Performance and Scalability: Apriori Implementation

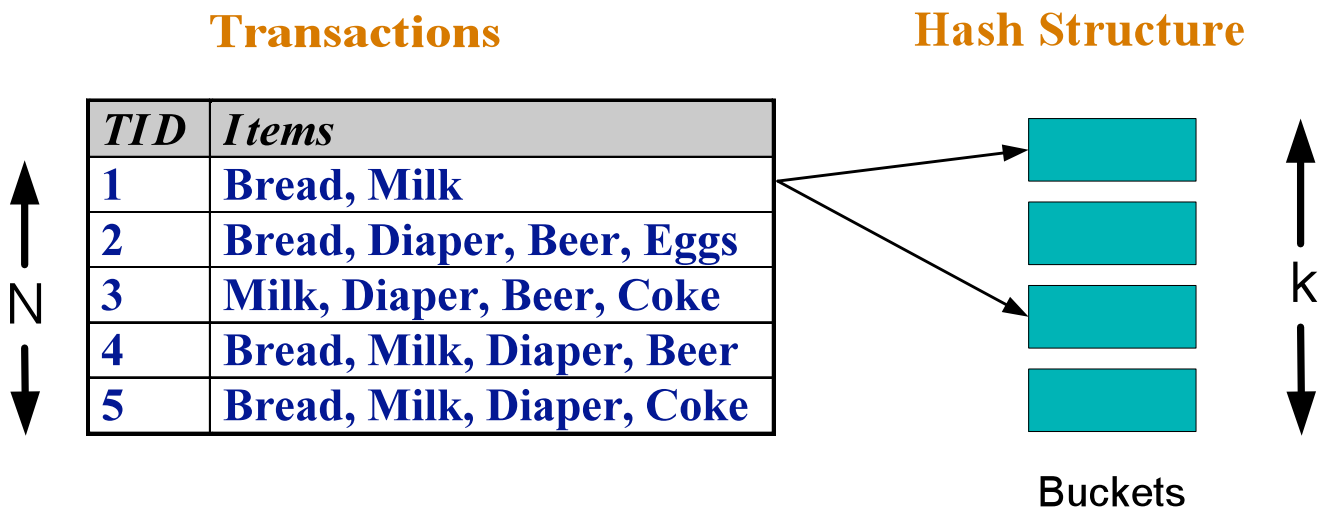
Apriori

```
1)  L_1 = {large 1-itemsets};
2)  for ( k = 2; L_{k-1} ≠ ∅; k++ ) do begin
3)     C_k = apriori-gen(L_{k-1}); // New candidates
4)     forall transactions t ∈ D do begin
5)        C_t = subset(C_k, t); // Candidates contained in t
6)        forall candidates c ∈ C_t do
7)           c.count++;
8)     end
9)     L_k = {c ∈ C_k | c.count ≥ minsup}
10) end
11) Answer = ⋃_k L_k;
```

R. Agrawal and R. Srikant.
Fast algorithms for mining association rules. VLDB, 487-499, 1994

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



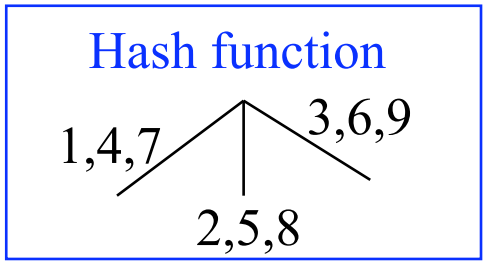
Generate Hash Tree

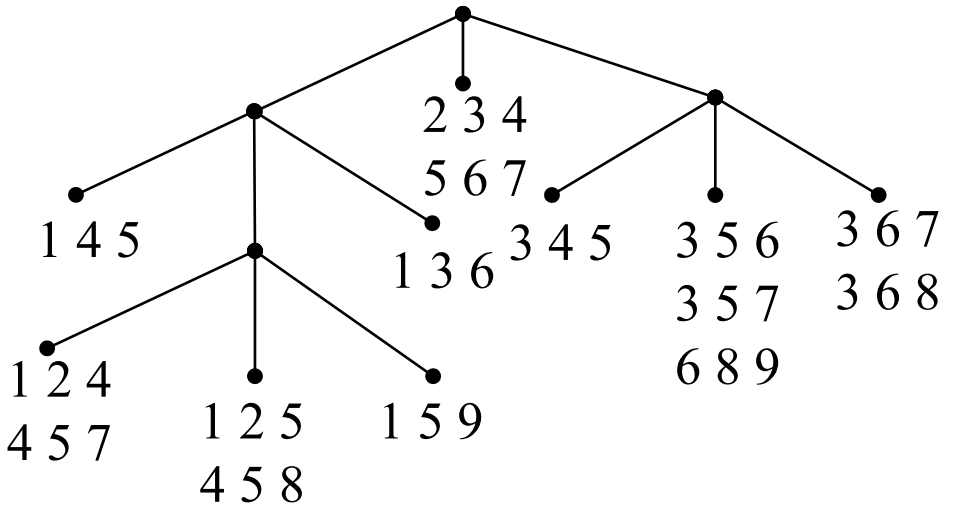
Suppose you have 15 candidate itemsets of length 3:

{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}

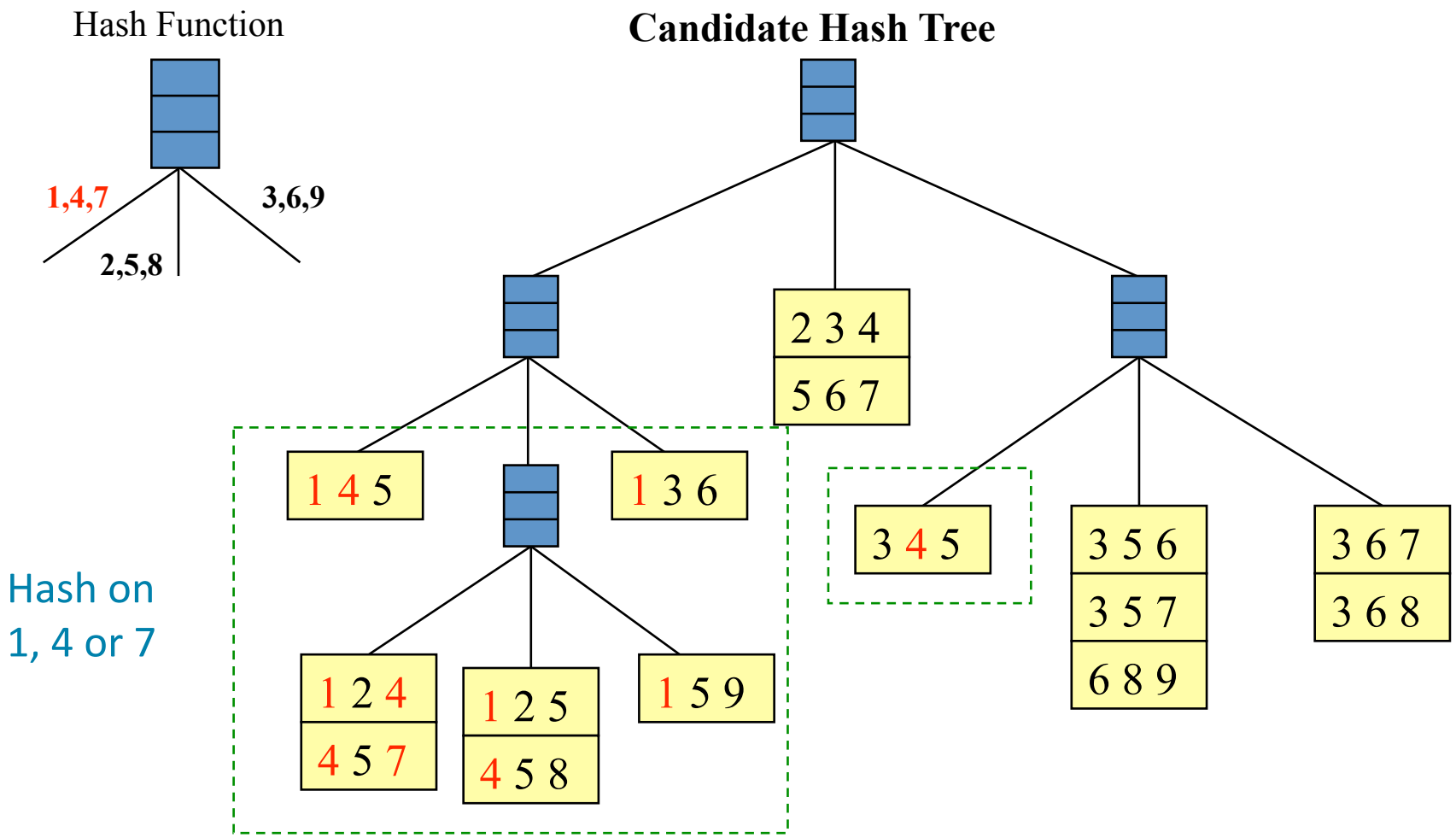
You need:

- **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)**

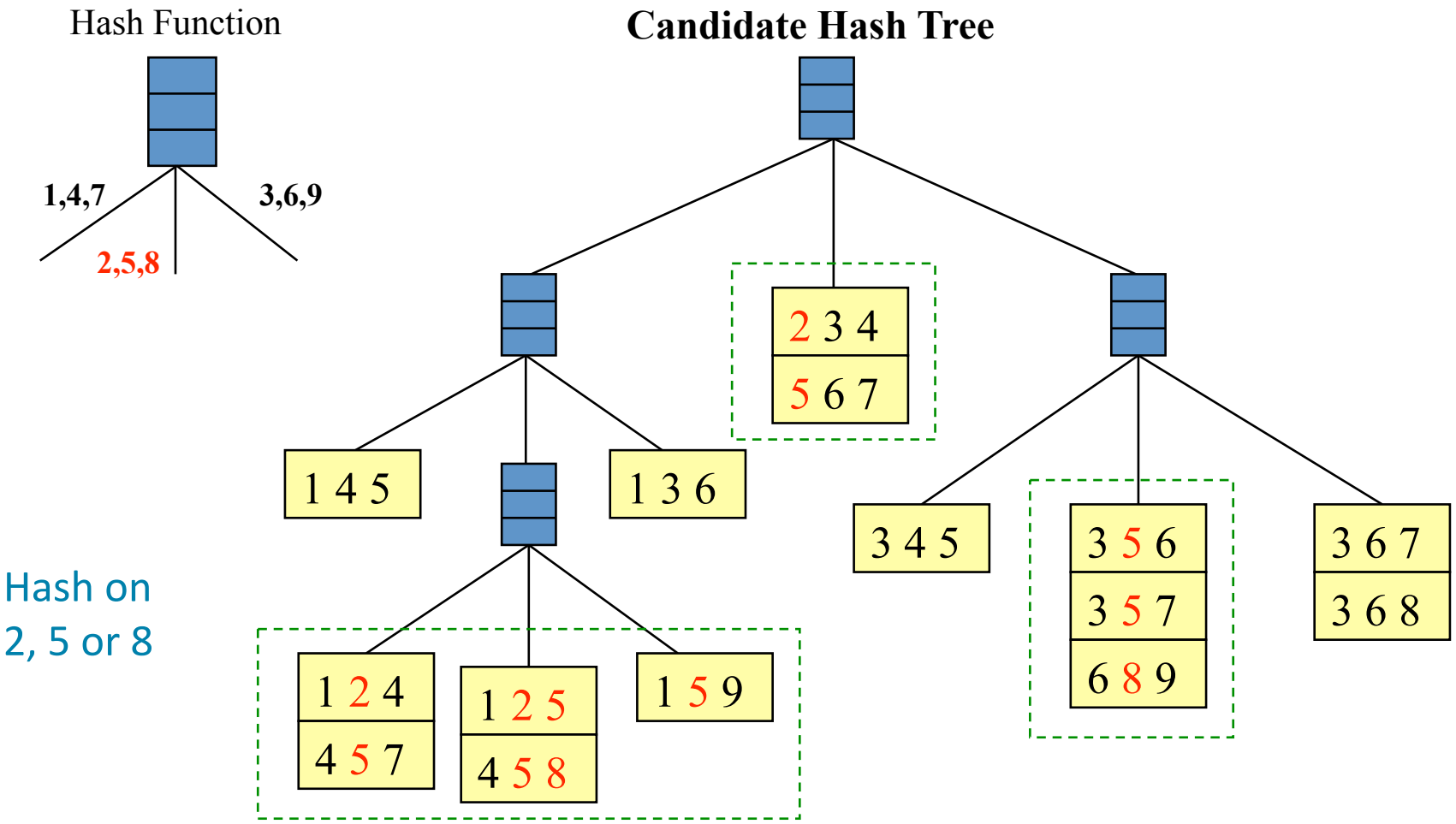




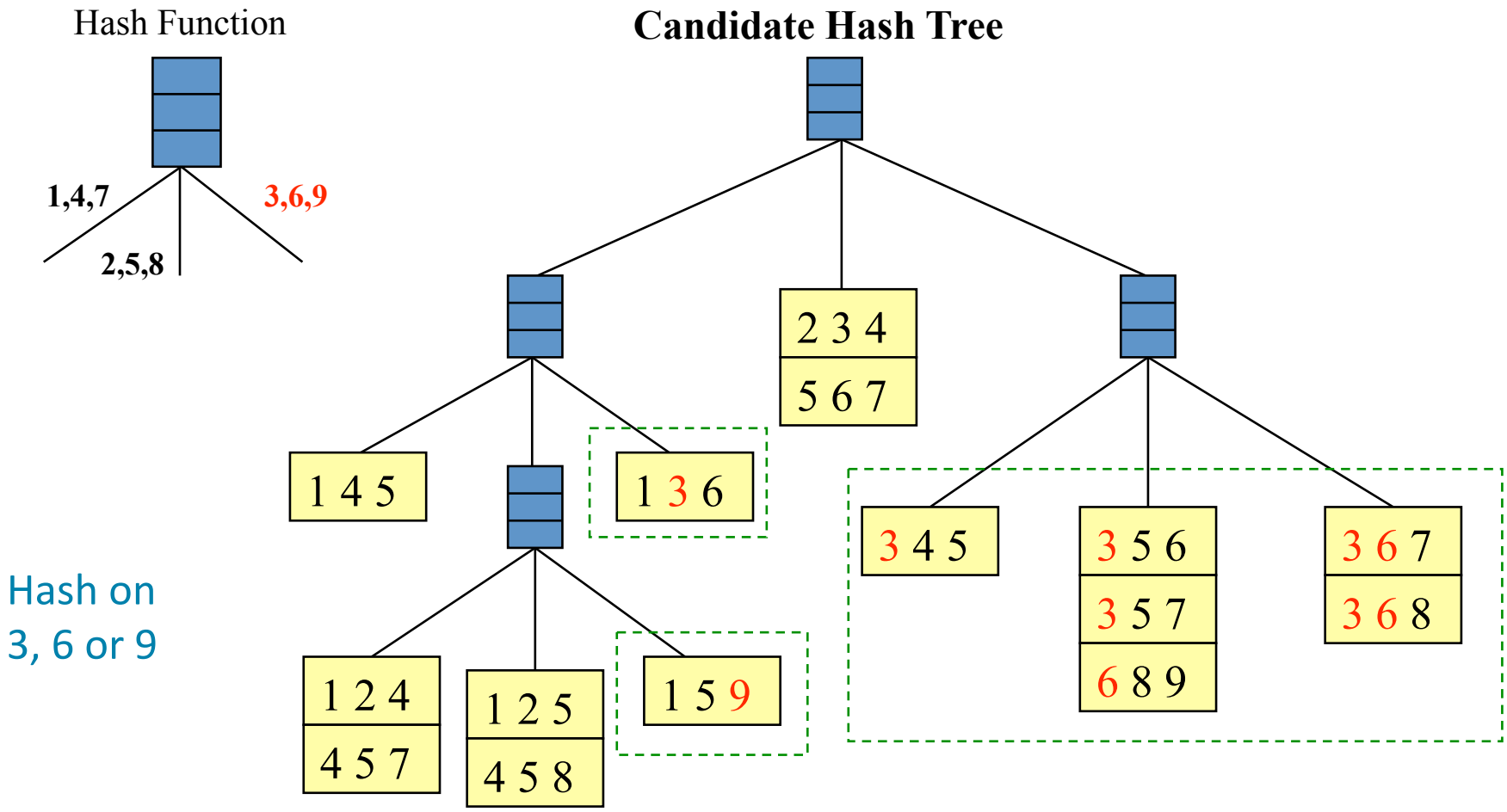
Association Rule Discovery: Hash tree



Association Rule Discovery: Hash tree

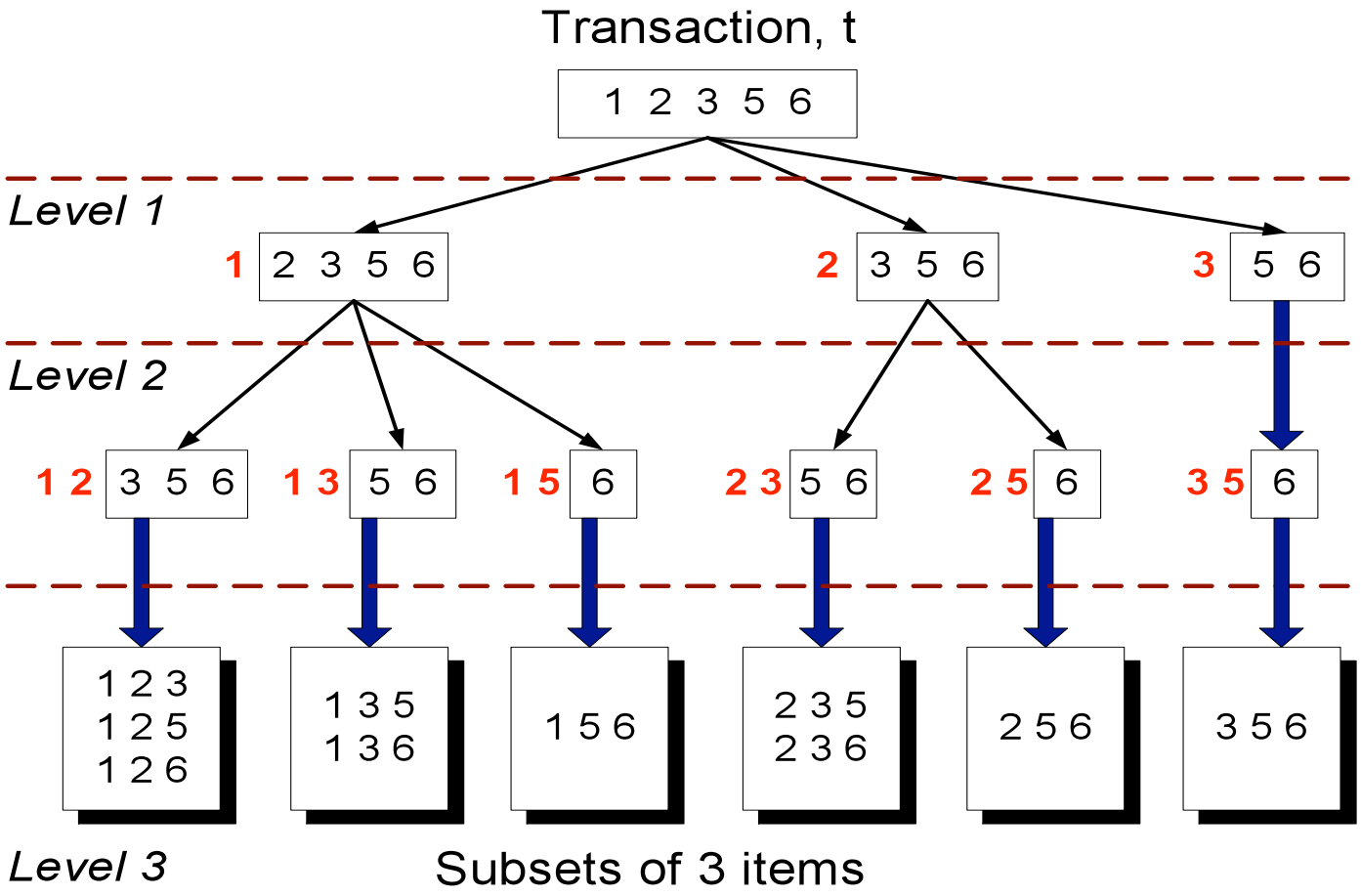


Association Rule Discovery: Hash tree

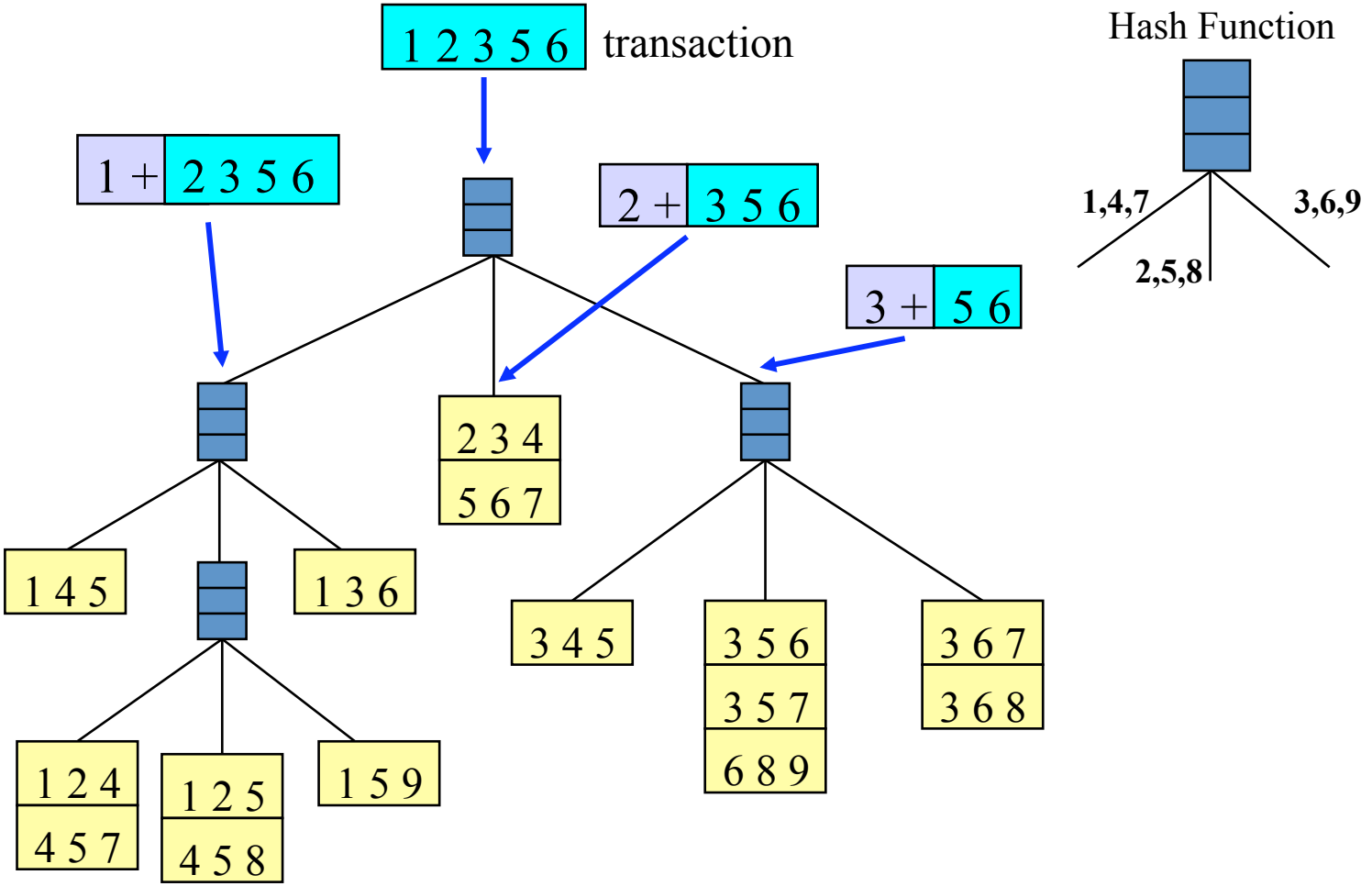


Subset Operation

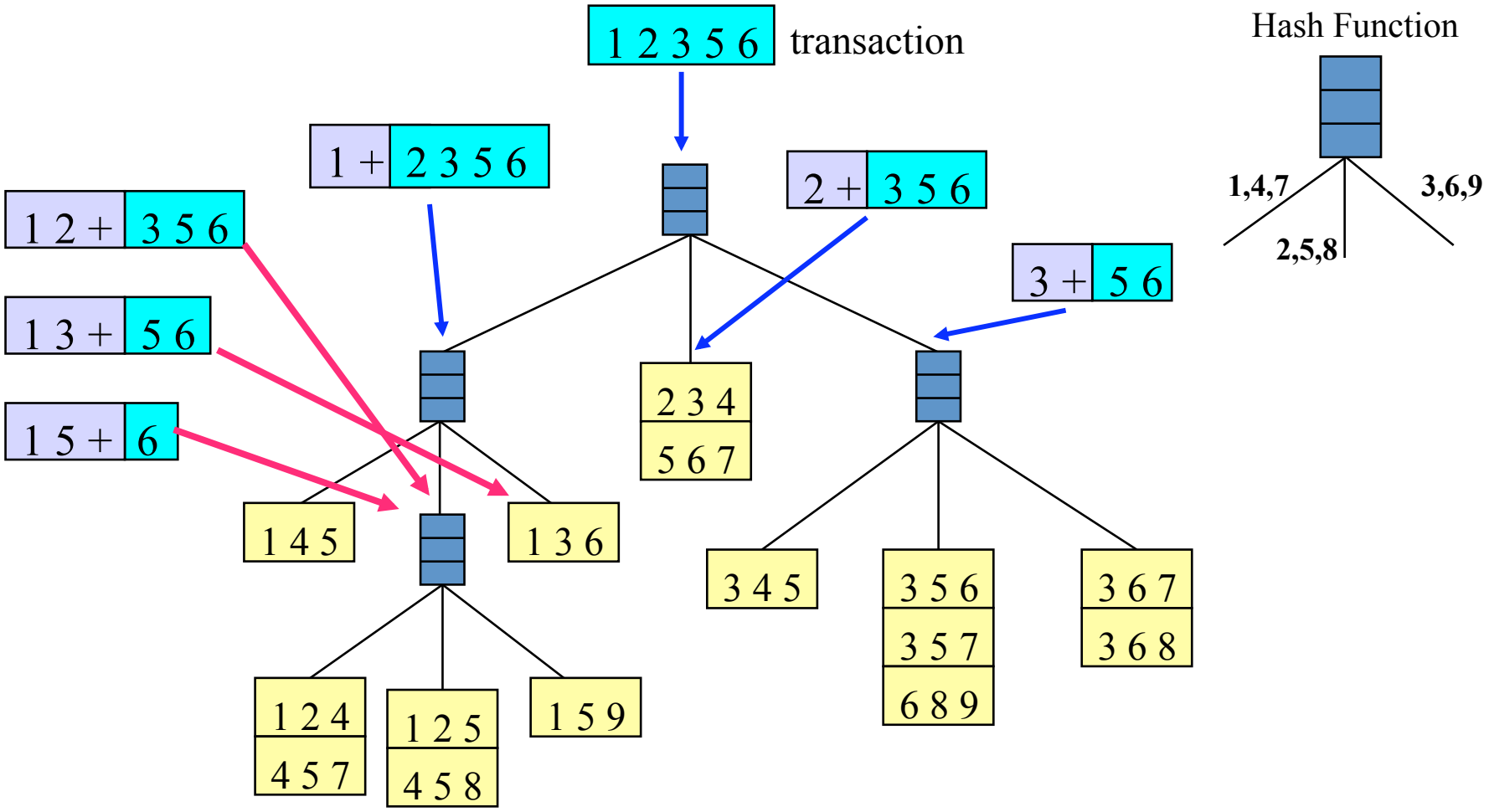
Given a transaction t, what are the possible subsets of size 3?



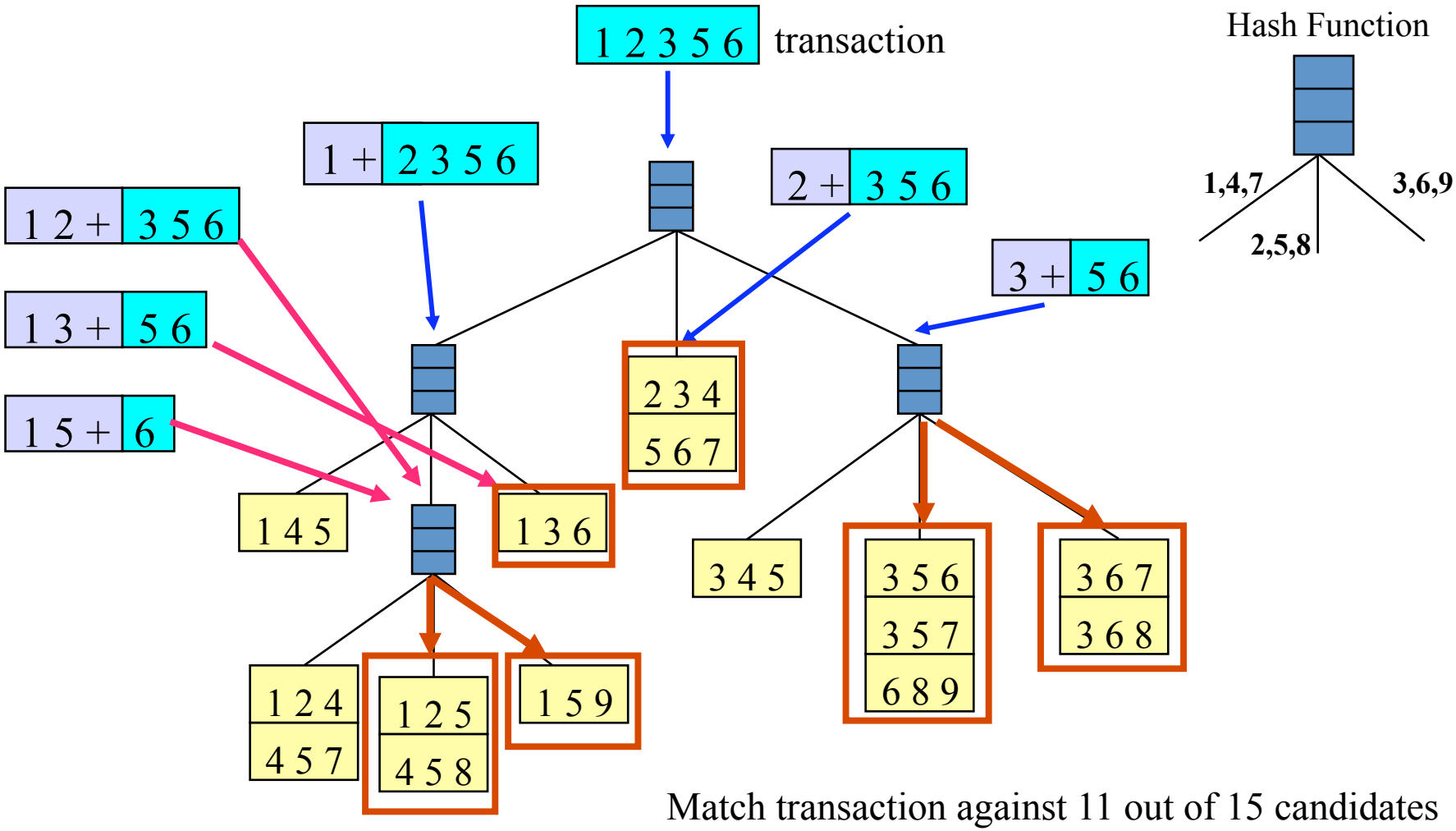
Subset Operation Using Hash Tree



Subset Operation Using Hash Tree



Subset Operation Using Hash Tree



Match transaction against 11 out of 15 candidates

Prefix Tree Representation

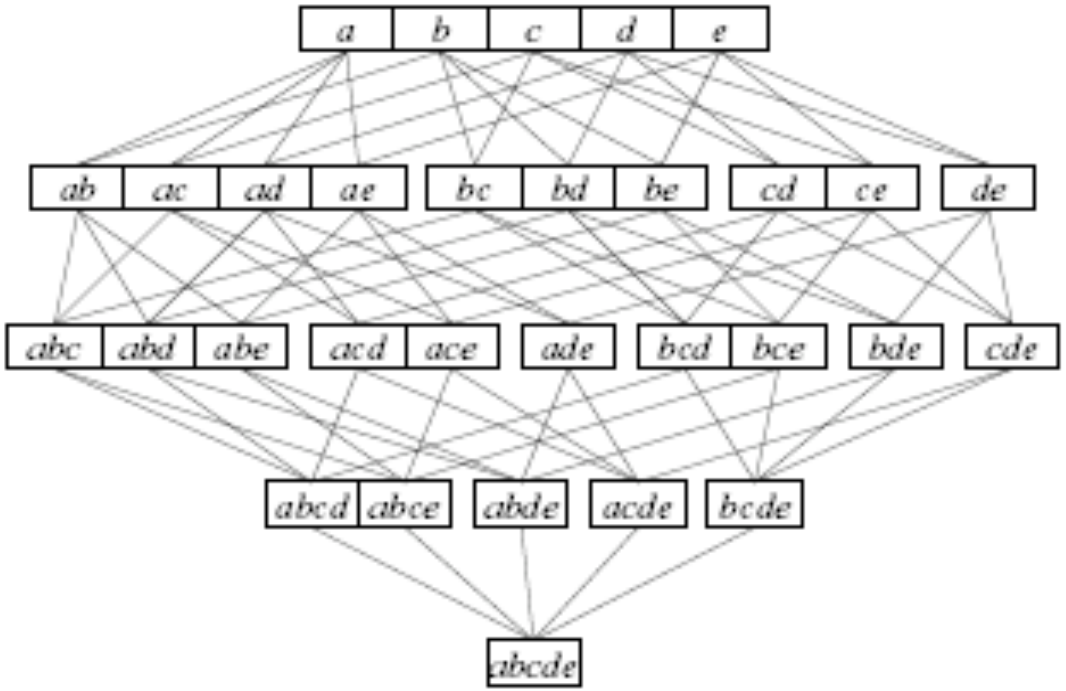


Figure 1. A subset lattice for five items (empty set omitted).

Efficient Implementations of Apriori and Eclat
Christian Borgelt., FIMI'03

Prefix Tree

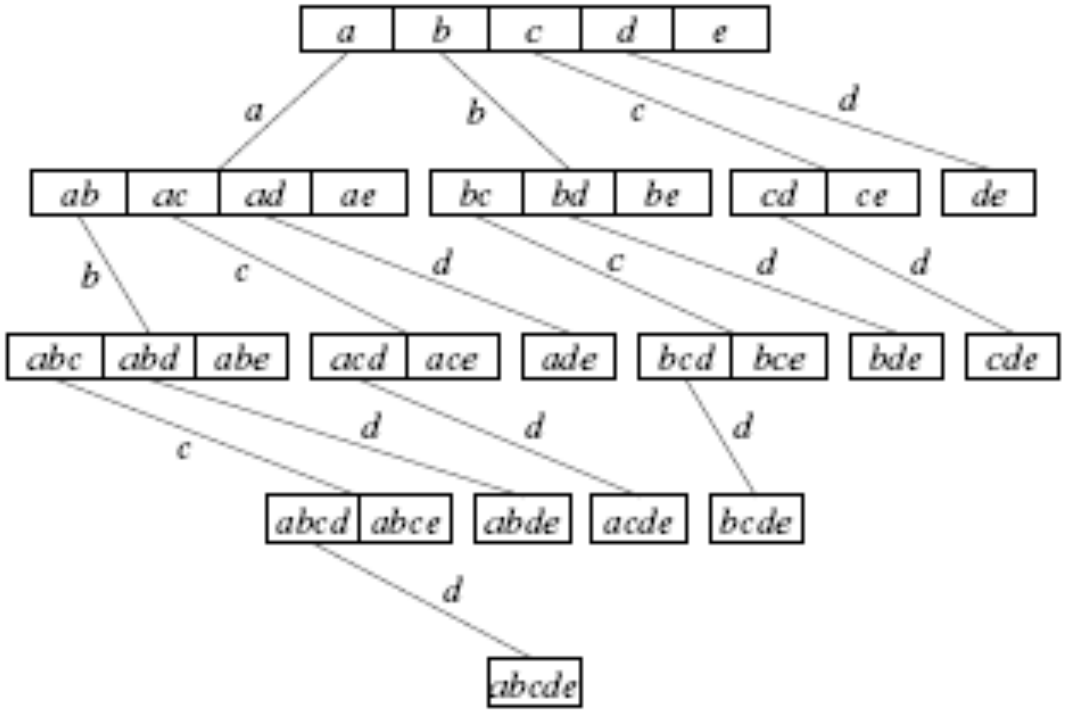


Figure 2. A prefix tree for five items (empty set omitted).

Prefix Tree Structure for Counting

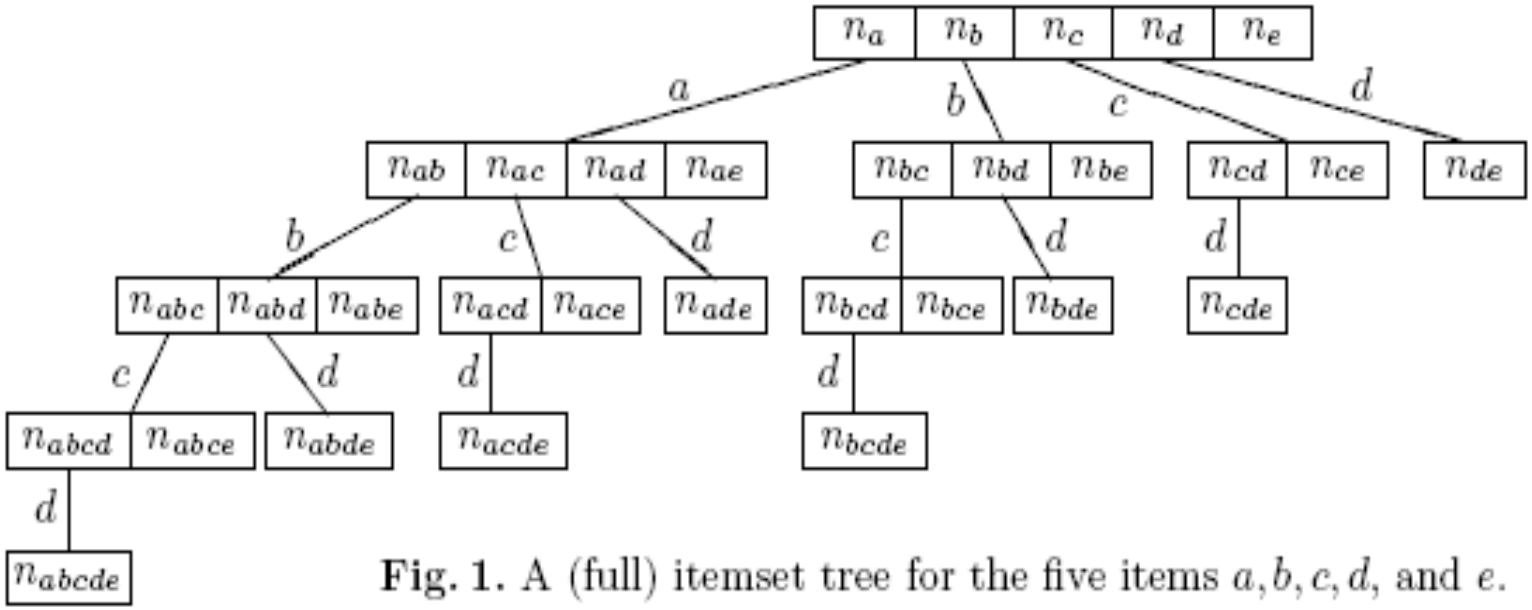


Fig. 1. A (full) itemset tree for the five items a, b, c, d, and e.

Other key optimization

- Reordering the items
  - Why is this relevant?
- Transaction Tree
  - Organize transaction into trees
  - Count through two trees

Important websites:

- FIMI workshop
  - Not only Apriori and FIM
    - FP-tree, ECLAT, Closed, Maximal
  - http://fimi.cs.helsinki.fi/
- Christian Borgelt's website
  - http://www.borgelt.net/software.html
- Ferenc Bodon's website
  - http://www.cs.bme.hu/~bodon/en/apriori/

References:

- Christian Borgelt**,** *Efficient Implementations of Apriori and Eclat*, FIMI'03
- Ferenc Bodon, *A fast APRIORI implementation*, FIMI'03
- Ferenc Bodon, *A Survey on Frequent Itemset Mining*, Technical Report, Budapest University of Technology and Economic, 2006

Scalability

- How to handle very large dataset?
- The dataset can not be stored in the main memory
- Performance of out-of-core datasets/ Performance of in-core datasets

Partition: Scan Database Only Twice

- Any itemset that is potentially frequent in DB must be frequent in at least one of the partitions of DB
  - Scan 1: partition database and find local frequent patterns
  - Scan 2: consolidate global frequent patterns
- A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association in large databases. In *VLDB'95*

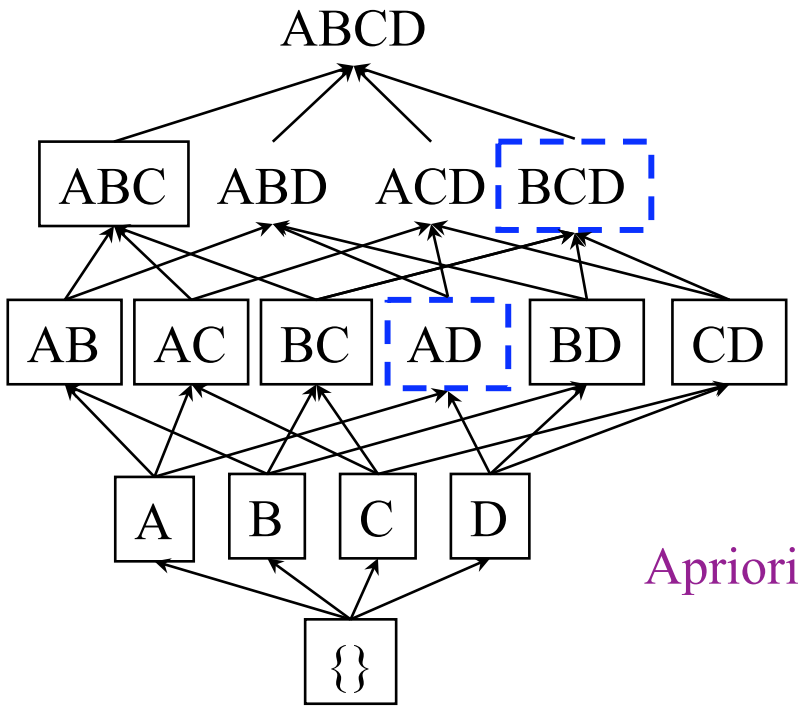
DHP: Reduce the Number of Candidates

- A *k*-itemset whose corresponding hashing bucket count is below the threshold cannot be frequent
  - Candidates: a, b, c, d, e
  - Hash entries: {ab, ad, ae} {bd, be, de} …
  - Frequent 1-itemset: a, b, d, e
  - ab is not a candidate 2-itemset if the sum of count of {ab, ad, ae} is below support threshold
- J. Park, M. Chen, and P. Yu. An effective hash-based algorithm for mining association rules. In *SIGMOD'95*

Sampling for Frequent Patterns

- Select a sample of original database, mine frequent patterns within sample using Apriori
- Scan database once to verify frequent itemsets found in sample, only *borders* of closure of frequent patterns are checked
  - Example: check *abcd* instead of *ab, ac, …, etc.*
- Scan database again to find missed frequent patterns
- H. Toivonen. Sampling large databases for association rules. In *VLDB'96*

DIC: Reduce Number of Scans



Itemset lattice

S. Brin R. Motwani, J. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket data. In *SIGMOD'97*

- Once both A and D are determined frequent, the counting of AD begins
- Once all length-2 subsets of BCD are determined frequent, the counting of BCD begins