In [16]:

```python
import sys

class Graph:
    def __init__(self, vertices):
        self.V = vertices
        self.graph = [[0 for _ in range(vertices)] for _ in range(vertices

    def add_edge(self, u, v, weight):
        self.graph[u][v] = weight
        self.graph[v][u] = weight

    def min_key(self, key, mst_set):
        min_val = sys.maxsize
        min_index = -1

        for v in range(self.V):
            if key[v] < min_val and not mst_set[v]:
                min_val = key[v]
                min_index = v

        return min_index

    def prim_mst(self):
        key = [sys.maxsize] * self.V
        parent = [-1] * self.V
        mst_set = [False] * self.V

        key[0] = 0  # Start with the first vertex

        for _ in range(self.V):
            u = self.min_key(key, mst_set)
            mst_set[u] = True

            # Update key values and parent pointers of adjacent vertices
            for v in range(self.V):
                if self.graph[u][v] > 0 and not mst_set[v] and key[v] > se
                    key[v] = self.graph[u][v]
                    parent[v] = u

        return [(parent[i], i) for i in range(1, self.V)]

# Example usage:
g = Graph(5)
g.add_edge(0, 1, 2)
g.add_edge(0, 3, 6)
g.add_edge(1, 2, 3)
g.add_edge(1, 3, 8)
g.add_edge(1, 4, 5)
g.add_edge(2, 4, 7)
g.add_edge(3, 4, 9)

mst = g.prim_mst()
print("Edges in the Minimum Spanning Tree:")
for u, v in mst:
    print(f"{u} - {v}")


```

```
58
59
60
61
62
```

```
Edges in the Minimum Spanning Tree:
0 - 1
1 - 2
0 - 3
1 - 4
```

In [ ]:

```
1
```