Walk:

- A walk can be defined as a sequence of edges and vertices of a graph.
- When we have a graph and traverse it, then that traverse will be known as a walk.
- In a walk, there can be repeated edges and vertices.
- The number of edges which is covered in a walk will be known as the Length of the walk. In a graph, there can be more than one walk.

For a walk, the following two points are important,

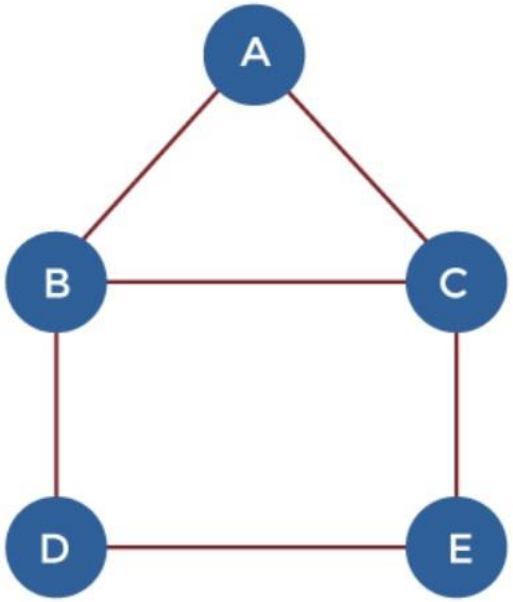
Edges can be repeated

Vertex can be repeated

For example: In this example, we have a graph, which is described as follows:

In the below graph, there can be many walks, but some of them are described as follows

Length of path = no. of edges in the path



1. A, B, C, E, D (Number of length = 4)
2. D, B, A, C, E, D, C (Number of length = 7)
3. E, C, B, A, C, E, D (Number of length = 6)

Types of Walks

There are two types of the walk, which are described as follows:

Open walk

Closed walk

Open Walk:

- A walk will be known as an open walk in the graph theory if the vertices at which the walk starts and ends are different. That means for an open walk, the starting vertex and ending vertex must be different.
- In an open walk, the length of the walk must be more than 0.

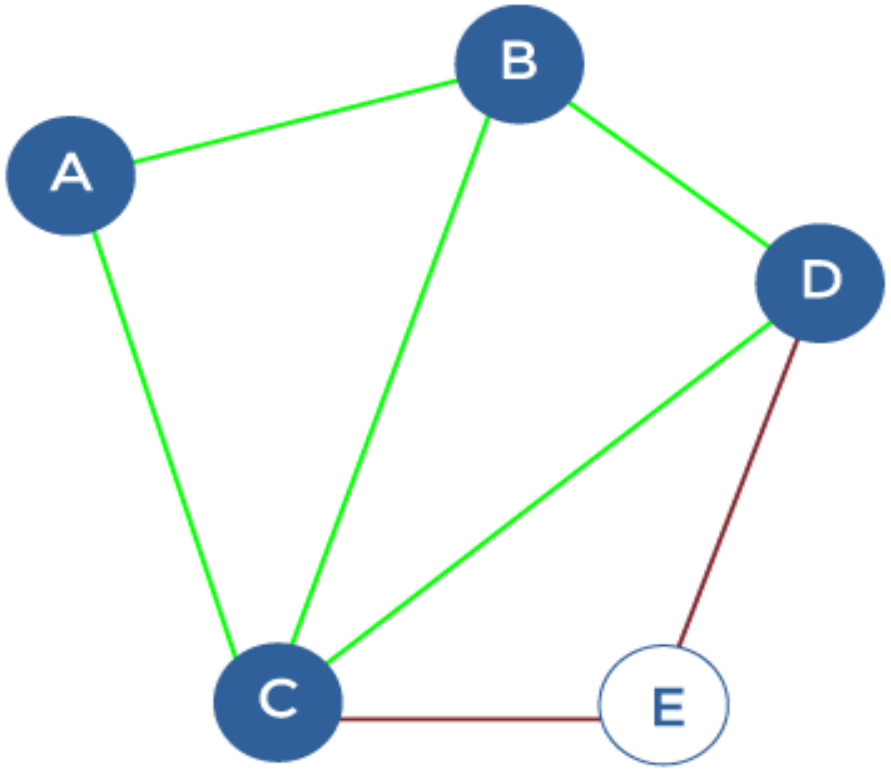
Closed Walk:

- A walk will be known as a closed walk in the graph theory if the vertices at which the walk starts and ends are identical.
- That means for a closed walk, the starting vertex and ending vertex must be the same. In a closed walk, the length of the walk must be more than 0.

In this graph, there is also a closed walk and an open walk, which are described as follows:

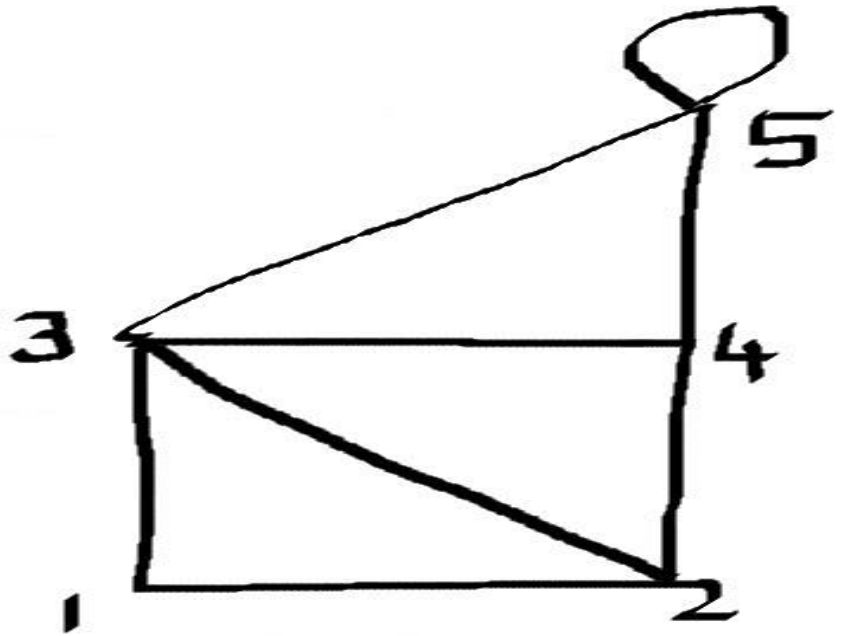
Closed walk = A, B, C, D, E, C, A

Open walk = A, B, C, D, E, C

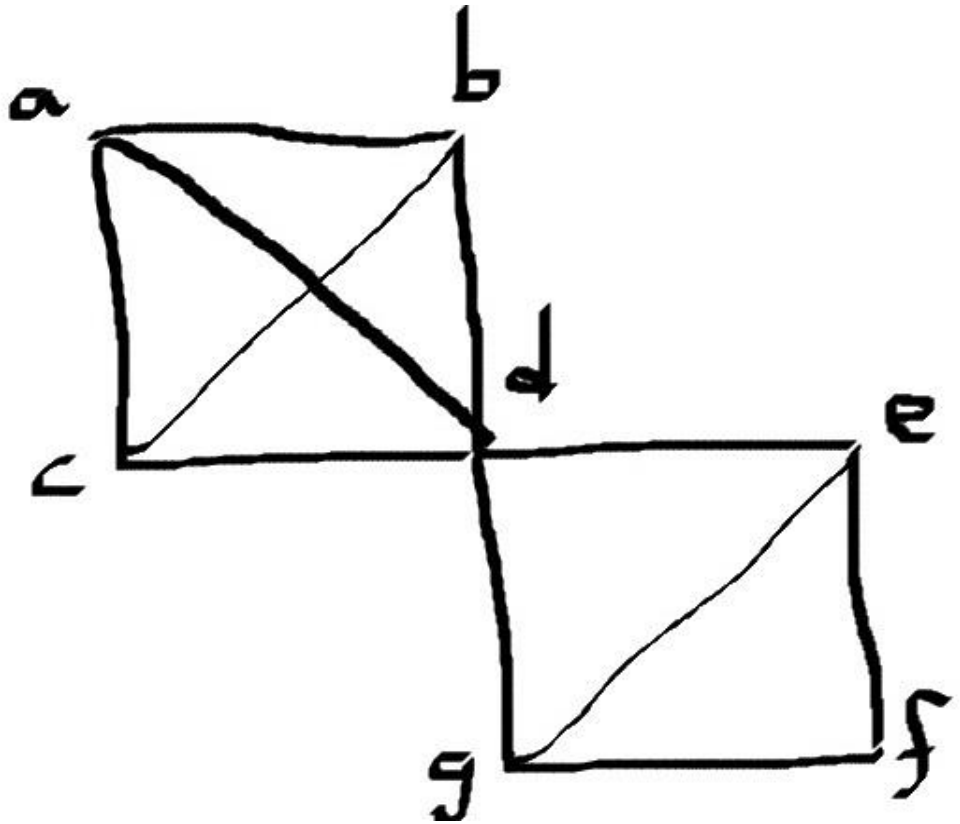


WALK

Example Closed walk =	1-2-3-4-2-1 1-2-3-1
Open walk =	1-2-3-4 1-2-3-4-2



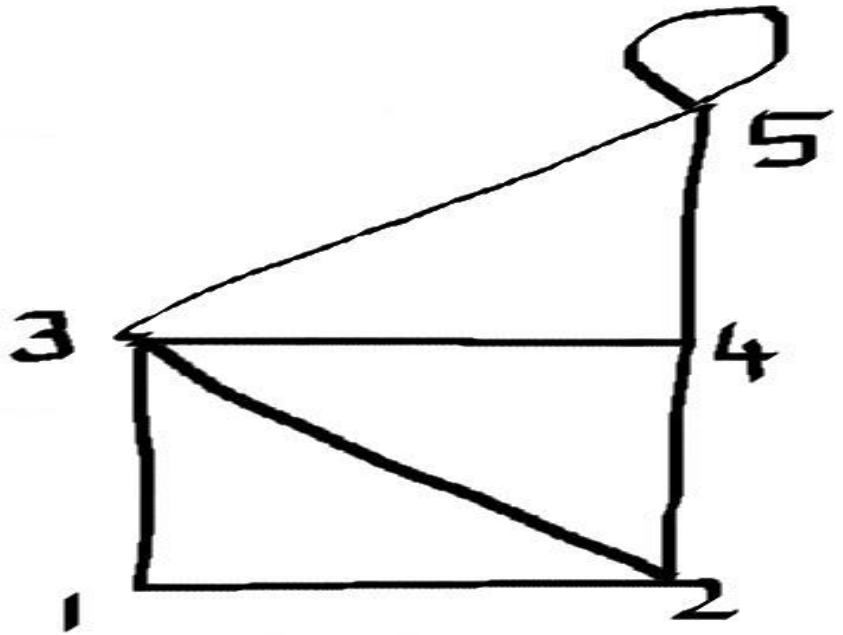
Sequence	Walk/ Not	Length
a,d,e,f,g	Yes	4
a,e,f,g	No	-
a,d,c,a,b,c	Yes	5
a,d,c,b,d,c	Yes	5



PATH

An open walk in which no vertex appears more than once is called a path

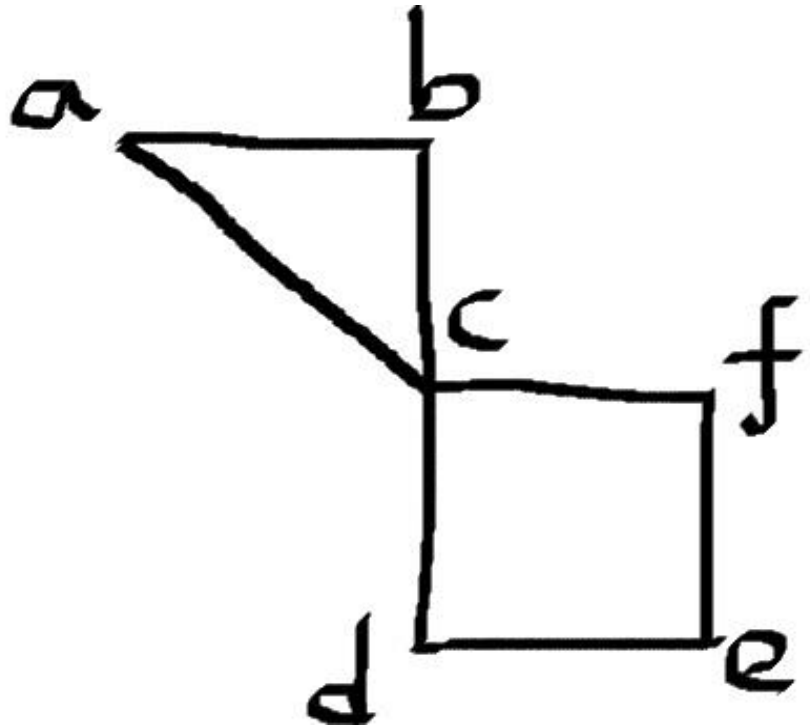
Sequence	Walk	Path
1-2-4-5	Open	Yes
5-3-1-2-3-4	open	No
5-5-4	open	no
5-4-3-5	closed	no



Simple path or Trial

Path without repetition of edges is called as trial or simple path.
A walk is a trial if all its edges are distinct.
Vertices can be repeated but not edges

Path	Simple/Not
a,c,d,e,f	Yes
a,c,d,e,f,c	Yes
a,d,c,a,b,c	No

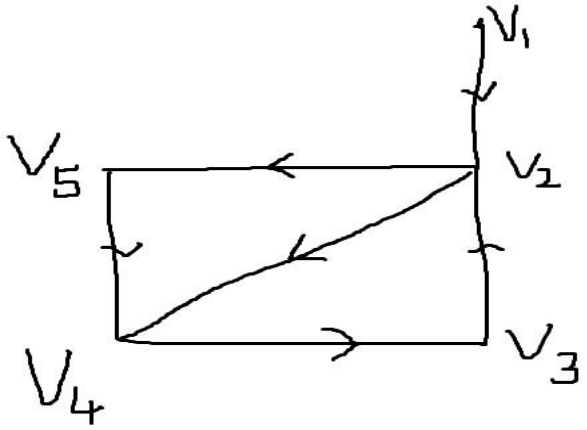


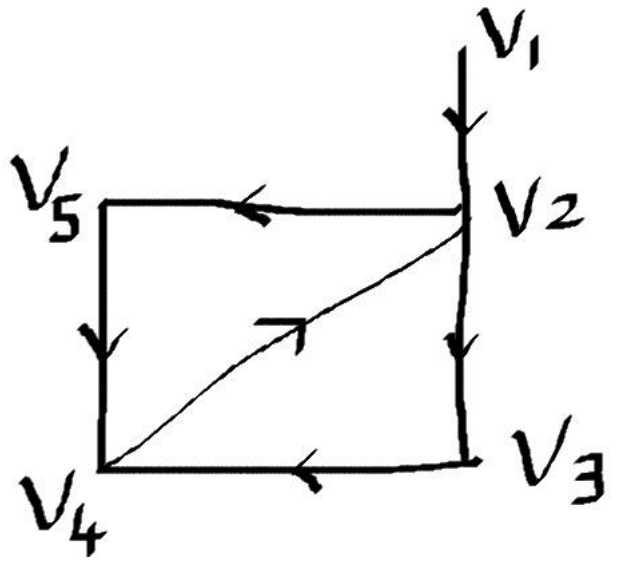
Simple path or Trial

Path without repetition of edges is called as trial or simple path.

A walk is a trial if all its edges are distinct. Vertices can be repeated but not edges

Path	Simple/Not
V1-V2-V4-V3-V2-V5-V4	YES
V1-V2-V3-V4-V2-V5-V4	YES





Dr.V.K.Ajay

Simple Circuit or Cycle

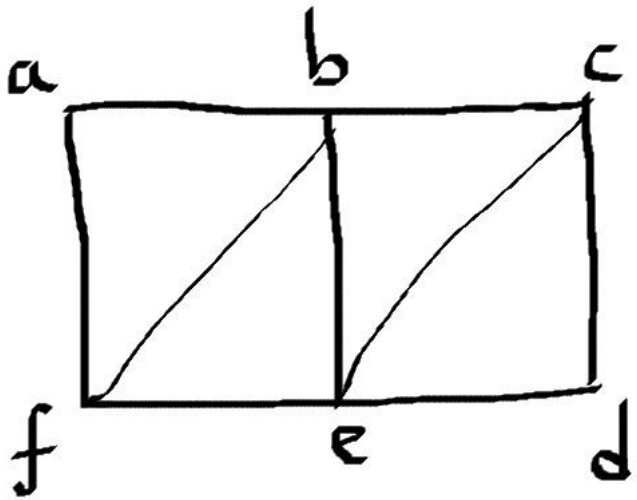
A Circuit is a closed trail. That is, a circuit has no repeated edges but may have repeated vertices.

Traversing a graph such that we do not repeat a vertex nor we repeat a edge but the starting and ending vertex must be same i.e. we can repeat starting and ending vertex only then we get a cycle.

Vertex not repeated

Edge not repeated

Path	Simple circuit /Not
a,b,f,a	YES (Cycle)
a,b,f,a,b,e,f,a	circuit not cycle
c,d,e,b,c	yes



Dr.V.K.Ajay

SUMMARY

REPETETION		
	VERTEX	EDGES
WALK	YES	YES
TRIAL	YES	NO
CIRCUIT	YES	NO
PATH	NO	NO
CYCLE	NO	NO

Dr.V.K.Ajay

Euler Path and Euler Circuit

Eulerian Path is a path in a graph that visits every edge exactly once.

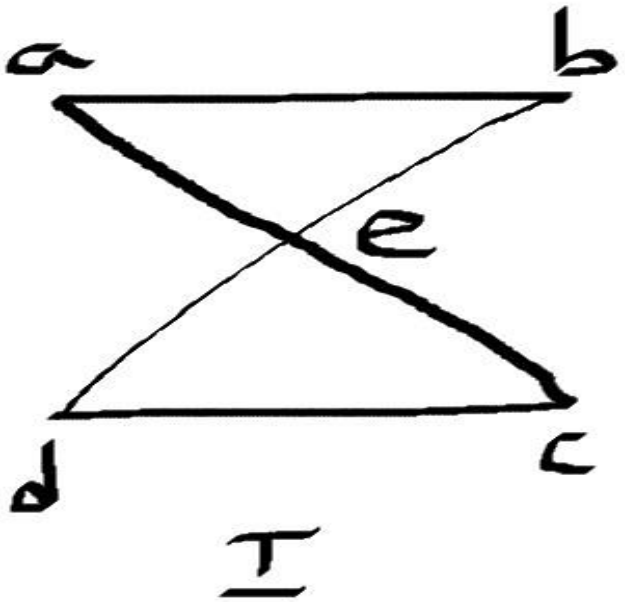
Eulerian Circuit is an Eulerian Path that starts and ends on the same vertex.

Euler Path: A simple path containing every edge of a graph

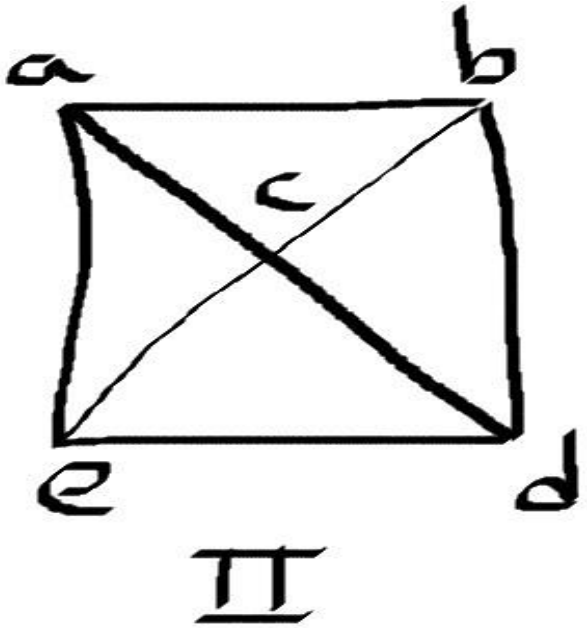
Euler Circuit : A simple circuit containing every edge of a graph

All edges covered , No repetition of edges	Euler path
All edges , No repetition of edges and initial vertex = end vertex	Euler circuit

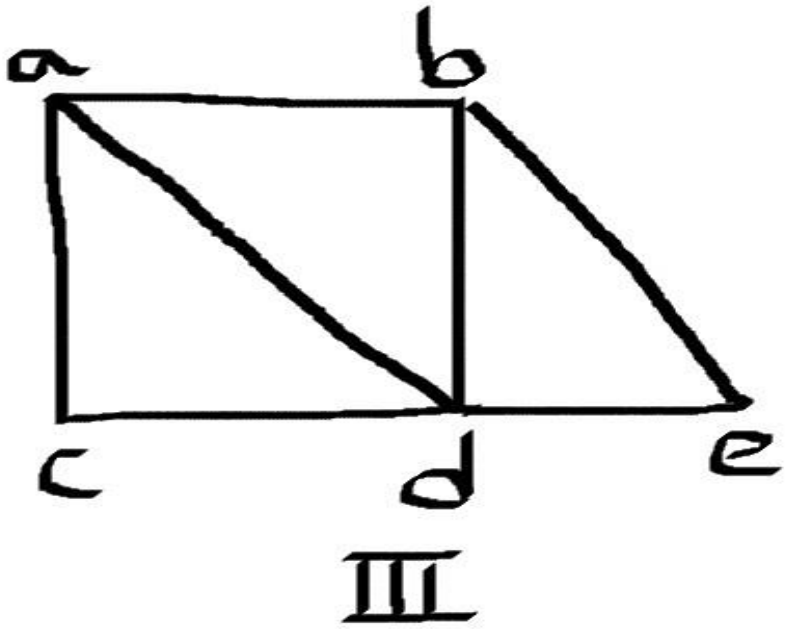
I	a,b,e,c,d,e,a	Euler circuit
II		Neither Euler circuit or Euler path
III	a,c,d,e,b,d,a,b	Euler path



I



II



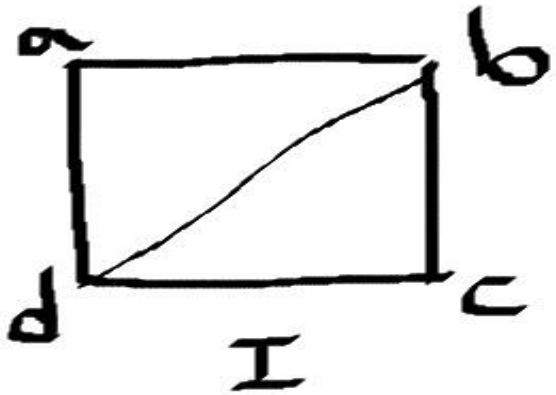
III

For the existence of Euler path/Euler circuit,
G-connected graph

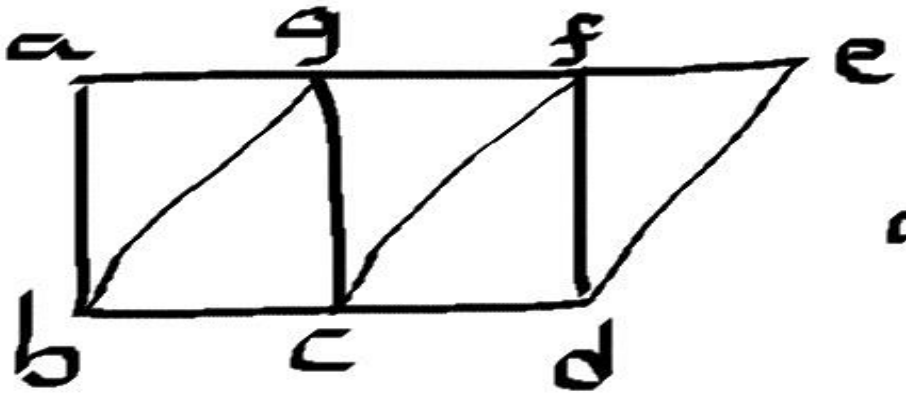
G has Euler circuit if and only if every vertex has even degree

G has Euler path but not Euler circuit if and only if it has exactly two vertices of odd degree

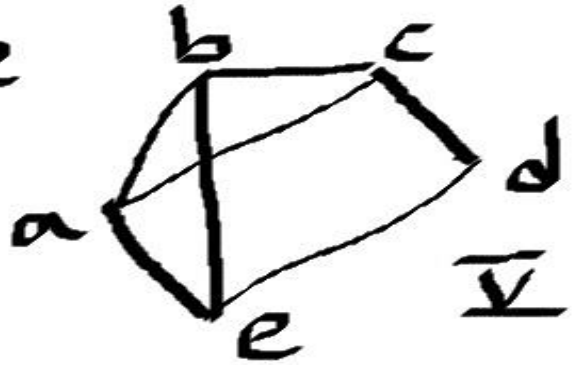
Exercises – Verify Euler Path or Euler Circuit



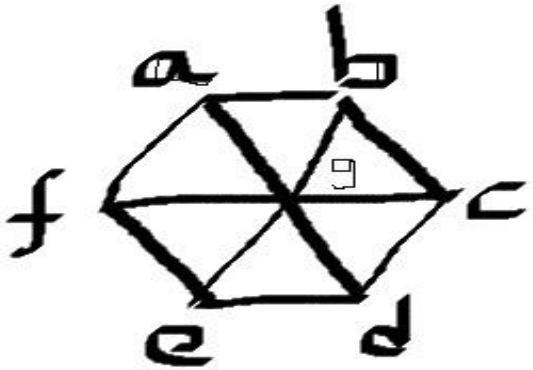
I



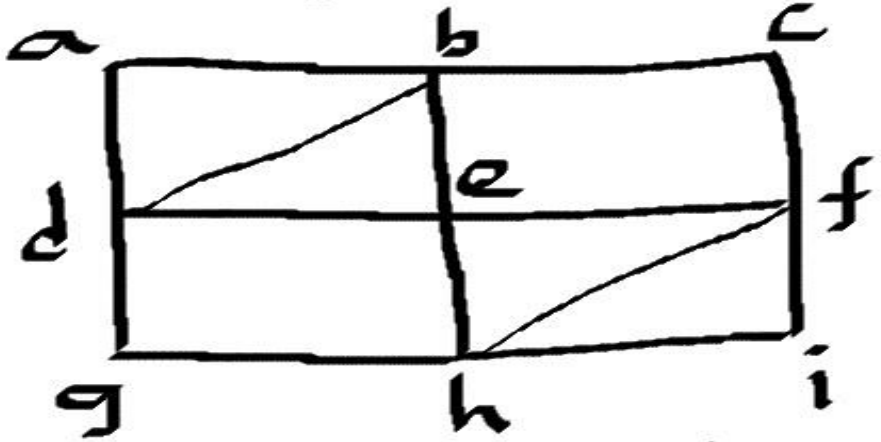
II



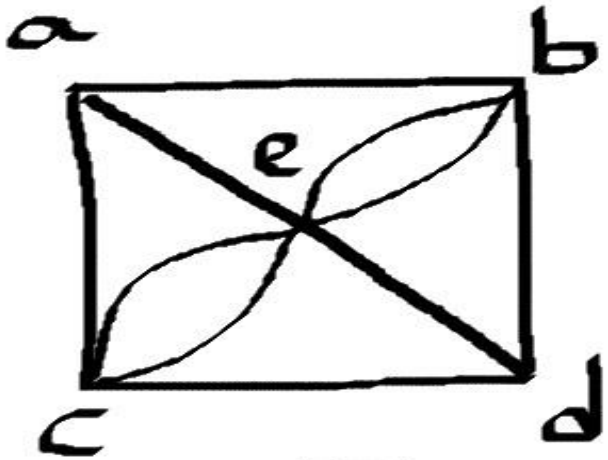
V



III



IV



VI

Euler Path and Euler Circuit

Graph	Euler path	Euler circuit
I		
II		
III		
IV		
V		
VI		

Euler Path and Euler Circuit

Graph	Euler path	Euler circuit
I	Yes d-b-c-d-a-b Exactly 2 vertices of odd degree	No
II	Yes (b-a-g-f-e-d-f-c-g-b-c-d) Exactly 2 vertices of odd degree	No
III	No (since 6 vertices of odd degree)	No
IV	Yes	Yes(each vertex has even degree)
V	No	No
VI	Yes Exactly 2 vertices of odd degree a-b-d-e-b-e-a-c-e-c-d	No

Konigsberg bridge problem

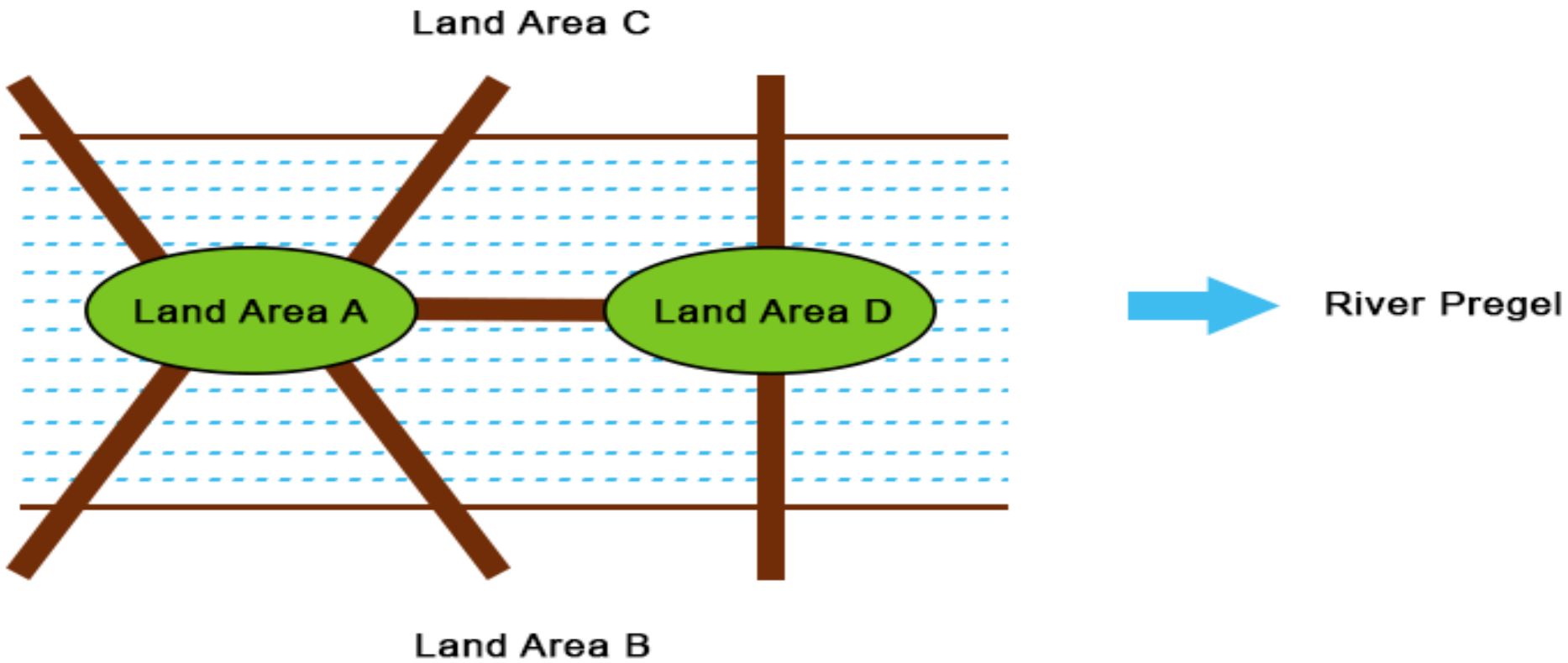
Dr.V.K.Ajay

Konigsberg bridge problem

- The Konigsberg is the name of the German city, but this city is now in Russia.
- In the below image, we can see the inner city of Konigsberg with the river Pregel.
- There are a total of four land areas in which this river Pregel is divided, i.e., A, B, C and D.
- There are total 7 bridges to travel from one part of the city to another part of the city.

Dr.V.K.Ajay

- The **Konigsberg bridge problem**



Konigsberg Bridge Problem

- **The Konigsberg bridge problem**

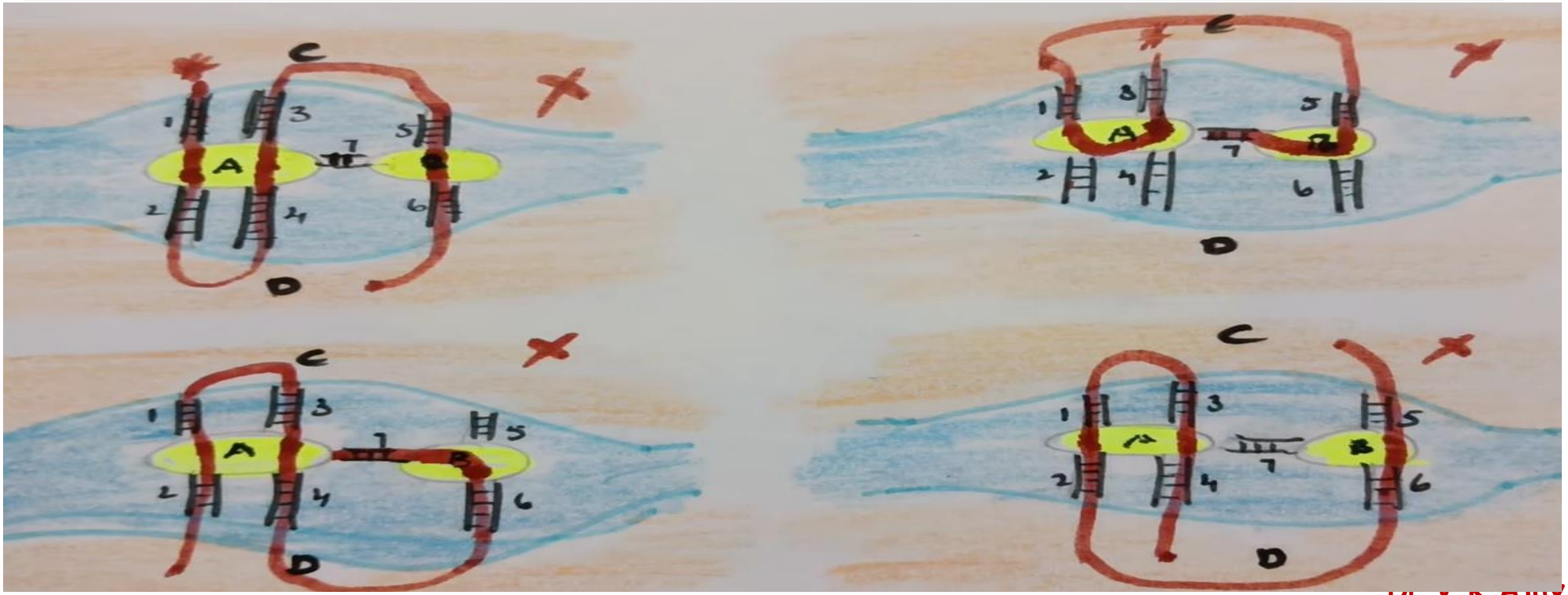
The Konigsberg Bridge contains the following problem which says:

Is it possible for anyone to cross each of the seven bridges only a single time and come back to the beginning point without swimming across the river if we begin this process from any of the four land areas that are A, B, C, and D?.

The problem was to start from anyone of the land area, walk across each bridge exactly once and return to the starting point

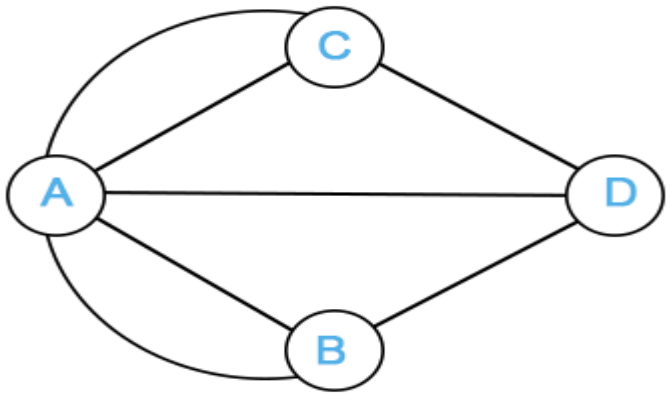
Konigsberg bridge problem

- The problem was to start from anyone of the land area, walk across each bridge exactly once and return to the starting point



Konigsberg bridge problem

- **Solution of Konigsberg Bridge problem**
- **In 1735, this problem was solved by Swiss mathematician Leon hard Euler.**
- **According to the solution to this problem, these types of walks are not possible.**
- **With the help of following graph, Euler shows the given solution.**



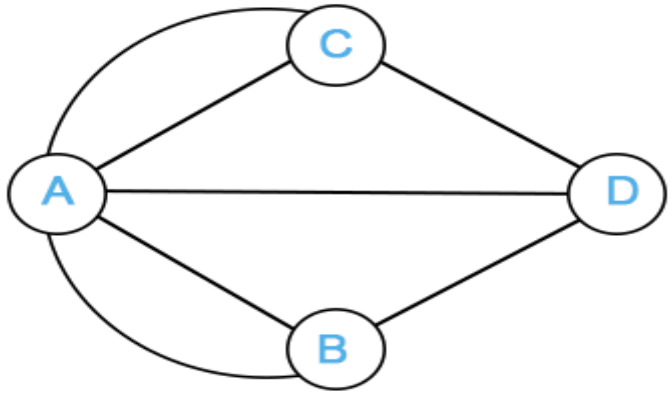
Graph Representation

Dr.V.K.Ajay

Konigsberg bridge problem

In the above graph, the following things have:

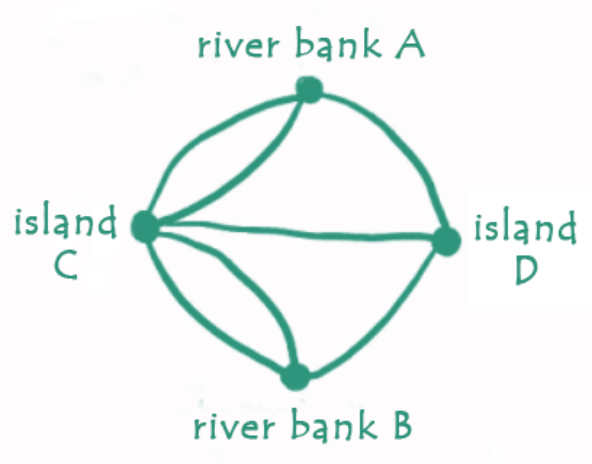
- **The vertices of this graph are used to show the landmasses.**
- **The edges are used to show the bridges.**



Graph Representation

Konigsberg bridge problem

In this diagram, he used two islands and 4 dots (vertices) for two riverbanks. With the help of A, B, C, and D, these dots have been marked. The 7 lines (arcs) are used to show the seven bridges. In the above diagram, 3 bridges (arcs) were used to join riverbank A, and 3 arcs were used to join riverbank B. As same, 5 bridges (arcs) were used to join island C, and 3 arcs were used to join island D. This shows that all the vertices of this network contain an odd number of arcs, so these types of vertices are called odd vertices. (For an even vertex, there must be an even number of bridges joining to it)



Konigsberg bridge problem

We have to keep in mind that the problem of Konigsberg was to travel around the town by crossing each bridge only a single time. On Euler's network, it meant that all the vertices had to be visited and traced over each arc only a single time. He worked on it and said that it would not be done, because to do this, we want the odd vertex, and for odd vertex, the trip must start and end at that vertex. So there can be only 2 odd vertices, and there can be only one starting and one end only if we can trace over each arc only once. Since there is a total of 4 odd vertices in the bridge, so it would not be possible to do.

Konigsberg bridge problem

So Euler observed that at the time of tracing the graph, if they try to visit a vertex, then the following things happen:

- **The graph will contain an edge that enters into the vertex.**
- **It will contain one more edge that leaves the vertex.**
- **Therefore, there must be an even number in the order of vertex.**

Theorem:

A connected graph has an Eulerian path(cycle) if degree of each vertex is even

Konigsberg bridge problem

About the network, Euler found that if a network contains the following things, only then that network will be traversable:

- The network does not contain any odd vertices. Here, the starting vertex and the end vertex must be the same. The starting vertex can be any vertex, but the ending point must be at the same starting vertex.
- Or the network contains two odd vertices. Here, the beginning point must be the one odd vertex, and the endpoint must be the other odd vertex.

A vertex is called **even vertex** if it has even number of edges

A vertex is called **odd vertex** if it has odd number of edges

Konigsberg bridge problem

Now,

- **Since there are total 4 odd vertices in the Konigsberg network, therefore Euler concluded that the network is not traversable.**
- **Thus, Euler finally concluded that it is not possible to traverse the desired walking tour of Konigsberg.**

Eulerian graphs

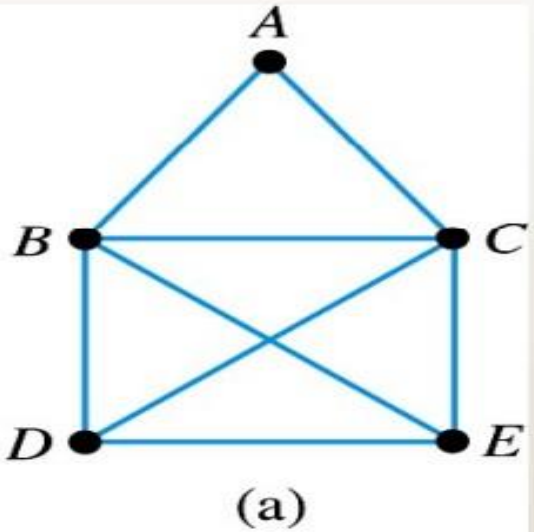
Euler Graph - A connected graph G is called an Euler graph, if there is a closed trail which includes every edge of the graph G.

Euler Path - An Euler path is a path that uses every edge of a graph exactly once. An Euler path starts and ends at different vertices.

Euler Circuit - An Euler circuit is a circuit that uses every edge of a graph exactly once. An Euler circuit always starts and ends at the same vertex. A connected graph G is an Euler graph if and only if all vertices of G are of even degree, and a connected graph G is Eulerian if and only if its edge set can be decomposed into cycles.

Examples - Eulerian graphs

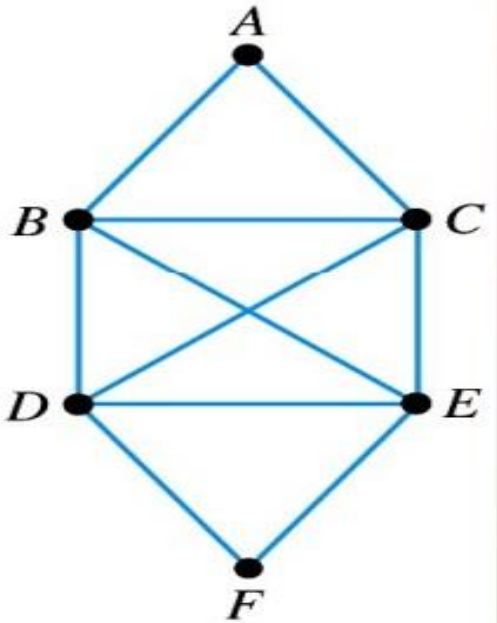
Euler path



(a)

An Euler path
D, E, B, C, A, B, D, C, E

Euler circuit

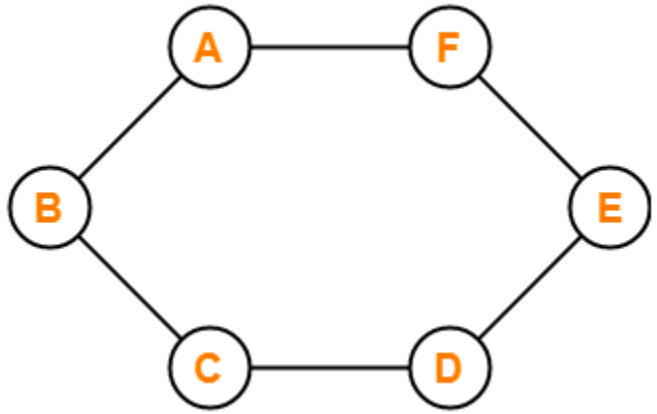


(b)

An Euler circuit
D, E, B, C, A, B, D, C, E, F, D

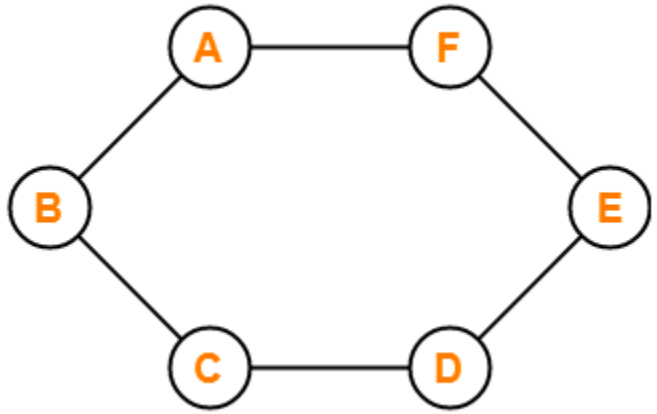
Hamilton Graph

- **Suppose a closed walk in the connected graph that visits every vertex of the graph exactly once (except starting vertex) without repeating the edges.**
- **Such a graph is called a Hamiltonian graph, and such a walk is called a Hamiltonian path. The Hamiltonian circuit is also known as Hamiltonian Cycle.**



Hamilton Graph

- **Hamilton path** - It is a simple path passing through every vertex exactly once
- **Hamilton Circuit** - It is a simple circuit passing through every vertex exactly once .A Hamiltonian path that starts and ends at the same vertex is called a Hamiltonian circuit
- **Hamilton graph** - A graph possessing a Hamilton circuit is called Hamilton graph



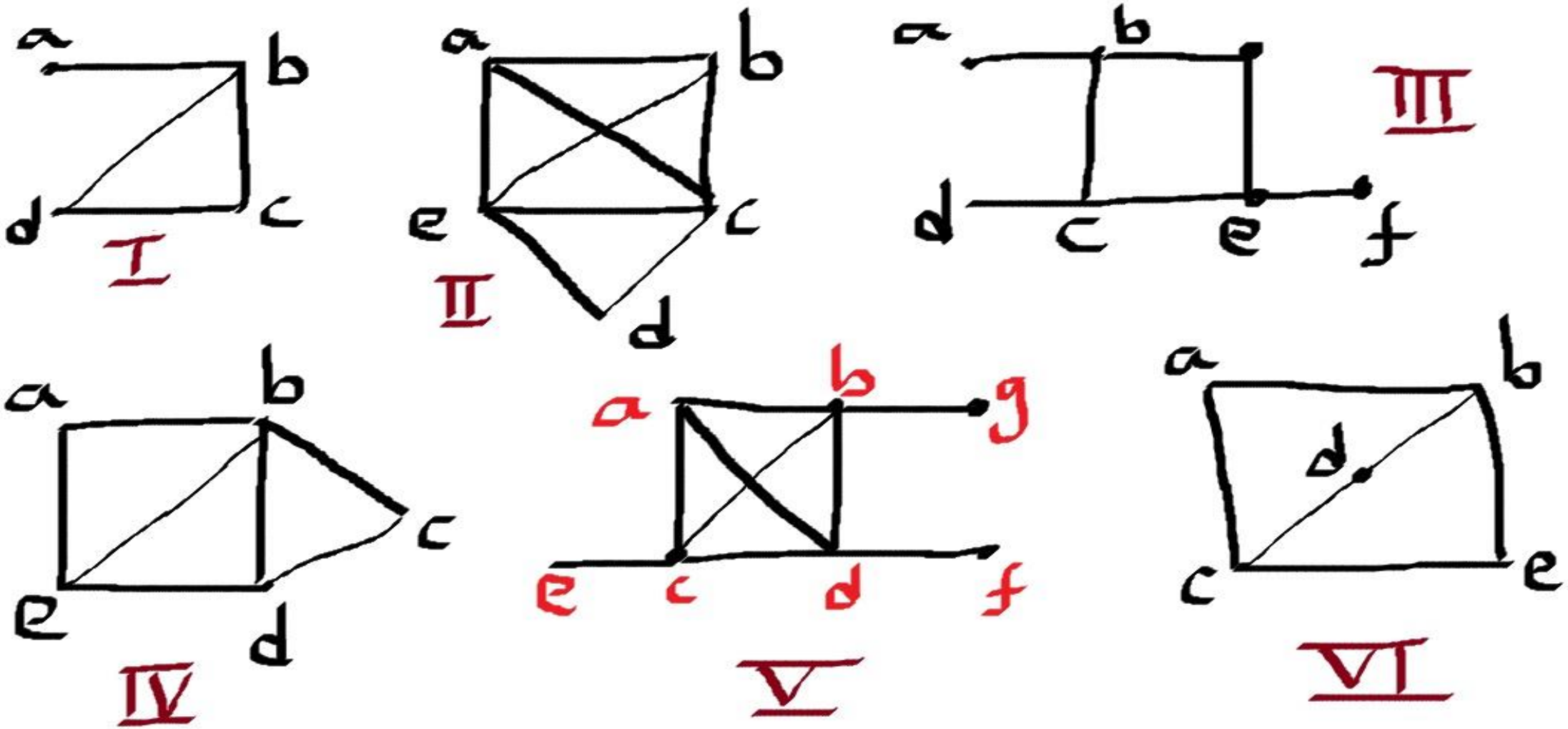
Hamilton Graph

In other words, A Hamiltonian path that starts and ends at the same vertex is called a Hamiltonian circuit. Every graph that contains a Hamiltonian circuit also contains a Hamiltonian path, but vice versa is not true. There may exist more than one Hamiltonian path and Hamiltonian circuit in a graph.

The graph shown in the above image consists of a closed path ABCDEFA which starts from vertex A and traverses all other vertices or nodes without traversing any of the nodes twice other than vertex A in the path of traversal. Therefore, the graph shown in the above image is a Hamilton graph.

Hamilton Graph

Which of the following graphs are Hamilton path or circuit



Ajay

Hamilton Graph - Exercise

2

	Hamilton path	Hamilton Circuit
I	YES (A-B-C-D)	NO
II	YES	YES (A-B-C-D-E)
III	NO	NO
IV	YES	YES (A-B-C-D-E-A)
V	NO	NO
VI	YES (A-B-E-C-D)	NO

Statement Of The Problem

The traveling salesman problem involves a salesman who must make a tour of a number of cities using the shortest path available and visit each city exactly once and only once and return to the original starting point. For each number of cities n ,the number of paths which must be explored is n!, causing this problem to grow exponentially rather than as a polynomial. There are bunch of algorithms offering comparably fast running time and still yielding near optimal solutions.

Traveling salesman problem (TSP)

Some solution methods of TSP include:

- Brute-force method.
- Branch and Bound

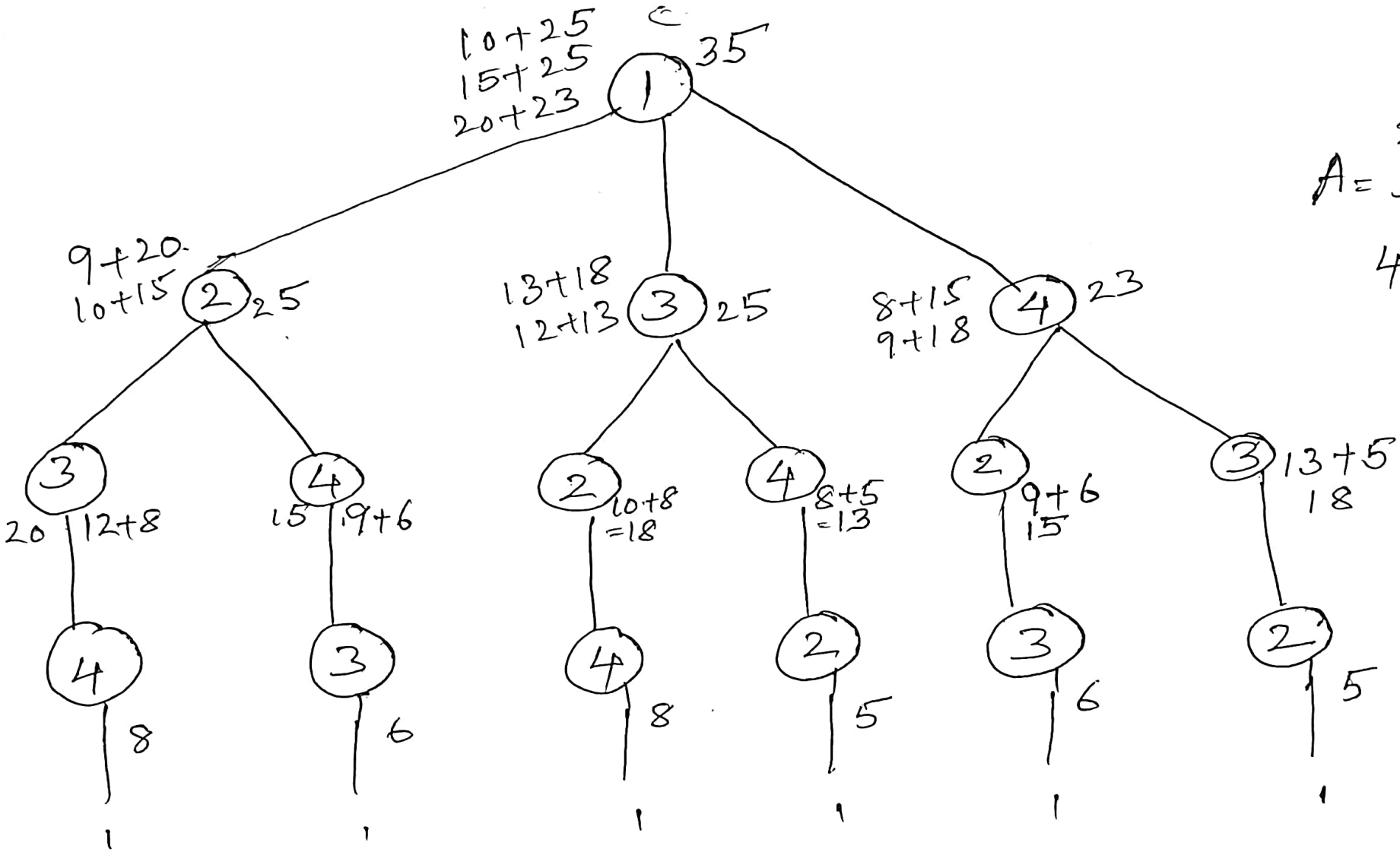
Traveling salesman problem (TSP)- The brute-force method

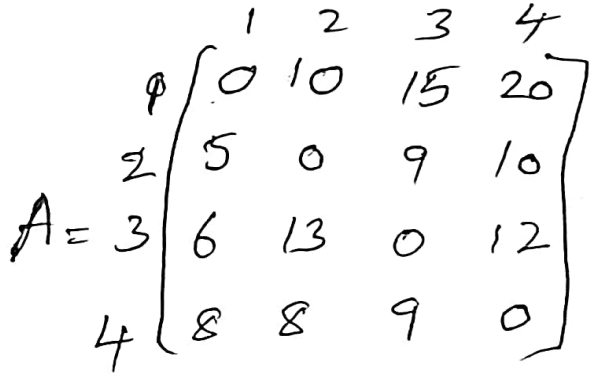
brute-force method

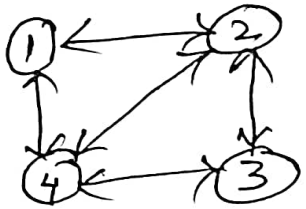
The brute-force method is to simply generate all possible tours and compute their distances. The shortest tour is thus the optimal tour. To solve TSP using Brute-force method we can use the following steps:

Step 1. calculate the total number of tours.
Step 2. draw and list all the possible tours.
Step 3. calculate the distance of each tour.
Step 4. choose the shortest tour, this is the optimal solution.

Traveling salesman problem (TSP)- The brute-force method





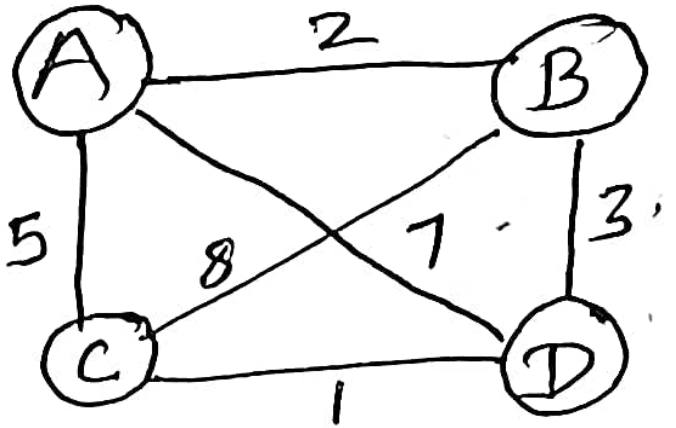


brute-force method – Example 2

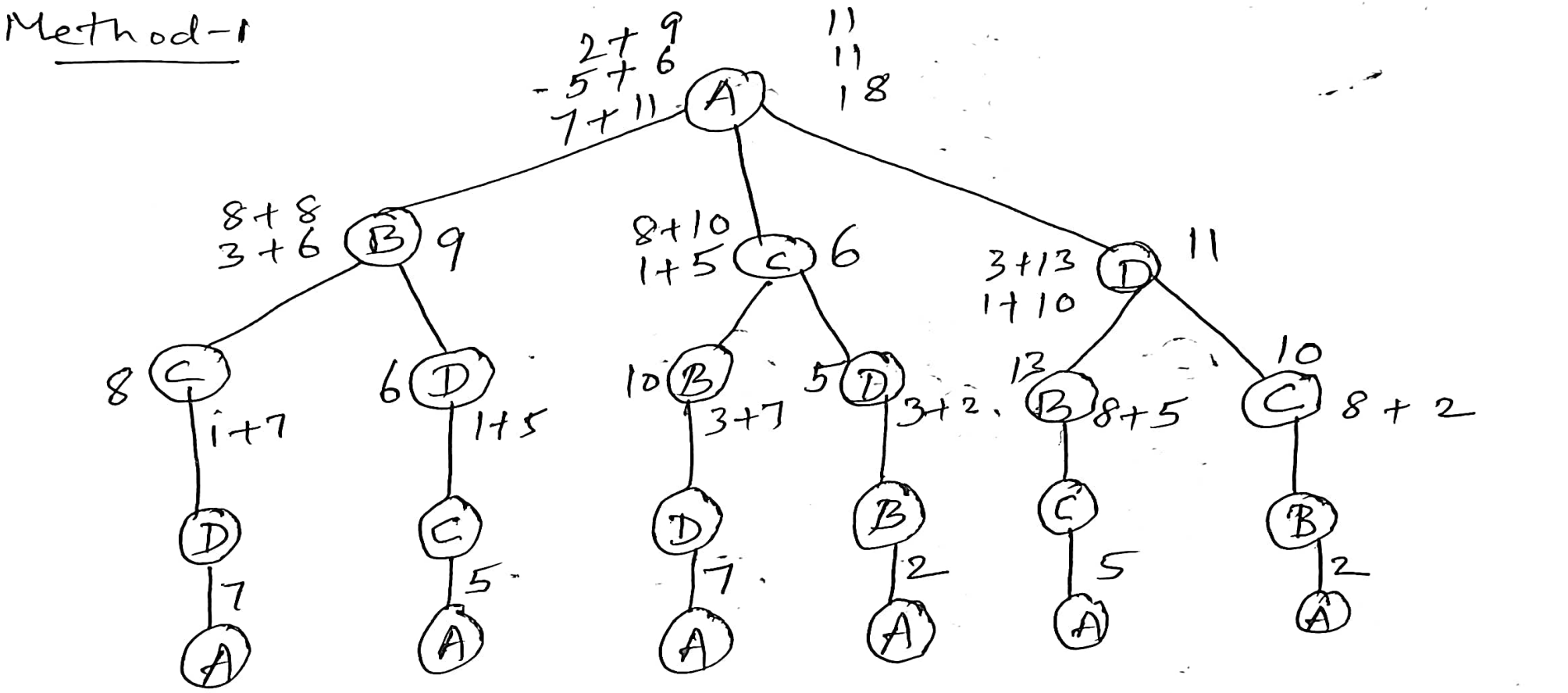
Example-2-TSP :

Adjacency Matrix =

	A	B	C	D
A	0	2	5	7
B	2	0	8	3
C	5	8	0	1
D	7	3	1	0



brute-force method – Example 2

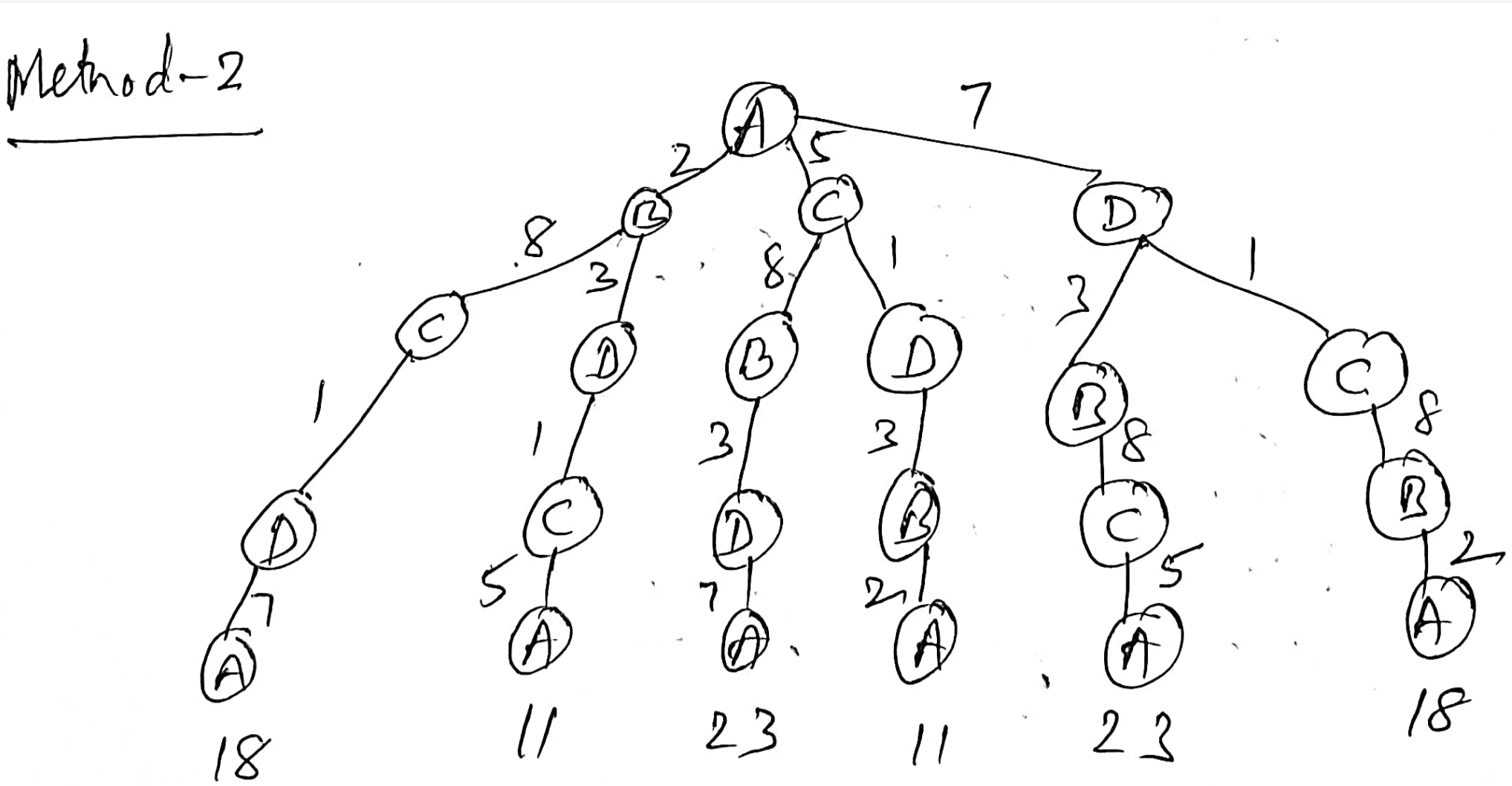


Minimum Cost = 11

Minimum Tour of graph = A → B → D → C → A

A → C → D → B → A

brute-force method – Example 2 – Method 2



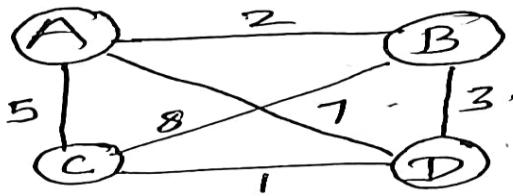
Traveling salesman problem (TSP)- The brute-force method

3a

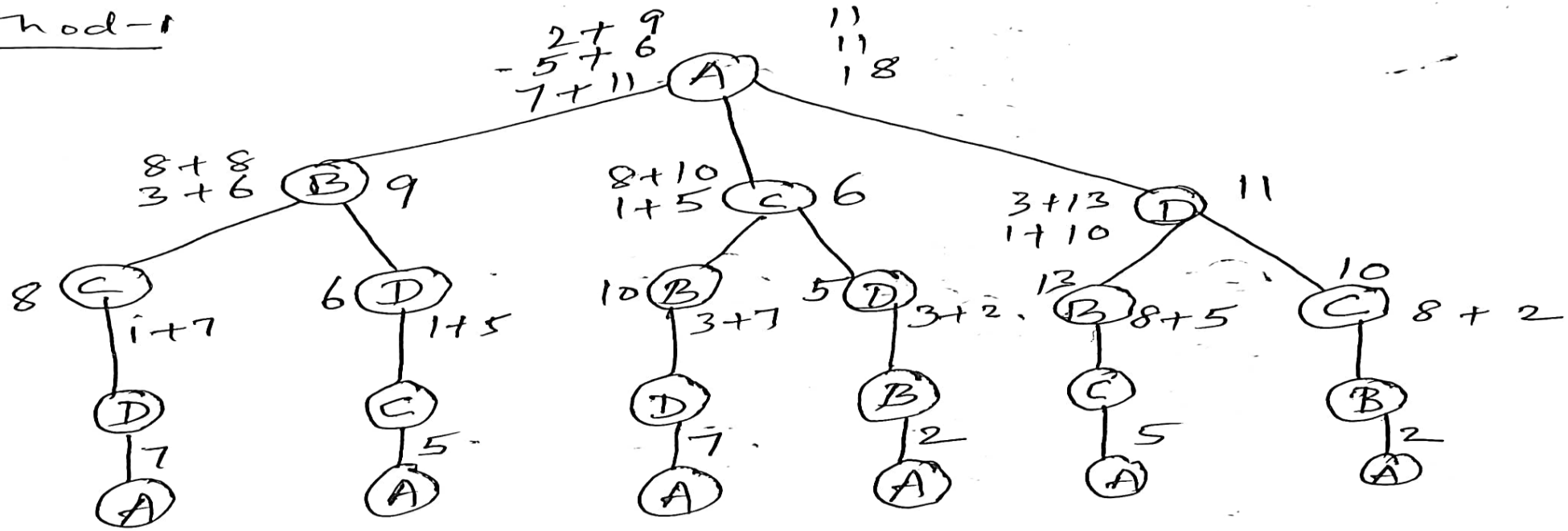
Example-2 - TSP :

Adjacency Matrix =

	A	B	C	D
A	0	2	5	7
B	2	0	8	3
C	5	8	0	1
D	7	3	1	0



Method-1



Minimum Cost = 11

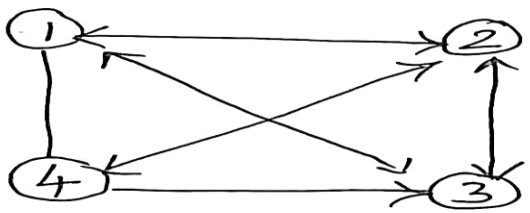
Minimum Tour of graph = A → B → D → C → A

A → C → D → B → A

Travelling Salesman Problem (Brute Force Approach - Dynamic Programming)

$$g(i,S) = \min_{k \in S} \{ C_{ik} + g(k, S-\{k\}) \} \quad \text{— (1)}$$

Given graph



Given Weights.

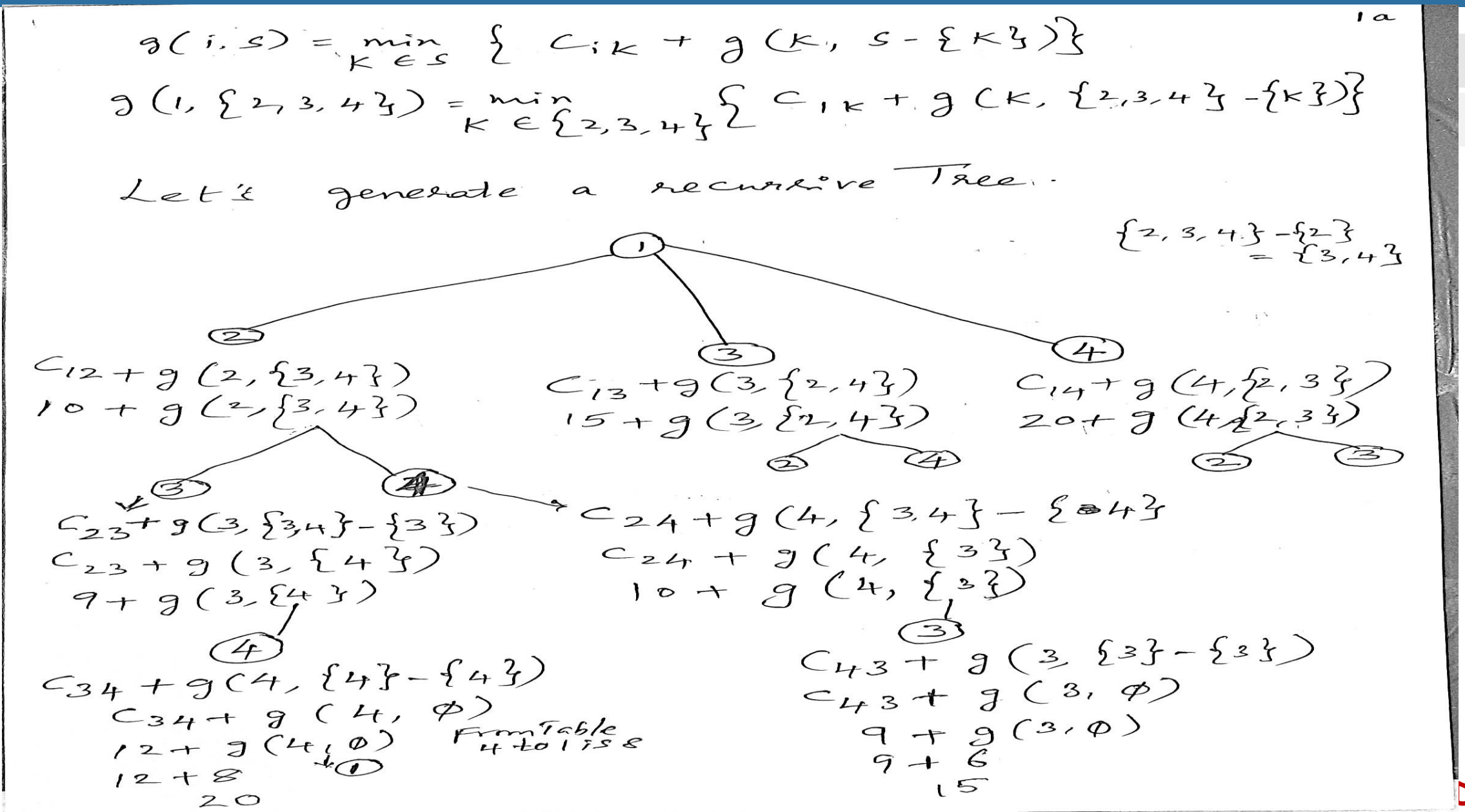
	1	2	3	4
1	0	10	15	20
2	5	0	9	10
3	6	13	0	12
4	8	8	9	0

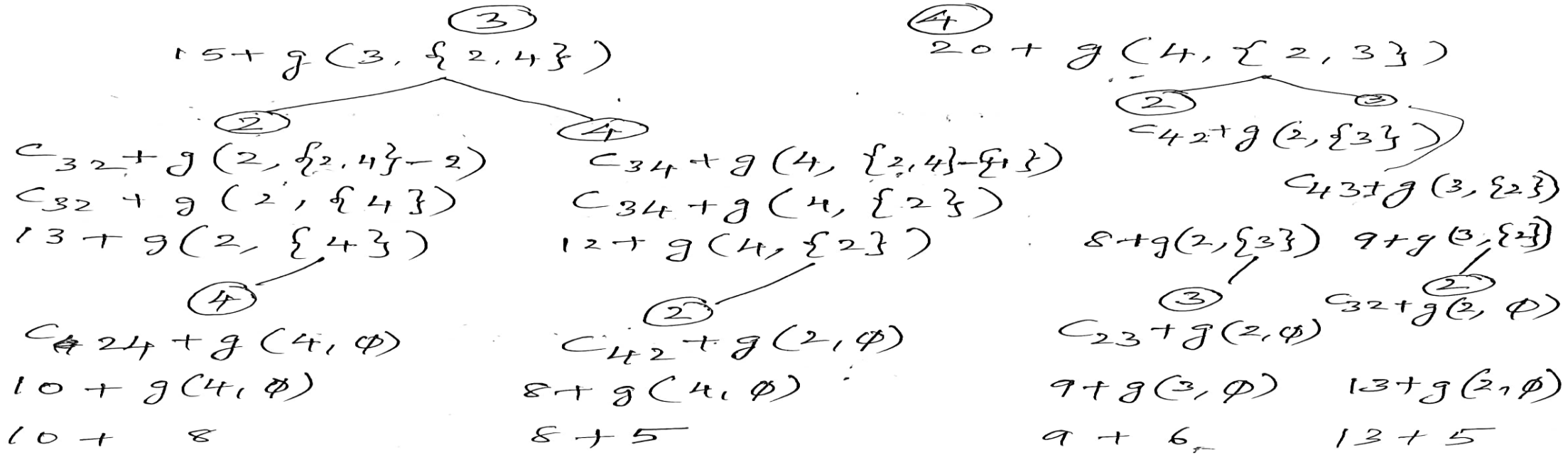
1) We have to start from Vertex and travel all other vertices once and we have to return to the same starting Vertex.
2) Cost of Travelling must be minimum.

Starting Vertex, $i = 1$; Cost function is termed as g.

Visit remaining Vertices, $S = \{2, 3, 4\}$

We can travel from $1 \longrightarrow 2$, $1 \longrightarrow 3$, $1 \longrightarrow 4$ $\Rightarrow k \in \{2, 3, 4\}$

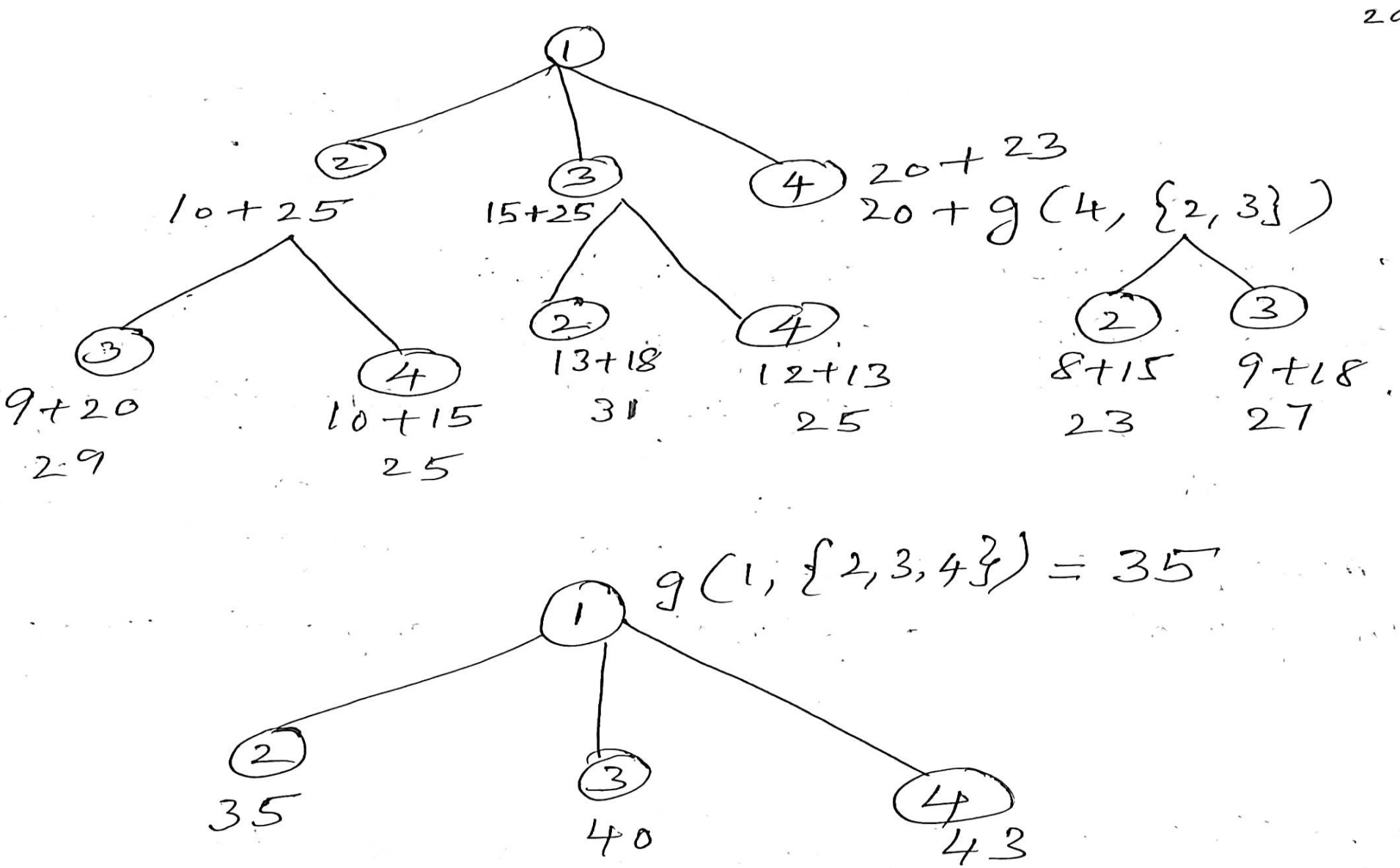

1a
g(i, S) = min_{k∈S} { c_ik + g(k, S − {k}) }
g(1, {2, 3, 4}) = min_{k∈{2,3,4}} { c_1k + g(k, {2, 3, 4} − {k}) }
Let's generate a recursive Tree.
1
2
3
4
{2, 3, 4} − {2} = {3, 4}
c12 + g(2, {3, 4})
10 + g(2, {3, 4})
c13 + g(3, {2, 4})
15 + g(3, {2, 4})
c14 + g(4, {2, 3})
20 + g(4, {2, 3})
c23 + g(3, {3,4} − {3})
c23 + g(3, {4})
9 + g(3, {4})
c24 + g(4, {3, 4} − {4})
c24 + g(4, {3})
10 + g(4, {3})
c34 + g(4, {4} − {4})
c34 + g(4, φ)
12 + g(4, φ)
12 + 8
20
From table 4 to 1 is 8
c43 + g(3, {3} − {3})
c43 + g(3, φ)
9 + g(3, φ)
9 + 6
15



The Tree is expanded,
Now, we get the Table and we get the values for last level first, then upper level & so on.

g(2, 0) = 5
g(3, 0) = 6
g(4, 0) = 8
g(2, {3}) = 15
g(2, {4}) = 18
g(3, {2}) = 18
g(3, {4}) = 20
g(4, {2}) = 13
g(4, {3}) = 15

g(2, {3,4}) = 25 (Minimum of 29, 25)
g(3, {2,4}) = 25 (31, 25)
g(4, {2,3}) = 23 (23, 27)
g(1, {2,3,4}) = 35 (35, 40, 43)


2a
1
2
3
4
10+25
15+25
20+23
20+g(4,{2,3})
3
4
2
4
2
3
9+20
29
10+15
25
13+18
31
12+13
25
8+15
23
9+18
27
g(1,{2,3,4})=35
1
2
3
4
35
40
43

Branch and bound method –

The Branch and Bound strategy divides a problem to be solved into a number of sub-problems. It is a system for solving a sequence of subproblems each of which may have multiple possible solutions and where the solution chosen for one sub-problem may affect the possible solutions of later sub-problems.
The branch and bound method provides a correct solution for the given problem. It is suitable for use when the number of vertices of graphs does not exceed 60, which is sufficient when it comes to problems such as package delivery.
This method is based on the idea that the sets are divided into two disjoint subsets at every step of the process - branching. Furthermore, one subset contains the path between the two selected towns and the other subset does not. For each of these subsets the lower restriction (bound) is calculated for the duration or for travel expenses. Finally, the subset which exceeds the estimated lower bound is eliminated.

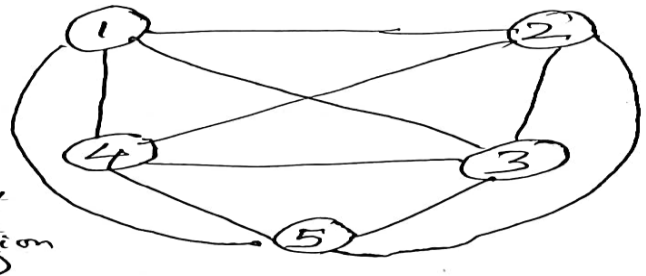
The procedure of branching is represented by a tree where the top is marked by the branching points of the set of solutions, and the edges mark the path between the two contiguous vertices in the graph which are used to model the problem and inside which the shortest Hamilton cycle is pursued.

Travelling Salesman Problem (Branch and Bound)

1. Reduce the Matrix.

	1	2	3	4	5	
1	∞	20	30	10	11	10 → Min value of row.
2	15	∞	16	4	2	2
3	3	5	∞	2	4	2
4	19	6	18	∞	3	3
5	16	4	7	16	∞	4

= 21 (Total Cost of Reduction of rows)



⇒

	∞	10	20	0	1
	13	∞	14	2	0
	1	3	∞	0	2
	16	3	15	∞	0
	12	0	2	12	∞
	1	0	3	0	0

Reduce Rows.

	1	2	3	4	5
1	∞	20	30	10	11
2	15	∞	16	4	2
3	3	5	∞	2	4
4	19	6	18	∞	3
5	16	4	7	16	∞

= 4 (Total Cost of reduction of Columns)

2. Reduce Columns

	∞	10	17	0	1
	12	∞	11	2	0
	0	3	∞	0	2
	15	3	12	∞	0
	11	0	0	12	∞

Total Cost of Reduction
4 + 21 = 25.

⇒ Minimum cost of Tour may be ≤ or > 25.

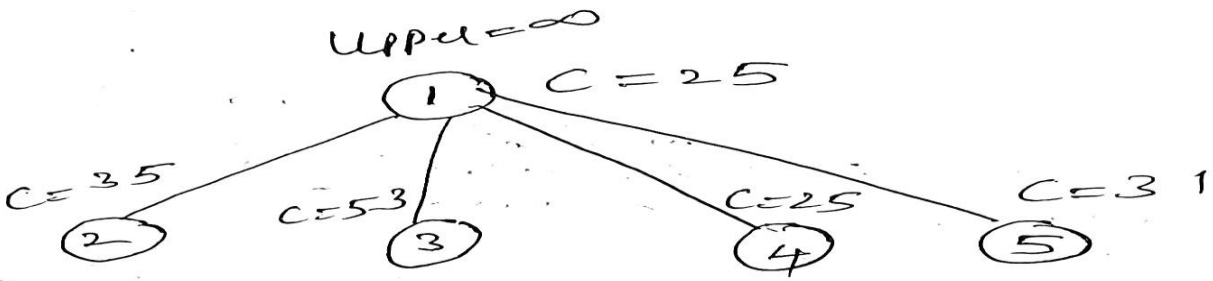
A matrix is said to be reduced, when atleast one row and one column must contain atleast one zero.

14a

Now, let us consider reduced matrix.
Let us solve using branch and bound with the help of state space tree

	1	2	3	4	5
1	∞	10	17	0	1
2	12	∞	11	2	0
3	0	3	∞	0	2
4	15	3	12	∞	0
5	11	0	0	12	∞

→ (4) (1)



For every vertex, we should find out cost.
Cost of First Matrix, ie reduced cost = 25 (C=25)
Let upper bound initially = ∞
We need not fix upper bound for each vertex.
Once, we reach end, then we update upperbound.

To find cost of 2, Consider 1 → 2 path

Make first row, second column as ∞

So from (1)(2) we get

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ \infty & \infty & \infty & \infty & \infty \\ 12 & \infty & 11 & 2 & 0 \\ 0 & \infty & \infty & 0 & 2 \\ 15 & \infty & 12 & \infty & 0 \\ 11 & \infty & 0 & 12 & \infty \end{bmatrix} \quad \text{—— (2)}$$

After coming from 1 to 2, we can't go from 2 to 1, So make it as ∞ (2, 1) in (2)

We get

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 11 & 2 & 0 \\ 0 & \infty & \infty & 0 & 2 \\ 15 & \infty & 12 & \infty & 0 \\ 11 & \infty & 0 & 12 & \infty \\ 0 & \infty & 0 & 0 & 0 \end{bmatrix} \begin{matrix} \infty \text{ (ignore)} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{matrix} \quad \text{—— (3)}$$

check if matrix (3) is reduced or not.

⟹ C(1, 2) + Cost of reduction + Any other reduction cost

⟹ C(1, 2) + 0 + 0

⟹ 10 + 25 + 0

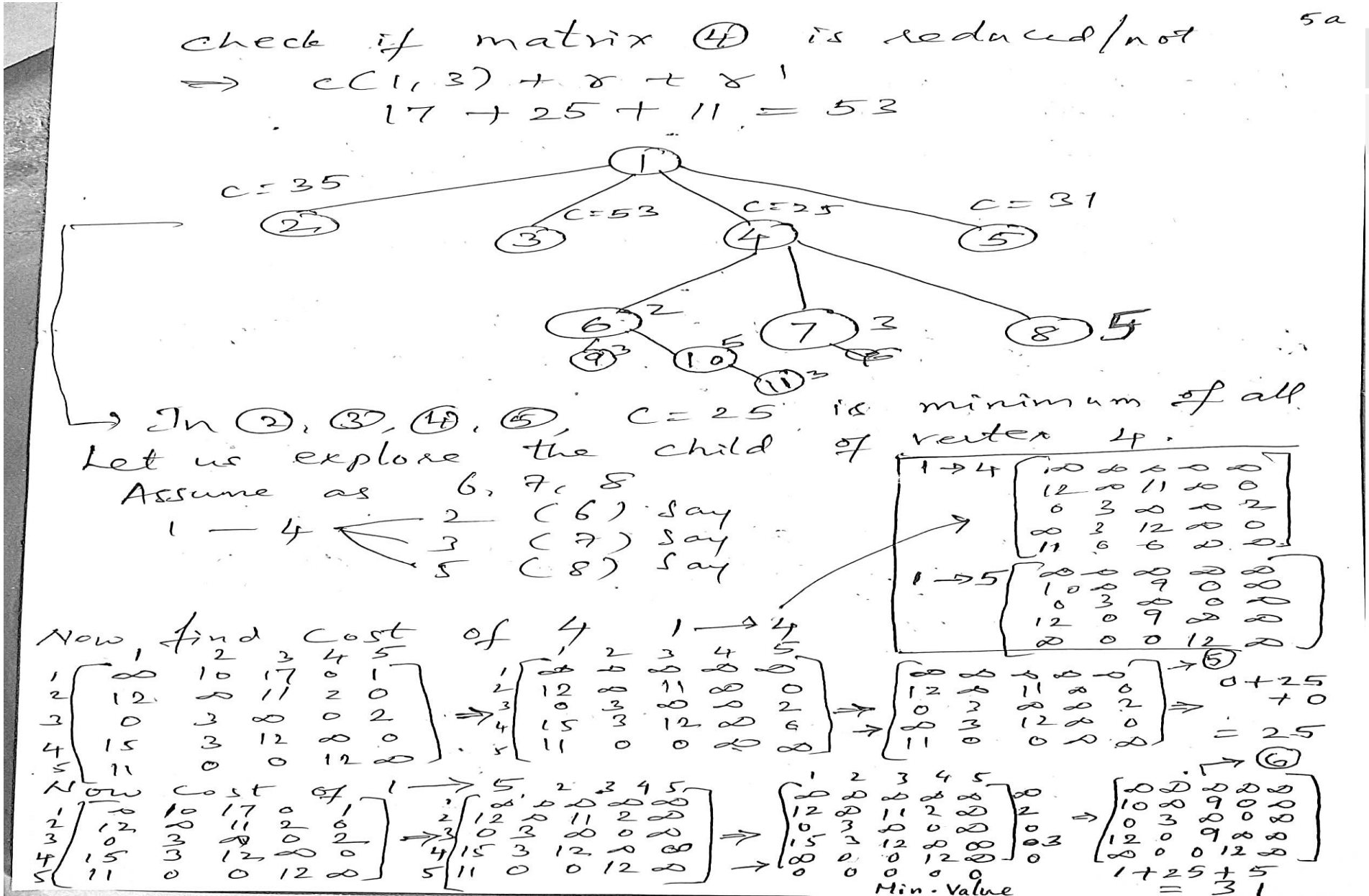
= 35

Now, find Cost of 3 ie 1 → 3

Take first row, & third column as ∞ in (1)

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 12 & \infty & \infty & 2 & 0 \\ \infty & 3 & \infty & 0 & 2 \\ 15 & 3 & \infty & \infty & 0 \\ 11 & 0 & \infty & 12 & 0 \\ 11 & 0 & \infty & 0 & 0 \end{bmatrix} \begin{matrix} \infty \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{matrix} \Rightarrow \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 1 & \infty & \infty & 2 & 0 \\ \infty & 3 & \infty & 0 & 2 \\ 4 & 3 & \infty & \infty & 0 \\ 0 & 0 & \infty & 12 & 0 \end{bmatrix} \quad \text{(4)}$$

Traveling salesman problem (TSP)- Branch and bound method



6

$C = 25$ is minimum.

Assume as 6, 7, 8.

$1 \rightarrow 4 \begin{cases} 2 & (6, \text{Say}) \\ 3 & (7, \text{Say}) \\ 4 & (8, \text{Say}) \end{cases}$

To find cost of 6 (ie node 2)

$4 \rightarrow 2$. Take reduced matrix in eqn (5). Make 4th row and second column as ∞. and 2 to 4, 2 to 1 as ∞ in matrix (5)

	1	2	3	4	5
1	∞	∞	∞	∞	∞
2	12	∞	11	∞	0
3	0	3	∞	∞	2
4	∞	3	12	∞	0
5	11	0	0	∞	∞

$\Rightarrow$

	1	2	3	4	5
1	∞	∞	∞	∞	∞
2	12	∞	11	∞	0
3	0	∞	∞	∞	2
4	∞	∞	∞	∞	∞
5	11	∞	0	∞	∞

$\Rightarrow$

	1	2	3	4	5	
1	∞	∞	∞	∞	∞	∞
2	∞	∞	11	∞	0	0
3	0	∞	∞	∞	2	0
4	∞	∞	∞	∞	∞	∞
5	11	∞	0	∞	∞	0
	0	∞	0	∞	0	

check reduced/not $\rightarrow$ Matrix of Node 6

$c(4,2) + r + r'$

$c(4,2) + c(4) + r'$

$3 + 25 + 0 = 28$

from (5) $\rightarrow$ 3 $\uparrow$ from (5) Cost = 25

from (5) $\rightarrow$ Take matrix (5)

16a

To find cost of 7 ie node 3, Take matrix 5

4 → 7 (3) ie 4 → 3 ie Do 4th row, 3rd Col as ∞. And 3 to 4, 3 to 1 as ∞

	1	2	3	4	5	
1	∞	∞	∞	∞	∞	∞
2	12	∞	∞	∞	0	0
3	∞	3	∞	∞	2	2
4	∞	∞	∞	∞	∞	∞
5	11	0	∞	∞	∞	0

Check reduction → 11 0 ∞ ∞ 0

Reduced Cost = 11 + 2 = 13

Reduced Matrix =

∞	∞	∞	∞	∞
1	∞	∞	∞	0
∞	1	∞	∞	0
∞	∞	∞	∞	∞
0	0	∞	∞	∞

→ Matrix of Node 7

c(4, 3) + r + r'

12 + 25 + 13 = 50

To find Cost of 8 (ie 5 node).

4 → 5 ⇒ Do 4th row, 5th column as ∞. And 5 to 4, 5 to 1 as ∞.

	1	2	3	4	5	
1	∞	∞	∞	∞	∞	∞
2	12	∞	11	∞	0	11
3	0	3	∞	∞	∞	0
4	∞	∞	∞	∞	∞	∞
5	11	0	0	∞	∞	0
	0	0	0	∞	∞	

Reduced = 11

⇒

∞	∞	∞	∞	∞
1	∞	0	∞	∞
0	3	∞	∞	∞
∞	∞	∞	∞	∞
∞	0	0	∞	∞

→ Matrix of Node 8

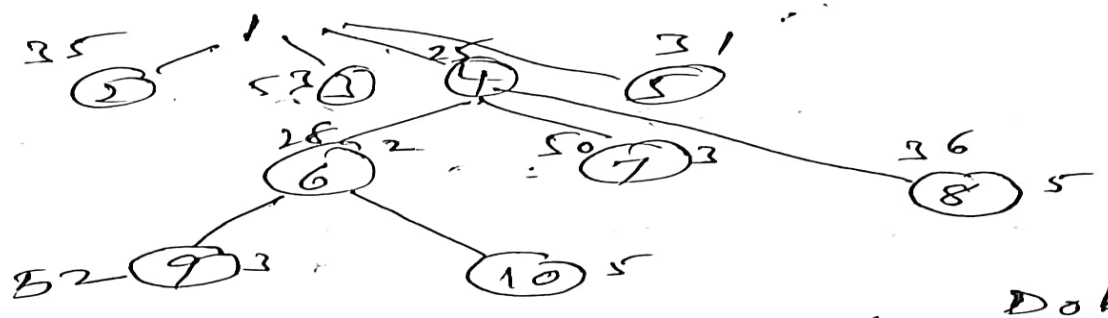
r(4, 5) + r + r'

0 + 25 + 11 = 36

Cost, C = 36 (for Vertex 8 ie node 5)

Traveling salesman problem (TSP)- Branch and bound method

From vertex ②, ③, ⑤, ⑥, ⑦, ⑧, Minimum is Vertex ⑥ (i.e Cost = 28)
So, expand ⑥



To find cost of node 9, take matrix 6;

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 11 & \infty & 0 \\ 0 & \infty & \infty & \infty & 2 \\ \infty & \infty & 0 & \infty & \infty \\ 11 & \infty & \infty & \infty & \infty \end{bmatrix} \longrightarrow \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & 0 \\ 0 & \infty & \infty & \infty & 2 \\ \infty & \infty & \infty & \infty & \infty \\ 11 & \infty & \infty & \infty & \infty \end{bmatrix}$$

Do row 2 Col 3 as ∞ And 3 to 1 as ∞

Also 3 to 1 as ∞

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & 0 \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ 11 & \infty & \infty & \infty & \infty \end{bmatrix} \Rightarrow \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & 0 \\ \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & \infty & \infty \end{bmatrix}$$

→ Matrix of Node 9

→ Cost of Parent node 6 is 28

$c(2,3) + c(6) + 13$

$11 + 28 + 13 = 52$

↳ reduced cost of Node 9 Matrix

See Matrix 6 Node 6

Traveling salesman problem (TSP)- Branch and bound method

To find cost of node 10, Take Matrix of node 6 (7a)

$2 \rightarrow 5 \Rightarrow$ Do 2nd row, 5th Column as ∞, And 5 to 1 as ∞.

$$\begin{bmatrix} & 1 & 2 & 3 & 4 & 5 \\ 1 & \infty & \infty & \infty & \infty & \infty \\ 2 & \infty & \infty & 11 & \infty & \infty \\ 3 & 0 & \infty & \infty & \infty & 2 \\ 4 & \infty & \infty & \infty & \infty & \infty \\ 5 & 11 & \infty & 0 & \infty & \infty \end{bmatrix} \rightarrow \begin{bmatrix} & 1 & 2 & 3 & 4 & 5 \\ 1 & \infty & \infty & \infty & \infty & \infty \\ 2 & \infty & \infty & \infty & \infty & \infty \\ 3 & 0 & \infty & \infty & \infty & \infty \\ 4 & \infty & \infty & \infty & \infty & \infty \\ 5 & \infty & \infty & 0 & \infty & \infty \end{bmatrix}$$

$\Rightarrow$ Reduced Cost = 0

Matrix of Node 10

11

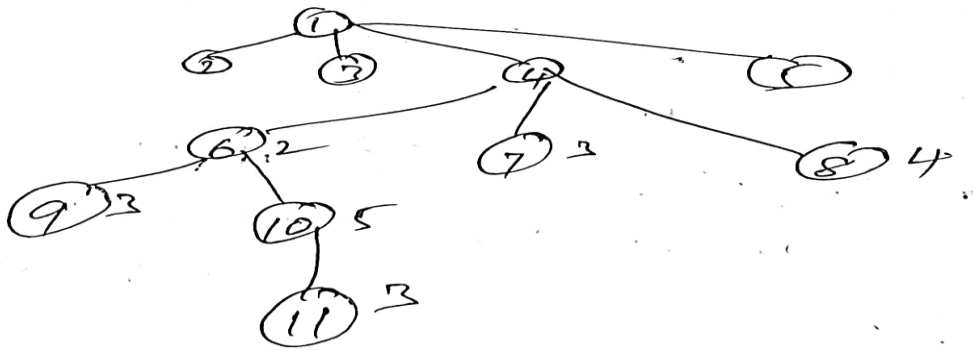
$c(2,5) + r + r'$

$c(2,5) + c(6) + 0$

$0 + 28 + 0 = 28.$

From vertices 2, 3, 5, 7, 8, 9, 10, minimum is 28 ie Vertex 10 (5)

So; expand 10 (1 – 4 – 2 – 5 – 3)



To find Cost of 11, Take matrix of Node 10

5 → 3 : Do 5th row, 3rd Column as ∞

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & \infty & \infty \end{bmatrix} \Rightarrow \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix}$$

Reduced Cost = 0.

$c(5,3) + r + r'$

$c(5,3) + c(10) + r' = \infty + 28 + 0$

$= 28.$

Finally we travelled from 1 – 4 – 2 – 5 – 3. → 1

And remaining is to 1.

So, Minimum Cost = 28

Update Upper bound from ∞ to 28.

When Upper bound is updated as 28, check the nodes whose Cost is greater than Upper.

So all nodes 2, 3, 5, 7, & 9 shall be ignored

Here last remaining node is 3, whose cost is 28

So, Cost of Tour is 28

And Tour is 1 — 4 — 2 — 5 — 3 — 1 which is minimum cost Tour.

Dijkstra's Algorithm

Dijkstra’s Algorithm

Dijkstra's algorithm finds the shortest path from one vertex to all other vertices.

It does so by repeatedly selecting the nearest unvisited vertex and calculating the distance to all the unvisited neighboring vertices.
Dijkstra's algorithm is often considered to be the most straightforward algorithm for solving the shortest path problem.

Dijkstra's algorithm is used for solving single-source shortest path problems for directed or undirected paths.
Single-source means that one vertex is chosen to be the start, and the algorithm will find the shortest path from that vertex to all other vertices.

Dijkstra's algorithm does not work for graphs with negative edges

Dijkstra's Algorithm

To find the shortest path, Dijkstra's algorithm needs to know which vertex is the source, it needs a way to mark vertices as visited, and it needs an overview of the current shortest distance to each vertex as it works its way through the graph, updating these distances when a shorter distance is found.

Dijkstra's Algorithm

How it works:

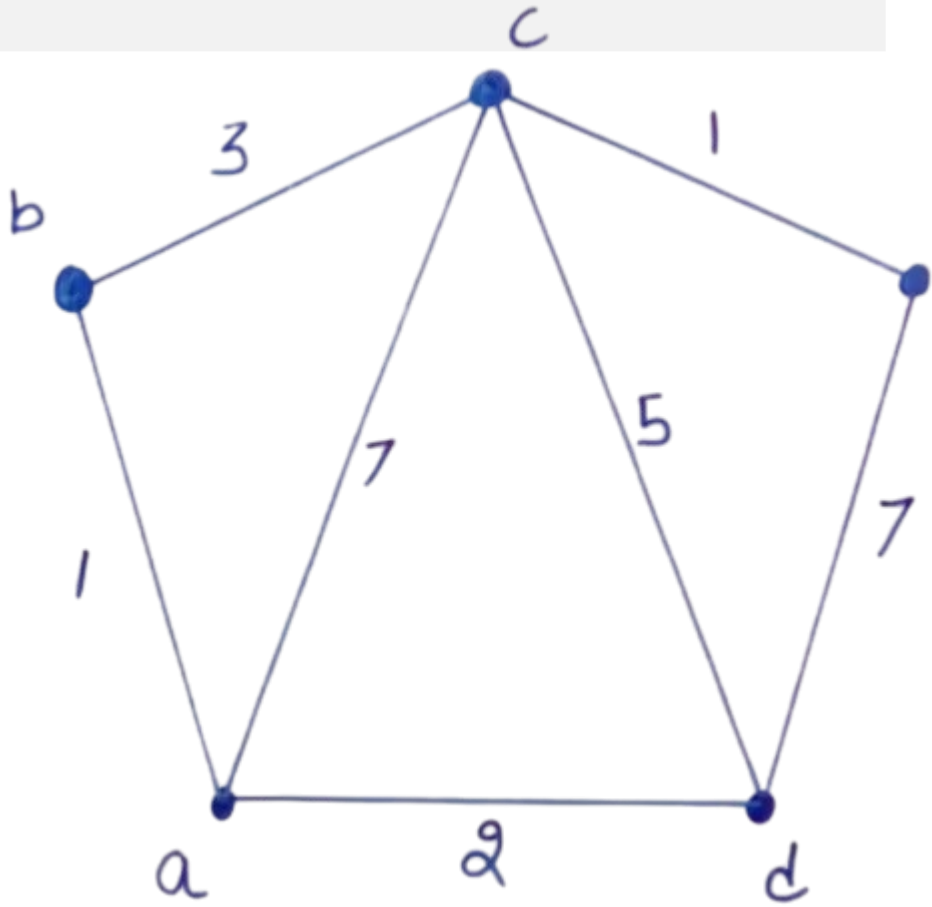
1. Set initial distances for all vertices: 0 for the source vertex, and infinity for all the other.
2. Choose the unvisited vertex with the shortest distance from the start to be the current vertex. So the algorithm will always start with the source as the current vertex.
3. For each of the current vertex's unvisited neighbor vertices, calculate the distance from the source and update the distance if the new, calculated, distance is lower.
4. We are now done with the current vertex, so we mark it as visited. A visited vertex is not checked again.
5. Go back to step 2 to choose a new current vertex, and keep repeating these steps until all vertices are visited.
6. In the end we are left with the shortest path from the source vertex to every other vertex in the graph.

Example - Dijkstra's Algorithm

Find the shortest distance and shortest path from 'a' to 'e' in the given weighted graph

From 'a' to 'e' shortest distance is 5
And the shortest path is a-b- c- e

a	b	c	d	e
0	∞	∞	∞	∞
0	1	7	2	∞
0	1	4	2	∞
0	1	4	2	9
0	1	4	2	5

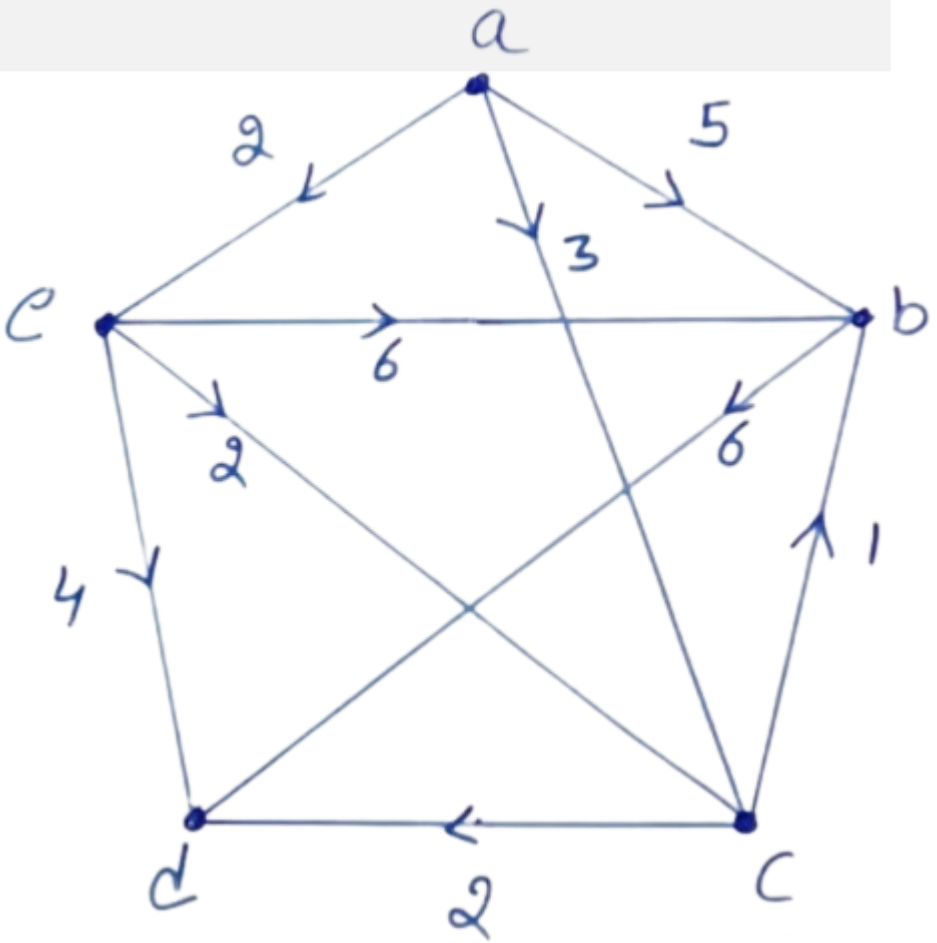


Example 2 - Dijkstra's Algorithm

Find the shortest distance and shortest path from 'a' to 'd' in the given weighted graph

From 'a' to 'd' shortest distance is 5
And the shortest path is a-c-d

a	b	c	d	e
0	∞	∞	∞	∞
0	5	3	∞	2
0	5	3	6	2
0	4	3	5	2
0	4	3	5	2

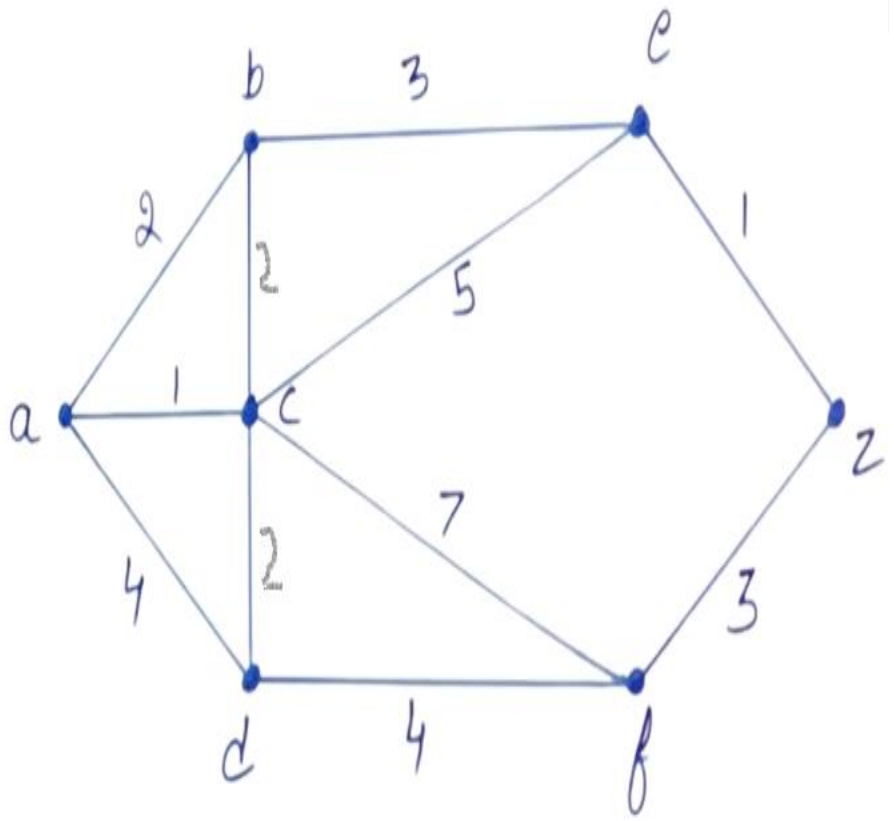


Example 2 - Dijkstra's Algorithm

Find the shortest distance and shortest path from 'a' to 'z' in the given weighted graph

From 'a' to 'z' shortest distance is 6
And the shortest path is a - b – e - z

a	b	c	d	e	f	z
0	∞	∞	∞	∞	∞	∞
0	2	1	4	∞	∞	∞
0	2	1	3	6	8	∞
0	2	1	3	5	8	∞
0	2	1	3	5	7	∞
0	2	1	3	5	7	6
0	2	1	3	5	7	6



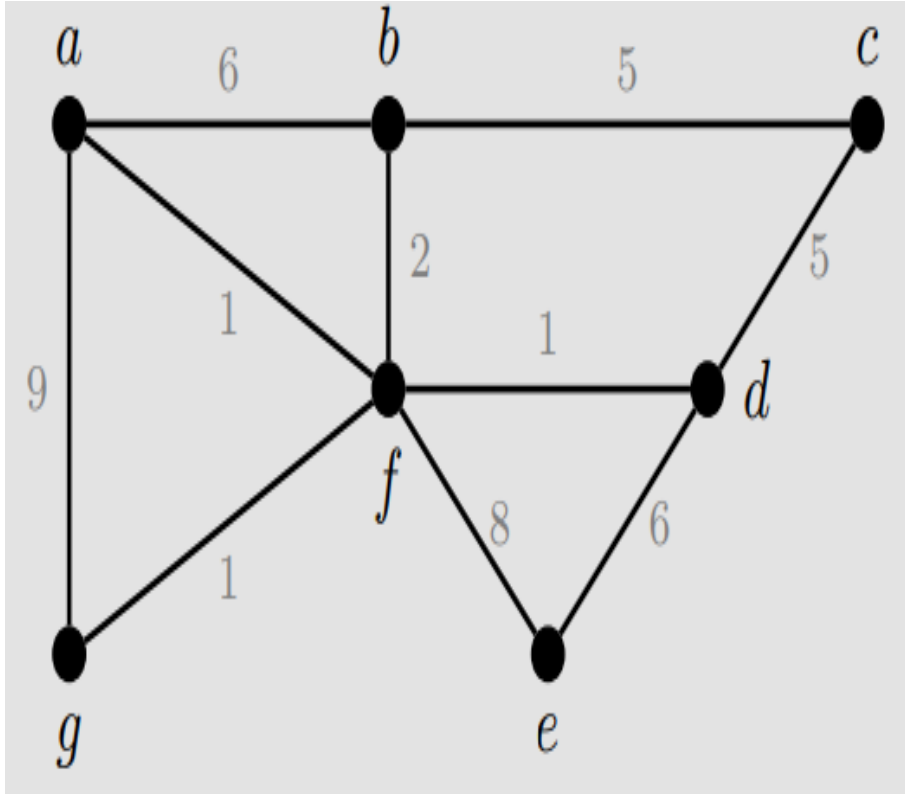
Dr.V.K.Ajay

Example 3 - Dijkstra's Algorithm

Find the shortest distance and shortest path from 'g' to 'c' in the given weighted graph

From 'g' to 'c' shortest distance is
And the shortest path is

g	a	b	c	d	e	f

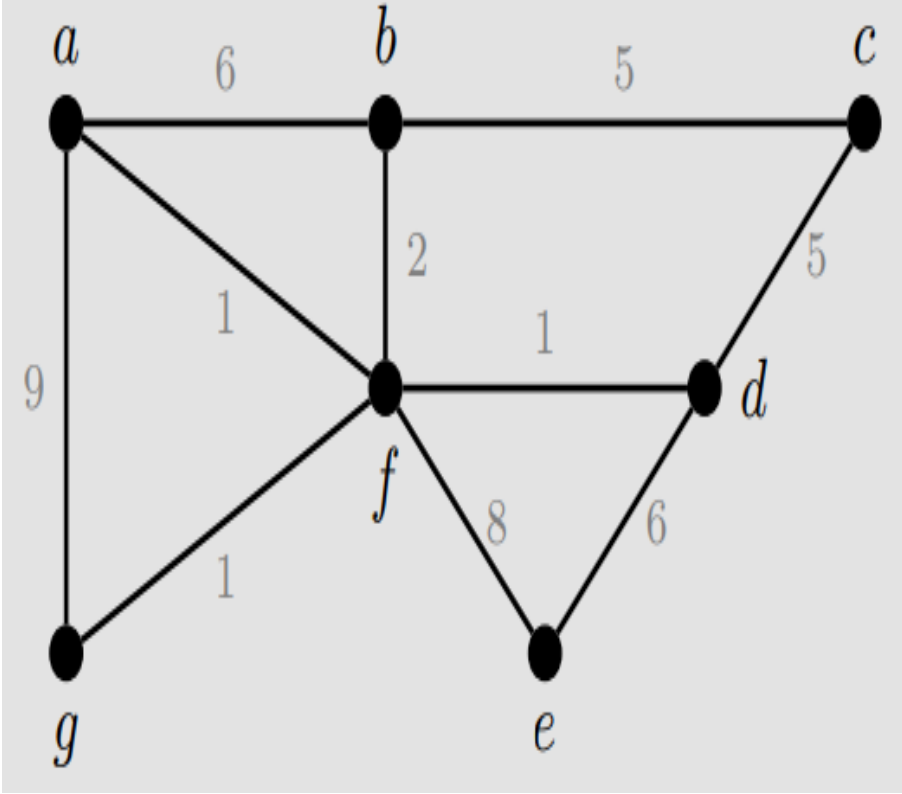


Example 3 - Dijkstra's Algorithm

Find the shortest distance and shortest path from 'g' to 'c' in the given weighted graph

From 'g' to 'c' shortest distance is 7
And the shortest path is g - f – d - c

g	a	b	c	d	e	f
0	∞	∞	∞	∞	∞	∞
0	9	∞	∞	∞	∞	1
0	2	3	∞	2	9	1
0	2	8,3	∞	2	9	1
0	2	3	7	2	8	1
0	2	3	8,7	2	8	1
0	2	3	7	2	8	1



Applications of Dijkstra's Algorithm:

- Google maps uses Dijkstra algorithm to show shortest distance between source and destination.
- In computer networking, Dijkstra's algorithm forms the basis for various routing protocols, such as OSPF (Open Shortest Path First) and IS-IS (Intermediate System to Intermediate System).
- Transportation and traffic management systems use Dijkstra's algorithm to optimize traffic flow, minimize congestion, and plan the most efficient routes for vehicles.
- Airlines use Dijkstra's algorithm to plan flight paths that minimize fuel consumption, reduce travel time.
- Dijkstra's algorithm is applied in electronic design automation for routing connections on integrated circuits and very-large-scale integration (VLSI) chips.

Walks Using Matrices

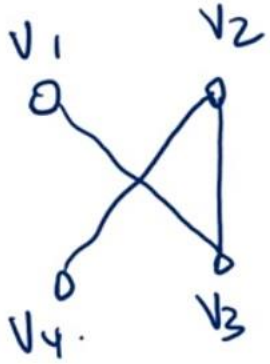
Theorem –

Let G be a graph with adjacency matrix A. Then for any integer $n > 0$ the entry a_{ij} in A^n counts the number of walks from v_i to v_j .

Let A(G) represents the adjacency matrix for graph G. Let $k \in N$.
Then $A^k(G) = (Z_{ij})$ will have Z_{ij} representing the number different of walks of length k from v_i to v_j

Walks Using Matrices – Example 1

$Z_{21} = 1$	1 walk of length 2 from V_2 to V_1	$V_2V_3V_1$
$Z_{11} = 1$	1 walk of length 2 from V_1 to V_1	$V_1V_3V_1$
$Z_{33} = 2$	2 walks of length 2 from V_3 to V_3	$V_3V_2V_3$ and $V_3V_1V_3$
The value inside matrix represent no. of walks and power represents length		

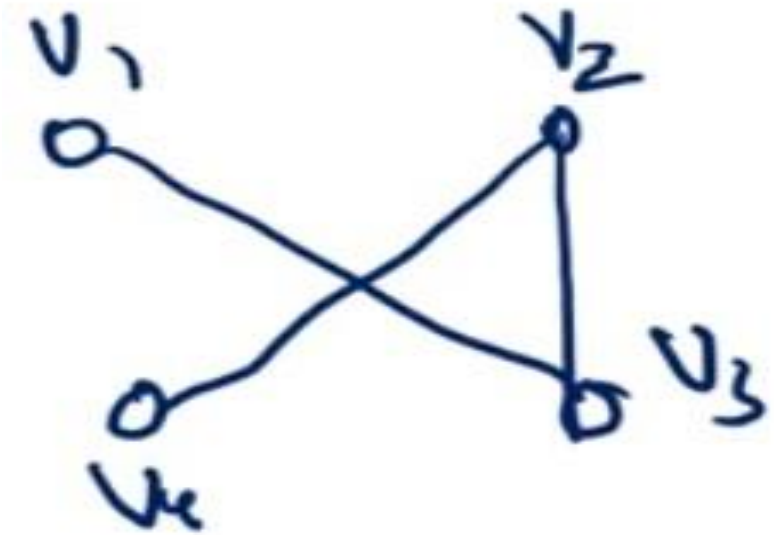


$$A(G) = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$A^2(G) = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 2 & 0 & 0 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

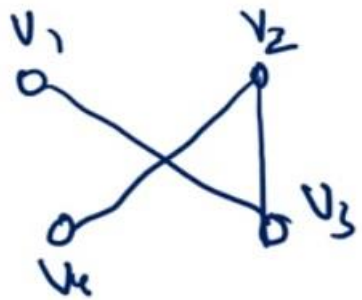
Walks Using Matrices – Example 2

$Z_{11} = 0$		
$Z_{23} = 3$		
$Z_{42} = 2$		



Walks Using Matrices – Example 2

$Z_{11} = 0$	0 walks of length 3 from V_1 to V_1	
$Z_{23} = 3$	3 walks of length 3 from V_2 to V_3	$V_2 V_3 V_1 V_3$ & $V_2 V_4 V_2 V_3$ & $V_2 V_3 V_2 V_3$
$Z_{42} = 2$	2 walks of length 3 from V_4 to V_2	$V_4 V_2 V_3 V_2$ and $V_4 V_2 V_4 V_2$
The value inside matrix represent no. of walks and power represents length		



$$A(G) = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$A^3(G) = \begin{bmatrix} 0 & 0 & 2 & 1 \\ 0 & 0 & 3 & 2 \\ 2 & 3 & 0 & 0 \\ 1 & 2 & 0 & 0 \end{bmatrix}$$