Naive Travelling Salesman Report

The given Java program solves the Traveling Salesman Problem (TSP) through the use of a brute-force permutation strategy. A summary of the findings, including the effectiveness of the naive strategy, is provided below:

**Input:** The user provides input in the form of a distance matrix M, where M[i][j] denotes the distance between cities i and j.

A four-city problem is used in the example.

**Output:** For the given input, the application successfully computes and shows the shortest route and shortest distance. For the given example:

Enter the number of cities: 4

Enter the distance matrix (M[i][j]):

0 10 15 20

10 0 35 25

15 35 0 30

20 25 30 0

Shortest Route: [0, 1, 3, 2, 0]

Shortest Distance: 80

**Brute-Force technique:** This program takes a brute-force method, creating all possible city combinations and computing the overall distance for each one. The permutation with the least distance is then chosen. While this strategy ensures the best result, it is inefficient for many cities due to its exponential time complexity. There are n! permutations to consider for n cities, resulting in a factorial temporal complexity that becomes prohibitive as n rises.

**Permutations:** The most computationally costly element of the process is generating all permutations. As n increases, the number of permutations grows factorially, resulting in significant processing cost.

**Performance:** For modest examples of the TSP, such as the 4-city problem, the program runs well. However, as the number of cities (n) increases, its performance rapidly diminishes. Due to the exponential time-based complexity, solving TSP instances with a large number of cities becomes impracticable.