Analytics and Visualization of Big Data

Fadel M. Megahed

Lecture 08: Frequent Itemset Mining and Association Rules

AUBURN UNIVERSITY
SAMUEL GINN
COLLEGE OF ENGINEERING

Department of Industrial and Systems Engineering

Spring 13

Outline for Topics Covered in Chapter 06

What is the Market-Basket Model?

- Definition of Frequent Itemsets
- Applications of Frequent Itemsets
- Association Rules
- Finding Association Rules with High Confidence

Market-Baskets and A-Priori Alg.

- Representation of Market-Basket Data
- Main Memory for Itemset Counting
- Monotoncity of Itemsets
- Tyranny of Counting Pairs
- The A-Priori Algorithm and for All Frequent Itemsets

Association Rule Discovery – The Market-Basket Model

- **Goal:** Identify items frequently bought together

- **Approach:** Process the sales data collected with barcode scanners to find dependencies among items
  - **A classic rule:** If one buys diaper and milk, then (s)he is likely to buy beer!! – **Layout: Putting 6-pack next to diapers?**

transaction id	customer id	products bought
tran1	cust33	p2, p5, p8
tran2	cust45	p5, p8, p11
tran3	cust12	p1, p9
tran4	cust40	p5, p8, p11
tran5	cust12	p2, p9
tran6	cust12	p9

→

Rules Discovered:
Products 5 and 8 are often bought together

The Market-Basket Model

tran1	cust33	p2, p5, p8
tran2	cust45	p5, p8, p11
tran3	cust12	p1, p9
tran4	cust40	p5, p8, p11
tran5	cust12	p2, p9
tran6	cust12	p9

- A large set of items
  - e.g., things sold in a supermarket
- A large set of baskets, each is a small subset of items
  - e.g., things a customer buys in one transaction
- A general many-many mapping (association) between two kinds of things
  - We look for connections among "items," not "baskets"

Association Rules – Approach

- Given a set of **baskets**, we want to discover **association rules**
  - People who bought {p5} tend to buy {p8}
    - Web Marketing in Amazon
- 2 step approach:
  - Find **frequent itemsets**
  - Generate **association rules**
- Input:

tran1	cust33	p2, p5, p8
tran2	cust45	p5, p8, p11
tran3	cust12	p1, p9
tran4	cust40	p5, p8, p11
tran5	cust12	p2, p9
tran6	cust12	p9

- Output:
  - Discovered Rule: Product 5 and 8 are likely to be bought together

Applications Related to INSY – Healthcare

- **Baskets** = patients; **items** = drugs & side-effects
  - Used to detect combinations of drugs that result in particular side-effects

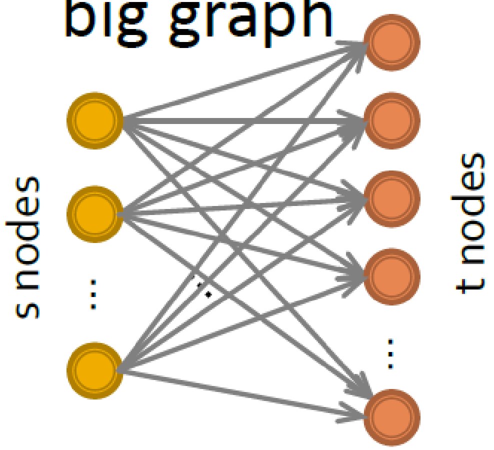
- **But requires extension**: absence of an item needs to be observed as well as presence.

PID	D1	D2	D3	D4	D5	D6	SE1	SE2
242	Yes	Yes	Yes	No	No	No	No	Yes
231	No	No	Yes	No	No	No	No	Yes
339	No	No	No	Yes	Yes	Yes	Yes	No
157	No	No	No	No	No	No	No	No
638	No	No	Yes	No	Yes	Yes	Yes	Yes
247	No	No	No	No	No	Yes	No	No
241	No	No	No	No	Yes	Yes	Yes	No

Finding communities in graphs (e.g., web)

Baskets = nodes; **items** = outgoing neighbors

- Searching for complete bipartite subgraphs $K_{s,t}$ of a big graph



A dense 2-layer graph

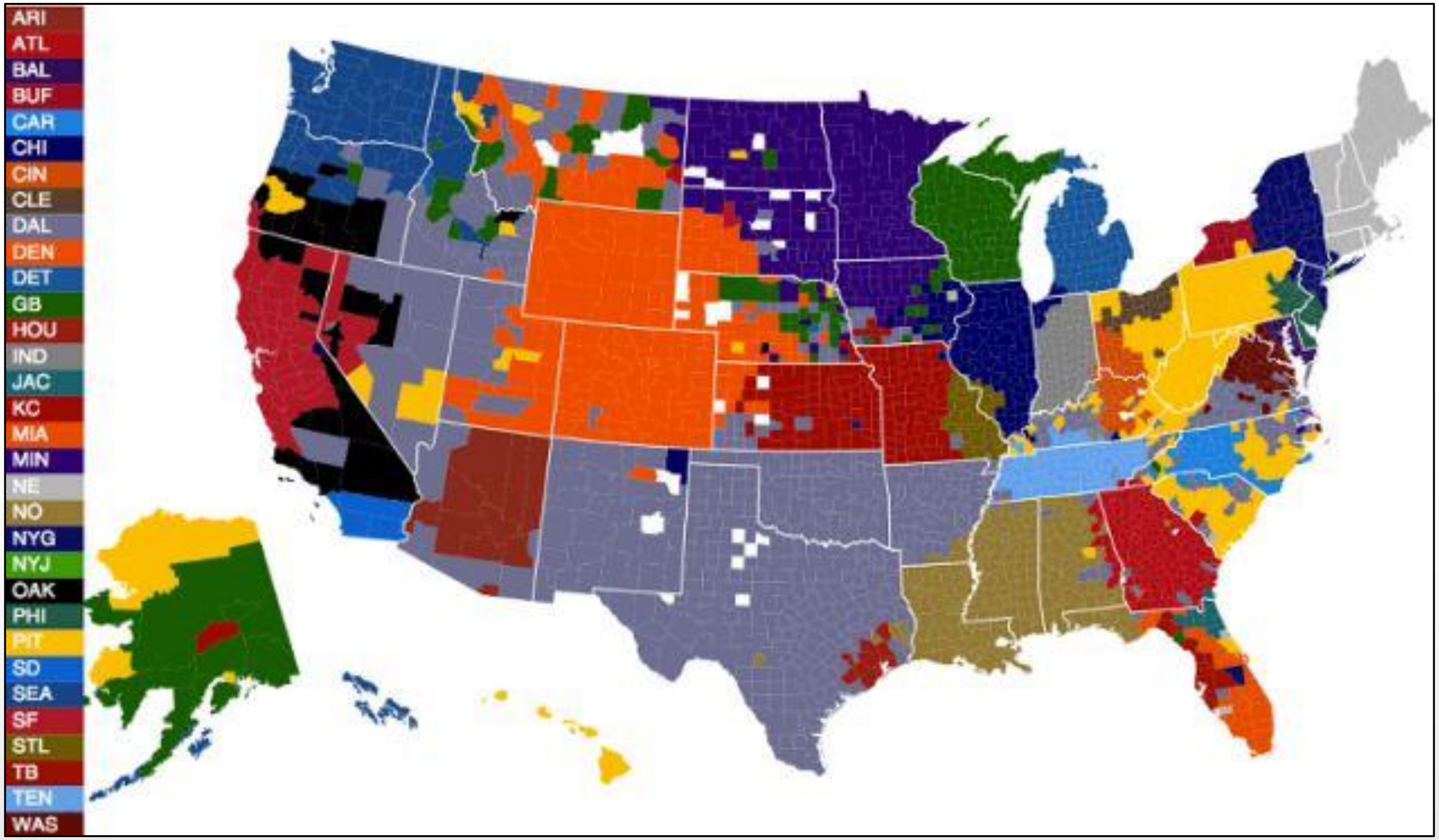
Use this to define topics:
What the same people on the left talk about on the right

- **How?**
  - View each node i as a bucket B_i of nodes i it points to
  - $K_{s,t}$ = a set Y of size t that occurs in s buckets B_i
  - Looking for $K_{s,t}$ → set of support s and look at layer t – all frequent sets of size t

Source: Figure from Jure Leskovic, Stanford CS246, Lecture Notes, see http://cs246.stanford.edu

Applications Related to INSY – Social Media Analysis

Using Facebook information, NYU PHd candidate Sean Taylor has generated a new series of maps showing the social and geographical breakdown of NFL fans



Source: Figure from http://www.dailymail.co.uk/news/article-2269915/Facebook-map-shows-USA-divided-NFL--enclave-Pittsburgh-fans-Oregon.html#axzz2K1yRIloN

Applications Related to INSY – Social Media-Like Analysis

Source: video.ted.com/talk/podcast/2012G/None/MalteSpitz_2012G.mp4

A Formal Definition for Frequent Itemsets

- **Simplest question:** Find sets of items that appear together "frequently" in baskets

- **Support** for itemset I: number of baskets containing all items in I
  - Often expressed as a fraction of the total number of baskets

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

Support of
{Beer, Bread} = 2

- Given a **support threshold *s***, then sets of items that appear in at least *s* baskets are called **frequent itemsets**

Example: Frequent Itemsets

- **Items** = {milk, coke, pepsi, beer, juice}

- **Minimum support** = 3 baskets

  B1 = {m, c, b}　　B2 = {m, p, j}
  B3 = {m, b}　　B4= {c, j}
  B5 = {m, p, b}　　B6 = {m, c, b, j}
  B7 = {c, b, j}　　B8 = {b, c}

- **What are the frequent itemsets?**

A Formal Definition of Association Rules

- Association Rules: If-then rules about the contents of baskets
- {i1, i2,…,ik} → j means: "if a basket contains all of i1,…,ik then it is likely to contain j"
- Confidence of this association rule is the probability of j given I = {i1,…,ik}

$$\boxed{\mathrm{conf}(I \rightarrow j) = \frac{\Pr[I \cup j]}{\Pr[I]} = \frac{\mathrm{support}(I \cup j)}{\mathrm{support}(I)}}$$

Source: Slide Adapted Jure Leskovic, Stanford CS246, Lecture Notes, see http://cs246.stanford.edu

Finding Association Rules

- Process: "all association rules with support ≥ s and confidence ≥ c." Based on this definition, we can see that the process is really made of two steps:
  - Step 1: Find all frequent itemsets I
    - Generate all itemsets whose support ≥ min support
  - Step 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 still computationally expensive

Example: Association Rules – Confidence and Interest

- **Items** = {milk, coke, pepsi, beer, juice}
- **Minimum support** = 3 baskets

  B1 = {m, c, b} B2 = {m, p, j}
  B3 = {m, c, b, n} B4= {c, j}
  B5 = {m, p, b} B6 = {m, c, b, j}
  B7 = {c, b, j} B8 = {b, c}

- **Generate the association rules (s=3, c=0.75)**
  - **Frequent itemsets:**
  - **Generate Rules:**

Interesting Association Rules → Bonferroni's Principle??

- **Not all high-confidence rules are interesting**
  - The rule $X \to milk$ may have high confidence for many itemsets X, because milk is just purchased very often (independent of X)

- **Interest** of an association rule $I \to j$: difference between its confidence and the fraction of baskets that contain j

$$\boxed{\text{Interest}(I \to j) = \text{conf}(I \to j) - \Pr[j]}$$

- Interesting rules are those with high positive or negative interest values

Compacting the Output: An Exercise

- **Maximal Frequent itemsets**: no immediate superset is frequent

- **Closed itemsets:** no immediate superset has the same count (count>0).

Itemset	Count
A	4
B	5
C	3
AB	4
AC	2
BC	3
ABC	2

For the above table, identify whether each of the following itemsets is frequent, maximal and/or closed. Use $s \geq 3$ to decide on whether an item is frequent or not

Outline for Topics Covered in Chapter 06

What is the Market-Basket Model?

- Definition of Frequent Itemsets
- Applications of Frequent Itemsets
- Association Rules
- Finding Association Rules with High Confidence

Market-Baskets and A-Priori Alg.

- Representation of Market-Basket Data
- Main Memory for Itemset Counting
- Monotoncity of Itemsets
- Tyranny of Counting Pairs
- The A-Priori Algorithm and for All Frequent Itemsets

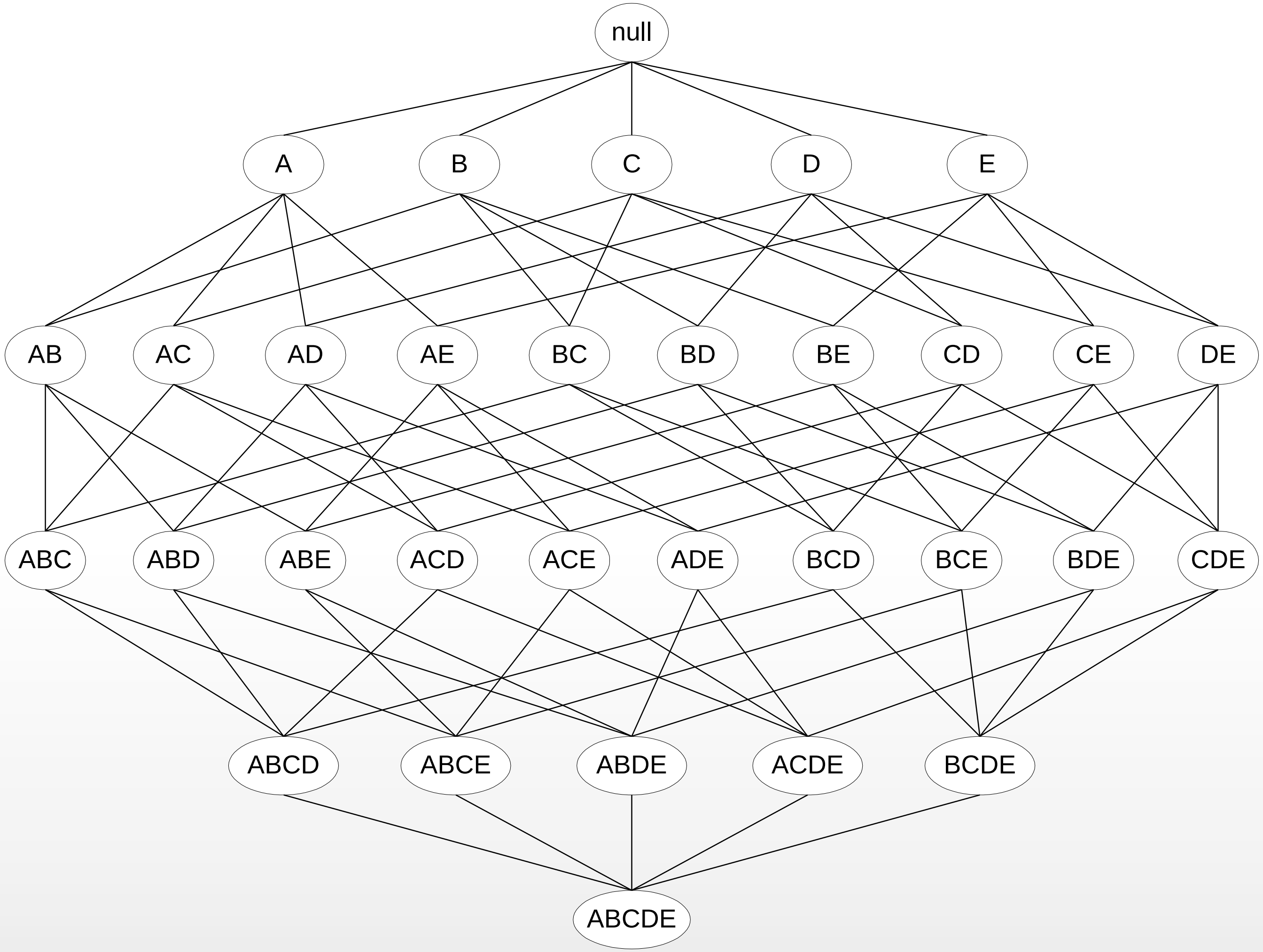
Main-Memory Bottlenecks

- In many algorithms to find frequent itemsets we need to worry about how **main memory** is used.
  - As we read baskets, we need to count something, e.g., occurrences of pairs.
  - The number of different things we can count is limited by the main memory.
  - Swapping counts in/out is a disaster.

Finding Frequent Pairs

- **The hardest problem often turns out to be finding the frequent pairs of items** $\{i_1, i_2\}$
- We'll concentrate on how to do that, then discuss extensions to finding frequent triples, etc.
- The approach:
  - We always need to generate all the itemsets
  - But we would only like to count/keep track of those itemsets that in the end turn out to be frequent

The Lattice of Itemsets



Source: figure from Shivnath Babu, Duke CPS196.3, Lecture Notes, see http://www.cs.duke.edu/courses/spring09/cps196.3/courseoutline.html

Naïve Algorithm

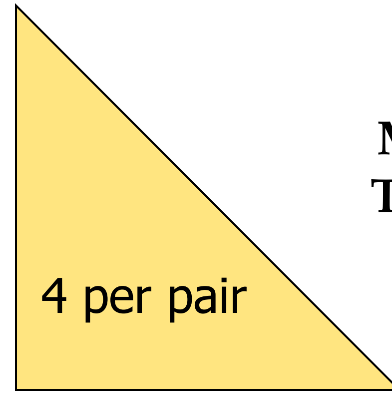
- Naïve approach to finding frequent pairs
- Read file once, counting in main memory the occurrences of each pair:
  - From each basket of n items, generate its n(n-1)/2 pairs by two nested loops
  - Fails if $(\#items)^2$ exceeds main memory
- Note: #items can be 100K (Wal-Mart) or 10B (Web pages)
  - Suppose 10^5 items, counts are 4-byte integers
  - Number of pairs of items: $10^5(10^5\text{-}1)/2 = 5*10^9$
  - Therefore, $2*10^{10}$ (20 gigabytes) of memory needed

Source: Slide Adapted Jure Leskovic, Stanford CS246, Lecture Notes, see http://cs246.stanford.edu

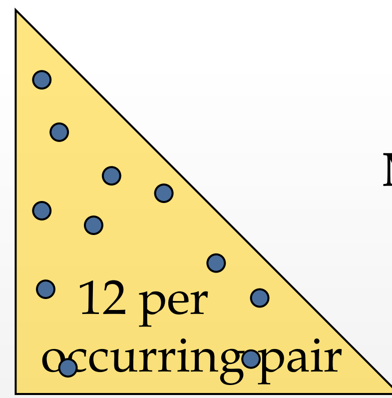
Details of Main-Memory Counting

- There are two basic approaches:
  1. Count all item pairs, using a triangular matrix.
  2. Keep a table of triples [i, j, c] = the count of the pair of items {i,j } is c.

- (1) requires only (say) 4 bytes/pair; (2) requires 12 bytes, but only for those pairs with >0 counts.



Method 1: Triangular Matrix



Method 2: Triples

The Triangular Matrix Approach

- Number items 1,2,…
- Keep pairs in the order *{1,2}, {1,3},…, {1,n }, {2,3}, {2,4},…,{2,n }, {3,4},…, {3,n },…{n -1,n }.*
- Find pair $\{i, j\}$ at the position

$$(i-1)(n-i/2) + j - i.$$

- Total number of pairs $n(n-1)/2$; total bytes about $2n^2$.

Source: Slide Adapted from Shivnath Babu, Duke CPS196.3, Lecture Notes, see earlier reference

The Triples Approach

- You need a hash table, with i and j as the key, to locate (i, j, c) triples efficiently.
  - Typically, the cost of the hash structure can be neglected.

- Total bytes used is about $12p$, where p is the number of pairs that actually occur.
  - Beats triangular matrix if at most 1/3 of possible pairs actually occur.

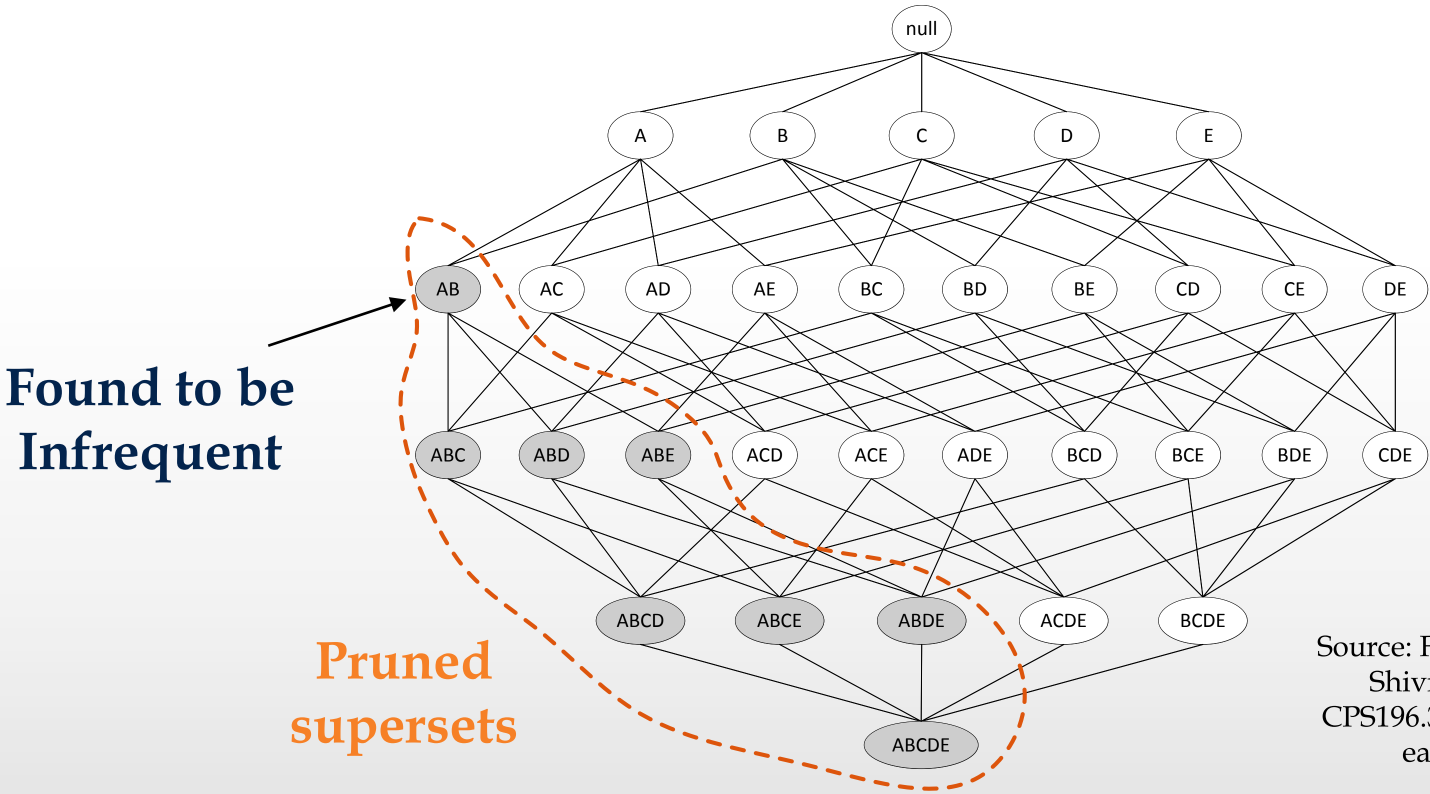
Source: Slide Adapted from Shivnath Babu, Duke CPS196.3, Lecture Notes, see earlier reference

Insights/Limitations from the Previous Two Approaches

Think/Pair/Share – Activity ☺

The A-Priori Algorithm

- Key idea: ***monotonicity*** - if a set of items appears at least s times, so does every subset.
  - **Contrapositive for pairs:** if item i does not appear in s baskets, then no pair including i can appear in s baskets.



Source: Figure Adapted from Shivnath Babu, Duke CPS196.3, Lecture Notes, see earlier reference

Illustrating the A-priori Principle*

- Consider the following market-basket data

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 from Shivnath Babu, Duke CPS196.3, Lecture Notes, see earlier reference

Illustrating the A-priori 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,

$6 + 6 + 1 = 13$

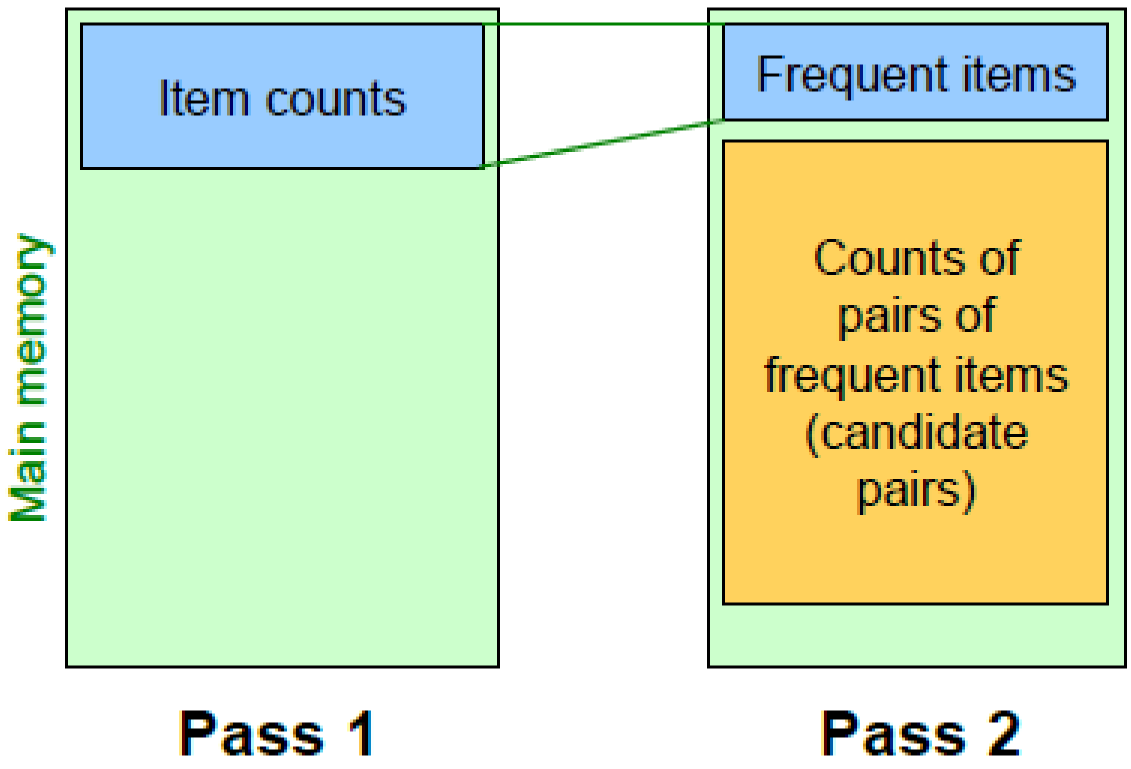
Details of the A-Priori Algorithm

- **Pass 1:** Read baskets and count in main memory the occurrences of each item.
  - Requires only memory proportional to #items.

- Pass 2: Read baskets again and count in main memory only those pairs both of which were found in Pass 1 to be frequent.
  - Requires memory proportional to square of frequent items only.
  - Plus a list of the frequent items (so you know what must be counted)

Source: Slide Adapted Jure Leskovic, Stanford CS246, Lecture Notes, see http://cs246.stanford.edu

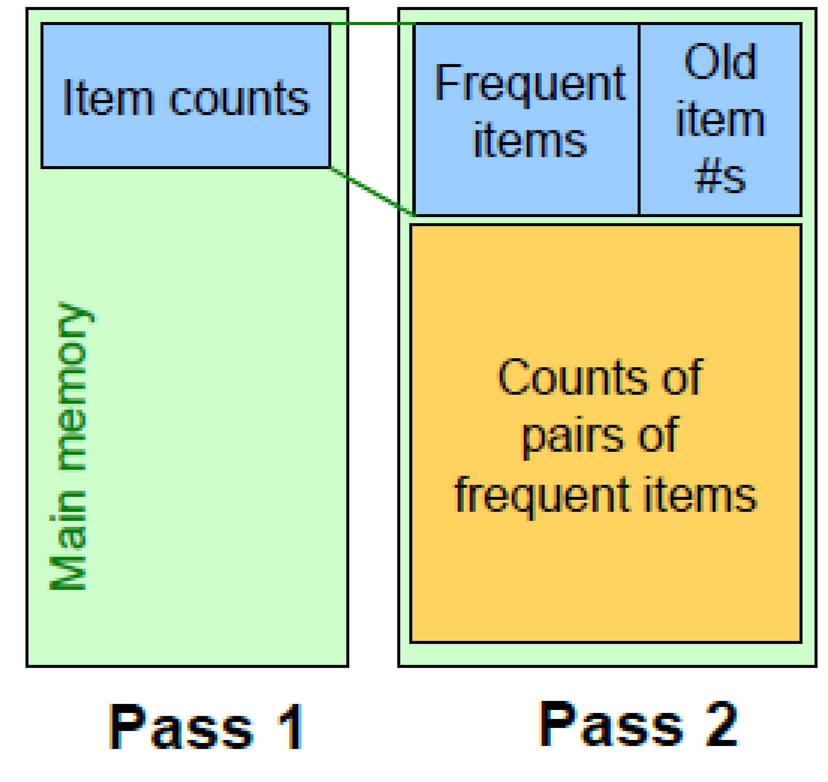
Main Memory: Picture of the A-Priori



Source: Slide Adapted Jure Leskovic, Stanford CS246, Lecture Notes, see http://cs246.stanford.edu

Detail for A-Priori

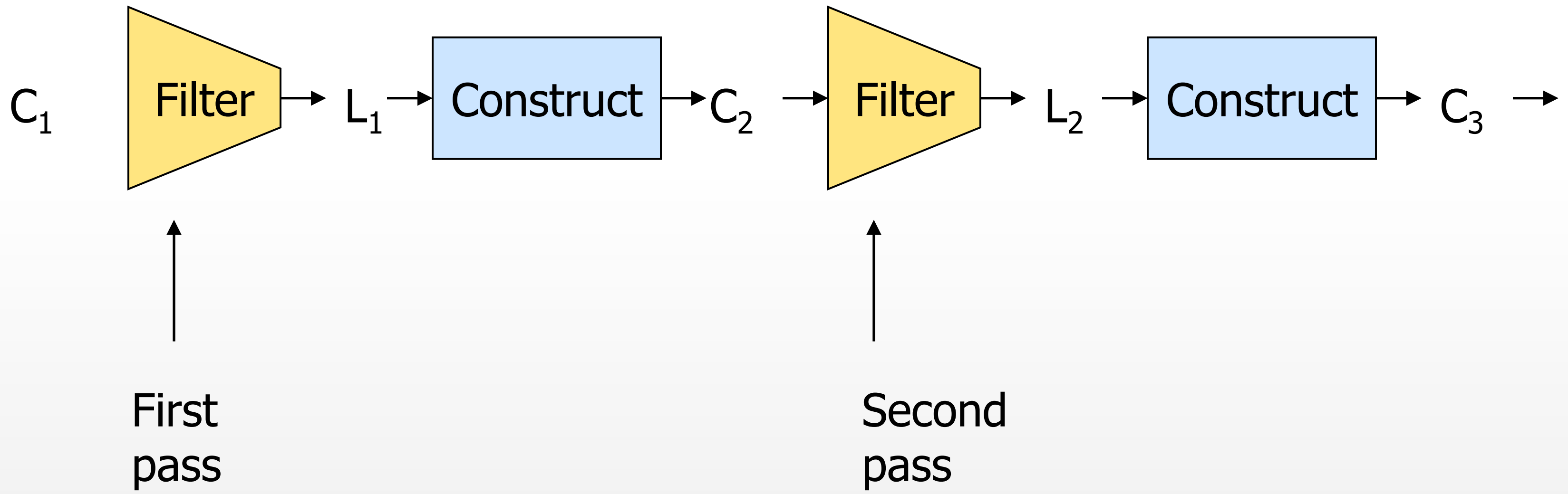
- You can use the triangular matrix method with n = number of frequent items.
  - Saves space compared with storing triples.
- **Trick:** number frequent items 1,2,… and keep a table relating new numbers to original item numbers.



Source: Slide Adapted Jure Leskovic, Stanford CS246, Lecture Notes, see http://cs246.stanford.edu

Frequent Triples, Etc.

- For each k, we construct two sets of k –tuples:
  - C_k = **candidate** k – tuples = those that might be frequent sets (support $\geq s$) based on information from the pass for k –1.
  - L_k = the set of truly frequent k –tuples.



Source: Slide Adapted Jure Leskovic, Stanford CS246, Lecture Notes, see http://cs246.stanford.edu

Example

- **Hypothetical steps of the A-Priori algorithm**
- C_1 = { {b} {c} {j} {m} {n} {p} }
- Count the support of itemsets in C_1
- Prune non-frequent: L_1 = { b, c, j, m }
- Generate C_2 = { {b,c} {b,j} {b,m} {c,j} {c,m} {j,m} }
- Count the support of itemsets in C2
- Prune non-frequent: L_2 = { {b,m} {b,c} {c,m} {c,j} }
- Generate C_3 = { {b,c,m} {b,c,j} {b,m,j} {c,m,j} }
- Count the support of itemsets in C_3
- Prune non-frequent: L_3 = { {b,c,m} }