Route Search Algorithm Using Uniform Cost Search

Thomas Kojo Quarshie

Student ID: 43102024

CS313_A: Project 2 (C++)

Department of Computer Science and Information Systems, Ashesi University

Dr. Robert Sowah & Dr. David Ebo Adjepon-Yamoah

November 29, 2022.

I started off by creating a class with its corresponding header file for each CSV file (i.e., Airport, Airline, and Routes) with the column in each file as its member variables. I then created a "Readfile" class that takes the name of a CSV file as a parameter and splits each row using the getline method and the newline character as a delimiter. The splatted rows were further split and stored in a vector by constructing a method that splits a line using a comma as a delimiter. The overall output in a vector made it easier to access a specific row in a file by an index. Moreover, each class I created also had getter methods which made it convenient to access specific data in the vector, e.g., getAirportcode(). However, I realized that the existence of extra commas in some records influenced their columns to increase more than the number of columns in a table. This problem persisted in the Airport file, which I decided to solve by writing a method that returns all Airport objects with sizes or lengths greater than 14. With this, I could identify where the extra commas were and constructed a constraint that combined two or more columns into one using the Airport ID.

Nonetheless, I constructed a "Route Distance" method from the Haversine formula referenced from GeeksForGeeks that computed the distance between a "SourceAirportcode" and a "Destination Airportcode." This was relevant because I applied this method in my Uniform cost Search algorithm to find the shortest path cost from one source airport code to its destination code. My search algorithm also utilized an unordered map data structure that efficiently tracks the source airport code as the key and the values as a vector of a vector containing the destination airport code. The search algorithm returns the optimal path by comparing the new path cost found with the previous. It returns a solution if the new path cost is less than the previous and is also the goal node. With this project, I have researched intensively into the data structures and the syntax of C++. I had to research more on vectors, unordered maps, deques, and the efficient way to utilize pointers. I faced many challenges, including overriding the equals method, casting a data type to another using "stoi" or "stod," extending the search to other destination airport codes, and the goal test method. I have been stressed out in this project because of the extensive research I had to do, but I consider it worthwhile.