Frequent Itemset Mining

Stony Brook University

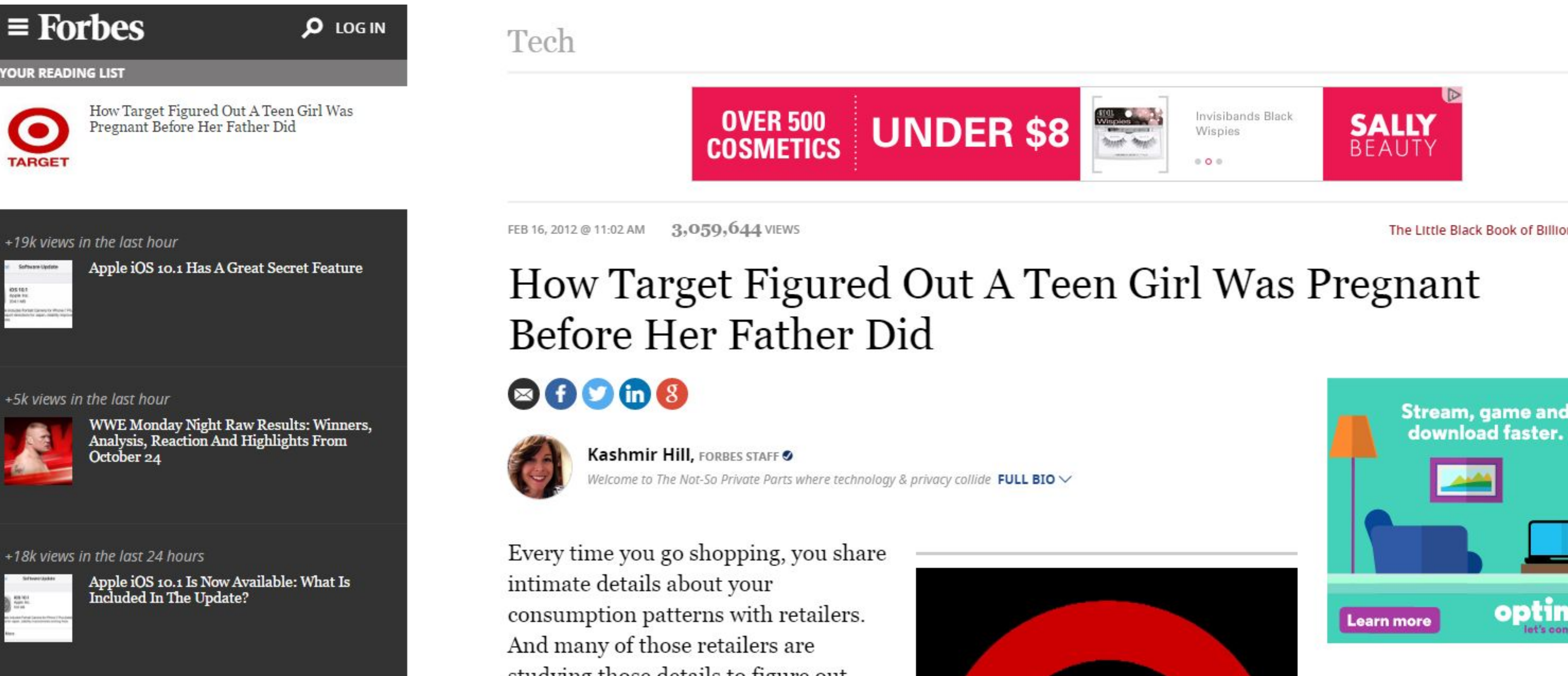
CSE545, Fall 2016

Frequent Itemset Mining *aka* Association Rules

Goal: Identify items that are often purchased together.

Frequent Itemset Mining *aka* Association Rules

Goal: Identify items that are often purchased together.



Frequent Itemset Mining *aka* Association Rules

Goal: Identify items that are often purchased together.

Classic Example:

If someone buys diapers and milk, then he/she is likely to buy beer

Don’t be surprised if you find six-packs next to diapers!

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

Market-Basket Model

Given:

- Set of potential *items*
- Instances of *baskets*

Each basket ($b \in$ *baskets*) is a subset of *items*
(i.e. the items bought in a single purchase)

Market-Basket Model

Given:

- Set of potential *items*
- Instances of *baskets*

Each basket ($b \in$ *baskets*) is a subset of *items*
(i.e. the items bought in a single purchase)

Find:

Frequent itemsets -- itemsets which appear together in at least *s* baskets
(*s* = *“support”)*

Association Rules -- if-then rules about the contents of baskets
(e.g. if basket contains 7-up and Snickers, then it likely to also contains Pop Secret)

Market-Basket Model

s(I) -- support, number of times appearing together.

Rule : $I \rightarrow j$ //given I items j is likely to appear

confidence -- How likely is j, given I:

$$c = \frac{s(I \cup \{j\})}{s(I)}$$

Association Rules -- if-then rules about the contents of baskets

(e.g. if basket contains 7-up and Snickers, then it likely to also contains Pop Secret)

Market-Basket Model

$s(I)$ -- support, number of times appearing together.

Rule : $I \rightarrow j$ //given I items j is likely to appear

confidence -- How likely is j, given I:

$$c = \frac{s(I \cup \{j\})}{s(I)}$$

Typical use: find all rules with at least a given *support* and a given *confidence*.

Association Rules -- if-then rules about the contents of baskets

(e.g. if basket contains 7-up and Snickers, then it likely to also contains Pop Secret)

Market-Basket Model

$s(I)$ -- support, number of times appearing together.

Rule : $I \rightarrow j$ //given I items j is likely to appear

confidence -- How likely is j, given I:

$$c = \frac{s(I \cup \{j\})}{s(I)}$$

Typical use: find all rules with at least a given *support* and a given *confidence*.

Why support?

Association Rules -- if-then rules about the contents of baskets

(e.g. if basket contains 7-up and Snickers, then it likely to also contains Pop Secret)

Market-Basket Model

$s(I)$ -- support, number of times appearing together.

Rule : $I \rightarrow j$ //given I items j is likely to appear

confidence -- How likely is j, given I:

$$c = \frac{s(I \cup \{j\})}{s(I)}$$

Typical use: find all rules with at least a given *support* and a given *confidence*.

Why support?

favors really common items --
can't recommend common
items "everywhere"

Association Rules -- if-then rules about the contents of baskets
(e.g. if basket contains 7-up and Snickers, then it likely to also contains Pop Secret)

Market-Basket Model

s(I) -- support, number of times appearing together.

Rule : $I \rightarrow j$ //given I items j is likely to appear

confidence -- How likely is j, given I:

$$c = \frac{s(I \cup \{j\})}{s(I)}$$

interest -- Difference between *c* and "expected *c*" :

$$int = c - \frac{s(\{j\})}{s(\text{all\_baskets})}$$

Association Rules -- if-then rules about the contents of baskets

(e.g. if basket contains 7-up and Snickers, then it likely to also contains Pop Secret)

Market-Basket Model

s(I) -- support, number of times appearing together.

Rule : $I \rightarrow j$ //given I items j is likely to appear

confidence -- How likely is j, given I:

$$c = \frac{s(I \cup \{j\})}{s(I)}$$

interest -- Difference between *c* and "expected *c*" :

$$int = c - \boxed{\frac{s(\{j\})}{s(\text{all\_baskets})}}$$

Association Rules -- if-then rules about the contents of baskets

(e.g. if basket contains 7-up and Snickers, then it likely to also contains Pop Secret)

Main-Memory Bottleneck

Imagine application: Process basket by basket, counting pairs, triples, etc...

Main-Memory Bottleneck

Imagine application: Process basket by basket, counting pairs, triples, etc...

- Counting itemsets in memory can run out of space quickly.

$$\binom{100,000}{2} = 5billion \text{ pairs}$$

- If storing in memory: just not enough space
- If storing on disk: too much swapping in and out with every increment

Main-Memory Bottleneck

Imagine application: Process basket by basket, counting pairs, triples, etc...

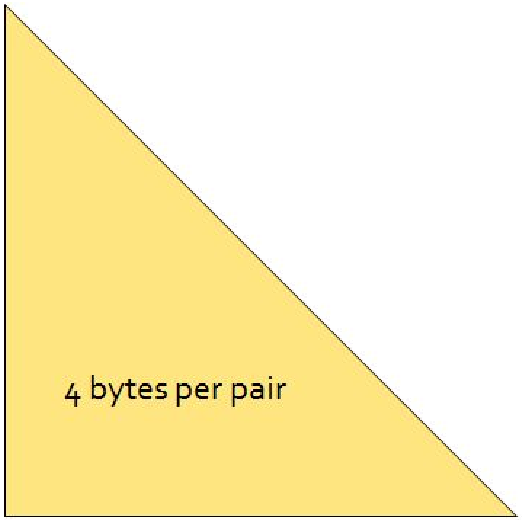
- Counting itemsets in memory can run out of space quickly.

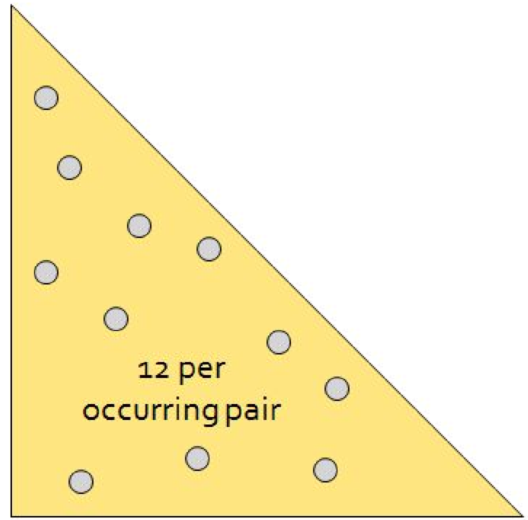
$$\binom{100,000}{2} = 5billion \text{ pairs}$$

- If storing in memory: just not enough space
- If storing on disk: too much swapping in and out with every increment

One partial solution: we can do a lot just counting pairs, since a triple can be evidenced by strong confidence of its 3 subset pairs.

2 Approaches to store pairs





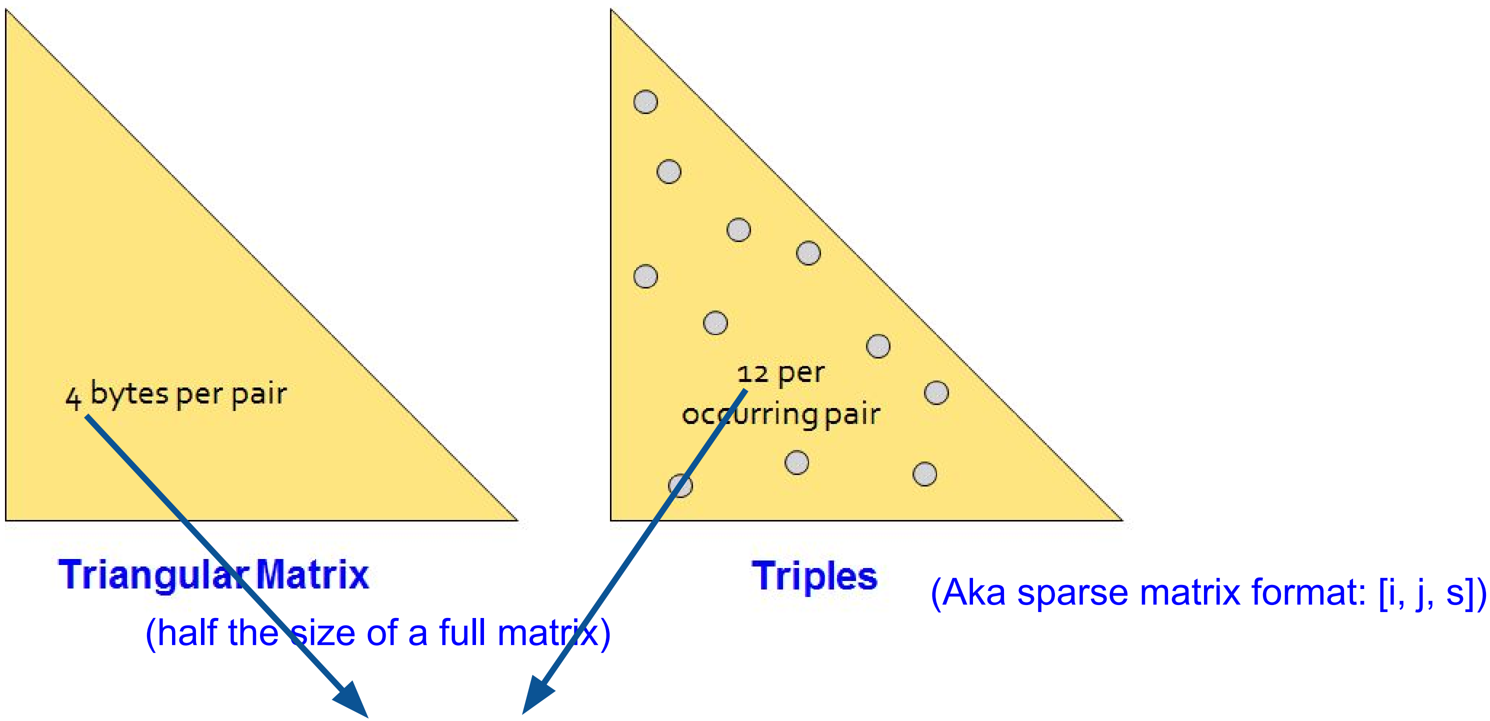
Triangular Matrix

(half the size of a full matrix)

Triples

(Aka sparse matrix format: [i, j, s])

2 Approaches to store pairs



Triples beats if we only have ⅓ of possible pairs

A’ Priori Algorithm

Can we use multiple passes and negate the need to store items in main memory?

Goal: Find frequent pairs.

A’ Priori Algorithm

Can we use multiple passes and negate the need to store items in main memory?

Goal: Find frequent pairs.

Key idea: *Monotonicity* -- If itemset I appears at least s times, then $J \subseteq I$ also appears at least s times.

Thus, if item i does not appear in s baskets, then no set including i can appear in s baskets. (using contrapositive of monotonicity)

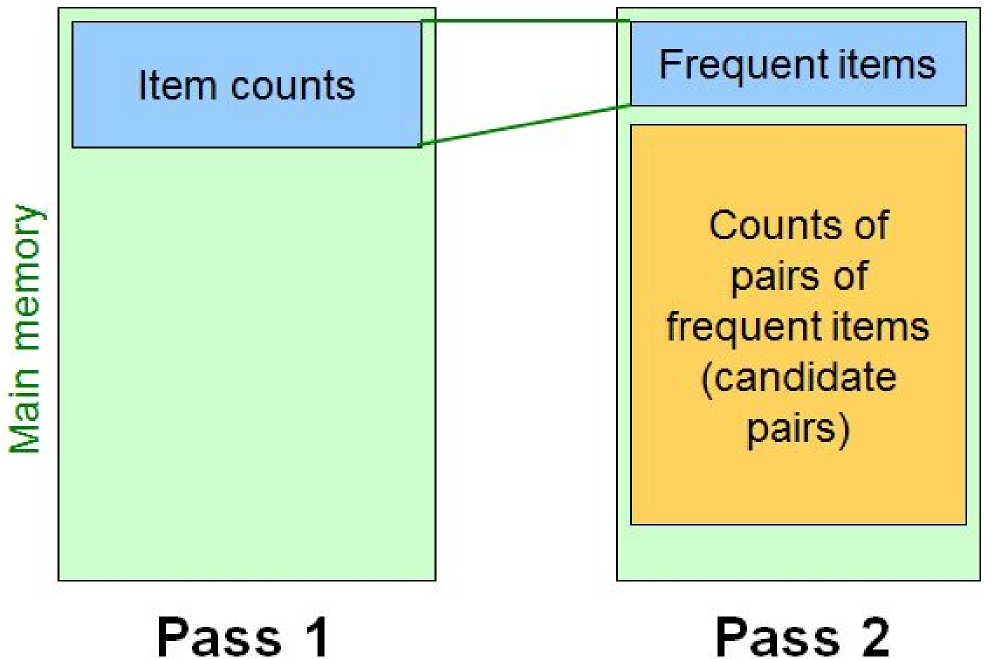
A’ Priori Algorithm

Can we use multiple passes and negate the need to store items in main memory?

Goal: Find frequent pairs.

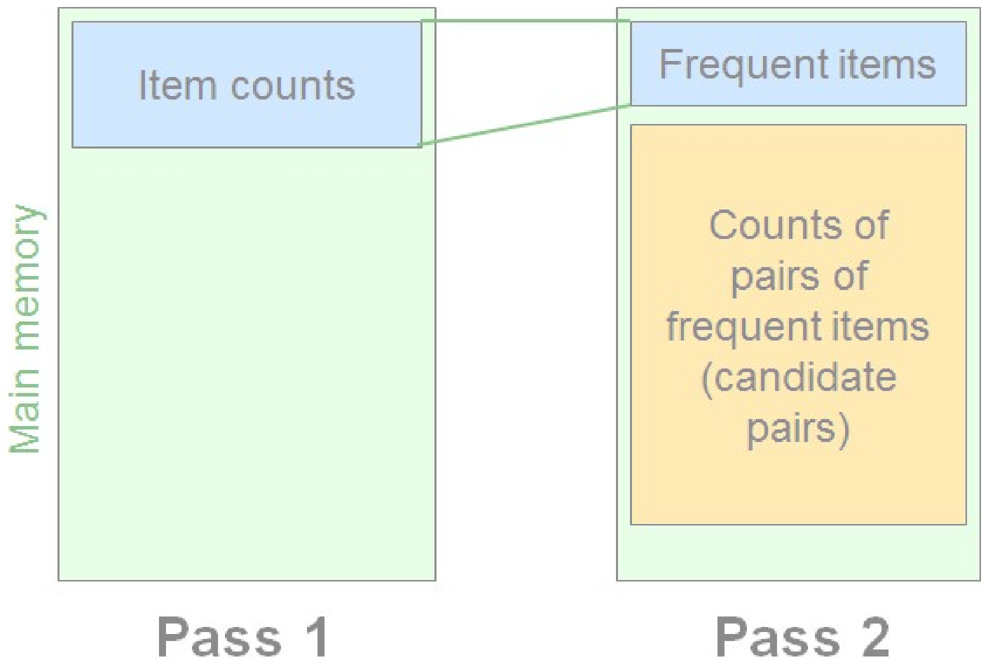
```
Pass 1: count basket occurrences of each item
    //frequent items -- appear at least s times

Pass 2: count pairs of frequent items
    //requires O(|frequent items|^2) + O(|frequent items|) memory
```

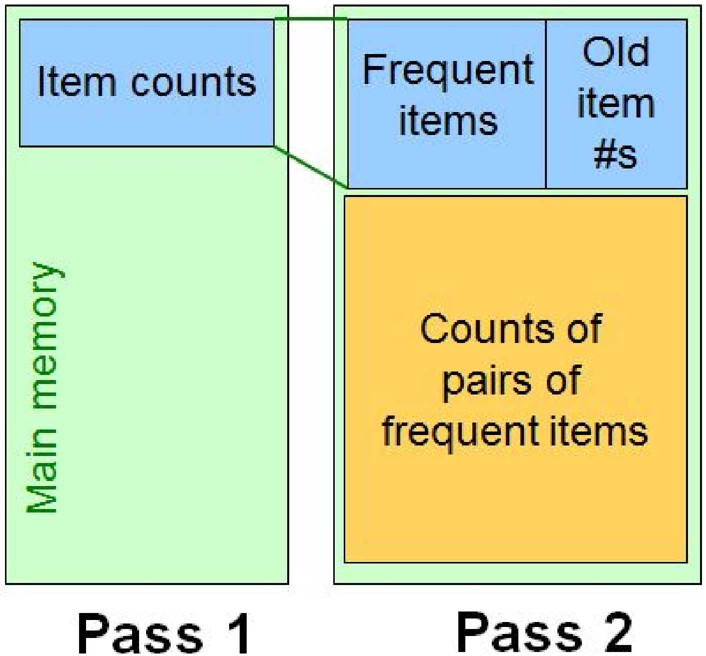
A’ Priori Algorithm



A’ Priori Algorithm

To use triangle matrix method, need to map to old numbers.





A’ Priori Algorithm: What about triples, etc...?

K_sets -- sets of size k

Pass 1: count basket occurrences of each item

    //frequent items -- appear at least s times

Pass 2: count pairs of *frequent items*

    //requires O(|*frequent items*|^2) + O(|*frequent items*|) memory

A' Priori Algorithm: What about triples, etc...?

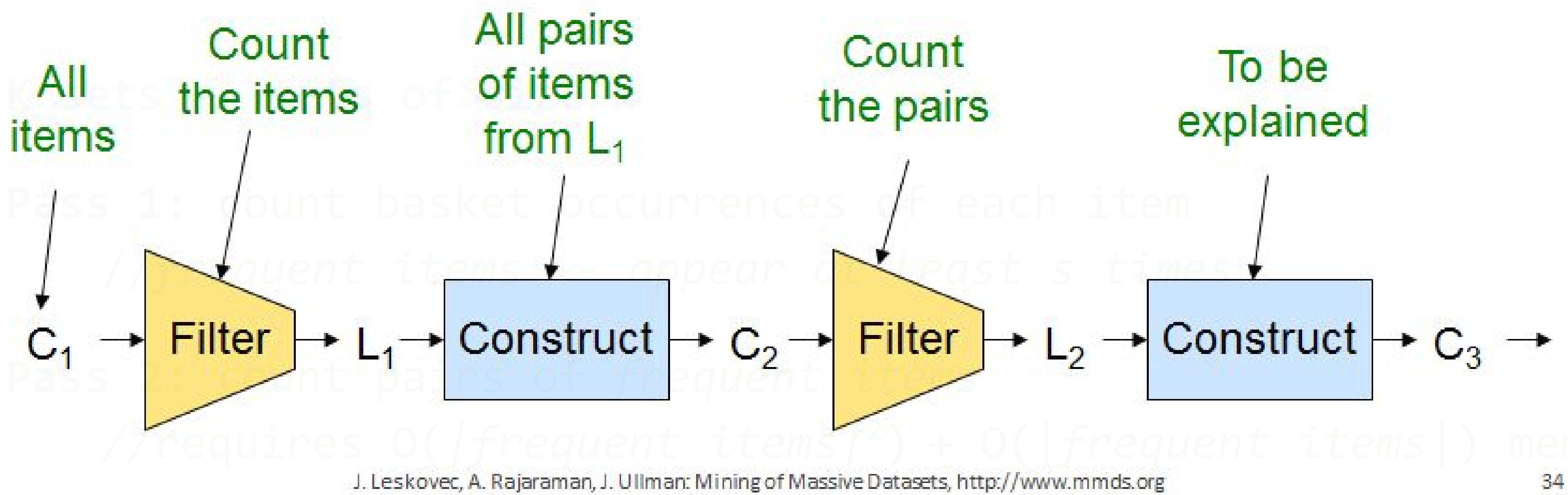
K_sets -- sets of size k

Pass 1: count basket occurrences of each item
    //frequent items -- appear at least s times

Pass 2: count pairs of *frequent items*
    //requires O(|*frequent items*|^2) + O(|*frequent items*|) memory

Pass 3+: count k_sets of *frequent (k-1)_sets -- C_k*
    //C_k *are* possible k_sets (meeting support threshold)

A' Priori Algorithm: What about triples, etc...?



Pass 3+: count k_sets of *frequent (k-1)_sets* -- C_k

//C_k are candidate k_sets

//L_k those meeting support threshold

A’ Priori Algorithm

- One pass for each *k*
- Space needed on kth pass is up to C choose k
  - In practice, memory often peaks at 2

Thus, often focus only on pairs.