PRIM'S ALGORITHM
MINI PROJECT

Rohan Kulkarni
A023 45207210002

❖ What is Prim's Algorithm?

➔ In Computer Science Prims Algorithm is a greedy Algorithm that finds a minimum spanning tree for a weighted undirected graph.
➔ It means that it find the subset of the edges that forms a tree that Includes every vertex, where the total weight of all the edges in the tree is minimized.

❖ Applications of Prims Algorithm

➔ In real-daily-life it can be used to design a network linking small outlying villages.
➔ A network of pipes for drinking water or natural gas ,an electrical grid, irrigation channels, a fibre-optic, placing microwave towers etc.
➔ Designing Local Area Networks (LAN's)
➔ Making electrical wire connections on a road panel
➔ Constructing highway roads or rail roads spanning over several cities.

❖ Prim's Algorithm Pseudocode –

➔ T = Φ;
➔ U = {1};
➔ While (U≠V)

Let (u , v) be the lowest cost edge such that u ∈ U and v ∈ V - U;

$T = T \cup \{(u , v)\}$

$$U = U \cup \{v\}$$

- **Example of a Code :**

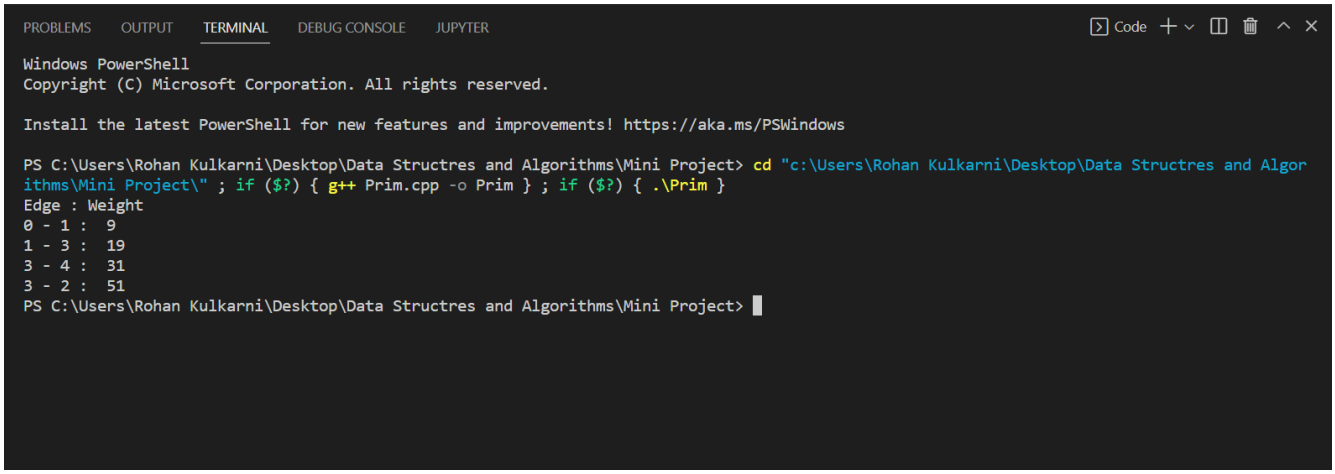
```
#include <cstring>
#include <iostream>
using namespace std;

#define INF 9999999

// number of vertices in grapj
#define V 5

// create a 2d array of size 5x5
//for adjacency matrix to represent graph

int G[V][V] = {
  {0, 9, 75, 0, 0},
  {9, 0, 95, 19, 42},
  {75, 95, 0, 51, 66},
  {0, 19, 51, 0, 31},
  {0, 42, 66, 31, 0}};

int main() {
  int no_edge;  // number of edge
```

```
// create a array to track selected vertex
// selected will become true otherwise false
int selected[V];

// set selected false initially
memset(selected, false, sizeof(selected));

// set number of edge to 0
no_edge = 0;

// the number of egde in minimum spanning tree will be
// always less than (V -1), where V is number of vertices in
//graph

// choose 0th vertex and make it true
selected[0] = true;

int x;  //  row number
int y;  //  col number

// print for edge and weight
cout << "Edge"
  << " : "
  << "Weight";
```

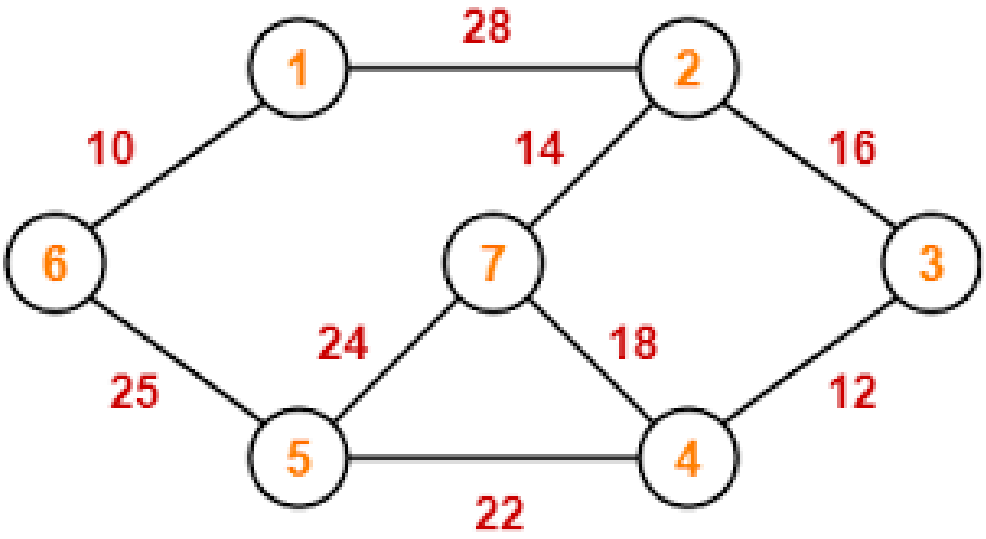
```
cout << endl;
while (no_edge < V - 1) {
 //For every vertex in the set S, find the all adjacent vertices
 // , calculate the distance from the vertex selected at step 1.
 // if the vertex is already in the set S, discard it otherwise
 //choose another vertex nearest to selected vertex  at step 1.

 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]) {  // not in selected and there is an edge
     if (min > G[i][j]) {
      min = G[i][j];
      x = i;
      y = j;
     }
    }
   }
  }
 }
 cout << x << " - " << y << " :  " << G[x][y];
```

```
    cout << endl;
    selected[y] = true;
    no_edge++;
  }

  return 0;
}
```

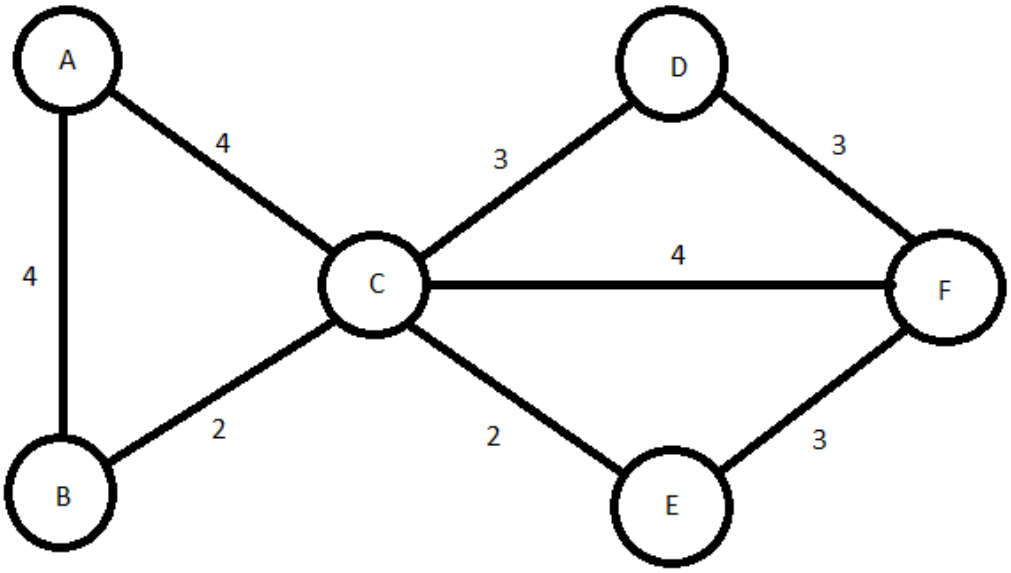
OUTPUT:



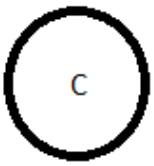
❖ **Diagrammatic Representation of Prim's Algorithm Along with solving an Example –**



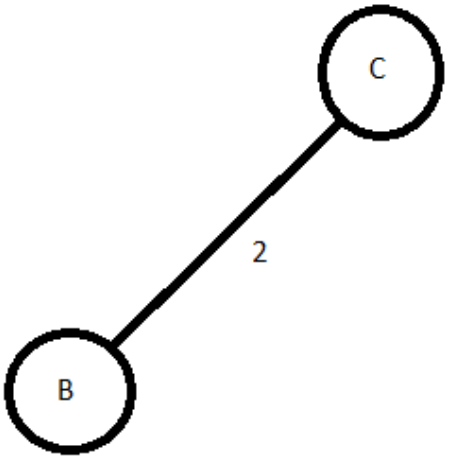
Example :



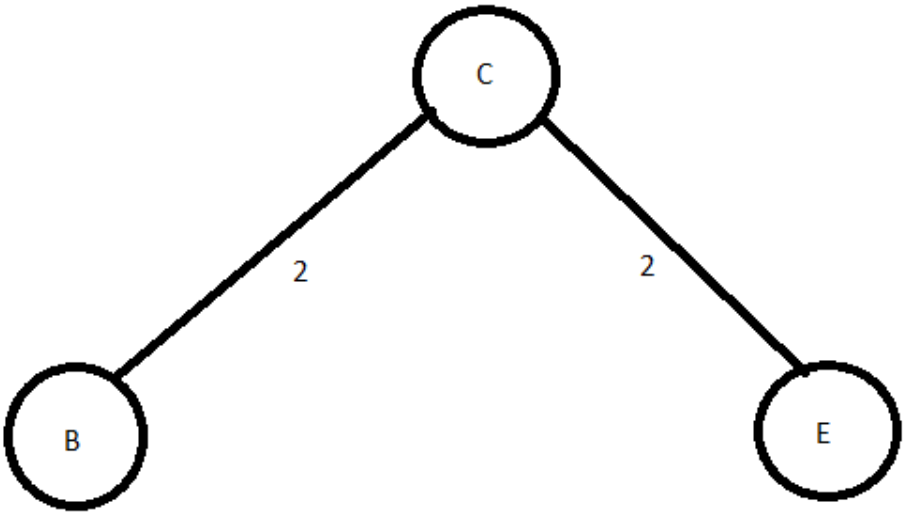
Step 1



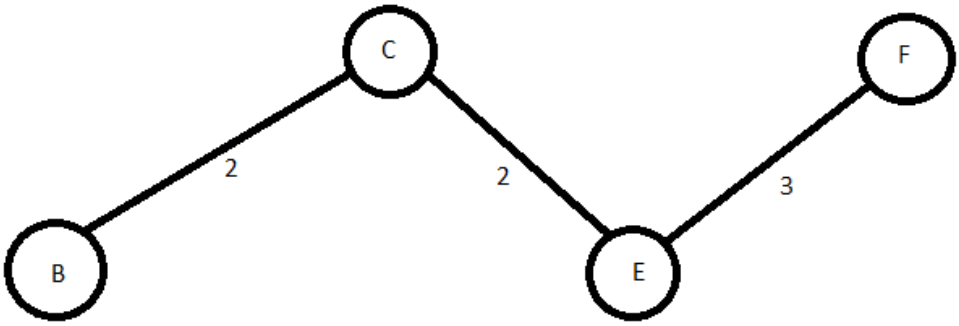
Step 2



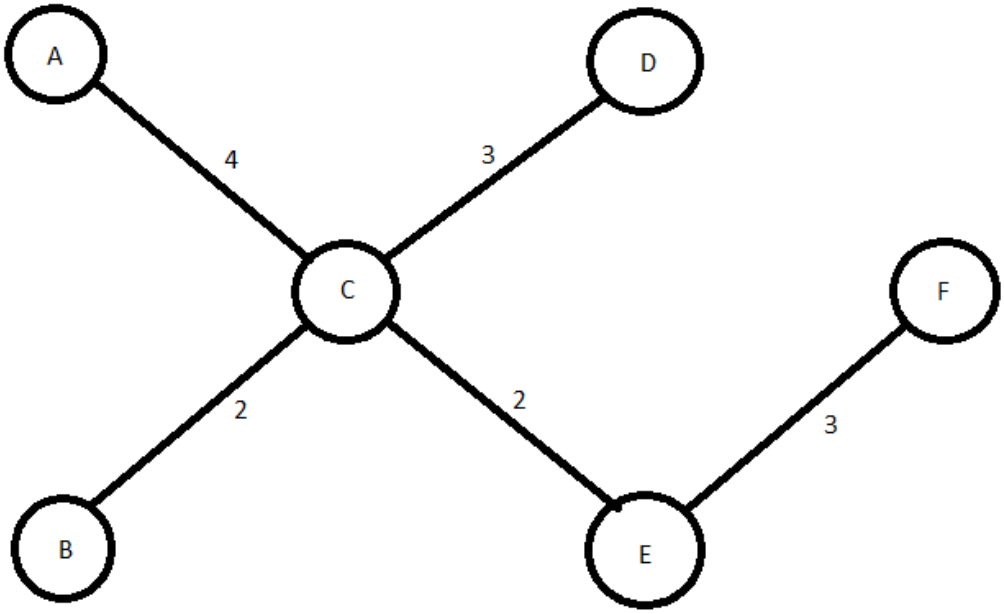
Step 3



Step 4



Step 5


A
D
4
3
C
F
2
2
3
B
E