Published by SET Publisher

Journal of Basic & Applied Sciences

ISSN (online): 1927-5129

Review Article: Graph Colouring and Applications

Faiza Shujat[1,*] and Abu Zaid Ansari[2]

[1]*Department of Mathematics, Taibah University, Madinah KSA*

[2]*Department of Mathematics, Faculty of Science, Islamic University of Madinah, Saudi Arabia*

Article Info:

Keywords:
Graph,
adjacency matrix,
graph isomorphism,
coloring,
chromatic number.

Timeline:
Received: December 13, 2024
Accepted: January 04, 2025
Published: January 21, 2025

Citation: Shujat F, Ansari AZ. Review Article: Graph Colouring and Applications. J Basic Appl Sci 2025; 21: 7-22.

DOI: https://doi.org/10.29169/1927-5129.2025.21.02

*Corresponding Author
E-mail: faiza.shujat@gmail.com

AMS classification 2020: 05C15, 05C60, 17C50.

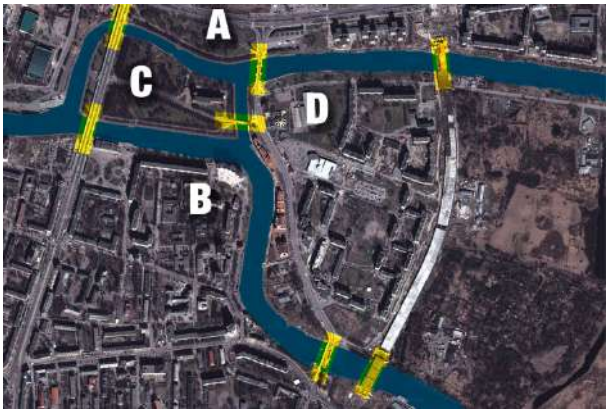
Abstract:

In this article, we present fundamental concept of graph theory and its application in real life. Section 1 deals with the basic definitions and literature review. In Section 2, we describe the algebraic properties of graph. The last section is dedicated to the colouring pattern of graph and its applications. All the information given in this article based on the references [1-8] and the references therein.

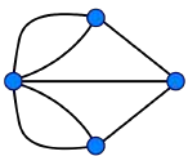
SECTION -1

1.1. Introduction

"Computer science is fundamentally a science of abstraction," according to A. Aho and J. Ulman. Real-world issues must be abstracted by computer scientists so that they may be represented and controlled by a computer. Abstraction might be a straightforward procedure at times. For instance, while designing computer circuits, logic is used. Final test schedule is another example. Associations between classes, students, and rooms must be considered for scheduling to be successful. Graphs are used to model such a collection of relationships between elements. To restate, the collection of elements (courses, students, and rooms) will not be very useful in our model. Because we need to examine the interactions between connections, we also require a collection of connections between pairs of things.



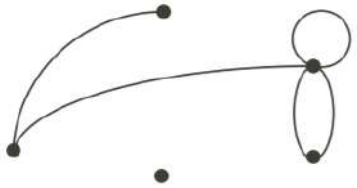
The renowned Swiss mathematician Leonhard Euler created the fundamental concept of graphs in the 18th century. He solved the well-known Königsberg bridge problem using graphs. This image was obtained from the internet. The German city of Königsberg, which is now Kaliningrad in Russia, was located on the Pregel River. It had two islands and a park on the river's banks. Seven bridges connected the mainland to the islands. The question of whether it was feasible to stroll around the town and cross each bridge just once was complicated. This is the problem's graph model. A graph is a collection of points, also known as vertices or nodes, that are joined by lines, also known as edges or arcs. A linked list is the most basic example you are aware of.



1.2. Basic Definitions

In this section we will give some basic concepts, related examples and fundamental theorems. We will start this section by Definition 1.2.1.

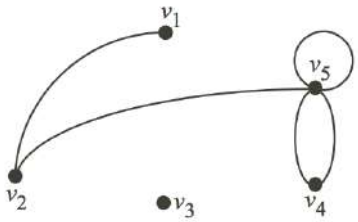
Definition 1.2.1. A graph is formed vertices and edges connecting the vertices.



Example 1.2.1.

Definition 1.2.2. Formally: a graph is a pair of sets (V,E), where V is the set of vertices and E the set of edges, formed by pairs of vertices. E is a multiset, in other words, its elements can occur more than once so that every element has a multiplicity. Often, we label the vertices with letters (for example: $a,b,c\ldots$ or $v_1,v_2\ldots$). Throughout this graduation project, we will label the elements of V in this way.

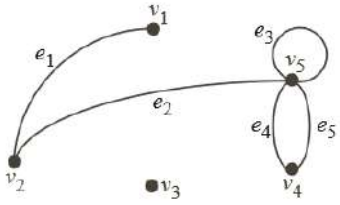
Example 1.2.2.



We have $V = \{v_1, v_2, v_3, v_4, v_5\}$ for the vertices and $E = \{(v_1,v_2), (v_2,v_5), (v_5,v_5), (v_5,v_4), (v_5,v_4)\}$ for the edges. Similarly, we often label the edges with letters $e_1,e_2,e_3,\ldots$

Remark 1.2.1. The two edges (u,v) and (v,u) are the same. In other words, the pair is not ordered.

Example 1.2.3. (Continuing from the previous example) We label the edges as follows:



So E = {e_1 e_2, e_3, e_4,e_5}.

In graphs, we have the following terminologies:

1. The two vertices *u* and *v* are ***end vertices*** of the edge (*u,v*).
2. Edges that have the same end vertices are ***parallel***.
3. An edge of the form (*v,v*) is a ***loop***.
4. A graph is ***simple*** if it has no parallel edges or loops.
5. A graph with no edges (i.e. *E* is empty) is ***empty***.
6. A graph with no vertices (i.e. *V* and *E* are empty) is a ***null graph***.
7. A graph with only one vertex is ***trivial***.
8. Edges are ***adjacent*** if they share a common end vertex.
9. Two vertices *u* and *v* are ***adjacent*** if they are connected by an edge, in other words, (*u,v*) is an edge.
10. The ***degree*** of the vertex *v*, written as *d*(*v*), is the number of edges with *v* as an end vertex. By convention, we count a loop twice and parallel edges contribute separately.
11. A ***pendant vertex*** is a vertex whose degree is 1.
12. An edge that has a pendant vertex as an end vertex is a ***pendant edge***.
13. An ***isolated vertex*** is a vertex whose degree is 0.

Example 1.2.4. (Continuing from Example 1.2.3)

• $v_4$and v_5 are end vertices of e_5.
• e_4 and $e_5$are parallel.
• e_3 is a loop.
• The graph is not simple.
• e_1 and e_2 are adjacent.
• v_1 and v_2 are adjacent.
• The degree of v_1 is 1 so it is a pendant vertex.
• e_1 is a pendant edge.
• The degree of v_5 is 5.
• The degree of $v_4$is 2.
• The degree of v_3 is 0 so it is an isolated vertex.

Throughout this project, we will label graphs with letters, for example: *G* = (*V, E*).

The minimum degree of the vertices in a graph G is denoted δ(*G*) (= 0 if there is an isolated vertex in *G*). Similarly, we write ∆(*G*) as the maximum degree of vertices in *G*.

Example 1.2.5. (Continuing from Example 1.2.3) δ(*G*) = 0 and ∆(*G*) = 5.

Remark 1.2.2. In this course, we only consider finite graphs, i.e. *V* and *E* are finite sets.

Since every edge has two end vertices, we get

Theorem 1.2.1. [**HANDSHAKING LEMMA**] The graph *G* = (*V*,*E*), where *V* = {$v_1,\ldots, v_n$} and *E* = {$e_1,\ldots,e_m$}, satisfies

$$\sum_{i=1}^{n} d(v_i) = 2m$$

Corollary 1.2.1. Every graph has an even number of vertices of odd degree.

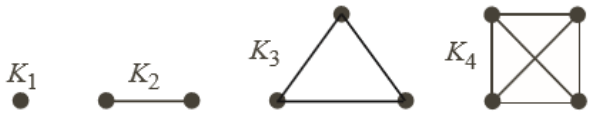
Proof. If the vertices $v_1,\ldots,v_k$ have odd degrees and the vertices $v_{k+1},\ldots,v_n$ have even degrees, then (Theorem 1.2.1)

$d(v_1) + \cdots + d(v_k) = 2m - d(v_{k+1}) - \cdots - d(v_n)$ is even. Therefore, *k* is even.

Example 1.2.6. (Continuing from Example 1.2.3) Now the sum of the degrees is 1 + 2 + 0 + 2 + 5 = 10 = 2 · 5. There are two vertices of odd degree, namely v_1 and v_5.

Complete Graph

A simple graph that contains every possible edge between all the vertices is called a complete graph. A complete graph with *n* vertices is denoted as K_n. The first four



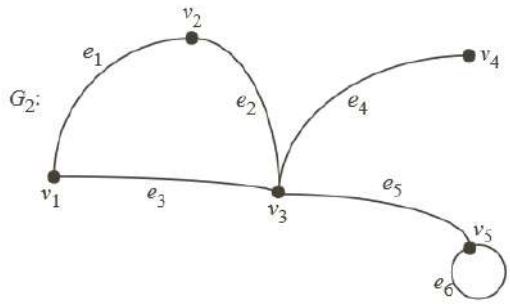
complete graphs are given below:

Subgraph

The graph $G_1 = (V_1,E_1)$ is a subgraph of $G_2 = (V_2,E_2)$ if

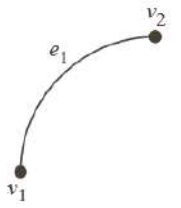
1. $V_1 \subseteq V_2$ and
2. Every edge of G_1 is also an edge of G_2.

Example 1.2.7. We have the graph

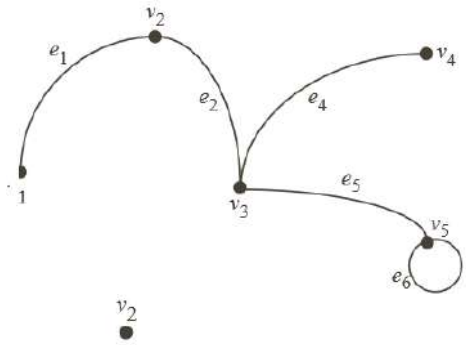


and some of its subgraphs are

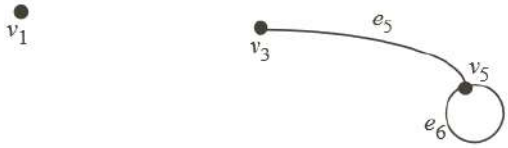
G_1:



G_1:

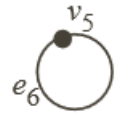


G_1:



and

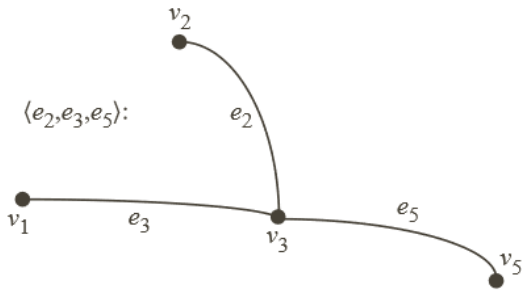
G_1:



Subgraph Induced by the Edges

The subgraph of $G = (V,E)$ induced by the edge set $E_1 \subseteq E$ is $G_1 = (V_1,E_1) =_{def}$. $<E_1>$,where V_1 consists of every end vertex of the edges in E_1.

Example 1.2.8. (Continuing from the above) From the original graph G, the edges e_2, e_3 and e_5 induce the subgraph



Subgraph Induced by the Vertices

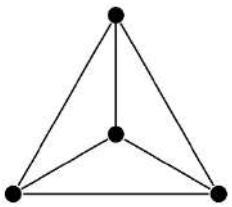
The subgraph of $G = (V,E)$ induced by the vertex set $V_1 \subseteq V$ is $G_1 = (V_1,E_1) =_{def}$. $<V_1>$,where E_1 consists of every edge between the vertices in V_1.

Example 1.2.9. (Continuing from Example 1.2.7) From the original graph G, the vertices v_1, v_3 and v_5 induce the subgraph

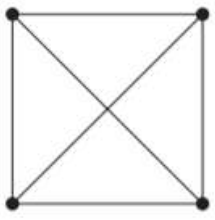
A complete subgraph of G is called a ***clique*** of G.

Planer Graph

A graph is called planer if it can be drawn in the figure without edges crossing (but two can be meet at the same vertex) such a drawing is called planer representation of the graph.



→ Planer graph



→Non-planer graph

Euler's Formula

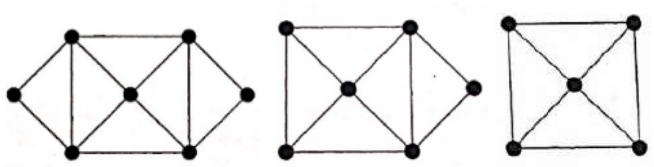
A planar representation of a graph splits the plane into regions, including an unbounded region. For instance,

the planar representation of the graph shown in the following figure splits the plane into six regions. These are labeled in the figure. Euler showed that all planar representations of a graph split the plane into the same number of regions. He accomplished this by finding a relationship among the number of regions, the number of vertices, and the number of edges of a planar graph.

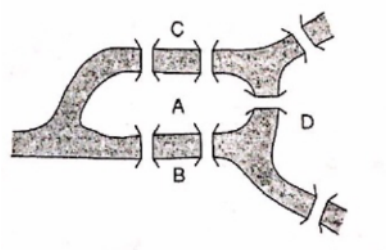
$$R = e - v + 2$$

1.3. Eulerian Graph

A connected graph *G* is **Eulerian** if there exists a closed trail containing every edge of *G*. such a trial is an **Eulerian trial**. Note that this definition requires each edge to be traversed once and once only. A non-Eulerian graph *G* is semi-Eularian if there exists a trail containing every edge of *G*. The following figure show graph that are Eulerian, semi-Eularian and non-Eularian, respectively.



Problems on Eularian graphs frequently appear in books on recreational mathematics. A typical problem might ask whether a given diagram can be drawn without lifting one's pencil from paper and without repeating any lines. The name 'Eulerian' arises from the fact that Euler was the first person to solve the famous **Konigsberg bridges problem** which asks whether you cross each of the seven bridges in the next exactly once and return to your starting point. This is equivalent to asking whether the graph has an Eulerian trail.



One question that immediately arises is 'can one find necessary and sufficient conditions for a graph to be Eulerian?' Before answering this question in Theorem 1.3.2, we prove a simple lemma.

Lemma1.3.1. If G is a graph in which the degree of each vertex is at least 2, then G contains a cycle.

Proof. If *G* has any loop of multiple edges, the result is trivial. We can therefore suppose that *G* is a simple graph. Let *v* be any vertex of *G* we construct a walk

$v \rightarrow v_1 \rightarrow v_2 \rightarrow \ldots$ inductively by choosing v_1 to be any vertex adjacent to *v* and, for each $i > 1$, choosing v_{i+1} to be any vertex adjacent to v_i except v_{i+1} ; the existence of each vertex is guaranteed by our hypothesis. Since *G* has only finitely many vertices, we must eventually choose a vertex that has been chosen before. If v_k is the required cycle.

Theorem 1.3.2. (Euler 1736). A connected graph G is Eulerian if and only if the degree of each vertex of G is even.

Proof. (→) Suppose that *P* is an Eulerian trail of *G*. Whenever *P* passes through a vertex, there is a contribution of 2 towards the degree of that vertex. Since each edge occurs exactly once in *P*, each vertex must have even degree.

(←) The proof is by induction on the number of edges of *G*. Suppose that the degree of each vertex is even. Since *G* is connected, each vertex has degree at least 2 and so, by Lemma 1.2.1*G* contains a cycle *C*. If *C* contains every edge of *G*, the proof is complete. If not, we remove from *G* the edges of *C* to form a new, possible disconnected, graph *H* with fewer edges than *G* and in which each vertex still have even degree. By the induction hypothesis, each component of *H* has an Eulerian trail. Since each component of *H* has at least one vertex in common with *C*, by connectedness, we obtain the required Eulerian trail of *G* by following the edges of *C* until a non-isolated vertex of *H* is reached, tracing the Eulerian trial of the component of *H* that contains that vertex, and then continuing along the edges of *C* until we reach a vertex belonging to another component of *H*, and so on. The whole process terminates when we return to the initial vertex.

Corollary 1.3.2. A connected graph is Eulerian if and only if its set of edges can be split up into disjoint cycles.

Corollary 1.3.3. A connected graph is semi-Eulerian if and only if it has exactly two vertices of odd degree.

Note that, in a semi-Eulerian graph, any semi-Eulerian trail must have one vertex of odd degree as its initial vertex and the other as its final vertex. Note also that, by the handshaking lemma, a graph cannot have exactly one vertex of odd degree.

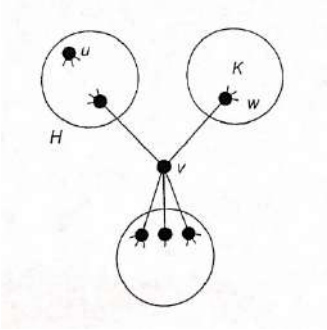
We conclude our discussion of Eulerian graph with an algorithm constructing an Eulerian trail in a giving Eulerian graph. The method is known as **Fleury's algorithm**.

Theorem 1.3.3. Let G be an Eulerian graph. Then the following construction is always possible, and produces an Eulerian trial of G.

Start at any vertex *u* and traverse the edges in an arbitrary manner, subject only to the following rules:

(i) Erase the edges as they are traversed, and if any isolated vertices result, erase them too;

(ii) At each stage, use a bridge only if there is no alternative.



Proof. We show first that the construction can be carried out at each stage. Suppose that we have just reached a vertex *v*. If $v \neq u$, then the subgraph *H* that remains is connected and contains only two vertices of odd degree, *u* and *v*. To show that the construction can be carried out, we must show that the removal of the next edge does not disconnected *H* – or, equivalently, that *v* is incident with at most one bridge. But if this is not the case, then there exists a bridge *vw* such that the component *K* of *H* – *vw* containing *w* does not contain *u* (see the next figure). Since the vertex *w* has odd degree in *K*, some other vertex of *K* must also have odd degree, giving the required contradiction. If $v = u$, the proof is almost identical, as long as there are still edges incident with *u.*

SECTION – 2: ALGEBRAIC PROPERTIES OF GRAPHS

2.1. Introduction

There are several graph operations. Amongst these the complement, difference of graph, union, intersection and symmetric difference are well-known. In this chapter, first we shall study the binary operations on simple graphs. Further, we prove that the collection simple graphs with two binary operations say ring sum and intersection of graphs forms a commutative ring with identity. Later, we will show how to represent graphs in several ways and lastly, we study the isomorphism between two simple graphs.

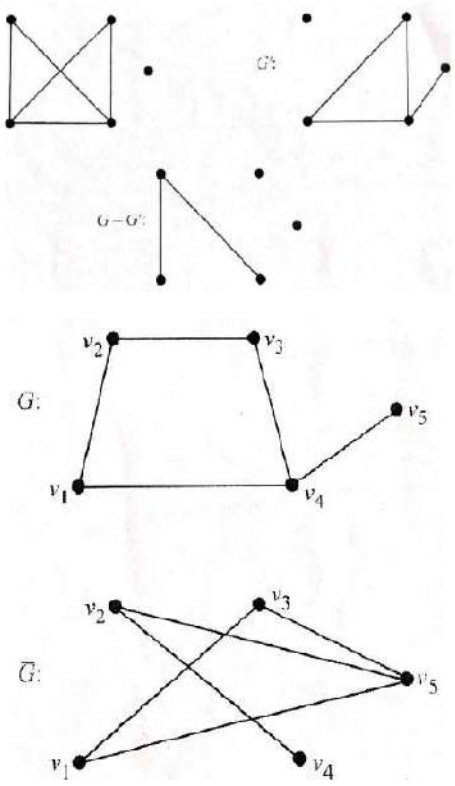
2.2. Binary Operations

The **complement** of the simple graph $G = (V, E)$is the simple graph $\bar{G} = (V, \bar{E})$, where the edges in $\bar{E}$are exactly the edges not in *E.*

The complement of the complete graph K_n is the empty graph with *n* vertices. Obviously, $\bar{\bar{G}} = G$.

If the graphs $G = (V, E)$ and $G' = (V', E')$ are simple and $V \supseteq V'$, then **the difference of graphs** is $G - G' = (V', E'')$ where E'' contains those edges from *G* that are not in *G'.*

Example 2.2.1.



Some binary operations between two simple graphs $G_1 = (V_1,E_1)$ and $G_2 = (V_2,E_2)$ are presented below:

a) The **union** is $G_1 \cup G_2 = (V_1 \cup V_2, E_1 \cup E_2)$.

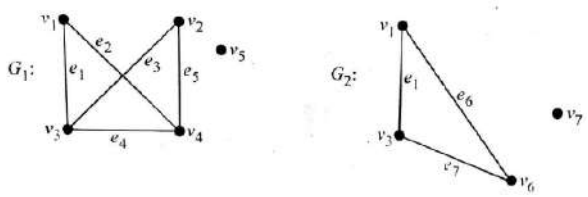
b) The **intersection** is $G_1 \cap G_2 = (V_1 \cap V_2, E_1 \cap E_2)$.

c) The ***ring sum*** $G_1 \oplus G_2$ is the subgraph of $G_1 \cup G_2$ induced by the edge set $E_1 \oplus E_2$.

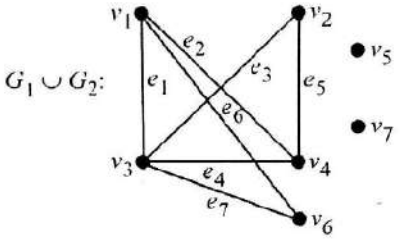
Note: The set operation $\oplus$ is the symmetric difference, i.e. $E_1 \oplus E_2 = (E_1 - E_2) \cup (E_2 - E_1)$ or equivalently $E_1 \oplus E_2 = (E_1 \cap \overline{E_2}) \cup (E_2 \cap \overline{E_1})$.

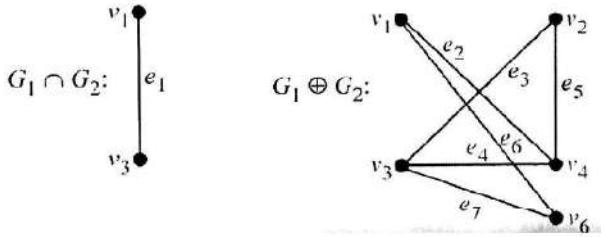
Since the ring sum is a subgraph induced by an edge set, there are no isolated vertices. All three operations are commutative and associative.

Example 2.2.3. For the graphs



we have,





Remark. The operations `∪', `∩' and `⊕' can also be defined for more general graphs other than simple graphs. Naturally, we have to` `keep track" of the multiplicity of the edges:

- ∪: The multiplicity of an edge in $G_1 \cup G_2$ is the larger of its multiplicities in G_1 and G_2.
- ∩: The multiplicity of an edge in $G_1 \cap G_2$ is the smaller of its multiplicities in G_1 and G_2.
- ⊕: The multiplicity of an edge in $G_1 \oplus G_2$ is $|m_1 - m_2|$, where m_1 is its multiplicity in G_1 and m_2 is its multiplicity in G_2.

(We assume zero multiplicity for the absence of an edge.) In addition, we can generalize the difference operation for all kinds of graphs if we take account of the multiplicity. The multiplicity of the edge e in the difference $G - G'$ is

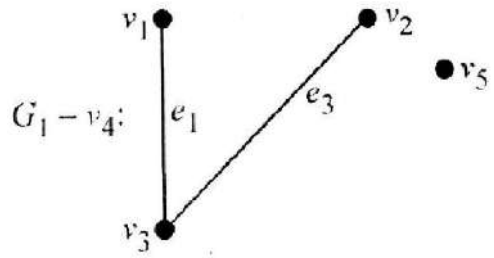
$$m_1 \dot{-} m_2 = \begin{cases} m_1 - m_2, \text{ if } m_1 \geq m_2 \\ 0, \text{ if } m_1 < m_2 \end{cases}$$

(Also known as the proper difference), where m_1 and m_2 are the multiplicities of e in G_1 and G_2, respectively.

Removal of a Vertex

If v is a vertex of the graph $G = (V, E)$, then $G - v$ is the subgraph of G induced by the vertex set $V - \{v\}$. We call this operation the removal of a vertex.

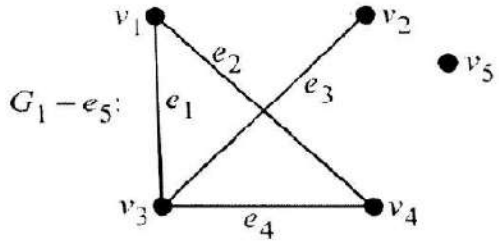
Example 2.2.4. (Continuing from Example 2.2.3)



Removal of an Edge

Similarly, if e is an edge of the graph $G = (V,E)$, then $G - e$ is graph (V,E'), where E' is obtained by removing e from E. This operation is known as removal of an edge. Notice that we are not talking about removing an edge as in Set Theory, because the edge can have non-unit multiplicity and we only remove the edge once.

Example 2.2.5. (Continuing from Example 2.2.3)



Short-circuit

If u and v are two distinct vertices of the graph $G = (V,E)$, then we can short-circuit the two vertices u and v and obtain the graph (V',E'), where

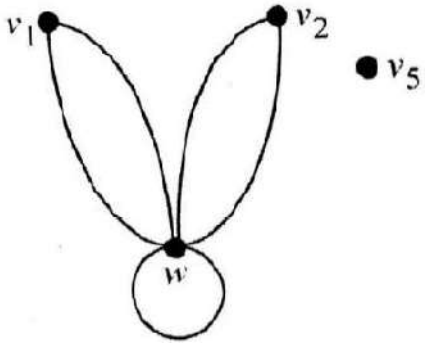
$V' = (V - \{u,v\}) \cup \{w\}$

$E' = (E - \{(v',u),(v',v) \mid v' \in V\}) \cup \{(v',w) \mid (v',u) \in E \text{ or } (v',v) \in E\} \cup \{(w,w) \mid (u,u) \in E \text{ or } (v,v) \in E\}$, where $w \notin V$ is the new vertex.

Note:

- Recall that the pair of vertices corresponding to an edge is not ordered.
- We have to maintain the multiplicity of the edges. In particular, the edge (*u, v*) becomes a loop.

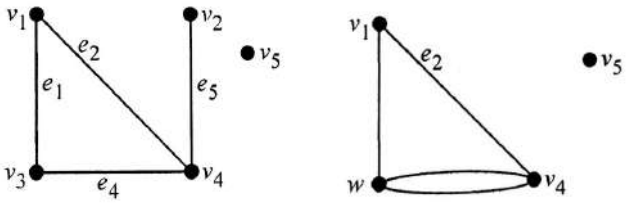
Example 2.2.6. (Continuing from Example 2.2.3) Short-circuit v_3 and v_4 in the graph G_1.



Edge Contracting

In the graph $G = (V,E)$, contracting the edge $e = (u,v)$ (not a loop) means the operation in which we first remove e and then short-circuit u and v. (Contracting a loop simply removes that loop.)

Example 2.2.7. (Continuing from Example 2.2.3) We contract the edge e_3 in G_1 by first removing e_3 and then short-circuiting v_2 and v_3.



Ring of Simple Graphs

The collection simple graphs with two binary operations say ring sum and intersection of graphs ($\oplus$, $\cap$) forms a commutative ring with identity.

1) ($G,\oplus$) is an Abelian Group.

- **Identity:**

$E_1 \oplus \Phi = (E_1 - \Phi) \cup (\Phi - E_1) = E_1 \cup \Phi = E_1$

$G_1 \oplus \phi = \phi \oplus G_1 = G_1$, where ϕ is null graph. Φ is empty set. Therefore ϕ is the identity of G.

- **Inverse:** $G_1 \oplus G_1 = \phi$, Hence the inverse of all elements in G is itself.
- **Associative:**

$(E_1 \oplus E_2) \oplus E_3$

$= (((E_1 \cap \overline{E_2}) \cup (\overline{E_1} \cap E_2)) \cap \overline{E_3}) \cup \overline{(E_1 \oplus E_2 \cap E_3)}$ Symmetric Difference

$= (((E_1 \cap \overline{E_2}) \cup (\overline{E_1} \cap E_2)) \cap \overline{E_3}) \cup \overline{(((E_1 \cup E_2) \cap (\overline{E_1} \cup \overline{E_2}))} \cap E_3)$ Symmetric Difference

$= (E_1 \cap \overline{E_2} \cap \overline{E_3}) \cup (\overline{E_1} \cap E_2 \cap \overline{E_3}) \cup \overline{(((E_1 \cup E_2) \cup (\overline{E_1} \cup \overline{E_2}) \cap E_3)}$ Intersection distributes over union and De' Morgan

$= (E_1 \cap \overline{E_2} \cap \overline{E_3}) \cup (\overline{E_1} \cap E_2 \cap \overline{E_3}) \cup (((\overline{E_1} \cup \overline{E_2}) \cup (E_1 \cup E_2) \cap E_3)$ De' Morgan

$= (E_1 \cap \overline{E_2} \cap \overline{E_3}) \cup (\overline{E_1} \cap E_2 \cap \overline{E_3}) \cup (\overline{E_1} \cap \overline{E_2} \cap E_3) \cup (E_1 \cap E_2 \cap E_3)$ Intersection distributes over

$E_1 \oplus (E_2 \oplus E_3)$

$= (E_1 \cap \overline{E_2 \oplus E_3}) \cup (\overline{E_1} \cap ((E_2 \cap \overline{E_3}) \cup (\overline{E_2} \cap E_3)))$ Symmetric Difference

$= (E_1 \cap ((E_2 \cup E_3) \cap (\overline{E_2} \cup \overline{E_3}))) \cup (\overline{E_1} \cap ((E_2 \cap \overline{E_3}) \cup (\overline{E_2} \cap E_3)))$ Symmetric Difference

$= (E_1 \cap \overline{((E_2 \cup E_3) \cap (\overline{E_2} \cup \overline{E_3}))}) \cup (\overline{E_1} \cap E_2 \cap \overline{E_3}) \cup (\overline{E_1} \cap \overline{E_2} \cap E_3)$ Intersection distributes over union and De' Morgan

$= (E_1 \cap ((\overline{E_2} \cup \overline{E_3}) \cap (E_2 \cup E_3))) \cup (\overline{E_1} \cap E_2 \cap \overline{E_3}) \cup (\overline{E_1} \cap \overline{E_2} \cap E_3)$ De' Morgan

$= (E_1 \cap \overline{E_2} \cap \overline{E_3}) \cup (E_1 \cap E_2 \cap E_3) \cup (\overline{E_1} \cap E_2 \cap \overline{E_3}) \cup (\overline{E_1} \cap \overline{E_2} \cap E_3)$ Intersection distributes over

Hence, $(G_1 \oplus G_2) \oplus G_3 = G_1 \oplus (G_2 \oplus G_3)$.

- **Abelian:** $G1 \oplus G2 = G2 \oplus G1$

$G_1 \oplus G_2$: subgraph of $G_1 \cup G_2$ induced by $E_1 \oplus E_2$

$G_2 \oplus G_1$: subgraph of $G_2 \cup G_1$ induced by $E_2 \oplus E_1$

$E_2 \oplus E_1 = (E_2 - E_1) \cup (E_1 - E_2)$

$= (E_1 - E_2) \cup (E_2 - E_1)$

$= E_1 \oplus E_2$

Hence, $G_1 \oplus G_2 = G_2 \oplus G_1$.

2) (G, ∩)is associative: $(G_1 \cap G_2) \cap G_3 = G_1 \cap (G_2 \cap G_3)$.

$(G_1 \cap G_2) \cap G_3 = ((V_1 \cap V_2) \cap V_3, (E_1 \cap E_2) \cap E_3)$

$= (V_1 \cap (V_2 \cap V_3), E_1 \cap (E_2 \cap E_3))$

$= G_1 \cap (G_2 \cap G_3)$.

3) Distributive

(i) $\boldsymbol{G_1 \cap (G_2 \oplus G_3) = (G_1 \cap G_2) \oplus (G_1 \cap G_3)}$

(ii) $(G_2 \oplus G_3) \cap G_1 = (G_2 \cap G_1) \oplus (G_3 \cap G_1)$

$$G_1 \cap (G_2 \oplus G_3) = (V_1, E_1) \cap ((V_2, V_3), (E_2 - E_3) \cup (E_2 - E_3))$$

$$= \big((V_1 \cap V_2) \cap (V_1 \cap V_3), (E_1 \cap (E_2 - E_3) \cup (E_1 \cap (E_3 - E_2))\big)$$

$$= \big((V_1 \cap V_2) \cap (V_1 \cap V_3), ((E_1 \cap E_2) - (E_2 \cap E_3) \cup ((E_2 \cap E_3) - (E_1 \cap E_2)\big)$$

$$= (G_1 \cap G_2) \oplus (G_1 \cap G_3)$$

Similarly, we can prove $(G_2 \oplus G_3) \cap G_1 = (G_2 \cap G_1) \oplus (G_3 \cap G_1)$.

4) Commutative $G_1 \cap G_2 = G_2 \cap G_1$

Let $G_1 = (V_1, E_1)$ and $G_2 = (V2, E_2)$

$G_1 \cap G_2 = (V_1 \cap V_2, E_1 \cap E_2)$

$= (V_2 \cap V_1, E_2 \cap E_1)$

$= G_2 \cap G_1$

5) The Identity with Respect to Intersection

If G = (V, E) is a simple graph, then G ∩ G = (E ∩ E, V ∩ V) = (E, V) = G

$G \cap G = G$, hence G itself is the identity of G with respect to the intersection.

Therefore, we get our conclusion.

2.3. Matrix Representation

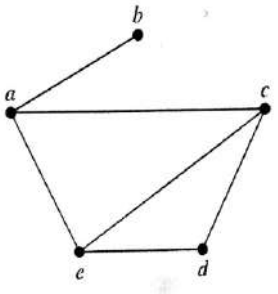
There are many useful ways to represent graphs. As we will see throughout this section, inworking with a graph it is helpful to be able to choose its most convenient representation. In this section, we will show how to represent graphs in several different ways. Sometimes, two graphs have exactly the same form, in the sense that there is a one-to-one correspondence between their vertex sets that preserves edges. In such a case, we say that the two graphs are isomorphic. Determining whether two graphs are isomorphic is an important problem of graph theory that we will study in this section.

Representing Graphs

One way to represent a graph without multiple edges is to list all the edges of this graph. Another way to represent a graph with no multiple edges is to use adjacency lists, which specify the vertices that are adjacent to each vertex of the graph.

Example 2.3.1. Use adjacency lists to describe the simple graph given in the following figure.

Solution: The following table lists those vertices adjacent to each of the vertices of the graph.

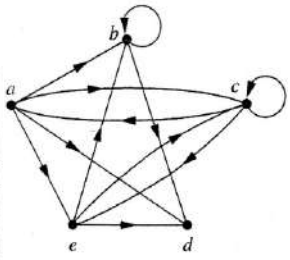


Vertex	Adjacent Vertices
a	b, c, e
b	a
c	a, d, e
d	c, e
e	a, c, d

Example 2.3.2. Represent the directed graph shown in the following figure by listing all the vertices that are the terminal vertices of edges starting at each vertex of the graph.

Solution: The following table represents the digraph shown in the following figure.

Initial Vertex	Terminal Vertices
a	b, c, d, e
b	b, d
c	a, c, e
d	
e	b, c, d



Adjacency Matrices

Carrying out graph algorithms using the representation of graphs by lists of edges, or by adjacency lists, can be cumbersome if there are many edges in the graph. To simplify computation, graphs can be represented using matrices. Two types of matrices commonly used to represent graphs will be discussed here. One is

based on the adjacency of vertices, and the other is based on incidence of vertices and edges.

Suppose that $G = (V, E)$ is a simple graph where $|V| = n$ and the vertices of G are listed arbitrarily as $v_1, v_2, \ldots, v_n$. The adjacency matrix A of G, with respect to this listing of the vertices, is the $n \times n$ zero–one matrix with 1 as it's (i, j)th entry when $v_i$and v_j are adjacent, and 0 as its (i,j)th entry, when they are not adjacent. In other words, if its adjacency matrix is $A = [a_{ij}]$, then

$$a_{ij} = \begin{cases} 1 & \text{if } \{v_i, v_j\} \text{ is an edge of } G, \\ 0 & \text{otherwise.} \end{cases}$$

Example 2.3.3. Use an adjacency matrix to represent the graph shown in the following figure.

Solution: We order the vertices as *a, b, c, d*. The matrix representation of this graph is

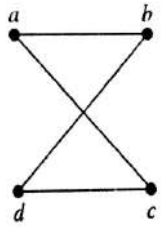
$$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}.$$

Example 2.3.4. Draw a graph with the adjacency matrix with respect to the ordering of vertices *a, b, c, d*.

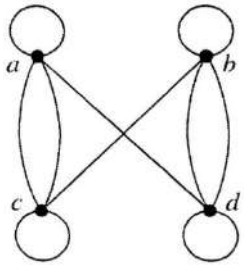
$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

Note that an adjacency matrix of a graph is based on the ordering chosen for the vertices.

Hence, there may be as many as $n!$ different adjacency matrices for a graph with n vertices, because there are $n!$ different orderings of n vertices. The adjacency matrix of a simple graph is symmetric, that is, $a_{ij} = a_{ji}$, because both of these entries are 1 when v_i and v_j are adjacent, and both are 0 otherwise. Furthermore, because a simple graph has no loops, each entry aii, $i = 1, 2, 3. . .n$, is 0. Adjacency matrices can also be used to represent undirected graphs with loops and with multiple edges. A loop at the vertex v_i is represented by a 1 at the (i, i)th position of the adjacency matrix. When multiple edges connecting the same pair of vertices $v_i$and v_j, or multiple loops at the same vertex, are present, the adjacency matrix is no longer a zero–one matrix, because the (i, j)th entry of this matrix equals the number of edges that are associated to $\{v_i, v_j\}$. All undirected graphs, including multigraphs and pseudo graphs, have symmetric Adjacency matrices.



Example 2.3.5. Use an adjacency matrix to represent the following pseudo graph.



Solution:

$$\begin{bmatrix} 1 & 0 & 2 & 1 \\ 0 & 1 & 1 & 2 \\ 2 & 1 & 1 & 0 \\ 1 & 2 & 0 & 1 \end{bmatrix}$$

We used zero–one matrices to represent directed graphs. The matrix for a directed graph $G = (V, E)$ has a 1 in its (i, j)th position if there is an edge from $v_i$to v_j ,where $v_1, v_2, \ldots, v_n$ is an arbitrary listing of the vertices of the directed graph. In other words, if $A = [a_{ij}]$ is the adjacency matrix for the directed graph with respect to this listing of the vertices, then

$$a_{ij} = \begin{cases} 1 & \text{if } \{v_i, v_j\} \text{ is an edge of } G, \\ 0 & \text{otherwise.} \end{cases}$$

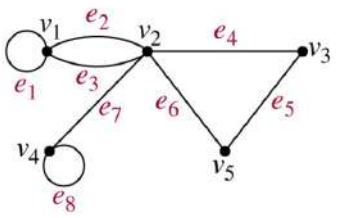
The adjacency matrix for a directed graph does not have to be symmetric, because there may not be an edge from v_j to v_i when there is an edge from v_i to v_j. Adjacency matrices can also be used to represent directed multigraphs. Again, such matrices are not zero–one matrices when there are multiple edges in the same direction connecting two vertices. In the adjacency matrix for a directed multigraph, a_{ij} equals the number of edges that are associated to (v_i, v_j).

Incidence Matrices

Another common way to represent graphs is to use incidence matrices. Let $G = (V, E)$ be an undirected graph. Suppose that $v_1, v_2, \ldots, v_n$ are the vertices and $e_1, e_2, \ldots, e_m$ are the edges of G. Then the incidence matrix with respect to this ordering of V and E is the $n \times m$ matrix $M = [m_{ij}]$, where

$$m_{ij} = \begin{cases} 1 & \text{when edge } e_j \text{ is incident with } v_i, \\ 0 & \text{otherwise.} \end{cases}$$

Example 2.3.6. Representation of the pseudo graphand its incidence matrix.



$$\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \begin{array}{cccccccc} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 \end{array} \\ \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} \end{array}.$$

2.4. GraphIsomorphism

We often need to know whether it is possible to draw two graphs in the same way. That is, do the graphs have the same structure when we ignore the identities of their vertices? For instance, in chemistry, graphs are used to model chemical compounds. Different compounds can have the same molecular formula but can differ in structure. Such compounds can be represented by graphs that cannot be drawn in the same way. The graphs representing previously known compounds can be used to determine whether a probably new compound has been studied before.

The simple graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are isomorphic if there exists a one-to-one and onto function f from V_1 to V_2 with the property that a and b are adjacent in G_1 if and only if $f(a)$ and $f(b)$ are adjacent in G_2, for all a and b in V_1. Such a function f is called an isomorphism. Two simple graphs that are not isomorphic are called non-isomorphic.

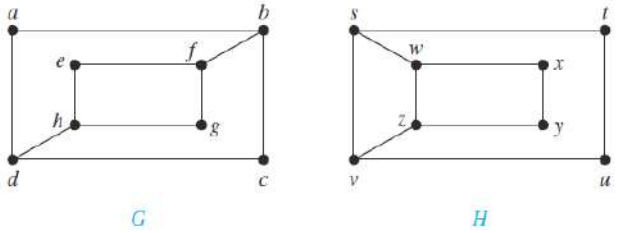
In other words, when two simple graphs are isomorphic, there is a one-to-one correspondence between vertices of the two graphs that preserves the adjacency relationship. Isomorphism of simple graphs is an equivalence relation.

Determining whether Two Simple Graphs are Isomorphic

It is often difficult to determine whether two simple graphs are isomorphic. There are $n!$ possible one-to-one correspondences between the vertex sets of two simple graphs with n vertices. Testing each such correspondence to see whether it preserves adjacency and non-adjacency is impractical if n is at all large.

Sometimes it is not hard to show that two graphs are not isomorphic. In particular, we can show that two graphs are not isomorphic if we can find a property only one of the two graphs has, but that is preserved by isomorphism. A property preserved by isomorphism of graphs is called a graph invariant. For instance, isomorphic simple graphs must have the same number of vertices, because there is a one-to-one correspondence between the sets of vertices of the graphs. Isomorphic simple graphs also must have the same number of edges, because the one-to-one correspondence between vertices establishes a one-to-one correspondence between edges. In addition, the degrees of the vertices in isomorphic simple graphs must be the same. That is, a vertex v of degree d in G must correspond to a vertex $f(v)$ of degree d in H, because a vertex w in G is adjacent to v if and only if $f(v)$ and $f(w)$ are adjacent in H.

Example 2.4.1. Determine whether the graphs shown in the following figure are isomorphic.



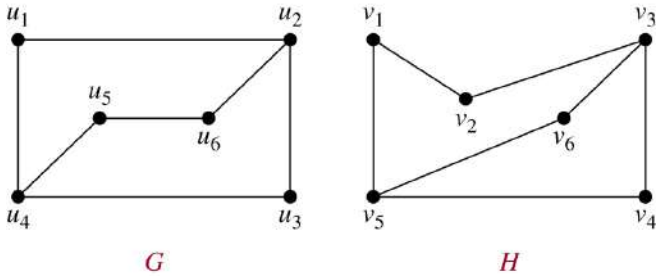
Solution: The graphs G and H both have eight vertices and 10 edges. They also both have four vertices of degree two and four of degree three. Because these invariants all agree, it is still conceivable that these graphs are isomorphic. However, G and H are not isomorphic. To see this, note that because $\deg(a) = 2$ in G, a must correspond to either t, u, x, or y in H, because these are the vertices of degree two in H.

However, each of these four vertices in H is adjacent to another vertex of degree two in H, which is not true for

a in *G*. Another way to see that *G* and *H* are not isomorphic is to note that the subgraphs of *G*and *H* made up of vertices of degree three and the edges connecting them must be isomorphic if these two graphs are isomorphic. However, these sub graphs, are not isomorphic.

To show that a function *f* from the vertex set of a graph *G* to the vertex set of a graph *H* is an isomorphism, we need to show that *f* preserves the presence and absence of edges. One helpful way to do this is to use adjacency matrices. In particular, to show that *f* is an isomorphism, we can show that the adjacency matrix of *G* is the same as the adjacency matrix of *H*, when rows and columns are labeled to correspond to the images under *f* of the vertices in *G* that are the labels of these rows and columns in the adjacency matrix of *G*.

We illustrate how this is done in the example. **Example 2.4.2.** Determine whether the graphs *G* and *H* displayed in the following figure are isomorphic.



G *H*

Solution: Both *G* and *H* have six vertices and seven edges. Both have four vertices of degree two and two vertices of degree three. It is also easy to see that the subgraphs of *G* and *H* consisting of all vertices of degree two and the edges connecting them are isomorphic. Because *G* and *H* agree with respect to these invariants, it is reasonable to try to find an isomorphism *f*.

We now will define a function *f* and then determine whether it is an isomorphism. Because deg(u_1) = 2 and because u_1 is not adjacent to any other vertex of degree two, the image of $u_1$must be either v_4 or v_6, the only vertices of degree two in *H* not adjacent to a vertex of degree two. We arbitrarily set $f(u_1) = v_6$. [If we found that this choice did not lead to isomorphism, we would then try $f(u_1) = v_4$.] Because u_2 is adjacent to u_1, the possible images of u_2 are $v_3$and v_5. We arbitrarily set $f(u_2) = v_3$. Continuing in this way, using adjacency of vertices and degrees as a guide, we set $f(u_3) = v_4$, $f(u_4) = v_5$, $f(u_5) = v_1$, and $f(u_6) = v_2$. We now have a one-to-one correspondence between the vertex set of *G* and the vertex set of *H*, namely, $f(u_1) = v_6$, $f(u_2) = v_3$, $f(u_3) = v_4$, $f(u_4) = v_5$, $f(u_5) = v_1$, $f(u_6) = v_2$. To see whether *f* preserves edges, we examine the adjacency matrix of *G* and the adjacency matrix of *H* with the rows and columns labeled by the images of the corresponding vertices in *G*,

$$\mathbf{A}_G = \begin{array}{c} \\ u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \end{array} \begin{array}{c} \begin{array}{cccccc} u_1 & u_2 & u_3 & u_4 & u_5 & u_6 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix} \end{array},$$

$$\mathbf{A}_H = \begin{array}{c} \\ v_6 \\ v_3 \\ v_4 \\ v_5 \\ v_1 \\ v_2 \end{array} \begin{array}{c} \begin{array}{cccccc} v_6 & v_3 & v_4 & v_5 & v_1 & v_2 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix} \end{array}.$$

Because $A_G = A_H$, it follows that *f* preserves edges. We conclude that *f* is an isomorphism,

So *G* and *H* are isomorphic. Note that if *f* turned out not to be an isomorphism, we would not have established that *G* and *H* are not isomorphic, because another correspondence of the vertices in *G* and *H* may be an isomorphism.

SECTION - 3: COLORING OF GRAPHS

3.1. Introduction

Problems related to the coloring of maps of regions. Such as maps of parts of the world. Have generated many results in graph theory. When a map is colored, two regions with a common border are customarily assigned different colors. One way to ensure that two adjacent regions never have the same color use a different color for each region. However, this is inefficient, and one maps with many regions it would be hard to distinguish similar colors. Instead, a small number of colors should be used whenever possible. Consider the problem of determining the least number of colors that, an be used to color a map so that adjacent regions never have the same color. For instance, for the map shown on the left Figure 3.1.1. Four colors suffice, but three colors are not enough. in the map on the right in Figure 3.1.1, three colors are sufficient (but two are not).

Each map in the plane can be are represented by a graph. To set up this correspondence, are represented by these vertices have a common border. Two regions that touch at only one point are not considered adjacent. The resulting graph is called the ***dual graph*** of the map. By the way in which dual graphs of maps are constructed, it is clear that any map in the plane has a planar dual graph. Figure 3.1.2 displays. The dual graph that correspond to the maps shown in Figure 3.1.1. The problem of coloring the regions of a map is equivalent to the problem of coloring the vertices of the dual graph so that no two adjacent vertices in this graph have the same color.

We now define a graph coloring.

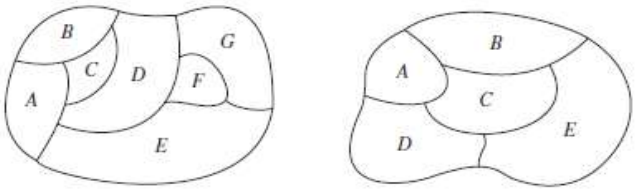


Figure 3.1.1.

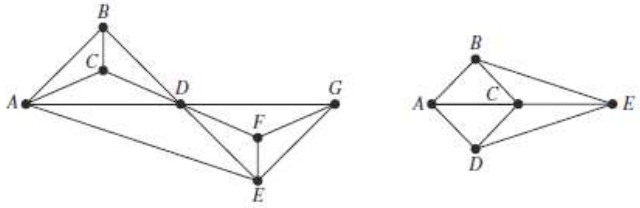


Figure 3.1.2.

3.2. Graph Coloring

A coloring of a simple graph is the assignment of a color to each vertex of the graph so that no two adjacent vertices are assigned the same color.

A graph can be colored by assign ng a different color to each of its vertices. However, for most graphs coloring can be found that uses fewer colors than the number of vertices in the graph. What is the least number of colors necessary?

Chromatic Number

The chromatic number of a graph is the least number of colors needed for a coloring of this graph. The chromatic number of a G is denoted by $\chi(G)$. (Here χ is the Greek letter chi) Note that asking for the chro-maticnumber of a planar graph is the same as asking for the minimum number of colors required to color a planar map so that no two adjacent region are assigned the same color. This question has been studied for more than 100 years. The answer is provided by one of the most famous theorems in Mathematics.

The Four Color Theorem

The four-color theorem was originally posed as a conjecture in the 1850. It was finally proved by the American mathematicians Kenneth Apple and Wolfgang Haken in 1976. Prior to 1976, many incorrect proofs where published, often with hard-to-find errors. In addition, many futile attempts were made to construct counterexamples by drawing maps that required more than four colors.

Perhaps the most notorious fallacious proof in all of mathematics is the incorrect prove of the four-color theorem published in 1879 by a London barrister and amateur mathematician. Alfred Kemp. Mathematician accepted his proof as correct until 1890. When Percy Hen wood found an error that made Kemps argument incomplete. However, Kemps line of reasoning turned out to be the basis of the successful proof given by Apple and Haken. Their proof relies on a careful case-by – case analysis carried out by computer. They showed that if the four-color theorem where false, there would have to be a counterexample of one of approximately 2000 different types, and they then showed that none of these types exists. They used over 1000 hours of computer time in their proof. This proof generated a large amount of controversy, because computers played such an important role in it. For example, could there be an error in a computer program that led to incorrect result? Was their argument really a proof if it depended on what could be unreliable computer output? Since their proof appeared, simpler proof that rely on checking fewer types of possible counterexample have been found and a proof using an automated proof system has been created. However, no proof not relying on a computer has yet been found.

Note that the four-color theorem applies only to planar graphs. No planar graphs can have arbitrarily large chromatic number, as will be shown in next examples.

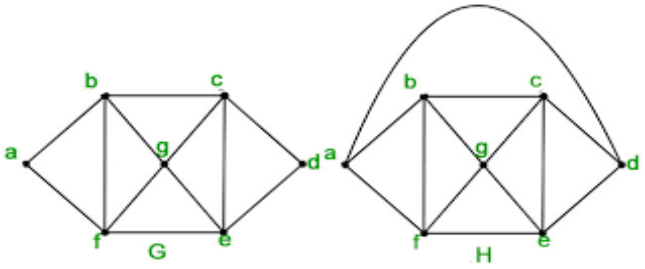
Two things are required to show that the chromatic number of a graph is k. First, we must show that the graph can be colored with k colors.

This can be done by constructing such a coloring. Second, we must show that the graph cannot be colored using fewer than k colors. Example 3.2.1 to

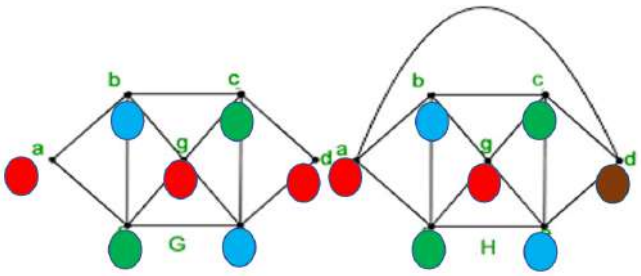
Example 3.2.4 illustrate how chromatic number can be found.

Theorem 3.2.1. The chromatic number of a planar graph is no greater than four.

Example 3.2.1. What are the chromatic numbers of the graphs *G* and *H* shown in the following Figure?



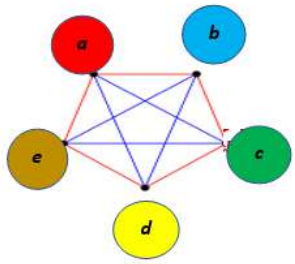
Solution: The chromatic number of at least three, because the vertices *a, b*, and *c* must be assigned different colors. To see if *G* can be colored with three colors, assign red to *a*, blue to *b*, and green to *c*. Then *d* can (and must) be colored red because it is adjacent to *b* and *c*. Furthermore, *e* can (and must) be colored green because it is adjacent only to vertices colored red and blue, and *f* can(and must) be colored blue because it is adjacent only to vertices colored red and green



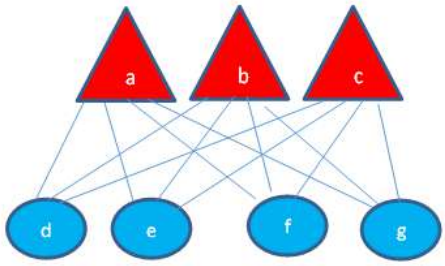
Finally *g* can (and must) be colored red because it is adjacent only to vertices colored blue and green. This produces a coloring of *G* using exactly three colors. Figure displays such a coloring.

The graph *H* is made up of the graph *G* with an edge connecting *a* and *g*. Any attempt to color *H* using three colors must follow the same reasoning as that used to color *G*. except at the last stage, when all vertices other than *g* have been colored. Then, because *g* is adjacent (in *H*)to vertices colored red, blue, and green, a fourth color, say brown, needs to be used. Hence, *H* has a chromatic number equal to 4.

Example 3.2.2. What is the chromatic number of K_n?



K_n



$K_{m,n}$

Solution: A coloring of K_n can be construct using *n* colors by assigning a different color to each vertex. Is there a coloring using fewer colors? The answer is no. No two vertices can be assigned the same color, because every two vertices of this graph are adjacent. Hence, the chromatic number of K_n is *n*. That is, $\chi(K_n) = n$. (Recall that k_n is not planar when $n \geq 5$.

So this result does not contradict the four color theorem). A coloring of K_5 using five colors is shown in the above figure.

Example 3.2.3. What is the chromatic number of the complete bipartite graph $K_{m,n}$. where *m* and *n* are positive integers?

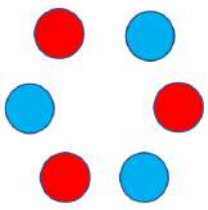
Solution: The number of colors needed may seem to depend on *m* and *n*. Only two colors are needed, because $K_{m,n}$ is a bipartite graph. Hence, $\chi(K_{m,n})=2$. This means that we can color the set of *m* vertices with one color and the set of *n* vertices with a second color. Because edges connect only a vertex from the set of *m* vertices and vertex from the set of *n* vertices, no two adjacent vertices have the same color. A coloring of $K_{3.4}$ with two colors is display in the above figure.

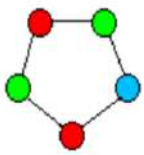
Example 3.2.4. What is the chromatic number of the graph C_n where $n \geq 3$? Where *Cn* is the cycle with *n* vertices.

Solution: We will first consider some individual cases. To begin, let n=6. Pick vertex and color it red. Proceed clock wise in the planar depiction of C_6 shown in the following figure. It is necessary to assign second color, say blue. To the next vertex reached. Continue in the clock wise direction: the third vertex can be colored red. The fourth vertex blue, and the fifth vertex red. Finally the sixth vertex, which is adjacent to the first, can be colored blue. Hence, the chromatic number of C_6 is 2. The following figure displays the coloring constructed here.

Next let n=5 and consider C_5. Pick a vertex and color it red. Proceeding clock wise, it is necessary to assign second color, say blue to the next vertex reached. Continuing in the clock wise direction, the third vertex can be colored red, and the fourth vertex can be colored blue. The fifth vertex cannot be colored either red or blue, because it is adjacent to the forth vertex can be colored blue. The fifth cannot be colored either red or blue, because it is adjacent to the fourth vertex and the first vertex. Consequently a third color is required for this vertex. Note that we would have also needed three colors if we had colored vertices in the counterclockwise direction. Thus the chromatic number of C_5 is 3. A coloring of C_5 using three colors is displayed in the following figure. In general, two colors are needed to color C_n when n is even. To construct such a coloring, simply pick a vertex and color its red. Proceed around the graph in a clockwise direction (using a planar representation of the graph) coloring the second vertex blue, the third vertex red, the fourth vertex can be colored blue, because the two vertices adjacent to it, namely the (n-1) and the first vertex, are both colored red.

When n is odd and n>1, the chromatic number of C_n is 3. To see this, pick an initial vertex. To use only two colors, it is necessary to alternate colors as the graph is traversed in a clockwise direction. However, fourth vertex reached is adjacent to two vertices of different colors, namely, the first and (n-1). Hence a third color must be used. We have shown that $\chi(C_n)$=2 if n is an even positive integer with n≥4 and $\chi(C_n)$=3 if n is an odd positive integer with n≥3.



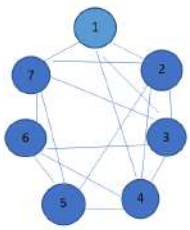


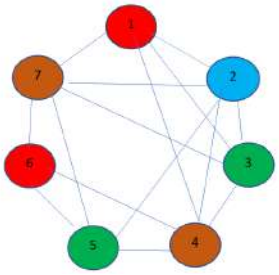
3.3. Applications of Graph Coloring

Graph coloring has a variety of applications to solve problem involving scheduling and assignments. (Note that because no efficient algorithm is known for graph coloring, this does not lead to efficient algorithm for scheduling and assignments.) Examples of such application will be given here. The first application deals with the scheduling of final exams.

Example 3.3.1. Scheduling Final Exam: How can the final exams at a university be scheduled so that no student has two exams at the same time?

Solution: This scheduling problem can be solved using a graph model. With vertices representing courses and with an edge between two vertices if there is a common student in the courses they represent. Each time slot for a final exam is represented by a different color. A scheduling of the exam corresponds to a coloring of the associated graph. For instance, suppose there are seven finals to be scheduled. Suppose the courses are numbered I through 7. Suppose at the following pairs of courses have common student: 1 and 2, 1 and 3, 1 and 4, 1 and 7, 2 and 3, 2 and 4, 2 and 5, 2 and 7, 3 and 4, 3 and 6, 3 and 7, 4 and 5, 4 and 6 , 5 and 6, 5 and 7,and6 and 7. In Figure 8 the graph associated with this set of classes is shown. A scheduling consists of a coloring of this graph. Because the chromatic number of this graph is 4, four time slots are needed. A coloring of the graph using four colors and the associated schedule are shown in the following figure.





Example 3.3.2. Frequency Assignments: Television channels 2 through 13 are assigned to station in North America so that no two station 150 miles can operation the same channel. How can the assignment of channels be modeled by graph coloring?

Solution: Construct graph by assignment a vertex to each station. Two vertices are connected by an edge if they are located within 150 miles of each other. An assignment of channels corresponds to coloring of the graph, where each color represents different channel.

An application of graph coloring to compilers is considered in next example.

Example 3.3.3. Index Registers: In efficient compilers the execution of loops is speeded up when frequency used variable are stored temporarily in index registers in the central processing unit instead of in regular memory. For a given loop, how many index registers are needed? This problem can be addressed using a graph coloring model, let each vertex of a graph represent a variable in the loop. There is an edge between two vertices if the variable they represent must be stored in index register at the same time during the execution of the loop. Thus, chromatic number of the graph gives the number of index register needed, because different register must be assignment to variable when the vertices representing these variables are adjacent in the graph.

COMPETENT INTEREST

The authors declared that there is no conflict of interest.

REFERENCES

[1] Bondy JA, Murty USR. Graph theory with applications, Elsevier Science Publishing Co., Inc., 1976.

[2] Harary F. Graph Theory, Addison-Wesley Publishing Company 1969.

[3] Trudeau RJ. Introduction to Graph Theory, Dover publication INC, New York 1993.

[4] Biggs N. Algebraic Graph Theory, Cambridge Mathematical Library 1994.

[5] Godsil C, Royle GF. Algebraic Graph Theory, Springer, 2001.

[6] Ruohonen K. Graph Theory, Tampere University of Technology 2008.

[7] Gross JL, Yellen J, Anderson M. Graph Theory and Its Applications, CRC Taylor and Francis 2023.

Rosen KH. Discrete Mathematics and Its Applications, Eighth edition, New York, NY: McGraw-Hill, 2019.