ECE 2574
Introduction to Data Structures and Algorithms

38: Introduction to Graphs

Chris Wyatt
Electrical and Computer Engineering
Virginia Tech

Graphs

One can approach graphs from different perspectives

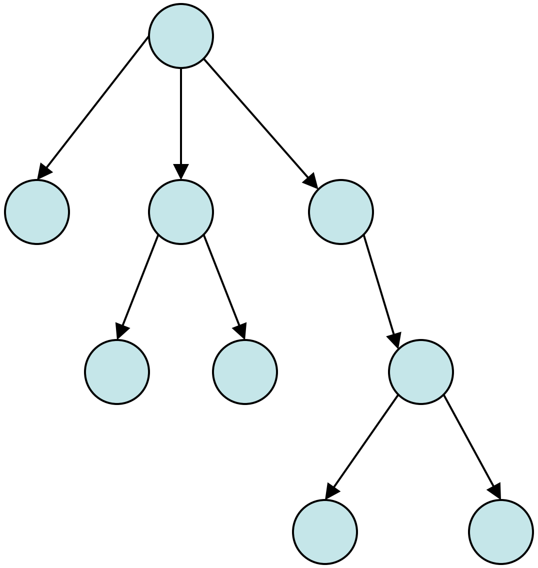
1) It is a data structure: extension of trees

2) Is is a mathematical construct

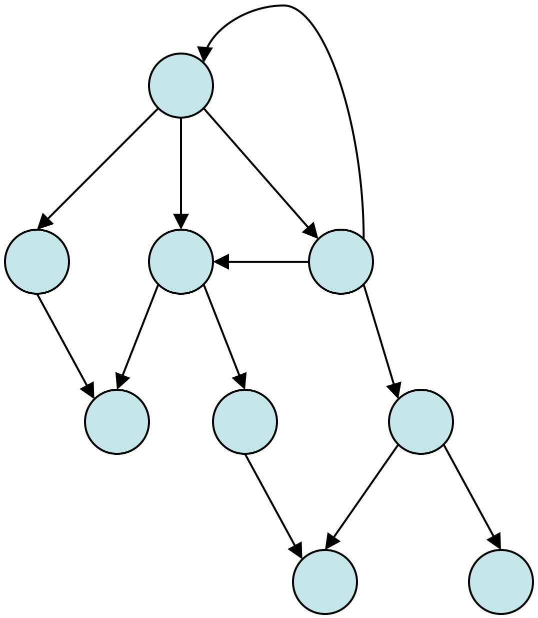
3) A model of many different real world problems

From trees to graphs

Tree



Graph



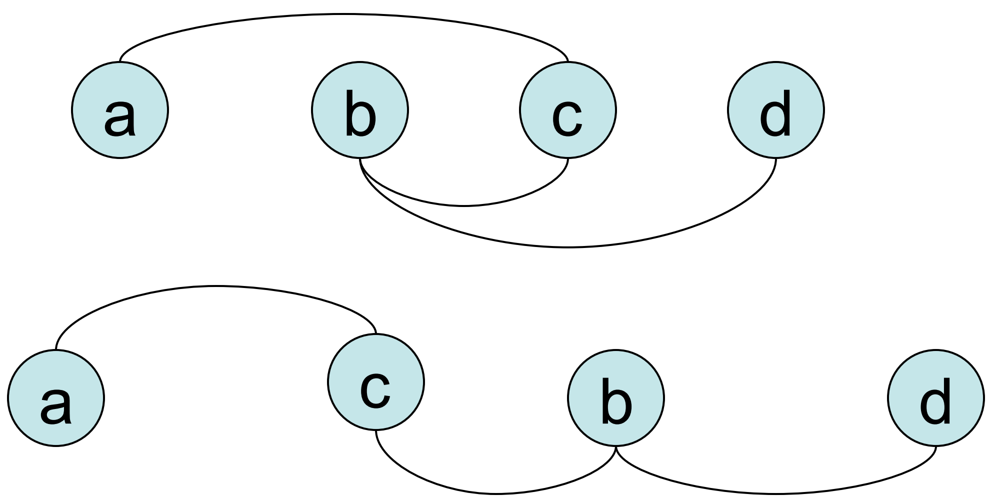
Definition of a graph

A graph is a collection of vertices (nodes), V, and a set of pairs of vertices, E.

G = {V,E}

Example: V = {a b c d} E = {(a,c) (c,b) (b,d)}

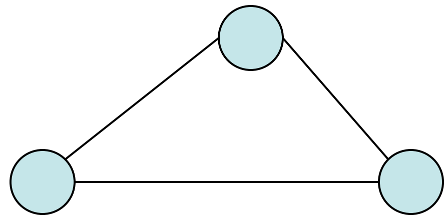
a and c are ***adjacent***



Terminology: Edges

Edges may be

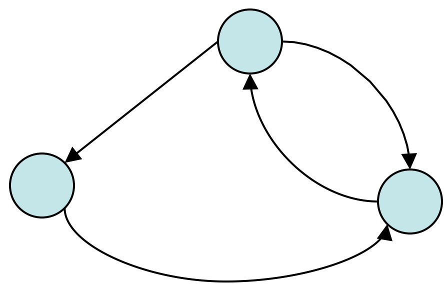
undirected



order of vertex pairs neglected

or

directed

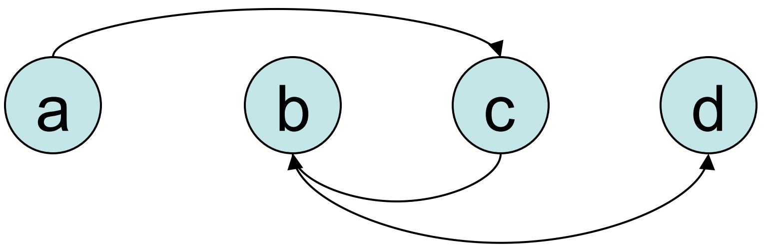


order of vertex pairs determines direction

Terminology: graphs vs multi-graphs

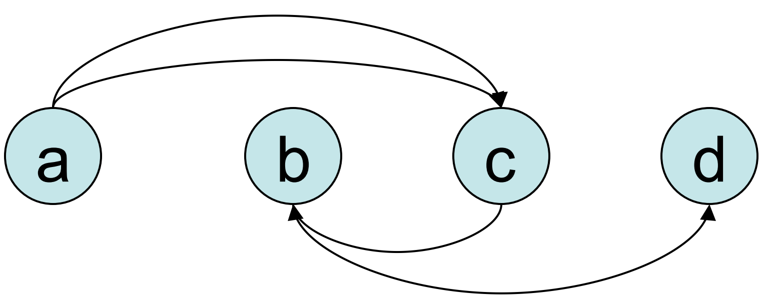
Graphs have at most one edge between two vertices

V = {a b c d} E = {(a,c) (c,b) (b,d)}



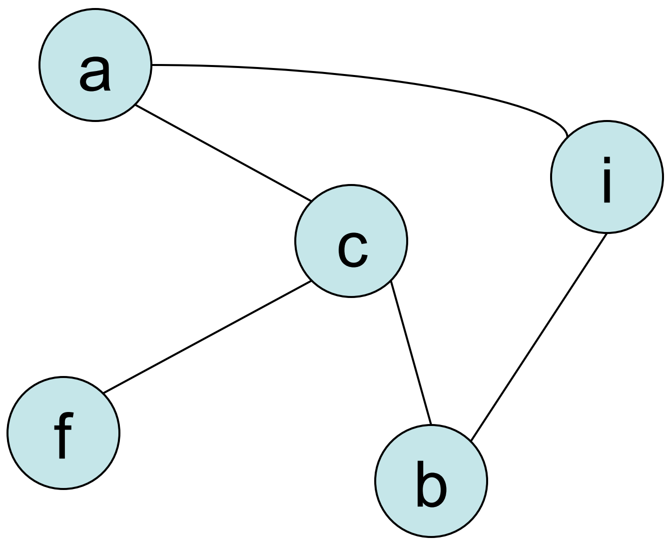
Multigraphs allow duplicate edges

V = {a b c d} E = { **(a,c)** (c,b) (b,d) **(a,c)** }

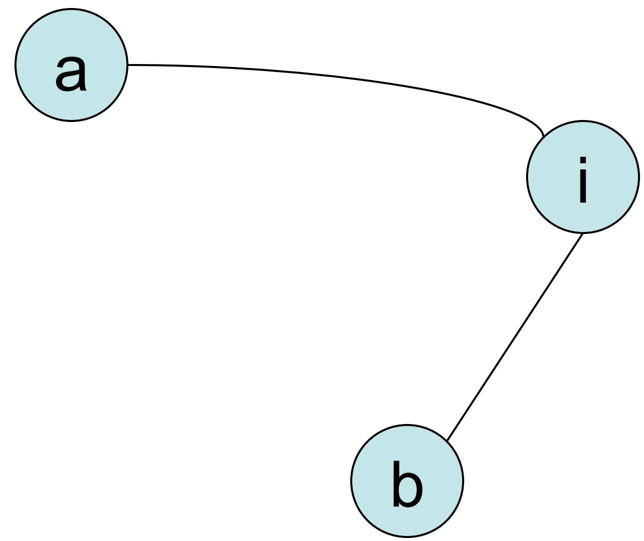


Terminology: subgraphs

Any subset of V and E forms a subgraph



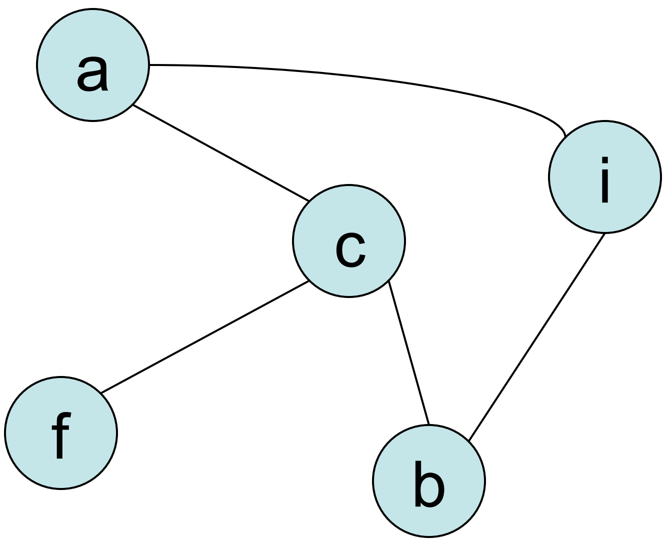
Undirected Graph G



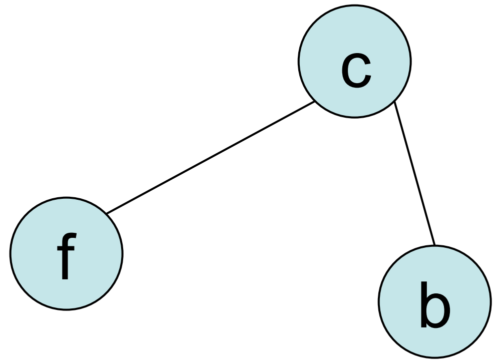
A subgraph of G

Terminology: subgraphs

Any subset of V and E forms a subgraph



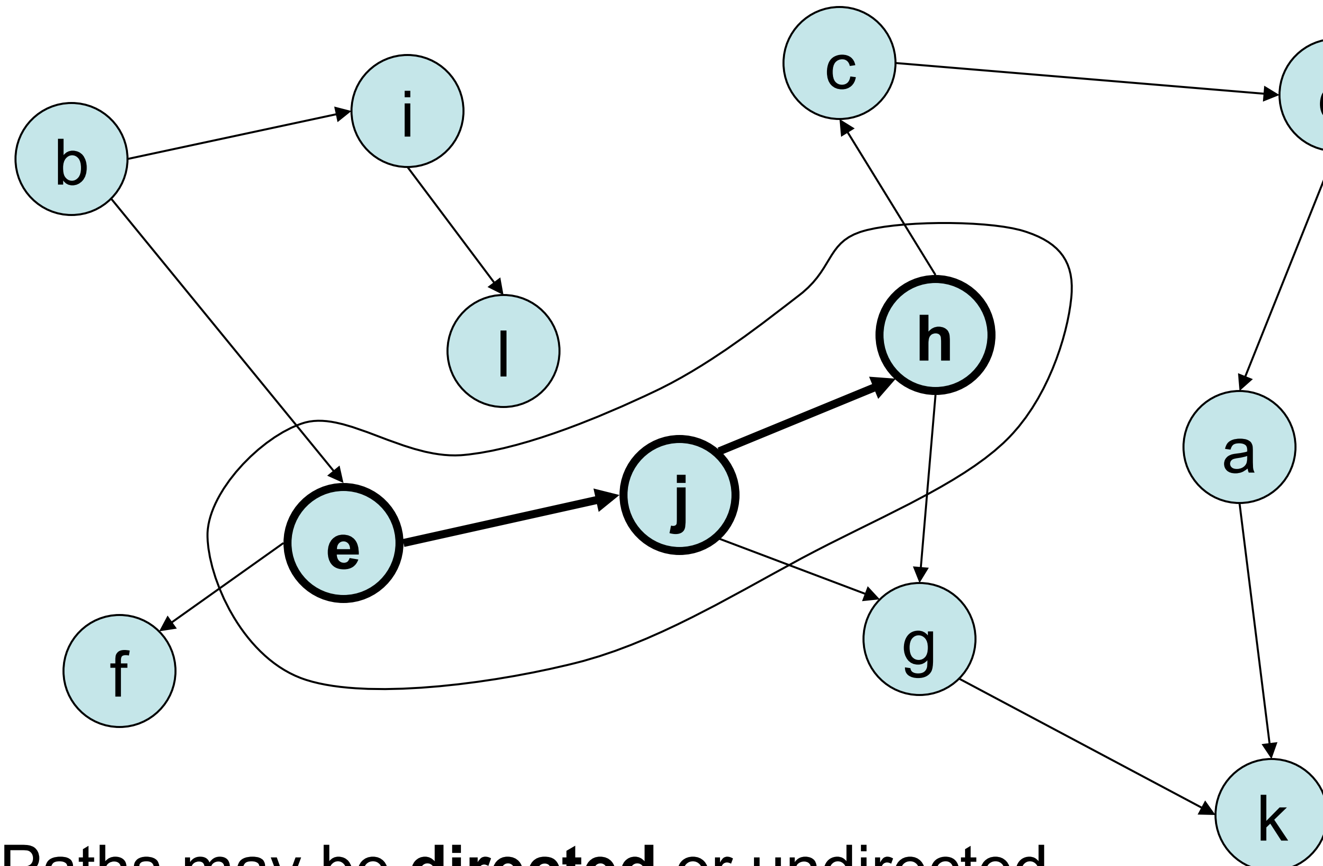
Undirected Graph G



Another subgraph of G

Terminology: paths

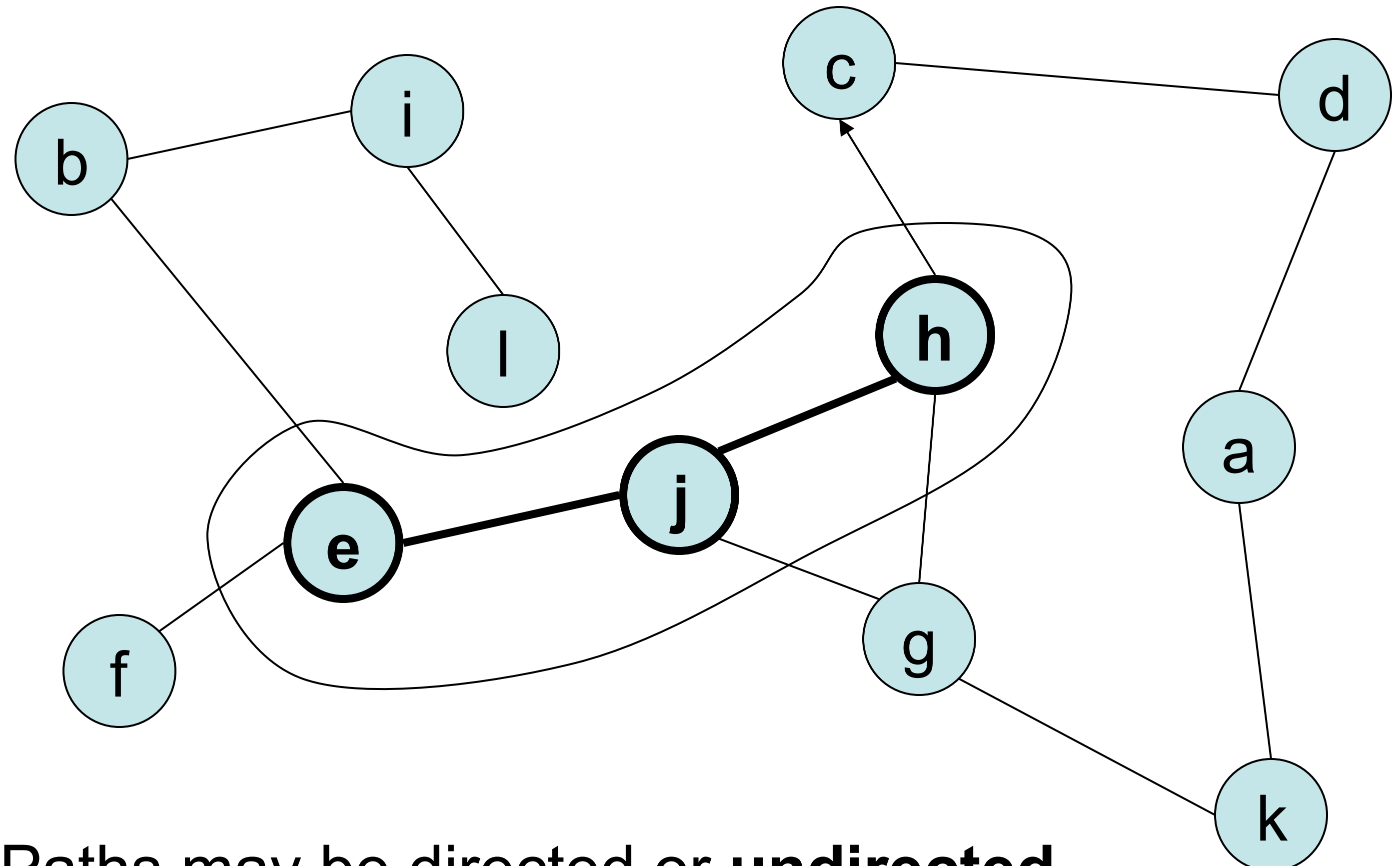
A path is a sequence of vertices connected by edges



Paths may be **directed** or undirected

Terminology: paths

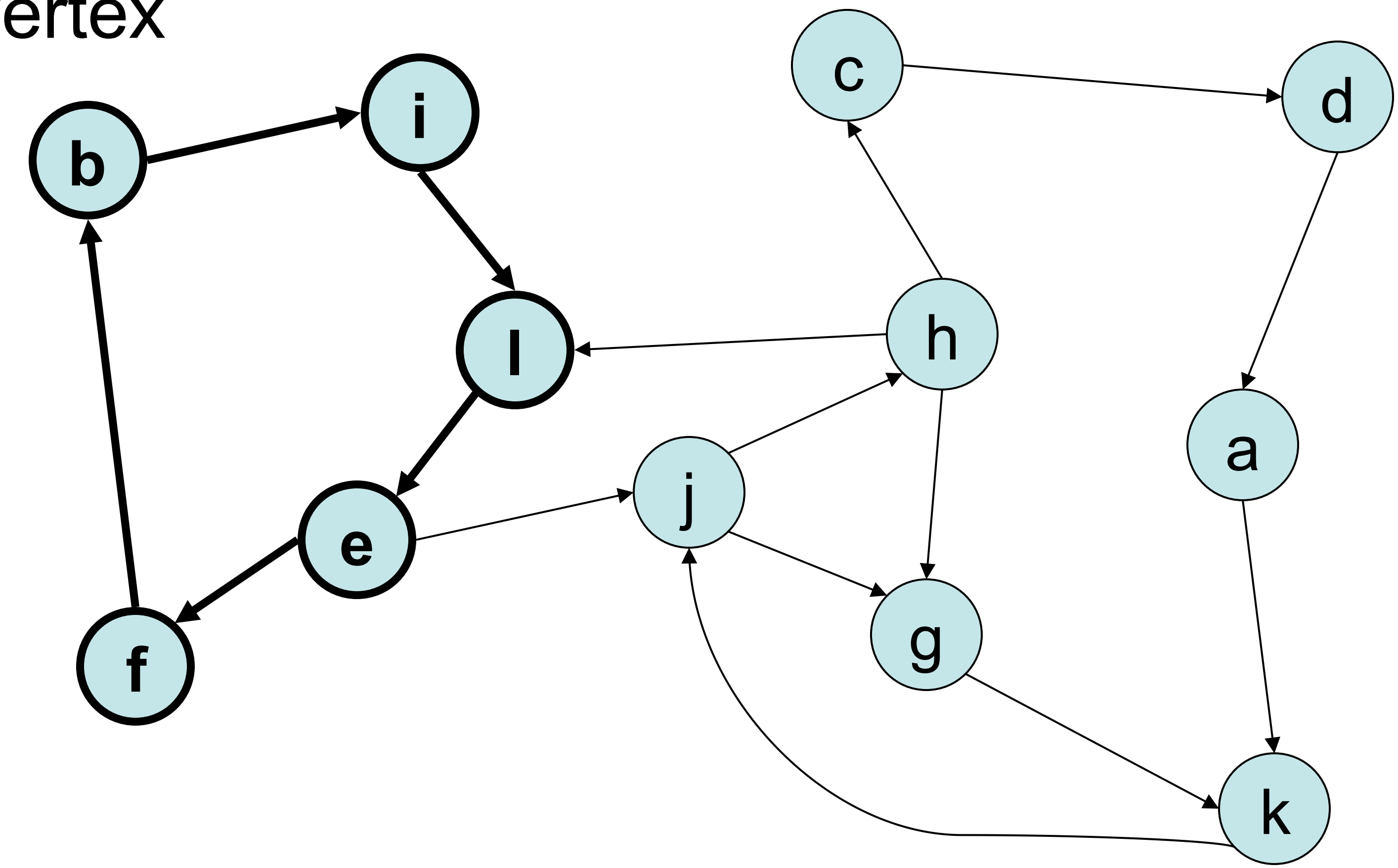
A path is a sequence of vertices connected by edges



Paths may be directed or **undirected**

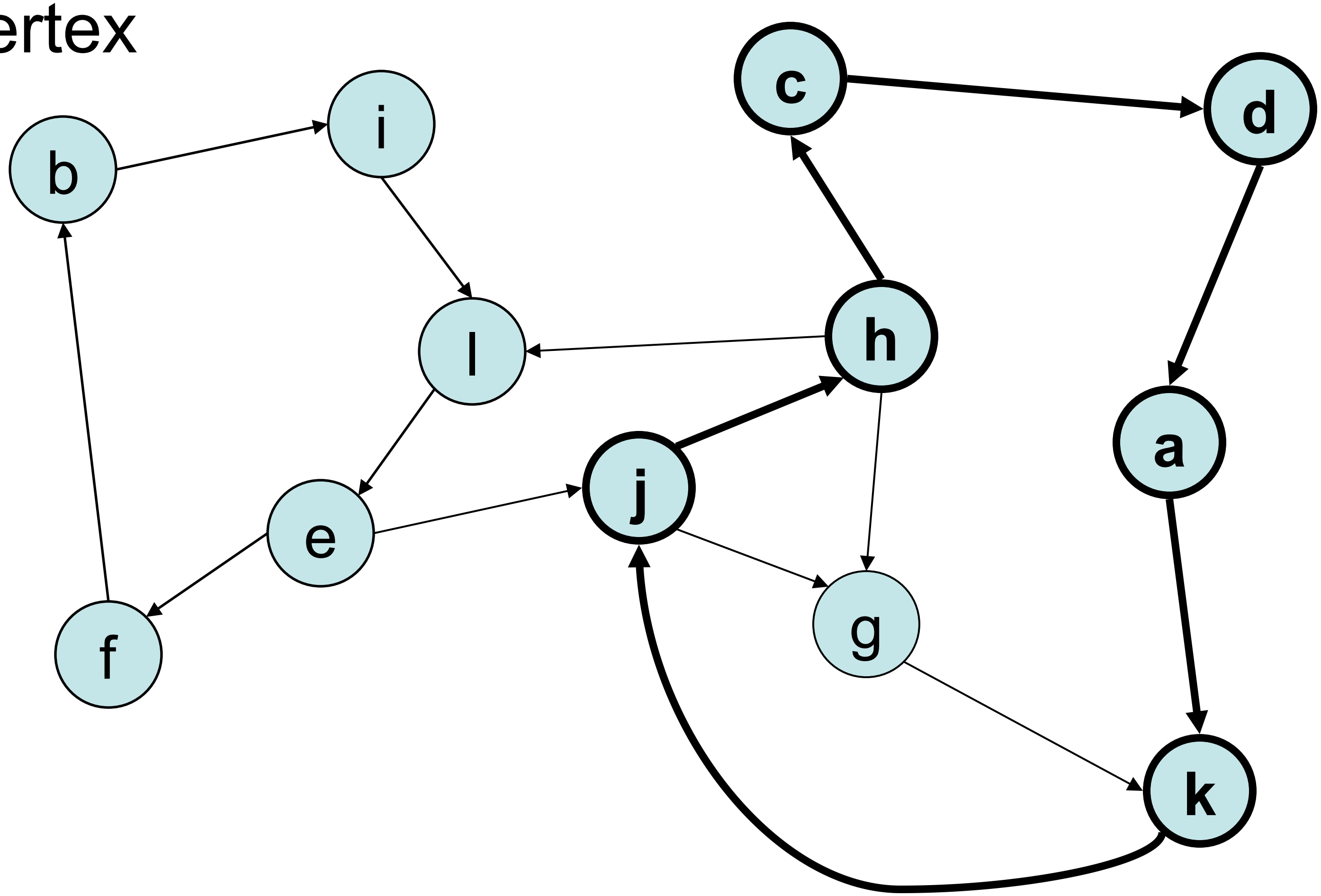
Terminology: cycle

A cycle is a path that starts and stops at the same vertex



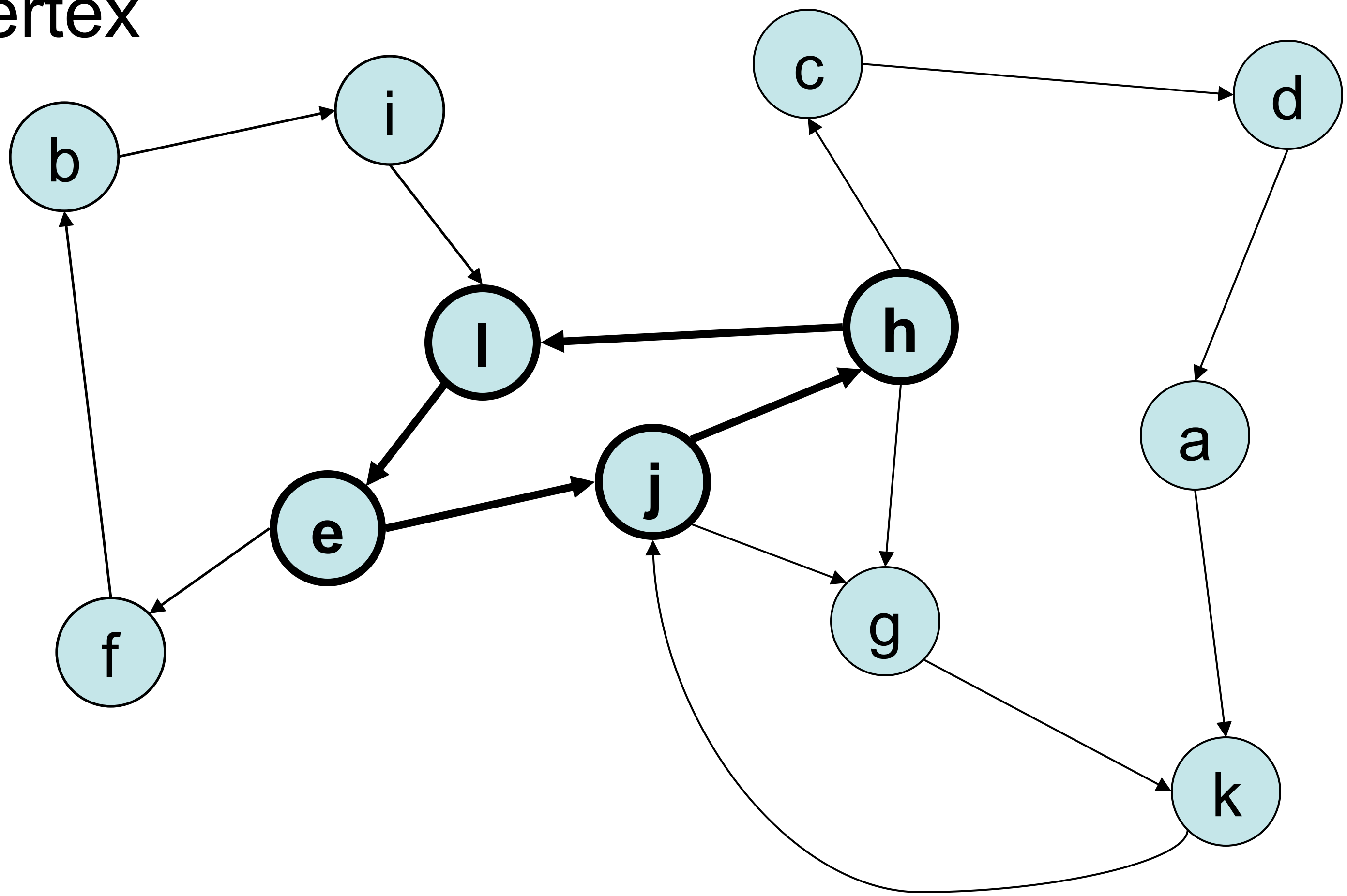
Terminology: cycle

A cycle is a path that starts and stops at the same vertex



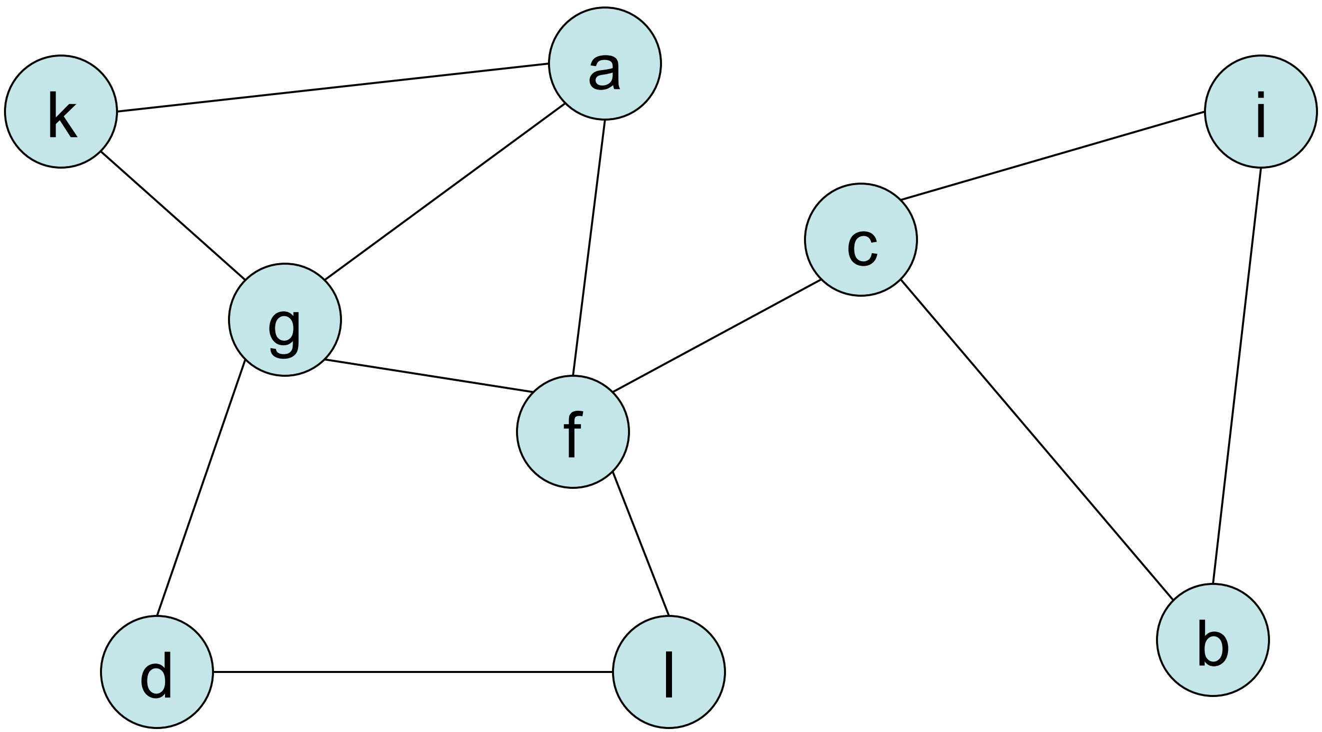
Terminology: cycle

A cycle is a path that starts and stops at the same vertex



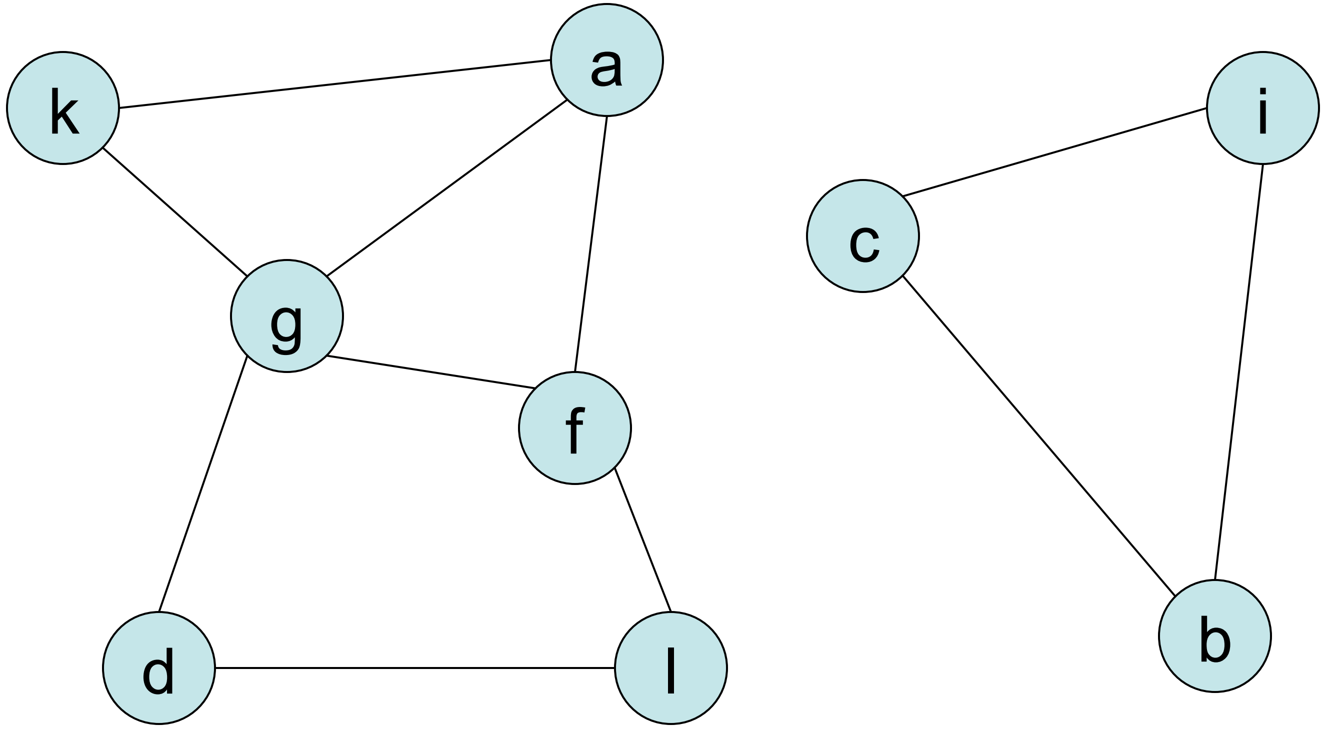
Terminology: connected / disconnected

A graph is connected if every pair of vertices are connected by at least one path



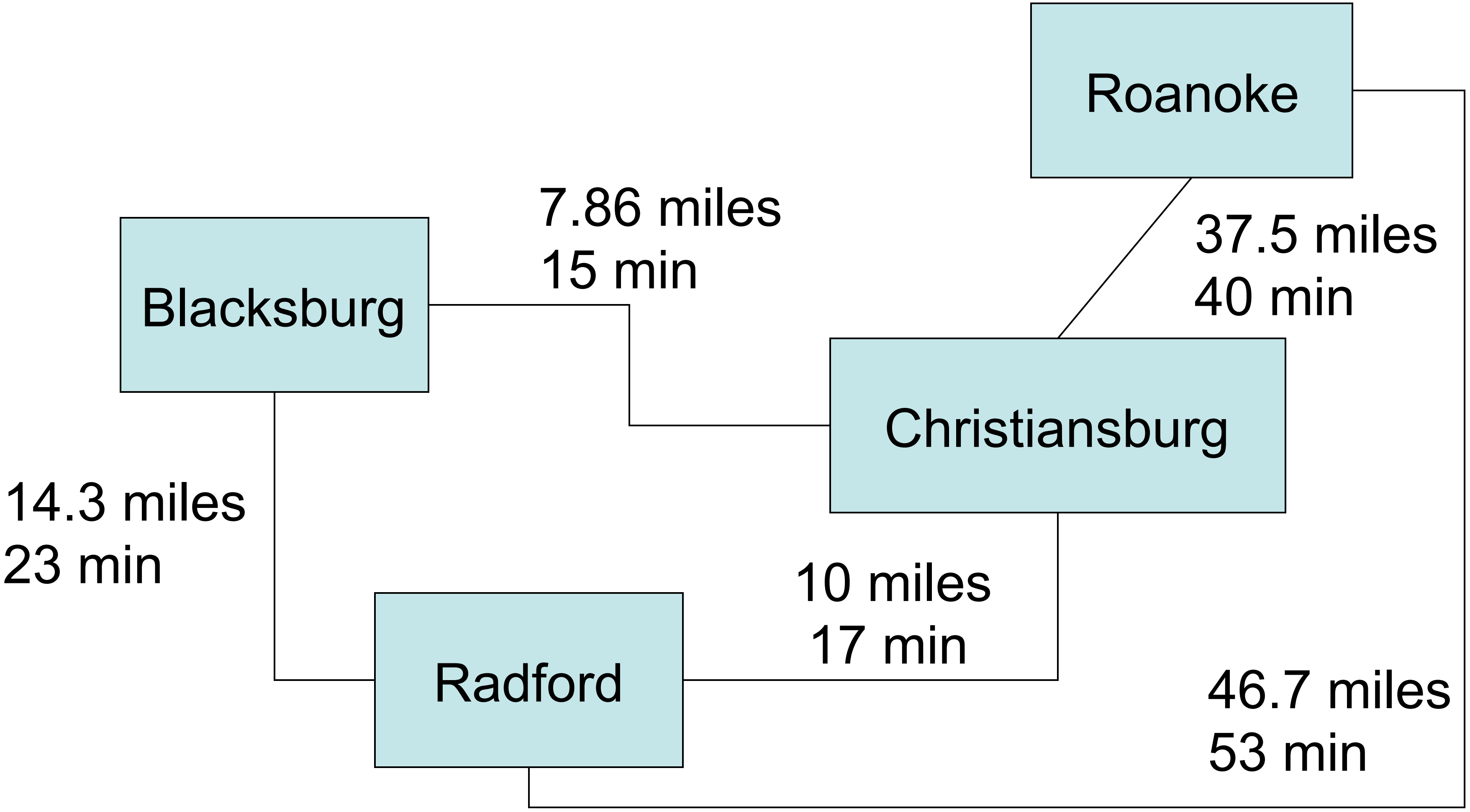
Terminology: connected / disconnected

Otherwise it is disconnected.



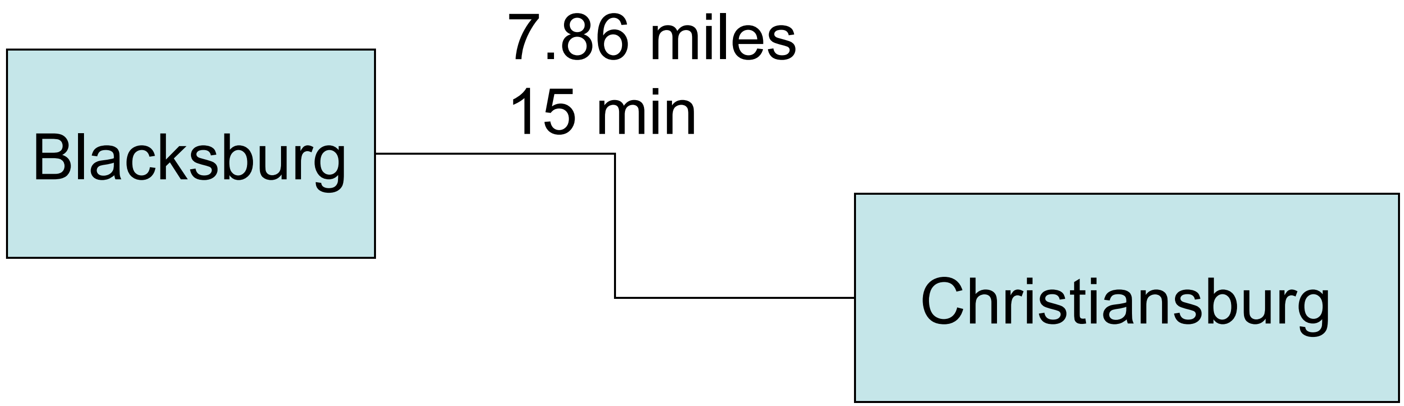
Vertices and Edges can have properties or attributes attached to them.

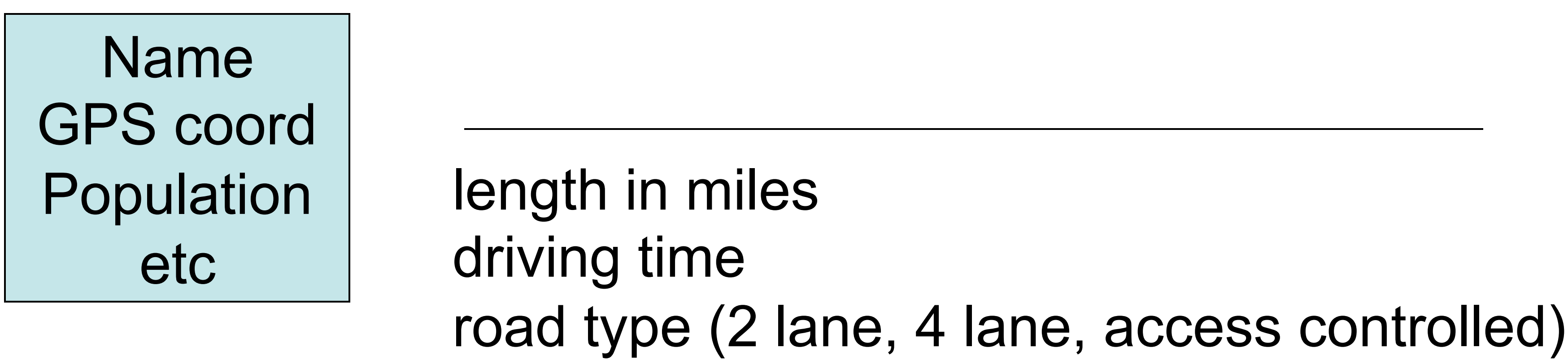
Example: driving routes between cities



Vertices and Edges can have properties or attributes attached to them.

Example: driving routes between cities





A commonly encountered graph is one where the edge property is a *weight*.

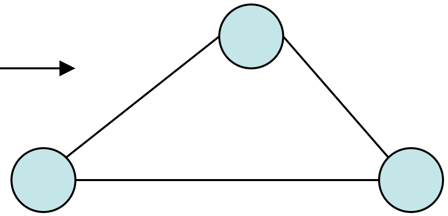
Weighted graphs (directed or undirected) have edges whose property is a cost.

Often one want to find paths connecting nodes where some function of the sum of the weights is optimal a min or max).

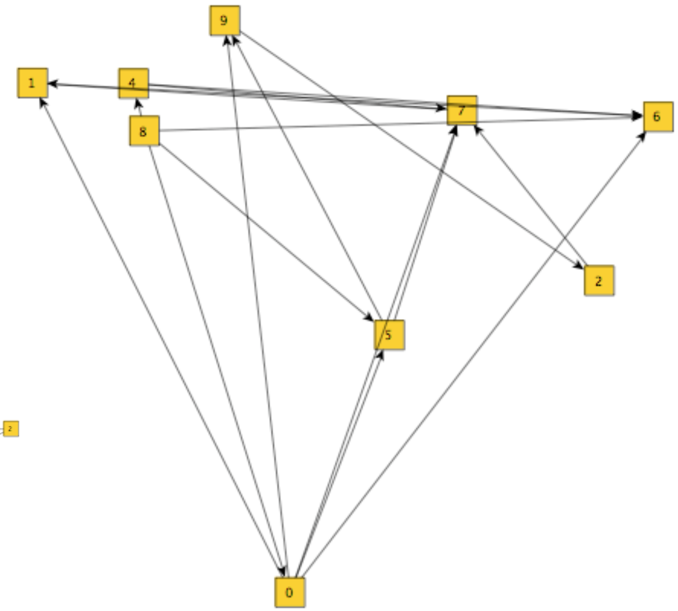
For example: what is the shortest route from Roanoke to Radford? what is the fastest? etc.

Some categories of graphs

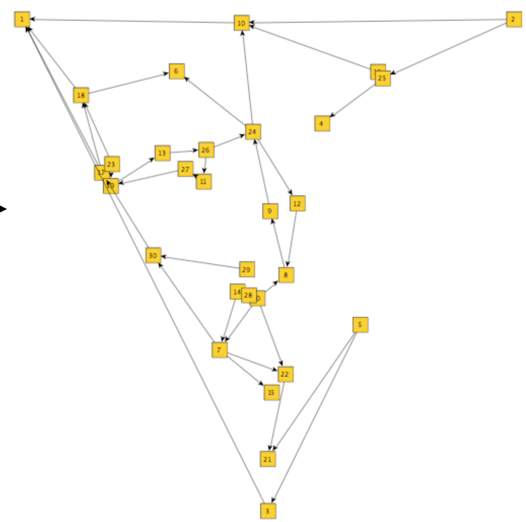
Complete



Random



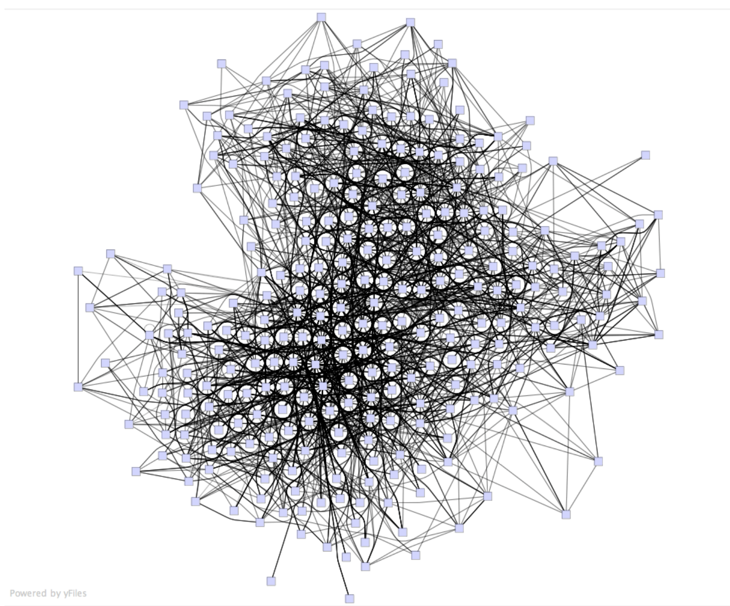
Planar



Small-World Graphs

Small world graphs often appear in the real world and have very interesting properties.

- Seven degrees of Kevin Bacon
- Large Scale Computer Networks
- Brains



Neural Connections in the worm C-elegans

279 vertices

6,417 edges

Example uses of graphs

Path planning
Layout routing
Games and puzzles
Many kinds of circuits
Networked systems
Optimization
Constraint Satisfaction
Logical Inference
Probabilistic Inference
..... on and on .....

Implementing Graphs

There is no graph data structure in the current standard C++ library.

It is easy to roll your own using existing standard library containers.

There is also the boost graph library (www.boost.org)

Three common approaches to representing graphs.

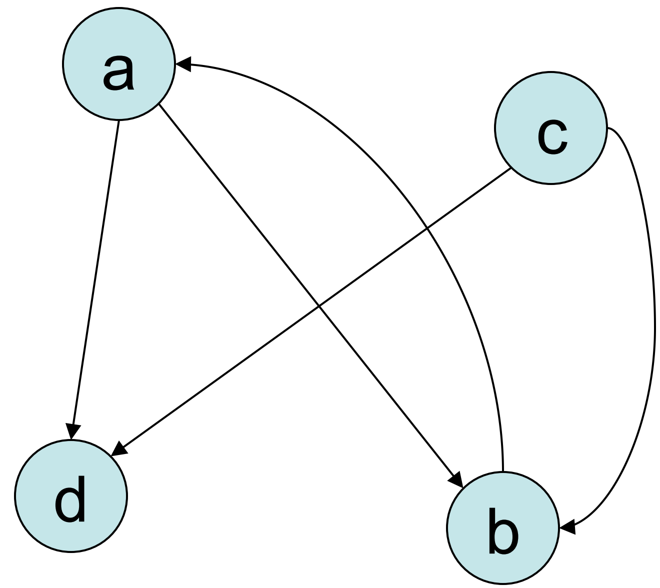
Adjacency matrix: given N vertices, the edges are indicated by an NxN matrix

Adjacency List: given N vertices, the edges are indicated by a list of connected vertices for each vertex.

Pointer based: given a pointer to a vertex, which contains pointers to it's adjacent vertices

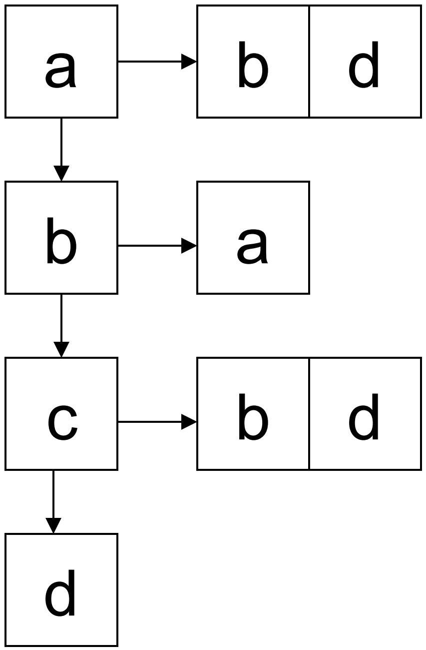
Graph using an adjacency matrix

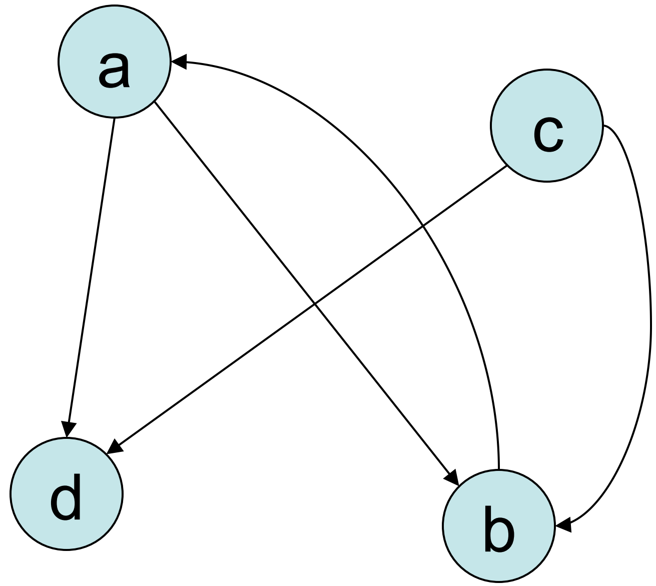
	a	b	c	d
a	0	1	0	1
b	1	0	0	0
c	0	1	0	1
d	0	0	0	0



- Undirected graphs have a symmetric matrix
- Weighted graphs have integer or real entries

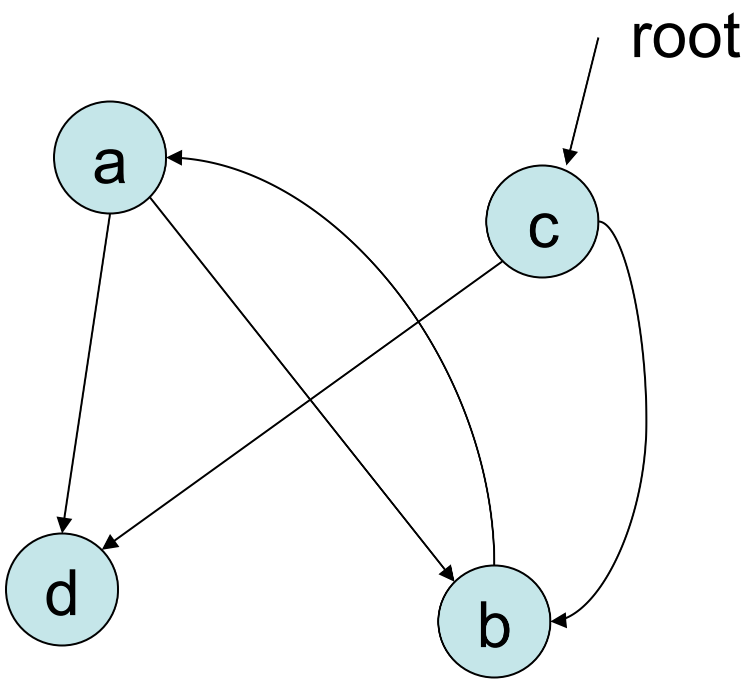
Graph using an adjacency list





- the lists could be vectors, linked, or trees

Graph using pointers



- graph must have a root and be connected.
  - Why?

Advantages/Disadvantages of implementations

Adjacency matrix

Advantages

1. simple
2. space efficient for dense graphs (~ complete)
3. fast access to all edges

Disadvantages

1. space inefficient for sparse graphs

Advantages/Disadvantages of implementations

Adjacency list

Advantages

1. space efficient for sparse graphs

Disadvantages

1. space inefficient for dense graphs
2. access to arbitrary edges slower

Advantages/Disadvantages of implementations

Pointer based

Advantages

1. space efficient for sparse graphs

Disadvantages

1. space inefficient for dense graphs
2. access to arbitrary edges slower
3. cannot represent disconnected graphs (easily)

Next Actions and Reminders

Read CH pp. 614-630 on graph traversals and algorithms

Program 5 is due 12/11

Please Fill out the SPOT survey!