ADA-LAB-7

Q) Find Minimum Cost Spanning Tree of a given undirected graph using Prim/Kruskal's algorithm.

CODE-

Prim's Algorithm-

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

int V;

int minKey(int key[], bool mstSet[]) {
   int min = INT_MAX, min_index;

   for (int v = 0; v < V; v++) {
      if (mstSet[v] == false && key[v] < min) {
         min = key[v];
         min_index = v;
      }
   }

   return min_index;
}

int printMST(int parent[], int graph[V][V]) {
   int sum = 0;
   printf("Edge \tWeight\n");
   for (int i = 1; i < V; i++) {
      printf("%d - %d \t%d \n", parent[i], i, graph[i][parent[i]]);
      sum += graph[i][parent[i]];
   }
   printf("weight=%d\n", sum);
}

void primMST(int graph[V][V]) {
   int parent[V];
   int key[V];
   bool mstSet[V];

   for (int i = 0; i < V; i++) {
      key[i] = INT_MAX;
      mstSet[i] = false;
   }

   key[0] = 0;
   parent[0] = -1;
```

```
    for (int count = 0; count < V - 1; count++) {
        int u = minKey(key, mstSet);
        mstSet[u] = true;

        for (int v = 0; v < V; v++) {
            if (graph[u][v] && mstSet[v] == false && graph[u][v] < key[v]) {
                parent[v] = u;
                key[v] = graph[u][v];
            }
        }
    }
    printMST(parent, graph);
}

int main() {
    printf("Enter the number of vertices: ");
    scanf("%d", &V);

    int graph[V][V];

    printf("Enter the adjacency matrix:\n");
    for (int i = 0; i < V; i++) {
        for (int j = 0; j < V; j++) {
            scanf("%d", &graph[i][j]);
        }
    }

    primMST(graph);

    return 0;
}
```

Krushkal's Algorithm-

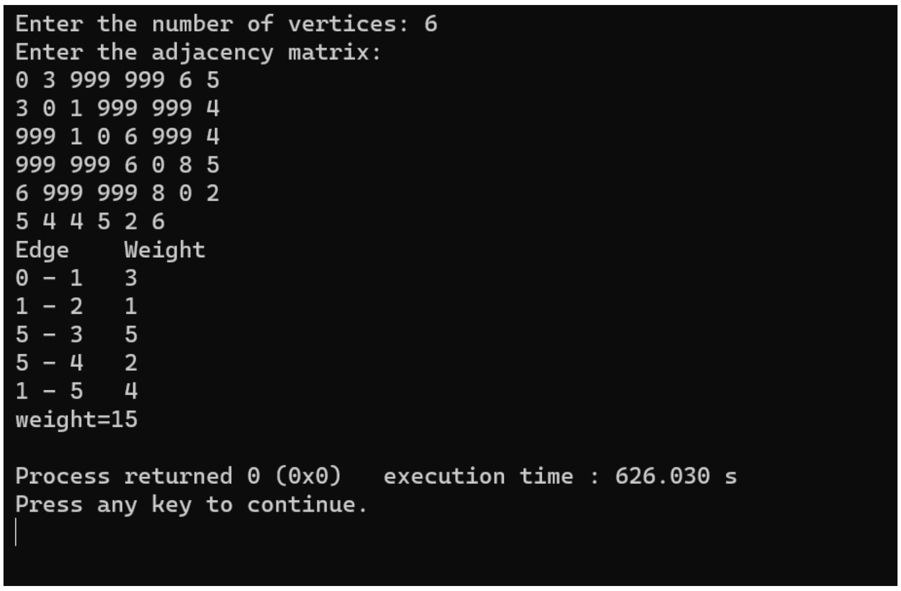
```
#include <stdio.h>

int find(int v, int parent[10])
{
   while (parent[v] != v)
   {
      v = parent[v];
   }
   return v;
}

void union1(int i, int j, int parent[10])
{
   if (i < j)
      parent[j] = i;
   else
```

```
    parent[i] = j;
}

void kruskal(int n, int a[10][10])
{
  int count, k, min, sum, i, j, t[10][10], u, v, parent[10];
  count = 0;
  k = 0;
  sum = 0;
  for (i = 0; i < n; i++)
    parent[i] = i;
  while (count != n - 1)
  {
    min = 999;
    for (i = 0; i < n; i++)
    {
      for (j = 0; j < n; j++)
      {

        if (a[i][j] < min && a[i][j] != 0)
        {
          min = a[i][j];
          u = i;
          v = j;
        }
      }
    }
    i = find(u, parent);
    j = find(v, parent);
    if (i != j)
    {
      union1(i, j, parent);
      t[k][0] = u;
      t[k][1] = v;
      k++;
      count++;
      sum = sum + a[u][v];
    }
    a[u][v] = a[v][u] = 999;
  }
  if (count == n - 1)
  {
    printf("spanning tree\n");
    for (i = 0; i < n - 1; i++)
    {
      printf("%d %d\n", t[i][0], t[i][1]);
    }
    printf("cost of spanning tree=%d\n", sum);
  }
  else
    printf("spanning tree does not exist\n");
}
```

```
int main()
{
  int n, i, j, a[10][10];
  printf("enter the number of nodes\n");
  scanf("%d", &n);
  printf("enter the adjacency matrix\n");
  for (i = 0; i < n; i++)
  {
    for (j = 0; j < n; j++)
      scanf("%d", &a[i][j]);
  }
  kruskal(n, a);
  return 0;
}
```

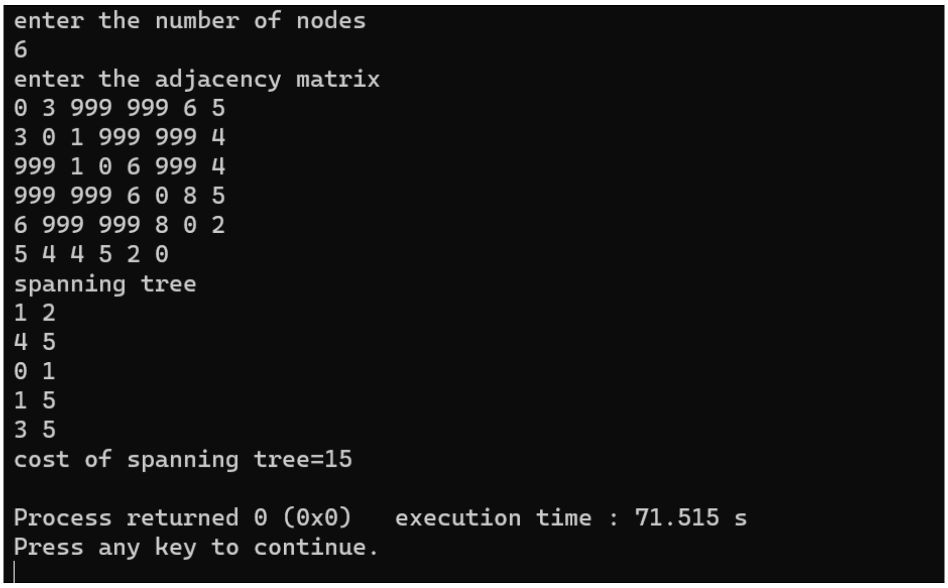
OUTPUT-

Prim’s Alogirthm-



```
Enter the number of vertices: 6
Enter the adjacency matrix:
0 3 999 999 6 5
3 0 1 999 999 4
999 1 0 6 999 4
999 999 6 0 8 5
6 999 999 8 0 2
5 4 4 5 2 6
Edge    Weight
0 - 1   3
1 - 2   1
5 - 3   5
5 - 4   2
1 - 5   4
weight=15

Process returned 0 (0x0)   execution time : 626.030 s
Press any key to continue.
```

Krushkal’s Algorithm-



```
enter the number of nodes
6
enter the adjacency matrix
0 3 999 999 6 5
3 0 1 999 999 4
999 1 0 6 999 4
999 999 6 0 8 5
6 999 999 8 0 2
5 4 4 5 2 0
spanning tree
1 2
4 5
0 1
1 5
3 5
cost of spanning tree=15

Process returned 0 (0x0)   execution time : 71.515 s
Press any key to continue.
```