8 Graphs and O-Notation

Königsberg Bridge Problem

**Problem:** Cross every bridge exactly once and return to your starting point.

**Solution:** Solved by Euler (1736).

**Key Idea:** Every time you enter a point you must be able to leave. Hence the number of lines at each point must be even. This is not true in the above graph.
Definition: A graph $G(V, E)$ consists of a finite nonempty set $V = V(G)$ of vertices and a set $E = E(G)$ of unordered pairs of distinct vertices of $V$ called edges.

Example: $G = G(V, E)$ is given below.

$V = \{v_1, v_2, v_3, v_4\}$.
$E = \{v_1v_2, v_2v_3, v_1v_4, v_3v_4, v_2v_4\}$.

The graph can be represented geometrically as follows:

Note: This is also a representation of $G$.

Graphs, as defined above, are often called simple graphs.

Terminology: $V$ consists of points, vertices, nodes and $E$ consists of lines, edges, arcs.
Other types of graphs can also be defined.

1. **Multigraph**: Multiple edges are allowed between any given pair of vertices.

2. **Pseudograph**: Loops and multiple edges are allowed.

3. **Directed Graph**: The edges are *ordered* pairs of vertices instead of unordered pairs of vertices.

4. **Network**: A graph or directed graph with a function $f : E \to R$.

*Note*: The term “graph” will always refer to a simple graph unless otherwise stated.
A few more definitions.

**Definition:** A **complete graph** on \( n \) vertices, denoted \( K_n \), is a graph with an edge joining each pair of distinct vertices.

**Example:**

---

**Definition:** A graph \( G \) is called **bipartite** if its vertices can be divided into two sets, \( U \) and \( W \), such that every edge in \( G \) joins a vertex of \( U \) to a vertex of \( W \).

**Example:**

---

**Definition:** A **complete bipartite graph** on \( (m,n) \) vertices, denoted \( K_{m,n} \), is a bipartite graph as defined above with \( |U| = m, |W| = n \), and with an edge joining each vertex in \( U \) to each vertex in \( W \).

**Example:**
Traversing a Graph

Definition: A walk in a graph $G$ is an alternating sequence of vertices and edges, $v_0e_1v_1e_2 \ldots v_{n-1}e_nv_n$, beginning and ending with vertices, such that every edge is immediately preceded and succeeded by the two vertices with which the edge is incident.

Consider $G$ below.

Example: $v_1v_6v_2v_6v_7v_3$ is a $v_1 - v_3$ walk (the edges are usually omitted).

Note: If $v_0 = v_n$ the walk is closed.

Definition: A trail is a walk in which all edges are distinct.

Note: The text calls this a path.

Example: $v_1v_2v_6v_5v_1v_6v_7$ is a trail.

Note: If $v_0 = v_n$ the trail is closed and is called a circuit.

Definition: A path is a trail in which all vertices are distinct.
Note: The text calls this a “simple path”.

Example: \( v_1v_6v_2v_3 \) is a path.

Definition: A walk \( v_0v_1v_2\ldots v_{n-1}v_n \) is a cycle if \( v_0v_1v_2\ldots v_{n-1} \) are all distinct and \( v_0 = v_n \).

Note: The text calls this a “simple circuit”.

Example: \( v_1v_2v_3v_7v_6v_1 \) is a cycle.

Definition: A graph is connected if there exists a path between every pair of vertices.

Example:

\( G \) is not connected.
**Definition:** A graph $H$ is a **subgraph** of a graph $G$ if, and only if, all vertices and edges of $H$ are also in $G$.

**Definition:** A graph $H$ is a **connected component** of a graph $G$ if, and only if

1. $H$ is a subgraph of $G$
2. $H$ is connected
3. no connected subgraph of $G$ has $H$ as a subgraph and contains vertices or edges that are not in $H$.

**Example:** $G$ has 2 components. $H$ is **not** a component of $G$.

**Definition:** The **degree** of a vertex is the number of edges incident to it.

**Example:**

\[
\text{deg (a)} = 2 \quad \text{deg (d)} = 4 \quad \text{deg (e)} = 1
\]

**Theorem 8.1** In any graph $G$, the sum of the degrees of all the vertices equals twice the number of edges of $G$.

**Proof:** Each edge contributes two to the sum of the degrees.

**Corollary 8.2** In any graph there is an even number of vertices of odd degree.
Definition: A **tree** is a connected acyclic graph, i.e., a connected graph with no cycles.

Example:

Definition: A **forest** is a disjoint union of trees.

Example:

**Theorem 8.3** If $T$ is a tree with $p$ points and $q$ lines, then $p = q + 1$.

**Proof:** The proof is by induction on $p$.

Since there is only one tree with 1 point (it has no lines), the result holds for $p = 1$.

Suppose the theorem holds for all trees with $p$ points and let $T'$ be a tree with $p + 1$ points. The proof follows easily from the following claims.
Claim 1: $T'$ has a point $u$ of degree 1.

Hint: Consider a longest path in $T'$.

Claim 2: $T = T' - u$ is a tree.

By induction, $T$ has $p$ points and $p - 1$ lines. Hence $T'$ has $p + 1$ points and $p$ lines. $\square$

Note: If we substitute the word “forest” for the word “tree” in the previous theorem, the result is false.

Reason: Claim 1 is false for forests.

Example:
Definition: Graphs $G$ and $H$ are **isomorphic**, denoted $G \simeq H$, if there exists a 1-1 correspondence $\phi$ between the vertices of $G$ and the vertices of $H$ that preserves adjacency, i.e.,

$$\{uv\} \in E(G) \iff \{\phi(u)\phi(v)\} \in E(H).$$

An isomorphism $\phi$ is given by

$$\begin{align*}
\phi(v_1) &= u_1 \\
\phi(v_2) &= u_3 \\
\phi(v_3) &= u_5 \\
\phi(v_4) &= u_2 \\
\phi(v_5) &= u_4 \\
\phi(v_6) &= u_6.
\end{align*}$$
Note: There is no known easy way (even for a computer) to determine if two graphs are isomorphic. If graphs $G$ and $H$ each have 20 vertices and 20 edges, and a computer checks “all possibilities” at 1 microsecond per check, it will take $1.9 \times 10^{23}$ years to determine if $G \simeq H$.

Note: $20! = 2432902008176640000$

Note: By Sterling’s Formula $n! \approx \sqrt{2\pi nn}e^{-n}$.

It is much easier to determine that two graphs are not isomorphic.

Definition: A property $P$ is called an isomorphic invariant if, and only if, given graphs $G$ and $H$, if $G$ has property $P$ and $G \simeq H$, then $H$ has property $P$.

Example: The number of vertices in a graph is an isomorphic invariant. So is the number of edges.
Planar Graphs

*Definition:* A graph $G$ is *planar* if it can be “embedded” in the plane.

*Example:* To see that $G$ is planar,

observe that $G$ can be drawn as follows.

*Definition:* An embedding of a planar graph in the plane is called a *plane graph.*
Theorem 8.4 (Euler’s Formula)
Let $G$ be a connected plane graph with $n$ vertices, $m$ edges and $r$ regions. Then
\[ n - m + r = 2. \]

Proof: By induction on $m$.

Suppose $m = 0$. Then $G$ is just a single vertex. Hence $n = 1, m = 0$ and $r = 1$ and the formula holds.

Now, by induction, let $m \geq 1$ and suppose the formula holds for all connected plane graphs having fewer than $m$ edges. Let $G$ be a connected plane graph having $m$ edges. If $G$ is a tree, then $m = n - 1$ and $r = 1$, so again the formula holds. Otherwise, let $e$ be an edge of $G$ that lies on a cycle, and consider the graph
\[ G' = G - e. \]
Then $G'$ has $m - 1$ edges and $r - 1$ regions. By induction
\[ n - (m - 1) + (r - 1) = 2 \]
i.e., $n - m + r = 2$. 

120
**Definition:** Let $e = uv$ be an edge in a graph $G$. An *elementary subdivision* of $G$ is the graph $G'$ obtained by eliminating $e$ and adding a new vertex $w$ plus edges $uw$ and $wv$.

**Definition:** A *subdivision* of a graph $G$ is a graph obtained by a sequence of elementary subdivisions (possibly none).

**Example:** $G$ is a subdivision of $H$

**Theorem 8.5 (Kuratowski)** A graph $G$ is planar if and only if it contains no subgraph that is a subdivision of either $K_5$ or $K_{3,3}$. 

Königsberg Bridge Problem (revisited)

Definition: A circuit in a graph $G$ is eulerian if it contains all the edges of $G$. A graph $G$ is eulerian if it contains an eulerian circuit.

**Theorem 8.6** A graph $G$ without isolated vertices is eulerian iff $G$ is connected and the degree of every vertex of $G$ is even.

*Note:* The theorem is not obvious.

**Outline of Proof:** ($\Rightarrow$) The argument is easy - essentially using the key idea at the beginning of the lecture.

($\Leftarrow$) Choose a longest trail $T$ (with respect to the number of edges) in $G$. We will show that $T$ is an eulerian circuit.

*Note:* The vertices are not necessarily distinct.

**Claim 1:** $v_0 = v_n$. 

122
Suppose otherwise. Then each time the vertex $v_n$ appears on $T$ two degrees are “used up”, except at the end of $T$. Since $\text{deg} (v_n)$ is even, there is an edge incident to $v_n$ that is not on $T$. However if we add this edge to $T$ we get a longer trail, a contradiction. Thus $v_0 = v_n$.

**Claim 2**: All edges of $G$ are on $T$.

Otherwise, since $G$ is connected, there must be an edge of the form $e_1$ or $e_2$ in $G$ as depicted below.

However now we again have a longer trail, a contradiction. □

**Note**: The book’s proof is algorithmic. It’s more complicated but it actually produces an eulerian circuit.

**Note**: It is easy (fast) using a computer to determine if a graph is eulerian. The reason is that there are fast (polynomial time) algorithms to determine if a graph is connected. It is also simple to check whether the degree of every vertex is even.
Definition: A **hamilton** cycle in a graph \( G \) is a cycle that includes all the vertices of \( G \). A graph is **hamiltonian** if it has a hamilton cycle.

Example:

\( H \) is hamiltonian, but \( G \) is not.

Note: It is *apparently* very difficult to write an algorithm that will quickly (in polynomial time) determine if a graph has a hamilton cycle. The problem falls into a large class of problems known as “NP-complete” problems. Until now

- No one has been able to find such a “fast” algorithm.
- No one can prove that such an algorithm does not exist.
- If such an algorithm can be found, it can also be used to “quickly” solve all other NP-complete problems.
O-Notation

In computer science, O-Notation (big-oh Notation) is often used to describe the running time of algorithms. First one needs to measure the size of the input data. This varies from problem to problem. Suppose the size of the input is \( n \). A fast algorithm is one that runs in “polynomial time”, i.e., the number of operations to run the program is a polynomial function of \( n \). An algorithm that requires an exponential running time is not very efficient.

See table.

Note the big difference between \( n^2 \) and \( 2^n \) and between \( n^2 \) and \( n \ln n \). By comparison, there is only a tiny difference between \( n^2 \) and \( 10n^2 \). This motivates the following definition.

\textit{Definition}: Let \( f(x) \) and \( g(x) \) be real-valued functions defined on the same set of real numbers. Then \( f \) is of order \( g \), written \( f(x) \) is \( O(g(x)) \), if, and only if, there exists a positive real number \( M \) and a real number \( x_0 \) such that

\[ |f(x)| \leq M \cdot |g(x)|, \quad \text{whenever } x > x_0. \]
The next theorem follows directly from the definition.

**Theorem 8.7** If \( r < s \), then \( x^r \) is \( O(x^s) \).

In general we have the following.

**Theorem 8.8** If \( a_0, a_1, a_2, \ldots, a_n \) are real numbers with \( a_n \neq 0 \), then

1. \( a_n x^n + a_{n-1} x^{n-1} + a_{n-2} x^{n-2} + \cdots + a_1 x + a_0 \) is \( O(x^m) \) \( \forall \ m \geq n \), and

2. \( a_n x^n + a_{n-1} x^{n-1} + a_{n-2} x^{n-2} + \cdots + a_1 x + a_0 \) is not \( O(x^m) \) \( \forall \ m < n \).

**Example:** \( f(x) = x^5 - 3x^4 + 12x^2 - 9 \) is \( O(x^5) \). This follows since for large positive \( x \),

\[
|x^5 - 3x^4 + 12x^2 - 9| \leq x^5 + 3x^4 + 12x^2 + 9 \\
\leq x^5 + 3x^5 + 12x^5 + 9x^5 \\
\leq 25x^5 \\
\leq 25|x^5|.
\]

Of course \( f(x) \) is also \( O(x^{1000}) \), but we usually try to find the “best” big-oh approximation.

**Example:**

\[
f(x) = \frac{x(x-1)}{2} \text{ is } O(x^2).
\]

However \( f(x) \) is not \( O(x) \). Clearly there does not exist a constant \( M > 0 \) and an integer \( x_0 \) such that

\[
\frac{x(x-1)}{2} \leq M \cdot x \ \forall \ x \geq x_0.
\]
Note: In computer science it is common to use big-oh notation in reference to functions defined on the integers. Integer valued functions are often used to count the number of steps required to execute an algorithm.

Note: When we say an algorithm is $O(f(n))$ we mean that in the “worst case” the algorithm will run in no more than $M|f(n)|$ steps.

Some examples.

- Most sorting algorithms (based on comparisons) are $O(n \ln n)$.
- Fast Fourier Transform (Cooley - Tukey 1965) is $O(n \ln n)$. Previously the best algorithms were $O(n^2)$.
- Linear Programming.

ex: Maximize $a_1x + a_2y$ subject to

\[
\begin{align*}
    b_{11}x + b_{12}y & \leq 7 \\
    b_{12}x + b_{22}y & \leq 5 \\
    x, y & \geq 0.
\end{align*}
\]

The Simplex Method (Danzig - 1948) is $O(2^n)$, however on average it is $O(n)$. More recent algorithms (Khachian - 1980, Karmarkar - 1984) have shown the problem to be polynomial ($O(n^6)$). However the average time complexity of Karmarkar’s algorithm is comparable to the Simplex Method.