MARMARA UNIVERSITY – FACULTY OF ENGINEERING

CSE 2046/2246 HOMEWORK #2 REPORT

Name, Surname: Hasan Şenyurt – Mert Akbal – Hakan Kenar – Ahsen Yağmur Kahyaoğlu

Student ID: 150120531 – 150119703 – 150119675 – 150119788

Department: Computer Engineering

Code – Algorithm

Our algorithm consists of two parts. Firstly, we find a tour with Nearest Neighbor algorithm, then we try to improve that tour. It is not just simple Nearest Neighbor. It is used to find initial bad tour as simple step.

```
#Checks nearest neighbor of the city by looking distance between city and unvisitied cities.
def nearestNeighbor(city, unvisited_cities):

    minimum_distance = float('inf')
    for neighbor_city in unvisited_cities:

        distance = calculateDistance(city, neighbor_city)

        #if new distance is smaller than minimum distance that is assigned before, assign new minimum distance and nearest neighbor.
        if distance < minimum_distance:
            minimum_distance = distance
            nearest_neighbor = neighbor_city

    return nearest_neighbor, minimum_distance
```

Figure-1: Nearest Neighbor Algorithm

Nearest neighbor algorithm checks nearest unvisited city for each city one by one. It calculates distance between cities by using Euclidian distance formula. When it finds minimum distance, then it means nearest neighbor is found. This algorithm is a simple step to start improving function.

```
#Finds half tsp tour, and returns the tour with coordinates and total distance in tour.
def findHalfTspTour(cities):

    total_distance = 0 #initial assignment for total distance.
    start_city = cities[0][1:] #take just coordinates not id.
    unvisited_cities = []

    for i in cities[1:]:
        unvisited_cities.append(i[1:]) #take all cities as unvisited except start city.

    tour = [start_city]
    initial_city = start_city

    #creating tour with searching nearest neighbor of the cities one by one.
    for i in range(len(tour),math.ceil(len(cities) / 2)):

        nearest_neighbor, distance = nearestNeighbor(initial_city, unvisited_cities) #choose nearest neighbor
        total_distance += distance #add distance to total distance.

        tour.append(nearest_neighbor) #add to tour
        unvisited_cities.remove(nearest_neighbor) #remove visited city in unvisited_cities list.
        initial_city = nearest_neighbor

    tour.append(start_city)  # Return to the start city at the end of the tour.
    total_distance += calculateDistance(tour[-1], tour[-2]) #distance between start point and last point before start

    return tour, total_distance
```

Figure-2: Finding Half TSP Tour with Nearest Neighbor

As we can see in Figure-2 above, this is simple nearest neighbor algorithm to find a tour. Firstly, we initialize first city, which is in input files, as starting city of the tour for all input files. Secondly, we search for nearest neighbor for each city in a loop, and as we find nearest neighbor of the city, we append nearest neighbor to tour. It continues until length of tour equals to length of cities divided by two. We calculated total distance of the tour as well to compare with improved tour's distance.

```python
""" this function is the key point of our project. we send tour which is found by nearest neighbor algorithm to check
    if there are better way to connect cities to create new tour. we take each city and create new tour by changing the
    route. It calculates the total distance again, if it is better than before, this will be our new route. We keep same
    start and end city."""
def improveTourAlgorithm(tour, distances):

    improved = True
    #initial best distance which is nearest neighbor result.
    best_distance = calculateTotalDistanceOfTour(tour, distances)

    #algorithm continues until there is no more improving.
    while improved:
        improved = False
        #starts from second index of the coordinates
        for i in range(1, len(tour) - 1):
            for j in range(i + 1, len(tour)):

                #creates new tour by changing cities in the tour.
                improved_tour = tour[:i] + tour[i:j][::-1] + tour[j:]
                #calculates new distance to check if it is improved or not.
                improved_tour_distance = calculateTotalDistanceOfTour(improved_tour, distances)

                #assign new tour if new distance is better.
                if improved_tour_distance < best_distance:
                    improved = True
                    tour = improved_tour
                    best_distance = improved_tour_distance

        return tour
```

Figure-3: Improving Tour Function

This is the function in Figure-3 above that we improve our tour that is found with Nearest Neighbor algorithm. Main point of the improving algorithm is creating new tours in current tour constantly until we find better tour. Function starts from second city to find new neighbor respectively. It looks at third city, forth city, fifth city until the end to find new neighbor to second city. It continues like this way. After finding best neighbor to second city, it searches for third city in the same way. After finding new tour, it calculates its total distance and compares with previous tour that is found. If it is smaller than previous tour, then our new tour becomes best tour. Until no more improving in terms of distance, algorithm stops. We can take a look at a simple example to explain algorithm better. Also, we always used Python 'List' as **data structures**.

```python
tour = [[1,2],[3,4],[5,6],[7,8],[9,10],[11,12],[1,2]]
```

Figure-4: Example Tour

Assume that we have a tour that is found by Nearest Neighbor algorithm as in figure-4 above. Our algorithm tries to find new tours by producing different tours by connecting each city one by one with the cities ahead. If each tour it produces is better than the previous one, our tour will be improved. Possible tours can be seen in Figure-5 below. Total distance of these tours will be calculated, and best one will be selected.

```
[[1, 2], [3, 4], [5, 6], [7, 8], [9, 10], [11, 12], [1, 2]]
[[1, 2], [5, 6], [3, 4], [7, 8], [9, 10], [11, 12], [1, 2]]
[[1, 2], [7, 8], [5, 6], [3, 4], [9, 10], [11, 12], [1, 2]]
[[1, 2], [9, 10], [7, 8], [5, 6], [3, 4], [11, 12], [1, 2]]
[[1, 2], [11, 12], [9, 10], [7, 8], [5, 6], [3, 4], [1, 2]]
[[1, 2], [3, 4], [5, 6], [7, 8], [9, 10], [11, 12], [1, 2]]
[[1, 2], [3, 4], [7, 8], [5, 6], [9, 10], [11, 12], [1, 2]]
[[1, 2], [3, 4], [9, 10], [7, 8], [5, 6], [11, 12], [1, 2]]
[[1, 2], [3, 4], [11, 12], [9, 10], [7, 8], [5, 6], [1, 2]]
[[1, 2], [3, 4], [5, 6], [7, 8], [9, 10], [11, 12], [1, 2]]
[[1, 2], [3, 4], [5, 6], [9, 10], [7, 8], [11, 12], [1, 2]]
[[1, 2], [3, 4], [5, 6], [11, 12], [9, 10], [7, 8], [1, 2]]
[[1, 2], [3, 4], [5, 6], [7, 8], [9, 10], [11, 12], [1, 2]]
[[1, 2], [3, 4], [5, 6], [7, 8], [11, 12], [9, 10], [1, 2]]
[[1, 2], [3, 4], [5, 6], [7, 8], [9, 10], [11, 12], [1, 2]]
```

Figure-5: Possible New Tours

Test Results (3 Example Input – 4 Test Input)

```
City ID's Half TSP Tour (Nearest Neighbor): [0, 75, 1, 2, 23, 22, 21, 25, 24, 46, 45, 44, 48, 47, 69, 68, 67, 50, 49, 52, 53, 54, 42, 43, 28, 29, 30, 31, 19, 20, 5, 6, 7, 8, 9, 10, 11, 12]
Total Distance: 55097
City ID's Half TSP Tour (Improved): [0, 75, 12, 11, 10, 9, 8, 7, 6, 5, 20, 19, 31, 30, 29, 28, 43, 42, 54, 53, 52, 49, 50, 67, 68, 69, 47, 48, 44, 45, 46, 24, 25, 21, 22, 23, 1, 2]
Total Distance: 54120
Distance improved: 977
Running Time in Seconds: 0.007
```

Figure-6: Results of Example Input 1

```
City ID's Half TSP Tour (Nearest Neighbor): [0, 2, 3, 279, 278, 4, 277, 276, 275, 274, 273, 272, 271, 16, 17, 18, 19, 20, 21, 128, 127, 126, 125, 30, 31, 32, 29, 28, 27, 26, 22, 25, 23, 24, 14, 13, 12,
 11, 10, 8, 7, 9, 6, 5, 1, 242, 243, 241, 240, 239, 238, 231, 232, 233, 234, 235, 236, 237, 246, 245, 244, 247, 250, 251, 230, 229, 228, 227, 226, 225, 224, 223, 222, 219, 218, 215, 214, 211, 210, 207,
 208, 209, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 138, 137, 136, 135, 134, 270, 269, 268, 267, 140, 139, 148, 141, 142, 143, 144, 145, 146, 147, 149, 150, 178, 151,
152, 156, 153, 155, 154, 129, 130, 131, 132, 133, 15, 248, 249, 206, 205, 204, 203, 202, 200, 199]
Total Distance: 1610
City ID's Half TSP Tour (Improved): [0, 1, 5, 6, 7, 9, 8, 10, 11, 12, 13, 14, 24, 23, 25, 22, 26, 27, 28, 29, 32, 31, 30, 125, 126, 127, 128, 21, 20, 19, 133, 132, 131, 130, 129, 154, 155, 153, 156, 15
2, 151, 178, 150, 149, 147, 146, 145, 199, 200, 202, 144, 143, 142, 141, 148, 139, 140, 138, 137, 136, 135, 134, 270, 269, 268, 267, 266, 265, 264, 263, 262, 203, 204, 205, 206, 253, 254, 257, 258, 259
, 261, 15, 18, 17, 16, 271, 272, 273, 274, 275, 276, 260, 277, 4, 278, 248, 249, 256, 255, 252, 209, 208, 207, 210, 211, 214, 215, 218, 219, 222, 223, 224, 225, 226, 227, 228, 229, 230, 251, 250, 247,
244, 245, 246, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 243, 242, 2, 3, 279]
Total Distance: 1503
Distance improved: 107
Running Time in Seconds: 0.227
```

Figure-7: Results of Example Input 2

```
Total Distance (Nearest Neighbor): 824056
Total Distance (Improved): 790904
Distance improved: 33152
Running Time in Seconds: 7029.291
```

Figure-8: Results of Example Input 3

```
Total Distance (Nearest Neighbor): 1610
Total Distance (Improved): 1503
Distance improved: 107
Running Time in Seconds: 0.189
```

Figure-9: Results of TEST INPUT 1

```
Total Distance (Nearest Neighbor): 143196
Total Distance (Improved): 135938
Distance improved: 7258
Running Time in Seconds: 7.915
```

Figure-10: Results of TEST INPUT 2

```
Total Distance (Nearest Neighbor): 5863
Total Distance (Improved): 5569
Distance improved: 294
Running Time in Seconds: 243.053
```

Figure-11: Results of TEST INPUT 4

```
Total Distance (Nearest Neighbor): 35227320
Running Time in Seconds: 448.883
```

Figure-12: Results of TEST INPUT 3 (Improving function took too much time, we could not get the results of improving function because of time limit. We decided to take NN algorithm as result only.)

Division of Labor

Hasan Şenyurt - 150120531: Research, NN Algorithm, Improving Function, Testing, Report

Mert Akbal - 150119703: Research, Improving Function, Report

Hakan Kenar - 150119675: Research, NN Algorithm, Testing, Report

Ahsen Yağmur Kahyaoğlu - 150119788: Research, NN Algorithm, Testing, Report