

Online Social Networks and Media

Introduction

Instructors:

Ευαγγελία Πιτουρά

<http://www.cs.uoi.gr/~pitoura>

Παναγιώτης Τσαπάρας

<http://www.cs.uoi.gr/~tsap>

Goal

Understand the importance of networks in life, technology and applications

Study the theory underlying social networks

Learn about algorithms that make use of network structure

Learn about the tools to analyze them

Today:

A taste of the topics to be covered

Some logistics

Some basic graph theory

Logistics

Textbooks:

Easley and Kleinberg free text-book on [Networks, Crowds and Markets](#)

M. E. J. Newman, [The structure and function of complex networks](#), SIAM Reviews, 45(2): 167-256, 2003

Reza Zafarani, Mohammad Ali Abbasi, Huan Liu, free text-book on [Social Media Mining](#)

Web page:

www.cs.uoi.gr/~tsap/teaching/cs-l14

20% Presentations and class participation

30% Assignments

50% Term Project (in 2 Phases)

No Final Exam

WHAT DO THE FOLLOWING
COMPLEX SYSTEMS HAVE IN
COMMON?

The Economy



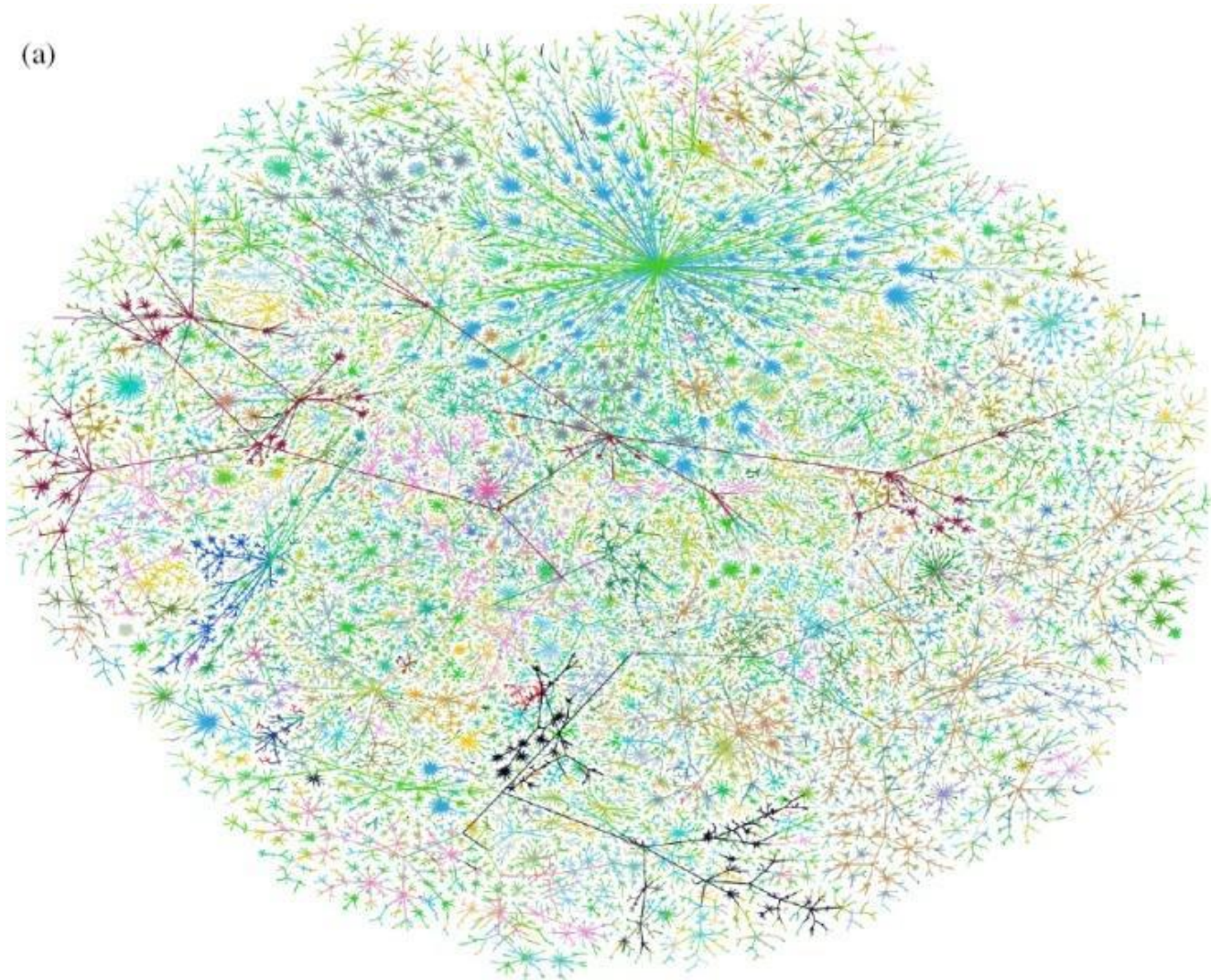
The Human Cell



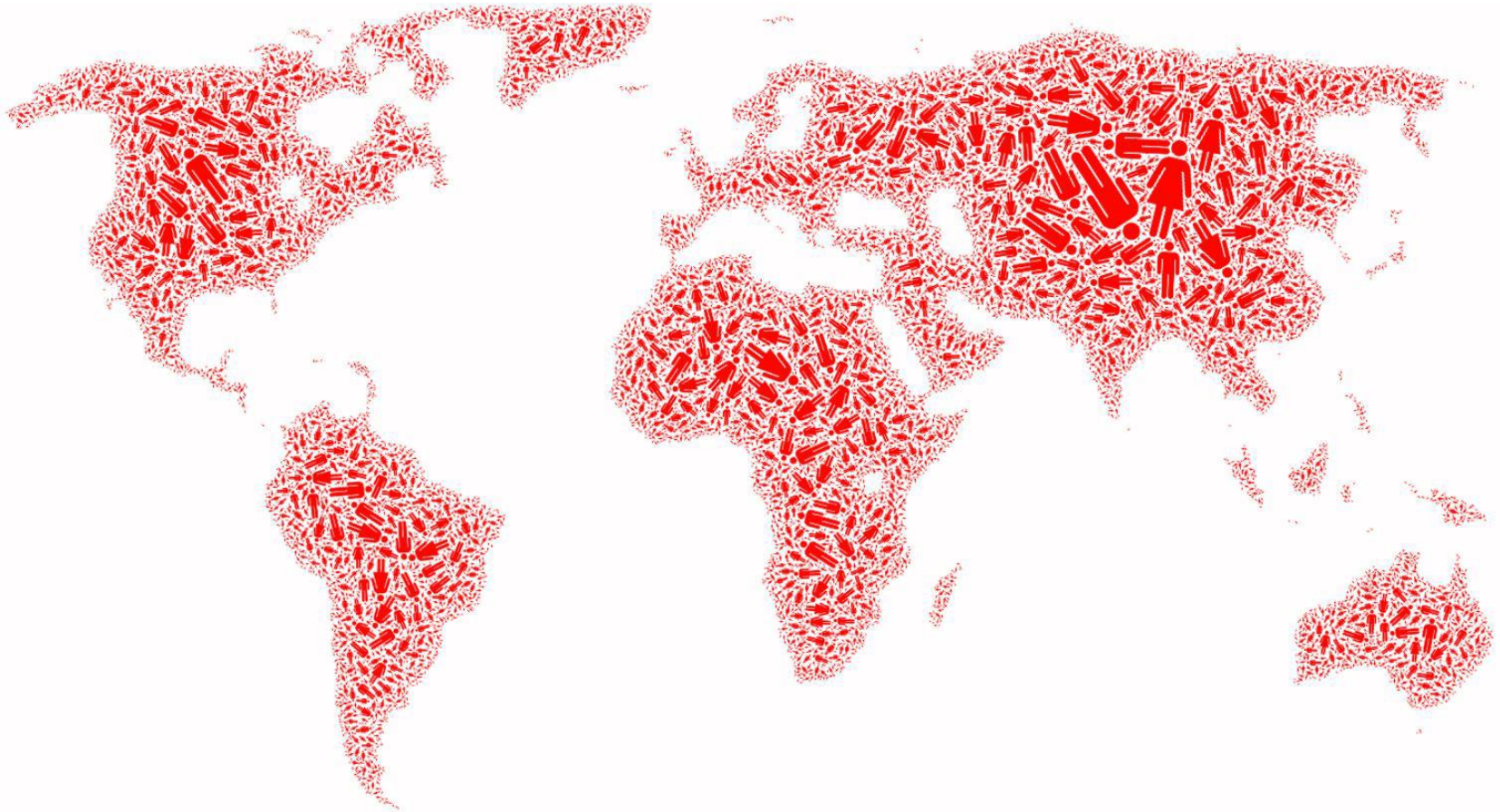
Traffic and roads



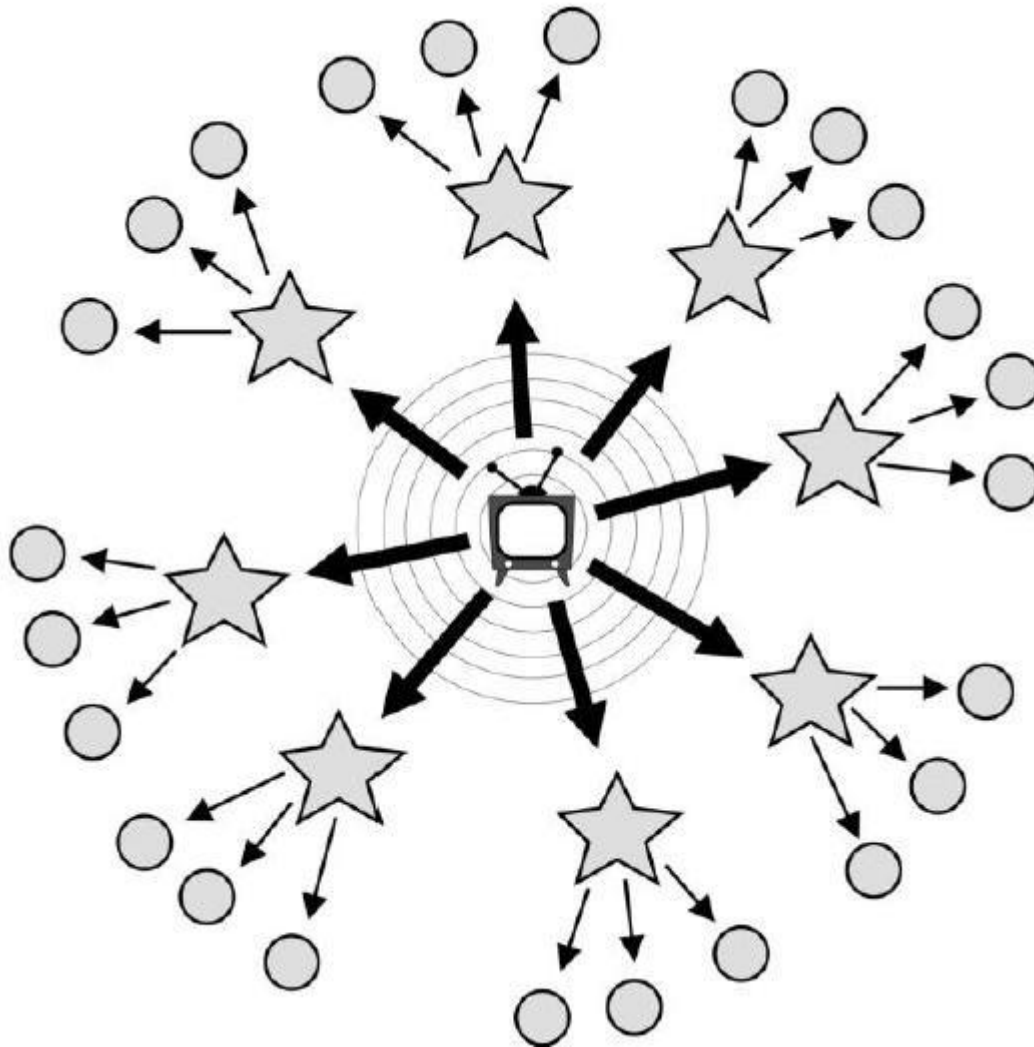
Internet



Society



Media and Information



THE NETWORK!

All of these systems can be modeled as
networks

What is a network?

- Network: a collection of **entities** that are interconnected with **links**.

Social networks



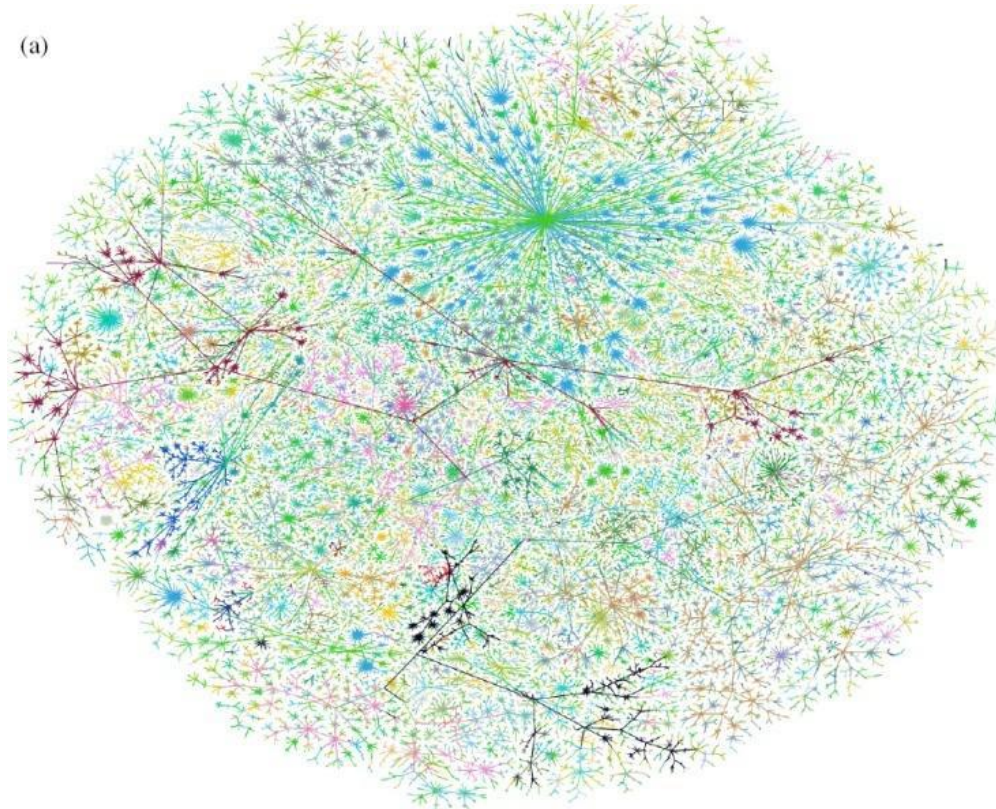
- **Entities:** People
- **Links:** Friendships

Communication networks



- **Entities:** People
- **Links:** email exchange

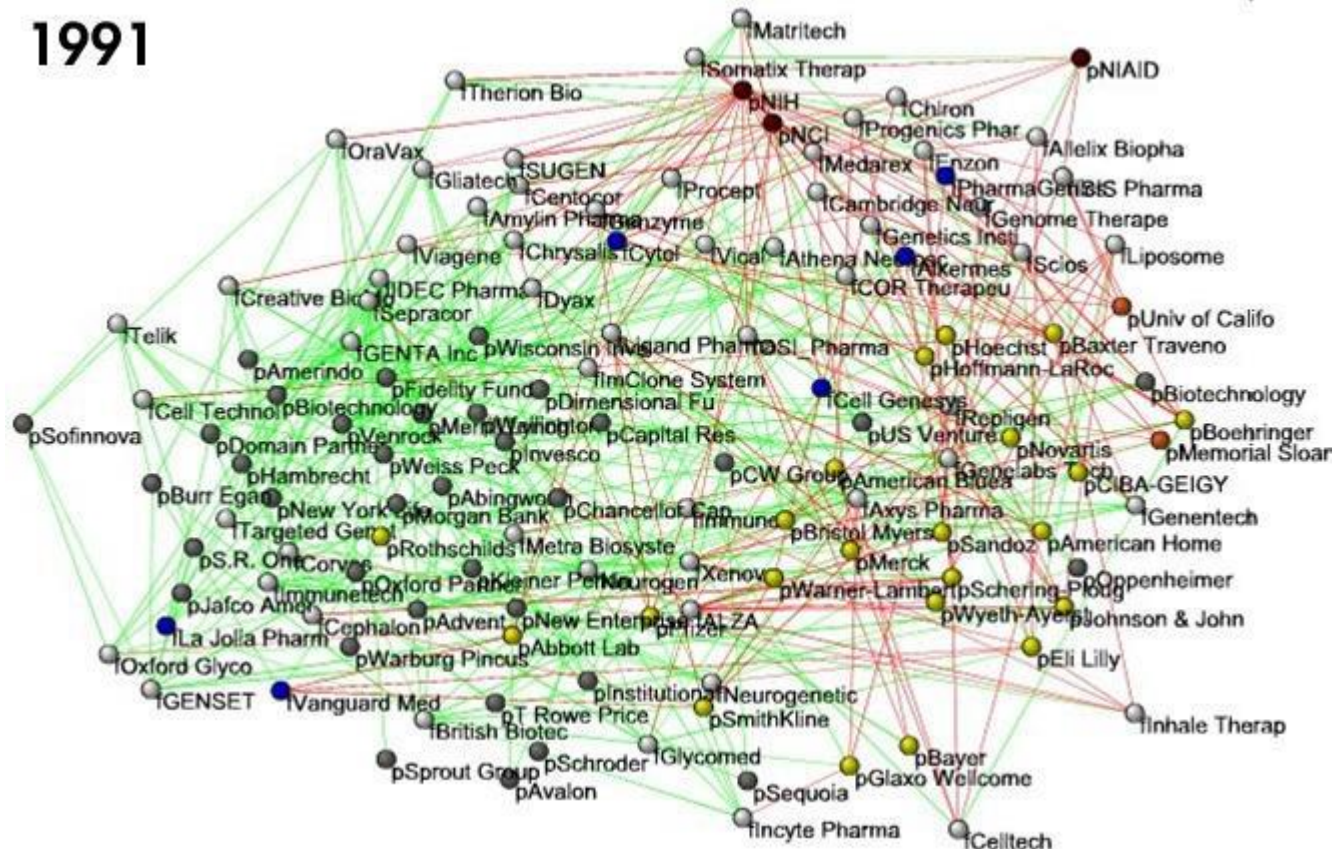
Communication networks



- **Entities:** Internet nodes
- **Links:** communication between nodes

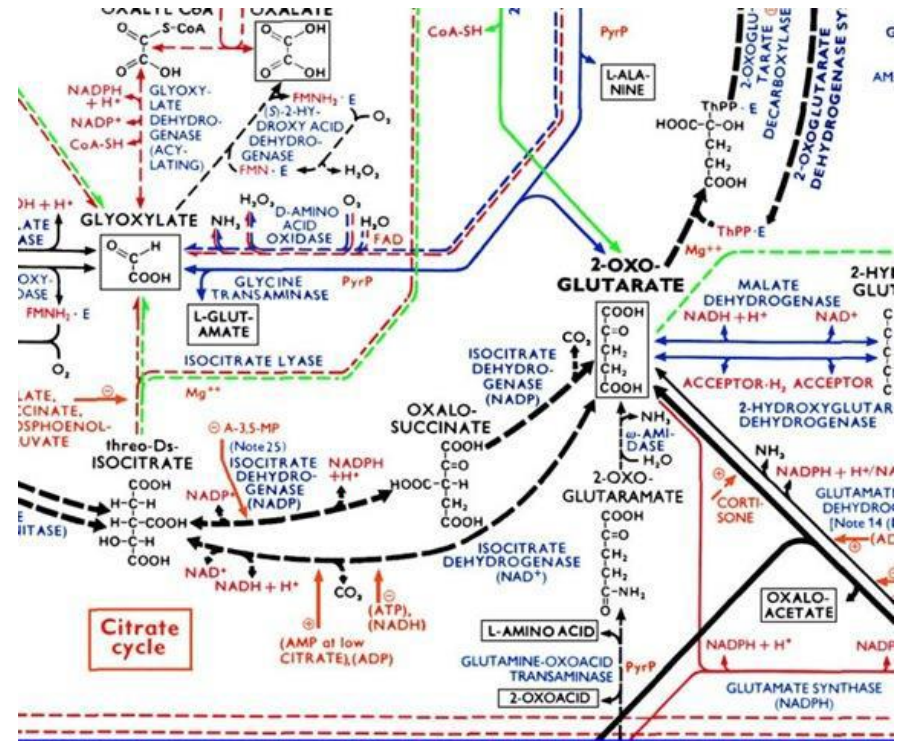
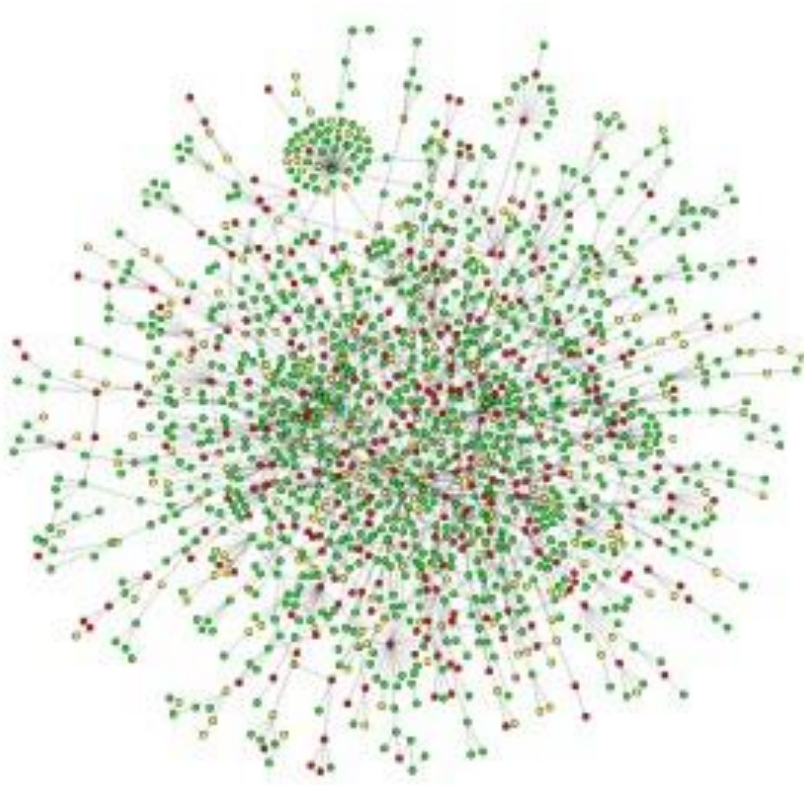
Financial Networks

1991



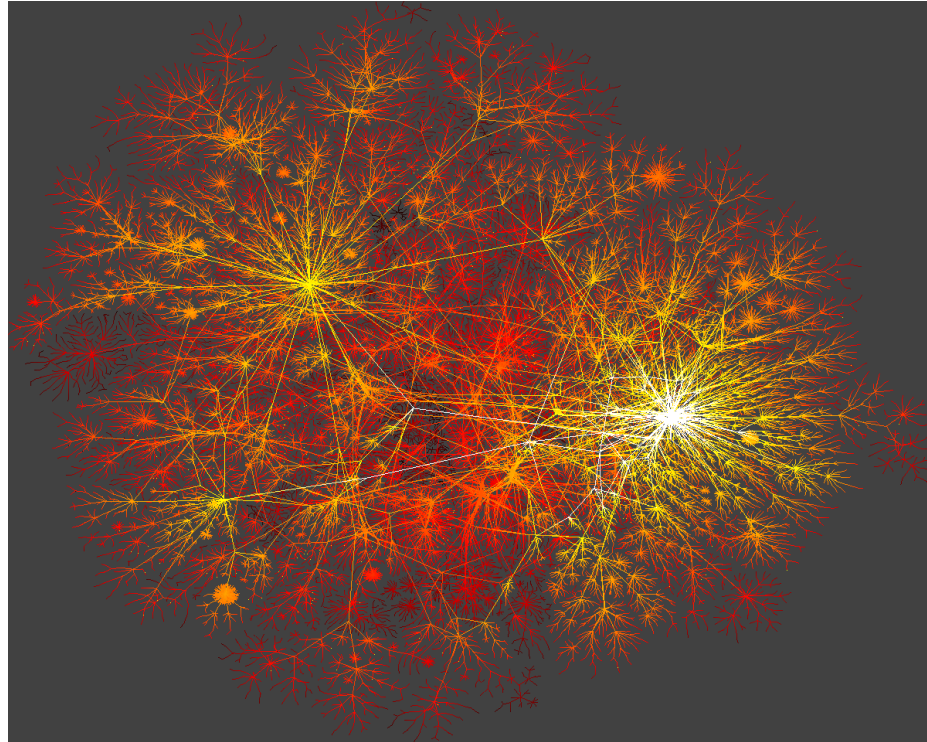
- Entities: Companies
- Links: relationships (financial, collaboration)

Biological networks



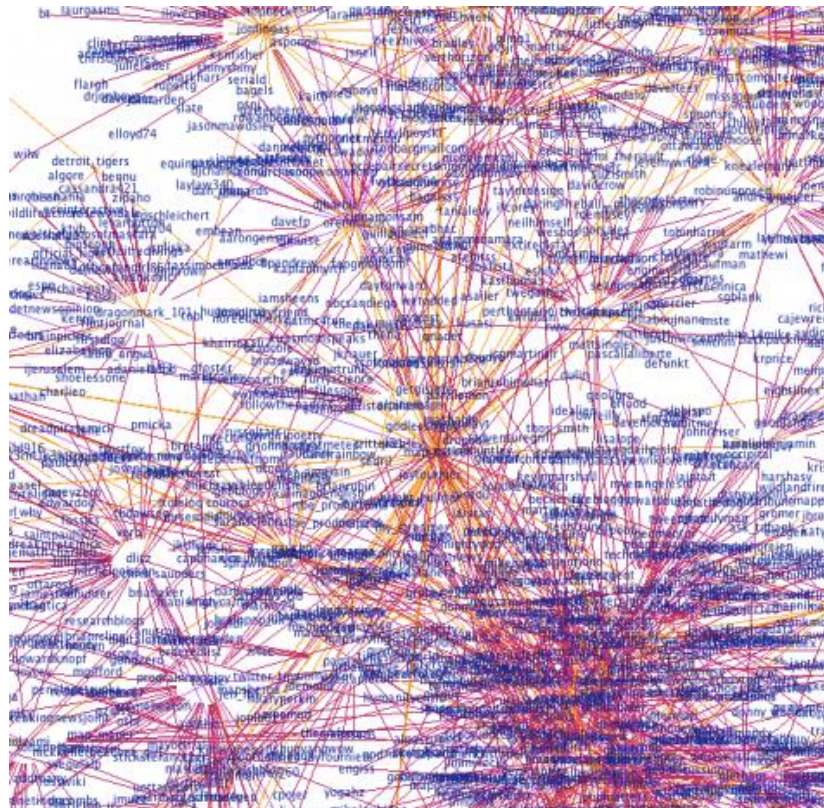
- **Entities:** Proteins
- **Links:** interactions
- **Entities:** metabolites, enzymes
- **Links:** chemical reactions

Information networks



- **Entities:** Web Pages
- **Links:** Links

Information/Media networks



- **Entities:** Twitter users
- **Links:** Follows/conversations

Many more

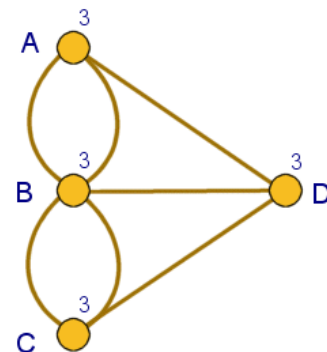
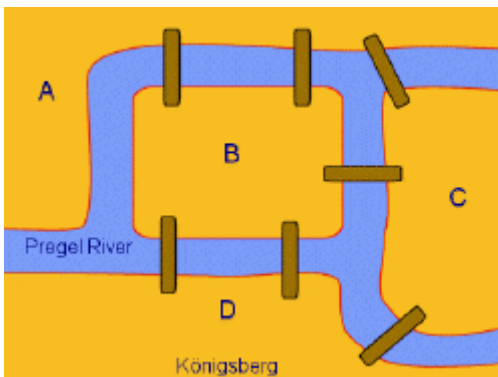
- Wikipedia
- Brain
- Highways
- Software
- Etc...

Why networks are important?

- We cannot truly understand a **complex system** unless we understand the **underlying network**.
 - Everything is **connected**, studying individual entities gives only a partial view of a system
- Two main themes:
 - What are the **structural properties** of the network?
 - How do **processes** happen in the network?

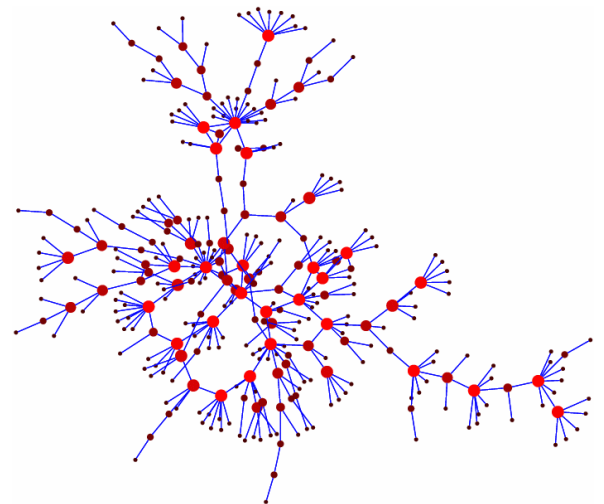
Graphs

- In mathematics, networks are called **graphs**, the entities are **nodes**, and the links are **edges**
- Graph theory starts in the 18th century, with Leonhard Euler
 - The problem of Königsberg bridges
 - Since then graphs have been studied extensively.



Networks in the past

- Graphs have been used in the past to model existing networks (e.g., networks of highways, social networks)
 - usually these networks were **small**
 - network can be studied **visual inspection** can reveal a lot of information



Networks now

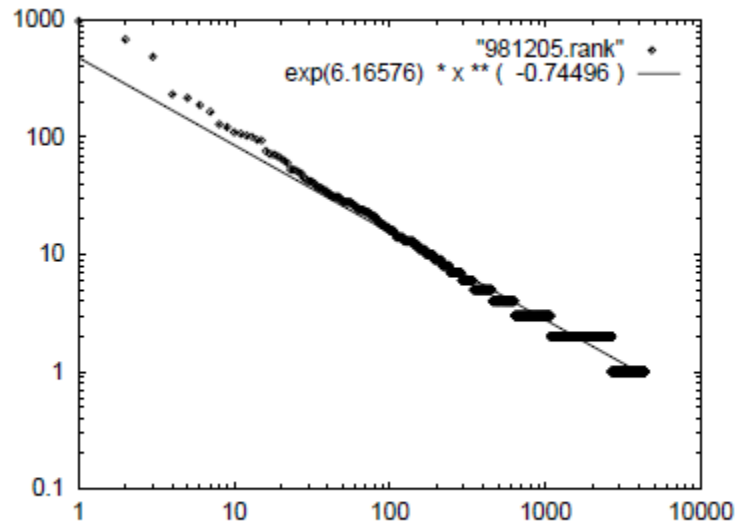
- **More** and **larger** networks appear
 - Products of **technology**
 - e.g., Internet, Web, Facebook, Twitter
 - Result of our ability to collect **more, better, and more complex** data
 - e.g., gene regulatory networks
 - Result of the willingness of **users** to **contribute** data
 - e.g., users making their relationships public online
- Networks of thousands, millions, or **billions** of nodes
 - Impossible to process visually
 - Problems become harder
 - Processes are more complex

Topics

- Measuring Real Networks
- Modeling the evolution and creation of networks
- Identifying important nodes in the graph
- Understanding information cascades and virus contagions
- Finding communities in graphs
- Link Prediction
- Storing and processing huge networks
- Network embeddings
- Other special topics

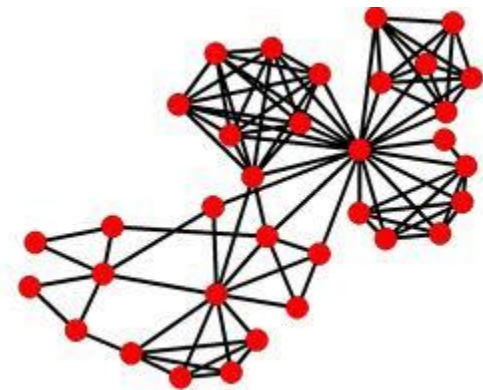
Understanding large graphs

- What does a network **look like**?
 - Measure different properties to understand the structure



(a) Int-12-98

degree of nodes



Triangles in the graph

Real network properties

- Most nodes have only a small number of neighbors (degree), but there are some nodes with very high degree (**power-law degree distribution**)
 - **scale-free** networks
- If a node **x** is connected to **y** and **z**, then **y** and **z** are likely to be connected
 - high **clustering coefficient**
- Most nodes are just a few edges away on average.
 - **small world** networks
- Networks from very diverse areas (from internet to biological networks) have similar properties
 - Is it possible that there is a unifying underlying generative process?

Generating random graphs

- **Classic graph theory** model (Erdős-Renyi)
 - each edge is generated independently with probability p
- Very well studied model but:
 - most vertices have about the same degree
 - the probability of two nodes being linked is independent of whether they share a neighbor
 - the average paths are short

Modeling real networks

- Real life networks are **not “random”**
- Can we define a **model** that **generates** graphs with statistical properties similar to those in real life?
- The **rich-get-richer** model

We need to accurately model the mechanisms that govern the evolution of networks (for prediction, simulations, understanding)

Ranking of nodes on the Web

- Is my home page as important as the facebook page?
- We need algorithms to compute the **importance of nodes** in a graph
- The **PageRank** Algorithm
 - A success story of network use



It is impossible to create a web search engine without understanding the web graph

Information/Virus Cascade

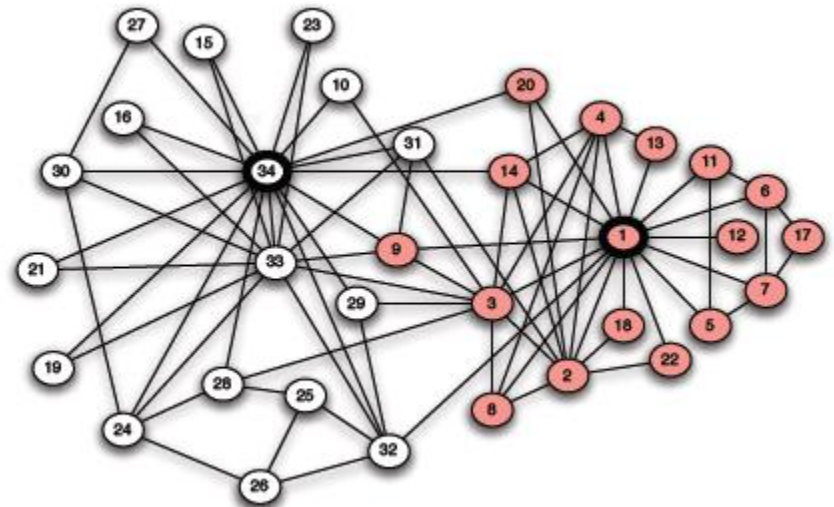
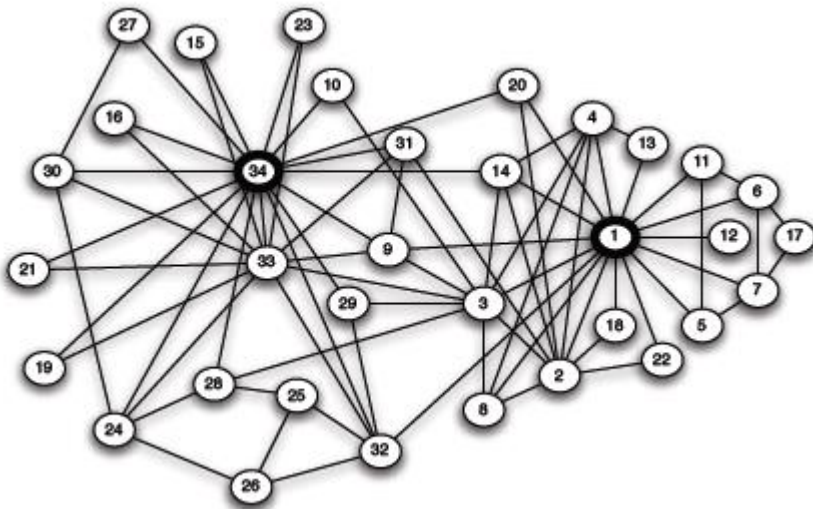
- How do **viruses spread** between individuals? How can we stop them?
- How does **information propagate** in social and information networks? What items become **viral**? Who are the **influencers** and trend-setters?
- We need **models and algorithms** to answer these questions

Online advertising relies heavily on online social networks and word-of-mouth marketing. There is currently need for models for understanding the spread of Ebola virus.

Clustering and Finding Communities

- What is **community**?
 - “Cohesive subgroups are subsets of actors among whom there are relatively strong, direct, intense, frequent, or positive ties.” [Wasserman & Faust '97]

Karate club example [W. Zachary, 1970]



Clustering and Finding Communities

- Input: a graph $G=(V,E)$
 - edge (u, v) denotes similarity between u and v
 - *weighted graphs*: weight of edge captures the degree of *similarity*
- **Clustering**: Partition the nodes in the graph such that nodes **within** clusters are **well interconnected** (high edge weights), and nodes **across** clusters are **sparsely interconnected** (low edge weights)

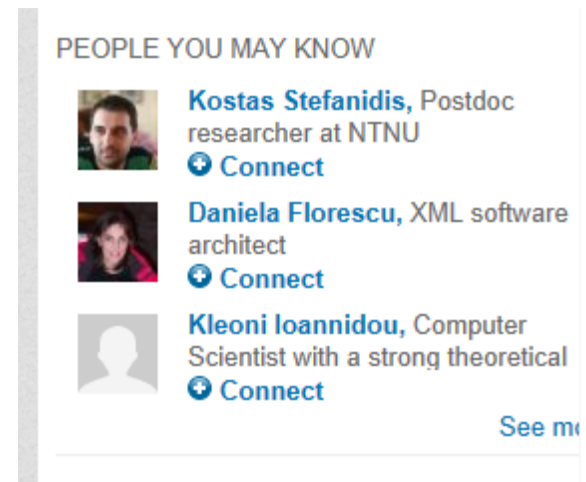
Community Evolution

- **Homophily:** “Birds of a feather flock together”
- Caused by two related social forces [Friedkin98, Lazarsfeld54]
 - *Social influence:* People become similar to those they interact with
 - *Selection:* People seek out similar people to interact with
- Both processes contribute to homophily, but
 - Social influence leads to community-wide homogeneity
 - Selection leads to fragmentation of the community
- Applications in online marketing
 - *viral marketing* relies upon social influence affecting behavior
 - *recommender systems* predict behavior based on similarity

How do we define and discover communities in large graphs? How do communities evolve?

Link Prediction

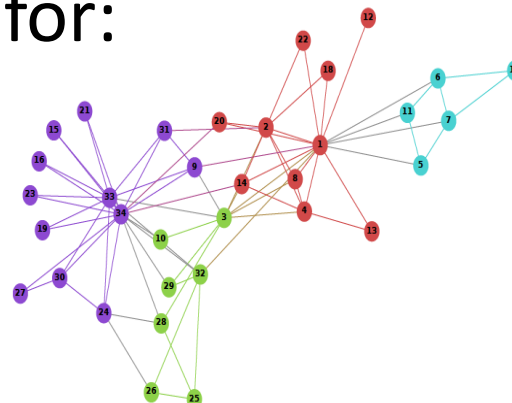
- Given a snapshot of a social network at time t , we seek to accurately **predict** the edges that will be added to the network during the interval from time t to a given future time t' .
- **Applications:**
 - Accelerate the growth of a social network (e.g., Facebook, LinkedIn, Twitter) that would otherwise take longer to form.
 - Identify suspect relationships



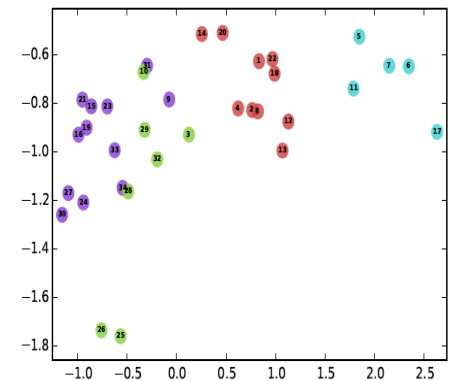
How do we predict future links?

Network Embeddings

- Project the nodes of the network into a **multidimensional real space**, such that the nodes that are close to each other in the network (or, semantically similar) are mapped to near-by points in this space
- An application of machine learning on graphs.
- Multiple applications for:
 - Link prediction
 - Node classification
 - More



Zachary's Karate Club Network:



2-dimensional node embedding

Network content

- Users on online social networks generate **content**.
- Mining the content **in conjunction** with the **network** can be useful
 - Do friends post similar content on Facebook?
 - Can we understand a user's interests by looking at those of their friends?
 - The importance of homophily
 - Social recommendations: Can we predict a movie rating using the social network?

Social Media

- Today **Social Media** (Twitter, Facebook, Instagram) have supplanted the traditional media sources
 - Information is generated and disseminated mostly online by users
 - E.g., the assassination of Bin Laden appeared first on Twitter
 - Twitter has become a global “**sensor**” detecting and reporting everything
- Interesting problems:
 - Automatically detect events using Twitter
 - Earthquake prediction
 - Crisis detection and management
 - Sentiment mining
 - Track the evolution of events: socially, geographically, over time.

Tools

NetworkX: a Python language software package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks.

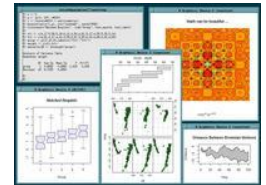
<http://networkx.lanl.gov/>



Stanford Network Analysis Platform (SNAP): general purpose, high performance system for analysis and manipulation of large networks written in C++ <http://snap.stanford.edu/snap/index.html>



R: free software environment for statistical computing and graphics. <http://www.r-project.org/>



Gephi: interactive visualization and exploration platform for all kinds of networks and complex systems, dynamic and hierarchical graphs <http://gephi.org/>



Frameworks for Processing Large Graphs

Large scale (in some cases billions of vertices, trillions of edges)

How to process graphs in parallel?

- Write *your own code*
- Use *MapReduce (general parallel processing)* *
- *Pregel* (bulk synchronous parallel model) introduced by Google in 2010*
- *Giraph* <http://incubator.apache.org/giraph/> (part of Hadoop software)

Storage?

*J. Dean, S. Ghemawat. *MapReduce: Simplified Data Processing on Large Clusters*. OSDI 2004: 137-150

** G. Malewicz, M. H. Austern, A. J. C. Bik, J. C. Dehnert, I. Horn, N. Leiser: *Pregel: a system for large-scale graph processing*. SIGMOD Conference 2010: 135-146

Data

Collected using available APIs (Twitter, Facebook, etc)

Using existing collections, e.g., from SNAP (more in the webpage), permission may be required

Stanford Large Network Dataset Collection

[60 large social and information network datasets](#)

Coauthorship and Citation Networks

[DBLP](#): Collaboration network of computer scientists

[KDD Cup Dataset](#)

Internet Topology

[AS Graphs](#): AS-level connectivities inferred from Oregon route-views, Looking glass data and Routing registry data

Yelp Data

[Yelp Review Data](#): reviews of the 250 closest businesses for 30 universities for students and academics to explore and research

Youtube dataset

[Youtube data](#): YouTube videos as nodes. Edge a->b means video b is in the related video list (first 20 only) of a video a.

Amazon product copurchasing networks and metadata

[Amazon Data](#): The data was collected by crawling Amazon website and contains product metadata and review information about 548,552 different products (Books, music CDs, DVDs and VHS video tapes).

Wikipedia

[Wikipedia page to page link data](#): A list of all page-to-page links in Wikipedia

[DBPedia](#): The DBpedia data set uses a large multi-domain ontology which has been derived from Wikipedia.

[Edits and talks](#): Complete edit history (all revisions, all pages) of Wikipedia since its inception till January 2008.

Movie Ratings

[IMDB database](#): Movie ratings from IMDB

[User rating data](#): Movie ratings from MovieLens

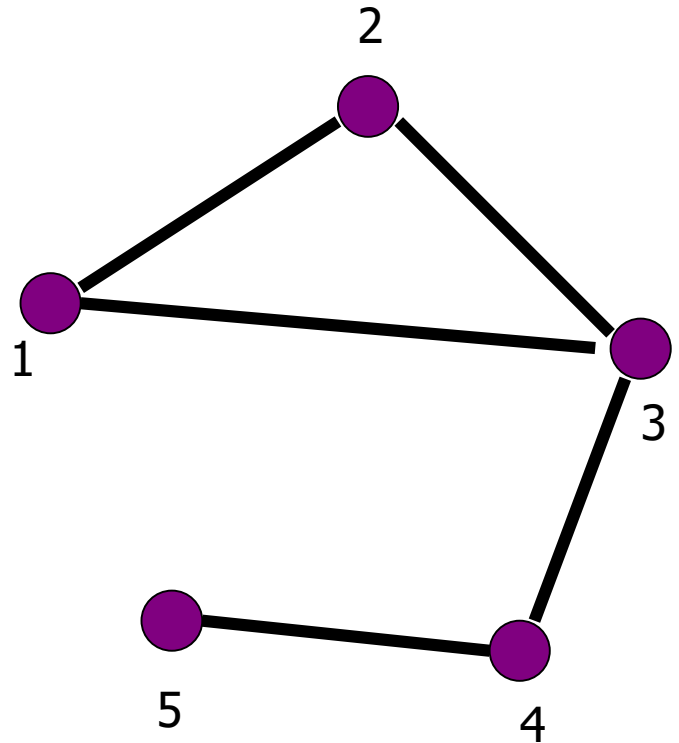
Acknowledgements

- Thanks to Jure Leskovec for some of the material from his course notes.
- M. E. J. Newman, **The structure and function of complex networks**, SIAM Reviews, 45(2): 167-256, 2003

Graph Theory Reminder

Undirected Graph

- Graph $G=(V,E)$
 - V = set of vertices (nodes)
 - E = set of edges



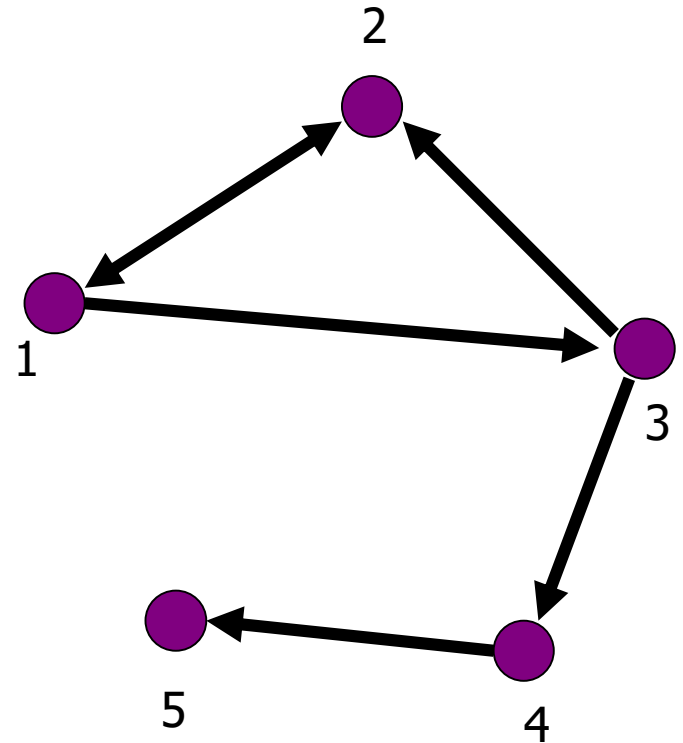
undirected graph

$V = \{1, 2, 3, 4, 5\}$

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

Directed Graph

- Graph $G=(V,E)$
 - V = set of vertices (nodes)
 - E = set of edges



directed graph

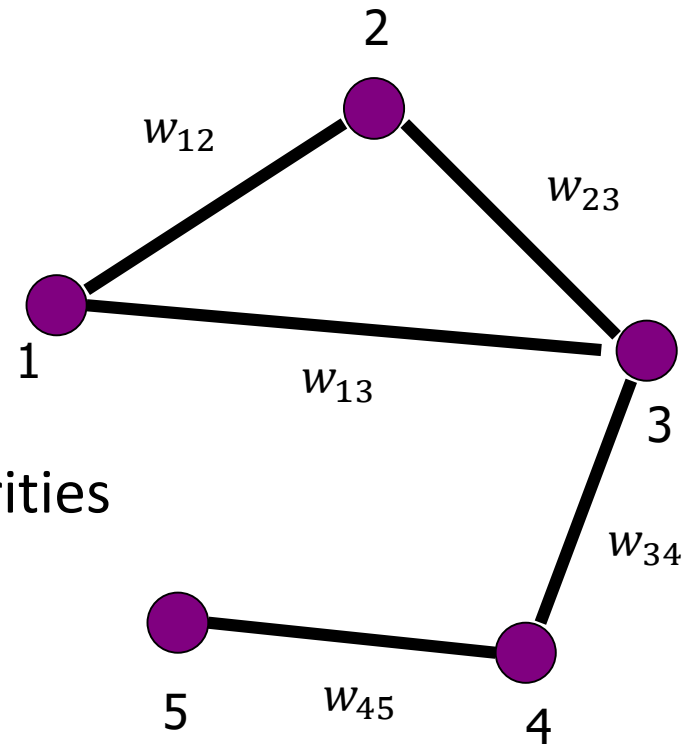
$V = \{1, 2, 3, 4, 5\}$

$E = \{\langle 1,2 \rangle, \langle 2,1 \rangle, \langle 1,3 \rangle, \langle 3,2 \rangle, \langle 3,4 \rangle, \langle 4,5 \rangle\}$

Weighted Graph

- Graph $G=(V,E)$
 - V = set of vertices (nodes)
 - E = set of edges and their weights

Weights can be either distances or similarities



Weighted graph

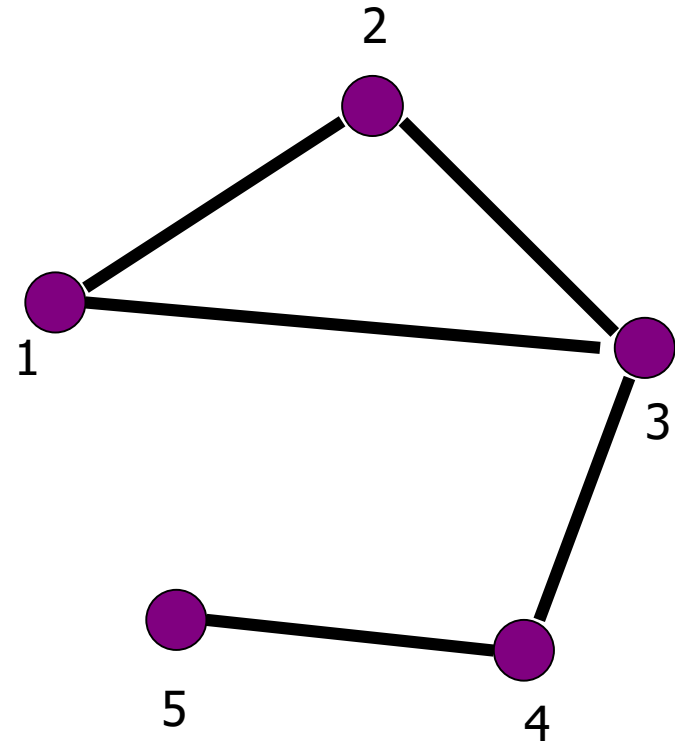
$V = \{1, 2, 3, 4, 5\}$

$E = \{(1,2, w_{12}), (1,3, w_{13}), (2,3, w_{23}), (3,4, w_{34}), (4,5, w_{45})\}$

Undirected graph

- Neighborhood $N(i)$ of node i
 - Set of nodes adjacent to i

- degree $d(i)$ of node i
 - Size of $N(i)$
 - number of edges incident on i



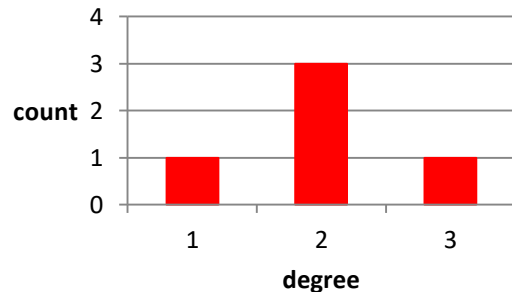
Undirected graph

- degree sequence

- $[d(1), d(2), d(3), d(4), d(5)]$
- $[2, 2, 3, 2, 1]$

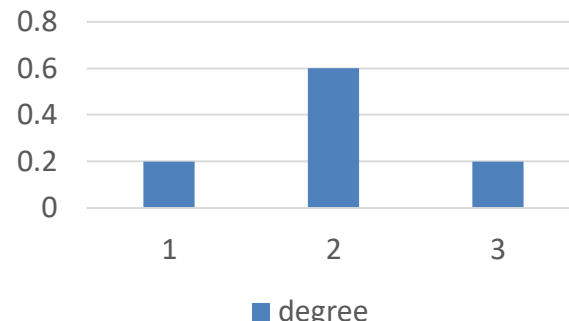
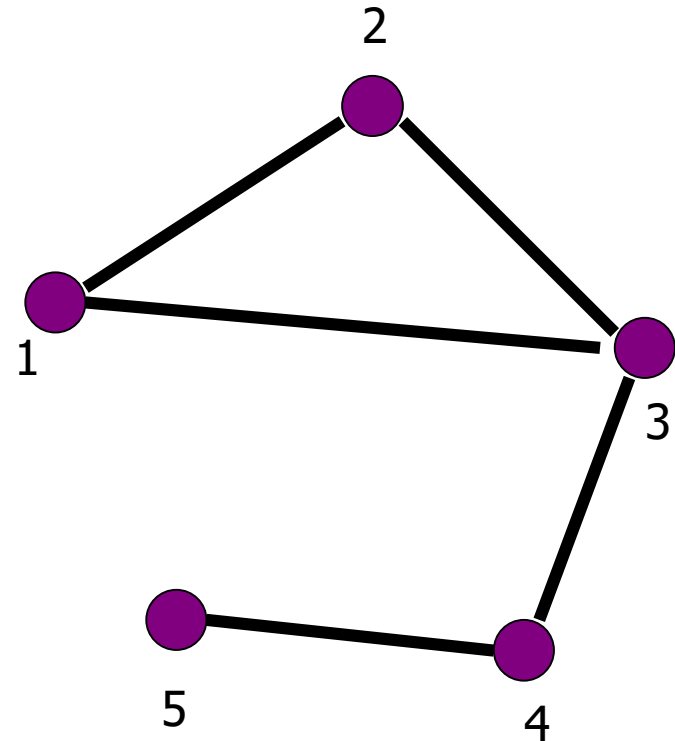
- degree histogram

- $[(1:1), (2:3), (3,1)]$



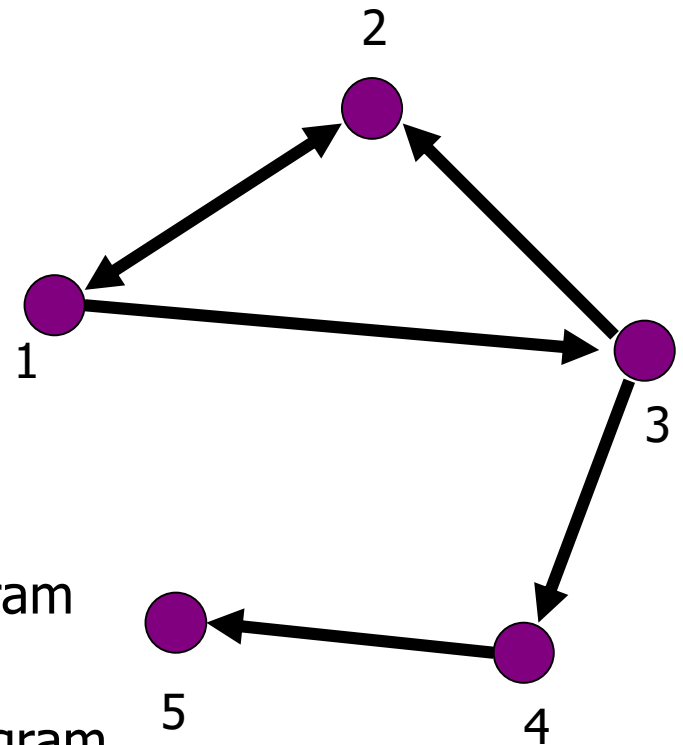
- degree distribution

- $[(1:0.2), (2:0.6), (3,0.2)]$



Directed Graph

- **in-degree** $d_{in}(i)$ of node i
 - number of edges incoming to node i
- **out-degree** $d_{out}(i)$ of node i
 - number of edges leaving node i



- | | |
|-----------------------|------------------------|
| ■ in-degree sequence | ■ in-degree histogram |
| ■ [1,2,1,1,1] | ■ [(1:4),(2:1)] |
| ■ out-degree sequence | ■ out-degree histogram |
| ■ [2,1,2,1,0] | ■ [(0:1),(1:2),(2:2)] |

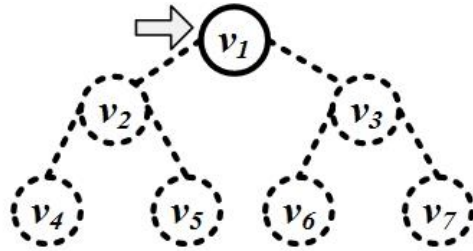
Graph Traversals

- A **traversal** is a procedure for visiting (going through) all the nodes in a graph

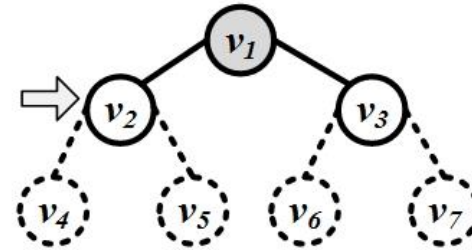
Depth First Search Traversal

- Depth-First Search (**DFS**) starts from a node i , selects one of its neighbors j from $N(i)$ and performs Depth-First Search on j before visiting other neighbors in $N(i)$.
 - The algorithm can be implemented using a *stack structure*

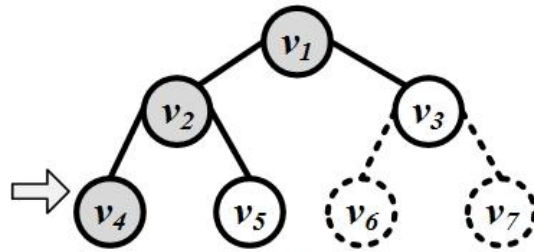
Example for a tree graph



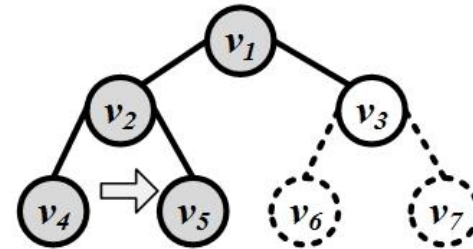
(1)



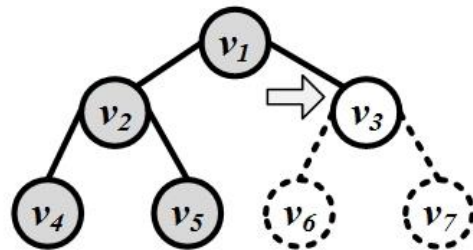
(2)



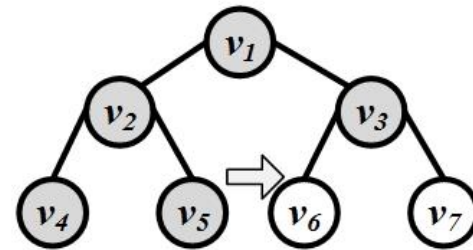
(3)



(4)



(5)

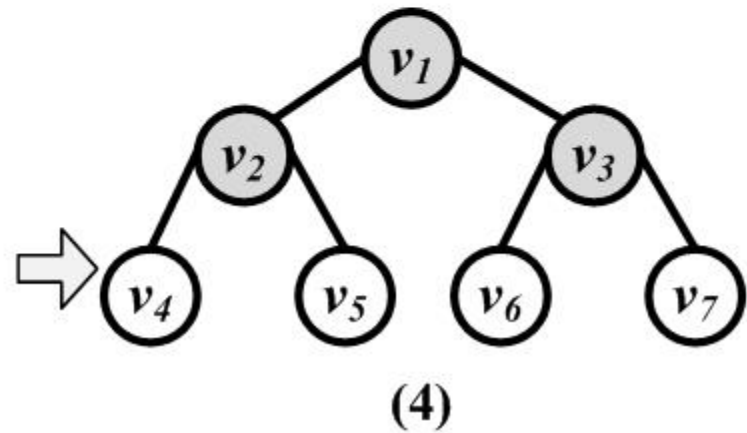
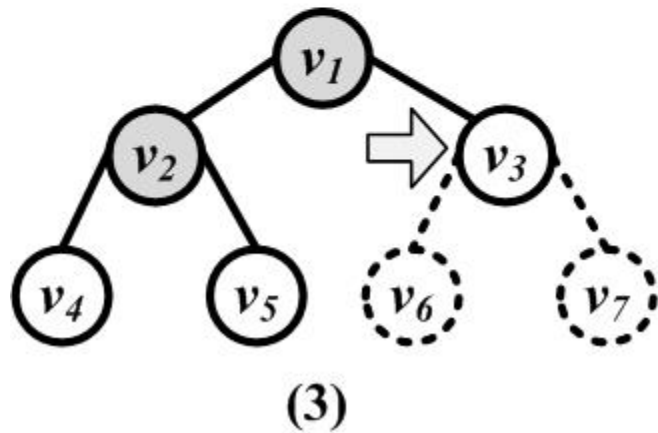
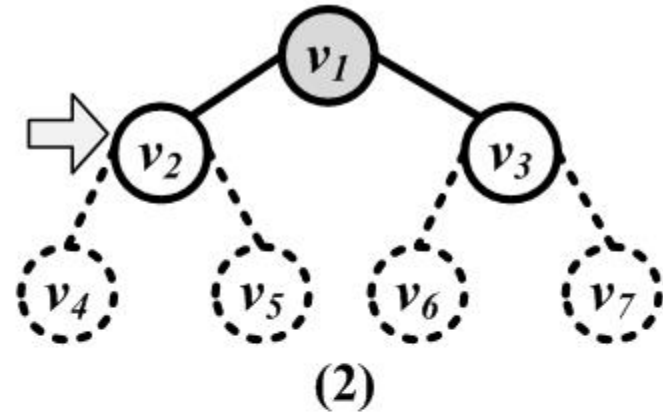
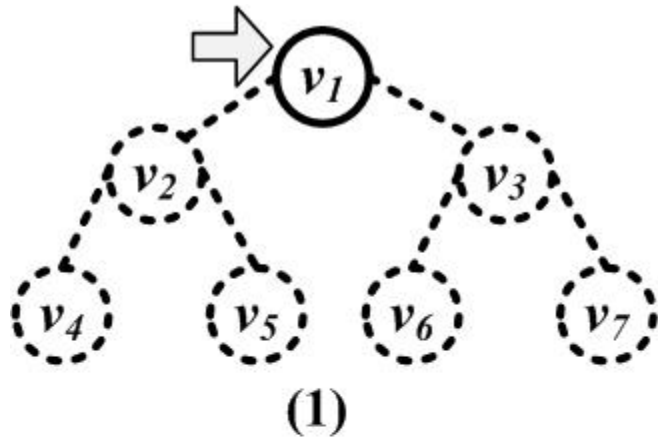


(6)

Breadth First Search Traversal

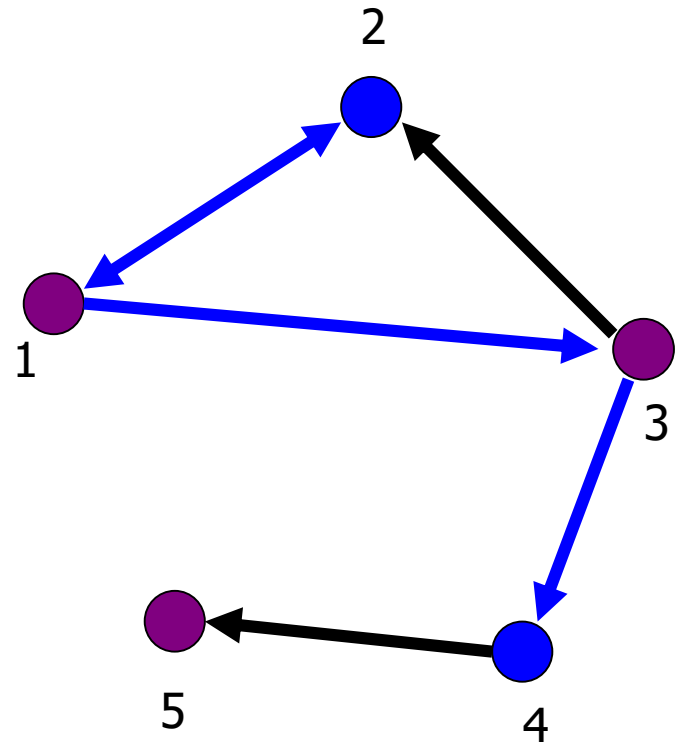
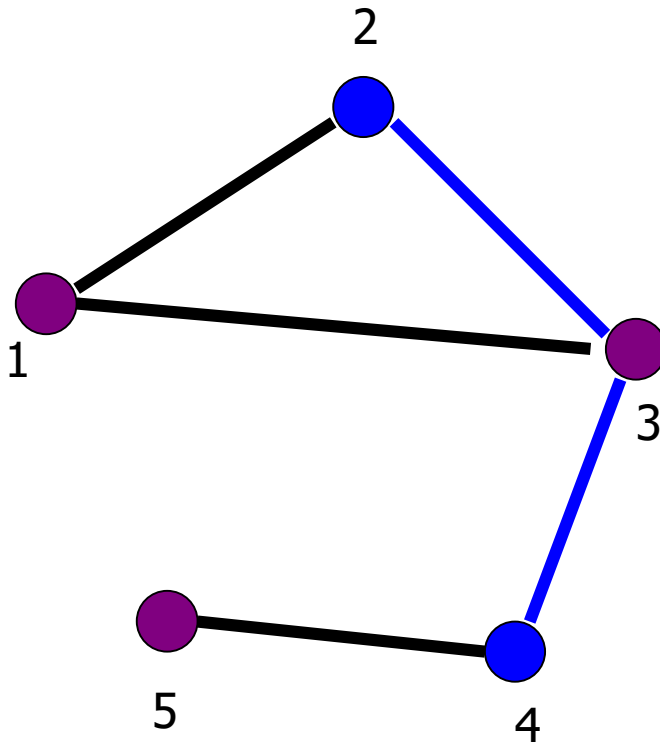
- Breadth-First-Search (**BFS**) starts from a node, visits all its immediate neighbors first, and then moves to the second level by traversing their neighbors.
 - The algorithm can be implemented using a *queue structure*

Example of BFS on a tree



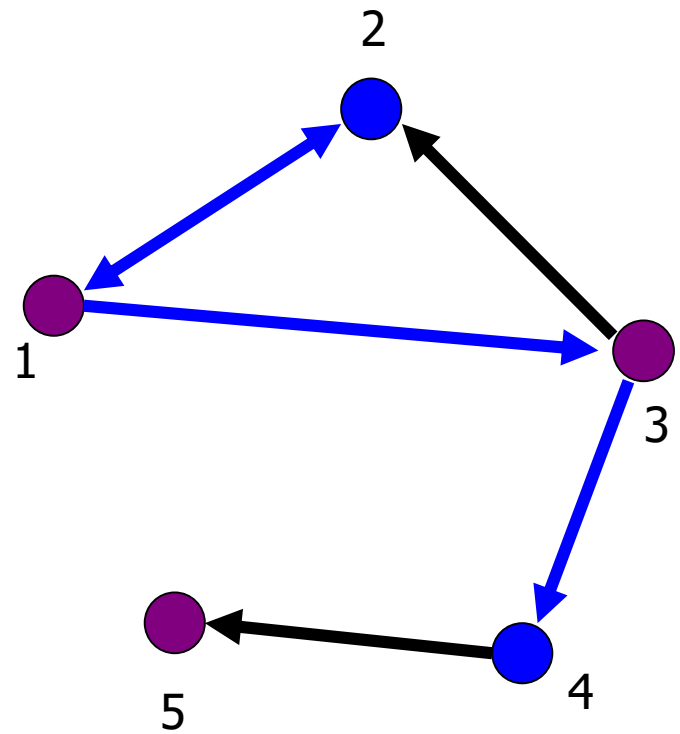
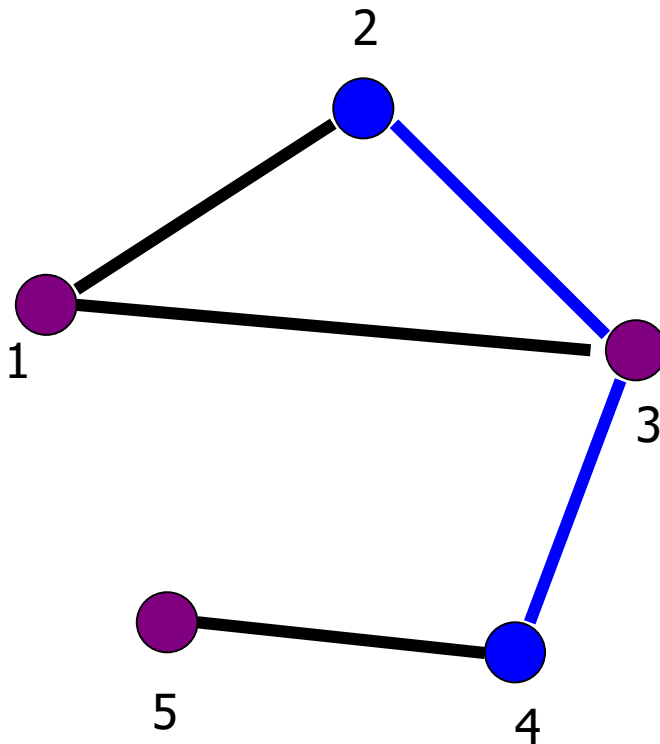
Paths

- Path from node i to node j : a sequence of edges (directed or undirected) from node i to node j
 - path **length**: number of edges on the path
 - nodes i and j are **connected**
 - **cycle**: a path that starts and ends at the same node



Shortest Paths

- Shortest Path from node i to node j
 - also known as **BFS path**, or **geodesic path**
 - We can find all shortest paths from a node using BFS

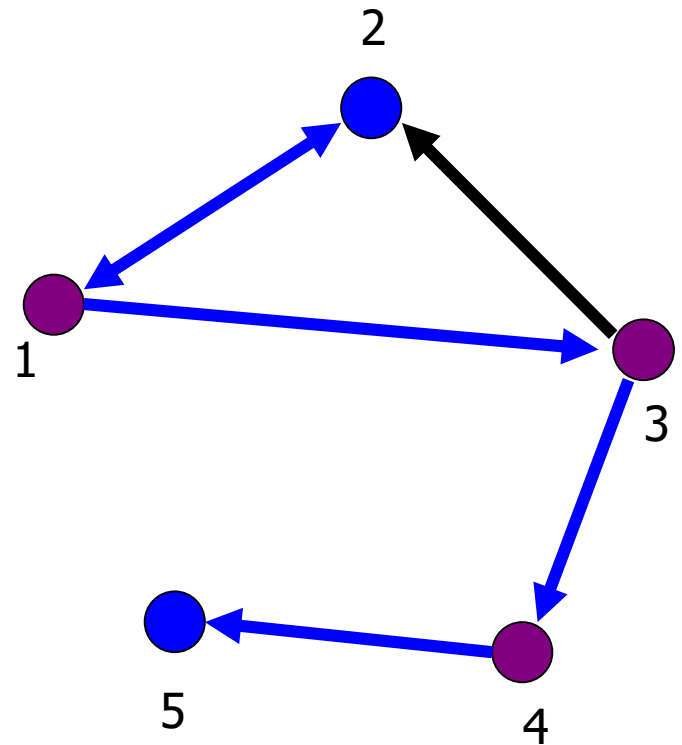
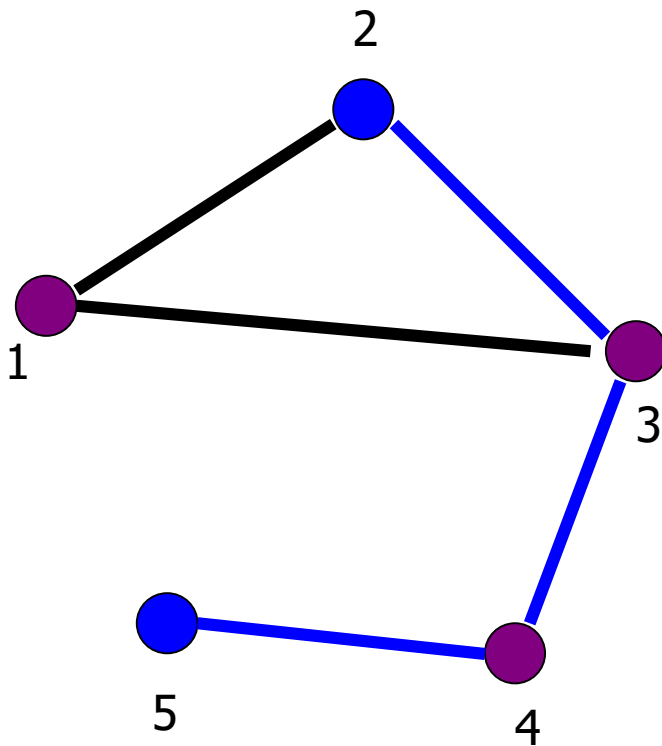


Shortest paths on weighted graphs

- Shortest paths on **weighted** graphs are harder to construct
 - There are several well known algorithms for finding **single-source**, or **all-pairs** shortest paths
 - For example: **Dijkstra's Algorithm**

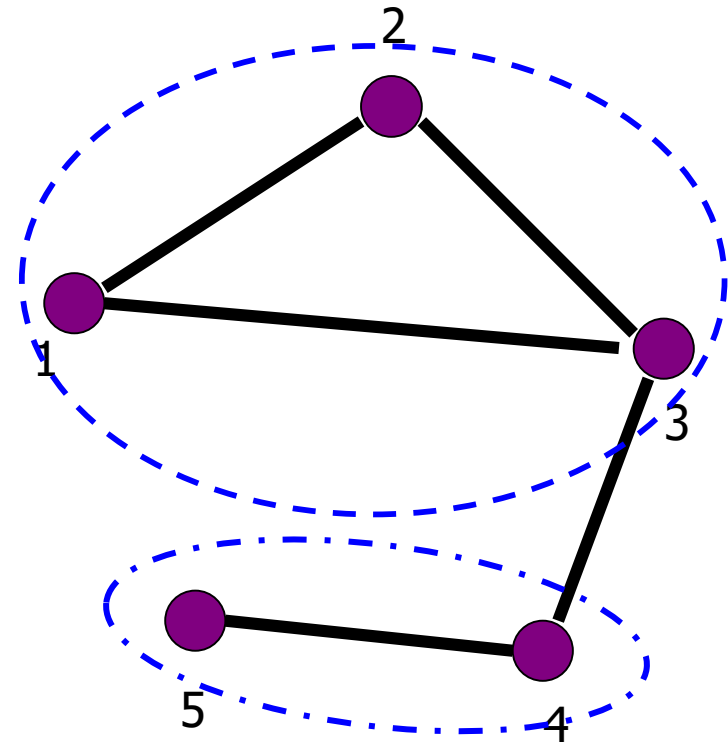
Diameter

- The **longest shortest path** in the graph



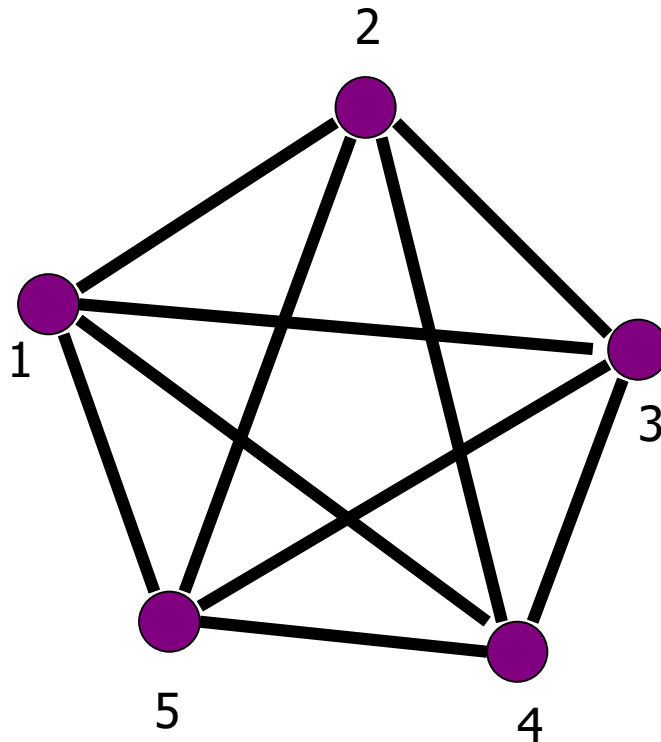
Undirected graph

- **Connected** graph: a graph where every pair of nodes is connected
- **Disconnected** graph: a graph that is not connected
- **Connected Components**: subsets of vertices that are connected



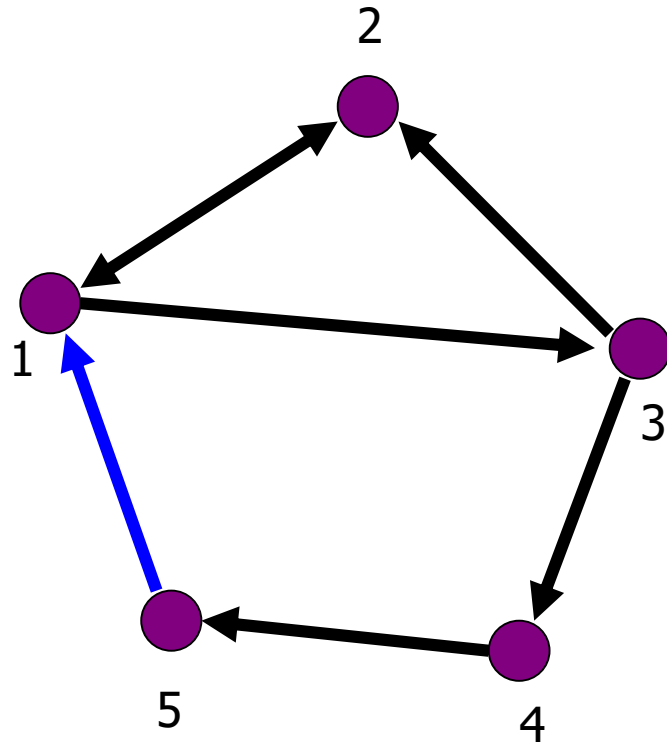
Fully Connected Graph

- **Clique** K_n
- A graph that has all possible $n(n-1)/2$ edges



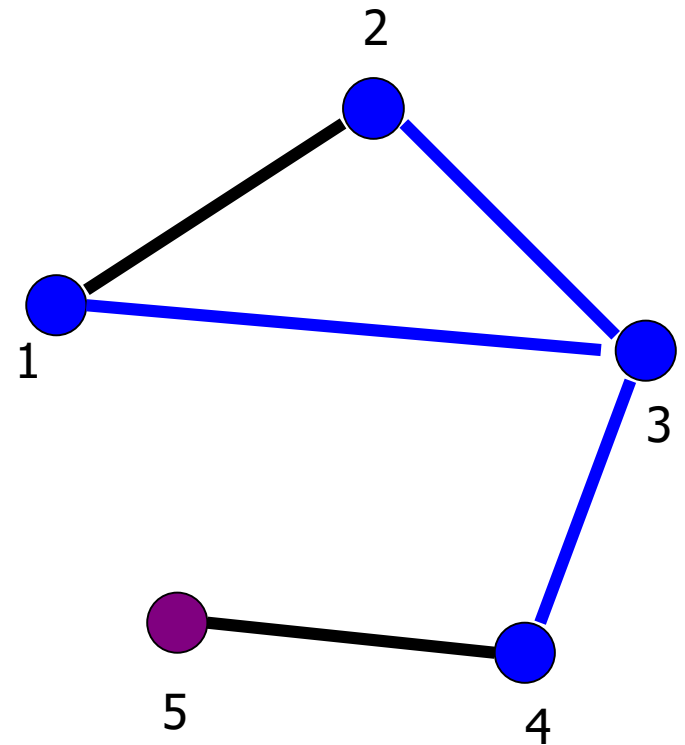
Directed Graph

- **Strongly connected graph:** there exists a path from every i to every j
- **Weakly connected graph:** If edges are made to be undirected the graph is connected



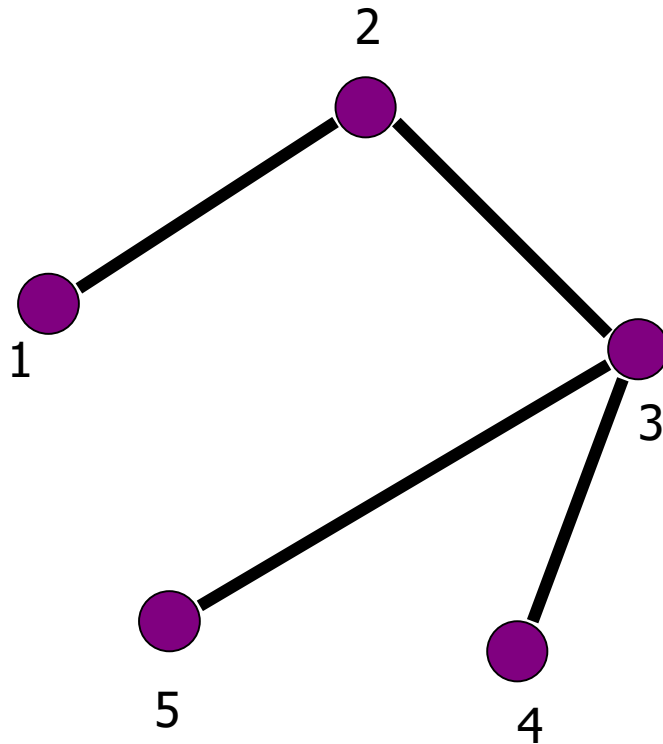
Subgraphs

- **Subgraph:** Given $V' \subseteq V$, and $E' \subseteq E$, the graph $G'=(V',E')$ is a subgraph of G .
- **Induced subgraph:** Given $V' \subseteq V$, let $E' \subseteq E$ is the set of all edges between the nodes in V' . The graph $G'=(V',E')$, is an induced subgraph of G



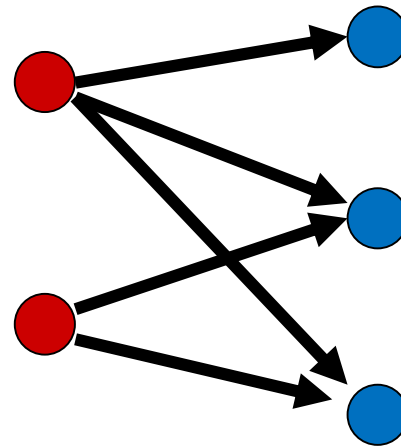
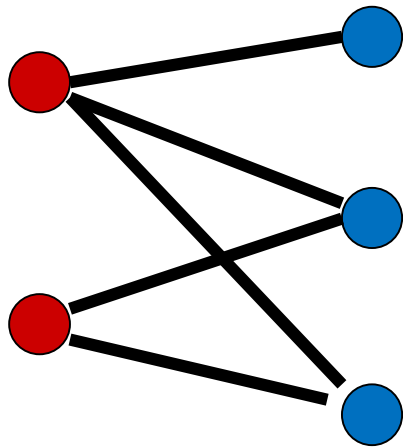
Trees

- Connected Undirected graphs without cycles



Bipartite graphs

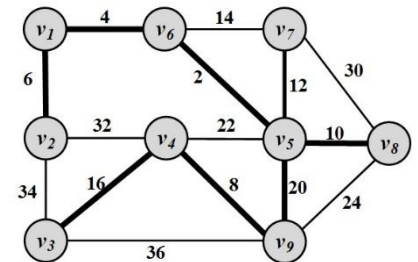
- Graphs where the set of nodes V can be partitioned into two sets L and R , such that there are edges only between nodes in L and R , and there is no edge within L or R



Spanning Tree

- For any connected graph, the **spanning tree** is a subgraph and a tree that includes all the nodes of the graph
- There may exist multiple spanning trees for a graph.
- For a weighted graph and one of its spanning tree, the weight of that spanning tree is the summation of the edge weights in the tree.
- Among the many spanning trees found for a weighted graph, the one with the minimum weight is called the

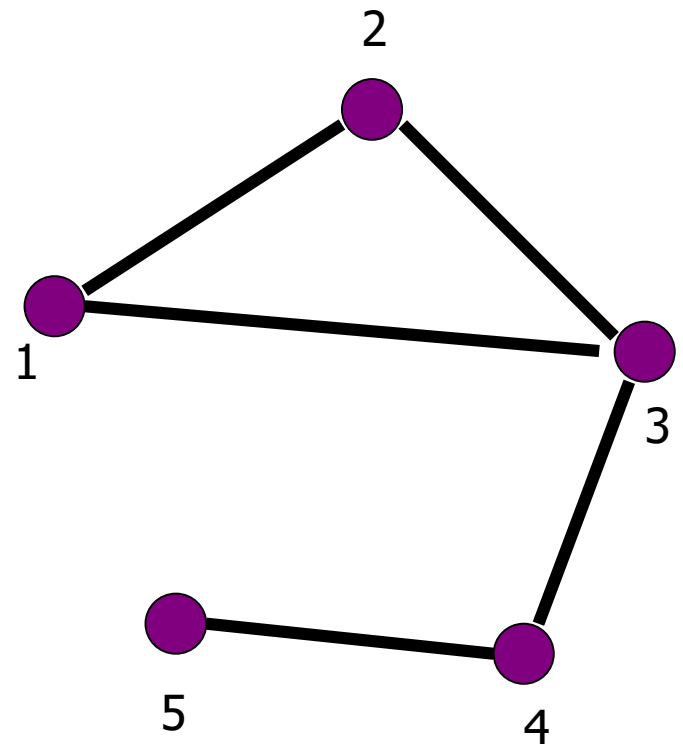
minimum spanning tree (MST)



Graph Representation

- Adjacency Matrix
 - **symmetric** matrix for undirected graphs

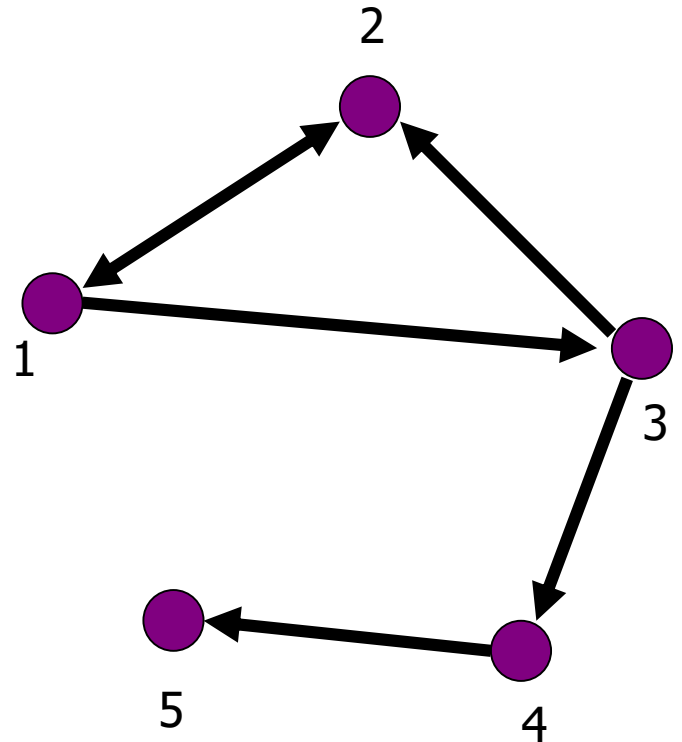
$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$



Graph Representation

- Adjacency Matrix
 - **unsymmetric** matrix for undirected graphs

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



Graph Representation

- Adjacency List
 - For each node keep a list with neighboring nodes

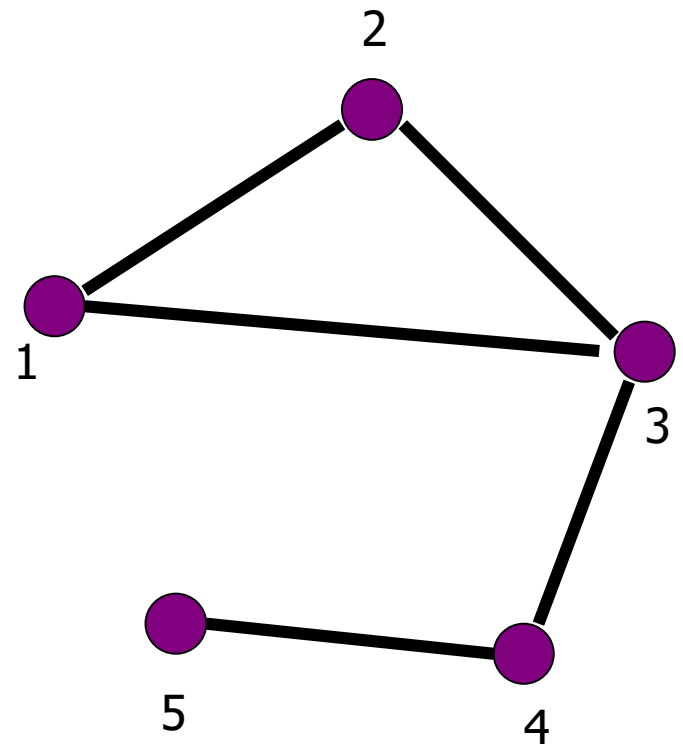
1: [2, 3]

2: [1, 3]

3: [1, 2, 4]

4: [3, 5]

5: [4]



Graph Representation

- Adjacency List
 - For each node keep a list of the nodes it points to

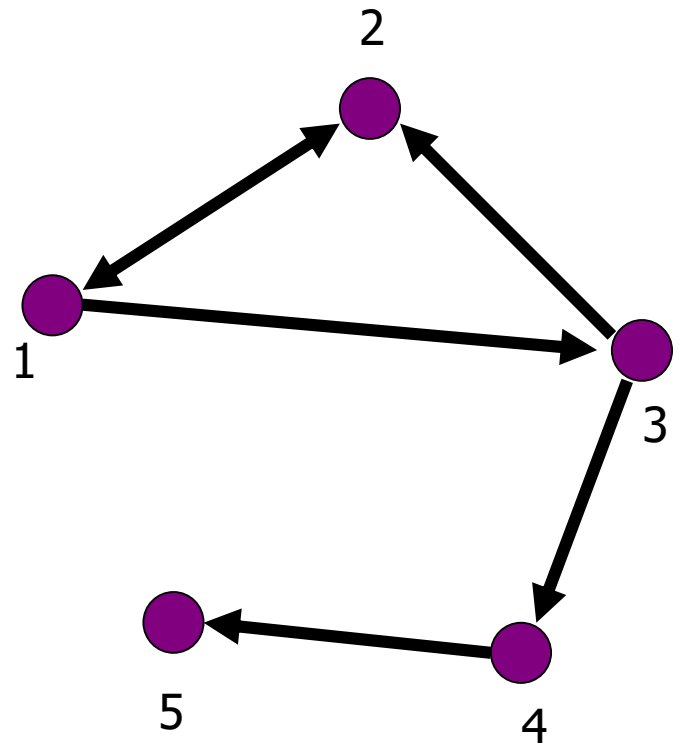
1: [2, 3]

2: [1]

3: [2, 4]

4: [5]

5: [null]



Graph Representation

- List of edges
 - Keep a list of all the edges in the graph

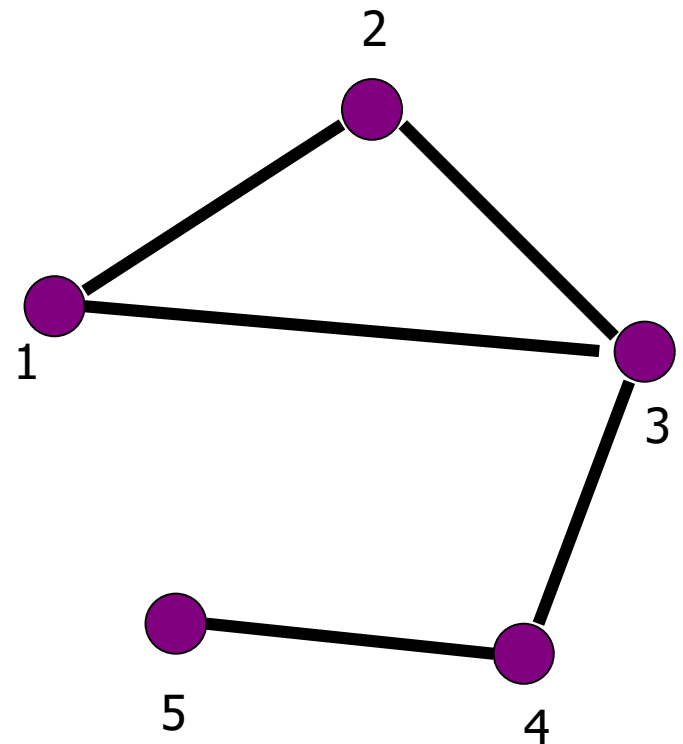
(1,2)

(2,3)

(1,3)

(3,4)

(4,5)



Graph Representation

- List of Edges
 - Keep a list of all the directed edges in the graph

(1,2)

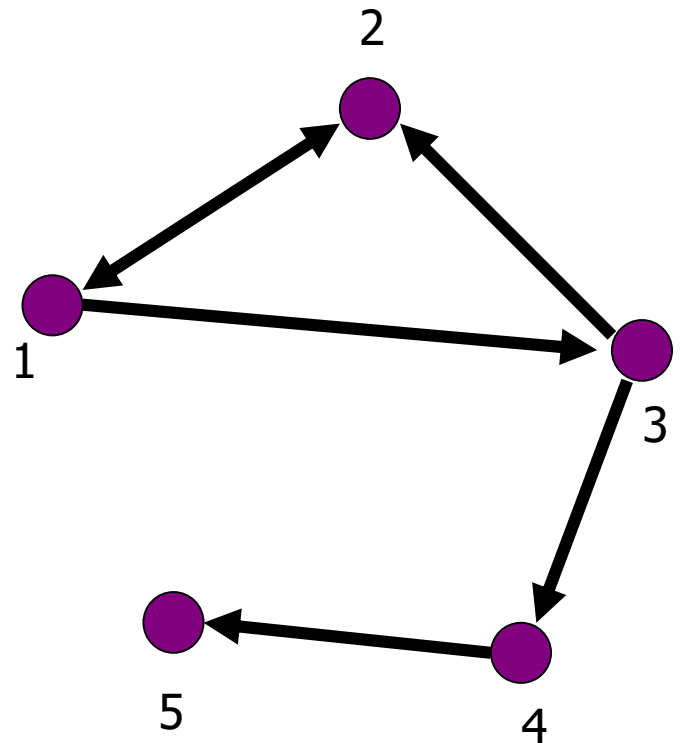
(2,1)

(1,3)

(3,2)

(3,4)

(4,5)



P and NP

- **P**: the class of problems that can be **solved** in polynomial time
- **NP**: the class of problems that can be **verified** in polynomial time, but there is **no known solution** in polynomial time
- **NP-hard**: problems that are at least as hard as any problem in **NP**