Date 02/03/2024

9.Aim: Write the python to implement Travelling Salesman ProblemProgram:

```
import sys

class TSP:
    def __init__(self, num_cities):
        self.num_cities = num_cities
        self.graph = [[0] * num_cities for _ in range(num_cities)]

    def add_edge(self, city1, city2, distance):
        self.graph[city1][city2] = distance
        self.graph[city2][city1] = distance

    def nearest_neighbor(self):
        visited = [False] * self.num_cities

        # Start from the first city
        current_city = 0
        visited[current_city] = True
        total_cost = 0
        tour = [current_city]

        for _ in range(self.num_cities - 1):
            min_distance = sys.maxsize
            nearest_city = None

            # Find the nearest unvisited city
            for city in range(self.num_cities):
                if not visited[city] and self.graph[current_city][city] < min_distance:
```

```
                min_distance = self.graph[current_city][city]
                nearest_city = city

        # Move to the nearest unvisited city
        visited[nearest_city] = True
        total_cost += min_distance
        tour.append(nearest_city)
        current_city = nearest_city

    # Return to the starting city
    total_cost += self.graph[tour[-1]][tour[0]]
    tour.append(tour[0])

    return tour, total_cost

# Example usage:
if __name__ == "__main__":
    tsp = TSP(4)
    tsp.add_edge(0, 1, 10)
    tsp.add_edge(0, 2, 15)
    tsp.add_edge(0, 3, 20)
    tsp.add_edge(1, 2, 35)
    tsp.add_edge(1, 3, 25)
    tsp.add_edge(2, 3, 30)

    tour, total_cost = tsp.nearest_neighbor()
    print("Nearest Neighbor Tour:", tour)
    print("Total Cost:", total_cost)
```

Output:\

```
Nearest Neighbor Tour: [0, 1, 3, 2, 0]
Total Cost: 80
```

Result: The given program has been executed successfully