Part 2.

Clustering of Trajectories

Question 7:

The usual theorem to estimate the distance between 2 points; which is Pythagoras theorem; cannot be made use here as the latitude and longitude is represented by considering the spherical aspect of Earth. So, the distance should be calculated by considering this spherical aspect of planet.

The best way to find the distance between 2 points represented by latitude and longitude is by making use of **Haversine Distance formula**. It remains well for even small distances.

Reference: https://en.wikipedia.org/wiki/Haversine_formula

The formula is:

$\theta = \frac{d}{r}$ which means $d = \theta * r$

Where:

D is the distance which we want to find and

Θ is the angle between 2 points represented by the latitude and longitude,

r is the radius of Earth

r can be assumed to an average of 6371 km for Earth

Θ is found out using the below equation

$$hav(\theta) = hav(\Phi 2 - \Phi 1) = \cos(\Phi 1)\cos(\Phi 2)hav(\lambda 2 - \lambda 1)$$

Φ1 and Φ2 represent latitude of 2 points 1 and 2 respectively whereas the

Λ1 and Λ2 represent longitude of 2 points 1 and 2 respectively

The hav() function which is called inside "$hav(\lambda 2 - \lambda 1)$", can be found out using below formula:

$$hav(\theta) = sin^2\left(\frac{\theta}{2}\right) = \frac{1 - \cos(\theta)}{2}$$

After finding the haversine angle (hav(Θ)), we have to find the Θ which is the central angle using:

$$\theta = \text{hav}^{-1}(\text{hav}(\theta)) = 2sin^{-1}(\sqrt{hav(\theta)})$$

So, once we have r and Θ, we can find the distance, 'd' using $d = \theta * r$

For example, consider a dataset with a fixed latitude, longitude for a timestamp (consider the first 2 rows from the given dataset for part1.)

Point	UserID	Latitude	Longitude	Altitude	Timestamp	Date
A	100	39.974408918	116.30352210	480.287355	40753.53069	2011-07-29
B	100	39.974397078	116.30352693	480.121151	40753.53070	2011-07-29

Condition	Representation	Result
Distance between A and B	d(A,B)	0.001379 km
Distance between B and A	d(B,A)	0.001379 km
Distance between A and A	d(A,A)	0.000000 km
Distance between B and B	d(B,B)	0. 000000 km

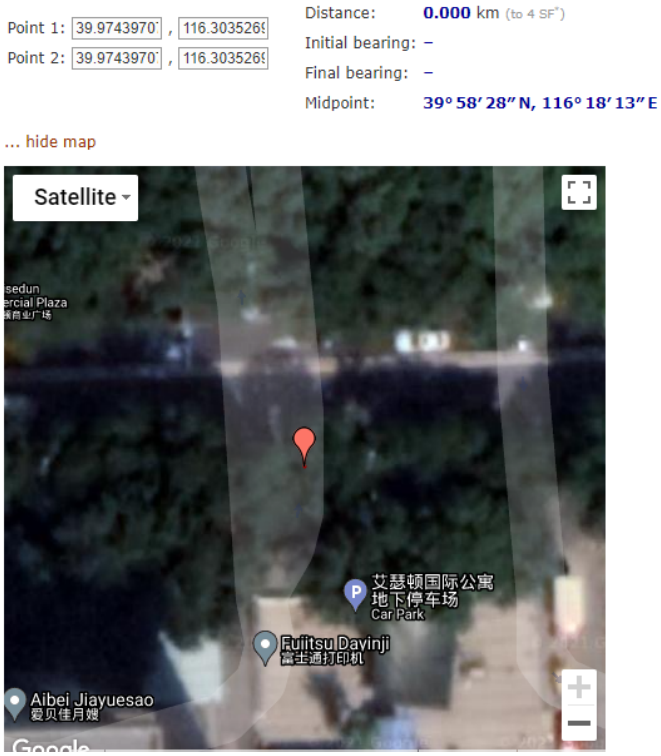
Refer https://www.movable-type.co.uk/scripts/latlong.html for above calculation

This proves that the distance measured is symmetric (i.e., d(x,y) = d(y,x) for any trajectories x,y)

d(A,B) = d(B,A)

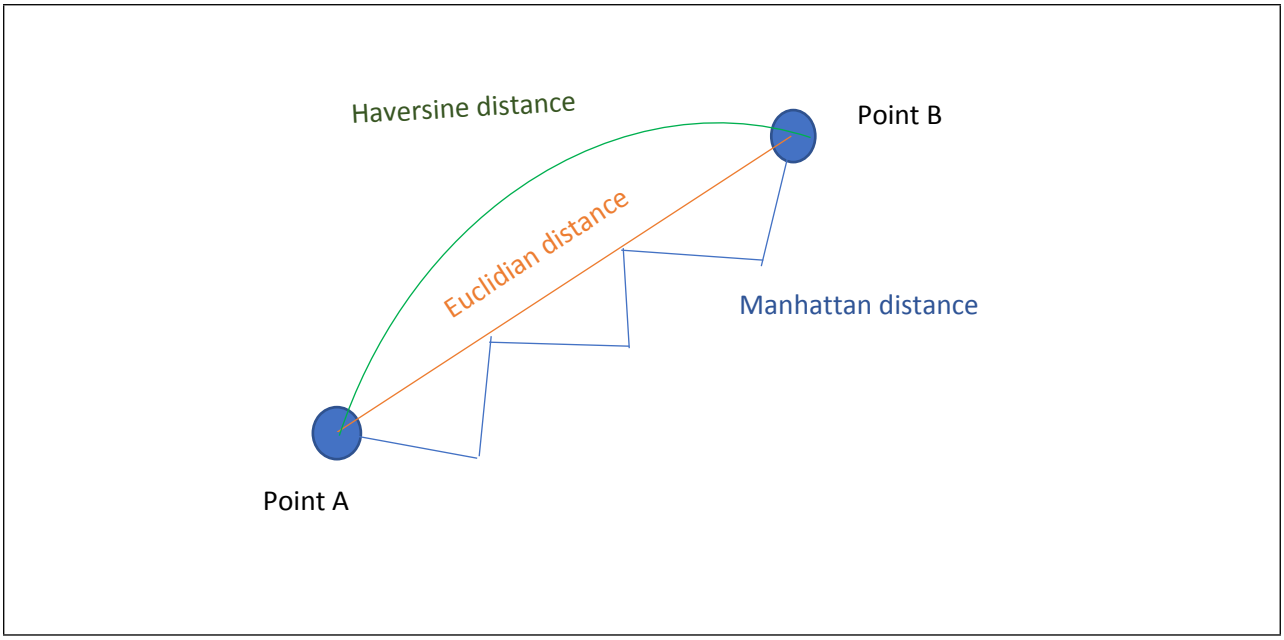
Distance is 0 between two trajectories (i.e., d(x,y) = 0 if and only if x=y)

d(A,A) = d(B,B) = 0



Another measure with which we can calculate the distance between 2 points. It is **Manhattan distance,** also known as the Taxicab distance or the City Block distance. It calculates the distance between two real-valued vectors considering the city is divided into several blocks. This measure do not give the spherical distance.

ManhattanDistance = sum for i to N sum |v1[i] – v2[i]|



Question 8:

For example, consider a dataset with a fixed latitude, longitude for a timestamp (consider the first 2 rows from the given dataset for part1.)

Point	UserID	Latitude	Longitude	Altitude	Timestamp	Date
A	100	39.974408918	116.30352210	480.287355	40753.53069	2011-07-2
B	100	39.974397078	116.30352693	480.121151	40753.53070	2011-07-29
C	100	39.973982524	116.303621837	478.4994553	40753.530729	2011-07-29

Condition	**Representation**	**Result**
Distance between A and B	d(A,B)	0.001379 km
Distance between B and A	d(B,A)	0.001379 km
Distance between A and A	d(A,A)	0.000000 km
Distance between A and C	d(A,C)	0.04817 km
Distance between B and C	d(B,C)	0.04680 km

Refer https://www.movable-type.co.uk/scripts/latlong.html for above calculation

This measure also satisfies the triangle equality (d(x,y) <= d(x,z) + d(z,y) for any trajectories x,y,z).

d(A,C) <= d(A,B) + d(B,C)

0.04680 + 0.001379 = 0.04817 km

Point 1: 39.9744089 , 116.303522
Point 2: 39.9739825 , 116.303621

Distance: **0.04817** km (to 4 SF)
Initial bearing: **169° 50′ 15″**
Final bearing: **169° 50′ 15″**
Midpoint: **39° 58′ 27″ N, 116° 18′ 13″ E**

... hide map

Point 1: 39.9743970 , 116.303526
Point 2: 39.9739825 , 116.303621

Distance: **0.04680** km (to 4 SF)
Initial bearing: **170° 02′ 57″**
Final bearing: **170° 02′ 57″**
Midpoint: **39° 58′ 27″ N, 116° 18′ 13″ E**

... hide map

Question 9:

In order to group users with similar trajectories, the best approach is KNN Algorithm.

K Nearest Neighbour (KNN) algorithm usually stores the similar features of existing data and then when a new data comes, it compares those features of new data and classify it. This is performed in data sets which consists of more than two labels. In our case, when similar trajectories are found, the algorithm classifies them after learning the existing trajectories based on the value of K given. Parameter tuning (selecting the correct value of K) should be done properly for better results.

KNN can be used on data which we already have and hence model will learn how to classify the future data. KNN does not make any explicit assumptions hence avoiding mismodelling errors and it is highly precise compared to other models. Here, for this dataset, KNN algorithm will do the calculation for the users sharing similar trajectories.

Because of the effectiveness and the precision KNN is always preferred. As we have a initial set of data from which various features can be extracted and used for prediction of future, this method can give better results for similar trajectories.