# Data Mining
# -- Association Rules

Instructor: Jen-Wei Huang

Office: 92528 in the EE building
jwhuang@mail.ncku

## FP-Growth [1]

- Mining frequent patterns without candidate generation
  - Depth-first search approach
- Grow long patterns from short ones using local frequent items only
  - “abc” is a frequent pattern
  - Get all transactions having “abc”, i.e., project DB on abc: DB|abc
  - “d” is a local frequent item in DB|abc → abcd is a frequent pattern

# Construct FP-tree

| *TID* | *Items bought* |
|---|---|
| **100** | ***{f, a, c, d, g, i, m, p}*** |
| **200** | ***{a, b, c, f, l, m, o}*** |
| **300** | ***{b, f, h, j, o, w}*** |
| **400** | ***{b, c, k, s, p}*** |
| **500** | ***{a, f, c, e, l, p, m, n}*** |

***min_support = 3***

1. Scan DB once, find frequent 1-itemset (single item pattern)
2. Sort frequent items in frequency descending order, f-list

**Header Table**

| *Item* | *frequency* | *head* |
|---|---|---|
| *f* | *4* | |
| *c* | *4* | |
| *a* | *3* | |
| *b* | *3* | |
| *m* | *3* | |
| *p* | *3* | |

F-list = f-c-a-b-m-p



# Construct FP-tree

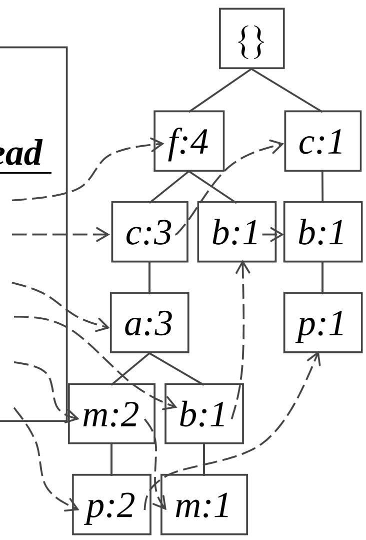
| *TID* | *Items bought* | *(ordered) frequent items* |
|---|---|---|
| **100** | ***{f, a, c, d, g, i, m, p}*** | ***{f, c, a, m, p}*** |
| **200** | ***{a, b, c, f, l, m, o}*** | ***{f, c, a, b, m}*** |
| **300** | ***{b, f, h, j, o, w}*** | ***{f, b}*** |
| **400** | ***{b, c, k, s, p}*** | ***{c, b, p}*** |
| **500** | ***{a, f, c, e, l, p, m, n}*** | ***{f, c, a, m, p}*** |

***min_support = 3***

3. Scan DB again, sort items in the transaction by frequency and construct FP-tree

**Header Table**

| *Item* | *frequency* | *head* |
|---|---|---|
| *f* | *4* | |
| *c* | *4* | |
| *a* | *3* | |
| *b* | *3* | |
| *m* | *3* | |
| *p* | *3* | |




F-list = f-c-a-b-m-p

# Partition Database

- Frequent patterns can be partitioned into subsets according to f-list
  - F-list = f-c-a-b-m-p
  - Patterns containing p
  - Patterns having m but no p
  - …
  - Patterns having c but no a nor b, m, p
  - Pattern f
- Completeness and non-redundancy

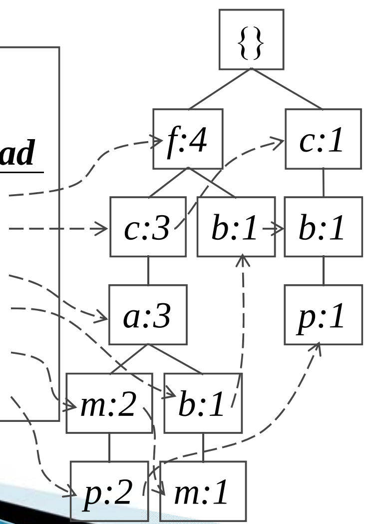


# Conditional Pattern Bases

- Starting at the least frequent item in the header table
- Traverse the FP-tree by following the link of each frequent item
- Accumulate all of *transformed prefix paths* of the item to form its conditional pattern base

**Header Table**

| *Item* | *frequency* | *head* |
|---|---|---|
| *f* | *4* | |
| *c* | *4* | |
| *a* | *3* | |
| *b* | *3* | |
| *m* | *3* | |
| *p* | *3* | |

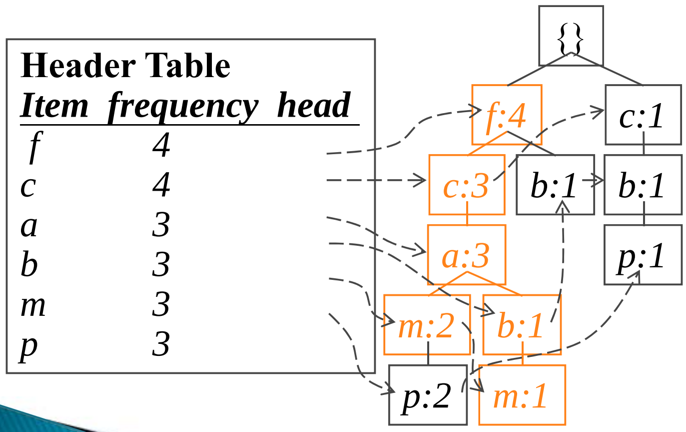



***Conditional* pattern bases**

| *item* | *cond. pattern base* |
|---|---|
| **c** | ***f:3*** |
| **a** | ***fc:3*** |
| **b** | ***fca:1, f:1, c:1*** |
| **m** | ***fca:2, fcab:1*** |
| **p** | ***fcam:2, cb:1*** |

# Conditional FP-trees

- For each pattern-base
  - Accumulate the count for each item in the base
  - Construct the FP-tree for the frequent items of the pattern base




***m-conditional*** **pattern base:**
***fca:2, fcab:1***

➔

{}
|
*f*:3 ➔
|
*c*:3
|
*a*:3

***m-conditional*** **FP-tree**

**All frequent patterns relate to *m***

***m,***
***fm, cm, am,***
***fcm, fam, cam,***
***fcam***



# Mining Conditional FP-tree

{}
|
*f*:3
|
*c*:3
|
*a*:3

***m-conditional*** **FP-tree**

Cond. pattern base of "am": (fc:3)

{}
|
*f*:3
|
*c*:3

***am-conditional*** **FP-tree**

Cond. pattern base of "cm": (f:3)

{}
|
*f*:3

***cm-conditional*** **FP-tree**

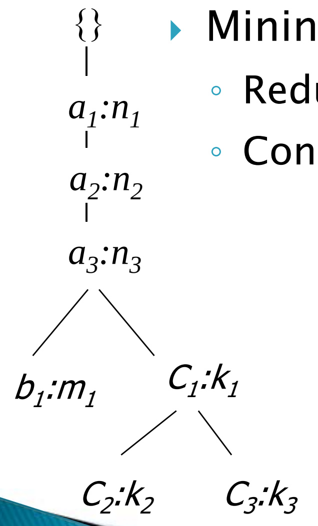
Cond. pattern base of "cam": (f:3)

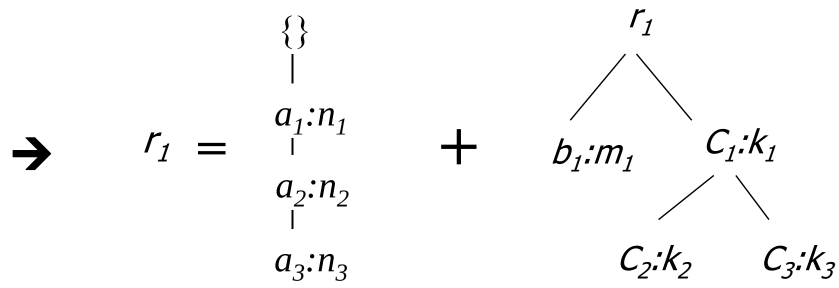
{}
|
*f*:3

***cam-conditional*** **FP-tree**

# Single Prefix Path

- Suppose a (conditional) FP-tree T has a shared single prefix-path P
- Mining can be decomposed into two parts
  - Reduction of the single prefix path into one node
  - Concatenation of the mining results of the two parts






# Benefits of FP-tree

- Completeness
  - Preserve complete information for frequent pattern mining
  - Never break a long pattern of any transaction
- Compactness
  - Reduce irrelevant info—infrequent items are gone
  - Items in frequency descending order: the more frequently occurring, the more likely to be shared
  - Never be larger than the original database (not count node-links and the *count* field)

# FP-Growth Algorithm

- Idea: Frequent pattern growth
  - Recursively grow frequent patterns by pattern and database partition
- Method
  - For each frequent item, construct its conditional pattern-base, and then its conditional FP-tree
  - Repeat the process on each newly created conditional FP-tree
  - Until the resulting FP-tree is empty, or it contains only one path—single path will generate all the combinations of its sub-paths, each of which is a frequent pattern



# Problems of FP-Growth

- What about if FP-tree cannot fit in memory?
  - DB projection

- First partition a database into a set of projected DBs
- Then construct and mine FP-tree for each projected DB

# ECLAT [3]

- Mining by exploring vertical data format
- Vertical format: t(AB) = $\{T_{11}, T_{25}, \ldots\}$
  - tid-list: list of trans.-ids containing an itemset
- Deriving frequent patterns based on vertical intersections
- Using diffset to accelerate mining
  - Only keep track of differences of tids
  - $t(X) = \{T_1, T_2, T_3\}$, $t(XY) = \{T_1, T_3\}$
  - Diffset $(XY, X) = \{T_2\}$



# Basic Extensions

- Max-pattern [5]
  - R. J. Bayardo. Efficiently mining long patterns from databases. SIGMOD'98.
- Closed-pattern [6]
  - N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. ICDT'99.
- Sequential pattern [7]
  - R. Agrawal and R. Srikant. Mining sequential patterns. ICDE'95

# Closed Patterns and Max-Patterns

- A long pattern contains a combinatorial number of sub-patterns, e.g., $\{a_1, \ldots, a_{100}\}$ contains $({}_{100}{}^{1}) + ({}_{100}{}^{2}) + \ldots + ({}_{1}{}^{1}{}_{0}{}^{0}{}_{0}{}^{0}) = 2^{100} - 1 = 1.27*10^{30}$ sub-patterns!
- Solution: *Mine closed patterns and max-patterns instead*
- An itemset X is closed if X is *frequent* and there exists *no super-pattern* Y ɔ X, *with the same support* as X
  - Closed pattern is a lossless compression of freq. patterns
  - Reducing the # of patterns and rules
- An itemset X is a max-pattern if X is frequent and there exists no frequent super-pattern Y ɔ X



# Examples

- Exercise. DB = $\{<a_1, \ldots, a_{100}>, < a_1, \ldots, a_{50}>\}$
  - Min_sup = 1.
- What is the set of closed itemset?
  - $<a_1, \ldots, a_{100}>$: 1
  - $< a_1, \ldots, a_{50}>$: 2
- What is the set of max-pattern?
  - $<a_1, \ldots, a_{100}>$: 1
- What is the set of all patterns?
  - !!

# Computational Complexity

- How many itemsets are potentially to be generated in the worst case?
  - The number of frequent itemsets to be generated is senstive to the min_sup threshold
  - When min_sup is low, there exist potentially an exponential number of frequent itemsets
  - The worst case: $M^N$ where M: # distinct items, and N: max length of transactions
- The worst case complexity vs. the expected probability
  - Ex. Suppose Walmart has $10^4$ kinds of products
    - The chance to pick up one product $10^{-4}$
    - The chance to pick up a particular set of 10 products: $\sim 10^{-40}$
    - What is the chance this particular set of 10 products to be frequent $10^3$ times in $10^9$ transactions?



# References


- [1] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. SIGMOD' 00
- [2] G. Grahne and J. Zhu, Efficiently Using Prefix-Trees in Mining Frequent Itemsets, Proc. ICDM'03 Int. Workshop on Frequent Itemset Mining Implementations (FIMI'03), Melbourne, FL, Nov. 2003
- [3] M. J. Zaki, Scalable Algorithms for Association Mining. IEEE Transactions on Knowledge and Data Engineering, 12(3):372-390. May/June 2000
- [4] M. J. Zaki and Karam Gouda, Fast Vertical Mining Using Diffsets. In 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. August 2003.
- [5] R. J. Bayardo. Efficiently mining long patterns from databases. SIGMOD'98.
- [6] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. ICDT'99.
- [7] R. Agrawal and R. Srikant. Mining sequential patterns. ICDE'95

# References

- Slides from Prof. J.-W. Han, UIUC
- Slides from Prof. M.-S. Chen, NTU
- Slides from Prof. W.-Z. Peng, NCTU