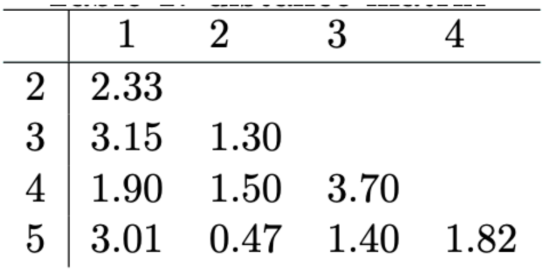
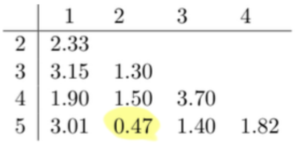
Q1 [20 Marks]

Apply the agglomerative hierarchical clustering algorithm with the following distance matrix and the single linkage. Plot the cluster tree and mark out all the merging levels.



	1	2	3	4
2	2.33			
3	3.15	1.30		
4	1.90	1.50	3.70	
5	3.01	0.47	1.40	1.82

Table 1： distance matrix



	1	2	3	4
2	2.33			
3	3.15	1.30		
4	1.90	1.50	3.70	
5	3.01	0.47	1.40	1.82

Step 1: merge 2,5 : at $\boxed{\text{level } 0.47}$ ⇒

$C_1=\{1\}$
$C_2=\{2,5\}$
$C_3=\{3\}$
$C_4=\{4\}$

updated proximity matrix:

$$\begin{array}{c|cccc} & C_1 & C_2 & C_3 & C_4 \\ C_1 & - & 2.33 & 3.15 & 1.90 \\ C_2 & 2.33 & - & 1.30 & 1.50 \\ C_3 & 3.15 & 1.30 & - & 3.70 \\ C_4 & 1.90 & 1.50 & 3.70 & - \end{array}$$

step 2: merge C2, C3 at $\boxed{\text{level } 1.30}$ ⇒

$C_1=\{1\}$
$C_2=\{2,3,5\}$
$C_3=\{4\}$

updated proximity matrix:

$$\begin{array}{c|ccc} & C_1 & C_2 & C_3 \\ C_1 & - & 2.33 & 1.90 \\ C_2 & 2.33 & - & 1.50 \\ C_3 & 1.90 & 1.50 & - \end{array}$$

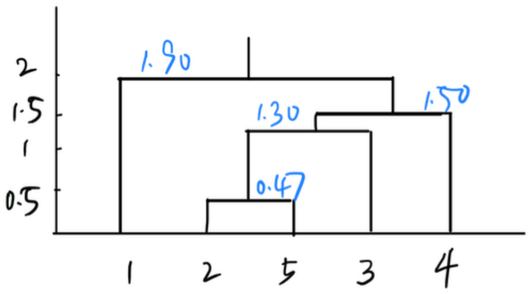
step 3: merge C2, C3 at level 1.50 => C1 = {1}
C2 = {2,3,4,5}

updated proximity matrix:

	C1	C2
C1	–	1.90
C2	1.90	–

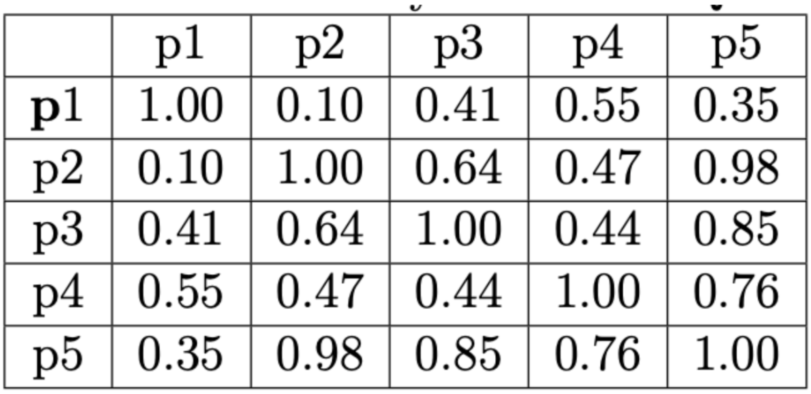
step 4: merge C1 . C2 at level 1.90

cluster tree



Q2 [20 Marks]

Use the similarity matrix in Table 2 to perform single-link hierarchical clustering. Show your results by drawing a dendrogram. The dendrogram should clearly show the order in which the clusters are merged.



	p1	p2	p3	p4	p5
p1	1.00	0.10	0.41	0.55	0.35
p2	0.10	1.00	0.64	0.47	0.98
p3	0.41	0.64	1.00	0.44	0.85
p4	0.55	0.47	0.44	1.00	0.76
p5	0.35	0.98	0.85	0.76	1.00

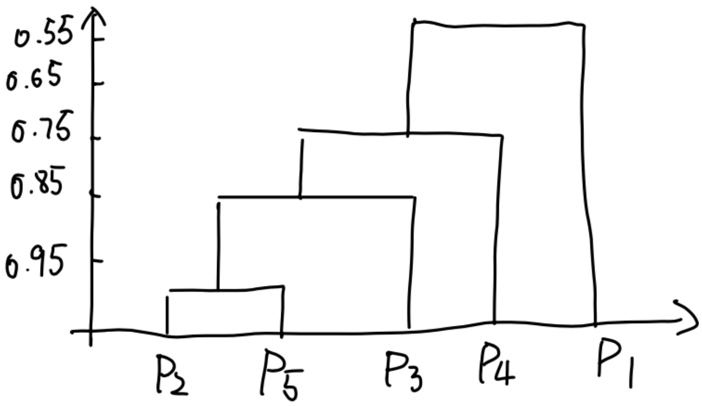
Table 2: Similarity matrix for Q2

S_0	1	2	3	4	5
1	1	0.1	0.41	0.55	0.35
2	0.1	1	0.64	0.47	0.98
3	0.41	0.64	1	0.44	0.85
4	0.55	0.47	0.44	1	0.76
5	0.35	0.98	0.85	0.76	1

S_1	1	2,5	3	4
1	1	0.35	0.41	0.55
2,5	0.35	1	0.85	0.76
3	0.41	0.85	1	0.44
4	0.55	0.76	0.44	1

S_2	1	2,5,3	4
1	1	0.41	0.55
2,5,3	0.41	1	0.76
4	0.55	0.76	1

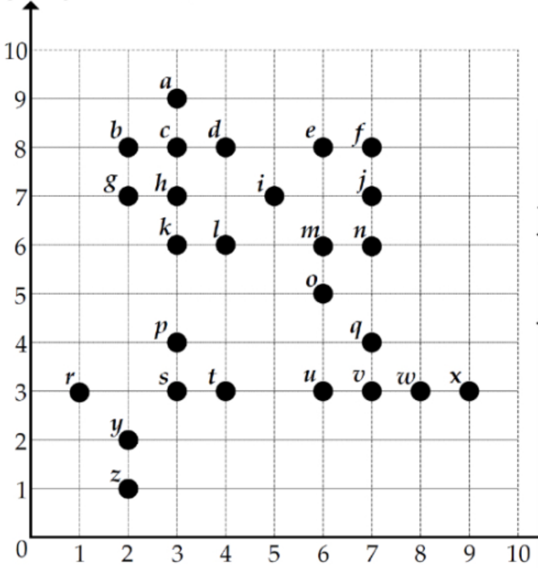
S_3	1	2,5,3,4
1	1	0.55
2,5,3,4	0.55	1



Q3 [30 Marks]

Apply DBSCAN with parameters MinPts=4 and Eps = √2 to get clustering results. **First**, for every data point, answer if it is a core, a border, or an outlier. **Second**, for data points that are not outliers, show the clusters detected. **Third**,

show your detailed steps of DBSCAN process, including the content of the queue you maintain, whenever a new core is found.



Answer:

First: Second:

	core	border	outlier	cluster
a	✓			1
b	✓			1
c	✓			1
d	✓			1
e	✓			1
f		✓		1
g	✓			1
h	✓			1
I	✓			1
J	✓			1
K	✓			1
L	✓			1
M	✓			1
N	✓			1
O	✓			1
P		✓		2
Q	✓			1
R		✓		2
S	✓			2
T		✓		2
U		✓		1
V	✓			1
W	✓			1
X		✓		1
Y	✓			2
Z		✓		2

Third: detail

process a

cluster 1: {a,b,c,d}

Queue: b,c,d

process: b

cluster 1: {a,b,c,d,g,h}

Queue: c,d,g,h

process: c

cluster 1: {a,b,c,d,g,h}

process: d

cluster 1: {a,b,c,d,g,h,i}

Queue: g,h,i

process: g

cluster 1: {a,b,c,d,g,h,i,k}

Queue: g,h,i,k

process: h

cluster 1: {a,b,c,d,g,h,i,k,l}

Queue: i,k,l

process: i

cluster 1: {a,b,c,d,g,h,i,k,l,e,m}

Queue: k,l,e,m

process: k

process: l

process: e

cluster 1: {a,b,c,d,g,h,i,k,l,e,m,f,j}

Queue: m, f,j

process: m

cluster 1: {a,b,c,d,g,h,i,k,l,e,m,f,j,n,o}

Queue: f,j,n,o

process: f

process: j

process: n

process: o

cluster 1: {a,b,c,d,g,h,i,k,l,e,m,f,j,n,o,q}

Queue: q

process: q

cluster 1= {a,b,c,d,g,h,i,k,l,e,m,f,j,n,o,q,u,v,w}

Queue: u,v,w

process: u

process: v.

process: w

cluster 1: {a,b,c,d,g,h,i,k,l,e,m,f,j,n,o,q,u,v,w,x}

Queue: x

process: x.

(Queue empty).

process: P

process: r

process: S

cluster 2: {P,S,t,y}

Queue: t,y

process: t

process: y

cluster 2: {P,S,t,y,r,z}

Queue: r,z

process: r

process: z.

Result:

Cluster 1: {a,b,c,d,e,f,g,h,i,j,k, l, m,n, o, q, u,v,w,x}

cluster 2: {p, r, s, t, y, z}

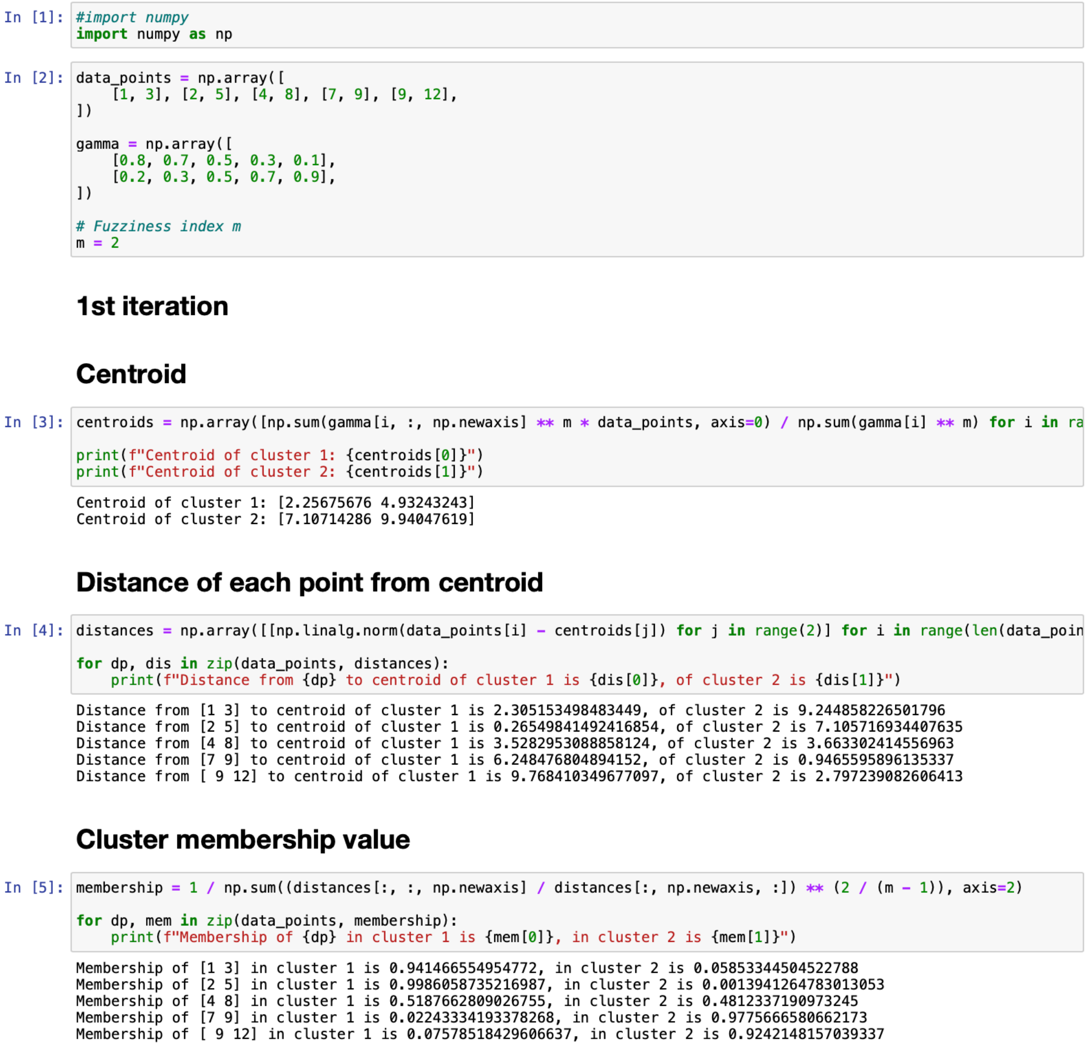
Q4 [20 Marks] Fuzzy Cluster

Assume there are 2 clusters in which the data is to be divided, initializing the data point randomly. Each data point lies in both clusters with some membership value which can be assumed anything in the initial state.

The table below represents the values of the data points along with their membership (gamma) in each cluster.

Cluster	(1,3)	(2,5)	(4,8)	(7,9)	(9,12)
1)	0.8	0.7	0.5	0.3	0.2
2)	0.1	0.3	0.5	0.7	0.9

Please work out the centroids, the distance of each point from centroid, and the cluster membership value.



```
In [1]: #import numpy
        import numpy as np
```

```
In [2]: data_points = np.array([
            [1, 3], [2, 5], [4, 8], [7, 9], [9, 12],
        ])

        gamma = np.array([
            [0.8, 0.7, 0.5, 0.3, 0.1],
            [0.2, 0.3, 0.5, 0.7, 0.9],
        ])

        # Fuzziness index m
        m = 2
```

1st iteration

Centroid

```
In [3]: centroids = np.array([np.sum(gamma[i, :, np.newaxis] ** m * data_points, axis=0) / np.sum(gamma[i] ** m) for i in ra

        print(f"Centroid of cluster 1: {centroids[0]}")
        print(f"Centroid of cluster 2: {centroids[1]}")
```

```
Centroid of cluster 1: [2.25675676 4.93243243]
Centroid of cluster 2: [7.10714286 9.94047619]
```

Distance of each point from centroid

```
In [4]: distances = np.array([[np.linalg.norm(data_points[i] - centroids[j]) for j in range(2)] for i in range(len(data_poin

        for dp, dis in zip(data_points, distances):
            print(f"Distance from {dp} to centroid of cluster 1 is {dis[0]}, of cluster 2 is {dis[1]}")
```

```
Distance from [1 3] to centroid of cluster 1 is 2.305153498483449, of cluster 2 is 9.244858226501796
Distance from [2 5] to centroid of cluster 1 is 0.26549841492416854, of cluster 2 is 7.105716934407635
Distance from [4 8] to centroid of cluster 1 is 3.5282953088858124, of cluster 2 is 3.663302414556963
Distance from [7 9] to centroid of cluster 1 is 6.248476804894152, of cluster 2 is 0.9465595896135337
Distance from [ 9 12] to centroid of cluster 1 is 9.768410349677097, of cluster 2 is 2.797239082606413
```

Cluster membership value

```
In [5]: membership = 1 / np.sum((distances[:, :, np.newaxis] / distances[:, np.newaxis, :]) ** (2 / (m - 1)), axis=2)

        for dp, mem in zip(data_points, membership):
            print(f"Membership of {dp} in cluster 1 is {mem[0]}, in cluster 2 is {mem[1]}")
```

```
Membership of [1 3] in cluster 1 is 0.941466554954772, in cluster 2 is 0.05853344504522788
Membership of [2 5] in cluster 1 is 0.9986058735216987, in cluster 2 is 0.0013941264783013053
Membership of [4 8] in cluster 1 is 0.5187662809026755, in cluster 2 is 0.4812337190973245
Membership of [7 9] in cluster 1 is 0.02243334193378268, in cluster 2 is 0.9775666580662173
Membership of [ 9 12] in cluster 1 is 0.07578518429606637, in cluster 2 is 0.9242148157039337
```

2nd iteration

In [6]:
```
membership[:, 0]
```

Out[6]:
```
array([0.94146655, 0.99860587, 0.51876628, 0.02243334, 0.07578518])
```

In [7]:
```
gamma = np.array([membership[:, 0], membership[:, 1]])

gamma
```

Out[7]:
```
array([[0.94146655, 0.99860587, 0.51876628, 0.02243334, 0.07578518],
       [0.05853345, 0.00139413, 0.48123372, 0.97756666, 0.92421482]])
```

In [8]:
```
print("2nd iteration result:\n")

centroids = np.array([np.sum(gamma[i, :, np.newaxis] ** m * data_points, axis=0) / np.sum(gamma[i] ** m) for i in ra

print(f"Centroid of cluster 1: {centroids[0]}")
print(f"Centroid of cluster 2: {centroids[1]}\n")

distances = np.array([[np.linalg.norm(data_points[i] - centroids[j]) for j in range(2)] for i in range(len(data_poin

for dp, dis in zip(data_points, distances):
    print(f"Distance from {dp} to centroid of cluster 1 is {dis[0]}, of cluster 2 is {dis[1]}")

print()

membership = 1 / np.sum((distances[:, :, np.newaxis] / distances[:, np.newaxis, :]) ** (2 / (m - 1)), axis=2)

for dp, mem in zip(data_points, membership):
    print(f"Membership of {dp} in cluster 1 is {mem[0]}, in cluster 2 is {mem[1]}")
```

```
2nd iteration result:

Centroid of cluster 1: [1.85854048 4.57240718]
Centroid of cluster 2: [ 7.48562706 10.12986197]

Distance from [1 3] to centroid of cluster 1 is 1.7915234069386294, of cluster 2 is 9.638375906828829
Distance from [2 5] to centroid of cluster 1 is 0.4503847359639798, of cluster 2 is 7.51049852587616
Distance from [4 8] to centroid of cluster 1 is 4.041564222520601, of cluster 2 is 4.0848388005237295
Distance from [7 9] to centroid of cluster 1 is 6.785144367450272, of cluster 2 is 1.2298055621513753
Distance from [ 9 12] to centroid of cluster 1 is 10.30386233611025, of cluster 2 is 2.4063959856546133

Membership of [1 3] in cluster 1 is 0.9666046400662307, in cluster 2 is 0.03339535993376918
Membership of [2 5] in cluster 1 is 0.9964168017412698, in cluster 2 is 0.0035831982587302144
Membership of [4 8] in cluster 1 is 0.5053250313519356, in cluster 2 is 0.4946749686480643
Membership of [7 9] in cluster 1 is 0.03180657102057524, in cluster 2 is 0.9681934289794247
Membership of [ 9 12] in cluster 1 is 0.051721374184511, in cluster 2 is 0.9482786258154889
```

3rd iteration

In [9]:
```
gamma = np.array([membership[:, 0], membership[:, 1]])

gamma
```

Out[9]:
```
array([[0.96660464, 0.9964168 , 0.50532503, 0.03180657, 0.05172137],
       [0.03339536, 0.0035832 , 0.49467497, 0.96819343, 0.94827863]])
```

In [10]:
```
print("3rd iteration result:\n")

centroids = np.array([np.sum(gamma[i, :, np.newaxis] ** m * data_points, axis=0) / np.sum(gamma[i] ** m) for i in ra

print(f"Centroid of cluster 1: {centroids[0]}")
print(f"Centroid of cluster 2: {centroids[1]}\n")

distances = np.array([[np.linalg.norm(data_points[i] - centroids[j]) for j in range(2)] for i in range(len(data_poin

for dp, dis in zip(data_points, distances):
    print(f"Distance from {dp} to centroid of cluster 1 is {dis[0]}, of cluster 2 is {dis[1]}")

print()

membership = 1 / np.sum((distances[:, :, np.newaxis] / distances[:, np.newaxis, :]) ** (2 / (m - 1)), axis=2)

for dp, mem in zip(data_points, membership):
    print(f"Membership of {dp} in cluster 1 is {mem[0]}, in cluster 2 is {mem[1]}")
```

```
3rd iteration result:

Centroid of cluster 1: [1.81711109 4.50607855]
Centroid of cluster 2: [ 7.50785987 10.17469156]

Distance from [1 3] to centroid of cluster 1 is 1.7134594075343288, of cluster 2 is 9.686508100517925
Distance from [2 5] to centroid of cluster 1 is 0.526694170939806, of cluster 2 is 7.55737740076521
Distance from [4 8] to centroid of cluster 1 is 4.119768327143095, of cluster 2 is 4.127270804928556
Distance from [7 9] to centroid of cluster 1 is 6.859859139110664, of cluster 2 is 1.2797741683314656
Distance from [ 9 12] to centroid of cluster 1 is 10.380402291244282, of cluster 2 is 2.35759051608785

Membership of [1 3] in cluster 1 is 0.9696588448063691, in cluster 2 is 0.030341155193631012
Membership of [2 5] in cluster 1 is 0.9951664022240737, in cluster 2 is 0.004833597775926174
Membership of [4 8] in cluster 1 is 0.5009097169852561, in cluster 2 is 0.4990902830147439
Membership of [7 9] in cluster 1 is 0.03363395551519961, in cluster 2 is 0.9663660444848003
Membership of [ 9 12] in cluster 1 is 0.04905290561941839, in cluster 2 is 0.9509470943805817
```