95.Prims Algorithm

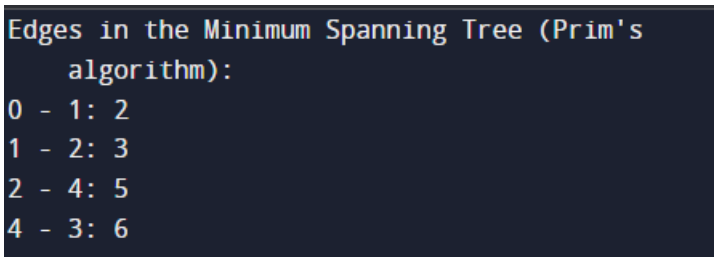
AIM: To find the minimum (or) shortest path by using the Prim’s algorithm

PROGRAM:

```
import heapq

def prim_mst(graph):
    n = len(graph)
    mst = []
    visited = [False] * n
    min_heap = []

    heapq.heappush(min_heap, (0, 0))

    while min_heap:
        weight, u = heapq.heappop(min_heap)

        if visited[u]:
            continue

        visited[u] = True

        if u != 0:
            mst.append((parent[u], u, weight))
                for v, weight in enumerate(graph[u]):
            if weight != float('inf') and not visited[v]:
                heapq.heappush(min_heap, (weight, v))
                parent[v] = u  # To reconstruct MST later

    return mst
graph = [
    [0, 2, float('inf'), 6, float('inf')],
```

```
    [2, 0, 3, 8, 5],
    [float('inf'), 3, 0, float('inf'), 7],
    [6, 8, float('inf'), 0, 9],
    [float('inf'), 5, 7, 9, 0]
]

parent = [-1] * len(graph)
mst = prim_mst(graph)

print("Edges in the Minimum Spanning Tree (Prim's algorithm):")
for u, v, weight in mst:
    print(f"{u} - {v}: {weight}")
```

OUTPUT:



TIME COMPLEXITY: O (E log V)