

Speeding-up Association Rules Mining

Mining Massive Datasets

Prof. Carlos Castillo — <https://chato.cl/teach>



Universitat
Pompeu Fabra
Barcelona

Sources

- Data Mining, The Textbook (2015) by Charu Aggarwal (Chapters 4, 5) – [slides by Lijun Zhang](#)
- Mining of Massive Datasets 2nd edition (2014) by Leskovec et al. ([Chapter 6](#)) – [slides](#)
- Data Mining Concepts and Techniques, 3rd edition (2011) by Han et al. (Chapter 6)
- Introduction to Data Mining 2nd edition (2019) by Tan et al. (Chapters 5, 6) – [slides ch5](#), [slides ch6](#)

Speeding up candidate generation

Level-wise pruning trick

- Let F_k be the set of frequent k -itemsets
- Let C_{k+1} be the set of $(k+1)$ -candidates
- $I \in C_{k+1}$ is frequent only if all the k -subsets of I are frequent
- Pruning
 - Generate all the k -subsets of I
 - If any one of them does not belong to F_k , then remove I

Candidates generation

- A Naïve Approach
 - Check all the possible combinations of frequent itemsets
- An Example of the Naïve Approach
 - itemsets: $\{abc\}$ $\{bcd\}$ $\{abd\}$ $\{cde\}$
 - $\{abc\} + \{bcd\} = \{abcd\}$
 - $\{bcd\} + \{abd\} = \{abcd\}$
 - $\{abd\} + \{cde\} = \{abcde\}$
 -

Candidates generation (cont.)

- Introduction of Ordering
 - Items in U have a lexicographic ordering
 - Itemsets can be order as strings
- A Better Approach
 - Order the frequent k -itemsets
 - Merge two itemset if the first $k-1$ items of them are the same

Candidates generation (cont.)

- Example
 - k-itemsets: $\{abc\}$ $\{abd\}$ $\{bcd\}$
 - $\{abc\} + \{abd\} = \{abcd\}$
- k-itemsets: $\{abc\}$ $\{acd\}$ $\{bcd\}$
 - No $(k+1)$ -candidates
- Early stop is possible
 - Do not need to check $\{abc\} + \{bcd\}$ after checking $\{abc\} + \{acd\}$
- Do we miss $\{abcd\}$?
 - No, due to the Downward Closure Property

Improving computation of support

Naïve support counting

- Naïve counting:

For each candidate $l_i \in C_{k+1}$

For each transaction T_j in T

Check whether l_i appears in T_j

- Limitation

- Inefficient if both $|C_{k+1}|$ and $|T|$ are large

Support counting with a data structure

- A Better Approach
 - Organize the candidate patterns in C_{k+1} in a data structure
- Use the data structure to accelerate counting
 - Each transaction in T_i examined against the subset of candidates in C_{k+1} that might contain T_i

Data structured for support counting based on hashing

Naïve counting:

For each $I_i \in C_{k+1}$

For all $T_j \in T$

If $I_i \subseteq T_j$

Add to $\text{sup}(I_i)$

Hashed counting:

For each $T_j \in T$

For $I_i \in \text{hashbucket}(T_j, C_{k+1})$

If $I_i \subseteq T_j$

Add to $\text{sup}(I_i)$

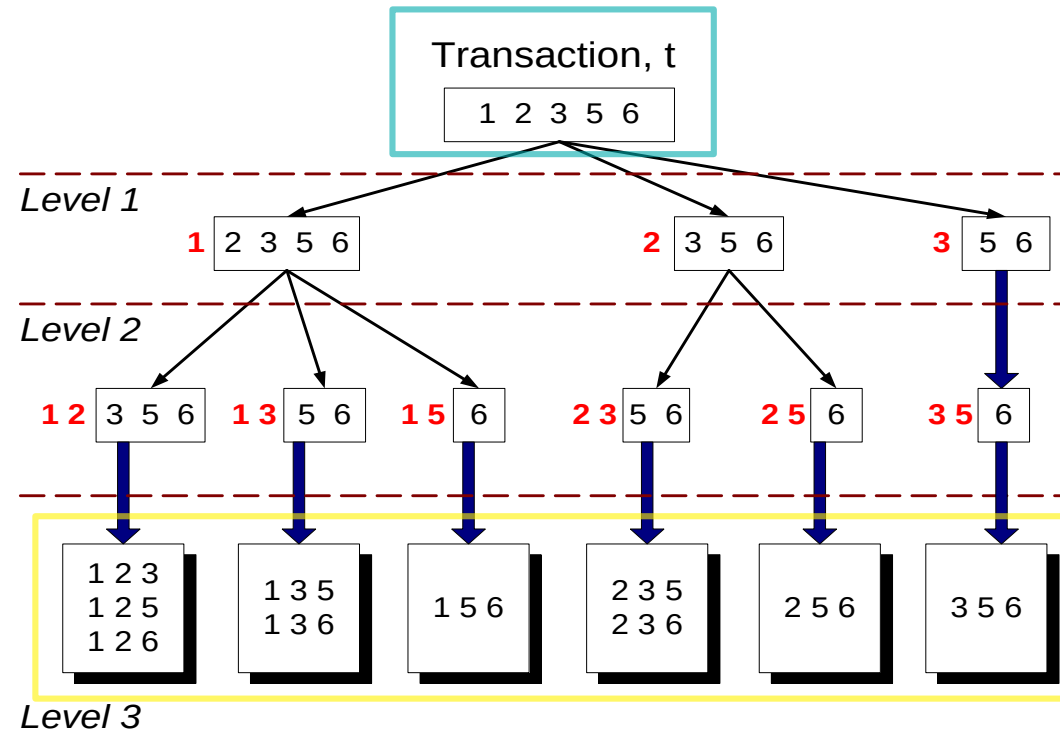
Which candidates are relevant?

Imagine 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}

Now, suppose we look
for this transaction:

{1 2 3 5 6}



Here we depict only the candidates that
appear in the transaction (10 out of 15)

Hash tree for itemsets in C_{k+1}

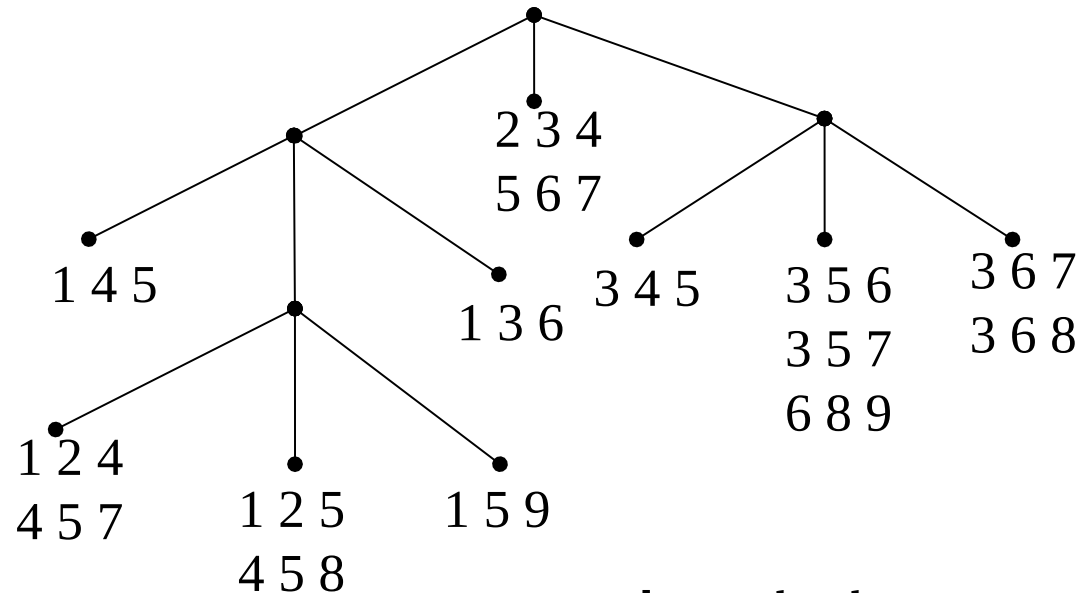
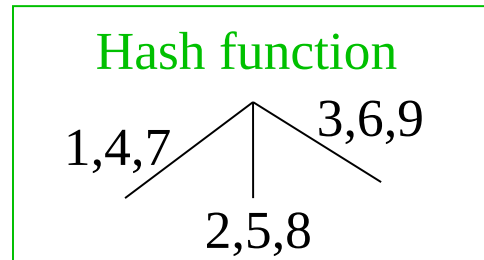
- A tree with fixed degree r
- Each itemset in C_{k+1} is stored in a leaf node
- All internal nodes use a hash function to map items to one of the r branches (can be the same for all internal nodes)
- All leaf nodes contain a lexicographically sorted list of up to `max_leaf_size` itemsets

Example hash tree

$r=3$ $\text{max_leaf_size}=3$

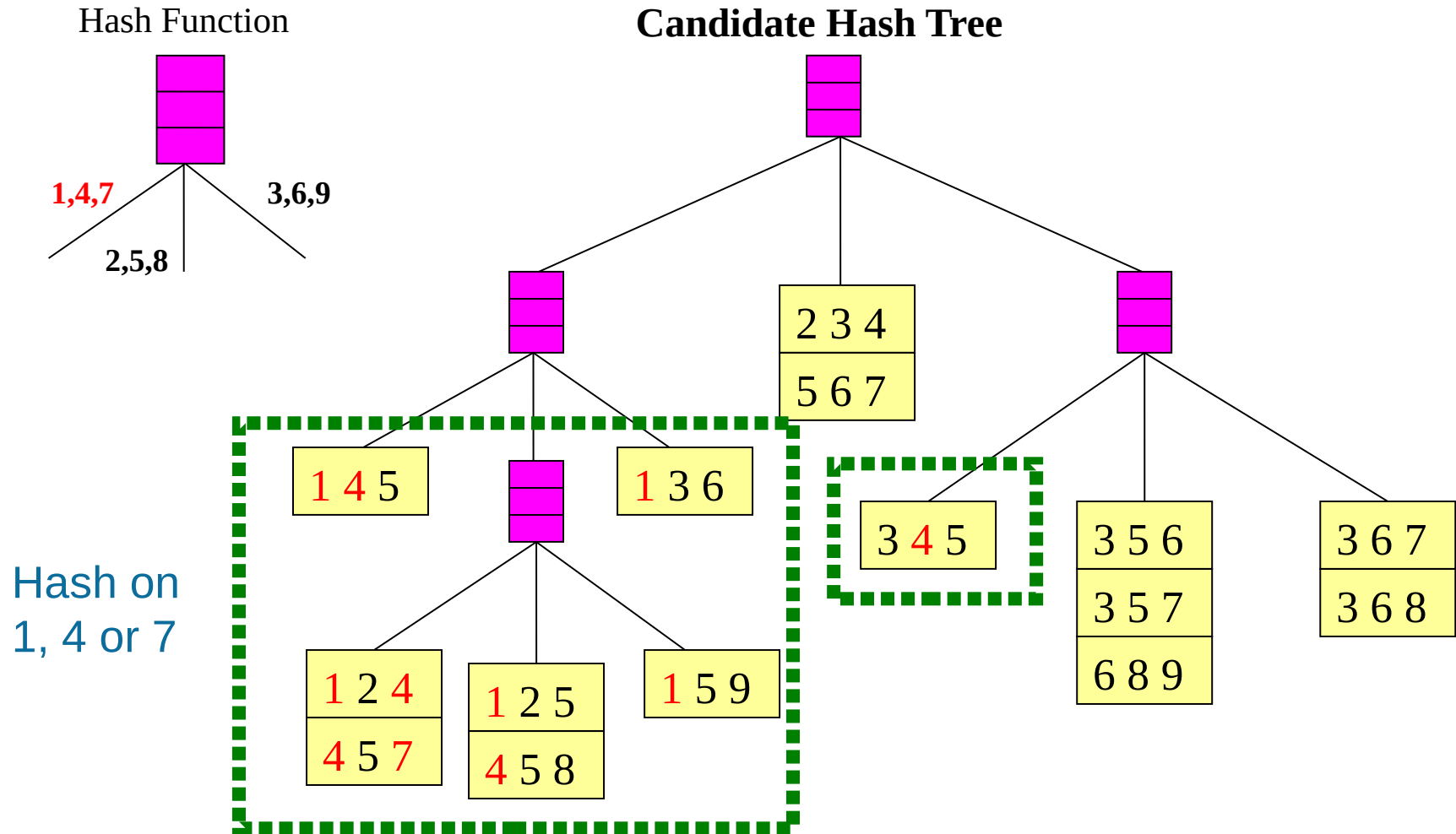
Candidate itemsets

{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}

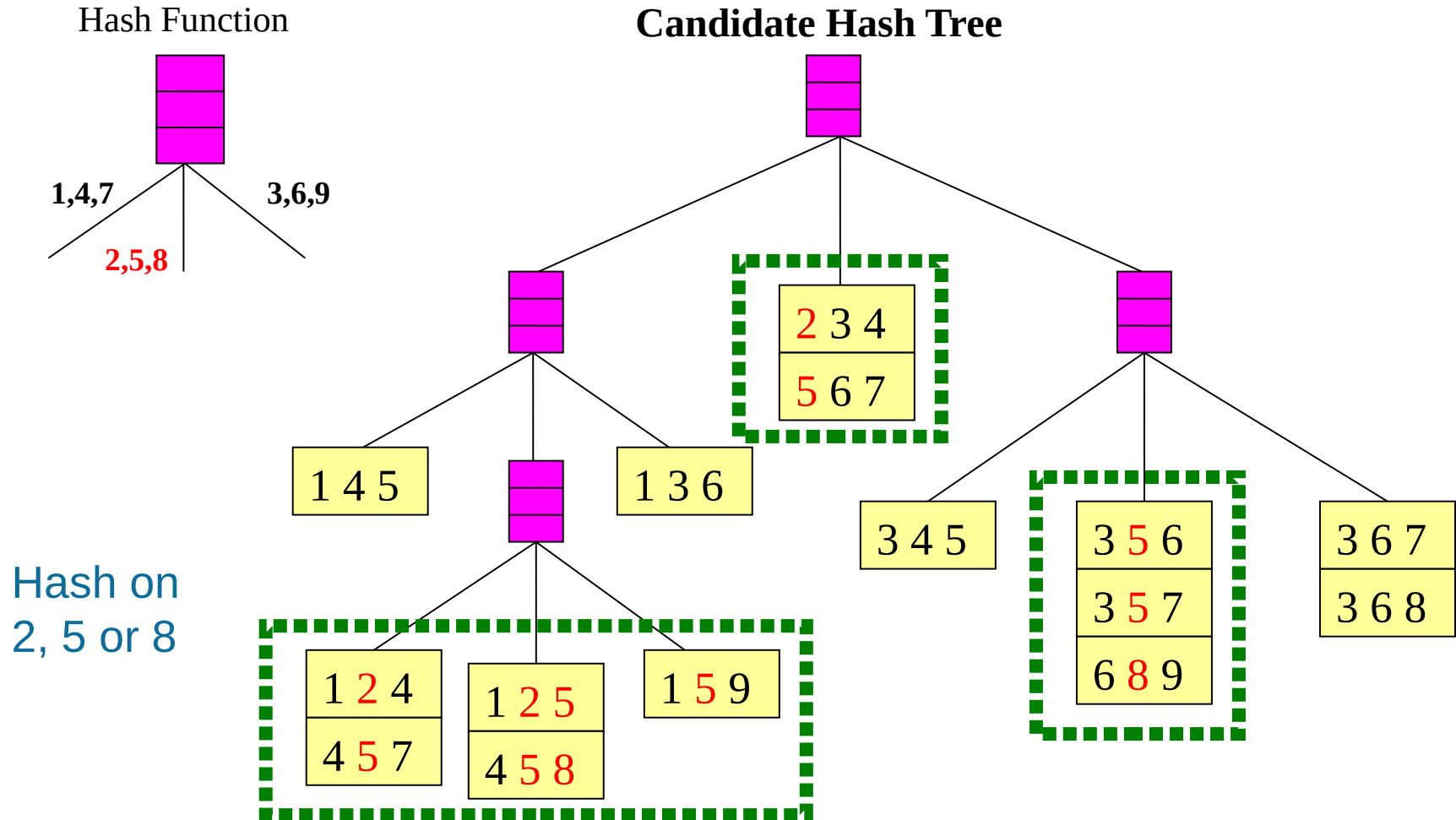


Important:
itemsets are sorted!

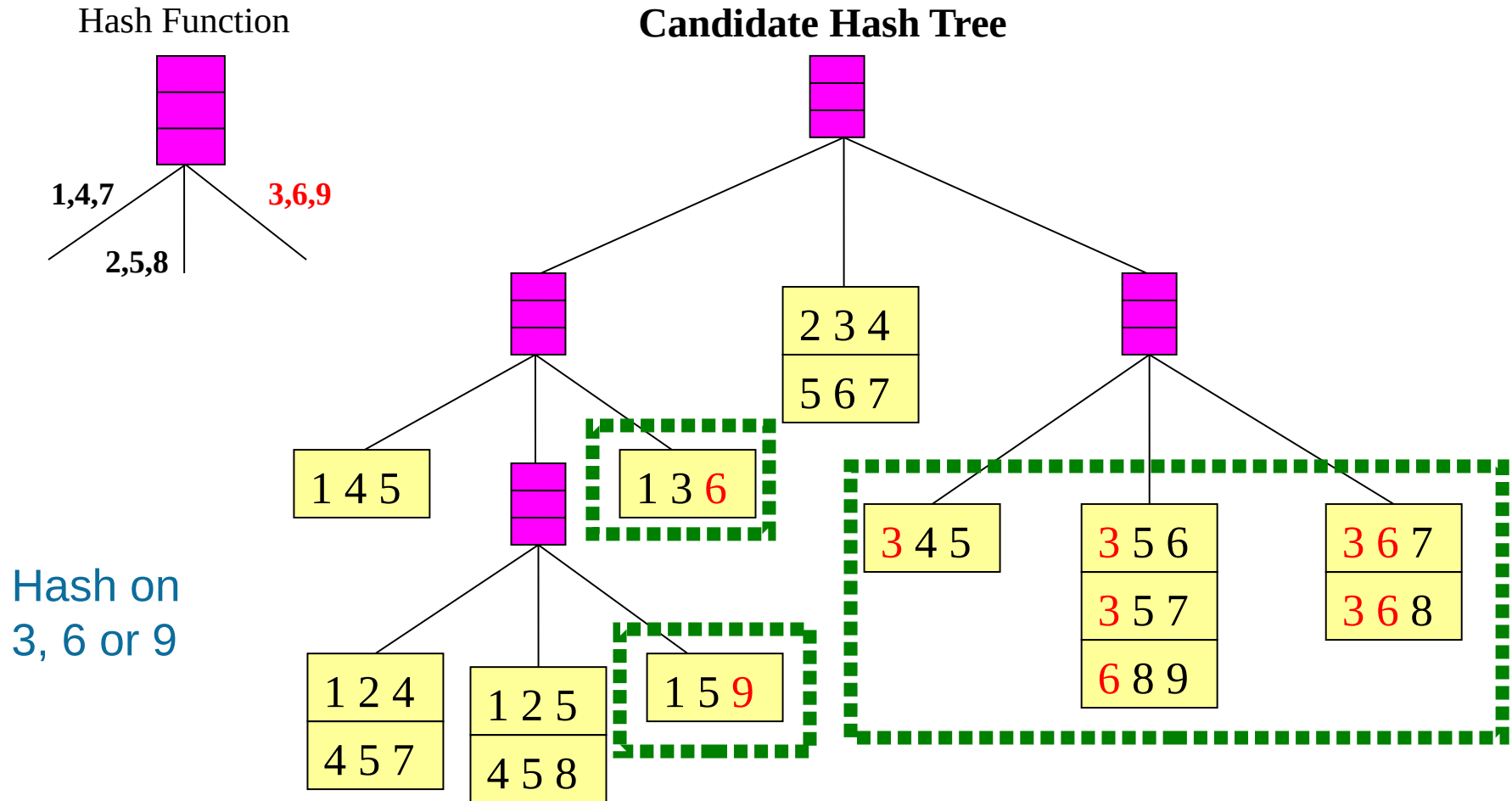
Example hash tree (cont.)



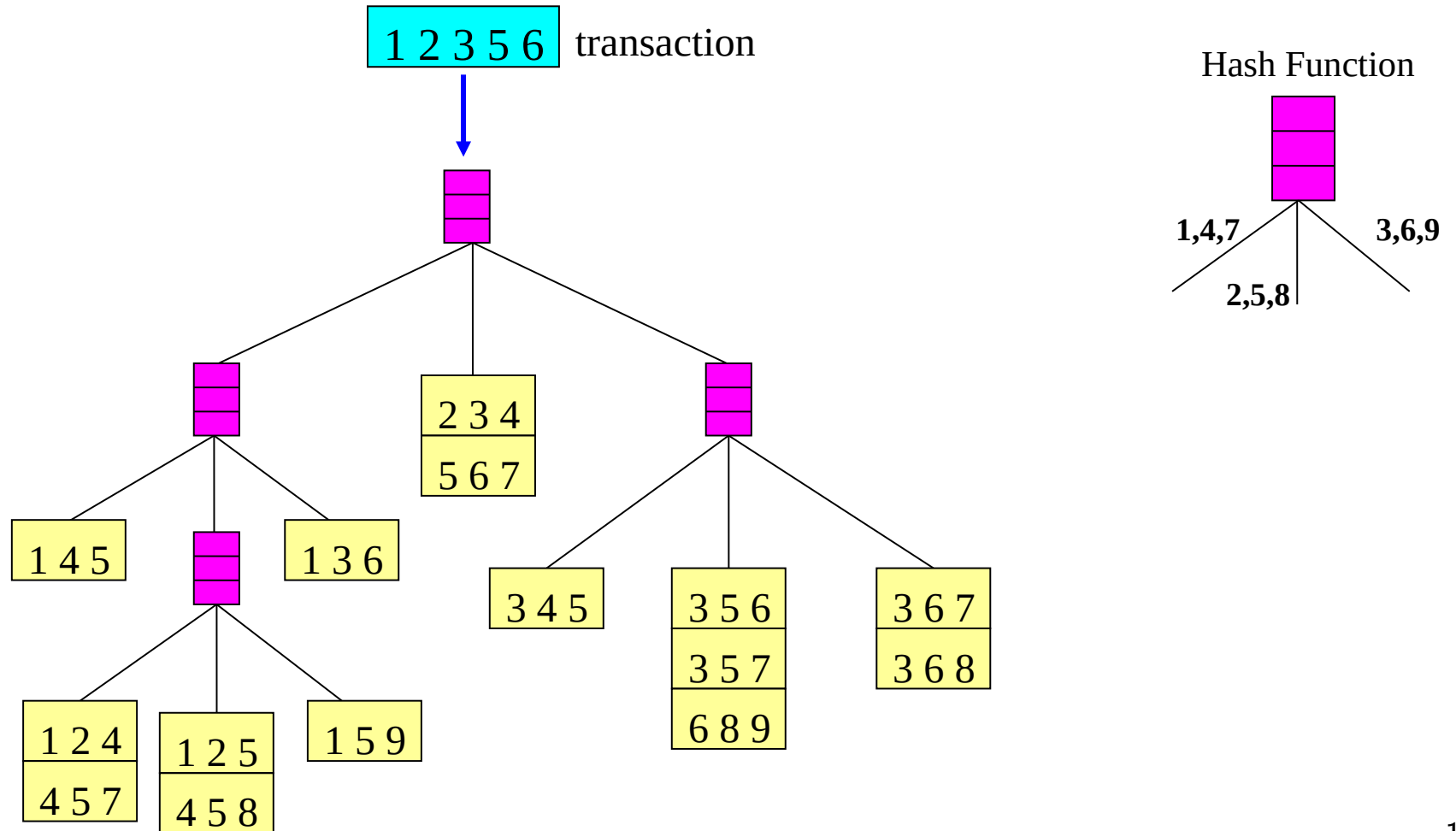
Example hash tree (cont.)



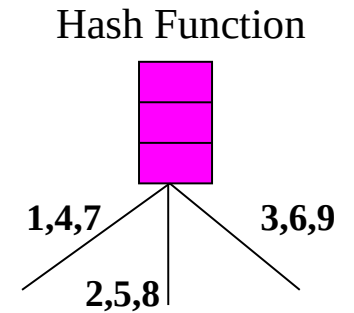
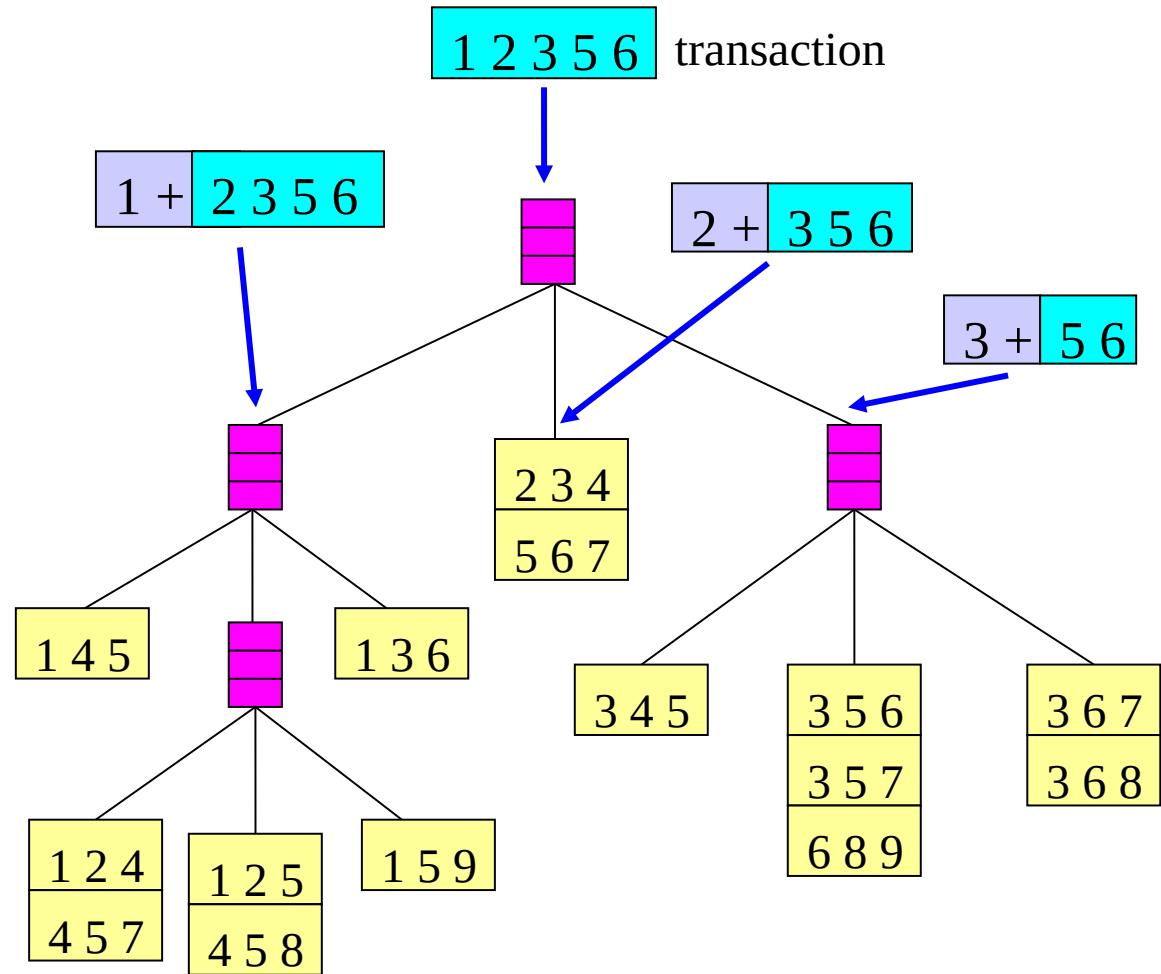
Example hash tree (cont.)



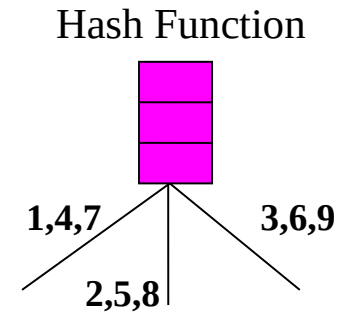
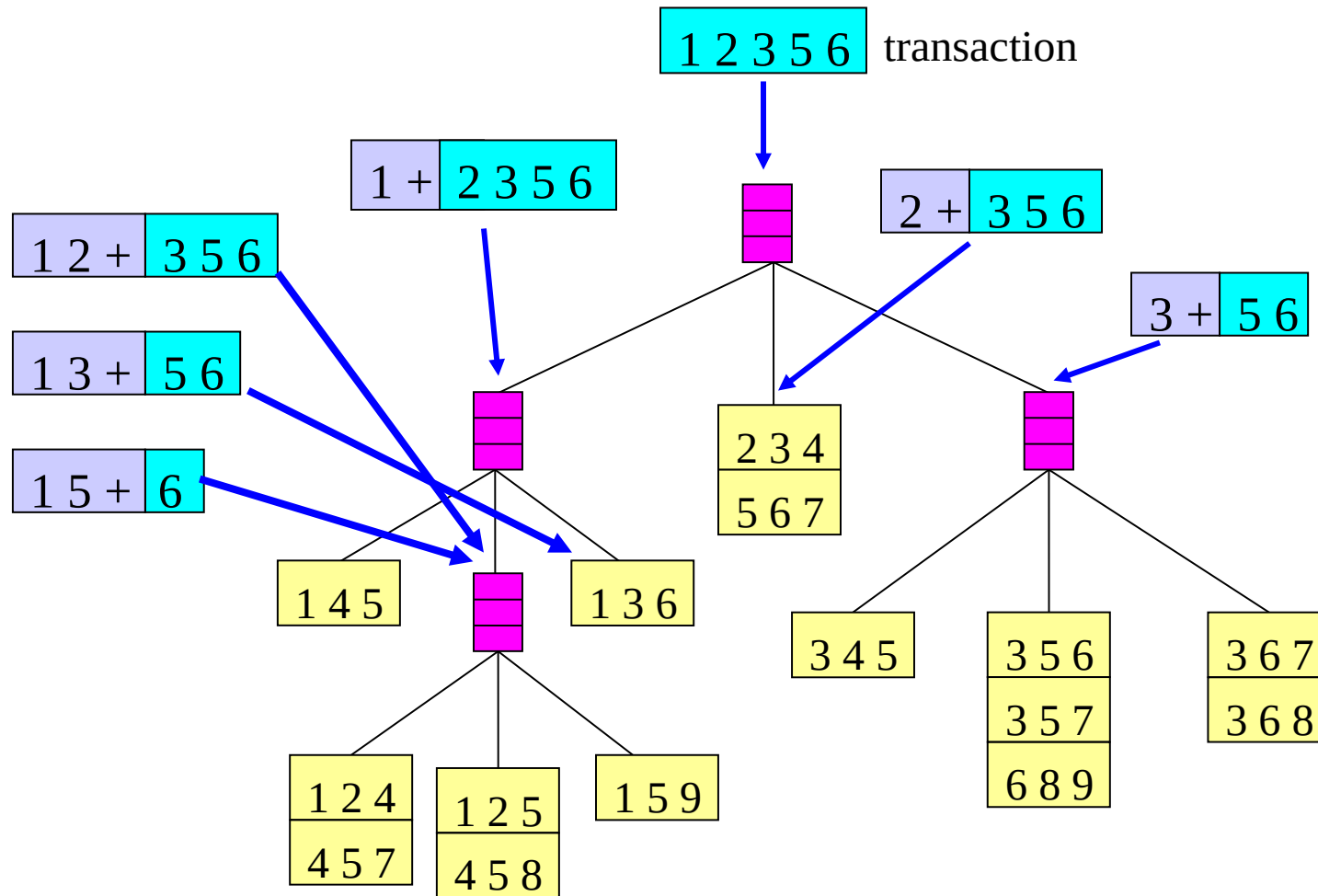
Checking which candidates might be in a transaction



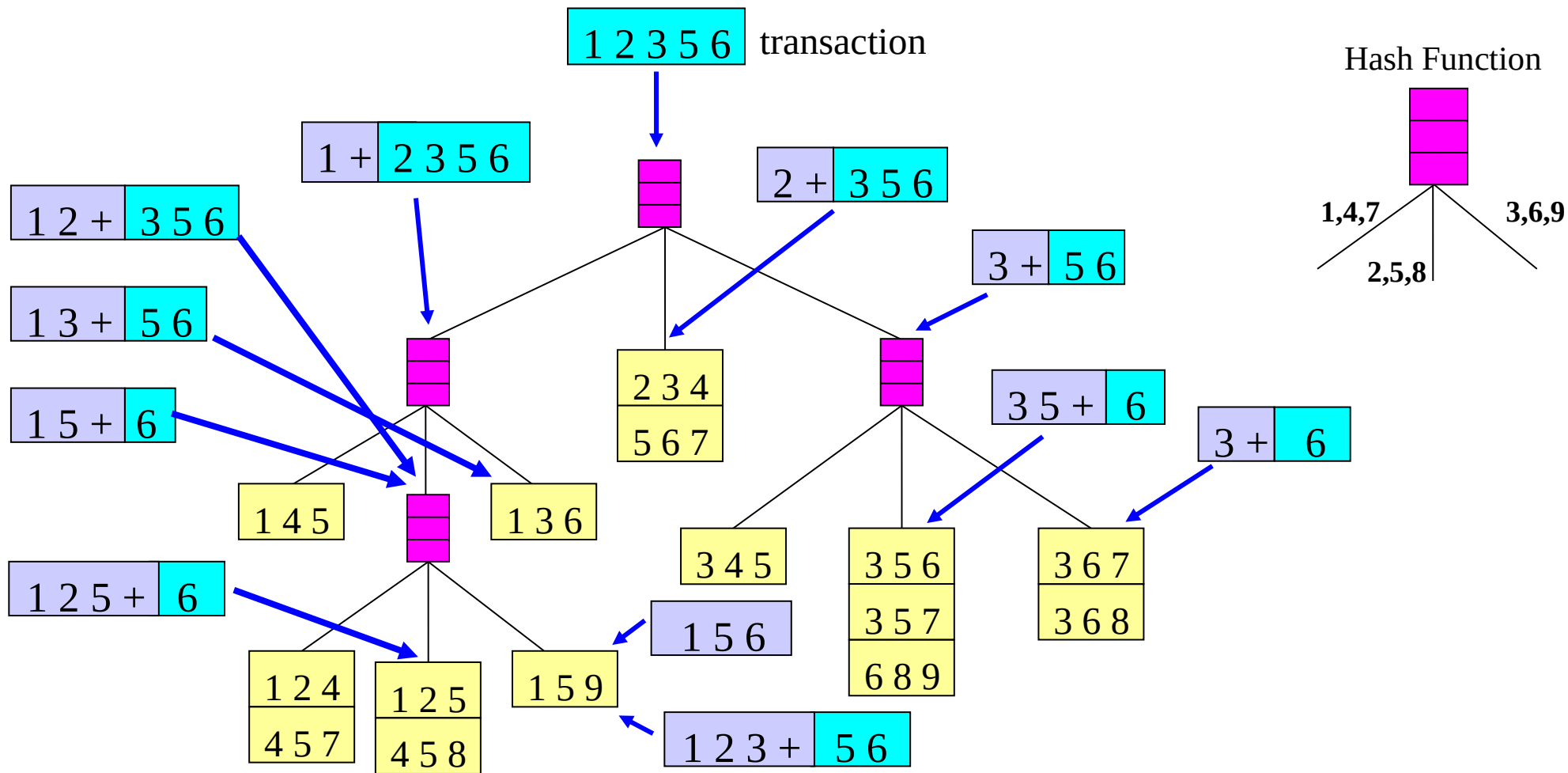
Checking which candidates might be in a transaction



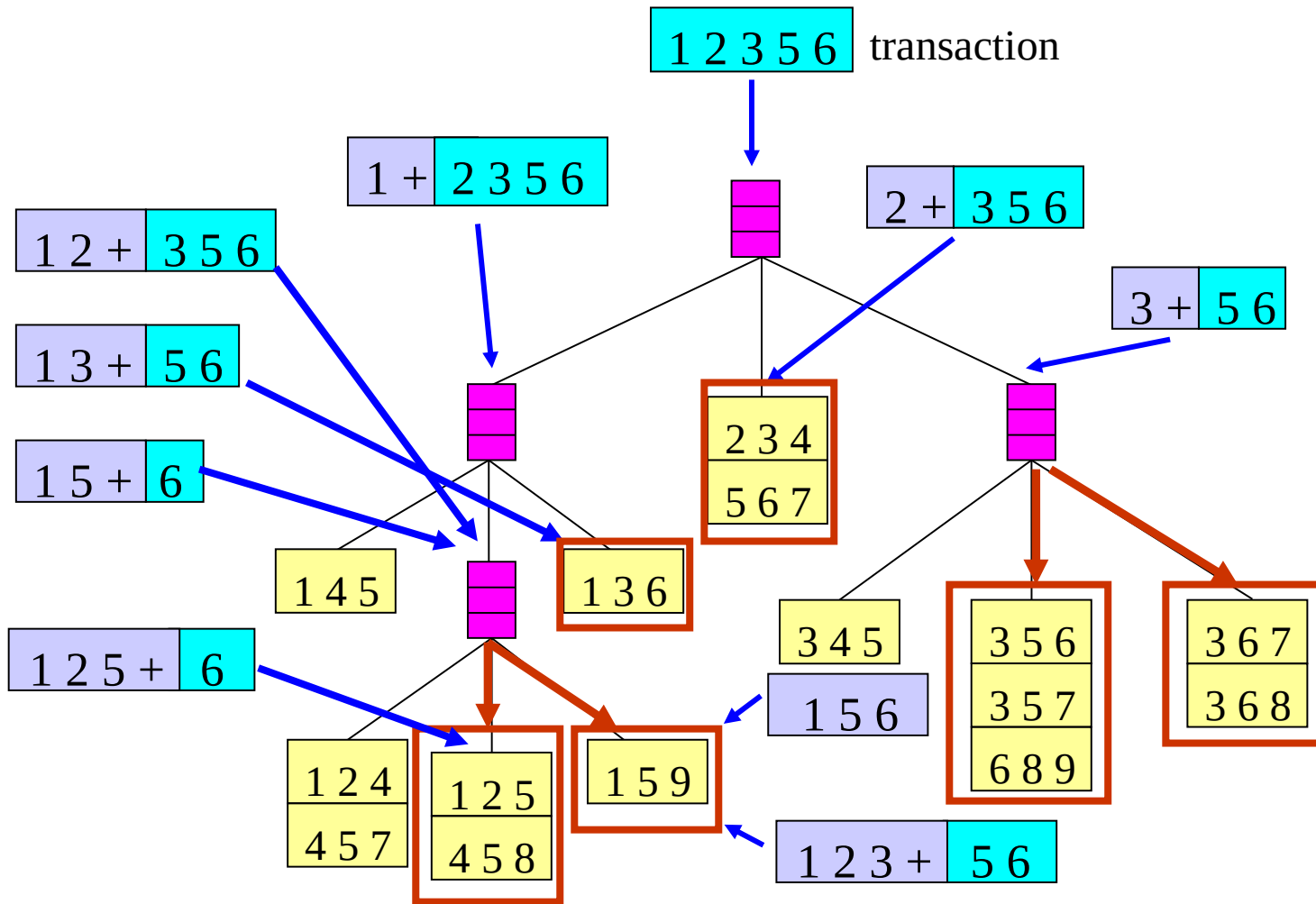
Checking which candidates might be in a transaction



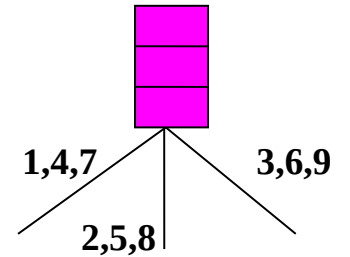
Checking which candidates might be in a transaction



Checking which candidates might be in a transaction

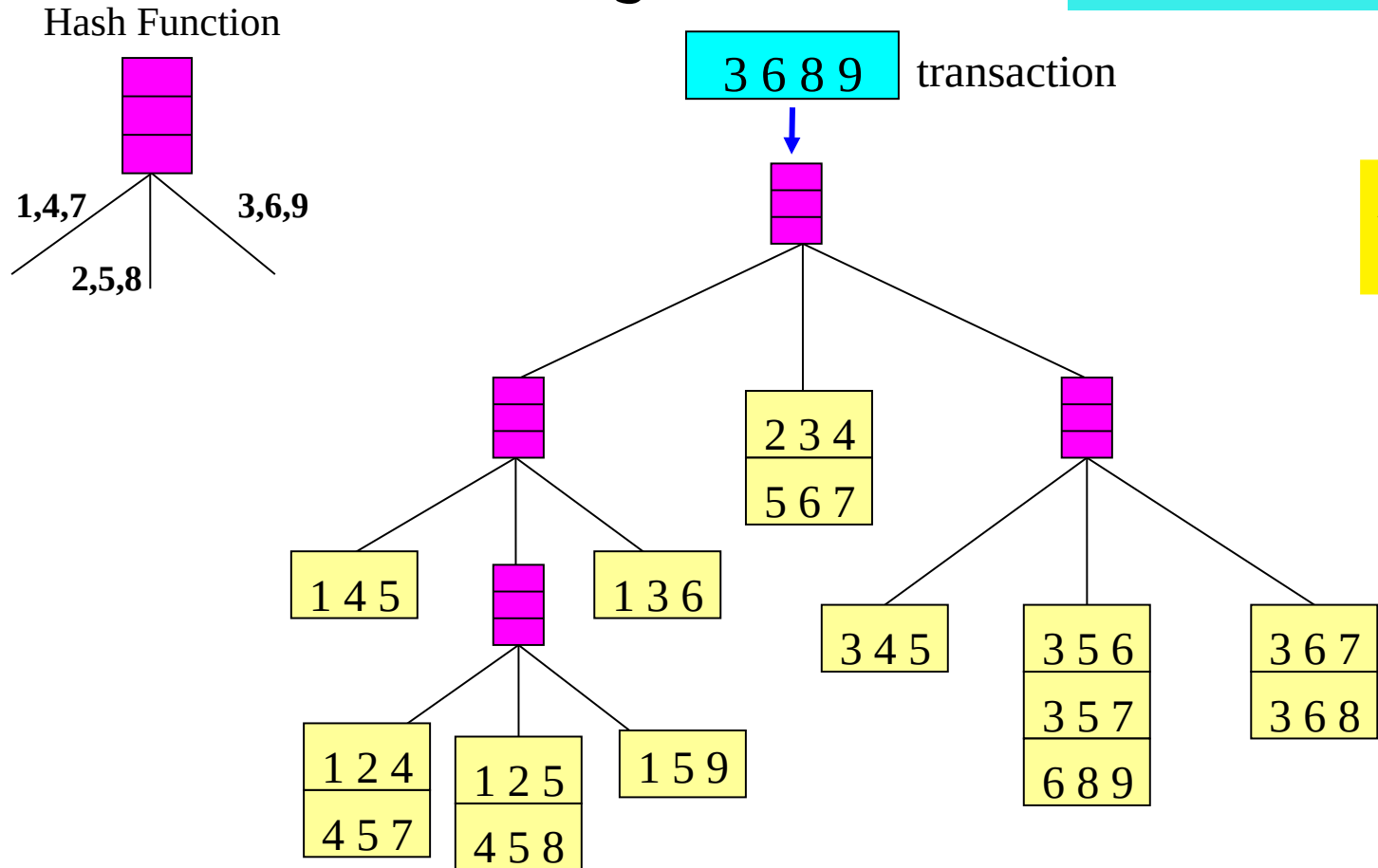


Hash Function



**Compare transaction
against 11 out of 15
candidates**

Exercise: Use the hash tree to determine which candidates might be in this transaction



Answer in
Nearpod Draw-it

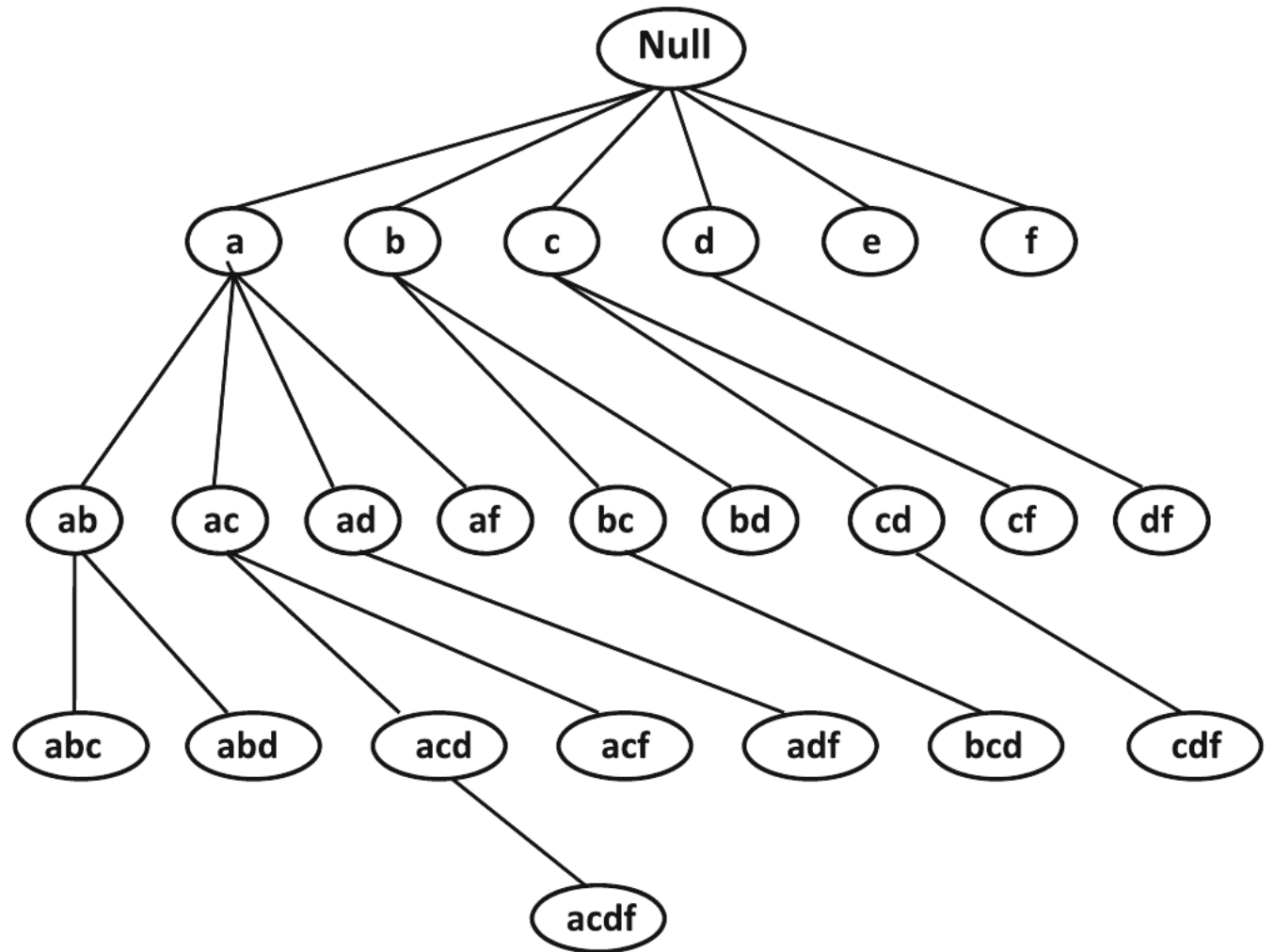
Enumeration-tree algorithms:

Lexicographic tree

- There is a node in the tree for each frequent itemset
- The root of the tree contains the null itemset
- If $I = \{i_1, i_2, \dots, i_k\}$ then the parent of I in the tree is $\{i_1, i_2, \dots, i_{k-1}\}$

Example

Note that, unlike the lattice, a parent can only be extended with an item that is lexicographically larger



Enumeration tree algorithm

Algorithm *GenericEnumerationTree*(Transactions: \mathcal{T} ,
Minimum Support: $minsup$)

begin
 Initialize enumeration tree \mathcal{ET} to single *Null* node;
 while any node in \mathcal{ET} has not been examined **do begin**
 Select one of more unexamined nodes \mathcal{P} from \mathcal{ET} for examination;
 Generate candidates extensions $C(P)$ of each node $P \in \mathcal{P}$;
 Determine frequent extensions $F(P) \subseteq C(P)$ for each $P \in \mathcal{P}$ with support counting;
 Extend each node $P \in \mathcal{P}$ in \mathcal{ET} with its frequent extensions in $F(P)$;
 end
 return enumeration tree \mathcal{ET} ;
end

Enumeration-tree-based implementation of Apriori

- Apriori constructs the enumeration tree in a breadth-first manner
- Apriori generates candidate $(k+1)$ -itemsets by merging two frequent k -itemsets of which the first $k-1$ items are the same \Rightarrow extension in the enumeration-tree

Summary

Things to remember

- Support and confidence on a rule
- Downward closure property
 - every subset of a frequent itemset is also frequent
 - hence, if an itemset X has a subset that is not frequent, X cannot be frequent
- Methods for candidate generation, pruning
- Algorithms for fast support computation

Exercises for this topic

- Data Mining, The Textbook (2015) by Charu Aggarwal
 - Exercises 4.9 → 9-10
- Mining of Massive Datasets 2nd edition (2014) by Leskovec et al.
 - Exercises 6.2.7 → 6.2.5 and 6.2.6
- Introduction to Data Mining 2nd edition (2019) by Tan et al.
 - Exercises 5.10 → 9-12