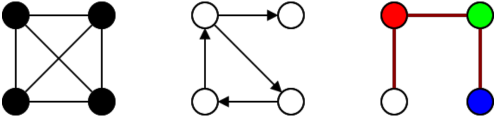
Graphs: Introduction

L.EIC

Algorithms and Data Structures

2025/2026



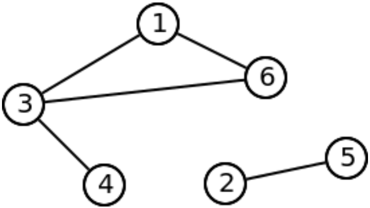
P Ribeiro, AP Tomas

Concept

Graph Definition

Formally, a **graph** is:

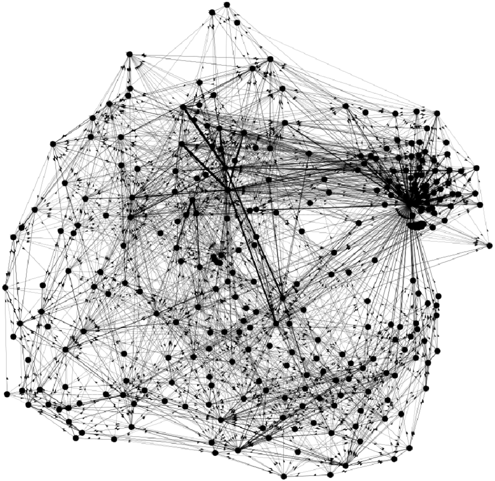
- A set of **nodes**/**vertices** (**V**).
- A set of **links**/**edges**/**connection** (**E**), that connect pairs of vertices



- $V = \{1, 2, 3, 4, 5, 6\}$
- $E = \{(1, 6), (1, 3), (3, 6), (3, 4), (2, 5)\}$

What are graphs for?

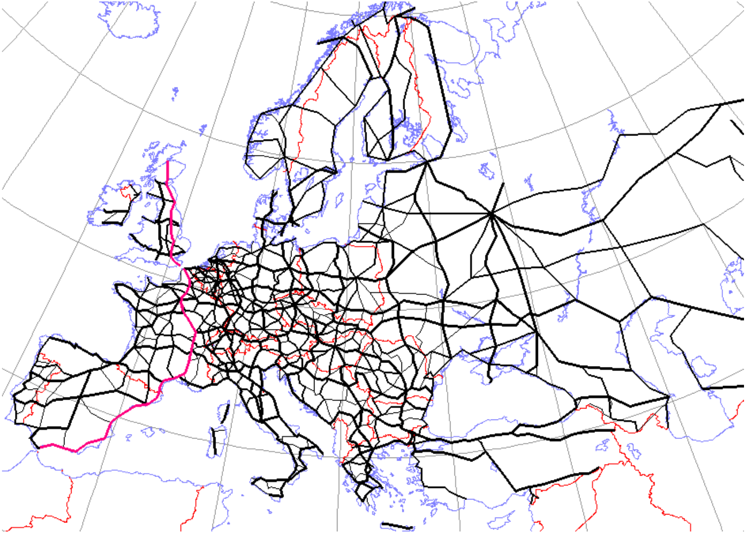
- Graphs are **ubiquitous** in Computer Science and they are present, implicitly or explicitly in many algorithms.
- They can be used in a multitude of applications.



Graph Examples

Networks that exist in the real "physical" world

- Road Network

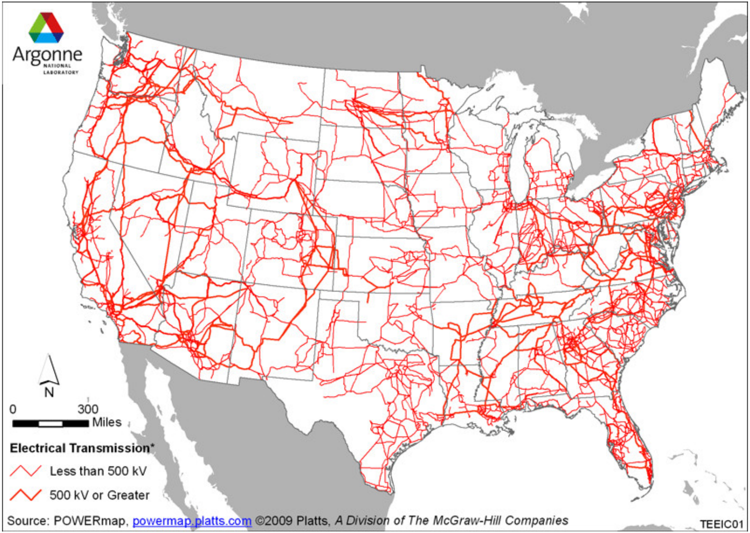


- Public Transportation (ex: subway, train)

Graph Examples

Networks that exist in the real "physical" world

- Power Grid

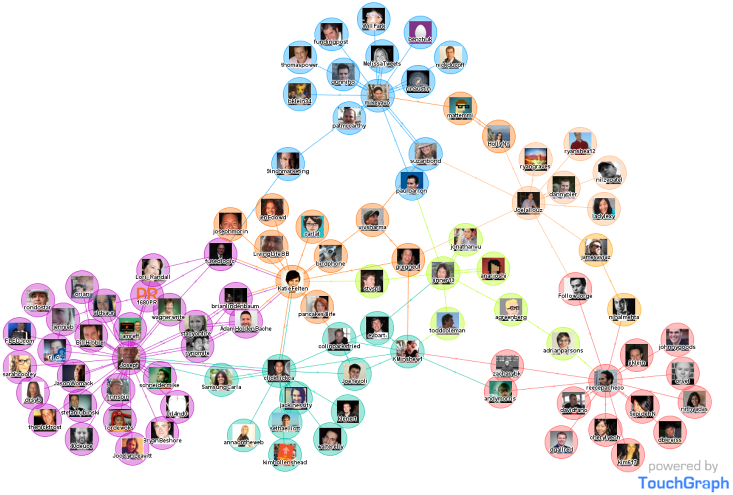


Graph Examples

Networks that exist in the real "physical" world

- Computer Network

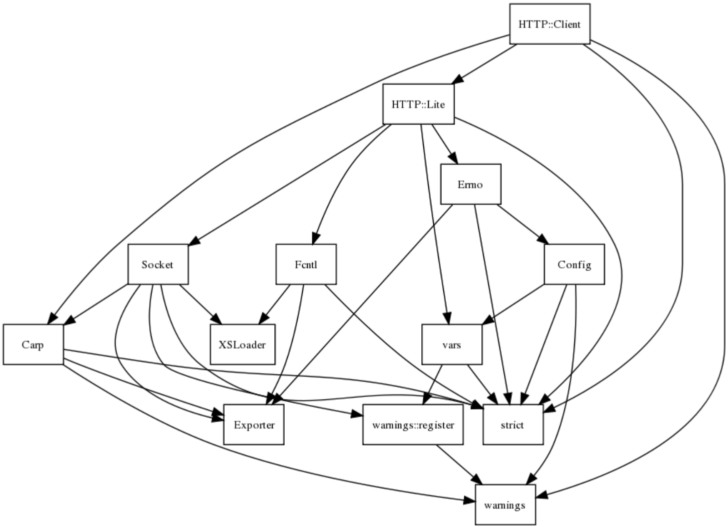
- Facebook (others: Twitter, emails, co-authorship of articles, ...)



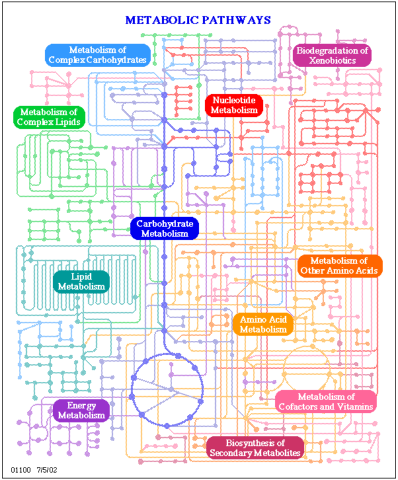
Graph Examples

Software Networks

- Module Dependencies (other examples: state, information flow, ...)



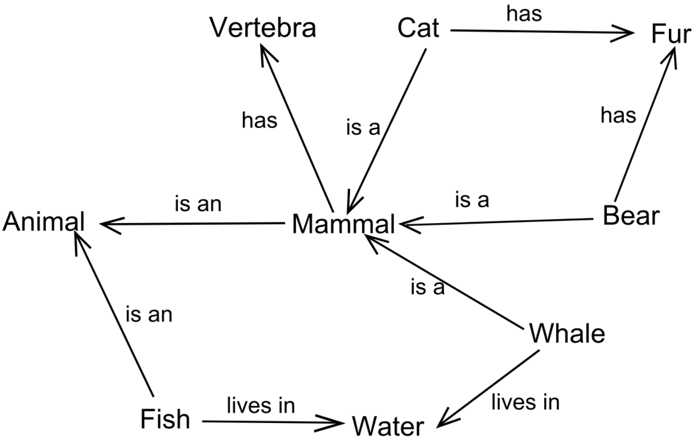
- Metabolic Networks (other examples: protein interaction, brain networks, food webs, phylogenetic trees, ...)



Graph Examples

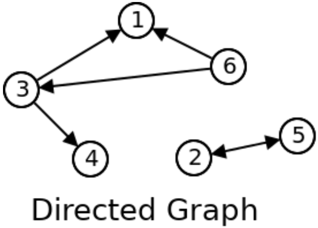
Other Graphs

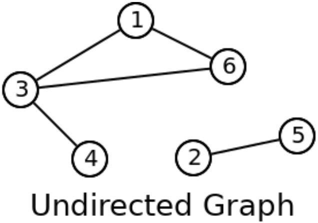
- Semantic Networks (other examples: world wide web, ...)



Terminology

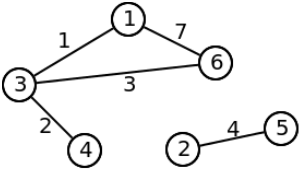
- **Directed** graph - each link has a starting node (**origin**) and an **end** node (order matters!). Usually we use arrows to indicate the direction.
- **Undirected** graphs - There is no origin or end, but just a connection



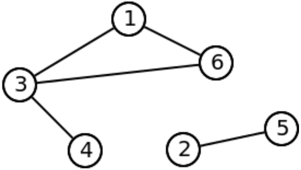


Terminology

- **Weighted** graph - there is a value associated with each link (it could be distance, cost, ...)
- **Unweighted** - there are no weights associated with a link



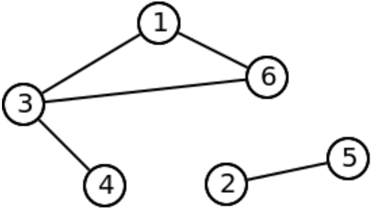
Weighted Graph



Unweighted Graph

Terminology

- **Degree** - number of connections of a node
- In directed graphs we can distinguish between **indegree** and **outdegree**



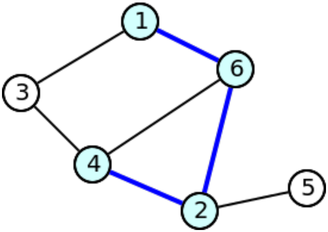
1 has degree 2
2 has degree 1
3 has degree 3
4 has degree 1
5 has degree 1
6 has degree 2

Terminology

- **Adjacent/neighbor** node: two nodes are neighbors if they are linked
- **Trivial graph**: graph with no edges and a single node
- **Self-loop**: link from a node to itself
- **Multigraph:** graph with multiple links between the same node pair
- **Simple graph:** graph without self-loops and without repeated links (we are mostly going to work with simple graphs)
- **Dense graph:** with many links when compared with the maximum possible - $|E|$ of the order of $\mathcal{O}(|V|^2)$
- **Sparse graph:** with few links when compared with the maximum possible - $|E|$ with lower order than $\mathcal{O}(|V|^2)$

Terminology

- **Path**: sequence of alternating nodes and edges, such that two consecutive nodes are linked (and respect the direction, in case of a directed graph).
- In simple graphs we typically describe a path using just the nodes.

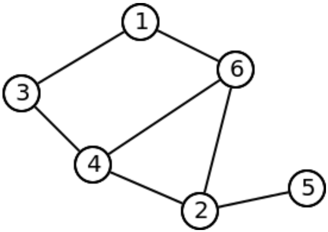


$1 \rightarrow 6 \rightarrow 2 \rightarrow 4$

- **Cycle**: path that starts and ends on the same node (ex: for the above graph, $1 \rightarrow 6 \rightarrow 4 \rightarrow 3 \rightarrow 1$ is a cycle)
- **Acyclic graph**: graph without cycles
- **DAG**: directed acyclic graph

Terminology

- **Size** of a path: number of edges in the path
- **Cost** of a path: if the graph is weighted, we can talk about the cost, which is the sum of the edge weights
- **Distance**: size/cost of the minimum path between two nodes
- **Diameter** of a graph: max distance between two nodes of a graph



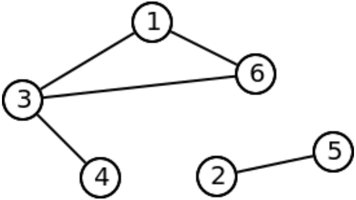
Diameter = 3

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

Distances between nodes

Terminology

- **Connected Component**: Subset of nodes where there is at least one path between each of them
- **Connected Graph**: Graph with just one connected component (there is a path between all pairs of nodes)

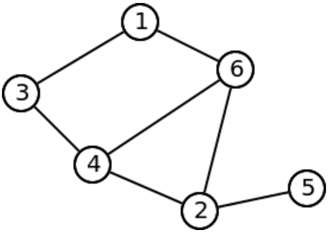


Graph with two connected components: $\{1, 3, 4, 6\}$ e $\{2, 5\}$

(this drawing is from an undirected graph; for directed graphs we can talk about **strongly connected components** - we will talk about them later)

Terminology

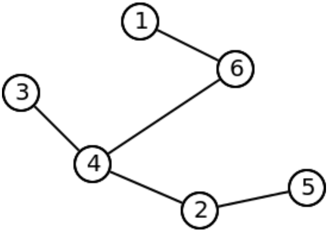
- **Subgraph**: subset of nodes and the edges between them
- **Complete** graph: with links between all pairs of nodes
- **Clique**: a complete subgraph
- **Triangle**: a clique with 3 nodes



Subgraph examples: $\{1, 3\}$, $\{1, 6, 2\}$, $\{2, 4, 5, 6\}$, etc
Example clique: $\{2, 4, 6\}$ (a triangle)

Terminology

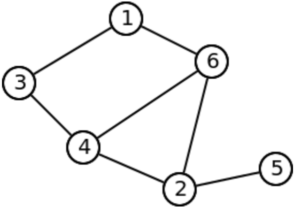
- **Tree**: simple, connected acyclic graph
  (if it has n nodes, then it will have $n - 1$ edges)
- **Forest**: set of multiple disconnected trees



Graph Representation

How to represent a graph?

- **Adjacency Matrix**: $|V| \times |V|$ matrix where the (i, j) cell indicates if there is a link between nodes i and j *(an undirected graph has a symmetric matrix)*; if the graph is weighted we can store the weight
- **Adjacency list**: each node stores a list of its neighbors; if the graph is weighted we have to store pairs (destination,weight)



	1	**2**	**3**	**4**	**5**	**6**
1			X			X
2				X	X	X
3	X			X		
4		X	X			X
5		X				
6	X	X		X		

Adjacency Matrix

(e.g. bool[][])

1: $3 \rightarrow 6$
2: $4 \rightarrow 5 \rightarrow 6$
3: $1 \rightarrow 4$
4: $2 \rightarrow 3 \rightarrow 6$
5: 2
6: $1 \rightarrow 2 \rightarrow 4$

Adjacency List

(e.g. vector<list>)

Graph Representation

Some pros and cons:

- **Adjacency Matrix**:
  - Very simple to implement
  - Quick to check if there is a connection between two nodes - $\mathcal{O}(1)$
  - Slow to traverse the neighbors - $\mathcal{O}(|V|)$
  - Lots of memory wasted (in sparse graphs) - $\mathcal{O}(|V|^2)$
  - Weighted graph implies simply to store the weight in the matrix
  - Adding/Removing edges is simply changing a cell - $\mathcal{O}(1)$

- **Adjacency List**:
  - Slow to see if there is a link between u and v - $\mathcal{O}(\text{degree(u)})$
  - Quick to traverse the neighbors - $\mathcal{O}(\text{degree(u)})$
  - Efficient usage of memory - $\mathcal{O}(|V| + |E|)$
  - Weighted graph implies adding an attribute to the list
  - Removing edge (u, v) implies traversing the list - $\mathcal{O}(\text{degree(u)})$

    Note: we can use for instante BSTs (set/map) to improve the efficiency of searching and removing to $\mathcal{O}(\log \mathit{degree}(u))$

Graph datasets

Here are some interesting websites with graphs:

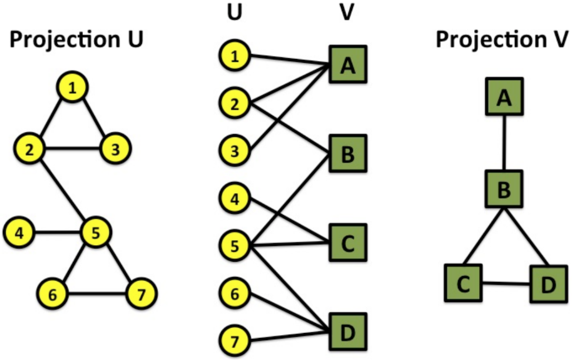
- **Network Repository:** http://networkrepository.com/
- **Konect:** http://konect.cc/
- **SNAP:** https://snap.stanford.edu/data/

Data & Network Collections. Find and interactively VISUALIZE and EXPLORE hundreds of network data

ANIMAL SOCIAL NETWORKS	816	INTERACTION NETWORKS	29	SCIENTIFIC COMPUTING	11
BIOLOGICAL NETWORKS	37	INFRASTRUCTURE NETWORKS	8	SOCIAL NETWORKS	77
BRAIN NETWORKS	116	LABELED NETWORKS	105	FACEBOOK NETWORKS	114
COLLABORATION NETWORKS	20	MASSIVE NETWORK DATA	21	TECHNOLOGICAL NETWORKS	12
CHEMINFORMATICS	646	MISCELLANEOUS NETWORKS	2668	WEB GRAPHS	36
CITATION NETWORKS	4	POWER NETWORKS	8	DYNAMIC NETWORKS	115
ECOLOGY NETWORKS	6	PROXIMITY NETWORKS	13	TEMPORAL REACHABILITY	38
ECONOMIC NETWORKS	16	GENERATED GRAPHS	221	BHOSLIB	36
EMAIL NETWORKS	6	RECOMMENDATION NETWORKS	36	DIMACS	78
GRAPH 500	8	ROAD NETWORKS	15	DIMACS10	84
HETEROGENEOUS NETWORKS	15	RETWEET NETWORKS	34	NON-RELATIONAL ML DATA	211

Bipartite Graphs

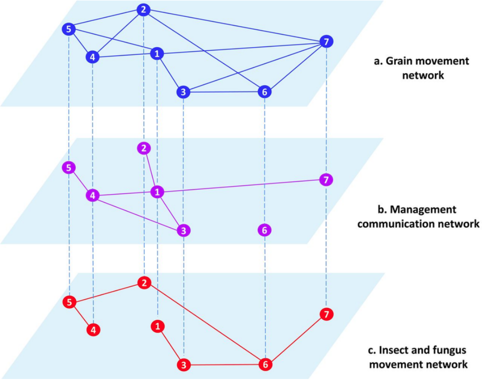
- A **bipartite graph** is a graph whose nodes can be divided into two disjoint sets U and V such that every edge connects a node in U to one in V



- Many (real world) networks come from projections (ex: actors and movies, diseases and genes)

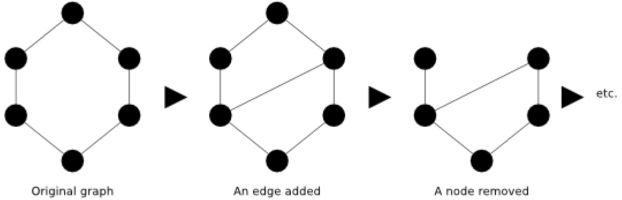
Other Graph Types: Multilayer / Multiplex

- Graphs can have different layers



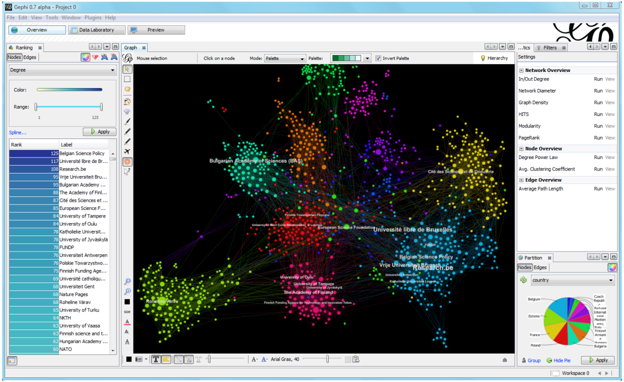
Other Graph Types: Temporal Networks

- Graphs can evolve over time



Extra: Graph Software (not needed for this course)

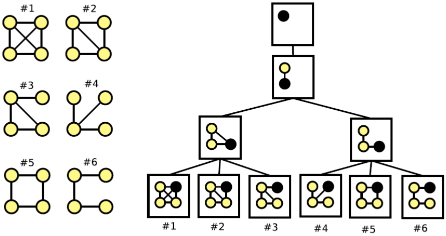
- Software such as **Gephi** or **Cytoscape** would allow you to visualize (and interact) with graphs



- Software packages such as **networkx** (Python) or **igraph** (C/C++, R, Python, Mathematica) provide graph implementations and many metrics and algorithms (they are *Network Science* packages)

Network Science / Graph Mining

My main research area



PhD Thesis (2011):

Efficient and Scalable Algorithms for Network Motifs Discovery

Publications: [personal website] [google scholar]

(and for instance a **Network Science course** at masters level)