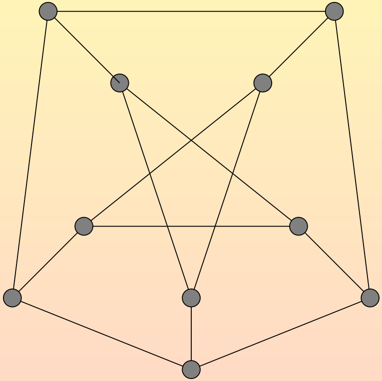
Algorithms: Graphs

Amotz Bar-Noy

CUNY

Spring 2012

Graphs



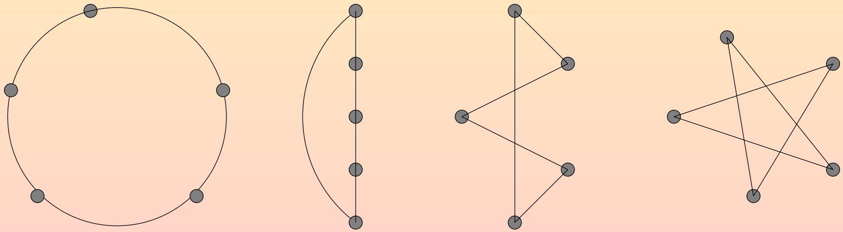
- **Definition:** A **graph** is a collection of **edges** and **vertices**. Each edge connects two vertices.

Graphs

- **Vertices:** Nodes, points, computers, users, items, ...
- **Edges:** Arcs, links, lines, cables, ...
- **Applications:** Communication, Transportation, Databases, Electronic Circuits, ...
- **An alternative definition:** A **graph** is a collection of subsets of size 2 from the set $\{1, \ldots, n\}$. A **hyper-graph** is a collection of subsets of any size from the set $\{1, \ldots, n\}$.

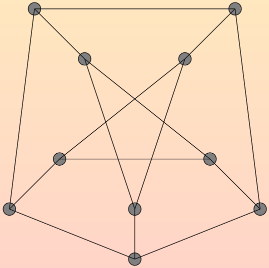
Drawing Graphs

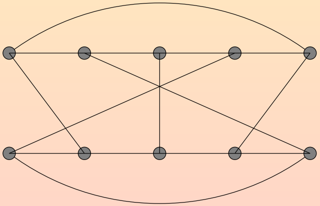
- Four possible drawings illustrating the same graph:



Drawing Graphs

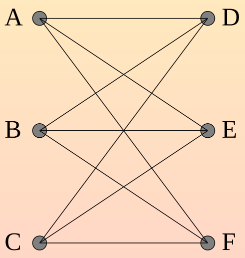
- Two drawings representing the same graph:

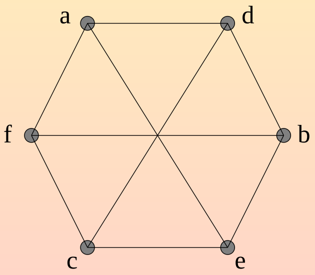




Graph Isomorphism

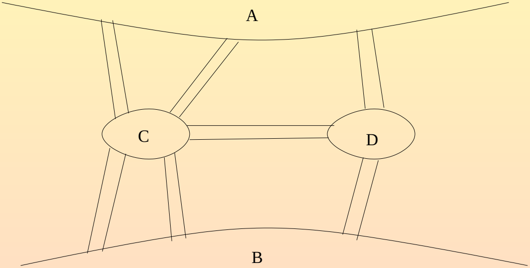
- Graph G_1 and graph G_2 are **isomorphic** if there is a **one-one** correspondence between their vertices such that the number of edges joining any two vertices of G_1 is equal to the number of edges joining the corresponding vertices of G_2.





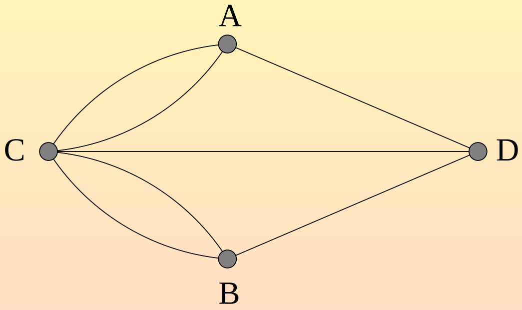
$$a \leftrightarrow A \quad b \leftrightarrow B \quad c \leftrightarrow C \quad d \leftrightarrow D \quad e \leftrightarrow E \quad f \leftrightarrow F$$

The Bridges of Königsberg



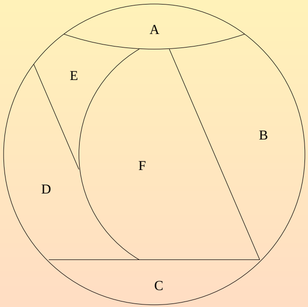
- Is it possible to traverse each of the 7 bridges of this town exactly once, starting and ending at any point?

The Bridges of Königsberg



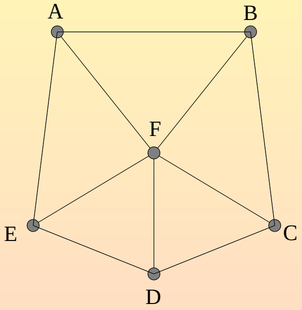
- Is it possible to traverse each of the edges of this graph exactly once, starting and ending at any vertex?
- Does a graph have an **Euler tour**?

The Four Coloring Problem



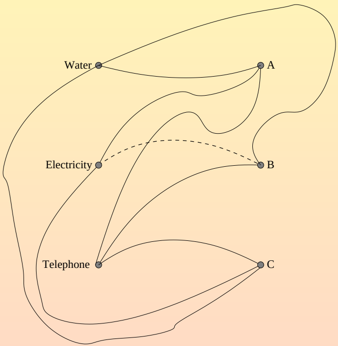
- Is it possible to color a map with at most 4 colors such that neighboring countries get different colors?

The Four Coloring Problem



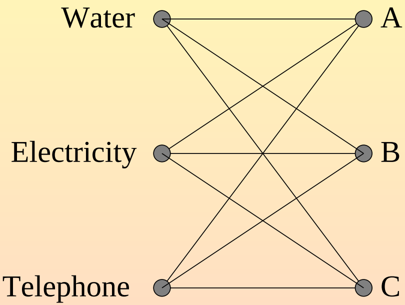
- Is it possible to color the vertices of this graph with at most 4 colors?
- Is it possible to color every **planar graph** with at most 4 colors?

The Three Utilities Problem



- Is it possible to connect the houses $\{A, B, C\}$ with the utilities {Water, Electricity,Telephone} such that cables do not cross?

The Three Utilities Problem



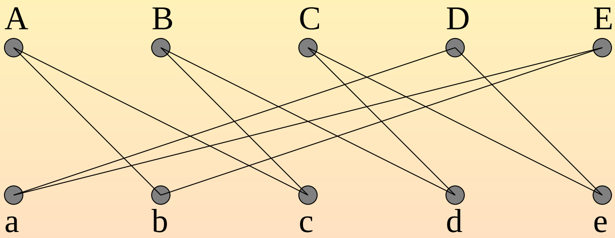
- Is it possible to draw the vertices and edges of this graph such that edges do not cross?
- Which graphs are **planar**?

The Marriage Problem

Anna loves:	Bob and Charlie
Betsy loves:	Charlie and David
Claudia loves:	David and Edward
Donna loves:	Edward and Albert
Elizabeth loves:	Albert and Bob

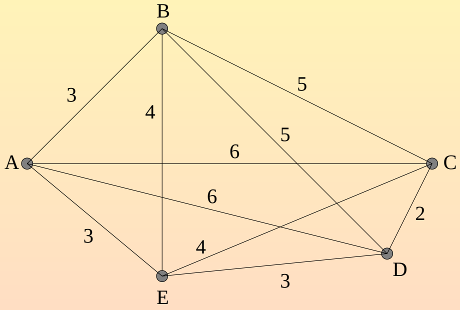
- Under what conditions a collection of girls, each loves several boys, can be married so that each girl marries a boy she loves?

The Marriage Problem



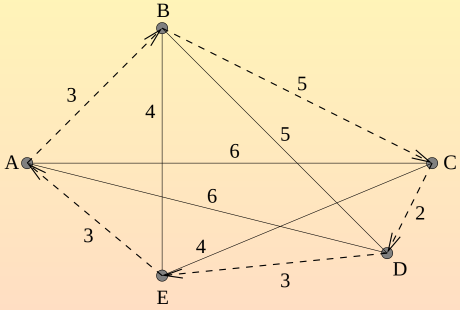
- Find in this graph a set of disjoint edges that cover all the vertices in the top side.
- Does a (bipartite) graph have a **perfect matching**?

The Travelling Salesperson Problem



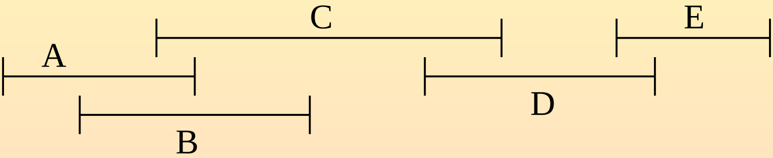
- A salesperson wants to sell products in the above 5 cities $\{A, B, C, D, E\}$ starting at A and ending at A while travelling as little as possible.

The Travelling Salesperson Problem



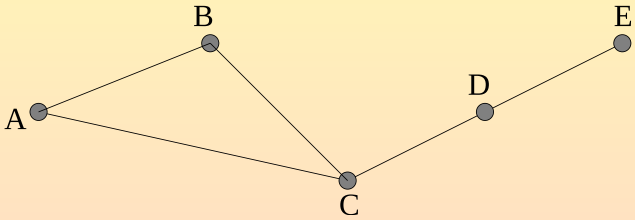
- Find the shortest path in this graph that visits each vertex at least once and starts and ends at vertex A.
- Find the shortest **Hamiltonian cycle** in a graph.

The Activity Center Problem



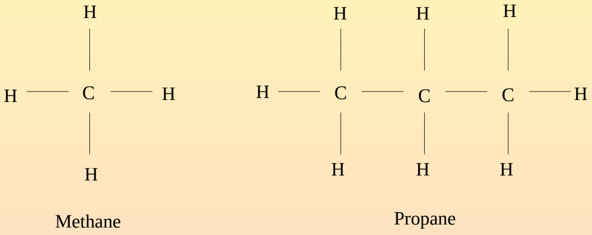
- What is the maximal number of activities that can be served by a single server?

The Activity Center Problem



- What is the maximal number of vertices in this graph with no edge between any two of them?
- Find a maximum **independent set** in a graph.

Chemical Molecules



- In the $C_x H_y$ molecule, y hydrogen atoms are connected to x carbon atoms. A hydrogen atom can be connected to exactly one carbon atom. A carbon atom can be connected to four other atoms either hydrogen or carbon.

Chemical Molecules

- How many possible structures exist for the molecule C_4H_{10}?
- How many **non-isomorphic** connected graphs exist with x vertices of degree 4 and y vertices of degree 1?
- Is there a (connected) graph whose **degree sequence** is $d_1 \geq \cdots \geq d_n$? How many non-isomorphic such graphs exist?

Some Notations

- $G = (V, E)$ - a graph G.
- $V = \{1, \ldots, n\}$ - a set of vertices.
- $E \subseteq V \times V$ - a set of edges.
- $e = (u, v) \in E$ - an edge.
- $|V| = V = n$ - number of vertices.
- $|E| = E = m$ - number of edges.

Directed and Undirected Graphs

- In **undirected graphs**: $(u, v) = (v, u)$.
- In **directed graphs (D-graphs)**: $(u \to v) \neq (v \to u)$.
- The **underlying** undirected graph $G' = (V', E')$ of a directed graph $G = (V, E)$:
  - Has the same set of vertices: $V = V'$.
  - Has all the edges of G without their direction: $(u \to v)$ becomes (u, v).

Undirected Edges

- Vertices u and v are the **endpoints** of the edge (u, v).
- Edge (u, v) is **incident** with vertices u and v.
- Vertices u and v are **neighbors** if edge (u, v) exists.
  - u is **adjacent** to v and v is **adjacent** to u.
- Vertex u has **degree** d if it has d neighbors.
- Edge (v, v) is a **(self) loop** edge.
- Edges $e_1 = (u, v)$ and $e_2 = (u, v)$ are **parallel** edges.

Directed Edges

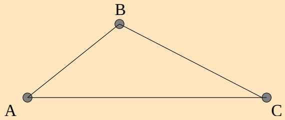
- Vertex u is the **origin** (**initial**) and vertex v is the **destination** (**terminal**) of the directed edge $(u \to v)$.
- Vertex v is the **neighbor** of vertex u if the directed edge $(u \to v)$ exists.
  - v is **adjacent** to u (but u is not adjacent to v).
- Vertex u has
  - **out-degree** d if it has d neighbors.
  - **in-degree** d if it is the neighbor of d vertices.

Weighted Graphs

- In **Weighted graphs** there exists a weight function: $w : E \rightarrow \Re$.
  - w : weight, distance, length, time, cost, capacity, ...
  - Weights could be negative.

Weighted Graphs

- In **Weighted graphs** there exists a weight function: $w : E \to \Re$.
  - w : weight, distance, length, time, cost, capacity, ...
  - Weights could be negative.



$$w(AC) \leq w(AB) + w(BC)$$

- Sometimes weights obey the **triangle inequality**. E.g., Distances in the plane.

Simple Graphs

- A **simple** directed or undirected graph is a graph with no parallel edges and no self loops.
- In a simple directed graph both edges: $(u \to v)$ and $(v \to u)$ could exist (they are not parallel edges).

Simple Graphs

- A **simple** directed or undirected graph is a graph with no parallel edges and no self loops.
- In a simple directed graph both edges: $(u \rightarrow v)$ and $(v \rightarrow u)$ could exist (they are not parallel edges).

- Number of Edges in Simple Graphs:
  - A simple undirected graph has at most $m = \binom{n}{2}$ edges.
  - A simple directed graph has at most $m = n(n-1)$ edges.
  - A **dense** simple (directed or undirected) graph has “many” edges: $m = \Theta(n^2)$.
  - A **sparse** (**shallow**) simple (directed or undirected) graph has “few” edges: $m = \Theta(n)$.

Labelled and Unlabelled Graphs

- In a **labelled** graph each vertex has a unique label (ID).
  - Usually the labels are: $1, \ldots, n$.

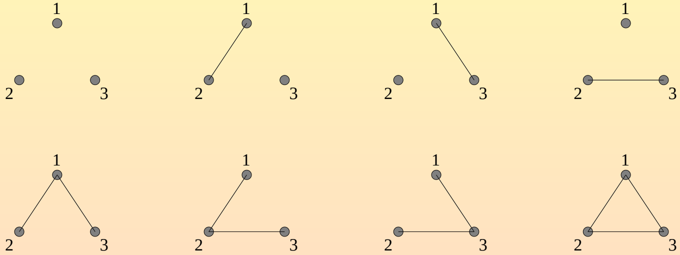
Labelled and Unlabelled Graphs

- In a **labelled** graph each vertex has a unique label (ID).
  - Usually the labels are: $1, \ldots, n$.

- **Observation:** There are $2^{\binom{n}{2}}$ **non-isomorphic** labelled graphs with n vertices.

- **Proof:** Each possible edge exists or does not exist.

Labelled Graphs



The 8 labelled graphs with $n = 3$ vertices.

Unlabelled Graphs

The 4 unlabelled graph with $n = 3$ vertices.

Paths and Cycles

- An undirected or directed **path** $\mathcal{P} = \langle v_0, v_1, \ldots, v_k \rangle$ of length k is an ordered list of vertices such that (v_i, v_{i+1}) or $(v_i \rightarrow v_{i+1})$ exists for $0 \leq i \leq k-1$ and all the edges are different.
- An undirected or directed **cycle** $\mathcal{C} = \langle v_0, v_1, \ldots, v_{k-1}, v_0 \rangle$ of length k is an undirected or directed path that starts and ends with the same vertex.
- In a **simple path**, directed or undirected, all the vertices are different.
- In a **simple cycle**, directed or undirected, all the vertices except $v_0 = v_k$ are different.

Special Paths and Cycles

- An undirected or directed **Euler path** (**tour**) is a path that traverses all the edges.
- An undirected or directed **Euler cycle** (**circuit**) is a cycle that traverses all the edges.
- An undirected or directed **Hamiltonian path** (**tour**) is a simple path that visits all the vertices.
- An undirected or directed **Hamiltonian cycle** (**circuit**) is a simple cycle that visits all the vertices.

Connected Graphs and Strongly Connected Directed Graphs

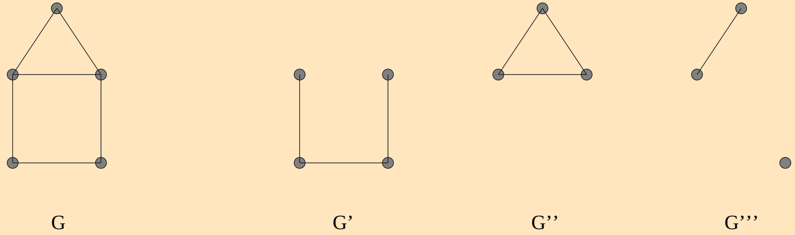
- **Connectivity:** In **connected** undirected graphs there exists a path between any pair of vertices.
- **Observation:** In a simple connected undirected graph there are at least $m = n - 1$ edges.
- **Strong connectivity:** In a **strongly connected** directed graph there exists a directed path from u to v for any pair of vertices u and v.
- **Observation:** In a simple strongly connected directed graph there are at least $m = n$ edges.

Weakly Connected Directed Graphs

- **Definition I:** In a **weakly connected** directed graph there exists a directed path either from u to v or from v to u for any pair of vertices u and v.
- **Definition II:** In a **weakly connected** directed graph there exists a path between any pair of vertices in the underlying undirected graph.
- **Observation:** The definitions are not equivalent: **Def. I** implies **Def. II** but **Def. II** does not imply **Def. I**.

Sub-Graphs

- A (directed or undirected) Graph $G' = (V', E')$ is a **sub-graph** of a (directed or undirected) graph $G = (V, E)$ if: $V' \subseteq V$ and $E' \subseteq E$.



- G', G'', G''' are sub-graphs of G

Connected Components - Undirected Graphs

- A connected sub-graph G' is a **connected component** of an undirected graph G if there is no connected sub-graph G'' of G such that G' is also a subgraph of G''.
- A connected component G' is a **maximal** sub-graph with the connectivity property.
- A connected graph has exactly one connected component.

Connected Components - Directed Graphs

- A strongly connected directed sub-graph G' is a **strongly connected component** of a directed graph G if there is no strongly connected directed sub-graph G'' of G such that G' is also a subgraph of G''.
- A strongly connected component G' is a **maximal** sub-graph with the strong connectivity property.
- A strongly connected graph has exactly one strongly connected component.

Counting Edges

- **Theorem:** Let G be a simple undirected graph with n vertices and k connected components then:
$$n - k \leq m \leq \frac{(n-k)(n-k+1)}{2}\,.$$

- **Corollary:** A simple undirected graph with n vertices is connected if it has m edges for:
$$m > \frac{(n-1)(n-2)}{2}$$

Assumptions

- Unless stated otherwise, **usually** a graph is:
  - Simple.
  - Undirected.
  - Connected.
  - Unweighted.
  - Unlabelled.

Forests and Trees

- **Forest:** A graph with no cycles.
- **Tree:** A connected graph with no cycles.

Forests and Trees

- **Forest:** A graph with no cycles.
- **Tree:** A connected graph with no cycles.

- **By definition:**
  - A tree is a connected forest.
  - Each connected component of a forest is a tree.

Trees

- **Theorem:** An undirected and simple graph is a tree if:
  - It is connected and has no cycles.
  - It is connected and has exactly $m = n - 1$ edges.
  - It has no cycles and has exactly $m = n - 1$ edges.
  - It is connected and deleting any edge disconnects it.
  - Any 2 vertices are connected by exactly one path.
  - It has no cycles and any new edge forms one cycle.

Trees

- **Theorem:** An undirected and simple graph is a tree if:
  - It is connected and has no cycles.
  - It is connected and has exactly $m = n - 1$ edges.
  - It has no cycles and has exactly $m = n - 1$ edges.
  - It is connected and deleting any edge disconnects it.
  - Any 2 vertices are connected by exactly one path.
  - It has no cycles and any new edge forms one cycle.

- **Corollary:** The number of edges in a forest with n vertices and k trees is $m = n - k$.

Rooted and Ordered Trees

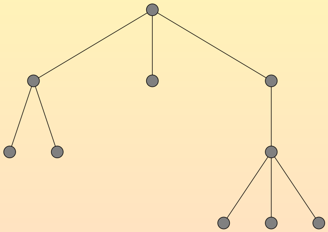
- **Rooted trees:**
  - One vertex is designated as the **root**.
  - Vertices with degree 1 are called **leaves**.
  - Non-leaves vertices are **internal** vertices.
  - All the edges are directed from the root to the leaves.

Rooted and Ordered Trees

- **Rooted trees:**
  - One vertex is designated as the **root**.
  - Vertices with degree 1 are called **leaves**.
  - Non-leaves vertices are **internal** vertices.
  - All the edges are directed from the root to the leaves.

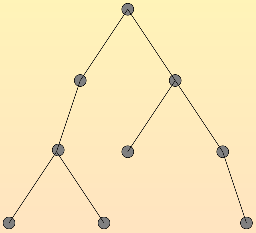
- **Ordered trees:**
  - **Children** of an internal **parent** vertex are ordered.

Drawing Rooted Trees



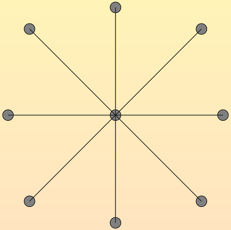
- Parents **above** children.
- Older children to the **left** of younger children.

Binary Trees



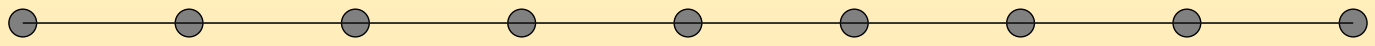
- **Binary trees:** The **root** has degree either 1 or 2, the **leaves** have degree 1, and the degree of **non-root internal** vertices is either 2 or 3.

Star Trees



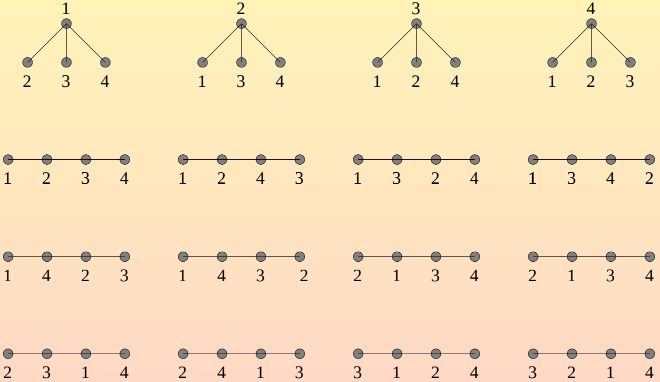
- **Star:** A rooted tree with 1 root and $n - 1$ leaves. The degree of one vertex (the root) is $n - 1$ and the degree of any non-root vertex is 1.

Path Trees



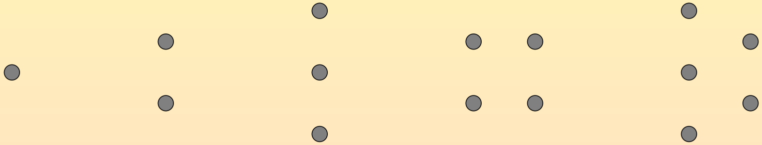
- **Path:** A tree with exactly 2 leaves.
- **Claim I:** The degree of a non-leave vertex is exactly 2.
- **Claim II:** The path is the only tree with exactly 2 leaves.

Counting Labelled Trees



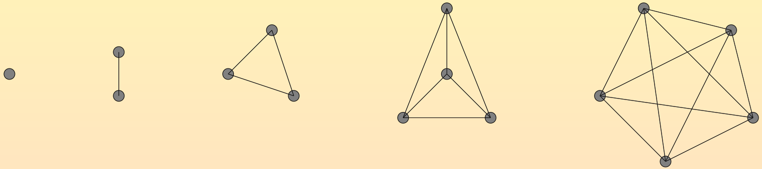
- **Theorem:** There are n^{n-2} distinct labelled n vertices trees.

Null Graphs



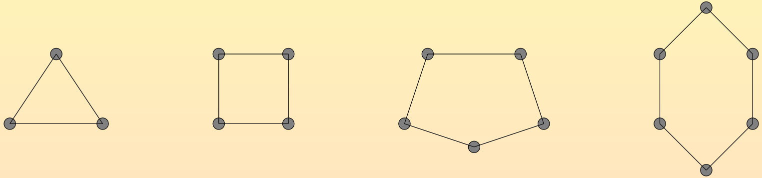
- **Null graphs** are graphs with no edges.
- The null graph with n vertices is denoted by N_n.
- In null graphs $m = 0$.

Complete Graphs



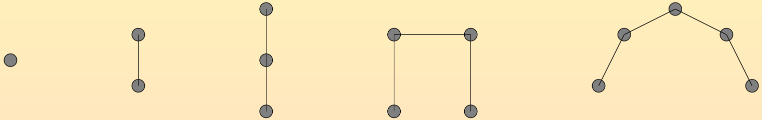
- **Complete graphs** (**cliques**) are graphs with all possible edges.
- The complete graph with n vertices is denoted by K_n.
- In complete graphs $m = \binom{n}{2} = \frac{n(n-1)}{2}$.

Cycles



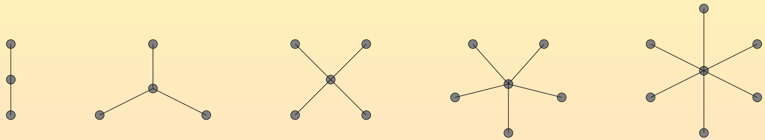
- **Cycles** (**rings**) are connected graphs in which all vertices have degree 2 ($n \geq 3$).
- The cycle with n vertices is denoted by C_n.
- In cycles $m = n$.

Paths



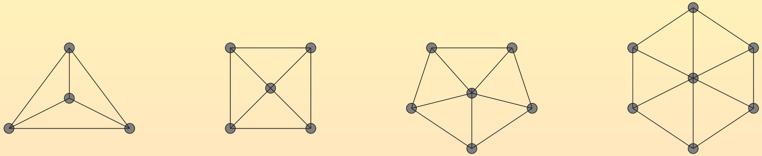
- **Paths** are cycles with one edge removed.
- The path with n vertices is denoted by P_n.
- In paths $m = n - 1$.

Stars



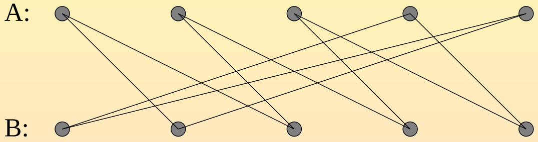
- **Stars** are graphs with one root and $n-1$ leaves.
- The star with n vertices is denoted by S_n.
- In stars $m = n - 1$.

Wheels



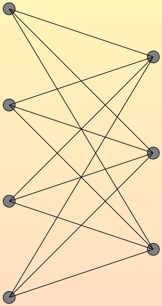
- **Wheels** are stars in which all the $n-1$ leaves form a cycle C_{n-1} ($n \geq 4$).
- The wheel with n vertices is denoted by W_n.
- In wheels $m = 2n - 2$.

Bipartite Graphs



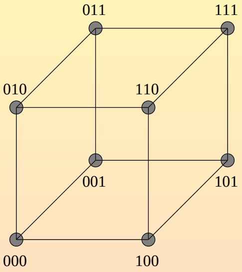
- **Bipartite graphs** $V = A \cup B$**:** each edge is incident to one vertex from A and one vertex from B.
- **Observation:** A graph is bipartite iff each cycle is of even length.

Complete Bipartite Graphs



- **Complete bipartite graphs $K_{r,c}$:** All possible $r \cdot c$ edges exist.

Cubes

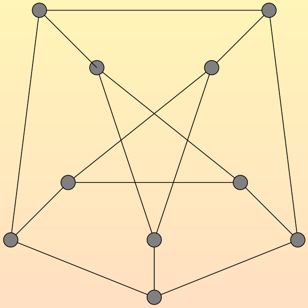


- There are $n = 2^k$ vertices representing all the 2^k binary sequences of length k.
- Two vertices are connected by an edge if their corresponding sequences differ by exactly one bit.

Cubes

- **Observation:** Cubes are bipartite graphs.
- **Proof:**
  - A: The vertices with even number of 1 in their binary representation.
  - B The vertices with odd number of 1 in their binary representation.
  - Any edge connects 2 vertices one from the set A and one from the set B.

d-regular Graphs

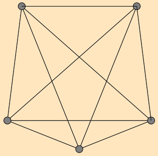


- In ***d*-regular graphs**, the degree of each vertex is exactly d.
- In ***d*-regular graphs**, $m = \frac{d \cdot n}{2}$.
- The Petersen Graph: a 3-regular graph.

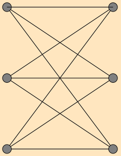
Planar Graphs

- **Definition: Planar graphs** are graphs that can be drawn on the plane such that edges do not cross each other.
- **Theorem:** A graph is planar iff it does not have sub-graphs **homeomorphic** to K_5 and $K_{3,3}$.
- **Theorem:** Every planar graph can be drawn with straight lines.

Non-Planar Graphs



K_5: the complete graph with 5 vertices.



$K_{3,3}$: the complete $\langle 3, 3 \rangle$ bipartite graph.

Platonic Graphs

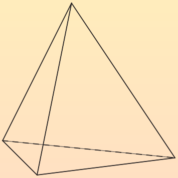
- Graphs that are formed from the vertices and edges of the five regular (**Platonic**) solids:
  - **Tetrahedron:** 4 vertices 3-regular graph.
  - **Octahedron:** 6 vertices 4-regular graph.
  - **Cube:** 8 vertices 3-regular graph.
  - **Icosahedron:** 12 vertices 5-regular graph.
  - **Dodecahedron:** 20 vertices 3-regular graph.

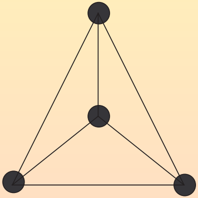
Platonic Graphs

- Graphs that are formed from the vertices and edges of the five regular (**Platonic**) solids:
  - **Tetrahedron:** 4 vertices 3-regular graph.
  - **Octahedron:** 6 vertices 4-regular graph.
  - **Cube:** 8 vertices 3-regular graph.
  - **Icosahedron:** 12 vertices 5-regular graph.
  - **Dodecahedron:** 20 vertices 3-regular graph.

- **Observation:** The platonic graphs are d-regular planar graphs.

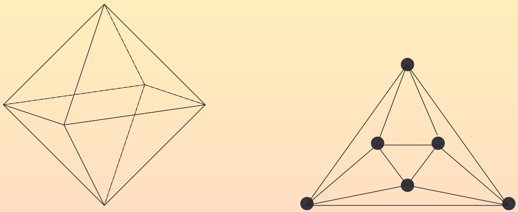
The Tetrahedron





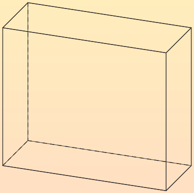
4 vertices; 6 edges; 4 faces; degree 3

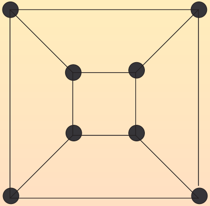
The Octahedron



6 vertices; 12 edges; 8 faces; degree 4

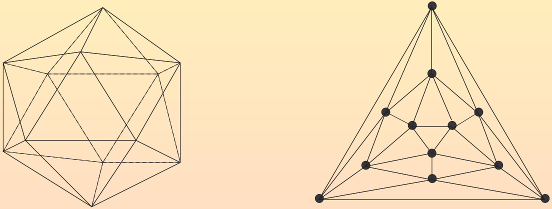
The Cube





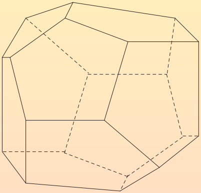
8 vertices; 12 edges; 6 faces; degree 3

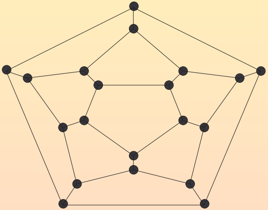
The Icosahedron



12 vertices; 30 edges; 20 faces; degree 5

The Dodecahedron





20 vertices; 30 edges; 12 faces; degree 3

Dual Planar Graphs

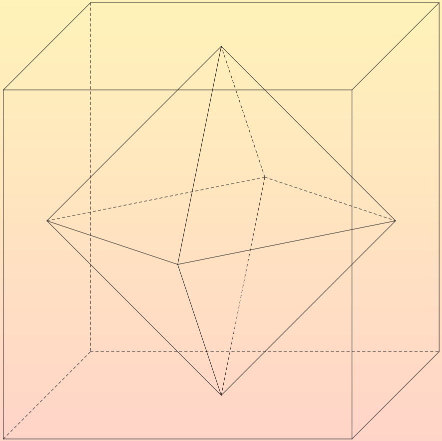
- In the **dual planar graph** G^* of a planar graph G, vertices correspond to faces of G and two vertices in G^* are joined by an edge if the corresponding faces in G share an edge.

Dual Planar Graphs

- In the **dual planar graph** G^* of a planar graph G, vertices correspond to faces of G and two vertices in G^* are joined by an edge if the corresponding faces in G share an edge.

- The **Octahedron** is the dual graph of the **Cube**.
- The **Cube** is the dual graph of the **Octahedron**.
- The **Icosahedron** is the the dual graph of the Dodecahedron.
- The **Dodecahedron** is the the dual graph of the **Icosahedron**.
- The **Tetrahedron** is the dual graph of itself.

Duality of the Cube and the Octahedron



Random Graphs

- **Definition I:**
  - Each edge exists with probability $0 \le p \le 1$.
  - **Observation:** Expected number of edges is $E(m) = p\binom{n}{2}$.

Random Graphs

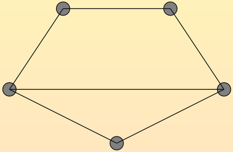
- **Definition I:**
  - Each edge exists with probability $0 \leq p \leq 1$.
  - **Observation:** Expected number of edges is $E(m) = p\binom{n}{2}$.

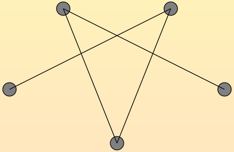
- **Definition II:**
  - A graph with m edges that is selected randomly with a uniform distribution over all graphs with m edges.

Interval Graphs

- Vertices represent **intervals** on the *x*-axis.
- An edge indicates that two intervals **intersect**.

Complement Graphs





- $\tilde{G} = (\tilde{V}, \tilde{E})$ is the **complement graph** of $G = (V, E)$ if $V = \tilde{V}$ and $(x, y) \in E \leftrightarrow (x, y) \notin \tilde{E}$.
- A graph G is **self-complementary** if it is isomorphic to $\tilde{G}$.

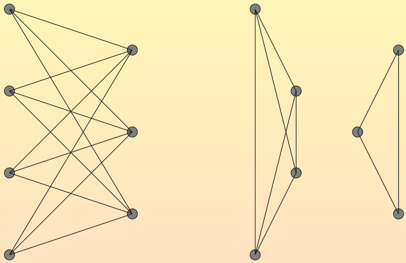
Complement Graphs

- **Lemma:** At least one of G and $\tilde{G}$ is connected.
- **Proof:**
  - Assume G is not connected.
  - The set of vertices V can be partitioned into 2 non-empty sets of vertices A and B such that all the edges between A and B are in $\tilde{G}$.
  - A complete bipartite graph is connected and therefore $\tilde{G}$ is connected.

Complement Graphs – Observation

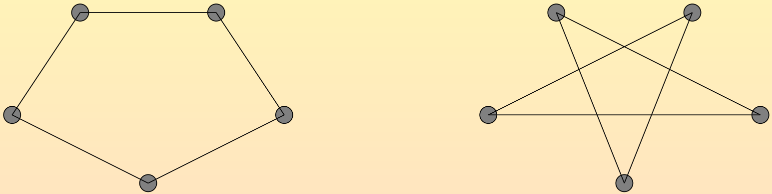
$$N_n = \tilde{K}_n.$$

Complement Graphs – Observation



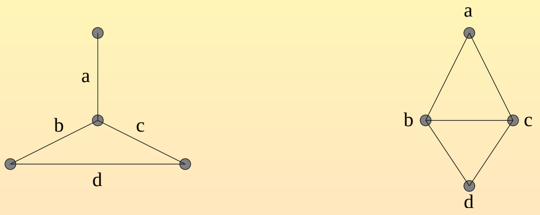
$$\tilde{K_{r,s}} = K_r \cup K_s.$$

Complement Graphs – Observation



$$C_5 = \tilde{C}_5.$$

Line Graphs

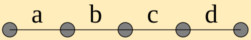


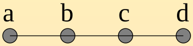
- In the **line graph** $L(G) = (E, F)$ of $G = (V, E)$ vertices correspond to edges of G and two vertices in $L(G)$ are joined by an edge if the corresponding edges in G share a vertex.
- **Definition:** $(e_i, e_j) \in F$ iff $e_i = (x, y)$ and $e_j = (y, z)$ for $x, y, z \in V$.
- **Observation:** $L(L(G)) = G$ is a **wrong** statement.

Line Graphs – Observation

$$L(C_n) = C_n.$$

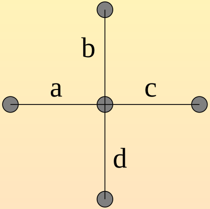
Line Graphs – Observation

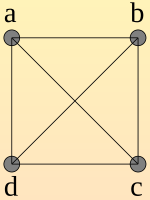




$$L(P_n) = P_{n-1}.$$

Line Graphs – Observation





$$L(S_n) = K_{n-1}.$$

Social Graphs

- **Definition:** The **social graph** contains all the **friendship** relations (edges) among n **people** (vertices).
- **I:** In any group of $n \geq 2$ people, there are 2 people with the same number of friends in the group.
- **II:** There exists a group of 5 people for which no 3 are mutual friends and no 3 are mutual strangers.
- **III:** Every group of 6 people contains either three mutual friends or three mutual strangers.

Data structure for Graphs

- **Adjacency lists:** $\Theta(m)$ memory.
- **An adjacency Matrix:** $\Theta(n^2)$ memory.
- **An incident matrix:** $\Theta(n \cdot m)$ memory.

The Adjacency Lists Representation

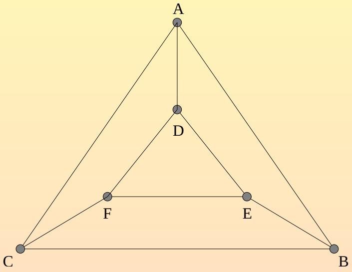
- Each vertex is associated with a linked list consisting of all of its neighbors.
- In a directed graph there are 2 lists: an incoming list and an outgoing list.
- In a weighted graph each record in the list has an additional field for the weight.

The Adjacency Lists Representation

- Each vertex is associated with a linked list consisting of all of its neighbors.
- In a directed graph there are 2 lists: an incoming list and an outgoing list.
- In a weighted graph each record in the list has an additional field for the weight.

- **Memory:** $\Theta(n+m)$.
  - Undirected graphs: $\sum_v Deg(v) = 2m$
  - Directed graphs: $\sum_v OutDeg(v) = \sum_v InDeg(v) = m$

Example – Adjacency Lists



$$\begin{array}{ccc} A & \rightarrow & (B, C, D) \\ B & \rightarrow & (A, C, E) \\ C & \rightarrow & (A, B, F) \\ D & \rightarrow & (A, E, F) \\ E & \rightarrow & (B, D, F) \\ F & \rightarrow & (C, D, E) \end{array}$$

The Adjacency Matrix Representation

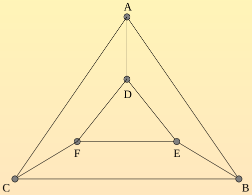
- A matrix A of size $n \times n$:
  - $A[u, v] = 1$ if (u, v) or $(u \to v)$ is an edge.
  - $A[u, v] = 0$ if (u, v) or $(u \to v)$ is not an edge.
- In simple graphs: $A[u, u] = 0$
- In undirected graphs: $A[u, v] = A[v, u]$
- In weighted graphs: $A[u, v] = w(u, v)$

The Adjacency Matrix Representation

- A matrix A of size $n \times n$:
  - $A[u, v] = 1$ if (u, v) or $(u \to v)$ is an edge.
  - $A[u, v] = 0$ if (u, v) or $(u \to v)$ is not an edge.
- In simple graphs: $A[u, u] = 0$
- In undirected graphs: $A[u, v] = A[v, u]$
- In weighted graphs: $A[u, v] = w(u, v)$

- **Memory:** $\Theta(n^2)$.
  - Independent of m that could be much smaller than $\Theta(n^2)$.

Example – Adjacency Matrix



	A	B	C	D	E	F
A	0	1	1	1	0	0
B	1	0	1	0	1	0
C	1	1	0	0	0	1
D	1	0	0	0	1	1
E	0	1	0	1	0	1
F	0	0	1	1	1	0

The Incident Matrix Representation

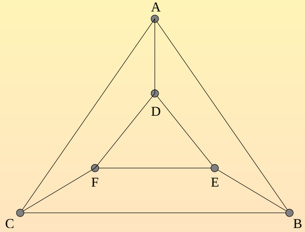
- A matrix A of size $n \times m$:
  - $A[v, e] = 1$ if undirected edge e is incident with v.
  - $A[u, e] = -1$ and $A[v, e] = 1$ for a directed edge $u \to v$.
  - Otherwise $A[v, e] = 0$.
- In simple graphs all the columns are different and each contains exactly 2 non-zero entries.
- In weighted undirected graphs: $A[v, e] = w(e)$ if edge e is incident with vertex v.

The Incident Matrix Representation

- A matrix A of size $n \times m$:
  - $A[v, e] = 1$ if undirected edge e is incident with v.
  - $A[u, e] = -1$ and $A[v, e] = 1$ for a directed edge $u \to v$.
  - Otherwise $A[v, e] = 0$.
- In simple graphs all the columns are different and each contains exactly 2 non-zero entries.
- In weighted undirected graphs: $A[v, e] = w(e)$ if edge e is incident with vertex v.

- **Memory:** $\Theta(n \cdot m)$.

Example – Incident Matrix



	(A,B)	(A,C)	(A,D)	(B,C)	(B,E)	(C,F)	(D,E)	(D,F)	(E,F)
A	1	1	1	0	0	0	0	0	0
B	1	0	0	1	1	0	0	0	0
C	0	1	0	1	0	1	0	0	0
D	0	0	1	0	0	0	1	1	0
E	0	0	0	0	1	0	1	0	1
F	0	0	0	0	0	1	0	1	1

Which Data Structure to Choose?

- Adjacency matrices are simpler to implement and maintain.
- Adjacency matrices are better for dense graphs.
- Adjacency lists are better for sparse graphs.
- Adjacency lists are better for algorithms whose complexity depends on m.
- Incident matrices are not efficient for algorithms.

Graphic Sequences

- The **degree** d_x of vertex x in graph G is the number of neighbors of x in G.
- **The hand-shaking Lemma:** $\sum_{i=1}^{n} d_i = 2m$.
- **Corollary:** Number of odd degree vertices is even.
- The **degree sequence** of G is $S = (d_1, \ldots, d_n)$.
- A sequence $S = (d_1, \ldots, d_n)$ is **graphic** if there exists a graph with n vertices whose degree sequence is S.

Non-Graphic Sequences

- $(3, 3, 3, 3, 3, 3, 3)$ is not graphic (equivalently, there is no 7-vertex 3-regular graph).
  - Since $\sum_{i=1}^{n} d_i$ is odd.

- $(5, 5, 4, 4, 0)$ is not graphic.
  - Since there are 5 vertices and therefore the maximum degree could be at most 4.

- $(3, 2, 1, 0)$ is not graphic.
  - Since there are 3 positive degree vertices and only one vertex with degree 3.

Graphic Sequences – Observations

- **I:** The sequence $(0, 0, \ldots, 0)$ of length n is graphic. Since it represents the null graph N_n.
- **II:** In a graphic sequence $S = (d_1 \geq \cdots \geq d_n)$ $d_1 \leq n - 1$.
- **III:** $d_{d_1+1} > 0$ in a graphic sequence of a non-null graph $S = (d_1 \geq \cdots \geq d_n)$.

Transformation

- Let $S = (d_1 \geq \cdots \geq d_n)$, then
  - $f(S) = (d_2 - 1 \geq \cdots \geq d_{d_1+1} - 1, d_{d_1+2} \geq \cdots \geq d_n)$.

Transformation

- Let $S = (d_1 \geq \cdots \geq d_n)$, then
  - $f(S) = (d_2 - 1 \geq \cdots \geq d_{d_1+1} - 1, d_{d_1+2} \geq \cdots \geq d_n)$.

- **Example:**
  - $S = (5, 4, 3, 3, 2, 1, 1, 1)$
  - $f(S) = (3, 2, 2, 1, 0, 1, 1)$

Lemma

- $S = (d_1 \geq \cdots \geq d_n)$ is graphic **iff** $f(S)$ is graphic.

Lemma

- $S = (d_1 \geq \cdots \geq d_n)$ is graphic **iff** $f(S)$ is graphic.
  - $\Leftarrow$ To get a graphic representation for S, add a vertex of degree d_1 to the graphic representation of $f(S)$ and connect this vertex to all vertices whose degrees in $f(S)$ are smaller by 1 than those in S.

Lemma

- $S = (d_1 \geq \cdots \geq d_n)$ is graphic **iff** $f(S)$ is graphic.
  - $\Leftarrow$ To get a graphic representation for S, add a vertex of degree d_1 to the graphic representation of $f(S)$ and connect this vertex to all vertices whose degrees in $f(S)$ are smaller by 1 than those in S.
  - $\Rightarrow$ To get a graphic representation for $f(S)$, omit a vertex of degree d_1 from the graphic representation of S. Make sure (**how?**) that this vertex is connected to the vertices whose degrees are $d_2, \ldots, d_{d_1+1}$.

Algorithm

Graphic($S = (d_1 \geq \cdots \geq d_n \geq 0)$)
  case $d_1 = 0$ **return TRUE** **(* Obs. I *)**
  case $d_1 \geq n$ **return FALSE** **(* Obs. II *)**
  case $d_{d_1+1} = 0$ **return FALSE** **(* Obs. III *)**
  otherwise return Graphic(Sort($f(S)$)) **(* Lemma *)**

Algorithm

- **Complexity:**
  - $O(m)$ for the transformations since $\sum_{i=1}^{n} d_i = 2m$.
  - $O(n^2)$ for the sorting (merging n times).

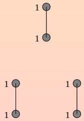
Algorithm

- **Complexity:**
  - $O(m)$ for the transformations since $\sum_{i=1}^{n} d_i = 2m$.
  - $O(n^2)$ for the sorting (merging n times).

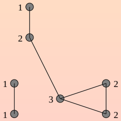
- Constructing the graph for $S = (d_1 \geq \cdots \geq d_n \geq 0)$: Follow the "$\Leftarrow$" part of the proof of the lemma starting with the sequence $(0, \ldots, 0)$ and ending with S.

Example

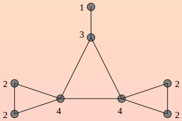
4	4	3	2	2	2	2	1	I
	3	2	1	1	2	2	1	
	3	2	2	2	1	1	1	II
		1	1	1	1	1	1	
		1	1	1	1	1	1	III
			0	1	1	1	1	
			1	1	1	1	0	IV
				0	1	1	0	
				1	1	0	0	V
					0	0	0	VI



III



II



I