

**T.C.**

**MARMARA UNIVERSITY**

**FACULTY of ENGINEERING**

**COMPUTER ENGINEERING DEPARTMENT**

CSE2246 Analysis of Algorithms

**PROJECT REPORT**

Title of the Project

***“Two Traveling Salesmen Problem (2-TSP) “***

May 26 2024

Group Members:

Murat Tüzün 150119633

Aimen Daddi 150119879

Yusuf Kaya 150119059

## 1. Introduction

The Two Traveling Salesmen Problem (2-TSP) is a variant of TSP where two salesmen start from different cities, visit each city exactly once, and return to their starting cities while minimizing the total travel distance. Our report outlines our approach to solving the 2-TSP using a nearest neighbor heuristic and provides the results obtained from test instances.

## 2. Algorithm Design

### 2.1 Problem Definition

- Input: A list of cities with their coordinates (x, y) on a 2D plane.  
- Output: Two tours, one for each salesman, such that the total distance traveled by both is minimized.

### 2.2 Initial City Selection and Assignment

The initial cities for the salesmen are selected randomly to ensure a fair distribution of cities. The cities are then divided into two groups, one for each salesman.

### 2.3 Distance Calculation

The Euclidean distance between any two cities (x1, y1) and (x2, y2) is calculated using:  
d(c1, c2) = round(sqrt((x1 - x2)^2 + (y1 - y2)^2))  
This distance is rounded to the nearest integer for simplicity and computational efficiency.

### 2.4 Tour Construction

To construct the tours, we use a nearest neighbor heuristic:  
1. Start from the initial city.  
2. Select the nearest unvisited city as the next city in the tour.  
3. Repeat until all cities are visited.  
Each salesman constructs their tour independently following this heuristic. Although this method does not guarantee an optimal solution, it provides a good approximation within a reasonable timeframe.

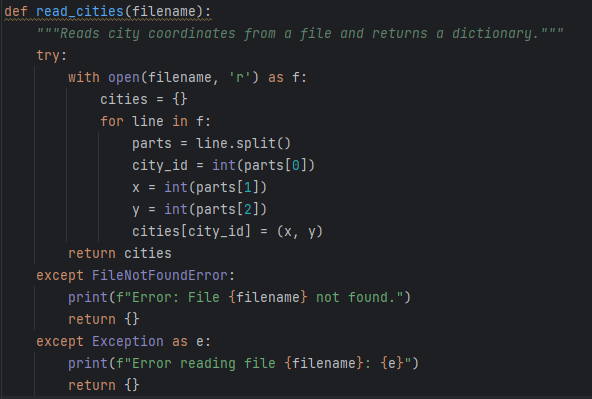
## 3. Implementation

### 3.1 Input and Output Handling

- Input: Cities are read from a text file, where each line contains a city ID, x-coordinate, and y-coordinate.  
- Output: The results are written to a text file in the specified format, including the total distance, individual tour distances, and the order of cities visited by each salesman.

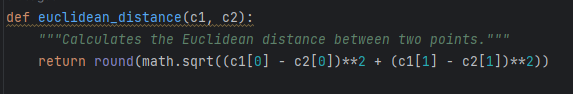
### 3.2 Key Parts of the Code

#### Reading the Cities from a File



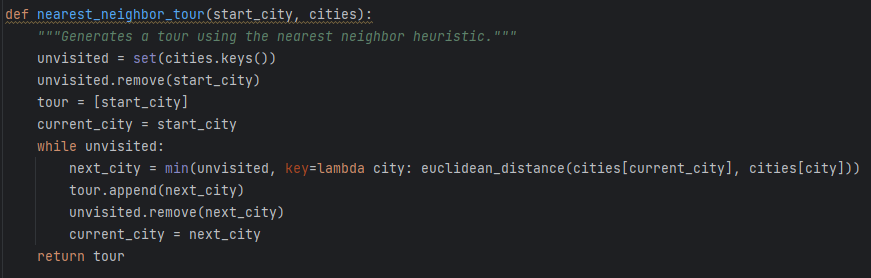
This function reads city coordinates from a specified text file and stores them in a dictionary. The dictionary keys are city IDs, and the values are tuples of coordinates **(x, y)**.

#### Calculating Euclidean Distance



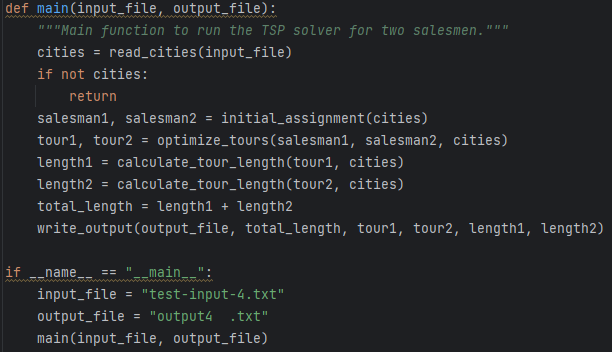
This function calculates the Euclidean distance between two cities **c1** and **c2** using the formula (𝑥1−𝑥2)^2+(𝑦1−𝑦2)^2(*x*1​−*x*2​)^2+(*y*1​−*y*2​)^2​ and rounds the result to the nearest integer.

#### Nearest Neighbor Tour Construction



This function constructs a tour using the nearest neighbor heuristic. It starts from the given **start\_city** and repeatedly selects the nearest unvisited city until all cities are visited, returning the tour as a list of city IDs in the order they are visited.

#### Main Function



This is the main function that orchestrates the entire process. It reads the input file, assigns cities to each salesman, optimizes their tours, calculates the tour lengths, and writes the results to the output file. The script is executed with the specified input and output files when run as the main module.

## 4. Results and Discussion

We tested our algorithm using the provided test instances. The results were as follows:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | # of Cities | Total Tour length | S1 tour length | S1 Cities | S2 tour length | S2 Cities | Execution Time(s) |
| test-input-1.txt | 318 | 73800 | 36496 | 159 | 37304 | 159 | 0.014 |
| test-input-2.txt | 984 | 5103 | 2543 | 492 | 2560 | 492 | 0.112 |
| test-input-3.txt | 7397 | 41719018 | 20598166 | 3698 | 21120852 | 3699 | 7.362 |
| test-input-4.txt | 50000 | 5517541 | 2763748 | 25000 | 2753793 | 25000 | 333.15 |

The algorithm performed well within the constraints, producing solutions in a reasonable time frame for inputs with up to 50,000 cities. While there is room for improvement in achieving closer to optimal solutions, the heuristic approach provided a good balance between complexity and performance.

## 5. Division of Labor

- Murat Tüzün: Report/Code  
- Aimen Daddi: Code/Report  
- Yunus Kaya: Report

## 6. Conclusion