MATHEMATICAL FOUNDATION FOR COMPUTER SCIENCE

Prepared by: Er. Ankit Kharel

Nepal college of information technology

GRAPH THEORY

SHORTEST PATH ALGORITHM:

1. The shortest paths is defined as the smallest weighted path from the starting vertex to the goal vertex out of all other paths in the weighted graph. Here, you can think “weighted” in the weighted path means the reaching cost to the goal vertex (some vertex).

2. One algorithm for finding the shortest path from a starting node to a target node in a weighted graph is **Dijkstra’s algorithm.**

3. The graph can either be directed or undirected.

4. One stipulation to using the algorithm is that the graph needs to have a nonnegative weight on every edge.

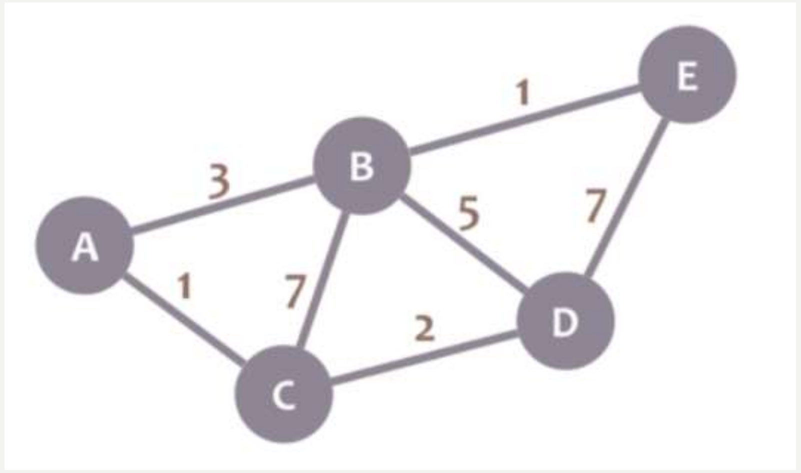
Edge Relaxation:

For the edge from the vertex *u* to the vertex *v*,

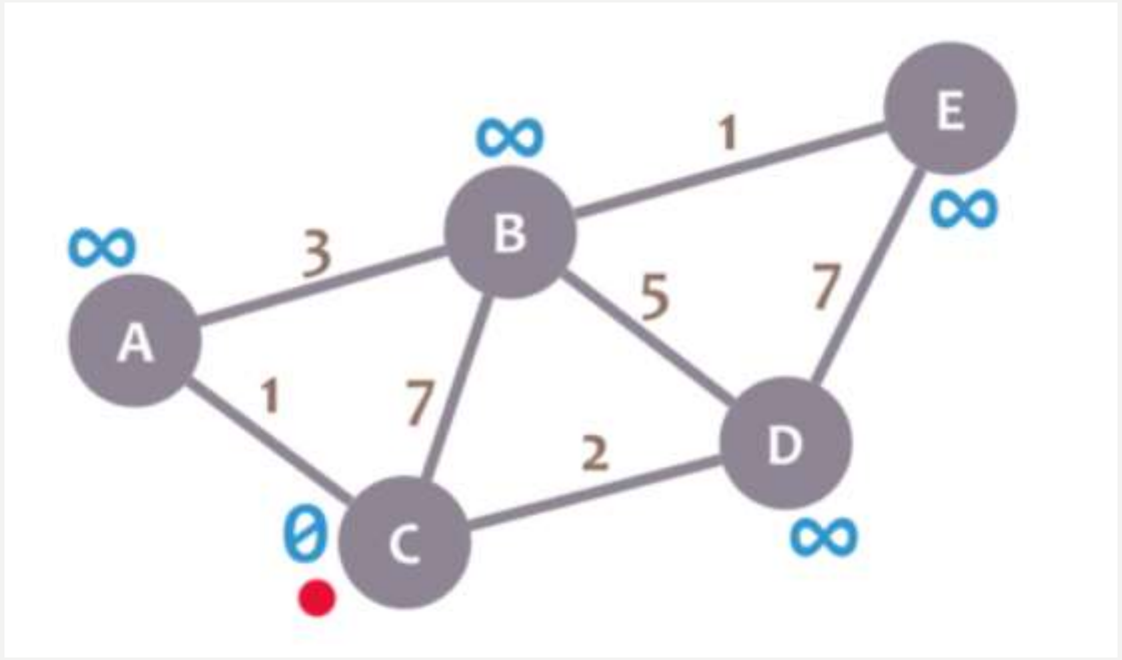
if ($d[u]+w(u,v)<d[v]$) is satisfied,

update $d[v] = d[u]+\mathrm{w}(u,v)$

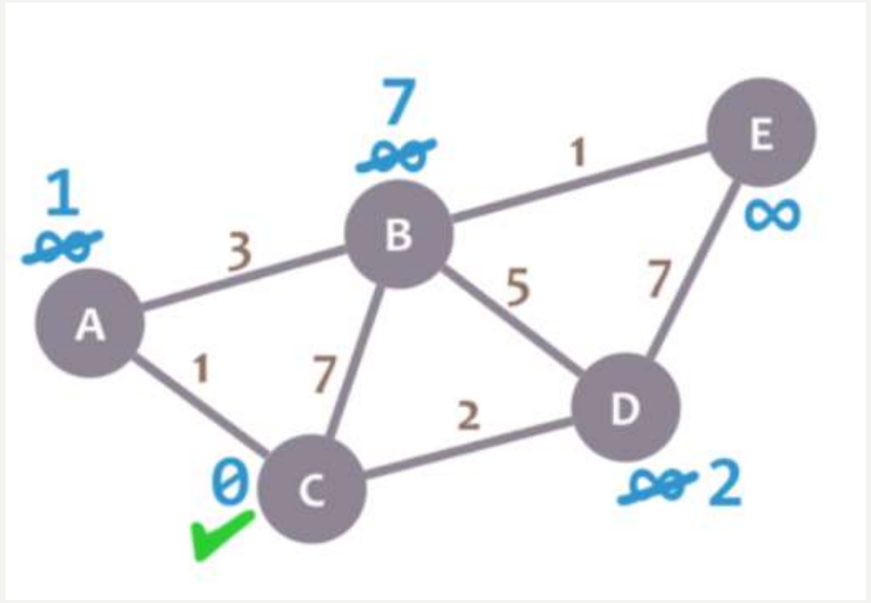
Let's calculate the shortest path between node C and the other nodes in our graph:



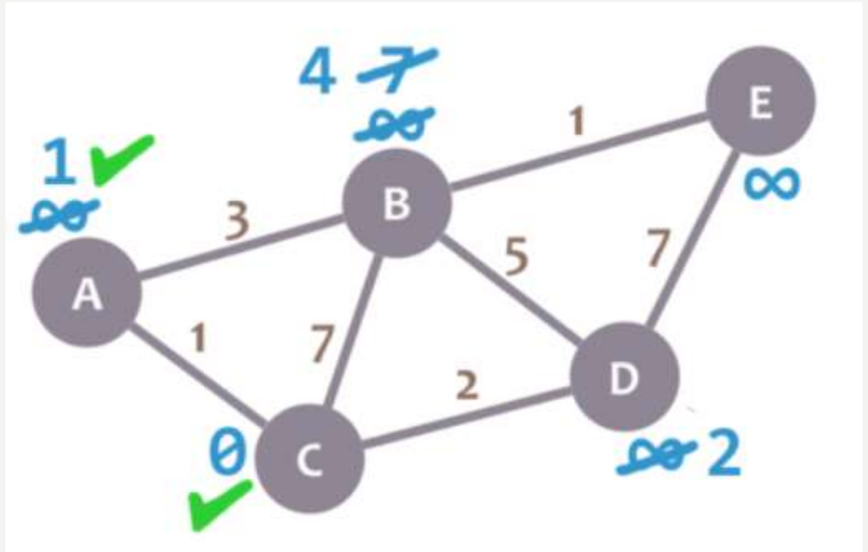
1. For node C, this distance is 0. For the rest of nodes, as we still don't know that minimum distance, it starts being infinity (∞):



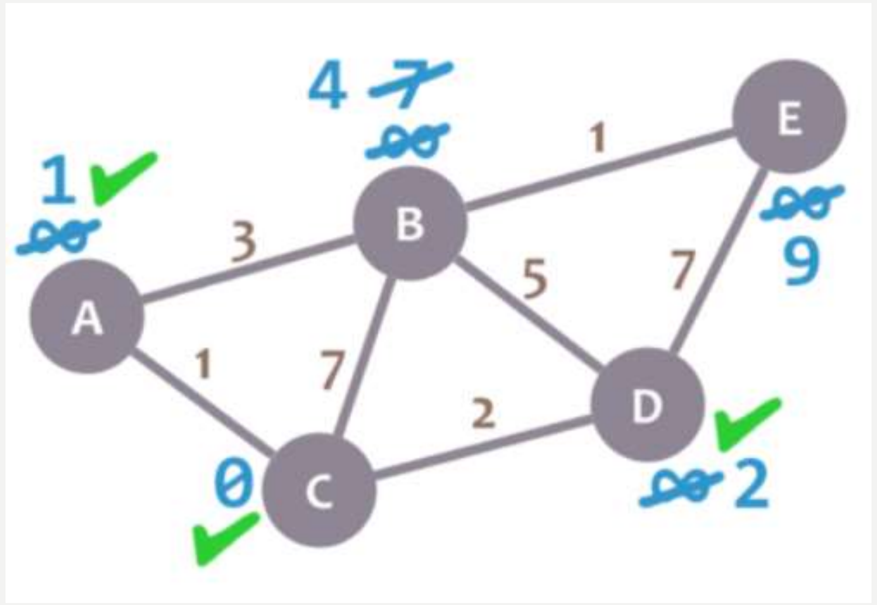
2. Now, we check the neighbors of our current node (A, B and D) in no specific order and perform relaxation if required.



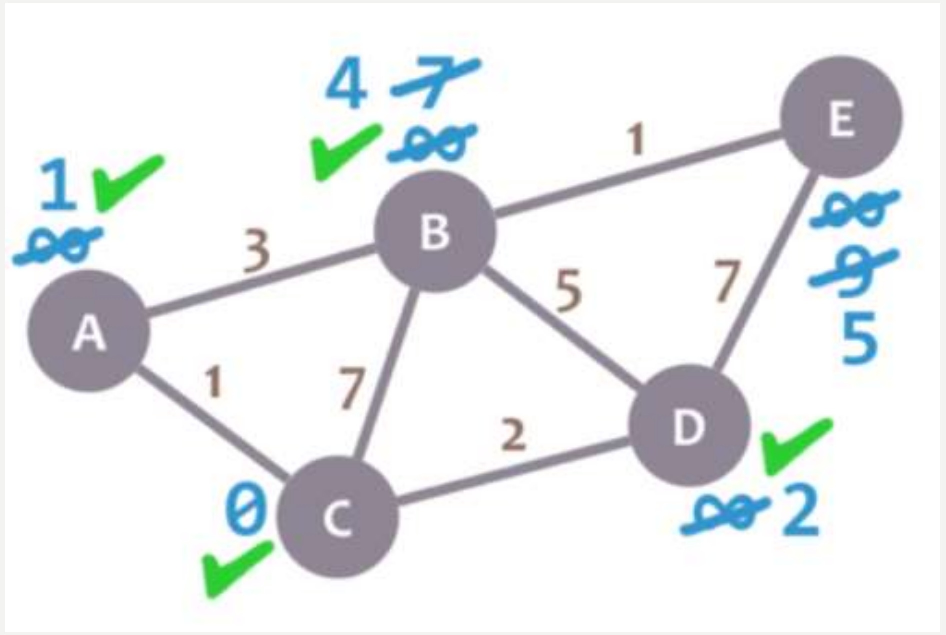
3. We now need to pick a new *current node*. That node must be the unvisited node with the smallest minimum distance .That's A and now we repeat the algorithm. We check the neighbors of our current node(which is B), ignoring the visited nodes.



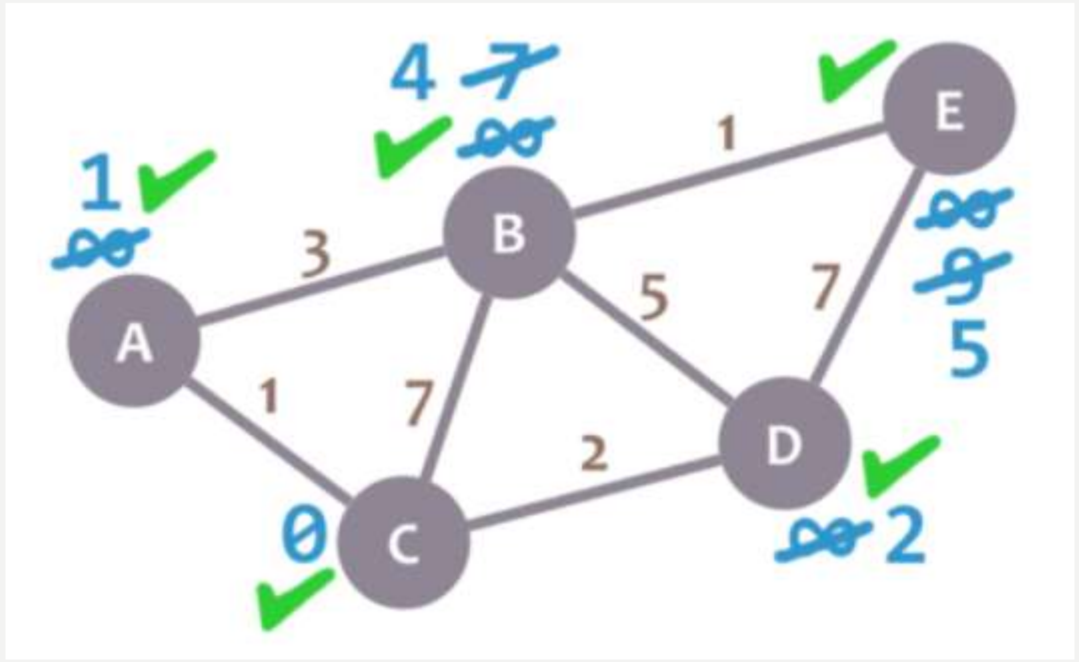
4. Now, select the node D(as it has the minimum distance among unvisited node) and repeat the algorithm.



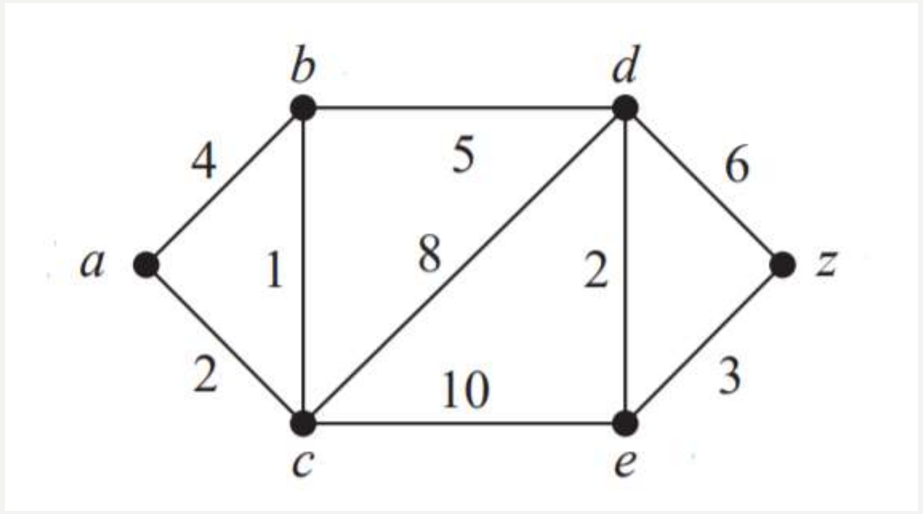
5. Now, select the node B(as it has the minimum distance among unvisited node) and repeat the algorithm.



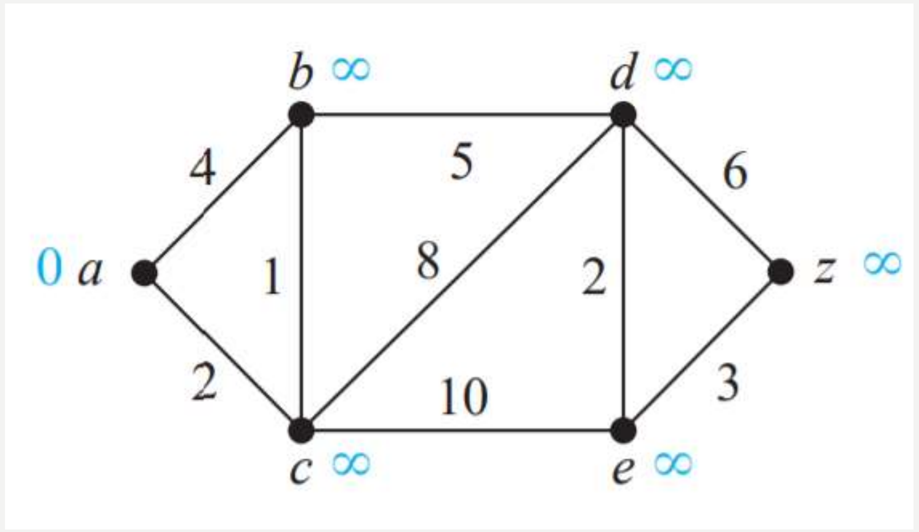
6. E doesn't have any non-visited neighbors, so we don't need to check anything. We mark it as visited.



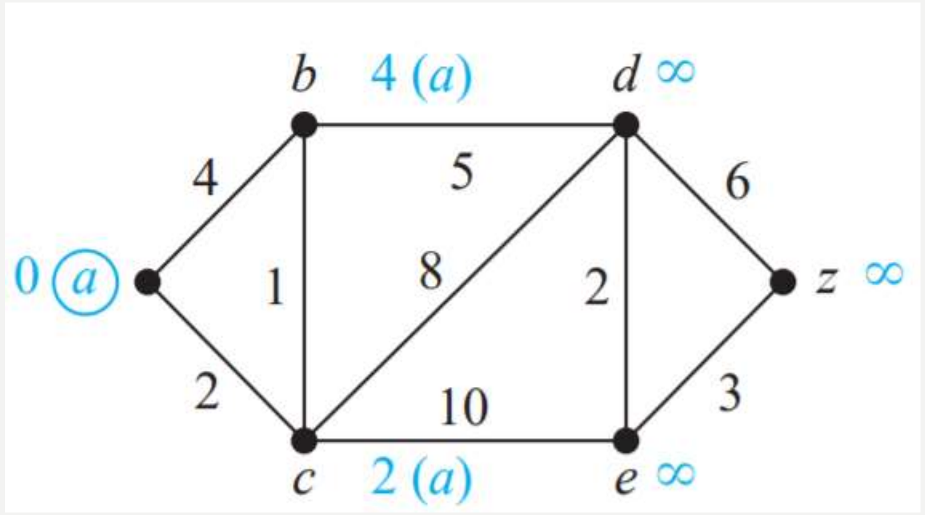
Use Dijkstra's algorithm to find the length of a shortest path between the vertices a and z in the weighted graph displayed in below Figure.



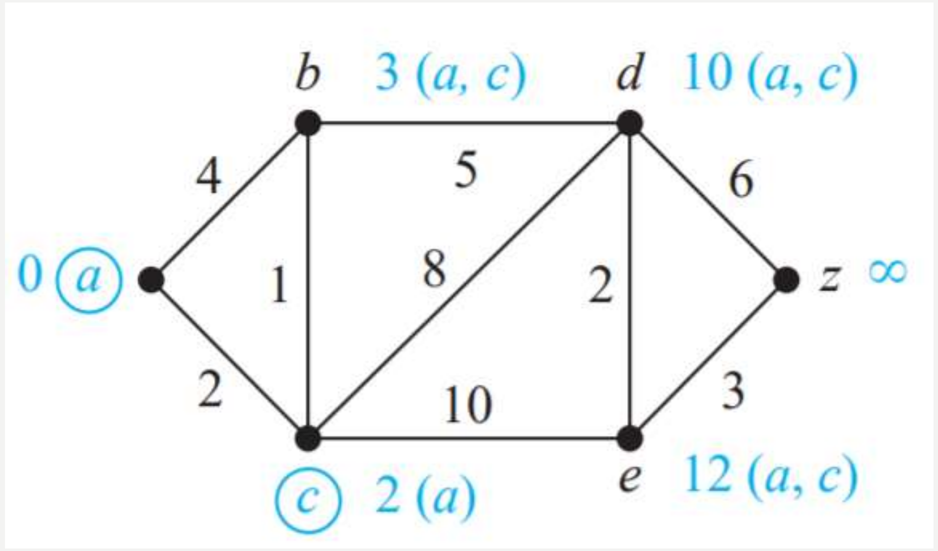
Step: 1



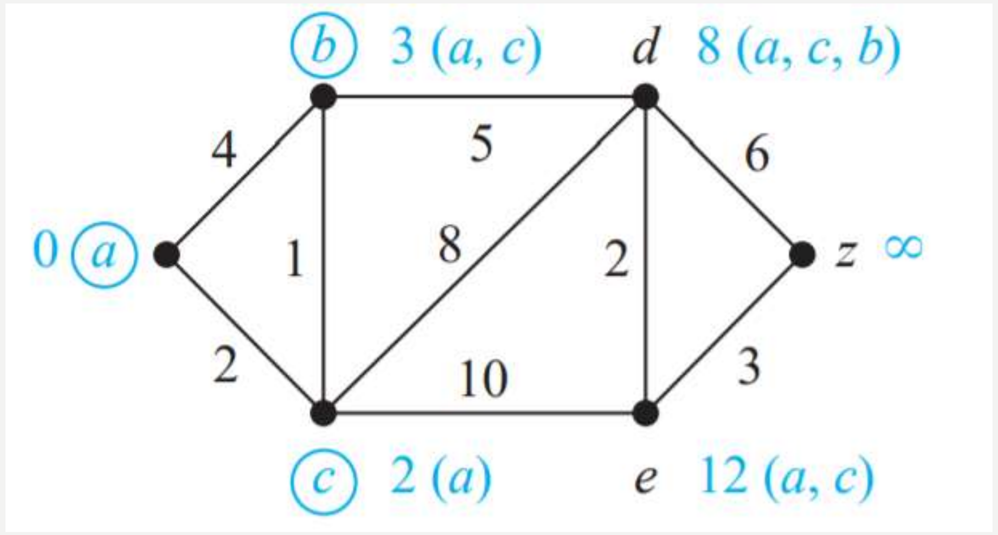
Step: 2


b
4 (a)
d ∞
4
5
6
0 a
1
8
2
z ∞
2
10
3
c
2 (a)
e ∞

Step: 3

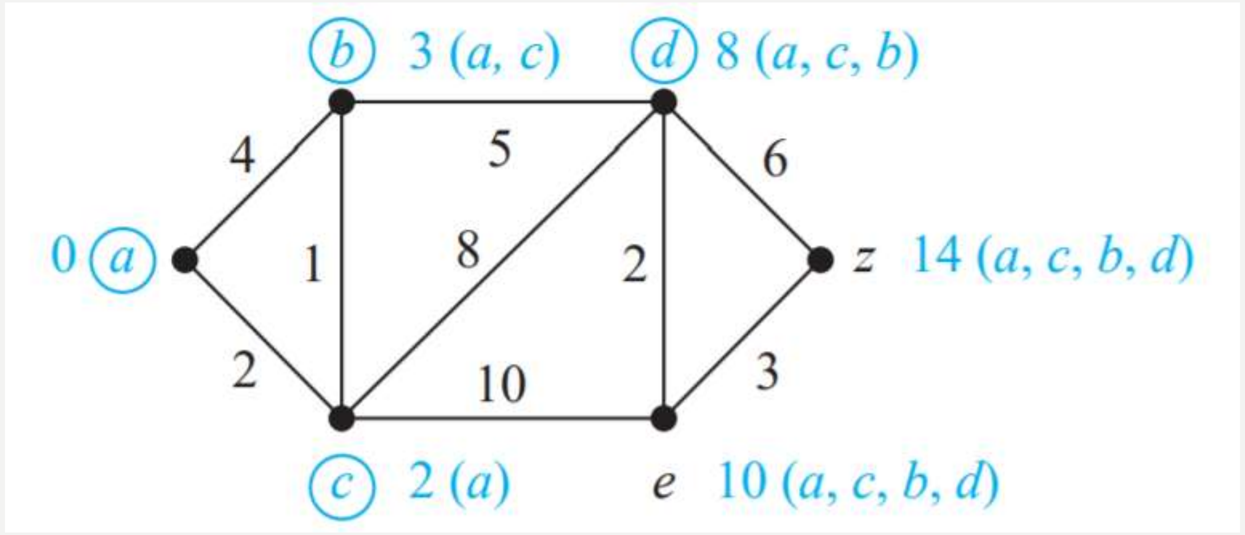

b
3 (a, c)
d
10 (a, c)
4
5
6
0 a
1
8
2
z ∞
2
10
3
c
2 (a)
e
12 (a, c)

Step: 4



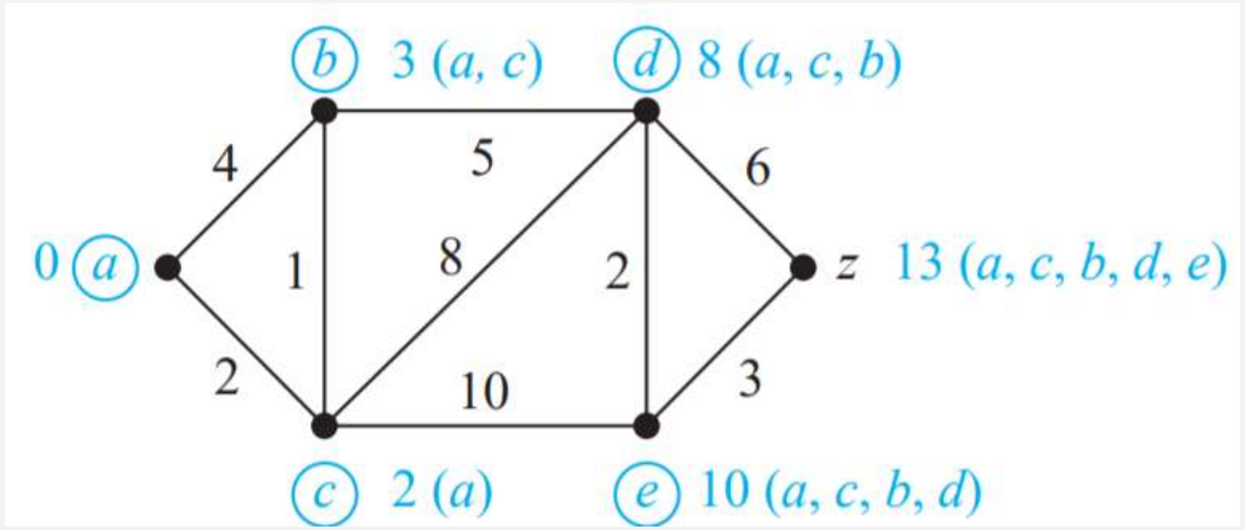
b 3 (a, c)
d 8 (a, c, b)
4
5
6
0 a
1
8
2
z ∞
2
10
3
c 2 (a)
e 12 (a, c)

Step: 5



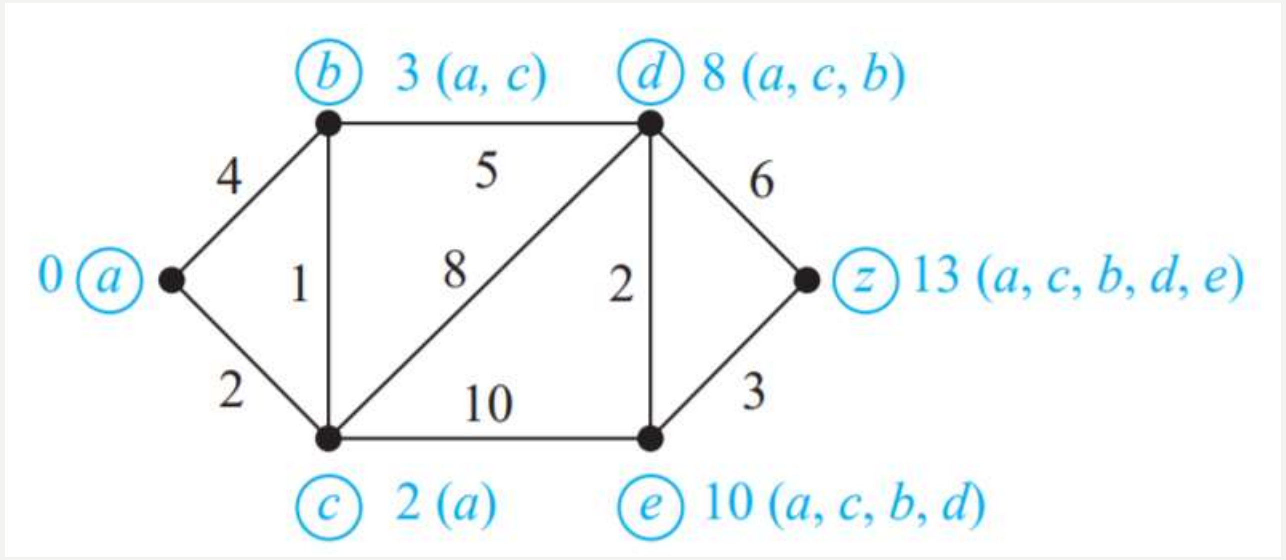
b 3 (a, c)
d 8 (a, c, b)
4
5
6
0 a
1
8
2
z 14 (a, c, b, d)
2
10
3
c 2 (a)
e 10 (a, c, b, d)

Step: 6



b 3 (a, c)
d 8 (a, c, b)
0 a
z 13 (a, c, b, d, e)
c 2 (a)
e 10 (a, c, b, d)
4
5
6
1
8
2
2
10
3

Step: 7



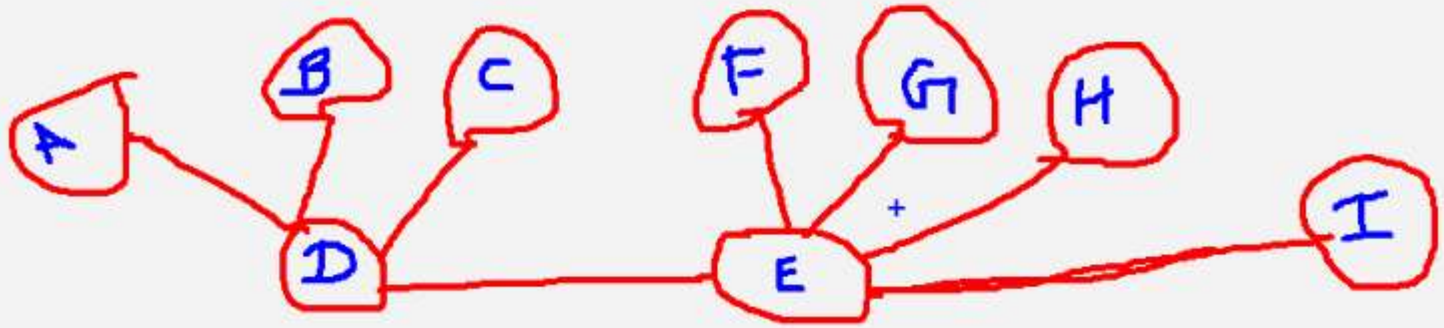
b 3 (a, c)
d 8 (a, c, b)
0 a
z 13 (a, c, b, d, e)
c 2 (a)
e 10 (a, c, b, d)
4
5
6
1
8
2
2
10
3

GRAPH TRAVERSAL:

- **Graph traversal** (also known as **graph search**) refers to the process of visiting (checking and/or updating) each vertex in a graph. Such traversals are classified by the order in which the vertices are visited. Tree traversal is a special case of graph traversal.

1. BFS(Breadth First Search):

The Breadth First Search (BFS) traversal is an algorithm, which is used to visit all of the nodes of a given graph. In this traversal algorithm one node is selected and then all of the adjacent nodes are visited one by one. After completing all of the adjacent vertices, it moves further to check another vertices and checks its adjacent vertices again. BFS uses Queue Data structure.



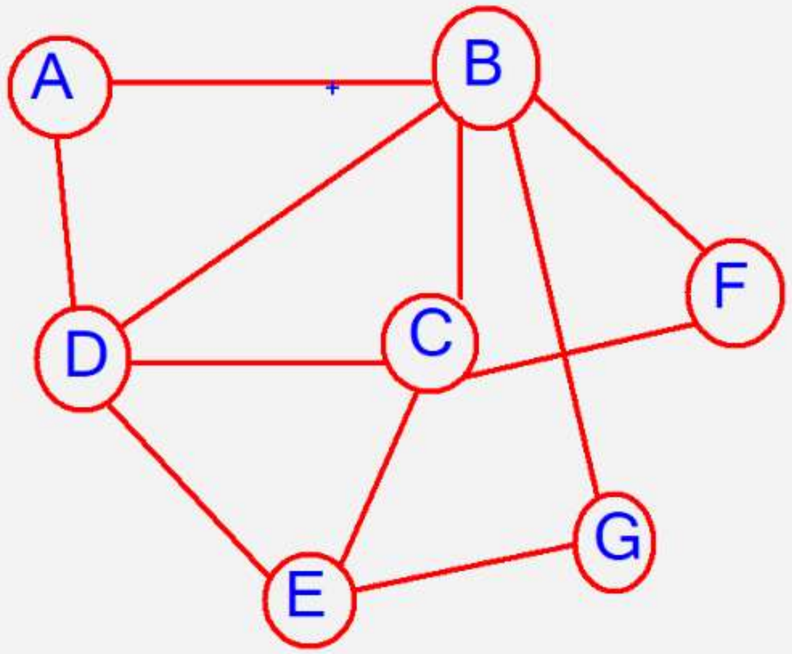
BFS : D, A, B, C, E, F, G, H, I

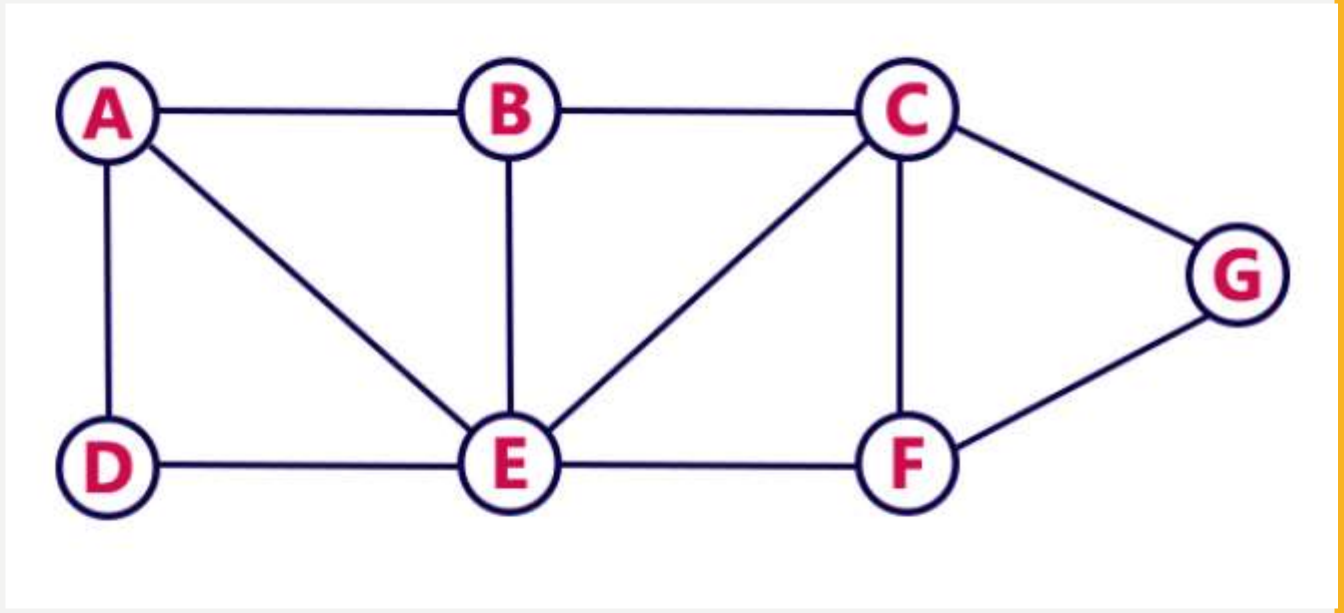
OR

BFS : D, E, C, B, A, I, H, G, F

GRAPH TRAVERSAL:

BFS(Breadth First Search):



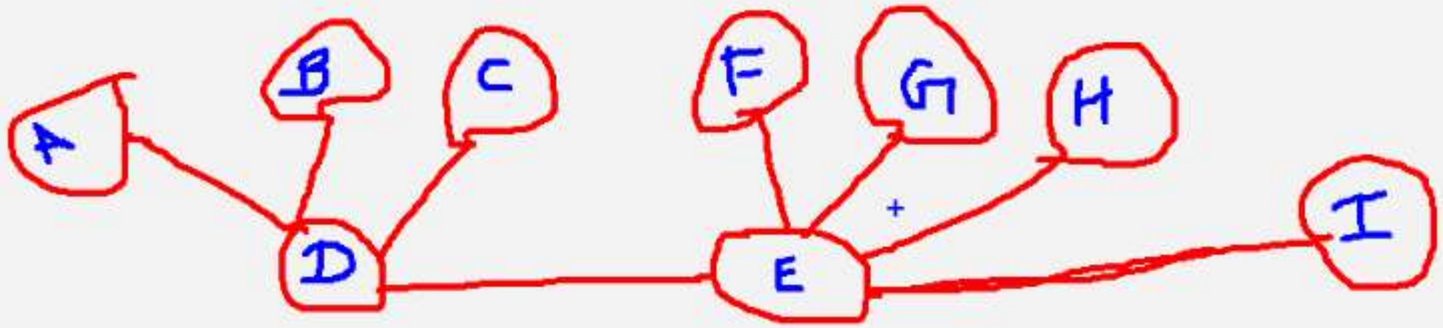


BFS : A, B, D, C, F, G, E

BFS : A, B, D, E, C, F, G

GRAPH TRAVERSAL:

2. **DFS(Depth First Search):**
   The Depth First Search (DFS) is a graph traversal algorithm. In this algorithm one starting vertex is given, and when an adjacent vertex is found, it moves to that adjacent vertex first and try to traverse in the same manner. DFS uses Stack Data structure.



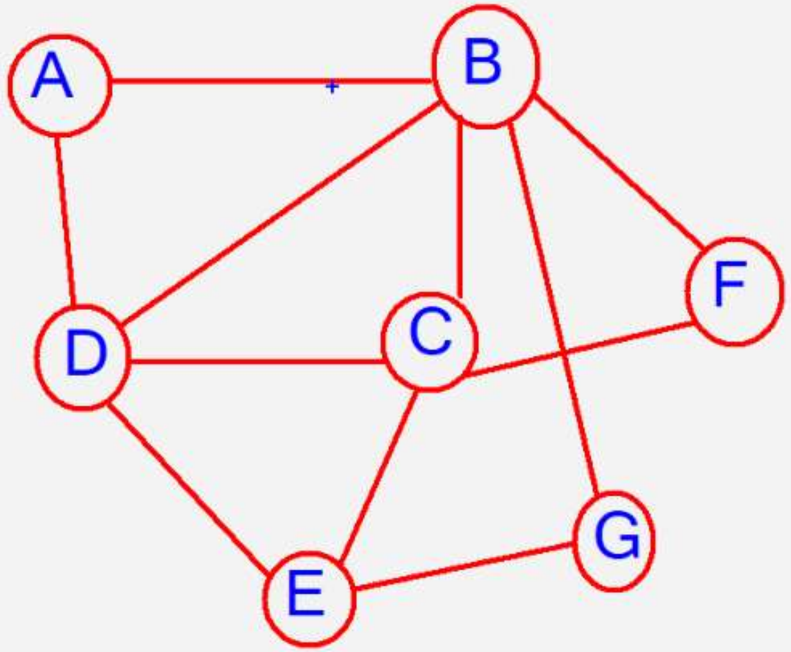
DFS : D, E, I, H, G, F, C, B, A

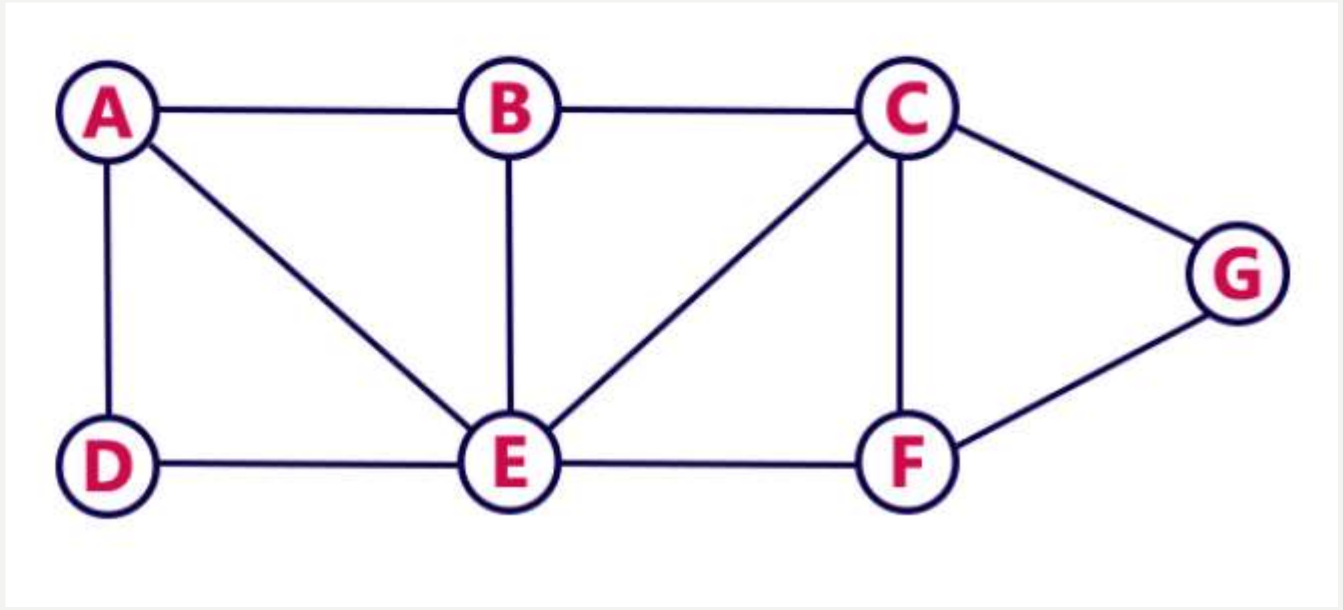
OR

DFS : D, A, B, C, E, F, G, H, I

GRAPH TRAVERSAL:

2. **DFS(Depth First Search):**



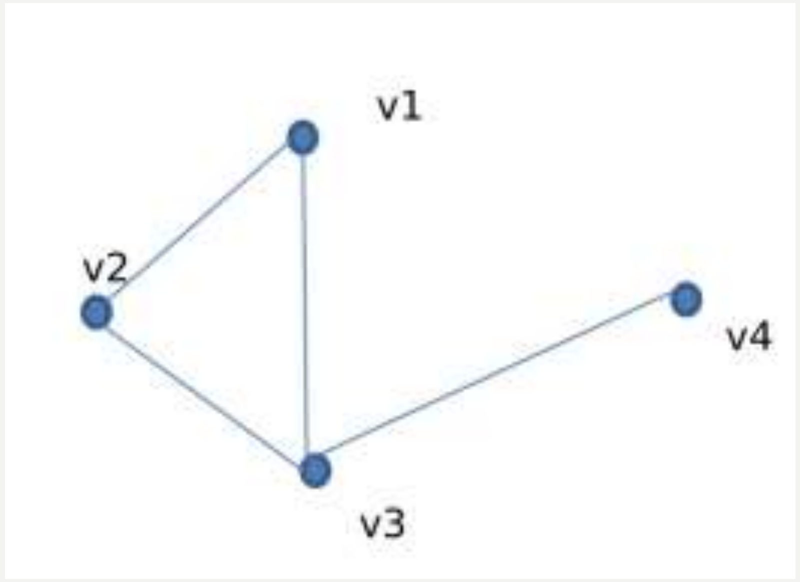


DFS :A, B, D, C ,E, G, F
OR
DFS: A, D, B, F, C, E, G

DFS :A, B, C, G, F, E, D
OR
DFS: A, E, D, B, C, F, G

DEGREE SEQUENCE:

- **Degree Sequence** of a graph is the list of degree of all the vertices of graph in non-increasing or non- decreasing order.

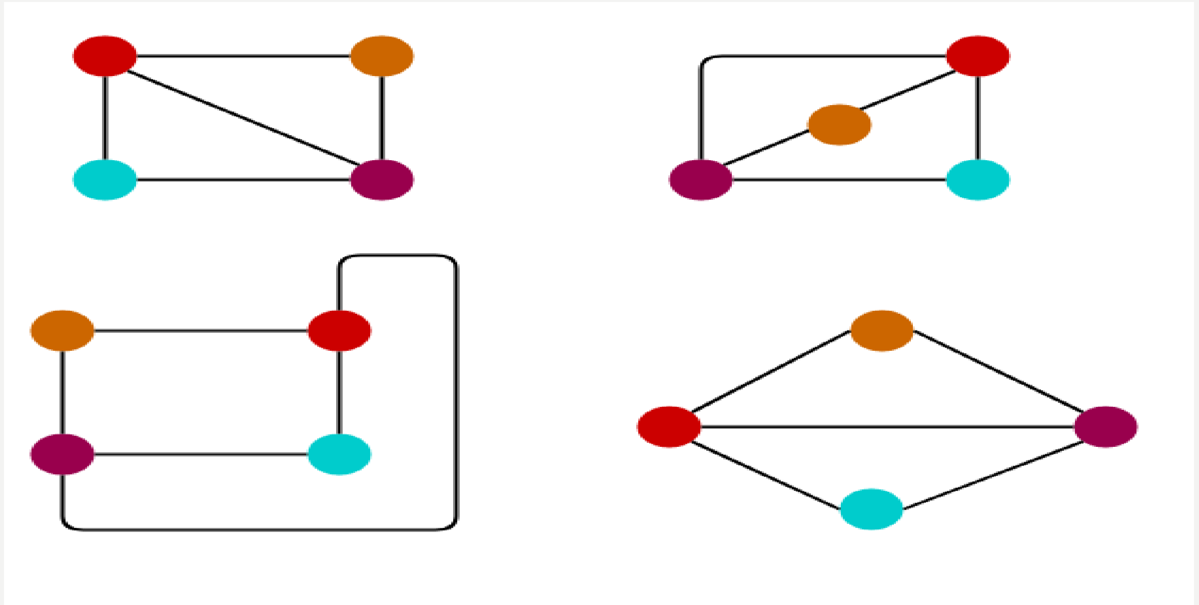


Non- increasing : 3, 2, 2, 1
Non- decreasing : 1, 2, 2, 3

ISOMORPHISM:

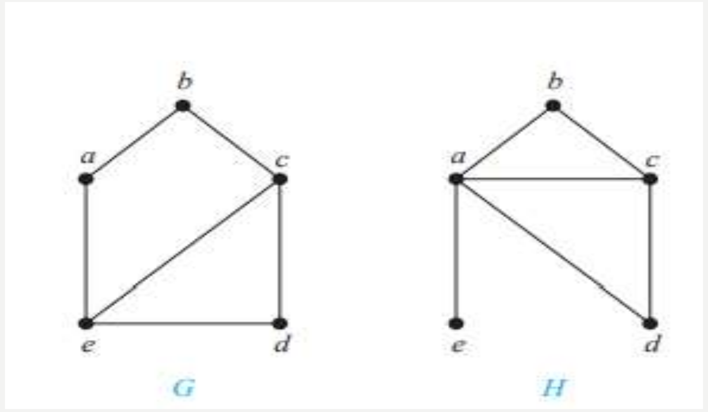
- Graph Isomorphism is a phenomenon of existing the same graph in more than one forms. Such graphs are called as **Isomorphic graphs**. When two simple graphs are isomorphic, there is a one-to-one correspondence between vertices of the two graphs that preserves the adjacency relationship.

- Following are the necessary condition for Two Graphs G and G to be isomorphic:
    1. Both Graphs should have same number of edges
    2. Both Graphs should have same number of vertices
    3. Degree sequence of both Graphs should be same
    4. Their edge connectivity is retained


Graph Isomorphism Example

ISOMORPHISM:

- Show that the graphs G = (V , E) and H = (W, F), displayed in Figure are isomorphic or not.

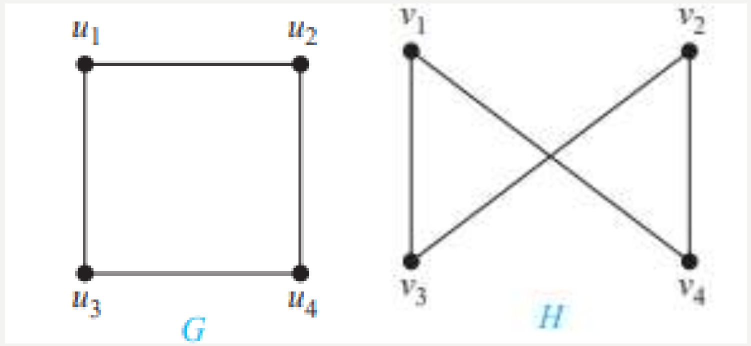


- Both Graph G and H have same number of vertices: 5
- Both Graph has same number of edges: 6
- Degree Sequence of G is : 3, 3, 2, 2, 2
- Degree Sequence of H is : 4, 3, 2, 2, 1

Since the degree sequence of Graphs are different. They are not isomorphic

ISOMORPHISM:

- Show that the graphs G = (V , E) and H = (W, F), displayed in Figure 8, are isomorphic



- Both Graph G and H have same number of vertices: 4
- Both Graph has same number of edges: 4
- Degree Sequence of G is : 2, 2, 2, 2
- Degree Sequence of H is : 2, 2, 2, 2
- Finding Correspondence between vertices:

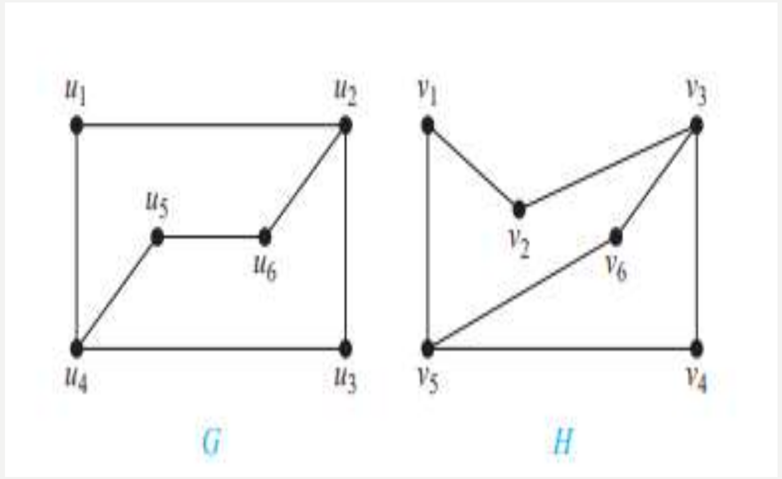
  (i)u_1=v_1
  (ii)u_2=v_4
  (iii)u_3=v_3
  (iv)u_4=v_2

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

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

Now, Find the adjacency matrix For G with ordering((u_1, u_2, u_3, u_4) and H for ordering(v_1,v_4,v_3,v_2)
Since both matrix are same the graph are isomorphic

- Show that the graphs G = (V , E) and H = (W, F), displayed in Figure are isomorphic or not



- Both Graph G and H have same number of vertices: 6
- Both Graph has same number of edges: 7
- Degree Sequence of G is : 3, 3, 2, 2, 2, 2
- Degree Sequence of H is : 3, 3, 2, 2, 2, 2
- Finding Correspondence between vertices:

  (i)$u_1=v_6$
  (ii)$u_2=v_3$
  (iii)$u_3=v_4$
  (iv)$u_4=v_5$
  (v)$u_5=v_1$
  (vi)$u_6=v_2$

$$G = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix} \qquad H = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Now, Find the adjacency matrix For G with ordering((u_1, u_2, u_3, u_4, u_5, u_6) and H for ordering(v_6, v_3 v_4, v_5, v_1, v_2)
If both matrix are same then the graph are isomorphic