DMDD final project positioning
Zixiao 20200405

Positioning

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
# import the magic code for using sql in jupyter notebook
%load_ext sql
# local database
# %sql mysql+pymysql://root:fjwwzx970814@localhost/mydb
# remote database
```

```
%sql mysql+pymysql://brickea_mac:fjwWZX970814@rm-
0xih4pk94w41k3c5j8o.mysql.rds.aliyuncs.com/mydb
```

```
'Connected: brickea_mac@mydb'
```

```
%sql mysql+pymysql://nity:BravoNity123@rm-
0xih4pk94w41k3c5j8o.mysql.rds.aliyuncs.com/mydb
```

```
'Connected: nity@mydb'
```

```
import sqlalchemy as sqlManager
```

```
# Create connection with database
connection =
sqlManager.create_engine('mysql+pymysql://brickea_mac:fjwWZX970814@rm-
0xih4pk94w41k3c5j8o.mysql.rds.aliyuncs.com/mydb?charset=utf8')
```

```
connection =
sqlManager.create_engine('mysql+pymysql://nity:BravoNity123@rm-
0xih4pk94w41k3c5j8o.mysql.rds.aliyuncs.com/mydb?charset=utf8')
```

Procedures and View test

```
%%sql
# View get_user_and_device
SELECT * FROM get_user_and_device
```

```
 * mysql+pymysql://brickea_mac:***@rm-
0xih4pk94w41k3c5j8o.mysql.rds.aliyuncs.com/mydb
15 rows affected.
```

idcustomer	firstName	lastName	iddevice	deviceType	NumConnectionsAvailable
1	first_name_0	last_name_0	2	Phone	3
1	first_name_0	last_name_0	6	Laptop	0
1	first_name_0	last_name_0	7	Ipad	0
2	first_name_1	last_name_1	3	Phone	4
2	first_name_1	last_name_1	9	Ipad	0
3	first_name_2	last_name_2	15	Phone	0
4	first_name_3	last_name_3	4	Phone	3
4	first_name_3	last_name_3	8	Ipad	0
5	first_name_4	last_name_4	5	Phone	5
6	first_name_5	last_name_5	10	Phone	4
7	first_name_6	last_name_6	11	Phone	4
8	first_name_7	last_name_7	12	Phone	5
9	first_name_8	last_name_8	13	Phone	5
10	first_name_9	last_name_9	14	Phone	4
11	first_name_9	last_name_9	1	Phone	5

```
%%sql
SELECT * FROM deviceBeaconConnection dc
WHERE dc.iddevice = 2
```

```
 * mysql+pymysql://brickea_mac:***@rm-
0xih4pk94w41k3c5j8o.mysql.rds.aliyuncs.com/mydb
```

```
3 rows affected.
```

idconnection	iddevice	idbeacon	distance	beaconRankbyDist
6	2	7	8.062	1
7	2	3	14.318	3
8	2	2	12.083	2

```
%%sql
# Procedure get_current_best_3_connections
CALL get_current_best_3_connections(2,@number,@c1,@c2,@c3);
```

```
 * mysql+pymysql://brickea_mac:***@rm-
0xih4pk94w41k3c5j8o.mysql.rds.aliyuncs.com/mydb
1 rows affected.
```

num_available_connections	connection_1	connection_2	connection_3
3	6	8	7

```
%%sql
SELECT x,y,distance FROM
(SELECT b.coordinatePoint,dc.distance FROM beacons b
INNER JOIN deviceBeaconConnection dc
ON b.idbeacon = dc.idbeacon AND dc.idconnection = 6) bd
INNER JOIN floorMapPoints fmp
ON fmp.coordinatePoint = bd.coordinatePoint;
```

```
 * mysql+pymysql://brickea_mac:***@rm-
0xih4pk94w41k3c5j8o.mysql.rds.aliyuncs.com/mydb
1 rows affected.
```

x	y	distance
11	14	8.062

```
%%sql
SELECT x,y,distance FROM
(SELECT b.coordinatePoint,dc.distance FROM beacons b
INNER JOIN deviceBeaconConnection dc
ON b.idbeacon = dc.idbeacon AND dc.idconnection = 7) bd
INNER JOIN floorMapPoints fmp
ON fmp.coordinatePoint = bd.coordinatePoint;
```

```
 * mysql+pymysql://brickea_mac:***@rm-
0xih4pk94w41k3c5j8o.mysql.rds.aliyuncs.com/mydb
1 rows affected.
```

x	y	distance
4	13	14.318

```
%%sql
SELECT x,y,distance FROM
(SELECT b.coordinatePoint,dc.distance FROM beacons b
INNER JOIN deviceBeaconConnection dc
ON b.idbeacon = dc.idbeacon AND dc.idconnection = 8) bd
INNER JOIN floorMapPoints fmp
ON fmp.coordinatePoint = bd.coordinatePoint;
```

```
 * mysql+pymysql://brickea_mac:***@rm-
0xih4pk94w41k3c5j8o.mysql.rds.aliyuncs.com/mydb
1 rows affected.
```

x	y	distance
7	15	12.083

```
%%sql
CALL get_3_connected_beacons_coordinate_and_distance(6,7,8)
```

```
 * mysql+pymysql://brickea_mac:***@rm-
0xih4pk94w41k3c5j8o.mysql.rds.aliyuncs.com/mydb
3 rows affected.
```

x	y	distance
11	14	8
4	13	14

7	15	12

Positing test

Trilateration Positioning Algorithm

```
# Trilateration Positioning Algorithm
def trilateration(beacons=None,distance=None):
    A = 2*(beacons.iloc[2].y - beacons.iloc[0].y)
    A_ = 2*(beacons.iloc[1].y - beacons.iloc[0].y)
    B = 2*(beacons.iloc[2].x - beacons.iloc[0].x)
    B_ = 2*(beacons.iloc[1].x - beacons.iloc[0].x)

    delta_1 = np.square(distance[0]) - np.square(distance[2]) +
np.square(beacons.iloc[2].y) - np.square(beacons.iloc[0].y) +
np.square(beacons.iloc[2].x) - np.square(beacons.iloc[0].x)
    delta_2 = np.square(distance[0]) - np.square(distance[1]) +
np.square(beacons.iloc[1].y) - np.square(beacons.iloc[0].y) +
np.square(beacons.iloc[1].x) - np.square(beacons.iloc[0].x)

    device_x = (delta_1 * A_ - delta_2 * A)/(B * A_ - B_ * A)
    device_y = (delta_1 * B_ - delta_2 * B)/(B_ * A - B * A_)

    return (round(device_x),round(device_y))
```

Get a particular device current position

```
%%sql
select * from get_user_and_device;
```

```
 * mysql+pymysql://brickea_mac:***@rm-
0xih4pk94w41k3c5j8o.mysql.rds.aliyuncs.com/mydb
15 rows affected.
```

idcustomer	firstName	lastName	iddevice	deviceType	NumConnectionsAvailable
1	first_name_0	last_name_0	2	Phone	3
1	first_name_0	last_name_0	6	Laptop	0
1	first_name_0	last_name_0	7	Ipad	0
2	first_name_1	last_name_1	3	Phone	4
2	first_name_1	last_name_1	9	Ipad	0
3	first_name_2	last_name_2	15	Phone	0

4	first_name_3	last_name_3	4	Phone	3
4	first_name_3	last_name_3	8	Ipad	0
5	first_name_4	last_name_4	5	Phone	5
6	first_name_5	last_name_5	10	Phone	4
7	first_name_6	last_name_6	11	Phone	4
8	first_name_7	last_name_7	12	Phone	5
9	first_name_8	last_name_8	13	Phone	5
10	first_name_9	last_name_9	14	Phone	4
11	first_name_9	last_name_9	1	Phone	5

```
# Get best 3 connections for a particular device
iddevice = 2
sql = 'CALL
get_current_best_3_connections('+str(iddevice)+',@number,@c1,@c2,@c3);'
best_3_connection = pd.read_sql(sql,connection)
best_3_connection
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	num_available_connections	connection_1	connection_2	connection_3
0	3	6	8	7

```
%%sql
select * from devicebeaconconnection
```

```
 * mysql+pymysql://brickea_mac:***@rm-
0xih4pk94w41k3c5j8o.mysql.rds.aliyuncs.com/mydb
42 rows affected.
```

idconnection	iddevice	idbeacon	distance	beaconRankbyDist

1	1	5	7.211	4
2	1	6	10.0	5
3	1	1	5.831	3
4	1	2	4.0	1
5	1	7	5.0	2
6	2	7	8.062	1
7	2	3	14.318	3
8	2	2	12.083	2
9	3	26	6.403	2
10	3	30	6.0	1
11	3	32	8.944	3
12	3	31	9.0	4
13	4	19	8.944	1
14	4	21	12.806	3
15	4	20	11.705	2
16	5	17	8.602	5
17	5	12	5.657	4
18	5	13	3.162	1
19	5	18	3.606	2
20	5	14	4.123	3
21	11	27	6.083	1
22	11	28	6.708	3
23	11	23	9.434	4
24	11	24	6.403	2
25	12	38	9.849	3
26	12	39	9.0	2
27	12	34	13.601	5
28	12	35	10.63	4
29	12	40	7.211	1
30	13	47	9.22	5
31	13	48	6.403	4

32	13	52	4.0	2
33	13	54	4.472	3
34	13	53	1.0	1
35	14	4	3.162	2
36	14	8	2.236	1
37	14	10	6.403	4
38	14	9	5.099	3
39	10	25	12.649	4
40	10	26	9.487	3
41	10	30	5.385	2
42	10	32	1.0	1

```
# get connected 3 beacons coordinate and distance
sql = 'CALL get_3_connected_beacons_coordinate_and_distance('\
                    +str(best_3_connection.connection_1[0])+','\
                    +str(best_3_connection.connection_2[0])+','\
                    +str(best_3_connection.connection_3[0])+')'
coordinate_distance = pd.read_sql(sql,connection)
coordinate_distance
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	x	y	distance
0	11	14	8
1	7	15	12
2	4	13	14

```
# Trilateration Positioning Algorithm
def trilateration(beacons=None,distance=None):
    A = 2*(beacons.iloc[2].y - beacons.iloc[0].y)
    A_ = 2*(beacons.iloc[1].y - beacons.iloc[0].y)
    B = 2*(beacons.iloc[2].x - beacons.iloc[0].x)
    B_ = 2*(beacons.iloc[1].x - beacons.iloc[0].x)

    delta_1 = np.square(distance[0]) - np.square(distance[2]) +
np.square(beacons.iloc[2].y) - np.square(beacons.iloc[0].y) +
np.square(beacons.iloc[2].x) - np.square(beacons.iloc[0].x)
    delta_2 = np.square(distance[0]) - np.square(distance[1]) +
np.square(beacons.iloc[1].y) - np.square(beacons.iloc[0].y) +
np.square(beacons.iloc[1].x) - np.square(beacons.iloc[0].x)

    device_x = (delta_1 * A_ - delta_2 * A)/(B * A_ - B_ * A)
    device_y = (delta_1 * B_ - delta_2 * B)/(B_ * A - B * A_)

    return (round(device_x),round(device_y))
```

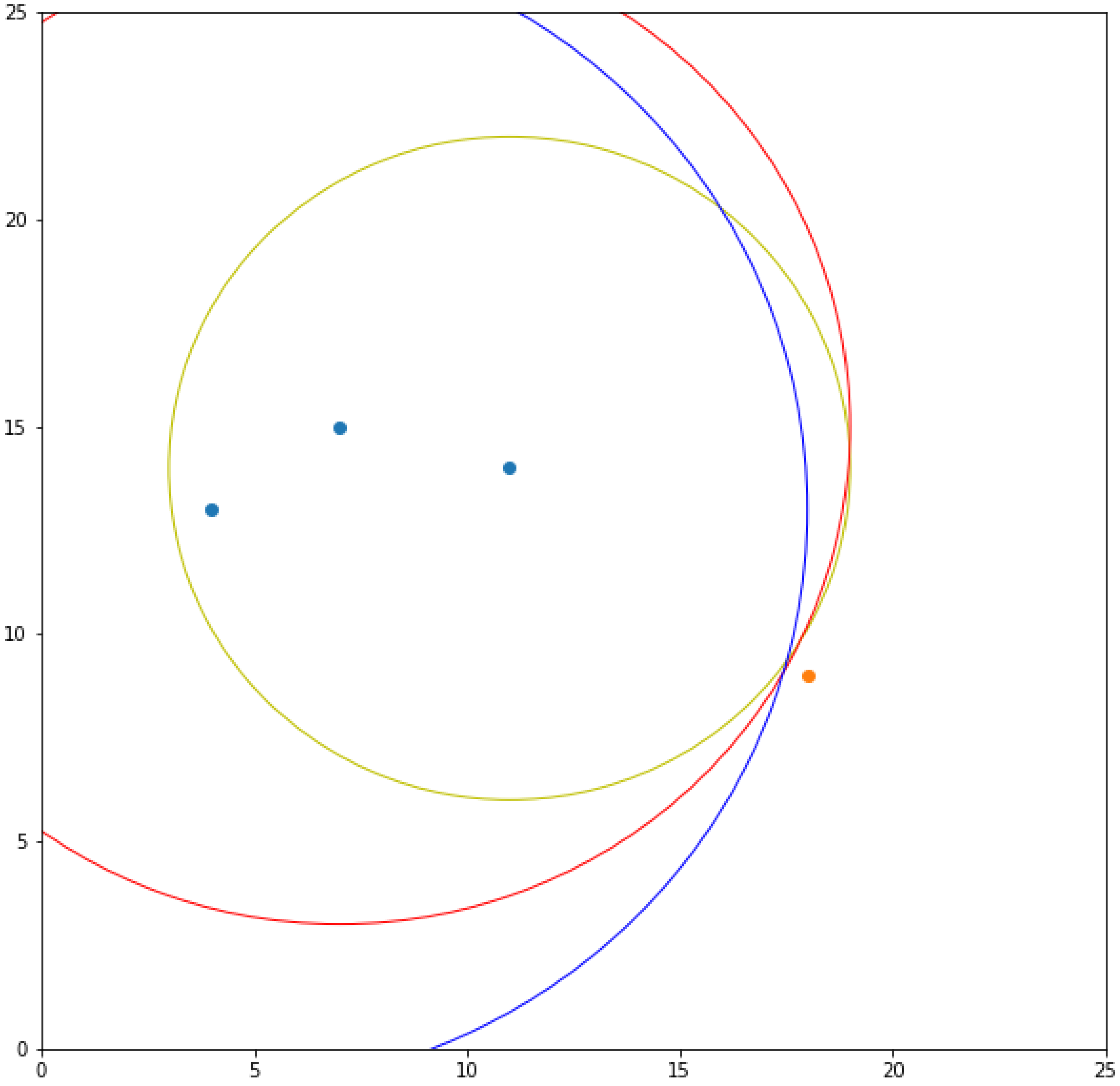
```
current_x,current_y = trilateration(beacons = coordinate_distance,distance
= coordinate_distance.distance)
(current_x,current_y)
```

```
(18.0, 9.0)
```

```
# Plot the current position
plt.figure(figsize=(10,10))
circle = plt.Circle((coordinate_distance.x[0], coordinate_distance.y[0]),
coordinate_distance.distance[0], color='y', fill=False)
plt.gcf().gca().add_artist(circle)
circle = plt.Circle((coordinate_distance.x[1], coordinate_distance.y[1]),
coordinate_distance.distance[1], color='r', fill=False)
plt.gcf().gca().add_artist(circle)
circle = plt.Circle((coordinate_distance.x[2], coordinate_distance.y[2]),
coordinate_distance.distance[2], color='b', fill=False)
plt.gcf().gca().add_artist(circle)

plt.scatter(coordinate_distance.x,coordinate_distance.y)
plt.scatter(current_x,current_y)
plt.xlim(0, 25)
plt.ylim(0, 25)
```

```
(0, 25)
```

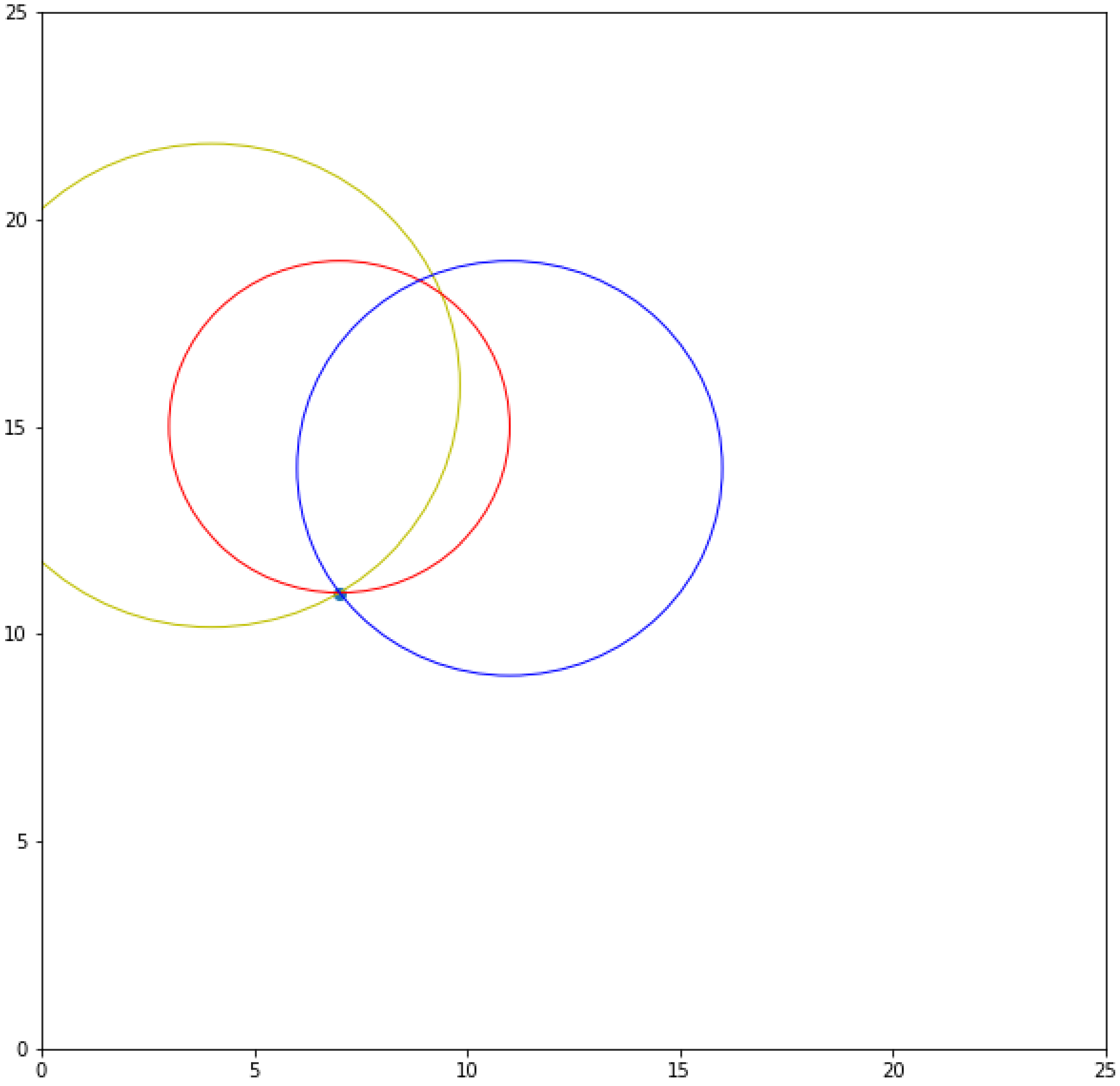


7 12 11 11 11 8

```
# Plot the current position
plt.figure(figsize=(10,10))
circle = plt.Circle((4, 16), 5.83, color='y', fill=False)
plt.gcf().gca().add_artist(circle)
circle = plt.Circle((7, 15), 4, color='r', fill=False)
plt.gcf().gca().add_artist(circle)
circle = plt.Circle((11, 14), 5, color='b', fill=False)
plt.gcf().gca().add_artist(circle)

plt.scatter(7,11)
plt.scatter(current_x,current_y)
plt.xlim(0, 25)
plt.ylim(0, 25)
```

```
(0, 25)
```


25
20
15
10
5
0
0
5
10
15
20
25