Centre for infrastructure, Sustainable Transportation and Urban Planning

Indian Institute of Science (IISc), Bengaluru
Summer Internship: Round 1

The aim of this exam is to test your problem-solving skills and basic understanding of Python. All questions are not compulsory, and partial solutions will also be considered while shortlisting for subsequent rounds. Hence, you are encouraged to submit the best possible answer for each question. Follow the instructions below precisely.

- Plagiarism will result in instant disqualification. You must write your own code.
- To make your submission, use the following Google form: submission link. You are allowed to make only one submission for this test. While submitting, you will be asked to upload three documents:
  - A python file (format: *.py*) containing all the codes related to Question 1
  - A python file (format: *.py*) containing all the codes related to Question 2
  - A report (format: *pdf*) summarizing your findings from both the questions. The report should include essential components such as clearly stated assumptions, informative visualizations, and your findings. Do not copy-paste code in the report.
- If possible, both the python files should be written according to PEP 8 – style guide (reference).
- Your submissions will be evaluated based on the quality of the report and codes. Hence, aim for better visualizations and efficient codes to increase your chances of selection. In case of a tie, the average runtime will be used as a tiebreaker.
- The test commences on 8^{th} April 2023 (10:00 AM). The last date for submission is 10^{th} April 2023 (10:00 AM). Late submissions will not be accepted.
- This project will be used to shortlist for the following projects (supervised by Dr. Tarun Rambha (link)):
  - Data science for public transit
  - Inverse reinforcement learning for tolling and parking
  - VISSIM modeler for parking
  - Real-time project scheduling assistant app

All the best.
C*i*STUP
IISc, Bengaluru

Question 1

In a bicycle-sharing system, bicycles are stored at fixed docking stations throughout the city. Users can rent bicycles from one docking station and return them to any other docking station. These systems are often used for short trips around the city, providing users with a convenient and eco-friendly mode of transportation. A dataset (link) containing 6,867 bicycle trips over one day is provided. The column descriptions are provided below.

- trip_id: Unique trip identifier.
- started_at: Start time of the trip
- ended_at: End time of the trip
- start_lat/start_lng: Latitude/Longitude of the starting depot
- end_lat/end_lng: Latitude/Longitude of the end depot

1. Write a function that removes all trips of duration 0 minutes and prints the following values on the console. Mention the same values in the report.
   - Maximum duration of the trip (in minutes).
   - Minimum duration of the trip (in minutes).
   - Total number of trips corresponding to the minimum duration.
   - Percentage of total circular trips. A trip is defined as circular if it starts and ends at the same location.
   - Total runtime for the function

   Hint: The question is designed to judge your basic skills in exploratory data analysis.

2. Filter the original dataset to include only the trips starting between 06:00 AM and 06:00 PM. Find the total number of feasible pairs of trips. Two trips, A and B, are defined as a feasible pair if they can be served in succession by the same bicycle, i.e., if the end location of trip A is the same as the start location of trip B and the start time of the trip B is greater than or equal to the end time of the trip A. For example, Trip Id 1733 and 1965 are feasible. In the report, mention the total feasible pairs of trips and runtime.
   Hint: The question is designed to judge your critical and analytical thinking.

3. Filter the original dataset to include only the first 100 trips (i.e., trip_id 1 to 100). In the report, mention the number of unique depots used to serve these trips. Next, find the shortest path distance between all the depots. To do so, do the following steps:
   - Download the underlying graph using the OSMnX module (reference) in Python.
   - Find the nearest node in the graph corresponding to each depot.
   - For every pair of nodes, run a shortest path algorithm (for example, Dijkstra's (reference)) to get the length of the shortest path. If the nodes are not reachable from each other, the function should return -1.

   In the report, mention the total runtime and the maximum and minimum (greater than 0) distance.