Name : Junyu Ren Student ID: z5195715

COMP 9318 Data Warehousing and Data Mining

Written Assignment 1

Q1

1) Data cube shows as follows:

Location	Time	Item	SUM(Quality)
Sydney	2005	PS2	1400
Sydney	2005	ALL	1400
Sydney	2006	PS2	1500
Sydney	2006	Wii	500
Sydney	2006	ALL	2000
Sydney	ALL	PS2	2900
Sydney	ALL	Wii	500
Sydney	ALL	ALL	3400
Melbourne	2005	XBox360	1700
Melbourne	2005	ALL	1700
Melbourne	ALL	XBox360	1700
Melbourne	ALL	ALL	1700
ALL	2005	PS2	1400
ALL	2006	PS2	1500
ALL	2006	Wii	500
ALL	2005	XBox360	1700
ALL	2005	ALL	3100
ALL	2006	ALL	2000
ALL	ALL	PS2	2900
ALL	ALL	Wii	500
ALL	ALL	XBox360	1700
ALL	ALL	ALL	5100

2) SQL statement shows as follows:

select Location, Time, Item, SUM(Quantity) from Sales group by Location, Time, Item union all
select Location, Time, ALL, SUM(Quantity) from Sales group by Location, Time union all
select Location, ALL, Item, SUM(Quantity) from Sales group by Location, Item union all
select ALL, Time, Item, SUM(Quantity) from Sales group by Time, Item union all
select Location, ALL, ALL, SUM(Quantity) from Sales group by Location union all
select ALL, Time, ALL, SUM(Quantity) from Sales group by Time union all
select ALL, ALL, Item, SUM(Quantity) from Sales group by Item union all
select ALL, ALL, ALL, SUM(Quantity) from Sales

3) Result shows as follows:

Location	Time	Item	SUM(Quantity)
Sydney	2006	ALL	2000
Sydney	ALL	PS2	2900
Sydney	ALL	ALL	3400
ALL	2005	ALL	3100
ALL	2006	ALL	2000
ALL	ALL	PS2	2900
ALL	ALL	ALL	5100

4) Map function is Index = 3 * 4 * Location + 4 * Time + Item = 12 * Location + 4 * Time + Item. (12, 4, 1) is the smallest coefficient group that Location, Time, Item can multiply, which ensures index and value are 1 to 1.

Tabular form with full data using mapping function $f_{\text{Location}}(x)$, $f_{\text{Time}}(x)$ and $f_{\text{Item}}(x)$ shows as follows:

ArrayIndex	Location	Time	Item	SUM(Quantity)
17	1	1	1	1400
16	1	1	0	1400
21	1	2	1	1500
23	1	2	3	500
20	1	2	0	2000
13	1	0	1	2900
15	1	0	3	500
12	1	0	0	3400
30	2	1	2	1700
28	2	1	0	1700
26	2	0	2	1700
24	2	0	0	1700
5	0	1	1	1400
9	0	2	1	1500
11	0	2	3	500
6	0	1	2	1700
4	0	1	0	3100
8	0	2	0	2000
1	0	0	1	2900
3	0	0	3	500
2	0	0	2	1700
0	0	0	0	5100

(ArrayIndex, Value) tabular form shows as follows with ArrayIndex sorted ascending:

ArrayIndex	Value
0	5100
1	2900
2	1700
3	500
4	3100
5	1400
6	1700
8	2000
9	1500
11	500
12	3400
13	2900
15	500
16	1400
17	1400
20	2000
21	1500
23	500
24	1700
26	1700
28	1700
30	1700

Q2

Step1: p2 and p5's similarity is 0.98, which is biggest among all values.So, put p2 and p5 together firstly.

Step2: recompute all similarity between p1,p3,p4,and p25 using group average.

Sim((2,5),1)=(0.1+0.35+0.98)/3 = 0.477

Sim((2,5),3)=(0.64+0.98+0.85)/3 = 0.823

Sim((2,5),4)=(0.47+0.98+0.76)/3 = 0.737

Similarity matrix update as follows:

	p_1	p_3	p_4	p_{25}
p_1	1.00	0.41	0.55	0.477
p_3	0.41	1.00	0.44	0.823
p_4	0.55	0.44	1.00	0.737
p_{25}	0.477	0.823	0.737	1.00

Step3: select current biggest value 0.823 and put p3 together with p25.

Step4: recompute all similarity between p1, p4 and p325 using group average.

Sim((3,2,5),1)=(0.41+0.1+0.35+0.64+0.98+0.85)/6 = 0.555

Sim((3,2,5),4)=(0.64+0.47+0.98+0.44+0.85+0.76)/6 = 0.69

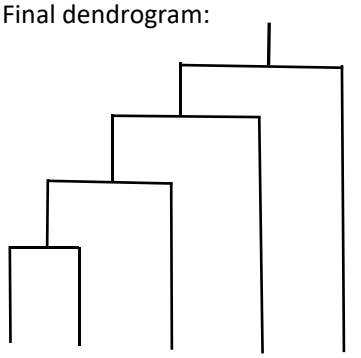
Similarity matrix update as follows:

	p_1	p_4	p_{325}
p_1	1.00	0.55	0.555
p_4	0.55	1.00	0.690
p_{325}	0.555	0.69	1.00

Step5: select current biggest value 0.690 and put p4 with p325 together.

Step6: lastly, only p1 remains alone. Put p1 together with p4325.

Final dendrogram:



Q3

1) After line 7 insert :

```
if previous_G = G:
    canStop ⟵ True
else:
    previous_G ⟵ G
```

2)

Denote initial total cost as **cost0**, total cost after executing line 5-7 as **cost1**, total cost after executing line 8-9 as **cost2**.

One iteration contains 2 for-loops.

The first loop reassign each point into a group whose central point is nearest to this point.For each group,since the central point don't change in this loop,the dist(p,ci) never increase,so cost(gi) never increase and cost(g1,g2,...,gk) never increase. We can denote as **cost1<= cost0**.

The second loop replace old central points by newly-computed ones in each group.For each group,the newly-calculated central point makes dist(p,ci) never increase,so cost(gi) never

increase and cost(g1,g2,...,gk) never increase. We can denote as **cost2 <= cost1**.

To concluded, **cost2 <= cost1 <= cost0** $\longrightarrow$ **cost2 <= cost0**, which means cost of k clusters at the end of each iteration never increases.cost2 = cost0 only holds when initial central point are optimal, no point goes to another group in loop 1 and newly-calculated central points are the same as old ones in loop 2.But it only has small possibility to happen.

3)

Firstly, It is a NP-hard problem to find global optimal of k-means algorithm.At most it will generate $k^n/k!$ clusters, which is a finite value.That means cost will decrease for finite times to reach the global optimal and the while-loop stops.So, it will always converge to a particular value.

Secondly, As for why it will get a local minima instead of global minima, k-means will randomly select initial points and before the algorithm starts, no one knows the value of k.Different value of k will have different converge cost.The cost is minimal under specified condition and may not the smallest from global view.When condition changes, a smaller cost may be generate.So,it is a local minima.

In conclusion, cost of clusters from k-means will always converge to a local minima.