SELECTED TOPICS IN NETWORK SYSTEMS BIOLOGY

Dr. Alioune Ngom
School of Computer Science
University of Windsor
angom@uwindsor.ca

Winter 2013
Explosion in the availability of **biological data**:

The goal of **systems biology**:
- Systems-level understanding of biological systems, e.g. the cell
  - The “whole is more than the sum of its parts”
- Analyze not only individual components, but their interactions as well and its functioning as a whole
- E.g.: Learn new biology from the topology of such interaction networks

However, biological data analysis research faces considerable challenges
- Incomplete and noisy data
- Large amount of data (size of biological networks)
- Complexity of components
- Computational infeasibility of many graph theoretic problems

We will discuss only some selected topics in NSB
Some Topics to be Discussed

Biological networks aspects:
- Basic biological concepts (e.g., DNA, genes, proteins, gene expression, …)
- Different types of biological networks
- Experimental techniques for acquiring the data and their biases
- Public databases and other sources of biological network data

Graph and Network theoretic aspects:
- Fundamental topics in graph theory and complex network analysis
- Basic graph algorithms (review)

Existing approaches for analyzing and modeling biological networks:
- Structural properties of large or complex networks
- Network models
- Network clustering
- Network alignment
- Network reconstruction
- Software tools for network analysis

Applications: **Predictions Problems in Biology**, such as
- Drug-Target Interaction prediction
- Protein Complex prediction
- Protein Function prediction
- Regulatory Network identification or reconstruction
- Other network-based prediction problems
Some Biology

- **Cell** - the building block of life
  - Cytoplasm and organelles separated by membranes:
    - Mitochondria, nucleus, etc.
Some Biology

Distinguish between:

- **Prokaryotes**
  - Single-celled, no cell nucleus
    - The genetic material in prokaryotes is not membrane-bound
  - The *bacteria* and the *archaea*
  - Model organism: *E.coli*

- **Eukaryotes**
  - Have "true" nuclei containing their DNA
  - May be unicellular, as in amoebae
  - May be multicellular, as in plants and animals
  - Model organism: *S. cerevisiae* (baker’s yeast)
Some Biology

- Nucleus contains **DNA**
  - Deoxyribonucleic acid
- **DNA nucleotides:** A and T, C and G
- DNA structure: double helix

![Diagram of DNA structure](image)

Figure 1.1: The structure of a DNA double helix: (a) an illustration of complementary nucleotides pairs; (b) a bio-chemical view of DNA; (c) organization of DNA into chromosome.
Some Biology

- **Chromosomes**

- **RNA**: similar to DNA, except T → U and single stranded

Figure 1.1: The structure of a DNA double helix: (a) an illustration of complementary nucleotides pairs; (b) a bio-chemical view of DNA; (c) organization of DNA into chromosome.
Some Biology

- **Main role of DNA**: long-term storage of genetic information

- **Genes**: DNA segments that carry this information
  - **Intron**: part of gene not translated into protein, spliced out of mRNA
  - **Exon**: mRNA translated into protein consists only of exon-derived sequences

- **Genome**: total set of (unique) genes in an organism
  - Every cell (except sex cells and mature red blood cells) contains the complete genome of an organism
Some Biology

- **Codons**: sets of three nucleotides
  - 4 nucleotides $\rightarrow 4^3 = 64$ possible codons

- Each codon codes for an *amino acid*
  - 64 codons produce **20** different amino acids
  - More than one codon stands for one amino acid

- **Polypeptide**:
  - String of amino acids, composed from a 20-character alphabet

- **Proteins**:
  - String composed of one or more polypeptides (70-3000 amino acids)
  - Sequence of amino acids is defined by a gene
  - *Gene expression*: information transmission from DNA to proteins

- **Proteome**: total set of proteins in an organism
Some Biology

- The 20 amino acids

<table>
<thead>
<tr>
<th>Amino Acid</th>
<th>Abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alanine</td>
<td>Ala</td>
</tr>
<tr>
<td>Arginine</td>
<td>Arg</td>
</tr>
<tr>
<td>Aspartic Acid</td>
<td>Asp</td>
</tr>
<tr>
<td>Asparagine</td>
<td>Asn</td>
</tr>
<tr>
<td>Cysteine</td>
<td>Cys</td>
</tr>
<tr>
<td>Glutamic Acid</td>
<td>Glu</td>
</tr>
<tr>
<td>Glutamine</td>
<td>Gln</td>
</tr>
<tr>
<td>Glycine</td>
<td>Gly</td>
</tr>
<tr>
<td>Histidine</td>
<td>His</td>
</tr>
<tr>
<td>Isoleucine</td>
<td>Ile</td>
</tr>
<tr>
<td>Leucine</td>
<td>Leu</td>
</tr>
<tr>
<td>Lysine</td>
<td>Lys</td>
</tr>
<tr>
<td>Methionine</td>
<td>Met</td>
</tr>
<tr>
<td>Phenylalanine</td>
<td>Phe</td>
</tr>
<tr>
<td>Proline</td>
<td>Pro</td>
</tr>
<tr>
<td>Serine</td>
<td>Ser</td>
</tr>
<tr>
<td>Threonine</td>
<td>Thr</td>
</tr>
<tr>
<td>Tryptophan</td>
<td>Trp</td>
</tr>
<tr>
<td>Tyrosine</td>
<td>Tyr</td>
</tr>
<tr>
<td>Valine</td>
<td>Val</td>
</tr>
</tbody>
</table>
Some Biology

- Levels of protein structure:
  - Primary protein structure: sequence of a chain of amino acids.
  - Secondary protein structure: occurs when the sequence of amino acids are linked by hydrogen bonds.
  - Tertiary protein structure: occurs when certain attractions are present between alpha helices and pleated sheets.
  - Quaternary protein structure: a protein consisting of more than one amino acid chain.
Some Biology

- **Genes vs. proteins**
  - Genes – passive; proteins – active

- **Protein synthesis**: from genes to proteins
  - *Transcription* (in nucleus)
  - *Splicing* (eukaryotes)
  - *Translation* (in cytoplasm)
Some Biology

Transcription (in nucleus)

- RNA polymerase enzyme builds an RNA strand from a gene (DNA is "unzipped")
- The gene is transcribed to messenger RNA (mRNA)
- Transcription is regulated by proteins called transcription factors
Some Biology

- **Splicing** (eukaryotes)
  - Regions that are not coding for proteins (introns) are removed from sequence
Some Biology

- **Translation** (in cytoplasm)
  - Ribosomes synthesize proteins from mRNA
  - mRNA is decoded and used as a template to guide the synthesis of a chain of amino acids that form a protein
  - Translation: the process of converting the mRNA codon sequences into an amino acid polypeptide chain
Some Biology

- **Microarrays:**
  - Measure mRNA abundance for each gene
  - The amount of transcribed mRNA correlates with gene expression:
    - The rate at which a gene produces the corresponding protein

It is hard to measure protein level directly!
Some Biology

- Every cell* contains the complete genome of an organism
- How is the variety of different tissues encoded and expressed?
Some Biology

Comparison of genome size

<table>
<thead>
<tr>
<th>Organisms</th>
<th>Genomes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Haemophilus influenza</td>
<td>Genome (MB): 1.83</td>
</tr>
<tr>
<td>Methanococcus jannaschii</td>
<td>1.66</td>
</tr>
<tr>
<td>Saccharomyces cerevisae (baker's yeast)</td>
<td>13</td>
</tr>
<tr>
<td>Caenorhabditis elegans (nematode worm)</td>
<td>97</td>
</tr>
<tr>
<td>Drosophila Melanogaster (fruit fly)</td>
<td>180</td>
</tr>
<tr>
<td>Mus musculus (laboratory mouse)</td>
<td>3200</td>
</tr>
<tr>
<td>Homo sapiens (man)</td>
<td>3500</td>
</tr>
<tr>
<td>Number of genes</td>
<td>1709</td>
</tr>
<tr>
<td></td>
<td>1682</td>
</tr>
<tr>
<td></td>
<td>6241</td>
</tr>
<tr>
<td></td>
<td>18,424</td>
</tr>
<tr>
<td></td>
<td>13,500</td>
</tr>
<tr>
<td></td>
<td>~30,000</td>
</tr>
<tr>
<td></td>
<td>~30,000</td>
</tr>
</tbody>
</table>

22,000?
Some Biology

-ome and –omics

- Genome and genomics
- Transcriptome and transcriptomics
- Proteome and proteomics
- Metabolome and metabolomics
- Interactome and interactomics
- Microbiome and microbiomics
- Etc …
Introduction: biological networks

- The goal of *systems biology*:
  - Systems-level understanding of biological systems
  - Analyze not only individual components, but their interactions as well and emergent behavior
  - In the rest of the course: Learn new biology from the topology of such interaction networks
    - Reconstruct a network from data sets
    - Mine a network
    - … etc
What is a *network* (or *graph*)?

- A set of nodes (vertices) and edges (links)
- Edges describe a relationship between the nodes
Introduction: biological networks

Networks model many real-world phenomena
Introduction: biological networks

- E.g., Facebook
Introduction: biological networks

- E.g., WWW
Introduction: biological networks

- E.g., Internet
Introduction: biological networks

- E.g., Airline routes
Introduction: biological networks

- **Biological nets**
  - E.g., Protein structure networks
Introduction: biological networks

- **Biological nets**
  E.g., Protein-protein interaction (PPI) networks
Introduction: biological networks

- Biological nets
  E.g., Metabolic networks

Metabolic network of *A. thaliana*
Introduction: biological networks

- Biological nets
  Other network types

- $X \rightarrow Y$ represents

- transcription network
  gene $x$  gene $y$

- neuron synaptic connection network

- ecological food web
  X $\rightarrow$ Eat $\rightarrow$ Y
Introduction: biological networks

- Types of biological networks:
  - Intra-cellular networks
    - Metabolic networks
    - Transcriptional regulation networks
    - Cell signaling networks
    - Protein-protein interaction (PPI) networks
    - Protein structure networks
  - Other biological networks
    - Neuronal synaptic connection networks
    - Brain functional networks
    - Ecological food webs
    - Phylogenetic networks
    - Correlation networks (e.g., gene expression)
    - Disease – “disease gene” association networks
    - Drug – “drug target” networks
Introduction: biological networks

- Intra-cellular networks
  - Metabolic networks
  - Transcriptional regulation networks
  - Cell signaling networks
  - Protein-protein interaction (PPI) networks
  - Protein structure networks

- All of these networks describe cellular functioning at different levels and often “overlap”
  - Cell relies on numerous highly interconnected interactions and chemical reactions between various types of molecules, e.g., proteins, DNA, RNA, metabolites, etc.
  - Various activities of cells are controlled by the action of molecules upon molecules
  - Proteins – central players
  - Main application of methods in this course: PPI networks
Metabolic networks

- Used for studying and modeling *metabolism*
  - Biochemical reactions in cells that allow an organism to:
    - Respond to the environment
    - Grow
    - Reproduce
    - Maintain its structure
    - ...
  - i.e., the main biochemical reactions needed to keep an organism in *homeostasis*
    - An internal regulation that maintains a stable, constant condition of a living system
Metabolic networks

- **Metabolites**
  - Small molecules such as glucose and amino acids
  - Also, macromolecules such as polysaccharides and glycan

- **Metabolic pathways**
  - Series of successive biochemical reactions for a specific metabolic function, e.g., glycolysis or penicillin synthesis, that convert one metabolite into another
  - **Enzymes**: proteins that catalyze (accelerate) chem. reactions

- Thus, in a metabolic pathway:
  - Nodes correspond to metabolites and enzymes
    - In an alternate order → **bipartite graphs**
  - Directed edges correspond to metabolic reactions
  - Simpler approaches: nodes are metabolites, directed edges are reactions that convert one metabolite into the other
Metabolic networks

- Example: part of glycolysis pathway

Glycolysis

Reactions + metabolites:
All metabolic pathways of a cell form a *metabolic network*

- Complete view of the cellular metabolism and material/mass flow through the cell
- Cell relies on this network to digest substrates from the environment, generate energy, and synthesize components needed for its growth and survival
- Used to, for example:
  - Cure human metabolic diseases through better understanding of the metabolic mechanism
  - Control infections of pathogens by understanding the metabolic differences between human and pathogens
Metabolic networks

- Constructed:
  - Partially experimentally
  - Partially from genome sequence (homology)
- Available for many organisms, from bacteria to human
- Available on-line:
  - KEGG (Kyoto Encyclopedia of Genes and Genomes)
    - Info on genes, proteins, reactions, pathways
    - Both for eukaryotes and prokaryotes
  - GeneDB—contains similar info
  - BioCyc, EcoCyc, MetaCyc
    - More specialized info on particular species
  - WIT, renamed to ERGO
KEGG: Kyoto Encyclopedia of Genes and Genomes

A grand challenge in the post-genomic era is a complete computer representation of the cell, the organism, the ecosystem, and the biosphere, which will enable computational prediction of higher-level complexity of cellular processes and organism behaviors from genomic and molecular information. Towards this end we have been developing a bioinformatics resource named KEGG as part of the research projects of the Kanehisa Laboratories in the Bioinformatics Center of Kyoto University and the Human Genome Center of the University of Tokyo.

Main entry point to the KEGG web service

- KEGG2
- KEGG Table of Contents
- Update notes
- Help

Data-oriented entry points

- Pathway maps and pathway modules
- Functional hierarchies and ontologies
- Human diseases
- Disease classification
- Drugs
- ATC drug classification
- KO system and ortholog annotation
- KO system
- Genes and proteins
- Genomes
- KEGG organisms
- Chemical compounds
- Compound classification
- Glycans
- Reactions
Further readings

Transcriptional regulation networks

- Model regulation of *gene expression*
  - Recall: gene $\rightarrow$ mRNA $\rightarrow$ protein

- Gene regulation
  - Gives a cell control over its structure and function, e.g.:
    - *Cellular differentiation* – a process by which a cell turns into a more specialized cell type
    - *Morphogenesis* (a process by which an organism develops its shape)
    - ...


Transcriptional regulation networks

- Nodes correspond to genes
  - DNA sequences which are transcribed into mRNAs that translate into proteins
- Directed edges correspond to interactions through which the products of one gene affect those of another
  - Protein-protein, protein-DNA and protein-mRNA interactions

- *Transcription factor* $X$ (protein product of gene $X$) binds regulatory DNA regions of gene $Y$ to regulate the production rate (i.e., stimulate or repress transcription) of protein $Y$
  - Note: proteins are products of gene expression that play a key role in regulation of gene expression
Problem

- Stimulation and repression of gene transcription are both represented the same way in the network

Available for *model organisms*

- Non-human species manipulated and studied to get insights into workings of other organisms
  - Baker's yeast, *S. cerevisiae* (Milo et al., 2002)
  - *E. coli* (Shen-Orr et al., 2002)
  - Sea urchin (Davidson et al., 2002)
  - Fruitfly, *D. melanogaster*

Available from: EcoCyc, GeneNet, KEGG, RegulonDB, Reactom, TRANSPATH, TRANSFAC
E. coli

(a)

(b) T F

(c) R G

Representation of the E. coli transcriptional regulatory network. a) Representation of the transcription-factor gene regulatory network of E. coli. Green circles represent transcription factors, brown circles denote regulated genes, and those with both functions are coloured in red. Projections of the network onto b) transcription factor and onto c) regulated gene nodes are also shown. Guzmán-Vargas and Santillán BMC Systems Biology 2008 2:13 doi:10.1186/1752-0509-2-13
Transcriptional regulation networks

- Further readings:
  - List of databases:
    - University of Pittsburg, Health Science Library
    - Online Bioinformatics Resources Collection
    - [http://www.hsls.pitt.edu/obrc/](http://www.hsls.pitt.edu/obrc/)
Cell signaling networks

- **Cell signaling**
  - Complex communication system that governs basic cellular activities, e.g., development, repair, immunity
  - Errors in signaling cause diseases
    - E.g., cancer, autoimmune diseases, diabetes…
Cell signaling networks

- **Signaling pathways**
  - Ordered sequences of signal transduction reactions in a cell, as shown in the previous figure
  - Cascade of reversible chemical modifications of proteins
    - E.g., phosphorylation catalyzed by protein kinases
- Signaling pathways in the cell form the **cell signaling network**
  - Nodes are proteins and edges are directed
Famous examples (lots of literature on them):

- **Mitogen-activated protein kinase (MAPK) pathway**
  - Originally called “ERK” pathway
  - MAPK protein: an enzyme, a *protein kinase*, which can attach *phosphate groups* to a target protein, causing its spatial reorganization and affecting its function
    - Other enzymes can restore protein’s initial function
  - E.g.:
    - MYC
      - An *oncogene* transcription factor expressed in a wide range of human cancers (oncogene – when mutated or over-expressed, the gene helps turn a normal into a tumor cell)
      - MAPK can *phosphorylate* (attach phosphate group to) MYC and alter gene transcription and cell cycle progression
    - EGFR = “epidermal growth factor receptor”
      - Activates MAPK pathway
      - Mutations affecting its expression/activity can result in cancer
Cell signaling networks

Famous examples (lots of literature on them) cont’d:

- **Hedgehog signaling pathway**
  - One of the key regulators of animal development
  - Conserved from fly to human
  - Establishes basis of fly body plan
  - Important during *embryogenesis* (the process by which the embryo develops) and *metamorphosis* (from larva to pupa to adult)

- **TGF-beta signaling pathway**
  - The “transforming growth factor” (TGF) signaling pathway
  - Involved in:
    - Cell growth
    - Cell differentiation
    - *Apoptosis* (programmed cell death)
Cell signaling networks

- Compared to metabolic networks:
  - Limited mass flow
  - Instead, sig. nets provide information transmission along a sequence of reactions – one enzyme modulates the activity of another one, which then modulates the activity of the third enzyme, etc., but enzymes are not consumed in the reactions they catalyze

- Compared to transcriptional reg. networks:
  - They overlap, but gene expression, i.e., transcription factors, can be seen as the “final targets” of signaling pathways

- Compared to PPI networks:
  - Signal transduction is indeed mediated between proteins, but PPIs are undirected without a defined input and output (as we will discuss soon)
  - Not all PPIs are involved in chemical reactions or part of signal transduction
  - Also, many components of signaling are not proteins

- These nets have much in common
- At the same time, they reflect different aspects of cellular activity
Protein-protein interaction (PPI) networks
Protein-protein interaction (PPI) networks

- A protein-protein interaction (PPI) usually refers to a physical interaction, i.e., binding between proteins.
- Can be other associations of proteins such as functional interactions – e.g., synthetic lethality.
PPIs are very important for structure and function of a cell:

- Participate in signal transduction
  - Play a role in many diseases (e.g., cancer)
- Can be stable interactions forming a protein complex
  (a form of a quaternary protein structure, set of proteins which bind to do a particular function, e.g., ribosome, hemoglobin – illustrated below)
Protein-protein interaction (PPI) networks

- PPIs are very important for structure and function of a cell:
  - Can be *transient interactions*
    - Brief interactions that modify a protein that can further change PPIs e.g., protein kinases (add a phosphate group to a target protein)
    - A protein can carry another protein, e.g., *nuclear pore importins* (proteins that carry other proteins from cytoplasm to nucleus and vice versa)
    - Transient interaction form the *dynamic part of PPI networks*
  - Some estimates state that about 70% of interactions are stable and 30% are dynamic
- PPI are essential to almost every process in a cell
- Thus, understanding PPIs is crucial for understanding life, disease, development of new drugs (most drugs affect PPIs)
Protein-protein interaction (PPI) networks

Methods to detect PPIs

- Biological and computational approaches
- None are perfect
  - High rates of false positives
    - Interactions present in the data sets that are not present in reality
  - High rates of false negatives
    - Missing true interactions
Protein-protein interaction (PPI) networks

Methods to detect PPIs

- PPIs initially studied individually by small-scale biochemical techniques (SS)
- However, large-scale (high-throughput) interaction detection methods (HT) are needed for high discovery rates of new protein interactions
- SS of better “quality,” i.e., less noisy than HT
- However, HT are more standardized, while SS are performed differently each time
- SS are biased – the focus is on the subsets of proteins interesting to particular researchers
- HT – view of the entire proteome
Protein-protein interaction (PPI) networks

Methods to detect PPIs

- Physical binding
  - Yeast 2-hybrid (Y2H) screening
  - Mass spectrometry of purified complexes
- Functional associations
  - Correlated mRNA expression profiles
  - Genetic interactions
  - In silico (computational) methods
- In many cases, functional associations do take the form of physical binding
Protein-protein interaction (PPI) networks

Yeast two-hybrid assay

- **Binary PPIs**

- Pairs of proteins to be tested for interaction are expressed as *artificial* (genetically engineered) *fusion proteins* in yeast:
  - One protein is fused to a reporter gene (a gene attached to another gene of interest)
  - The other is fused to a transcription factor
  - Any interaction between them is detected by the transcriptional activation of the reporter gene
Protein-protein interaction (PPI) networks

Yeast two-hybrid assay

- One protein (in PPI) is “bait”, the other is “prey”
- Potential problem:
  - Interest in a particular pathway of, say 15 proteins
  - These 15 proteins are all “baits”
  - There is an order of magnitude more “preys”
  - This imposes a particular structure on the PPI network by experimental design without reflecting the underlying network topology
- To avoid this, a matrix of $n \times n$ needs to be probed, where each bait is also a prey (Mark Vidal’s lab, Harvard)
Yeast two-hybrid assay

- This method is scalable to entire proteome
- Directly tests a protein pair for an interaction
- But high noise rate (50%, even up to 70%)
  - Because Y2H investigates interactions between:
    - artificial, fusion proteins
    - in the yeast
    - in the yeast’s nucleus
  - Each of these steps is noisy
- Proteins need to be in their native environment, not in nucleus
  - E.g., although proteins can physically bind, they never do so inside cells, because of different localization, or because they are never simultaneously expressed
Protein-protein interaction (PPI) networks

Mass spectrometry of purified complexes
- Individual proteins are tagged and used as hooks to biochemically purify whole protein complexes
- Complexes separated and components identified by mass spectrometry (MS)
  - MS measures mass-to-charge ratio of ions
- TAP (Tandem Affinity Purification)
- HMS-PCI (High-Throughput MS Protein Complex Identification)
- Not binary but co-complex data
Protein-protein interaction (PPI) networks

Mass spectrometry of purified complexes

- We know what proteins are in the complexes, but not how they are connected
  - Spoke model
  - Matrix model
Mass spectrometry of purified complexes

- **Pros:**
  - Detects real complexes in their physiological settings
  - Consistency check is possible by tagging several members of a complex
  - Good for screening permanent/stable interactions

- **Cons:**
  - Might miss some complexes that are not present under given cellular conditions
  - Tagging may disturb complex formation
  - Loosely associated components can be washed off during purification
Protein-protein interaction (PPI) networks

Functional associations
- Correlated mRNA expression profiles
  - Results in a gene expression correlation network
Functional associations

- Genetic interactions
  - Two non-essential genes that cause lethality when mutated at the same time form a synthetic lethal interaction
  - Such genes are often functionally associated and their encoded proteins may also interact physically
  - Charles Boone’s group from University of Toronto published genetic interaction networks
Protein-protein interaction (PPI) networks

Functional associations
- Genetic interactions
Protein-protein interaction (PPI) networks

Functional associations

- In silico (computational) methods
  - Gene fusion (if two genes are present in one species and fused in another)
  - ...


Protein-protein interaction (PPI) networks

Biases within PPI networks

- The following is lost:
  - Spatial information
  - Temporal information
  - Information about experimental conditions
  - Strength of interactions
  - Number of experiments confirming interactions

- PPI network: proteome + interactome
  - Proteome: a set of all unique proteins in an organism;
  - How does protein concentration affect the topology:
    - More instances of a protein in the cell → more interacting partners in the network?
Protein-protein interaction (PPI) networks

Quality and completeness of PPI data

- Data sets produced by different methods are often complementary.
- Even data sets obtained by the same technique complement each other to some (large) extent.
- Completeness of data sets:
  - **Yeast**: ~50% (~6K proteins, ~30K-60K interactions)
  - **Human**: ~10% (~25K proteins, ~260K interactions; ~300 million pairs to test)
  - **Fly**
  - **Worm**
  - Recently, herpes viruses (genome-wide coverage)
Protein-protein interaction (PPI) networks

PPI databases*

- Biological General Repository for Interaction Datasets (BioGRID)
- Human Protein Reference Database (HPRD)
- Saccharomyces Genome Database (SGD)
- Munich Information Center for Protein Sequences (MIPS)
- Database of Interacting Proteins (DIP)
- Molecular Interactions Database (MINT)
- Online Predicted Human Interaction Database (OPHID)
- VirusMINT

- The lack of standardization
  - Different databases use different naming conventions
  - Inconsistencies in mapping between them
  - This can seriously jeopardize network topological analyses

*Distinguish between binary and co-complex data.
Protein-protein interaction (PPI) networks

- Additional readings:
  - Chapter 4 of “Knowledge Discovery in Proteomics” by Wiggle and Jurisica
Protein structure networks

“Residue interaction graphs” (RIGs) model protein structures

- Nodes are amino acid residues
- Undirected, unweighted edges exist between amino acids that are in close proximity in the protein’s 3-dimensional structure
  - E.g., within 5 Angstroms (1 Å = 10⁻¹⁰ meters)

Different network types: summary

Proteins
Metabolites
Metabolism
Gene regulation
Cell signaling
PPIs
Other biological networks

- Neuronal synaptic connection networks

- Brain functional networks
  - Simultaneous (correlated) activities of brain regions during a task

- Ecological food webs

- Phylogenetic networks (trees)
  - Evolutionary relationships between species
Other biological networks

- Correlation networks (e.g., gene expression)
  - Different from transcriptional regulation networks
  - Not a direct result of experiments
  - Determined by:
    - Collecting large amounts of high-throughput data
    - Calculating the correlations between all elements
  - Biolayout Express 3-D: a tool for generating correlation networks
Other biological networks

☐ Disease – “disease gene” association networks
  - Link diseases that are caused by a same gene
  - Link genes if they cause a same disease

☐ Drug – “drug target” association networks
  - Link drugs if they target a same gene
  - Link genes if they are targeted by a same drug
Other biological networks

Further readings

- Neuronal synaptic connection networks

- Brain functional networks

- Ecological food webs, phylogenetic networks, correlation networks

- Disease-disease gene association networks

- Drug-drug target networks
Other real-world networks

- Technological networks:
  - WWW
  - Internet
  - Electric circuits
  - Software call graphs

- Transportation networks:
  - Roads, airlines, railways

- Social networks:
  - Friendships/relationships (Facebook, MySpace)
  - Collaborations between scientists/movie stars
  - Spread of infections and diseases
  - Economic networks
  - Relationships between organizations (companies, NGOs, etc.)
  - City/country trading relationships
  - Migrations
  - Disaster response networks
Introduction to graph theory

- Basic definitions and graph types
- Graph representations
- Running times of algorithms
- Complexity classes
- Graph traversing and shortest path problems
Introduction to graph theory

- Basic definitions and graph types
- Graph representations
- Running times of algorithms
- Complexity classes
- Graph traversing and shortest path problems
Introduction to graph theory

Good textbook (not mandatory):

Chapter 3
Graphs
Introduction to graph theory

- Begun in 1735
- Bridges of Konigsberg (today’s Kaliningrad): walk all 7 bridges without crossing a bridge twice

Solved by: parity of nodes
Introduction to graph theory

- **Graph** – mathematical object consisting of a set of:
  - \( V = \text{nodes} \) (vertices, points).
  - \( E = \text{edges} \) (links, arcs) between pairs of nodes.
  - Denoted by \( G = (V, E) \).
  - Captures pairwise relationship between objects.
  - **Graph size** parameters: \( n = |V|, m = |E| \).
Introduction to graph theory

**Graph** – mathematical object consisting of a set of:

- \( V = \text{nodes} \) (vertices, points).
- \( E = \text{edges} \) (links, arcs) between pairs of nodes.
- Denoted by \( G = (V, E) \).
- Captures pairwise relationship between objects.
- **Graph size** parameters: \( n = |V|, m = |E| \).

\[
V = \{1, 2, 3, 4, 5, 6, 7, 8\} \\
E = \{\{1,2\}, \{1,3\}, \{2,3\}, \{2,4\}, \{2,5\}, \{3,5\}, \{3,7\}, \{3,8\}, \{4,5\}, \{5,6\}\} \\
n = 8 \\
m = 11
\]
Introduction to graph theory

For graph $G(V,E)$:
- $V$ is also denoted by $V(G)$ and $V_G$
- $E$ is also denoted by $E(G)$ and $E_G$; $E \subseteq V \times V$
- Know set operators:
  - $A = B$ – equal sets
  - $A \cup B$ – union of sets $A$ and $B$
  - $A \cap B$ – intersection of sets $A$ and $B$
  - $a \in A$ – $a$ is an element of set $A$

### Example

- $V = \{1, 2, 3, 4, 5, 6, 7, 8\}$
- $E = \{\{1,2\}, \{1,3\}, \{2,3\}, \{2,4\}, \{2,5\}, \{3,5\}, \{3,7\}, \{3,8\}, \{4,5\}, \{5,6\}\}$
- $n = 8$
- $m = 11$
Introduction to graph theory

For graph \( G(V,E) \):

- If edge \( e=\{u,v\} \in E(G) \), we say that \( u \) and \( v \) are adjacent or neighbors.
- \( u \) and \( v \) are incident with \( e \).
- \( u \) and \( v \) are end-vertices of \( e \).
- An edge where the two end vertices are the same is called a loop, or a self-loop.

\[ V = \{ 1, 2, 3, 4, 5, 6, 7, 8 \} \]
\[ E = \{ \{1,2\}, \{1,3\}, \{2,3\}, \{2,4\}, \{2,5\}, \{3,5\}, \{3,7\}, \{3,8\}, \{4,5\}, \{5,6\} \} \]
\[ n = 8 \]
\[ m = 11 \]
Introduction to graph theory

- **Edge types:**
  - **Undirected:** E.g., distance between two cities, PPIs, friendships…
  - **Directed:** ordered pairs of nodes.
    - E.g. metabolic reactions, transcriptional regulation,…
  - **Loops:** usually we assume no loops.

- **Graph types:**
  - Undirected
  - Directed
  - Mixed (some edges directed some undirected)
  - Weighted
    - (weights on edges or nodes)
Introduction to graph theory

- Undirected edges have **end-vertices**

- Directed edges have a **source** (head, origin) and **target** (tail, destination) vertices

- **Multigraphs** – contain multiple edges

- **Simple graphs**
  - Undirected
  - No loops or multiple edges

- **Hypergraphs**: $E \subseteq 2^V$ (all subsets of elements of $V$); i.e., each edge (hyperedge) is a subset of vertices.

  E.g.

  $V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$

  $E = \{e_1, e_2, e_3, e_4\} = \{\{v_1, v_2, v_3\}, \{v_2, v_3\}, \{v_3, v_5, v_6\}, \{v_4\}\}$
Hypergraphs: $E \subseteq 2^V$ (all subsets of elements of $V$); i.e., each edge (hyperedge) is a subset of vertices. 

E.g.

$V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$
$E = \{e_1, e_2, e_3, e_4\} = \{\{v_1, v_2, v_3\}, \{v_2, v_3\}, \{v_3, v_5, v_6\}, \{v_4\}\}$

Not used much in network biology, but could be

E.g., metabolic networks where several substances react with each other to build other substances
An undirected graph $G = (V, E)$ is **bipartite** if the nodes can be colored red or blue such that every edge has one red and one blue end.

**Applications:**
- Stable marriage: men = red, women = blue.
- Scheduling: machines = red, jobs = blue.
- Metabolic networks: metabolites = blue, enzymes = red.

![a bipartite graph](image)
Def. A cycle is a path \( v_1, v_2, \ldots, v_{k-1}, v_k \) in which \( v_1 = v_k \), \( k > 2 \), and the first \( k-1 \) nodes are all distinct.

Simple cycle:
- all vertices and edges are distinct
- each edge is preceded and followed by its end-vertices
- E.g.: 1-2-3-7-8-3-1 in figure above is not a simple cycle, \( C \) above is

Cycles denoted by \( C_k \), where \( k \) is the number of nodes in the cycle
Introduction to graph theory

- Def.
  - An undirected graph is a **tree** if it is connected and does not contain a cycle.
  - **Forest** – does not contain a cycle (so it’s a union of trees)

- Theorem. (illustration only) Let G be an undirected graph on n nodes. Any two of the following statements imply the third.
  - G is connected.
  - G does not contain a cycle.
  - G has n-1 edges.
Introduction to graph theory

- **Rooted tree.** Given a tree $T$, choose a root node $r$ and orient each edge away from $r$.

- **Importance:** Models hierarchical structure.
Introduction to graph theory

- E.g.: **Phylogeny trees**. Describe evolutionary history of species.
Introduction to graph theory

- **Minimum spanning tree.** Given a connected graph $G = (V, E)$ with real-valued edge weights (costs) $c_e$, an **MST** is a subset of the edges $T \subseteq E$ such that $T$ is a **spanning tree** (contains all nodes of $G$) whose sum of edge weights is minimized.

- **Cayley's Theorem.** There are $n^{n-2}$ spanning trees of $K_n$.

G = (V, E)

$T$, $\sum_{e \in T} c_e = 50$

can't solve by brute force
Introduction to graph theory

- MST is fundamental problem with diverse applications.

  - Network design:
    - telephone, electrical, hydraulic, TV cable, computer, road

  - Approximation algorithms for NP-hard problems:
    - traveling salesperson problem, Steiner tree

  - Indirect applications:
    - bottleneck paths
    - identifying functional modules in weighted gene co-expression networks
    - predicting subgraph structure of protein complexess from affinity purification studies
    - clustering
    - ...

Introduction to graph theory

Greedy Algorithms used to find an MST:

- **Kruskal's algorithm.** Start with $T = \emptyset$. Consider edges in ascending order of cost. Insert edge $e$ in $T$ unless doing so would create a cycle.

- **Prim's algorithm.** Start with some root node $s$ and greedily grow a tree $T$ from $s$ outward. At each step, add the cheapest edge $e$ to $T$ that has exactly one endpoint in $T$.

- **Remark.** Algorithms produce an MST.
Def. A **complete graph (clique)** is a graph on $n$ nodes with all possible edges between the nodes.

Denoted by $K_n$, where $n$ is the number of nodes in the clique.
Introduction to graph theory

- **Def.** A **path** in an undirected graph \( G = (V, E) \) is a sequence \( P \) of nodes \( v_1, v_2, \ldots, v_{k-1}, v_k \) with the property that each consecutive pair \( v_i, v_{i+1} \) is joined by an edge in \( E \).

- So, nodes can repeat, but edges do not.

- **Def.** A **walk**: a path in which edges/nodes can be repeated.

- **Def.** A path is **simple** if all nodes are distinct.

- A simple path with \( n \) nodes is denoted by \( P_n \).

---

Walk: \{2,3\}, \{3,1\}, \{1,2\}, \{2,3\}, \{3,4\}

Path: \{3,1\}, \{1,2\}, \{2,3\}, \{3,4\}

Simple Path: \{1,2\}, \{2,3\}, \{3,4\}

Cycle
Introduction to graph theory

- **Def:** Two vertices are *connected* if there is a path between them.

- **Def.** An undirected graph is *connected* if for every pair of nodes $u$ and $v$, there is a path between $u$ and $v$.

- **Def.** Maximal connected subgraph of $G$: subgraph that is connected and is not contained in any other subgraph of $G$.

- **Def.** A connected component of $G$ is a maximal connected subgraph of $G$.

- **Def.** A subgraph $G'(V', E')$ of $G(V, E)$: $V' \subseteq V$, $E' \subseteq E$ (details later).

- **E.g.**

![Graph G with 3 connected components](image)
A shortest path between two vertices – a path of minimal length.

Length – number of edges.

Distance between u and v – the length of a shortest path between them (or $\infty$ if a path does not exist)

Subgraphs: $G'(V', E')$ is subgraph of $G(V, E)$: $V' \subseteq V$, $E' \subseteq E$

- Induced subgraph: $E'$ contains all edges from $E$ that exist between nodes in $V'$, i.e., that have both end-points in $V'$
- Partial subgraph: contains some of the edges in $E$ that have both end-points in $V'$.

Partial subgraph: 3-node path

Induced subgraph: 3-node cycle: $C_4=K_4$
Introduction to graph theory

- **Degree** of a vertex: the number of edges incident to the vertex (in undirected graphs)
- **In-degree** and **out-degree** in directed graphs: the number of edges coming into / going out of the vertex.
- E.g.

  ![Graph example](image)

  - **Property**: $\sum_{v \in V} \deg(v) = 2m$, where $|V|=n$, $|E|=m$.
    Proof: each end-vertex is counted twice.

  - **Property**: in a simple graph, $m \leq n(n-1)/2$.
    Proof: each vertex has degree at most $(n-1)$. 

  ![Graph example](image)
Introduction to graph theory

- **Graph isomorphism**

  E.g.: Petersen graph, G:

- Are there other ways of drawing G?
- Is graph H below the same as G above?
Introduction to graph theory

- Graph isomorphism

Example:

\[ G \text{ has no } C_4 \]

\[ \Rightarrow G \neq H \]

\[ \text{E.g. } \]

\[ G \]

\[ \text{redraw: } \]

\[ \Rightarrow G = H \]
Introduction to graph theory

- **Graph isomorphism**

Definitions:

An *isomorphism* $f$ from graph $G$ to graph $H$ is a bijection: $f : V(G) \rightarrow V(H)$ such that $xy$ is an edge of $G$ iff $f(x)f(y)$ is an edge of $H$.

An *automorphism* is an isomorphism from a graph to itself.

The automorphisms of a graph $G$ form a group, called the *automorphism group of* $G$, and commonly denoted by Aut($G$).

For a node $x$ of graph $G$, the *automorphism orbit of* $x$ is $\text{Orb}(x) = \{ y \in V(G) | y = f(x) \text{ for some } f \in \text{Aut}(G) \}$, where $V(G)$ is the set of nodes of graph $G$. 
Introduction to graph theory

- **Group**: an algebraic structure consisting of a set together with an operation that combines any two of its elements to form a third element.
- Operation must satisfy group axioms: closure, associativity, identity and invertibility.
- E.g. of a group: Integers with addition operation: \((\mathbb{Z},+)\).

<table>
<thead>
<tr>
<th></th>
<th>Addition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Closure:</strong></td>
<td>(a + b) is an integer</td>
</tr>
<tr>
<td><strong>Associativity:</strong></td>
<td>(a + (b + c) = (a + b) + c)</td>
</tr>
<tr>
<td><strong>Commutativity:</strong></td>
<td>(a + b = b + a)</td>
</tr>
<tr>
<td><strong>Existence of an identity element:</strong></td>
<td>(a + 0 = a)</td>
</tr>
<tr>
<td><strong>Existence of inverse elements:</strong></td>
<td>(a + (-a) = 0)</td>
</tr>
</tbody>
</table>

(not needed, if satisfied, commutative group)

- The composition (the application of one function to the results of another) of two automorphisms is another automorphism
- The set of automorphisms of a given graph, under the composition operation, forms a group, the automorphism group of the graph
Introduction to graph theory

- E.g.:
  - Node $a$ can be mapped to $c$ by an automorphism, while node $b$ can only be mapped to itself.
  - Thus: $\text{Orb}(a) = \{a, c\}$, $\text{Orb}(b) = \{b\}$.

- E.g.: in the graphs below, colors denote automorphism orbits.
Introduction to graph theory

- Basic definitions and graph types
- Graph representations
- Running times of algorithms
- Complexity classes
- Graph traversing and shortest path problems
**Graph Representation: Adjacency Matrix**

- **Adjacency matrix.** n-by-n matrix with $A_{uv} = 1$ if $(u, v)$ is an edge.
  - Two representations of each edge (symmetric matrix for undirected graphs; not for directed graphs).
  - Space: proportional to $n^2$.
    - Not efficient for **sparse graphs** (small number of edges compared to the maximum possible number of edges in the graph), e.g., biological networks
    - Algorithms might have longer running time if this representation used
  - Checking if $(u, v)$ is an edge takes $\Theta(1)$ time.
  - Identifying all edges takes $\Theta(n^2)$ time.
Graph Representation: Adjacency List

- **Adjacency list.** Node indexed array of lists.
  - Two representations of each edge.
  - Space proportional to \( m + n \).
  - Checking if \((u, v)\) is an edge takes \( O(deg(u)) \) time.
  - Identifying all edges takes \( \Theta(m+n) \) time = linear time for \( G(V,E) \).
  - Requires \( O(m+n) \) space. Good for dealing with sparse graphs.

degree = number of neighbors of \( u \)

1. \( 2 \to 3 \)
2. \( 1 \to 3 \)
3. \( 1 \to 2 \)
4. \( 2 \to 5 \)
5. \( 2 \to 3 \)
6. \( 5 \to 4 \)
7. \( 3 \to 8 \)
8. \( 3 \to 7 \)
Introduction to graph theory

- Basic definitions and graph types
- Graph representations
- Running times of algorithms
- Complexity classes
- Graph traversing and shortest path problems
Running Times of Algorithms

- **Algorithm:**
  - A sequence of computational steps that transform the input into the output
  - A tool for solving a well-specified computational problem
- **Examples:**
  - BLAST, Smith-Waterman,…
- **Analyzing an algorithm:**
  - Predicting the resources (computational time and memory space) required
- The **running time** of an algorithm: the number of steps executed as a function of its input (for G, input is V and E)
- **Worst-case running time**: the longest running time (an upper bound) for any input of size n.
- **Order of growth**: consider only the leading term of the running time, since lower-order terms are relatively insignificant for large n.
  - E.g., \( O(an^2 + bn + c) \) is \( O(n^2) \)
- An algorithm is **efficient** if its running time is low order polynomial (quadratic)
Running Times of Algorithms

Why It Matters?

Table 2.1 The running times (rounded up) of different algorithms on inputs of increasing size, for a processor performing a million high-level instructions per second. In cases where the running time exceeds $10^{25}$ years, we simply record the algorithm as taking a very long time.

<table>
<thead>
<tr>
<th>$n$</th>
<th>$n$</th>
<th>$n \log_2 n$</th>
<th>$n^2$</th>
<th>$n^3$</th>
<th>$1.5^n$</th>
<th>$2^n$</th>
<th>$n!$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n = 10$</td>
<td></td>
<td>&lt; 1 sec</td>
<td>&lt; 1 sec</td>
<td>&lt; 1 sec</td>
<td>&lt; 1 sec</td>
<td>&lt; 1 sec</td>
<td>4 sec</td>
</tr>
<tr>
<td>$n = 30$</td>
<td></td>
<td>&lt; 1 sec</td>
<td>&lt; 1 sec</td>
<td>&lt; 1 sec</td>
<td>&lt; 1 sec</td>
<td></td>
<td>18 min</td>
</tr>
<tr>
<td>$n = 50$</td>
<td></td>
<td>&lt; 1 sec</td>
<td>&lt; 1 sec</td>
<td>&lt; 1 sec</td>
<td></td>
<td>11 min</td>
<td>36 years</td>
</tr>
<tr>
<td>$n = 100$</td>
<td></td>
<td>&lt; 1 sec</td>
<td>&lt; 1 sec</td>
<td>&lt; 1 sec</td>
<td>1 sec</td>
<td>12,892 years</td>
<td>$10^{17}$ years</td>
</tr>
<tr>
<td>$n = 1,000$</td>
<td></td>
<td>&lt; 1 sec</td>
<td>&lt; 1 sec</td>
<td>&lt; 1 sec</td>
<td>18 min</td>
<td>very long</td>
<td>very long</td>
</tr>
<tr>
<td>$n = 10,000$</td>
<td></td>
<td>&lt; 1 sec</td>
<td>&lt; 1 sec</td>
<td>&lt; 1 sec</td>
<td>2 min</td>
<td>very long</td>
<td>very long</td>
</tr>
<tr>
<td>$n = 100,000$</td>
<td></td>
<td>&lt; 1 sec</td>
<td>2 sec</td>
<td>3 hours</td>
<td>32 years</td>
<td>very long</td>
<td>very long</td>
</tr>
<tr>
<td>$n = 1,000,000$</td>
<td></td>
<td>1 sec</td>
<td>20 sec</td>
<td>12 days</td>
<td>31,710 years</td>
<td>very long</td>
<td>very long</td>
</tr>
</tbody>
</table>

↑

linear  quadratic  cubic
Asymptotic Order of Growth

- **Upper bounds.** \( f(n) \) is \( O(g(n)) \) if there exist constants \( c > 0 \) and \( n_0 \geq 0 \) such that for all \( n \geq n_0 \) we have \( f(n) \leq c \cdot g(n) \).

- **Lower bounds.** \( f(n) \) is \( \Omega(g(n)) \) if there exist constants \( c > 0 \) and \( n_0 \geq 0 \) such that for all \( n \geq n_0 \) we have \( f(n) \geq c \cdot g(n) \).

- **Tight bounds.** \( f(n) \) is \( \Theta(g(n)) \) if \( f(n) \) is both \( O(g(n)) \) and \( \Omega(g(n)) \).

- **Ex:** \( f(n) = 32n^2 + 17n + 32 \).
  - \( f(n) \) is \( O(n^2), O(n^3), \Omega(n^2), \Omega(n), \) and \( \Theta(n^2) \).
  - \( f(n) \) is not \( O(n), \Omega(n^3), \Theta(n), \) or \( \Theta(n^3) \).
Introduction to graph theory

- Basic definitions and graph types
- Graph representations
- Running times of algorithms
- Complexity classes
- Graph traversing and shortest path problems
Complexity Classes

- **Polynomial-time algorithms:**
  - On input size $n$, their running time is $O(n^k)$
  - Not all problems can be solved in polynomial time (poly-time).
  - Intuition:
    - Polynomial time algorithms are tractable or “easy”
    - Problems that require “super-polynomial time” are “hard”

- Complexity classes:
  - **P**: problems that are solvable in polynomial time
  - **NP**: their solutions are verifiable in polynomial time, i.e., *decision problems* for which there exists a poly-time certifier

Remark. NP stands for *nondeterministic* polynomial time.
Complexity Classes

- E.g.
  - *Hamiltonian Cycle* of a graph $G(V,E)$ is a simple cycle that contains each vertex in $V$.
  - Problem: does a graph have a Hamiltonian Cycle? – NP-complete
  - If solution given, sequence $(v_1, v_2, v_3, ..., v_n)$ – easy to check in poly-time whether each $v_i, v_{i+1}$ in $E$ for all $i$ and $v_n, v_1$ in $E$.
Complexity Classes: $P$, $NP$

- **$P$.** Decision problems for which there is a poly-time algorithm.
- **$NP$.** Decision problems for which there is a poly-time certifier.

**Claim.** $P \subseteq NP$.

**Proof.** Consider any problem $X$ in $P$.

- By definition, there exists a poly-time algorithm $A$ that solves $X$.
- If we can solve in poly-time, we can verify a solution in poly time. □

**Does $P = NP$?** [Cook 1971, Edmonds, Levin, Yablonski, Gödel]

Is the decision problem as easy as the certification problem?
Clay $1$ million prize.

If yes: Efficient algorithms for $TSP$, $FACTOR$, $SAT$, …
If no: No efficient algorithms possible for $TSP$, $SAT$, …

Consensus opinion on $P = NP$? Probably no.
**Complexity Classes: P, NP**

**NP-complete.** A problem Y in NP with the property that for every problem X in NP, $X \leq_p Y$ (X is poly-time reducible to Y).

A is **poly-time reducible** to B if there exists a function $f: A \rightarrow B$ such that $\alpha$ is a yes instance for A if and only if $f(\alpha)$ is a yes instance for B and if $f$ is poly-time computable.

Problem L is **NP-complete** if:
- L is in NP
- every problem in NP is poly-time reducible to L (i.e., L is **NP-hard**: “at least as hard as any NP problem”)
Introduction to graph theory

- Basic definitions and graph types
- Graph representations
- Running times of algorithms
- Complexity classes
- Graph traversing and shortest path problems
Graph Traversing

- Given a graph $G(V,E)$, explore every vertex and every edge
- Using adjacency list is more efficient
- Example algorithms:
  - Depth-first search (DFS)
  - Breadth-first search (BFS)
Graph Traversing

- BFS example:
Graph Traversing

- BFS: code from LEDA

```cpp
(bfs)≡
void BFS(const graph& G, node s, node_array<int>& dist)
{
    queue<node> Q;
    node v,w;

    forall_nodes(w,G) dist[w] = -1;
    dist[s] = 0;
    Q.append(s);
    while (!Q.empty())
    { v = Q.pop();
        forall_adj_nodes(w,v)
           if (dist[w] < 0)
               { Q.append(w);
                   dist[w] = dist[v] + 1;
               }
    }
}
```

- Running time of BFS: linear, $O(|V|+|E|)$, using adjacency list
Graph Traversing

- **DFS applications:**
  - Determines whether $G$ is connected
  - Computes the connected components of $G$ (strongly connected components of a *digraph* = directed graph)
  - Path / cycle finding
  - Topological sort (ordering of vertices of digraph $G(V,E)$ such that for every edge $(u,v)$ in $E$, $u$ appears before $v$ in the ordering)
  - Linear running time

- **BFS applications:**
  - Computes the *distance* from $s$ to each reachable vertex in unweighted $G$
  - Finds *shortest paths* from $s$ to all other nodes in unweighted $G$
  - Finds a simple cycle, if there is one
  - Computes the connected components of $G$
Graph Traversing

- **Single-source shortest path problems (SSSPP):**
  - Given a source vertex \( s \), find distances and shortest paths from \( s \) to all other vertices
  - BFS works on unweighted graphs
  - **Dijkstra’s algorithm** for weighted graphs:
    - Each node is labeled with its distance from the source node along the best known path
      - Initially, all nodes are labeled with infinity
    - As the algorithm proceeds, labels may change
    - Label can be:
      - Non-permanent
      - Permanent
    - Initially, all labels are non-permanent
    - When label represents the shortest possible path from the source to that node, make it permanent
Graph Traversing

Dijkstra’s Shortest Path Algorithm

- Step 1: initially all nodes are “non-permanent”
- Step 2: set the source node (A) as permanent
  - A is at the same time the “current node”
- Step 3:
  - Examine all non-permanent nodes $i$ adjacent to the current node
  - For each $i$, calculate the cumulative distance from the source node to $i$ via the current node
  - Relabel $i$ with the newly computed distance
    - But if $i$ already has a shorter cumulative distance than the calculated one, then to NOT relabel.
  - Also, label $i$ with the name of the current node (as predecessor)
  - Compare labels (distances) of all non-permanent nodes and choose the one with the smallest value. Change the node to permanent and set it as the current node.
  - Repeat step 3 until all nodes become permanent.
Graph Traversing

Dijskstra’s Shortest Path Algorithm

Example
Graph Traversing

Dijkstra’s Shortest Path Algorithm

Example
Graph Traversing

Dijkstra’s Shortest Path Algorithm

Example
Graph Traversing

Dijkstra’s Shortest Path Algorithm

Example
Graph Traversing

Dijkstra’s Shortest Path Algorithm

Example
Graph Traversing

Dijkstra’s Shortest Path Algorithm

Example
Graph Traversing

Dijkstra’s Shortest Path Algorithm

Example
Graph Traversing

Dijkstra’s Shortest Path Algorithm

Example
Graph Traversing

Dijkstra’s Shortest Path Algorithm

Example
Graph Traversing

Dijkstra’s Shortest Path Algorithm

Example
Graph Traversing

Dijkstra’s Shortest Path Algorithm

Example
Graph Traversing

Dijkstra’s Shortest Path Algorithm

Example
Graph Traversing

Dijkstra’s Shortest Path Algorithm

Example
Graph Traversing

Dijkstra’s Shortest Path Algorithm

Example
Graph Traversing

Dijkstra’s Shortest Path Algorithm (SSSPP)

- Does not allow negative weights on edges
- Similar to BFS (BFS is this algorithm but with all weights equal to 1)
- Time complexity varies on implementation:
  - $O(|V|^2)$ – this is $O(|E|)$ for dense graphs
  - $O(|E| \log |V|)$ – good for sparse graphs (for them, $O(|E|)$ is of $O(|V|)$)
  - $O(|V| \log |V| + |E|)$ – good for both sparse and dense graphs
Graph Traversing

Belman-Ford Algorithm for SSSPP:

- Works on weighted, directed graphs
- Allows negative weights on edges, but no negative weight cycles
- If there is a negative weight cycle reachable from source vertex s, it reports no solution exists; otherwise produces the shortest paths and their weights

**B-F algorithm (G,s)**

- For each $v \in V$
  - $d[v] = \infty$
- $d[s] = 0$
- For $i=1$ to $|V|-1$
  - For each edge $(u,v) \in E$
    - If $(d[v] > d[u]+w(u,v))$
      - $d[v] = d[u] + w(u,v)$
  - For each edge $(u,v) \in E$
    - If $d[v] > d[u]+w(u,v)$
      - Return FALSE (negative weight cycle found)
Graph Traversing

All-pairs shortest paths

- Goal: create an $n \times n$ matrix of distance $\delta(u,v)$
- Use B-F algorithm once from each vertex as a source
  - But $O(|V|^2 |E|)$ running time, i.e., $O(|V|^4)$ running time for dense graphs
  - Can do slightly better with Dijkstra’s from each node, but no negative weight edges

- Use adjacency matrix representation of $G$ with entries being weights of edges, $w_{ij}$
- Negative weights are allowed, but no negative weight cycles (detects them if exist)
- **Floyd-Warshall algorithm**
  - Output:
    - A matrix of distances, $D$ (or equivalently, costs, $C$)
    - A predecessor matrix, $\Pi$
    - Dynamic programming algorithm (breaking down into smaller subproblems)
Graph Traversing

All-pairs shortest paths

- **Floyd-Warshall algorithm**
  - For $i=1$ to $n$
    - For $j=1$ to $n$
      - $d^{(0)}_{ij} = w_{ij}$ (\(d^{(k-1)}_{ij}\) is length of the shortest \(i,j\)-path using only \(\{1, 2, \ldots, k\}\) nodes)
  - For $k=1$ to $n$
    - For $i=1$ to $n$
      - For $j=1$ to $n$
        - $d^{(k)}_{ij} = \min\{d^{(k-1)}_{ij}, d^{(k-1)}_{ik} + d^{(k-1)}_{kj}\}$
  - Return $D^{(n)}$ (matrix of distances, or costs $C$)

```plaintext
for i = 1 to N
  for j = 1 to N
    if there is an edge from i to j
      dist[0][i][j] = the length of the edge from i to j
    else
      dist[0][i][j] = INFINITY

for k = 1 to N
  for i = 1 to N
    for j = 1 to N
      dist[k][i][j] = min(dist[k-1][i][j], dist[k-1][i][k] + dist[k-1][k][j])
```

\(O(n^3)\) time

\(O(n^2)\) space

(store only previous matrix)
Graph Traversing

Example

*Floyd-Warshall algorithm*

\[
D^{(0)} = \begin{pmatrix}
0 & 3 & 8 & \infty & -4 \\
\infty & 0 & \infty & 1 & 7 \\
\infty & 4 & 0 & \infty & \infty \\
2 & \infty & -5 & 0 & \infty \\
\infty & \infty & \infty & 6 & 0
\end{pmatrix}
\]

\[
\pi^{(0)} = \begin{pmatrix}
\text{NIL} & 1 & 1 & \text{NIL} & 1 \\
\text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\
\text{NIL} & 3 & \text{NIL} & \text{NIL} & \text{NIL} \\
4 & \text{NIL} & 4 & \text{NIL} & \text{NIL} \\
\text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL}
\end{pmatrix}
\]

\[
D^{(1)} = \begin{pmatrix}
0 & 3 & 8 & \infty & -4 \\
\infty & 0 & \infty & 1 & 7 \\
\infty & 4 & 0 & \infty & \infty \\
2 & \infty & -5 & 0 & -2 \\
\infty & \infty & \infty & 6 & 0
\end{pmatrix}
\]

\[
\pi^{(1)} = \begin{pmatrix}
\text{NIL} & 1 & 1 & \text{NIL} & 1 \\
\text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\
\text{NIL} & 3 & \text{NIL} & \text{NIL} & \text{NIL} \\
4 & 1 & 4 & \text{NIL} & 1 \\
\text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL}
\end{pmatrix}
\]

\[
D^{(2)} = \begin{pmatrix}
0 & 3 & 8 & 4 & -4 \\
\infty & 0 & \infty & 1 & 7 \\
\infty & 4 & 0 & 5 & 11 \\
2 & 5 & -5 & 0 & -2 \\
\infty & \infty & \infty & 6 & 0
\end{pmatrix}
\]

\[
\pi^{(2)} = \begin{pmatrix}
\text{NIL} & 1 & 1 & 2 & 1 \\
\text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\
\text{NIL} & 3 & \text{NIL} & 2 & 2 \\
4 & 1 & 4 & \text{NIL} & 1 \\
\text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL}
\end{pmatrix}
\]

\[
D^{(3)} = \begin{pmatrix}
0 & 3 & 8 & 4 & -4 \\
\infty & 0 & \infty & 1 & 7 \\
\infty & 4 & 0 & 5 & 11 \\
2 & -1 & -5 & 0 & -2 \\
\infty & \infty & \infty & 6 & 0
\end{pmatrix}
\]

\[
\pi^{(3)} = \begin{pmatrix}
\text{NIL} & 1 & 1 & 2 & 1 \\
\text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\
\text{NIL} & 3 & \text{NIL} & 2 & 2 \\
4 & 3 & 4 & \text{NIL} & 1 \\
\text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL}
\end{pmatrix}
\]

\[
D^{(4)} = \begin{pmatrix}
0 & 3 & -1 & 4 & -4 \\
3 & 0 & -4 & 1 & -1 \\
7 & 4 & 0 & 5 & 3 \\
2 & -1 & -5 & 0 & -2 \\
8 & 5 & 1 & 6 & 0
\end{pmatrix}
\]

\[
\pi^{(4)} = \begin{pmatrix}
\text{NIL} & 1 & 4 & 2 & 1 \\
4 & \text{NIL} & 4 & 2 & 1 \\
4 & 3 & \text{NIL} & 2 & 1 \\
4 & 3 & 4 & \text{NIL} & 1 \\
4 & 3 & 4 & 5 & \text{NIL}
\end{pmatrix}
\]

\[
D^{(5)} = \begin{pmatrix}
0 & 1 & -3 & 2 & -4 \\
3 & 0 & -4 & 1 & -1 \\
7 & 4 & 0 & 5 & 3 \\
2 & -1 & -5 & 0 & -2 \\
8 & 5 & 1 & 6 & 0
\end{pmatrix}
\]

\[
\pi^{(5)} = \begin{pmatrix}
\text{NIL} & 3 & 4 & 5 & 1 \\
4 & \text{NIL} & 4 & 2 & 1 \\
4 & 3 & \text{NIL} & 2 & 1 \\
4 & 3 & 4 & \text{NIL} & 1 \\
4 & 3 & 4 & 5 & \text{NIL}
\end{pmatrix}
\]