Name: Maxwell Aladago ID: 15992018

Report of Implementation of Ashesi Career Fair Programming Competition Entry

All the three tasks described in the instructions of the competition have been tackled. The solutions were implemented in the Java programming language using a Netbeans IDE. The rest of the report is as follows: Approach to solving task1 is described in Section II. Next in Section III and Section IV, approaches to solving task2 and task3 are respectively discussed. The paper is concluded with the references used in understanding and solving the three tasks.

SECTION I: TASK1: VEHICLE PARKING LOCATION AT NIGHT – APPROACH

The primary criterion for solving this task is to *search for the most frequent trip which had the car's engine switched off and the next trip happened a day after*. This approach still works perfectly even if the file sorting order changes. To implement that algorithm, the program starts by opening the *2.vehicle_log.csv* file for reading. In each row of the file, the program checks whether the engine is off. If it's, it goes further to check whether the time was recorded in the night. If both conditions are met, the id and location for that trip is recorded and kept for further processing. After all the rows in the file has been read, the program assess the recorded trip ids and picks out the location of the most frequent id as the cars packing location at night. The latitude and longitude of this location are converted to 2 d.ps and the results written to a file *output-task2.txt* at root of the project.

SECTION II: TASK2: POINTS OF INTEREST – APPROACH

The main criterion for solving this problem is that, *a point is within the 'geofence' of a point of interest if the distance between it and the point of interest is less than or equal to the radius.* However, since the number of entrances are required, the program assumes that, the input file is organized in a way that, in an entrance into a region, the feature recordings cannot be placed arbitrary over the file. That's the file is organized chronologically for each entrance. To solve this, the program opens a file called *1.interest_points.csv.* It then creates *PointsOfInterest (POI)* objects and spawns them in a thread for concurrent execution. For each row of *2.vehicle_log.csv* file, each *POI* reads its latitude and longitude. The distance between the *POI* and the location of the row (in km) is determined using the Harvesine Formula[1]. The Harvesine Formula was used because the Law of Cosines for Spherical Trigonometry does not give accurate results for small distances[2]. However, both formulae attained the same results for the test data, . Since all the latitudes of the points of interest are 5. *, the earth radius is calculated as *radius (in km) = 6378 - 21 * sin (latitude) instead of the average radius of 6371km.* Once a location is detected such that, the distance between it and POI is less than or equal to the Radius given in file, record its time and assess the next n rows until a location outside the range is found. Increment the total time spent within that point by the difference between entrance time and exit time. Also increase the number of entrances by 1. After all rows are considered, the program writes the average time(in minutes) and number of times the vehicle entered into a POI along with the properties of that POI to a file call *output-task2.txt*.

SECTION III: DETECTION OF UNUSUAL TRIPS – APPROACH

A trip is considered unusual in this implementation if it starts at a very odd time, or it has an odd maximum heading, or an odd minimum heading, or an odd latitude mean or an odd longitude mean. These features of a trip are evaluated after all rows in the 2.vehicle log file have been sorted and grouped into distinct trips. In this implementation, a trip category is considered 'odd' *if the size of the category is less than 1 less the square root of the number of trips under consideration/the number of distinct groups.* Groups of trips for outliers detection is computed as follows:

1. All trips whose average speed is less than 6km/hr or which lasted less than 10 minutes (600 seconds) or which covered less than 500 meters are excluded from the analysis.
2. All trips which started within the same hour are put into a distinct category
3. For both maximum headings and minimum headings, trips whose values differ by only the last significant digit are put into the same category
4. All trips whose distance are in a range of 100 meters (0.1 km) are put into one category.
5. For both latitude mean and longitude mean, trips which has the same values to 2.dp are put into the category.

Outliers returned from all the above categorization are returned as a set whose contents is then written to a file.
Note: All formulations or approaches describe in this section are not proven. I crafted them for this problem.

References

[1] N.A. Straight-Line Distance calculation. 2008.
https://engineering.purdue.edu/ece477/Archive/2008/Spring/S08-Grp03/nb/datasheets/haversine.pdf
[2]Rick, D. Longitude and Latitude Calculations. 1999. http://mathforum.org/library/drmath/view/51879.html