The A-Priori Algorithm

Monotonicity of "Frequent"
Candidate Pairs
Extension to Larger Itemsets

A-Priori Algorithm

- A two-pass approach called *a-priori* limits the need for main memory.
- Key idea: *monotonicity*: if a set of items appears at least *s* times, so does every subset of *s*.
- Contrapositive for pairs: if item *i* does not appear in *s* baskets, then no pair including *i* can appear in *s* baskets.

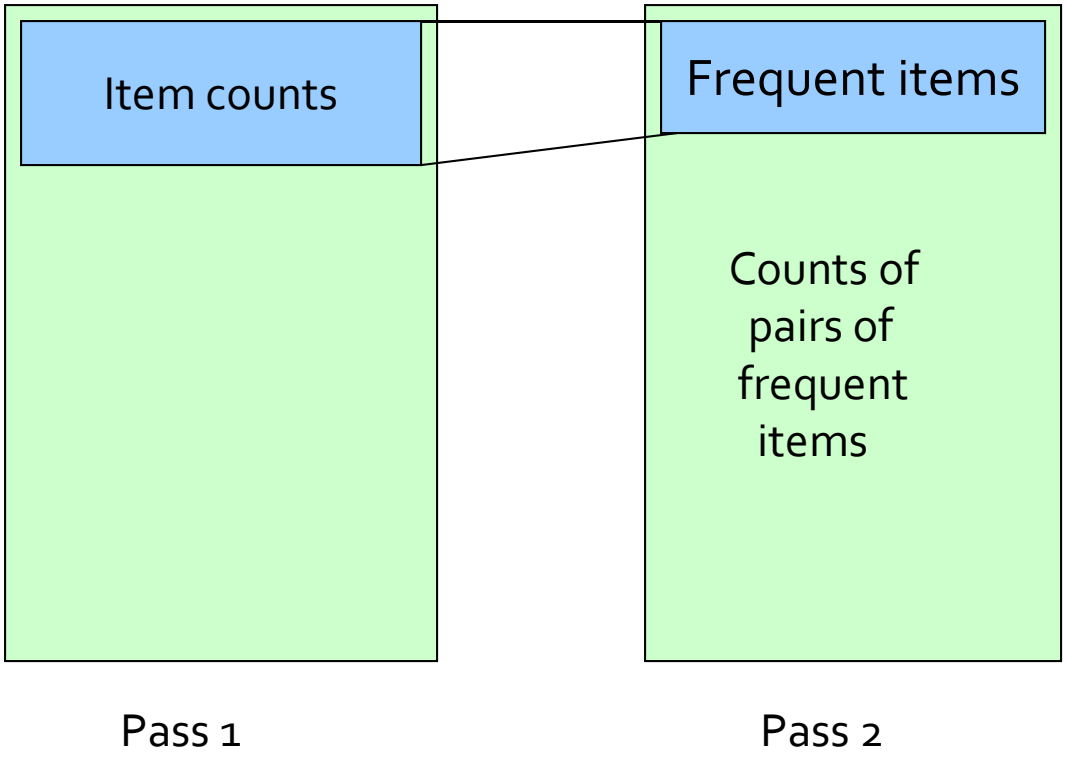
A-Priori Algorithm – (2)

- Pass 1: Read baskets and count in main memory the occurrences of each item.
  - Requires only memory proportional to #items.
- Items that appear at least *s* times are the *frequent items*.

A-Priori Algorithm – (3)

- 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 (for counts), plus a list of the frequent items (so you know what must be counted).

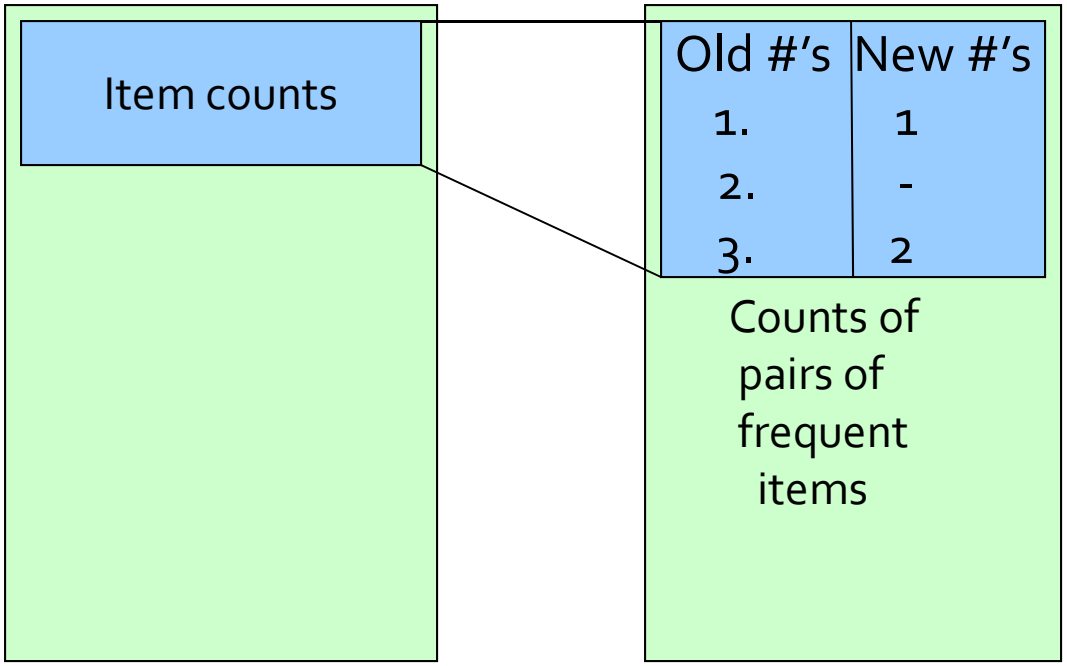
Picture of A-Priori



Detail for A-Priori

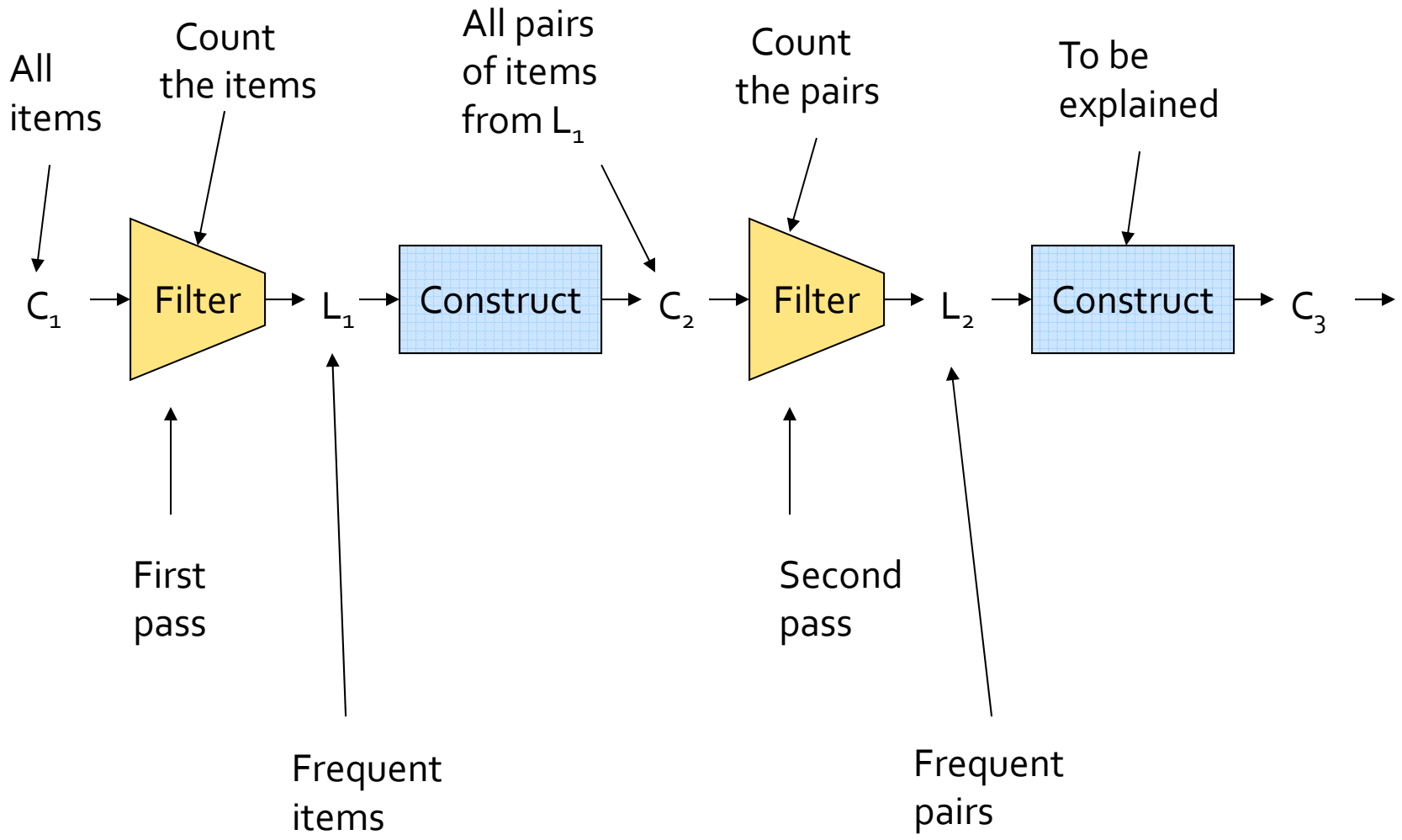
- You can use the triangular matrix method with *n* = number of frequent items.
  - May save space compared with storing triples.
- Trick: number frequent items 1,2,… and keep a table relating new numbers to original item numbers.

A-Priori Using Triangular Matrix



Frequent Triples, Etc.

- For each k, we construct two sets of *k-sets* (sets of size k):
  - C_k = *candidate* k-sets = 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-sets.


All items
Count the items
All pairs of items from L_1
Count the pairs
To be explained
C_1
Filter
L_1
Construct
C_2
Filter
L_2
Construct
C_3
First pass
Second pass
Frequent items
Frequent pairs

Passes Beyond Two

- C_1 = all items
- In general, L_k = members of C_k with support $\geq s$.
  - Requires one pass.
- C_{k+1} = (k+1)-sets, each k of which is in L_k .

Memory Requirements

- At the k^{th} pass, you need space to count each member of C_k.
- In realistic cases, because you need fairly high support, the number of candidates of each size drops, once you get beyond pairs.