MARMARA UNIVERSITY
FACULTY OF ENGINEERING

Algorithm Analysis

Project-2

Half Traveling Salesman Problem (H-TSP)

Due Date: 06.06.2023

	Name Surname	Student Id
1	Niyazi Ozan Ateş	150121991
2	Ahmet Arda Nalbant	150121004
3	Hasan Özeren	150121036
4	Umut Bayar	150120043

Division of Labor

We all started on a Discord call to understand and discuss of how to find the optimal solution to the given problem. We concluded to use our knowledge of our lecture Probability and Statistics for Computer Engineering.

Ahmet Arda Nalbant had the idea of finding the median of both x and y and use them to find the closest point to these values. He also created that part of the coding.

Niyazi Ozan Ateş came to the idea to use the Closest Neighbour Algorithm. Also, he came to the point to use the upper and lower points of this list. So, we can go to the closest point from the top or bottom of the list and wrote the code for that.

Hasan Özeren was keeping himself busy with getting the correct input from the input file and giving the correct output to the output file. He also managed to put the correct values to an ArrayList. We found this easy to work with and its flexible.

Umut Bayar gave us the idea to create another class that keep track of all the information of the nodes. He also, wrote the code to find the correct distance of the H-TSP.

The report is written through Discord by participating all together.

Purpose

The purpose of this project was to find as optimal as possible solution for the half travelling salesman problem. This means that we go through some two-dimensional coordinates only once. We do this until half of the coordinates has been visited and after that we return to the starting point. We could have started from any point. We will explain the reason of choosing a particular point later.

Algorithm

The code is written in java. We start the algorithm by asking the user for an input file from the console. If it is the case that the input file is found, we will start the program. Otherwise, we give an error to the console.

We have created two classes. The first one is our main class. The second one holds the information of the points. Such as the ID, x, and y values. After we have found the input file, we will create an ArrayList and put all the points into the ArrayList by reading it from the input file. The reason why we take the inputs as nodes from another class in the Arraylist is that the use of the extra arraylist uses extra memory and it is easier to access the values from a single arraylist.

To get a better solution we decided to try to find the best possible starting point based on the probabilities of the point distribution in general. We have used a method that we have seen in our lecture called Probability and Statistics for Computer Engineering. We concluded to find the median of both x and y. After we have found these coordinates, we find the closest point to these coordinates. This will be our starting point. The reason we have used this method is because the median will most of the time eliminate to reach the outliners. Which gives us a better starting point and given us most of the time one of the best solutions.

The calculations will differ based on the input size. If the size in odd, we take the middle point as the coordinates for the median. Otherwise, it is even, and we will take the average of the two middle points to get the coordinates of both x and y.

To create the path of the Half Travelling Salesman Problem we create an algorithm that uses a $O(n^2)$ time complexity. Our starting point is the point that we have found in the previous part. This point was the closest point to the x and y median. Also, we create a new ArrayList to keep track of the path. Our algorithm looks like the Closest Neighbour Algorithm.

We search for the closest point for the start and end of the list. We do this n/2 times by looking through a n sized list. If it is the case that we have found a closer point to the starting point than the ending point of the list, we will add it on the top of the path. So, the new starting point will be the last found point in this case. Otherwise, we have found a point that is close to the ending point than the starting point of the list. In this case we add the new point to the end of the path. While doing this we will remove the closest point from the list. The reason for this is so we don't get duplicated points.

The reason of using this method is because it has a high probability to find a good solution. Also, it is fast and in multiple aspects of mathematics it eliminates creating unnecessary paths and calculation.

Another task for this project was to find the total distance the salesman has travelled. We did this by going through the whole path and at in the end we add the distance from the starting and ending point. This is because the salesman needs to end at its starting point. We do this with the given formula as given below:

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

At last, we print the distance and the IDs of the path to both the console and output file.

Result and Conclusion

As the conclusion we believe that we have found a pretty good solution to the problem. Both memory and time wise. Based on the input examples we even have found a distance 60 times smaller than the given output examples. This is a huge improvement for such a normal sized input. While for finding the optimal solution the big O is equal to n!, we have found a close to the optimal solution with a n^2 time complexity.

The reason for this is because we don't check every single combination, which is n! combinations, but we make a prediction on the starting poin. While doing this we use a two-sided closest neighbour algorithm that will find a good solution for the given problem.

Example

Input

```
0 200 800
1 3600 2300
2 3100 3300
3 4700 5750
4 5400 5750
5 5608 7103
6 4493 7102
7 3600 6950
8 3100 7250
9 4700 8450
10 5400 8450
11 5610 10053
12 4492 10052
13 3600 10800
14 3100 10950
15 4700 11650
16 5400 11650
17 6650 10800
18 7300 10950
19 7300 7250
20 6650 6950
21 7300 3300
22 6650 2300
23 5400 1600
24 8350 2300
25 7850 3300
26 9450 5750
27 10150 5750
```

Output