**Algorithm Term Project**

Team K

텍스트, 로고, 상징, 등록 상표이(가) 표시된 사진

자동 생성된 설명

|  |  |
| --- | --- |
| 과목명 | 알고리즘 |
| 지도교수 | 이상웅 교수님 |
| 학과 | 인공지능전공 |
| 조원 | 202135748 김준우  202334528 이희승  202334454 남예원  202135837 정호정 |

가천대학교

Contents

**1. Prologue** --------------------------------------------------------------

**2. Program Structure** ---------------------------------------------------

2.1 TSP Algorithm

2.2 Program Function

**3. Operation mechanism**------------------------------------------------

**1. Prologue**

This project is a web system that searches for optimal delivery routes based on origin and multiple destinations. It calculates efficient routes using the TSP algorithm, which goes from origin to destination and back to origin.

**2. Program Structure**

2.1 TSP Algorithm

You can refer to the figure above to understand the Traveling Salesperson Problem (TSP) problem. This figure shows all the five nodes and all the paths between the nodes, and each node represents the point to visit. TSP's goal is to find the shortest path starting from the starting node (No. 1), visiting all the nodes once, and then returning to the starting node again. To resolve the problem, we explore all possible paths, calculate the distance of each path, and select the path with the shortest distance. However, the number of possible paths increases exponentially with the number of points, and if there are n points (n-1)!, you need to explore four paths. For example, to visit all five nodes in the figure above, you need to navigate 4! = 24 paths..

**2.2 Program Functions**

The project is structured with several key functions, each serving a specific role in implementing the features of the delivery optimization system. Below is an overview of the primary functions used in the project, written in Python, integrating with the Flask server and Kakao API for data processing and visualization.

**1. is\_in\_seoul**

This function determines whether a given coordinate is within the 15km radius of Seoul's center using the Haversine formula to calculate the distance.

**2. classify\_points**

It categorizes input points as being either inside or outside Seoul's radius and appends the result as an attribute to each point.

**3. haversine**

The function calculates the great-circle distance between two coordinates using the Haversine formula, returning the result in kilometers.

**4. calculate\_speed\_and\_adjusted\_distance**

Based on whether the points are inside or outside Seoul and the distance between them, this function computes the travel speed and adjusts the distance for realistic estimations.

**5. calculate\_time\_and\_distance**

It calculates the travel time in minutes and adjusted distance between two points using the Haversine formula and speed adjustments.

**6. calculate\_weight**

This function computes the weight between two points by combining 60% of the distance and 40% of the travel time to aid in finding the optimal route.

**7. create\_weighted\_distance\_matrix**

It generates a matrix containing the weights and detailed distance/time information for all point-to-point connections.

**8. calculate\_segment\_times**

The function calculates travel times for each segment of the optimal route, formatting the output in minutes or hours as needed.

**9. solve\_tsp**

This function implements a brute-force solution to the Traveling Salesperson Problem, exploring all possible routes and selecting the one with the lowest weight.

**10. get\_ordered\_coordinates**

It organizes the points in the order determined by the optimal route computed by the TSP algorithm.

**3. Operation mechanism**

The system calculates the optimal path based on user input and visually displays it. The main components consist of the user, web, Kakao API, server, and algorithm.

When the user enters the origin and destination through the web interface, the web module sends this data to the server in JSON format. During this process, the Kakao Maps API supports location search and map loading.

The server receives the user input and forwards it to the algorithm module, which returns the optimal path data by considering distance, weight, and time. The server communicates with the Kakao Mobility API to request route information to be displayed on the map.

The Flask server runs on localhost:5000 and serves as the central hub for this process. The optimized route data is sent back to the web module and displayed visually, enabling users to easily understand the results.

<<How to execute it ?>>

1. Execution server.py

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

2. “Open with live server” for Rocket Delivery.html

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명

🡪 This approach enables the desired functionality to retrieve data from the server through the algorithm file and display it on the web page.

<<mechanism>>

1) Point of origin, destination settings, selected locations are displayed on the map

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

2) From Rocket Delivery.html, get the coordinates for the origin and destination through the kakao.maps.services.Places() object and hand it over to server.py .

텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

Send to server

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

3) server.py requests calculation of optimal routes through algorithms.py

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

4) Calculate optimal paths, distance, time at algorithms.py







5) The server sends a POST request to the Kakao Mobility API to display the route on the map.

지도, 텍스트, 아틀라스이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명6) On the Web, the results from the server are displayed with each button.