Frequent Itemsets 1

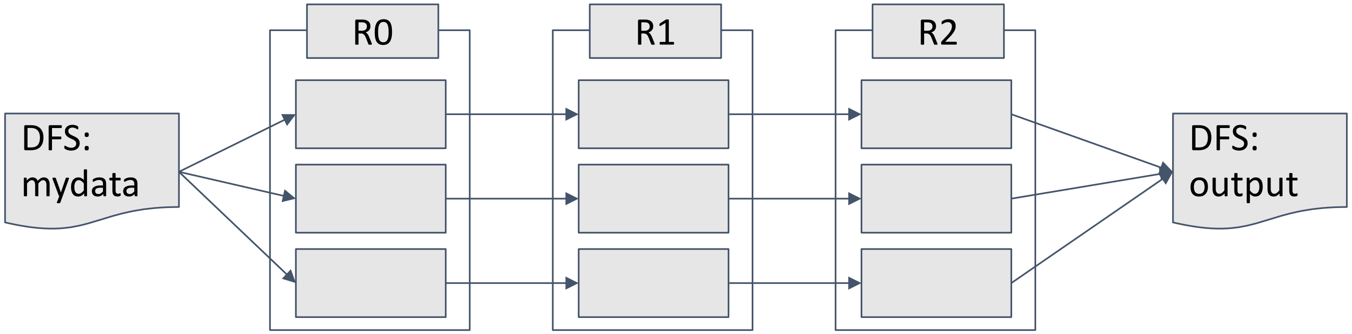
EE412: Foundation of Big Data Analytics

Announcements

- Homeworks
  - HW0 will be posted today (deadline: 09/21)
  - HW1 will be posted next Thursday (deadline: 10/05)
- Classum
  - I saw the first question and great answers
  - Thank you!

Recap: Spark

- Spark extends MapReduce with a DAG of functions
  - **Data abstraction:** Resilient distributed datasets (RDDs)
  - **Operations:** Map, FlatMap, Filter, Reduce, Join, etc.
  - **Two improvements:** Lazy evaluation, and the lineage of RDDs



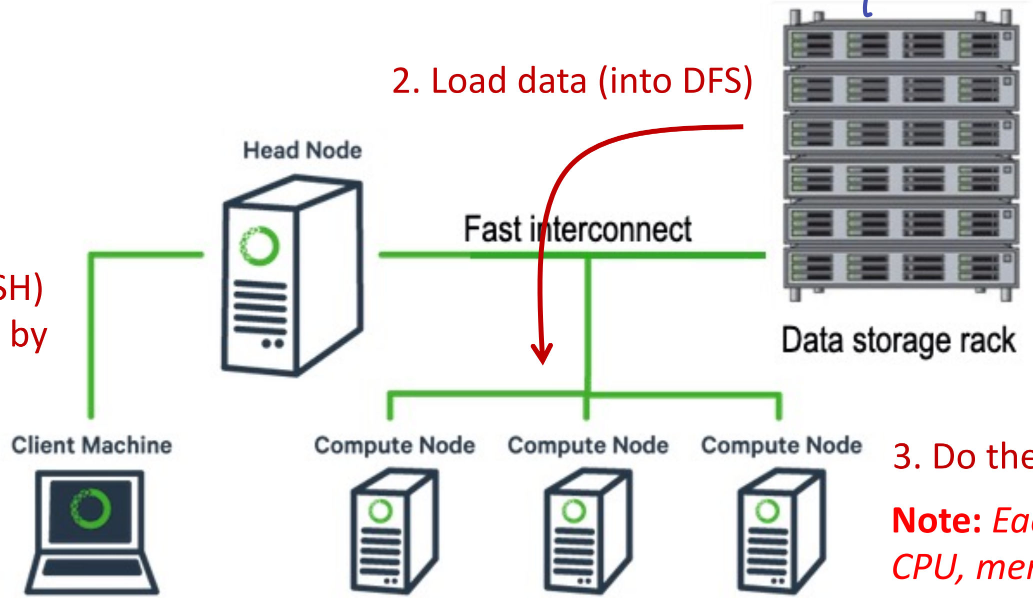
Recap: Computing Cluster

- **Q:** How exactly does a distributed cluster (Hadoop or Spark) work?

specialized for storage

2. Load data (into DFS)

1. Submit a job (SSH)
4. Receive a result by calling an Action



Compute node:
Actual computer
Runs workers
(= chunk server)
(= compute server)

Worker:
Virtual process
Does jobs (or tasks)

3. Do the job

Note: *Each node has its own CPU, memory, and disk*

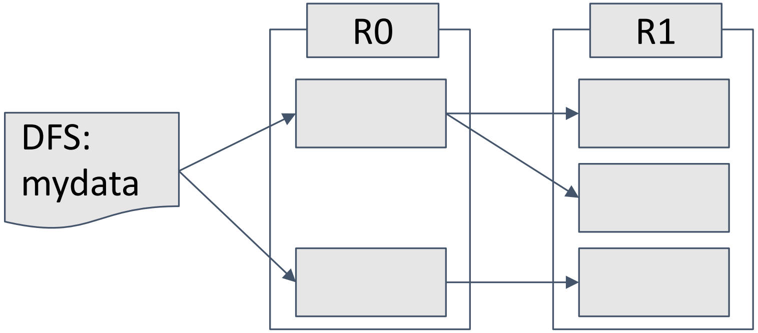
Source: Corredor and Rodriguez (2021)

node 1:N worker 1:N jobs

Recap: Details on FlatMap

- **Q1:** Can FlatMap create multiple partitions from one?
  - **A:** Yes, it is a one-to-many function
- **Q2:** Then, does it have **wide dependencies**?
  - **A:** No, each partition in R1 still comes from only one partition

narrow dependencies



Outline

1. **Frequent Itemsets**
2. Association Rules
3. Finding Frequent Pairs
4. A-Priori Algorithm

Association Rule Discovery

- **Market-basket model** for supermarket shelf management:
  - **Goal:** Identify items that are bought together by many customers
  - **Approach:** Process sales data to find dependencies among items
  - **A classic rule:**
    - If someone buys diaper and milk, then he/she is likely to buy beer
    - Don't be surprised if you find six-packs next to diapers!
- **Association rules:** People who bought $\{x, y\}$ tend to buy $\{z, v\}$

The Market-Basket Model

- Many-to-many mapping between items and baskets
  - A large set of **items** (e.g., things sold in a supermarket)
  - A large set of **baskets** (e.g., things a customer buys at one time)
    - Each basket is a small subset of items

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

Source: Stanford CS246 (2022)

Frequent itemsets

- **Goal:** Find sets of items that appear together frequently in baskets
- **Support** for itemset I: # of baskets containing all elements in I
  - Given a support threshold s
  - Itemset I is a **frequent itemset** if it appears at least in s baskets

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

Source: Stanford CS246 (2022)

Support of {Beer, Bread} is 2

Example

- Items = {milk, coke, pepsi, beer, juice}
- Support threshold = 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}

- Frequent itemsets:

{m}, {c}, {b}, {j}, {m, b}, {b, c}, {c, j}

Example

- Items = {milk, coke, pepsi, beer, juice}
- Support threshold = 3 baskets

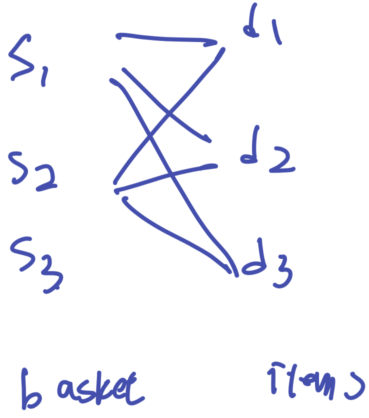
B1 = {m, c, b}	B2 = {m, p, j}
B3 = {m, b}	B4 = {c, j}
B5 = {m, p, b}	B6 = {m, c, b, j}
B7 = {c, b, j}	B8 = {b, c}

- Frequent itemsets:

{m}, {c}, {b}, {j}, {m, b}, {b, c}, {c, j}

Market-Basket Model as Abstract

- Items and baskets are abstracts
  - **Supermarket:** items = products; baskets = sets of products
    - Bread and milk (not too interesting)
    - Hot dog and mustard (opportunity for clever marketing)
    - Diapers and beers (why?)
  - **Topic discovery:** items = words; baskets = documents
  - **Plagiarism:** items = documents; baskets = sentences
    - Notice items do not have to be "in" baskets
  - **Biomarkers:** items = drugs & side effects; baskets = patients



Outline

1. Frequent Itemsets
2. **Association Rules**
3. Finding Frequent Pairs
4. A-Priori Algorithm

Association Rules

- **Given** an itemset I and an item j
- **Association rule** $I \to j$ means that if a basket contains all of the items in I, then it is likely to contain j as well

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}

Association rule: {m, b} → c

Is this a good rule?

Confidence

- We want to find significant/interesting rules
- **Confidence** of $I \rightarrow j$ = Probability of j given I in a basket
  - $\text{conf}(I \rightarrow j) = P(j|I) = \text{P}(I \cup \{j\}) \,/\, \text{P}(I)$
  - $\text{conf}(I \rightarrow j) = \text{support}(I \cup \{j\}) \,/\, \text{support}(I)$

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}

Support({m, b, c}): 2
Support({m, b}): 4
Confidence: 2/4 = 0.5

Association rule: {m, b} → c

??

Interest

- Not all high-confidence rules are interesting
  - The rule X → milk may have high confidence if milk is purchased a lot
- **Interest** of $I \rightarrow j$ = $\mathrm{conf}(I \rightarrow j) - P(j) \; = P(j|I) - P(j)$
  - $P(j)$: Fraction of baskets containing j in the dataset

B1 = {m, c, b}	B2 = {m, p, j}	Support({m, b, c}): 2
B3 = {m, b}	B4 = {c, j}	Support({m, b}): 4
B5 = {m, p, b}	B6 = {m, c, b, j}	Confidence: 2/4 = 0.5
B7 = {c, b, j}	B8 = {b, c}	Interest : 0.5 - 5/8 = -1/8
		(negative value?)

Association rule: {m, b} → c

Mining Association Rules

- **Goal:** Find all rules with support $\geq s$ and confidence $\geq c$
  - **Note:** Support of a rule $I \rightarrow j$ is the support of $I \cup \{j\}$
- **Hard part:** Finding the frequent itemsets
  - Identifying association rules is easy if we have frequent itemsets

Mining Association Rules

- **Step 1:** Find all frequent itemsets I (we will explain this next)
- **Step 2:** Rule generation
  abcd
  ab
  c, d
  - For every subset A of I, generate a rule $A \to I \setminus A$
    - **Variant 1:** Single pass to compute the rule confidence
      - $\text{conf}(\{a, b\} \to \{c, d\}) = \text{support}(\{a, b, c, d\}) / \text{support}(\{a, b\})$
    - **Variant 2:**
      always strong
      - **Observation:** If $\{a, b, c\} \to \{d\}$ is below confidence, then so is $\{a, b\} \to \{c, d\}$
      - Can eliminate stronger rules by starting from weaker ones
  - Output the rules above the confidence threshold

s(ab) ≥ s(abc)

abc → d
∪
ab → cd
∪

$\boxed{a \to bcd}$

Mining Association Rules

- **Example**
  - Support threshold $s = 3$
  - Confidence threshold $c = 3/4$

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}

1. Frequent itemsets:
{b,m}, {b,c}, {c,m}, {c,j}, {m,c,b}

Mining Association Rules

- **Example**
  - Support threshold $s = 3$
  - Confidence threshold $c = 3/4$

B1 = {m, c, b}
B3 = {m, c, b, n}
B5 = {m, p, b}
B7 = {c, b, j}

B2 = {m, p, j}
B4 = {c, j}
B6 = {m, c, b, j}
B8 = {b, c}

1. Frequent itemsets:
{b,m}, {b,c}, {c,m}, {c,j}, {m,c,b}

2. Generated rules:
{b, c} → m: conf = 3/5 weaker
{b, m} → c: conf = 3/4
{m, c} → b: conf = 3/3
b → {c, m}: conf = 3/6 stronger
…

Mining Association Rules

- **Example**
  - Support threshold $s = 3$
  - Confidence threshold $c = 3/4$

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}

1. Frequent itemsets:
{b,m}, {b,c}, {c,m}, {c,j}, {m,c,b}

2. Generated rules:
~~{b, c} → m: conf = 3/5~~
{b, m} → c: conf = 3/4
{m, c} → b: conf = 3/3
~~b → {c, m}: conf = 3/6~~
…

Compacting the Output

- Giving a store manager a million association rules is not practical
  - *What can we do other than adjusting support threshold?*
- To reduce # of rules, we can post-process frequent itemsets into
  - **Maximal frequent itemsets:**
    - No immediate superset is frequent
    - Gives more pruning
  - **Closed itemsets:**
    - No immediate superset has the same support (> 0)
    - Stores not only frequent information, but exact supports/counts

Example: Maximal / Closed Itemsets

- **Example** (s=3)

comparison w/ threshold

comparison w/ support

	Support	Maximal	Closed	
A	4	No	No	
B	5	No	Yes	BC is freq.
C	3	No	No	support(BC) = 3
AB	4	Yes	Yes	ABC is not freq.
AC	2	No	No	
BC	3	Yes	Yes	support(ABC) < 3
ABC	2	No	Yes	

Pop Quiz

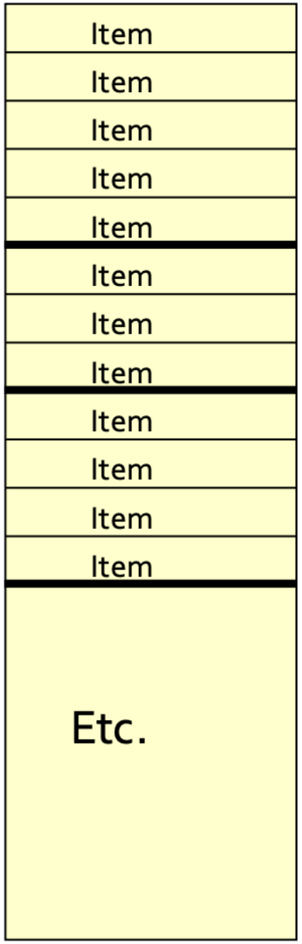
- Suppose we have the following baskets:
  - B1 = {apple, butter, milk}
  - B2 = {apple, butter, banana, walnuts, milk}
  - B3 = {butter, milk, banana}
  - B4 = {apple, milk}
- **Q1.** Compute the support for itemset {apple, butter} 2
- **Q2.** Compute the confidence for the rule {apple} -> {butter} $\frac{2}{3}$
- **Q3.** Compute the interest of {apple} -> {butter} $\frac{2}{3} - \frac{3}{4} = \frac{8-9}{12} = -\frac{1}{6}$
- **Q4.** Is {apple} -> {butter} maximal and/or closed?

Outline

1. Frequent Itemsets
2. Association Rules
3. **<u>Finding Frequent Pairs</u>**
4. A-Priori Algorithm

Representation of Market-Basket Data

- Let's go back to finding frequent itemsets
- Market-basket data is stored on disk basket-by-basket
  - Either in a DFS (baskets are objects in files) or as a file
- The data can be too large to fit in main memory
  - Baskets are small but we have many baskets and items



Source: Stanford CS246 (2022)

Computational Cost

- The true cost of mining disk-resident data is usually # of disk I/Os
- In practice, association-rule algorithms read the data in passes
- We measure the cost by **# of passes** an algorithm makes over data

Main-Memory Bottleneck

- For many frequent-itemset algorithms, **main memory** is the issue
  - As we read baskets, we need to count something, e.g., item occurrences
  - The number of different things we can count is limited by main memory
  - Swapping counts in/out is a disaster
- **Minor technique:** We first represent items by consecutive integers from 1 to n where n is the number of distinct items
  - Can construct a hash table that translates items to integers

Finding Frequent Pairs

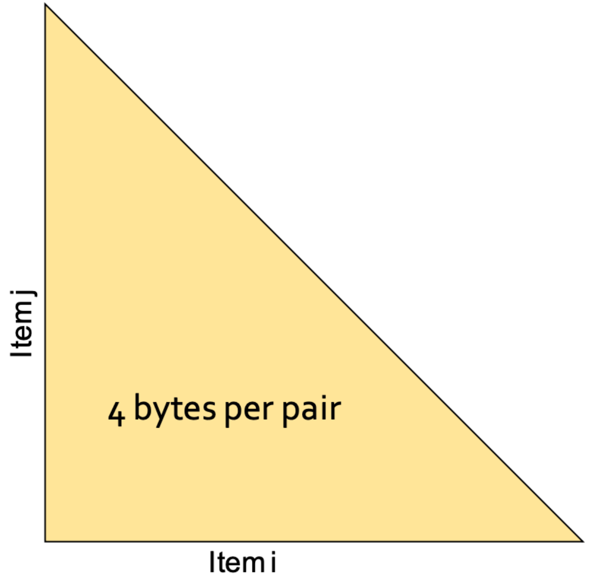
- The hardest problem is finding frequent **pairs of items** $\{i, j\}$
  - **Why?** Frequent pairs are common, frequent triples are rare
  - Probability of being frequent drops exponentially with size
- Let's first concentrate on pairs, then extend to larger sets

Counting Pairs in Memory

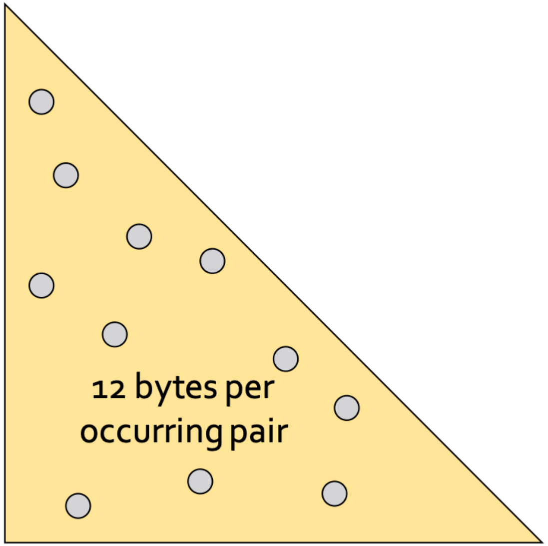
no

- **Goal:** Count the number of occurrences of each pair of items (i, j)
- **Triangular-matrix method** dense most pars occurs in data
  - Use a large matrix (i.e., an array) to store all counts
- **Triples method** sparse only few pair matches
  - Keep a hash table of triples (i, j, c), storing count c for each pair (i, j)
  - If integers are 4 bytes, we need about 12 bytes for each pair with $c > 0$
  - Plus some additional overhead for the hash table

Comparison of Two Approaches



Triangular Matrix



Triples (item i, item j, count)

Source: Stanford CS246 (2022)

Comparison of Two Approaches

- **Triangular-matrix method**
  - Count pair of items (i, j) only if $i < j$
  - Pairs are stored in lexicographic order:
    - {1,2}, {1,3}, …, {1,n}, {2,3}, {2,4}, …,{2,n}, … {n-1,n}
  - Total bytes = $4 \times n(n-1)/2$
- **Triples Method**
  - Uses 12 bytes per pair only if count > 0
  - Total bytes = $\boxed{12 \times p \times n}$?.?

크기가 3배 차이
P=1/3

- The triples method is better if $p < 1/3$

Naïve Algorithm

- Count in main memory the occurrences of each pair
  - From each basket b of n_b items, generate its $n_b(n_b - 1)/2$ pairs
- Fails if (# of items)^2 exceeds main memory easily happen
  - **Remember:** # of items can be 100K (Wal-Mart) or 10B (Web pages)
    - If we have 1M items, 2 terabytes of memory is needed
    - What if we have 1B items?
- *How can we do better when we have too many items?*

Outline

1. Frequent Itemsets
2. Association Rules
3. Finding Frequent Pairs
4. **<u>A-Priori Algorithm</u>**

A-Priori Algorithm

- **A-Priori** is a two-pass approach which limits memory usage
- **Key idea: Monotonicity** (inequality 이용)
  - If a set of items I appears at least s times, so does every subset J of I
- **Contrapositive for pairs:** (대우 명제)
  - If item i does not appear in s baskets, then no pair including i can

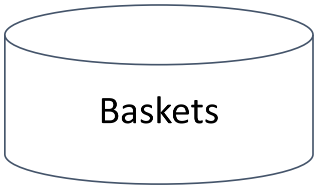
A-Priori Algorithm

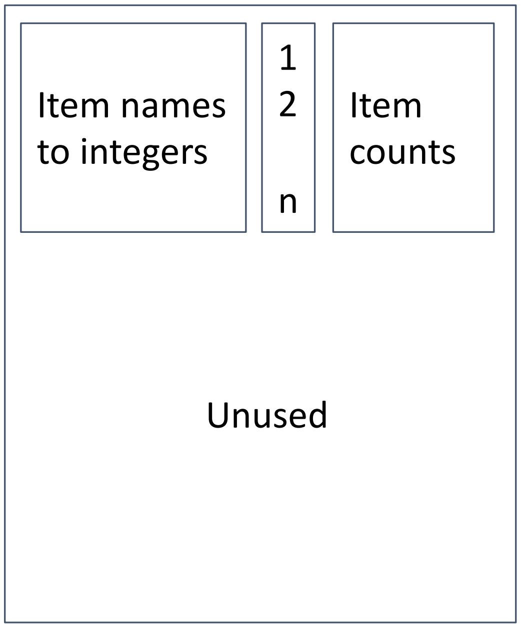
- **Pass 1:** Read baskets and count # of occurrences of each item
  - Requires only memory proportional to # of items
- **Mark** the items that appear $\geq s$ times as frequent items
- **Pass 2:** Read baskets again keeping track of only the freq. items
  - Count only those pairs where both elements are frequent (from Pass 1)
  - Requires memory proportional to square of # of frequent items
  - Plus a list of the frequent items (so you know what must be counted)

여기가 fail?. threshold 가 너무 작다.

A-Priori Algorithm (Pass 1)

- Table 1: Translate item names to integers $1, \ldots, n$
- Table 2: Initialize an array of counts for n items to 0

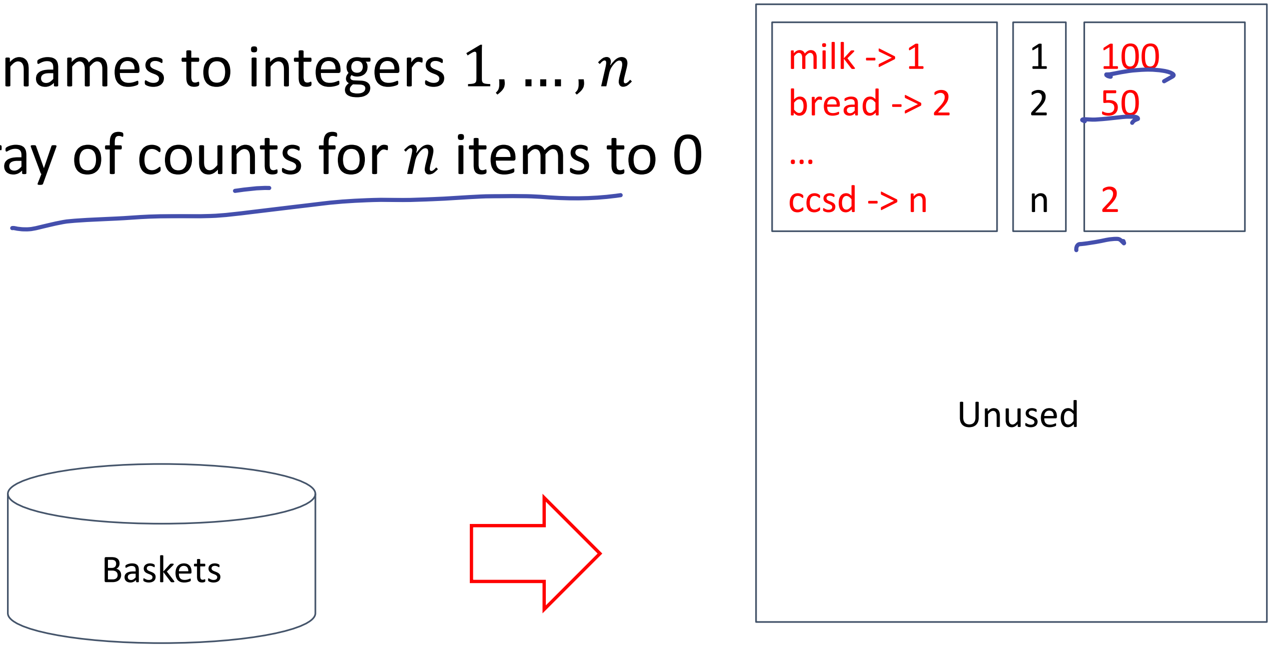




<main memory>

A-Priori Algorithm (Pass 1)

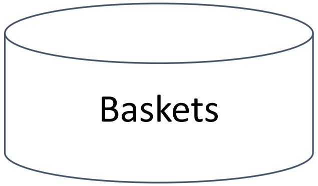
- Table 1: Translate item names to integers $1, \dots, n$
- Table 2: Initialize an array of counts for n items to 0

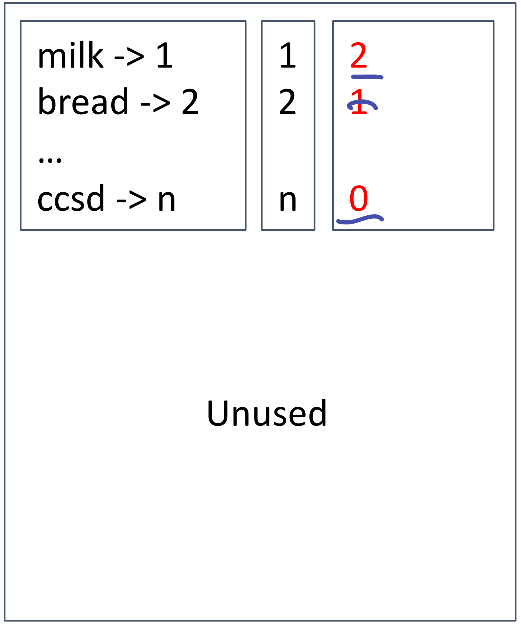


<main memory>

A-Priori Algorithm (between Pass 1 & 2)

- Find frequent items with support $\geq s$
- Create new numbering $1, \ldots, m$ for the freq. items
- Create a table where the i-th entry is 0 if item i is not frequent or a unique integer in $[1, m]$ otherwise

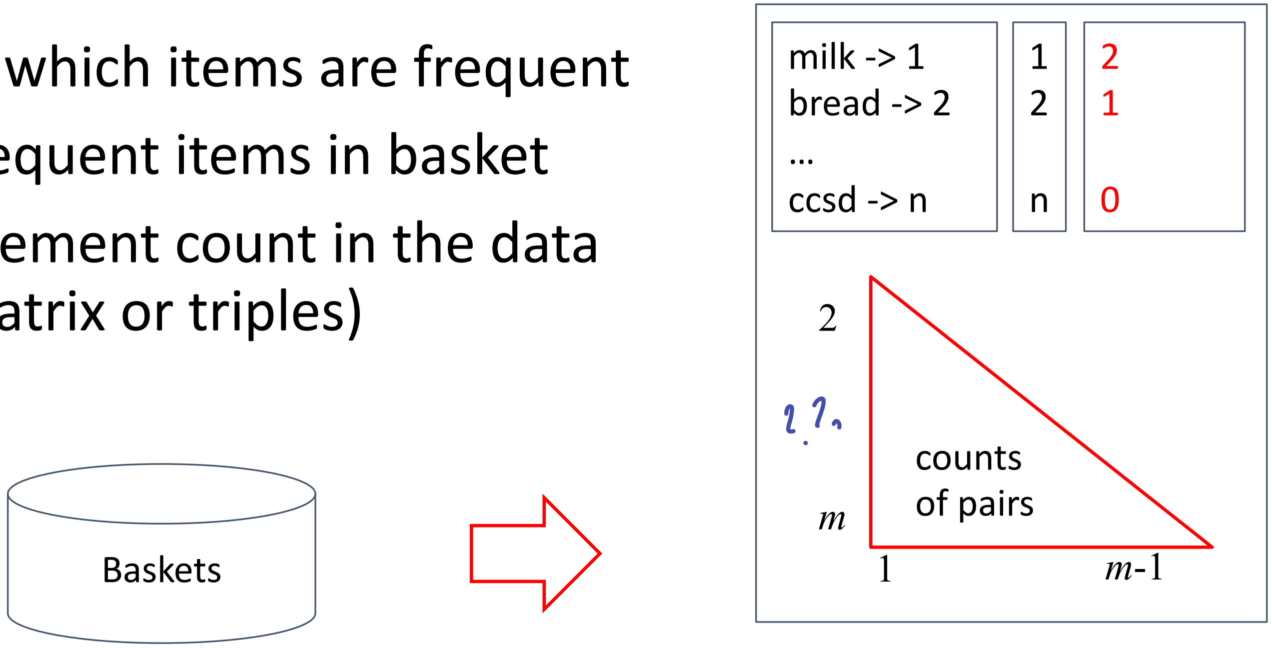




<main memory>

A-Priori Algorithm (Pass 2)

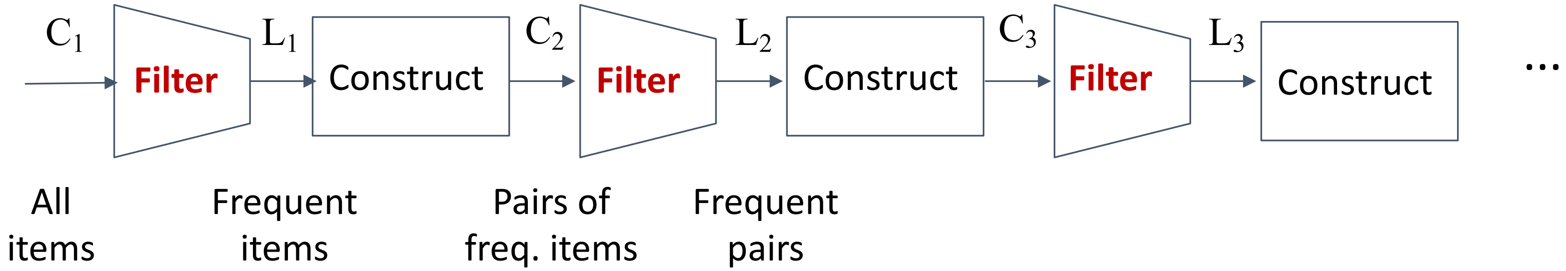
- For each basket, check which items are frequent
- Generate all pairs of frequent items in basket
- For each such pair, increment count in the data structure (triangular matrix or triples)



<main memory>

Frequent Triples

- Same technique is used for finding larger frequent itemsets
- When we move from size $k-1$ to k, we use two sets of itemsets:
  - C_k: Set of candidate k-tuples that might be frequent sets
  - L_k: Set of truly frequent k-tuples
- If no frequent itemsets are found, we are done by monotonicity



Example

- C1 = { {1} {2} {3} {4} {5} {6} {7} {8} {9} {10} }
- Count the support of itemsets in C1
- Prune non-frequent itemsets: L1 = { {1}, {2}, {3}, {4}, {5} }

Example

- C1 = { {1} {2} {3} {4} {5} {6} {7} {8} {9} {10} }
- Count the support of itemsets in C1
- Prune non-frequent itemsets: L1 = { {1}, {2}, {3}, {4}, {5} }
- Generate C2 = { {1,2} {1,3} {1,4} {1,5} {2,3} {2,4} {2,5} {3,4} {3,5} {4,5} }
- Count the support of itemsets in C2
- Prune non-frequent itemsets: L2 = { {1,2} {2,3} {2,4} {3,4} {4,5} }

Example

support

- C1 = { {1} {2} {3} {4} {5} {6} {7} {8} {9} {10} }
- Count the support of itemsets in C1
- Prune non-frequent itemsets: L1 = { {1}, {2}, {3}, {4}, {5} }
- Generate C2 = { {1,2} {1,3} {1,4} {1,5} {2,3} {2,4} {2,5} {3,4} {3,5} {4,5} }
- Count the support of itemsets in C2
- Prune non-frequent itemsets: L2 = { {1,2} {2,3} {2,4} {3,4} {4,5} }
- Generate C3 = { {2,3,4} } {1,2,3}, {1,2,4}, {2,4,5}, {3,4,5}
- ...

Pop Quiz

- Apply the A-Priori algorithm with support threshold 3:

  B1 = {apple, butter, milk}

  B2 = {apple, butter, banana, walnuts, milk}

  B3 = {butter, milk}

  B4 = {apple, milk}

Summary

1. Frequent Itemsets
   - Market-basket model
2. Association Rules
   - Confidence, Interest, Maximal or closed itemsets
3. Finding Frequent Pairs
   - Triangular-matrix method, Triples method
4. A-Priori Algorithm
   - How the two-pass algorithm works