# PRACTICAL - 7

**AIM** : To perform prim’s algorithm

**PSEUDO CODE :**

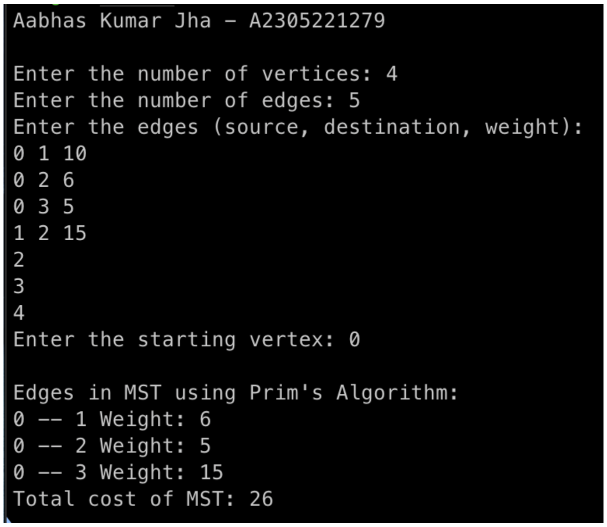
```
PrimsMST(Graph graph)
Step 1: Initialize an empty set to keep track of selected vertices (MST).
Step 2: Initialize a priority queue (min heap) to store edges with their weights.
Step 3: Choose an arbitrary starting vertex as the initial vertex.
Step 4: Add all edges incident to the initial vertex into the priority queue.
Step 5: Mark the initial vertex as selected.
Step 6: while the MST set doesn't contain all vertices, do steps 7-9
Step 7:         Get the edge with the minimum weight from the priority queue.
Step 8:         If adding the edge to the MST doesn't create a cycle, then do step 9
Step 9:         Add the selected edge to the MST set.
Step 10:        Add all edges incident to the newly added vertex into the priority queue.
Step 11: Return the MST set.
```

**CODE** :

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_EDGES 50
#define MAX_VERTICES 30
#define INF 1000000  // Define a large value for infinity
struct Edge {
  int source, destination, weight;
};
struct Graph {
  int numVertices, numEdges;
  struct Edge edges[MAX_EDGES];
};
int minKey(int key[], int mstSet[], int numVertices) {
  int min = INF, min_index;
  for (int v = 0; v < numVertices; v++)
```

```
        if (!mstSet[v] && key[v] < min)
            min = key[v], min_index = v;
    return min_index;
}

void prims(struct Graph graph) {
    int parent[MAX_VERTICES];
    int key[MAX_VERTICES];
    int mstSet[MAX_VERTICES];

    for (int i = 0; i < graph.numVertices; i++) {
        key[i] = INF;
        mstSet[i] = 0;
    }

    printf("Enter the starting vertex: ");
    int startVertex;
    scanf("%d", &startVertex);

    key[startVertex] = 0;
    parent[startVertex] = -1;

    for (int count = 0; count < graph.numVertices - 1; count++) {
        int u = minKey(key, mstSet, graph.numVertices);
        mstSet[u] = 1;

        for (int v = 0; v < graph.numVertices; v++) {
            if (graph.edges[v].weight && !mstSet[v] && graph.edges[v].weight < key[v]) {
                parent[v] = u;
                key[v] = graph.edges[v].weight;
            }
        }
    }

    printf("\nEdges in MST using Prim's Algorithm:\n");
    int totalCost = 0;
    for (int i = 1; i < graph.numVertices; i++) {
        printf("%d -- %d Weight: %d\n", parent[i], i, graph.edges[i].weight);
        totalCost += graph.edges[i].weight;
    }
    printf("Total cost of MST: %d\n", totalCost);
}

int main() {
    printf("Aabhas Kumar Jha - A2305221279\n\n");
    int numVertices, numEdges;
    printf("Enter the number of vertices: ");
    scanf("%d", &numVertices);

    struct Graph graph;
    graph.numVertices = numVertices;
    graph.numEdges = 0;

    printf("Enter the number of edges: ");
    scanf("%d", &numEdges);

    printf("Enter the edges (source, destination, weight):\n");
    for (int i = 0; i < numEdges; i++) {
        int source, destination, weight;
        scanf("%d %d %d", &source, &destination, &weight);
        graph.edges[graph.numEdges].source = source;
        graph.edges[graph.numEdges].destination = destination;
        graph.edges[graph.numEdges].weight = weight;
        graph.numEdges++;
    }

    prims(graph);

    return 0;
}
```

**OUTPUT** :



```
Aabhas Kumar Jha - A2305221279

Enter the number of vertices: 4
Enter the number of edges: 5
Enter the edges (source, destination, weight):
0 1 10
0 2 6
0 3 5
1 2 15
2
3
4
Enter the starting vertex: 0

Edges in MST using Prim's Algorithm:
0 -- 1 Weight: 6
0 -- 2 Weight: 5
0 -- 3 Weight: 15
Total cost of MST: 26
```

**COMPLEXITY :** O(V^2)

# PRACTICAL - 8

**AIM** : To perform kruskal’s algorithm

**PSEUDO CODE :**

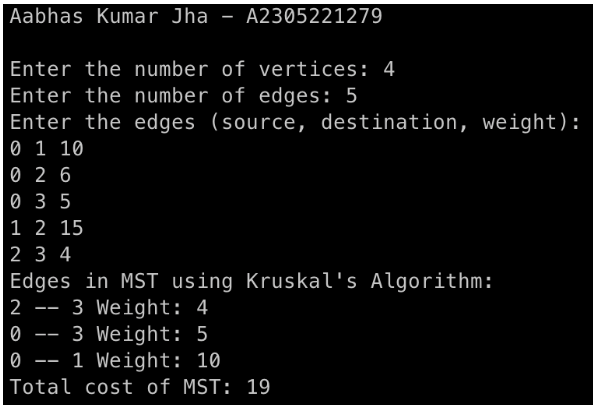
```
KruskalMST(Graph graph)
Step 1: Sort all the edges of the graph in ascending order of their weights.
Step 2: Initialize an empty set to keep track of selected edges (MST).
Step 3: Initialize a Union-Find data structure (Disjoint Set) to manage connected components.
Step 4: for each edge (u, v) in the sorted order, do steps 5-7
Step 5:         if adding edge (u, v) doesn't create a cycle, then do step 6
Step 6:         Add edge (u, v) to the MST set.
Step 7:         Merge the sets containing u and v in the Union-Find structure.
Step 8: Return the MST set.
```

**CODE** :

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

#define MAX_EDGES 50
#define MAX_VERTICES 30

struct Edge {
  int source, destination, weight;
};

struct Graph {
  int numVertices, numEdges;
  struct Edge edges[MAX_EDGES];
};

int parent[MAX_VERTICES];
```

```
struct Graph createGraph(int numVertices) {
  struct Graph graph;
  graph.numVertices = numVertices;
  graph.numEdges = 0;
  return graph;
}

void addEdge(struct Graph *graph, int source, int destination, int weight) {
  graph->edges[graph->numEdges].source = source;
  graph->edges[graph->numEdges].destination = destination;
  graph->edges[graph->numEdges].weight = weight;
  graph->numEdges++;
}

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

void unionSets(int x, int y) {
  int xset = find(x);
  int yset = find(y);
  parent[xset] = yset;
}

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

void kruskals(struct Graph graph) {
  struct Edge result[MAX_EDGES];
  int numEdgesInMST = 0;
  int totalCost = 0;

  for (int i = 0; i < graph.numVertices; i++)
      parent[i] = -1;

  qsort(graph.edges, graph.numEdges, sizeof(struct Edge), compareEdges);

  for (int i = 0; i < graph.numEdges; i++) {
      int source = graph.edges[i].source;
      int destination = graph.edges[i].destination;

      int sourceParent = find(source);
      int destinationParent = find(destination);

      if (sourceParent != destinationParent) {
          result[numEdgesInMST++] = graph.edges[i];
          unionSets(sourceParent, destinationParent);
          totalCost += graph.edges[i].weight;
      }
  }

  printf("Edges in MST using Kruskal's Algorithm:\n");
  for (int i = 0; i < numEdgesInMST; i++)
      printf("%d -- %d Weight: %d\n", result[i].source, result[i].destination, result[i].weight);
  printf("Total cost of MST: %d\n", totalCost);
}

int main() {
  printf("Aabhas Kumar Jha - A2305221279\n\n");
  int numVertices, numEdges;
  printf("Enter the number of vertices: ");
  scanf("%d", &numVertices);

  struct Graph graph = createGraph(numVertices);

  printf("Enter the number of edges: ");
  scanf("%d", &numEdges);

  printf("Enter the edges (source, destination, weight):\n");
  for (int i = 0; i < numEdges; i++) {
      int source, destination, weight;
      scanf("%d %d %d", &source, &destination, &weight);
      addEdge(&graph, source, destination, weight);
  }
  kruskals(graph);
  return 0;
}
```

**OUTPUT** :



```
Aabhas Kumar Jha - A2305221279

Enter the number of vertices: 4
Enter the number of edges: 5
Enter the edges (source, destination, weight):
0 1 10
0 2 6
0 3 5
1 2 15
2 3 4
Edges in MST using Kruskal's Algorithm:
2 -- 3 Weight: 4
0 -- 3 Weight: 5
0 -- 1 Weight: 10
Total cost of MST: 19
```