Ministry Of high Education
and Scientific Research
University of Babylon
Dep of Mathematics

Operations on graphs

Qasim Ali Abd Ahmed

Supervisor

Dr. Ahmed Abed Ali

Dedication

To his fragrant biography, and enlightened thought;

He was the first credit for my attainment of higher education

(My beloved father), may God prolong his life.

To the one who put me on the path of life, and made me calm,

She nursed me until I was big

(My dear mother), may God bless her soul.

to my brothers; Those who had a great impact on many obstacles and hardships.

To all my honorable teachers; Who did not hesitate to extend a helping hand to me

Abstract :

Operations on graphs are mathematical operations that modify the structure and properties of a graph. They include adding/removing vertices/edges, finding shortest paths, computing vertex degree, and identifying connected components. These operations are critical in analyzing and designing complex systems like social, transportation, and computer networks.

Content :

1- Introduction

1-1- What is a graph

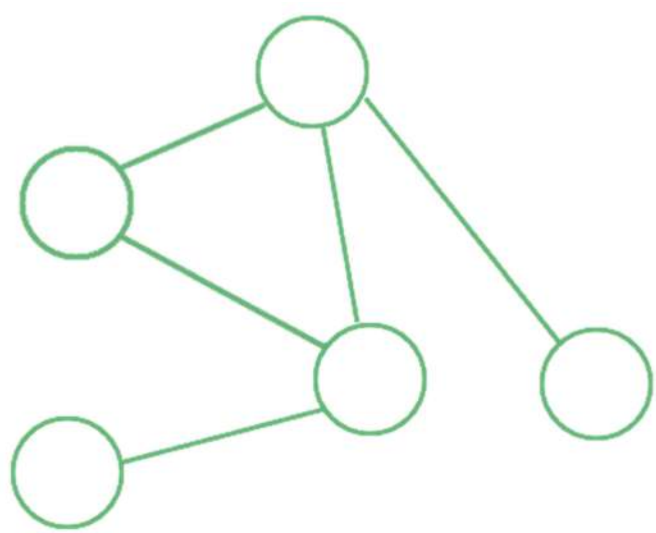
a graph is a mathematical structure that consists of a set of vertices (also known as nodes) and a set of edges that connect those vertices. The edges represent the relationships or connections between the vertices.

Graphs can be represented in many different ways, including as diagrams or as data structures in computer programs. In a diagram representation, the vertices are usually depicted as points or circles, and the edges are depicted as lines connecting those points. In a data structure representation, the vertices and edges are typically stored in arrays or linked lists.

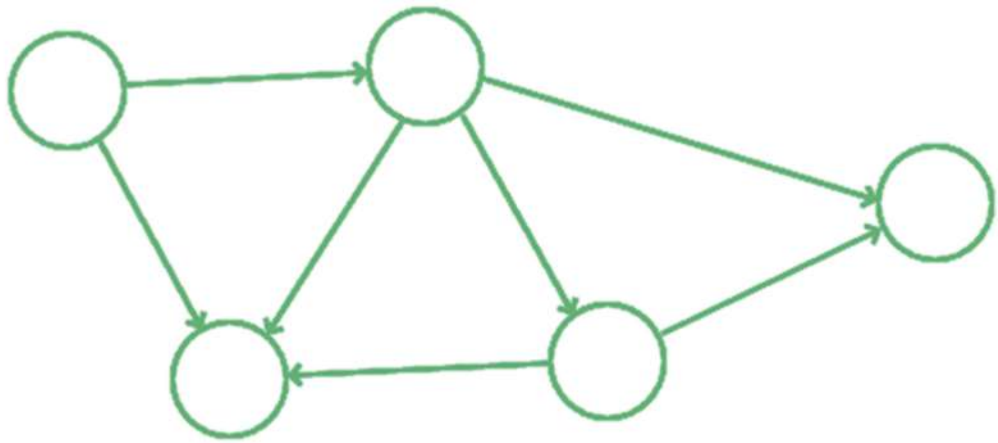
Graphs can be classified based on various properties, such as their size, connectivity, or directedness.

Some common types of graphs include:

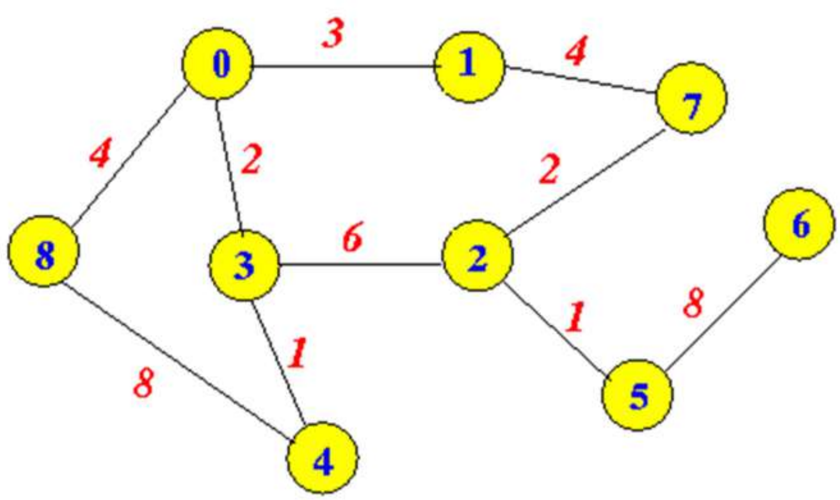
1. Undirected graphs: These are graphs where the edges do not have a direction. If there is an edge connecting vertex A to vertex B, then there is also an edge connecting vertex B to vertex A.



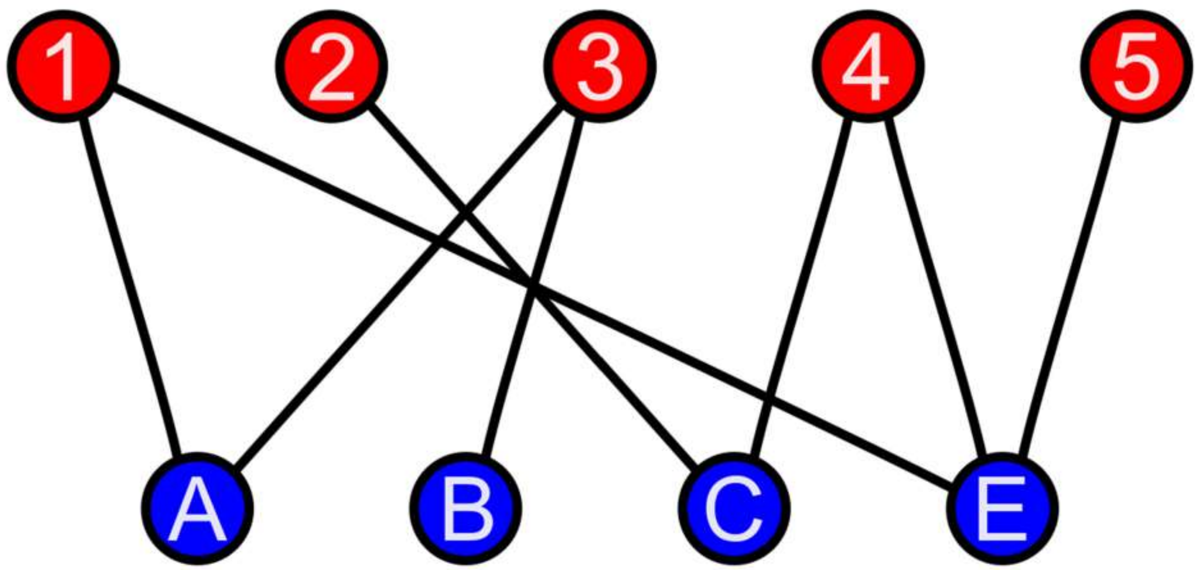
2. Directed graphs: These are graphs where the edges have a direction. If there is an edge connecting vertex A to vertex B, then there may or may not be an edge connecting vertex B to vertex A.



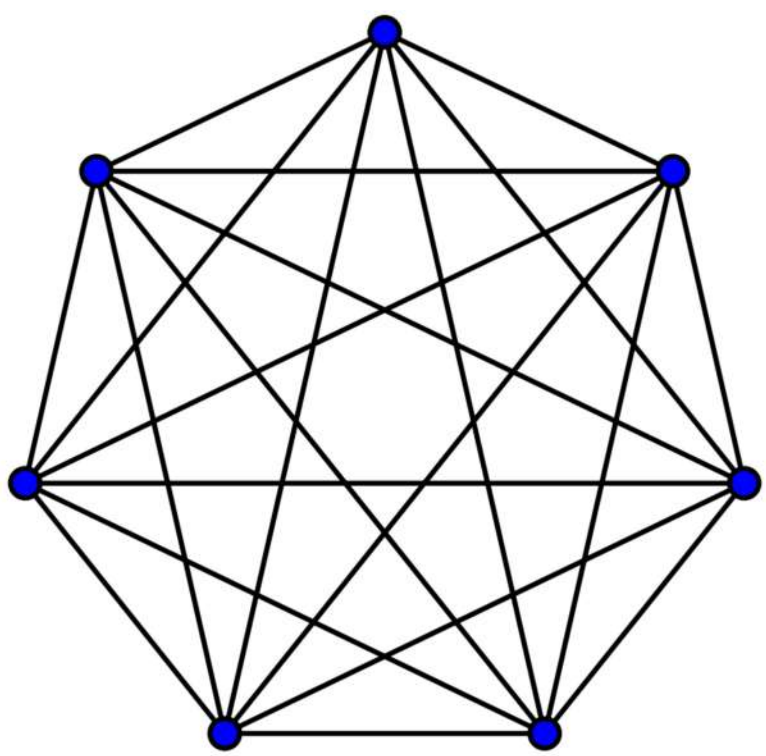
3. Weighted graphs: These are graphs where each edge is assigned a weight or value that represents some kind of cost, distance, or other attribute.



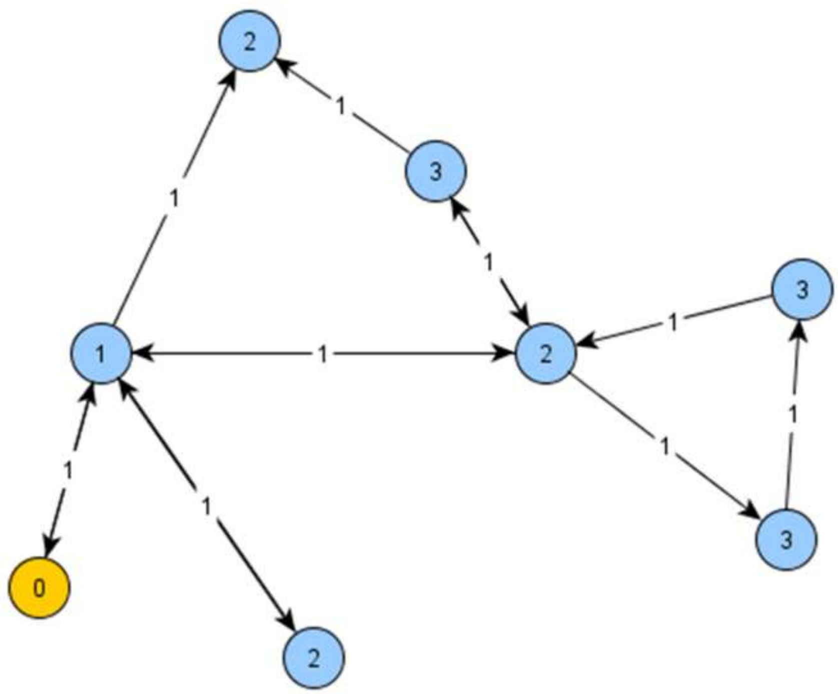
4. Bipartite graphs: These are graphs where the vertices can be divided into two groups such that every edge connects a vertex in one group to a vertex in the other group.



5. Complete graphs: These are graphs where every pair of vertices is connected by an edge.



6. Sparse graphs: These are graphs where the number of edges is much smaller than the maximum number of edges that could be present.



Graphs are used in many different fields, including computer science, mathematics, physics, and social sciences. They are a powerful tool for modeling and analyzing complex systems, networks, and relationships between objects or concepts.

1-2- Applications Of Graph

- Social media analysis: Social media platforms generate vast amounts of data in real-time, which can be analyzed using graphs to identify trends, sentiment, and key influencers. This can be useful for marketing, customer service, and reputation management.

- Network monitoring: Graphs can be used to monitor network traffic in real-time, allowing network administrators to identify potential bottlenecks, security threats, and other issues. This is critical for ensuring the smooth operation of complex networks.
- Financial trading: Graphs can be used to analyze real-time financial data, such as stock prices and market trends, to identify patterns and make trading decisions. This is particularly important for high-frequency trading, where even small delays can have a significant impact on profits.
- Internet of Things (IoT) management: IoT devices generate vast amounts of data in real-time, which can be analyzed using graphs to identify patterns, optimize performance, and detect anomalies. This is important for managing large-scale IoT deployments.
- Autonomous vehicles: Graphs can be used to model the real-time environment around autonomous vehicles, allowing them to navigate safely and efficiently. This requires real-time data from sensors and other sources, which can be processed using graph algorithms.
- Disease surveillance: Graphs can be used to model the spread of infectious diseases in real-time, allowing health officials to identify outbreaks and implement effective containment strategies. This is particularly important during pandemics or other public health emergencies.

1-3- Advantages and Benefits of Graph

Advantages of Graph:

- **Representing complex data:** Graphs are effective tools for representing complex data, especially when the relationships between the data points are not straightforward. They can help to uncover patterns, trends, and insights that may be difficult to see using other methods.
- **Efficient data processing:** Graphs can be processed efficiently using graph algorithms, which are specifically designed to work with graph data structures. This makes it possible to perform complex operations on large datasets quickly and effectively.
- **Network analysis:** Graphs are commonly used in network analysis to study relationships between individuals or organizations, as well as to identify important nodes and edges in a network. This is useful in a variety of fields, including social sciences, business, and marketing.
- **Pathfinding:** Graphs can be used to find the shortest path between two points, which is a common problem in computer science, logistics, and transportation planning.
- **Visualization**: Graphs are highly visual, making it easy to communicate complex data and relationships in a clear and concise way. This makes them useful for presentations, reports, and data analysis.
- **Machine learning**: Graphs can be used in machine learning to model complex relationships between variables, such as in recommendation systems or fraud detection.

Benefits of Graph:

1. Simplicity: Graphs provide a simple and intuitive way to represent and understand complex relationships between objects or entities.
2. Scalability: Graphs can handle large amounts of data and can be used to analyze complex systems with many interconnected components.
3. Flexibility: Graphs can be used to represent many different types of relationships and can be applied to a wide range of problems and domains.
4. Interdisciplinary: Graphs are used in many different fields, from computer science to social sciences to biology, making them a valuable tool for interdisciplinary research and collaboration.

Overall, the benefits and advantages of using graphs make them an essential tool for understanding and analyzing complex systems and relationships.

2- Operations on graphs

Operations on graphs refer to the various ways in which a graph can be manipulated or transformed. These operations can be used to analyze, compare, or modify graphs in various ways.

Some common operations on graphs include:

1. Graph Traversal: This involves visiting each vertex or edge of a graph in a systematic manner. Examples of traversal algorithms include depth-first search and breadth-first search.
2. Graph Search: This involves finding a particular vertex or edge in a graph. Examples of search algorithms include binary search and Dijkstra's algorithm.
3. Graph Connectivity: This involves determining whether a graph is connected or not, and if so, how many connected components it has. This is important in various applications such as network analysis and social network analysis.
4. Graph Coloring: This involves assigning colors to vertices of a graph such that no two adjacent vertices have the same color. This is used in various applications such as scheduling and map coloring.
5. Graph Isomorphism: This involves determining whether two graphs are structurally identical, i.e., they have the same number of vertices and edges, and the same connectivity pattern.
6. Graph Transformation: This involves modifying a graph in some way, such as adding or deleting vertices or edges, or changing their weights or labels.

These are just a few examples of the many operations that can be performed on graphs. The choice of operation depends on the specific application and the type of problem being solved.

2-1- Union

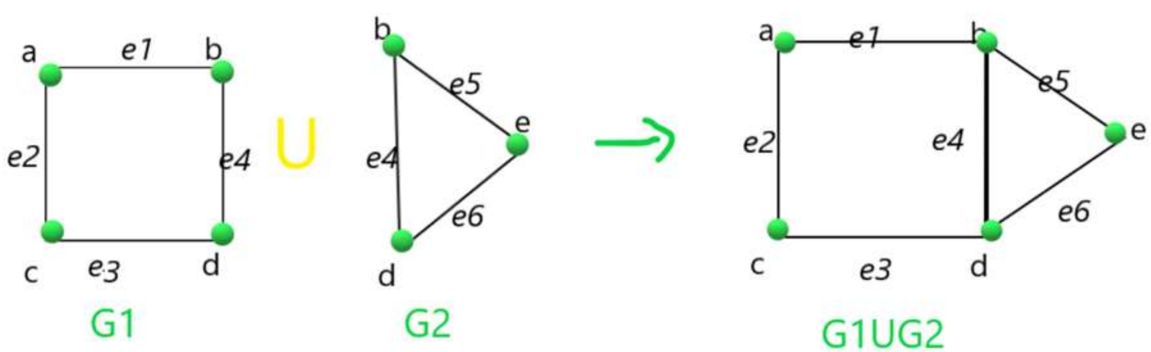
In graph theory, the union operation is used to combine two graphs into a single graph. The resulting graph contains all the vertices and edges of the original graphs.

The union of two graphs G1 = (V1, E1) and G2 = (V2, E2) is denoted as G1 ∪ G2 and is defined as follows:

1. The vertex set of G1 ∪ G2 is the union of the vertex sets of G1 and G2, i.e., V(G1 ∪ G2) = V1 ∪ V2.
2. The edge set of G1 ∪ G2 is the union of the edge sets of G1 and G2, i.e., E(G1 ∪ G2) = E1 ∪ E2.

If there are common vertices or edges in G1 and G2, they are not duplicated in G1 ∪ G2. That is, the resulting graph contains only one copy of each vertex and each edge.

The union operation is useful in various graph problems, such as graph connectivity, graph coloring, and graph traversal. For example, if two graphs represent different networks, the union of these graphs can be used to analyze the connectivity between the networks. Similarly, if two graphs represent different parts of a larger system, the union of these graphs can be used to analyze the overall behavior of the system.



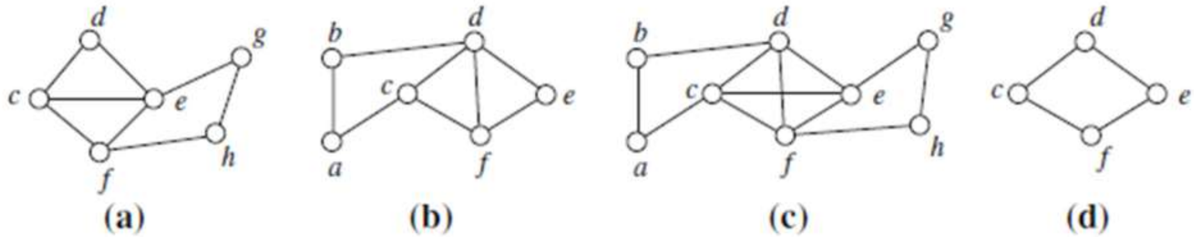


Fig. 2.10 (a) G_1, (b) G_2, (c) $G_1 \cup G_2$, and (d) $G_1 \cap G_2$

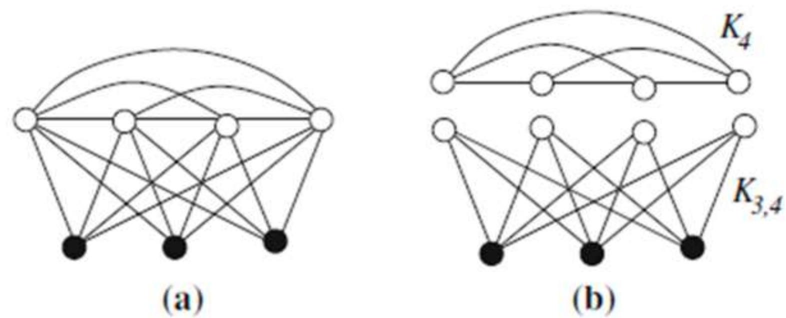


Fig. 2.11 (a) The graph representing handshakes, (b) K_4 and $K_{3,4}$

Figures 2.10(a) and (b) show two graphs *G*1 and *G*2, and Figs. 2.10(c) and (d) illustrate their union and intersection, respectively.

Clearly, we can define the union and intersection of more than two graphs in a similar way.

These operations on graphs can be used to solve many problems veryeasily. We now present such an application of these operations on two graphs [2]. Suppose there are $h + g$ people in a party; h of them are hosts and g of them areguests.

Each person shakes hands with each other except that no host shakes hands with any other host. The problem is to find the total number of handshakes. As usual, we transform the scenario into a graph problem as follows. We form a graph with $h + g$ vertices; h of them are black vertices, representing the hosts and the other g vertices are white, representing the guests.

The edges of the graph represent the handshakes. Thus, there is an edge between every pair of vertices except for that there is no edge between any pair of black vertices. Thus, the problem now is to count the number of edges in the graph thus formed. The graph is illustrated for $h = 3$ and

$g = 4$ in Fig. 2.11(a).

To solve the problem, we note that the graph can be thought of as a union of two

graphs: a complete graph *Kg* and a complete bipartite graph *Kh,g* as illustrated in Fig. 2.11(b). Since there is no common edge between the two graphs, their intersection contains no edges. Thus, the total number of edges in the graph (i.e., the total number of handshakes in the party) is $n(n - 1)/2 + m \times n$.

2-2- Join

In graph theory, there are several operations that can be performed on graphs, including the join operation. The join of two graphs G1 and G2, denoted by G1 ⋈ G2, is defined as follows:

- The vertex set of G1 ⋈ G2 is the union of the vertex sets of G1 and G2, that is, V(G1 ⋈ G2) = V(G1) ∪ V(G2).
- Two vertices v and u in G1 ⋈ G2 are adjacent if and only if:
    - v and u are both in G1, and are adjacent in G1, or
    - v and u are both in G2, and are adjacent in G2, or
    - v is in G1 and u is in G2, or vice versa.

In other words, the join of two graphs combines their vertices and edges in a way that preserves their individual structures.

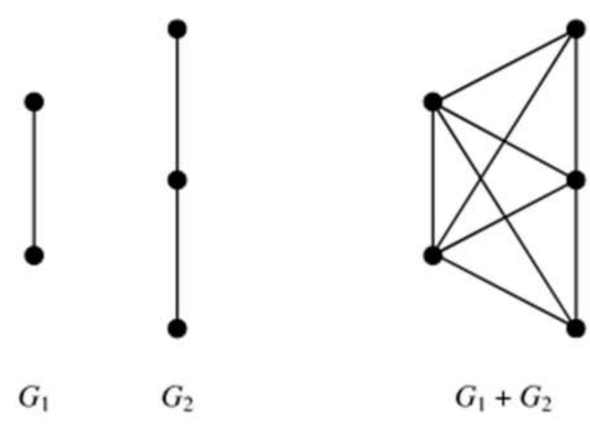
One way to visualize the join of two graphs is to imagine placing one graph on top of the other, and then connecting each vertex in the top graph to every vertex in the bottom graph. This creates a new graph that has all the edges of both original graphs, plus additional edges between the vertices of the two graphs.

It is important to note that the join operation is not commutative, that is, G1 ⋈ G2 is not necessarily the same as G2 ⋈ G1. Additionally, the join of two disconnected graphs is always a disconnected graph, while the join of two connected graphs is not always connected.

The join operation can be useful in various applications, including network analysis, where it can be used to model relationships between different types of entities. For example, if G1 represents a set of people and G2 represents a set of organizations, the join of G1 and G2 can be used to represent the relationships between people and organizations (i.e., which people belong to which organizations).

Certainly, here are some examples of the join operation with corresponding images:

Example 1: Joining two complete graphs



The join G=G1+G 2 of graphs G1 and G2 with disjoint point sets V 1 and V2 and edge sets X1 and X2 is the graph union G 1 union G 2 together with all the edges joining V1 and V2 (Harary 1994, p. 21). Graph joins are implemented in the Wolfram Language as GraphJoin[G1, G2].

A complete k-partite graph K (i,j,...) is the graph join of empty graphs on i, j, ... nodes. A wheel graph is the join of a cycle graph and the singleton graph. Finally, a star graph is the join of an empty graph and the singleton graph (Skiena 1990, p. 132).

The following table gives examples of some graph joins. Here Kn denotes an empty graph (i.e., the graph complement of the complete graph K n), Cn a cycle graph, and K1 the singleton graph.

product	result
$\overline{K}_i + \overline{K}_j + \ldots$	complete k-partite graph $K_{i,j,\ldots}$
$C_n + K_1$	wheel graph W_{n+1}
$\overline{K}_n + K_1$	star graph S_{n+1}
$C_n + \overline{K}_2$	n-dipyramidal graph
$C_m + \overline{K}_n$	cone graph $C_{m,n}$
$\overline{K}_m + P_n$	fan graph
$m\,K_{n-1} + K_1$	windmill graph $D_n^{(m)}$

2-3- Cartesian product

The Cartesian product is a binary operation that combines two sets to create a new set consisting of all possible pairs of elements from the original sets. In the context of graphs, the Cartesian product can be used to create a new graph that combines the vertices and edges of two existing graphs.

Given two graphs G1 = (V1, E1) and G2 = (V2, E2), their Cartesian product G1 x G2 is a new graph defined as follows:

- The vertex set of G1 x G2 is the Cartesian product of V1 and V2, i.e., V(G1 x G2) = V1 x V2.
- Two vertices (u1, v1) and (u2, v2) in G1 x G2 are adjacent if and only if either:
  - u1 = u2 and (v1, v2) is an edge in G2, or
  - v1 = v2 and (u1, u2) is an edge in G1.

In other words, each vertex in G1 x G2 corresponds to a pair of vertices, one from G1 and one from G2, and two vertices in G1 x G2 are adjacent if and only if their corresponding vertices in G1 and G2 are adjacent in their respective graphs.

The Cartesian product of graphs can be useful in a variety of applications, including network design, social network analysis, and theoretical computer science. For example, the Cartesian product of two trees is always a tree, which can be useful in designing hierarchical network topologies.

- The order in which the graphs are multiplied matters. That is, G1 x G2 is not necessarily equal to G2 x G1. In fact, they can be quite different graphs.
- The Cartesian product of two graphs is not necessarily a simple graph. That is, it may have loops or multiple edges.
- The Cartesian product of a graph with itself (i.e., G x G) is sometimes denoted by G^2, and represents the "2-step neighborhood" of each vertex in G. In other words, two vertices in G^2 are adjacent if and only if they can be reached from each other by following two edges in G.
- The Cartesian product of two connected graphs is always connected. However, the diameter of the product graph can be quite large, especially if the original graphs are large.
- The Cartesian product of two graphs can be used to model certain types of interactions between the vertices of the two original graphs. For example, if G1 represents a group of people and G2 represents a set of interests, then G1 x G2 represents the relationships between people and interests (i.e., which people are interested in which topics).
- The Cartesian product of a graph with a single vertex (i.e., G x {v}) is isomorphic to the original graph G.

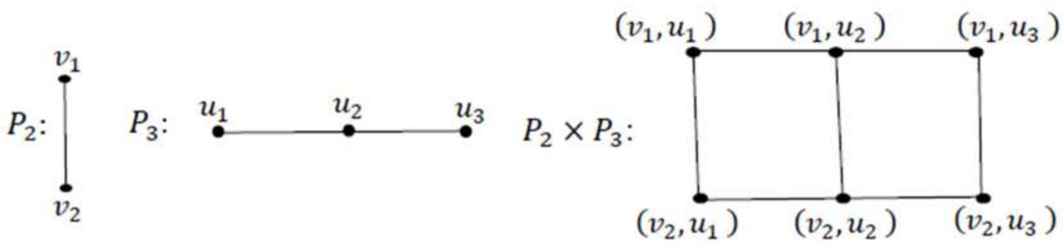


Figure 1.23: The Cartesian of two graphs

2-4- Composition

One important class of operations on graphs is composition.

Composition of graphs is an operation that combines two graphs to create a new graph. Let G and H be two graphs. The composition of G and H is denoted by G ∘ H and defined as follows:

1. The vertex set of G ∘ H is the Cartesian product of the vertex sets of G and H: V(G ∘ H) = V(G) × V(H).
2. Two vertices (u, v) and (u', v') in G ∘ H are adjacent if and only if one of the following conditions hold:

a. u = u' and v is adjacent to v' in H.

b. v = v' and u is adjacent to u' in G.

The composition of two graphs can be visualized as follows: We take the vertex set of one graph and replace each vertex with a copy of the second graph. We then connect the vertices in the new graph according to the rules above.

For example :

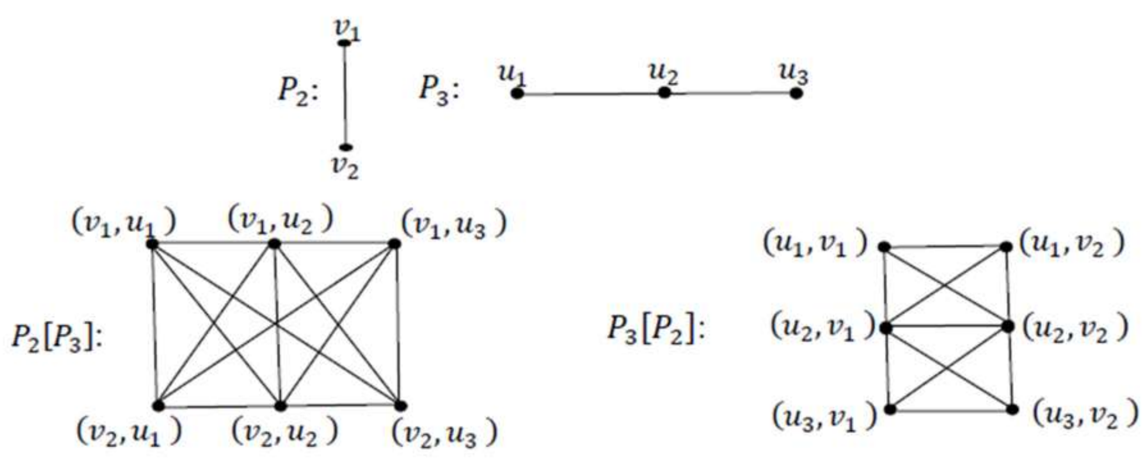


Figure 1.24: The composition of two graphs

Here, we replaced each vertex of G with a copy of H, and connected the vertices in the new graph according to the rules above.

Composition of graphs has several important properties. For example:

1. Composition is associative: (G ∘ H) ∘ K = G ∘ (H ∘ K) for any graphs G, H, and K.
2. Composition is not commutative in general: G ∘ H is not necessarily equal to H ∘ G.
3. Composition of graphs can be used to represent the composition of relations. If G represents the relation R and H represents the relation S, then G ∘ H represents the composition of R and S.

In summary, composition of graphs is an important operation in graph theory and computer science. It allows us to combine two graphs to create a new graph, and has several useful properties.

There are several types of composition in operations on graphs, each with its own specific definition and properties. Here are some common types of composition:

1. Cartesian product: The Cartesian product of two graphs G and H, denoted by G × H, is defined as the graph with vertex set V(G) × V(H) and edge set consisting of all pairs ((u,v),(u',v')) such that either u = u' and v is adjacent to v' in H, or v = v' and u is adjacent to u' in G. The Cartesian product is commutative and associative, and has applications in computer science, such as in the construction of parallel algorithms and in the study of distributed computing.
2. Strong product: The strong product of two graphs G and H, denoted by G **×** H, is defined as the graph with vertex set V(G) × V(H) and edge set consisting of all pairs ((u,v),(u',v')) such that either u = u' and v is adjacent to v' in H, or v = v' and u is adjacent to u' in G, or both. The strong product is associative, but not

commutative, and has applications in the study of graph coloring and graph isomorphism.

3. Lexicographic product: The lexicographic product of two graphs G and H, denoted by G ⊗ H, is defined as the graph with vertex set V(G) × V(H) and edge set consisting of all pairs ((u,v),(u',v')) such that either u is adjacent to u' in G, or u = u' and v is adjacent to v' in H, or u is a vertex of a maximum independent set in G and u' is a neighbor of u in G, and v = v'. The lexicographic product is associative, but not commutative, and has applications in the study of graph coloring and domination.
4. Tensor product: The tensor product of two graphs G and H, denoted by G ⊗ H, is defined as the graph with vertex set V(G) × V(H) and edge set consisting of all pairs ((u,v),(u',v')) such that u is adjacent to u' in G and v is adjacent to v' in H. The tensor product is associative and commutative, and has applications in the study of algebraic graph theory and graph isomorphism.

These are just a few examples of the many types of composition in operations on graphs. Each type of composition has its own specific properties and applications, making them useful tools for studying graphs and their properties.

2-5- Corona

Corona product is a type of binary operation on graphs, which takes two graphs G and H as inputs and produces a new graph, denoted by G ∘ H or H ∘ G, depending on the order of the inputs. The corona product is defined as follows:

- The vertex set of the corona product G ∘ H consists of all vertices of G and H, plus one new vertex for each vertex of G, which is adjacent to all vertices in H.
- The edge set of G ∘ H consists of all edges of G and all edges of H, plus for each vertex v in G, an edge between v and each vertex in the neighborhood of v in H.

Intuitively, the corona product creates a copy of H for each vertex in G, and then connects each copy of H to the corresponding vertex in G. The new graph is similar to the union of G and H, but with additional edges between each vertex of G and its corresponding copy of H.

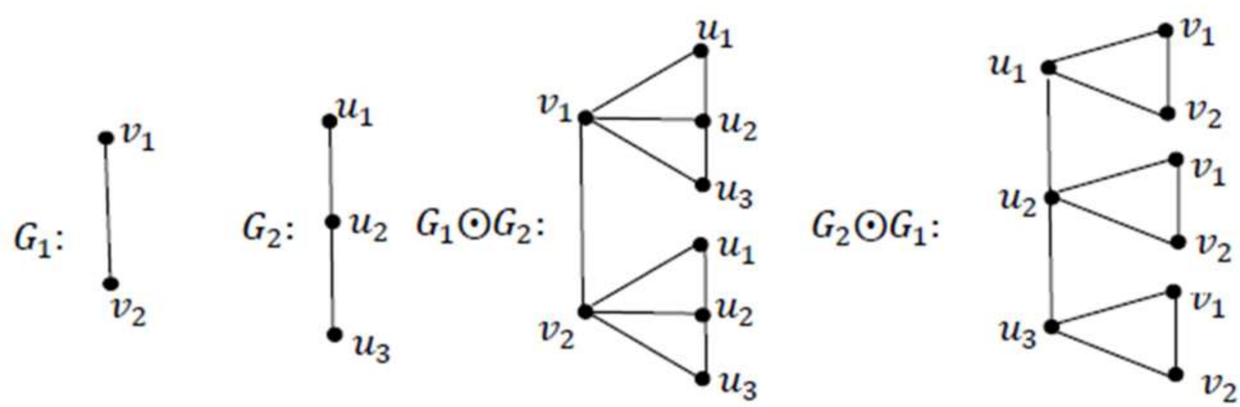


Figure 1.26: Two graphs and their coronas

2-5-1- Properties of Corona Product:

1. The corona product is associative, meaning that (G ∘ H) ∘ K = G ∘ (H ∘ K) for any three graphs G, H, and K.
2. The corona product is not commutative, meaning that G ∘ H is not necessarily equal to H ∘ G for any two graphs G and H.
3. If G and H are both nonempty graphs, then the corona product G ∘ H is a connected graph.
4. If G and H are both regular graphs of degrees d_G and d_H, respectively, then the corona product G ∘ H is a regular graph of degree d_G + d_H.
5. The corona product can be used to construct a class of graphs called corona graphs, which are graphs that can be obtained as the corona product of a path and a cycle. Corona graphs have many interesting properties and applications in graph theory.

2-5-2- Applications of Corona Product:

1. The corona product has applications in the study of graph coloring, where it can be used to construct new graphs with interesting coloring properties.
2. The corona product can be used to model the structure of wireless sensor networks, where the vertices of G represent the sensor nodes and the vertices of H represent the base stations.
3. The corona product has applications in the study of graph embeddings, where it can be used to construct new embeddings of graphs into higher-dimensional spaces.
4. The corona product has applications in computer science, where it can be used to construct new data structures for representing and manipulating graphs.

In summary, the corona product is a useful binary operation on graphs that creates a new graph by combining two existing graphs and connecting them in a specific way. The properties and applications of the corona product make it a valuable tool in the study of graph theory and its applications.

3- *Some resultant graphs from graph operations*

In this section, graphs are given as an example of operations (given in previous section) between other two graphs.

3-1- Wheel Graph

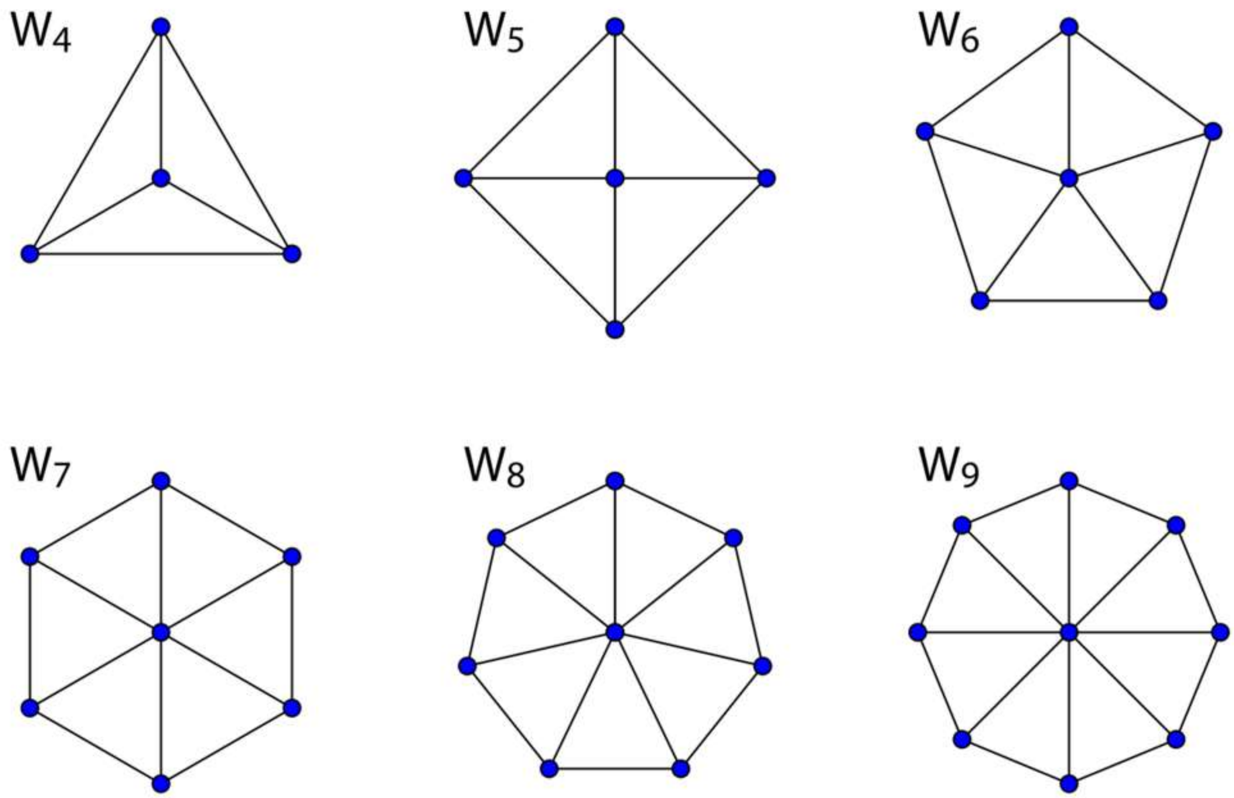
A wheel graph is a type of graph that consists of a central vertex (called the hub) and a cycle of vertices (called the rim) that are all connected to the hub. The wheel graph is denoted by Wn, where n is the number of vertices in the rim.

Formally, the wheel graph Wn is defined as follows:

- The vertex set of Wn is V = {v0, v1, v2, ..., vn-1}.
- The edge set of Wn is E = {(v0, v1), (v0, v2), ..., (v0, vn-1)} U {(vi, vi+1) | i = 1, 2, ..., n-2} U {(vn-1, v1)}.

In other words, the hub vertex v0 is connected to all vertices in the rim, and the vertices in the rim are connected in a cycle. The number of vertices in the wheel graph is n+1, since there are n vertices in the rim and one hub vertex.

The wheel graph is a special case of a spider graph, which is a graph that consists of a central vertex and a set of vertices connected to the central vertex but not to each other. The wheel graph is also a special case of a complete multipartite graph, which is a graph that consists of a set of vertices partitioned into subsets, with all vertices in one subset being connected to all vertices in another subset. Specifically, the wheel graph is a complete bipartite graph with one part consisting of a single vertex (the hub) and the other part consisting of the vertices in the rim.



- Properties and characteristics of wheel graphs:

1. Degree: The degree of the hub vertex v0 in the wheel graph Wn is n, while the degree of each vertex in the rim is 3.
2. Diameter: The diameter of the wheel graph Wn is 2, which means that the maximum distance between any two vertices in the graph is 2.
3. Planarity: The wheel graph Wn is a planar graph for n ≥ 3, which means that it can be drawn on a plane without any edges crossing.

In fact, the wheel graph is a special case of a wheel and spoke graph, which is a planar graph consisting of a central vertex (the hub) and a set of vertices connected to it by spokes.

4. Hamiltonian cycle: The wheel graph Wn has a Hamiltonian cycle, which is a cycle that passes through every vertex exactly once. The Hamiltonian cycle of the wheel graph starts and ends at the hub vertex v0, and visits each vertex in the rim exactly once.
5. Symmetry: The wheel graph Wn has rotational symmetry, which means that it looks the same after rotating it by any multiple of $2\pi/n$ radians around the hub vertex v0. Specifically, the wheel graph has n-fold rotational symmetry.
6. Coloring: The chromatic number of the wheel graph Wn is 3 for $n \geq 3$, which means that it can be colored with 3 colors in such a way that no two adjacent vertices have the same color. This is known as a 3-coloring of the wheel graph.

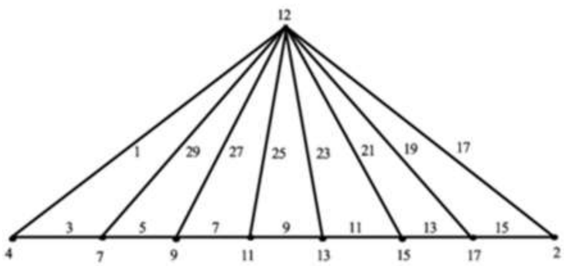
3-2- Double Fan

A double fan graph is a type of graph that consists of two fan graphs connected by a single edge. A fan graph is a type of graph that consists of a central vertex (called the hub) and a set of vertices connected to the hub by edges. The fan graph is sometimes also called a star graph because of its appearance.

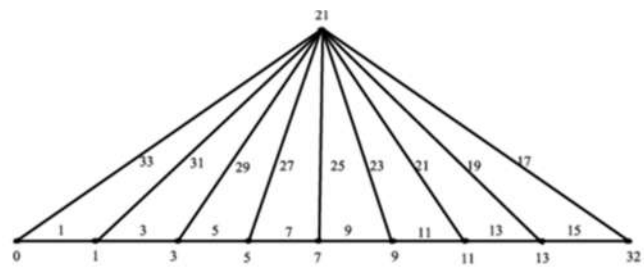
Formally, the double fan graph DF(m, n) is defined as follows:

- The vertex set of DF(m, n) is V = {v1, v2, ..., vm, w1, w2, ..., wn}.
- The edge set of DF(m, n) is E = {(vi, wj) | i = 1, 2, ..., m; j = 1, 2, ..., n} U {(v1, v2), (v1, v3), ..., (v1, vm)} U {(w1, w2), (w1, w3), ..., (w1, wn)}.

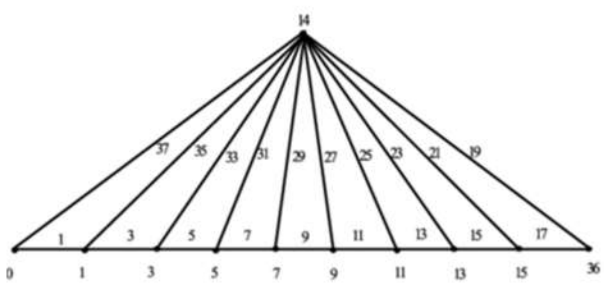
In other words, the double fan graph consists of two fan graphs with m and n vertices respectively, connected by a single edge between the hub vertices. The vertex sets of the two fan graphs are disjoint, and the edge set consists of all possible edges between the vertices in the two fan graphs, as well as the edges connecting the hub vertices to the vertices in their respective fan graphs.



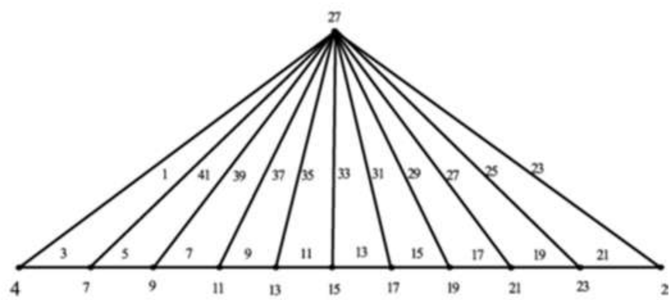
F_8



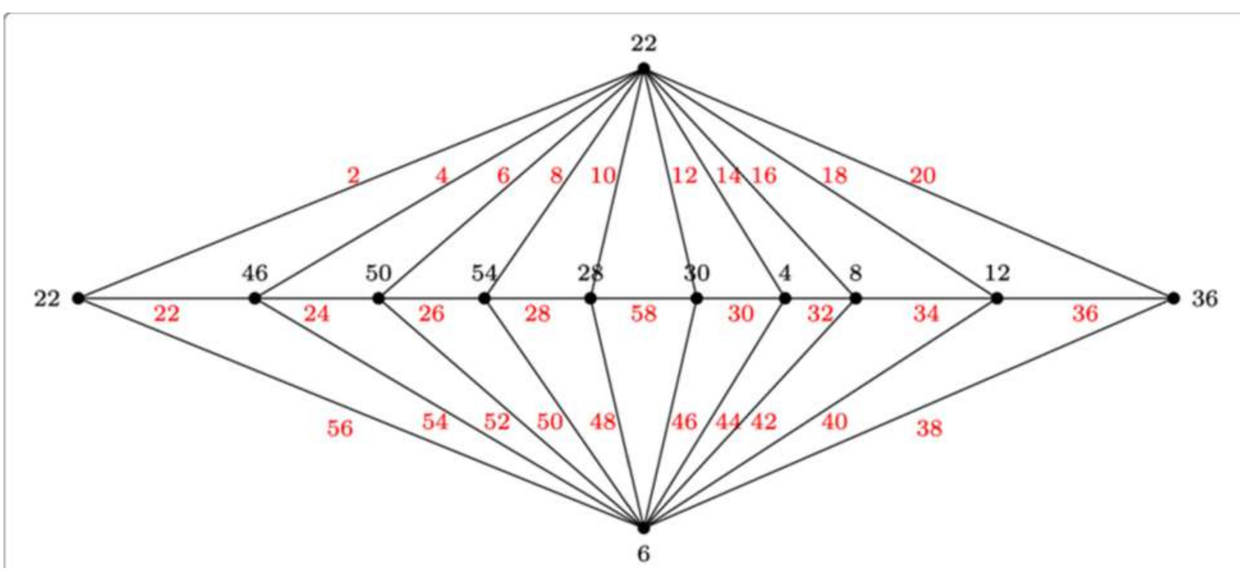
F_9



F_{10}



F_{11}



Some properties of the double fan graph DF(m, n) include:

1. Degree: The degree of the hub vertices v1 and w1 is m and n respectively, while the degree of each of the other vertices in the fan graphs is 1.
2. Diameter: The diameter of the double fan graph DF(m, n) is 2 if both m and n are greater than 1, and 1 otherwise.
3. Planarity: The double fan graph DF(m, n) is not planar if both m and n are greater than 1. This is because it contains a subgraph that is isomorphic to the complete bipartite graph K3,3, which is not planar.
4. Hamiltonian cycle: The double fan graph DF(m, n) has a Hamiltonian cycle if both m and n are greater than 1.
5. Coloring: The chromatic number of the double fan graph DF(m, n) is 2 if both m and n are greater than 1, and 1 otherwise. This means that the double fan graph can be colored with two colors in such a way that no two adjacent vertices have the same color.

3- References

1- Bondy, J. A.; Murty, U. S. R. (2008). *Graph Theory*. Graduate Texts in Mathematics. Springer. p. 29.

2- Graham, R.L.; Grötschel, M.; Lovász, L. (1995). *Handbook of Combinatorics*. MIT Press.

3- Zwillinger, Daniel (2002). *CRC Standard Mathematical Tables and Formulae* (31st ed.). Chapman & Hall/CRC

4- Md. Saidur Rahman Basic Graph Theory

5- Reingold, O.; Vadhan, S.; Wigderson, A. (2002). "Entropy waves, the zig-zag graph product, and new constant-degree expanders"