Graphs & Graph Data Structures (ADTs)

I. Examples of graphs / networks

$G(V, E)$ - graph or network

- V: set of nodes / vertices
  - objects
  - entities
- E: set of edges or arcs
  - links
  - relationships
  - associations

Table 1: Examples of real-world networks

Graph / Network	Node / Vertex	Edge / Arc
Internet	Computer / router	Cable / wireless data connection
WWW	Web page	Hyperlink
Facebook	Person	Friendship
Paper citation graph	Paper	Citation
Power Grid	Generating station	Transmission line
Neural network	Neuron	Synapse
Calling graph in software engin.	Function	Caller-callee relation
Airport network	Airport in a city	Flights connection
US roads	City	Road
Railroad network	Stations	Train routes
Course schedule network	Course, student, professor	Attend / teach relation

Note: Make sure you went over **Required** readings first.

II. Types of Graphs

1. Directionality of edges

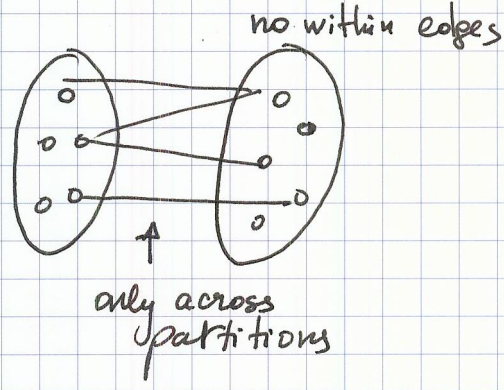
- Directed (arcs) $e \in E$, $e = (u, v)$: $u \rightarrow v$
- Undirected $e \in E$, $e = (u, v)$ u — v

2. Frequency of updates

- Static (rare updates)
- Dynamic (frequent updates), time-evolving graphs

3. Number of partitions

- Unipartite
- Bipartite
- Multipartite

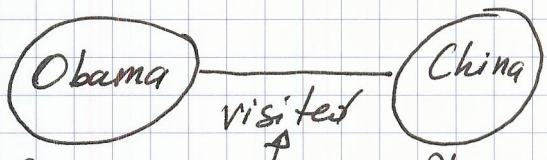


4. Density of edges

- Dense, $|E| \sim |V|^2$, $|E| = m$, $|V| = n$, $m = O(n)$
- Sparse, $|E| \sim |V|$ $m = O(n)$

5. Labels of graph elements (nodes/edges)

- Labeled / Attributed
- Unlabeled

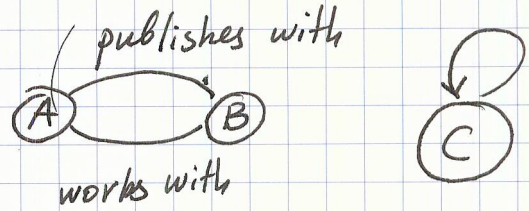


E.g: semantic web (node label)

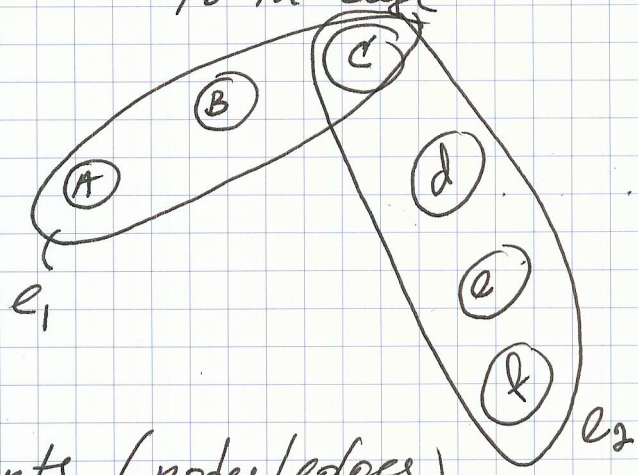
Node ID is unique but node may have many labels

6. Edge Type

- Multi-edges
- Self-loops



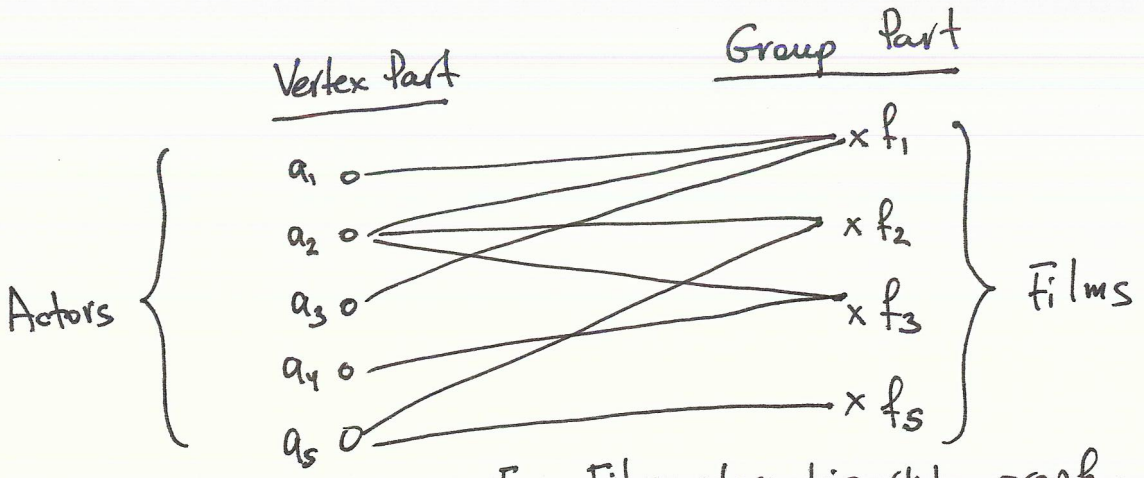
- Hypergraphs, multiple vertices belong to the edge
  hyperedges



- Simple graphs
  (no multi-edges,
  no self-loops
  no hyperedges

(7) Weights of elements (nodes/edges)

- Weighted (e.g., distance, similarity, importance)
- Unweighted



Ex.: Film actors bipartite graph

Hypergraph: Film actors

Edge f_1 contains vertices a_1, a_2, a_3

Edge f_2 contains vertices a_2, a_5

⋮

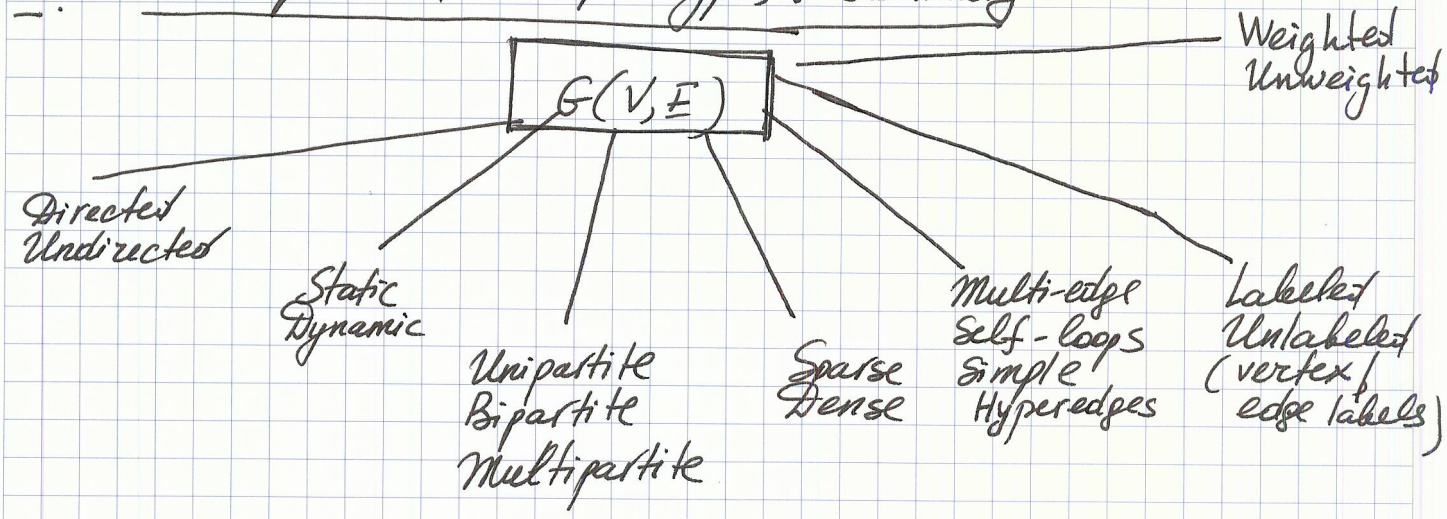
Ex.: Film actors hypergraph

Network	Vertex	Group	Section
Film actors	Actor	Cast of a film	3.5
Coauthorship	Author	Authors of an article	3.5
Boards of directors	Director	Board of a company	3.5
Social events	People	Participants at social event	3.1
Recommender system	People	Those who like a book, film, etc.	4.3.2
Keyword index	Keywords	Pages where words appear	4.3.3
Rail connections	Stations	Train routes	2.4
Metabolic reactions	Metabolites	Participants in a reaction	5.1.1

Table 6.2: Hypergraphs and bipartite graphs. Examples of networks that can be represented as hypergraphs or equivalently as bipartite graphs. The last column gives the section of this book in which each network is discussed.

Source: The book by Newman

III. Examples of Graph Types: Summary



Exercise: Characterize each graph in Table 1 using type information above

Ex: Internet = (Computers, Wireless Connections)
G = (V, E)

- Undirected
- Dynamic
- Unipartite
- Sparse (?)
- Simple
- Unlabeled
- Weighted (avg. connection time?)

(IV) Core Methods of the Graph ADT (Abstract Data Type)

Update Methods

(A.) Vertex

- insertVertex(o)
  insert a vertex storing element o
- removeVertex(v)
  - erase vertex v & its incident edges

(B.) Edge

- insertEdge(u, v, o)
  insert (u, v) edge storing element o
- removeEdge(e)
  erase edge e

(C.) Labels / Attributes

⋮

(D.) Weights

⋮

Find / Lookup / Accessor Methods

(A.) Adjacent relations

- areAdjacent(u, v)
  true iff u and v are adjacent

  u.isAdjacentTo(v)
- areAdjacentEdges(e, w)
  true iff edge e and w are adjacent

(B) Incident relations

- e.endVertices():
  list of two vertices incident to e
- e.opposite(v)
  vertex opposite of v on edge e

(C) Lookup elements Reference element

- v.getElement()
  - reference to element associated w/ vertex v
- e.getElement()
  reference to element associated w/ edge e

Iterators / Enumerators Collection Methods

(A.) Vertex Set

- vertices()
  list of all vertices in G
- adjacentVertices(v)
  all neighbors of vertex v

(B.) Edge Set

- edges()
  list of all edges in G
- incidentEdges(v)
  list of edges incident to v

(V) Graph Data Structures: Summary

Note: Make sure to go over Big-O refresher slides

1. Adjacency Matrix (Vertex Adjacency)
   - Boolean Matrix (true iff v is adjacent to u)
   - Matrix with Edge Weights
   - With Edge List Structure

2. Adjacency List (Vertex Adjacency)
   - Undirected Graph
   - Directed Graph:
     - Successor-based
     - Predecessor-based

3. Adjacency Tree (Vertex Adjacency)
   - Alleviate some disadvantage of adj. list due to high cost of finding and removing edges

4. Hybrid matrix/list structures

5. Edge Lists

6. Heaps
   - Not for storing graphs but
   - For storing values on graphs (e.g. values associated with vertices)
   - Vertex weighted graphs
     - min/max weight vertex

V. A Graph ADT: Adjacency Matrix (see Required Reading)

V. B

Graph ADT: Adjacency List

(see Required Reading)

V.C Graph ADT : Adjacency Tree (To be covered later)

(VI) Comparison of Graph Data Structures

Let G be a simple, undirected graph w/ n vertices & m edges. Let $deg(v)$ denote the degree of node v, or $|N(v)|$ - the # of neighbors

Method	Adj List: Space $O(n+m)$	Adj matrix: Space $O(n^2)$
Update Methods:	Time Complexity:	
insertVertex()	$O(1)$	$O(n^2)$
insertEdge()	$O(1)$	$O(1)$
removeVertex()	$O(deg(v))$ $\max deg(v) = n-1$	$O(n^2)$
removeEdge()	$O(1)$	$O(1)$
Accessor / Lookup Methods:		
areAdjacent(u, v)	$O(\min\{deg(u), deg(v)\})$	$O(1)$
Iterator / Enumerator Methods:		
incidentEdges(v) ($N(v)$ - neighbors)	$O(deg(v))$	$O(n)$