**Prim's Minimum Spanning Tree (MST) algorithm:-**

**import sys**

**# Library for INT\_MAX**

**class Graph():**

**def \_\_init\_\_(self, vertices):**

**self.V = vertices**

**self.graph = [[0 for column in range(vertices)]**

**for row in range(vertices)]**

**# A utility function to print the constructed MST stored in parent[]**

**def printMST(self, parent):**

**print("Edge \tWeight")**

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

**print(parent[i], "-", i, "\t", self.graph[i][parent[i]])**

**# A utility function to find the vertex with minimum distance value**

**def minKey(self, key, mstSet):**

**# Initialize min value**

**min = sys.maxsize**

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

**if key[v] < min and mstSet[v] == False:**

**min = key[v]**

**min\_index = v**

**return min\_index**

**# Function to construct and print MST for a graph**

**def primMST(self):**

**# Key values used to pick minimum weight edge in cut**

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

**parent = [None] \* self.V # Array to store constructed MST**

**# Make key 0 so that this vertex is picked as first vertex**

**key[0] = 0**

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

**parent[0] = -1 # First node is always the root of**

**for cout in range(self.V):**

**# Pick the minimum distance vertex from**

**# the set of vertices not yet processed.**

**u = self.minKey(key, mstSet)**

**# Put the minimum distance vertex in the shortest path tree**

**mstSet[u] = True**

**# Update dist value of the adjacent vertices**

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

**# graph[u][v] is non zero only for adjacent vertices of m**

**# mstSet[v] is false for vertices not yet included in MST**

**if self.graph[u][v] > 0 and mstSet[v] == False and key[v] > self.graph[u][v]:**

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

**parent[v] = u**

**self.printMST(parent)**

**V=int(input("Enter no of vertices"))**

**g = Graph(V)**

**n=int(input("Enter total no of edges"))**

**for i in range(0, n):**

**u = int(input("Enter start vertex: "))**

**v = int(input("Enter end vertex: "))**

**c = int(input("Enter the cost of edge "))**

**g.graph[u][v] = c**

**g.graph[v][u] = c**

**g.primMST();**