p1:

A) What is the number of frequent itemsets for each dataset?
Which dataset will produce the most number of frequent itemsets?

- Data Set C will produce the most number of frequent itemsets.
- Data Set A：(2**5 - 1) * 2 = 62
  Data Set B：2**3 - 1 = 7
  Data Set C：7 + 3 + 127 = 137

B) Which dataset will produce the longest frequent itemset?

- Data Set C

C) Which dataset will produce frequent itemsets with highest maximum support?

- Data set B

D) Which dataset will produce frequent itemsets containing items with widely varying support levels (i.e., itemsets containing items with mixed support, ranging from 20% to more than 70%)?

- Data Set B

E) What is the number of maximal frequent itemsets for each dataset? Which dataset will produce the most number of maximal frequent itemsets?

- Data set A: 2
  Data set B: 1
  Data set C:3
- Data set C produce the most number of maximal frequent itemsets

F) What is the number of closed frequent itemsets for each dataset? Which dataset will produce the most number of closed frequent itemsets?

- Data set A:{abcde}{fghij}50% 2
  Data set B:{ab}100% 1
  Data set C:{abc} 40%; {bcd} 20%;{defghij} 20% 3;
- Data set C;

P2:

Consider the following set of candidate 3-itemsets:
{1, 2, 4}, {1, 3, 5}, {1, 4, 6}, {2, 3, 5}, {2, 5, 6}, {3, 4, 5}, {3, 5, 6}, {2, 4, 6}

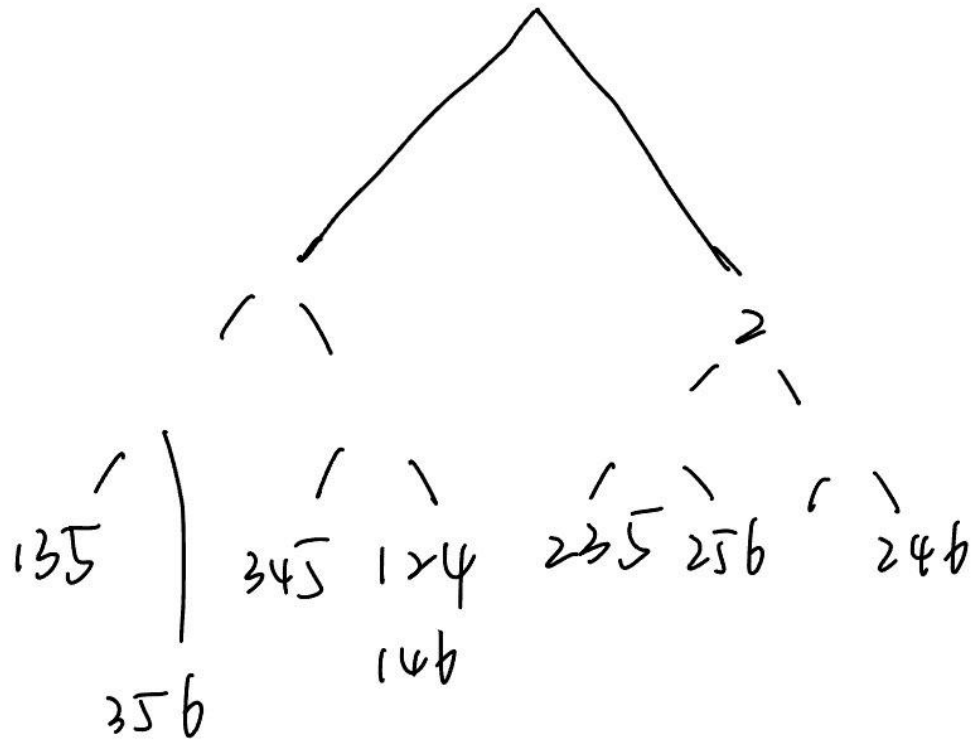
a. Construct a hash tree for the above candidate 3-itemsets. Assume the tree uses a hash function where all odd-numbered items are hashed to the left child of a node, while the even-numbered items are hashed to the right child. A candidate k-itemset is inserted into the tree by hashing on each successive item in the candidate and then following the appropriate branch of the tree according to the hash value. Once a leaf

node is reached, the candidate is inserted based on one of the following conditions:

Condition 1: If the depth of the leaf node is equal to k (the root is assumed to be at depth 0), then the candidate is inserted regardless of the number of itemsets already stored at the node.

Condition 2: If the depth of the leaf node is less than k, then the candidate can be inserted as long as the number of itemsets stored at the node is less than maxsize. Assume maxsize=2 for this question.

Condition 3: If the depth of the leaf node is less than k and the number of itemsets stored at the node is equal to maxsize, then the leaf node is converted into an internal node. New leaf nodes are created as children of the old leaf node. Candidate itemsets previously stored in the old leaf node are distributed to the children based on their hash values. The new candidate is also hashed to its appropriate leaf node.



•

b. How many leaf nodes are there in the candidate hash tree? How many internal nodes are there?

- There are 7 leaf nodes, 7 internal nodes

c. Consider a transaction that contains the following items: {1, 2, 3, 5, 6}. Using the hash tree constructed in part (a), which leaf nodes will be checked against the transaction? What are the candidate 3-itemsets contained in the transaction?

- Step1:find the candidate 3-itemsets in { 1、2、3、5、6}

{1、3、5} 、{2、3、5}、{2、5、6}、{3、5、6}

Step2：comparison

{1, 3, 5} Left（1）→ left（3）→ left（5）

{2, 3, 5} Right（2）→ left（3）→ left（5）

{2, 5, 6} Right（2）→ left（5）→ right（6）

{3, 5, 6} Left（3）→ left（5）→ right（6）

P3：

The Apriori algorithm uses a generate-and-count strategy for deriving frequent itemsets. Candidate itemsets of size k+1 are created by joining a pair of frequent itemsets of size k (this is known as the candidate generation step). A candidate is discarded if any one of its subsets is found to be infrequent during the candidate pruning step. Suppose the Apriori algorithm is applied to the data set shown in Table 2.0 with minsup=30% , i.e., any itemset occurring in less than 3 transactions is considered to be infrequent.

Table: 2.0

Transaction ID	**Items** Bought
T1	a, b, x, y
T2	b, x, y
T3	a, y, z
T4	a, b, x, z
T5	x, y
T6	b, z
T7	a, x, y, z
T8	a, b
T9	b, y, z
T10	a, b, x, y

A. Draw an itemset lattice representing the data set given in Table 2.0 . Label each node in the lattice with the following letter(s):

i. N: If the itemset is not considered to be a candidate itemset by the Apriori algorithm. There are two reasons for an itemset not to be considered as a candidate itemset: (1) it is not generated at all during the candidate generation step, or (2) it is generated during the candidate generation step but is subsequently removed during the candidate pruning step because one of its subsets is found to be infrequent.

ii.F: If the candidate itemset is found to be frequent by the Apriori algorithm.

Iii.I: If the candidate itemset is found to be infrequent after support counting.

b. What is the percentage of frequent itemsets (with respect to all itemsets in the lattice)?

c. What is the pruning ratio of the Apriori algorithm on this data set? (Pruning ratio is defined as the percentage of itemsets not considered to be a candidate because (1) they are not generated during candidate generation or (2) they are pruned during the candidate pruning step.)

d. What is the false alarm rate (i.e., percentage of candidate itemsets that are found to be infrequent after performing support counting)?

- Itemset Lattice

Itemsets size	Itemset		occurrence number
1	{a}	F	6
1	{b}	F	7
1	{x}	F	6

Itemsets size	Itemset		occurrence number
1	{y}	F	7
1	{z}	F	5
2	{a, b}	F	4
2	{a, x}	F	4
2	{a, y}	F	4
2	{a, z}	F	3
2	{b, x}	F	4
2	{b, y}	F	4
2	{b, z}	F	3
2	{x, y}	F	5
2	{x, z}	I	2
2	{y, z}	F	3
3	{a, b, x}	F	3
3	{a, b, y}	I	2
3	{a, x, y}	F	3
3	{b, x, y}	F	3
3	{b, y, z}	I	1
3	{x, y, z}	I	1
3	{a, b, z}	I	1
3	{a, x, z}	N	2
3	{a, y, z}	I	2
3	{b, x, z}	N	1
4	{a, b, x, y}	N	2
4	{a, b, x, z}	N	1
4	{a, b, y, z}	N	0
4	{a, x, y, z}	N	1
4	{b, x, y, z}	N	0
5	{a, b, x, y, z}	N	0

- B. the percentage of frequent itemsets in is 17 /(2**5 - 1) = 17/31
- C.the pruning ratio of the Apriori algorithm on this data set is 8 / 31
- D. the false alarm rate is （31 - 8 -17）/（31 - 8） = 6 / 23