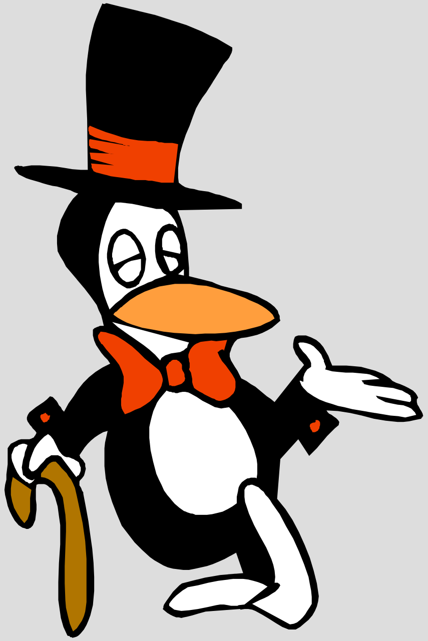
Association Rule Algorithms

- developed to analyze market basket data
- identified groups of market items that customers tended to buy in association with each other
- People who buy soap and shampoo have an increased probability of buying hairspray.

Define:

- $I = \{i_1, i_2, \ldots, i_m\}$ *where i* is a set of items
- Let database *D* consist of a set of records called transactions, *T*
- Each transactions *T* contains a set of items such that $T \subseteq I$
- Let $X \subseteq I$
- *T* contains *X* if $X \subseteq T$

More Formally



- Given a set of items I, (or a set of variables from each transaction or record)
- an association rule is a probabilistic implication $X \Rightarrow Y$ where X and Y are subsets of I and $X \cap Y = \phi$

Support

Support – A rule's statistical significance in a dataset.

- A set of items is said to satisfy a transaction *T* in the dataset if each item's value equals 1, or is contained in that transaction.
- Given an association rule $X \Rightarrow Y$ where X and Y are two disjoint sets of items, then the support of the rule is the number of records that satisfy X U Y divided by the number of records.
- s% of records that contain X U Y

CPU Busy = 1 ⇒ Network Busy = 1

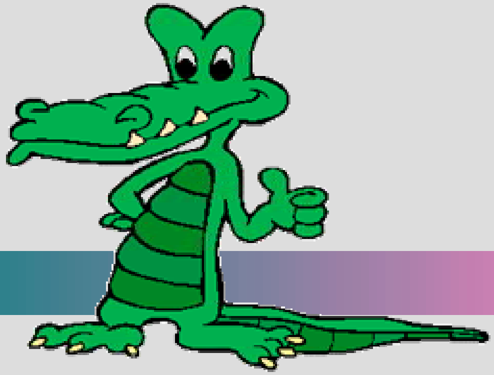
Support

CPU busy	Network Busy	Response Time High	Network Busy High
1	0	1	0
1	1	0	0
1	1	0	1
1	0	1	1
0	1	0	0
1	0	1	1
1	1	0	0
1	1	1	1
0	1	1	1
1	1	0	1

#records where CPU Busy and Network Busy are 1 = 5

total # records = 10

support = 5/10 = 50%



Confidence

Confidence – The strength of a rule relative to the dataset.

Given an association rule $X \Rightarrow Y$ where X and Y are two disjoint sets of items, then the confidence of the rule is the number of records that satisfy X U Y divided by the number of records that satisfy X.

c% of transactions that such that if X is in *T* then Y is in *T*

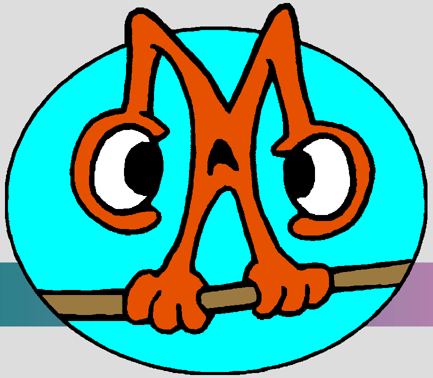
CPU busy = 1 ⇒ Network Busy = 1 Confidence

CPU busy	Network Busy	Response Time High	Network Busy High
1	0	1	0
1	1	0	0
1	1	0	1
1	0	1	1
0	1	0	0
1	0	1	1
1	1	0	0
1	1	1	1
0	1	1	1
1	1	0	1

#records where CPU Busy 1 = 8

#records where CPU Busy and Network Busy are 1 = 5

confidence = 5/8 = 63%



Very Interesting!

A rule that has a support above a user defined threshold, *minsup*, and confidence above a user defined threshold *minconf*, is an interesting association rule Apriori

Brute Force Approach

{1,2,3,4}

{1, 2, 3} {1, 2, 4} {1, 3, 4} {2, 3, 4}

{1, 2} {1, 3} {1, 4} {2, 3} {2, 4} {3, 4}

{1} {2} {3} {4}

List all itemsets and calculate support.

Given *n* items, number of itemsets is 2^n

For highly dimensional datasets this is very bad (Curse of dimensionality)

Apriori

(Agrawal, Imielinsk, and Swami) is an association rule algorithm that efficiently finds rules by finding sets of high support (large itemsets) and then generating confident rules from highly supported sets.

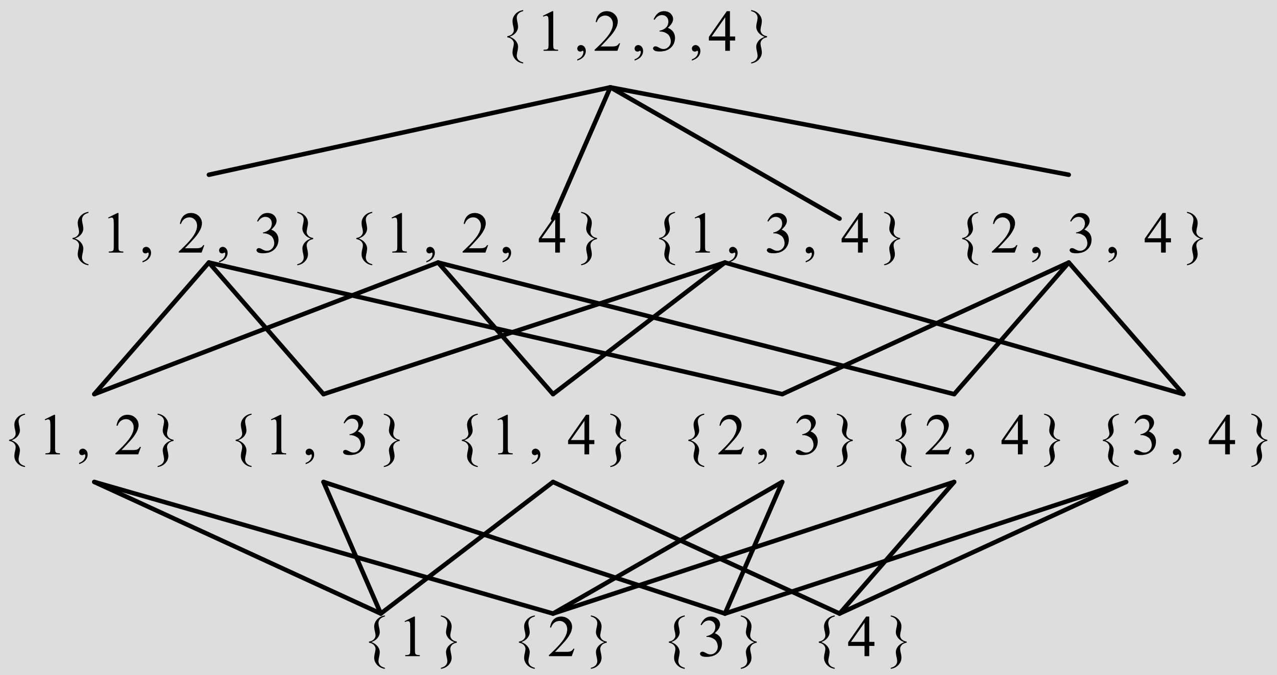
Two Parts

1. Find all large itemsets with minimum support
2. Find all rules from these itemsets with minimum confidence

Part 2 is linear and “easy”

Part 1 is exponential

Subsets of Large Itemsets are Large



The set of all itemsets form a lattice. All size 1 itemsets (1-itemsets) are candidates for being large

Assume

- Each transaction has an identification number associated with it (TID)
- Items are kept in lexicographic order within a transaction
- Itemsets contain a count field, initialized to 0, to keep track of support

Notation

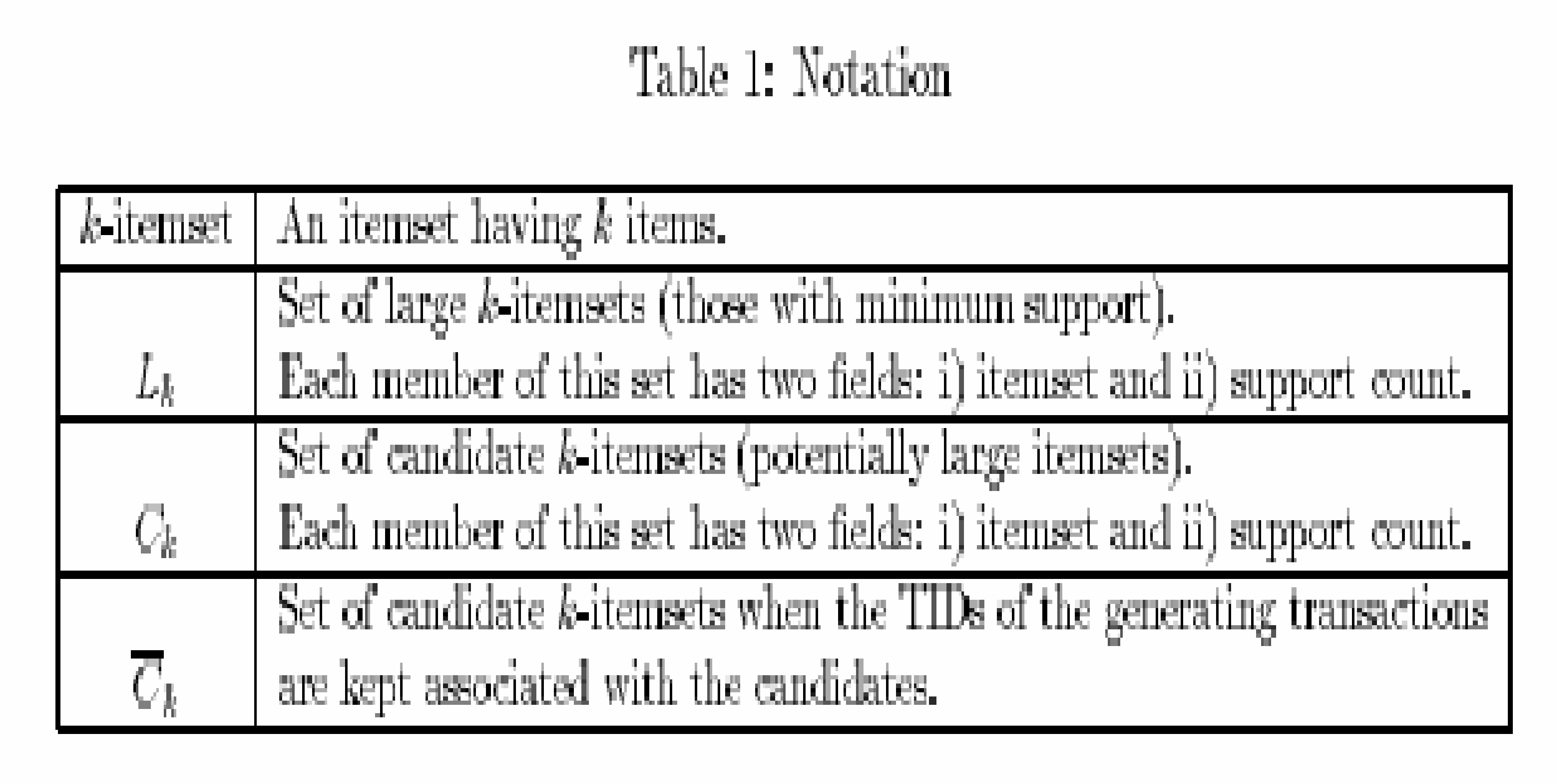
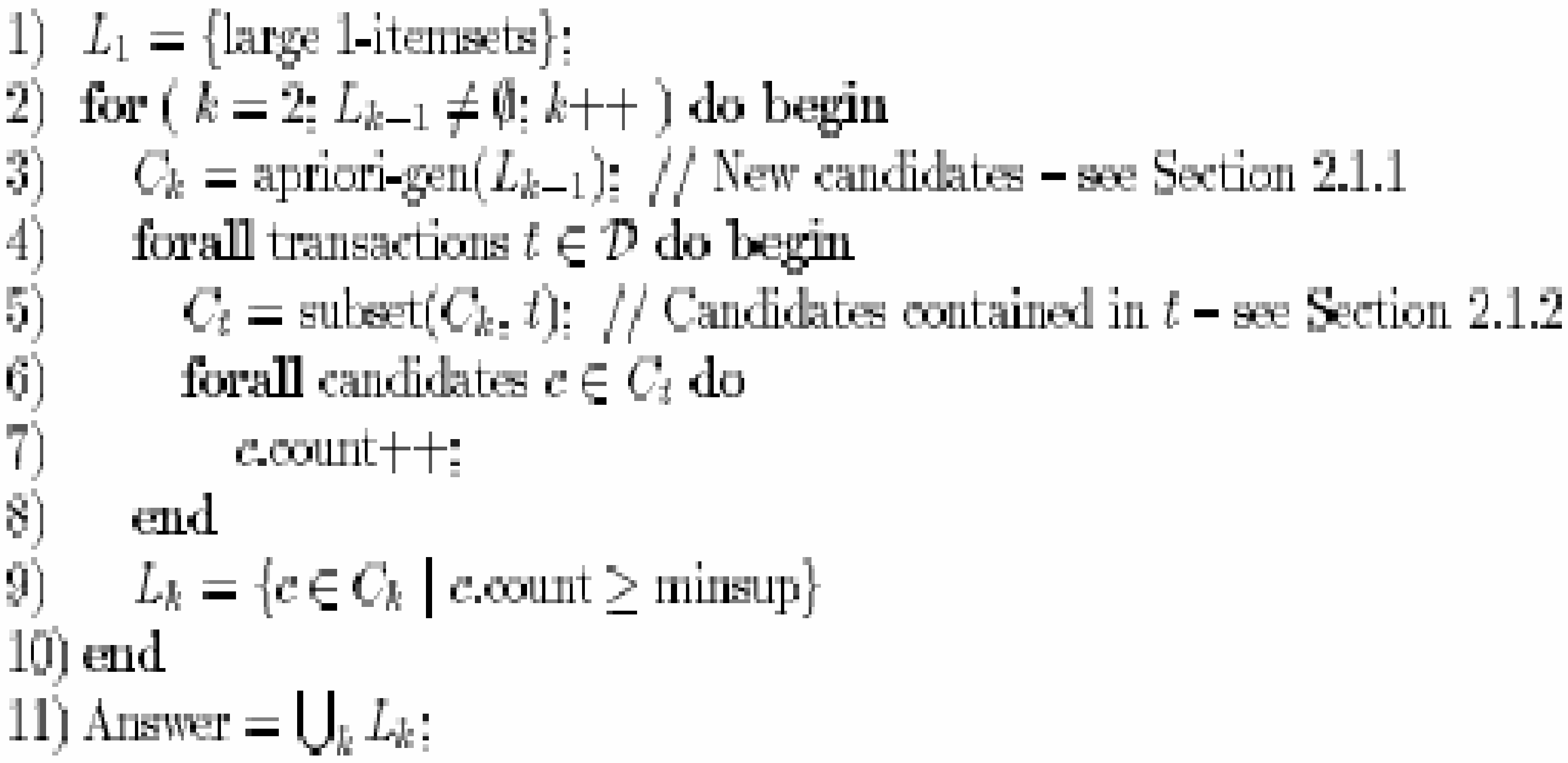


Table 1: Notation

k-itemset	An itemset having k items.
L_k	Set of large k-itemsets (those with minimum support). Each member of this set has two fields: i) itemset and ii) support count.
C_k	Set of candidate k-itemsets (potentially large itemsets). Each member of this set has two fields: i) itemset and ii) support count.
$\overline{C}_k$	Set of candidate k-itemsets when the TIDs of the generating transactions are kept associated with the candidates.

Algorithm Apriori



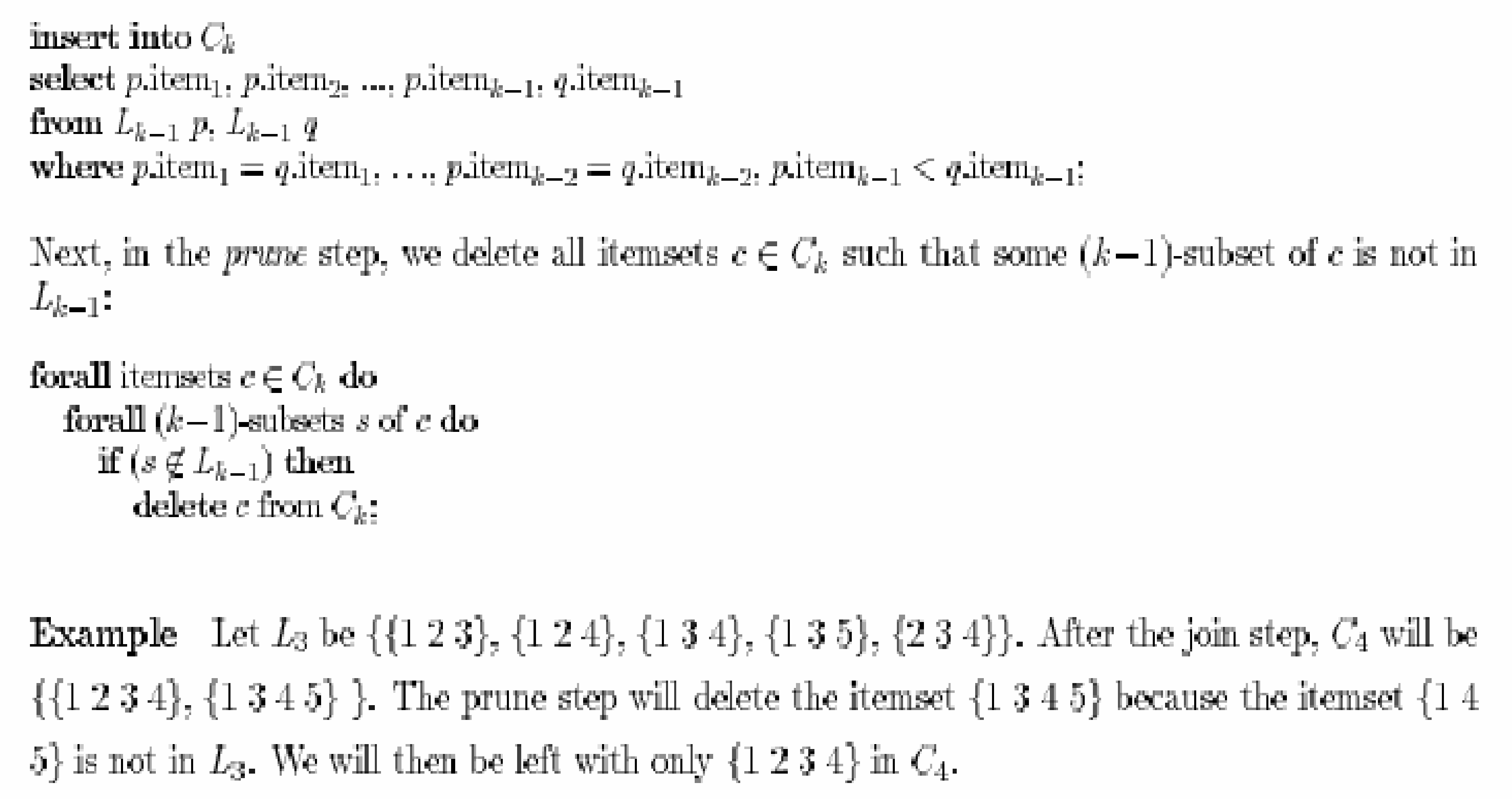
```
$L_1$ = {large 1-itemsets};
for ( $k = 2$; $L_{k-1} \neq \emptyset$; $k$++ ) do begin
    $C_k$ = apriori-gen($L_{k-1}$); // New candidates – see Section 2.1.1
    forall transactions $t \in \mathcal{D}$ do begin
        $C_t$ = subset($C_k$, $t$); // Candidates contained in $t$ – see Section 2.1.2
        forall candidates $c \in C_t$ do
            $c$.count++;
    end
    $L_k = \{c \in C_k \mid c.\text{count} \geq \text{minsup}\}$
end
Answer = $\bigcup_k L_k$;
```

Figure 1: Algorithm Apriori

Apriori Candidate Generation

$L_{k=1}$ is joined with L_{k-1}

p U q is inserted into C_k if p and q have the same, first k – 2 items



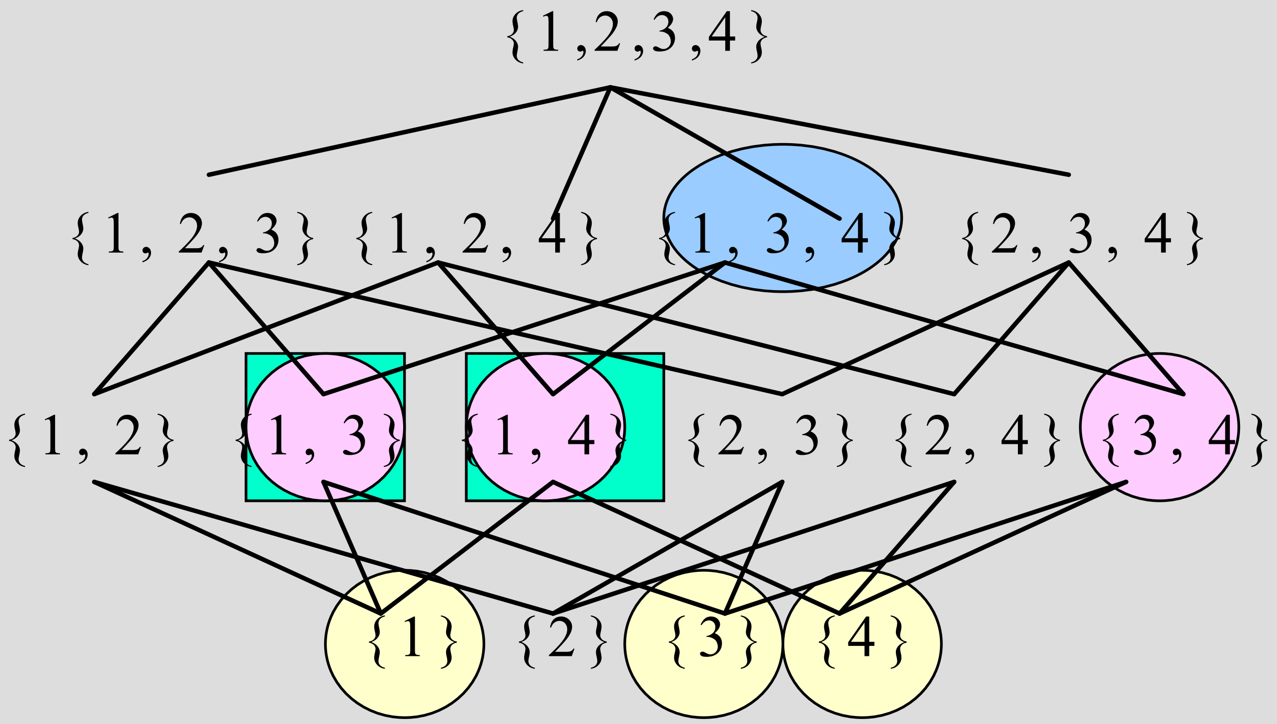
```
insert into $C_k$
select $p.\text{item}_1$, $p.\text{item}_2$, ...., $p.\text{item}_{k-1}$, $q.\text{item}_{k-1}$
from $L_{k-1}$ $p$, $L_{k-1}$ $q$
where $p.\text{item}_1 = q.\text{item}_1$, . . ., $p.\text{item}_{k-2} = q.\text{item}_{k-2}$, $p.\text{item}_{k-1} < q.\text{item}_{k-1}$;
```

Next, in the *prune* step, we delete all itemsets $c \in C_k$ such that some $(k-1)$-subset of c is not in L_{k-1}:

```
forall itemsets $c \in C_k$ do
  forall $(k-1)$-subsets $s$ of $c$ do
    if ($s \notin L_{k-1}$) then
      delete $c$ from $C_k$;
```

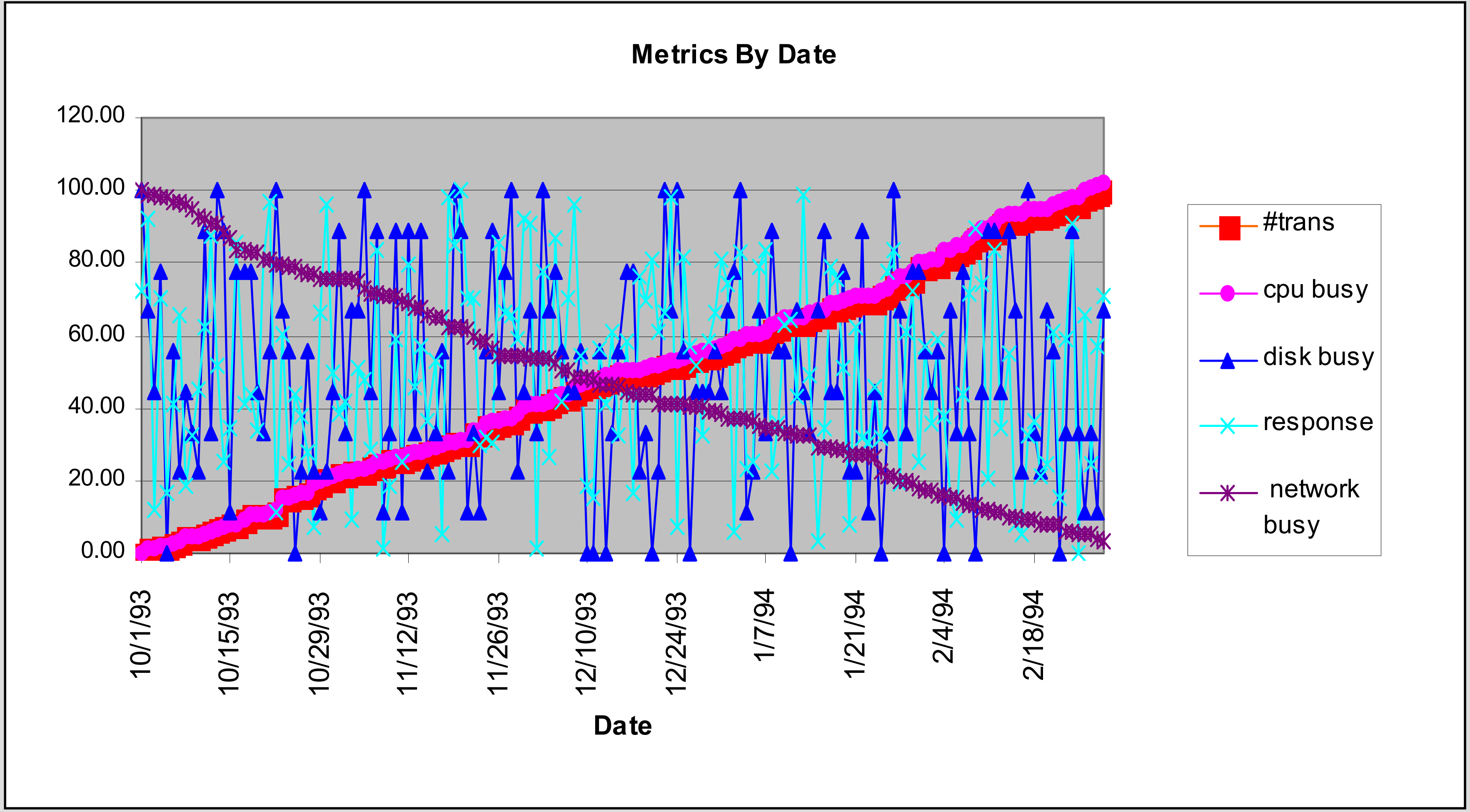
Example Let L_3 be {{1 2 3}, {1 2 4}, {1 3 4}, {1 3 5}, {2 3 4}}. After the join step, C_4 will be {{1 2 3 4}, {1 3 4 5} }. The prune step will delete the itemset {1 3 4 5} because the itemset {1 4 5} is not in L_3. We will then be left with only {1 2 3 4} in C_4.

Supersets of Itemsets found to be large in the k^{th} iteration are candidates for large in the k+1 iteration



Apriori is a level-wise algorithm. Still exponential, but on average better than brute force. Still cursed!

The Sample Data



Results From Apriori On the Sample CP Data

Numeric data is thresholded then translated into boolean values.

minsup = 50%   minconf = 80%

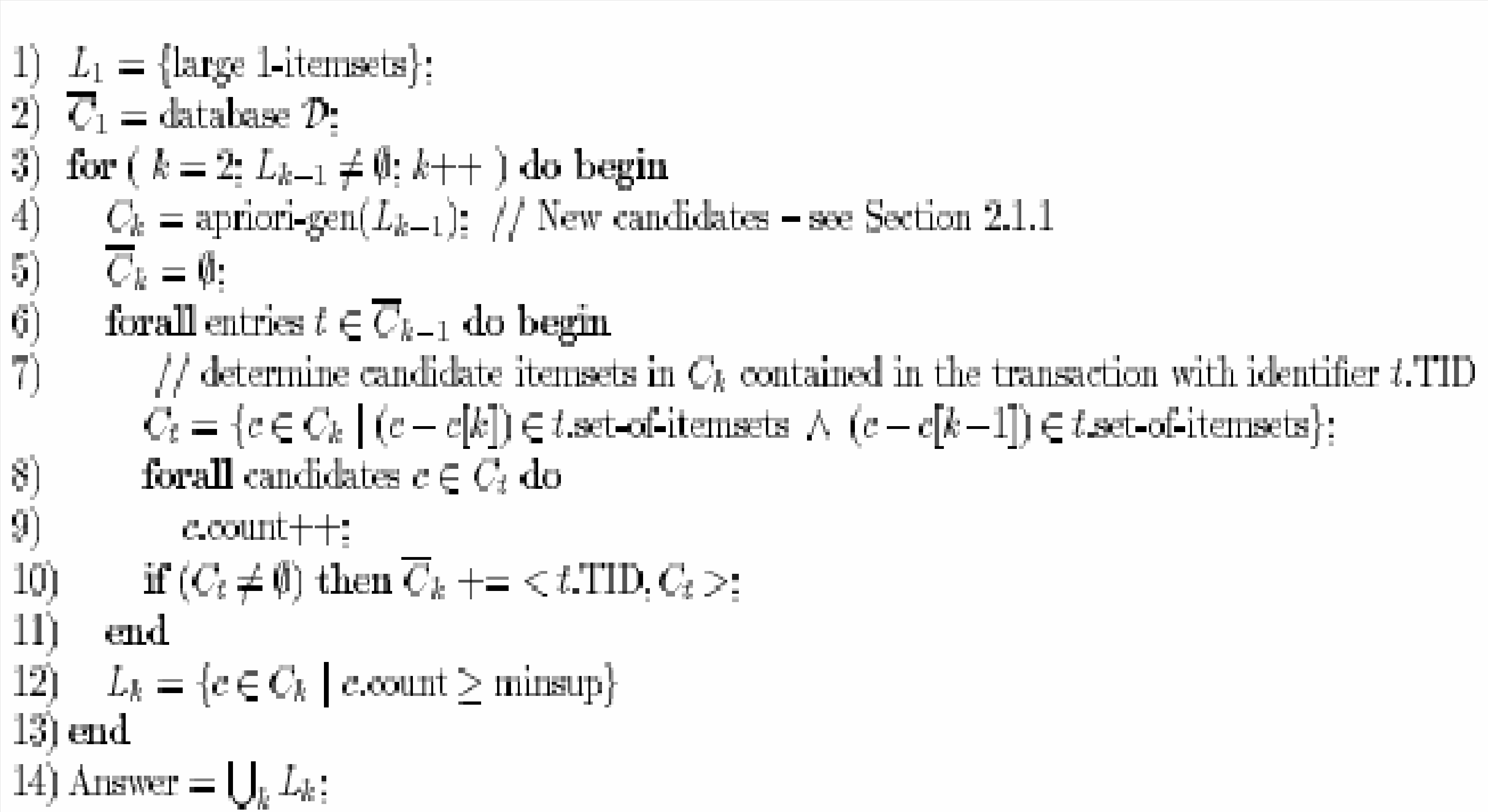
# transactions ⇒ date

(support = 51.0%, confidence = 98.7%)

date ⇒ # transactions

(support = 50.3%, confidence = 100.0%)

Apriori TID



```
$L_1$ = {large 1-itemsets};
$\overline{C}_1$ = database $\mathcal{D}$;
for ( $k = 2$; $L_{k-1} \neq \emptyset$; $k$++ ) do begin
    $C_k$ = apriori-gen($L_{k-1}$); // New candidates – see Section 2.1.1
    $\overline{C}_k = \emptyset$;
    forall entries $t \in \overline{C}_{k-1}$ do begin
        // determine candidate itemsets in $C_k$ contained in the transaction with identifier $t$.TID
        $C_t = \{c \in C_k \mid (c - c[k]) \in t.\text{set-of-itemsets} \wedge (c - c[k-1]) \in t.\text{set-of-itemsets}\}$;
        forall candidates $c \in C_t$ do
            $c$.count++;
        if ($C_t \neq \emptyset$) then $\overline{C}_k$ += < $t$.TID, $C_t$ >;
    end
    $L_k = \{c \in C_k \mid c.\text{count} \geq \text{minsup}\}$
end
Answer = $\bigcup_k L_k$;
```

Database

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

$\overline{C}_1$

TID	Set-of-Itemsets
100	{ {1}, {3}, {4} }
200	{ {2}, {3}, {5} }
300	{ {1}, {2}, {3}, {5} }
400	{ {2}, {5} }

L_1

Itemset	Support
{1}	2
{2}	3
{3}	3
{5}	3

C_2

Itemset
{1 2}
{1 3}
{1 5}
{2 3}
{2 5}
{3 5}

$\overline{C}_2$

TID	Set-of-Itemsets
100	{ {1 3} }
200	{ {2 3}, {2 5}, {3 5} }
300	{ {1 2}, {1 3}, {1 5}, {2 3}, {2 5}, {3 5} }
400	{ {2 5} }

L_2

Itemset	Support
{1 3}	2
{2 3}	2
{2 5}	3
{3 5}	2

C_3

Itemset
{2 3 5}

$\overline{C}_3$

TID	Set-of-Itemsets
200	{ {2 3 5} }
300	{ {2 3 5} }

L_3

Itemset	Support
{2 3 5}	2

Rule Generation (1 of 2)

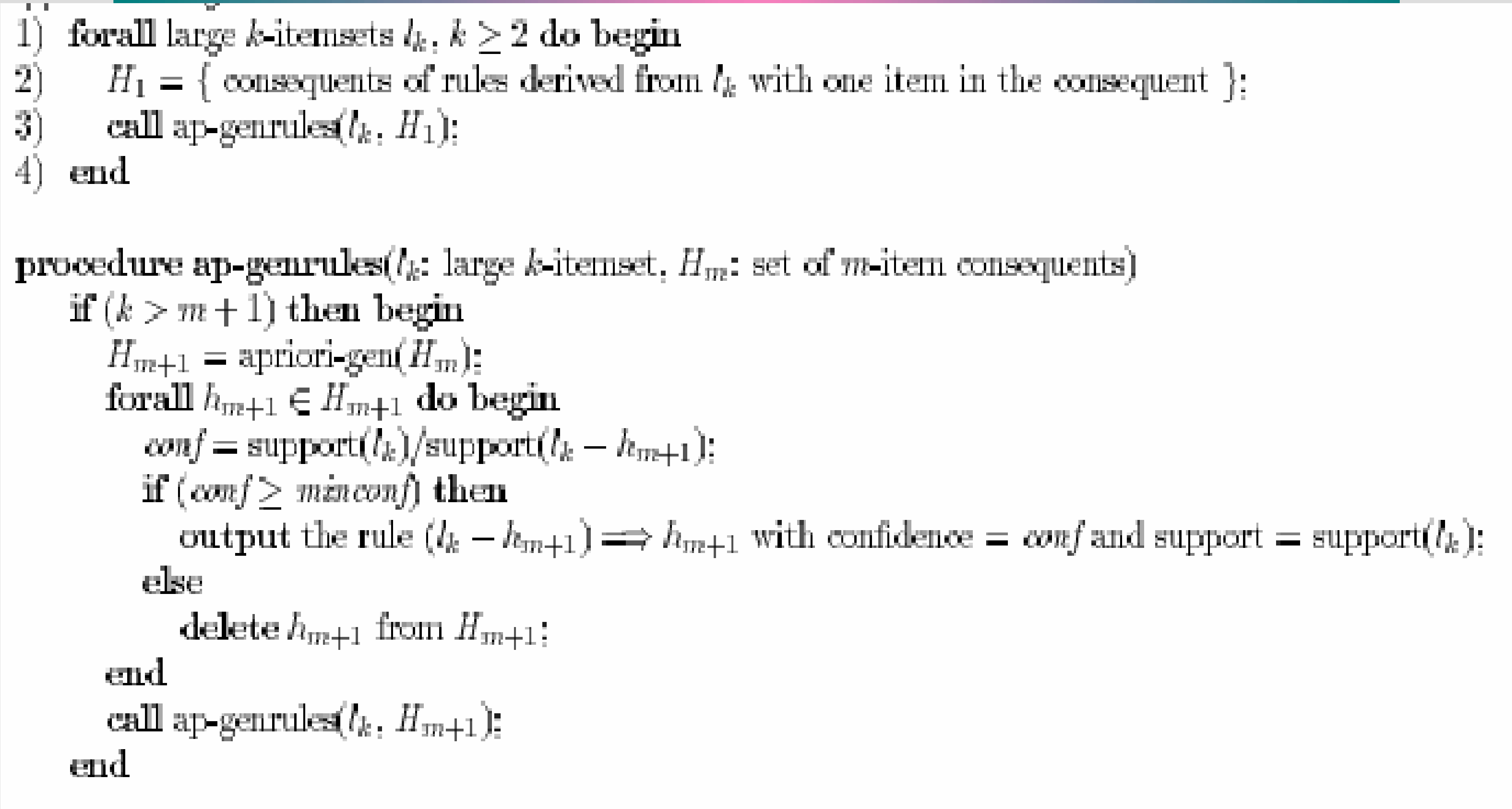
- Given a large itemset l, let $a \subset l$
- Then the confidence of $a \Rightarrow (l - a)$ is support (l) / support (a)
- Let $\tilde{a} \subset a$ then the confidence of

  $\tilde{a} \Rightarrow (l - \tilde{a}) <=$ confidence of $a \Rightarrow (l - a)$
- Therefore if $a \Rightarrow (l - a)$ has confidence less than minconf then $\tilde{a} \Rightarrow (l - \tilde{a})$ has confidence less than minconf.

Rule Generation (2 of 2)

- Hence if (l – ã) ⇒ ã has confidence of at least minconf then (l – a) ⇒ a has confidence of at least minconf
  - Ex. if AB ⇒CD holds, then ABC ⇒ D holds and ABD ⇒ C holds
- Given a large itemset l_k, generate all rules with a single item in the consequent
- Use rule consequents and apriori gen procedure to generate rules with consequents of 2 items. Continue recursively

Rule Generation Algorithm



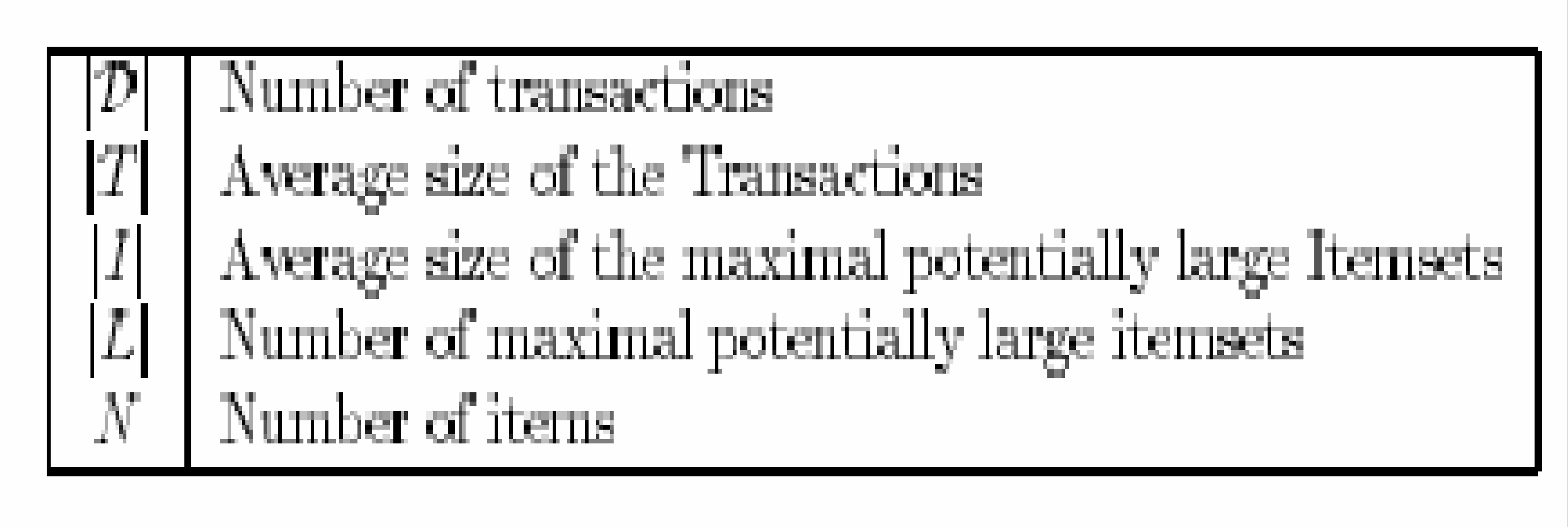
```
forall large $k$-itemsets $l_k$, $k \geq 2$ do begin
    $H_1$ = { consequents of rules derived from $l_k$ with one item in the consequent };
    call ap-genrules($l_k$, $H_1$);
end

procedure ap-genrules($l_k$: large $k$-itemset, $H_m$: set of $m$-item consequents)
    if ($k > m + 1$) then begin
        $H_{m+1}$ = apriori-gen($H_m$);
        forall $h_{m+1} \in H_{m+1}$ do begin
            $conf$ = support($l_k$)/support($l_k - h_{m+1}$);
            if ($conf \geq minconf$) then
                output the rule ($l_k - h_{m+1}$) $\Longrightarrow h_{m+1}$ with confidence = $conf$ and support = support($l_k$);
            else
                delete $h_{m+1}$ from $H_{m+1}$;
        end
        call ap-genrules($l_k$, $H_{m+1}$);
    end
```

Synthetic Data

- Mimics transactions in retail environment
- people tend to buy sets of items together
- transactions are clustered around a mean and few transactions contain more than one large item
- sizes of large itemsets are also clustered around a mean with few large itemsets having large numbers of items

Parameters



$\lvert\mathcal{D}\rvert$	Number of transactions
$\lvert T\rvert$	Average size of the Transactions
$\lvert I\rvert$	Average size of the maximal potentially large Itemsets
$\lvert L\rvert$	Number of maximal potentially large itemsets
N	Number of items

Synthetic Data

- Size of next transaction picked from a Poisson distribution with mean μ equal to |T|
- Assign items to transaction
  - each transaction is assigned several potentially large itemsets
  - if too large for transaction 50% of time put there anyway, other 50% of time placed in next transaction
  - Choose large itemsets from a set *T* of large itemsets. number of itemsets in *T* = |L|
  - the size of an itemset is given by choosing from a Poisson distribution with μ = | *I*|
  - choose itemsets randomly for first set
  - For subsequent sets, some fraction of the previous itemset are chosen (large itemsets frequently have common items. Remainder are chosen randomly
  - Associate with each itemset in *T* a weight corresponding to the probability that the itemset will be picked (weight is picked form an exponential distribution with unit mean and then normalized so the sum of the weights for all itemsets in *T* is 1.
  - Choose the next itemset for a transaction from *T* by tossing an |L| sided weighted coin.

Synthetic Data

- All itemsets in large itemsets are not bought together
- Assign *T* a corruption level *c*
- drop an item from the itemset when inserting into the transaction as long as a uniformly distributed random number between 0 and 1 is less than c
- This means that for an itemset of size *l*, add *l* items 1 – *c* of the time, *l* – 1 items c(1-c) of the time *l* - 2 items c^2(1 – c) of the time
- corruption level for an itemset is fixed and chosen from a normal distribution with mean 0.5 and variance 0.1

Association Rule Algorithms

- Association rule algorithms show co-occurrence of variables.
- tend to generate many more rules than what is seen in decision trees
- gives the ability to find rare and less obvious patterns in the data.
- attempts to use the rules generated by association rule algorithms - has met with mixed results.