Spring 2023 DS 210 Final Project — Jit Ping Lee

Dataset Picked

I picked the April 2023 Citibikes User Data. Citibikes is a bike sharing platform in New York City where users take a bike from a "station" and return it to another "station". This is similar to Bluebikes in Boston.

The user data comes in a csv file and has one row for each trip taken for the month of April 2023. Some metrics calculated include the start and end location of the trip and the start and end time of the trip.

Link To Datasets: https://s3.amazonaws.com/tripdata/index.html (Filename: 202304-citibike-tripdata.csv.zip)

This program works for any month after 2021 because they were using a different format before that.

Research Question

I wanted to find out the shortest travelling time between any one station in the Citibike network to all other Citibike stations.

Even if no ride has been taken before between those two stations, I will use the shortest path algorithms to find the shortest travelling time between the stations.

Initial Analysis of Dataset

Based on a counter I printed out in my Rust file, I know that I processed minimally 2.8 million Citibikes rides (some were not processed if for example, the start and end stations are the same).

I also know that there were 1863 unique Citibikes stations. These were my nodes.

As I was using a weighted directed graph, the weights for each edge was the average travelling time from one station to another. The 2.8 million rides got condensed into 472,093 rides. These were my edges.

Algorithms Used

I used Dijkstra's shortest path algorithm. This algorithm worked for me as I was looking to find the shortest travelling time from one station to another station whereby a ride might have never been taken before. It is technically possible for the shortest travelling time from one station to another to be an indirect route (e.g. A-B-C is faster than A-C). The algorithm would have suggested the indirect routing as it produces the shortest travelling time.

Spring 2023 DS 210 Final Project — Jit Ping Lee

Input Required And Output Produced

Users need to input a csv file containing the ride data. They also need to input the station name of the start and end stations they want to calculate the shortest paths for.

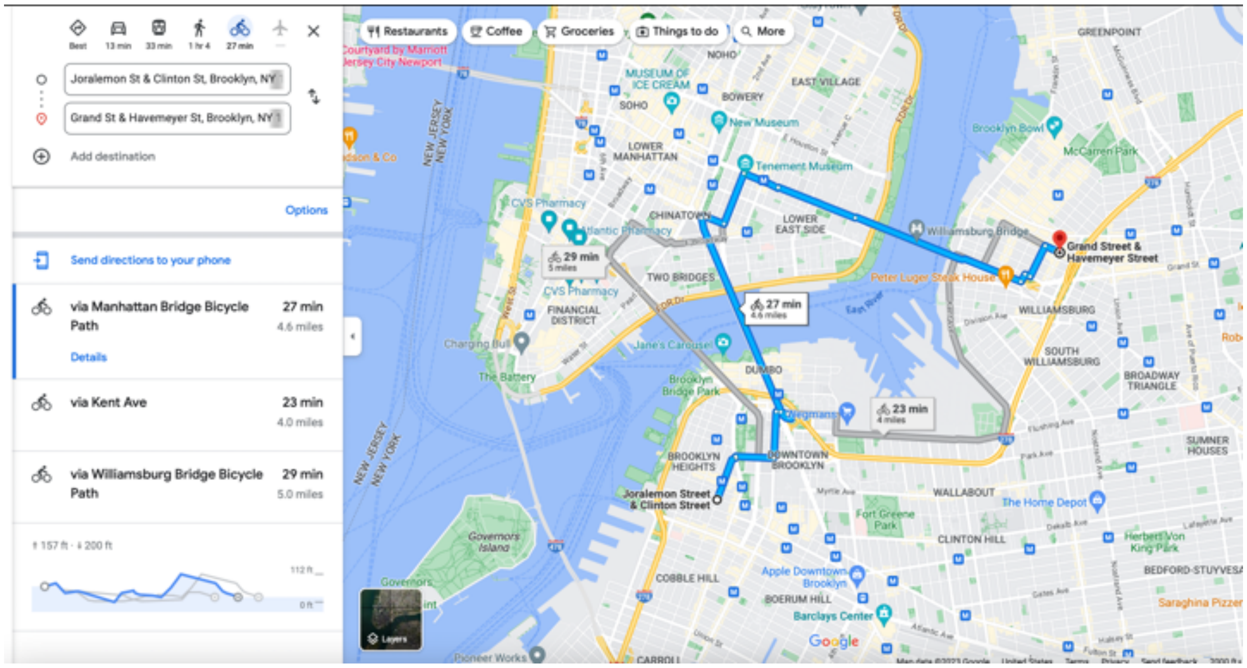
I have two outputs after data processing.

I print out the travelling time between a start and end Citibike station as specified by the user. I debated between letting users enter a station_id I generated for each station or the actual name of the station but chose the station name as the station_id is not intuitive to a user.

I also print out the travelling time from the start Citibike station to all other Citibike stations in a .txt file. The name of the .txt file is different for each Citibike station. This means that previous files will not be overwritten by a new file.

Interesting Comments

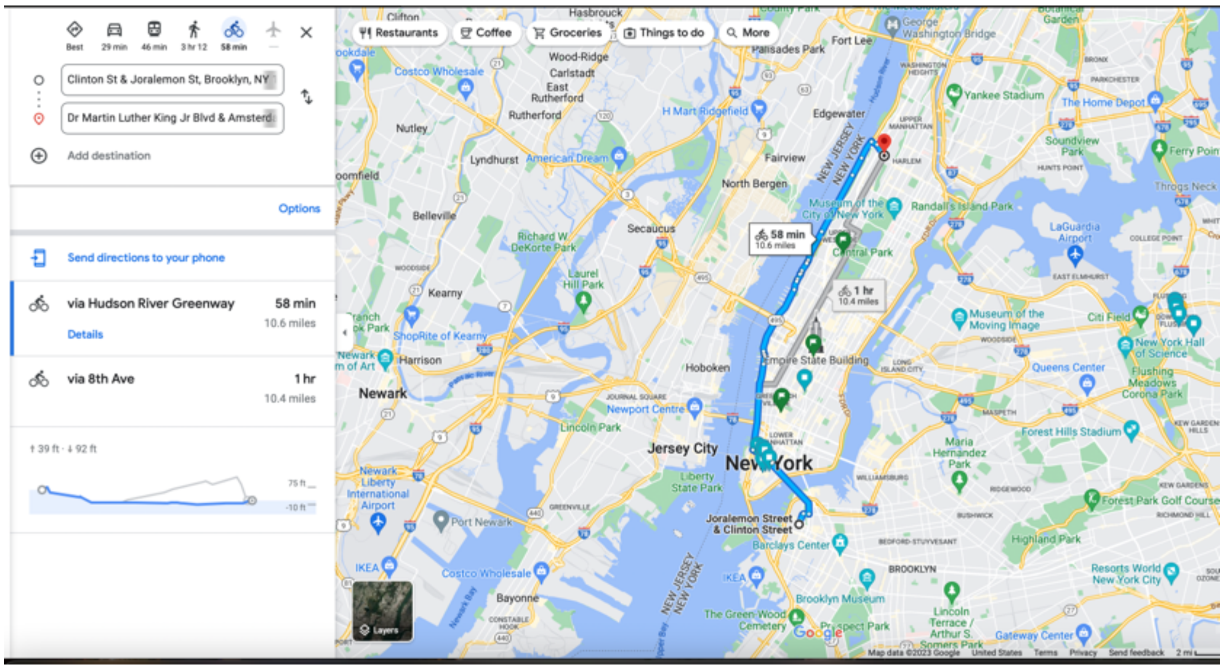
To make sure that my results make sense, I decided to validate my results in Google Maps. For example, my algorithm reports that the shortest travelling time from “Clinton St & Joralemon St” to “Grand St & Havemeyer St” is 20 minutes 23 seconds. This is similar to the 24 minutes offered on Google Maps. The return journey was estimated to be 20 minutes 9 seconds. The fact that the numbers are not wildly off suggests that my algorithm was working.


Joralemon St & Clinton St, Brooklyn, NY
Grand St & Havemeyer St, Brooklyn, NY
Add destination
Options
Send directions to your phone
via Manhattan Bridge Bicycle Path 27 min 4.6 miles
Details
via Kent Ave 23 min 4.0 miles
via Williamsburg Bridge Bicycle Path 29 min 5.0 miles
29 min 5 miles
27 min 4.6 miles
23 min 4 miles
Joralemon Street & Clinton Street
Grand Street & Havemeyer Street
Williamsburg Bridge
LOWER MANHATTAN
CHINATOWN
LOWER EAST SIDE
DUMBO
BROOKLYN HEIGHTS
DOWNTOWN BROOKLYN
WILLIAMSBURG
SOUTH WILLIAMSBURG
GREENPOINT
Governors Island

An interesting question is whether the estimates will be less accurate as the travelling time increases. On one hand, riders get tired when cycling longer distances, a consideration the shortest paths algorithm does not take into account. For example, when cycling a long distance journey from A-C, fatigue will play a bigger role as compared to two different riders cycling A-

B and B-C. On the other hand, riders might have to detour when reaching a further destination connected only by Citibikes stations.

I decided to check the cycling time from “Clinton St & Joralemon St” in Brooklyn to “Amsterdam Ave & W 125 St” in uptown Manhattan. My program predicted a cycling time of 42 minutes and 21 seconds while Google Maps predicts a cycling journey of 58 minutes. It appears that in New York, where there are 1863 Citibike Stations, the concern of riders having to “detour” when travelling between various Citibike stations to get to their destination as part of the Shortest Paths algorithm is minimal. The difference in travelling time is greater when distances are further.



Tests Run

I ran six tests for my project.

My first tests makes sure that the csv file is being processed properly. This includes making sure that a single Citibike station is not being assigned two IDs. I have written code that skips assigning a station ID if a station has been assigned before. I also have a test to ensure that a row is skipped if the starting and ending stations are the same. I also wrote a test to ensure that if a trip duration is calculated to be negative, I discard the trip too.

My second test ensures that “duplicate” edges ie. Multiple trips from one station to another are treated correctly. This means ensuring that they are not producing two similar edges after processing and that the average value is calculated correctly.

My third test ensures that the function I wrote converting seconds to minutes and seconds is working.

My fourth test ensures that my shortest path can find the shortest travelling time between two Citibike stations even if this edge has not existed beforehand.

My fifth test ensures that an error message is produced if an invalid station name is given.

My sixth test ensures that the shortest paths algorithm does not create a new edge if there is no way two Citibike stations are linked to one another.

References

https://doc.rust-lang.org/std/primitive.str.html (for the .trim_matches() function to remove the double quotation marks)

https://docs.rs/chrono/latest/chrono/ (For finding the duration between my start and end time, in particular the parse from string function)