Experiment No – 03

Aim: Implement Greedy search algorithm for Prim's Minimal Spanning Tree Algorithm.

Source Code:

```
import sys
class Graph():
    def __init__(self, vertices):
        self.V = vertices
        self.graph = [[0 for column in range(vertices)]
                      for row in range(vertices)]
    def printMST(self, parent):
        print("Edge \tWeight")
        for i in range(1, self.V):
            print(parent[i], "-", i, "\t", self.graph[i][parent[i]])
    def minKey(self, key, mstSet):
        # Initialize min value
        min = sys.maxsize
        for v in range(self.V):
            if key[v] < min and mstSet[v] == False:
                min = key[v]
                min_index = v
        return min_index
    def primMST(self):
        # Key values used to pick minimum weight edge in cut
        key = [sys.maxsize] * self.V
        parent = [None] * self.V
        key[0] = 0
        mstSet = [False] * self.V
        parent[0] = -1  # First node is always the root of
        for cout in range(self.V):
            u = self.minKey(key, mstSet)
```

```
            # Put the minimum distance vertex in
            # the shortest path tree
            mstSet[u] = True
            for v in range(self.V):
                if self.graph[u][v] > 0 and mstSet[v] == False \
                and key[v] > self.graph[u][v]:
                    key[v] = self.graph[u][v]
                    parent[v] = u
        self.printMST(parent)
# Driver's code
if __name__ == '__main__':
    g = Graph(5)
    g.graph = [[0, 2, 0, 6, 0],
               [2, 0, 3, 8, 5],
               [0, 3, 0, 0, 7],
               [6, 8, 0, 0, 9],
               [0, 5, 7, 9, 0]]
    g.primMST()
```

Output:

```
Edge      Weight
0 - 1     2
1 - 2     3
0 - 3     6
1 - 4     5
```