Question 1

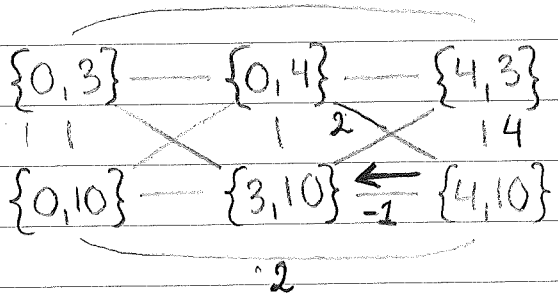
a) i. A node contains k objects and represent a proposal of the cluster medoids.

ii. Two nodes are neighbors when they differ with exactly one object. A node has $k(n-k)$ neighbors

iii. PAM and CLARANS are the algorithms with the graphs containing the most nodes. They search the same graph with the difference that PAM checks all neighbors of a node while CLARANS checks only the set maximum number of neighbors. CLARA takes samples from the data to search and therefore the graph contains fewer nodes.

iv. None of the algorithms promise a global optimum, but PAM guarantees to find a local optimum (which the two others don't) 2

b)



The node {4,10} is chosen as starting point.

Swapping costs to it's neighbors are calculated with $TC_{ih} = \sum C_{ijh} = \sum_j (d(x_h, x_j) - d(x_i, x_j))$. They can be seen in the graph.

The algoritm then moves on to the neighbor with lowest swapping cost (as long as it's negative). If there are only positive TC_{ih} then the algorithm stops. 3

In this case the algorithm moves on to {3,10}.

Question 2

a) i. The BIRCH algorithm consists of two steps:

1 Build a CF-tree → how?

2 Use a cluster algorithm of your choise on the sub clusters in the leaf nodes.

ii. Cluster Feature Vector contains information about the cluster: n = number of observations

LS = sum of the observations

SS = sum of the squared observations

For the case given, the CF is:

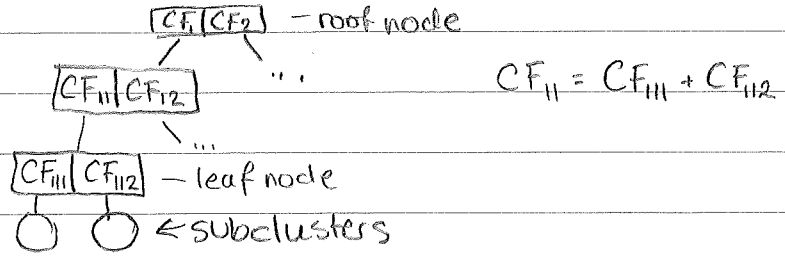
$n = 3$

$LS = (1+1+2 = 4,\ 2+3+2 = 7)$

$SS = (1+1+4 = 6,\ 4+9+4 = 17)$

$CF = (3, (4,7), (6,17))$

iii. The CF tree is a height balanced tree. It consists of a root-node, middle level nodes and leaf nodes. The leaf nodes contains information on the data points in the subclusters with the help of CFs. The leaf-nodes are connected to parent nodes wich holds information about their children (with combined CFs). In the end the root node contains information about all the data points by combining it's childrens CFs. An example:



$CF_{11} = CF_{111} + CF_{112}$

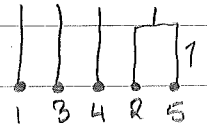
iv. Branching factor specifies how many children a parent can have. Threshold states the max. allowed diameter of subclusters

Question 2

b) All clusters start out as individual clusters. The two clusters that are the least dissimilar (according to complete link) are joined together in each step. In the end, all data points form a single cluster.

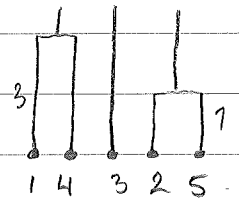
Step 1:

	1	2	3	4	5
1	0				
2	5	0			
3	9	10	0		
4	3	2	6	0	
5	7	(1)	4	8	0



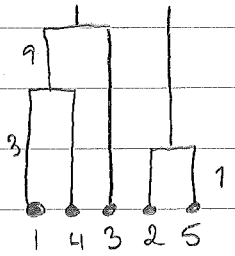
Step 2:

	1	3 4 2,5
1	0	
3	9	0
4	(3)	6 0
2,5	7	10 8 0



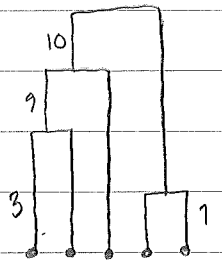
Step 3:

	1,4	3	2,5
1,4	0		
3	(9)	0	
2,5	(8)	10	0



Step 4:

	1,3,4	2,5
1,34	0	
2,5	(10)	0



Question 3

A core point is a data point that has at least min.point of data points within its neighborhood, defined by radius ε. ε and min.point is set by the user. A point p is directly density-reachable from point q if q is a core point and p lies inside it's ε-neighborhood. The point p is density reachable from q if there exists a chain of points (q as start and p as end) where each point is directly density-reachable from the point one step previous in the chain. The points p and o are density-connected if there exists a point q from wich both p and o are density-reachable. In DBSCAN a cluster is formed by density-connected data points. If a point isn't density-connected to any other points it's considered to be an outlier since it's not placed in a dense enough region.

Question 4

a) To compute the distance between K and L the following formula is used:

$$d_{K,L} = \frac{\sum_{f=1}^{p} \delta_1^f \times d_{KL}^f}{\sum_{f=1}^{p} \delta^f}$$

where d_{KL}^f is the distance between the objects for attribute f and $\delta^f = 0$ if f has missing value or f is asymmetric binary and is of case 0,0. Otherwise $\delta^f = 1$

$d_{KL}^A = \sqrt{(10-10)^2 + (500-505)^2} = 5$ $\quad d_{KL}^B = |2-1| + |1-3| + |1-1| = 3$

$d_{KL}^C = 0$ $\quad d_{KL}^D = 1$ $\quad d_{KL}^E = 1$ $\quad d_{KL}^F = 0$

$$d_{KL} = \frac{1\times 5 + 1\times 3 + 1\times 0 + 1\times 1 + 1\times 1 + 0\times 0 + 0\times d_{KL}^G}{5\times 1} = \frac{10}{5} = 2$$

2

b) Create ranking for the observations $r = \{r_1, \ldots, M_r\}$ where M_r is the highest rank and r_1 is the lowest rank. Then you need to normalize the ranks by

$$Z_i = \frac{r_i - 1}{M_r - 1}$$

An interval-based measure can now be used on the normalized z-values.

1

Question 5

a) min.sup = 1

C_1

Item	S
A	6
B	3
C	2
D	3

→ L_1

L_1
A
B
C
D

↑ output

Candidates of size 1 check support

C_2

Items	S
AB	3
AC	2
AD	3
BC	1
~~BD~~	~~0~~
CD	1

→ L_2

L_2
AB
AC
AD
BC
CD

↑ output

Candidates size 2 created by self join (joined by prefix) check support

C_3

Items	S
ABC	1
~~ABD~~	~~—~~
ACD	1

→ L_3

L_3
ABC
ACD

↑ output

Candidates size 3 created by self join prune itemsets that have a subset not included in L_2. Check support

Question 5

b)

C_1

Item	C	S
A	✓	6
B	✓	3
C	✓	–
D	✓	–

→ L_1

A
B
C
D

→ C_2

Item	C	S
AB	✓	3
AC	✓	2
AD	✓	3
BC	✓	1
~~BD~~	~~✓~~	~~0~~ too low support
~~CD~~	~~✗~~	~~–~~ constraint not fullfilled

→ L_2

AB
AC
AD
BC

The constraint is antimonotone!

C_3

Item	C	S
ABC	✓	1
~~ABD~~	~~–~~	~~–~~
~~ACD~~	~~–~~	~~–~~

} subsets not in L_2

→ L_3

ABC

2

The constraint is checked after the controll that all subsets of the candidates are frequent (is part of L_{k-1}) and before checking support. If the constraint is false for an itemset this itemset is pruned and there is no need to check support for this itemset.

c) min.conf ≥ 0.5

1

$AB \rightarrow C$ $\quad c = \frac{Sup(ABC)}{Sup(AB)} = \frac{1}{3}$ ✗

$AC \rightarrow B$ $\quad c = \frac{Sup(ABC)}{Sup(AC)} = \frac{1}{2}$ ✓

$BC \rightarrow A$ $\quad c = \frac{Sup(ABC)}{Sup(BC)} = \frac{1}{1}$ ✓

Because this rule has too low confidence we don't need to check the rules $A \rightarrow BC$ and $B \rightarrow AC$

$C \rightarrow AB$ $\quad c = \frac{sup(ABC)}{sup(C)} = \frac{1}{2}$ ✓

Question 6

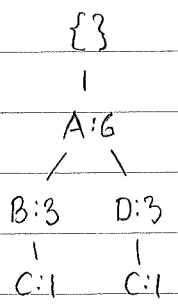
a) Sort the items after support and remove those that aren't frequent (sup < min.sup = 1). Reorder transactions.

Item	A	B	D	C
Sup	6	3	3	2

IN THE OUTPUT

TID	Items
1	ABC
2	ADC
3	AB
4	AB
5	AD
6	AD

Build the FP-tree based on the reordered transactions



Item	
A	-
B	A:3
D	A:3
C	AB:1, AD:1

Repeat the steps above for the conditional data bases

A-conditional is empty, no need to check

B-conditional

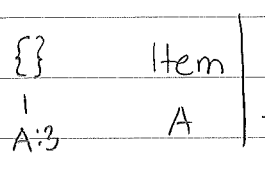
A, A, A

Item A

Sup 3

↓

OUTPUT: AB



AB-cond empty, no need to check further

(continue question 6a)

D-conditional

A, A, A

Item A

Sup 3

↓

OUTPUT: AD

{}
|
A:3

Item	
A	–

AD-cond empty, no need to check

C-conditional

AB, AD

Item A B D } ORDER AFTER SUPPORT

Sup 2 1 1 }

↓

OUTPUT: AC, BC, DC

{}
A:2
/ \
B:1 D:1

Item	
A	–
B	A:1
D	A:1

BC-conditional

A

Item A

Sup 1

↓

Output: ABC

{}
|
A:1

DC-conditional

A

Item A

Sup 1

↓

Output: ADC

{}
|
A:1

2

b) Antimonotone constraints are incorporated the same way support checking is. If an item/itemset doesn't fullfill the constraint it is pruned before the reordering of the items in the data base and is thus not part of any resulting tree or conditional data base. 0 / 2 missing place)

Monotone are checked when support is checked. If not fullfilled the item/itemset is removed from output but not from tree or data base. If the constraint is true for an item/itemset you needn't check it for the conditional data bases that follow that item/itemset. 0

Question 6

c) The main advantage of FP grow over Apriori algorithm is that it doesn't produce candidates and thus saves time (especially for low minimum support) and storage. In addition it only need to scan the original data base once, then it can be discarded (unlike in Apriori algorithm)

Question 7

a) A convertible monotone constraint is $avg(S) \geq p$ where S is the prices of the itemset and items are orclered after acending price (item with lowest price first). Adding a new item with higher price can only increase average price and that mean that the constraint is convertible monotone.

A convertible antimonotone constraint is $avg(S) \geq p$ when the items are ordered after decending price. Adding an item with lower price than previous items can only decrease average price and the constraint is convertible monotone.

In both cases it can be shown that the constraint $avg(S) \geq p$ is neither monotone nor antimonotone since adding an item (when there is no specific order) can both increase and decrease the price - we can't know.

3