Problem 5 Prim's Algorithm – Minimum Spanning Tree

```
import heapq
class Graph:
    def __init__(self):
        self.graph = {}

    def add_edge(self, u, v, weight):
        if u not in self.graph:
            self.graph[u] = []
        if v not in self.graph:
            self.graph[v] = []
        self.graph[u].append((v, weight))
        self.graph[v].append((u, weight))

    def prim_mst(self):
        start_vertex = next(iter(self.graph))
        mst = []
        visited = set()
        priority_queue = [(0, start_vertex)]

        while priority_queue:
            weight, current_vertex = heapq.heappop(priority_queue)
            if current_vertex in visited:
                continue

            visited.add(current_vertex)
            if weight != 0:
                mst.append((current_vertex, weight))

            for neighbor, edge_weight in self.graph[current_vertex]:
```

```
            if neighbor not in visited:
                heapq.heappush(priority_queue, (edge_weight, neighbor))

        return mst

g = Graph()
g.add_edge('A', 'B', 1)
g.add_edge('A', 'C', 4)
g.add_edge('B', 'C', 2)
g.add_edge('B', 'D', 5)
g.add_edge('C', 'D', 1)
g.add_edge('D', 'E', 3)
g.add_edge('E', 'F', 2)
g.add_edge('F', 'D', 2)

mst = g.prim_mst()
print("Minimum Spanning Tree:")
for vertex, weight in mst:
    print(f"Edge: {vertex} - {weight}")
```