Name	RUPALI SAWALE
UID no.	2021700056
Experiment No.	6

AIM:	Greedy Method – Prim’s Algorithm.
Program	
PROBLEM STATEMENT:	Use Greedy Programming method to find the minimum weight of a spanning tree.
ALGORITHM/ THEORY:	**Prim's Algorithm** is a greedy algorithm that is used to find the minimum spanning tree from a graph. Prim's algorithm finds the subset of edges that includes every vertex of the graph such that the sum of the weights of the edges can be minimized. Prim's algorithm starts with the single node and explores all the adjacent nodes with all the connecting edges at every step. The edges with the minimal weights causing no cycles in the graph got selected. Prim's algorithm is a greedy algorithm that starts from one vertex and continue to add the edges with the smallest weight until the goal is reached. The steps to implement the prim's algorithm are given as follows - ◦ First, we have to initialize an MST with the randomly chosen vertex. ◦ Now, we have to find all the edges that connect the tree in the above step with the new vertices. From the edges found, select the minimum edge and add it to the tree. ◦ Repeat step 2 until the minimum spanning tree is formed.
PROGRAM:	(see below)

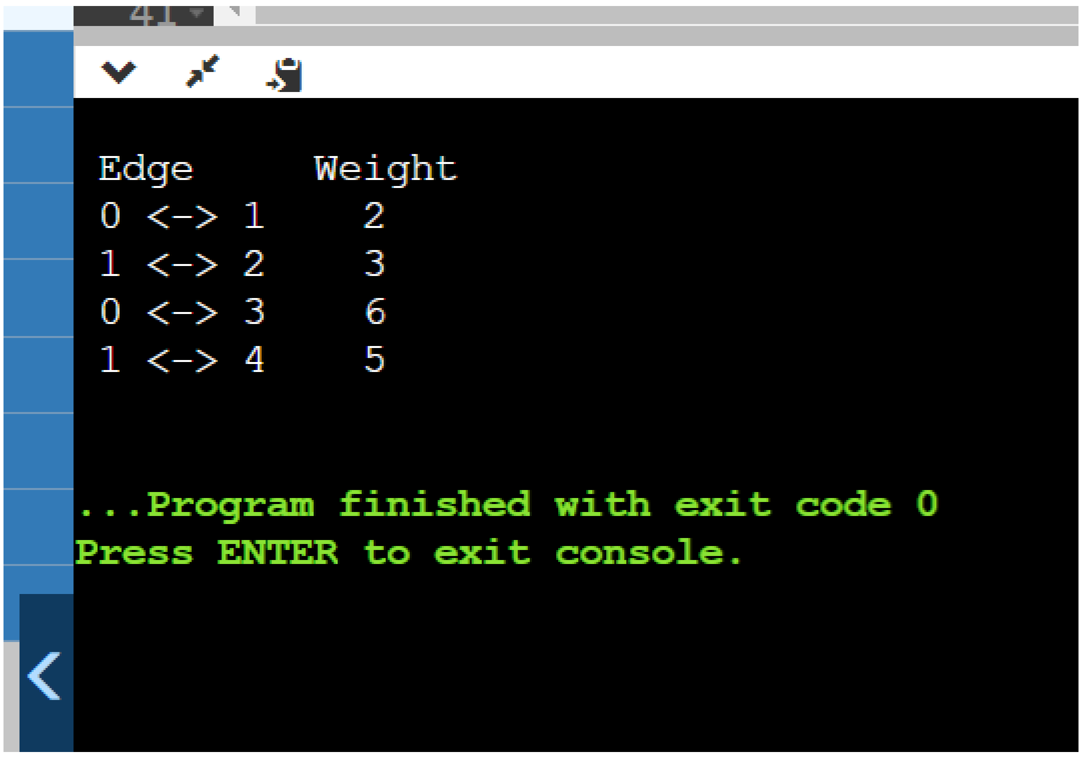
```
#include <stdio.h>
#include <limits.h>
#define vertices 5  /*Define the number of vertices in the graph*/
/* create minimum_key() method for finding the vertex that has minimum key-value and that is not added in MST yet */
int minimum_key(int k[], int mst[])
{
    int minimum  = INT_MAX, min,i;

    /*iterate over all vertices to find the vertex with minimum key-value*/
    for (i = 0; i < vertices; i++)
```

```
        if (mst[i] == 0 && k[i] < minimum )
            minimum = k[i], min = i;
    return min;
}
/* create prim() method for constructing and printing the MST.
The g[vertices][vertices] is an adjacency matrix that defines the graph for MST.*/
void prim(int g[vertices][vertices])
{
    /* create array of size equal to total number of vertices for storing the MST*/
    int parent[vertices];
    /* create k[vertices] array for selecting an edge having minimum weight*/
    int k[vertices];
    int mst[vertices];
    int i, count,edge,v; /*Here 'v' is the vertex*/
    for (i = 0; i < vertices; i++)
    {
        k[i] = INT_MAX;
        mst[i] = 0;
    }
    k[0] = 0; /*It select as first vertex*/
    parent[0] = -1;   /* set first value of parent[] array to -1 to make it root of MST*/
    for (count = 0; count < vertices-1; count++)
    {
        /*select the vertex having minimum key and that is not added in the MST yet from
the set of vertices*/
        edge = minimum_key(k, mst);
        mst[edge] = 1;
        for (v = 0; v < vertices; v++)
        {
            if (g[edge][v] && mst[v] == 0 && g[edge][v] <  k[v])
            {
                parent[v]  = edge, k[v] = g[edge][v];
            }
        }
     }
     /*Print the constructed Minimum spanning tree*/
     printf("\n Edge \t  Weight\n");
     for (i = 1; i < vertices; i++)
     printf(" %d <-> %d    %d \n", parent[i], i, g[i][parent[i]]);

}
int main()
{
    int g[vertices][vertices] = {{0, 2, 0, 6, 0},
                                {2, 0, 3, 8, 5},
                                {0, 3, 0, 0, 7},
                                {6, 8, 0, 0, 9},
                          {0, 5, 7, 9, 0},
                          };
    prim(g);
    return 0;
}
```

RESULT:



```
Edge      Weight
0 <-> 1     2
1 <-> 2     3
0 <-> 3     6
1 <-> 4     5


...Program finished with exit code 0
Press ENTER to exit console.
```

CONCLUSION:	Learned about greedy approach and it’s implementation by using prim’s algorithm .