CSC 212: Data Structures and Abstractions

Graphs

Prof. Marco Alvarez

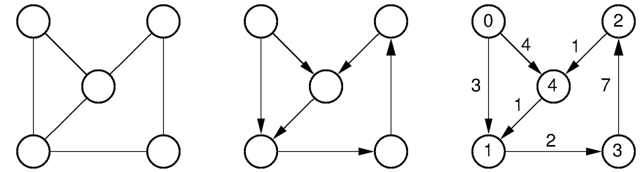
Department of Computer Science and Statistics
University of Rhode Island

Fall 2025

THINK BIG WE DO℠

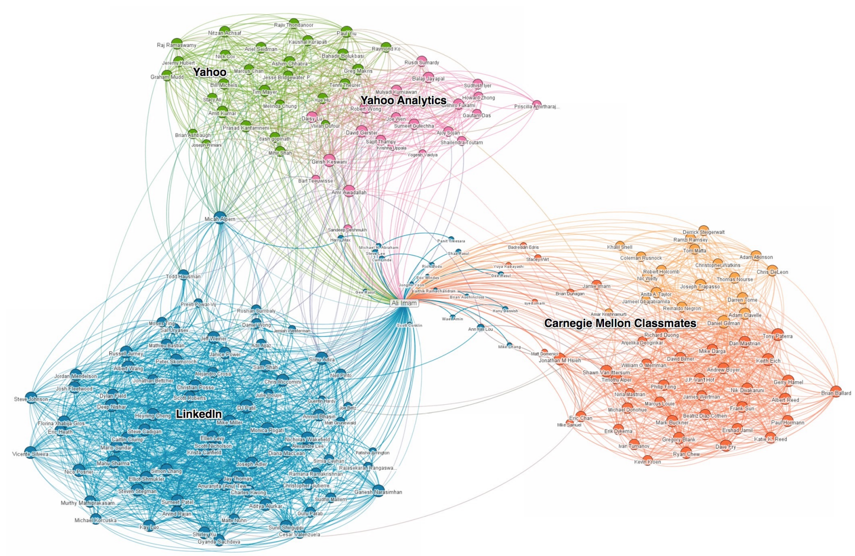
What is a graph?

A *graph* G is an ordered pair $G = (V, E)$, comprising a finite set of *nodes* V and a finite set of *edges* E

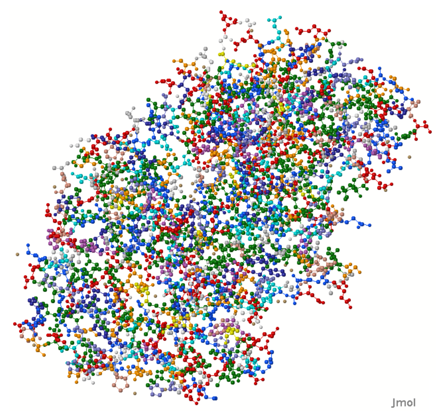


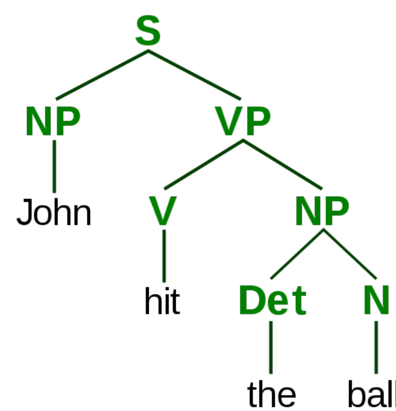
- Edges
  - ✓ can be **undirected** (u, v) or **directed** $(u \rightarrow v)$
  - ✓ may have additional attributes: weights or labels
- Why study graphs?
  - ✓ broadly useful abstraction with efficient algorithms
  - ✓ real-world applications

Graphs everywhere

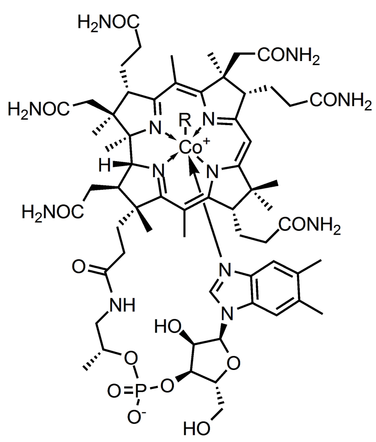


Graphs everywhere

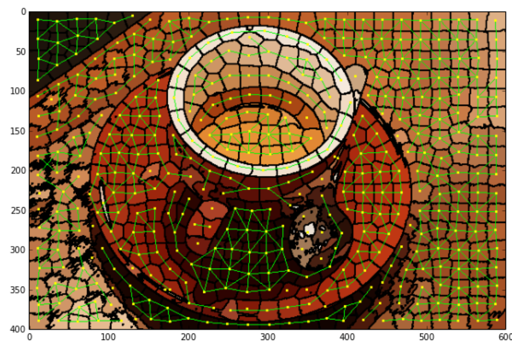




Graphs everywhere



R = 5'-deoxyadenosyl, Me, OH, CN



Applications

- Graphs model relationships
  - ✓ networks, dependencies, maps, molecules, social graphs, etc.
- Real-world graphs
  - ✓ social networks
    - vertices = users, edges = friendships/follows
  - ✓ road networks
    - vertices = intersections, edges = streets
  - ✓ course prerequisites
    - vertices = courses, edges = directed dependencies
  - ✓ computer networks
    - vertices = routers, edges = links

Types of graphs

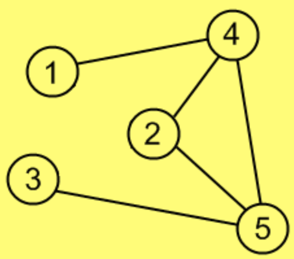
- Based on edges
  - ✓ **undirected** — all edges have no orientation
  - ✓ **directed (digraphs)** — all edges point from one vertex to another
- Based on edge attributes
  - ✓ **weighted** — edges store weights
  - ✓ **unweighted** — edges do not carry information
- Based on structure
  - ✓ **simple** — no self-loops, no parallel edges
  - ✓ **multigraph** — parallel edges allowed
  - ✓ **pseudograph** — self-loops allowed
- Based on Density
  - ✓ **sparse** — $|E| \approx |V|$
  - ✓ **dense** — $|E| \approx |V|^2$

Basic terminology

Term	Meaning
Vertex	A node in the graph
Edge	A connection between two vertices
Adjacent	Two vertices connected by an edge
Neighbors	Adjacent vertices
Incident	An edge that touches a vertex
Degree	Number of incident edges
In-degree / Out-degree	Directed version of degree
Path	Sequence of edges connecting vertices (no repeated edges, undirected or directed)
Cycle	Path that starts and ends at the same vertex (undirected or directed)
Connected graph	Every vertex reachable (undirected graphs)

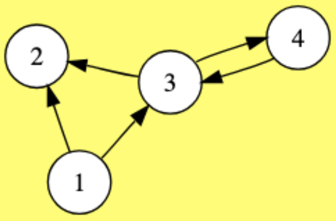
Practice

- Given this undirected graph
  - ✓ determine the degrees of all vertices



Practice

- Given this directed graph
  - ✓ determine the in-degree and out-degree of all vertices
  - ✓ identify all cycles



Graph representations

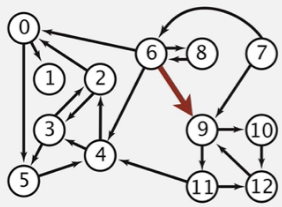
- Why do we need data structures for graphs?
  - ✓ efficiency of storage and operations depends on representation
  - ✓ sparse graphs and dense graphs require different structures
  - ✓ some algorithms prefer matrix access, others list traversal
- Representations
  - ✓ adjacency matrix
  - ✓ adjacency list
  - ✓ edge list

Adjacency matrix

- Definition
  - ✓ $n \times n$ matrix where $n = |V|$ and element at (row u column v) is 1 if edge exists connecting vertices u and v, 0 otherwise
  - ✓ for weighted graphs, replace 1 by the weight w and 0 by a special value that indicates the edge does not exist
- Space Complexity
  - ✓ $\Theta(V^2)$ regardless of actual number of edges
- When is it useful?
  - ✓ representing dense graphs
  - ✓ fast $O(1)$ adjacency queries

Symbol tables (maps) can be used to convert between names and integers

Example



to

from	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	1	0	0	0	1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0
2	1	0	0	1	0	0	0	0	0	0	0	0	0
3	0	0	1	0	0	1	0	0	0	0	0	0	0
4	0	0	1	1	0	0	0	0	0	0	0	0	0
5	0	0	0	0	1	0	0	0	0	0	0	0	0
6	0	0	0	0	1	0	0	0	1	1	0	0	0
7	0	0	0	0	0	0	1	0	0	1	0	0	0
8	0	0	0	0	0	0	1	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	1	1	0
10	0	0	0	0	0	0	0	0	0	0	0	0	1
11	0	0	0	0	1	0	0	0	0	0	0	0	1
12	0	0	0	0	0	0	0	0	0	1	0	0	0

Note: parallel edges disallowed

Image credit: COS 226 @ Princeton

Adjacency list

- Definition
  - ✓ list (or dictionary) where each vertex stores all its adjacent vertices
  - ✓ for weighted graphs, add the weight information to each element
    - $u : (v, w_1), (x, w_2)$
- Space Complexity
  - ✓ $\Theta(V + E)$ ideal for sparse graphs
- Advantages
  - ✓ efficient traversal
  - ✓ compact storage

Example

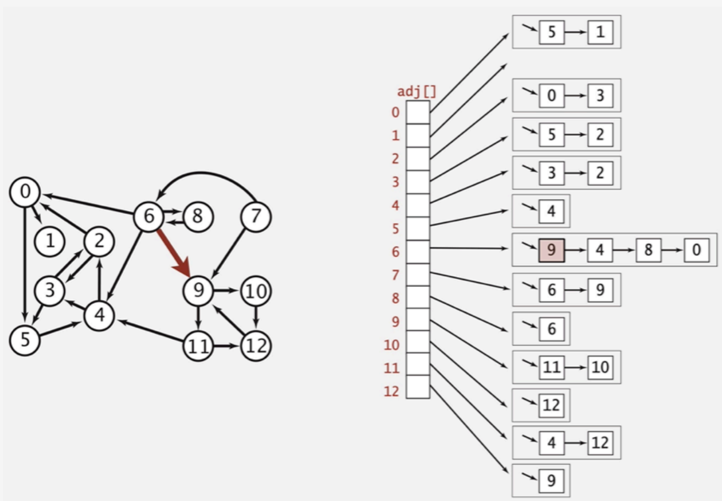


Image credit: COS 226 @ Princeton

Edge list

- Definition
  - ✓ a simple list of all edges: $[(u_1, v_1), (u_2, v_2), \ldots, (u_n, v_n)]$
  - ✓ for weighted graphs, add the weight information to each element
    - (u, v, w)
- Space Complexity
  - ✓ $\Theta(E)$
- Advantages
  - ✓ very compact
  - ✓ good for input parsing
- Cons
  - ✓ slow adjacency checks

Practice

- Given an undirected graph:
  - $V = \{0,1,2,3\}, \quad E = \{(0,1), (0,2), (1,2), (2,3)\}$
  - draw the corresponding representations: adjacency matrix, adjacency list, edge list

Practice

- Given an directed graph:
  - $V = \{0,1,2,3\}, \quad E = \{(0,1), (0,2), (1,2), (2,3)\}$
  - draw the corresponding representations: adjacency matrix, adjacency list, edge list

Practice

- Draw an undirected graph with 5 vertices where each vertex has degree >= 2
  - draw the corresponding representations: adjacency matrix, adjacency list, edge list
  - identify all cycles
  - is the graph connected?

Practice

- For each of the 3 representations, indicate the computational cost of:
  - checking if two vertices are adjacent
  - iterating all of the neighbors of a vertex u