Outline

- Introduction and basic definitions
- Bipartite graphs, stable marriage problem
- Operations on graphs, graph isomorphisms, connectivity
- Planar graphs
- Erdos-Renyi graphs (examples of random graphs)
- Trees
- Shortest path problems
  - Dijkstra’s link-state routing algorithm
  - Spanning trees of a graph
  - The Traveling Salesman Problem
Undirected and simple graphs

- A **undirected** graph $G = (V, E)$ consists of $V$, a nonempty set of vertices, and $E$, a set of **unordered pairs** of elements of $V$ called edges.
- That is, for an undirected graph $G=(V,E)$:
  $$\forall e \in E, \exists u,v \in V \text{ such that } e = (u,v)=(v,u).$$
- An edge $e$ is itself a **loop** if $e = (u,u)$ for some $u \in V$.
- An undirected graph may have edges that are loops and multiple (parallel) edges between the same pair of vertices.
- An **simple** graph is an undirected graph that contains no edges that are loops and no parallel edges.
- A **pseudograph** is an undirected graph that contains at least one loop or pair of parallel edges.

Directed Graphs

- A **directed graph** $G = (V, E)$ consists of a set $V$ of vertices and a set $E$ of edges that are **ordered** pairs of elements in $V$.
- That is, for a directed graph $G=(V,E)$:
  $$\forall e \in E, \exists u,v \in V \text{ s.t. } e = (u,v),$$
  where the (directed) edge $(u,v)$≠$(v,u)$.
- A (directed) edge $e$ is a loop if $e = (u,u)$ for some $u \in V$.
- So a directed edge $e=(u,v)$ has a tail $u$ and a head $v$, and is typically depicted by an arrow, i.e.,
  $$u \rightarrow v$$
Example – Trains between cities

- How can we represent a network of (bi-directional) railways connecting a set of cities?
- We should use a **simple graph** with an edge \((a, b)\) indicating a direct train connection between cities \(a\) and \(b\).

![Graph](image)

Example – Round-robin play

- In a round-robin tournament, each team plays against each other team exactly once.
- How can we represent the results of the tournament (which team beats which other team)?
- We should use a **directed graph** with an edge \((a, b)\) indicating that team \(a\) beats team \(b\).

![Graph](image)
Undirected edges

- **Definition:** Two vertices $u$ and $v$ in an undirected graph $G$ are called **adjacent** (or **neighbors**) in $G$ if $(u, v)$ is an edge in $G$.
- If $e = (u, v)$,
  - the edge $e$ is called **incident with** the vertices $u$ and $v$
  - the edge $e$ is also said to **connect** $u$ and $v$
- The vertices $u$ and $v$ are called **endpoints** of the edge $(u, v)$.

Vertex degree

- **Definition:** The **degree** of a vertex in an undirected graph is the number of edges incident with it, except that a loop at a vertex contributes twice to the degree of that vertex.
- In other words, you can determine the degree of a vertex in a displayed graph by **counting the lines** that touch it.
- The degree of the vertex $v$ is denoted by $\text{deg}(v)$. 
Isolated and pendant vertices

- A vertex of degree 0 is called **isolated**, since it is not adjacent to any vertex.
- **Note:** A vertex with a **loop** at it has at least degree 2 and, by definition, is **not isolated**, even if it is not adjacent to any other vertex.
- A vertex of degree 1 is called **pendant**. It is adjacent to exactly one other vertex.

Example – Vertex types

**Example:**

- Which vertices in the following graph are isolated, which are pendant, and what is the maximum degree?
- What type of graph is it?

**Solution:**

- Vertex f is isolated, and vertices a, d and j are pendant.
- The maximum degree is $\text{deg}(g) = 5$.
- This graph is a pseudograph (undirected edges with loops).
Example – Vertex degrees

Let us look at the same graph again and determine the number of edges and the sum of the degrees of all vertices:

Result:
- There are 9 edges, and the sum of all degrees is 18.
- This is easy to explain: Each new edge increases the sum of degrees by exactly two.

Undirected Graphs - Handshaking Theorem

- The Handshaking Theorem: If \( G = (V, E) \) is an undirected graph with \( |E| \) edges, then
  \[ 2|E| = \sum_{v \in V} \text{deg}(v) \]
- Proof: every edge contributes 2 to \( \sum_{v \in V} \text{deg}(v) \), 1 each for its incident vertices (even an edge loop by definition). Q.E.D.
- Example: How many edges are there in a graph with 10 vertices, each of degree 6?
- Solution:
  - The sum of the degrees of the vertices is \( 6 \cdot 10 = 60 \).
  - According to the Handshaking Theorem, it follows that \( 2|E| = 60 \), so there are \( |E| = 30 \) edges.
Undirected graphs – A corollary

**Theorem:** An undirected graph has an even number of vertices of odd degree.

**Proof:**

- Let $V_1$ and $V_2$ be the set of vertices of even and odd degrees, respectively.
- Thus, we have a partition of $V$: $V_1 \cap V_2 = \emptyset$, $V_1 \cup V_2 = V$.
- So, by Handshaking Theorem,
  
  $$2|E| = \sum_{v \in V} \deg(v) = \sum_{v \in V_1} \deg(v) + \sum_{v \in V_2} \deg(v)$$

- Since both $2|E|$ and $\sum_{v \in V_1} \deg(v)$ are even,
  
  $\sum_{v \in V_2} \deg(v)$ must be even.

- Since $\deg(v)$ if odd for all $v \in V_2$, $|V_2|$ must be even.

- QED

---

Edges of directed graphs

- **Definition:** When $(u, v)$ is an edge of the directed graph $G$ (i.e., with directed edges), $u$ is said to be **adjacent to** $v$, and $v$ is said to be **adjacent from** $u$.

- The vertex $u$ is called the **initial (tail) vertex** of $(u, v)$, and $v$ is called the **terminal (head) vertex** of $(u, v)$:
  
  $$u \rightarrow v$$

  is a depiction of $(u,v)$.

- The initial vertex and terminal vertex of an edge loop are the same.
In/out-degree of a vertex in a directed graph

- **Definition:** In a graph with directed edges, the **in-degree** of a vertex \( v \), denoted by \( \text{deg}(v) \), is the number of edges with \( v \) as their terminal vertex.

- The **out-degree** of \( v \), denoted by \( \text{deg}^+(v) \), is the number of edges with \( v \) as their initial vertex.

- **Question:** How does adding a loop to a vertex change the in-degree and out-degree of that vertex?

- **Answer:** It increases both the in-degree and the out-degree by one.

---

**Vertex degree - Example**

**Example:** What are the in-degrees and out-degrees of the vertices \( a, b, c, d \) in this graph.

Note that \( (d,b) \) and \( (b,d) \) are **not** parallel as they are in different directions.

\[
\begin{align*}
\text{deg}(a) &= 1 \\
\text{deg}^+(a) &= 2 \\
\text{deg}(d) &= 2 \\
\text{deg}^+(d) &= 1 \\
\text{deg}(b) &= 4 \\
\text{deg}^+(b) &= 2 \\
\text{deg}(c) &= 0 \\
\text{deg}^+(c) &= 2
\end{align*}
\]
Total vertex degree of a connected graph

• **Theorem:** If $G = (V, E)$ be a graph with directed edges, then

$$\sum_{v \in V} \text{deg}^{-}(v) = \sum_{v \in V} \text{deg}^{+}(v) = |E|.$$ 

• This is easy to see, because every edge contributes exactly 1 to both the sum of in-degrees (the edge’s head) and the sum of out-degrees (the edge’s tail).

Complete graphs

• The **complete graph** on $n$ vertices, denoted $K_n$, is the simple graph that contains exactly one edge between each pair of distinct vertices.

• Note that the number of (undirected) edges is $\binom{n}{2}$.
Cyclic Graphs

• The cycle $C_n$, $n \geq 3$, consists of $n$ vertices $v_1, v_2, ..., v_n$ and edges 
  $\{v_1, v_2\}, \{v_2, v_3\}, ..., \{v_{n-1}, v_n\}, \{v_n, v_1\}$.
  
• Note that the number of edges is $n$.

Wheel Graphs

• We obtain the wheel $W_n$ when we add an additional vertex to the cycle $C_n$, for $n \geq 3$, and connect this new vertex to each of the $n$ vertices in $C_n$ by adding new edges.
  
• The number of edges is $2(n-1)$.
Cubic Graphs

- **Definition:** The *n*-cube, denoted by $Q_n$, is the graph that has vertices representing the $2^n$ bit strings of length $n$.

- Two vertices are adjacent if and only if their bit-string representation differs in exactly one bit position (Hamming distance 1).

- Exercise: Prove by induction that the number of vertices of $Q_n$ is $2^n$ and the number of edges is $n2^{n-1}$, for integers $n > 0$.

![](image1.png)

Bipartite Graphs

- **Definition:** A simple graph $G=(V,E)$ is called bipartite if its vertex set $V$ can be partitioned into two disjoint nonempty sets $V_1$ and $V_2$ such that every edge in $E$ in the graph connects a vertex in $V_1$ with a vertex in $V_2$ (so that no edge in $E$ connects either two vertices in $V_1$ or two vertices in $V_2$).

- Recall our discussion of relations between two sets (domain and co-domain)

- For example, consider a graph that represents each person in a village by a vertex and each marriage by an edge.

- This graph is bipartite, because each edge connects a vertex in the subset of males with a vertex in the subset of females (if we think of traditional marriages).
Bipartite Graphs - Examples

- **Example I:** Is $C_3$ bipartite?
- No, because there is no way to partition the vertices into two sets so that there are no edges with both endpoints in the same set.

- **Example II:** Is $C_6$ bipartite?
- Yes, because we can display $C_6$ as on right:

Complete Bipartite Graphs

- **Definition:** The complete bipartite graph $K_{m,n}$ is the graph that has its vertex set partitioned into two subsets of size $m$ and $n$, respectively.
- Two vertices are connected if and only if they are in different subsets.
- Clearly, $K_{m,n}$ has $mn$ edges.
Stable Marriage Problem

- Suppose a bipartite graph connecting a set M vertices to a set N vertices.
- The objective is to find pairs of vertices \((a,b) \in M \times N\) best matched according to their preferences.
- Each vertex \(a \in M\) (resp. \(b \in N\)) has a list strictly ordered by preference \(N_a \subseteq N\) (resp. \(M_b \subseteq M\)).
- A marriage \(L \subseteq M \times N\) is maximal if \(|L| = \min\{|M|, |N|\}\).
- A maximal marriage \(L \subseteq M \times N\) is not stable if \(\exists (a,b) \notin L\) s.t. \(a\) and \(b\) both prefer each other over their existing partners in \(L\).
- So stability here is a kind of stalemate.
- **Exercise:** Relate this problem to that of the SLIP scheduling algorithm (McKeown et al.) for network switches.

Stable Marriage – Gale-Shapley Algorithm

- In each iteration of the GSA with \(L = \emptyset\) initially.
  1. Each \(a \in M\) “proposes” to the most preferred \(b \in N_a\) and \(N_a \rightarrow N_a \setminus \{b\}\)
  2. Each \(b \in M\) is then engaged to the most preferred \(a \in N_b\) among proposals in the previous Step 1 (of the current iteration) and its existing engagement (if any) in \(L\), and \((a,b)\) replaces the existing engagement for both \(a\) and \(b\) in \(L\).
  3. If \(\exists a \in M\) s.t. \(N_a \neq \emptyset\), then go to step 1.
- So, GSA is a kind of “greedy” algorithm.
- Note that the second part of step 1 means that GSA will terminate in a finite number of steps.
- **Exercise:** Argue by contradiction that the GSA will reach a maximal stable marriage \(L\) when \(|M| = |N|\).
Stable Marriage – example

• Consider a 3×3 example matching vertex-sets M={A,B,C} and N={X,Y,Z} with preferences decreasing from left to right
• There are three stable marriage arrangements:
  1. M-vertices get their first choice and N-vertices their third: (AY, BZ, CX)
  2. all participants get their second choice: (AX, BY, CZ)
  3. N-vertices get their first choice and M-vertices their third: (AZ, BX, CY)

Operations on Graphs - Subgraph

• **Definition:** A subgraph of a graph G = (V, E) is a graph H = (W, F) where W⊆V and F⊆E such that F only connects nodes in W.
• **Note:** Of course, H is a valid graph, so we cannot remove any end-point vertices of remaining edges when pruning G to create H.
• **Example:**

```
graph K_5

subgraph of K_5
```
Operations on Graphs - Unions

- **Definition:** The union of two simple graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is the simple graph with vertex set $V_1 \cup V_2$ and edge set $E_1 \cup E_2$.

- The union of $G_1$ and $G_2$ is denoted by $G_1 \cup G_2$.

![Graphs](image)

Representing graphs by listing edges: undirected, directed examples

Note redundancy in table representation of undirected graph as, e.g., $(a,b)=(b,a)$.

<table>
<thead>
<tr>
<th>Vertex</th>
<th>Adjacent Vertices</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b, c, d</td>
</tr>
<tr>
<td>b</td>
<td>a, d</td>
</tr>
<tr>
<td>c</td>
<td>a, d</td>
</tr>
<tr>
<td>d</td>
<td>a, b, c</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Initial Vertex</th>
<th>Terminal Vertices</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>c</td>
</tr>
<tr>
<td>b</td>
<td>a</td>
</tr>
<tr>
<td>c</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>a, b, c</td>
</tr>
</tbody>
</table>
Representing Graphs – Adjacency Matrix

- **Definition:** Let $G = (V, E)$ be a simple graph with $|V| = n$. Suppose that the vertices of $G$ are listed in arbitrary order as $v_1, v_2, \ldots, v_n$.

- The **adjacency matrix** $A$ (or $A_G$) of $G$, with respect to this listing of the vertices, is the $n \times n$ zero-one matrix with 1 as its $(i, j)$th entry when $v_i$ and $v_j$ are adjacent, and 0 otherwise.

- In other words, for an adjacency matrix $A = [a_{ij}]$,
  - $a_{ij} = 1$ if $\{v_i, v_j\}$ is an edge of $G$, and
  - $a_{ij} = 0$ otherwise.

**Adjacency Matrix Example**

- **Example:** What is the adjacency matrix $A_G$ for the following graph $G$ based on the order of vertices a, b, c, d?

- **Solution:**

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

  **Note:** Adjacency matrices of undirected graphs are always symmetric.
Incidence matrices

- **Definition:** Let $G = (V, E)$ be an undirected graph with $|V| = n$. Suppose that the vertices and edges of $G$ are listed in arbitrary order as $v_1, v_2, ..., v_n$ and $e_1, e_2, ..., e_m$, respectively.
- The **incidence matrix** of $G$ with respect to this listing of the vertices and edges is the $n \times m$ zero-one matrix with 1 as its $(i, j)$th entry when edge $e_j$ is incident with $v_i$, and 0 otherwise.
- In other words, for an incidence matrix $M = [m_{ij}]$,
  - $m_{ij} = 1$ if edge $e_j$ is incident with $v_i$
  - $m_{ij} = 0$ otherwise.

Representing Graphs

- **Example:** What is the incidence matrix $M$ for the following graph $G$ based on the order of vertices $a, b, c, d$ and edges $1, 2, 3, 4, 5, 6$?
- **Solution:**

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

**Note:** Incidence matrices of undirected graphs contain two 1s per column for edges connecting two vertices and one 1 per column for loops.
Isomorphism of Graphs - Definition

- The simple graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are **isomorphic** if there is a bijection (an one-to-one and onto function) $f: V_1 \rightarrow V_2$ with the property that: $\forall a, b \in V_1$, $a$ and $b$ are adjacent in $G_1$ if and only if $f(a)$ and $f(b)$ are adjacent in $G_2$.

- Such a function $f$ is called an **isomorphism**.

- In other words, $G_1$ and $G_2$ are isomorphic if their vertices can be ordered in such a way that the adjacency matrices $M_{G_1}$ and $M_{G_2}$ are identical.

Checking Isomorphism of Graphs

- From a visual standpoint, $G_1$ and $G_2$ are isomorphic if they can be arranged in such a way that their displays are identical (of course without changing adjacency).

- Unfortunately, for two simple graphs, each with $n$ vertices, there are $n!$ possible **isomorphisms** that we have to check in order to show that these graphs are isomorphic.

- However, showing that two graphs are **not** isomorphic can be easy.
Disproving graph isomorphism by checking invariants

• For this purpose we can check invariants, that is, properties that two isomorphic simple graphs must both have.

• For example, they must have the same:
  – number of vertices $|V|$
  – number of edges $|E|$
  – degree distribution of vertices
  – distribution of minimum paths between vertex pairs (cf. Dijkstra algorithm)
  – distribution of minimum coloring (such that no two adjacent nodes have the same color)

• Note that two graphs that differ in any of these invariants are not isomorphic, but two graphs that match in all of them are not necessarily isomorphic.

Isomorphism of Graphs – Example I

Are the following two graphs isomorphic?

![Graph 1](image1.png)

![Graph 2](image2.png)

Solution:
• Yes, they are isomorphic, because they can be arranged to look identical.
• You can see this if in the right graph you move vertex $b$ to the left of the edge $\{a, c\}$.
• Then the isomorphism $f$ from the left to the right graph is:
  \[ f(a) = e, f(b) = a, f(c) = b, f(d) = c, f(e) = d. \]
Isomorphism of Graphs – Example II

• Are these two graphs isomorphic?

Solution:
• No, they are not isomorphic, because they differ in the degrees of their vertices.
• Vertex d in right graph is of degree one, but there is no pendant vertex in the left graph.

Graph Connectivity – Definition of Path

• A path of length \( n \in \mathbb{Z}^+ \) from vertex \( u \) to \( v \) in an undirected or directed graph is a sequence of edges \( e_1, e_2, \ldots, e_n \) of the graph such that
\[
e_1 = (x_0, x_1), e_2 = (x_1, x_2), \ldots, e_n = (x_{n-1}, x_n), \quad \text{where } x_0 = u \text{ and } x_n = v.
\]
• That is, successive edges on the path meet at the path’s interior vertices \( x_1, x_2, \ldots, x_{n-1} \) (meet head-to-tail for a directed graph).
• When the graph is simple, we denote this path by its vertex sequence \( x_0, x_1, \ldots, x_n \), since it uniquely determines the path.
• The path is a circuit if it begins and ends at the same vertex, that is, if \( u = v \).
• The path or circuit is said to pass through or traverse \( x_1, x_2, \ldots, x_{n-1} \).
• A path or circuit is simple if it does not contain the same edge more than once.
Graph Connectivity - Definition

- An undirected graph is called **connected** if there is a path between every pair of distinct vertices in the graph.
- For example, any two computers in a network can communicate if and only if the graph of their network is connected.
- **Note:** A graph consisting of only one vertex is always connected, because it does not contain any pair of distinct vertices.

Connectivity - Examples

- **Example:** Are the following graphs connected?

  1. ![Connected Graph](image)
     - Yes.
  2. ![Disconnected Graph](image)
     - No.
  3. ![Connected Graph](image)
     - Yes.
  4. ![Disconnected Graph](image)
     - No.
Connected components of a disconnected graph

- **Definition:** A graph that is not connected is the union of two or more connected subgraphs, each pair of which has no vertex in common.
- These disjoint connected subgraphs are called the **connected components** of the graph.

**Example:** What are the connected components in the following graph?

**Solution:** The connected components are the graphs with vertices \{a, b, c, d\}, \{e\}, \{f\}, \{f, g, h, j\}.
Strong and Weak Directed Graph Connectivity

- **Definition**: An *directed* graph is **strongly connected** if there is a path from \( a \) to \( b \) and from \( b \) to \( a \) whenever \( a \) and \( b \) are vertices in the graph.

- **Definition**: An *directed* graph is **weakly connected** if there is a path between any two vertices in the *underlying undirected* graph.

**Connectivity - Example**

**Example**: Are the following directed graphs strongly or weakly connected?

- **Weakly connected**, because, for example, there is no path from \( b \) to \( d \).

  ![Weakly Connected Example](attachment:weakly_connected_example.png)

- **Strongly connected**, because there are paths between all possible pairs of vertices.

  ![Strongly Connected Example](attachment:strongly_connected_example.png)
Undirected Planar Graphs

- A graph is **planar** if it can be drawn so that its edges do not intersect except at vertices.
- Kuratowski proved that a graph is planar if it contains neither a fully connected 5-vertex subgraph nor a fully connected 3×3 bipartite subgraph.
- **Exercise:** Prove Kuratowski’s theorem by first taking two cases: graphs with fewer than 7 vertices or more than 6, and exhaustively consider all (sub)graphs (up to isomorphism) of the former case.

Planar Graphs – Euler’s formula

- For a planar graph, let $N=|V|$ be the number of vertices, $L=|E|$ the number of edges and $F$ the number of “faces”, i.e., contiguous regions bounded by edges including the outer region beyond the graph, i.e. a partition of the plane.
- **Euler’s formula for connected planar graphs without pendants:** $N-L+F=2$
- For the example graph at left, $N=4$, $L=5$ and $F=3$.
- Note that the other graph is isomorphic to the first, hence also planar with $N=4$, $L=5$ and $F=3$.
- **Exercise:** Try to prove Euler’s formula by induction on the number of vertices (a vertex will increase the number of faces by the number of edges also added minus 1, i.e., $N-L+F$ doesn’t change).
A family of random graphs: Erdos-Renyi

- Consider a simple undirected graph $G(V,E)$ with $N=|V|$ vertices/nodes and $L=|E|$ edges/links.
- So, the number of edges $L \leq C(N,2)$
- Recall that the sum of node degrees $= 2L$ (Handshake Theorem).
- Let $G(N; p)$ be the random simple graph (a random object) consisting of $N$ vertices where each edge independently exists with probability $p$, i.e., an independent Bernoulli random variable for each edge.
- So, the distribution of the number of edges $L$ is binomial:
  \[ P(L = m) = C(C(N,2),m) p^m (1 - p)^{C(N,2) - m} , \]
  \[ EL = \sum_{m=0}^{C(N,2)} C(N,2) m P(L=m) = C(N,2)p \quad \text{(which is intuitive)} \]
- Let $D$ be the random variable representing the degree of a node of the graph:
  \[ P(D=k) = C(N-1,k) p^k (1-p)^{N-1-k} , \]
  So, $N ED=2EL$ (handshake) $\Rightarrow$ $ED=(N - 1)p$ (intuitively)

Erdos-Renyi graphs (cont)

- As $N \to \infty$ and $p \to 0$, the graph $G(N;p)$ becomes sparse.
- By the law of small numbers, if
  \[ \lim_{N \to \infty, p \to 0} p C(N,2) = ED \]
  with $0 < ED < \infty$, then the (binomial) degree distribution tends to Poisson:
  \[ P(D=k) = (ED)^k e^{-ED} / k! \]
- Recall:
  - A path in a graph is a series of connected edges.
  - A graph is connected if every pair of vertices is connected by a path.
- Among the results proved by Erdos and Renyi in 1960 are:
  - If $p(N) > \log(N)/N$, then $\lim_{N \to \infty} P(G(N;p(N)) \text{ is connected}) = 1$, i.e., $G(N;p(N))$ becomes connected almost surely (a.s.)
  - If $p = 1/N$, then as $N \to \infty$, the largest (principle) connected component of $G(N;p)$ will a.s. have on the order of $N^{2/3}$ vertices.
Trees

- In a graphical tree with directed edge \((a,b)\), i.e., “\(a \rightarrow b\)”, vertex \(b\) is the child of \(a\) and vertex \(a\) is the parent of \(b\).
- The leaves of a tree are the vertices without any children.
- A directed graph \(G=(V,E)\) is called a **tree** if:
  - it is weakly connected and with only \(|V|-1\) edges, i.e., \(|E|=|V|-1\) (consistent with Euler’s formula for planar graphs);
  - There is a single root vertex without any parents, i.e., zero in-degree (consistent with \(|E|=|V|-1\)); and
  - There is a directed path from the root to every leaf.
- Note that the number of edges of the tree are the minimal number required for it to be thus connected; if \(|E|>|V|-1\), one can prove by the pigeonhole principle that there must be a non-root vertex that either has two different parents or parallel edges.
- Thus, every non-root vertex has a **unique** parent vertex (in-degree =1), and a tree has no “weak” circuit paths.
- Undirected trees can be similarly defined, where any node can play the role of the root.

**Trees – Example with 8 leaves, depth 3**

- Note that for a tree with bidirectional edges, any node can “serve” as root.
- The choice of root may affect the depth of the tree, however.
Counting Trees

• A binary/Boolean tree is one where the out-degree of every parent node, including the root, is 2.

• How many different (non-isomorphic) binary trees $T(N)$ are there that span a set of $N$ distinct leaf vertices?

• Solution:
  – Obviously, $T(1)=1$ (single node with no edges).
  – Thinking inductively for $N>1$, if the $N^{th}$ leaf $X$ is added to a binary tree with $N-1$ leaves, one of the existing leaf nodes $Y$ is replaced by a parent node and $X$ and $Y$ become children of that parent; since there are $N-1$ ways that this can happen (minding graph isomorphisms (i.e., it doesn’t matter whether the children are arranged $X,Y$ or $Y,X$):
    $$T(N) = (N-1)T(N-1)$$
  – Thus, $T(N)=(N-1)!$

Counting Trees (cont)

• How many different (non-isomorphic) binary trees $T(L)$ are there that span a set of $N$ distinct vertices (leaf or parent), i.e., $|V|=|L|$?

• Consider a root with a certain collection of $R \leq N$ vertices to the right of it, and $N-R$ vertices to the left.

• $T(N) = \sum_{R=0}^{\lfloor N/2 \rfloor} C(N,R)T(R)T(N-R)$

• Note how we stop summing at $\lfloor N/2 \rfloor$ to avoid counting isomorphic graphs we have already counted.

• Obviously, $T(1)=1$.

• Exercise: Write a program that can compute $T(N)$ for all $N$. 

(s) 2013 George Kesidis
Tree Traversal (Node Enumeration)

- Different ways of traversing a tree, visiting each node once.
- Each way can be used to enumerate the nodes of or search the tree.
- For a tree depicted with root at top and leaves below, can **recursively** traverse tree **depth-first**, e.g. first from left then right in:
  1. **Pre-order** - record parent nodes on downward edge moves
  2. **In-order** - record child nodes on upward edge moves
- Alternatively, **breadth-first**, e.g. first from left to right, then down.
- **Exercise:** Given a doubly-linked data-structure (one record per node) of a tree, write a recursion for depth-first search. And for breadth-first, how much memory is required?

Tree Traversal Example

```
Pre-order (left to right): radeijbcfgklmh
Post-order (left to right): dijeabklmghcr
Breadth first (left to right): rabcdeghijklm
```
Shortest Path Problems

• We can assign weights to the edges of graphs, for example to represent the distance between cities in a railway network:

![Graph with cities and weights](image)

• Such weighted graphs can also be used to model computer networks with response times or costs as weights.

• One of the most interesting questions that we can investigate with such graphs is:

• What is the **shortest path** between two vertices in the graph, that is, the path with the **minimal sum of weights** along the way?

• This corresponds to the shortest train connection or the fastest connection in a computer network.

• Note that when all edges have the same weight, the additive path cost is proportional to the number of edges, a quantity commonly expressed as “hops”.


Shortest Path Problems: Example

- Suppose we are planning the construction of a highway from city A to city K.
- Different construction alternatives and their “edge” costs $g \geq 0$ between directly connected cities (nodes) are given in the following graph.
- The problem is to determine the highway (edge sequence) with the minimum total (additive) cost.

Bellman’s principle of optimality

- Bellman’s principle of optimality - shortest path example
- If C belongs to an optimal (by edge-additive cost $J^*$) path from A to B, then the sub-path A to C and C to B are also optimal,
- i.e., any sub-path of an optimal path is optimal (easy proof by contradiction).
Dijkstra’s link-state routing algorithm

- Dijkstra’s iterated algorithm uses the predecessor node of destination (path penultimate node) and is based on complete link-state (edge-state) information consistently shared among all nodes:
  \[ J^*(A, B) = \min\{J^*(A, C) + g(C, B) \mid C \text{ is a predecessor of } B\}, \]
  i.e., C and B are adjacent nodes (endpoints of the same edge).

- **Note:** The iterated distributed Bellman-Ford algorithm instead uses the successor node of the path origin and only nearest-neighbor distance-vector information sharing:
  \[ J^*(A, B) = \min\{g(A, C) + J^*(C, B) \mid C \text{ is a successor of } A\} \]

- Both Dijkstra’s (OSPF, ISIS) and the dBF (RIP) algorithms are the basis of different Interior Gateway Routing Protocols (IGPs) in the Internet.

---

Example: Dijkstra’s algorithm at A

Initially set of optimal paths (by destination) \( \Omega = \{A\} \), \( J^*(A, A) = 0 \)

Dijkstra’s algo finds optimal paths in order of increasing lengths

1. \( A \to B \) (via A), \( \Omega = \{A, B\} \), \( J^*(A, B) = 8 = g(A, B) + J^*(A, A) \)

2. \( A \to C \) (via A), \( \Omega = \{A, B, C\} \), \( J^*(A, C) = 10 = g(A, C) + J^*(A, A) \)

3. \( C \to F \) (\( A \to F \) via C), \( \Omega = \{A, B, C, F\} \), \( J^*(A, F) = 17 = g(C, F) + J^*(A, C) \)

4. \( B \to E \) (\( A \to E \) via B), \( \Omega = \{A, B, C, F, E\} \), \( J^*(A, E) = 18 = g(B, E) + J^*(A, B) \)

5. \( B \to D \) (\( A \to D \) via B), \( \Omega = \{A, B, C, F, E, D\} \), \( J^*(A, D) = 22 = g(B, D) + J^*(A, B) \)

6. \( E \to H \) (\( A \to H \) via E) etc. (exercise)
Spanning Trees

- A **spanning tree** of a graph is a tree subgraph whose vertices are a certain subset (or perhaps all) of the graph’s vertices.
- Edges may be weighted.
- Note that Dijkstra’s algorithm creates a spanning tree.
- **Kruskal’s algorithm** is a greedy approach that finds a spanning tree of *minimum total weight*.
- **Steiner trees** are minimal spanning trees for subgraphs.
- The Internet’s IGMP uses a decentralized approach to find a tree spanning subscribers (leaves) to a multicast session.

Spanning Trees – Kruskal’s algorithm

1. Initially, \( G=(V,E) \) is an undirected graph with weighted edges.
2. Also create an initial forest \( F \) (a set of trees), where initially each vertex of the graph is a (single-vertex) tree of the forest, i.e., \( F=V \).
3. While \( E \neq \emptyset \) and \( |F| > 1 \)
   i. remove an edge with minimum weight from \( E \)
   ii. if the removed edge connects two different trees in \( F \) then combine these two trees into a single tree in \( F \), i.e., \( |F| \rightarrow |F| - 1 \)

- **Exercise:** Prove by contradiction that if the graph is connected, then at the termination of Kruskal’s algorithm, \( |F| = 1 \) and the single element in \( F \) is a minimum spanning tree.
- **Exercise:** Run Kruskal’s algorithm on the example used for Dijkstra’s algorithm. Compare the resulting spanning tree with that obtained by Dijkstra’s algorithm operating at A in terms of total edge weight.
The Traveling Salesman Problem

- The **traveling salesman problem** is one of the classical problems in computer science.

- A traveling salesman wants to visit a number of cities and then return to his starting point. Of course he wants to save time and energy, so he wants to determine the **shortest path** for his trip.

- We can represent the cities and the distances between them by a weighted, complete, undirected graph.

- The problem then is to find the circuit of minimum total weight that visits each vertex exactly once.

---

Example: What path would the traveling salesman take to visit the following cities?

![Diagram of cities](image)

Solution: The shortest path is Boston, New York, Chicago, Toronto, Boston (2,000 miles).
The Traveling Salesman Problem

• **Question:** Given \( n \) vertices, how many different cycles \( C_n \) can we form by connecting these vertices with edges?

• **Solution:** We first choose a starting point. Then we have \((n - 1)\) choices for the second vertex in the cycle, \((n - 2)\) for the third one, and so on, so there are \((n - 1)!\) choices for the whole cycle.

• However, this number includes identical cycles that were constructed in **opposite directions**. Therefore, the actual number of different cycles \( C_n \) is \((n - 1)!/2\).

The Traveling Salesman Problem

• Unfortunately, no algorithm solving the TSP with polynomial worst-case time complexity has been devised yet.

• This means that for large numbers of vertices, solving the TSP may be impractical.

• Other computationally difficult problems can be shown to be equivalent in complexity to the TSP, i.e., they’re also “NP-hard”.

• Sometimes there are more computationally feasible approaches but that lead to suboptimal solution.

• Or just apply vast, distributed computational resources particularly to a simple (e.g. exhaustive search) method.