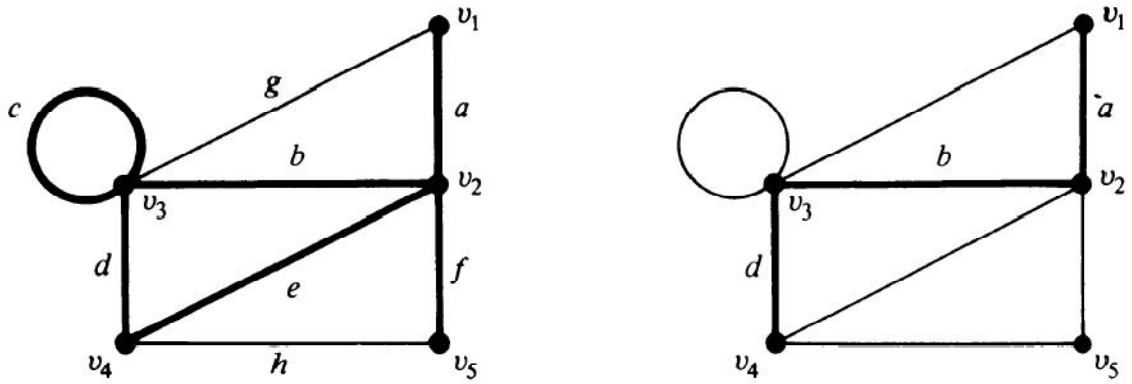
UNIT VI

Walks, Paths and Circuits

6.1 Walk

A walk is an alternating sequence of vertices and edges beginning and ending with vertices such that each edge is incident with the vertices preceding and following it. No edge is appears more than once in a walk. A vertex however may appear more than once.



(a) An Open Walk (b) A Path of Length Three

Fig 3.9

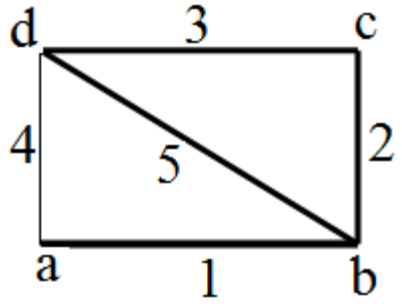
e.g. v1 a v2 b v3 c v3 d v4 e v2 f v5 is a walk shown with heavy lines. A walk is also referred to as an **edge train** or a **chain**. The set of vertices and edges that makes a given walk in a graph G is a subgraph in G

Vertices with which a walk begins and ends are called its **terminal walks**. Vertices v1 and v5 are the terminal vertices of the walk in the example above.

6.1.1 Types of walk

Close walk: If the beginning and ending vertices in a walk are same then it is called closed walk.

Open walk: if the beginning and ending vertices in a walk are different then it is called open walk.



Close Walk: b 2 c 3 d 5 b

Open Walk: a 1 b 5 d 3 c 2 b

Fig 3.10

6.2 Path

An open walk in which one vertex appears more than once is called a path or a simple path or an elementary path.

e.g a 1 b 2 c 3 d 4 a is a path.

In fig 3.9 (a), v1 a v2 b v3 d v4 is a path whereas v1 a v2 b v3 c v3 d v4 e v2 f v5 is not a path. A path does not intersect itself.

6.2.1 Length of a path: The number of edges in a path is called length of a path. An edge which is not a self loop is path of length one. A self loop can be included in a walk but not in path. Fig 3.9 (b) is path of length three.

6.3 Circuit

A closed walk in which no vertex (except the initial and the final vertex) appears more than once is called a circuit. In the other words, a close walk in which beginning and ending vertices are same. That is circuit is closed and non-intersecting walk.

e.g Fig 3.10 Circuit: a 1 b 2 c 3 d 4 1

Fig 3.9 (a) Circuit: v2 b v3 d v4 e v2

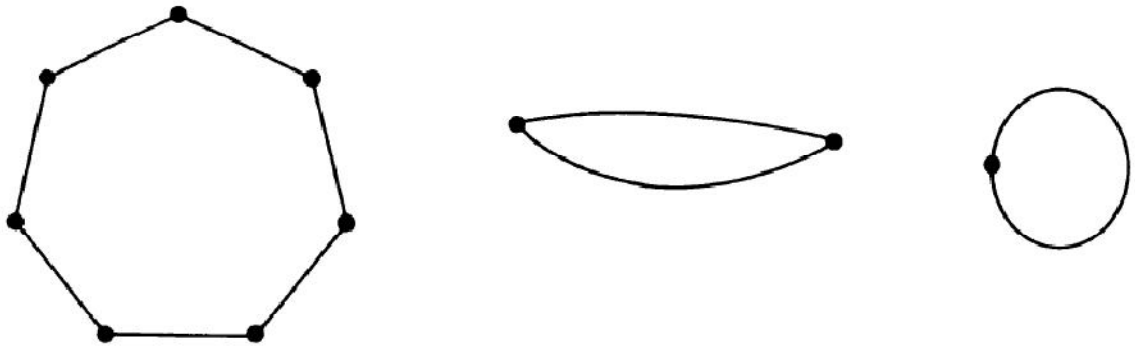
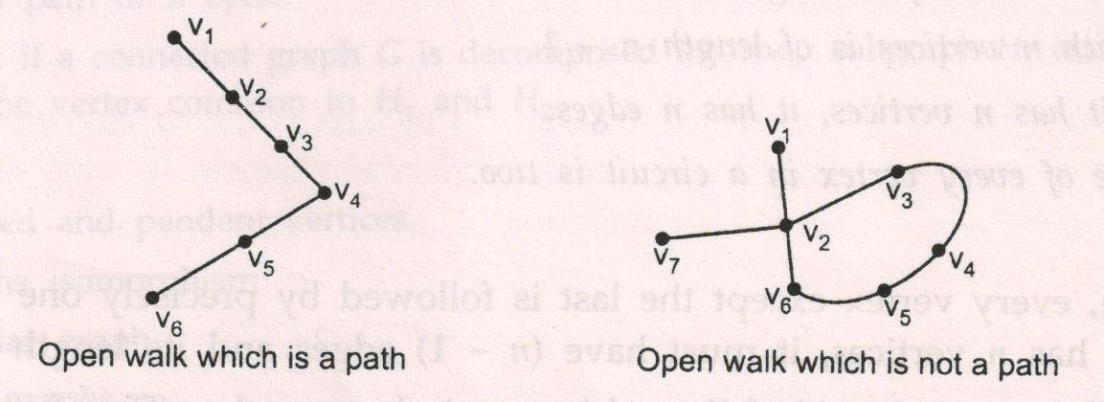


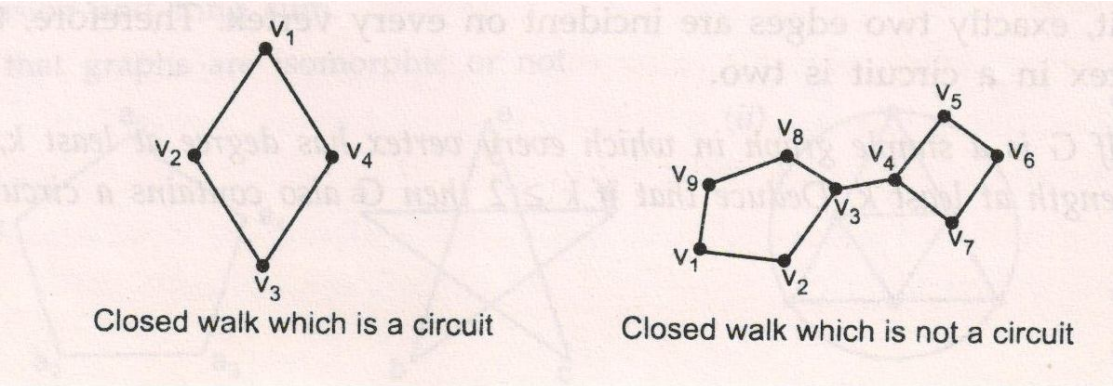
Fig 3.11

All the graphs in the fig 3.11 are circuit. A circuit is also called a ***cycle, elementary cycle, circular path*** and ***polygon***.

- Every self-loop is a circuit, but not every circuit is a self-loop.
- In walks, paths and circuit, no edge can appear more than once
- A path is an open walk, but open walks need not to be a path. E.g.



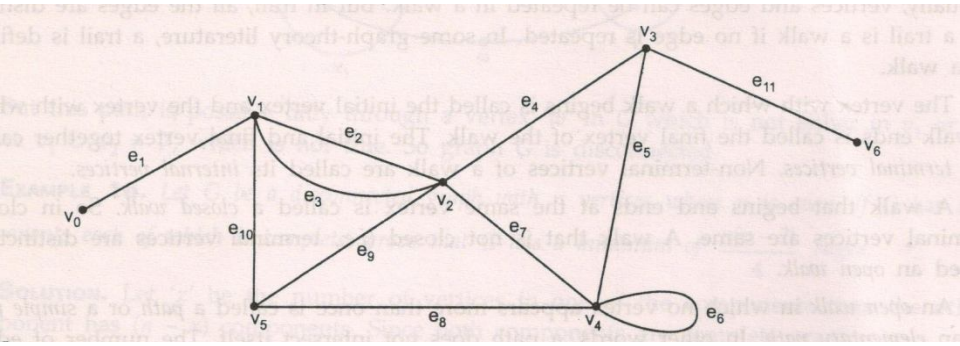
- A vertex can appear more than once in a walk but not in a path
- A circuit is a closed walk, but a closed walk needs not to be a circuit. E.g.



Examples:

In the graph below, determine whether the following are paths, simple paths, walk or circuits.

1. v0 e1 v1 e10 v5 e9 v2 e2 v1
2. v5 v2 v3 v4 v4 v4 v5
3. v4 e7 v2 e9 v5 e10 v1 e3 v2 e9 v5



Solution

1. The sequence has a repeated vertex v1, but does not have a repeated edge so it is a walk.
2. It is a circuit since it has no repeated edge, starts an end at same vertex. It is not a cycle because v4 is repeated.

3. The sequence has a repeated vertex and repeated edge hence it is not a path, cycle or circuit.

Example:

Prove the following:

1. A path with ***n*** vertices is of length ***n – 1***.
2. If a circuit has ***n*** vertices, it has ***n*** edges.
3. The degree of every vertex in a circuit is two.

Solution

1. In a path, except the last is followed by precisely once edge. Therefore, if a path has n vertices, it must have (n – 1) edges and its length is (n – 1).
2. In a circuit, every vertex is followed by precisely one edge. Therefore, if a circuit has n vertices, it must have n edges.
3. In a circuit, exactly two edges are incident on every vertex. Therefore, the degree of every vertex in a circuit is two.

6.4 Euler Graph

If some closed walk in a graph contains all edges of the graph, then the walk is called an **Euler line** and the graph an **Euler graph**.

A path in a graph G is called Euler path if it includes every edges exactly once. Since the path contains every edge exactly once, it is also called Euler tail. An Euler path that is a circuit is called Euler circuit. So a graph which has an Eulerian circuit is called an Eulerian graph.

By definition, a walk is always connected. Since the Euler line contains all the edges of the graph, an Eulerian graph is always connected.

6.4.1 Properties of Euler graph

- A connected undirected graph is Eulerian if every graph vertex has an even degree.
- An undirected graph is Eulerian if it is connected and can be decomposed into edge-disjoint cycles.
- If an undirected graph G is Eulerian then its line graph L(G) is Eulerian too.
- A directed graph is Eulerian if it is connected and every vertex has equal indegree and outdegree.
- A directed graph is Eulerian if it is connected and can be decomposed into edge disjoined directed cycles.

6.4.2 Theorem:

A given connected graph G is an Euler graph if and only if all vertices of G are of even degree.

Proof:

Suppose that **G** is an Euler graph. Euler graph is a closed walk. In tracing this walk, we observe that every time the walk meets a vertex ***v*** it goes through two new edges incident on ***v*** – with one we "entered" and with the other "exited". This is true not only of all intermediate vertices of the walk but also of the terminal vertex, because we "exited" and "entered" the same vertex at the beginning and end of the walk respectively. Thus if **G** is an Euler graph, the degree of every vertex is even.

6.4.3 Theorem:

A connected graph G is an Euler graph if and only if it can be decomposed into circuit.

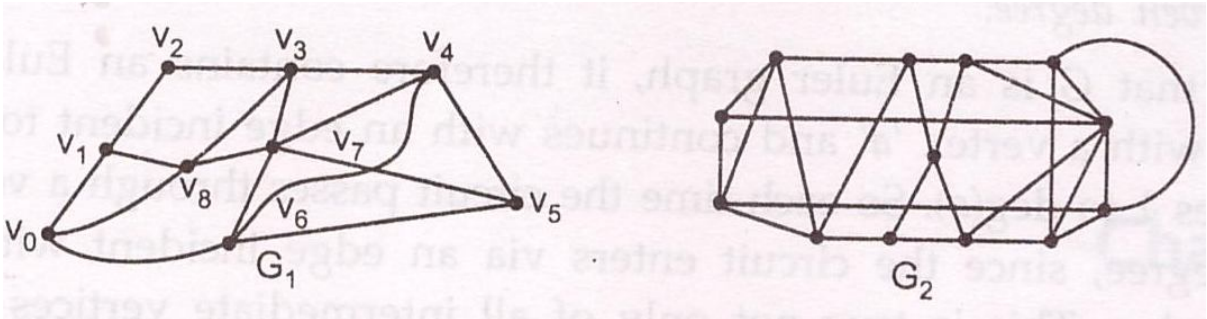
Proof:

Suppose graph **G** can be decomposed into circuit; that is, **G** is a union of edge-disjoint circuits. Since the degree of every vertex in a circuit is two, the degree of every vertex in **G** is even. Hence G is an Euler graph.

Conversely, let **G** be an Euler graph, consider a vertex v_1. There are at least two edges incident at v_1. Let one of these edges is between v_1 and v_2. Since vertex v_2 is also of even degree, it must have at least another edge, say between v_2 and v_3. Proceeding in this way, we eventually arrives at a vertex that has previously been traversed, thus forming a circuit **T**. Let us remove **T** from **G**. All vertices in the remaining graph must also be of even degree. From the remaining graph remove another circuit in exactly the same way as we removed **T** from **G**. Continue this process until no edges are left. Hence the theorem.

Example:

Decide which of the following graphs shown in the figure have an Euler circuit, Euler trail or neither.



Solution:

- In graph G1, more than two vertices are odd degree. So this graph is neither Euler circuit nor Euler trail.
- The graph G2 has an Euler trail since there are only two vertices of odd degree

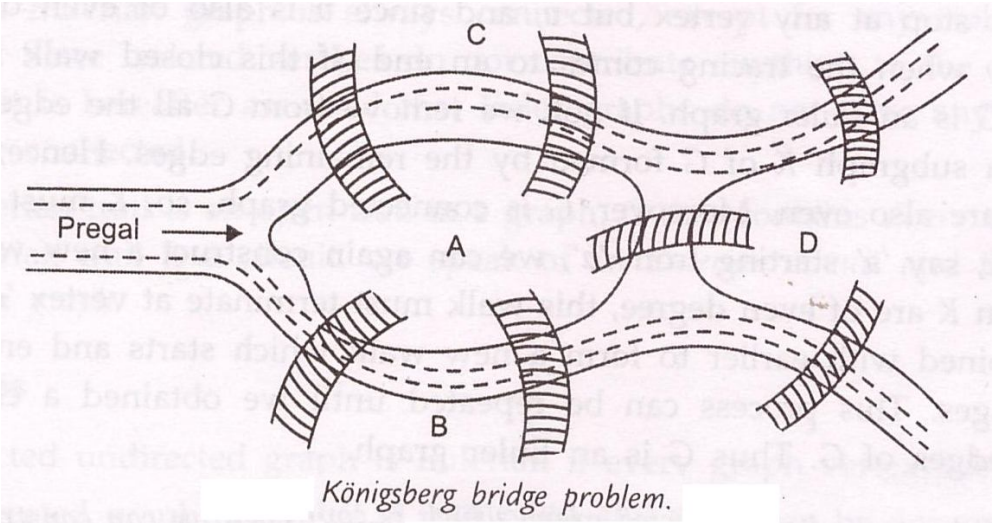
Points to be remember:

- If no. of odd degree is 0 then there is at least one Euler circuit.
- If no. of odd degree is 1 then such a graph is not possible.
- If no. of odd degree is 2 then there is no Euler circuit but at least one Euler trail.
- If no. of odd degree is more than 2 then there are no Euler circuit or Euler trail

Thus the theorem of Euler is "A connected multigraph G contains an Eulerian trail, if the number of vertices 'n' with odd degree is either 0 or 2. If n = 0, there exists an Eulerian circuit".

6.5 Koingsberg Bridge Problem

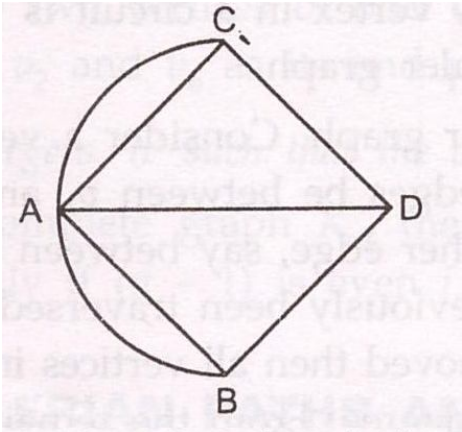
Given is the inner city of koingsberg with river Pregel, which divides the city into four land regions A, B, c and D. In order to travel from one part of the city to another one there exists 7 bridges.



Königsberg bridge problem.

The problem was to start at any of the four land areas of the city A, b, c and D, walk over each of the seven bridges exactly once, and return to the starting point (without swimming across the river).

This problem was solved by Leonhard Euler in 1736 using graph as shown in the graph given below.

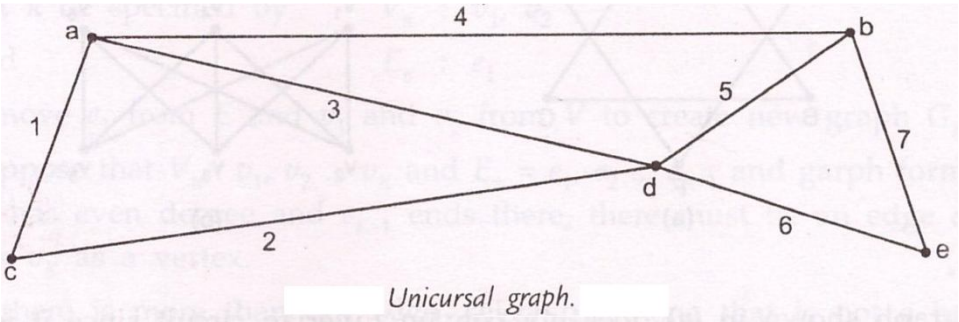


Graph of königsberg bridge.

The vertices represented the land areas and the edges represented the bridges. Euler proved that a solution for this problem does not exist. Looking to the graph we find that not all its vertices are of even degree. Hence it is not an Euler graph. So it is not possible to walk over each of the seven bridges exactly once and return to the starting point.

6.6 Unicursal Graph

For a graph to be Euler, it should be closed. For example



Unicursal graph.

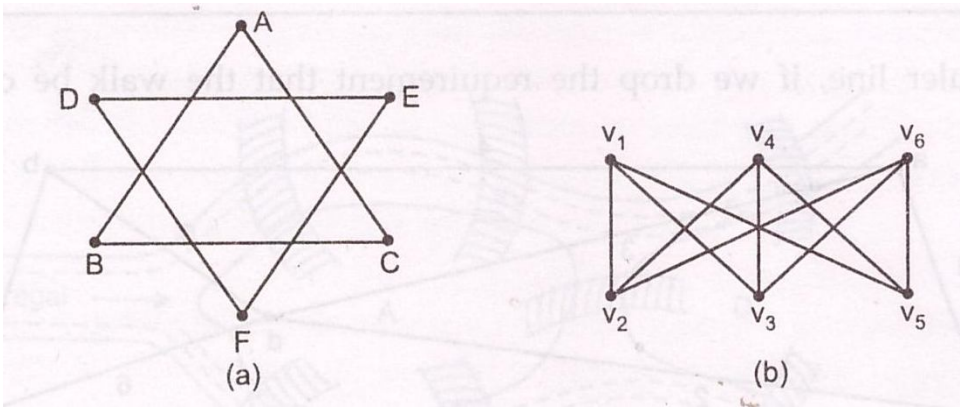
The walk a 1 c 2 d 3 a 4 b 5 d 6 e 7 b in the figure above includes all the edges of the graph and does not retrace any edge, is not closed. In this the initial vertex is ‘a’ and the final vertex is ‘b’.

An open walk that includes all edges of a graph without retracing any edge is called **unicursal line** or an **open Euler line**. A graph that has a unicursal line is called a unicursal graph.

By adding an edge between the initial and final vertices of a unicursal line, we will get Euler line. Thus a connected graph is unicursal if and only if it has exactly two vertices of odd degree.

Example

Show that the graphs in figure below contain no Eulerian circuit.

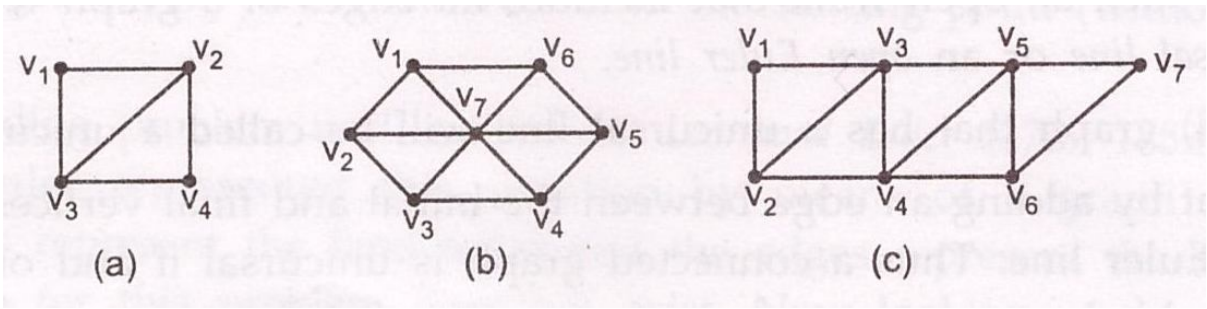


Solution

- The graph (a) does not contain Eulerian circuit since it is not connected.
- In the graph (b) all the vertices are of degree 3. Hence it does not contain Eulerian circuit.

Example:

Which graphs shown in figure have an Euler path?



Solution:

- In fig (a), graph contains exactly two vertices v2 and v3 of odd degree. Hence, it has an Eulerian path that must have v2 and v3 as its end point.
- In fig (b), the graph has no Euler path since it has six vertices of odd degree.
- In fig (c), graph has exactly two vertices of odd degree, v2 and v6. Hence it has an Euler path that must have v2 and v6 as its end points.

Example:

Find all positive integers 'n' such that the complete graph Kn is Eulerian.

Solution:

We know that in the complete graph K_n, the degree of every vertex is n – 1. Therefore K_n is Eulerian if and only if (n – 1) is even i.e. if and only if n is odd.

6.7 Constructing Eulerian Paths and Circuits

Consider a connected graph with at most two vertices of odd degree. We can construct an Eulerian path (not a cycle) using Fleury's algorithm.

We start with a vertex of odd degree. If graph has none then start with any vertex. At each step move across an edge whose deletion does not result in more than one connected component and delete the edge. At the end of the algorithm there are no edges left, and the sequence of edges moved across forms an Eulerian cycle if the graph has no vertices of odd degree or an Eulerian path if there are two vertices of odd degree.

Fleury's Algorithm

Let G = (V, E) be a connected graph with each vertex of even degree.

Step1:

Select an edge e_1 that is not a bridge in graph **G**. Let the end vertices of e_1 be v_1 and v_2.

Remove e_1 from v_1 & v_2 to create new graph **G1**.

Step2:

Suppose that $V = v_1, v_2 ... v_k$ and $E = e_1, e_2 ... e_{k-1}$ and graph form is G_{k-1}. Since v_k has even degree and e_{k-1} ends there, there must be an edge e_k in G_{k-1} that has v_k as a vertex.

If there is more than one such edge, select one that is not a bridge for G_{k-1}. Suppose e_k has v_k and v_{k+1} vertices then

$V = v_1\ v_2\ ...\ v_k, v_{k+1}$

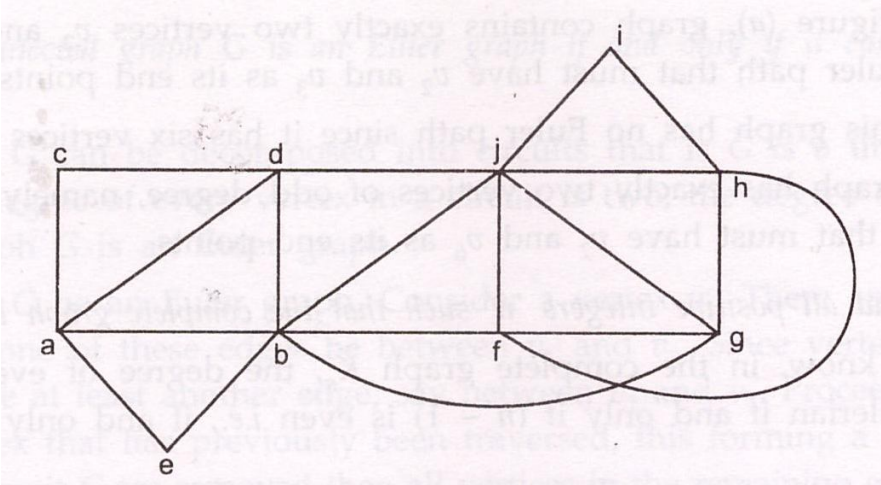
$E = e_1\ e_2\ ...\ e_{k-1}, e_k$

Step3:

Repeat step2 until no edges remain in E and stop.

Example:

Using Fleury's algorithm, find Euler circuit in the graph.



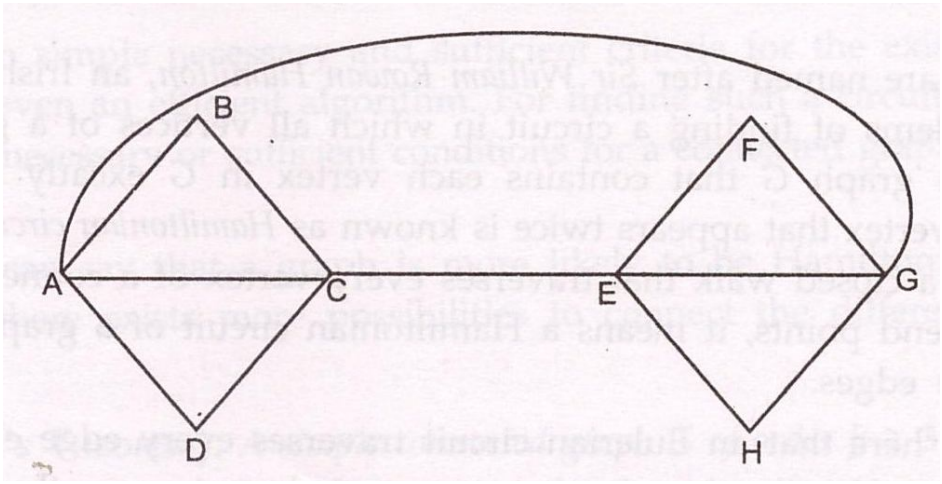
Solution:

Degree of all vertices are even. There exist an Eulerian circuit in it. Arbitrarily choose vertex 'a'.

Current Path	Next Edge
V: a	{a, b}
V: a, b	{b, d}
V: a, b, d	{d, c}
V: a, b, d, c	{c, a}
V: a, b, d, c, a	{a, e}
V: a, b, d, c, a, e	{e, b}
V: a, b, d, c, a, e, b	{b, f}
V: a, b, d, c, a, e, b, f	{f, g}
V: a, b, d, c, a, e, b, f, g	{g, h}
V: a, b, d, c, a, e, b, f, g, h	{h, f}
V: a, b, d, c, a, e, b, f, g, h, f	{f, j}
V: a, b, d, c, a, e, b, f, g, h, f, j	{j, b}
V: a, b, d, c, a, e, b, f, g, h, f, j, b	{b, g}
V: a, b, d, c, a, e, b, f, g, h, f, j, b, g	{g, j}
V: a, b, d, c, a, e, b, f, g, h, f, j, b, g, j	{j, h}
V: a, b, d, c, a, e, b, f, g, h, f, j, b, g, j, h	{h, i}
V: a, b, d, c, a, e, b, f, g, h, f, j, b, g, j, h, i	{i, j}
V: a, b, d, c, a, e, b, f, g, h, f, j, b, g, j, h, i, j	{j, d}
V: a, b, d, c, a, e, b, f, g, h, f, j, b, g, j, h, i, j, d	{d, a}
V: a, b, d, c, a, e, b, f, g, h, f, j, b, g, j, h, i, j, d, a	This is Euler circuit

Example:

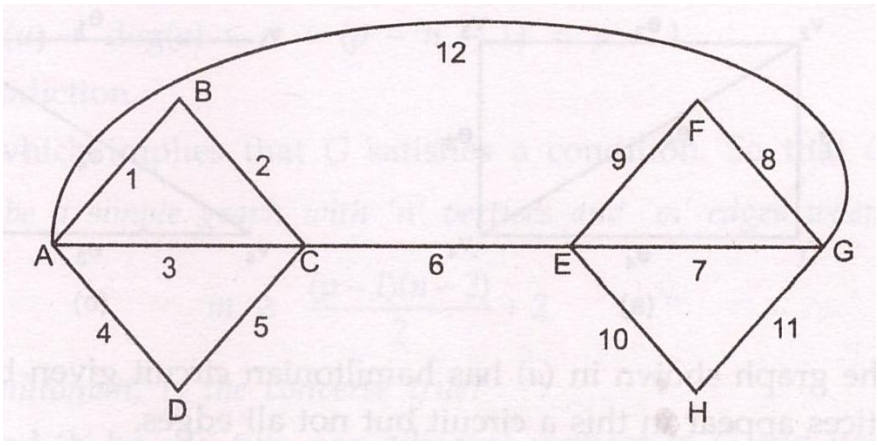
Using Fleury's algorithm, construct an Euler circuit for the graph.



Solution:

Suppose we start from vertex A.

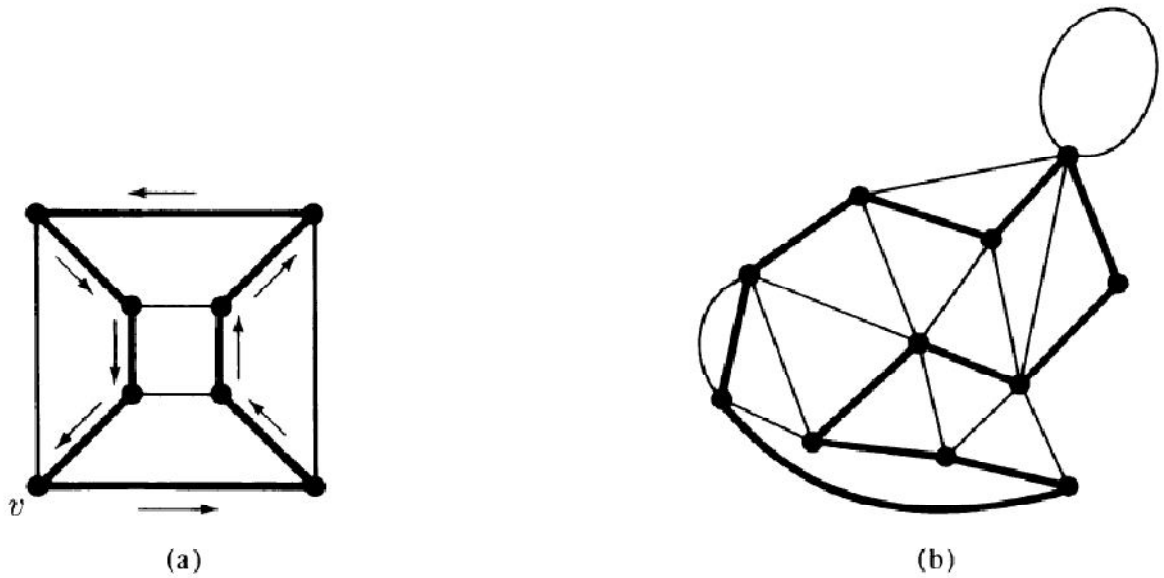
Current Path	Next Edge
V: A	{A, B}
V: A, B	{B, C}
V: A, B, C	{C, A}
V: A, B, C, A	{A, D}
V: A, B, C, A, D	{D, C}
V: A, B, C, A, D, C	{C, E}
V: A, B, C, A, D, C, E	{E, G}
V: A, B, C, A, D, C, E, G	{G, F}
V: A, B, C, A, D, C, E, G, F	{F, E}
V: A, B, C, A, D, C, E, G, F, E	{E, H}
V: A, B, C, A, D, C, E, G, F, E, H	{H, G}
V: A, B, C, A, D, C, E, G, F, E, H, G	{G, A}
V: A, B, C, A, D, C, E, G, F, E, H, G, A	This is Euler Circuit



6.8 Hamiltonian Graph

A Hamiltonian circuit in a connected circuit is a closed walk that traverses every vertex of G exactly once except for the terminal vertices i.e. starting and end vertex. Hamiltonian circuit of graph having ***n*** vertices consists of exactly ***n*** edges.

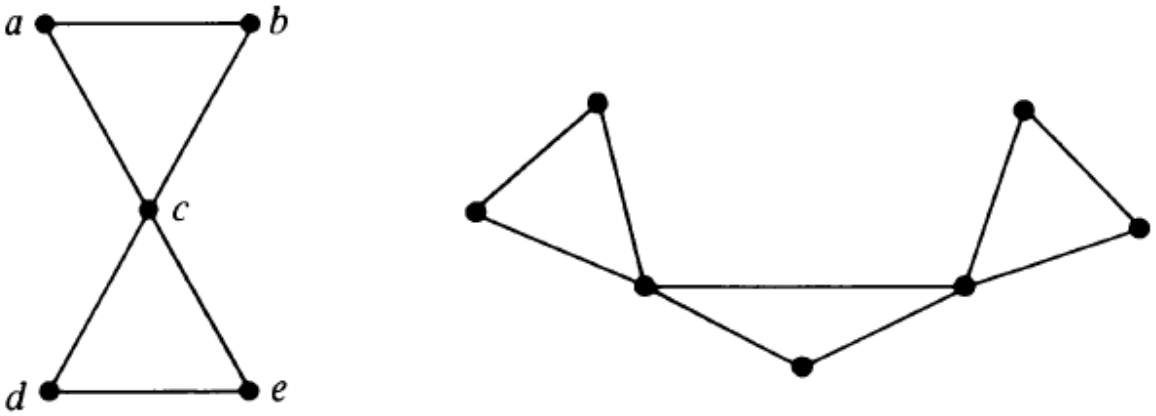
An Eulerian circuit traverses every edge exactly once with vertices may be repeated while Hamiltonian circuit traverses each vertex exactly once with edges may be repeated.



Hamiltonian Circuit

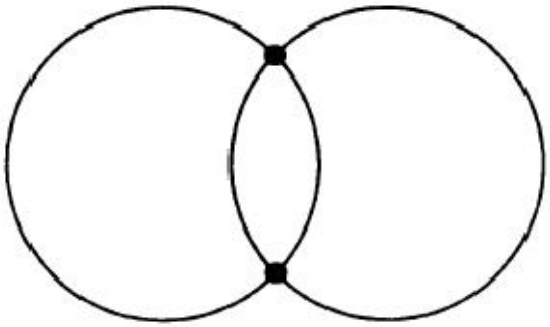
In the graph (a) starting vertex is v, traversing is shown by heavy line and one can traverse every vertex of the graph exactly once to get Hamiltonian circuit. A Hamiltonian circuit is also shown in the graph (b) by heavy lines.

Not every connected graph has Hamiltonian circuit. For example the two graphs shown below are not Hamiltonian circuit.



Example:

Is the graph shown below is a Hamiltonian circuit?



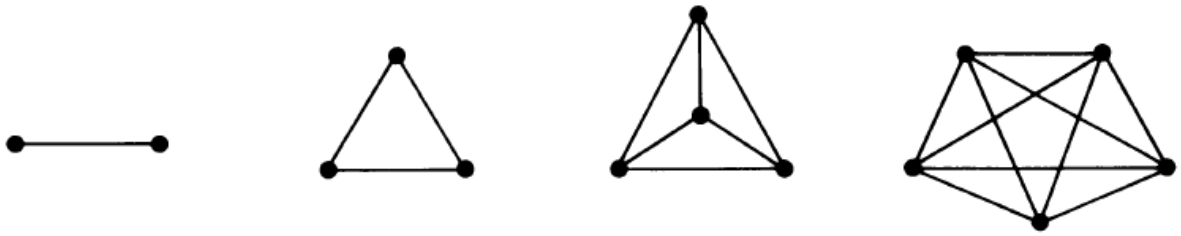
6.8.1 Hamiltonian Path

If we remove any one edge from a Hamiltonian circuit, we are left with a path. This path is called ***Hamiltonian path***. Hamiltonian path in a graph G traverses every vertex of G. Since Hamiltonian path is a subgraph of a Hamiltonian circuit, every graph that has a Hamiltonian circuit also has a Hamiltonian path. However it is not necessary that every Hamiltonian path may have Hamiltonian circuit. The length of a Hamiltonian path in a connected graph of ***n*** vertices is ***n – 1***.

Hamiltonian circuit (or path) traverses every vertex exactly once. Hence it cannot include a self loop or a set of parallel edges. Thus we required simple graphs to find the existence of a Hamiltonian circuit (or path).

6.8.2 Hamiltonian circuit from complete graph

A simple graph in which there exists an edge between every pair of vertices is called a ***complete graph***. Complete graph of two, three, four and five vertices are shown below:



Complete graphs of two, three, four, and five vertices.

Complete graph is also referred to as a ***universal graph***. Since every vertex is joined with every other vertex through one edge, the degree of every vertex is ***n – 1*** in a complete graph of ***n*** vertices. Also the total number of edges in G is ***n(n – 1)/2.***

It is easy to construct a Hamiltonian circuit in a complete graph of n vertices. Let the vertices be numbered $v_1, v_2, ..., v_n$. Since an edge exist between any two vertices, we can start from v_1

and traverse to v_2, and v_3, and so on to v_n, and finally from v_n to v_1. This is a Hamiltonian circuit

6.8.3 Number of Hamiltonian circuits in a graph

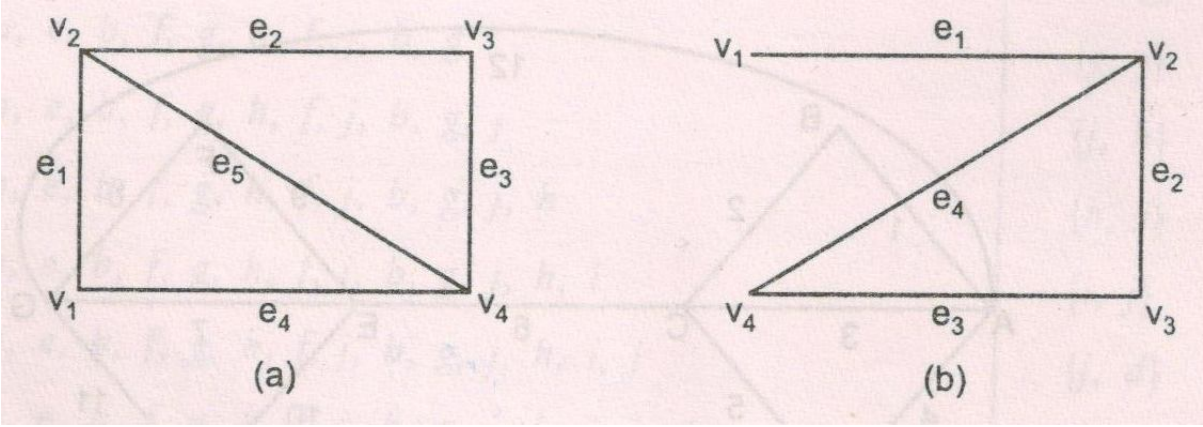
A graph may contain more than one Hamiltonian circuit and all are known as edge disjoint Hamiltonian circuits in a graph. In a complete graph with ***n*** vertices there are ***(n – 1)/2*** edge-disjoint Hamiltonian circuit, if n is an odd number >=3.

A complete graph G of ***n*** vertices has ***n(n – 1)/2*** edges, and a Hamiltonian circuit in **G** consist of ***n*** edges. Therefore, the number of edge-disjoint Hamiltonian circuits in **G** cannot exceed ***(n – 1)/2.*** There are ***(n – 1)/2*** edge-disjoint Hamiltonian circuits when ***n*** is odd.

- A sufficient condition (but not necessary) for a simple graph **G** to have a Hamiltonian circuit is that degree of every vertex in **G** be at least ***n/2*** where ***n*** is the number of vertices in **G.**
- If a given graph **G** has ***n*** vertices then Hamiltonian path and Hamiltonian circuit must contain (n – 1) and n edges respectively.
- If a vertex v in graph G has degree ***'m'*** then a Hamiltonian path must contain at least one edge incident on ***v*** and at most two edges incident on ***v*** but Hamiltonian circuit contains exactly two edges incident on ***v***.
- Once the Hamiltonian circuit we are building has passed through a vertex v, then all other unused edged incident on v can be deleted because only two edges incident on v can be included in a Hamiltonian circuit.

Example1:

Which of the graphs shown are Hamiltonian circuits?



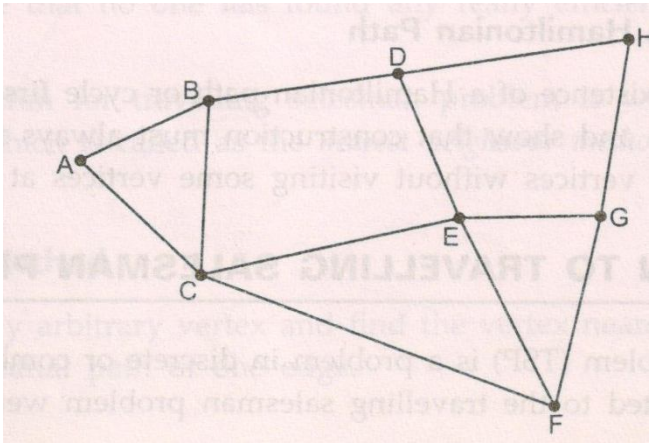
Solution:

The graph shown in fig (a) has Hamiltonian circuit given by v1 e1 v2 e2 v3 e3 v4 e4 v1.

The graph shown in fig (b) is not a Hamiltonian circuit as it is not a circuit.

Example2:

Determine whether a Hamiltonian circuit or path exists in the graph given below:

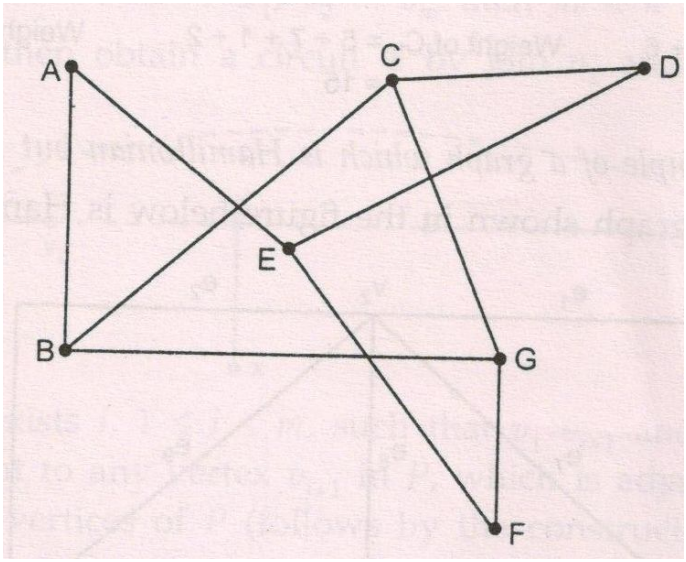


Solution

Starting from A. Move like this A B D H G F E C A. It is a Hamiltonian circuit.

Example3:

In the graph given below, find it form Hamiltonian circuit and Hamiltonian path.



Traversing graph in the sequence

A B G F E D C B A	NO Hamiltonian circuit
A E F G C D E	NO Hamiltonian circuit
A B G F E D C	Hamiltonian path

6.9 The Travelling Salesman Problem (TSP)

The Travelling Salesman Problem states that "A salesman is required to visit a number of cities during a trip. He has to travel every city exactly once and return back to the starting city. Given the distance between the cities, what order should he travel to cover minimum mileage?"

We can get a graph by representing city by vertex and the roads between them by edges. Here in graph, every edge is associated with a weight i.e. $w(e_i)$ that represents the distance between the two vertex (cities). Since salesman has to travel every city i.e. vertex exactly once, the graph in the TSP is closely related to Hamiltonian circle. The objective of Travelling Salesman Problem is to find a Hamiltonian circle of minimum total weight.

Many TSP's are symmetric, i.e. the distance between the two cities A and B is symmetric. The distance from A to B is the same as that from B to A. The travel cost will be symmetric. For asymmetric TSP, there will be **(n – 1)!** different tours for **n** cities. For first city, there are (n – 1) choices for second city visited, (n – 2) choices for the third, and so on. For symmetric case there are half as many distinct solution i.e. **(n – 1)!/2** for n city TSP which is total number of different Hamiltonian circuits in a complete graph of ***n*** vertices because Hamilton circuit has been counted twice.

To find the shortest path for travelling salesman, we can use ***nearest neighbour method*** to solve the problem.

Nearest Neighbour Method

Step1: Choose any arbitrary vertex and find the vertex nearest to this vertex to form an initial path of one edge.

Step2: Add the path obtained in Step1 in a vertex by vertex.

Step3: Let k be the latest vertex that has been added to the path. Now among the rest of the vertices, which have not picked up so far, pick the closest one to k and add to the path. Repeat the process until all the vertices of the given graph G is included in the path.

Step4: Join starting and the last vertex by an edge and form a circuit.

This method is also called ***greedy method*** as it selects the shortest path from a vertex every time and doesn't care whether this will lead to a wrong result or not.

Example:

A newspaper agent daily drops the newspaper to the area assigned in such a manner that he has to cover all the houses in the respective area with minimum travel cost. Compute the minimum travel cost. The area assigned to the agent where he has to drop the newspaper is shown in the figure.

Solution:

The cost-adjacency matrix of the graph G is as follows:

	H1	H2	H3	H4	H5	H6	H7	H8
H1	0	5	0	6	0	4	0	7
H2	5	0	2	4	3	0	0	0
H3	0	2	0	1	0	0	0	0
H4	6	4	1	0	7	0	0	0
H5	0	3	0	7	0	0	6	4
H6	4	0	0	0	0	0	3	0
H7	0	0	0	0	6	3	0	2
H8	7	0	0	0	4	0	2	0

	H1	H2	H3	H4	H5	H6	H7	H8
(H1)	0	5	0	6	0	(4)	0	7
H2	5	0	2	4	3	0	0	0
H3	0	2	0	1	0	0	0	0
H4	6	4	1	0	7	0	0	0
H5	0	3	0	7	0	0	6	4
H6	4	0	0	0	0	0	3	0
H7	0	0	0	0	6	3	0	2
H8	7	0	0	0	4	0	2	0

Mark area H6 because it is the minimum cost area reachable from area H1 then select minimum cost area reachable from H6.

	H1	H2	H3	H4	H5	H6	H7	H8
(H1)	0	5	0	6	0	(4)	0	7
H2	5	0	2	4	3	0	0	0
H3	0	2	0	1	0	0	0	0
H4	6	4	1	0	7	0	0	0
H5	0	3	0	7	0	0	6	4
(H6)	4	0	0	0	0	0	(3)	0
H7	0	0	0	0	6	3	0	2
H8	7	0	0	0	4	0	2	0

Mark area H7 because it is the minimum cost area reachable from area H6 and then selects minimum cost area reachable from H7.

	H1	H2	H3	H4	H5	H6	H7	H8
(H1)	0	5	0	6	0	(4)	0	7
H2	5	0	2	4	3	0	0	0
H3	0	2	0	1	0	0	0	0
H4	6	4	1	0	7	0	0	0
H5	0	3	0	7	0	0	6	4
(H6)	4	0	0	0	0	0	(3)	0
(H7)	0	0	0	0	6	3	0	(2)
H8	7	0	0	0	4	0	2	0

Mark area H8 and select minimum cost area reachable from H8

	H1	H2	H3	H4	H5	H6	H7	H8
(H1)	0	5	0	6	0	(4)	0	7
H2	5	0	2	4	3	0	0	0
H3	0	2	0	1	0	0	0	0
H4	6	4	1	0	7	0	0	0
H5	0	3	0	7	0	0	6	4
(H6)	4	0	0	0	0	0	(3)	0
(H7)	0	0	0	0	6	3	0	(2)
(H8)	7	0	0	0	(4)	0	2	0

Mark area H5 and select minimum cost area reachable from H5.

	H1	H2	H3	H4	H5	H6	H7	H8
(H1)	0	5	0	6	0	(4)	0	7
H2	5	0	2	4	3	0	0	0
H3	0	2	0	1	0	0	0	0
H4	6	4	1	0	7	0	0	0
(H5)	0	(3)	0	7	0	0	6	4
(H6)	4	0	0	0	0	0	(3)	0
(H7)	0	0	0	0	6	3	0	(2)
(H8)	7	0	0	0	(4)	0	2	0

Mark area H2 and select minimum cost area reachable from H2

	H1	H2	H3	H4	H5	H6	H7	H8
H1	0	5	0	6	0	4	0	7
H2	5	0	2	4	3	0	0	0
H3	0	2	0	1	0	0	0	0
H4	6	4	1	0	7	0	0	0
H5	0	3	0	7	0	0	6	4
H6	4	0	0	0	0	0	3	0
H7	0	0	0	0	6	3	0	2
H8	7	0	0	0	4	0	2	0

Mark area H3 and select minimum cost area reachable from H3.

	H1	H2	H3	H4	H5	H6	H7	H8
H1	0	5	0	6	0	4	0	7
H2	5	0	2	4	3	0	0	0
H3	0	2	0	1	0	0	0	0
H4	6	4	1	0	7	0	0	0
H5	0	3	0	7	0	0	6	4
H6	4	0	0	0	0	0	3	0
H7	0	0	0	0	6	3	0	2
H8	7	0	0	0	4	0	2	0

Mark area H4 and select minimum cost area reachable from H4 and its H1.

	H1	H2	H3	H4	H5	H6	H7	H8
H1	0	5	0	6	0	4	0	7
H2	5	0	2	4	3	0	0	0
H3	0	2	0	1	0	0	0	0
H4	6	4	1	0	7	0	0	0
H5	0	3	0	7	0	0	6	4
H6	4	0	0	0	0	0	3	0
H7	0	0	0	0	6	3	0	2
H8	7	0	0	0	4	0	2	0

So, using the greedy strategy we get the following:

H1 → (4) H6 → (3) H7 → (2) H8 → (4) H5 → (3) H2 → (2) H3 → (1) H4 → (6) H1

Thus, the minimum travel cost = 4 + 3 + 2 + 4 + 3 + 2 + 1 + 6

= 25