Name:	Omkar Deshmukh
UID:	2021700018
Batch:	CSE(DS)D1
Exp:	05
Aim:	Prims Algorithm

Algorithms:

The working of Prim's algorithm can be described by using the following steps:

Step 1: Determine an arbitrary vertex as the starting vertex of the MST.

Step 2: Follow steps 3 to 5 till there are vertices that are not included in the MST (known as fringe vertex).

Step 3: Find edges connecting any tree vertex with the fringe vertices.

Step 4: Find the minimum among these edges.

Step 5: Add the chosen edge to the MST if it does not form any cycle.

Step 6: Return the MST and exit.

Theory: Prim's algorithm is also a Greedy algorithm. This algorithm always starts with a single node and moves through several adjacent nodes, in order to explore all of the

connected edges along the way. The algorithm starts with an empty spanning tree.

The idea is to maintain two sets of vertices. The first set contains the vertices already included in the MST, and the other set contains the vertices not yet included. At every step, it considers all the edges that connect the two sets and picks the minimum weight edge from these edges. After picking the edge, it moves the other endpoint of the edge to the set containing MST.

Code:-

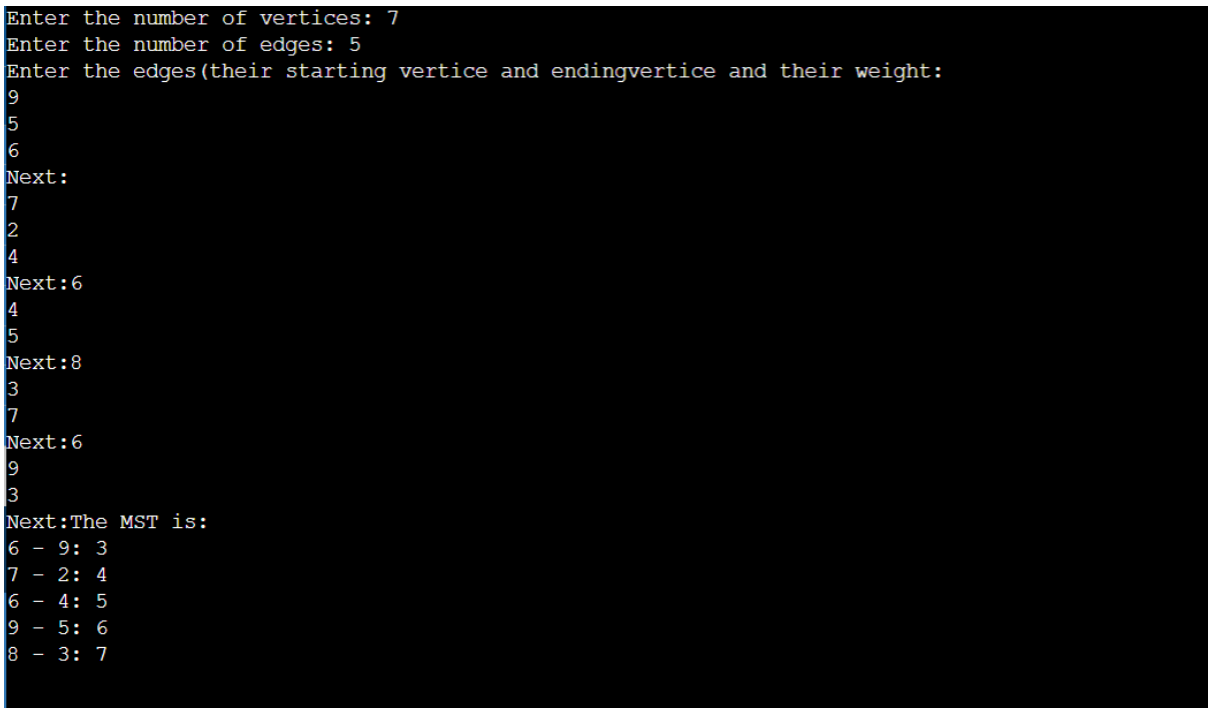
```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <limits.h>
#define MAX_VERTICES 100
#define INF INT_MAX
typedef struct {
 int u, v, weight;
} Edge;
int parent[MAX_VERTICES];
Edge edges[MAX_VERTICES];
int num_edges = 0;
int find(int v) {
 if (parent[v] != v) {
 parent[v] = find(parent[v]);
```

```
 }
 return parent[v];
}
void union_sets(int u, int v) {
 parent[find(u)] = find(v);
}
// Comparator function for sorting edges by weight
int compare_edges(const void* a, const void* b) {
 Edge* e1 = (Edge*)a;
 Edge* e2 = (Edge*)b;
 return e1->weight - e2->weight;
}
// Find the MST of a graph with n vertices and m edges
void mst(int n, int m, Edge* edges) {
 for (int i = 0; i < n; i++) {
 parent[i] = i;
 }
 // Sort the edges by weight
 qsort(edges, m, sizeof(Edge), compare_edges);
 for (int i = 0; i < m && num_edges < n - 1; i++) {
 int u = edges[i].u;
 int v = edges[i].v;
```

```
 if (find(u) != find(v)) {
 union_sets(u, v);
 edges[num_edges++] = edges[i];
 }
 }
}
int main() {
 int n, m;
 printf("Enter the number of vertices: ");
 scanf("%d", &n);
 printf("Enter the number of edges: ");
 scanf("%d", &m);
 printf("Enter the edges(their starting vertice and
endingvertice and their weight:\n");
 for (int i = 0; i < m; i++)
 {
 scanf("%d%d%d", &edges[i].u, &edges[i].v,
&edges[i].weight);
 printf("Next:");
 }
 mst(n, m, edges);
 printf("The MST is:\n");
```

```
 for (int i = 0; i < num_edges; i++) {
 printf("%d - %d: %d\n", edges[i].u, edges[i].v,
edges[i].weight);
 }
 return 0;
}
```

Output:-



Conclusion:-From this experiment, I have understood how prims algoritm is used to find minimal spanning tree.