Association (Part I)

nanopoulos@ismll.de

Association Rule Mining

Given a set of transactions, find rules that will predict the occurrence of an item based on the occurrences of other items in the transaction

Market-Basket transactions

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

Example of Association Rules

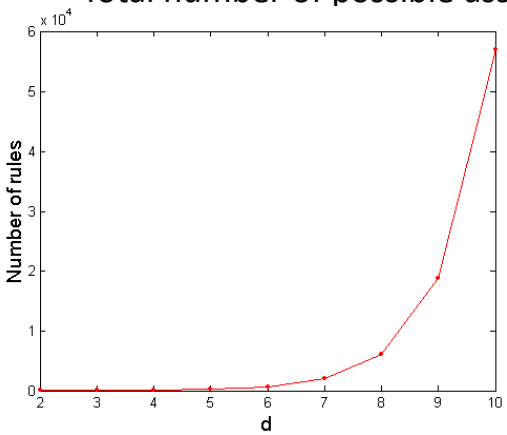
{Diaper} → {Beer},
{Milk, Bread} → {Eggs,Coke},
{Beer, Bread} → {Milk},

Implication means co-occurrence, not causality!

Many possible rules!

Given d unique items:

- Total number of sets of items = 2^d
- Total number of possible association rules:



$$R = \sum_{k=1}^{d-1}\left[\binom{d}{k} \times \sum_{j=1}^{d-k}\binom{d-k}{j}\right]$$

$$= 3^d - 2^{d+1} + 1$$

If d=6, R = 602 rules

Definition: Frequent Itemset

- **Itemset**
  - A collection of one or more items
    - Example: {Milk, Bread, Diaper}
  - k-itemset
    - An itemset that contains k items
- **Support count (σ)**
  - Frequency of occurrence of an itemset
  - E.g. σ({Milk, Bread,Diaper}) = 2
- **Support**
  - Fraction of transactions that contain an itemset
  - E.g. s({Milk, Bread, Diaper}) = 2/5
- **Frequent Itemset**
  - An itemset whose support is greater than or equal to a *minsup* threshold

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

Definition: Association Rule

- Association Rule
  - An implication expression of the form $X \rightarrow Y$, where X and Y are itemsets
  - Example: {Milk, Diaper} → {Beer}

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

- Rule Evaluation Metrics
  - Support (s)
    - Fraction of transactions that contain both X and Y
  - Confidence (c)
    - Measures how often items in Y appear in transactions that contain X

Example:

$$\{\text{Milk}, \text{Diaper}\} \Rightarrow \text{Beer}$$

$$s = \frac{\sigma(\text{Milk}, \text{Diaper}, \text{Beer})}{|\text{T}|} = \frac{2}{5} = 0.4$$

$$c = \frac{\sigma(\text{Milk}, \text{Diaper}, \text{Beer})}{\sigma(\text{Milk}, \text{Diaper})} = \frac{2}{3} = 0.67$$

Association Rule Mining Task

Given a set of transactions T, the goal of association rule mining is to find all rules having

support ≥ *minsup* threshold

confidence ≥ *minconf* threshold

Mining Association Rules

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

Example of Rules:

{Milk,Diaper} → {Beer} (s=0.4, c=0.67)
{Milk,Beer} → {Diaper} (s=0.4, c=1.0)
{Diaper,Beer} → {Milk} (s=0.4, c=0.67)
{Beer} → {Milk,Diaper} (s=0.4, c=0.67)
{Diaper} → {Milk,Beer} (s=0.4, c=0.5)
{Milk} → {Diaper,Beer} (s=0.4, c=0.5)

Observations:

- All the above rules are binary partitions of the same itemset: {Milk, Diaper, Beer}
- Rules originating from the same itemset have identical support but can have different confidence
- Thus, we may decouple the support and confidence requirements

Mining Association Rules

Two-step approach:

1. Frequent Itemset Generation
   - Generate all itemsets whose support ≥ minsup
2. Rule Generation
   - Generate high confidence rules from each frequent itemset, where each rule is a binary partitioning of a frequent itemset

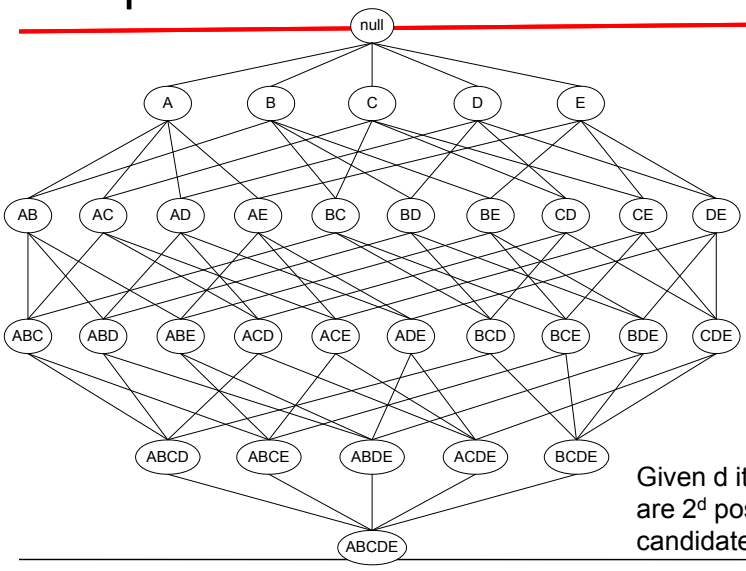
Frequent itemset generation is the most computationally expensive

Generating Frequent Itemsets: Naive algorithm

```
d <- |I|
N <- |D|
for each subset x of I do
    σ(x) <- 0
    for each transaction T in D do
        if x is a subset of T then
            σ(x) <- σ(x) + 1
    if minsup <= σ(x)/N then
        add s to frequent subsets
```

9

The powerset of an itemset



Given d items, there are 2^d possible candidate itemsets

Analysis of naive algorithm

- $O(2^d)$ subsets of *I*
- Scan n transactions for each subset
- $O(2^d n)$ tests of s being subset of T
- Growth is exponential in the number of items!
- Can we do better?

Frequent Itemset Generation Strategies

Reduce the number of candidates (M)

- Complete search: $M=2^d$
- Use pruning techniques to reduce M

Reduce the number of comparisons (NM)

- Use efficient data structures to store the candidates or transactions
- No need to match every candidate against every transaction

Reducing Number of Candidates

Apriori principle:

If an itemset is frequent, then all of its subsets must also be frequent

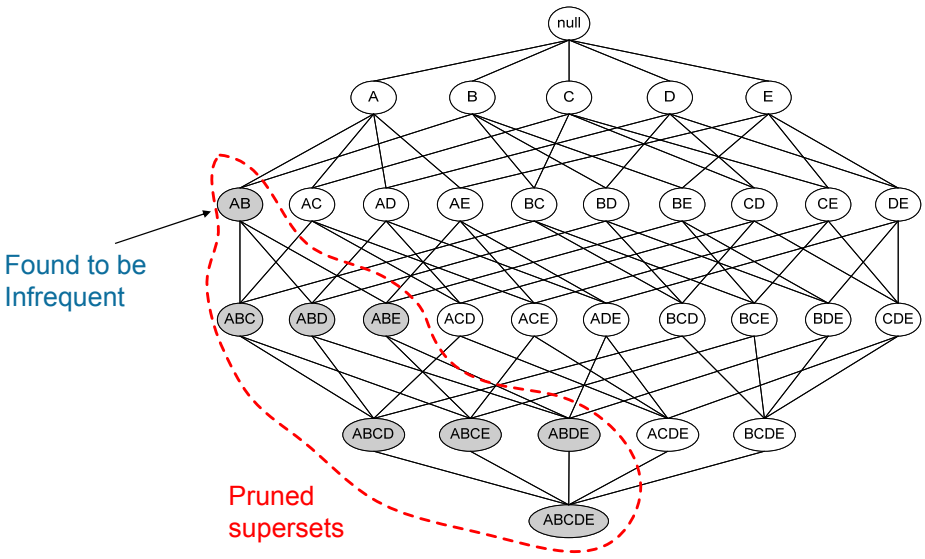
Apriori principle holds due to the following property of the support measure:

$$\forall X, Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y)$$

Support of an itemset never exceeds the support of its subsets

This is known as the anti-monotone property of support

Illustrating Apriori Principle



Illustrating Apriori Principle

Item	Count
Bread	4
Coke	2
Milk	4
Beer	3
Diaper	4
Eggs	1

Items (1-itemsets)

Itemset	Count
{Bread,Milk}	3
{Bread,Beer}	2
{Bread,Diaper}	3
{Milk,Beer}	2
{Milk,Diaper}	3
{Beer,Diaper}	3

Pairs (2-itemsets)

(No need to generate candidates involving Coke or Eggs)

Minimum Support = 3

Triplets (3-itemsets)

Itemset	Count
{Bread,Milk,Diaper}	3

If every subset is considered,
$^6C_1 + {}^6C_2 + {}^6C_3 = 41$
With support-based pruning,
6 + 6 + 1 = 13

The Apriori Algorithm

Join Step: C_k is generated by joining $L_{k-1}$with itself

Prune Step: Any (k-1)-itemset that is not frequent cannot be a subset of a frequent k-itemset

Pseudo-code:

C_k: Candidate itemset of size k
L_k : frequent itemset of size k

L_1 = {frequent items};
for (k = 1; L_k !=∅; k++) **do begin**
  C_{k+1} = candidates generated from L_k;
  for each transaction t in database do
    increment the count of all candidates in C_{k+1} that are contained in t
  L_{k+1} = candidates in C_{k+1} with min_support
  end
return $\cup_k L_k$;

16

How to Generate Candidates?

Suppose the items in L_{k-1} are listed in an order

Step 1: self-joining L_{k-1}

insert into C_k

select $p.item_1, p.item_2, \ldots, p.item_{k-1}, q.item_{k-1}$

from $L_{k-1}\ p, L_{k-1}\ q$

where $p.item_1=q.item_1, \ldots, p.item_{k-2}=q.item_{k-2}, p.item_{k-1} < q.item_{k-1}$

Step 2: pruning

forall ***itemsets c in*** C_k do

forall ***(k-1)-subsets s of c*** do

if *(s is not in* L_{k-1}*)* **then delete** *c* **from** C_k

17

Example of Generating Candidates

L_3={*abc, abd, acd, ace, bcd*}

Self-joining: L_3*L_3

abcd from *abc* and *abd*

acde from *acd* and *ace*

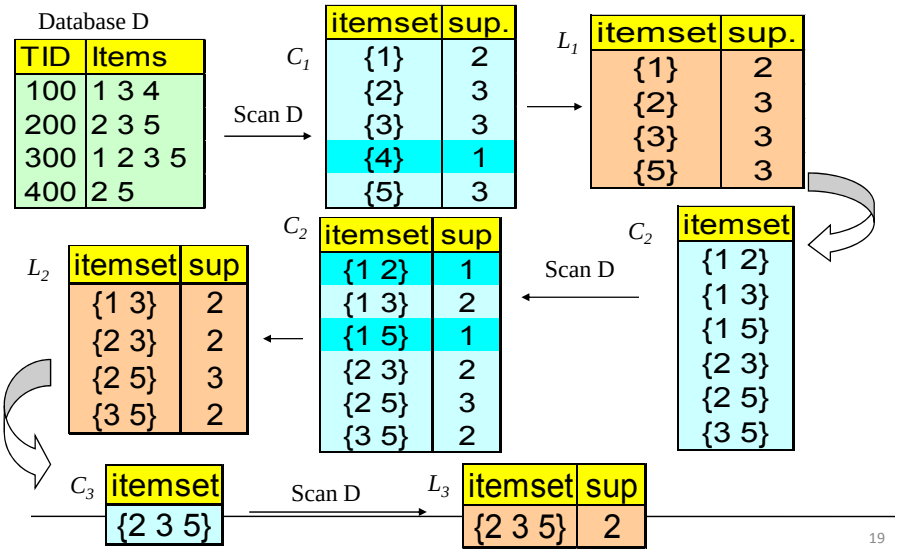
Pruning:

acde is removed because *ade* is not in L_3

C_4={*abcd*}

18

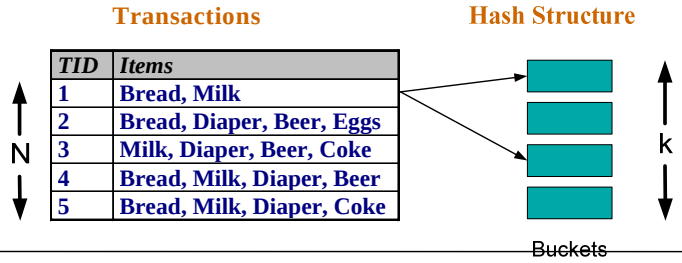
The Apriori Algorithm — Example



Reducing Number of Comparisons

Candidate counting:

- Scan the database of transactions to determine the support of each candidate itemset
- To reduce the number of comparisons, store the candidates in a hash structure
  - Instead of matching each transaction against every candidate, match it against candidates contained in the hashed buckets



Generate Hash Tree

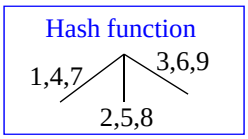
Suppose you have 15 candidate itemsets of length 3:

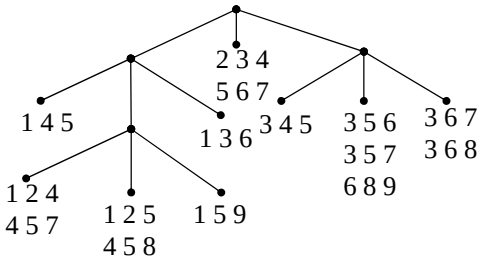
{1 4 5}, {1 2 4}, {4 5 7}, {1 2 5}, {4 5 8}, {1 5 9}, {1 3 6}, {2 3 4}, {5 6 7}, {3 4 5}, {3 5 6}, {3 5 7}, {6 8 9}, {3 6 7}, {3 6 8}

You need:

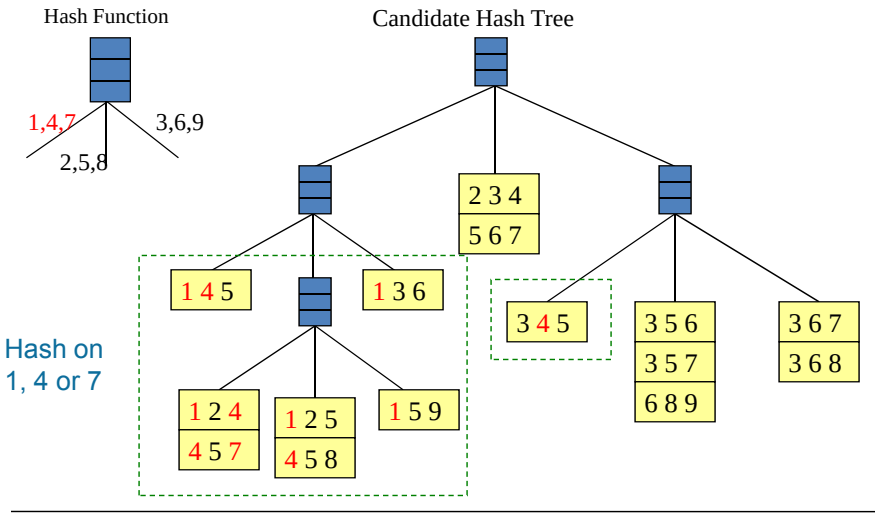
• Hash function

• Max leaf size: max number of itemsets stored in a leaf node (if number of candidate itemsets exceeds max leaf size, split the node)

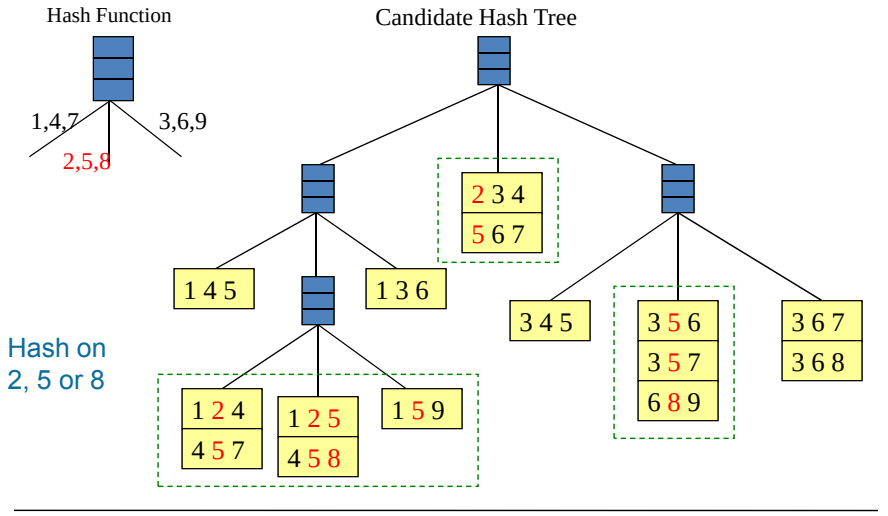




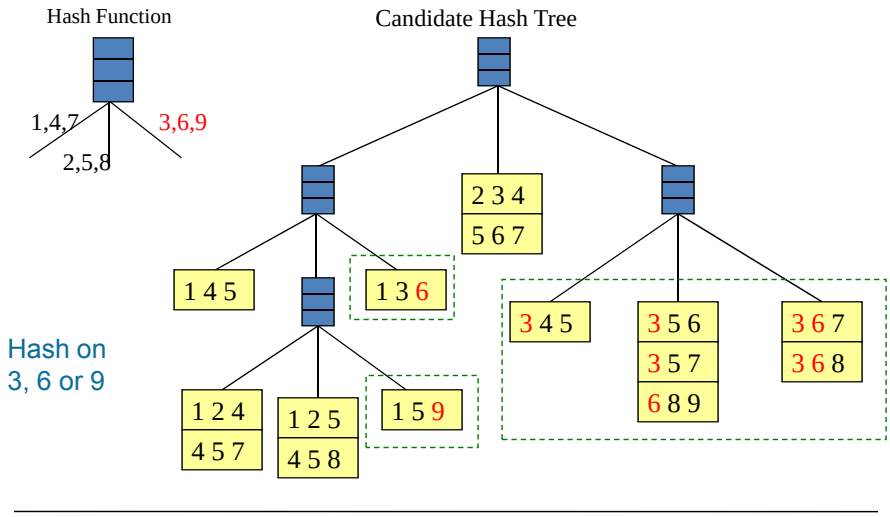
Hash tree



Hash tree

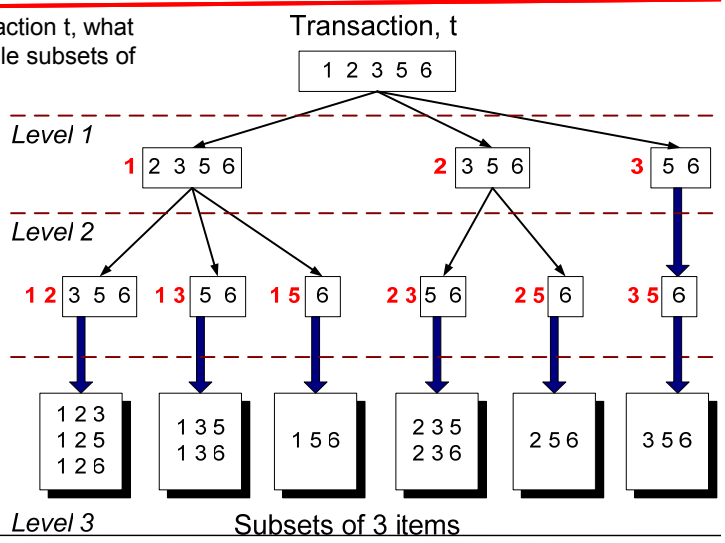


Hash tree

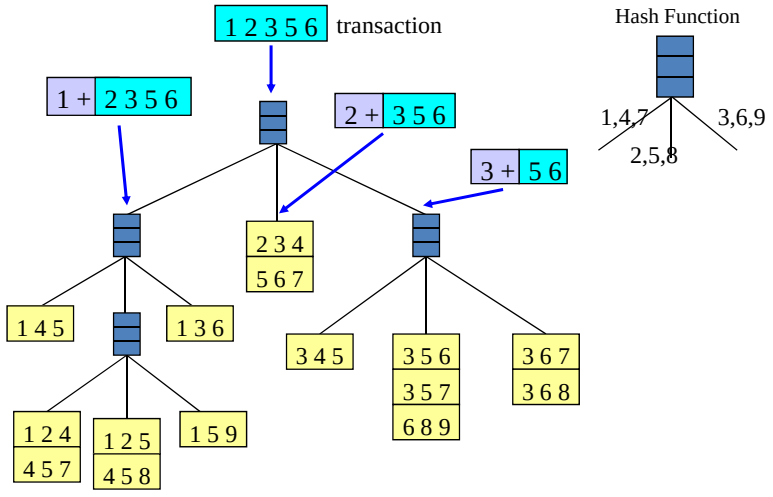


Subset Operation

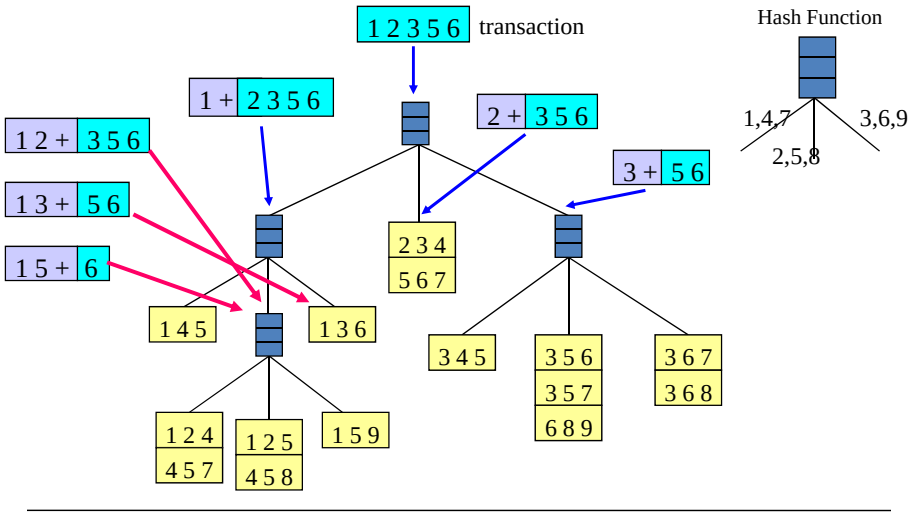
Given a transaction t, what are the possible subsets of size 3?



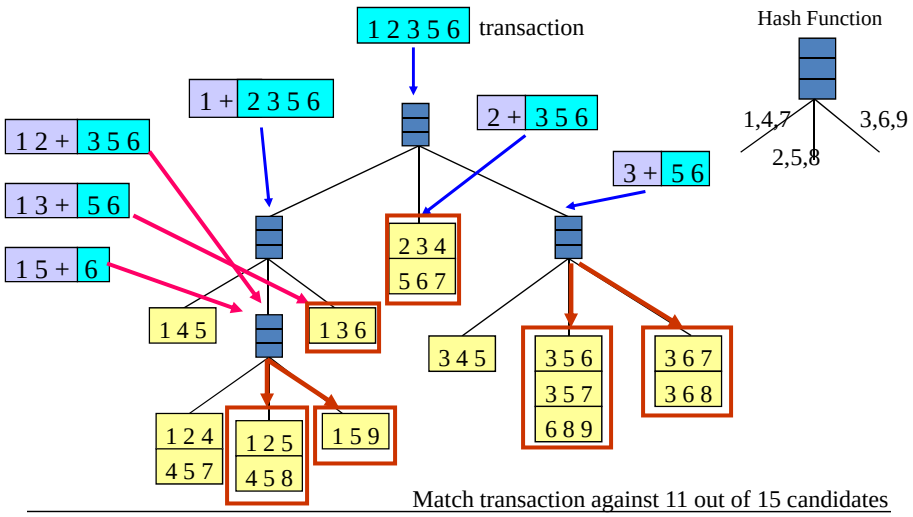
Subset Operation Using Hash Tree



Subset Operation Using Hash Tree



Subset Operation Using Hash Tree



Factors Affecting Complexity

Choice of minimum support threshold
- lowering support threshold results in more frequent itemsets
- this may increase number of candidates and max length of frequent itemsets

Dimensionality (number of items) of the data set
- more space is needed to store support count of each item
- if number of frequent items also increases, both computation and I/O costs may also increase

Size of database
- since Apriori makes multiple passes, run time of algorithm may increase with number of transactions

Average transaction width
- transaction width increases with denser data sets
- This may increase max length of frequent itemsets and traversals of hash tree (number of subsets in a transaction increases with its width)

Generating rules (2nd sub-problem)

Given a frequent itemset L, find all non-empty subsets $f \subset L$ such that $f \rightarrow L - f$ satisfies the minimum confidence requirement

If {A,B,C,D} is a frequent itemset, candidate rules:

ABC $\rightarrow$D,	ABD $\rightarrow$C,	ACD $\rightarrow$B,	BCD $\rightarrow$A,
A $\rightarrow$BCD,	B $\rightarrow$ACD,	C $\rightarrow$ABD,	D $\rightarrow$ABC
AB $\rightarrow$CD,	AC $\rightarrow$ BD,	AD $\rightarrow$ BC,	BC $\rightarrow$AD,
BD $\rightarrow$AC,	CD $\rightarrow$AB,		

If $|L| = k$, then there are $2^k - 2$ candidate association rules (ignoring $L \rightarrow \varnothing$ and $\varnothing \rightarrow L$)

30

Rule Generation with anti-monotone property

How to efficiently generate rules from frequent itemsets?

In general, confidence does not have an anti-monotone property

c(ABC →D) can be larger or smaller than c(AB →D)

But confidence of rules generated from the same itemset has an anti-monotone property

e.g., L = {A,B,C,D}:

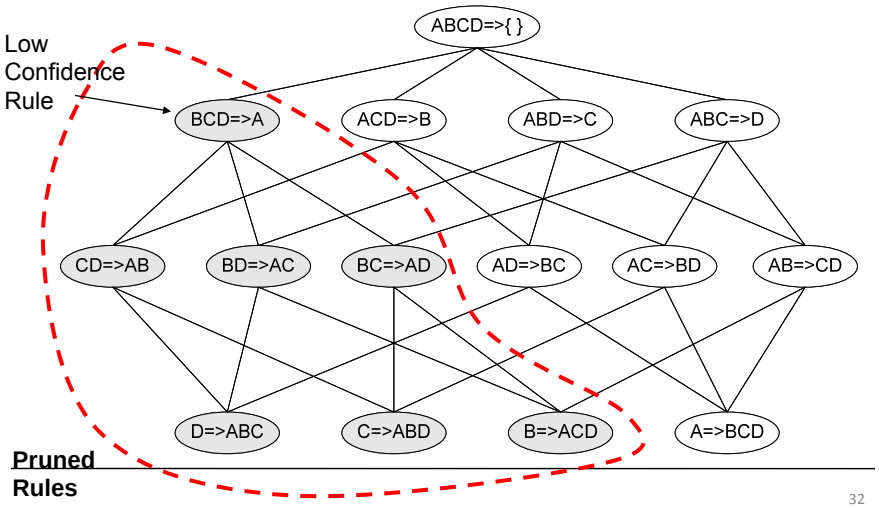
$$c(ABC \rightarrow D) \geq c(AB \rightarrow CD) \geq c(A \rightarrow BCD)$$

Confidence is anti-monotone w.r.t. number of items on the RHS of the rule

31

Rule Generation: example of anti-monotonicity

Lattice of rules



32