applications - 2

- The web graph
- Software module dependencies
- Computation structure
- Scheduling constraints
- Collaboration graphs
- State graphs of machines and protocols
- Many, many others

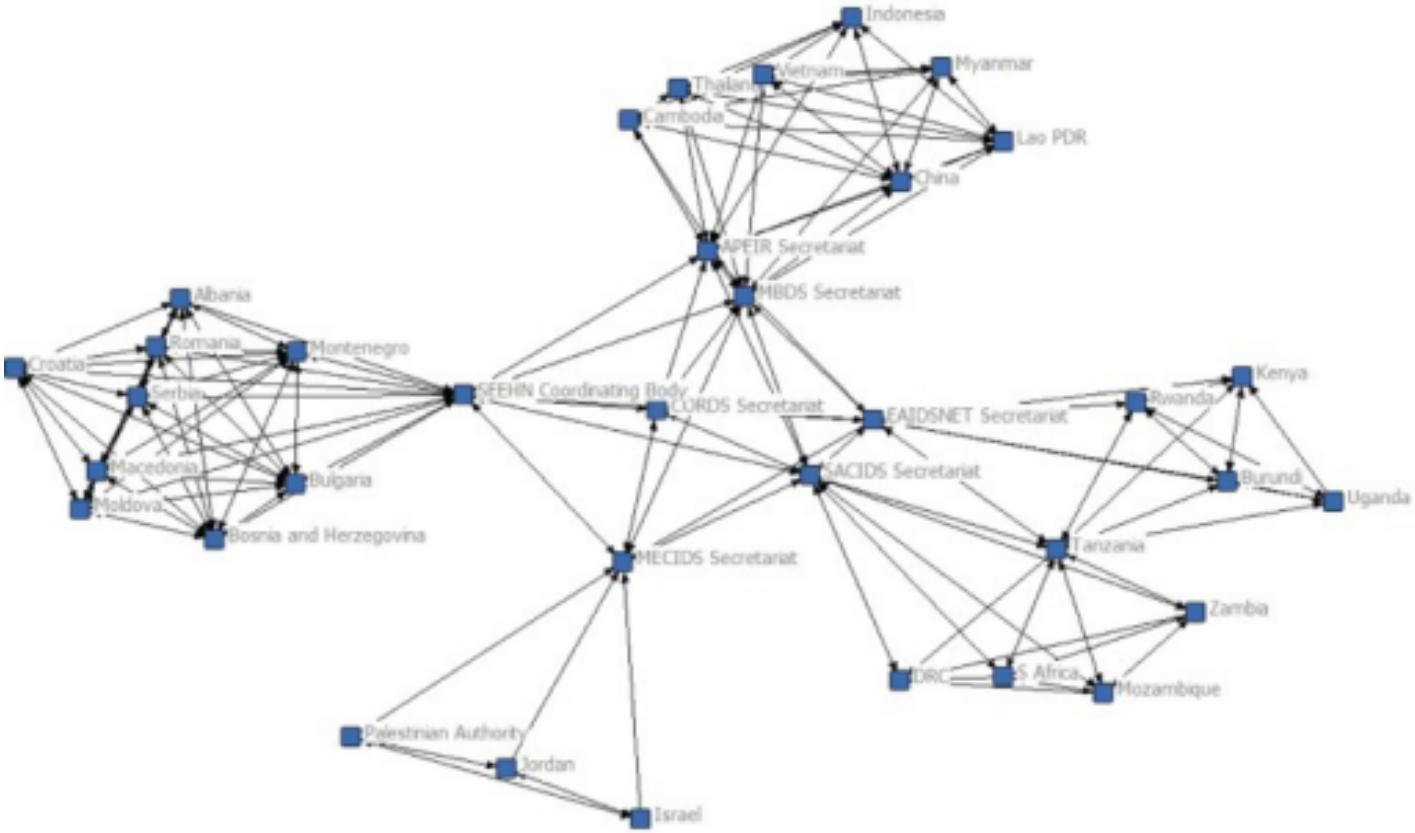
Graphs: Applications - 3



Article: Gene networks offer entry point to unraveling autism

From `https://spectrumnews.org/news/gene-networks-offer-entry-point-to-unraveling-autism/`

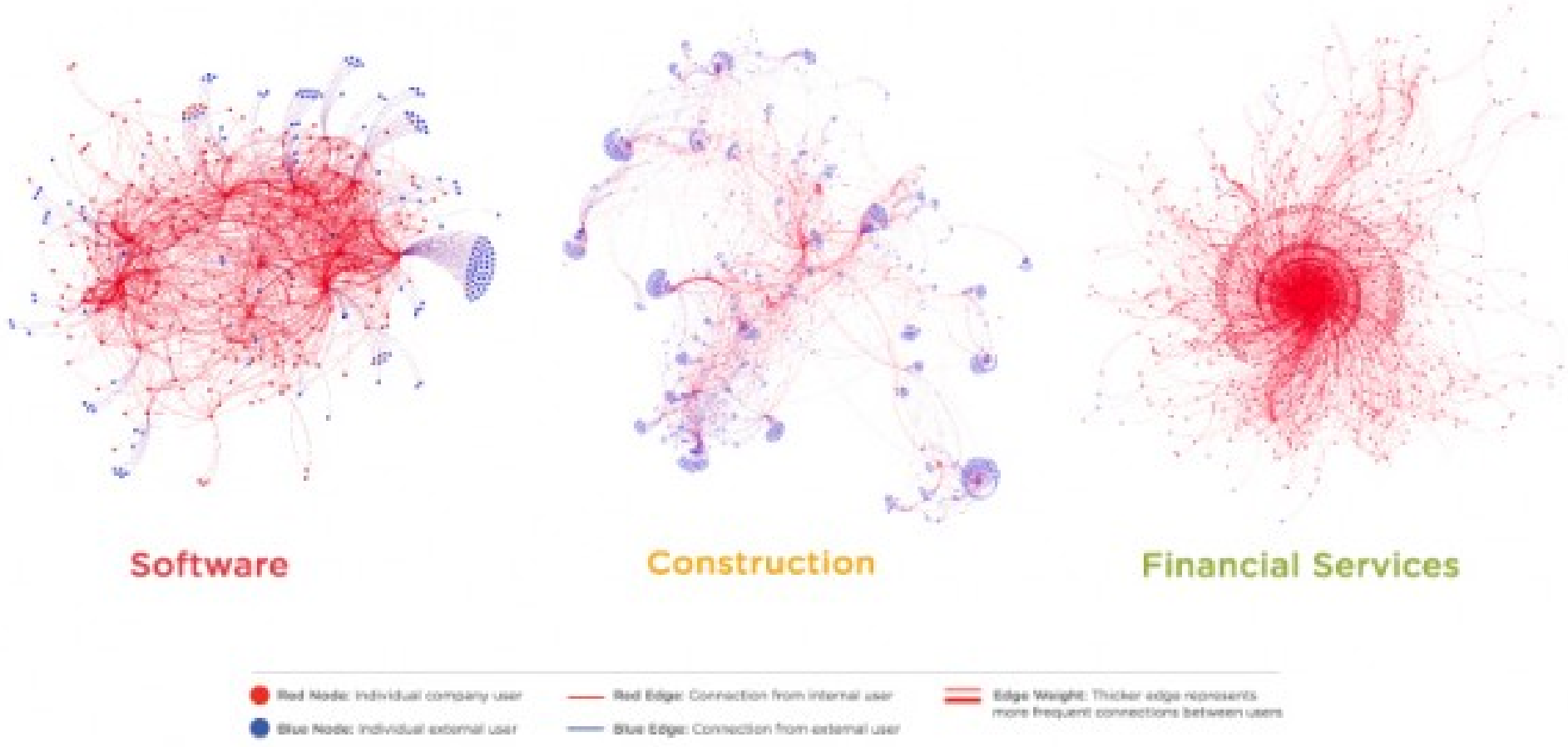
Graphs: Applications - 4



A social network graph illustrating the connections among countries and regional networks in CORDS (CORDS=Connecting Organizations for Regional Disease Surveillance;

From `https://openi.nlm.nih.gov/detailedresult.php?img=PMC3557911_EHTJ-6-19913-g001&req=4`

Graphs: Applications - 5



Collaboration graph among people in the same company

From `https://linkurio.us/blog/visualizing-business-organizations-the-collaboration-graph/`

Definitions - 1

- $G = (V, E)$, V = set of nodes/vertices, E = set of edges
- Edges incident on a vertex
- Adjacent vertices
- degree of a node
- neighborhood of a node
- Self-loop

Definitions - 2

- Edge Types:
  - Directed edge: ordered pair of vertices (u, v)
    - u : origin, v : destination
  - Undirected edge: unordered pair of vertices (u, v)
- Graph Types:
  - Directed graph: all the edges are directed
  - Undirected graph: all the edges are undirected
- Paths:
  - Simple Paths
  - Cycles
  - Simple cycles: no vertex repeated

Elementary Properties

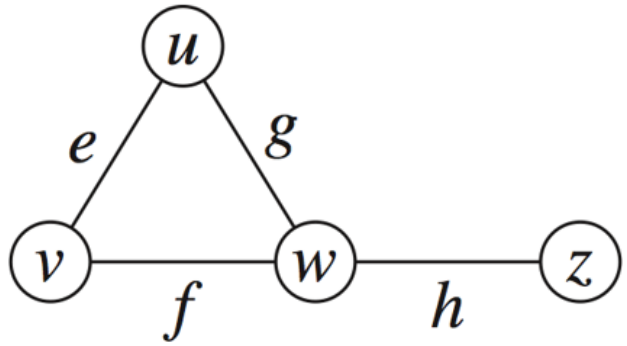
- The sum of degrees is even (equals twice the number of edges in an undirected graph)
- The sum of indegrees equals sum of outdegrees in a directed graph
- In an undirected graph $m \leq \frac{n(n-1)}{2}$
  What is the bound for directed graphs?

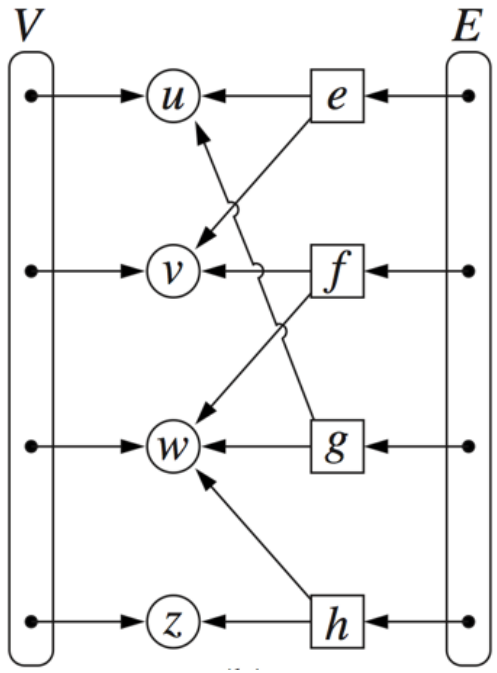
Graph Representations

- Edge list
- Adjacency list
- Adjacency matrix

Edge Lists

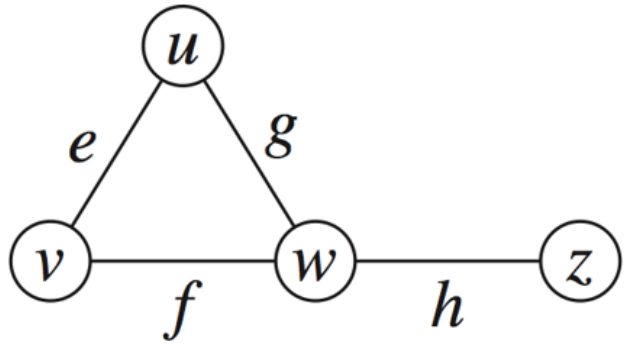
- Vertex object: reference to position in vertex sequence
- Edge object: origin vertex object, destination vertex object, reference to position in edge sequence
- Vertex sequence: sequence of vertex objects
- Edge sequence: sequence of edge objects

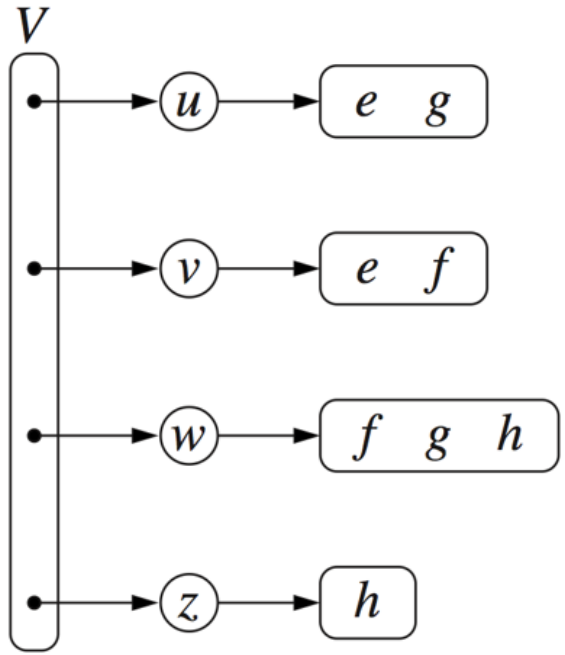




Adjacency Lists

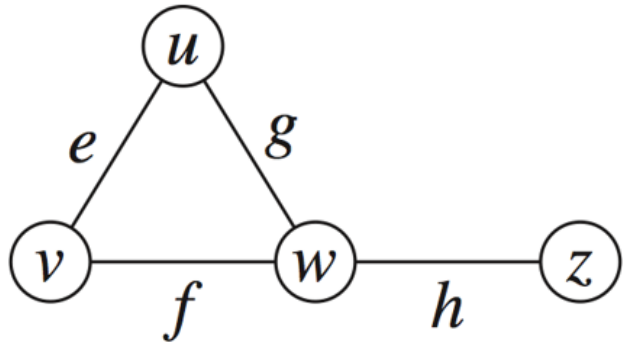
- Incidence sequence for each vertex: sequence of references to edge objects of incident edges
- Augmented edge objects: references to associated positions in incidence sequences of end vertices





Adjacency Matrix

- 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



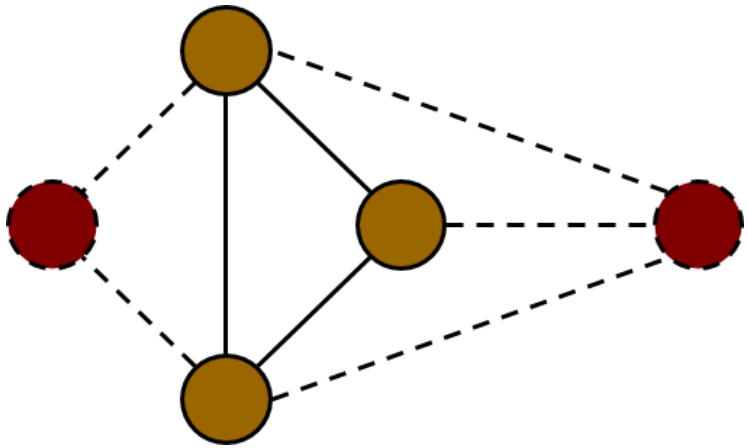
		0	1	2	3
u	0		e	g	
v	1	e		f	
w	2	g	f		h
z	3			h	

Performance

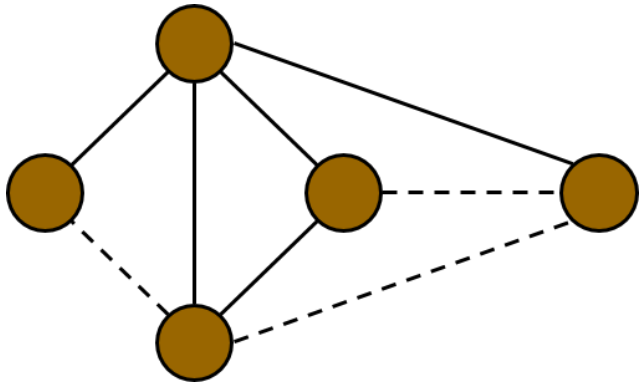
▪ n vertices, m edges ▪ no parallel edges ▪ no self-loops	Edge List	Adjacency List	Adjacency Matrix
Space	$n + m$	$n + m$	n^2
incidentEdges(v)	m	$\deg(v)$	n
areAdjacent (v, w)	m	$\min(\deg(v), \deg(w))$	1
insertVertex(o)	1	1	n^2
insertEdge(v, w, o)	1	1	1
removeVertex(v)	m	$\deg(v)$	n^2
removeEdge(e)	1	1	1

Subgraphs

- A subgraph S of a graph G is a graph such that
  - The vertices of S are a subset of the vertices of G
  - The edges of S are a subset of the edges of G
- A spanning subgraph of G is a subgraph that contains all the vertices of G



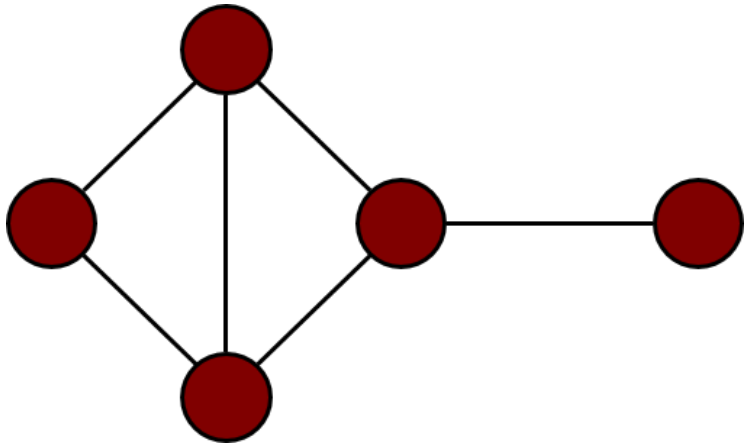
Subgraph



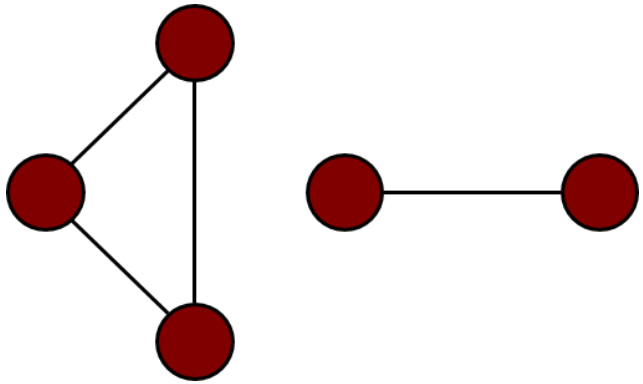
Spanning subgraph

Connected graphs

- A graph is connected if there is a path between every pair of vertices
- A connected component of a graph G is a maximal connected subgraph of G



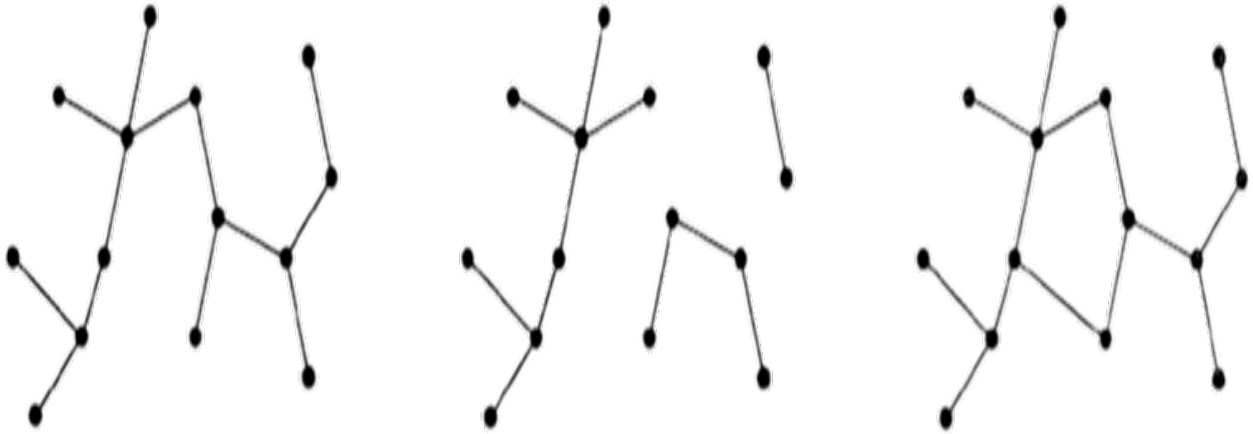
Connected graph



Disconnected graph with two connected components

Trees

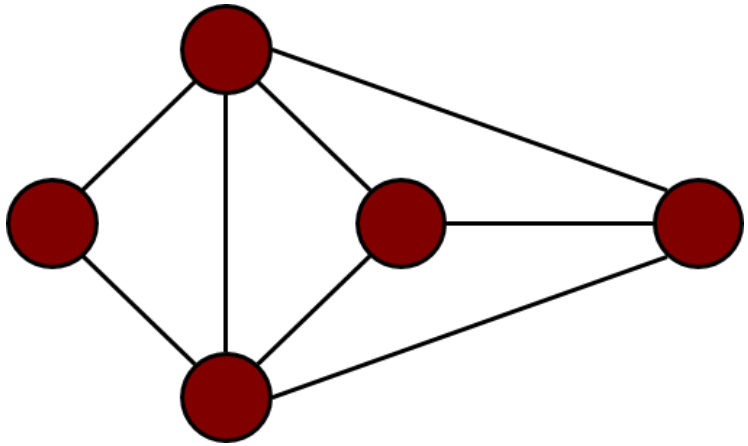
- A tree is a connected, acyclic, undirected graph
- A forest is a set of trees (not necessarily connected)



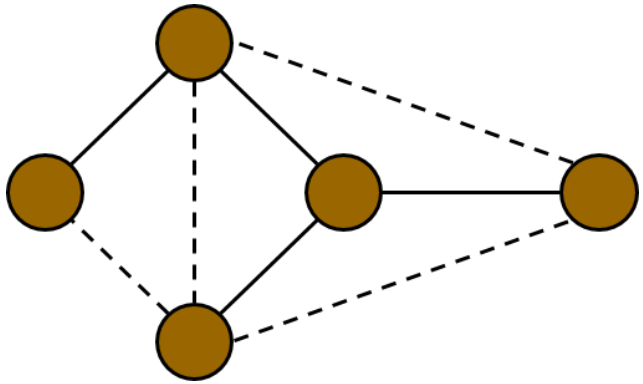
Tree, forest, a cyclic graph

Spanning Trees

- A spanning tree of a connected graph is a spanning subgraph that is a tree
- A spanning tree is not unique unless the graph is a tree
- Spanning trees have applications to the design of communication networks
- A spanning forest of a graph is a spanning subgraph that is a forest



graph



Spanning tree

Graph Problems

- Connectivity: Are all vertices reachable from each other?
- Reachability: Is a node v reachable from a node u?
- Graph Isomorphism
- Graph Coloring
- And many others

Graph ADT

- Accessor methods
  - numVertices(): Returns the number of vertices in the graph
  - numEdges(): Returns the number of vertices in the graph
  - getEdge(u, v): Returns edge from u to v
  - endVertices(e): an array of the two endvertices of e
  - opposite(v, e): the vertex opposite to v on e
  - outDegree(v): Returns number of outgoing edges
  - inDegree(v): Returns number of incoming edges

Graph ADT - 2

- Update methods
  - insertVertex(x): insert a vertex storing element x
  - insertEdge(u, v, x): insert an edge (u,v) storing element x
  - removeVertex(v): remove vertex v (and its incident edges)
  - removeEdge(e): remove edge e
- Iterator methods
  - incomingEdges(v): Incoming edges to v
  - outgoingEdges(v): Outgoing edges from v
  - vertices(): all vertices in the graph
  - edges(): all edges in the graph

Graph ADT - Implementation

- Many implementations
- Look at Prof. Elder's slides for details on the net.datastructures library