SWE 510 – Data Structures and Algorithms
Assignment 1 Report – Migros Delivery

Deniz Baran ASLAN
ID: 2021719183
Date: 25.04.2022

Algorithm Explanation

This script is an implementation of the traveling salesman problem in Java, using arrays and ArrayLists. It performs the following operations in order:

- Prompt the user for input.
- Read coordinates from the input file.
- Store the coordinates in an array.
- Calculate the distances between each node and store them in a 2d array.
- Start at Migros node.
- Iterate over remaining nodes to find the nearest one.
- Move to the nearest node.
- Repeat until all nodes are visited.
- Move back to Migros node.
- Add up the distance.
- Print the resulting path and total distance.

Program Inputs

The script needs an input file that contains node coordinates, with one node designated as “Migros”. Once run, the script also prompts the user for input regarding which file should be read. (e.g. “input02.txt”)

Program Outputs

In the absence of any unexpected errors, the script prints out the following:

- The name of the input file selected by the user.
- The shortest path that starts from Migros, visits each node once, and returns to Migros at the end.
- The total distance traveled.

In case the specified input file cannot be found, the program prints an error message and quits. This portion of the code was taken from the “Reading Text Files” guide.

Sample Program Input and Output

```
Which input file should be read?
>input01.txt
File is: input01.txt
Shortest Route: [11, 12, 10, 6, 8, 4, 9, 5, 7, 1, 3, 2, 11]
Distance: 4.26184571418472

File is: input02.txt
Shortest Route: [10, 9, 8, 7, 12, 11, 3, 4, 5, 2, 1, 6, 10]
Distance: 3.972706470425156

File is: input03.txt
Shortest Route: [10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 12, 11, 10]
Distance: 3.022090246134521
```

```
File is: input04.txt
Shortest Route: [7, 8, 6, 2, 3, 4, 5, 1, 7]
Distance: 3.7487628231434895

Which input file should be read?
>input06.txt
File is: input06.txt
File cannot be found.
```

Disclaimer

My original plan was to use brute-force calculation of each permutation of possible paths and pick the shortest one. I was unable to implement this solution using only arrays and ArrayLists, so I opted to implement a nearest neighbor approach instead. I understand that this method might not always yield the optimum path.