**Name :** Atharva Arjun Goral

**Roll No :** 3101033

**Batch :** B  **Div :** A

## **AI**

# **Assignment No. 3**

**CODE :**

import heapq

def prim\_mst(graph, num\_nodes):

visited = [False] \* num\_nodes

min\_heap = [(0, 0)] # (cost, start\_node)

mst\_cost = 0

mst\_edges = []

parent = [-1] \* num\_nodes

while min\_heap:

cost, u = heapq.heappop(min\_heap)

if visited[u]:

continue

visited[u] = True

mst\_cost += cost

if parent[u] != -1:

mst\_edges.append((parent[u], u, cost))

for v, weight in graph[u]:

if not visited[v]:

heapq.heappush(min\_heap, (weight, v))

parent[v] = u

return mst\_cost, mst\_edges

# User input for the graph

num\_nodes = int(input("Enter number of nodes: "))

graph = {i: [] for i in range(num\_nodes)}

num\_edges = int(input("Enter number of edges: "))

print("Enter edges in the format: node1 node2 weight")

for \_ in range(num\_edges):

u, v, w = map(int, input().split())

graph[u].append((v, w))

graph[v].append((u, w))

mst\_cost, mst\_edges = prim\_mst(graph, num\_nodes)

print("Minimum Spanning Tree Cost:", mst\_cost)

print("Edges in MST:")

for edge in mst\_edges:

print(edge[0], "-", edge[1], "with cost", edge[2])

**OUTPUT :**

Enter number of nodes: 5

Enter number of edges: 7

Enter edges in the format: node1 node2 weight

0 1 2

0 3 6

1 2 3

1 3 8

1 4 5

2 4 7

3 4 9

Minimum Spanning Tree Cost: 16

Edges in MST:

0 - 1 with cost 2

1 - 2 with cost 3

2 - 4 with cost 5

4 - 3 with cost 6