PRATICAL-5

AIM: Implementation of Traveling Salesman Problem.

◉ SOURCE CODE :

```
from itertools import permutations

def tsp_brute_force_with_steps(graph, start):
    # Get all cities except the starting one
    cities = list(range(len(graph)))
    cities.remove(start)

    # Generate all permutations of the cities
    min_cost = float('inf')
    best_path = []

    print("Steps:")
    for perm in permutations(cities):
        current_cost = 0
        current_path = [start] + list(perm) + [start]

        # Calculate the cost of the current permutation
        step_costs = []
        for i in range(len(current_path) - 1):
            step_cost = graph[current_path[i]][current_path[i+1]]
            step_costs.append(step_cost)
            current_cost += step_cost

        print(f"Path: {current_path}, Step Costs: {step_costs}, Total Cost: {current_cost}")

        if current_cost < min_cost:
            min_cost = current_cost
            best_path = current_path

    print("\nOptimal Path:", best_path)
    print("Minimum Cost:", min_cost)
    return min_cost, best_path

# Example Usage
graph = [
    [0, 10, 15, 20],
    [10, 0, 35, 25],
    [15, 35, 0, 30],
    [20, 25, 30, 0]
]
start_city = 0
tsp_brute_force_with_steps(graph, start_city)
```

◉ OUTPUT :

```
Steps:
Path: [0, 1, 2, 3, 0], Step Costs: [10, 35, 30, 20], Total Cost: 95
Path: [0, 1, 3, 2, 0], Step Costs: [10, 25, 30, 15], Total Cost: 80
Path: [0, 2, 1, 3, 0], Step Costs: [15, 35, 25, 20], Total Cost: 95
Path: [0, 2, 3, 1, 0], Step Costs: [15, 30, 25, 10], Total Cost: 80
Path: [0, 3, 1, 2, 0], Step Costs: [20, 25, 35, 15], Total Cost: 95
Path: [0, 3, 2, 1, 0], Step Costs: [20, 30, 35, 10], Total Cost: 95

Optimal Path: [0, 1, 3, 2, 0]
Minimum Cost: 80
```