```
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):
        Mst=[]
        Visited=set()
        Start_node=list(self.graph.keys())[0]
        Visited.add(start_node)
        Edges=[(cost,start_node,neighbor)for neighbor,cost in self.graph[start_node]]
        Heapq.heapify(edges)


        While edges:
            Cost,u,v=heapq.heappop(edges)
            If v not in visited:
                Visited.add(v)
                Mst.append((cost,u,v))
                For neighbor,n_cost in self.graph[v]:
                    If neighbor not in visited:
```

```
                Heapq.heappush(edges,(n_cost,v,neighbor))
        Return mst


Graph = Graph()
Num_edge=int(input(“ no of edges”))
For _ in range(num_edge):
    U,v,weight=input(“enter u v weight “).split()
    Graph.add_edge(u,v,int(weight))
Mst=graph.prim_mst()
Print(“mst”)
For edge in mst:
    Print(edge)
```