UNIT 2  PATHS, CYCLES AND TRAILS

Structure                                      Page No
2.1 Introduction                               19
   Objectives                                   
2.2 Connected Graphs                           19
2.3 Cut-Vertex and Cut-Edge (Bridge)           21
2.4 Bipartite Graphs                           23
2.5 Eulerian Graphs                            25
2.6 Summary                                    26
2.7 Hints/Solutions                            26

2.1 INTRODUCTION

In this section, we shall study three important classes of graphs – connected graphs, bipartite graphs and Eulerian graphs. These graphs find extensive applications in many different areas. In any network, reachability from one vertex to any other vertex is of utmost practical use. For example, in a housing colony it is desirable that every house is connected by road for reachability, by telephone lines for communication, etc. In fact, as we show in this unit, they can all be modelled as problems involving paths or cycles in a graph or digraph.

Objectives

After studying this unit, you should be able to

- explain the terms edge-connectivity, vertex-connectivity, walk, trail, path, cut vertex, cut edge, bipartite graph;
- explain the terms Eulerian graph, Eulerian trail and state necessary and sufficient conditions for a graph to be Eulerian.

2.2 CONNECTED GRAPHS

In this section we investigate the extent to which a given graph is connected. In particular, we discuss the question: how many edges do we need to remove from a given connected graph so that it becomes disconnected? This, and other similar questions related to connectivity, is important ones to consider when designing telecommunications networks, road systems and other networks for example, in a telecommunications network it is essential that the network should still be operable if some of the links between the exchanges become damaged, or are blocked by other calls.

To check the connectivity of a graph we need to discuss some important terms as being used.

A **walk** is a list $v_0, e_1, v_1, ..., e_k, v_k$ of vertices and edges such that, for $1 \leq i \leq k$, the edge $e_i$ has endpoints $v_{i-1}$ and $v_i$. A **trail** is a walk with no repeated edge. A $u,v$- walk or $u,v$- trail has first vertex $u$ and the last vertex $v$ known as endpoints. If, in addition, all the vertices are different in a trail then it is called a **path**.
The length of a walk, trail, path, or cycle is its number of edges. A walk or trail is closed if its endpoints are the same.

Note: All paths are trails. All trails are walks. Now we advise you to start reading the Text Book DW.

Now we shall see some more examples of paths and trails.

Example 1: Let us consider the following lists of vertices and edges.

\[ W_1 : v_1, e_1, v_2, e_2, v_3, e_3, v_6, e_6, v_2, e_2, v_8, e_8, v_4 \]

\[ W_2 : v_1, e_6, v_4, e_4, v_5, e_5, v_4, e_8, v_2, e_1, v_1 \]

\[ T : v_1, e_1, v_2, e_8, v_4, e_4, v_7, v_2, e_2, v_3 \]

\[ P : v_1, e_1, v_2, e_2, v_3, e_3, v_6, e_4, e_5 \]

Then we observe the following:

i) \( W_1 \) is a walk which is not a path.

ii) \( W_2 \) is a closed walk which is not a path.

iii) \( T \) is a trail which is not a path

iv) \( P \) is a path.

Different types of possible traversing in a network are walk, trail and a path. Suppose from our house we walk to reach our friend’s house in a housing colony. All the three types of traversing are possible. It could be as if a drunkard or a senseless person does, namely, we keep on traveling through the same road and repeating the same spots or junction. Usually we avoid repetition and use an optimal path (may be unknowingly).

Example 2: Let us consider the following graph \( G \) with \( V(G) = \{u, v, w, x, y, z\} \) and \( E(G) = \{e_1, e_2, \ldots, e_9\} \).

Fig. 1
We can reach from $u$ to $y$, in many ways of which $u e_1, x e_5, w e_4, v e_4, v e_7, z e_9, y$ is a walk with length 6, $u e_2, v e_4, w e_5, x e_6, z e_7, w e_8, y$ is a trail with length 6 which is not a path and $u e_1, x e_5, w e_8, y$ is a path with length 3 which is one of the shortest. Obviously the last type is often the chosen case. As, any two vertices can be reached by a path, the graph is connected (intuitively, it lies in one piece).

However, the following graph is disconnected

![Graph](attachment:graph.png)

**Fig. 2**

How many components are there? There are 5 components.

Now we want you to try some exercises.

---

**E1)** In the following graph

![Graph](attachment:graph2.png)

**Fig. 3**

(i) find all the paths from $s$ to $z$;
(ii) find a trail of length 8 which is not a path;
(iii) are there any cycles containing both $s$ and $z$? Justify your answer.

---

### 2.3 CUT-VERTEX AND CUT-EDGE (BRIDGE)

For many applications it is necessary to know more about a graph than just whether or not it is connected. For example, in telecommunications networks there are usually several different paths between a given pair of subscribers (vertices). In such a situation, it is important to know how many links (edges) can be blocked or broken without preventing a call being made between the two subscribers. In order to answer this and similar questions, we need to investigate connected graphs a little further. In a graph by deleting a vertex or an edge can increase the number of components. Although deleting an edge can
only increase the number of components by 1, deleting a vertex can increase it by many.

When we obtain a subgraph by deleting a vertex, it must be a graph, so deleting the vertex also delete all edges incident to it.

In a graph G, deletion of a vertex ‘v’ means that the vertex ‘v’ and all the edges incident on ‘v’ are removed. The deletion of an edge ‘e’ means that only the edge ‘e’ is removed and its end vertices are retained.

Let us do an example.

Example 3: Let G is given as in Fig. 4

In some graphs G there may be a vertex v whose removal from G increases the number of components of G. Similarly, there may be an edge e whose removal from G increases the number of components. In particular, if G is connected, G − v and G − e will be disconnected. Such a vertex v is called a cut–vertex and such an edge e is called a cut-edge (bridge).

In the above example, v is such a vertex, but e is not such an edge. However, there are cut-edges in G. (Try to identify them)

Example 4: Let v be a cut-vertex of a simple graph G. Let us prove that \( \overline{G} − v \) is connected.

Suppose v is a cut vertex of G.

Let \( u, w \in V(G) \) and u and w belong to different component of \( G − v \). That is there is a path from u to w through v. But in \( \overline{G} \) there is an edge joining u and w. There may be another path joining u and w through v in \( G \). Therefore deletion of v in \( \overline{G} \) will not make the graph disconnected.
From Theorem 1.2.14 in the **Text Book DW**, it follows that in a tree (Unit 4), every edge is a cut-edge, but the Petersen graph has no cut-edge (how?). Clearly, no vertex of a complete graph is a cut vertex and no edge of a cycle is a cut-edge.

Identification of cut-vertices and cut-edges are very crucial in some networks (in a communication network related to defense etc.). Note that a vertex \( v \) in a graph \( G \) is a cut vertex if and only if there is vertices \( u, w \) in \( G \) such that \( v \) is on every \( u - w \) path.

Now we want you to try some exercises.

---

**E2)** Let \( G \) be a graph. For \( v \in V(G) \) and \( e \in E(G) \), describe the adjacency and incidence matrices of \( G - v \) and \( G - e \) in terms of the corresponding matrices for \( G \).

**E3)** Let \( v \) be a vertex of a connected simple graph \( G \). Prove that \( v \) has a neighbour in every component of \( G - v \). Conclude that no graph has a cut-vertex of degree 1.

---

### 2.4 BIPARTITE GRAPHS

In this section we will talk about a very important class of graphs know as bipartite graphs. A **bipartite graph** (or **bigraph**) is a graph whose vertices can be divided into two disjoint sets \( U \) and \( V \) such that every edge connects a vertex in \( U \) to one in \( V \); that is, \( U \) and \( V \) are independent sets. Equivalently, a bipartite graph is a graph that does not contain any odd-length cycles. For more information on bipartite graphs you should read text book DW.

**READ DW: Chapter 1, Section 1.2, from page 24 up to page 26.**

Here, we will have a closer look at the structure of a bipartite graph discussed in Fig. 5. Recall that in a bipartite graph \( G \), we can partition the vertex set of \( G \) into two sets \( X \) and \( Y \) such that an edge of \( G \) joins a vertex of \( X \) with a vertex of \( Y \). If every vertex of \( X \) joins every vertex of \( Y \), then we have a complete bipartite graph.

Bipartite graphs occur in job assignment problems, marriage problem (Chapter 3 of **Text Book DW**) etc.

![Fig. 5](attachment:image_url)

Theorem 1.2.18 in **Text Book DW** states that, a graph \( G \) is bipartite if and only if all its cycles are of even length. Thus, no complete graph is bipartite and the Petersen graph is not bipartite. Further, since a tree has no cycles, every tree is
Graph Theory

trivially a bipartite graph. Also, the star \((K_{1,n})\) is the only complete bipartite tree.

**Example 4**: Let us Consider the following graph \(G\).

![Graph G](image)

This has no cycles of odd length (list all cycles of \(G\) and convince yourself). Following the proof of Theorem 1.2.18 in *Text Book DW*, we can convince yourself that \(G\) is indeed bipartite by redrawing the same. Consider \(X = \{1, 3, 5, 7\}\) and \(Y = \{2, 4, 6, 8\}\)

Then \(G = \)

Consider \(T = \)

which can be redrawn as follows with

\(X = \{1, 3, 5, 7, 8, 10\}\) and \(Y = \{2, 4, 6, 9, 11, 12\}\)
Note that $C_4$ – the cycle on 4 vertices is isomorphic to the complete bipartite graph $K_{2,2}$.

Try some exercises.

E4) Prove that a bipartite graph has a unique bipartition if and only if it is connected.

## 2.5 EULERIAN GRAPHS

An explorer wishes to explore all the routes between a numbers of cities. Can he arrange his tour so as to traverse each route only once?

Recall the Konigsberg Bridge Problem (KBP) mentioned in Unit 1. The problem was to check whether it is possible to start at any land area, walk across each bridge exactly once and get back to the starting place. Using the graph theoretic terminologies, the problem was to find a closed trail containing all the edges. As it has been observed that a necessary condition for existence of such a trail is that all vertices degrees be even. Later on Euler stated that these conditions are also sufficient.

We define a graph as Eulerian if it has a closed trail containing all edges. For example Fig. 2 in Unit 1, no such closed trail exists. Hence the graph G is not Eulerian.

### Example 7:
Let us try to draw some Eulerian graphs.

![Fig. 7](image)

In Fig. 7 (a) we have an Eulerian trail as abcdefgcegfbfa where as in Fig. 7(b) we have Eulerian trail as abcadcfbefdea.

### Example 8:
Let us determine the values of m and n such that $K_{m,n}$ is Eulerian. We know that a graph is Eulerian iff every vertex of G has even degree. So if m and n are even then $K_{m,n}$ is Eulerian.
**Example 9:** Let us prove that every Eulerian simple graph with an even number of vertices has an even number of edges.

A graph $G$ is Eulerian iff every vertex has even degree. Suppose that $G$ is an Eulerian simple graph with an even number of vertices. Since $G$ is Eulerian, every vertex has even degree. Therefore the number of edges of $G$ is even.

Now let us try some related exercises.

---

**E5)** Determine whether $K_4$ contains the following:

a) A walk that is not a trail.

b) A trail that is not closed and is not a path.

c) A closed trail that is not a cycle.

**E6)** In the graph below, find all the maximal paths, maximal cliques and maximal independent sets. Also find all the maximum paths, maximum cliques, and maximum independent sets.

![Graph](image.png)

**E7)** What is the minimum number of trails needed to decompose the Petersen graph? Is there a decomposition into this many trails using only paths?

---

**2.6 SUMMARY**

In this unit, we have covered the following:

1. Edge-connectivity, vertex-connectivity, walk, trail, path, cut vertex, cut edge, bipartite graph.

2. Eulerian graph, Eulerian trail and necessary and sufficient conditions for a graph to be Eulerian.

---

**2.7 HINTS/SOLUTIONS**

**E1)** (i) length 3: svwz

length 4: stvwxz and svwyz

length 5: stuvwxz, stvwxyz, and sywxyz

length 6: stuvwxyz and stvwxyz

length 7: stuvwxyz.
(ii) length 8: stuvwxyzw.

(iii) Any cycle containing both s and z must consist of a path from s to z following by a path from z to s. However, all paths from s to z contain both v and w, and all paths from z to s contain v or w, so either v or w must occur twice. Since this is not allowed, there can be no cycle containing both s and z.

E2) For G – v:

**Adjacency Matrix:** If a vertex v is removed from G. Then A(G – v) will be an \((n-1) \times (n-1)\) matrix. The row and column corresponding to the vertex v is removed. No change in the remaining entries.

**Incidence Matrix:** In M(G – v) the row corresponding the vertex v and columns corresponding to the edges adjacent to v are removed - No change in other entries.

For G – e:

**Adjacency Matrix:** In A(G – e) the entries corresponding to vertices adjacent to this edge will be decreased by 1. No change in other entries.

**Incidence Matrix:** In M(G – e) the column corresponding to that edge is deleted. No change in other entries.

E3) Let v be a vertex of a connected simple graph G. Then G – v will be disconnected graph if v is a cut vertex. In G – v the vertex v and all edges adjacent to v are also removed. Therefore the adjacent vertices of v belongs to different components of G – v. That is v has a neighbour in every component of G – v.

If \(d(v) = 1\) then G – v will not make any change therefore v will not be a cut vertex.

E4) Hint: Prove by contradiction method.

E5) 

![Diagram](image)

a) \(v_1e_1v_2e_2v_3e_3v_4e_4v_1e_5e_6v_4\)

b) \(v_3e_3v_4e_4v_1e_5e_3e_2v_2\)

c) not possible.
E6) Clique

\[
\begin{align*}
\text{Path} \\
\text{Independent set} \\
\{v_1, v_4\} \text{ and } \{v_2, v_4\}.
\end{align*}
\]

E7) Minimum number of trails needed is 5.

If G is a connected graph with exactly \(2n(n \geq 1)\) odd vertices then edge set of G can be partitioned into \(n\ open trail.

\[
2n = 2 \times 5 = 10 \quad n = 5
\]

Therefore 5 trails.

\[
G: T_1 \cup T_2 \cup T_3 \cup T_4 \cup T_5
\]

\[
T_1: v_7, e_{11}, v_{10}, e_{15}, v_8, e_4, v_3
\]

\[
T_2: v_4, e_9, v_9, e_{11}, v_6, e_{12}, v_8
\]

\[
T_3: v_9, e_{14}, v_7, e_7, v_2, e_2, v_1
\]

\[
T_4: v_6, e_6, v_1, e_1, v_5, e_{10}, v_{10}
\]

\[
\]

*****