Travel Recommendation System Project

Ali Ehab Mohamed Abdelrazek

320210045

# Interface

Graphical user interface, application

Description automatically generated

1. **User Input:** Here is the input for the User Code, which is retrieved from the dataset.

**2.** **Budget:** Here is the user’s available budget, from which flight prices are deducted.

3. **Number of Cities to be Visited:** Here is the user’s input of the desired maximum number of city visits.

4. **Starting City:** The city the user will begin their travel. **This is an optional field.**

5. **Begin button:** Initiates the recommendation program and views the output.

6. **Connect Button:** Connects the program to the available dataset. **This must be performed before anything else.** Allows the user to access Starting City options.

7, **Output:** The output resulted from the Begin button shows up here.

# Analysis

## **CSVtoDataTable Function**

This function converts the given dataset, provided in the form of a .CSV file, to a DataTable which can be used in the program. The **Time Complexity** of this function is O(n + m), where n is the number of columns and m the number of rows.

## **Connect\_Click Function**

This is the first function that is called, and is called by clicking the **Connect Button**. This is where CSVtoDataTable() is also called to convert the dataset to a DataTable. This function then creates a new column “priceT” of data type decimal and copies all entries from the “price” column to it, but converted to decimal (since values from a DataTable cannot be changed once they are entered). Then, it converts the first instance of each entry from the “from” column to a string and adds it to a list cities. It has a **Time Complexity** of O(n + m), where n is the number of rows and m is the number of cities.

## **begin\_Click Function**

This function is called upon clicking of the **Begin Button**. First it clears any previous outputs. It checks if the **Starting City** dropdown list has a selected value or not. If not, it defaults to “Recife (PE)”. It proceeds to call the function findNext. Everything apart from the findNext function is of Time Complexity O(1).

## **findNext Function**

This is the bulk and main function of the program. First it creates a list of unvisited places, which is a copy of cities only the starting city is removed from it. It creates a for loop for each city to be visited. This function uses a **Greedy Algorithm** to pick the cheapest option, such that it does not exist in previously visited cities and exists in unvisited cities. The price of the city is deducted from the budget, as long as the budget is greater than or equal to it. If the price is more than the budget, it immediately breaks out of the for loop. Then it prints each of the visited cities and the remaining budget. The **Time Complexity** is O(n\*m\*log(m)), where n is the number of cities to be visited, m the number of rows in the DataTable. As such, this does not always create an optimal solution, as the output may not be equal to the number of desired visits. I was unable to implement this function using Dynamic Programming.

# Time Complexity for each Function

|  |  |
| --- | --- |
| Function | Time Complexity |
| CSVtoDataTable | O(n+m) |
| Connect\_Click | O(n+m) |
| Begin\_Click | O(n\*mlogm) |
| findNext | O(n\*mlogm) |

# Sample Cases

## Sample Case 1

Graphical user interface, text, application

Description automatically generated

## Sample Case 2

Graphical user interface, application

Description automatically generated

## Sample Case 3

Graphical user interface, application

Description automatically generated

## Sample Case 4

Graphical user interface, application

Description automatically generated

**Thank you for your time.**