EXPERIMENT 2

Objective: Implementation of Prim’s algorithm to find Minimum Spanning Tree (MST).
.

Brief Theory:

```
MST-PRIM(G, w, r)
for each vertex u ∈ G.V
    u.key = ∞
    u.π = NIL
r.key = 0
Q = ∅
for each vertex u ∈ G.V
    INSERT(Q, u)
while Q ≠ ∅
    u = EXTRACT-MIN(Q)          // add u to the tree
    for each vertex v in G.Adj[u]   // update keys of u's non-tree neighbors
        if v ∈ Q and w(u, v) < v.key
            v.π = u
            v.key = w(u, v)
            DECREASE-KEY(Q, v, w(u, v))
```

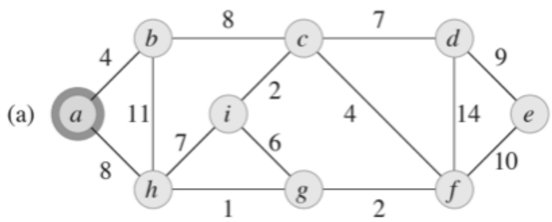
Prim's Algorithm is a greedy method to find the MST of a connected, weighted graph. It starts from any node and grows the MST by repeatedly adding the smallest edge connecting a vertex in the MST to one outside it.

<u>Steps:</u>

1. Initialize a starting node and use a data structure to store edges by weight.
2. Select the smallest edge connecting MST to an unvisited vertex.
3. Add the edge and vertex to the MST, updating the queue with new edges.
4. Repeat until all vertices are included in the MST.

Tasks:

1) Implement Prim's algorithm to find the Minimum Spanning Tree (MST) for the given connected, weighted graph using an adjacency matrix.



2) Modify the implementation to use a priority queue for selecting the smallest edge more efficiently.
3) Create a program where the user can input graph nodes, edges, and weights, then compute the MST using Prim's algorithm.

Apparatus and components required: Computer with C or C++ Compiler and Linux/Windows platform.

Experimental/numerical procedure: Coding, compilation, editing, run and debugging.