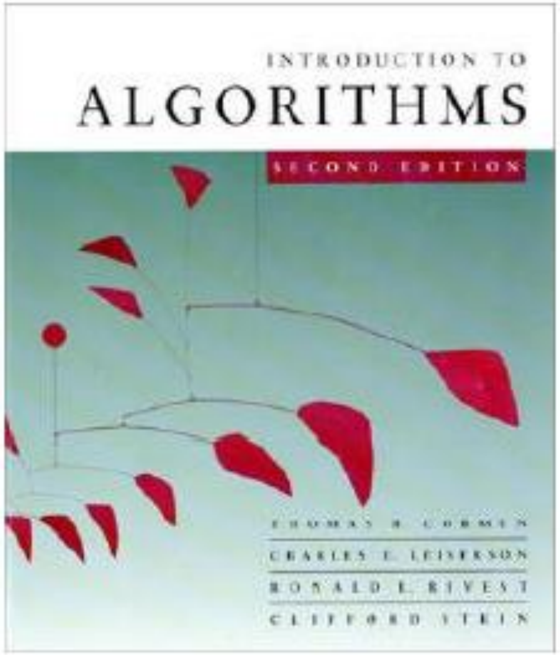
Introduction to Algorithms



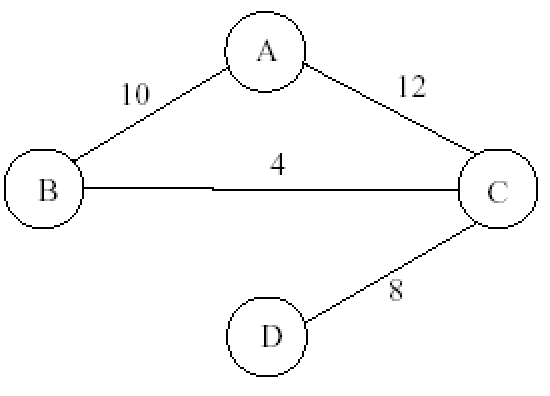
Chapter 22: Elementary Graph Algorithms

Graph Terminology

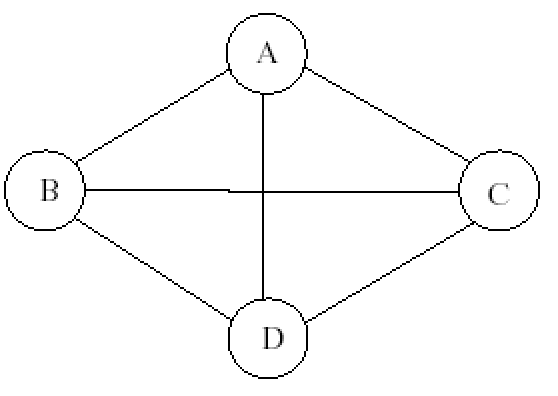
- A graph $G = (V, E)$
  - V = set of vertices
  - E = set of edges
- In an *undirected graph:*
  - $edge(u, v) = edge(v, u)$
- In a *directed* graph:
  - $edge(u,v)$ goes from vertex u to vertex v, notated $u \rightarrow v$
  - $edge(u, v)$ is not the same as $edge(v, u)$

Graph Terminology

- **If each edge in the graph carries a value, then the graph is called** *weighted graph***.**
  - **A weighted graph is a graph $G = (V, E, W)$, where each edge, $e \in E$ is assigned a real valued weight, $W(e)$.**
- **A** *complete graph* **is a graph with an edge between every pair of vertices.**
  - **A graph is called** *complete graph* **if every vertex is a**



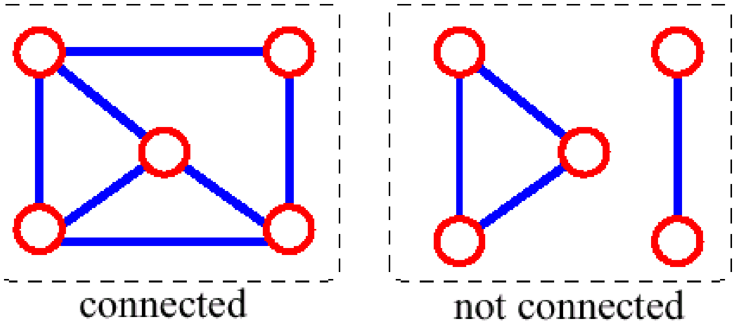
weighted graph



complete graph

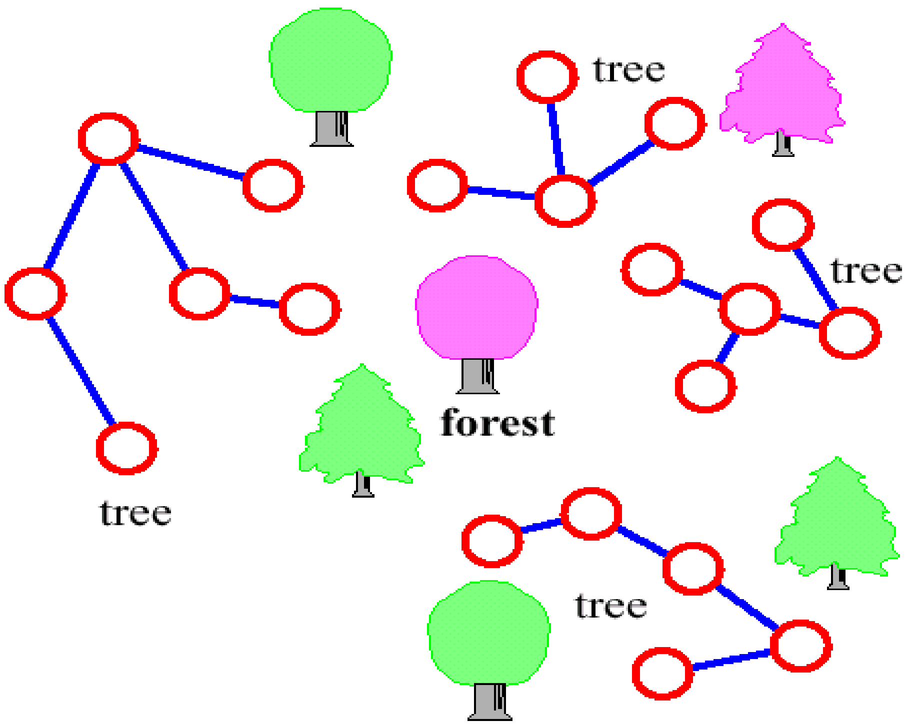
Graph Terminology

- **connected graph:** **any two vertices are connected by some path**
  - **An undirected graph is** *connected* **if, for every pair of vertices** ***u*** **and** ***v*** **there is a path from** ***u*** **to** ***v*****.**

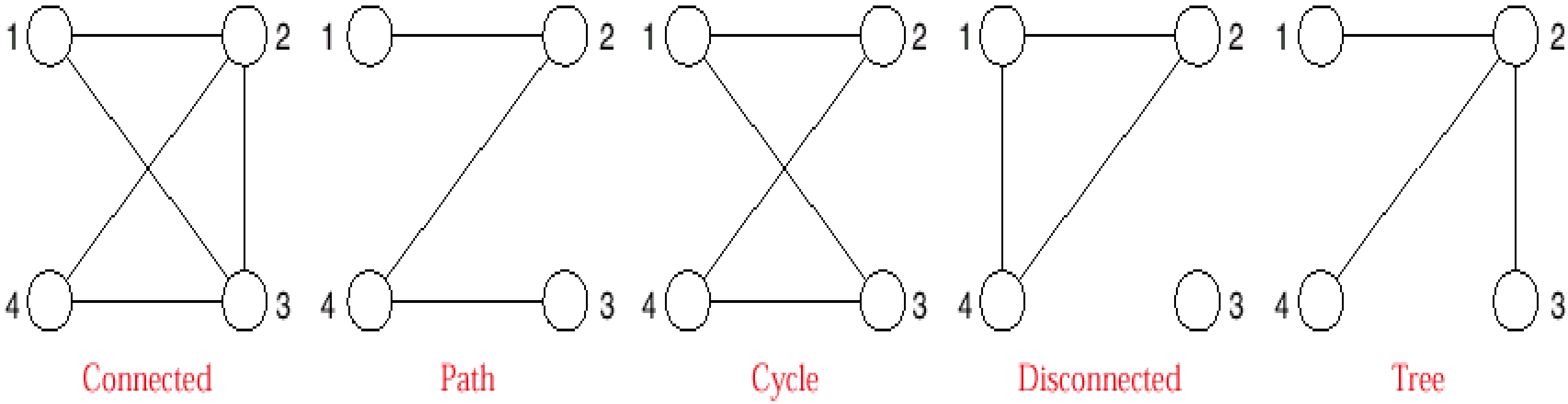


Graph Terminology

- **tree** - connected graph without cycles
- **forest** - collection of trees



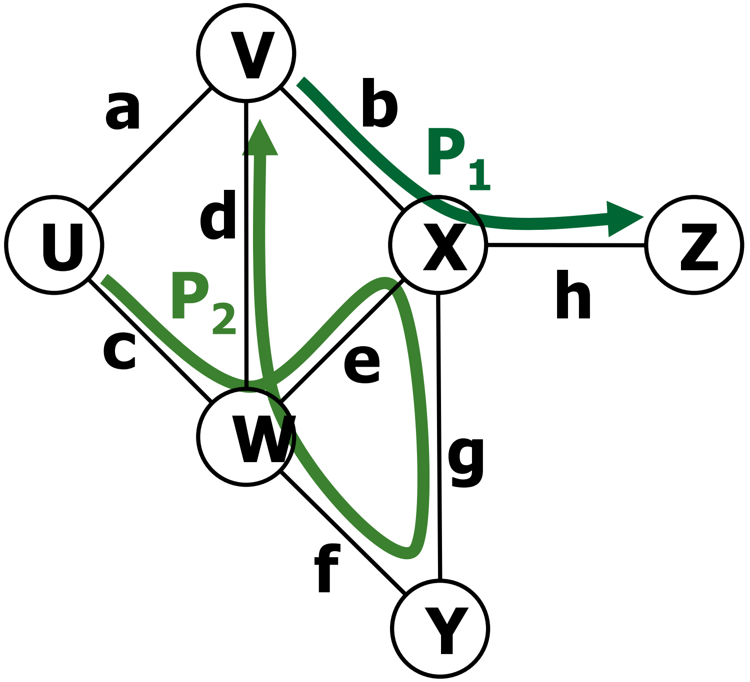
Graph Terminology



Graph Terminology

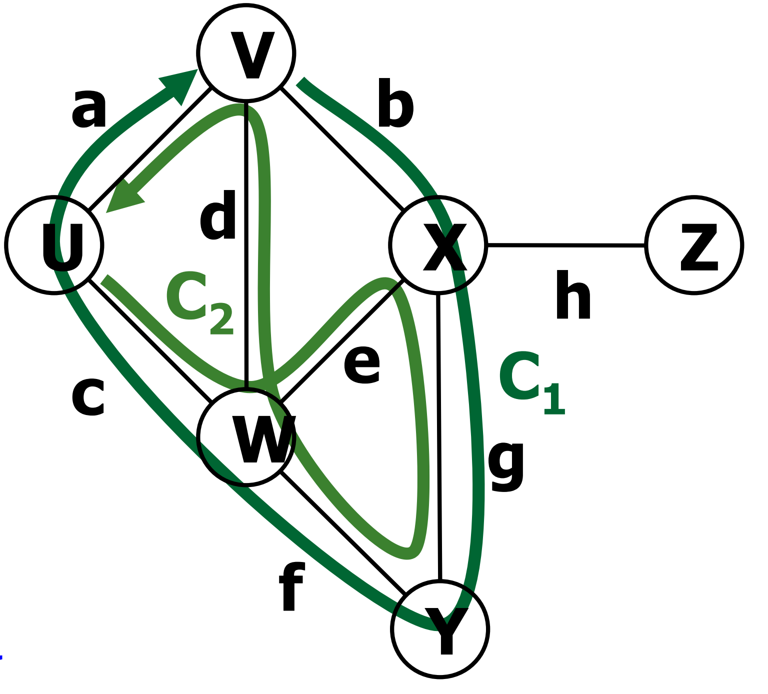
- Path
  - **sequence of alternating vertices and edges**
  - **begins with a vertex**
  - **ends with a vertex**

- Simple path
  - **path such that all its vertices and edges are** distinct**.**
- Examples
  - **P_1 = (V, X, Z) is a simple path.**
  - **P_2 = (U, W, X, Y, W, V) is a path that is not simple.**



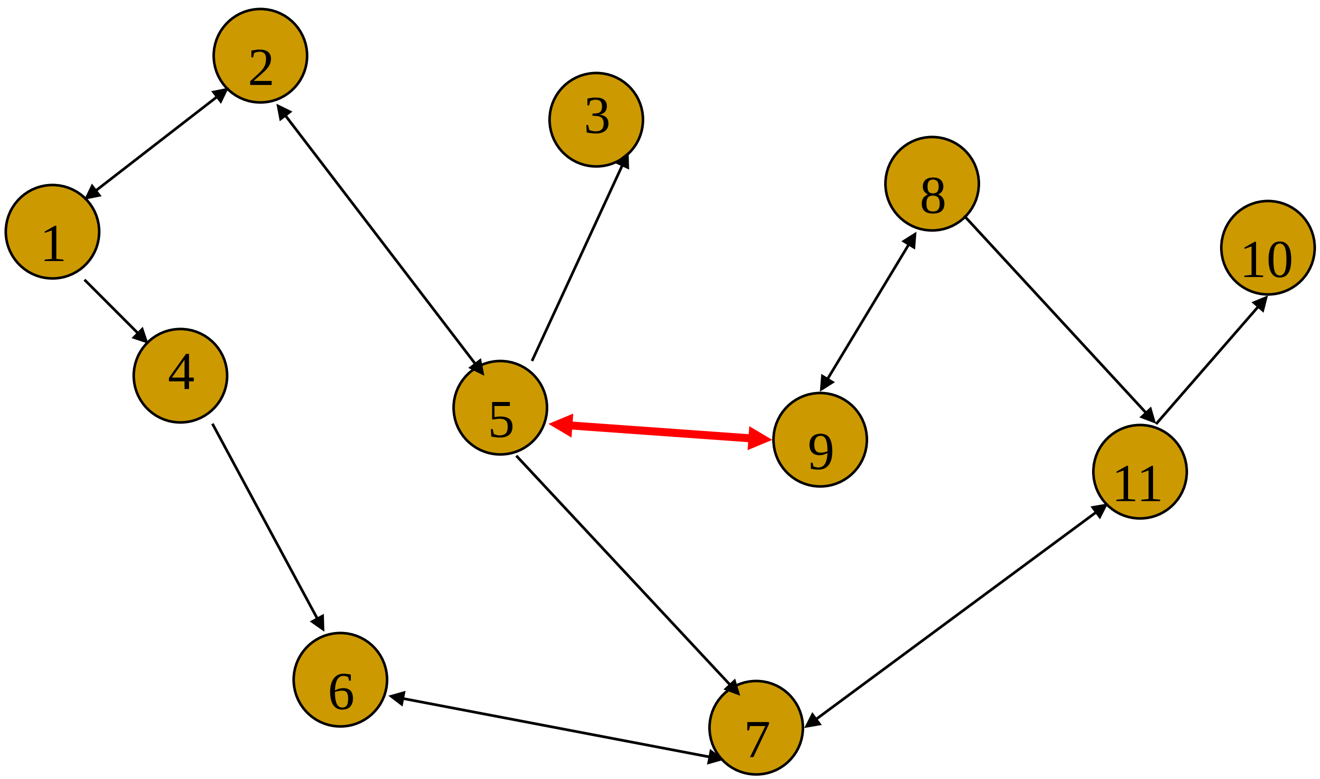
Graph Terminology

- Cycle
  - **circular sequence of alternating vertices and edges**
- **Simple cycle**
  - **cycle such that all its vertices and edges are** distinct
- **Examples**
  - **C_1 = (V, X, Y, W, U, V) is a simple cycle**
  - **C_2 = (U, W, X, Y, W, V, U) is a cycle that is not simple**



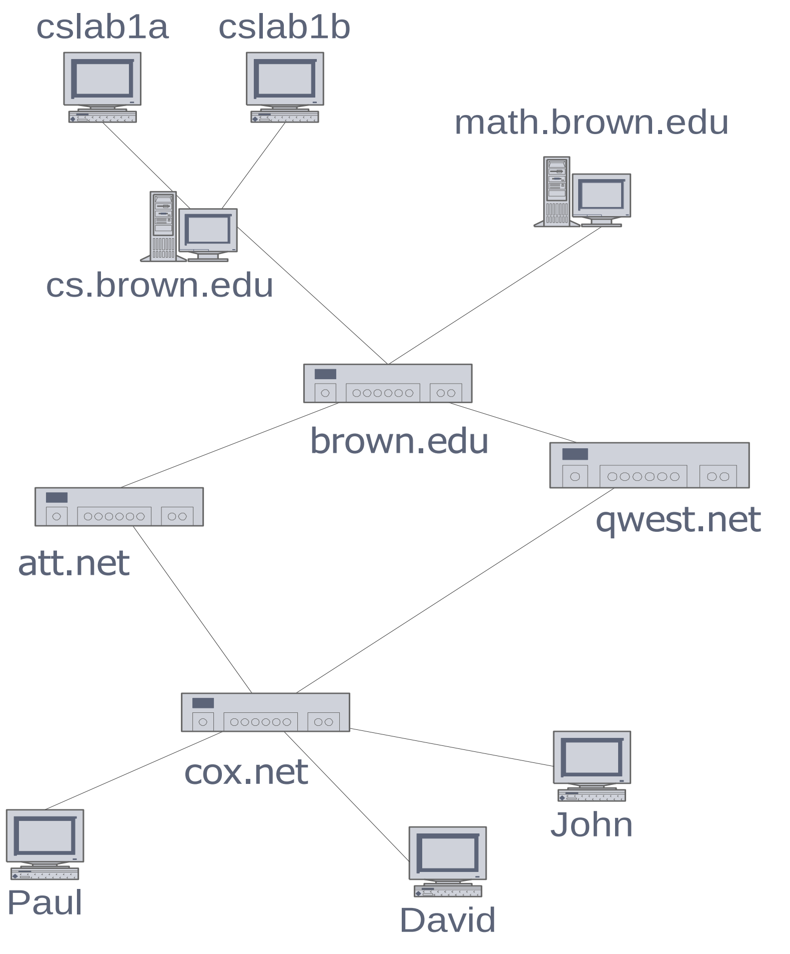
Street Map

- **Some streets are one way**
- **A *bidirectional* link represented by 2 directed edge**
  - **(5, 9) (9, 5)**



Computer Networks

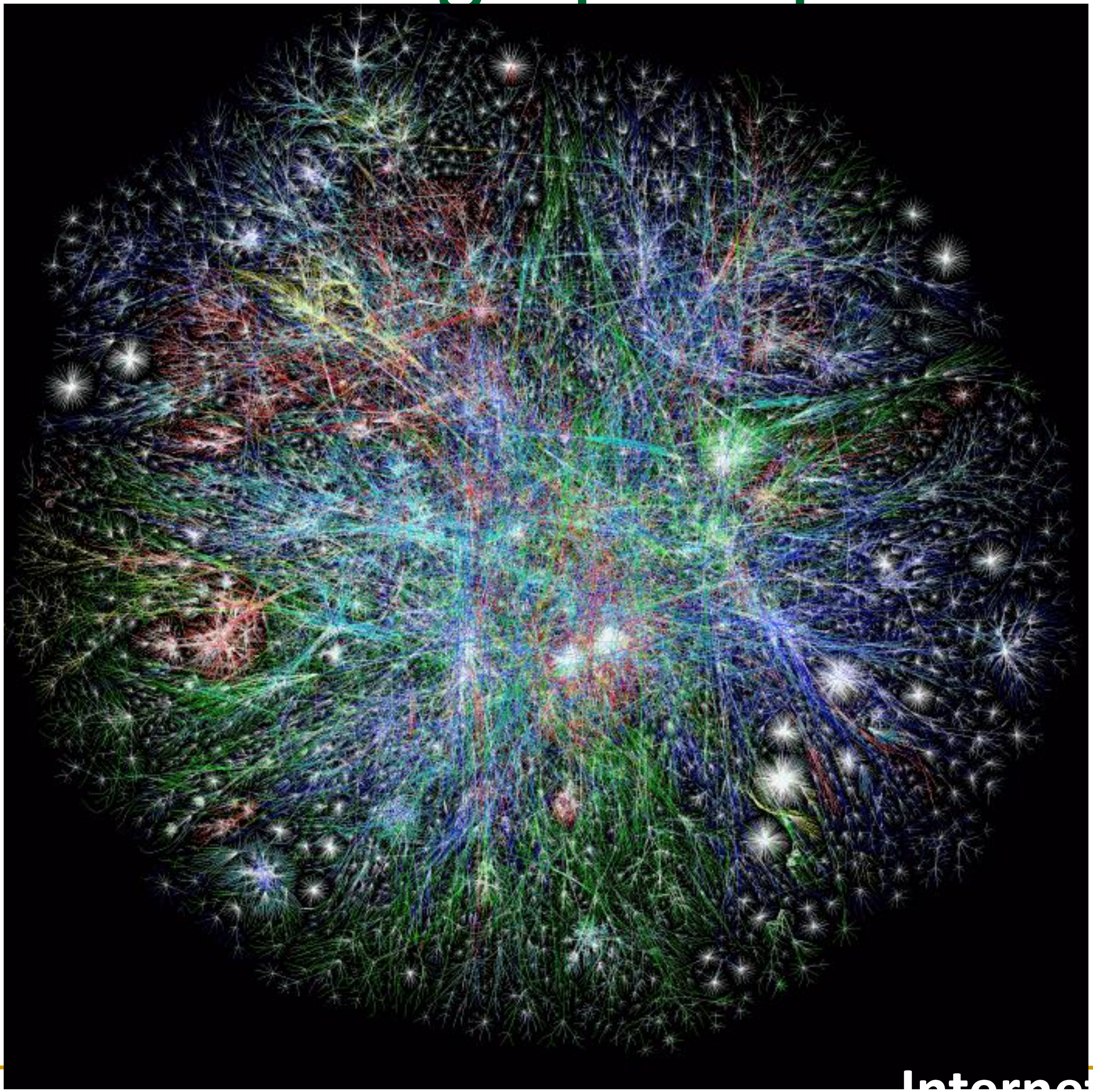
- Electronic circuits
  - Printed circuit board
- Computer networks
  - Local area network
  - Internet
  - Web



Graphs are omnipresent

Airline Route maps

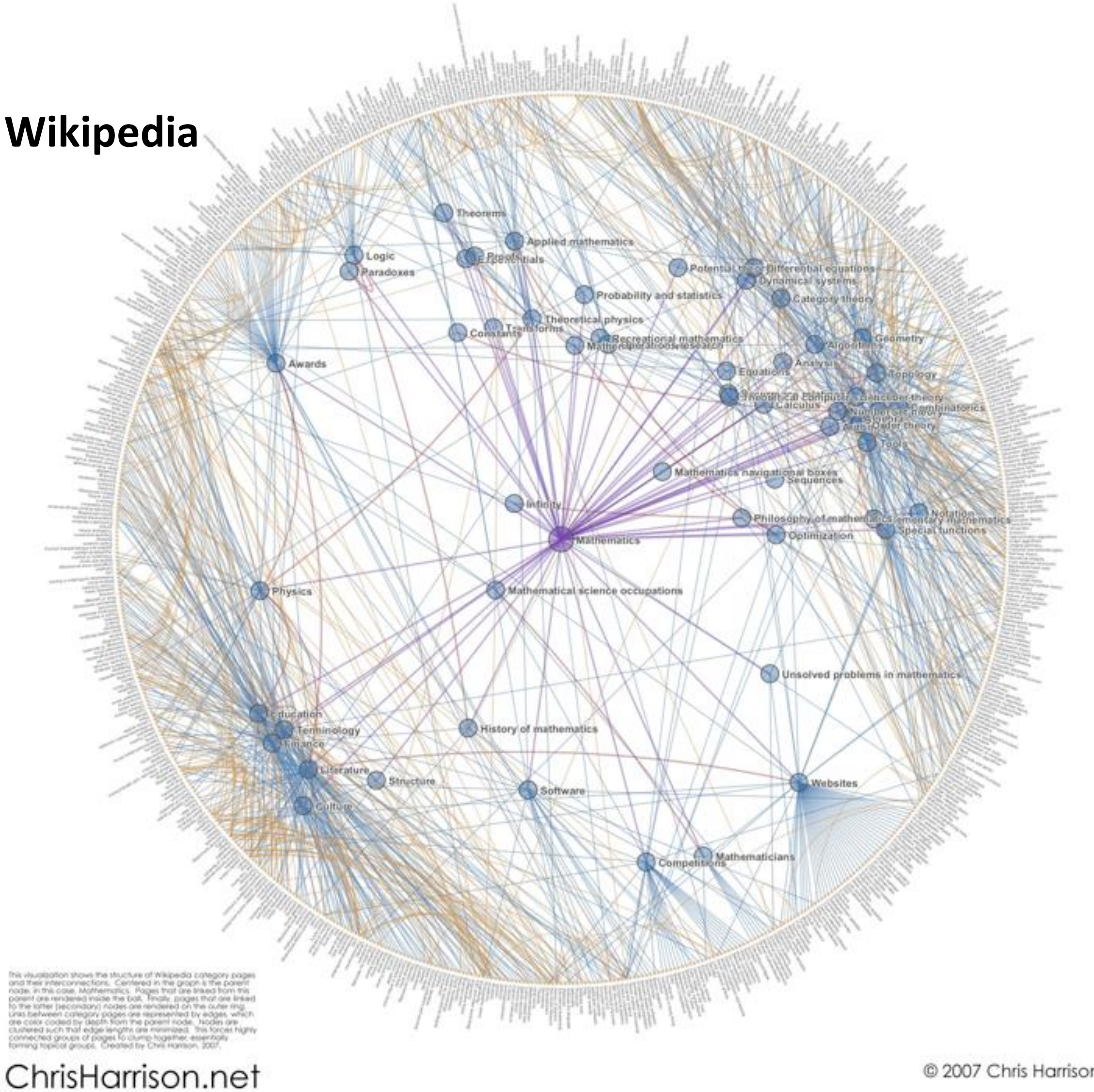
What does this graph represent?



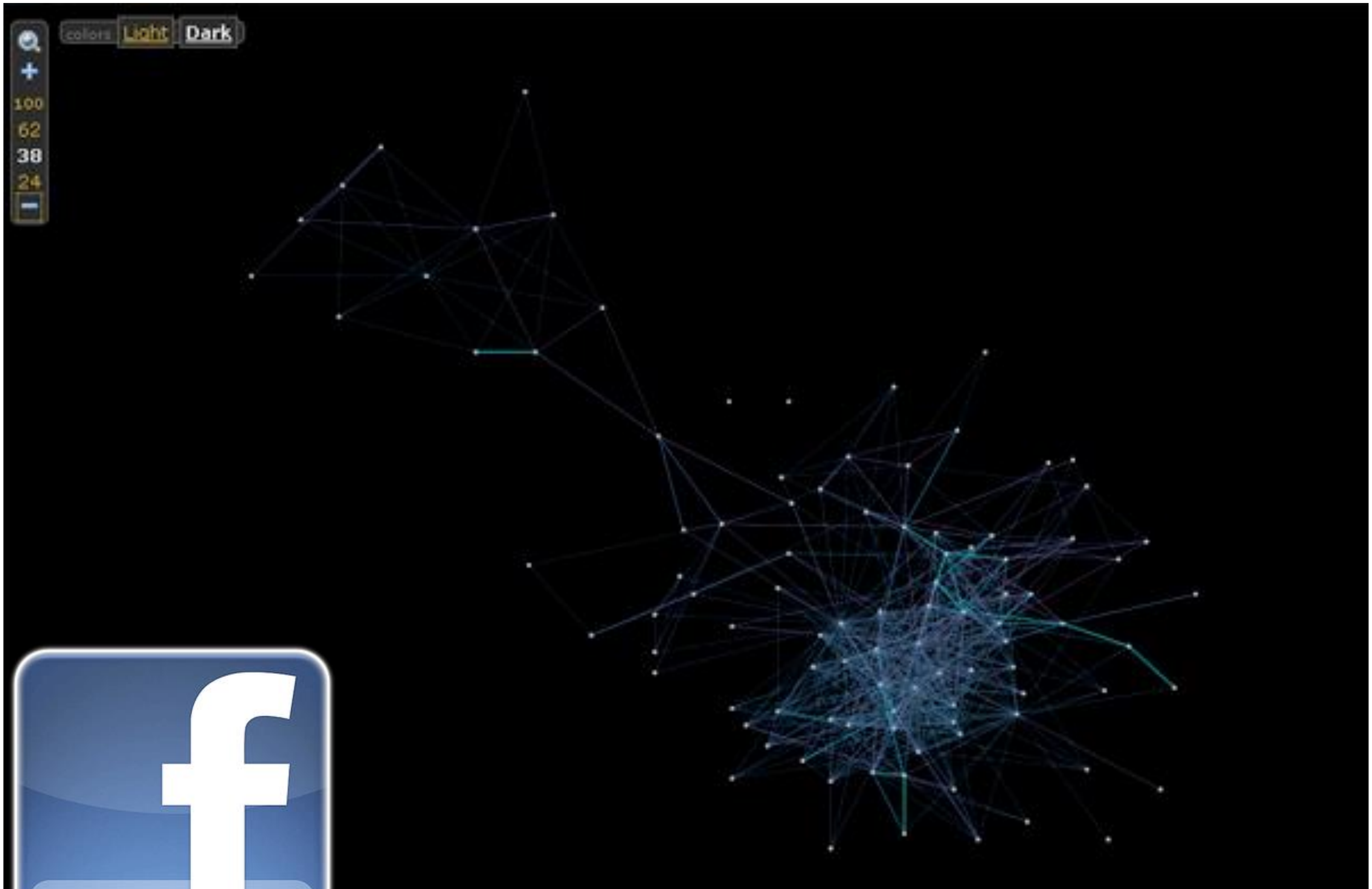
Internet

And this one?

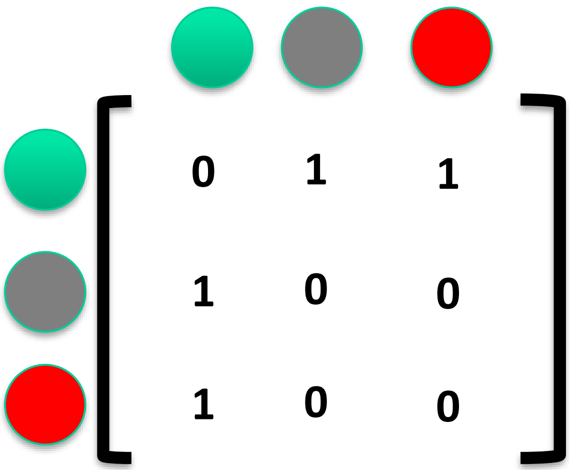
Math articles on Wikipedia



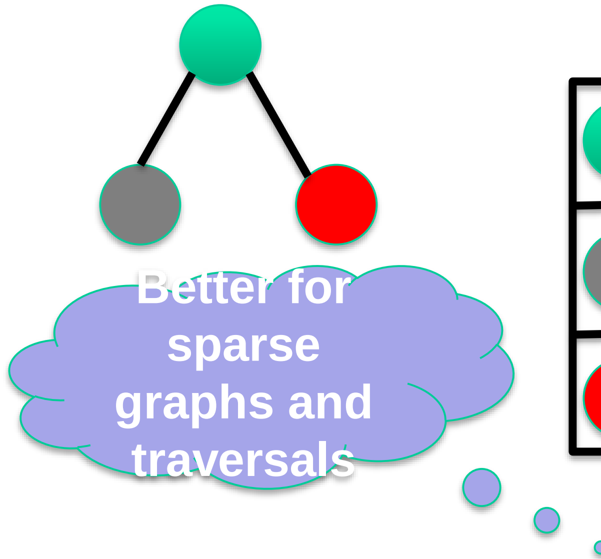
And this one?

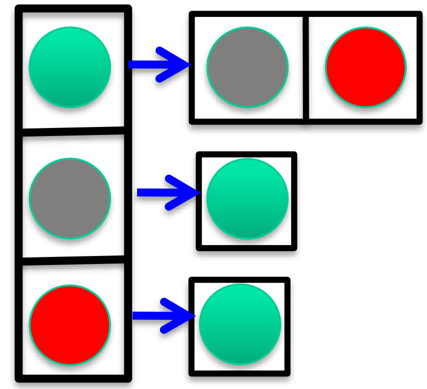


Graph representations



Adjacency matrix



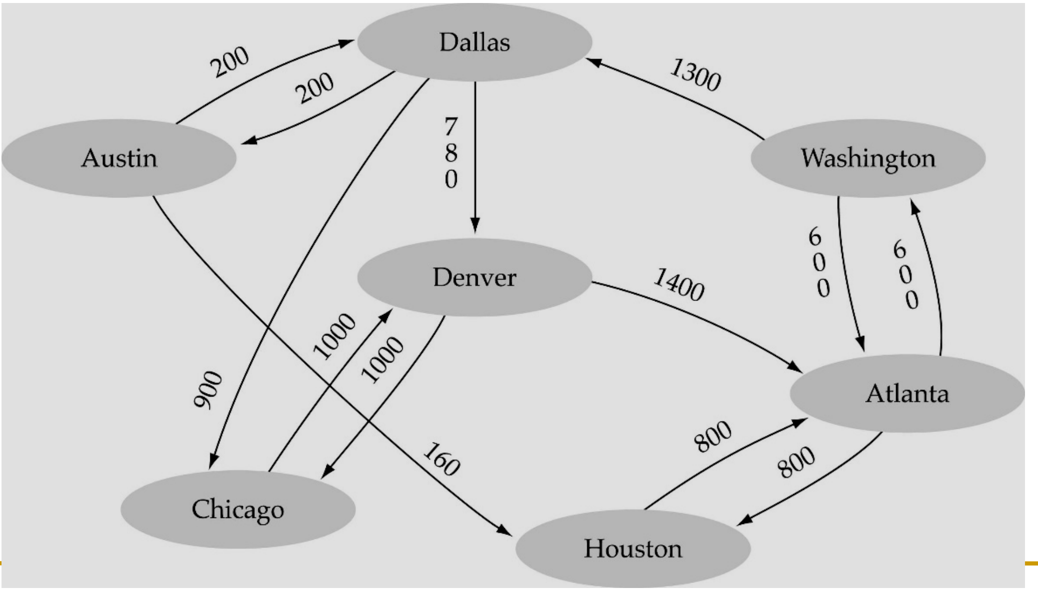


Adjacency List

Graph implementation

Array-based implementation

- A 1D array is used to represent the vertices
- A 2D array (adjacency matrix) is used to represent the edges



Array-based implementation

graph

.numVertices 7

.vertices

[0]	"Atlanta "
[1]	"Austin "
[2]	"Chicago "
[3]	"Dallas "
[4]	"Denver "
[5]	"Houston "
[6]	"Washington"
[7]	
[8]	
[9]	

.edges

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
[0]	0	0	0	0	0	800	600	•	•	•
[1]	0	0	0	200	0	160	0	•	•	•
[2]	0	0	0	0	1000	0	0	•	•	•
[3]	0	200	900	0	780	0	0	•	•	•
[4]	1400	0	1000	0	0	0	0	•	•	•
[5]	800	0	0	0	0	0	0	•	•	•
[6]	600	0	0	1300	0	0	0	•	•	•
[7]	•	•	•	•	•	•	•	•	•	•
[8]	•	•	•	•	•	•	•	•	•	•
[9]	•	•	•	•	•	•	•	•	•	•

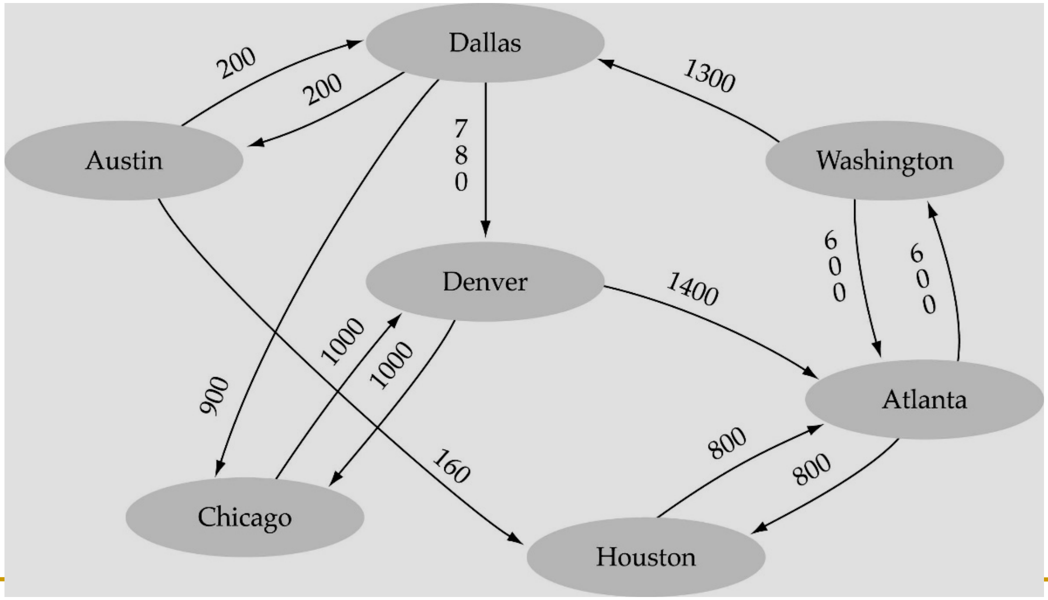
(Array positions marked '•' are undefined)

Graph implementation (cont.)

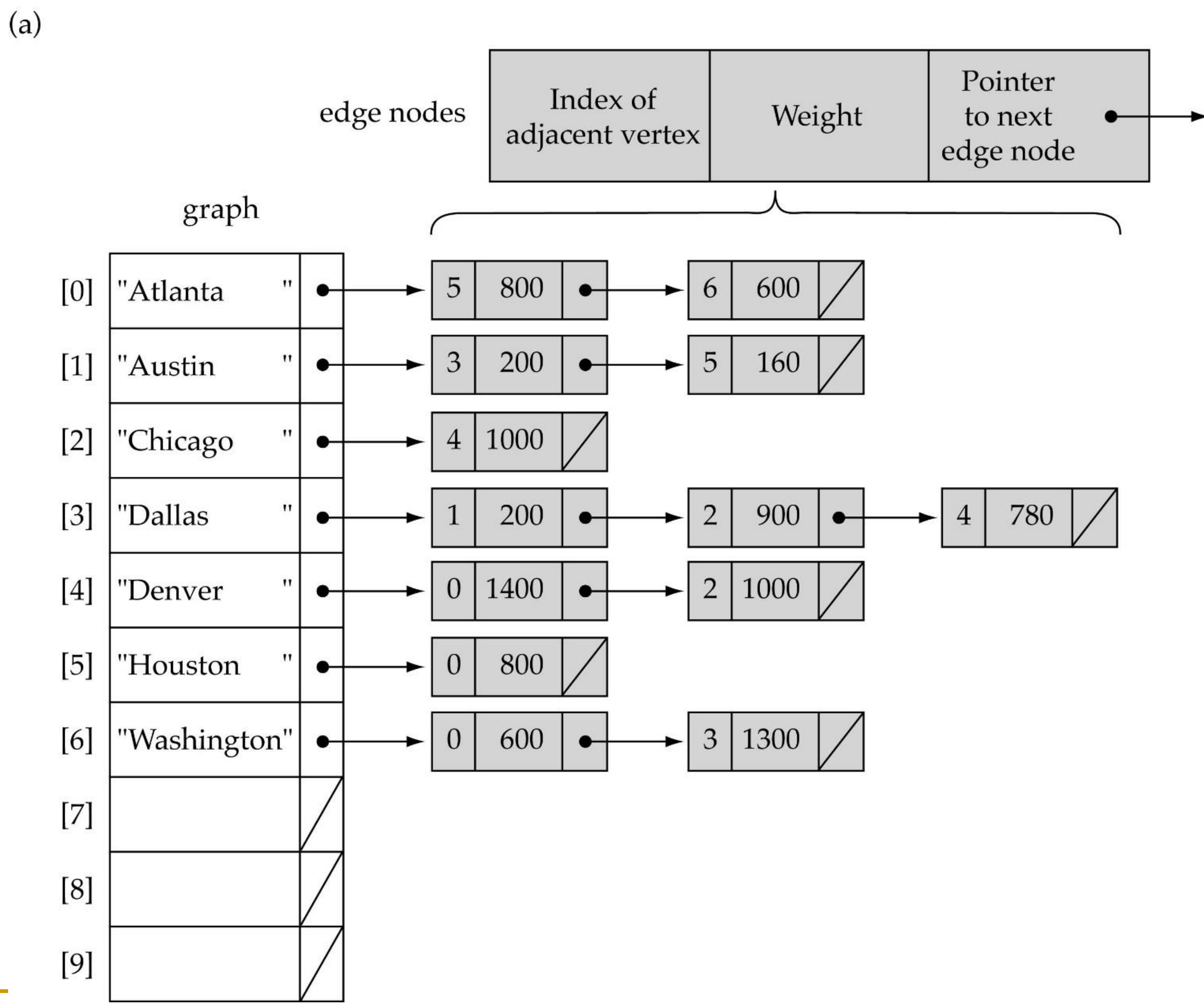
<u>Linked-list implementation</u>

A 1D array is used to represent the vertices

A list is used for each vertex *v* which contains the vertices which are adjacent from *v* (adjacency list)



Linked-list implementation



Adjacency matrix vs. adjacency list representation

Adjacency matrix

- Good for dense graphs
- Connectivity between two vertices can be tested quickly

Adjacency list

- Good for sparse graphs Vertices adjacent to another vertex can be found quickly