

DAA LAB ASSIGNMENT-4

1.Prims Algorithm:

CODE:

```
#include <stdio.h>
#include <limits.h>
#define V 5
int min_key(int key[], int mst_set[]) {
    int min = INT_MAX, min_index;
    for (int v = 0; v < V; v++) {
        if (mst_set[v] == 0 && key[v] < min) {
            min = key[v];
            min_index = v;
        }
    }
    return min_index;
}
void prim(int graph[V][V]) {
    int parent[V];
    int key[V];
    int mst_set[V];
    for (int i = 0; i < V; i++) {
        key[i] = INT_MAX;
```

```

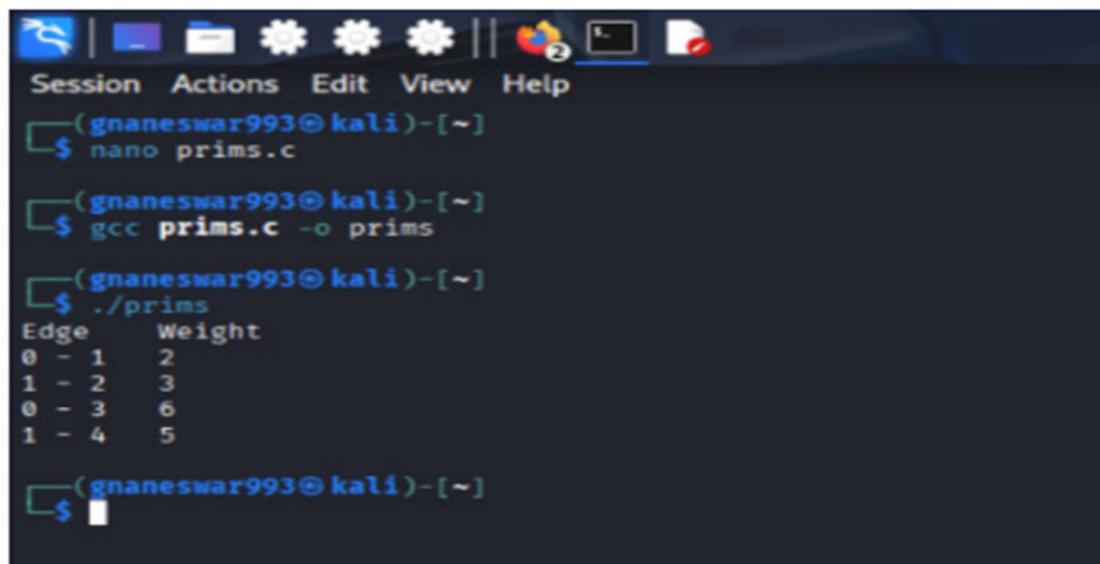
mst_set[i] = o;
}

key[o] = o;
parent[o] = -1;
for (int count = o; count < V - 1; count++) {
    int u = min_key(key, mst_set);
    mst_set[u] = 1;
    for (int v = o; v < V; v++) {
        if (graph[u][v] && mst_set[v] == o && graph[u][v] <
key[v]) {
            key[v] = graph[u][v];
            parent[v] = u;
        }
    }
    printf("Edge %d-%d %d\n", parent[i], i, graph[i][parent[i]]);
}
int main() {
    int graph[V][V] = {
        {0, 2, 0, 6, 0},
        {2, 0, 3, 8, 5},

```

```
{o, 3, o, o, 7},  
{6, 8, o, o, 9},  
{o, 5, 7, 9, o}  
};  
prim(graph);  
return o;  
}
```

OUTPUT:



The screenshot shows a terminal window with a dark blue background and white text. At the top, there is a menu bar with icons for Session, Actions, Edit, View, and Help. Below the menu, the terminal prompt shows the user is at the root directory (~) on a Kali Linux system. The user has run a series of commands to compile a C program named 'prims.c' and then execute it. The output of the program shows the edges and their weights from the resulting Minimum Spanning Tree.

```
(gnaneswar993㉿kali)-[~]  
$ nano prims.c  
(gnaneswar993㉿kali)-[~]  
$ gcc prims.c -o prims  
(gnaneswar993㉿kali)-[~]  
$ ./prims  
Edge      Weight  
0 - 1      2  
1 - 2      3  
0 - 3      6  
1 - 4      5  
(gnaneswar993㉿kali)-[~]  
$ █
```

2.KRUSKAL'S ALGORITHM:

PROGRAM:

```
#include <stdio.h>
#include <stdlib.h>

#define V 5
#define E 7
typedef struct {
    int u, v, weight;
} Edge;

int find(int parent[], int i) {
    if (parent[i] == -1)
        return i;
    return find(parent, parent[i]);
}

void union_set(int parent[], int x, int y) {
    int xroot = find(parent, x);
    int yroot = find(parent, y);
    parent[xroot] = yroot;
}

int compare(const void *a, const void *b) {
    return ((Edge*)a)->weight - ((Edge*)b)->weight;
}
```

```

void kruskal(Edge edges[], int parent[]) {
    qsort(edges, E, sizeof(Edge), compare);
    int mst_weight = 0;
    printf("Edge |tWeight|\n");
    for (int i = 0; i < E; i++) {
        int u = edges[i].u;
        int v = edges[i].v;
        int w = edges[i].weight;
        int x = find(parent, u);
        int y = find(parent, v);
        if (x != y) {
            printf("%d - %d |t%d|\n", u, v, w);
            mst_weight += w;
            union_set(parent, x, y);
        }
    }
    printf("Total weight of MST: %d\n", mst_weight);
}

int main() {
    Edge edges[] = {
        {0, 1, 2},
        {0, 3, 6},
        {1, 2, 3},
        {1, 4, 5},

```

```

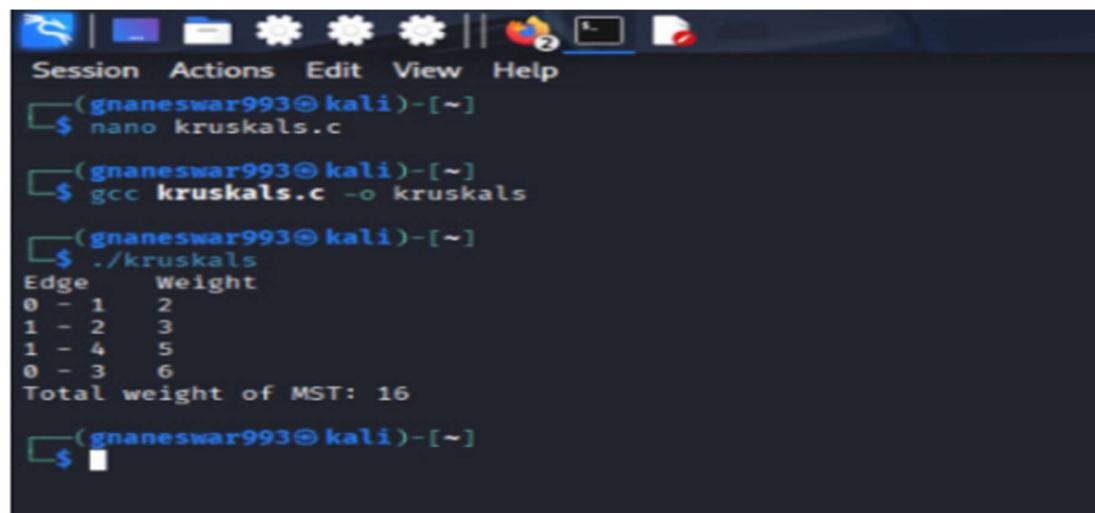
{2, 4, 7},
{3, 4, 9},
{2, 3, 8}

};

int parent[V];
for (int i = 0; i < V; i++) {
    parent[i] = -1;
}
kruskal(edges, parent);
return o;
}

```

OUTPUT:



```

Session Actions Edit View Help
(gnaneswar993㉿kali)-[~]
$ nano kruskals.c
(gnaneswar993㉿kali)-[~]
$ gcc kruskals.c -o kruskals
(gnaneswar993㉿kali)-[~]
$ ./kruskals
Edge      Weight
0 - 1      2
1 - 2      3
1 - 4      5
0 - 3      6
Total weight of MST: 16
(gnaneswar993㉿kali)-[~]
$ █

```

K.NAGA GNANESWARA REDDY

CH.SC.U4CSE24219