Q1: (**FP-growth**) FP-growth algorithm to find all frequent pattern with minimum support = 3. To verify your work, there are 14 of them. Show your steps.

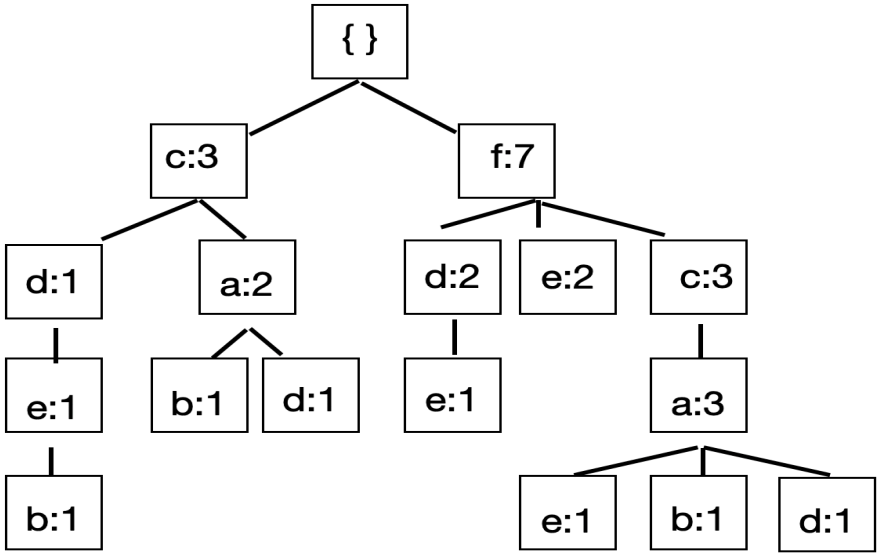
Scan DB once, find frequent 1-itemset and sort it:

Item	frequency head
f	7
c	6
a	5
d	5
e	5
b	3

F-list = f-c-a-d-e-b

TID	Items	ordered
1	a,b,c	c,a,b
2	a,c,d	c,a,d
3	d,e,f	f,d,e
4	e, f	f,e
5	a,c,e,f	f,c,a,e
6	a,b,c,f	f,c,a,b
7	b,c,d,e	c,d,e,b
8	e,f	f,e
9	a,c,d,f	f,c,a,b
10	d,f	f,d

Scan DB again, construct FP-tree:



Find frequent itemsets:

Patterns containing	Conditional pattern base	Frequent
f	null	{f}

c	f:3	{c, fc}
a	c:2, fc:3	{a, ca, fa, fca}
d	c:1, ca:1, f:2, fca:1,	{d, cd, fd}
e	cd:1, fd:1, f:2	{e, fe}
b	cde:1, ca:1, fca:1,	{b, cb}

There are 14 frequent patterns with minimum support = 3:
{f, c, a, d, e, b, fc, ca, fa, cd, fd, fe, cb, fca}.

Q2: (**Apriori**) Find all frequent 3-itemsets **candidates** using Apriori algorithm with alternate $F_{k-1} \times F_{k-1}$Fk−1×Fk−1 Method mentioned page 45 of the chap 4 slides. To save your work, assume we have found all 2-itemsets already (you are allowed to reuse the result found in Q1). Then, perform candidate pruning over your result.

According to Q1:
F_2 = {fc, ca, fa, cd, fd, fe, cb} is the set of frequent 2-itemsets.
Merge each of them from F_2 to generate the set of candidates 3-itemset:
L_3 = {fca, fcd, fda, fec, fcb, cad, cab, fae, cdb, fde}
Candidate pruning:
Prune {fda} because (da) is infrequent.
Prune {fec}because {ec} is infrequent.
Prune {fcb} because {fb} is infrequent.
Prune {cad} because {ad} is infrequent.
Prune {cab} because {ab} is infrequent
Prune {fae} because {ae} is infrequent.
Prune {cdb} because {cd} is infrequent.
Prune {fde} because {de} is infrequent.
Therefore, after candidates pruning: **candidate 3-itemsets**: L_3 = {fca, fcd}
Support counting:
Count the support by scanning the DB: {fca:3}, {fcd:1}
Candidate elimination
Eliminate candidates {fcd}
Therefore, frequent 3-itemsets is{fca}

Q3: (**Min-Hash**) Given the set of shingles {A,B,C,D,E,F,G,H} and the following three documents D_1,D_2,D_3, compute the MinHash for them against each of the permutation p_1,p_2,p_3 Calculate the Jaccard similarity between these documents and the similarity of MinHash of these documents.

Documents	D_1	D_2	D_3
Shingle	{B,D,F,H}	{A,B,H}	{E,F}

Documents	D_1	D_2	D_3
A	0	1	0
B	1	1	0
C	0	0	0
D	1	0	0
E	0	0	1
F	1	0	1
G	0	0	0
H	1	1	0

Permutation	Order
P_1	BDEFHGAC
P_2	CDEFABHG
P_3	ACBDGFEH

As for p_1, BDEFHGAC,

p_1	D_1	D_2	D_3
B (1)	1	1	0
D (2)	1	0	0
E (3)	0	0	1
F (4)	1	0	1
H (5)	1	1	0
G (6)	0	0	0
A (7)	0	1	0
C (8)	0	0	0

As for p_2, CDEFABHG,

p_2	D_1	D_2	D_3
C (1)	0	0	0
D (2)	1	0	0
E (3)	0	0	1
F (4)	1	0	1
A (5)	0	1	0
B (6)	1	1	0
H (7)	1	1	0
G (8)	0	0	0

As for p_3, ACBDGFEH,

p_3	D_1	D_2	D_3
A (1)	0	1	0
C (2)	0	0	0
B (3)	1	1	0
D (4)	1	0	0
G (5)	0	0	0

F (6)	1	0	1
E (7)	0	0	1
H (8)	1	1	0

Signature matrix:

	D_1	D_2	D_3
p_1	1	1	3
p_2	2	5	3
p_3	3	1	6

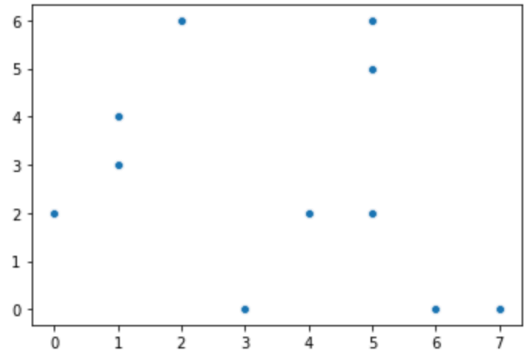
Calculate the similarities:

	D_1-D_2	D_1-D_3	D_2-D_3
Jaccard similarity	0.4	0.2	0
MinHash similarity	0.33	0	0

Q4: (**MST**) Create 3 clusters using minimum spanning tree (MST) with the following coordinates. Please reproduce the diagram in your

```
points = np.array([[1,3],[1,4], [0,2],[2,6],[3,0],[4,2],[5,2],[5,5],[6,0],[5,6],[7,0]] )
sns.scatterplot(points[:,0],points[:,1])
```

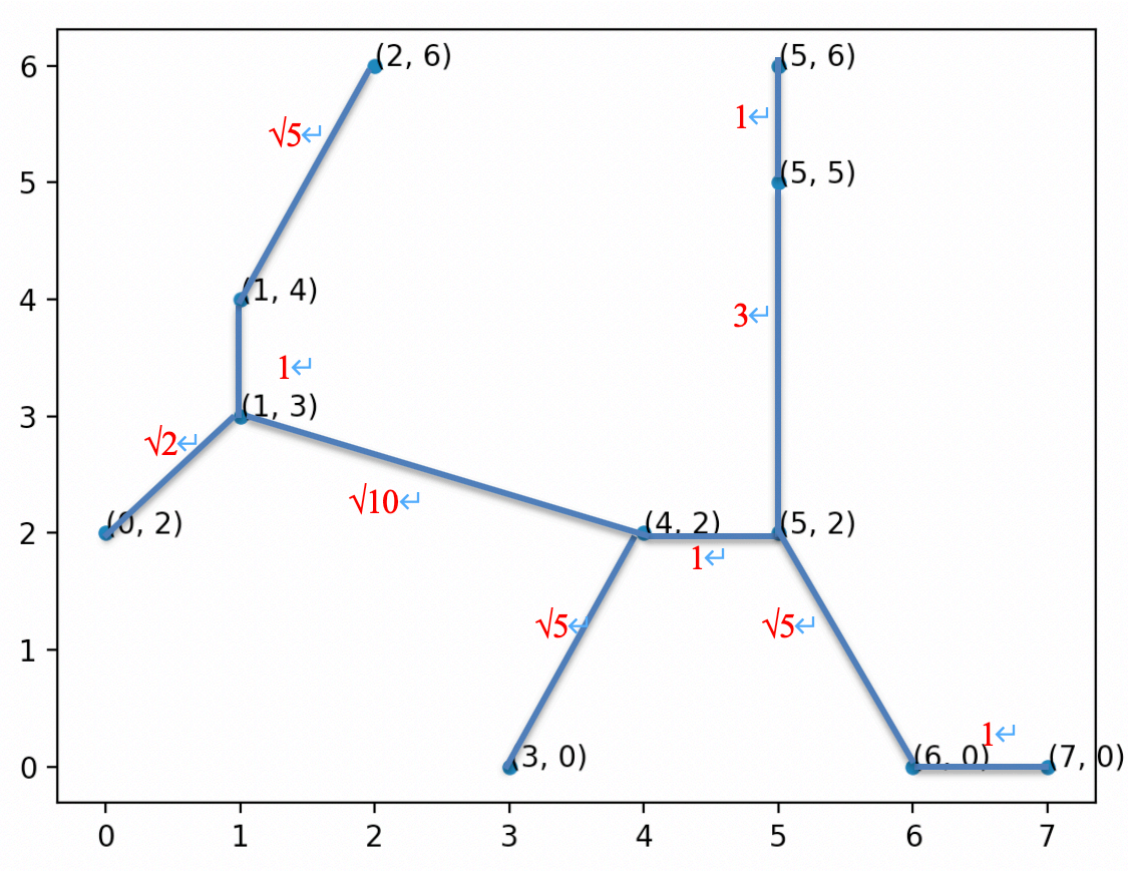
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a26f105d0>
```



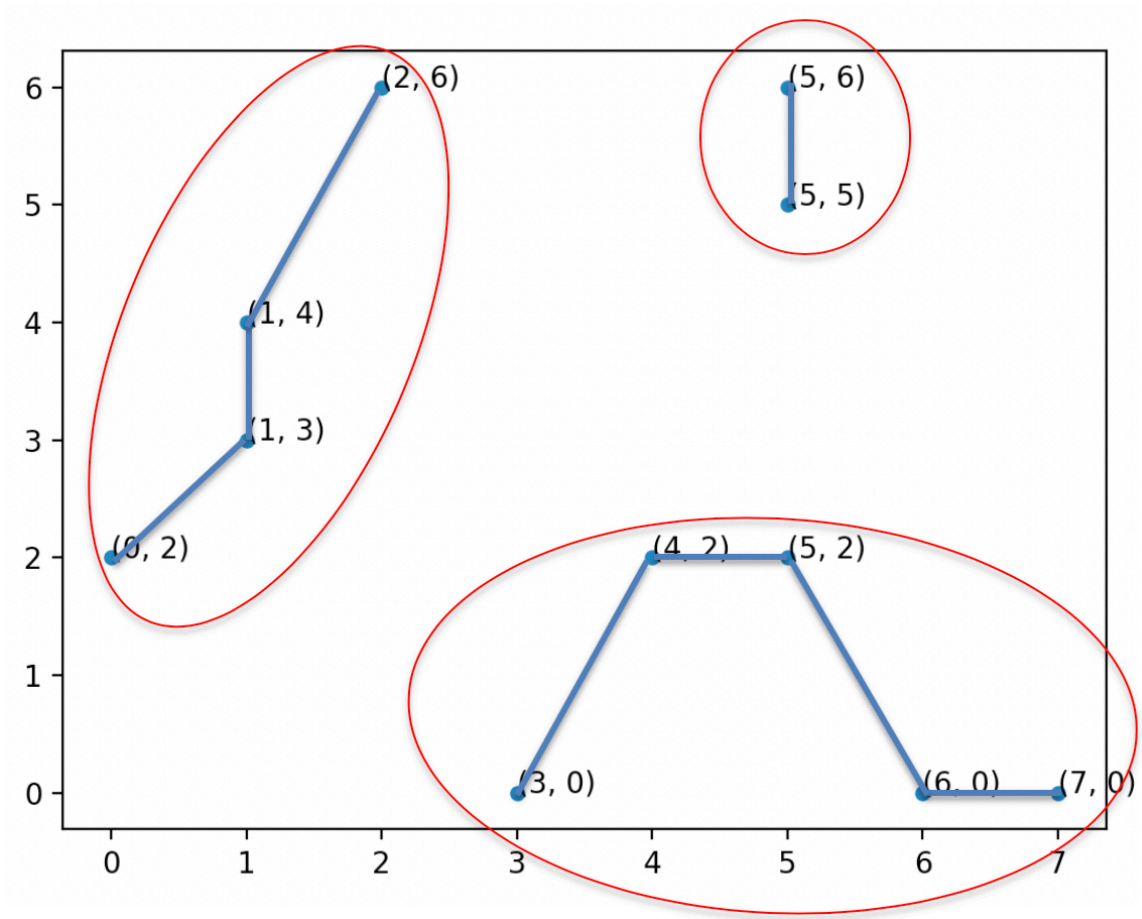
Calculate the distance between each point:

(1,3)	0									
(1,4)	1	0								
(0,2)	√2	√5	0							
(2,6)	√10	√5	2√5	0						
(3,0)	√13	2√5	√13	√37	0					
(4,2)	√10	√13	4	2√5	√5	0				
(5,2)	√17	2√5	5	5	2√2	1	0			
(5,5)	2√5	√17	√34	√10	√29	√10	3	0		
(6,0)	√34	√41	2√10	2√13	3	2√2	√5	√26	0	
(5,6)	5	2√5	√41	√9	2√10	√17	4	1	√37	0
(7,0)	3√5	2√13	√53	√61	4	√11	2√2	√29	1	2√10	0
distance	**(1,3)**	**(1,4)**	**(0,2)**	**(2,6)**	**(3,0)**	**(4,2)**	**(5,2)**	**(5,5)**	**(6,0)**	**(5,6)**	**(7,0)**

Generate the minimum spanning tree (MST):



Erase two longest lines √10 (point (1,3) and point (4.2)) and 3(point (5,2) and point (5,5)) and generate 3 clusters:



Cluster1: (0,2), (1,3), (1,4), (2,6),
Cluster2: (5,5), (5,6),
Cluster3: (3,0), (4,2), (5,2), (6,0), (7,0),