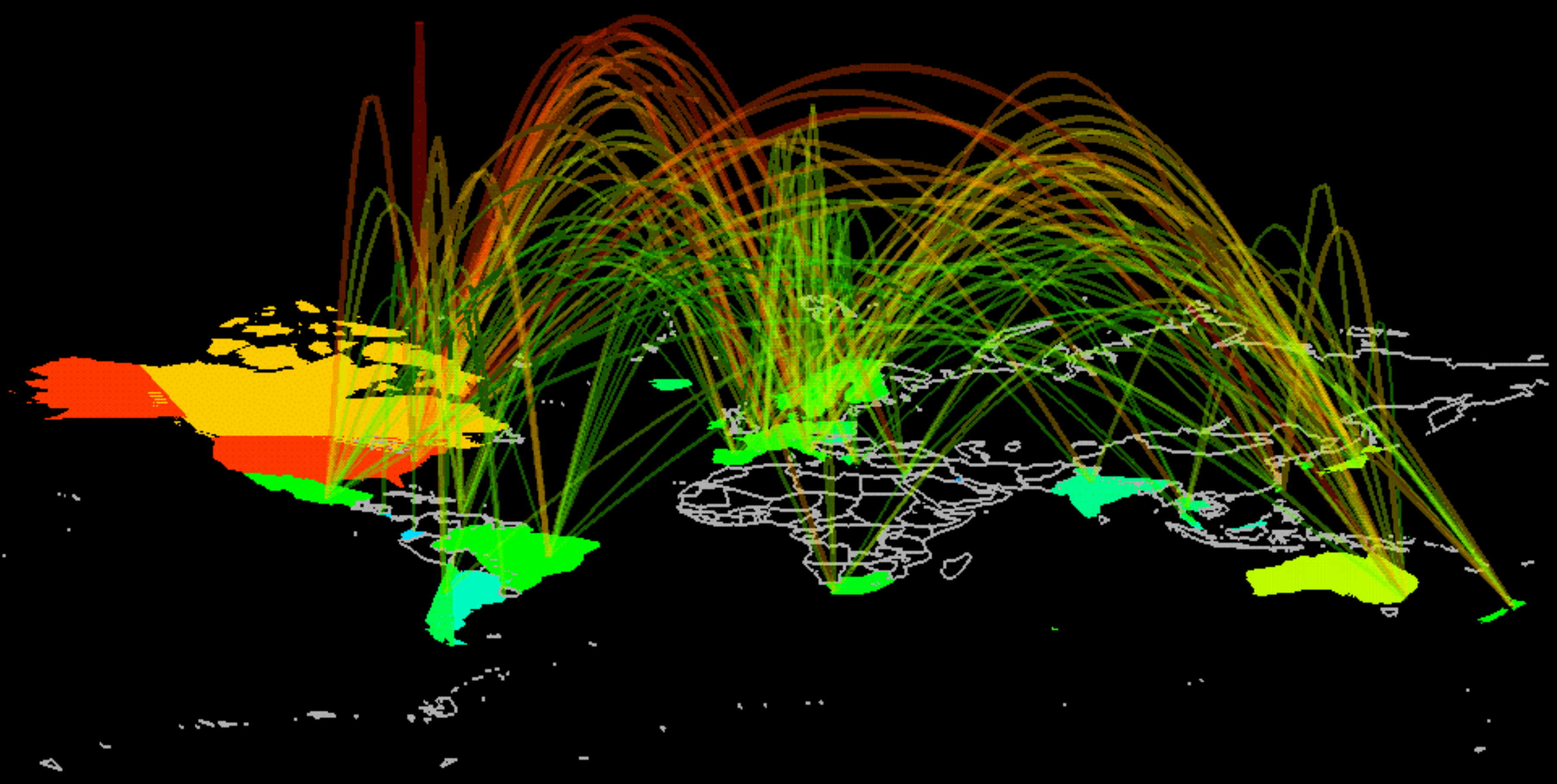
Module #7a:
Theory of Network Visualization

Objective

- Define hierarchical data and related terms
- List example tasks for hierarchical and network data
- Understand approaches to draw 2D trees
- Describe treemap, SunBurst, and other techniques

JOHNS HOPKINS UNIVERSITY | Engineering for Professionals

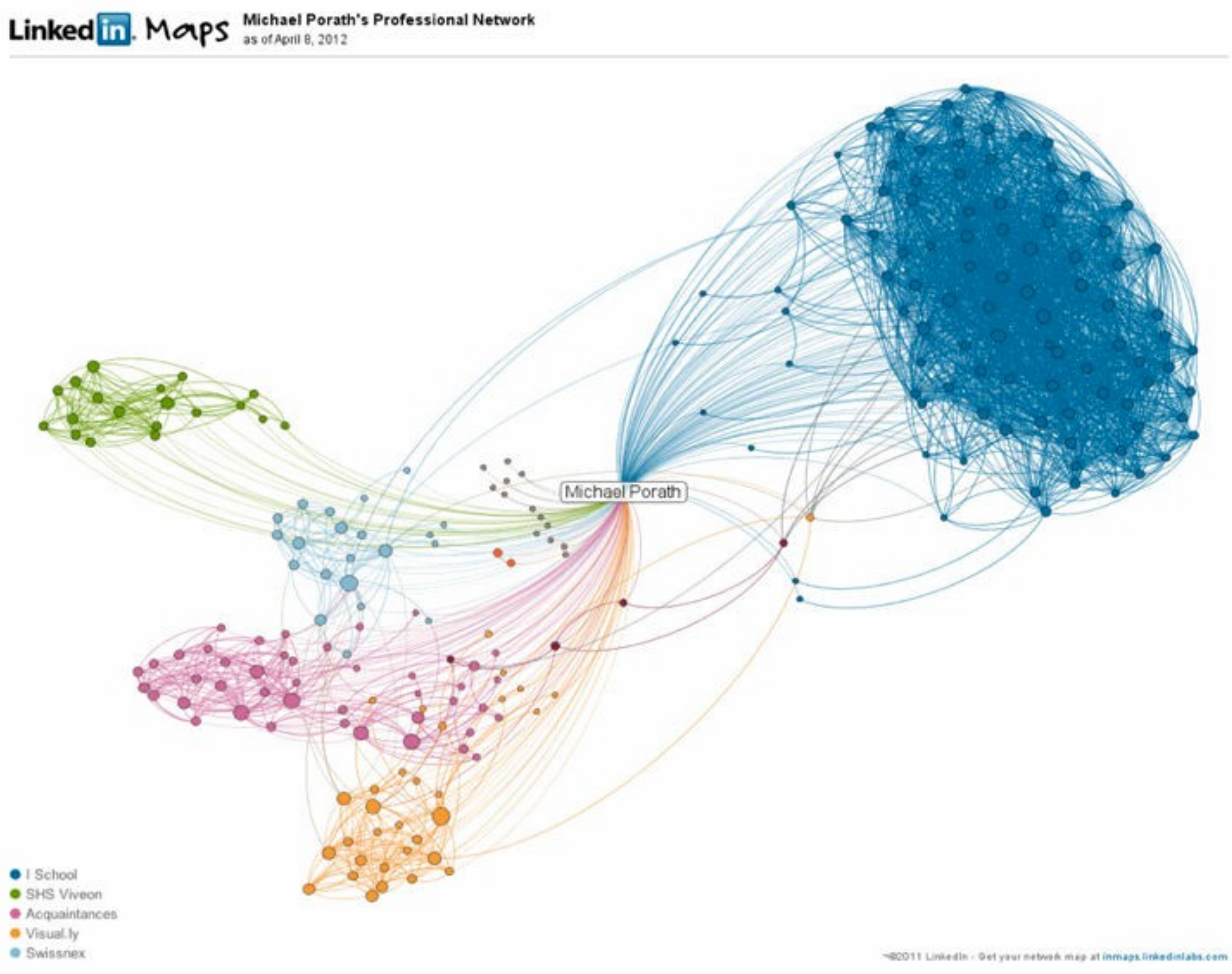
Connected World



Stephen G. Eick et al - Bell Laboratories

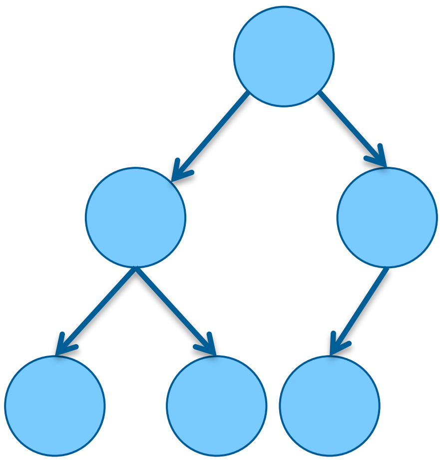
Social Networks

Social Networks

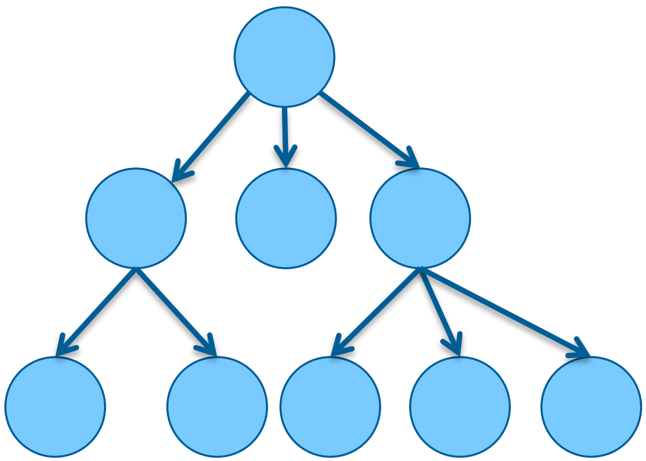


http://inmaps.linkedinlabs.com/

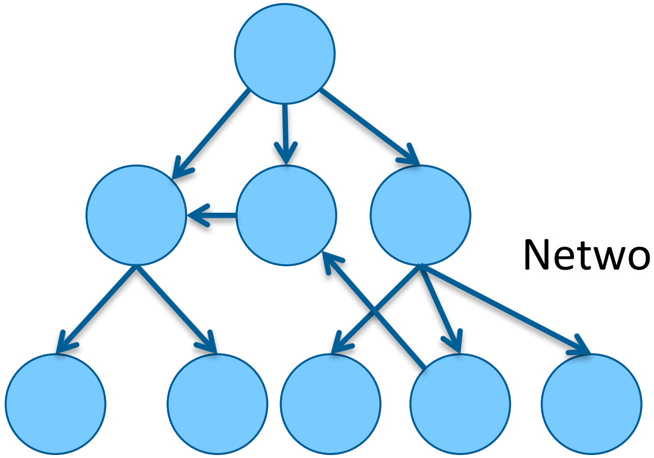
Properties of networks



Tree

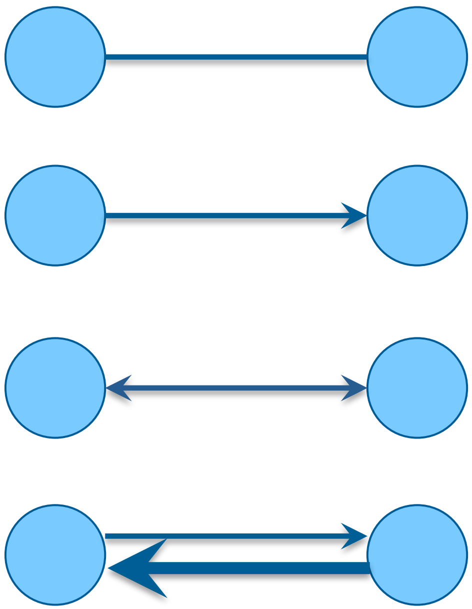


Tree



Network

Properties



Undirected edges

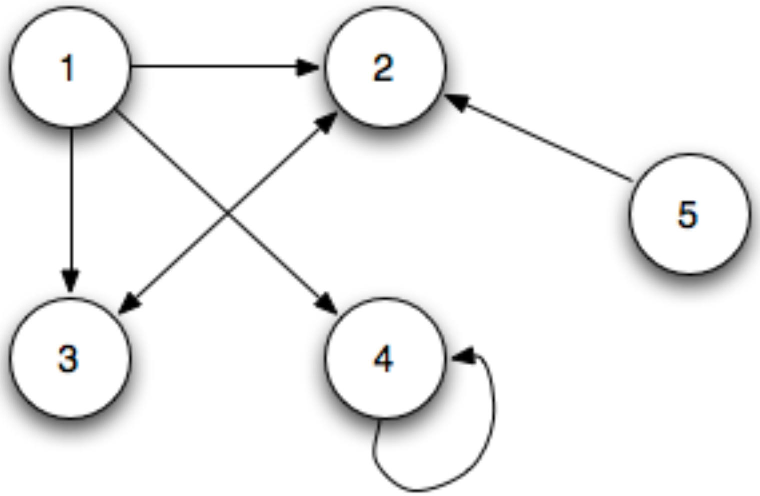
Directed edges

Bi-directional, symmetrical edges

Bi-directional, asymmetrical edges

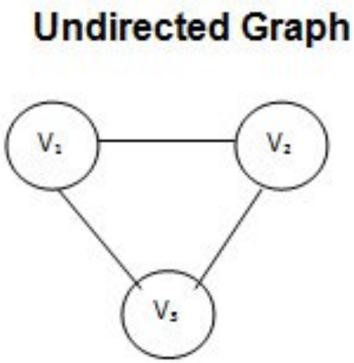
Graph terminology I

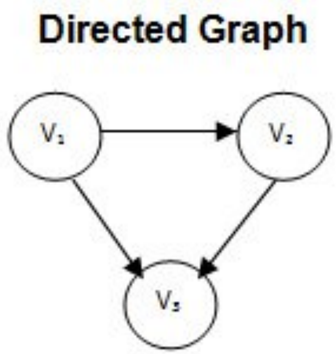
- A graph is a collection of nodes (or vertices, singular is vertex) and edges (or arcs)
  - Each node contains an element
  - Each edge connects two nodes together (or possibly the same node to itself) and may contain an edge attribute



Graph terminology I

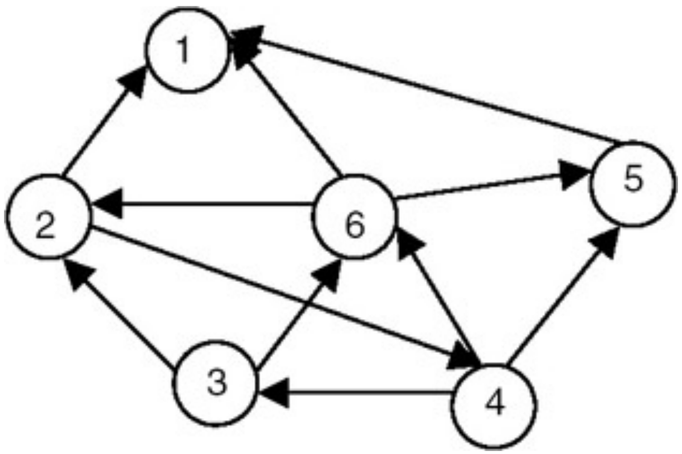
- A directed graph is one in which the edges have a direction
- An undirected graph is one in which the edges do not have a direction





Graph terminology II

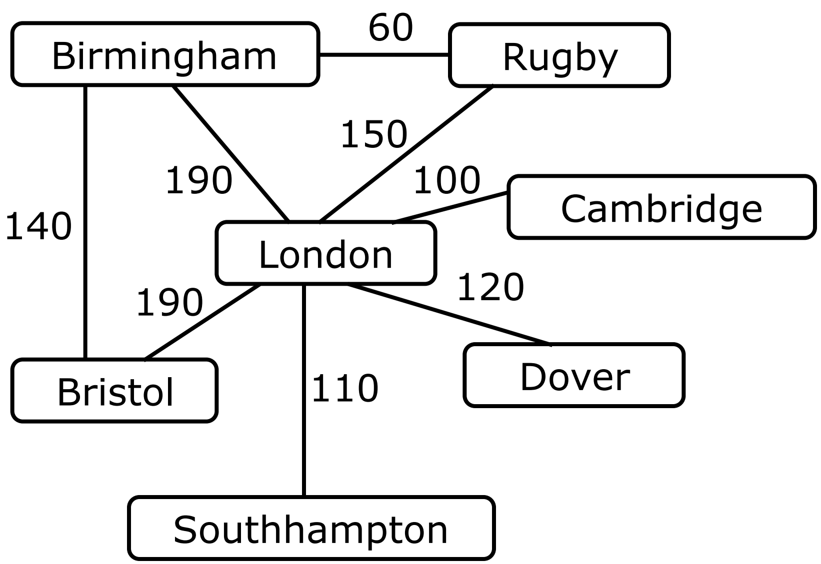
- The size of a graph is the number of *nodes* in it
- If two nodes are connected by an edge, they are neighbors (and the nodes are adjacent to each other)
- The degree of a node is the number of edges it has
- For directed graphs,
  - The in-degree of a node is the number of in-edges it has
  - The out-degree of a node is the number of out-edges it has



	Indegree	Outdegree
1	3	0
2	2	2
3	1	2
4	1	3
5	2	1
6	2	3

Graph terminology III

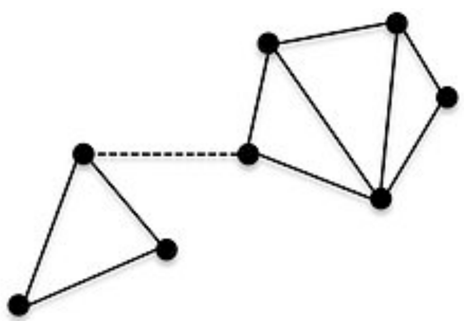
- A path is a list of edges such that each node (but the last) is the predecessor of the next node in the list
- A cycle is a path whose first and last nodes are the same

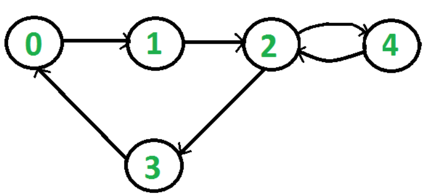


- Example: (London, Bristol, Birmingham, London, Dover) is a path
- Example: (London, Bristol, Birmingham, London) is a cycle
- A cyclic graph contains at least one cycle
- An acyclic graph does not contain any cycles

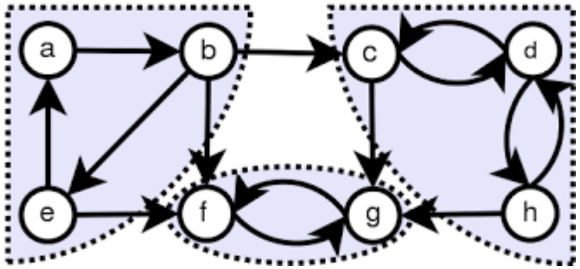
Graph terminology IV

- An undirected graph is connected if there is a path from every node to every other node
- A *directed graph* is strongly connected if there is a path from every node to every other node
- Node X is reachable from node Y if there is a path from Y to X
- A subset of the nodes of the graph is a connected component (or just a component) if there is a path from every node in the subset to every other node in the subset

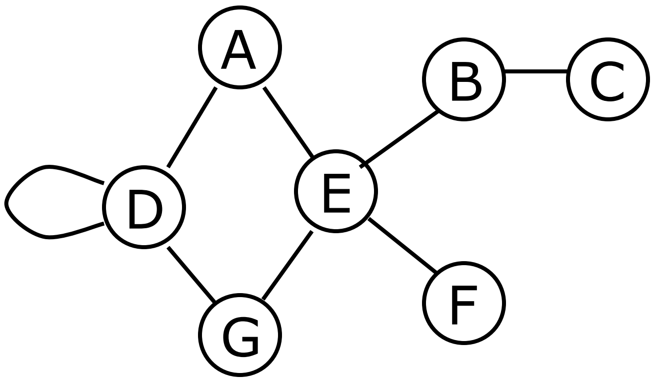




Strongly Connected



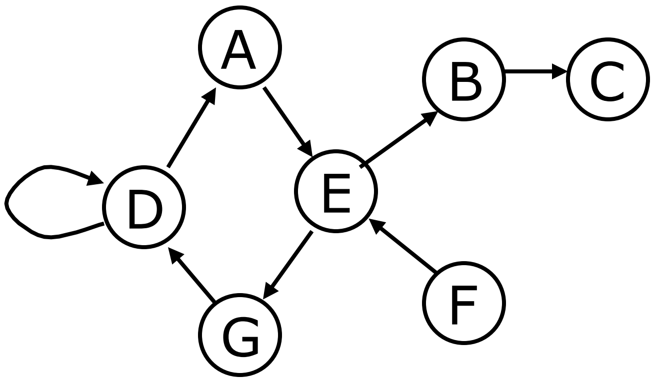
Adjacency-matrix representation I



	A	B	C	D	E	F	G
A				●	●		
B			●		●		
C		●					
D	●			●			●
E	●	●				●	●
F					●		
G				●	●		

- One simple way of representing a graph is the adjacency matrix
- A 2-D array has a mark at `[i][j]` if there is an edge from node `i` to node `j`
- The adjacency matrix is symmetric about the main diagonal

Adjacency-matrix representation II



	A	B	C	D	E	F	G
A					●		
B			●				
C							
D	●			●			
E		●					●
F					●		
G				●			

- An adjacency matrix can equally well be used for digraphs (directed graphs)
- A 2-D array has a mark at `[i][j]` if there is an edge from node `i` to node `j`
- Only suitable for *small* graphs!

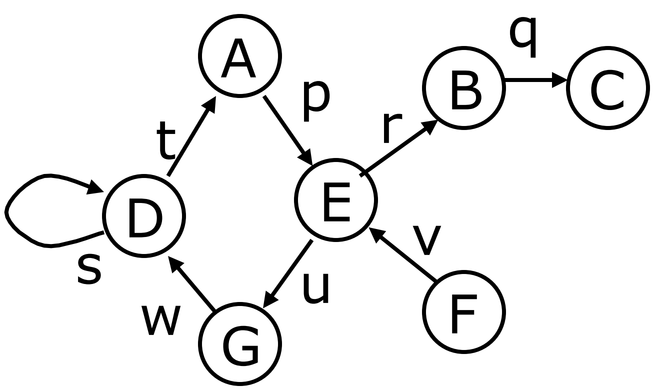
Pros and Cons of Adjacency Matrices

- Pros:
  - Simple to implement
  - Easy and fast to tell if a pair (i,j) is an edge: simply check if A[i][j] is 1 or 0
- Cons:
  - No matter how few edges the graph has, the matrix takes $O(n^2)$ in memory

Edge-set representation I

- An edge-set representation uses a *set* of nodes and a *set* of edges
  - The sets might be represented by, say, linked lists
  - The set links are stored in the nodes and edges themselves

Edge-set representation II



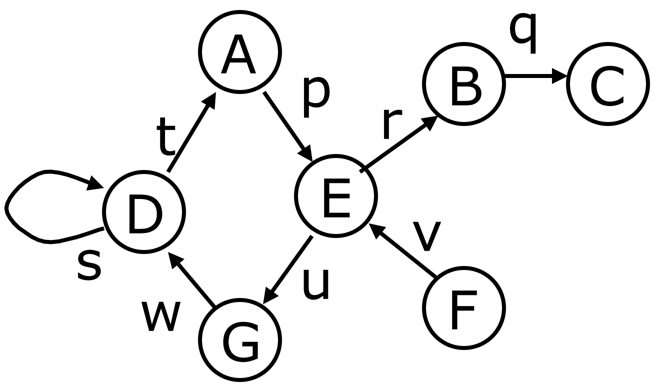
nodeSet = {A, B, C, D, E, F, G}

edgeSet = { p: (A, E),
q: (B, C), r: (E, B),
s: (D, D), t: (D, A),
u: (E, G), v: (F, E),
w: (G, D) }

- Here we have a set of nodes, and each node contains only its element (not shown)

- Each edge contains references to its source and its destination (and its attribute, if any)

Adjacency-set representation II

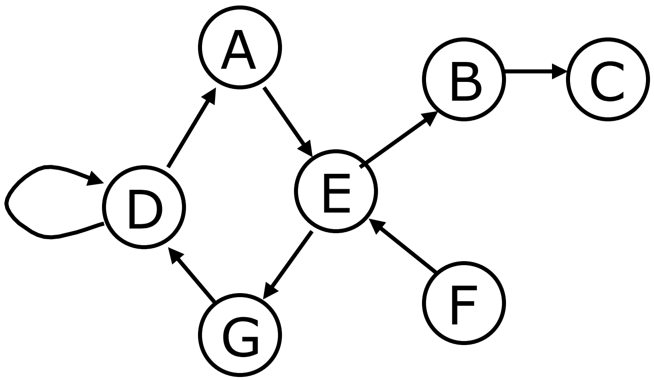


{ A → { p }, B → { q }, C → { }, D → { s, t }, E → { r, u }, F → { v }, G → { w } }

p: (A, E)
q: (B, C)
r: (E, B)
s: (D, D)
t: (D, A)
u: (E, G)
v: (F, E)
w: (G, D)

- Here we have a set of nodes, and each node refers to a set of edges
- Each edge contains references to its source and its destination (and its attribute, if any)

Adjacency-set representation II

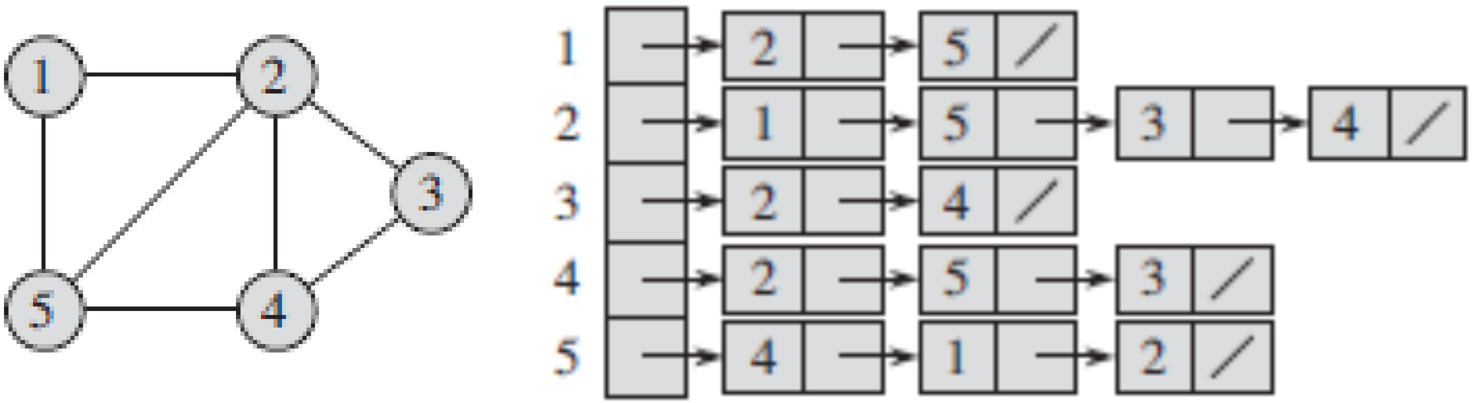


A → { E }
B → { C }
C → { }
D → { D, A }
E → { B, G }
F → { E }
G → { D }

- Here we have a set of nodes, and each node refers to a set of other (pointed to) nodes
- The edges are *implicit*

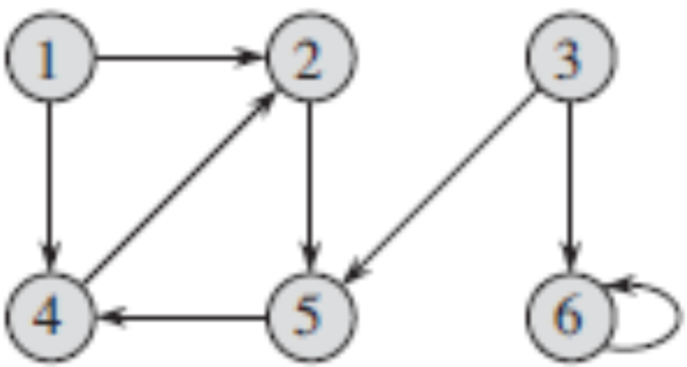
Graph representations: Adjacency List

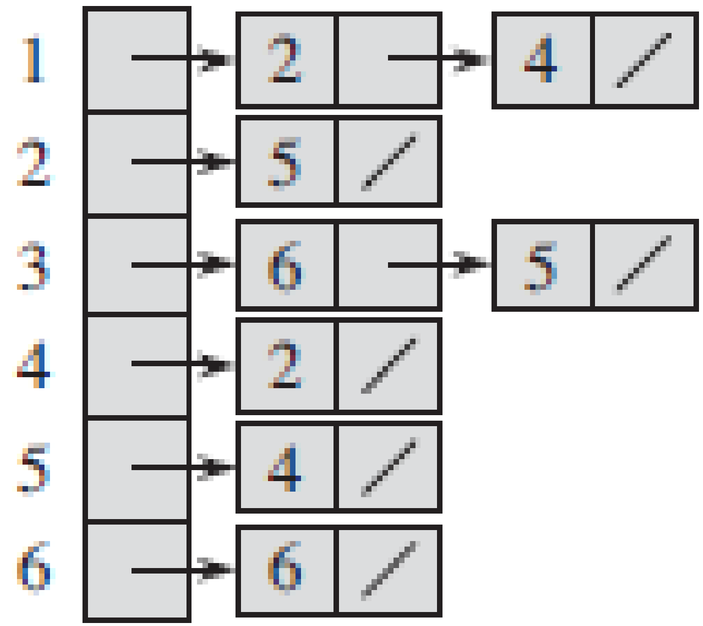
- **Undirected** weighted graph



Graph representations: Adjacency List

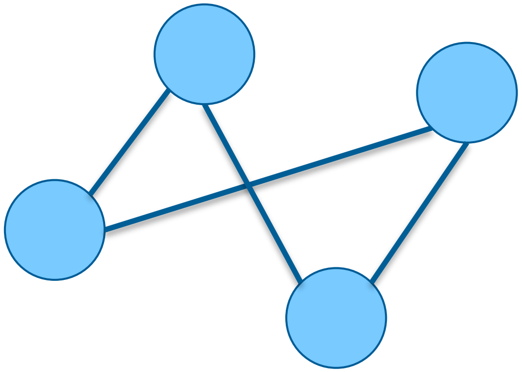
- **Directed** weighted graph

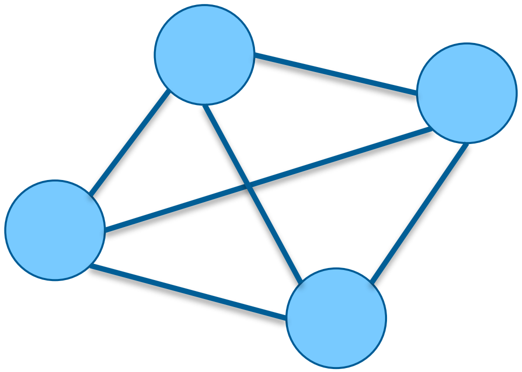




Topology: Planar graphs

Graphs that can be laid out without edge crossing

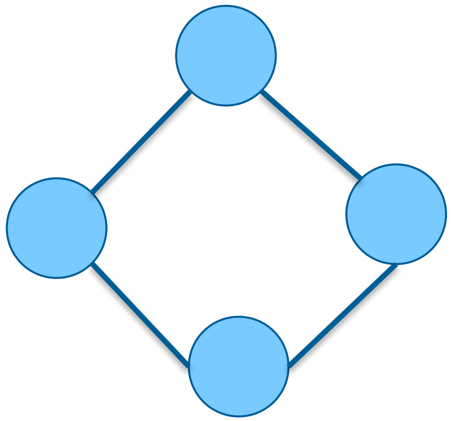


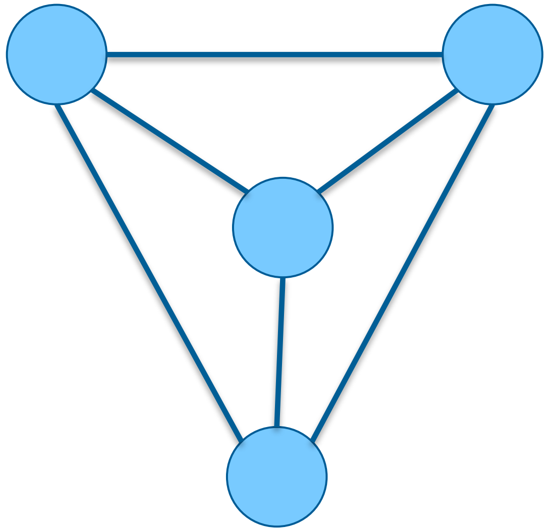


Which on is planar?

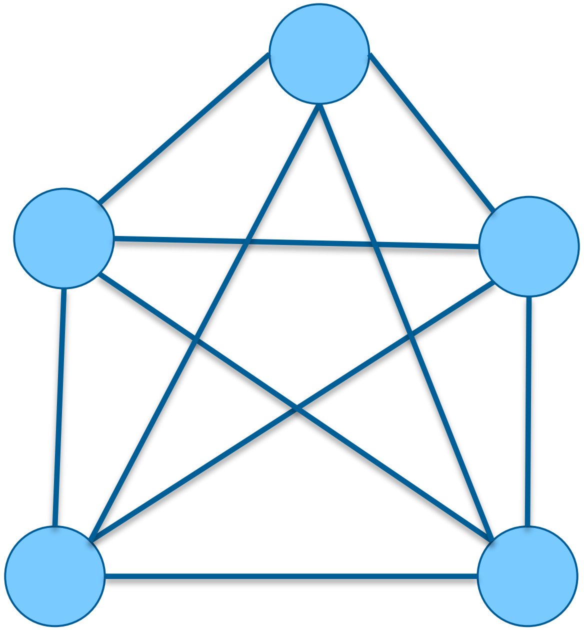
Topology: Planar graphs

Graphs that can be laid out without edge crossing





Topology: Planar graphs



Non-planar

JOHNS HOPKINS
WHITING SCHOOL of ENGINEERING

© The Johns Hopkins University 2016, All Rights Reserved.