Name: SUN RUI
Student ID: 18083229g

Ans1:

a)

Dissim(0001, 0150) = 1 - $\frac{|\{\text{Romance,Drama}\}|}{|\{\text{Drama,Romance,Sci-Fiction,Mystery,Fiction}\}|}$ = $\frac{3}{5}$ = 0.6

Dissim(0001, 0553) = 1 - $\frac{|\{\text{ Sci-Fiction}\}|}{|\{\text{Romance,Sci-Fiction,Drama,Mystery,Action,Thriller,Horror}\}|}$ = $\frac{6}{7}$ $\approx$ 0.857

Dissim(0001, 1011) = 1 - $\frac{|\{\text{ NULL}\}|}{|\{\text{Romance,Sci-Fiction,Drama,Mystery,Horror,Thriller}\}|}$ = 1

Dissim(0001, 3997) = 1 - $\frac{|\{\text{Sci-Fiction}\}|}{|\{\text{Romance,Sci-Fiction,Drama,Mystery,Action,Crime}\}|}$ = $\frac{5}{6}$ $\approx$ 0.833

Dissim(0150, 0553) = 1 - $\frac{|\{NULL\}|}{|\{\text{Drama,Romance,Fiction,Action,Sci-Fiction,Thriller,Horror}\}|}$ = 1

Dissim(0150, 1011) = 1 - $\frac{|\{NULL\}|}{|\{\text{Drama,Romance,Fiction,Action,Crime,Sci-Fiction}\}|}$ = 1

Dissim(0150, 3997) = 1 - $\frac{|\{NULL\}|}{|\{\text{Drama,Romance,Fiction,Action,Crime,Sci-Fiction}\}|}$ = 1

Dissim(0553, 1011) = 1 - $\frac{|\{\text{Thriller,Horror}\}|}{|\{\text{Action,Sci-Fiction,Thriller,Horror}\}|}$ = $\frac{2}{4}$ = 0.5

Dissim(0553, 3997) = 1 - $\frac{|\{\text{Action,Sci-Fiction}\}|}{|\{\text{Action,Sci-Fiction,Thriller,Horror,Crime}\}|}$ = $\frac{3}{5}$ = 0.6

Dissim(1011, 3997) = 1 - $\frac{|\{\text{NULL}\}|}{|\{\text{Action,Crime,Sci-Fiction,Horror,Thriller}\}|}$ = 1

dissimilarity matrix:

	0001	0150	**0553**	1011	3997
0001	0	-	-	-	-
0150	0.6	0	-	-	-
0553	0.857	1	0	-	-
1011	1	1	**0.5**	0	-
3997	0.833	1	0.6	1	0

b)

merge 0553 and 1011 (0.5), we have:

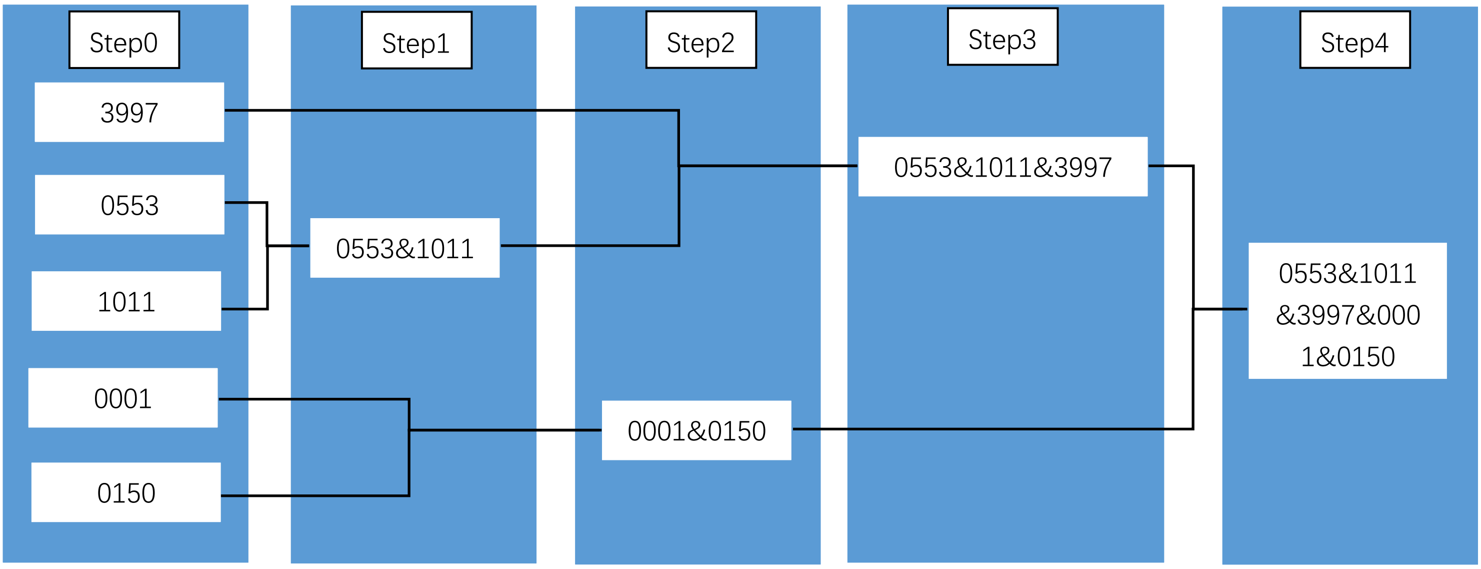
	0553&1011	**0001**	0150	3997
0553&1011	0	-	-	-
0001	1	0	-	-
0150	1	**0.6**	0	-
3997	1	0.833	1	0

merge 0001 and 0150 (0.6), we have:

	0553&1011	0001&0150	3997
0553&1011	0	-	-
0001&0150	1	0	-
3997	**1**	1	0

merge 0553&1011 and 3997 (0.6), we have:

	0001&0150	0553&1011&3997
0001&0150	0	-
0553&1011&3997	1	0



c)

Firstly, store all dissimilarities in a list, each dissimilarity stands for a pair of items;

Secondly, sort the list by ASC;

Thirdly, travel the ascending list:

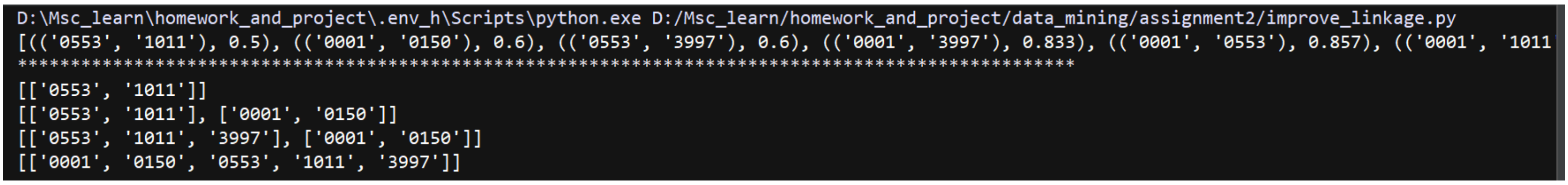
If the pair has not existed in any clusters, then the pair is a new cluster;

If both of two items of a pair have been included in different two clusters, then combine these two clusters;

If both of two items of a pair have been included in a same cluster, do noting;

If one item of a pair has existed in a cluster, but another one is not in any clusters, then put it into the cluster.

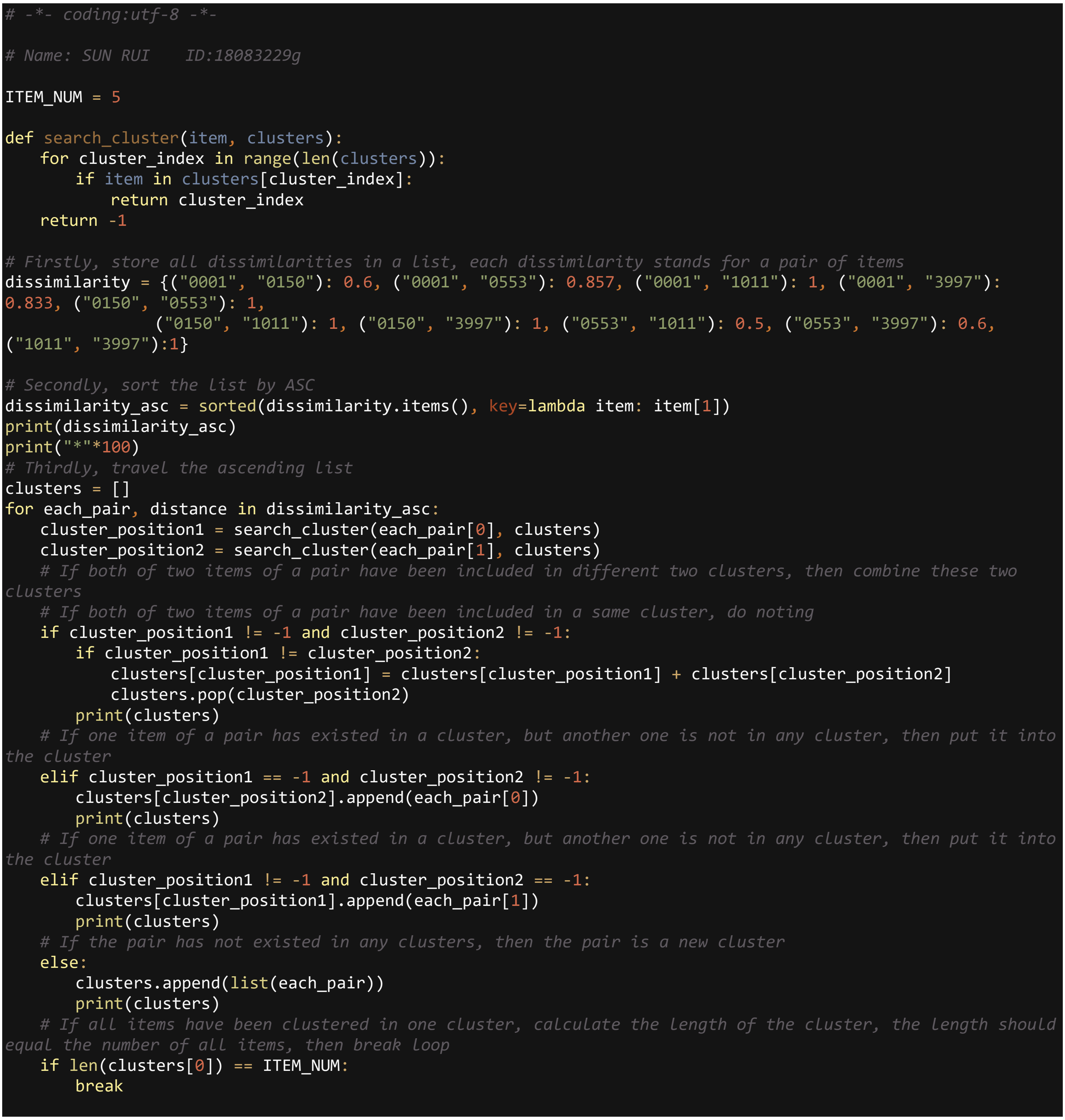
Let us can see a demo by Python:



```
D:\Msc_learn\homework_and_project\.env_h\Scripts\python.exe D:/Msc_learn/homework_and_project/data_mining/assignment2/improve_linkage.py
[(('0553', '1011'), 0.5), (('0001', '0150'), 0.6), (('0553', '3997'), 0.6), (('0001', '3997'), 0.833), (('0001', '0553'), 0.857), (('0001', '1011
*****************************************************************
[['0553', '1011']]
[['0553', '1011'], ['0001', '0150']]
[['0553', '1011', '3997'], ['0001', '0150']]
[['0001', '0150', '0553', '1011', '3997']]
```

The result is same as above result, it can reduce Space Complexity and do not need to update distances between two items.

The code reference:



```
# -*- coding:utf-8 -*-

# Name: SUN RUI    ID:18083229g

ITEM_NUM = 5

def search_cluster(item, clusters):
    for cluster_index in range(len(clusters)):
        if item in clusters[cluster_index]:
            return cluster_index
    return -1

# Firstly, store all dissimilarities in a list, each dissimilarity stands for a pair of items
dissimilarity = {("0001", "0150"): 0.6, ("0001", "0553"): 0.857, ("0001", "1011"): 1, ("0001", "3997"): 0.833, ("0150", "0553"): 1,
                 ("0150", "1011"): 1, ("0150", "3997"): 1, ("0553", "1011"): 0.5, ("0553", "3997"): 0.6, ("1011", "3997"):1}

# Secondly, sort the list by ASC
dissimilarity_asc = sorted(dissimilarity.items(), key=lambda item: item[1])
print(dissimilarity_asc)
print("*"*100)
# Thirdly, travel the ascending list
clusters = []
for each_pair, distance in dissimilarity_asc:
    cluster_position1 = search_cluster(each_pair[0], clusters)
    cluster_position2 = search_cluster(each_pair[1], clusters)
    # If both of two items of a pair have been included in different two clusters, then combine these two clusters
    # If both of two items of a pair have been included in a same cluster, do noting
    if cluster_position1 != -1 and cluster_position2 != -1:
        if cluster_position1 != cluster_position2:
            clusters[cluster_position1] = clusters[cluster_position1] + clusters[cluster_position2]
            clusters.pop(cluster_position2)
        print(clusters)
    # If one item of a pair has existed in a cluster, but another one is not in any cluster, then put it into the cluster
    elif cluster_position1 == -1 and cluster_position2 != -1:
        clusters[cluster_position2].append(each_pair[0])
        print(clusters)
    # If one item of a pair has existed in a cluster, but another one is not in any cluster, then put it into the cluster
    elif cluster_position1 != -1 and cluster_position2 == -1:
        clusters[cluster_position1].append(each_pair[1])
        print(clusters)
    # If the pair has not existed in any clusters, then the pair is a new cluster
    else:
        clusters.append(list(each_pair))
        print(clusters)
    # If all items have been clustered in one cluster, calculate the length of the cluster, the length should equal the number of all items, then break loop
    if len(clusters[0]) == ITEM_NUM:
        break
```

Ans2

a)

	P1	P2	P3	P4	P5	P6	P7	P8
P1	0							
P2	4	0						
P3	8.49	6.32	0					
P4	3.61	3.61	5	0				
P5	7.81	5.39	1	4.47	0			
P6	7.21	4.47	2	4.12	1	0		
P7	8.06	4.12	7.28	7.21	6.32	5.39	0	
P8	2.24	3.61	6.40	1.41	5.83	5.39	7.62	0

b)

initial centroids:

	Group 1	Group 2	Group 3
Record	P1	P4	P7
Cluster Mean	(2,10)	(5,8)	(1,2)

Calculate distances to cluster mean:

	Group 1	Group 2	Group 3
	Distance to P1	Distance to P4	Distance to P7
P1	**0**	-	-
P2	4	**3.61**	4.12
P3	8.49	**5**	7.28
P4	-	**0**	-
P5	7.81	**4.47**	6.32
P6	7.21	**4.12**	5.39
P7	-	-	**0**
P8	2.24	**1.41**	7.62

New centroids:

	Group 1	Group 2	Group 3
Record	P1	Mean of (P2, P3, P4, P5, P6, P8)	P7
Cluster Mean	(2, 10)	(5.33, 5.833)	(1, 2)

i) the new cluster: C1={P1}, C2={P2, P3, P4, P5, P6, P8}, C3={P7}

ii) The centroids of the new clusters: (2, 10) of C1, (5.33, 5.833) of C2, (1, 2) of C3

Ans3

a)

P(Activist)=2/6, P(Follower)=2/6, P(Superstar)=2/6

A1		
P(Many|Activist)=1/2	P(Many|Follower)=0/2	P(Many|Superstar)=2/2
P(Few|Activist)=1/2	P(Few|Follower)=2/2	P(Few|Superstar)=0/2
A2		
P(Many|Activist)=1/2	P(Many|Follower)=2/2	P(Many|Superstar)=1/2
P(Few|Activist)=1/2	P(Few|Follower)=0/2	P(Few|Superstar)=1/2
A3		
P(High|Activist)=1/2	P(High|Follower)=2/2	P(High|Superstar)=0/2
P(Low|Activist)=1/2	P(Low|Follower)=0/2	P(Low|Superstar)=2/2

A	P(X|Activist)P(Activist)=0.042 P(X|Follower)P(Follower)=0 **P(X|Superstar)P(Superstar)=0.167**	Target: Activist Predict: Superstar
B	P(X|Activist)P(Activist)=0.042 **P(X|Follower)P(Follower)=0.333** P(X|Superstar)P(Superstar)=0	Target: Activist Predict: Follower
C	P(X|Activist)P(Activist)=0.042 **P(X|Follower)P(Follower)=0.333** P(X|Superstar)P(Superstar)=0	Target: Follower Predict: Follower
D	P(X|Activist)P(Activist)=0.042 P(X|Follower)P(Follower)=0 **P(X|Superstar)P(Superstar)=0.167**	Target: Superstar Predict: Superstar
E	P(X|Activist)P(Activist)=0.042 P(X|Follower)P(Follower)=0 **P(X|Superstar)P(Superstar)=0.167**	Target: Superstar Predict: Superstar
F	P(X|Activist)P(Activist)=0.042 **P(X|Follower)P(Follower)=0.333** P(X|Superstar)P(Superstar)=0	Target: Follower Predict: Follower

So, the classification rate is 4/6

b)

G	Activist	1/2*1/2*P(A3|Activist)*2/6=1/12*P(A3|Activist)	A3=High	1/24
			A3=Low	1/24
	Follower	0*0*P(A3|Follower)*2/6=0*P(A3|Follower)	A3=High	0
			A3=Low	0
	Superstar	2/2*1/2*P(A3|Superstar)*2/6=1/6*P(A3|Superstar)	A3=High	0
			A3=Low	**1/6**

H	Activist	P(A1|Activist)*1/2*1/2*2/6=1/12*P(A1|Activist)	A1=Many	1/24
			A1=Few	1/24
	Follower	P(A1|Follower)*2/2*2/2*2/6=1/3*P(A1|Follower)	A1=Many	**1/6**
			A1=Few	**1/6**
	Superstar	P(A1|Superstar)*1/2*0*2/6=0*P(A1|Superstar)	A1=Many	0
			A1=Few	0

According to above table:
User G can be classified to Superstar with the largest probability 1/6 as A3=Low.
User H can be classified to Follower with the largest probability 1/6 as A1= Many or Few.