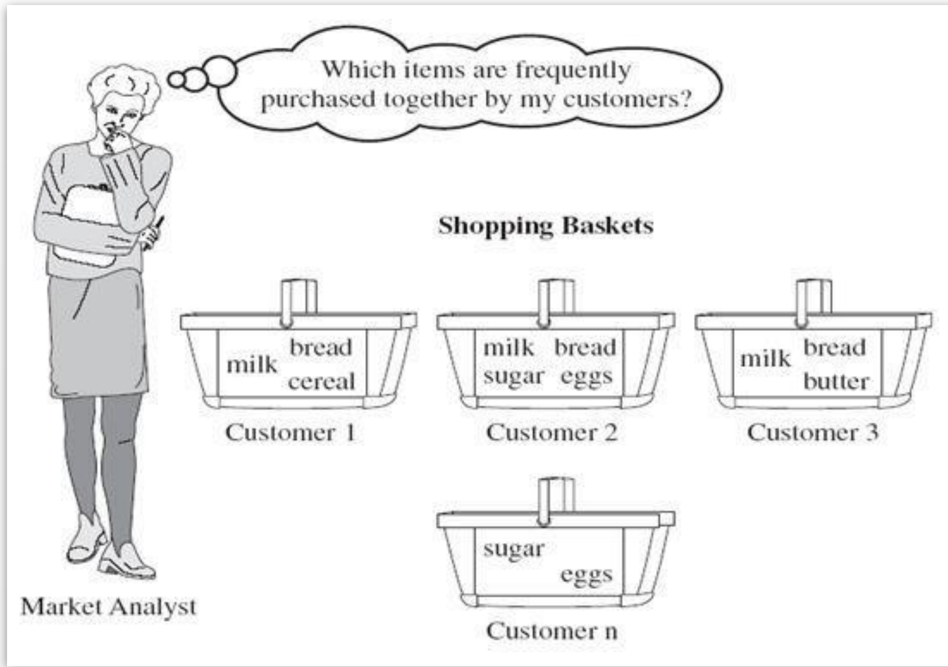
Mining Frequent Itemsets with A-Priori

Market Basket Analysis



Baskets of items

TID	*Items*
1	**Bread, Coke, Milk**
2	**Beer, Bread**
3	**Beer, Coke, Diaper, Milk**
4	**Beer, Bread, Diaper, Milk**
5	**Coke, Diaper, Milk**

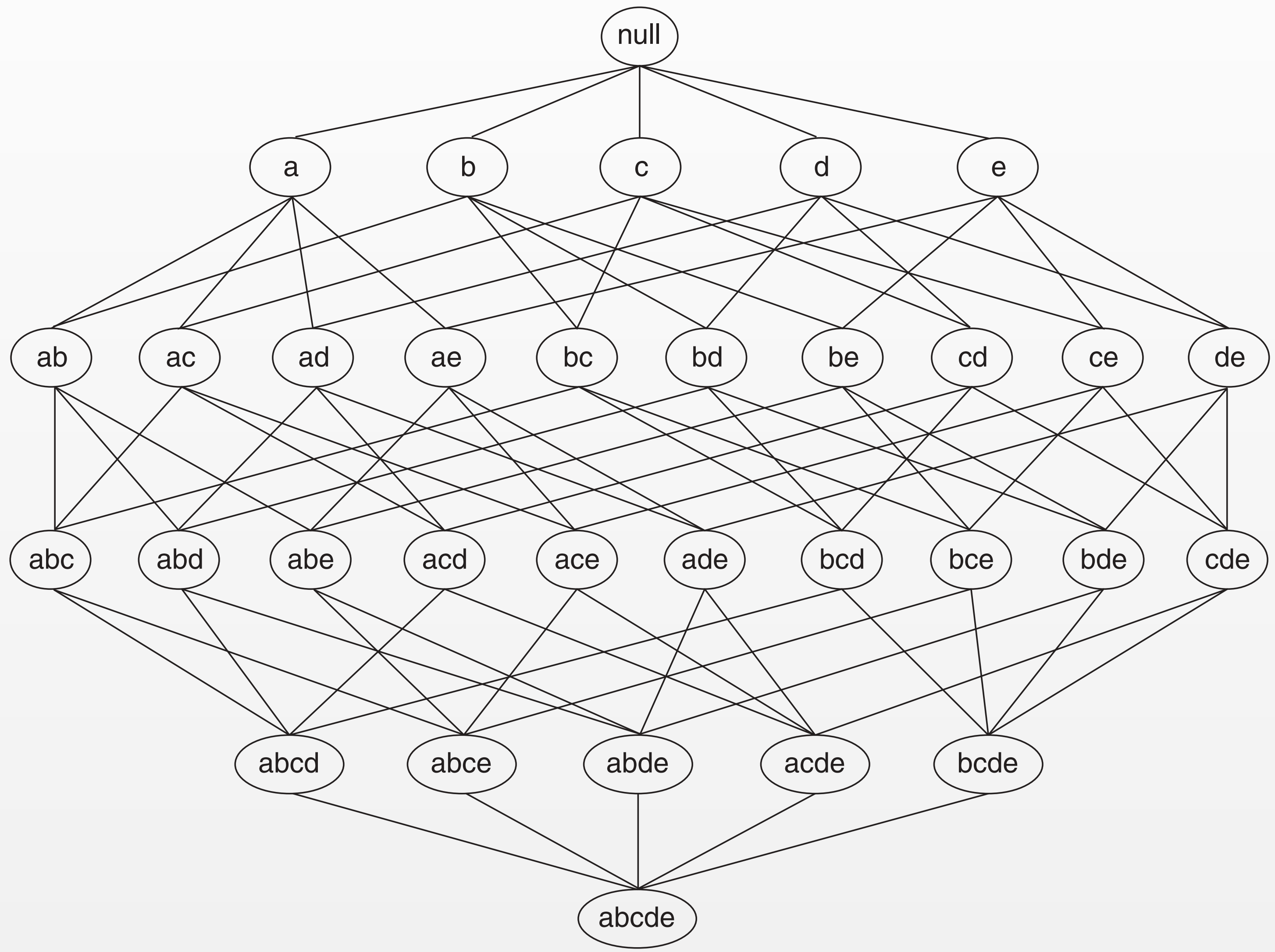
Association Rules

{Milk} --> {Coke}
{Diaper, Milk} --> {Beer}

Finding Frequent Item Sets

Let I be the set of all items

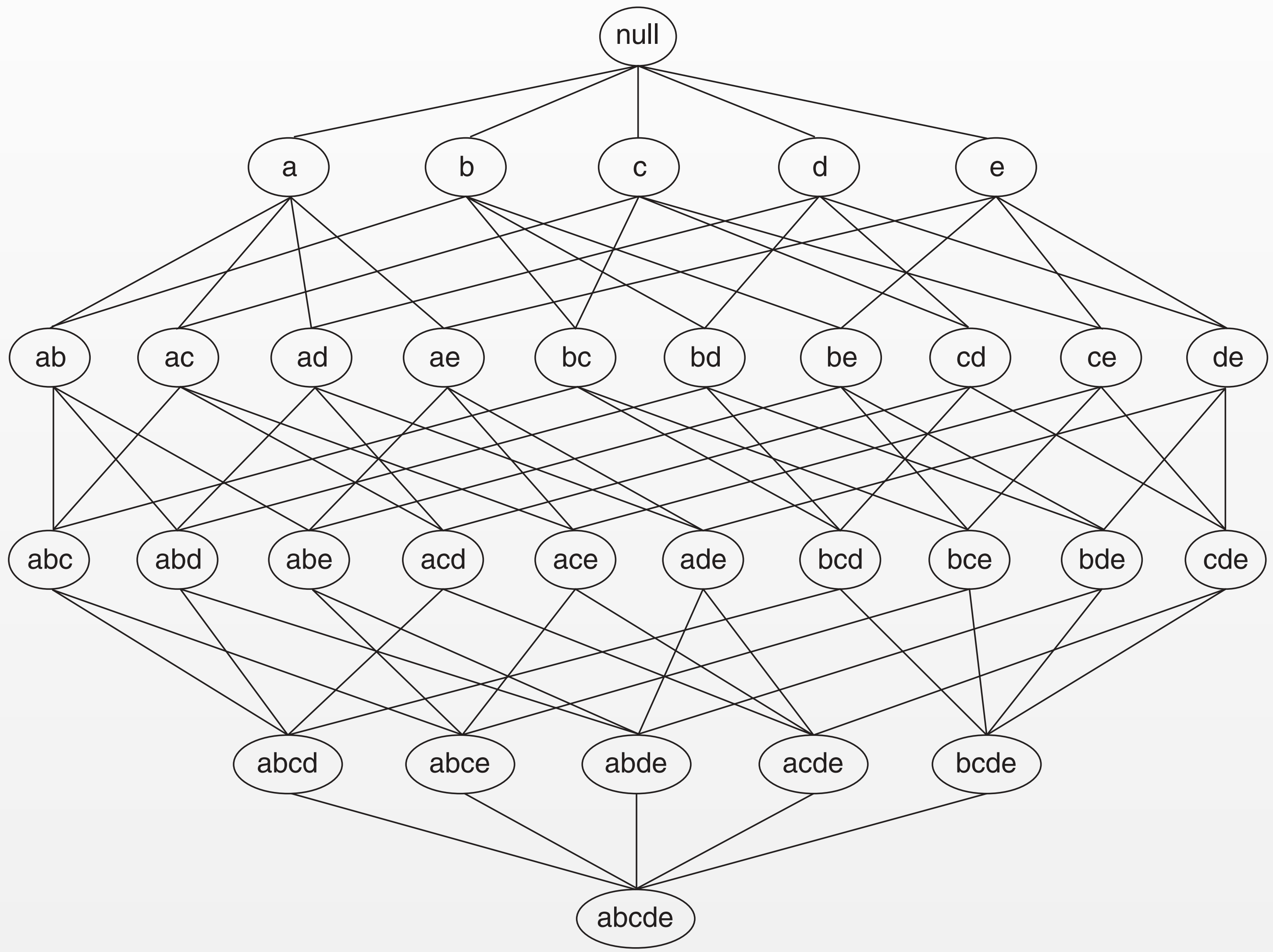
If $M = |I|$, how many possible itemsets are there?



adapted from: Tan, Steinbach & Kumar, "Introduction to Data Mining", http://www-users.cs.umn.edu/~kumar/dmbook/ch6.pdf

Finding Frequent Item Sets

Answer: $2^M - 1$; Cannot enumerate all possible sets



adapted from: Tan, Steinbach & Kumar, "Introduction to Data Mining", http://www-users.cs.umn.edu/~kumar/dmbook/ch6.pdf

Anti-monotone Property

A function f (defined on sets) is said to follow the anti-monotone property if

$$\forall A, B \in 2^I : \quad A \subseteq B \Rightarrow f(A) \geq f(B)$$

I is the set of all items
2^I denotes the power set of I

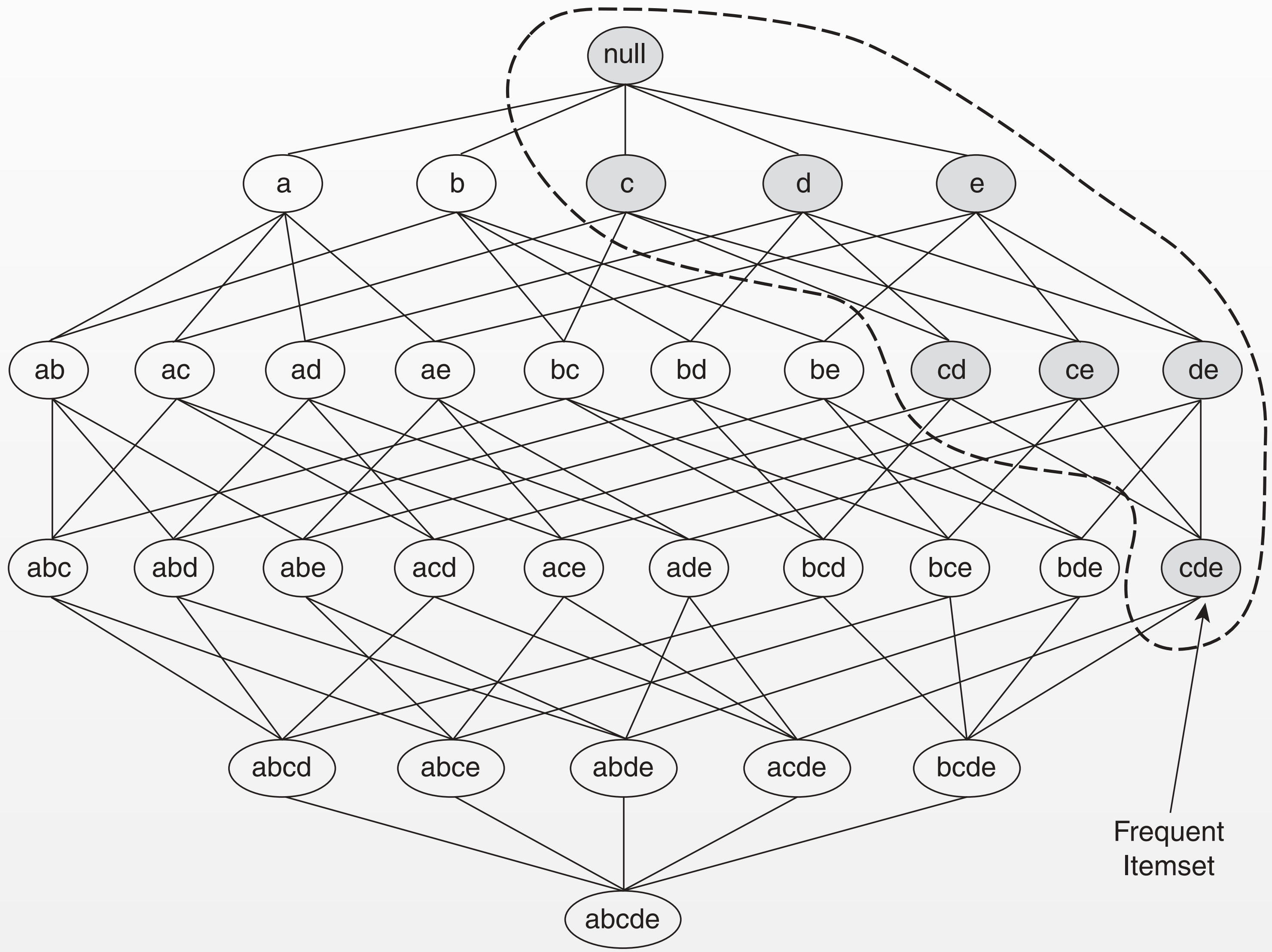
Support follows the anti-monotone property

$$\sigma(A) = |\{t \in T : A \subset t\}|$$

$\sigma : 2^I \rightarrow \mathbf{N}$
$N = \{0, 1, \ldots, \infty\}$, the set of natural numbers
T: the set of all transactions

Intuition: A-priori Principle

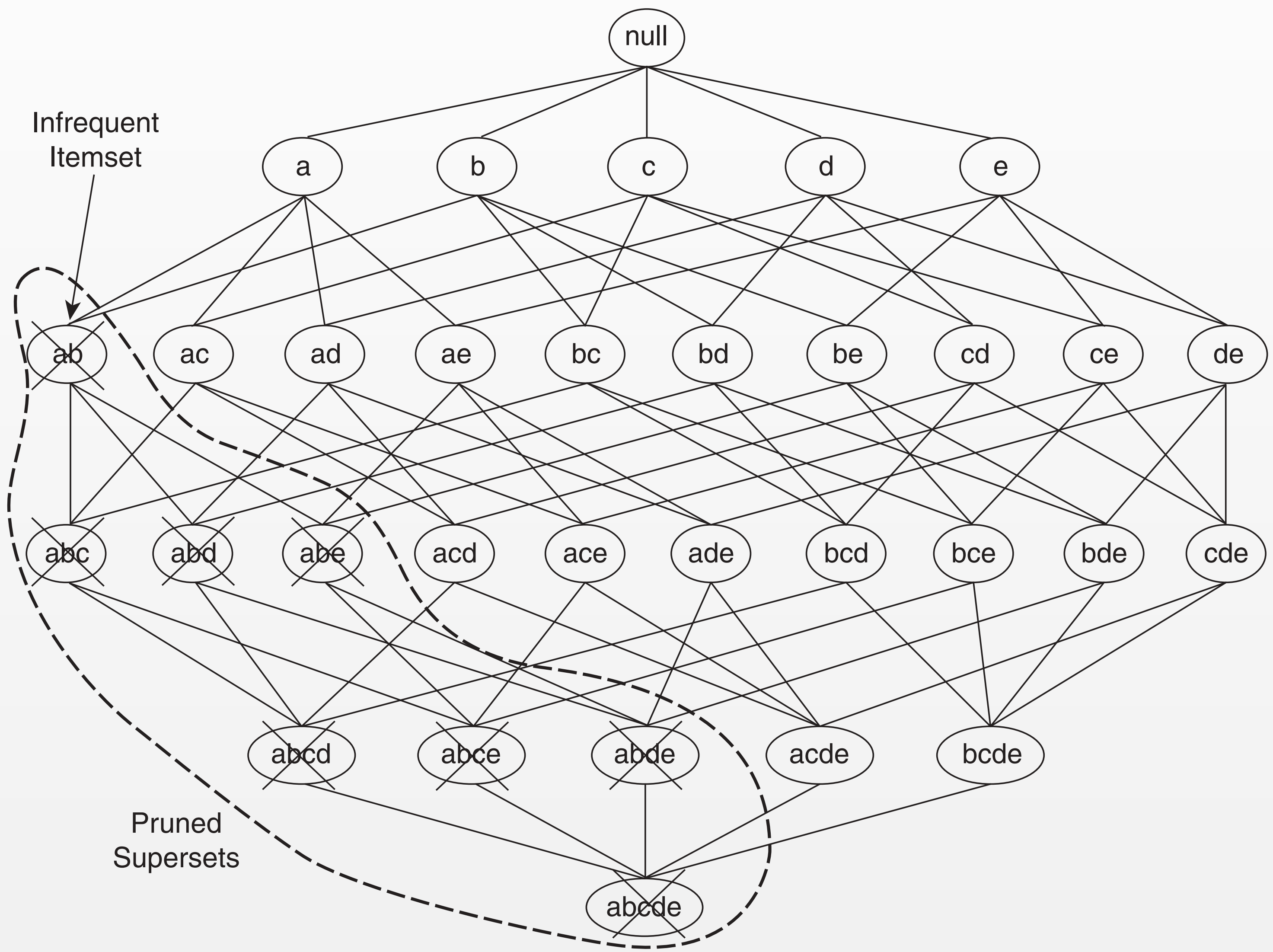
Observation: Subsets of a frequent item set are **also** frequent



adapted from: Tan, Steinbach & Kumar, "Introduction to Data Mining", http://www-users.cs.umn.edu/~kumar/dmbook/ch6.pdf

Intuition: A-priori Principle

Corollary: If a set is **not** frequent, then its supersets are **also** not frequent



adapted from: Tan, Steinbach & Kumar, “Introduction to Data Mining”, http://www-users.cs.umn.edu/~kumar/dmbook/ch6.pdf

A-priori Algorithm

1. Find all frequent itemsets of size 1
   (*only have to check* $M = |I|$ *possible sets*)
2. For $k = 1,2,\ldots M$
   - Extend frequent itemsets of size $k - 1$ to create *candidate* itemsets of size k
   - Find candidate sets that are frequent

A-priori Algorithm

Algorithm 6.1 Frequent itemset generation of the *Apriori* algorithm.

$k = 1$.
$F_k = \{\, i \mid i \in I \wedge \sigma(\{i\}) \geq N \times minsup\}$.   {Find all frequent 1-itemsets}
repeat   Interpret minsup as a fraction here
  $k = k + 1$.
  $C_k = \text{apriori-gen}(F_{k-1})$.   {Generate candidate itemsets}
  for each transaction $t \in T$ **do**
    $C_t = \text{subset}(C_k, t)$.   {Identify all candidates that belong to t}
    for each candidate itemset $c \in C_t$ **do**
      $\sigma(c) = \sigma(c) + 1$.   {Increment support count}
    end for
  end for
  $F_k = \{\, c \mid c \in C_k \wedge \sigma(c) \geq N \times minsup\}$.   {Extract the frequent k-itemsets}
until $F_k = \emptyset$
Result $= \bigcup F_k$.

adapted from: Tan, Steinbach & Kumar, "Introduction to Data Mining", http://www-users.cs.umn.edu/~kumar/dmbook/ch6.pdf

Matching Transactions to Candidate Sets

Algorithm 6.1 Frequent itemset generation of the *Apriori* algorithm.

$k = 1$.
$F_k = \{\, i \mid i \in I \wedge \sigma(\{i\}) \geq N \times minsup \}$. {Find all frequent 1-itemsets}
repeat
  $k = k + 1$.
  $C_k =$ apriori-gen(F_{k-1}). {Generate candidate itemsets}
  for each transaction $t \in T$ **do**
    $C_t =$ subset(C_k, t). {Identify all candidates that belong to t}
    for each candidate itemset $c \in C_t$ **do**
      $\sigma(c) = \sigma(c) + 1$. {Increment support count}
    end for
  end for
  $F_k = \{\, c \mid c \in C_k \wedge \sigma(c) \geq N \times minsup \}$. {Extract the frequent k-itemsets}
until $F_k = \emptyset$
Result $= \bigcup F_k$.

adapted from: Tan, Steinbach & Kumar, “Introduction to Data Mining”, http://www-users.cs.umn.edu/~kumar/dmbook/ch6.pdf

Generating Candidates C_k

Objectives

1. ***No Duplicates:*** A candidate itemsets must be unique.
2. **Completeness:** At least, all frequent k-itemsets should be included.
3. **No infrequent subsets**: A candidate should not have any infrequent subset.

Generating Candidates C_k

Questions

1. *How many k-itemsets are there?*
2. *How to reduce number of candidates using the computations already performed?*
   1. $F_{k-1} \times F_1$*: combine frequent (k-1)-itemsets with frequent 1-itemsets to get candidates of size k.*

      $$\{a, b, c\} \cup \{d\} = \{a, b, c, d\}$$

   2. $F_{k-1} \times F_{k-1}$*: combine frequent (k-1)-itemsets with other frequent (k-1)-itemsets that differs in only 1 item to get candidates of size k.*

      $$\{a, b, c\} \cup \{a, b, d\} = \{a, b, c, d\}$$

Generating Candidates C_k

Duplicates

Combining sets arbitrary pairs of sets from F_{k-1} and F_1 will lead to duplicate candidates

$$\{a, b, c\} \cup \{d\} = \{a, b, c, d\}$$
$$\{a, b, d\} \cup \{c\} = \{a, b, c, d\}$$

Each candidate of size k could be generated k times

Combining sets arbitrary pairs of sets from F_{k-1} will lead to duplicate candidates

$$\{a, b, c\} \cup \{a, b, d\} = \{a, b, c, d\}$$
$$\{a, b, c\} \cup \{a, c, d\} = \{a, b, c, d\}$$

Each candidate of size k could be generated $\binom{k}{k-2}$ times

Generating Candidates C_k

Solution to duplicates

Sort the items

- **Item ordering:** Define an ordering on all items
  - Either by assigning a unique id to each item. The items are ordered based on their ID.
  - Or by a lexicographic ordering on the item string; e.g. 'coke' < 'cookie'
- Assume that the items in the itemsets in F_{k-1} are sorted. $a_1 < a_2 < \ldots < a_k$ in $A = \{a_1, a_2, \ldots, a_k\}$

ID based ordering wil have computational advantage since comparing numbers is cheaper than comparing strings

Generating Candidates C_k

Solution to duplicates

$F_{k-1} \times F_1$: Combine $A \in F_{k-1}$ and $B = \{b\} \in F_1$ only if $\forall a \in A \quad a < b$.

Combine if all items in A are less than the only item in B

- Combine $\{a, c, e\}$ with $\{f\}$ to give candidate $\{a, c, e, f\}$.
- Do not combine $\{a, c, e\}$ with $\{d\}$.
- No duplicates: Each candidate has only one way of being generated. $\{a, c, e, f\}$ can only be generated by combining $\{a, c, e\}$ and $\{f\}$.
- Completeness: If $\{a, c, e, f\}$ is indeed frequent, $\{a, c, e\}$ and $\{f\}$ have to be present in F_3 and F_1. And they would get combined to generate $\{a, c, e, f\}$.
- Subsets of generated candidates might still be infrequent

Generating Candidates C_k

Solution to duplicates

Combine if the first $k-2$ items in A and B are the same and the last element of A is less than that of B.

$F_{k-1} \times F_{k-1}$: Combine $A \in F_{k-1}$ and $B \in F_{k-1}$ only if $a_i = b_i$, for $i = 1,2\ldots k-2$ and $a_{k-1} < b_{k-1}$.

- Combine $\{a, c, e\}$ with $\{a, c, f\}$ to give candidate $\{a, c, e, f\}$.
- Do not combine $\{a, c, e\}$ with $\{a, b, e\}$.
- No duplicates: Each candidate has only one way of being generated. $\{a, c, e, f\}$ can only be generated by combining $\{a, c, e\}$ and $\{a, c, f\}$.
- Completeness: If $\{a, c, e, f\}$ is indeed frequent, $\{a, c, e\}$ and $\{a, c, f\}$ have to be present in F_3. And they would get combined to generate $\{a, c, e, f\}$.
- Subsets of generated candidates might still be infrequent

Generating Candidates C_k

How to efficiently find itemsets that could be combined?

- **Itemset Ordering:** Use the ordering on items to define an ordering of itemsets
  - Assume that the items in each itemset are presorted.
    $a_1 < a_2 < \ldots < a_k$ in $A = \{a_1, a_2, \ldots, a_k\}$
  - $A < B$ if $a_i < b_i$, where i is the index of the first item differing in A and B.
    {apple, bread, coke, sauce} $<$ {apple, bread, cookie, milk} or $\{4,7,21,50\} < \{4,7,25,40\}$

Elements of F_{k-1} sorted with the itemset ordering

$$\{a, b, c\}, \{a, b, e\}, \{a, b, g\}, \{a, c, d\}, \{a, c, g\} \ldots$$

$F_{k-1} \times F_{k-1}$

$\{a, b, c\}$ can't be combined with any itemset beyond $\{a, b, g\}$. So no need to compare beyond $\{a, c, d\}$

Without exploiting the itemset ordering $|F_{k-1}|(|F_{k-1}| - 1)/2$ comparisons need to be made

Generating Candidates C_k

Pruning candidates with infrequent subsets

- For a candidate of size k, one only needs to check subsets of size $k-1$.
- Enumerate subsets of size $k-1$ by removing one element at a time from the candidate.
- Search for the subsets one after the other in F_{k-1} until a subset is not found or the list of subsets is exhausted
  - Binary search could be performed if F_{k-1} is sorted under the itemset ordering for an efficient search.
  - Alternatively a hash tree could be build to store the itemsets of F_{k-1} for an efficient search.
- If a subset wasn't found the candidate should be discarded.

If all size $k-1$subsets are frequent so are subsets of smaller sizes.

Each candidate of size k will give k subsets of size $k-1$

For each candidate $O(k|F_{k-1}|)$ compariso might be needed in the worst case, if searching naively.
This can be improved b binary search $O(k\log(|F_{k-1}|))$ or constructing hash table

Generating Candidates C_k

1. *Self-joining:* Find pairs of sets in F_{k-1} that have first $k-2$ items in common and differ by **one** element.
2. *Pruning:* Remove all candidates with infrequent subsets

Example: Generating Candidates C_k

- *Frequent itemsets of size 2:*
  {b,c}:5, {b,m}:4,{c,j}:3 {c,m}:3
- *Self-joining:*
  {b,c,m}, {c,j,m}
- *Pruning:*
  ~~{c,j,m}~~ since ~~{j,m}~~ not frequent
- Frequent items of size 3:
  {b,c,m}

B_1 = {b, c, m} B_2 = {j, m, p}
B_3 = {b, m} B_4= {c, j}
B_5 = {b, c, m} B_6 = {b, c, j, m}
B_7 = {b, c, j} B_8 = {b, c}

Matching Transactions to Candidate Sets

Algorithm 6.1 Frequent itemset generation of the *Apriori* algorithm.

$k = 1$.
$F_k = \{\ i \mid i \in I \wedge \sigma(\{i\}) \geq N \times minsup\}$. {Find all frequent 1-itemsets}
repeat
    $k = k + 1$.
    $C_k =$ apriori-gen(F_{k-1}). {Generate candidate itemsets}
    for each transaction $t \in T$ **do**
        $C_t =$ subset(C_k, t). {Identify all candidates that belong to t}
        for each candidate itemset $c \in C_t$ **do**
            $\sigma(c) = \sigma(c) + 1$. {Increment support count}
        end for
    end for
    $F_k = \{\ c \mid c \in C_k \wedge \sigma(c) \geq N \times minsup\}$. {Extract the frequent k-itemsets}
until $F_k = \emptyset$
Result $= \bigcup F_k$.

adapted from: Tan, Steinbach & Kumar, "Introduction to Data Mining", http://www-users.cs.umn.edu/~kumar/dmbook/ch6.pdf

A-priori Algorithm

Algorithm 6.1 Frequent itemset generation of the *Apriori* algorithm.

$k = 1.$

$F_k = \{\ i \mid i \in I \wedge \sigma(\{i\}) \geq N \times minsup\}.$ {Find all frequent 1-itemsets}

repeat

Interpret minsup as a fraction here

    $k = k + 1.$

    $C_k = \text{apriori-gen}(F_{k-1}).$ {Generate candidate itemsets}

    for each transaction $t \in T$ **do**

        $C_t = \text{subset}(C_k, t).$ {Identify all candidates that belong to t}

        for each candidate itemset $c \in C_t$ **do**

            $\sigma(c) = \sigma(c) + 1.$ {Increment support count}

        end for

    end for

    $F_k = \{\ c \mid c \in C_k \wedge \sigma(c) \geq N \times minsup\}.$ {Extract the frequent k-itemsets}

until $F_k = \emptyset$

Result $= \bigcup F_k.$

adapted from: Tan, Steinbach & Kumar, “Introduction to Data Mining”, http://www-users.cs.umn.edu/~kumar/dmbook/ch6.pdf

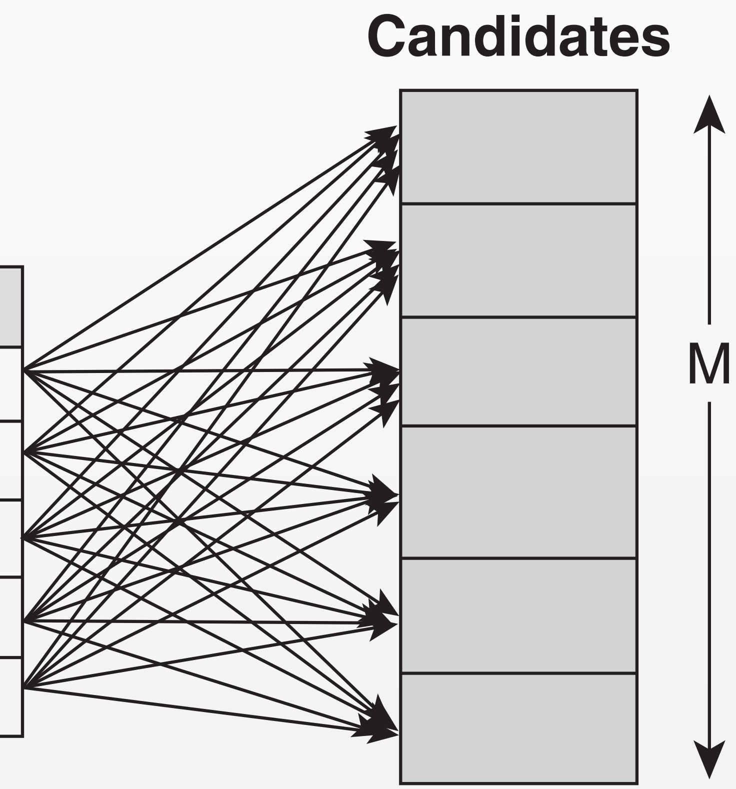
Problem: Naive Matching is Expensive

Cost: O(N M), where N is number of baskets and M is number of candidates

Transactions

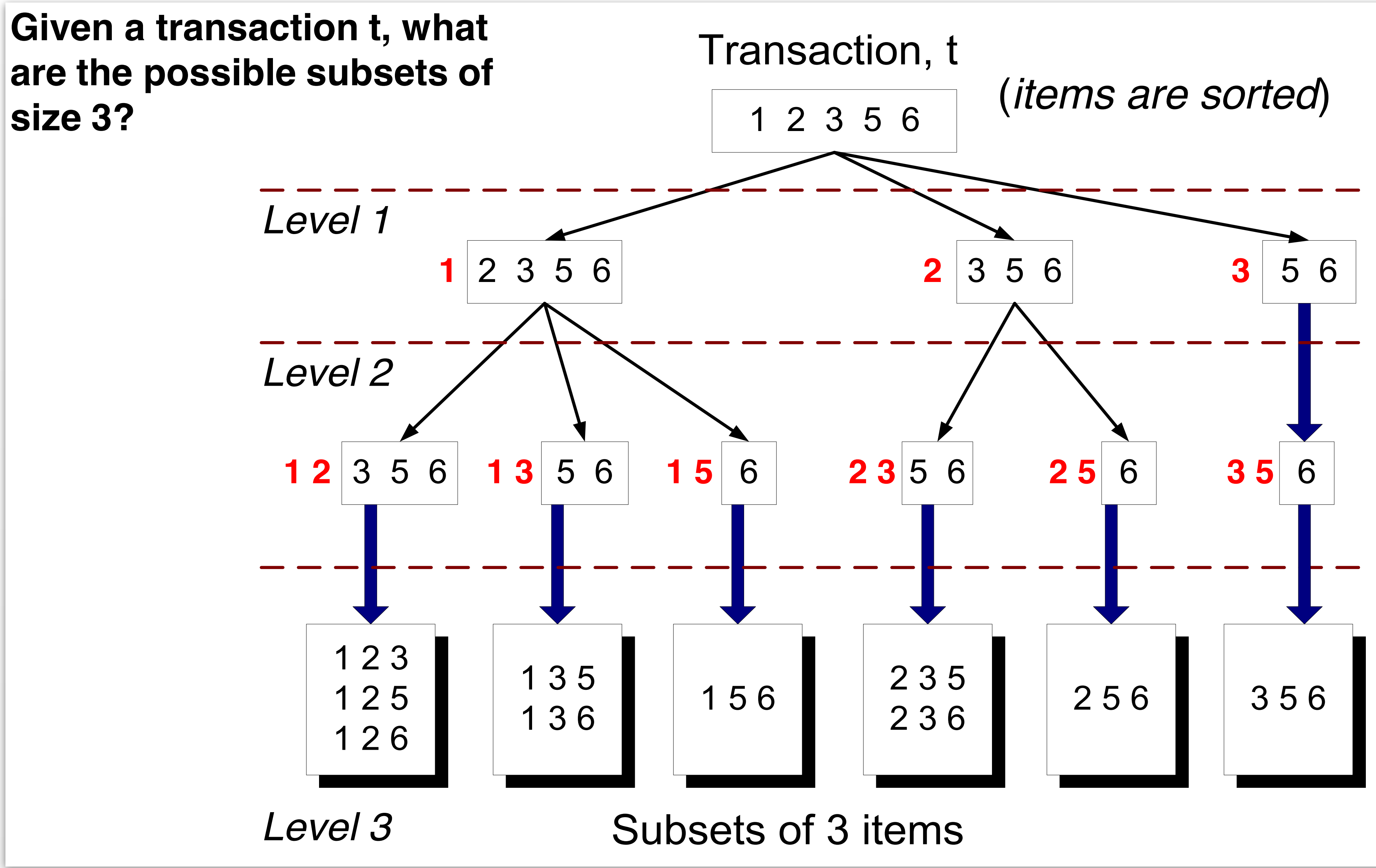
TID	Items
1	Bread, Milk
2	Bread, Diapers, Beer, Eggs
3	Milk, Diapers, Beer, Coke
4	Bread, Milk, Diapers, Beer
5	Bread, Milk, Diapers, Coke

N



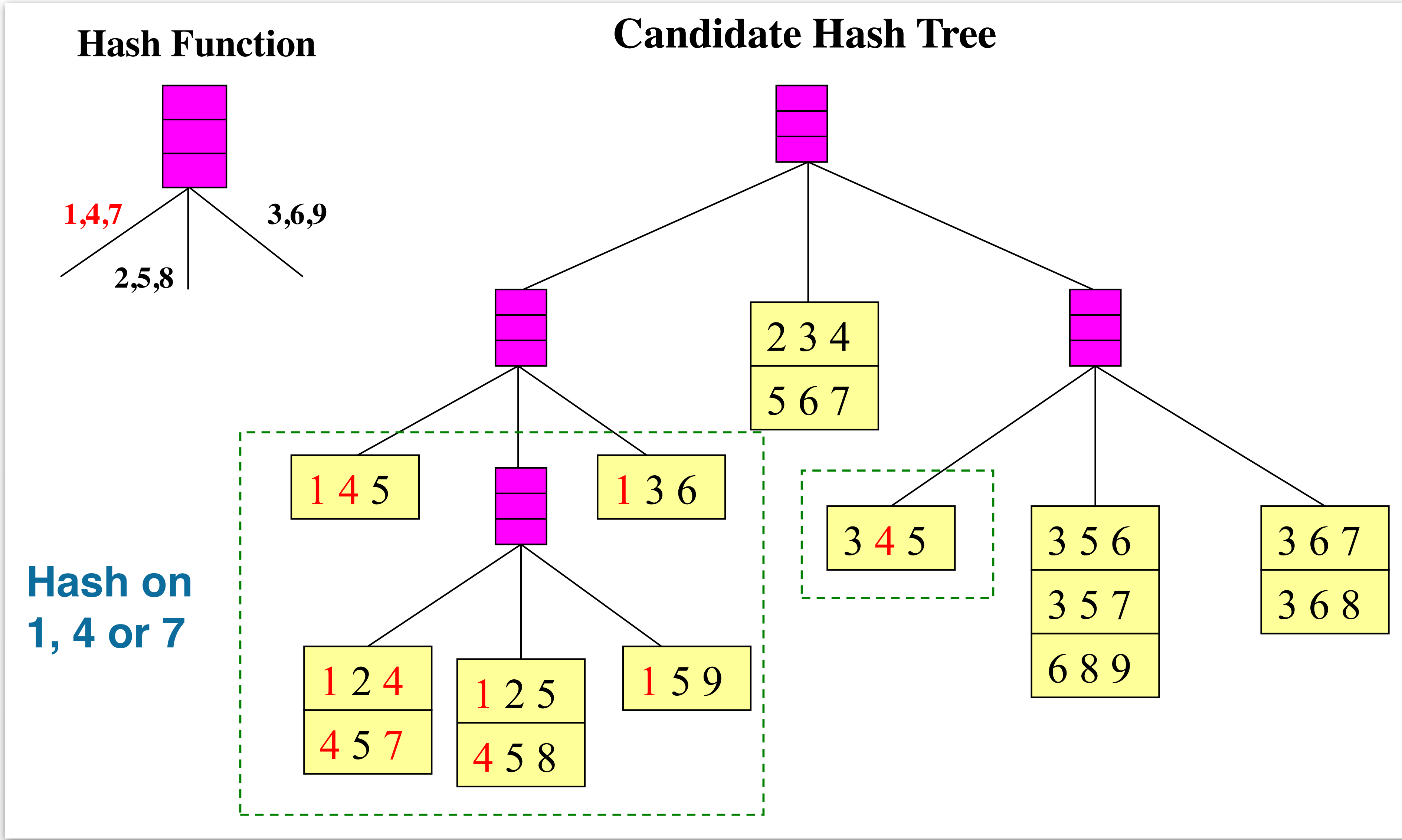
adapted from: Tan, Steinbach & Kumar, "Introduction to Data Mining", http://www-users.cs.umn.edu/~kumar/dmbook/ch6.pdf

Strategy 1: Enumerating Transaction Subsets



adapted from: Tan, Steinbach & Kumar, "Introduction to Data Mining", http://www-users.cs.umn.edu/~kumar/dmbook/ch6.pdf

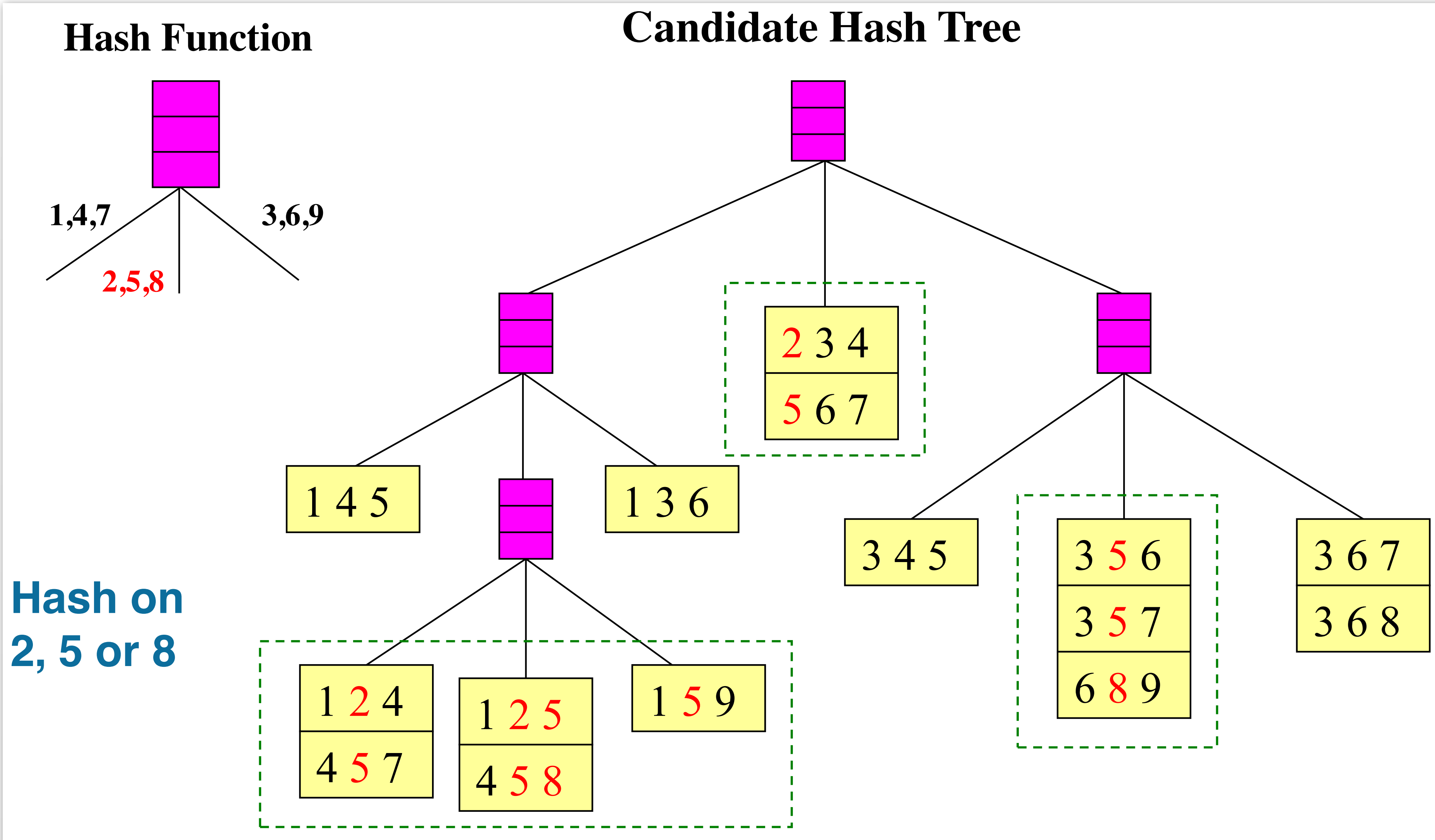
Hash Tree for Itemsets



15 candidate 3-itemsets, distributed across 9 leaf nodes

adapted from: Tan, Steinbach & Kumar, "Introduction to Data Mining", http://www-users.cs.umn.edu/~kumar/dmbook/ch6.pdf

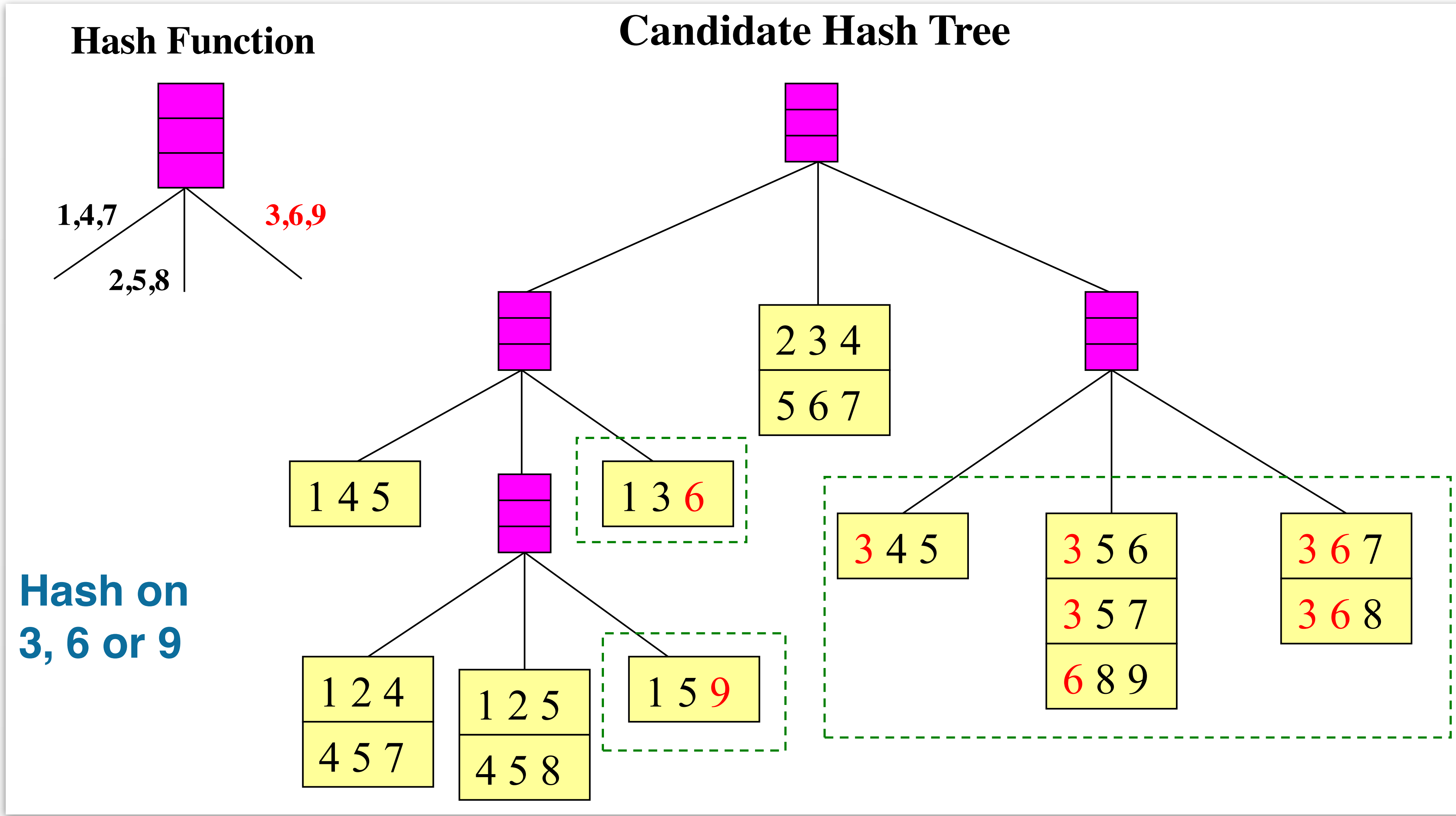
Hash Tree for Itemsets



15 candidate 3-itemsets, distributed across 9 leaf nodes

adapted from: Tan, Steinbach & Kumar, “Introduction to Data Mining”, http://www-users.cs.umn.edu/~kumar/dmbook/ch6.pdf

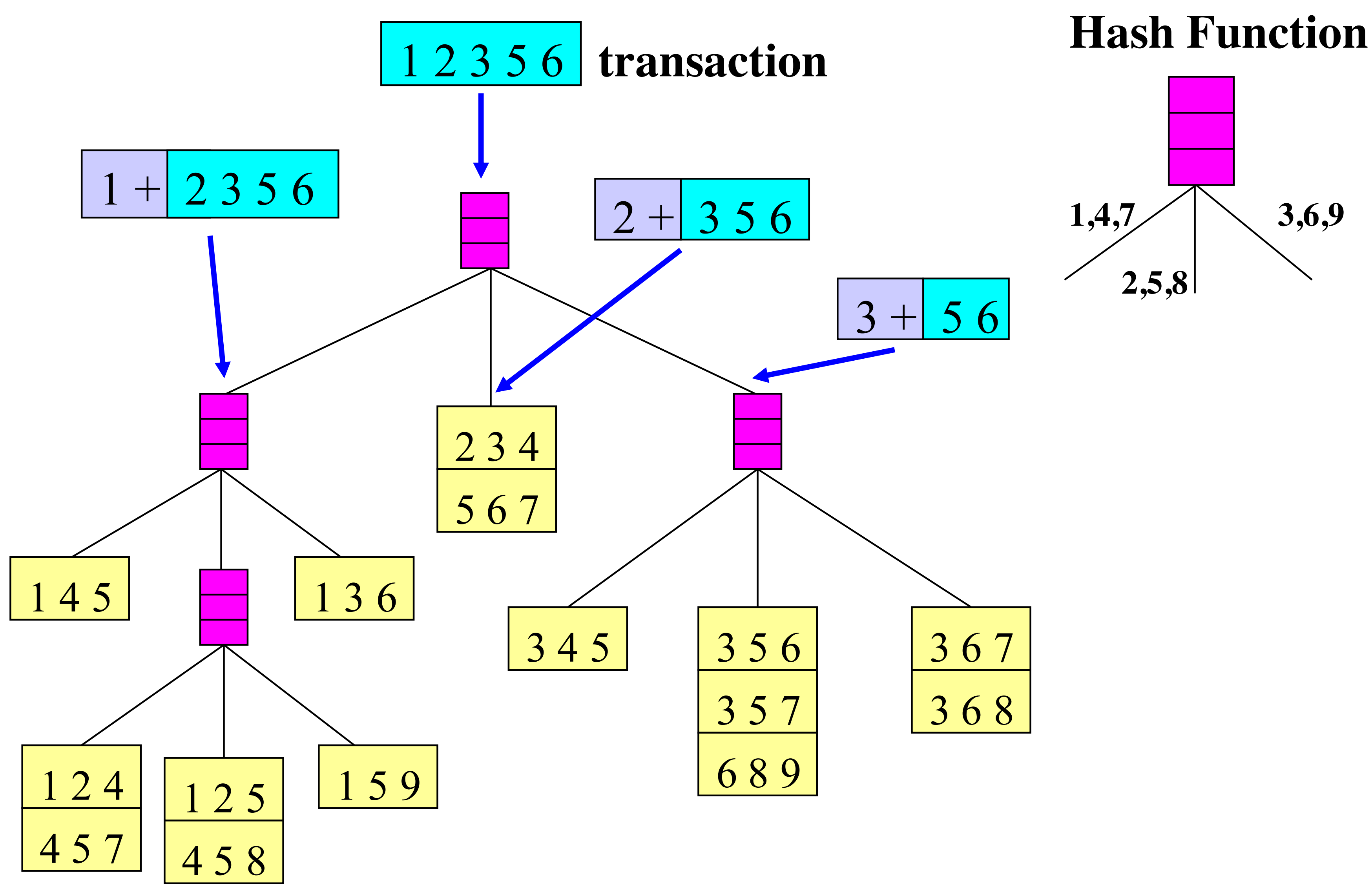
Strategy 2: Hashing Itemsets



15 candidate 3-itemsets, distributed across 9 leaf nodes

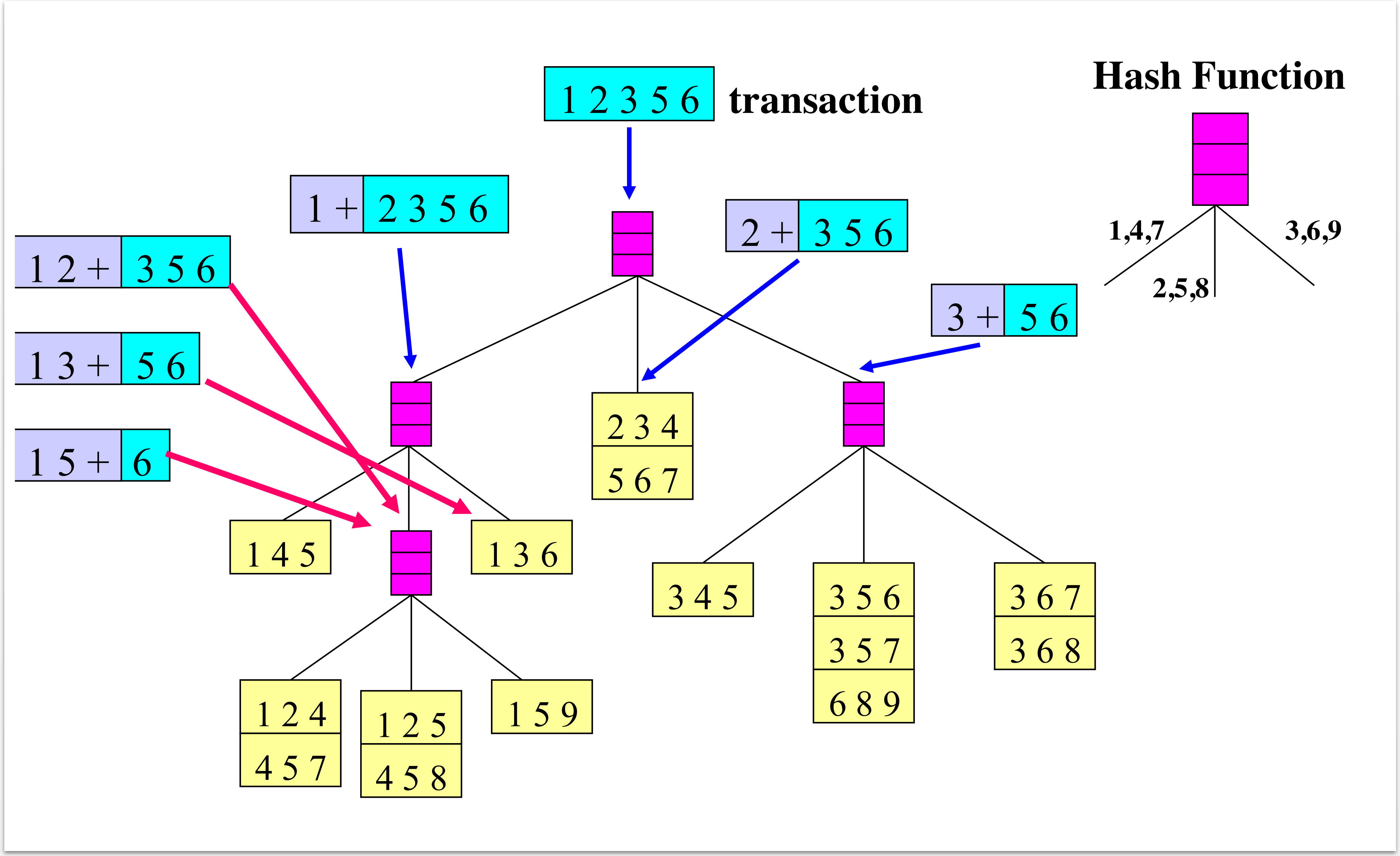
adapted from: Tan, Steinbach & Kumar, "Introduction to Data Mining", http://www-users.cs.umn.edu/~kumar/dmbook/ch6.pdf

Strategy 2: Hash Tree for Candidates



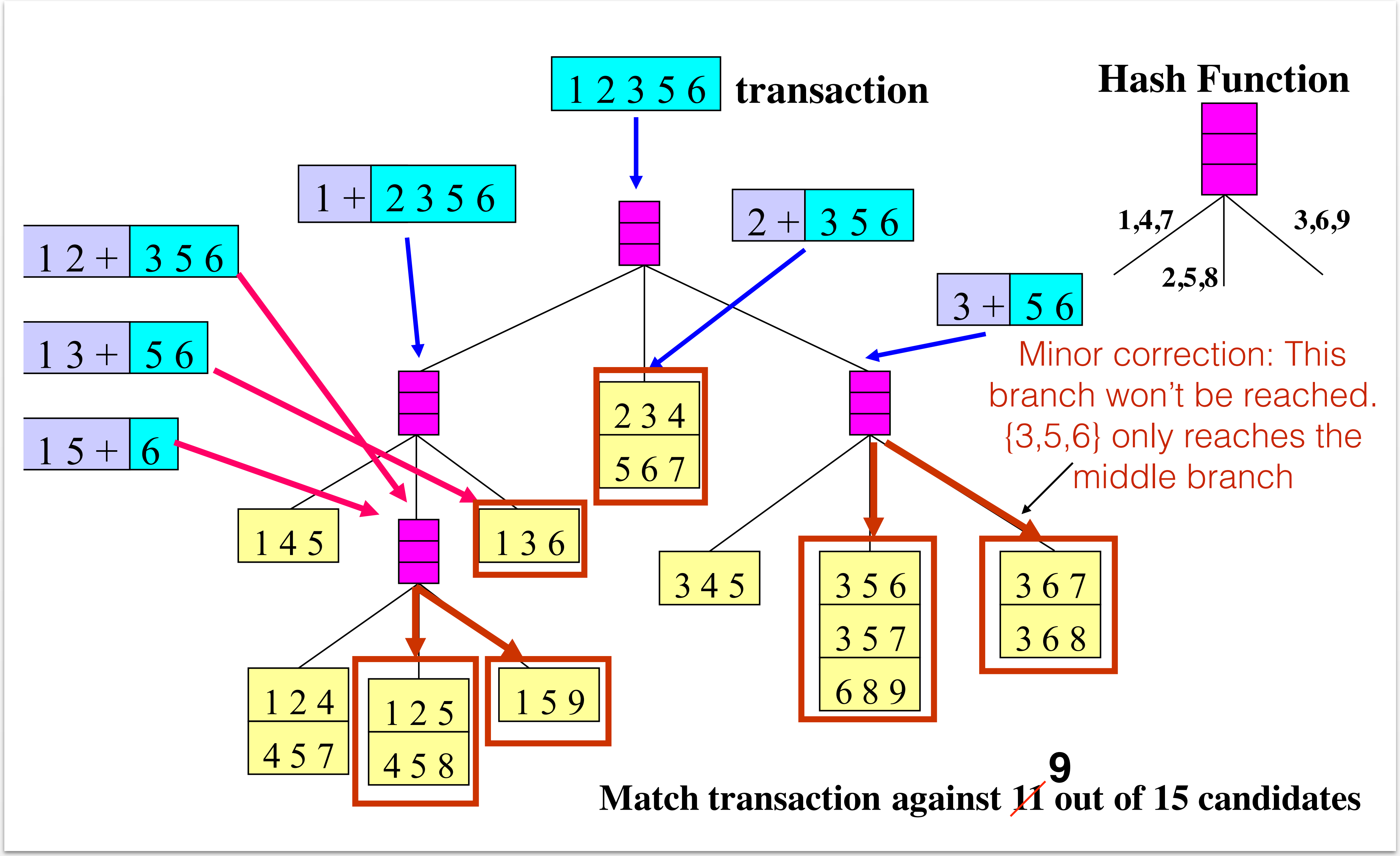
adapted from: Tan, Steinbach & Kumar, "Introduction to Data Mining", http://www-users.cs.umn.edu/~kumar/dmbook/ch6.pdf

Strategy 2: Hash Tree for Candidates



adapted from: Tan, Steinbach & Kumar, "Introduction to Data Mining", http://www-users.cs.umn.edu/~kumar/dmbook/ch6.pdf

Strategy 2: Hash Tree for Candidates



adapted from: Tan, Steinbach & Kumar, "Introduction to Data Mining", http://www-users.cs.umn.edu/~kumar/dmbook/ch6.pdf

A-priori Algorithm

Algorithm 6.1 Frequent itemset generation of the *Apriori* algorithm.

$k = 1.$
$F_k = \{\ i \mid i \in I \wedge \sigma(\{i\}) \geq N \times minsup\}.$ $\quad$ {Find all frequent 1-itemsets}
repeat $\quad$ Interpret minsup as a fraction here
    $k = k + 1.$
    $C_k =$ apriori-gen$(F_{k-1}).$ $\quad$ {Generate candidate itemsets}
    for each transaction $t \in T$ **do**
        $C_t =$ subset$(C_k, t).$ $\quad$ {Identify all candidates that belong to t}
        for each candidate itemset $c \in C_t$ **do**
            $\sigma(c) = \sigma(c) + 1.$ $\quad$ {Increment support count}
        end for
    end for
    $F_k = \{\ c \mid c \in C_k \wedge \sigma(c) \geq N \times minsup\}.$ $\quad$ {Extract the frequent k-itemsets}
until $F_k = \emptyset$
Result $= \bigcup F_k.$

adapted from: Tan, Steinbach & Kumar, "Introduction to Data Mining", http://www-users.cs.umn.edu/~kumar/dmbook/ch6.pdf

Rule Generation

- Items of each frequent itemset Y can be partitioned into the consequent and the the antecedent to give a rule. For an $X \subset Y$
$$X \rightarrow Y - X$$
- $Y = \{a, b, c\}$ could give the six rule $\{a, b\} \rightarrow \{c\}, \{a, c\} \rightarrow \{b\}, \{b, c\} \rightarrow \{a\}, \{a\} \rightarrow \{b, c\}, \{b\} \rightarrow \{a, c\}, \{c\} \rightarrow \{a, b\}$.
- A frequent k-itmeset can potentially give to $2^k - 2$ rules.
- Not all rules are confident

$$C(X \rightarrow Y - X) = \sigma(Y)/\sigma(X) < \text{minconf}$$

Pick a subset of th k items as a consequent. The remaining items become the antecedent. Remo $Y \rightarrow \emptyset$ and $\emptyset \rightarrow$

X is also frequent by anti-monotonicity. However, the rule might not meet the minimum confidence threshold.

- How to find confident association rule without enumerating them all?

Rule Generation

Rule Pruning

- **Theorem 6.2.** *If a rule $X \longrightarrow Y - X$ does not satisfy the confidence threshold, then any rule $X' \longrightarrow Y - X'$, where X' is a subset of X, must not satisfy the confidence threshold as well.*

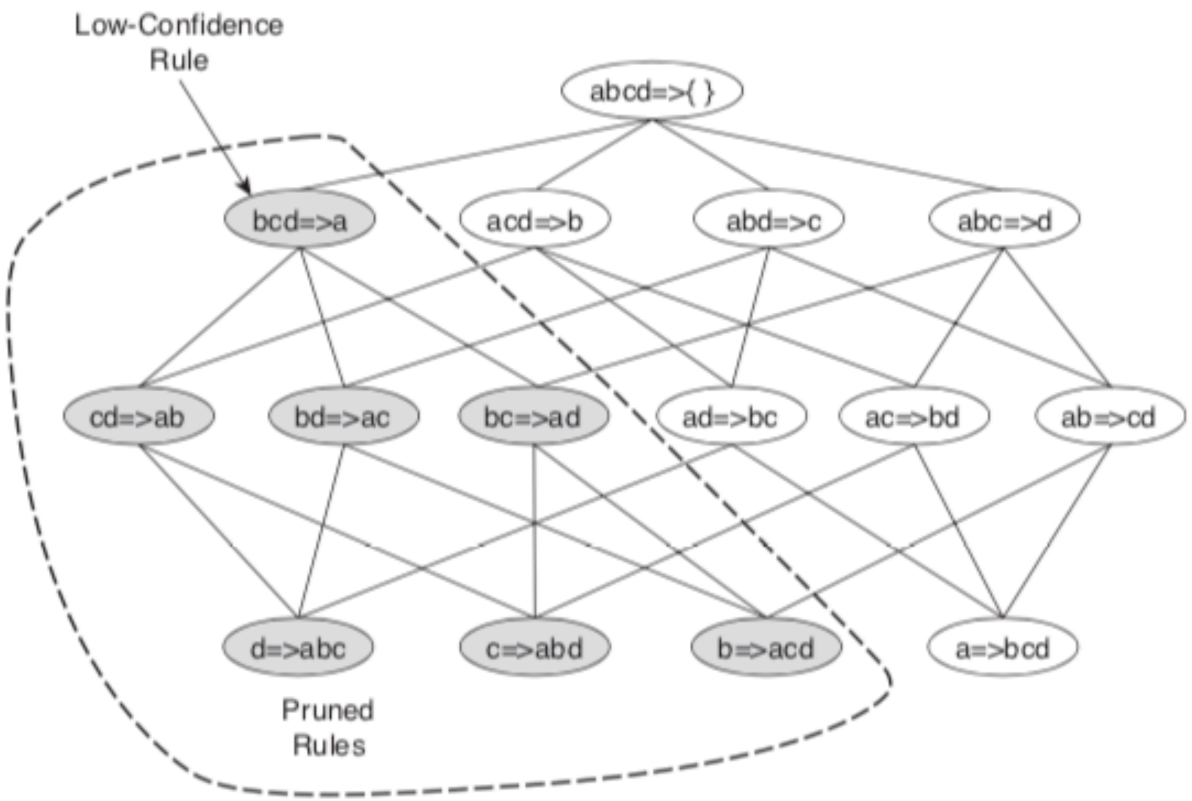


Figure 6.15. Pruning of association rules using the confidence measure.

Rule Generation

Algorithm 6.2 Rule generation of the *Apriori* algorithm.

for each frequent k-itemset f_k, $k \geq 2$ **do**
    $H_1 = \{i \mid i \in f_k\}$ {1-item consequents of the rule.}
    call `ap-genrules`(f_k, H_1.)
end for

Algorithm 6.3 Procedure `ap-genrules`(f_k, H_m).

$k = |f_k|$ {size of frequent itemset.}
$m = |H_m|$ {size of rule consequent.}
if $k > m + 1$ **then**
    $H_{m+1} =$ apriori-gen(H_m).
    for each $h_{m+1} \in H_{m+1}$ **do**
        $conf = \sigma(f_k)/\sigma(f_k - h_{m+1})$.
        if $conf \geq minconf$ **then**
            output the rule $(f_k - h_{m+1}) \longrightarrow h_{m+1}$.
        else
            delete h_{m+1} from H_{m+1}.
        end if
    end for
    call ap-genrules(f_k, H_{m+1}.)
end if

Compacting the Output

- To number of frequent item sets can be exponential in the number of items.
- Might be useful to work with compact representations
- Maximal frequent itemsets: No immediate superset is frequent
    - Gives more pruning

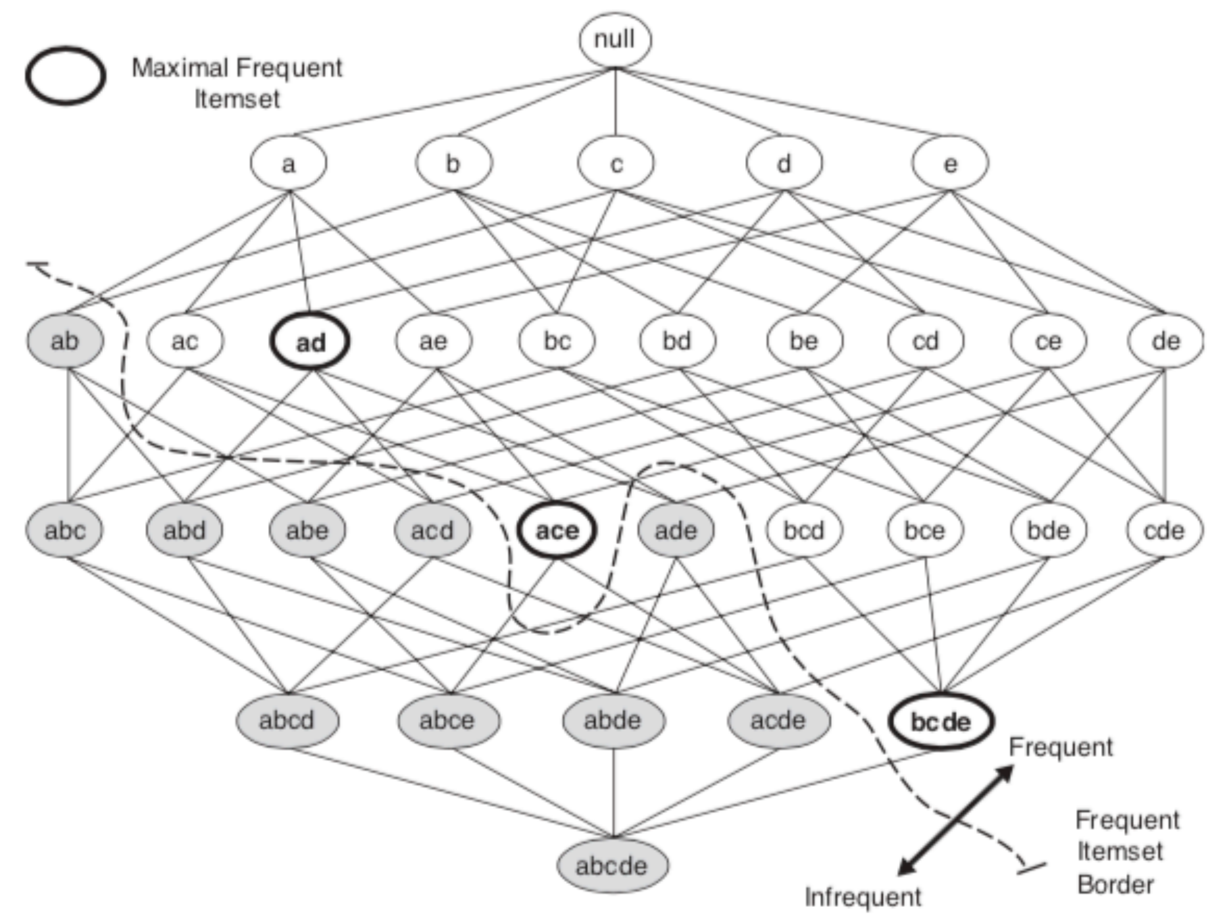


Figure 6.16. Maximal frequent itemset.

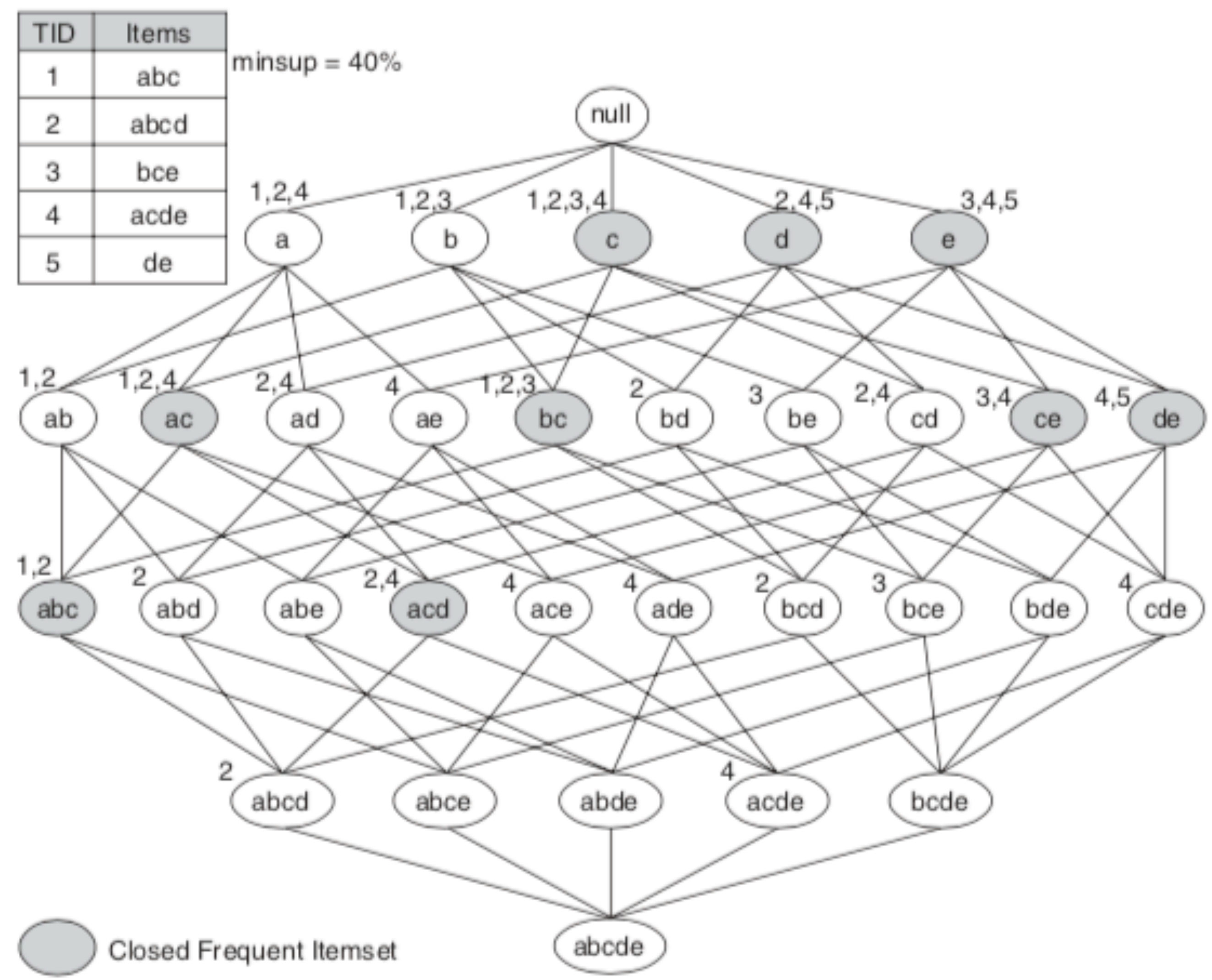
adapted from: J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org

Compacting the Output

Closed frequent itemsets:

- No immediate superset has same count
- Stores not only frequent information, but exact counts
- The counts of non-closed frequent items can be obtained as the maximum of its closed frequent superset
- Redundant association rules are not generated if using closed frequent itemsets.

$\{b\} \rightarrow \{a\}$ and $\{b, c\} \rightarrow \{a\}$ will have the same support and confidence because $\{b\}$ is not closed, but $\{b, c\}$ is



TID	Items
1	abc
2	abcd
3	bce
4	acde
5	de

adapted from: J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org

Example: Maximal vs Closed

$B_1 = \{m, c, b\}$ $B_2 = \{m, p, j\}$

$B_3 = \{m, b\}$ $B_4 = \{c, j\}$

$B_5 = \{m, c, b\}$ $B_6 = \{m, c, b, j\}$

$B_7 = \{c, b, j\}$ $B_8 = \{b, c\}$

Frequent itemsets:

{m}:5, **{c}:6**, **{b}:6**, **{j}:4**,
{m,c}:3, **{m,b}:4**, **{c,b}:5**, **{c,j}:3**,
{m,c,b}:3

Closed

Maximal

Example: Maximal vs Closed