ASSOCIATION RULE MINING

BEIYU LIN

FREQUENT ITEMSET GENERATION

- Brute-force approach:
  - Each itemset in the lattice is a candidate frequent itemset
  - Count the support of each candidate by scanning the database

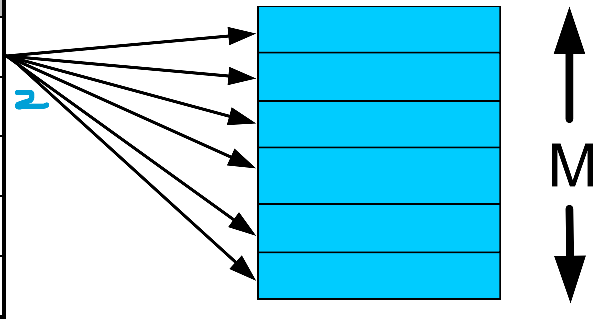
Transactions

N

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

w

msef= **List of Candidates**



  - Match each transaction against every candidate
  - Complexity ~ O(NMw) => Expensive since $M = 2^d$!!!

FREQUENT ITEMSET GENERATION STRATEGIES

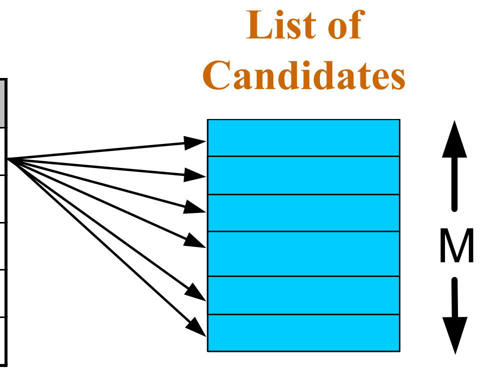
- Reduce the number of candidates (M)
  - Complete search: $M=2^d$
  - Use pruning techniques to reduce M
- Reduce the number of transactions (N)
  - Reduce size of N as the size of itemset increases
  - Used by DHP and vertical-based mining algorithms
- Reduce the number of comparisons (NM)
  - Use efficient data structures to store the candidates or transactions
  - No need to match every candidate against every transaction

Transactions

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

N

w



Given a transaction {B, M, D, C}, find all possible subset with size 3 from this transaction.

REDUCING NUMBER OF CANDIDATES

- Apriori principle:
  - If an itemset is frequent, then all of its subsets must also be frequent
- Apriori principle holds due to the following property of the support measure:

$$\forall X, Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y)$$

  - Support of an itemset never exceeds the support of its subsets
  - This is known as the anti-monotone property of support

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

Support (ratio)

s({Milk, Bread, Diaper}) = 2/5 = # itemsets / total # of transaction $= \frac{2}{5}$

σ **Support count:** # of the itemsets that show in the transaction $= 2$

Confidence $= \frac{\# X \cup Y}{\# X} = \frac{2}{3}$

$s = \frac{\sigma}{\text{totalT}}$

$X = \{M, B\}$

$Y = \{D\}$

$X \rightarrow Y$

$X \cup Y = \{M, B, D\}$

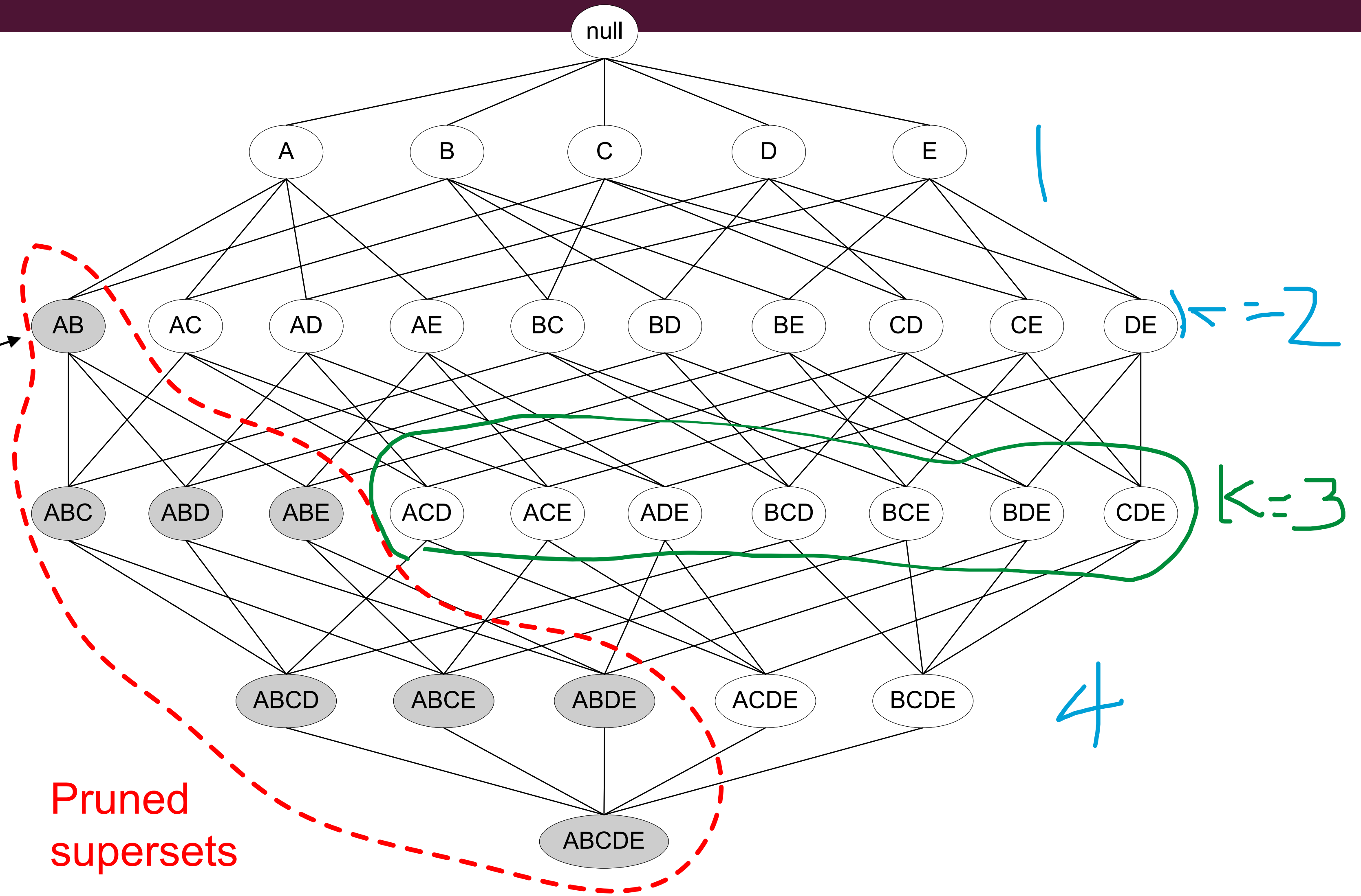
ILLUSTRATING APRIORI PRINCIPLE

$\sigma \geq \min \sigma$

$s \geq \min s$

$s = \dfrac{\sigma}{T}$

Found to be Infrequent



SUPPORT COUNTING OF CANDIDATE ITEMSETS

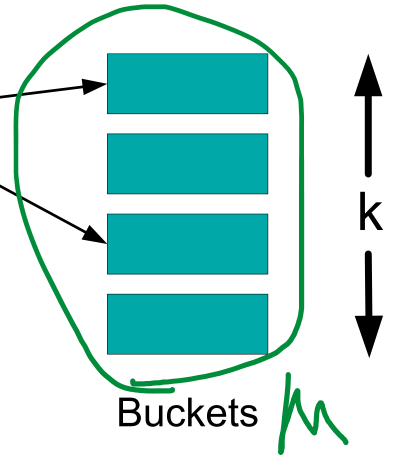
- To reduce number of comparisons, store the candidate itemsets in a hash structure / hash function
  - Instead of matching each transaction against every candidate, match it against candidates contained in the hashed buckets

Transactions

N

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

Hash Structure



N . M

SUPPORT COUNTING: AN EXAMPLE

Suppose you have 15 candidate itemsets of length 3: → Reduce M from Apriori Alg

{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}

How many of these itemsets are supported by transaction (1,2,3,5,6)?

itemsets pruned

N

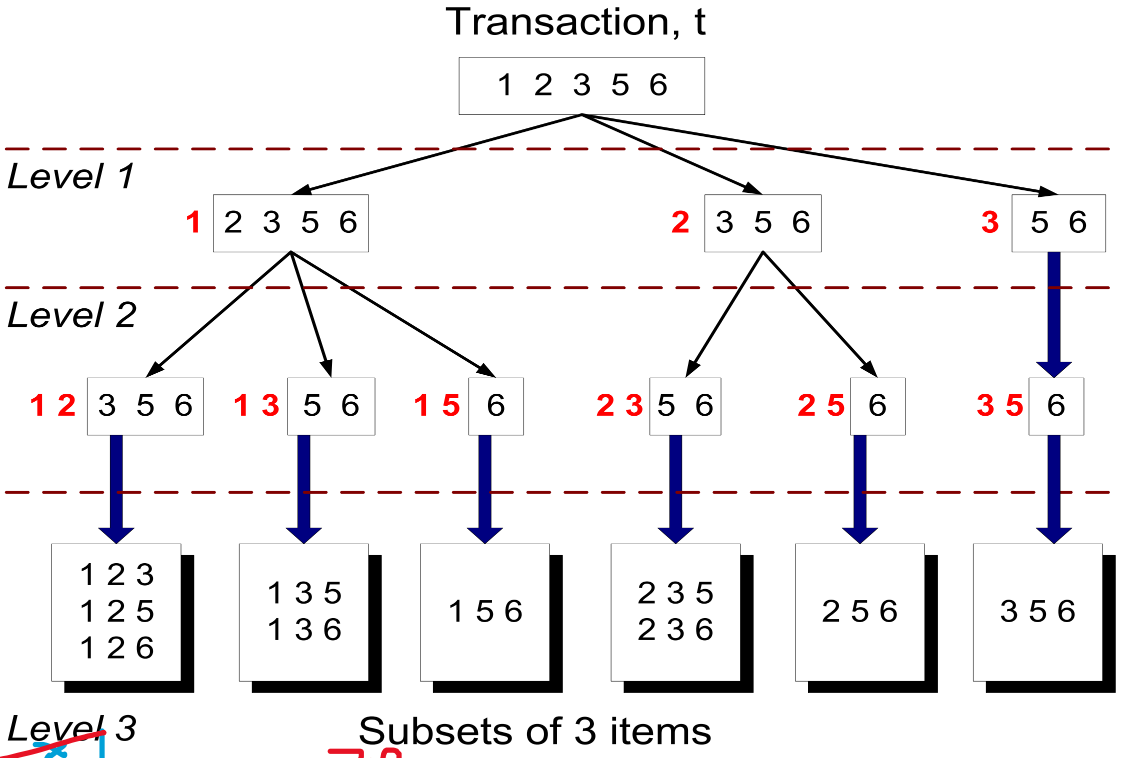
1	
⋮	
5	12356

M

145
124

k=3

$$C^n_r = C^5_3 = \frac{n!}{r!(n-r)!} = \frac{5!}{3!\,2!} = \frac{5\times4\times\cancel{3\times2\times1}}{\cancel{3\times2\times1}\cdot\cancel{2\times1}} = \frac{20}{2} = 10$$



SUPPORT COUNTING: AN EXAMPLE

Suppose you have 15 candidate itemsets of length 3: → Reduce M from Apriori Alg

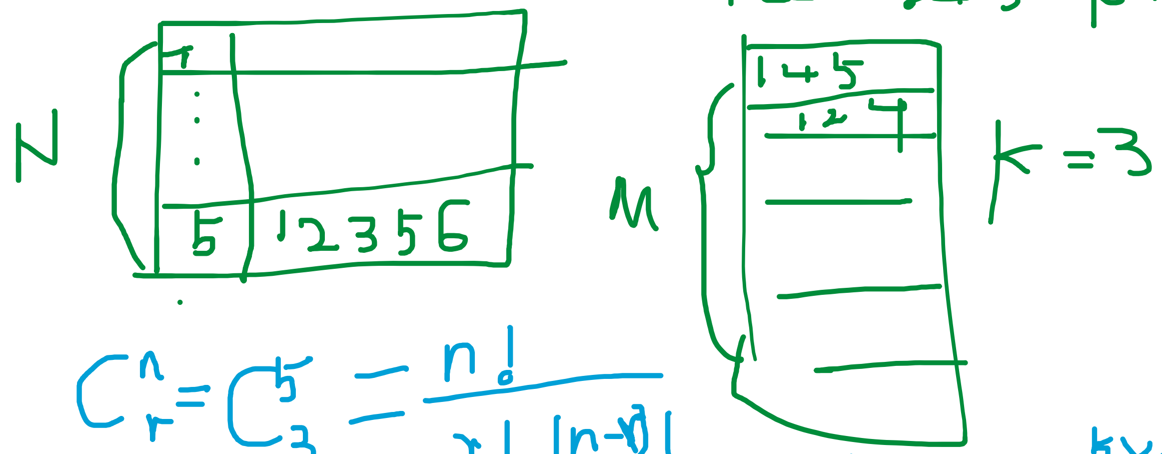
{1 4 5}, {1 2 4}, {4 5 7}, {1 2 5}, {4 5 8}, {1 5 9}, {1 3 6}, {2 3 4}, {5 6 7}, {3 4 5}, {3 5 6}, {3 5 7}, {6 8 9}, {3 6 7}, {3 6 8}

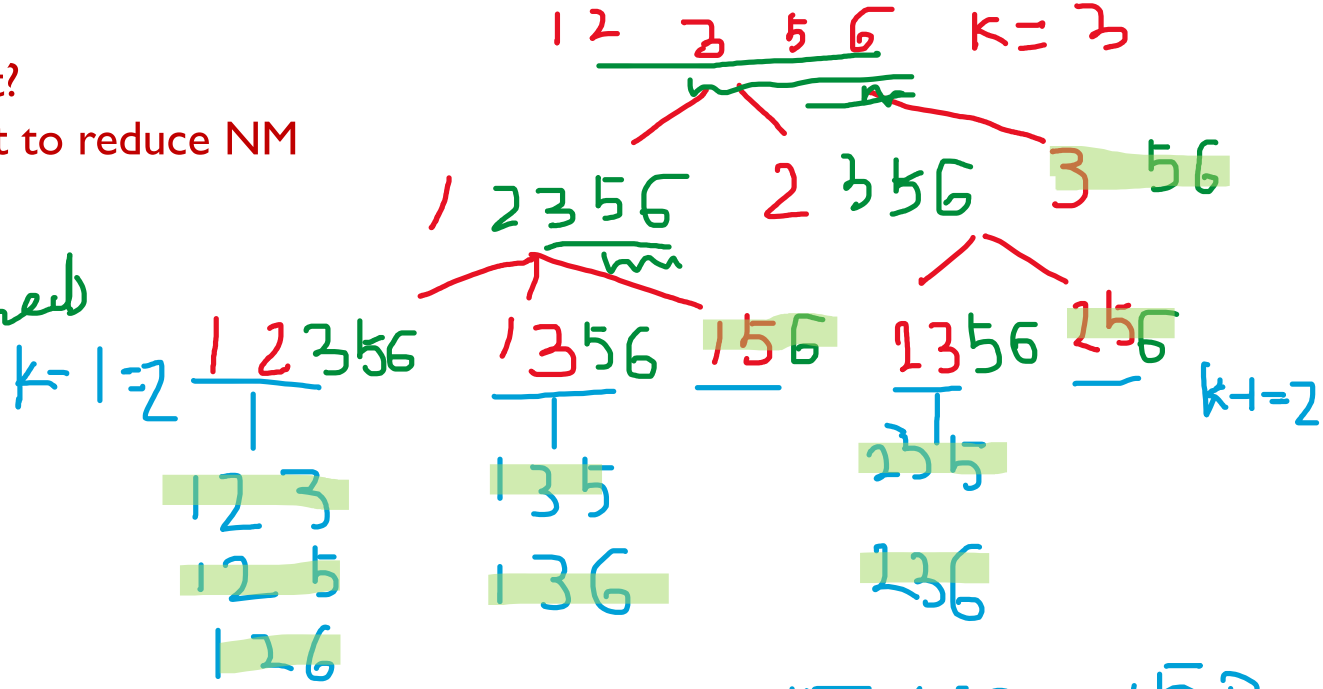
How to find all the subsets with k items from an itemset?

Given an itemset {1,2,3,5,6} ⇔ one transaction, we want to reduce NM

How many of these itemsets are supported by transaction (1,2,3,5,6)?

itemsets pruned





$C^n_r = C^5_3 = \frac{n!}{r!\,(n-r)!} = \frac{5!}{3!\,2!} = \frac{5\times4\times\cancel{3\times2\times1}}{\cancel{3\times2\times1}\cdot2\times1} = \frac{20}{2} = 10$

$15\times10 = 150$

SUPPORT COUNTING USING A HASH TREE

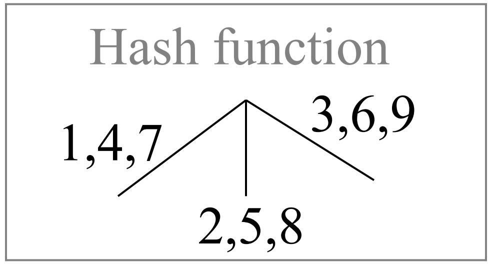
Suppose you have 15 candidate itemsets of length 3:

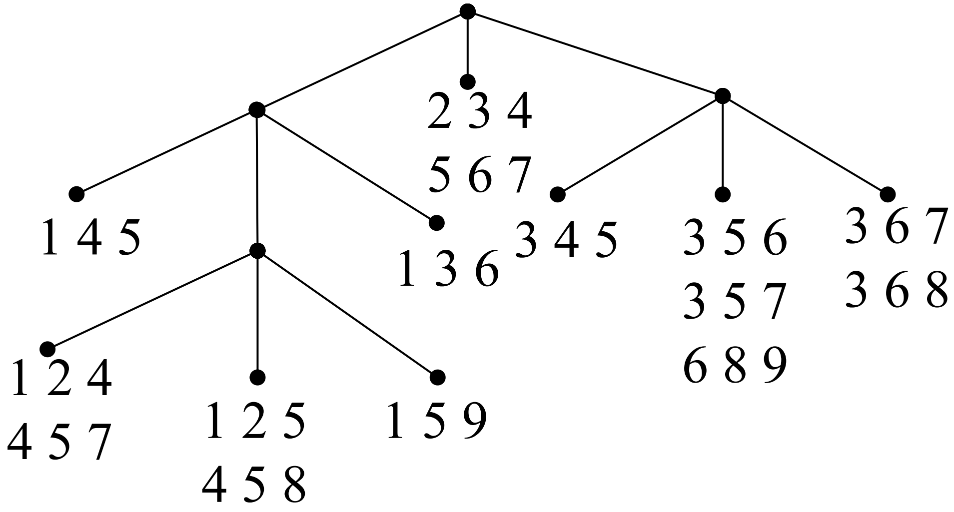
{1 4 5}, {1 2 4}, {4 5 7}, {1 2 5}, {4 5 8}, {1 5 9}, {1 3 6}, {2 3 4}, {5 6 7}, {3 4 5}, {3 5 6}, {3 5 7}, {6 8 9}, {3 6 7}, {3 6 8}

You need:

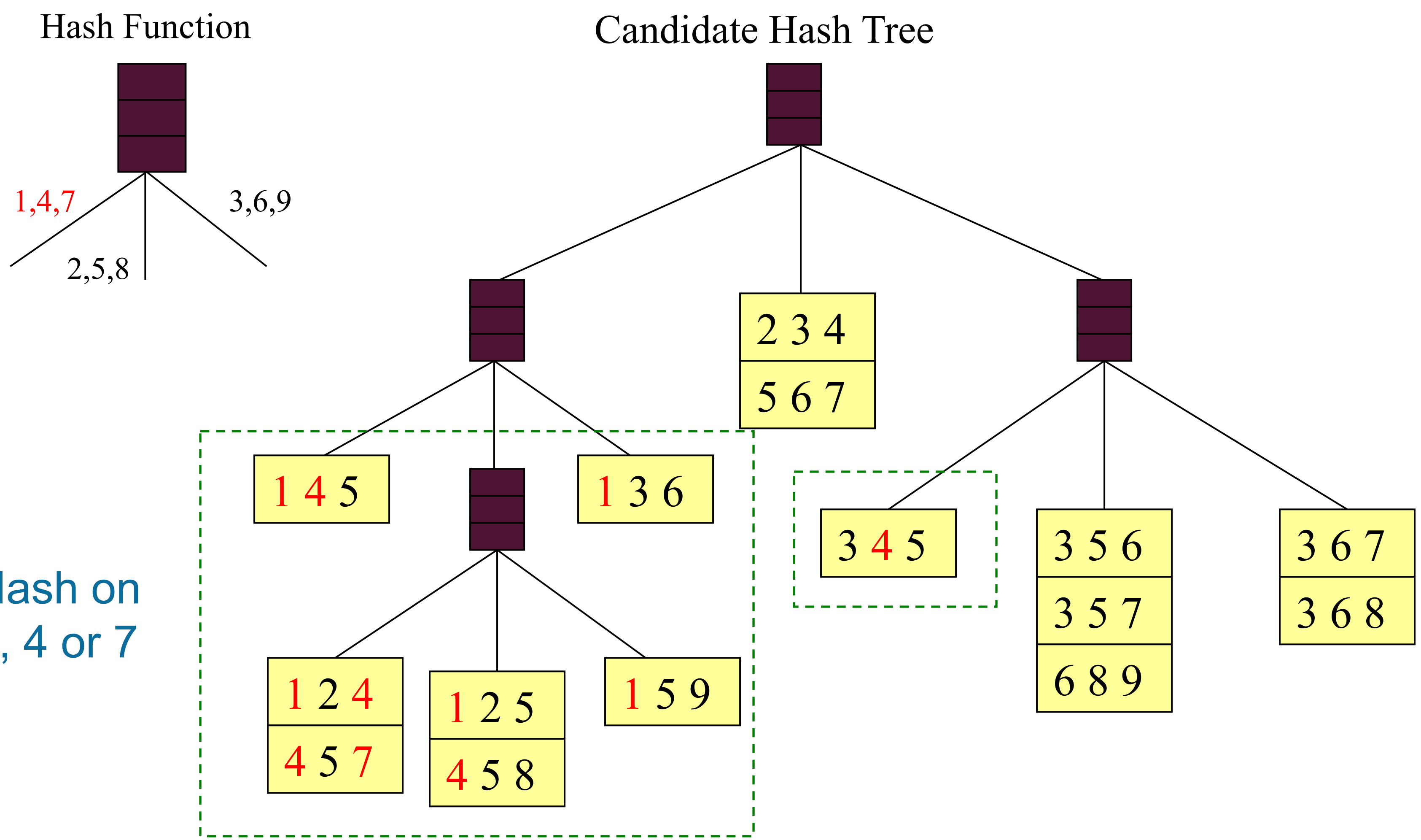
• Hash function

• Max leaf size: max number of itemsets stored in a leaf node (if number of candidate itemsets exceeds max leaf size, split the node)



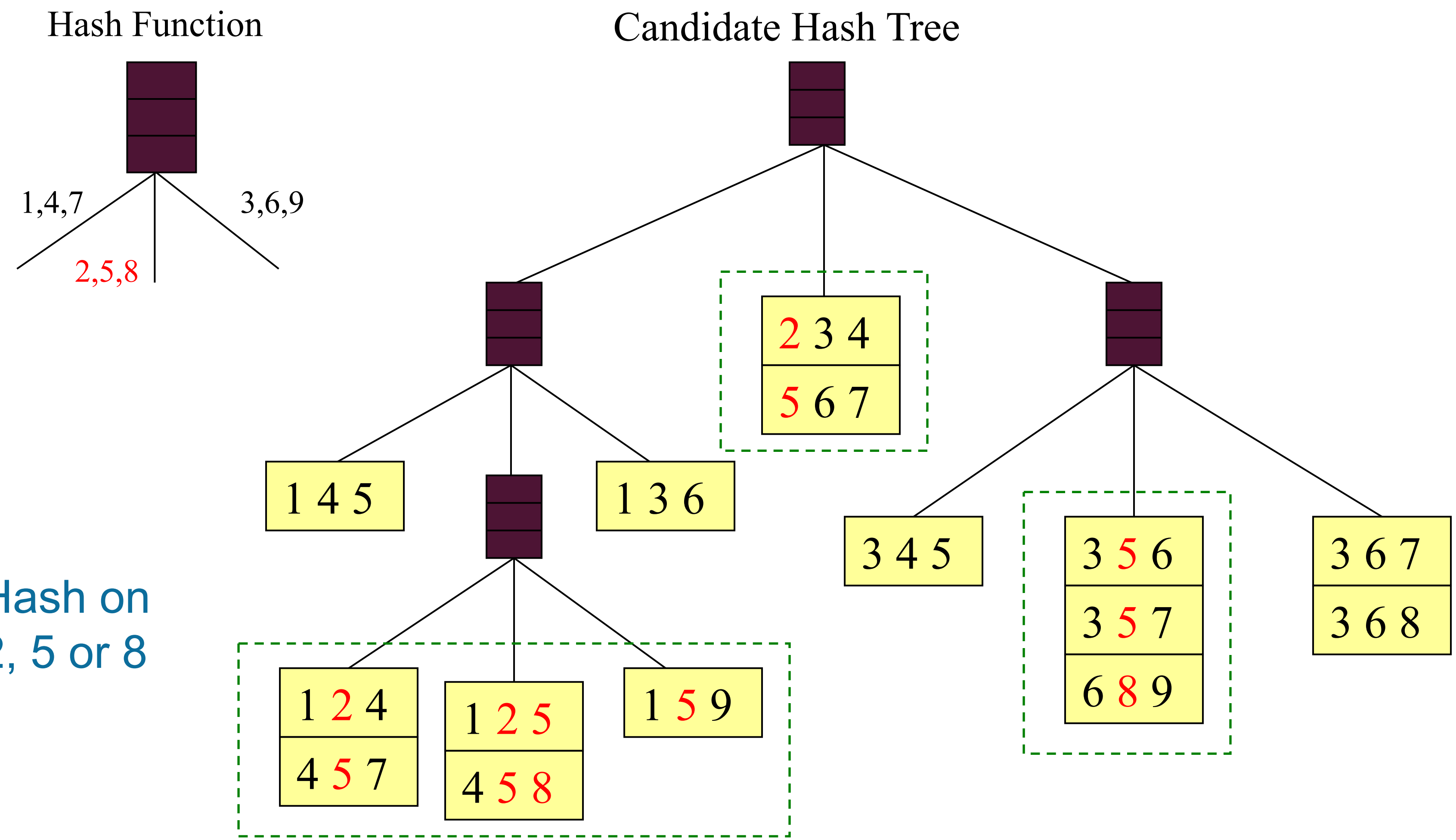


SUPPORT COUNTING USING A HASH TREE



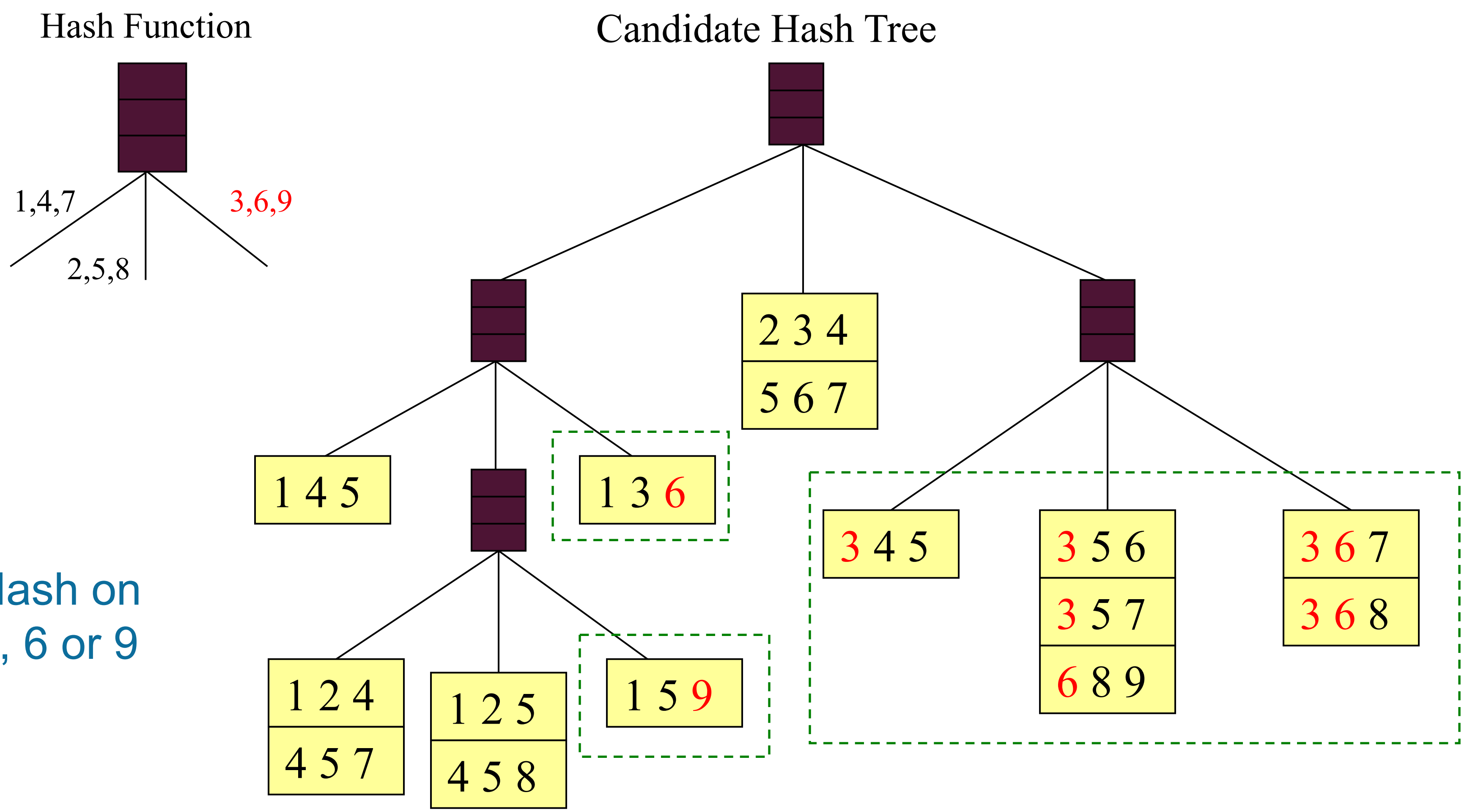
Hash on
1, 4 or 7

SUPPORT COUNTING USING A HASH TREE



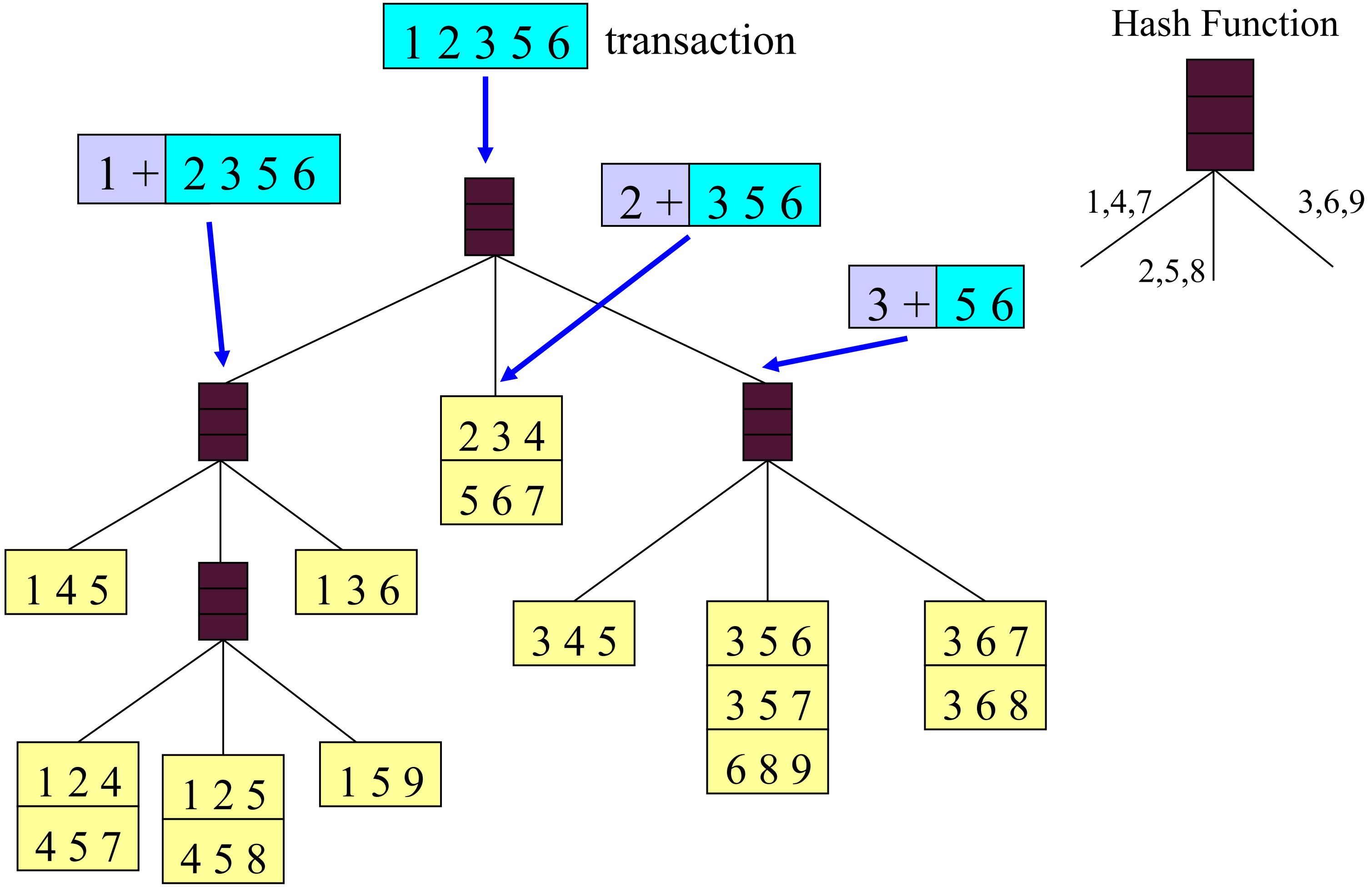
Hash on
2, 5 or 8

SUPPORT COUNTING USING A HASH TREE

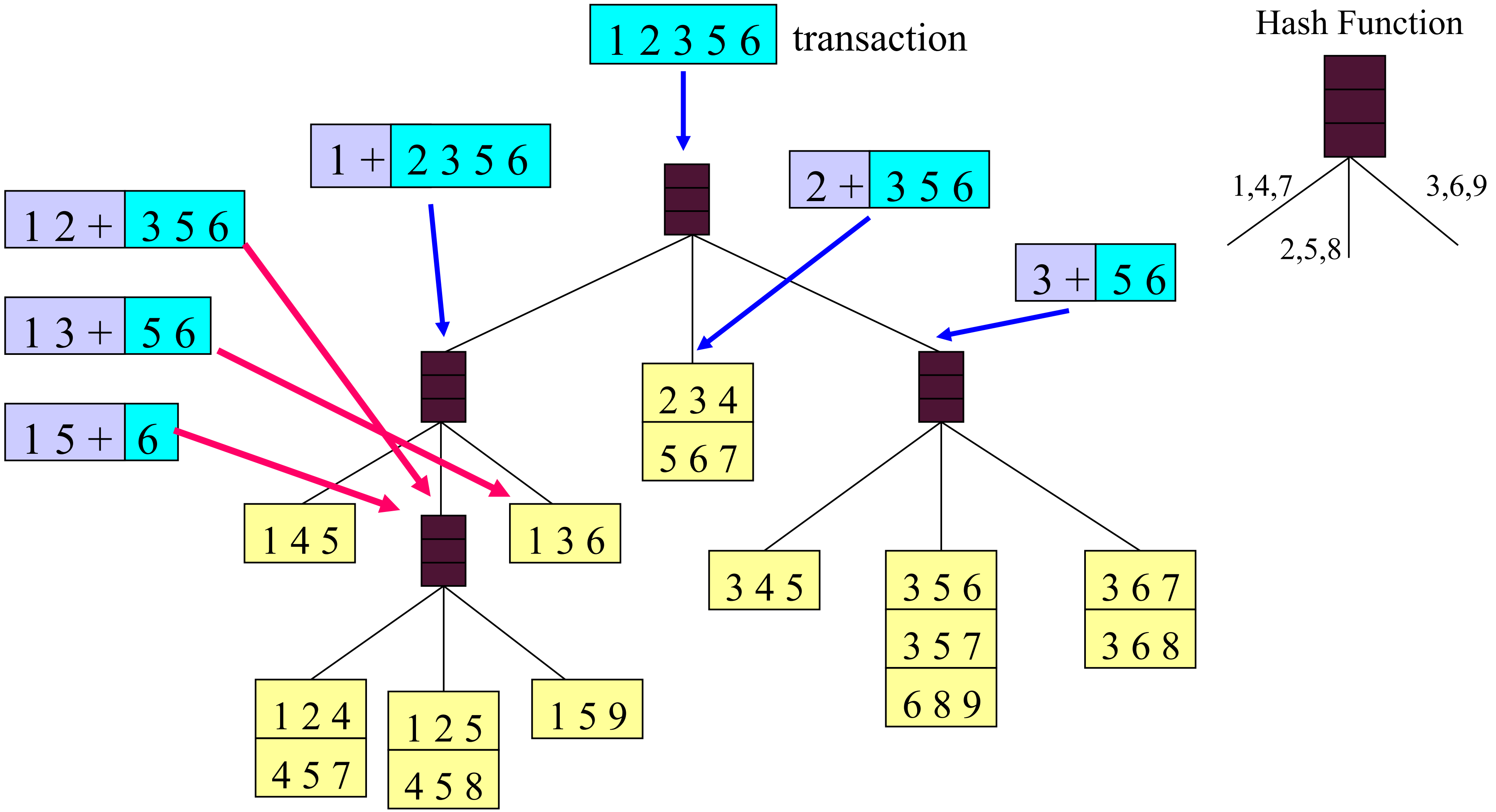


Hash on
3, 6 or 9

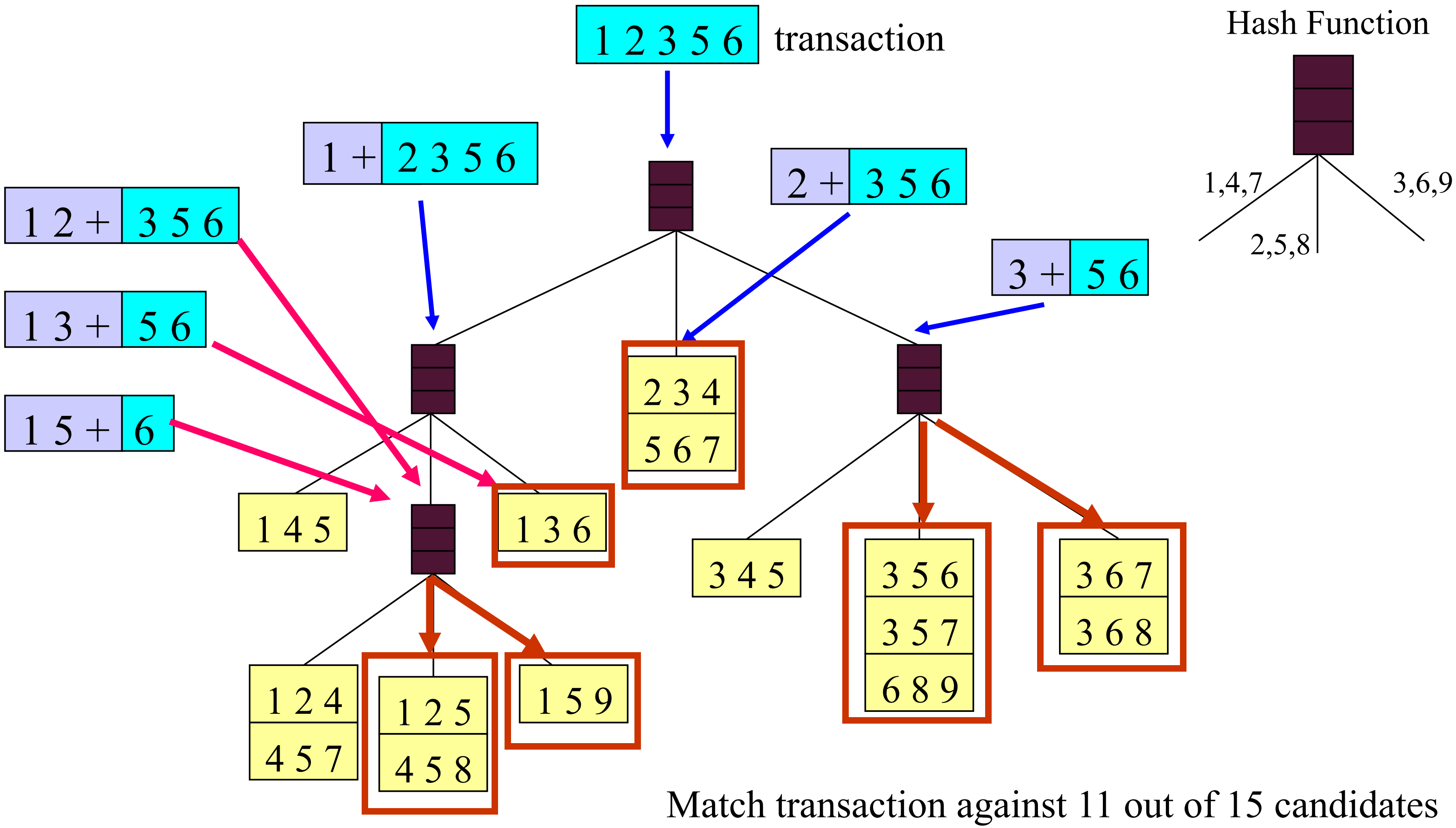
SUPPORT COUNTING USING A HASH TREE



SUPPORT COUNTING USING A HASH TREE



SUPPORT COUNTING USING A HASH TREE



Match transaction against 11 out of 15 candidates