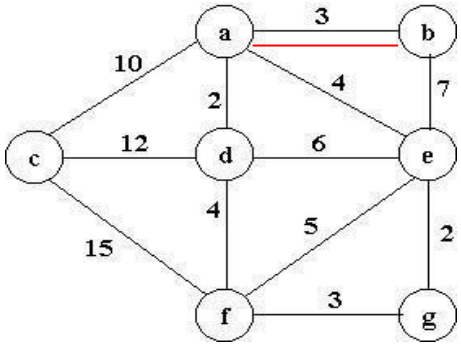
AIM ***Minimum Spanning Tree Algorithm: Prim's and Kruskal***

Assessment

1. Given a graph with weighted edges, obtain a minimal spanning tree using Prim's algorithm



Algorithm:

- **a.** Choose a starting vertex
- **b.** Start a loop till when all the nodes are reached
- **c.** Selec the edge connecting to the next node that has minimum weight which is not visited
- **d.** Add the selected edge and the vertex to the minimum spanning tree t
- **e.** Exit

Code:

```
#include <stdio.h>
#include <stdbool.h>
#include <string.h>

#define INF 9999999

#define V  7

int G[V][V] = {
    {0, 3, 10, 2, 4, 0, 0},
    {3, 0, 0, 0, 7, 0, 0},
    {10, 0, 0, 12, 0, 15, 0},
```

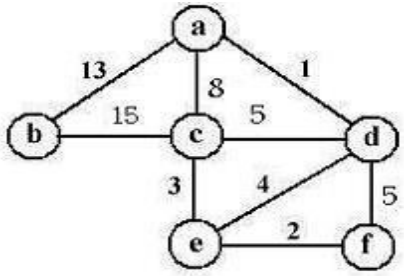
```
    {2, 0, 12, 0, 6, 4, 0},
    {4, 7, 0, 6, 0, 5, 2},
    {0, 0, 15, 4, 5, 0, 3},
    {0, 0, 0, 0, 2, 3, 0}};

int main() {
  int no_edge;
  int selected[V];
  memset(selected, false, sizeof(selected));
  no_edge = 0;
  selected[0] = true;

  int x;
  int y;

  printf("Edge : Weight\n");

  while (no_edge < V - 1) {

    int min = INF;
    x = 0;
    y = 0;

    for (int i = 0; i < V; i++)
    {
      if (selected[i])
      {
        for (int j = 0; j < V; j++)
        {
          if (!selected[j] && G[i][j])
          {
            if (min > G[i][j])
            {
              min = G[i][j];
              x = i;
              y = j;
            }
          }
        }
      }
    }
    printf("%d - %d : %d\n", x, y, G[x][y]);
    selected[y] = true;
    no_edge++;
  }

  return 0;
```

```
}
```

Output:

```
Edge : Weight
0 - 3 : 2
0 - 1 : 3
0 - 4 : 4
4 - 6 : 2
6 - 5 : 3
0 - 2 : 10
```

2. **The State Water department is planning to implement a new drinking water supply system that should connect its 6 districts. In the following graph, the nodes represent the districts and the values associated with the edges represent the cost of laying the water channel between the two districts. The graph shows all the possible connections among the nodes.Use Kruskal's algorithm to find the cost optimal solution for the given graph.**



Code:

```
#include <stdio.h>


#define V 6
int parent[V];
#define inf 99999


int find(int i)
{
  while (parent[i] != i)
    i = parent[i];
  return i;
}
```

```
void union1(int i, int j)
{
  int a = find(i);
  int b = find(j);
  parent[a] = b;
}


void kruskalMST(int cost[][V])
{
  int mincost = 0;


  for (int i = 0; i < V; i++)
    parent[i] = i;


  int edge_count = 0;
  while (edge_count < V - 1)
  {
    int min = inf, a = -1, b = -1;
    for (int i = 0; i < V; i++)
    {
      for (int j = 0; j < V; j++)
      {
        if (find(i) != find(j) && cost[i][j] < min)
        {
          min = cost[i][j];
          a = i;
          b = j;
        }
      }
    }


    union1(a, b);
    printf("Edge %d:(%d, %d) cost:%d \n",
           edge_count++, a, b, min);
    mincost += min;
  }
  printf("\n Minimum cost= %d \n", mincost);
}


int main()
```

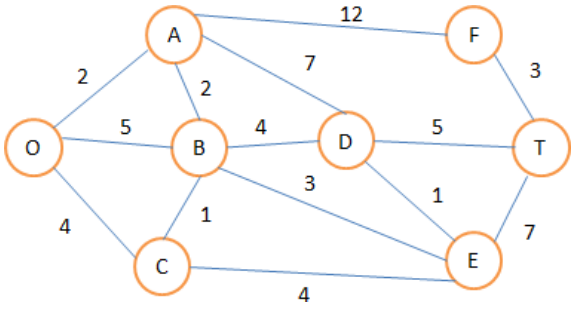
```
{
  /*the nodes in the question can be interpreted as:
  a=0;
  b=1;
  c=2;
  d=3;
  e=4;
  f=5;
  */
  int cost[][V] = {
      {inf, 13, 8, 1, inf, inf},
      {13, inf, 15, inf, inf, inf},
      {8, 15, inf, 5, 3, inf},
      {1, inf, 5, inf, 4, 5},
      {inf, inf, 3, 4, inf, 2},
      {inf, inf, inf, 5, 2, inf}};


  kruskalMST(cost);


  return 0;
}

Output:
Minimum Cost : 23
```

Algorithm:

a. **Sort all the edges from low weight to high**
b. **Take the edge with the lowest weight and add it to the spanning tree.**
c. **If adding the edge created a cycle, then reject the edge.**
d. **Keep adding the edge until all the vertices are visited**

AIM *Single Source Shortest Path Algorithm*

3. Assume that, the management of a school (located in place O) has decided to operate the school buses for the convenience of their staff and students. They wanted to cover all the places that are interconnected as shown in the figure below. Suggest a suitable algorithm to find the shortest route from the school to all other locations for helping the management to plan their bus service. Illustrate the step-by-step procedure for calculation of the shortest path (source O destination F)


A
F
O
B
D
T
C
E
12
7
2
3
2
5
4
5
3
1
4
1
7
4

Algorithm:

1. Create a shortest path tree set that keeps track of vertices included
2. Assign all distances values as infinite
3. While shortest path tree doesn't contain all the vertex
    a. Pick a vertex which is not included and calculate the minimum distance
    b. Include that vertex int the tree
    c. Update the values of the distances as calcultated.

Code:

```
#include <limits.h>
#include <stdio.h>
#include <stdbool.h>


#define V 8


int minDistance(int dist[], bool sptSet[])
{
    int min = INT_MAX, min_index;


    for (int v = 0; v < V; v++)
        if (sptSet[v] == false && dist[v] <= min)
            min = dist[v], min_index = v;


    return min_index;
}


void printSolution(int dist[])
{
    printf("Vertex \t\t Distance from Source\n");
    for (int i = 0; i < V; i++)
        printf("%d \t\t %d\n", i, dist[i]);
}


void dijkstra(int graph[V][V], int src)
{
    int dist[V];
    bool sptSet[V];
    for (int i = 0; i < V; i++)
        dist[i] = INT_MAX, sptSet[i] = false;


    dist[src] = 0;
```

```
    for (int count = 0; count < V - 1; count++)
    {
        int u = minDistance(dist, sptSet);


        sptSet[u] = true;


        for (int v = 0; v < V; v++)


            if (!sptSet[v] && graph[u][v] && dist[u] != INT_MAX && dist[u] +
graph[u][v] < dist[v])
                dist[v] = dist[u] + graph[u][v];
    }


    printSolution(dist);
}


int main()
{
    int graph[V][V] = {{0, 2, 5, 4, 0, 0, 0, 0},
                       {2, 0, 2, 0, 7, 0, 0, 12},
                       {5, 2, 0, 1, 4, 3, 0, 0},
                       {4, 0, 1, 0, 0, 4, 0, 0},
                       {0, 7, 4, 0, 0, 1, 5, 0},
                       {0, 0, 3, 4, 1, 0, 7, 0},
                       {0, 0, 0, 0, 5, 7, 0, 3},
                       {0, 12, 0, 0, 0, 0, 3, 0}};


    dijkstra(graph, 0);
```

```
    return 0;
}
```

Output: