

```
//IMPLEMENTATIO OF PRISM'S ALGORITHM
```

```
#include<iostream>
```

```
using namespace std;
```

```
const int V=4;
```

```
int min_Key(int key[], bool visited[])
```

```
{
    int min = 999, min_index;
    for (int v = 0; v < V; v++)
    {
        if (visited[v] == false && key[v] < min)
        {
            min = key[v];
min_index = v;
        }
    }
    return min_index;
}
```

```
int print_MST(int parent[], int cost[V][V])
```

```
{
    int minCost=0; cout<<"Edge
\tWeight\n"; for (int i = 1; i < V; i++)
    {
        cout<<parent[i]<<" - "<<i<<" \t"<<cost[i][parent[i]]<<" \n";
        minCost=minCost+cost[i][parent[i]];
    }
    cout<<"Total cost is"<<minCost;
}
```

```
void find_MST(int cost[V][V])
```

```
{
    int parent[V], key[V];
    bool visited[V];

    for (int i = 0; i < V; i++)
    {
        key[i] = 999; visited[i] = false;
        parent[i]=-1;
    }
    key[0] = 0;
parent[0] = -1;
    for (int x = 0; x < V - 1; x++)
    {
        int u = min_Key(key, visited);
visited[u] = true;
        for (int v = 0; v < V; v++)
        {

```

```

        if (cost[u][v]!=0 && visited[v] == false && cost[u][v] < key[v])
        {
            parent[v] = u;
            key[v] = cost[u][v];
        }
    }
    print_MST(parent, cost);
}

```

```

int main()
{
    int cost[V][V];
    cout<<"Enter the vertices for a graph with 4 vetices:-";
    for (int i=0;i<V;i++)
    {
        for(int j=0;j<V;j++)
        {
            cin>>cost[i][j];
        }
    }
    find_MST(cost);
    return 0;
}

```

Output:-

```

avcoe@avcoe-HP-Pro-3330-MT:~$ cd
avcoe@avcoe-HP-Pro-3330-MT:~/Avcoe $ g++ prisma.cpp
avcoe@avcoe-HP-Pro-3330-MT:~/Avcoe$ ./a.out

```

Enter the vertices for a graph with 4 vetices:-0

4

2

7

4

0

0

6

2

0

0

9

7

6

9

0

Edge Weight

0 - 1 4

0 - 2 2

1 - 3 6

Total cost is12

