Assignment 4

DS TOOLS
LAST ASSIGNMENT

Supervised by
Dr. Mohamed
Eng. Sohaila

TEAM NAMES AND IDS

Name	ID
MAHMOUD ESSAM	20221460231
ABDELRAHMAN ASHRAF	20221374041
ABDULLAH HUSSIEN	20221427861
FARES MUHAMMED	20221461330
ZYAD ASHRAF HAFEZ	20221374025

ASSIGNMENT 4

Question No.1: Given the following dissimilarity matrix between three objects

P(X) =

0	1	4
1	0	2
4	2	0

A) Sketch a dendrogram using single linkage strategy?

IDX	1	2	3
1	0	1	4
2	1	0	2
3	4	2	0

IDX	1,2	3
1,2	0	2
3	2	0

IDX	1,2,3
1,2,3	0

FINAL SHAPE

"((1,2),3)"

B) Sketch a dendrogram using complete linkage strategy?

IDX	1	2	3
1	0	1	4
2	1	0	2
3	4	2	0

IDX	1,2	3
1,2	0	4
3	4	0

IDX	1,2,3
1,2,3	0

FINAL SHAPE
"((1,2),3)"

C) Write down the corresponding Python steps of parts (A) include a proper cut-off point?

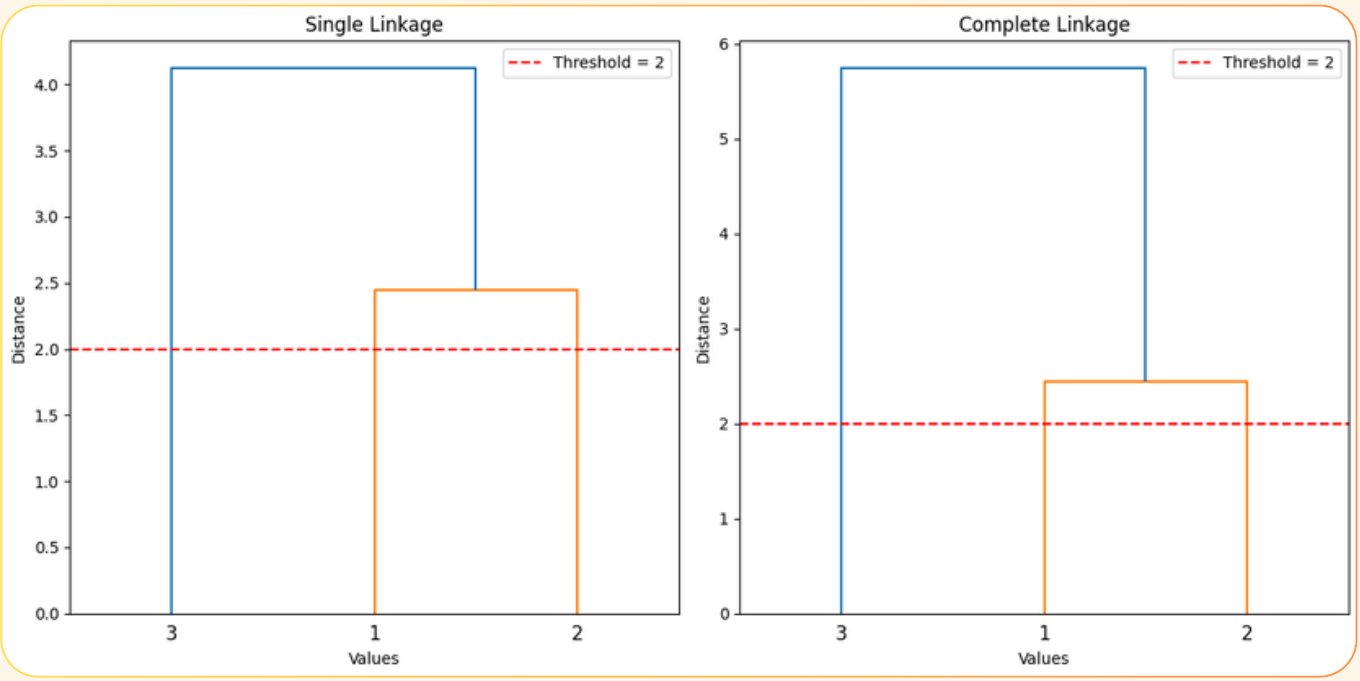
```
# Given dissimilarity matrix
P_X = np.array([[0, 1, 4], [1, 0, 2], [4, 2, 0]])
threshold = int(input("Set Threshold for proper: "))
# Perform hierarchical clustering using single linkage
Z_single = linkage(P_X, method="single")
# Perform hierarchical clustering using complete linkage
Z_complete = linkage(P_X, method="complete")
```

```
[0.        1.        2.44948974 2.        ]
[2.        3.        4.12310563 3.        ]
```

```
[0.        1.        2.44948974 2.        ]
[2.        3.        5.74456265 3.        ]
```

```
# Perform Proper Cut-off
# “needed number of clusters based on hyperparameter value”
clusters_single = fcluster(Z_single, t=threshold,
criterion="distance")
print("Single Linkage Clusters:", clusters_single[-1])
```

Based on the input value, will give him all number of clusters can the dendrogram handle “With using Matplotlib for plotting results”



Based on the value of input, will find that he has broken 3 lines, about the all values which leads to 3 clusters

---

d) Using the first and second object as initial two centers for clusters C_1 and C_2 respectively. What is the hard membership of the third object in C_1 and C2

In this situation, will calculate the distance from (Third Object) to C1, and (Third Object with C2) using euclidean distance

distance_to_C1 = np.sqrt((P_X[2, 0] - P_X[0, 0]) ** 2 + (P_X[2, 1] - P_X[0, 1]) ** 2) 4.123106
distance_to_C2 = np.sqrt((P_X[2, 0] - P_X[1, 0]) ** 2 + (P_X[2, 1] - P_X[1, 1]) ** 2) 3.605551

Will find that Object so near to Second cluster more that first one so (Object Belongs to C2)

E) What is the center of the new formed cluster in part (D) if X1=[1,3], X2=[2,5], X3=[3,7]

Let's Get Distance Matrix Firstly for those objects

Columns	X1	X2	X3
X1	0	2.236	4.472
X2	2.236	0	2.236
X3	4.472	2.236	0

We know that our old Clusters are

C1 = [0, 1, 4]

C2 =[1, 0, 2]

so by building our hardmembership matrix

Columns	C1	C2	Position
X1	1.323169	3.480152	C1
X2	3.018519	1.258409	C2
X3	6.125999	4.588652	C2

So by taking the mean values of X1 for C1 divided by 3, and same operation for C2, will find new Centroids for both are

C1=(1,3)

C2=(2.5,6)

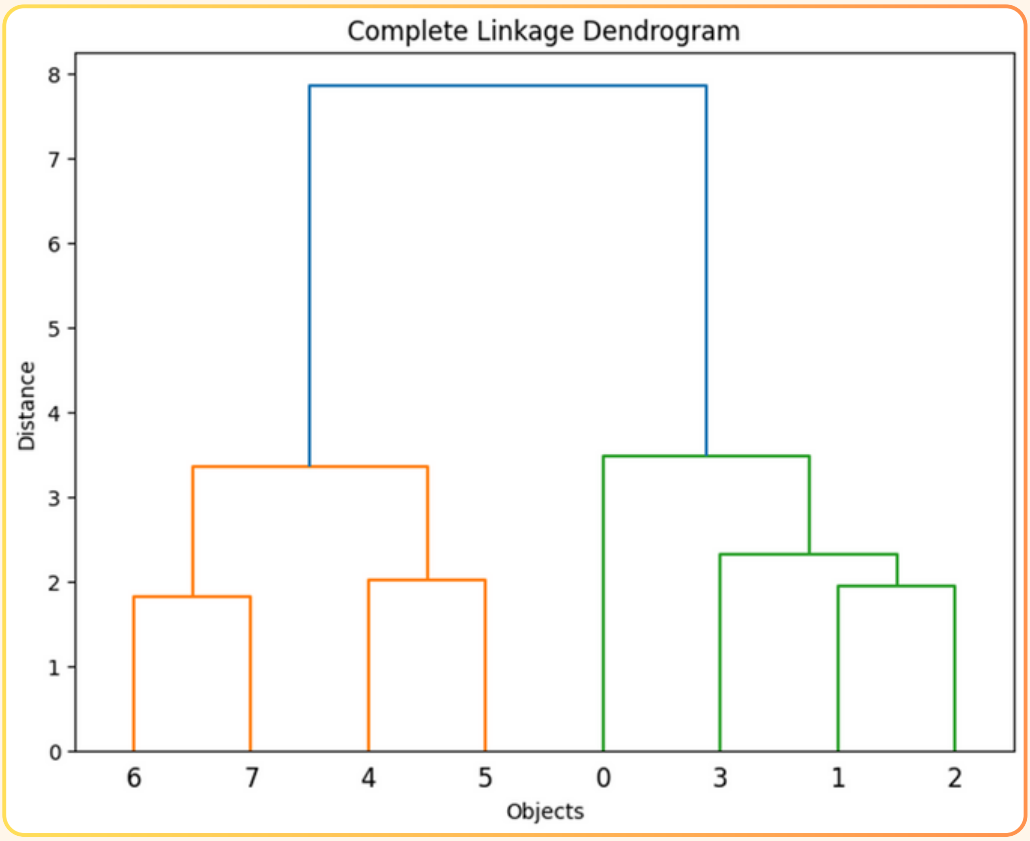
"All used codes are inserted with report in teams"

Question No.2: hierarchical, Medoid-based and density-based clustering

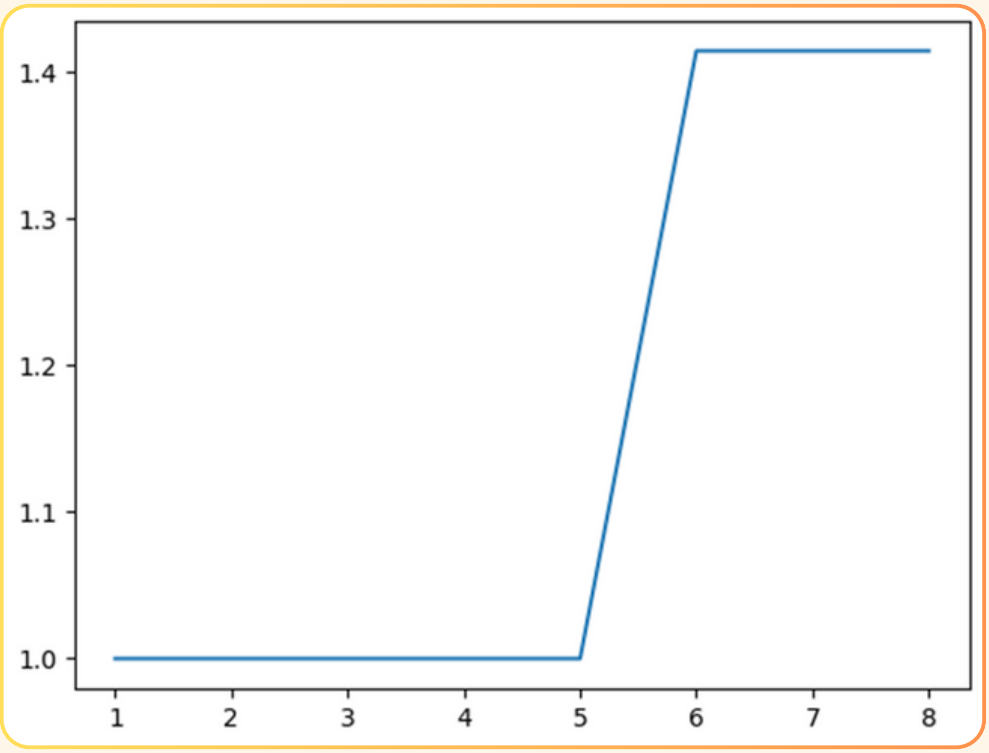
A) what's the cost of replacing m1 with (2,2)?

The cost is determined by the discrepancy in the loss function between two distinct centroids. The visual representation makes it clear which points belong to each cluster, except for (1, 3) and (3, 3), as their distances to both centroids are identical. We'll assume both points belong to the first cluster, as it doesn't impact the objective function's value. Initially, the objective function J(centroids) equals 7.8284, calculated as $0 + 1 + 2 + \sqrt{2} + 0 + 1 + 1 + \sqrt{2}$. After replacement, J(centroids) becomes 7.24264, computed as $0 + 3\sqrt{2} + 1 + 1 + 1$.

C) Cluster above data using complete linkage strategy and sketch the corresponding dendrogram

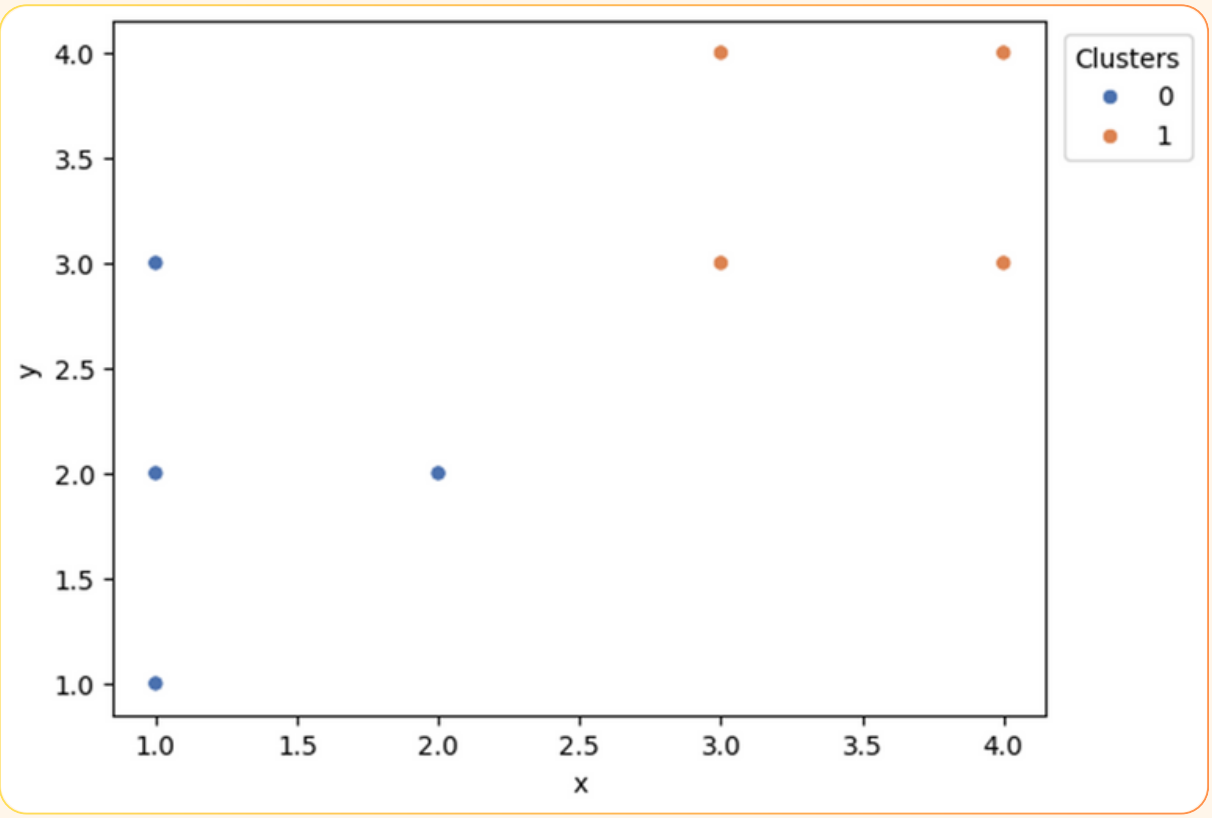


C) Sketch k-dist graph for k=3 and estimate suitable value for Epson.

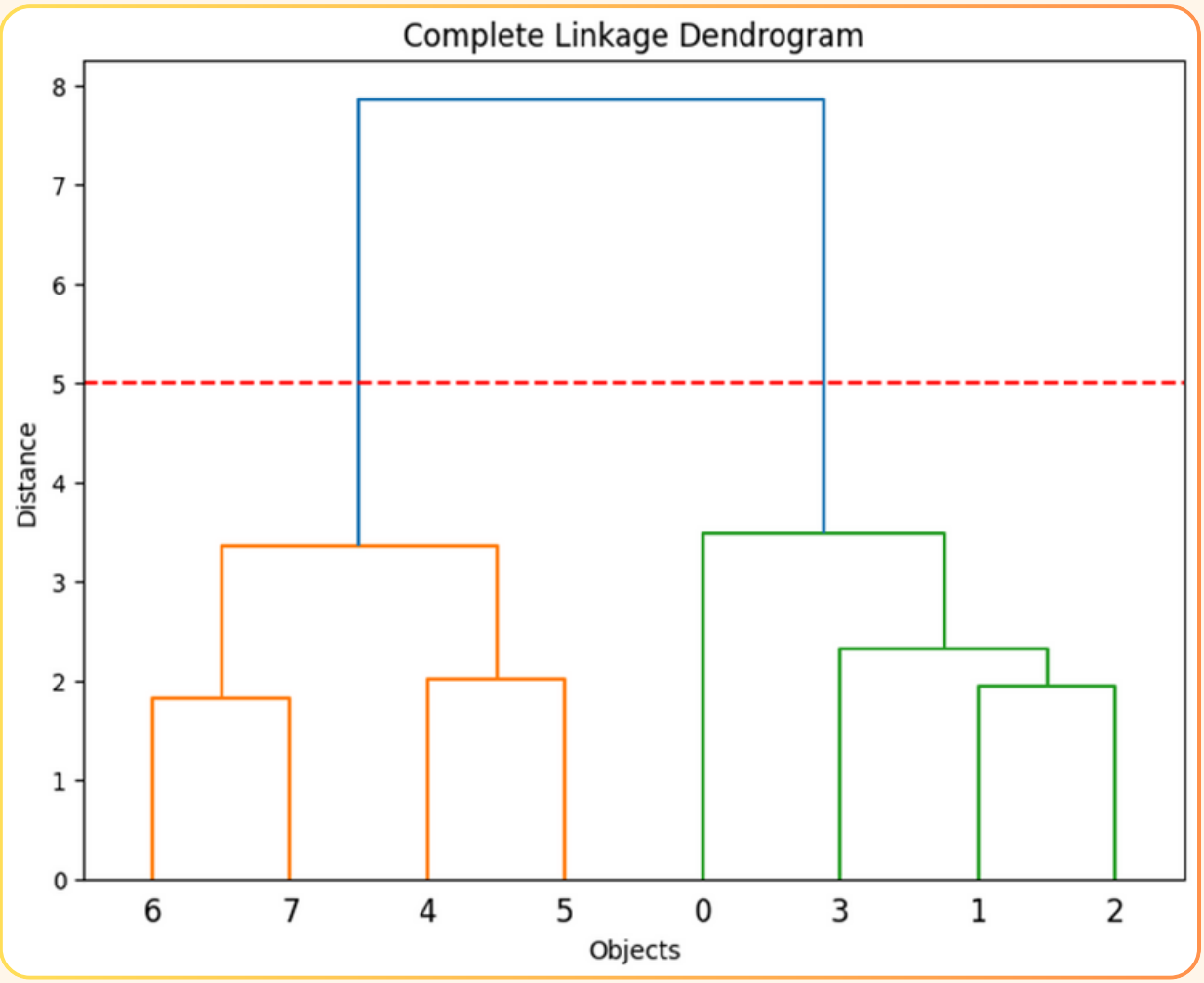


From the graph it seems as if the best value for Epson is 1

D) Apply DBSCAN on the given dataset starting from (4,4) using k=1 and Epson=√2



D) Estimate cutting edges for the dendrogram in part B to produce the same clusters producedby DBSCAN in part D



Question No.3: Given the following dissimilarity matrix between Five objects

P(X) =

0	1	5	6.4	7.4
1	0	4.2	5.7	6.7
5	4.2	0	1.4	2.5
6.4	5.7	1.4	0	1.1
7.4	6.7	2.5	1.1	0

A) Sketch a dendrogram using single linkage strategy?

IDX	1	2	3	4	5
1	0	1	5	6.4	7.4
2	1	0	4.2	5.7	6.7
3	5	4.2	0	1.4	2.5
4	6.4	5.7	1.4	0	1.1
5	7.4	6.7	2.5	1.1	0

IDX	1,2	3	4	5
1,2	0	4.2	5.7	6.7
3	4.2	0	1.4	2.5
4	5.7	1.4	0	1.1
5	6.7	2.5	1.1	0

IDX	1,2	3	4,5
1,2	0	4.2	5.7
3	4.2	0	1.4
4	5.7	1.4	0

IDX	1,2	3,4,5
1,2	0	4.2
3	4.2	0

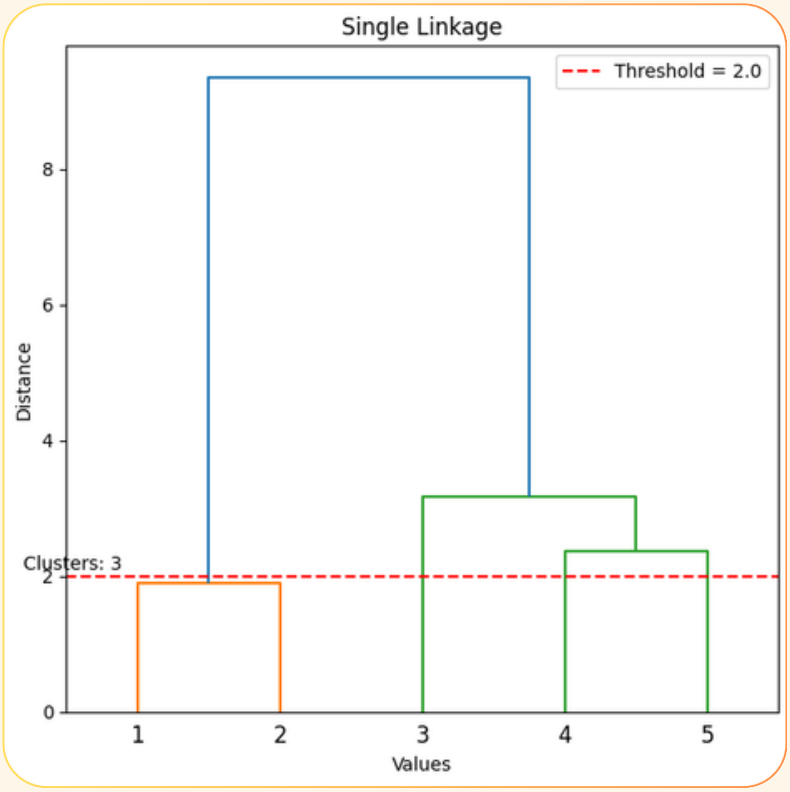
IDX	1,2,3,4,5
1,2,3,4,5	0

FINAL SHAPE

((1,2),(3,(4,5))

B) Select a proper cut off point and output the corresponding clustering?

Based on the code attached with the report number 3
will find that by using proper cut off at distance = 2
the result will lead for 3 clusters from this dendrogram



C) Draw a dendrogram using complete linkage strategy?

IDX	1	2	3	4	5
1	0	1	5	6.4	7.4
2	1	0	4.2	5.7	6.7
3	5	4.2	0	1.4	2.5
4	6.4	5.7	1.4	0	1.1
5	7.4	6.7	2.5	1.1	0

IDX	1,2	3	4	5
1,2	1	5	6.4	7.4
3	5	0	1.4	2.5
4	6.4	1.4	0	1.1
5	7.4	2.5	1.1	0

IDX	1,2	3	4,5
1,2	0	5	6.4
3	5	0	1.4
4	6.4	1.4	0

IDX	1,2	3,4,5
1,2	0	6.4
3	6.4	0

IDX	1,2,3,4,5
1,2,3,4,5	0

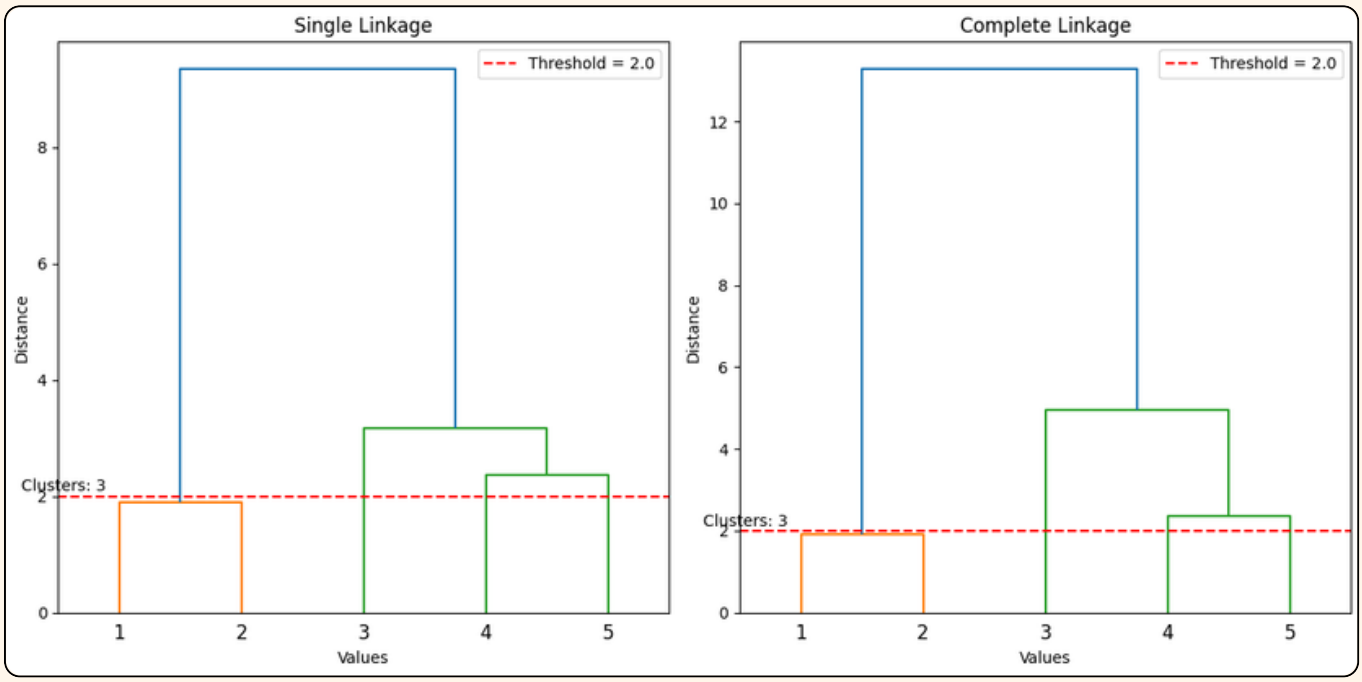
FINAL SHAPE

((1,2),(3,(4,5))

D) Write down the corresponding Python steps of parts A,B,C?

All is done in the notebook attached with Report

E) Comment on the resulting dendrograms in A and C?



Main Difference between two is how the calculation of distance
from our detailed answer we have achieved same dendrogram, so will find the similarity between complete and single, but main difference is the how of calculation is achieved
where Single calculate min distance = min(d(z,(x,y))
while complete one = max(d(z,(x,y))

F) What is the complexity of the algorithm in terms of number of objects?

As we said, both performing same action, and main difference is how the achievement happening.
so if we saw it as a tree way, it will achieve firstly N^2
and due to the way of divide and conquer of the tree, by taking each 2 values together, we are breaking the value as Log(n)
so multiply both together at same time, will achieve complexity
$O(N^2 \text{Log}(N))$

Question No.4: Complete The dots

a) The size of similarity matrix between elements of X is N*N
b) The size of membership matrix of X in k clusters is N*K
c) The size of Linkage matrix output by matlab is-- N-1*4
d) The number of internal nodes of dendrogram is N-1
e) In order to determine the pair of clusters that is going to be merged at the r + 1 level, N-R pairs of clusters have to be considered.
f) The complexity of heirarchical clustering is $O(N^2)$

Question No.(5): "(True/False) "

1. Number of clusters is a required parameter for hierarchical algorithms [FALSE]
2. Cancer Diagnosis is a clustering application [False]
3. Image segmentation is a classification problem [False]
4. Noise is a random error or variance in a measured variable [TRUE]

Q6) Compute Accuracy, Sensitivity and Specificity of the following classification results?

First we imported the data as a dictionary in python.
Then,We converted it to a pandas dataframe to make the operations done on the data easier.

ID	Income	Marital Status	Refund	Actual Class	Predicted Class
1	95K	M	Y	-1	1
2	120K	S	Y	-1	-1
3	140K	S	N	1	1
4	80K	M	N	-1	-1
5	160K	D	Y	-1	1
6	100K	M	Y	-1	-1
7	90K	S	Y	-1	1
8	75K	D	N	-1	-1
9	170K	D	N	1	1
10	125K	S	N	1	-1

Then we made confusion matrix using confusion_matrix built in function from sickt.learn

	Predicted 0	Predicted 1
Actual 0	4	3
Actual 1	1	2

Then we declared the total number of records and correct predictions to calculate accuracy.

accuracy = correct_predictions / total_predictions

Then we declared the needed values for all needed rules.

TP = conf_matrix[1,1] **TN = conf_matrix[0,0]**
FN = conf_matrix[1,0] **FP = conf_matrix[0,1]**

sensitivity = tp / (tp + fn) → Sensitivity: 66.67%
specificity = tn / (tn + fp) → Specificity: 57.14%

Q7) Consider the following k-nearest table of data points. What are the accuracy, specificity precision, recall, f-measure and sensitivity of k-nearest neighbor on them assuming the actual labels of x1,x3, and x5 is 1 and x2,x4 and x6 is -1 using leave-one-out cross validation method (k=1)?

Object Xi	Nearest Neighbors
X1	x3, x4, x6
X2	x4, x6, x2
X3	x5, x2, x3
X4	x6, x3, x1
X5	x4, x2, x1
X6	x4, x1, x1

We see the nearest neighbor label and compare it with the actual label of the point , if they match , the prediction is correct,if not then it's false prediction

x1: Predicted label 1 (correct, actual label 1)
x2: Predicted label -1 (correct, actual label -1)
x3: Predicted label 1 (correct, actual label 1)
x4: Predicted label -1 (correct, actual label -1)
x5: Predicted label -1 (incorrect, actual label 1)
x6: Predicted label -1 (correct, actual label -1)

accuracy = correct predictions / total
=5/6 * 100 =83.33%

Precision = (2 + 2) / (2 + 0) = 4 / 2 = 2
Recall (Sensitivity) = (2 + 2) / (2 + 0) = 4 / 2 = 2
Specificity = (2) / (2 + 0) = 2 / 2 = 1
F-measure = 2 * (2 * 2) / (2 + 2) = 8 / 4 = 2

Question No.(8): "compute accuracy , Sensitivity and Specificity of Model M1 & M2 "

1. Apply model rules to predict classes:

M1 correctly predicts 3 classes (-1, -1, -1) and incorrectly predicts 4 classes (1, 1, 1, 1).
M2 correctly predicts 2 classes (-1, -1) and incorrectly predicts 5 classes (1, 1, 1, 1, -1).

2. Calculate accuracy:

-Accuracy = (number of correct predictions) / (total number of predictions)

- M1 accuracy = 3/7 = **0.4286**
- M2 accuracy = 2/7 = **0.2857**

3. Calculate sensitivity:

-Sensitivity = (number of true positives) / (number of actual positives)

- M1 sensitivity = 0/4 = **0**
- M2 sensitivity = 0/4 = **0**

4. Calculate specificity:

-Specificity = (number of true negatives) / (number of actual negatives)

- M1 specificity = 3/7 = **0.4286**
- M2 specificity = 2/7 = **0.3157**

Question No.(9): "Predict using KNN (3)"

Distance = sqrt((age1 - age2)^2 + (loan1 - loan2)^2)

1. Identify the k nearest neighbors:

Sort the data points by their distance to the new point, ascending.
Select the top 3 nearest neighbors (k=3):
Age 35, Loan $60,000, House Price Index 256 (Distance 5,000)
Age 40, Loan $62,000, House Price Index 216 (Distance 7,000)
Age 25, Loan $40,000, House Price Index 135 (Distance 15,000)

2. Average the house price indices of the nearest neighbors:

(256 + 216 + 135) / 3 = 202.33

3. Predicted house price index:

The predicted house price index for age 38 and loan $55,000 is 202.33

Question No.10: "compute accuracy , Sensitivity and Specificity of Model applied LOOCV"

ID	Class	Predicted class
1	1	-1
2	-1	1
3	-1	1
4	1	-1
5	1	1
6	1	1
7	1	1
8	1	1
9	-1	-1
10	-1	-1
11	-1	-1
12	1	-1
13	-1	1
14	-1	1

Accuracy = ½

sensitivity = 4 / 7 = 0.571,

specificity = 3 / 7 = 0.42857

Question No.11: "compute accuracy , Sensitivity and Specificity of Model applied LOOCV"

Let's assess the induction rules for the given input [x1, x2] = [0.4, 0.6]:

Criterion for w1: x1 > 1/4 and x2 ⩽ 3/4

This criterion is met because 0.4 > 1/4 and 0.6 ⩽ 3/4.

Criterion for w2: x1 > 1/4 and x2 > 1/2 and x1 ⩽ 1/2

This criterion is not fulfilled as 0.4 > 1/4, but 0.6 > 1/2, and 0.4 ⩽ 1/2 is false.

Criterion for w3: x1 > 1/4 and x2 > 1/2 and x1 > 3/4

This criterion is not satisfied since 0.4 > 1/4, but 0.6 > 1/2, and 0.4 > 3/4 is false.

Criterion for w4: x1 > 1/4 and x2 > 1/2 and x1 > 3/4

This criterion is not met as 0.4 > 1/4, but 0.6 > 1/2, and 0.4 > 3/4 is false.

Upon evaluation, the input [0.4, 0.6] fulfills the condition for w1. Consequently, the class assigned to [0.4, 0.6] is w1.

Question No.12: "compute the information gain,with and complete your decision tree and Classify an unseen sample"

Step 1: Calculate Entropy for the Class Label 'Cheat': $-\sum pi/\log2(pi)$

Step 2: Calculate Information Gain for 'Refund' and 'Marital Status':

$IG(T,A)=H(T)-\sum v\in A(|Tv|/|T|)\cdot H(Tv)$

- IG(T,A) is the Information Gain for attribute A
- H(T) is the entropy of the target class before the split
- |T| is the total number of instances in the dataset.
- Tv is the subset of instances for a particular value v of attribute A
- H(Tv)H(Tv) is the entropy of the subset Tv for attribute A

Information Gain for Refund : 0.395815602003358 4

Information Gain for Marital Status : 0.2812908992306927

```
Refund
├──── Yes
│      └──── No
└──── No
       ├──── Married
       │      └──── No
       └──── Not Married
              └──── Income
                     ├──── <= 125K
                     │      └──── No
                     └──── > 125K
                            └──── Yes
```

Income = 60K, Marital Status = Single, Refund = No

Path through the tree:

Refund (No) → Not Married → Income (<= 125K) → Predicted Class: NO

Question No.13: Classify Using NB Classifier

P (buys_computer | yes) = 4 / 6, P (buys_computer | yes) = 2 / 6

Age	Yes	No
>40	¾	¼
<=30	½	½

Age	Yes	No
>40	¾	¼
<=30	½	½

Age	Yes	No
>40	¾	¼
<=30	½	½

Credit	Yes	No
Fair	⅘	⅕
Excellent	0	1

P (X = age<=30, income = medium, student = yes, credit_rating = fair)

P (buys_computer | yes) * P (age<=30 | yes) * P (income = medium | yes) * P (student = yes | yes) * P (credit_rating = fair | yes) **= 4/6 * ½ * ⅔ * ¾ * ⅘ = 2/15**

P (buys_computer | no) * P (age<=30 | no) * P (income = medium | no) * P (student = yes | no) * P (credit_rating = fair | no) **= 2/6 * ½ * ⅓ * ¼ * ⅕ = 1/360**

P (X | yes) = (2/15) / ((2/15) + (1/360)) = 0.9796

P (X | no) = (1/360) / ((2/15) + (1/360)) = 1-(P(X | yes)) = 0.0204

Question No.(14): Add label and information to nodes and edges in the following example and show the output

Let's Edit the code
Firstly, let's setting labels for adding edges

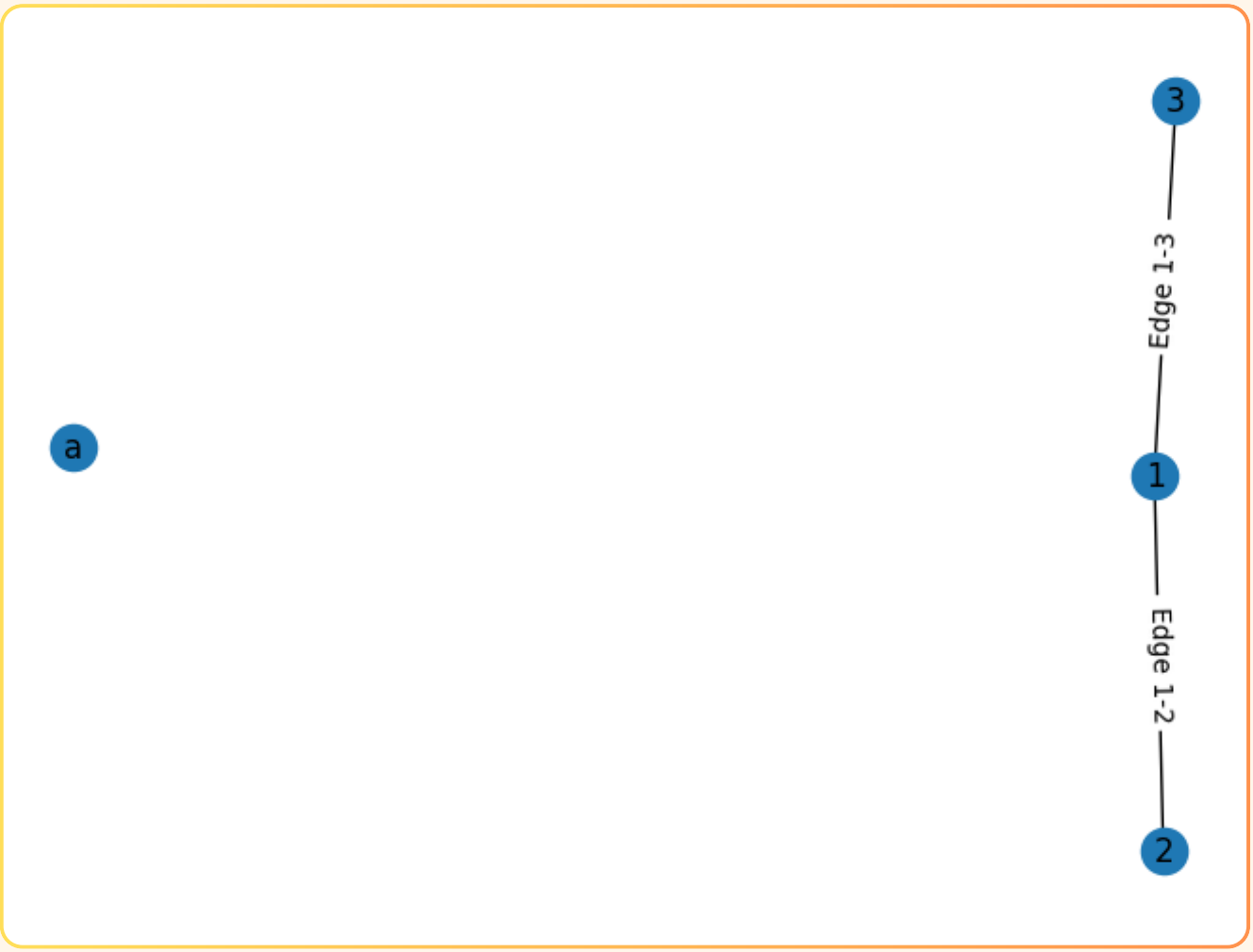
G.ADD_EDGES_FROM([(1, 2, {"LABEL": "EDGE 1-2"}), (1, 3, {"LABEL": "EDGE 1-3"})])
Simple Dictionary!

Let's add Label of A!

G.ADD_NODE("A", LABEL="NODE A")

Some Results?

Number of nodes: 4
Number of edges: 2
Nodes: [(1, {}), (2, {}), (3, {}), ('A', {'LABEL': 'NODE A'})]
Edges: [(1, 2, {'LABEL': 'EDGE 1-2'}), (1, 3, {'LABEL': 'EDGE 1-3'})]
Neighbors of node 1: [2, 3]
Degree of node 1: 2



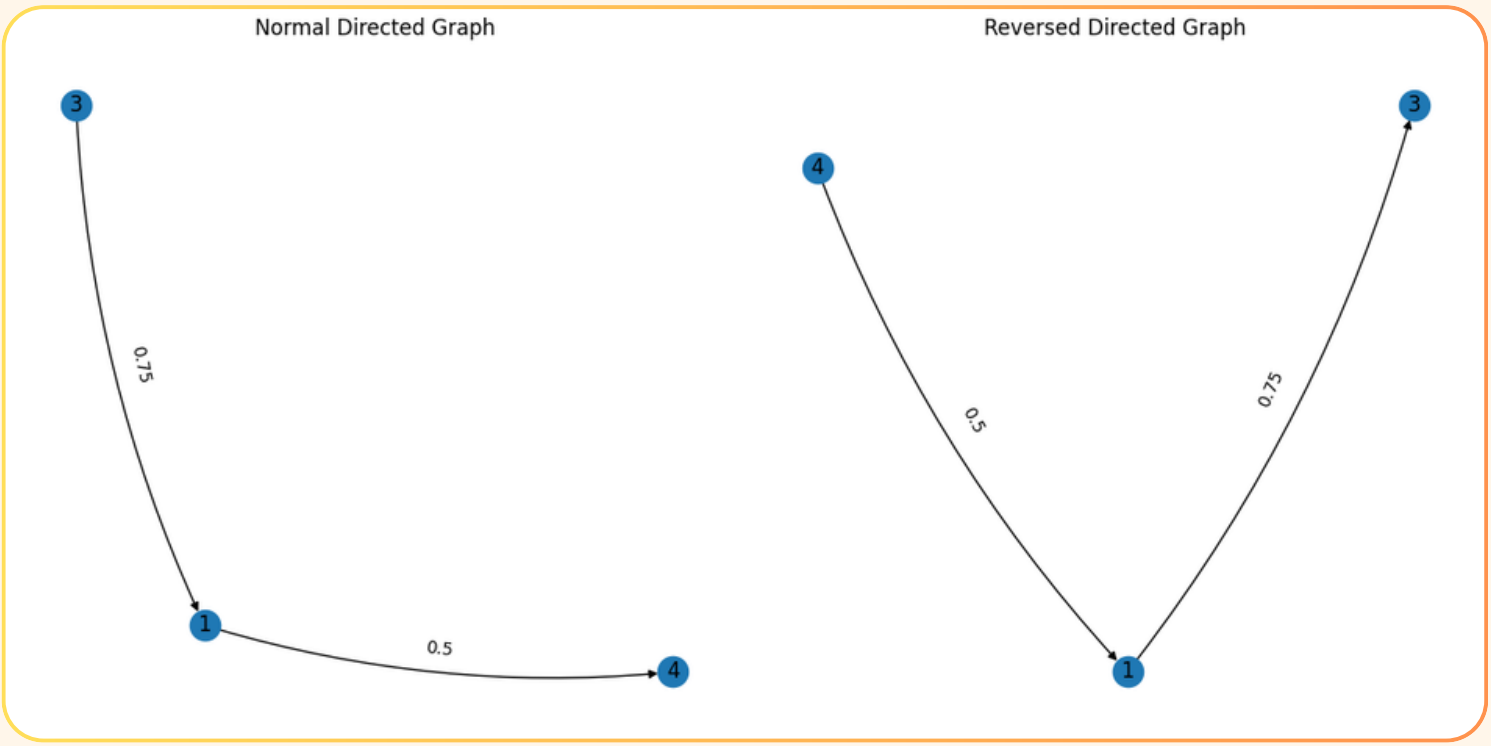
Question No.(15): reverse the direction of edges and rerun the following example to show the output

Simple Function, it's Reverse!
Will make some edits in code for print, and shows the reverse before and after

dg_reverse = dg.reverse()
That's all!, easy ha?

Some Results?

Weighted out-degree of node 1: 0.5
Weighted degree of node 1: 1.25
Successors of node 1: [4]
Predecessors of node 1: [3]



difference? Yep there's difference
will find that direction from 3 to 1 has inversed to be from 1 to 3
with same weights for directions, that's all!

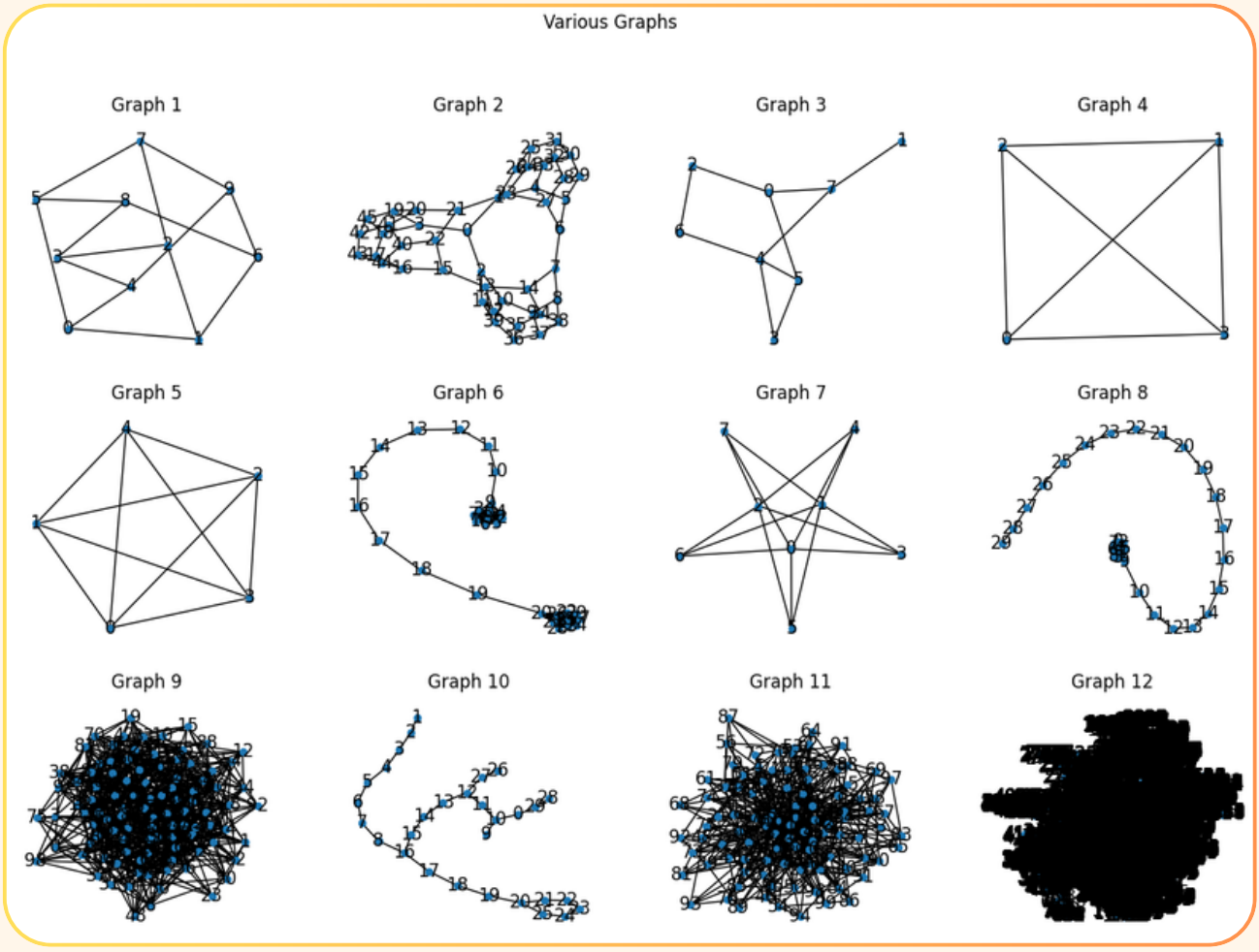
Question No.(16): Show the output of the following code

Showing some outputs? easy

with running same old code, with printing all needed information like, number of edges, and nodes and average degree based on weights

Purpose of showing it is to know the difference between each pattern

Name of graph is petersen with 10 nodes and 15 edges.
Name of graph is tutte with 46 nodes and 69 edges.
Name of graph is maze with 8 nodes and 10 edges.
Name of graph is tet with 4 nodes and 6 edges.
Name of graph is K_5 with 5 nodes and 10 edges.
Name of graph is barbell with 30 nodes and 101 edges.
Name of graph is K_3_5 with 8 nodes and 15 edges.
Name of graph is lollipop with 30 nodes and 65 edges.
Name of graph is er with 100 nodes and 741 edges.
Name of graph is ws with 30 nodes and 30 edges.
Name of graph is ba with 100 nodes and 475 edges.
Name of graph is red with 14629 nodes and 14628 edges.



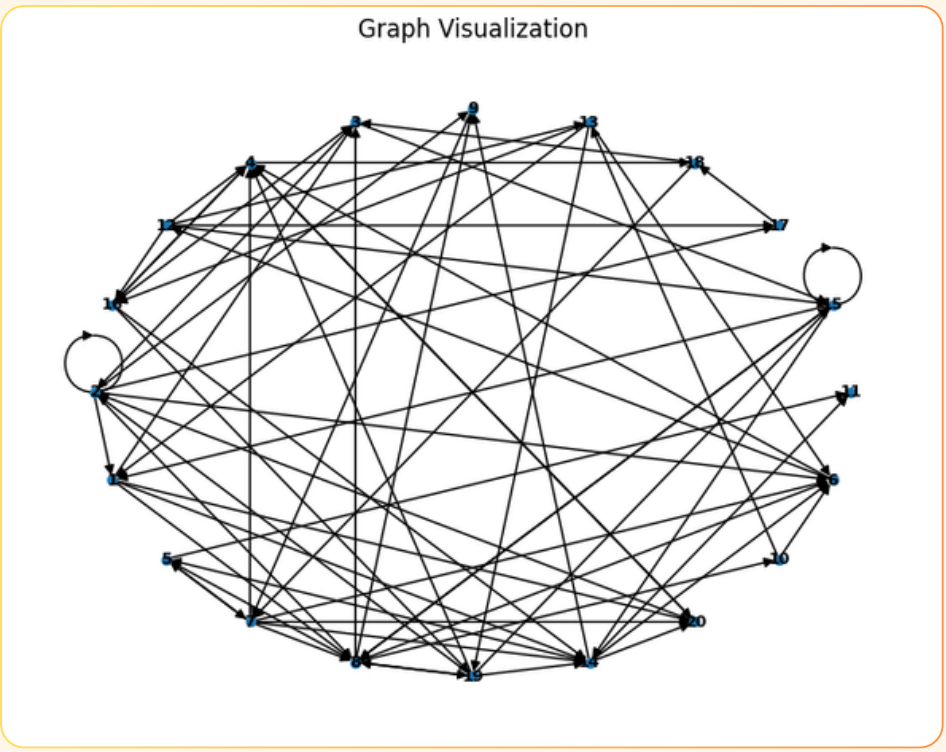
Question No.(17): replace the dataset in the following code by any of your choice and print nodes, edges count and average degree

All we need to do firstly, build random file handle the problem of the data

```
# Specify the number of nodes and edges
num_nodes = 20
num_edges = 70
# Generate edges with random weights
edges = [(random.randint(1, num_nodes), random.randint(1, num_nodes),
random.uniform(0.1, 1.0)) for _ in range(num_edges)]
# Write the edges with weights to the file
with open('my_file.txt', 'w') as file:
    for edge in edges:
        file.write(f"{edge[0]} {edge[1]} {edge[2]}\n")
print(f"Generated {num_nodes} nodes and {num_edges} edges with weights in
my_file.txt.")
```

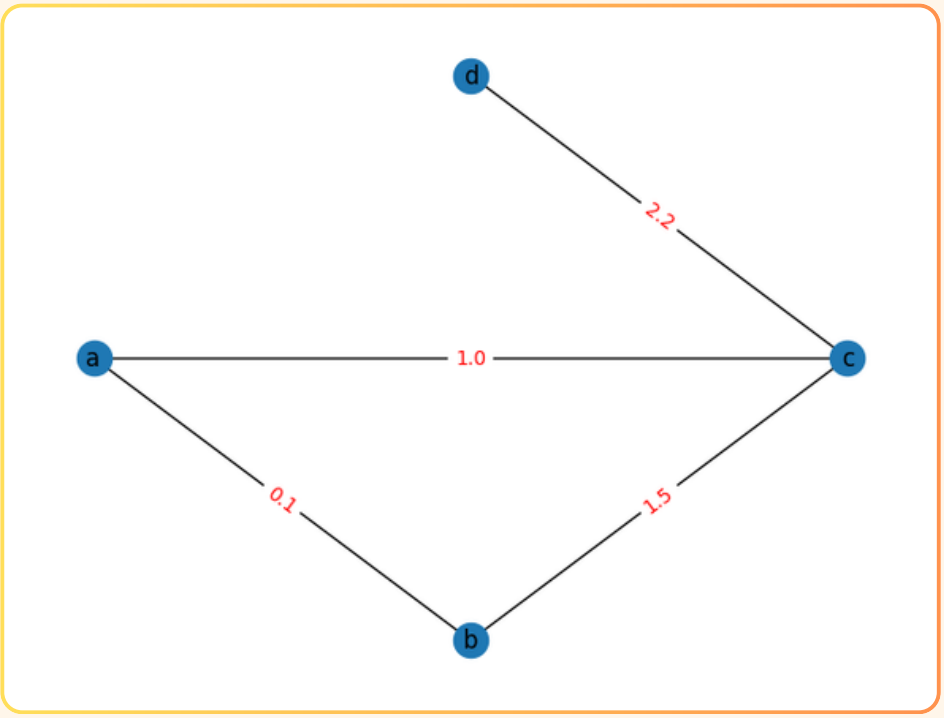
with running same old code, with printing all needed information like, number of edges, and nodes and average degree based on weights

Nodes: 20
Edges: 67
Average degree (considering weights): 3.599163970892539



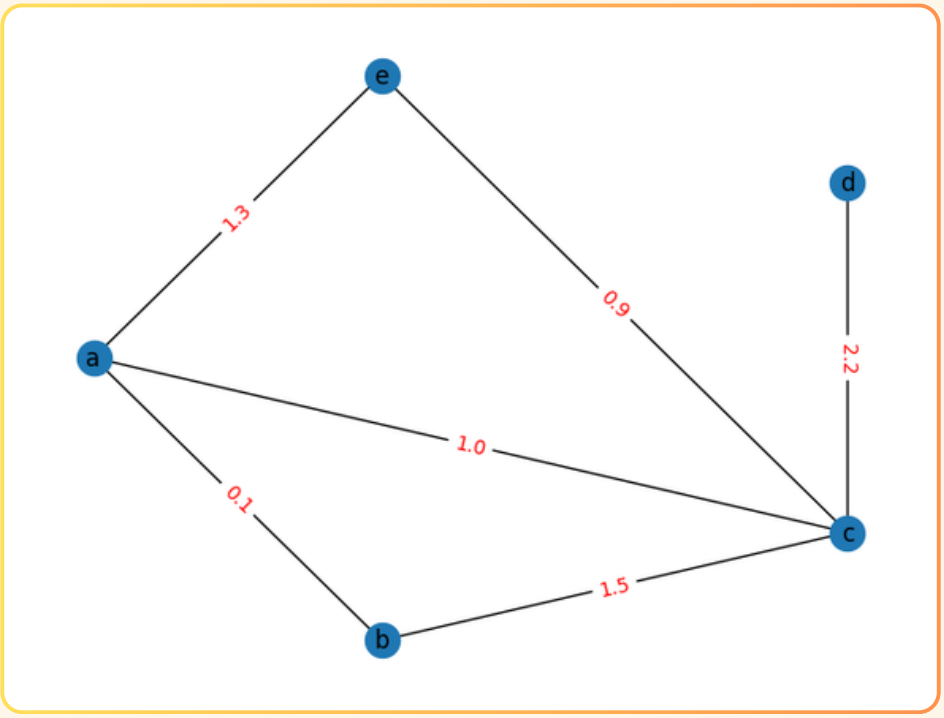
Question No.(18): Add two nodes e with edge to c and a having weight 0.9 and 13 respectively and compute the shortest path between a and d and the minimum spanning tree.

Based on the draw before the action with this weights will find that optimal one will be from b to a, then a to c and c to d



"Adding E to the draw"

After adding e to the draw will find that will change some by starting from a, a to c, to c then to d as if I wanted to go from a to d



Before the action happening

Before Adding The new node from e to c, and e to a with 0.9 and 1.3 respectively
Without Weights
['b', 'c', 'd'] With Weights ['b', 'a', 'c', 'd']

After the action happening

After Adding The new node from e to c, and e to a with 0.9 and 1.3 respecitvely
Without Weights ['a', 'c', 'd'] With Weights ['a', 'c', 'd'] with minimum_spanning_tree [('a', 'b'), ('a', 'c'), ('c', 'e'), ('c', 'd')]