# Passenger Clearance Order System

Yuhan Xu

**Summary**

In this part of the content, I used Greedy Algorithm to extend one of the homework part of the previous Dijkstra algorithm. Here, I let the program use the Greedy Algorithm to go to the airport where the passenger needs to go. The program will calculate the order of passenger release based on the airport information of the online database provided in the code. The links of the database are as follows: https://raw.githubusercontent.com/gtarik/airports\_iata\_rices/Airports\_Iata\_Codes.csv

**Input Example**

My input is a csv file, my code will be read it and run the program.

passenger\_infomation.csv:

passenger\_id,airport,city,state,country,airliner

A,LAX,Los Angeles,CA,US,AAL

B,ORD,Chicago,IL,US,DAL

C,KFK,New York,NY,US,AAL

D,HKG,Hong Kong,Hong Kong,China,CPA

E,LHR,London,London,UK,BAW

F,SYD,Sydney,Sydney,Australia,AAL

G,HNL,Honolulu,HI,US,UAL

H,ANC,Anchorage,AK,US,DAL

**Output Example**

Expected output:

C KFK New York NY US AAL

E LHR London London UK BAW

B ORD Chicago IL US DAL

F SYD Sydney Sydney Australia AAL

A LAX Los Angeles CA US AAL

H ANC Anchorage AK US DAL

G HNL Honolulu HI US UAL

D HKG Hong Kong Hong Kong China CPA

**Business Requirement**

### Input Requirements

1. The path to a CSV file containing passenger information should be provided as input to the script.

2. The CSV file provided by airport should have a header row and the following fields for each passenger: passenger\_id, airport, city, state, country, and airliner.

### Output Requirements

1. The script should output a list of passengers sorted based on the distance between their departure cities, in ascending order.

2. For each passenger, the script should output the following information: passenger\_id, airport, city, state, country, and airliner.

3. The output should be printed to the console.

**Output after Code running**

Please input the path of your csv file：/home/xuyuhan/Desktop/BU-learn-S2023/METCS566O1/HW6/pythonProject/passenger\_information.csv

C KFK New York NY US AAL

E LHR London London UK BAW

B ORD Chicago IL US DAL

F SYD Sydney Sydney Australia AAL

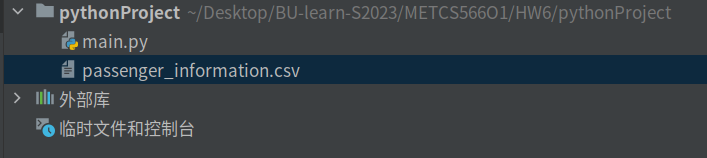
A LAX Los Angeles CA US AAL

H ANC Anchorage AK US DAL

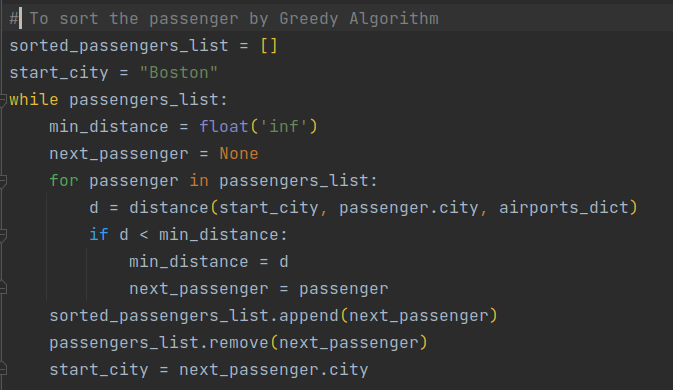
G HNL Honolulu HI US UAL

D HKG Hong Kong Hong Kong China CPA

**Screenshot of the file structure**

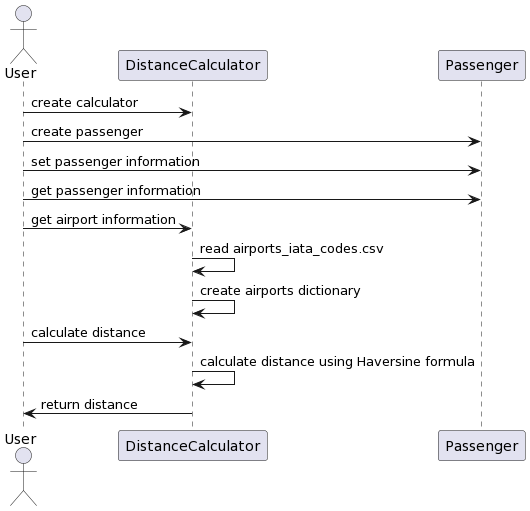


**Code for Greedy Algorithm**



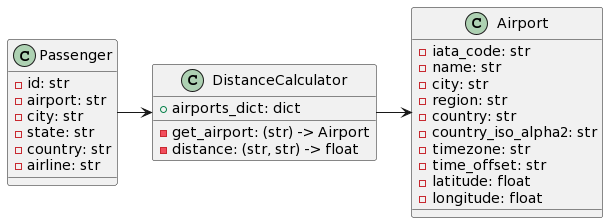
This code applies a greedy algorithm in the process of sorting the list of passengers based on their cities. The algorithm selects the passenger with the shortest distance from the current city (starting with "Boston")

**Sequence Model**

****

Sequence model

**Class Model**



class model

**Time Efficiency**

The algorithm reads a CSV file containing passenger information and then sorts the passengers by the distance between their starting city and the previous passenger's destination city, using the Greedy algorithm. The distances are calculated using the latitude and longitude of the airports in the cities, which are obtained from another CSV file. The algorithm has a time complexity of O(n^2), where n is the number of passengers. This is because for each passenger, the algorithm loops through the remaining passengers to find the one with the shortest distance to the previous passenger.

If the input data count increases, the algorithm's efficiency will decrease, as the number of iterations required to sort the passengers will increase. In practical terms, this means that the algorithm may take longer to run, which could be problematic if the application needs to process a large number of passengers in real-time. To improve the efficiency of the algorithm, one could consider using a different sorting algorithm that has a lower time complexity, such as merge sort or quick sort. Additionally, precomputing the distances between all pairs of cities and storing them in a lookup table could also improve the algorithm's efficiency.

**Comletely Code**

