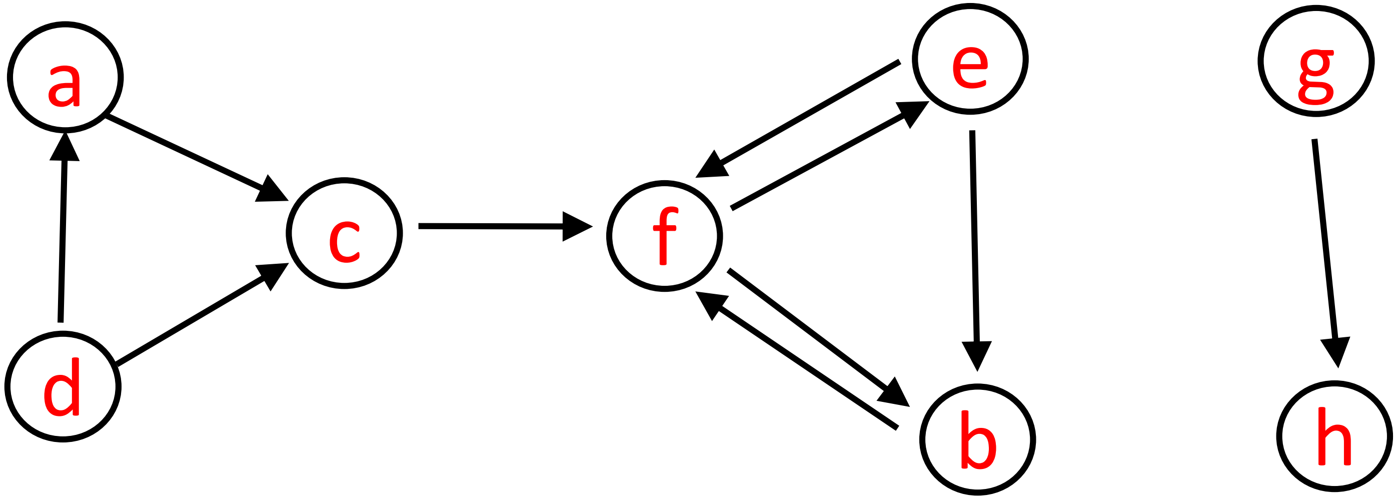
COMP 250

Lecture 27

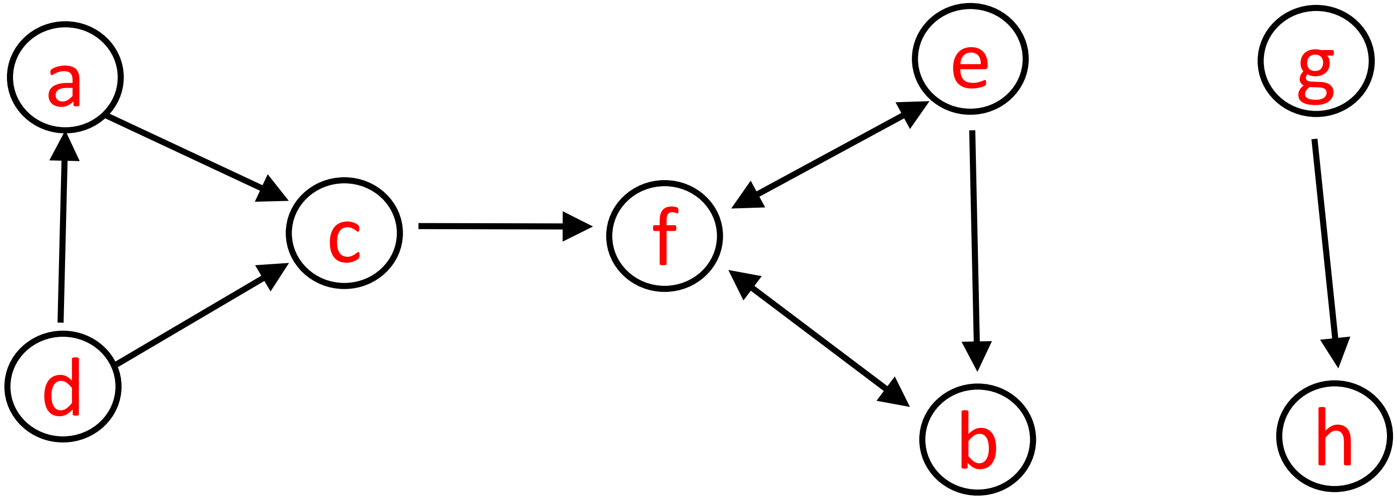
graphs

Nov. 11, 2016

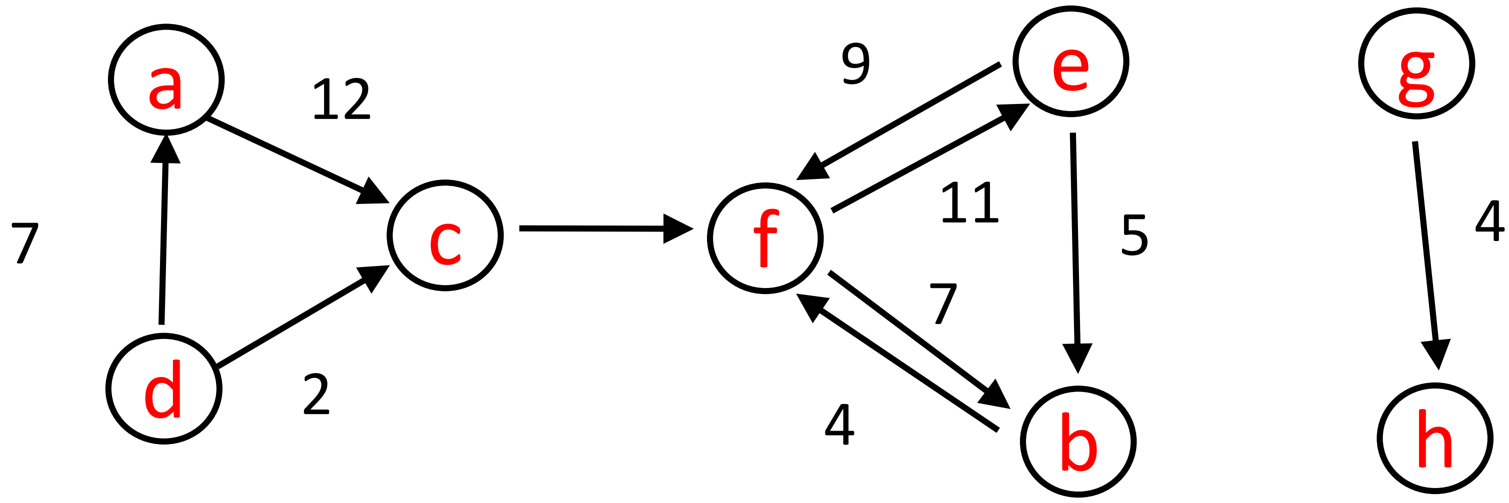
Example



Same Example – different notation



Weighted Graph



Definition

A *directed graph* is a set of *vertices*

$$V = \{v_i \;:\; i \in 1, \ldots, n\}$$

and set of ordered pairs of these vertices called *edges*.

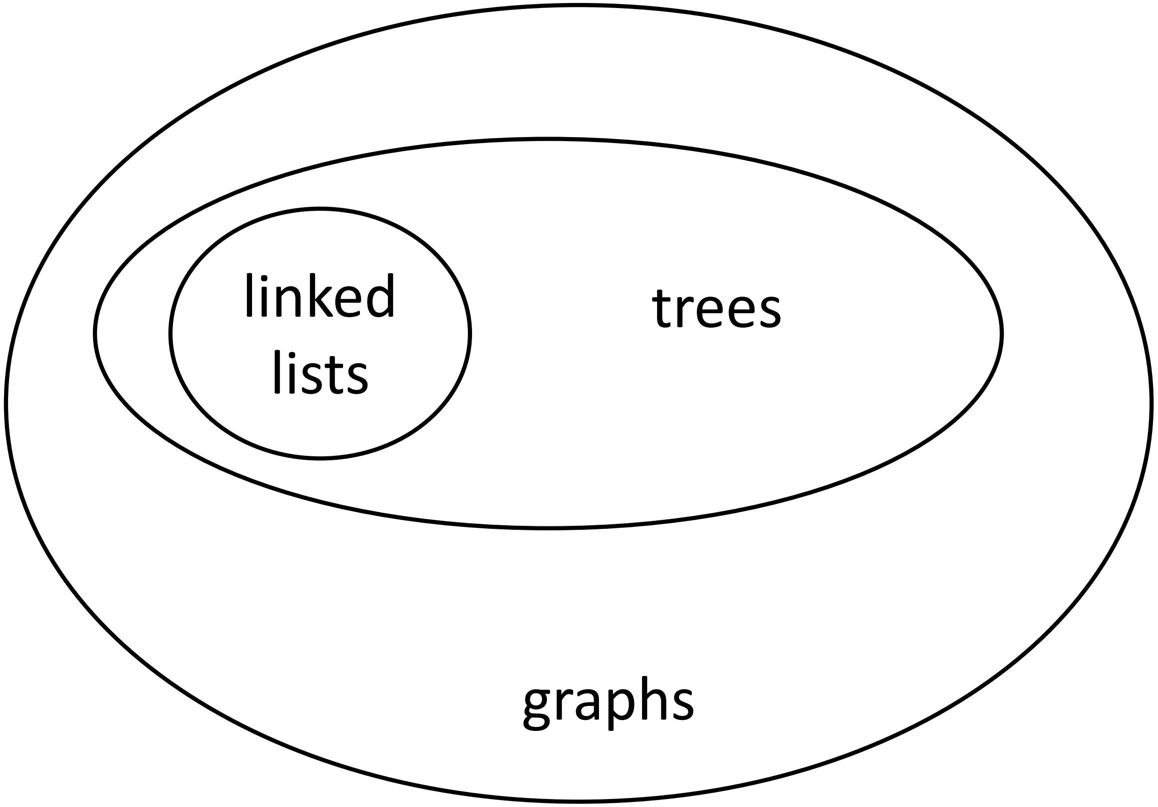
$$E = \{\left(v_i, v_j\right) \;:\; i, j \in 1, \ldots, n\}$$

In an *undirected* graph, the edges are *unordered* pairs.

$$E = \{\;\{v_i, v_j\} \;:\; i, j \in 1, \ldots, n\}$$

Examples

Vertices	Edges
airports	flights
web pages	links (URLs)
Java Objects	references



linked lists

trees

graphs

Data Structures for Graphs

Vertices

- hashmap: label → vertex

```
class  Graph<T>
 {
       HashMap< String, Vertex<T> >    vertexMap;
             :
 }
```

Data Structures for Graphs

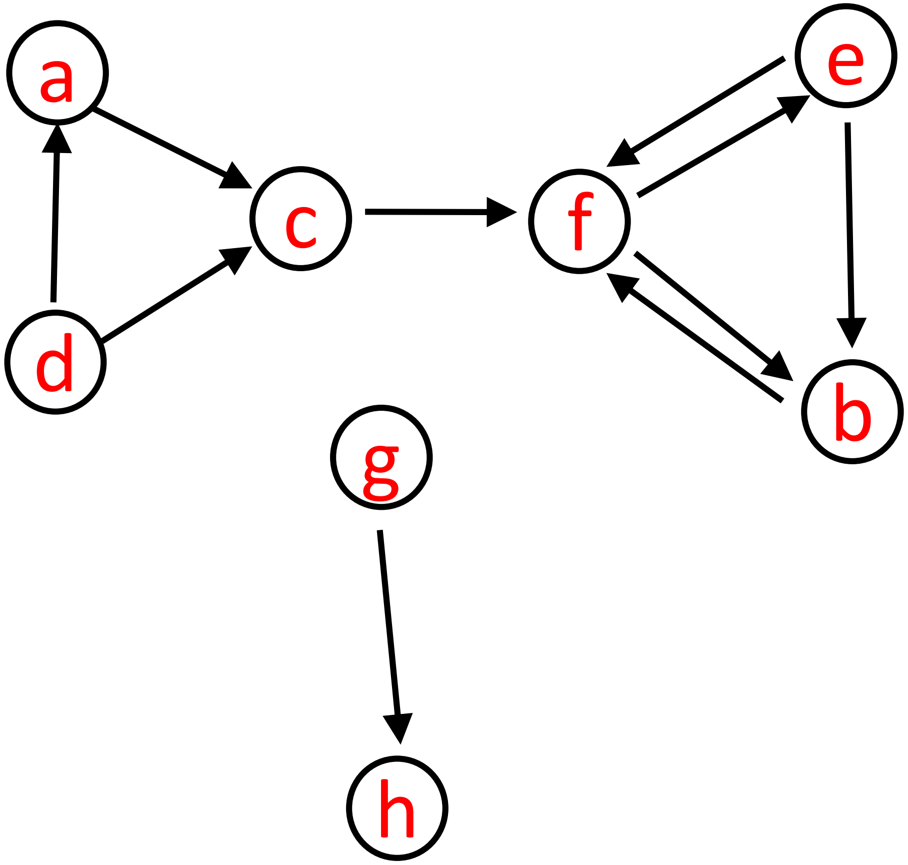
Vertices

- hashmap: label → vertex

Edges

- “adjacency list”
- “adjacency matrix”

Adjacency List



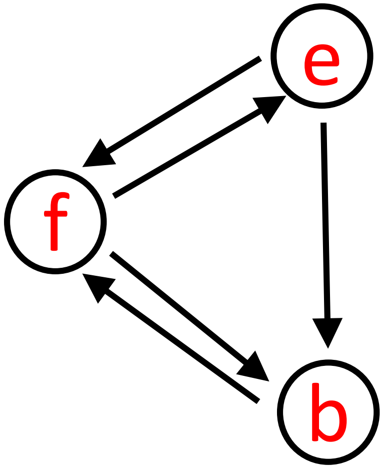
v	**v.adjList**
a	c
b	f
c	f
d	a, c
e	b, f
f	b, e
g	h
h	

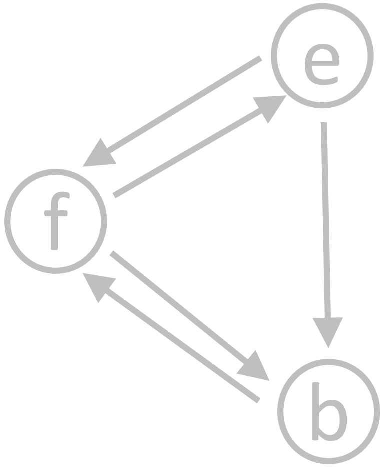
Here each adjacency list is sorted, but that is not always possible (or necessary).

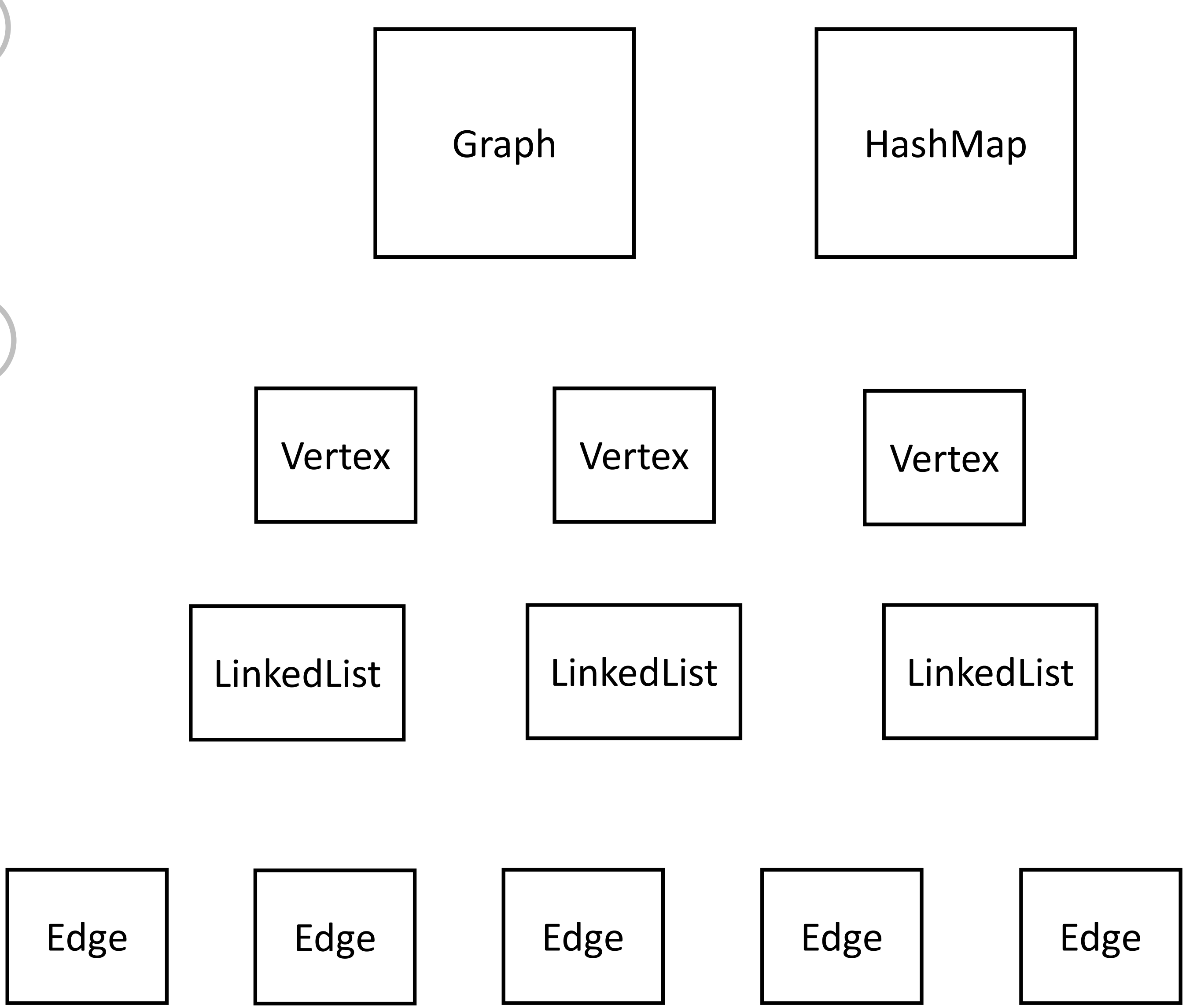
```
class  Graph<T>
 {
      HashMap< String, Vertex<T> >    vertexMap;
           :
}

class Vertex<T>
{
       LinkedList<Edge>    adjList;



}

class Edge
{
        Vertex                   endVertex;


}
```

```
class  Graph<T>
 {
     HashMap< String, Vertex<T> >    vertexMap;
         :
}

class Vertex<T>
{
     LinkedList<Edge>    adjList;
     String               key;
     boolean             visited;
     T                    element;
}

class Edge
{
      Vertex            endVertex;
      double            weight;
          :
}
```

How many objects ?







Graph

HashMap

Vertex

Vertex

Vertex

LinkedList

LinkedList

LinkedList

Edge

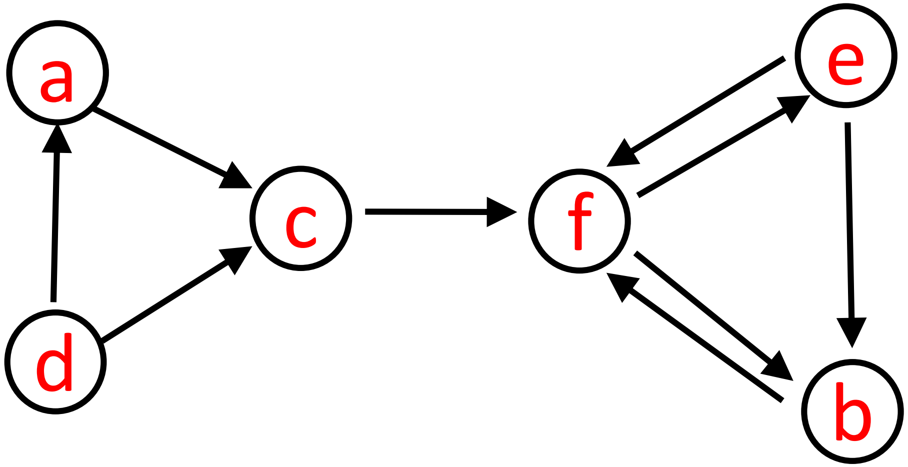
Edge

Edge

Edge

Edge

Adjacency Matrix

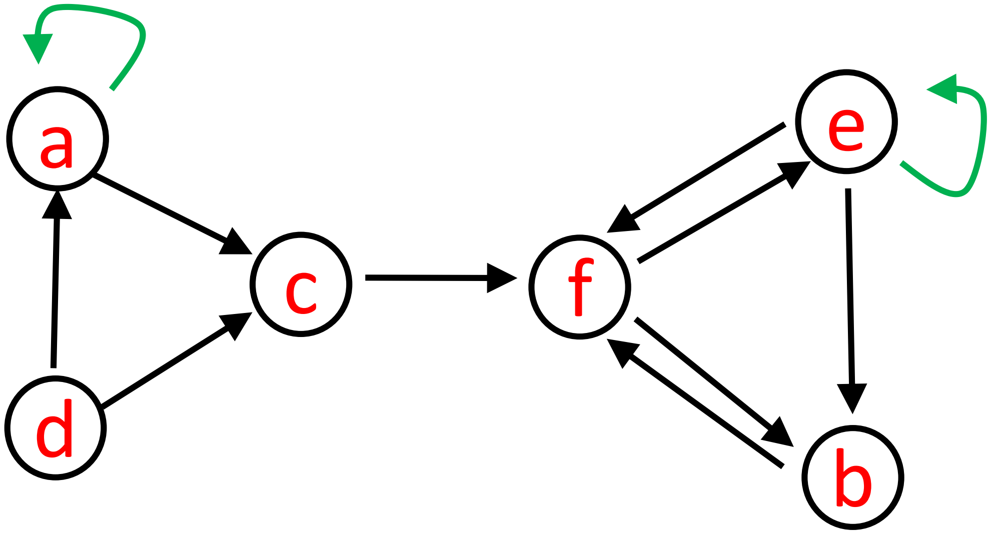


$$\begin{array}{c|cccccc|}
 & a & b & c & d & e & f \\
\hline
a & 0 & 0 & 1 & 0 & 0 & 0 \\
b & 0 & 0 & 0 & 0 & 0 & 1 \\
c & 0 & 0 & 0 & 0 & 0 & 1 \\
d & 1 & 0 & 1 & 0 & 0 & 0 \\
e & 0 & 1 & 0 & 0 & 0 & 1 \\
f & 0 & 1 & 0 & 0 & 1 & 0 \\
\hline
\end{array}$$

boolean adjMatrix[6][6]

Note we require a mapping from vertex names to 0, 1, …. , n-1.

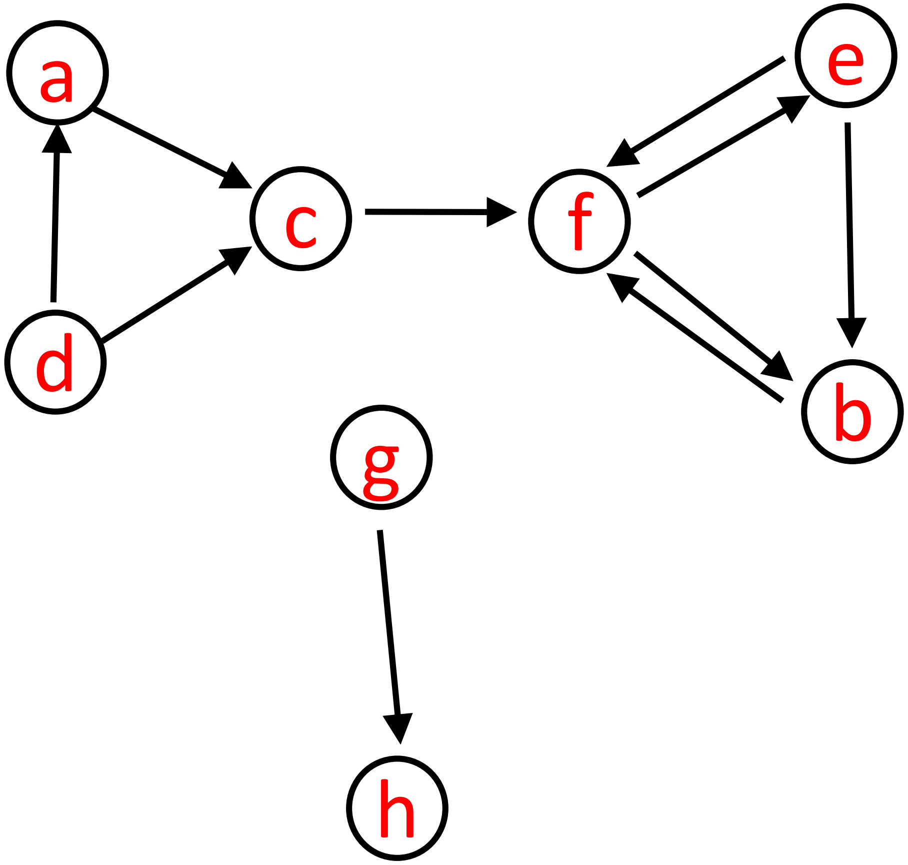
Adjacency Matrix



$$
\begin{array}{c|cccccc|}
 & a & b & c & d & e & f \\
\hline
a & 1 & 0 & 1 & 0 & 0 & 0 \\
b & 0 & 0 & 0 & 0 & 0 & 1 \\
c & 0 & 0 & 0 & 0 & 0 & 1 \\
d & 1 & 0 & 1 & 0 & 0 & 0 \\
e & 0 & 1 & 0 & 0 & 1 & 1 \\
f & 0 & 1 & 0 & 0 & 1 & 0 \\
\hline
\end{array}
$$

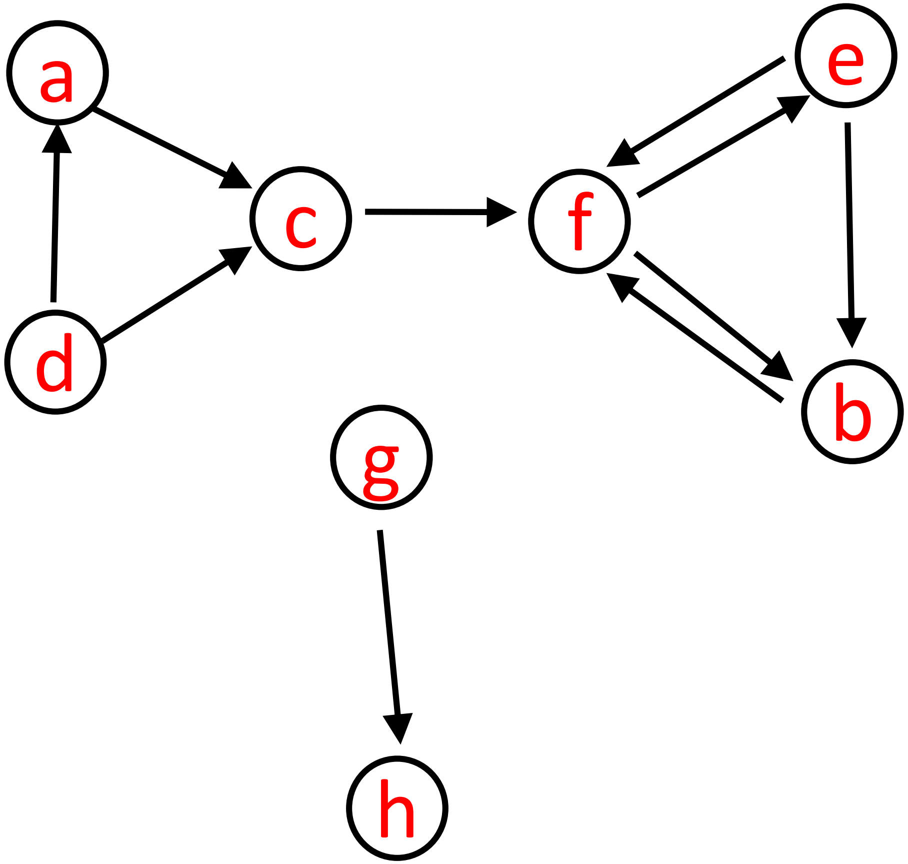
boolean adjMatrix[6][6]

Terminology: “in degree”



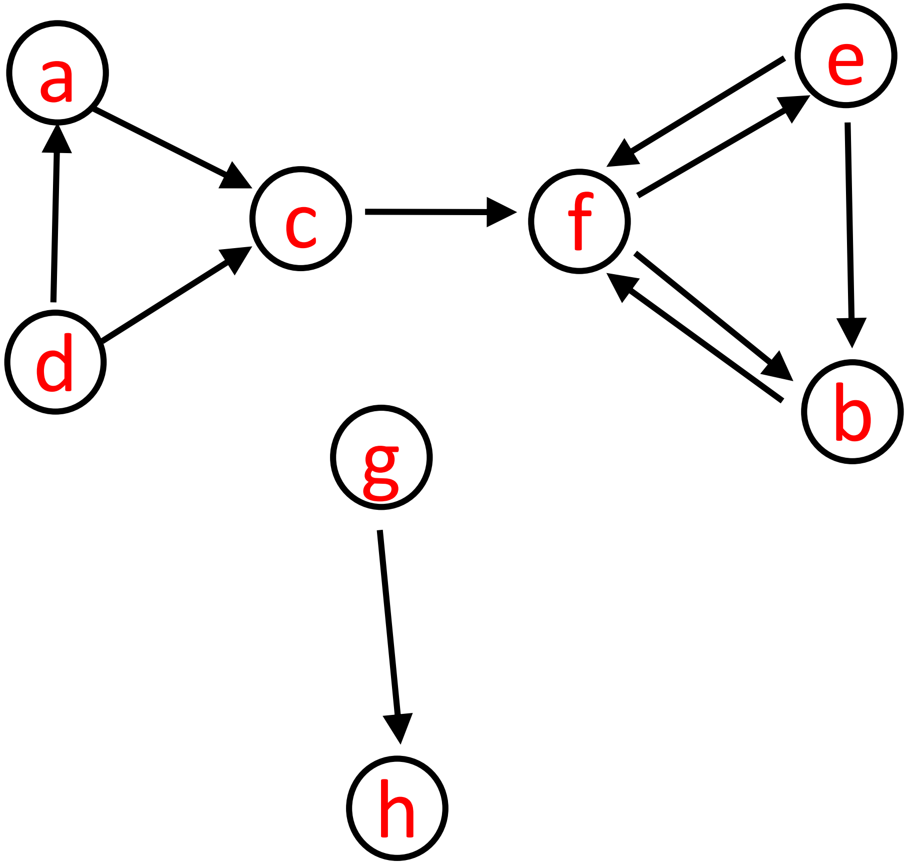
v	in degree
a	1
b	2
c	2
d	0
e	1
f	3
g	0
h	1

Terminology: “out degree”



v	**out degree**
a	1
b	1
c	1
d	2
e	2
f	2
g	1
h	0

Terminology: path



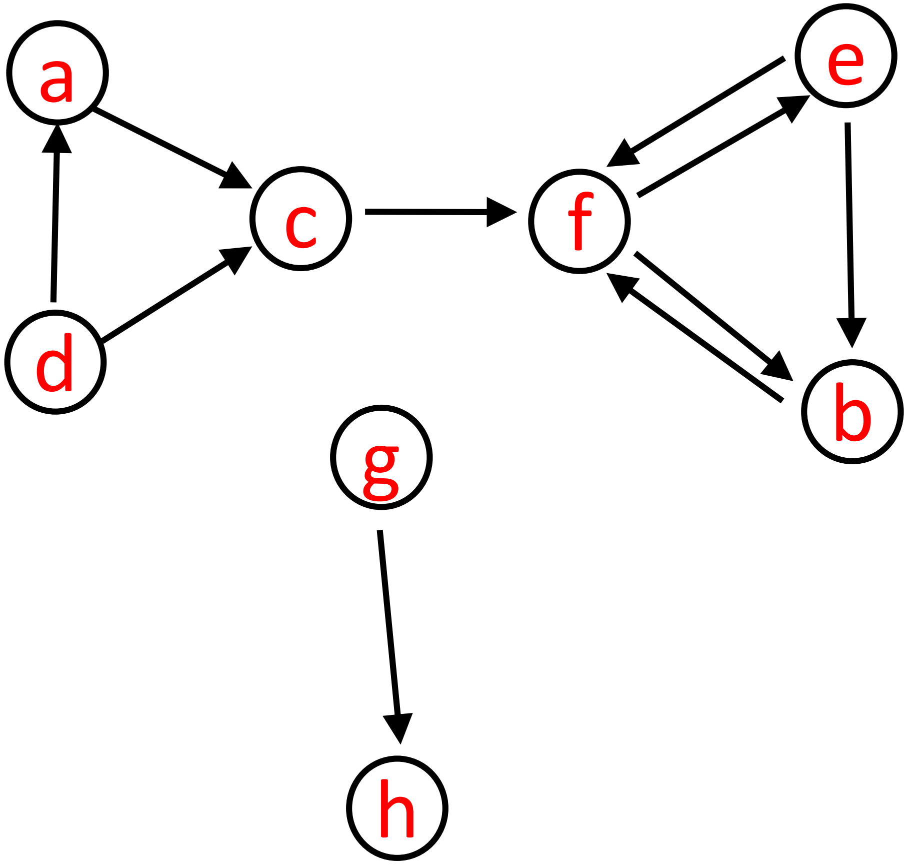
A sequence of vertices that are connected by edges.

Equivalently, a sequence of edges such that end vertex of one edge is the start vertex of next edge.)

Examples

- acfeb
- dac
- febf
- …..

Terminology: cycle



A path such that the last vertex is the same as the first vertex.

Examples

- febf
- efe
- fbf
- …

Graph algorithms in COMP 251

Given a graph, what is the shortest (weighted) path between two vertices?

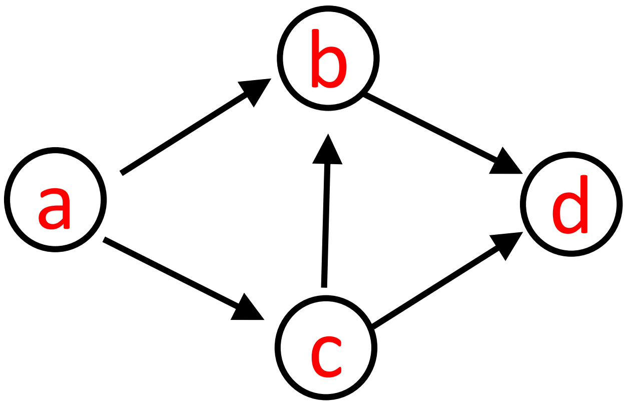
Given a graph and weights that describe flow capacities, what is the maximum flow between two given vertices ?

Given two sets X and Y, and each ranks the members of the other set, find a stable “match”.

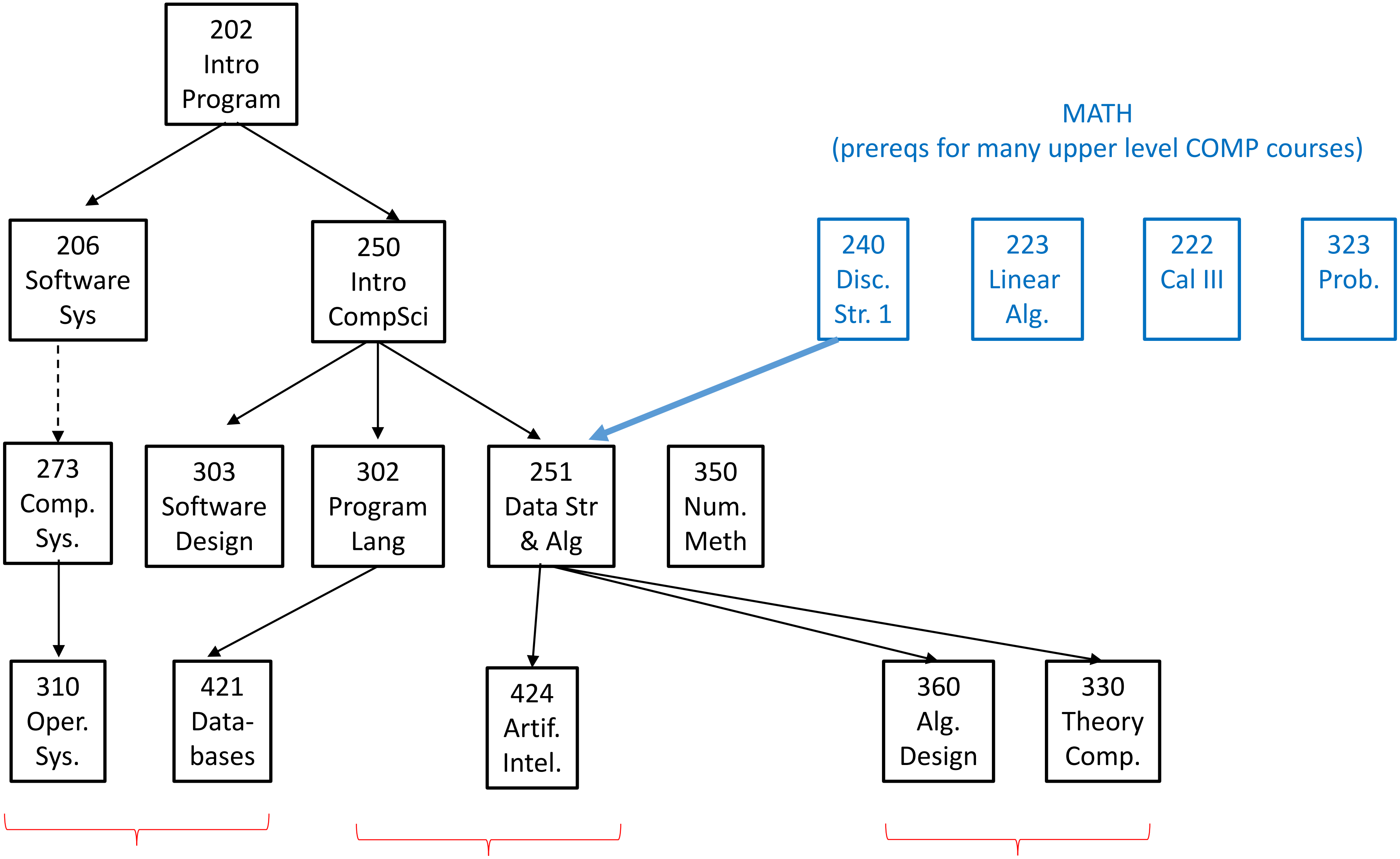
Directed *Acyclic* Graph

no cycles

Used to capture dependencies.



There are three paths from a to d.



MATH
(prereqs for many upper level COMP courses)

202 Intro Program

206 Software Sys

250 Intro CompSci

240 Disc. Str. 1

223 Linear Alg.

222 Cal III

323 Prob.

273 Comp. Sys.

303 Software Design

302 Program Lang

251 Data Str & Alg

350 Num. Meth

310 Oper. Sys.

421 Data-bases

424 Artif. Intel.

360 Alg. Design

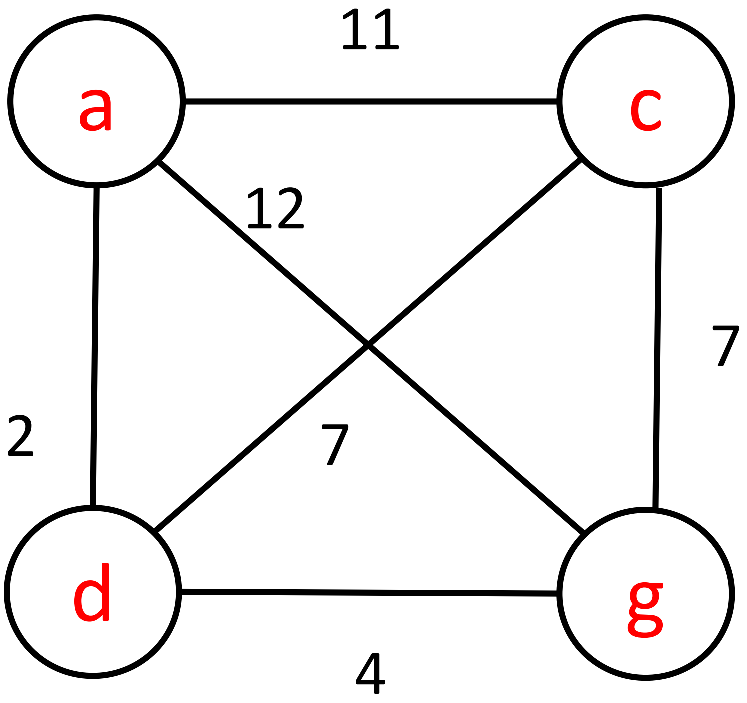
330 Theory Comp.

SYSTEMS
(compilers, networks,
distributed sys,
concurrency, web,..)

APPLICATIONS
(graphics, vision,
bioinf, games,
machine learning..)

THEORY
(crypto, optimization, game theory,
logic, correctness, computability..)

“Travelling Salesman” COMP 360
(Hamiltonian circuit)



Find the shortest cycle that visits all vertices once.

If there is an edge between each pair of vertices, then how many cycles are there?

Announcements

- Midterm 2 on Monday

STBIO	S3/3	[A-C]
ENGTR	0100	[D-J]
ADAMS	AUD	[K-W]
RPHYS	114	[X-Z]

covers from lecture 10-24

- Assignment 4 will be posted next week