EKİN NOHUTÇU 150116067
ERTUĞRUL SAĞDIÇ 150116061

ANALYSIS OF ALGORITHMS ASSIGNMENT #3
REPORT

In this project, we were required to find an optimal solution for Travelling Salesman Problem. TSP is a NP-Hard problem and its hard to find optimal solutions. We did our best and used the Nearest Neighbour to find initial solution of the problem and then we used Two-Opt algorithm to find an optimal solution for Travelling Salesman Problem.

TSP asks the following question: Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?

Inputs are the n cities, with their locations (x and y coordinates) in a 2D grid.
Output is ordering (tour) of these cities so that total distance to travel is minimized.

We can find the distance between two cities by using Euclidean algorithm computed as follows:

$$d(c_1, c_2) = round\left(\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}\right)$$

For verifying our algorithm we used the tsp-verifier.py which is given by the instuctor.

To approach an optimal solution, we have used Nearest Neighbour and Two-Opt methods.

Algorithm for Optimal Solution:

1) We read the input file and assign every city into City object with the values: ID,x and y coordinates.
2) For every City object, we stored them in ArrayList because we don't know the size of the input.
3) We took this ArrayList and send it into Nearest Neighbour algorithm to find non-optimal path for optimize this path.
4) Then, we have implemented the Two-Opt algorithm to optimize the Nearest Neighbour path. Because the result of Nearest Neighbour algorithm does not give the optimal solution most of the time.
5) The resulting output is the optimal solution for our algorithm and it is very close to the our instructors results.

6) After finding the results, we have verified our outputs by using the tsp-verifier.py file.

Nearest Neighbour Algorithm:

Nearest Neighbour algorithm starts with the first City and it finds the nearest city iteratively and returns an ArrayList of cities which gives non-optimal solution.
First we assign the first element of our City ArrayList and assign it to City object and remove it from the initial list. Then, we create a new ArrayList for our final solution and add the removed city into this solution ArrayList.
This continues iteratively and final solution ArrayList is our non-optimal solution.

Two-Opt Algorithm:

Two-Opt algorithm starts with initial solution and iteratively looks for improvement opportunities in the neighbourhood of that solution. In our project, first we give the starting ArrayList of the tour that we get output from the Nearest Neighbour algorithm. Then, we assigned the tour length of initial path to the best tour length variable. Then, we choose the first four city for comparing their distances between each of them. This continues for every nodes in a for loop to choose the shortest path for every four nodes and when find, swap the cities and continues. If there is no more optimizations (swaps) the algorithm returns the ArrayList of cities that is the optimum solution for our algorithm.
In Swap function, we add first i'th cities into new arraylist. Then we add the cities from i'th to j'th in reverse order. Finally we add the rest of the cities.

****In this project, we used Discord app and implement the algorithm and project report together.**

REFERENCES:

- https://towardsdatascience.com/around-the-world-in-90-414-kilometers-ce84c03b8552
- http://160592857366.free.fr/joe/ebooks/ShareData/Heuristics%20for%20the%20Traveling%20Salesman%20Problem%20By%20Christian%20Nillson.pdf
- https://www.ijeat.org/wp-content/uploads/papers/v4i6/F4173084615.pdf

OUTPUTS OF OUR ALGORITHM

OUTPUT OF EXAMPLE-INPUT-1:

```
Total number of cites: 76
------------------------
GREEDY ALGORITHM TOUR LENGTH:
150393
------------------------
TWO-OPT OPTIMIZATION TOUR LENGTH:
126642

Process finished with exit code 0
```

OUTPUT OF EXAMPLE-INPUT-2:

```
Total number of cites: 280
------------------------
GREEDY ALGORITHM TOUR LENGTH:
2808
------------------------
TWO-OPT OPTIMIZATION TOUR LENGTH:
2760

Process finished with exit code 0
```

OUTPUT OF EXAMPLE-INPUT-3:

```
Total number of cites: 15112
------------------------
GREEDY ALGORITHM TOUR LENGTH:
1964948
------------------------
TWO-OPT OPTIMIZATION TOUR LENGTH:
1709386

Process finished with exit code 0
```

OUTPUT OF TEST-INPUT-1:

```
Total number of cites: 280
------------------------
GREEDY ALGORITHM TOUR LENGTH:
2808
------------------------
TWO-OPT OPTIMIZATION TOUR LENGTH:
2760

Process finished with exit code 0
```

OUTPUT OF TEST-INPUT-2:

```
Total number of cites: 1002
------------------------
GREEDY ALGORITHM TOUR LENGTH:
338693
------------------------
TWO-OPT OPTIMIZATION TOUR LENGTH:
291114

Process finished with exit code 0
```

OUTPUT OF TEST-INPUT-3:

```
Total number of cites: 33810
------------------------
GREEDY ALGORITHM TOUR LENGTH:
77901311
------------------------
TWO-OPT OPTIMIZATION TOUR LENGTH:
71325508

Process finished with exit code 0
```

OUTPUT OF TEST-INPUT-4:

```
Total number of cites: 2924
------------------------
GREEDY ALGORITHM TOUR LENGTH:
12602
------------------------
TWO-OPT OPTIMIZATION TOUR LENGTH:
11236

Process finished with exit code 0
```