**Elif Balcı**
150117069

# Travelling Salesman Problem

**14th June 2020**

## Introduction

Travelling salesman problem is finding the shortest route starting from any node, visiting all of the nodes and returning to the first node. TSP is known to be a non-polynomial problem and there are a lot of algorithms to solve it. Some of the algorithms can be listed as Nearest Neighbor, Lin-Kernighan, Simulated Annealing, Tabu-Search, Genetic Algorithms, Tour Data Structure, Ant Colony Optimization, Tour Data Structure, etc.[1]

In this project nearest neighbor algorithm to establish an initial route and 2-OPT algorithm to optimize it. This report includes detailed information about these two algorithms and failed attempts for solving the travelling salesman problem.

## Project Structure

Main Class: Main Class is used for file I/O and creating the object that will help to solve the problem.

InputHandler Class: The InputHandler class is created to read and create an array list from the lines of the city text file.

City Class: City Class takes the array list of lines and creates cities out of it and save them to static City array list

MatrixCreator Class: Main duty of the MatrixCreator class is to create an adjacency matrix from the Cities array list. Class also includes the method that computes

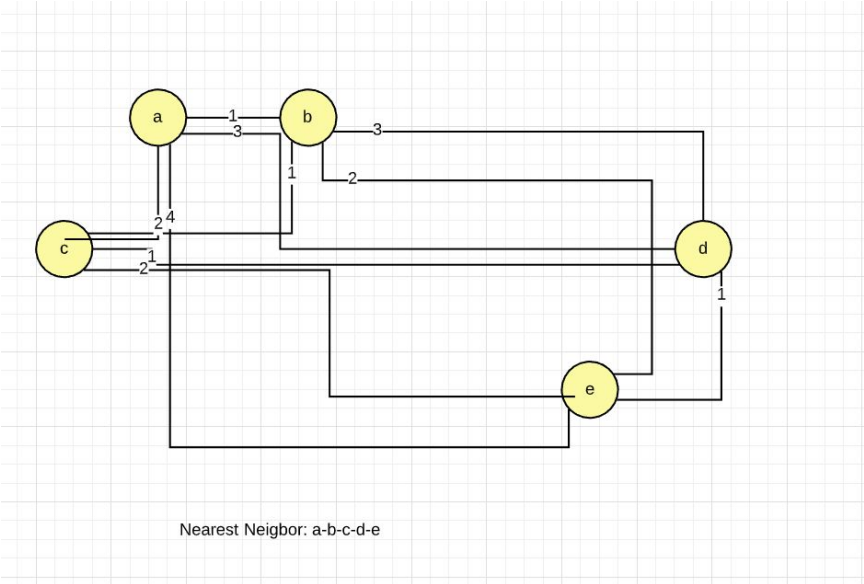
TSPSolver Class: TSPSolver Class is the class that contains all implementations of algorithms. Algorithms will be explained.

TSPSolverVeryBigList Class: This class is added after checking the input list. Using a 2-dimensional adjacency matrix was causing out of memory error. So this class won't use an adjacency matrix but compute distances only if needed.

## Nearest Neighbor

Nearest neighbor algorithm is probably one of the easiest to implement.

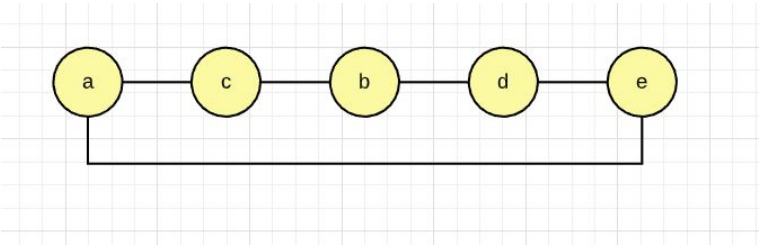
Starting at a random node, salesmen should visit the nearest unvisited city until every city in the list is visited. When all cities are visited, salesmen should return to the first city.



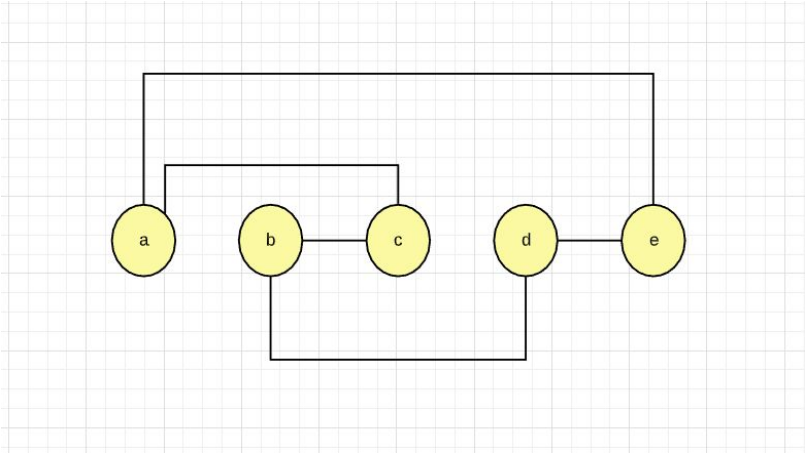

## 2 - OPT

2-opt heuristic can be defined briefly as checking if is there any 2-changes that make the route shorter.[2] There can n*(n-3) /2 changes can be made for a route that contains n cities.

After computing the length of those routes, the route should be changed only if there is an improvement. This is the 2-opt in nutshell.




List before 2-opt




A probable outcome of 2-opt

An example can be viewed above. If there is a route as a,b,c,d,a; a possible variation for 2-opt algorithm could be a,c,b,d,e.

2-opt algorithm does not have a polynomial time. It does not guarantee approximate correctness.

But empirically, typically returns the tour with total cost close to the minimum possible.[2]

## A Note for the Implementation

In TSPSolver Class, I observed that finding the best starting point for the nearest neighbor before 2-opt heuristic is sometimes effective although sometimes it makes the route longer. Hence a method to find the best starting point has been implemented and is called when the input list is short. This way in some cases route length decreased up to %5 percent.

Test outputs:

Test-1: 2675, Test-2: 271741, Test-3: 70876333, Test-4: 10822

## References

[1] Christian Nilsson “Heuristics for the Traveling Salesman Problem” January 2003

[2] Tim RoughGarden - Tim Roughgarden Lectures “Algorithms for NP-Hard Problems Section 20.4: The 2-OPT Heuristic for the TSP” May 2020

[3]http://on-demand.gputechconf.com/gtc/2014/presentations/S4534-high-speed-2-opt-tsp-solver.pdf