

Principles of Computer Science II

Introduction to Graph Theory

Ioannis Chatzigiannakis

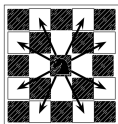
Sapienza University of Rome

Lecture 16



A little bit of Chess

- ▶ Knights move using a particular pattern.
- ▶ Knights can move two steps in any of four directions (left, right, up, and down) followed by one step in a perpendicular direction,
- ▶ Two points are connected by a line if moving from one point to another is a valid knight move.



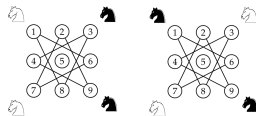
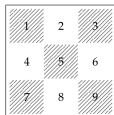
A Chess Puzzle

- ▶ Two white and two black knights on a 3×3 chessboard.
- ▶ Two Knights cannot occupy the same square.
- ▶ Starting from the top configuration,
- ▶ Can they move, using the usual chess knight's moves,
- ▶ To occupy the bottom configuration?



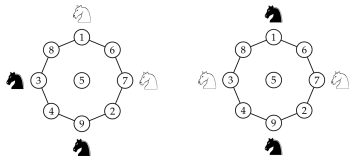
Chess Diagrams

- ▶ A Chess Diagram is used to represent movements of chess pieces on the board.
- ▶ Example of a 3×3 chessboard.
- ▶ Two points are connected by a line if moving from one point to another is a valid knight move.



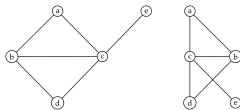
Chess Diagrams – Equivalent Representations

- ▶ An equivalent representation of the resulting diagram.
- ▶ Now it is easy to see that knights move around a "cycle".
- ▶ Every knight's move corresponds to moving to a neighboring point in the diagram – clockwise or counterclockwise
- ▶ white-white-black-black **cannot** be transformed into white-black-white-black

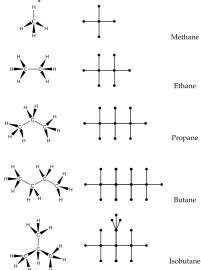


Chess Diagrams & Graphs

- ▶ Chess Diagrams are examples of *graphs*.
- ▶ The points are called vertices and lines are called edges.
- ▶ A simple graph of five vertices and six edges.
- ▶ We denote a graph by $G = G(V, E)$, where
 - ▶ V represents the set of vertices
 $V = \{a, b, c, d, e\}$
 - ▶ E represents the set of edges
 $E = \{(a, b), (a, c), (b, c), (b, d), (c, d), (c, e)\}$



Hydrocarbons as Graphs and Structural Isomers



Basic Definitions

- ▶ We denote $|V| = n$ – the number of vertices.
- ▶ We denote $|E| = m$ – the number of edges.
- ▶ Two vertices u, v are called **adjacent** or **neighboring** vertices if there exists an edge $e = (u, v)$.
- ▶ We say that edge e is **incident** to vertices u and v .
- ▶ We say that vertices u and v are **incident** to edge e .
- ▶ A **loop** is an edge from a node to itself: (u, u) .
- ▶ Two or more edges that have the same endpoints (u, v) are called **multiple edges**.
- ▶ The graph is called **simple** if it does not have any loops or multiple edges.

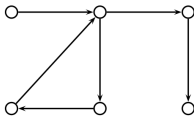
Degree of the Vertex

- ▶ The number of edges incident to a given vertex v is called the **degree of the vertex** and is denoted $d(v)$.
- ▶ For every graph $G = G(V, E)$, $\sum_{u \in V} d(u) = 2 \cdot |m|$.
- ▶ Notice that an edge connecting vertices v and w is counted in the sum twice: first in the term $d(v)$ and again in the term $d(w)$.



Directed & Undirected Graphs

- ▶ Many Bioinformatics problems make use of **directed graphs**.
- ▶ An edge can be **undirected** or **directed**.
- ▶ An undirected edge e is considered an unordered pair, in other words we assume that (u, v) and (v, u) are the same edge.
- ▶ A directed edge $e = (u, v)$ and $e' = (v, u)$ are different edges.
- ▶ If the edges have a direction, the **graph is directed** (digraph).
- ▶ If a graph has no direction, it is referred as **undirected**.



Directed Graphs

- ▶ In directed graphs, each vertex u has:
 - ▶ $indegree(u)$ – the number of incoming edges,
 - ▶ $outdegree(u)$ – the number of outgoing edges.
- ▶ For every directed graph $G = G(V, E)$,

$$\sum_{u \in V} indegree(u) = \sum_{u \in V} outdegree(u)$$



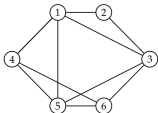
Subgraphs & Complete Graphs

- ▶ A **subgraph** G' of G consists of a subset of V and E . That is, $G' = (V', E')$ where $V' \subset V$ and $E' \subset E$.
- ▶ A **spanning subgraph** contains all the nodes of the original graph.
- ▶ If all the nodes in a graph are pairwise adjacent, the graph is called **complete**.



Triangles, Walks, Trails, Paths & Cycles

- ▶ A **triangle** in an undirected graph is a triplet (u, v, w) , where $u, v, w \in V$ such that $(u, v), (v, w), (w, u) \in E$.
- ▶ A **walk** is a sequence of vertices and edges of a graph – Vertex can be repeated. Edges can be repeated.
- ▶ **Trail** is a walk in which no edge is repeated.
- ▶ **Path** is a trail in which no vertex is repeated.
- ▶ Paths that start and end at the same vertex are referred to as **cycles**.



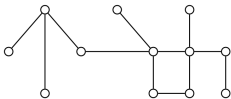
Paths

- ▶ A path of length k is a sequence of nodes (v_0, v_1, \dots, v_k) , where we have $(v_i, v_{i+1}) \in E$.
- ▶ If $v_i \neq v_j$ for all $0 \leq i < j \leq k$ we call the **path simple**.
- ▶ If $v_0 = v_k$ for all $0 \leq i < j \leq k$ and $v_0 = v_k$ the **path is a cycle**.
- ▶ A path from node u to node v is a path (v_0, v_1, \dots, v_k) such that $v_0 = u$ and $v_k = v$.



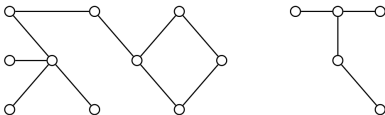
Graph Connectivity

- ▶ Two nodes u and v are **connected** if there is a path from u to v .
- ▶ A graph is called **connected** if all pairs of vertices can be connected by a path, otherwise we say that the graph is **disconnected**.
- ▶ A graph is called **complete** if there is an edge between every two vertices.



Graph Connectivity

- ▶ Disconnected graphs can be **decomposed** into a set of one or more **connected components**.



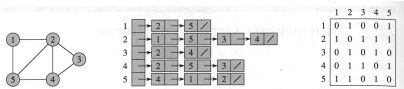
Forests & Trees

- ▶ A simple graph that does not contain any cycles is called a **forest**.
- ▶ A forest that is connected is called a **tree**.
- ▶ A tree has $n - 1$ edges.
- ▶ Any two of the following three statements imply that a graph is a tree (and thus they also imply the third one):
 1. The graph has $n - 1$ edges.
 2. The graph does not contain any cycles.
 3. The graph is connected.



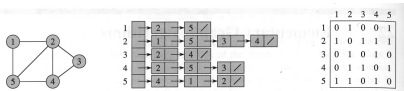
Representation of Graphs

- ▶ Two standard ways to represent a graph $G(V, E)$:
 1. A collection of adjacency lists.
 - ▶ Usually preferred for **sparse** graphs.
 - ▶ Sparse graph: $|E|$ is much less than $|V|^2$.
 2. An adjacency matrix.
 - ▶ Usually preferred for **dense** graphs.
 - ▶ Dense graph: $|E|$ is close to $|V|^2$.



Adjacency List

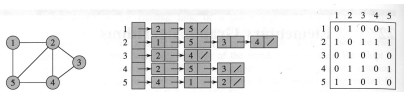
- ▶ Adjacency List Representation
- ▶ Consists of an array Adj of $|V|$ lists, one for each vertex in V .
- ▶ For each $u \in V$, the adjacency list $Adj[u]$ contains all the vertices adjacent to u in G .
- ▶ The vertices are stored in arbitrary order.



Adjacency Matrix

- ▶ Adjacency Matrix Representation of $G(V, E)$
- ▶ We assume that vertices are numbered $1, 2, \dots, |V|$.
- ▶ The matrix $|V| \times |V|$ matrix.
- ▶ $A = (a_{i,j})$, where

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



Adjacency List and Adjacency Matrix Examples

► Adjacency Matrix Representation



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

