**ASSIGNMENT 8**

Name: Aaryan Aher

PRN: 22210645

Div:- TY A [1]

Roll No.: 371003

**Title:**  Implement any scheme to find the optimal solution for the Traveling Salesperson problem and then solve the same problem instance using different algorithmic strategies and determine the optimal solution.

**CODE:**

# Function to solve the Traveling Salesman Problem (TSP) using recursion and backtracking

# graph: 2D list representing distances between cities

# v: list to track visited cities

# n: total number of cities

# current\_pos: the current city position

# cost: the total cost incurred so far

# count: number of cities visited so far

# ans: the minimum cost found for a complete tour

def tsp(graph, v, n, current\_pos, cost, count, ans):

    # Base case: if all cities are visited and there is a path back to the starting city

    if count == n and graph[current\_pos][0] > 0:

        # Update the answer with the minimum cost of completing the tour

        ans = min(ans, cost + graph[current\_pos][0])

        return ans

    # Try to visit all cities (from current position to other cities)

    for i in range(n):

        # Check if city i is not visited and there's a path from current\_pos to i

        if not v[i] and graph[current\_pos][i] > 0:

            # Mark city i as visited

            v[i] = True

            # Recursively call tsp to explore further paths and update the minimum cost

            ans = tsp(graph, v, n, i, cost + graph[current\_pos][i], count + 1, ans)

            # Backtrack: unmark city i to try other possibilities

            v[i] = False

    return ans  # Return the minimum cost found

# Main function to setup the problem and call the tsp function

def main():

    # Number of cities

    n = 4

    # Distance matrix representing the graph where graph[i][j] is the distance from city i to city j

    graph = [

        [0, 10, 15, 20],  # Distances from city 0 to others

        [10, 0, 35, 25],  # Distances from city 1 to others

        [15, 35, 0, 30],  # Distances from city 2 to others

        [20, 25, 30, 0]   # Distances from city 3 to others

    ]

    # Boolean list to track visited cities (initialize with False)

    v = [False] \* n

    # Initialize the answer (minimum cost) to infinity

    ans = float('inf')

    # Mark the first city (city 0) as visited

    v[0] = True

    # Call the tsp function starting from city 0, with an initial cost of 0 and 1 city visited

    ans = tsp(graph, v, n, 0, 0, 1, ans)

    # Print the minimum cost found for the tour

    print(ans)

# Driver code

if \_\_name\_\_ == "\_\_main\_\_":

    main()

**OUTPUT:**

