BFS

1:Start

2:Enter the number of vertices

Step 3:Read the graph in matrix form

4:Enter the starting vertex

5:Repeat the step4&5 until the queue is empty

6:Take the front item of the queue and add it to the visited list

7:Create a list of that vertex adjacent node.Add the one which aren’t in the Visited list to the back of the queue
8:Stop

DFS

1. Start
2. Create a structure node with element vertex and *next
3.Read number of vertices and assign to v
4.Initialize adj[] with a null
5 .Read edges and insert them in adj[]
6.Represent the edges into adjacency list
7. Acquire only for the new node
8. Insert the node in the new node
9. Go to end of the linked list
10. Keep an array visited[] to keep track of the visited vertices
11. if visited[i]=1 represent that vertex is has been visited before and the DFS function for some already visited node need not be called
12. Repeat the step until all the vertex are visited
13.Print the vertex
14.Stop

PRIMS

1.Begin

2.Create edge list of given graph, with their weights.

3.Draw all nodes to create skeleton for spanning tree.

4.Select an edge with lowest weight and add it to skeleton and delete edge from edge list.

5.Add other edges. While adding an edge take care that the one end of the edge should always be in the skeleton tree and its cost should be minimum.

6.Repeat step 5 until n-1 edges are added.

7.Return.

KRUSKAL’S

1.Start

2.Read the elements of the graph

3.if graph[i][j]!=0 then

4.Create a edge list of given graph with their weight

5.Sort the edge list according to their weights in ascending order

6.Pick up the edge at the top of the edge list

7.Remove this edgr from edge list

8.Connect the edge .If by connecting the vertices a cycle is created then discard the edge

9.Repeat the step 6 to 8 list of edge is visited

10.Print the minimum cost

11.Stop

SETS

1. Start
2. Read the no: of elements in first set
3. Read the elements of first set
4. Read the no: of elements in second set
5. Read the elements of second set
6. Check k=0 display resultant set is null
7. Otherwise display element of resultant set
8. Stop