Algorithms and Data Structures (CSci 115)

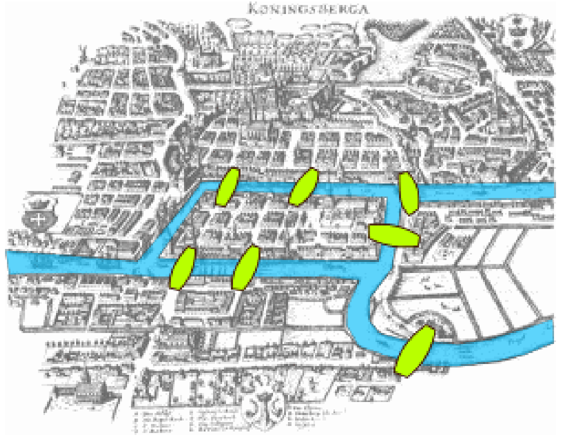
California State University Fresno
College of Science and Mathematics
Department of Computer Science
H. Cecotti

Learning outcomes

- **Graphs**
  - Definitions and principles
  - Representation of graphs
  - Examples

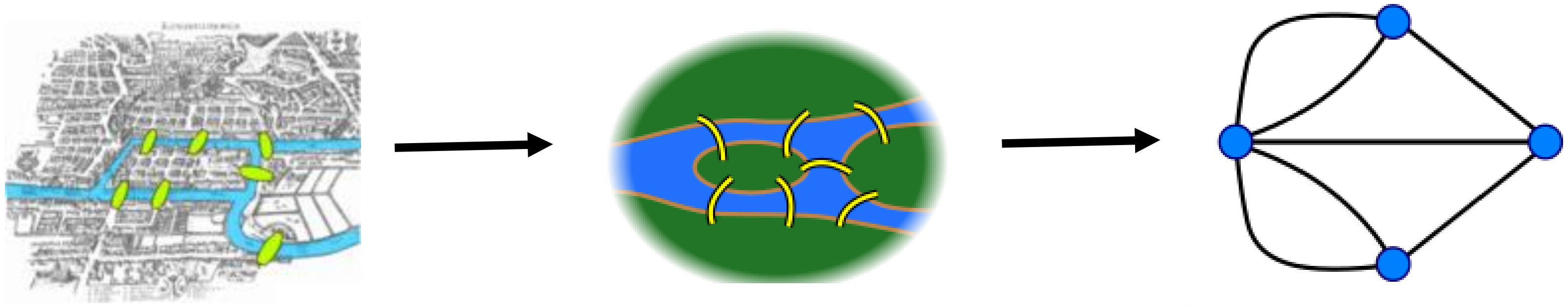
Motivations

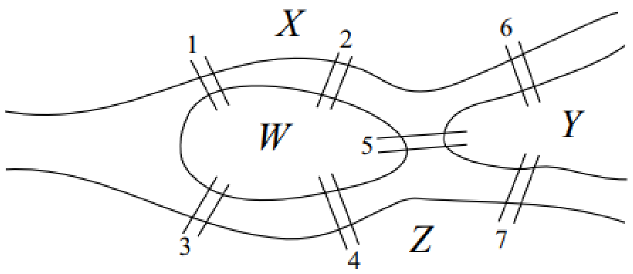
- The Königsberg Bridge Problem
  - ➢ The city of Königsberg (Kaliningrad)
    - o Located on the Pregel river in Prussia.
  - ➢ The river divided the city into 4 separate landmasses including the island of Kneiphopf.
    - o These 4 regions were linked by seven bridges.
  - ➢ Problem
    - o Residents of the city wondered if
      - • it were possible to leave home, cross each of the 7 bridges exactly once, and return home.
  - ➢ The Swiss mathematician Leonhard Euler (1707-1783) thought about this problem …
  - ➢ → Graph theory

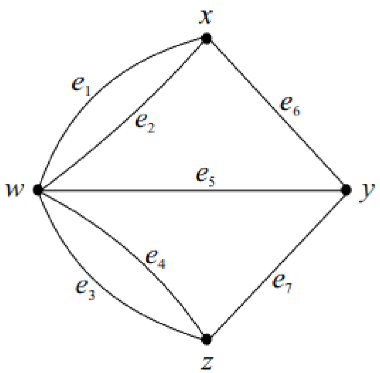


Motivations

- The Königsberg Bridge Problem
  - From a practical problem to a formal problem…

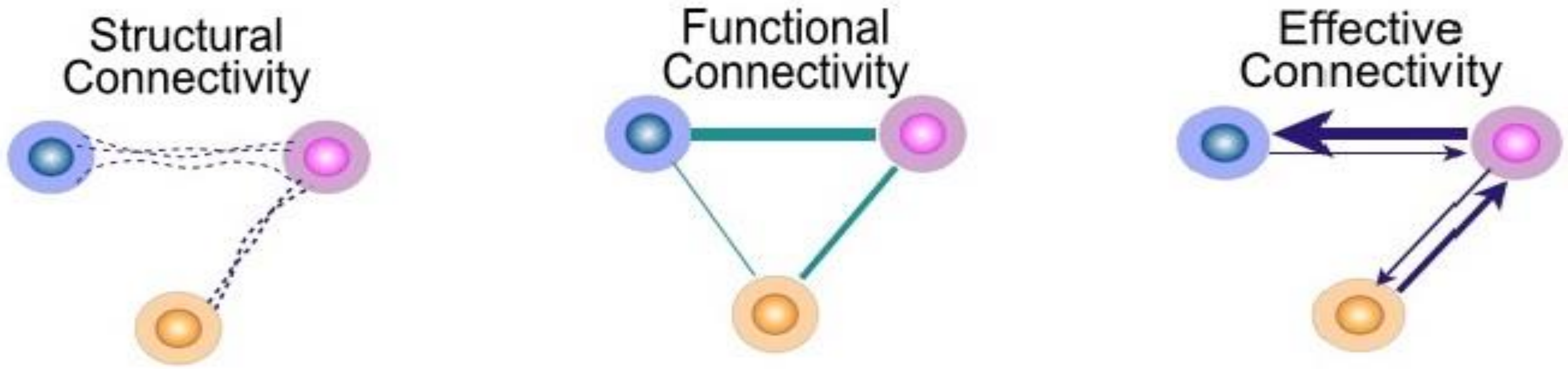






Applications

- Brain connectivity analysis



  - Anatomical information Relationships between areas Direction of the information flow

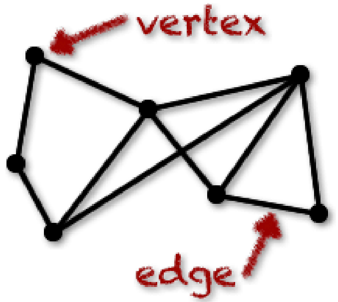
- Nodes
  - From/Source
  - Destination/Sink

Rationale

- We need graph for problems such as:
  - Finding the Shortest path
- Graphs for the representation of problems used with
  - **Greedy** algorithms
  - **Dynamic** programming

Graphs

- Definitions
  - *Graph G* = (V, E)
    - V = set of vertices
    - E = set of edges (arcs) $\subseteq (V \times V)$



  - Types of graphs
    - **Undirected**: edge $(u, v) = (v, u)$; for all v, $(v, v) \notin E$ (No self loops.)
    - **Directed**: (u, v) is a edge from u to v, denoted as $u \rightarrow v$. Self loops are allowed.
    - **Weighted**: Each edge has an associated weight, given by a weight function $w : E \rightarrow \mathbf{R}$.
    - **Mixed**: some edges may be directed and some may be undirected
    - **Multigraph**: multiple edges are two or more edges that connect the same two vertices.
    - **Dense**: $|E| \approx |V|^2$.
    - **Sparse**: $|E| << |V|^2$.
  - $|E| = O(|V|^2)$

Graphs

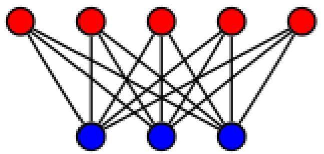
- Definitions
  - $|V|$: # of vertices
  - $|E|$: # of edges
  - If $(u, v) \in E$, then vertex v is **adjacent** to vertex u.
  - Adjacency relationship is:
    - Symmetric if G is undirected.
    - **Not necessarily** so if G is directed.
  - If G is **connected**:
    - There is a path between every pair of vertices.
    - $|E| \geq |V| - 1$.
    - if $\mathbf{|E| = |V| - 1}$**, then** $\mathbf{G}$ **is a tree**.

Graphs

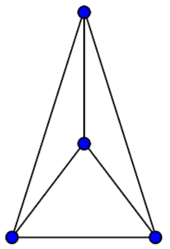
- Definitions
  - **Degree** of a vertex v: the number of edges attached to the vertex v
  - **Simple** graph
    - Undirected, both multiple edges and loops are disallowed
    - the edges form a *set*
      - each edge is an **unordered pair of *distinct* vertices**.
    - With n vertices, the degree of every vertex is at most $n - 1$.

Graphs

- Main types of graphs
  - **Connected** graphs
    - Every unordered pair of vertices in the graph is connected
    - → there is a path from any point to any other point in the graph
  - **Bipartite** graphs
    - Vertices can be divided into 2 disjoint and independent sets U and V
    - Every edge connects a vertex in U to one in V

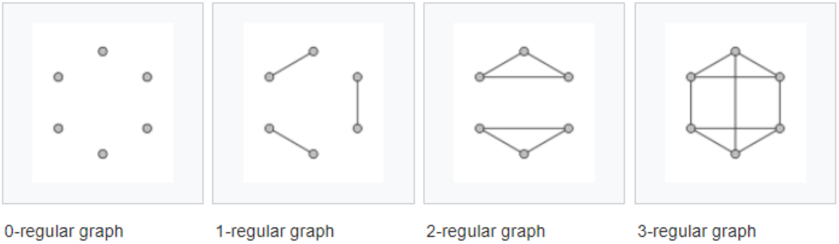


  - **Planar** graph
    - Vertices and edges can be drawn in a plane such that no two of the edges intersect
  - **Cycle** graphs
    - Connected graphs in which the degree of all vertices is 2
  - **Tree**
    - A connected graph with no cycles

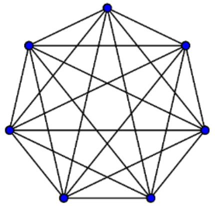


Graphs

- Main types of graphs
  - **Regular** graphs
    - Each vertex has the same number of neighbors
      - every vertex has the same degree



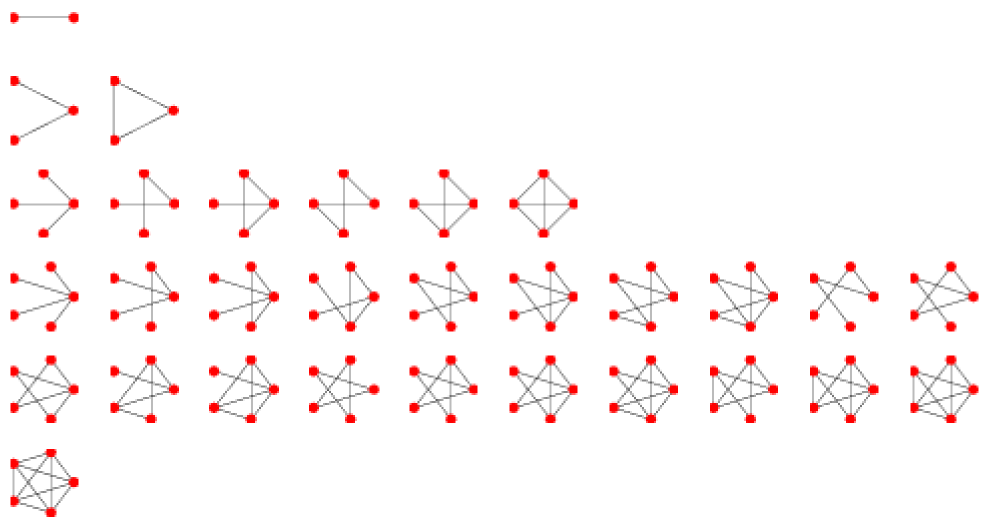
  - **Complete** graphs
    - A simple undirected graph, every pair of distinct vertices is connected by a unique edge



  - **Finite** graphs
    - The vertex set and the edge set are finite sets
  - **Eulerian**
    - If the graph is both connected and has a closed trail containing all edges of the graph
      - A walk with no repeated edges

Graphs

- Example
  - Connected graph



Warning: You may find different definitions in the literature for the path Path/Simple Path

Paths

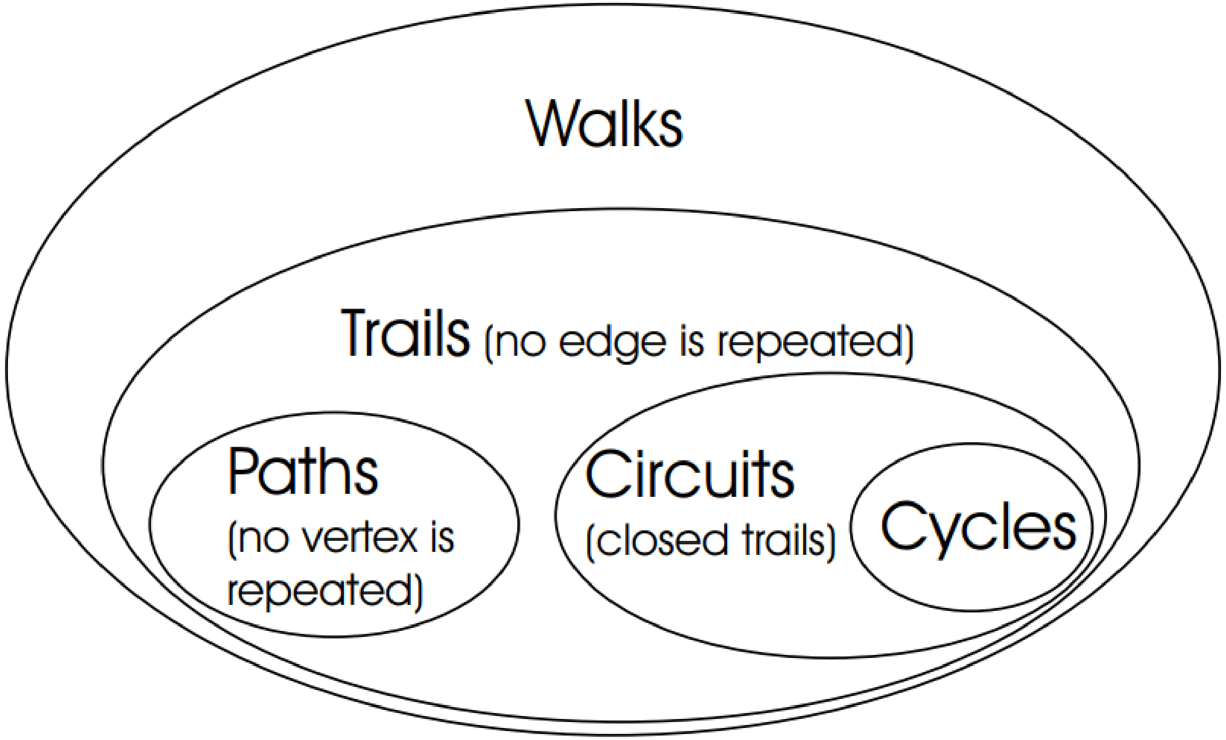
- Definitions
  - A **Walk:** a finite or infinite sequence of edges that joins a sequence of vertices.
  - A **Trail:** a walk in which all edges are distinct.
  - A **Path:** a sequence of vertices $v_1, \ldots, v_k$ where each (v_i, v_{i+1}) is an edge
    - **Simple path**: A path that does not repeat vertices
  - **Path Length**: # of edges in the path
  - A **Circuit**: A path that begins and ends at the same vertex
  - A **Cycle**: A circuit that doesn't repeat vertices
  - **Euler path:** A path that travels through **all edges** of a connected graph
  - **Euler circuit:** A circuit that visits **all edges** of a connected graph. An Euler circuit starts and ends at the same vertex.

Paths

- Some theorems
  - Euler's theorem 1 (**Circuit**)
    - If a graph has any vertex of **odd** degree → it cannot have an Euler circuit.
    - If a graph is connected and every vertex is of **even** degree → it has at least 1 Euler circuit.
  - Euler's theorem 2 (**Path**)
    - If a graph has more than 2 vertices of **odd** degree → it cannot have an Euler path.
    - If a graph is connected and has just 2 vertices of odd degree → it has at least one Euler path.
      - Any such path must start at one of the odd-vertices and end at the other odd vertex.

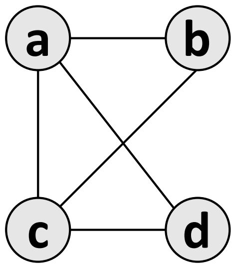
Paths

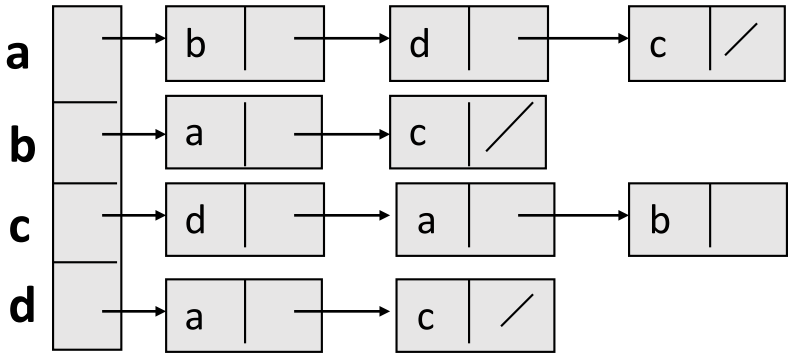
- Graphical representation of the definitions:



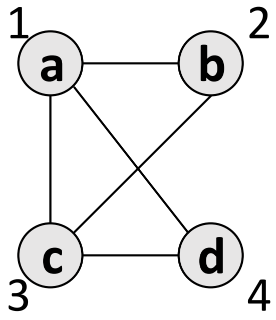
Representation & implementation

- Adjacency Lists.





- Adjacency Matrix.

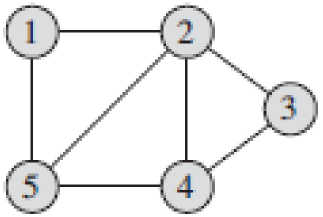


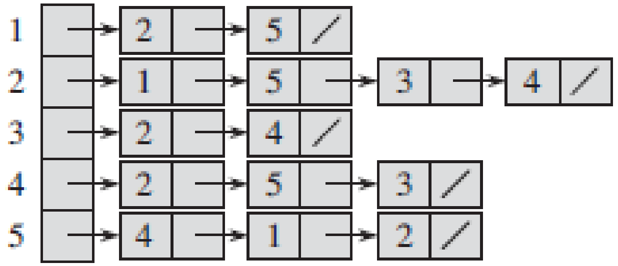
	1	2	3	4
1	0	1	1	1
2	1	0	1	0
3	1	1	0	1
4	1	0	1	0

Representation & implementation

- Examples:

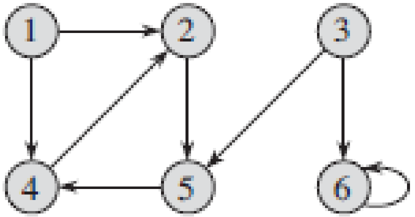
  - ➢Graph 1:

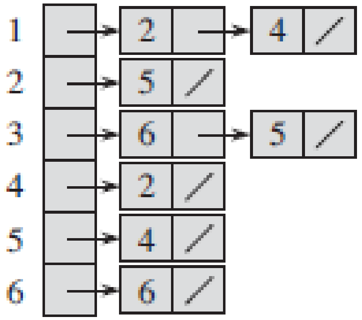




	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

  - ➢Graph 2:

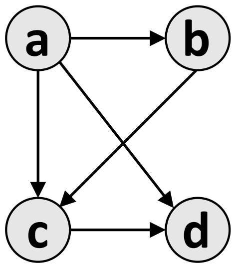


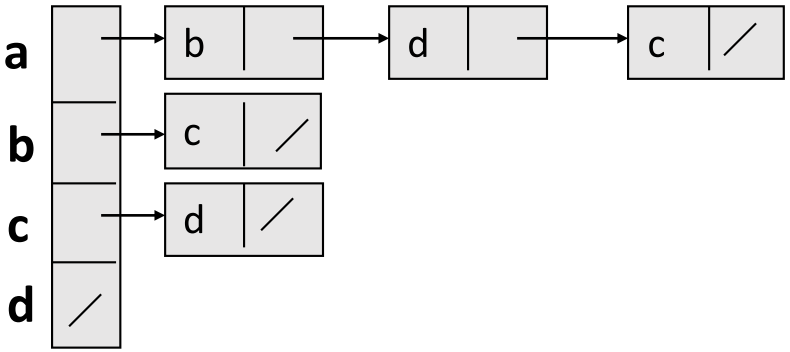


	1	2	3	4	5	6
1	0	1	0	1	0	0
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	0	1	0	0	0	0
5	0	0	0	1	0	0
6	0	0	0	0	0	1

Adjacency lists

- Data structure:
  - Consists of an array *Adj* of $|V|$ lists.
  - One list per vertex.
  - For $u \in V$, $Adj[u]$ consists of all vertices adjacent to u.





Remark:
If weighted arcs, then store weights also in adjacency lists.

Adjacency lists

- For **directed** graphs:
  - Sum of lengths of all adj. lists is

    $\sum_{v \in V} \text{out-degree}(v) = |E|$ (out-degree(v): number of edges leaving v)

  - Total storage: $\Theta(V+E)$
- For **undirected** graphs:
  - Sum of lengths of all adj. lists is

    $\sum_{v \in V} \text{degree}(v) = 2|E|$

  - Total storage: $\Theta(V+E)$

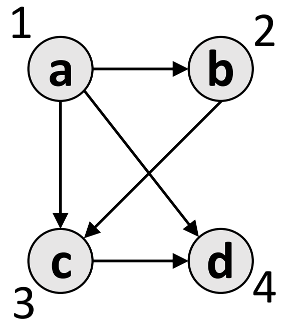
Adjacency lists

- Advantages
  - Space-efficient, when a graph is sparse
  - Can be modified to support many types of graphs
- Disadvantages
  - Determining if an edge $(u,v) \in G$: not efficient.
    - Have to search in u's adjacency list → $\Theta(\text{degree}(u))$ time.
    - $\Theta(V)$ in the worst case !

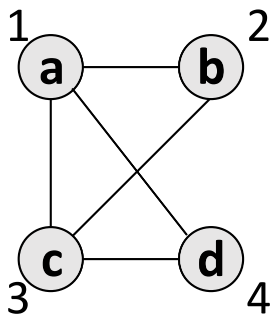
Adjacency matrix

- Matrix A
  - Size $|V| \times |V|$
  - Number vertices from 1 to $|V|$ in some arbitrary manner.
  - A is defined by:

$$A[i,j] = a_{ij} = \begin{cases} 1 & \text{if } (i,j) \in E \\ 0 & \text{otherwise} \end{cases}$$



	1	2	3	4
1	0	1	1	1
2	0	0	1	0
3	0	0	0	1
4	0	0	0	0



	1	2	3	4
1	0	1	1	1
2	1	0	1	0
3	1	1	0	1
4	1	0	1	0

$A = A^T$ for undirected graphs.
→ Triangle for storage

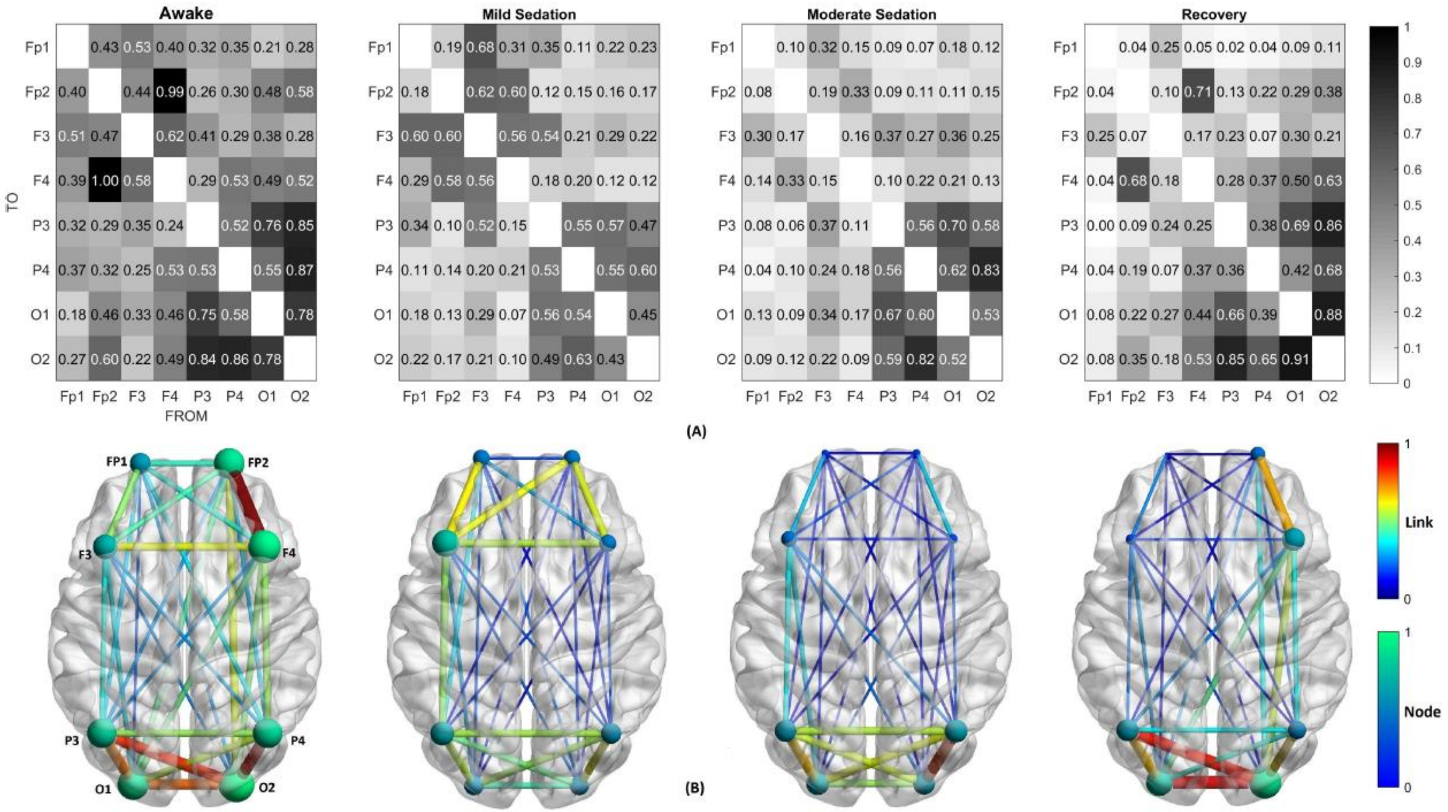
Adjacency matrix

- Space:
  - ➢ $O(N^2)$
- Edge insertion/deletion:
  - ➢ $O(1)$
- Find all adjacent vertices to a vertex:
  - ➢ $O(N)$

	a	*b*	*c*	*d*	*e*
a		1		1	1
b	1		1	1	
c		1		1	1
d	1	1	1		
e	1		1		

Application

- Example: Brain connectivity analysis
  - ➢ Potential CSci 198 projects ☺



Example

- We consider a complete binary tree on 7 vertices
  - Adjacency list

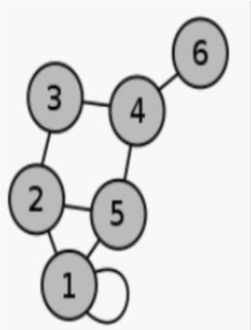
$$
\begin{aligned}
&1 : 2, 3\\
&2 : 1, 4, 5\\
&3 : 1, 6, 7\\
&4 : 2\\
&5 : 2\\
&6 : 3\\
&7 : 3.
\end{aligned}
$$

Adjacency matrix

$$
\begin{bmatrix}
0 & 1 & 1 & 0 & 0 & 0 & 0\\
1 & 0 & 0 & 1 & 1 & 0 & 0\\
1 & 0 & 0 & 0 & 0 & 1 & 1\\
0 & 1 & 0 & 0 & 0 & 0 & 0\\
0 & 1 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 1 & 0 & 0 & 0 & 0\\
0 & 0 & 1 & 0 & 0 & 0 & 0
\end{bmatrix}
$$

Degree matrix

- Definition
  - We consider a graph G=(V,E) with |V| =n
  - → the degree matrix D of G is a diagonal matrix (size n x n)
    - with d_{ij}= Deg(v_i) if i=j , 0 otherwise
  - Special case
    - Directed graph → Degree = **In**Degree or **Out**Degree
- Example



$$\begin{pmatrix} 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Laplacian matrix (simple graph)

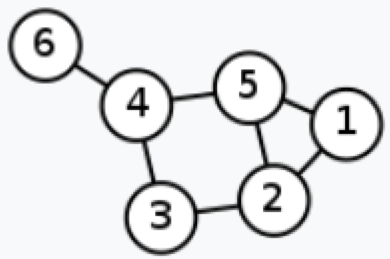
- We consider a graph G with n vertices
  - The Laplacian matrix L_{nxn} is: L=D-A
  - Where
    - D: the degree matrix
    - A: the adjacency matrix
      - Simple graph → matrix contains only 0 and 1, diagonal = 0
  - L_{ij}=
    - $Deg(v_i)$ if (i==j)
    - -1 if i!=j and v_i is adjacent to v_j
    - 0 otherwise

Incidence matrix

- A graph whose oriented incidence matrix M
  - Size $|E| \times |V|$ with element M_{ev}
    - for edge e connecting vertex **i** and **j**, with **i > j** and vertex v given by:
  - $M_{ev} = 1$ if v==i
  - $M_{ev} = -1$ if v==j
  - $M_{ev} = 0$ otherwise
- The Laplacian matrix $L = M^T M$
  - → Eigen values of L are all non-negative

Example

- Graph
  - → Adjacency matrix
  - → Degree matrix
  - → Laplacian matrix

Labeled graph	Degree matrix	Adjacency matrix	Laplacian matrix
	$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix}$

Conclusion

- Different types of graphs
  - Important
    - To specify properly the graph that is used for a given problem
      - Set of edges, vertices…
  - Considerations for the implementation
    - Number of vertices: fixed or not?
      - Lists
      - Matrix
  - Graphs with Matrices
    - Advanced techniques → related to Linear Algebra !!

Questions ?

- Reading
  - Introduction to Algorithms, Chapter 22.