Anastasia Kovalchuk – 19317511

CS211 Data Structures & Algorithms

1. *Explanation of TSP algorithm used.*

The Traveling Salesman Problem (TSP) in our project is concerned with finding the quickest way of delivering Apache orders to all the houses. To tackle this problem, I used **search heuristics.** The advantage of these methods is that you can keep them relatively simple and intuitive, while they can find solutions that are (close to) the optimum. Also, they scale better so applying them to a larger TSP will less likely crash your machine. However, a disadvantage is that you will not know how far you are from the optimum. The search heuristic I used in my project is **Nearest Neighbours**.

Nearest Neighbours is one of the simplest heuristics out there. It is part of the family of constructive search heuristics, meaning it gradually builds a route. Starting with one house and stopping only when all houses have been visited. It is greedy in nature; at each step it chooses the house that is closest to the current house.

Every time the algorithm is ran it will generate exactly the same solution. However, this is a big downside of this algorithm. Because of its greedy nature, it will always go for immediate gains and miss out on opportunities that will pay out in a longer term. Nevertheless, Nearest Neighbours has given me a feasible solution.

2. *Code Structure (Objects, Classes, Graphics, etc.)*

My code is split into two classes: *tsp* and *Window*. *Tsp* is simply used to create the interface window for the application. It takes the initialized width (850) and height (540) as the Window's parameters.

Window is the prominent class and is used to instantiate the GUI window, take in user input and produce an output. The main components of the GUI are created at the beginning of the class including the **JFrame** itself, two **Panels** to contain the text areas and buttons, two **JLabels** which allow text to be displayed, two **Buttons** to allow for submitting and resetting the data, two **JTextAreas** for the input and output displays and a **JScrollPane** to allow the input area to be scrollable.

For the graphics I used two imported Java libraries: **java.awt.*** and **javax.swing.***. Importing these libraries allowed me to use helpful methods when creating and setting up my window such as **.setBounds(), .setCursor, .setResizable(), .setLayout(), .setDefaultCloseOperation(), setLocationRelativeTo(),** etc. These methods helped me to quickly get most of the functionality for my interface working. I also added ActionListeners to both of my buttons for something to happen when they are clicked.

I also have a method called 'split' which takes the user input as a **String** and separates it into five String **ArrayLists:** order numbers, addresses, minutes waiting, north co-ordinates and west co-ordinates. I split the input string into a substring before the first comma (*addition*), and the leftover string after the first comma (*leftover*). Then I use an int variable *count* to decide which ArrayList will the *addition* String be added to. After *addition* has been added to an ArrayList, *count* is incremented to move onto the next ArrayList and add the corresponding String to it. Then the method is called recursively on the leftover of the String to cut up the rest of the input. I use this so that each piece of information (order number, adddress, minutes waiting, etc.) is added to the correct ArrayList.

Another method I use is called 'getDistance'. It takes in four Strings as its parameters: the latitude and longitude of location 1 and latitude and longitude of location 2. This method returns a double which is the distance between two co-ordinates in kilometres.

The most important method is my sorting method. It plans out the final route using the 'Nearest Neighbour' algorithm by taking in the ArrayLists of north and west co-ordinates and then calling the 'nearestPoint' method to find the closest point to Apache Maynooth. I decided to combine the north and west co-ordinates into a single 2D array since it was easier to access points that way.

In my 'nearestPoint' method I take in the starting co-ordinates as a 1D array, the north and west co-ordinates as a 2D array and the order numbers as an ArrayList. First, I find the closest house from the starting co-ordinates of Apache Maynooth. Then I add this point to the final route ArrayList, 'route' and call the 'otherPoints' method to do the same job essentially but on the rest of the locations.

Finally, the 'otherPoints' method is a recursive method to get the final route for the rest of the houses. I store the value and index of the most recently visited house by accessing the last element in the 'route' ArrayList. Then using this index, I get the latitude and longitude co-ordinates of the most recently visited house to get the distance from it to the next house. I keep adding the house with the shortest distance to the one before it to the 'route' and eventually when the route has the same number of orders as the order ArrayList initially, it means all houses have been visited and the output route can be printed.

3. Learning Outcomes

Working on this project has been an amazing learning experience for me. I learnt that optimization problems can appear to be simple, but in fact are often very complex and have an enormous number of potential solutions. Solving them can be done with exact methods, but those often require time, and a lot of computing power and especially larger optimization problems will often be too complex to solve. But, applying common sense and some creativity we can build algorithms that require less processing power and can perform very well. In fact, these solutions can even outperform the exact methods if time is a constraint.

Although I wasn't able to use the '2-Opt' algorithm in this project, I still learned a lot about it and how it works. I understood that it is a simple **local search** algorithm for solving the traveling salesman problem. The main idea behind it is to take a route that crosses over itself and reorder it so that it does not. As a developer, it will be my everyday work to solve problems and use algorithms to solve problems very efficiently. I'm glad that I got introduced to some very great algorithms through doing this project.

This was also a great chance for me to learn about using graphics and creating a GUI. I used many resources (linked below) to learn small but useful concepts and code snippets that helped shaped my code overall such as creating a simple window using JFrame, how to use RGB colour values to set JPanel background colour, creating read-only (non-editable) JTextFields, creating a JTextField or JTextArea, basic drawing, etc.

4. Appendix

```
import java.awt.*; // AWT = Abstract Window Toolkit
import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.*;


public class tsp
{
    private static final int width = 850;
    private static final int height = 540;

    public static void main(String[] args)
    {
        new Window(width, height);
    }
}


// Create a simple GUI window
class Window
{
    // ArrayLists for house details
    static ArrayList<String> order = new ArrayList<>();
    static ArrayList<String> address = new ArrayList<>();
    static ArrayList<String> minutes = new ArrayList<>();
    static ArrayList<String> north = new ArrayList<>();
    static ArrayList<String> west = new ArrayList<>();
    static ArrayList<String> route = new ArrayList<>(); // An ArrayList of the final route
```

```
    // Main components of the window
    JFrame frame;
    Panel rightPanel, bottomPanel;


    // Inner components that are used in the window
    Font font;
    JLabel info, input;
    Button btnSubmit, btnReset;
    static JTextArea tfInput, tfResult;
    JScrollPane scrollPane;
    static int count = 0;


    // Default constructor
    Window() {}


    // Create and display a window
    Window(int width, int height)
    {
        // Create and set up the window
        frame = new JFrame("Maynooth's Apache routey");
        frame.setBounds(0, 0, width, height);
        frame.setCursor(new Cursor(Cursor.HAND_CURSOR));
        frame.setResizable(false);
        frame.setLayout(null);
        // When the window is closed, the application also stops
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);


        // Display the window in the center of the screen
        frame.setLocationRelativeTo(null);


        createRightPanel();
        createBottomPanel();

        frame.add(rightPanel);
        frame.add(bottomPanel);

        // Set the window's size automatically by looking at what the JFrame contains
        // frame.pack();
        frame.setVisible(true);
    }


    private void createRightPanel()
    {
        rightPanel = new Panel();
```

```
        rightPanel.setBounds(650, 0, 200, 600);
        rightPanel.setBackground(new Color(114, 211, 237));
        rightPanel.setLayout(null);


        final String message = "<html> Order numbers in order of routey: </html>";


        info = new JLabel(message);
        info.setBounds(10, 0, 175, 50);
        info.setFont(new Font("SansSerif", Font.BOLD, 14));
        info.setFocusable(false);


        tfResult = new JTextArea();
        tfResult.setBounds(10, 45, 165, 265);
        tfResult.setFont(new Font("SansSerif", Font.BOLD, 15));
        tfResult.setEditable(false);
        tfResult.setLineWrap(true);
        tfResult.setWrapStyleWord(true);


        btnReset = new Button("Reset");
        btnReset.setBounds(10, 320, 165, 80);
        btnReset.setBackground(Color.LIGHT_GRAY);
        btnReset.setFocusable(true);
        btnReset.setFont(new Font("SansSerif", Font.BOLD, 24));


        btnReset.addActionListener(new ActionListener() {
            // When the "submit" button is clicked, this method is run
            @Override
            public void actionPerformed(ActionEvent e)
            {
                tfInput.setText("");
                tfResult.setText("");
                System.out.println("Text field reset");
                order.removeAll(order);
                address.removeAll(address);
                minutes.removeAll(minutes);
                north.removeAll(north);
                west.removeAll(west);
            }
        });


        btnSubmit = new Button("Submit");
        btnSubmit.setBounds(10, 410, 165, 80);
        btnSubmit.setBackground(Color.LIGHT_GRAY);
        btnSubmit.setFont(new Font("SansSerif", Font.BOLD, 24));
```

```
        btnSubmit.addActionListener(new ActionListener() {
            // This method gets called when the "submit" button is clicked
            @Override
            public void actionPerformed(ActionEvent e)
            {
                // Include Apache's co-ordinates
                order.add("0");
                address.add("Maynooth Apache Pizza");
                minutes.add("0");
                north.add("53.38197");
                west.add("-6.59274");


                System.out.println("Input submitted");
                String text = tfInput.getText();
                split(text);
                System.out.println(" ");


                sort(north, west);
            }
        });



        // Creates the right panel and adds all the above inner components
        rightPanel.add(info);
        rightPanel.add(btnSubmit);
        rightPanel.add(btnReset);
        rightPanel.add(tfResult);


        info.setVisible(true);
        tfResult.setVisible(true);
        btnReset.setVisible(true);
        btnSubmit.setVisible(true);
        rightPanel.setVisible(true);
    }


    // Create panel to display and take in user input
    private void createBottomPanel()
    {
        bottomPanel = new Panel();
        bottomPanel.setBounds(0, 0, 850, 540);
        bottomPanel.setBackground(new Color(114, 211, 237));
        bottomPanel.setLayout(null);
```

```
        final String message = "<html> Input the order number, house address, minutes
waiting so far & the co-ordinates: </html>";


        input = new JLabel(message);
        input.setBounds(10, -45, 600, 120);
        input.setFont(new Font("SansSerif", Font.BOLD, 14));
        input.setFocusable(false);


        tfInput = new JTextArea(20, 20);
        tfInput.setBounds(10, 45, 630, 445);
        tfInput.setLineWrap(true);
        tfInput.setWrapStyleWord(true);
        tfInput.setFont(new Font("SansSerif", Font.BOLD, 15));
        scrollPane = new JScrollPane(tfInput);
        scrollPane.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
        scrollPane.setSize(650, 450);
        scrollPane.setBounds(10, 45, 640, 445);


        // Creating the bottom panel and adding components to it
        bottomPanel.add(input);
        bottomPanel.add(scrollPane);


        input.setVisible(true);
        tfInput.setVisible(true);
        bottomPanel.setVisible(true);
    }


    // Method to split input String into separate ArrayLists
    public void split(String input)
    {
        String text = input.replaceAll("\n", ",");
        int n = text.indexOf(",");
        String addition = text.substring(0, n);
        String leftover = text.substring(n+1);


        // If count is 0, add substring until comma to ArrayList 'order'
        if (count == 0)
        {
            order.add(addition);
            count++;
        }
```

```
        // If count is 1, add substring until comma to ArrayList 'address'
        else if (count == 1)
        {
            address.add(addition);
            count++;
        }


        // If count is 2, add substring until comma to ArrayList 'minutes'
        else if (count == 2)
        {
            minutes.add(addition);
            count++;
        }


        // If count is 3, add substring until comma to ArrayList 'north'
        else if (count == 3)
        {
            north.add(addition);
            count++;
        }


        // If count is 4, add substring until comma to ArrayList 'west'
        else if (count == 4)
        {
            west.add(addition);
            count = 0;
        }

        try { split(leftover); } catch (StringIndexOutOfBoundsException e) {
west.add(leftover); return; }
    }


    // Getting the distance between two co-ordinates in km
    public double getDistance (String latitude1, String longitude1, String latitude2,
String longitude2)
    {
        /* Point A */
        double lat1 = Double.parseDouble(latitude1); // Latitude of location 1
        double long1 = Double.parseDouble(longitude1); // Longitude of location 1


        // Convert the values of latitude & longitude from degrees to radians
        lat1 = Math.toRadians(lat1);
        long1 = Math.toRadians(long1);
```

```
        /* Point B */
        double lat2 = Double.parseDouble(latitude2); // Latitude of location 2
        double long2 = Double.parseDouble(longitude2); // Longitude of location 2


        // Convert the values of latitude & longitude to radians
        lat2 = Math.toRadians(lat2);
        long2 = Math.toRadians(long2);


        /* Haversine Formula */
        double dlong = long2 - long1;
        double dlat = lat2 - lat1;


        double distance = Math.pow(Math.sin(dlat / 2), 2)
                        + Math.cos(lat1) * Math.cos(lat2)
                        * Math.pow(Math.sin(dlong / 2), 2);


        double result = 2 * Math.asin(Math.sqrt(distance));


        double radius = 6371; // Radius of Earth in kilometers


        // Distance in kilometers from location 1 to location 2 rounded to 2 decimal points
        double output = Math.round((result * radius) * 100.0) / 100.0;
        return output;
    }


    // Method to sort the order numbers using 'Nearest Neighbour' algorithm
    public void sort (ArrayList<String> north, ArrayList<String> west)
    {
        double[] start = {53.38197, -6.59274};
        int size = north.size(); // +1 because of order 0
        boolean northOrWest = true;

        // Combining north and west co-ordinates into one array
        double[][] points = new double[size][2];
        points[0][0] = 53.38197;
        points[0][1] = -6.59274;


        // Filling up 'points' array
        for (int i = 1; i < points.length; i++) {
            for (int j = 0; j < points[i].length; j++) {
                if (northOrWest) {
                    points[i][j] = Double.parseDouble(north.get(i));
```

```
                    northOrWest = false;
                }
                else if (!northOrWest) {
                    points[i][j] = Double.parseDouble(west.get(i));
                    northOrWest = true;
                }
            }
        }


        // Getting the first delivery point
        nearestPoint(start, points, order);
    }


    public void nearestPoint (double[] start, double[][] points, ArrayList<String> order)
    {
        final int x = 0;
        final int y = 1;


        int pos = 0;

        double[] closestPoint = points[1];


        double closestDist = getDistance(String.valueOf(start[x]),
String.valueOf(start[y]), String.valueOf(closestPoint[x]),
String.valueOf(closestPoint[y]));


        /* Traverse the array */
        for (int i = 0; i < points.length; i++)
        {

            double dist = getDistance(String.valueOf(start[x]), String.valueOf(start[y]),
String.valueOf(points[i][x]), String.valueOf(points[i][y]));


            if (dist < closestDist && dist != 0.0) {
                closestDist = dist;
                closestPoint = points[i];
                pos = i;
            }
        }
        route.add(order.get(pos));


        otherPoints();
```

```
        System.out.println("Final Route: " + Arrays.deepToString(route.toArray()));
    }


    // Getting the route for the rest of the deliveries
    public void otherPoints ()
    {
        /* Getting the most recently visited house */
        String recent = route.get(route.size()-1);
        int index = order.indexOf(recent);


        /* Getting the starting co-ordinates */
        String latitude1 = north.get(index);
        String longitude1 = west.get(index);


        double closestDist = 9999.0;


        for (int i = 0; i < order.size(); i++)
        {
            // If the route already contains that order number, skip it (it has been
visited)
            if (route.contains(order.get(i)))
            {

            } else {
                /* Getting the value and index of next house to visit */
                String next = order.get(i);
                int nextInd = order.indexOf(next);


                String latitude2 = north.get(nextInd);
                String longitude2 = west.get(nextInd);


                double dist = getDistance(latitude1, longitude1, latitude2, longitude2);


                if (dist < closestDist)
                {
                    closestDist = dist;
                    route.add(order.get(i));
                }
            }
        }
```

```
        if (route.size() == order.size())
        {
            String result1 = Arrays.deepToString(route.toArray()).replaceAll("\\s","");
            String result2 = result1.replaceFirst("0,","");
            tfResult.setText(result2.substring(1,result2.length()-1));
            return;
        } else {
            otherPoints();
        }
    }
}
```

5. ***References***

- *Create a Simple Window Using JFrame - https://www.thoughtco.com/create-a-simple-window-using-jframe-2034069*
- *How to use rgb color values to set JPanel background color - https://coderanch.com/t/567959/java/rgb-color-values-set-JPanel*
- *Create read-only (non-editable) JTextField - https://examples.javacodegeeks.com/desktop-java/swing/jtextfield/create-read-only-non-editable-jtextfield/*
- *Creating a JTextField or JTextArea - https://mathbits.com/JavaBitsNotebook/GUIs/TextField.html*
- *Conversion of Latitude/Longitude into Image Co-ordinates - https://stackoverflow.com/questions/24874693/conversion-of-latitude-longitude-into-image-coordinates-pixel-coordinates-on-a*
- *Basic Drawing - https://zetcode.com/gfx/java2d/basicdrawing/*
- *How to get the index of second comma in a string - https://stackoverflow.com/questions/22669044/how-to-get-the-index-of-second-comma-in-a-string*
- *Rounding to 2 decimal places - https://intellipaat.com/community/35143/how-to-round-up-to-2-decimal-places-in-java*
- *Program for distance between two points on earth - https://www.geeksforgeeks.org/program-distance-two-points-earth/*
- *Finding the Nearest 2D Point in Java - https://www.youtube.com/watch?v=kK3V6GusRnA&ab_channel=AdamGaweda*
- *Remove Character From String - https://www.journaldev.com/18361/java-remove-character-string*