Brandon LaPointe & Param Rajguru

CSC 365 Assignment 3 Demo Substitute Write Up

For assignment 3, we initially started working on the additional parameters of the business json file needed to calculate the distances of the four closest businesses, grabbing the latitude and longitude of each business and making the proper adjustments to the business objects to also hold the newly obtained information. The next step was to make all of the adjustments necessary to the byte buffers in order to properly store all of this new information within the files and to then read the information back, adding new segments to the byte buffer to include two new doubles for the latitude and longitude data of each business.

With this step complete, we moved on to adding the Haversine formula method to calculate the distances between the latitude and longitude points of the businesses in order to gather the four closest businesses of every business and got this information to output onto the console but ran into a problem where many of the businesses displayed null for the four closest neighbors. This ended up being a result of the reordering of the businesses array by distance to the compared business every call, making it so many businesses were never run through the algorithm. The problem was fixed easily once finally identified by adding an extra line before the call to reorganize the array to make a copy of the array and then reorganize the copy instead of the original business array.

With all of the businesses successfully outputting their closest neighbors using the Haversine formula, we moved on to the report of the number of disjoint sets and persistent storage of the HashMap containing the key of each business to values of a list of pairs, each containing the neighboring business objects paired with their distance from the key business. After that we started working on implementing the shortest path using Dijkstra's algorithm and number of disjoint sets using union and find. Both seemed to be working with smaller test inputs but when we integrated them with our main code we were facing some issues with the shortest path method which needed to be debugged before we ran out of time. With the successful output of the disjoint sets and persistent storage of the neighboring businesses working properly utilizing Serializable, we moved onto graphing the output and applying the graph to the GUI and this is where we got caught up before running out of time.

We have a hard coded, separate Graph Display main method GUI implementation working properly to at least display the graph, including nodes and edges, using a simplified adjacency matrix and print the shortest path to another node on the console but started running into issues with implementation of the code within the main application as the graph just failed to display and the shortest path was never output. After scanning through the code, we fixed the issue which involved the retrieval of values for the adjacency matrix and graphing but couldn’t figure out proper formatting of the graph to neatly display the data as it is all clustered towards the upper corners of the GUI (which was because we were using x and y coordinates as similarities and not the coordinates and would have been easy to fix if we weren’t so focused on the main issues that needed to be addressed first) and is impossible to distinguish different nodes in the dense clusters. We also couldn’t fix the display of the shortest path to the cluster center in the main application unless we hardcoded it to find the shortest path to the business being searched which just resulted in a single path of the business itself. With that said, we couldn’t complete the remainder of the assignment involving the GUI interaction to display the selected node and pathing to the nearest reachable cluster center. Our plan was to merge the two GUIs for proper input and outputs and once on successful application of shortest path we would grab the edges and change their color from black to say red on the GUI to display the shortest path using Dijkstra’s algorithm and display another graph with changed red edges depicting the closest path from Business A to B.

Included below are screenshots of the hard code output and the main application output in its present form:

**Initial issues of null neighbor list:**

A screenshot of a computer

Description automatically generated

**Graph test output:**

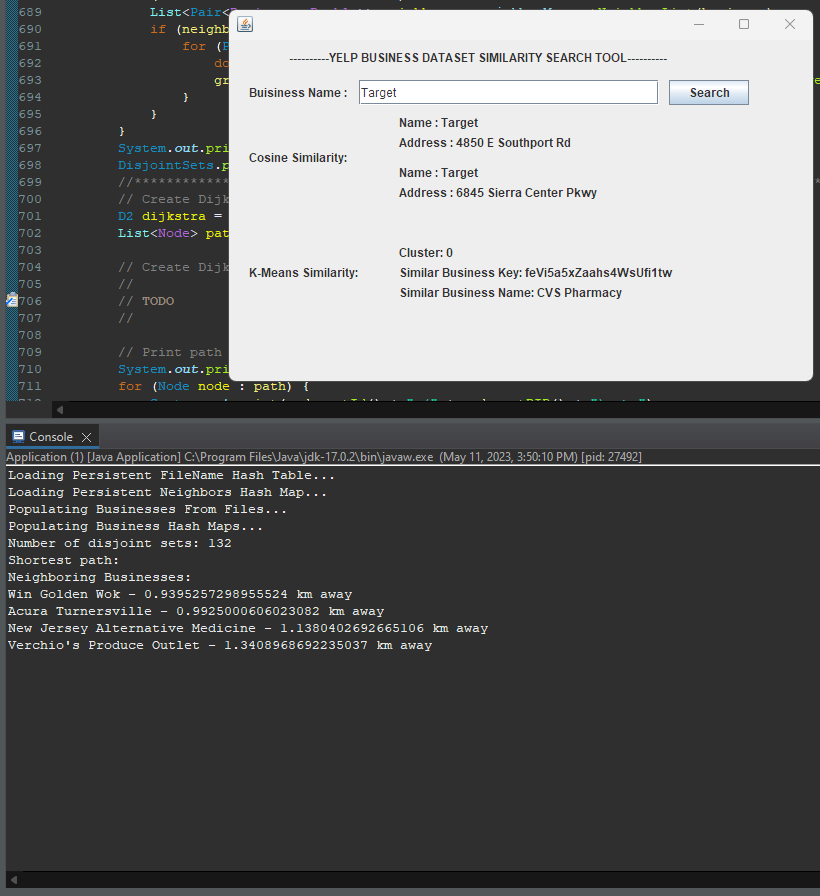
A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence**Application graph output:**

**Console output after business search:**



**Working Dijkstra’s and number of disjoint sets on test outputs:**

A screenshot of a computer

Description automatically generated