Unit 8: Bridges & Algorithms

UNIT STRUCTURE

8.1 Learning Objectives
8.2 Introduction
8.3 Bridges
8.4 Breadth First Algorithm
8.5 Depth First Algorithm
8.6 Kruskal's Algorithm
8.7 Prims Algorithm
8.8 Let Us Sum Up
8.9 Answers to Check Your Progress
8.10 Further Reading
8.11 Model Questions

8.1 LEARNING OBJECTIVES

After going through this unit, you will be able to-

- definition and properties of bridges in graph theory.
- different types and workings of algorithms like breadth first search, depth first search etc.

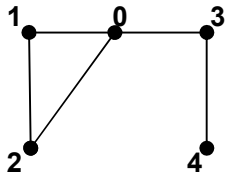
8.2 INTRODUCTION

Graph traversal and minimum spanning tree algorithms are fundamental concepts in graph theory, with numerous applications in computer science, engineering and other fields. In this chapter, we explore some of three algorithms and understand their working.

8.3 BRIDGES

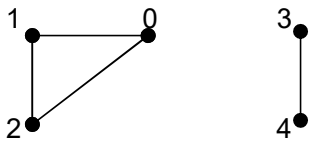
In graph theory, a bridge (also known as a cut edge or articulation edge) is an edge in a connected graph whose removal makes the graph disconnected.

Example:

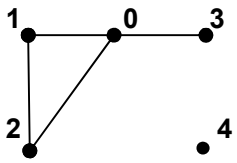


Here the bridges are:

(i) (0, 3)



(ii) (3, 4)



Clearly, we can observe that the removal of a bridge increases the number of connected components of the graph.

Properties of Bridges:

(i) A bridges cannot be a part of any cycle in the graph.

(ii) If an end point is a Bride, then both of its end-points are typically cut vertices (unless one of the end points is a leaf modem I, e, a vertex of degree one)

(iii) A graph which is connected and has 'n' vertices, can have at most (n-1) brides.

(iv) A bridge is a single edge whose removal disconnects a connected graph while a cut-set of edges whose removal disconnects the graph. Therefore, every bridge is a cut set of size one.

(v) A graph with no bridges is called `bridgeless' Clearly, a bridgeless graph is 2-edge connected.

W CHECK YOUR PROGRESS

Q 1. Identify the bridges in the following graph:



8.4 BREADTH FIRST ALGORITHM

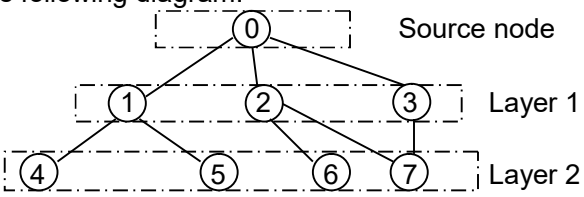
Breadth First search (BFS) is a traversing algorithm where we start traversing from a selected node (source or starting node) and traverse graph layer wise, thus exploring the neighbour nodes (nodes which are directly connected to the source node). We then move to the next level neighbour nodes.

As the name suggests, we are required to traverse the graph breadth wise as follows:

(i) First move horizontally and visit all the nodes of the current layer.

(ii) Move to the next layer.

Consider the following diagram:



In BFS, we must trovers all the nodes in layer 1 before we move on to the nodes in Layer 2.

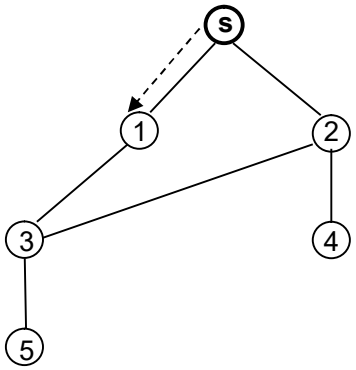
We start traversing from 0 and visit its child nodes 1, 2 and 3. We store them in the order in which they are visited. This allows us to visit the child nodes of 1 first (I, e 4 & 5), then of 2 (i. e 6 & 7) and then of 3 (i. e 7) etc.

To make the process easy, we use a queue to store the node and mark it as 'visited' until all its neighbours (vertices that all directly connected to it) are marked. The queue follows the First In First out queuing method and therefore, the neighbours of the will be visited in the order in which they were in the order in which they were inserted in the node, i.e, the node that was inserted first will be visited first and so on.

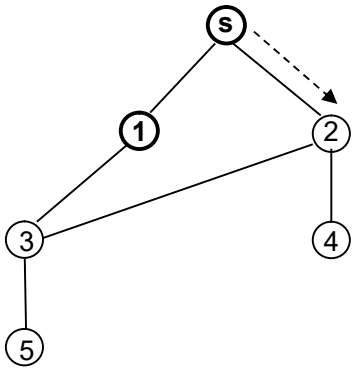
Example:

Traversing Process

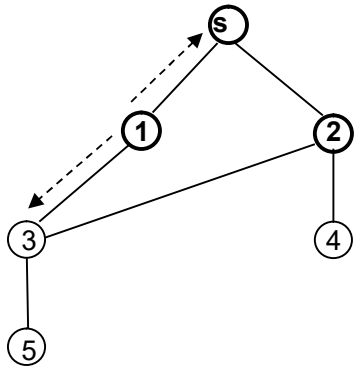
A.



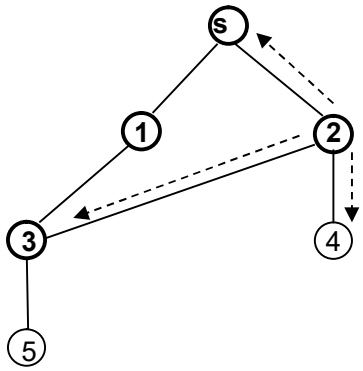
B.



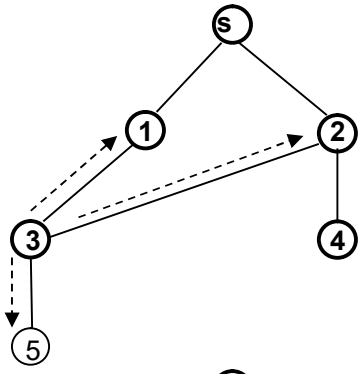
C.



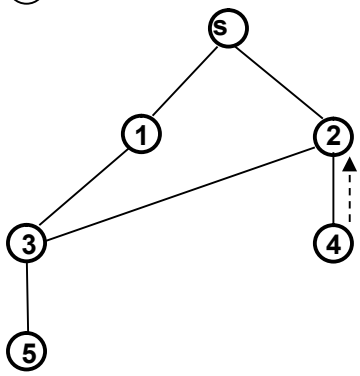
D.



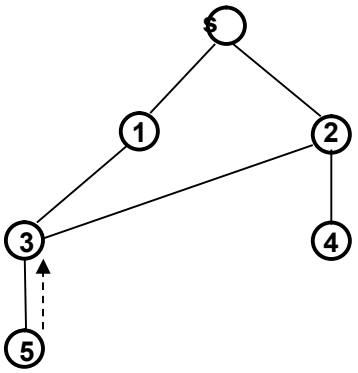
E.



F.



G.



Traversing will start from the source node and push 'S' in queue. S will be marked as visited.

First iteration

- S will be popped from queue
- Neighbours of S, i.e, 1 and 2 will be traversed
- 1 and 2 which have not been traversed earlier are traversed. They will be :
  (i) Pushed in the queue
  (ii) 1 and 2 will be marked visited

Second iteration

- 1 is propped from the queue
- Neighbours of 1 i, e S and 3 are traversed
- S is ignored because it is marked as 'visited'
- 3 which has not been traversed earlier is traversed. It is
  (i) Pushed in the queue
  (ii) Marked as visited.

Third iteration

- 2 is popped from the queue.
- Neighbours of 2 i. e s, 3 and 4 are traversed.
- 3 and s are ignored because they are marked as 'visited'.
- 4, which has not been traversed earlier, is traversed. It is:
  (i) Pushed in the queue
  (ii) Marked as visited.

Fourth iteration

- 3 is popped from the queue.
- Neighbours of 3, i. e, 1, 2 and 5 are traversed.
- 1 and 2 are ignored because they are marked as 'visited'.
- 5, which is not traversed earlier, is traversed. It is :
  (i) Pushed in the queue
  (ii) Marked as visited.

Fifth iteration

- 4 will be popped from the queue.

- Neighbours of 4, i.e, 2 is traversed.
- 2 is ignored because it is already marked as 'visited'.

Sixth iteration

- 5 is popped from the queue.
- Neighbours of 5, i. e 3 is traversed
- 3 is ignored because it is already marked 'visited'.

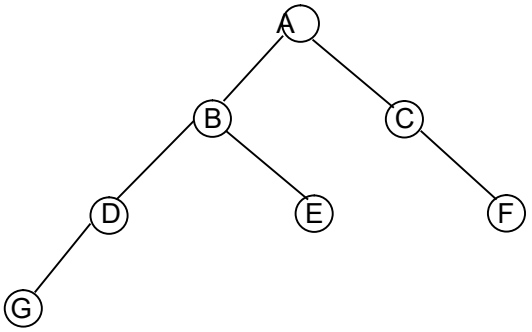
The queue is empty now and it comes out of the loop. All the nodes have been traversed by using BFS.

8.5 DEPTH FIRST ALGORITHM

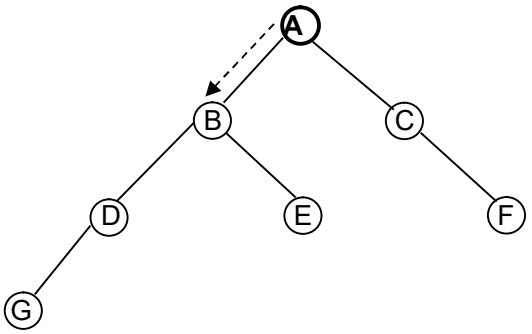
It is a powerful technique of systematically traversing the edges of a given graph such that each vertex is visited at least once. This technique is also called backtracking on a graph. It explores as far as possible along each branch before backtracking.

Here, backtrack means that when we are moving forward and there are no more nodes along the current path, then we move backwards on the same path to find nodes to traverse. All the nodes will be visited on the current path till all the unvisited nodes have been traversed after which the next path shall be selected.

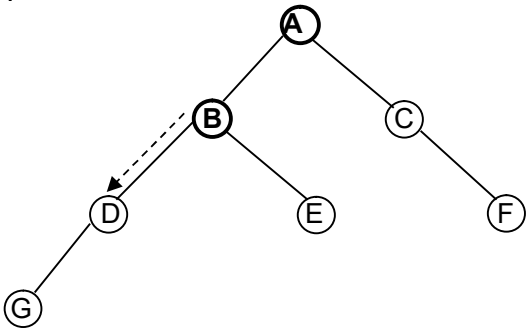
Example:



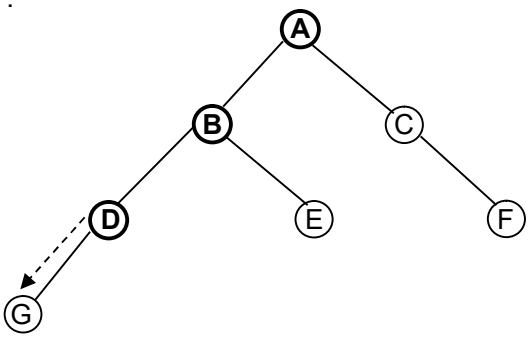
Step 1 : Start from the source node A



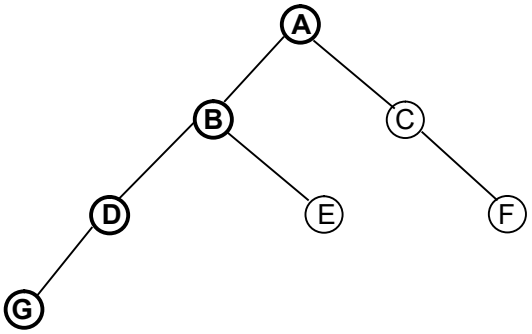
Step 2 :



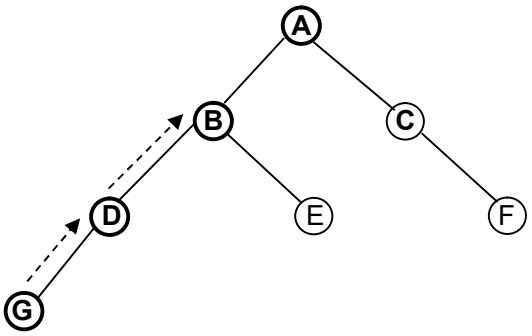
Step 3 :



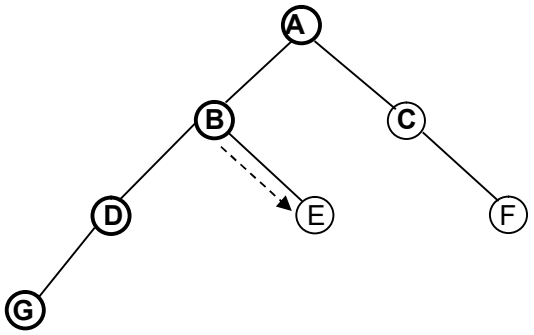
Step 4 :



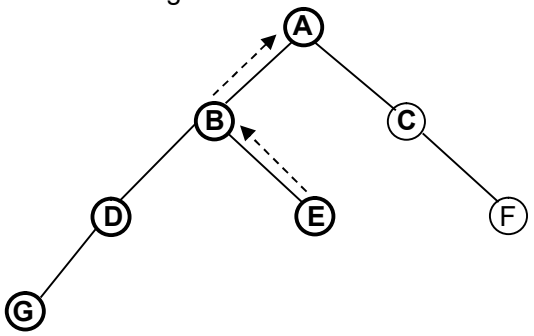
Step 5 : Backtracking



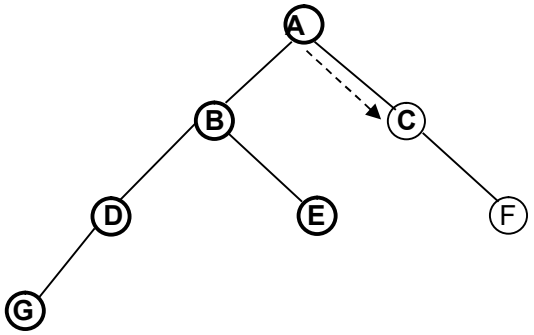
Step 6 :



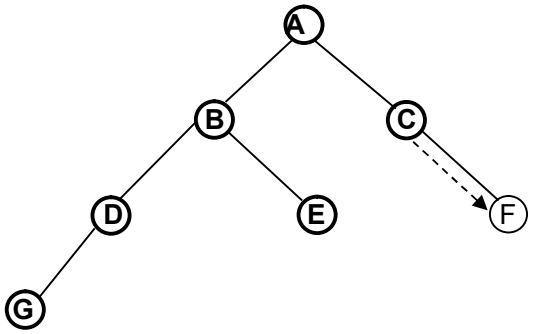
Step 7 : Backtracking



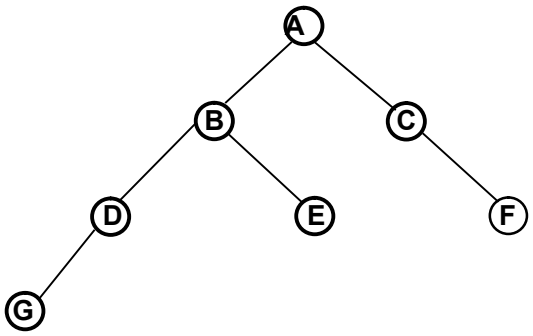
Step 8 :



Step 9 :



Step 10 :



Thus, all the vertices have been traversed.
∴ The traversed path is :
A B D G D B E B A C F
Note : The same example if we do in terms of BSF, then the traversed path will be:
A B A C F C A B E B D G

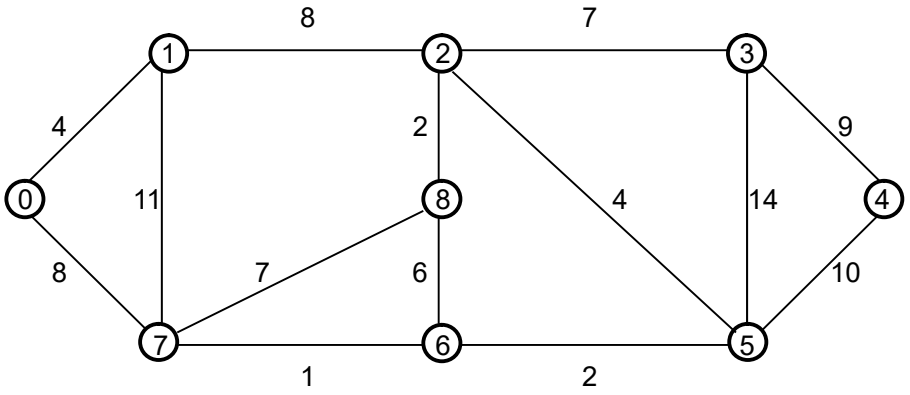
8.6 KRUSKAL'S ALGORITHM

Kruskal's algorithm is used to find a minimum spanning tree (MST) in a connected, undirected, edge –weighted graph.
The steps involved are:

- Sort all the edges in an increasing order.
- Pick the smallest edge. Check if it forms a cycle with the spanning tree forms so far. If cycle is not formed, include this edge. Else, discord it.
- Repeat step 2 until there are (ν-1) edges (ν is number of vertices) in the spanning tree.

This algorithm picks the minimum weighted edge at first and the maximum weighted edge at last. Thus it makes a locally optimal choice in each step in order edge at last. Thus, it makes a locally optimal choice in each in order to find the optimal solution.

Example:



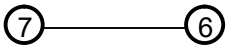
The graph contains 9 vertices and 14 edges. So the minimum spanning tree formed will be having (9 – 1) = 8 edges.

Sorting:

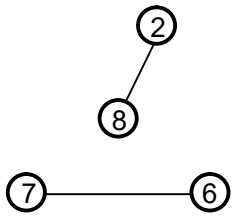
Weight	Source	Destination
1	7	6
2	8	2
2	6	5
4	0	1
4	2	5
6	8	6
7	2	3
7	7	8
8	0	7
8	1	2
9	3	4
10	5	4
11	1	7
14	3	5

Now, pick all the edges one by one from the sorted list of edges.

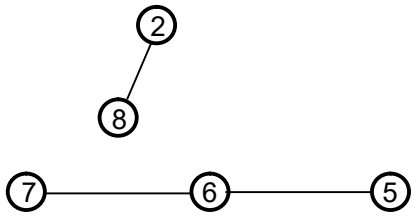
Step 1 : Pick edge 7 – 6 , No cycle is formed, So include it.



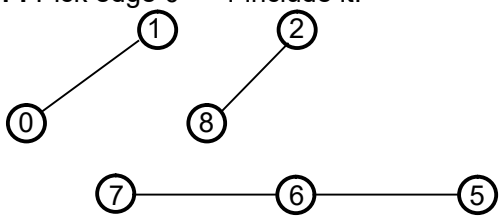
Step 2 : Pick edge 8 ------- 2. No cycle is formed so include it.



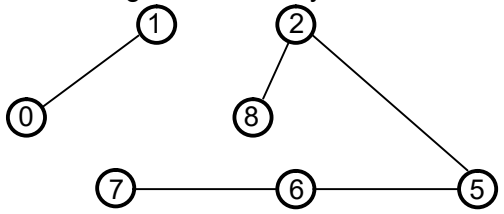
Step 3 : Pick edge 6 — 5. No cycle is formed so include it.



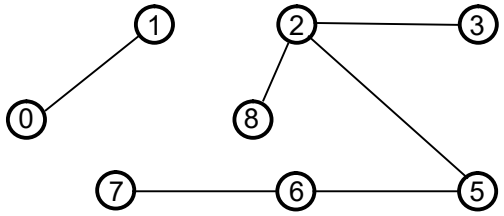
Step 4 : Pick edge 0 — 1 include it.



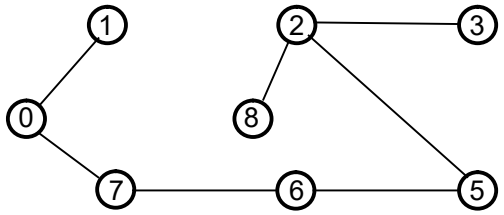
Step 5 : Pick edge 2 — 5. No cycle is formed So include it.



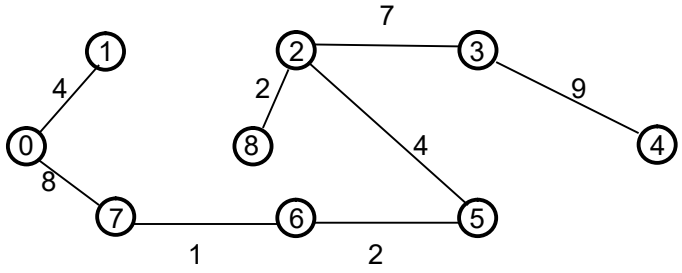
Step 6 : Pick edge 8 — 6 . Since including this edge results in a cycle, so we discard it. Pick edge 2 — 3. Since no cycle is formed, so include it.



Step 7 : Pick edge 7 — 8. Since including this edge results in a cycle, discard it. Pick edge 0 - 7. No cycle is formed, so include it.



Step 8 : Pick edge 1 — 2. Since including this edge results in the cycle, discard it. Pick edge 3 — 4. No cycle is formed so include it.



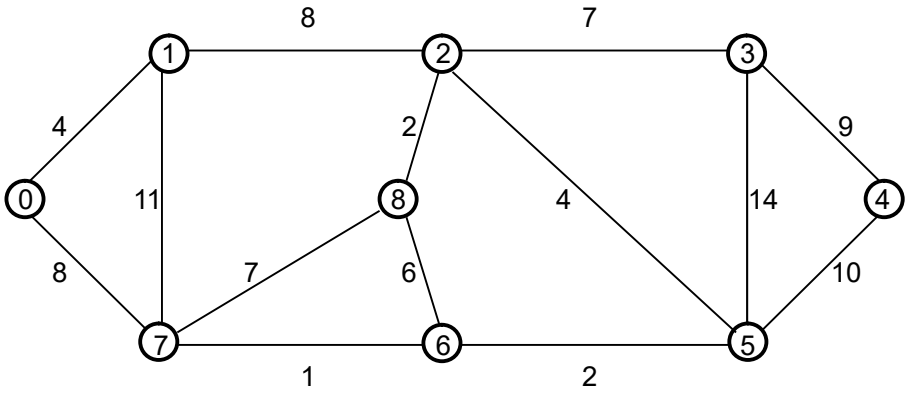
Since the number of edges = 8, So the algorithm stops here.
∴ Length = 4+8+1+2+4+2+7+9
= 37

8.7 PRIMS ALGORITHM

Prim's Algorithm is used to find a minimal spanning tree for a weighted undirected graph.
How the algorithm works:

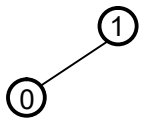
- We start with an empty MST and an arbitrarily chosen starting vertex. This vertex is added to this vertex is added to the MST.
- Repeatedly we select the edge with the minimum weight that connects a vertex already in the MST to a vertex not yet in the MST.
- We add this minimum weight edge and the newly connected vertex to the MST.
- This process is continued until all vertex of the graph are included in the MST.

Example:



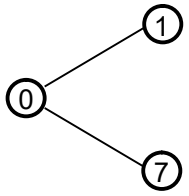
Step 1 : Start with vertex 0.
We have two edges from 0 :
0 – 1 with weight 4
0 – 7 with weight 8

We pick edge 0 – 1 as it has minimum weight. So we add vertex 1.



Step 2 : We now have edges
1 – 2 with weight 8
1 – 7 with weight 11
0 – 7 with weight 8.

We pick either 1 – 2 or 0 – 7 since they have minimum weight. They have same minimum weight.

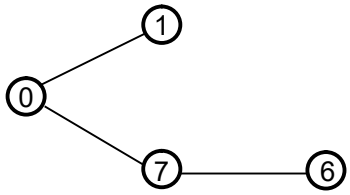


So we added vertex (7)

Step 3 : We now have edges.

7 – 6 with weight 1
7 – 8 with weight 7
1 – 2 with weight 8
1 – 7 with weight 11

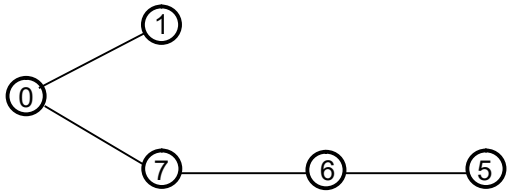
We pick edge 7 – 6 and thus add vertex (6)



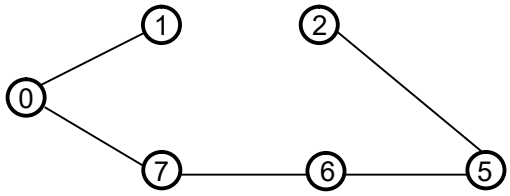
Step 4 : We now have edges

6 – 5 with weight 2
7 – 8 with weight 7
1 – 2 with weight 8
6 – 8 with weight 6

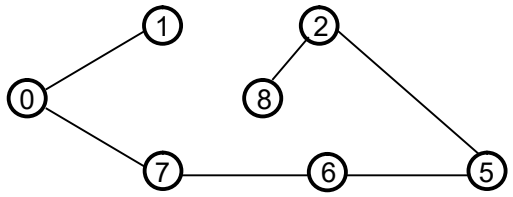
Therefore, we pick edge 6 – 5 with minimum weight and add vertex 5.



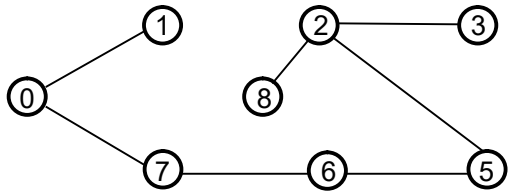
Step 5 : We next choose edge 5 – 2 with weight 4 (as it is minimum) and add vertex 2.



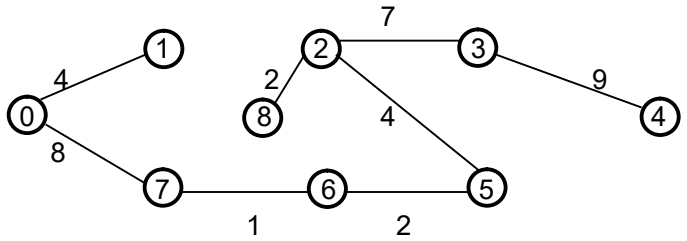
Step 6 : We choose edge 2 – 8 with weight 2 and thus vertex 8.



Step 7 : We choose edge 2 – 3 with weight 7 and thus add vertex 3.



Step 8 : Finally we choose 3 – 4 with weight 9.



All the vertices are included, so the MST is complete and thus, the algorithm stops.

∴ Length (Total weight)

= 4+8+2+1+2+4+7+9

= 37.

2 8.8 LET US SUM UP

- In this unit, we explored bridges and certain algorithms which are crucial in graph theory, like prim's algorithm, Breadth search algorithm, Depth search algorithm, Depth search algorithm, Kruskals algorithm etc. These concepts are important in understanding graph structures and has important applications in:
- Network Design and Optimization: Designing efficient networks, Optimization routes and ensuring reliability.
- Data Analysis and clustering: Analyzing graph structures, clustering data and identifying patterns.
- Route planning and logistics: Finding shortest paths, optimizing routes and scheduling.

K 8.9 ANSWERS TO CHECK YOUR PROGRESS

Answer to Q 1:



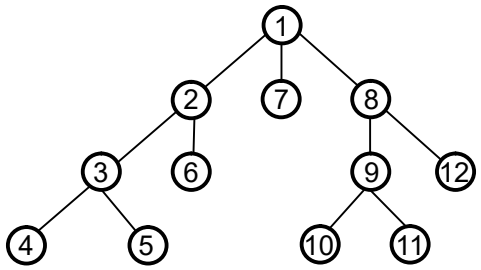
The bridges are : 0 – 1,
1 – 2,
2 – 3.

O8.10 FURTHER READING

1) N. Deo, “Graph Theory with Applications to Engineering and Computer Science’’
2) M. Dror, “Arc Routing: Theory, Solutions & Applications’’
3) Thomas H. Cormen, C. E. Leiserson, R. L. Rivrst, C Stem, ``Introduction to Algorithm’’.

M8.11 MODEL QUESTIONS

Q 1: What is the difference between prim’s algorithm and Kruskal’s algorithm.
Q 2: Employ BFS and DFS to the following graph.



*** ***** ***