

## //Assignment 5(b)

```
import sys
```

```
class Graph:
```

```
    def __init__(self, vertices):
```

```
        self.V = vertices
```

```
        self.graph = [[0 for _ in range(vertices)] for _ in range(vertices)] # Adjacency Matrix
```

```
# Print the MST
```

```
def print_mst(self, parent):
```

```
    print("\nEdges in the Minimum Spanning Tree (Prim's Algorithm):")
```

```
    total_cost = 0
```

```
    for i in range(1, self.V):
```

```
        if parent[i] is not None and parent[i] != -1:
```

```
            print(f"{parent[i]} -- {i} == {self.graph[i][parent[i]]}")
```

```
            total_cost += self.graph[i][parent[i]]
```

```
    print("Minimum Spanning Tree Cost:", total_cost)
```

```
# Find vertex with minimum key value
```

```
def min_key(self, key, mstSet):
```

```
    min_val = sys.maxsize
```

```
    min_index = -1
```

```
    for v in range(self.V):
```

```
        if key[v] < min_val and not mstSet[v]:
```

```
            min_val = key[v]
```

```
            min_index = v
```

```
return min_index
```

```
# Prim's Algorithm
```

```
def prim_mst(self):
```

```
    key = [sys.maxsize] * self.V
```

```
    parent = [None] * self.V
```

```
    key[0] = 0
```

```
    parent[0] = -1
```

```
    mstSet = [False] * self.V
```

```
    for _ in range(self.V):
```

```
        u = self.min_key(key, mstSet)
```

```
        if u == -1: # No valid vertex left (disconnected graph)
```

```
            break
```

```
        mstSet[u] = True
```

```
        for v in range(self.V):
```

```
            if self.graph[u][v] > 0 and not mstSet[v] and key[v] > self.graph[u][v]:
```

```
                key[v] = self.graph[u][v]
```

```
                parent[v] = u
```

```
    self.print_mst(parent)
```

```
# ----- Main Program -----
```

```
if __name__ == "__main__":
```

```
# Example: 5 departments
```

```
# 0 = Main Gate, 1 = CS Dept, 2 = IT Dept, 3 = Library, 4 = Hostel
```

```
g = Graph(5)
```

```
# Fill adjacency matrix with distances
```

```
g.graph = [
```

```
    [0, 10, 8, 0, 0], # Gate
```

```
    [10, 0, 5, 3, 0], # CS
```

```
    [8, 5, 0, 7, 6], # IT
```

```
    [0, 3, 7, 0, 4], # Library
```

```
    [0, 0, 6, 4, 0]  # Hostel
```

```
]
```

```
# Run Prim's Algorithm
```

```
g.prim_mst()
```

//OUTPUT

Edges in the Minimum Spanning Tree (Prim's Algorithm):

0 -- 2 == 8

2 -- 4 == 6

2 -- 1 == 5

1 -- 3 == 3

Minimum Spanning Tree Cost: 22