IKN Special Lecture / 情報知識ネットワーク特論 Data Mining 3
Frequent Itemset Mining: DFS (Depth-first search) algorithms (Eclat and LCM algorithms)

Hiroki Arimura, Takuya Kida

Graduate School of Info. Sci. and Tech, Hokkaido University
email: {arim,kida}@ist.hokudai.ac.jp

Slide PDF: http://www-ikn.ist.hokudai.ac.jp/ikn-tokuron/
Oct 2014

今日の内容

Lecture 3: Frequent Itemset Mining: DFS (Depth-first search) algorithms

- Depth-first search algorithms
  - Backtrack algorithm
  - Algorithms: Eclat, FP-growth, LCM

Points

- Introduction to enumeration algorithms
- Set enumeration tree
- DFS by backtracking

Backgrounds

Frequent Itemset Mining

- Finding all "frequent" sets of elements (items) appearing no more than σ times in a given transaction data base.
- Introduced by Agrawal and Srikant [VLDB'94]
- One of the most popular data mining problem
- Basis for more complicated / sophisticated data mining problems

Definitions: Database

- A set $\Sigma = \{ 1, ..., n \}$ of items (elements)
- Transaction database
  - A set $T = \{ t_1, ..., t_m \}$ of subsets of Σ
  - Each subset $t \subseteq \Sigma$ is called a tuple (record)

Alphabet of items

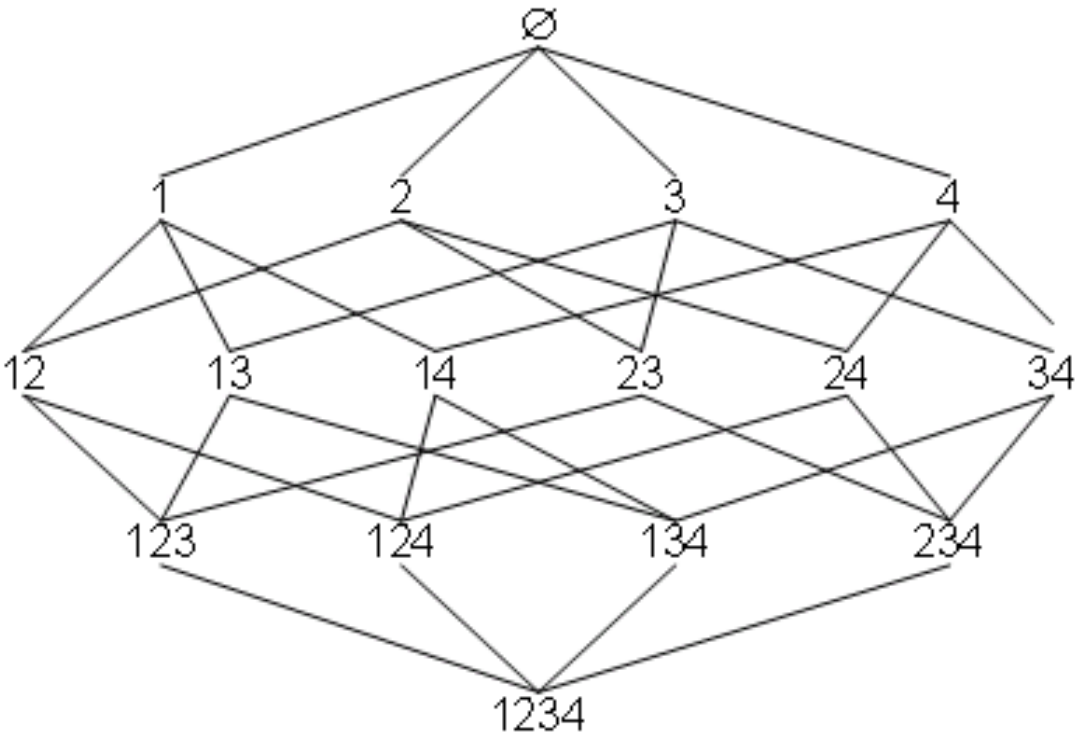
$I = \{1,2,3,4\}$

Transaction database

id	tuples
1	1, 3
2	2, 4
3	1, 2, 3, 4
4	1, 2, 4

Definitions: Itemset lattice

- Item set: any subset $X \subseteq \boldsymbol{\Sigma} = \{ 1, \ldots, n \}$
- (Item) set lattice $\mathcal{L} = (2^{\boldsymbol{\Sigma}}, \subseteq)$
  - The power set $2^{\boldsymbol{\Sigma}} = \{ X : X \subseteq \boldsymbol{\Sigma} \}$
  - The subset relation $\subseteq$ over $2^{\boldsymbol{\Sigma}}$



Example:
The set lattice
for$\Sigma = \{1,2,3,4\}$

Definitions: Frequent sets

- An itemset X appears in a tuple t: $X \subseteq t$
- The occurrence of X in a database ***T***:
  $Occ(X, \boldsymbol{T}) = \{ t \in \boldsymbol{T} : X \subseteq t \}$
- The frequency of X: $Fr(X, T) = |\, Occ(X, T) \,|$
- Minimum support (minsup): an integer $0 \leqq \sigma \leqq |\boldsymbol{T}|$
- X is σ-frequent (frequent) in T if $Fr(X, T) \geqq \sigma$.

Alphabet of items

$I = \{A,B,C,D\}$

Occurrences and frequences of itemsets

Occ(**3**, T) = {1, 3}
Fr(**3**, T) = 2

Occ(**24**, T) = {2, 3, 4},
Fr(**24**, T) = 3

Transaction database

id	tuples
1	1, 3
2	2, 4
3	1, 2, 3, 4
4	1, 2, 4

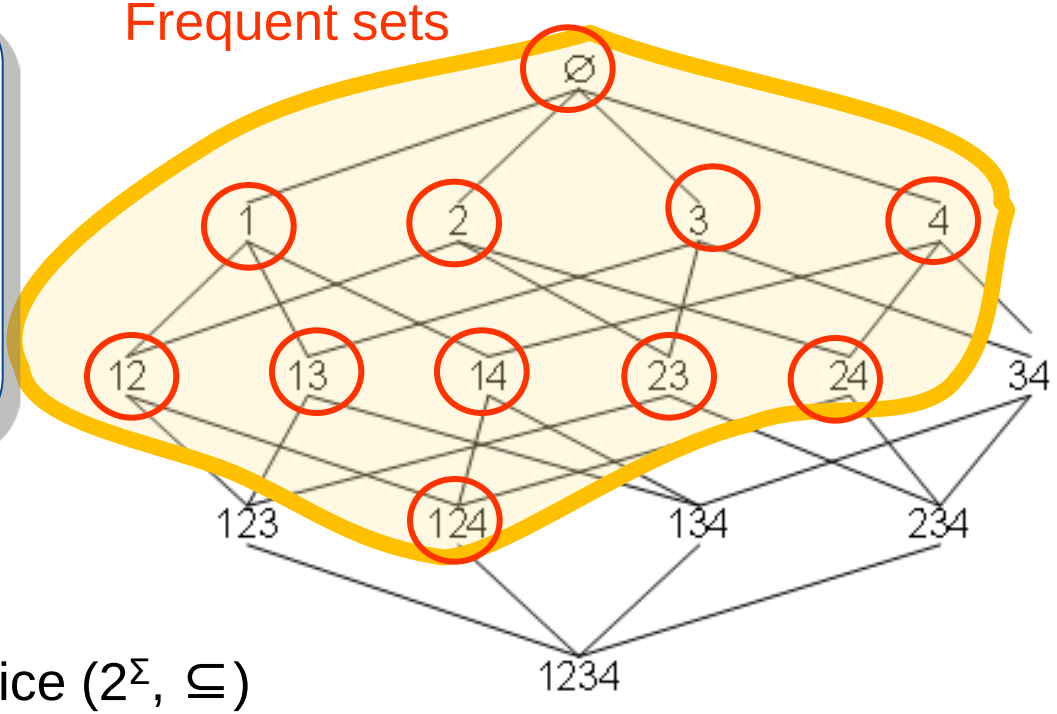
Definitions: Frequent sets

- The occurrence of X in a database ***T***:
  $Occ(X, \boldsymbol{T}) = \{ t \in \boldsymbol{T} : X \subseteq t \}$
- X is σ-frequent (frequent) in T if $Fr(X, T) = |\, Occ(X, T) \,| \geqq \sigma$.

minsupσ= **2**

Frequent sets
∅,
1, 2, 3, 4,
12, 13, 14,
23, 24, 124

Frequent sets



	1	2	3	4	5
t1	○		○		
t2		○		○	
t3	○	○	○	○	
t4		○	○		○
t5	○	○		○	

database

The itemset lattice $(2^{\Sigma}, \subseteq)$

Definitions: Problem

Frequent Itemset Mining Problem

- Given: A transaction database $\boldsymbol{T}$ and a non-negative integer $0 \leqq \sigma \leqq |\boldsymbol{T}|$
- Task: Enumerate **all "frequent" itemsets X** in $\boldsymbol{T}$ that have frequency at least σ ($Fr(\mathbf{X}) \geqq \sigma$)

- $\mathscr{F}$: the class of all σ-frequent itemsets
- The number $|\mathscr{F}|$ of solutions is exponential in the number $\boldsymbol{n}$ of items.
- a typical enumeration problem.

機械学習アルゴリズムへの応用

ブースティング (Boosting) [Freund, Shapire 1996]

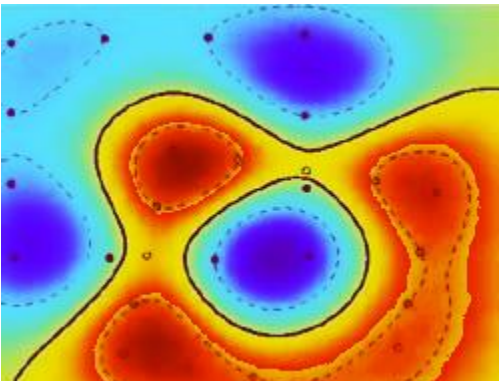
- 多数の機械学習アルゴリズムを統合して高精度予測
- オンライン予測の理論と深い関連

SVM (Support Vector Machines) [Vapnik 1996]

- マージン最大化による現在の state-of-the-art methods
- カーネル法を用いた高次元空間と多様なデータへの拡張

これらの学習アルゴリズムと，重みつきアイテム集合マイニング
を組み合わせて利用できる

キーワード: Boosting, SVM, オンライン予測, ニューラルネット, 計算学習理論
国際会議: NIPS, ICML, COLT, ALT



- V. Vapnik, Statistical Learning Theory, Wiley, 1998. (textbook)
- N. Cristianini and J. Shawe-Taylor, An introduction to support vector machines and other kernel-based learning methods, Cambridge, 2000. (textbook)
- Y. Freund and R. E. Schapire, A decisiontheoretic generalization of on-line learning and an application to boosting, JCSS, 55, 119-139, 1997. (AdaBoost)
- 金森, 畑埜, 渡辺,ブースティング: 学習アルゴリズムの設計技法, 森北出版 (text book)

Computational Complexity of Data Mining Algorithms

How to model efficient data mining algorithms?

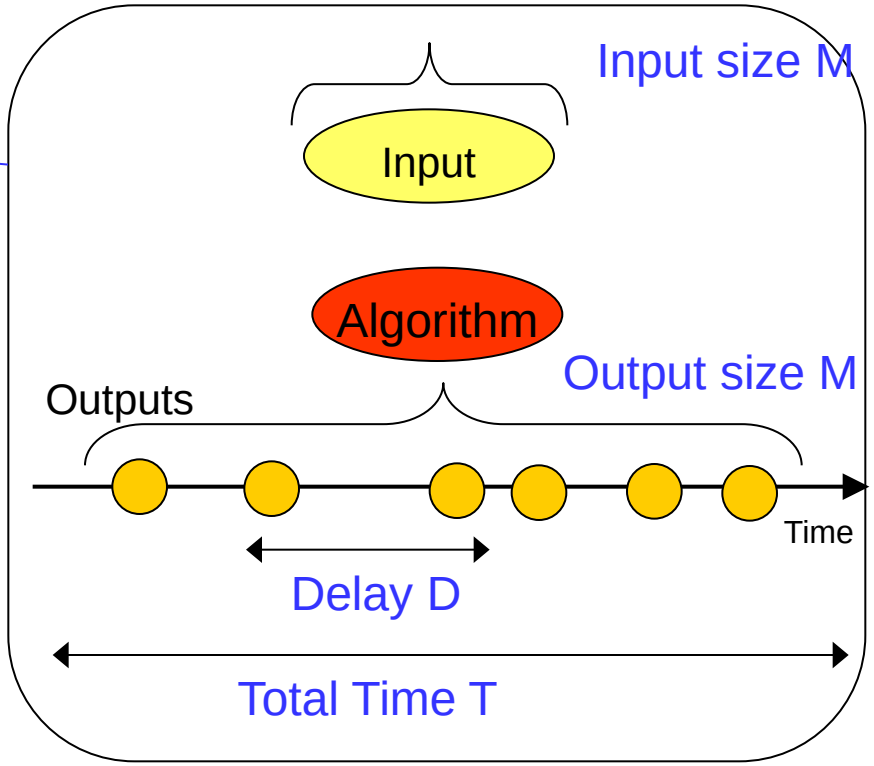
- **Light-weight**
- **High-throughput**

Computational Complexity of Data Mining Algorithms

Modeling data mining as enumeration

- **Idea: Mesure the computation time per solution**

- **Output-polynomial (OUT-POLY)**
  - Total time = poly(N, M)
- **polynomial-time enumeration, or amortized polynomial-delay (POLY-ENUM)**
  - Amotized delay is poly(Input), or
  - Total time = M·poly(N)
- **polynomial-delay (POLY-DELAY)**
  - Maximum of delay is poly(Input)



\+ **polynomial-space (POLY-SPACE)**

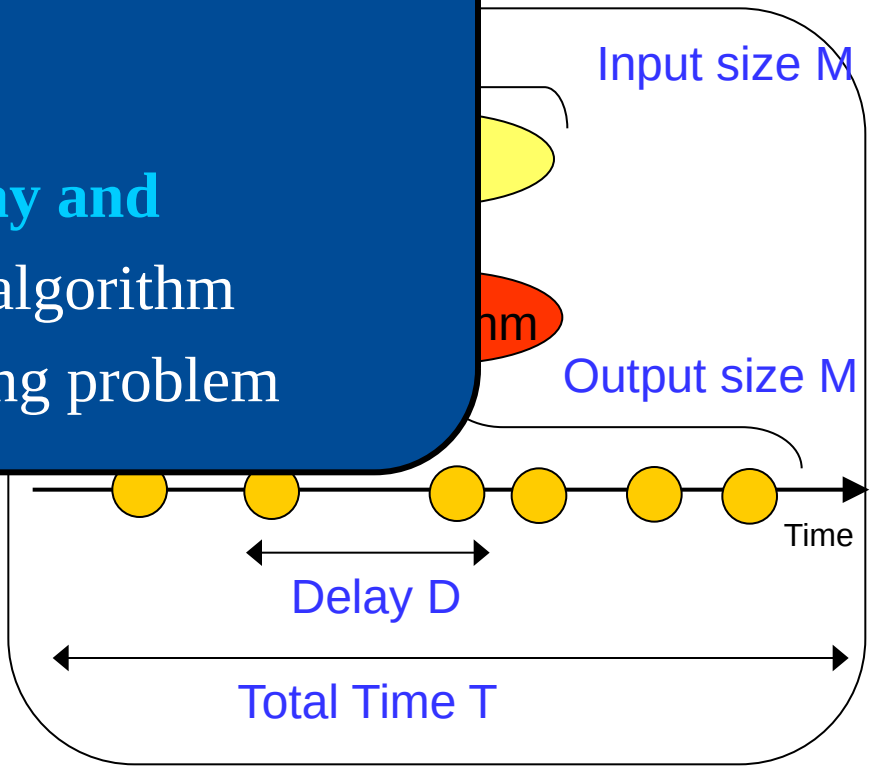
Modeling data mining as enumeration

- **Idea: Mesure the computation time per solution**

Ultimate Goal:

To design
polynomial delay and
polynomial space algorithm
for a given data mining problem

- **Output-p**
  - Total t
- **polynom**
  (POLY-E
  - Amotized
  - Total time i utput·poly(Input)



- **polynomial-delay (POLY-DELAY)**
  - Maximum of delay is poly(Input)

+ polynomial-space (POLY-SPACE)

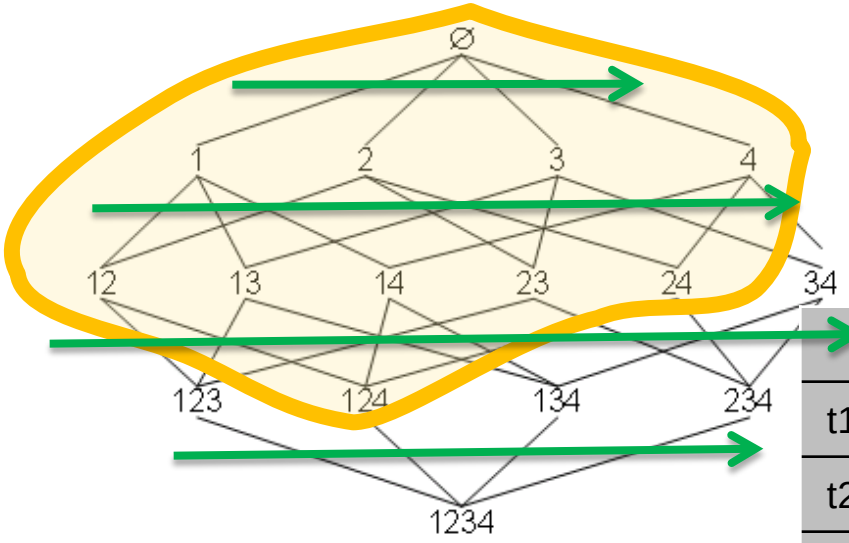
Frequent Itemset Mining Algorithms

DFS algorithm (Backtracking algorithm)

Two approaches to Frequent sets mining

Apriori algorithm [1994]

- Breadth-first search (BFS)
- Horizontal data layout



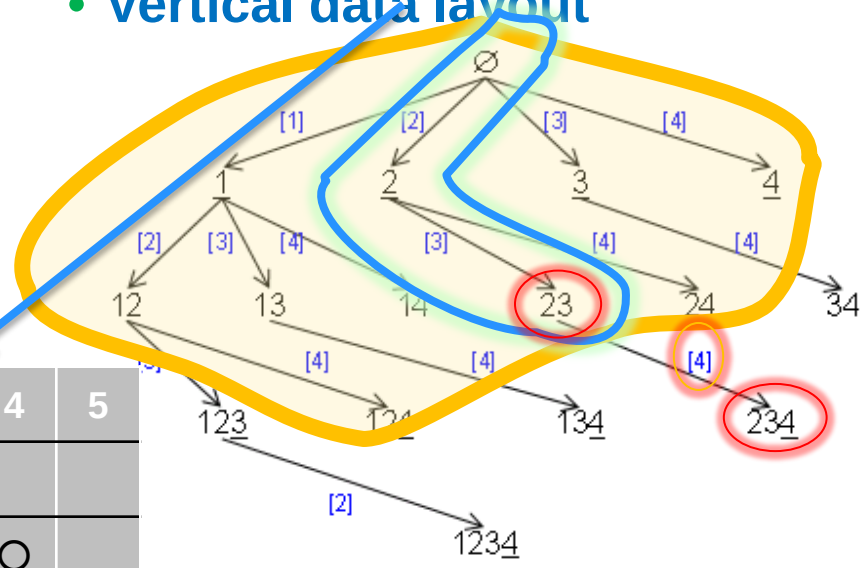
- 1st generation
- External memory algorithm

	1	2	3	4	5
t1	○		○		
t2		○		○	
t3	○	○	○	○	
t4		○	○		○
t5	○	○		○	

database

Backtrack algorithm [1997-1998]

- Depth-first search (DFS)
- Vertical data layout



- 2nd generation
- In-core algorithm
- Space efficient

DFS(Depth-first search) algorithm

Backtracking Algorithm

- The 2nd generation frequent itemset miners
- Eclat [Zaki et al., KDD'97], [Morishita DS'98], [Bayardo SIGMOD'98]

DFS (Depth-first search)

- **The set enumeration tree** $\mathcal{T} = (\mathcal{F}, E_{pa}, \phi)$ for the class of frequent sets
  - Starting from ϕ, search $\mathcal{F}$ from smaller to larger
- **Pattern growth approach**:
  - Grow each itemset by attaching a new item*, the tail element*.
- **Implementation**: The search structure is implicitly implemented by a recursive procedure.

Vertical Data Layout

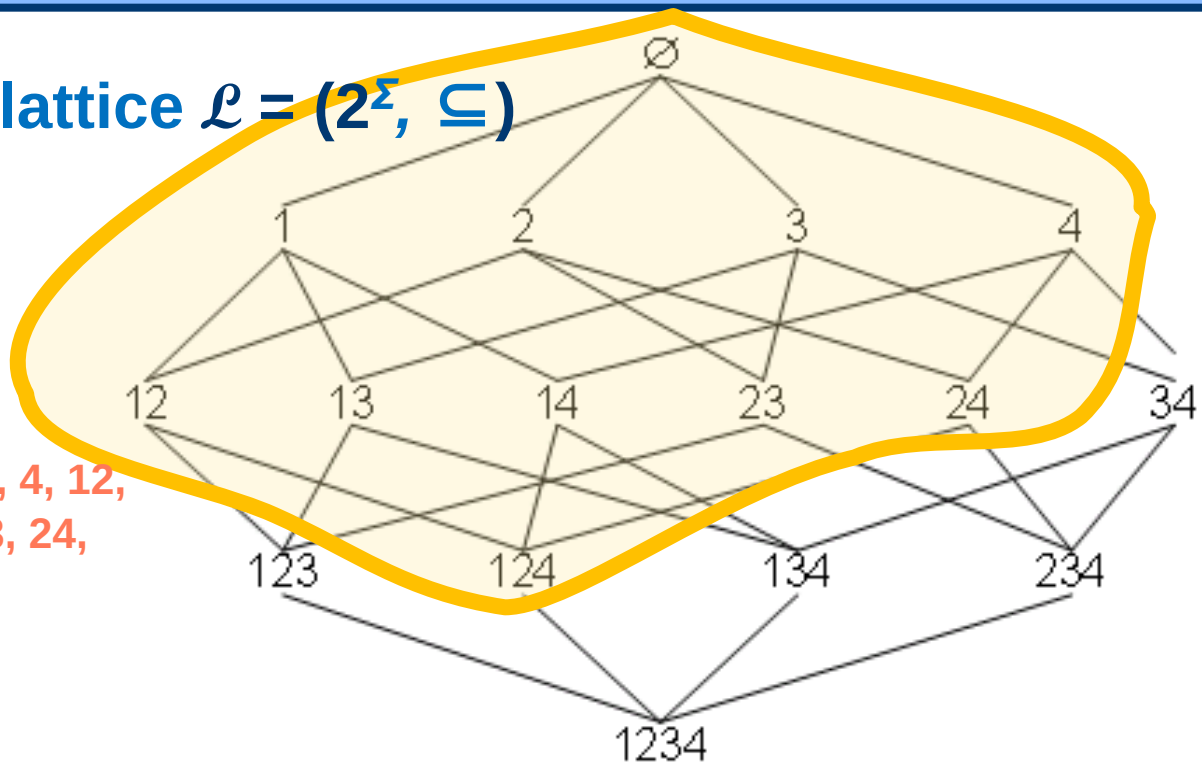
- For each enumerated itemset X, maintain its occurrence list Occ(X) to compute the frequencies.
- Incremental update of Occ(X) is possible: "*Downward closure*"

DFS(Depth-first search) algorithm

DFS algorithm

- How to generate all subsets $X \subseteq \Sigma$ in depth-first search?
- Use enumeration of subsets! (岡本先生の講義の「部分集合列挙」)

The set lattice $\mathcal{L} = (2^{\Sigma}, \subseteq)$



The set lattice for
Σ = {1,2,3,4}

σ = 2

$\mathcal{F}$ = { ∅, 1, 2, 3, 4, 12, 13, 14, 23, 24, 124 }

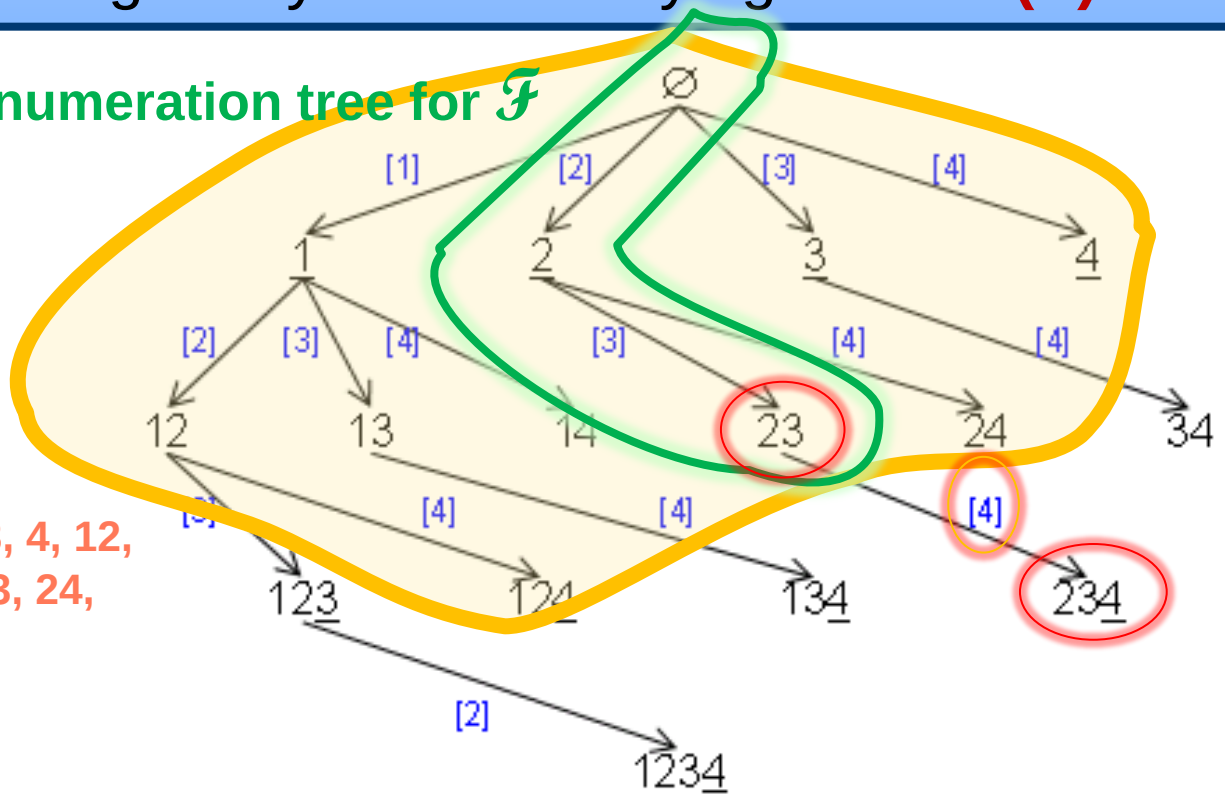
演習

- 問題 1: アイテムの集合Σ = {1, ..., n}が与えられたときに, そのすべての部分集合を出力するプログラムを書け
- (1) Σ = {1,2,3,4}に対して, プログラムの解をかけ.
- (2) プログラムを与えよ. 好きな言語でよい(日本語, English, 擬似言語, C, C++, Pascal, Java, ruby, python, LISP, Prolog, etc. etc.)
- (3: easy) Σ = {1,..., n}から3個のアイテムを組み合わせて作られる集合をすべて出力するプログラムをかけ.
- Problem 1: Give a computer program that receives a set {1,...,n} of n items (elements) and enumerates all of its subsets $X \subseteq \Sigma$ without repetition.

The set enumeration tree (the *family tree*)

- A spanning tree for all frequent itemsets
- Assign the unique parent **Pa(Y) = Y – { max(Y) } =: X** to each non-empty frequent set **Y.**
- Given a parent X, we can generate its children **Y = X ∪ { i }** by attaching every item ***i*** satisfying ***i* > max(X).**

The set enumeration tree for $\mathcal{F}$



The set lattice for
Σ = {1,2,3,4}

σ = 2

$\mathcal{F}$ = { ∅, 1, 2, 3, 4, 12, 13, 14, 23, 24, 124 }

DFS algorithm for Frequent Itemset Mining

Algorithm Backtrack

- BacktrackExpand(∅, Occ(∅), 0, n).

procedure BacktrackExpand(X, Occ(X), k, n)

- Input: X:itemset, Occ(X), k: tail, n: maximum item;
- **if** Freq(X) = |Occ(X)| < σ **then**,
  - **return**. //Backtrack
- **Output** a frequent set **X**.
- **for** i = k+1,...,n **do**:
  - 出現リストOcc(X∪{i})を計算する.
  - BacktrackExpand(**X∪{i}**, Occ(X∪{i}), i, n) .

Efficient update of Occ(X)

Properties:

- For any X1, X2, $Occ(X \cup Y) = Occ(X) \cap Occ(Y)$. (basic)
- For any set X, item a, $Occ(X \cup \{a\}) = \{ t \in Occ(X) : a \in t \}$

items

tuples					
t1	1	2	3	4	
t2	2	4	5		
t3	1	2	4	5	…
t4	2	3			
t5	1	2	4		

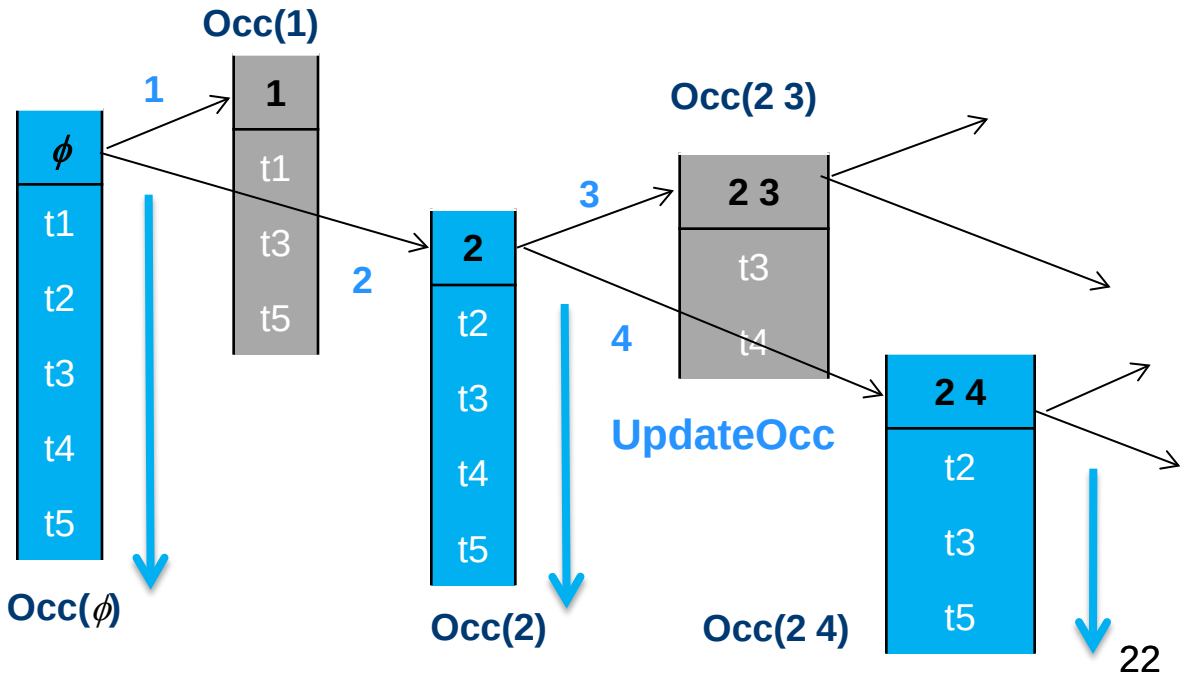
database ***D***

items

1	2	3	4	5
t1	t2	t1	t2	t4
t3	t3	t3	t3	
t5	t4	t4	t5	
	t5			

Vertical layout: The occurrence list representation

Update of occurrence lists:
Downward closure [Zaki et al '97]



DFS over the set enumeration tree

Efficient update of Occ(X)

Properties:

- For any X1, X2, $Occ(X \cup Y) = Occ(X) \cap Occ(Y)$. (basic)
- For any set X, item a, $Occ(X \cup \{a\}) = \{ t \in Occ(X) : a \in t \}$

手続き UpdateOcc(X, a, O(X))

Input: $X \subseteq \Sigma$と, $a \in \Sigma$, O(X).

Output: $O(X \cup \{a\})$.

- $Y = X \cup \{a\}$; $Occ(Y) = \emptyset$;
- **foreach** $t \in O(X)$ do
  - **if** $a \in t$ **then** $Occ(Y) = Occ(Y) \cup \{t\}$;
- **return Occ(Y);**

Downward closure [Zaki et al '97]

DFS: Main result

Theorem

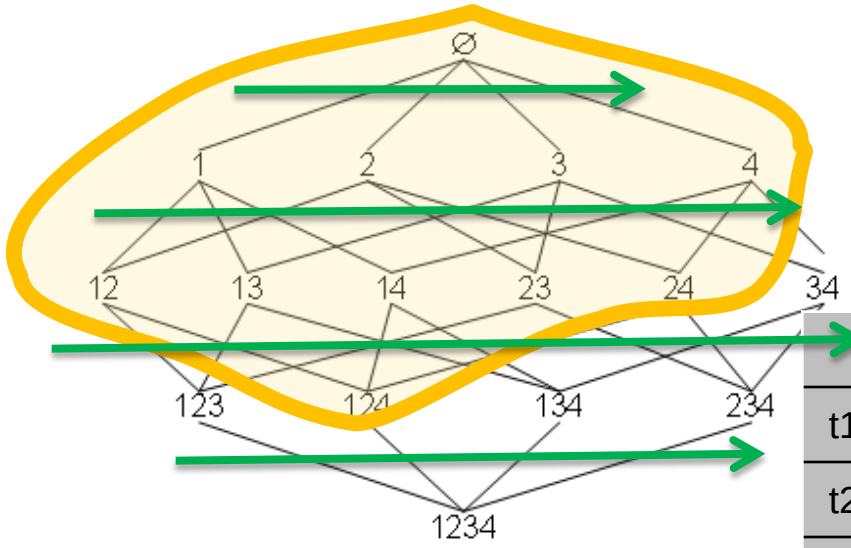
- The algorithm **Backtrack** can be implemented to enumerates all σ-frequent itemsets **X** in a given transaction database ***T***
  - in $O(l \cdot |Occ(X)|)$ time per frequent itemset
  - using $O(l \cdot |Occ(X)|)$ space
- where l is the maximum length of tuples in ***T*** and $|Occ(X)|$ is the number of occurrences of X.
- Space and time efficient (poly-delay & poly-space)

- On the other hand, the algorithm **Apriori** requires $O(l \cdot |Occ(X)|)$ time per frequent itemset and $O(\max_i ||\mathcal{F}_i||)$ space.

Summary: BFS vs. DFS

Apriori algorithm [1994]

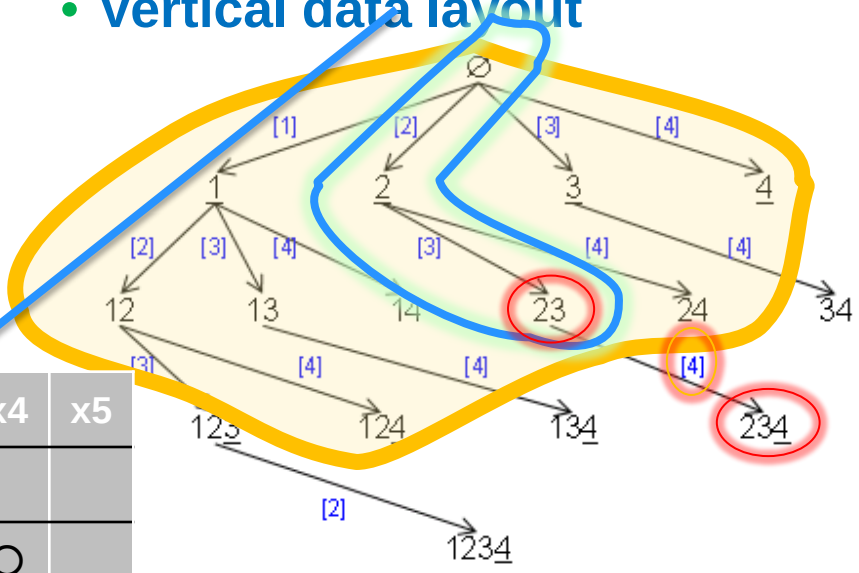
- Breadth-first search (BFS)
- Horizontal data layout



- 1st generation
- External memory algorithm

Backtrack algorithm [1997-1998]

- Depth-first search (DFS)
- Vertical data layout

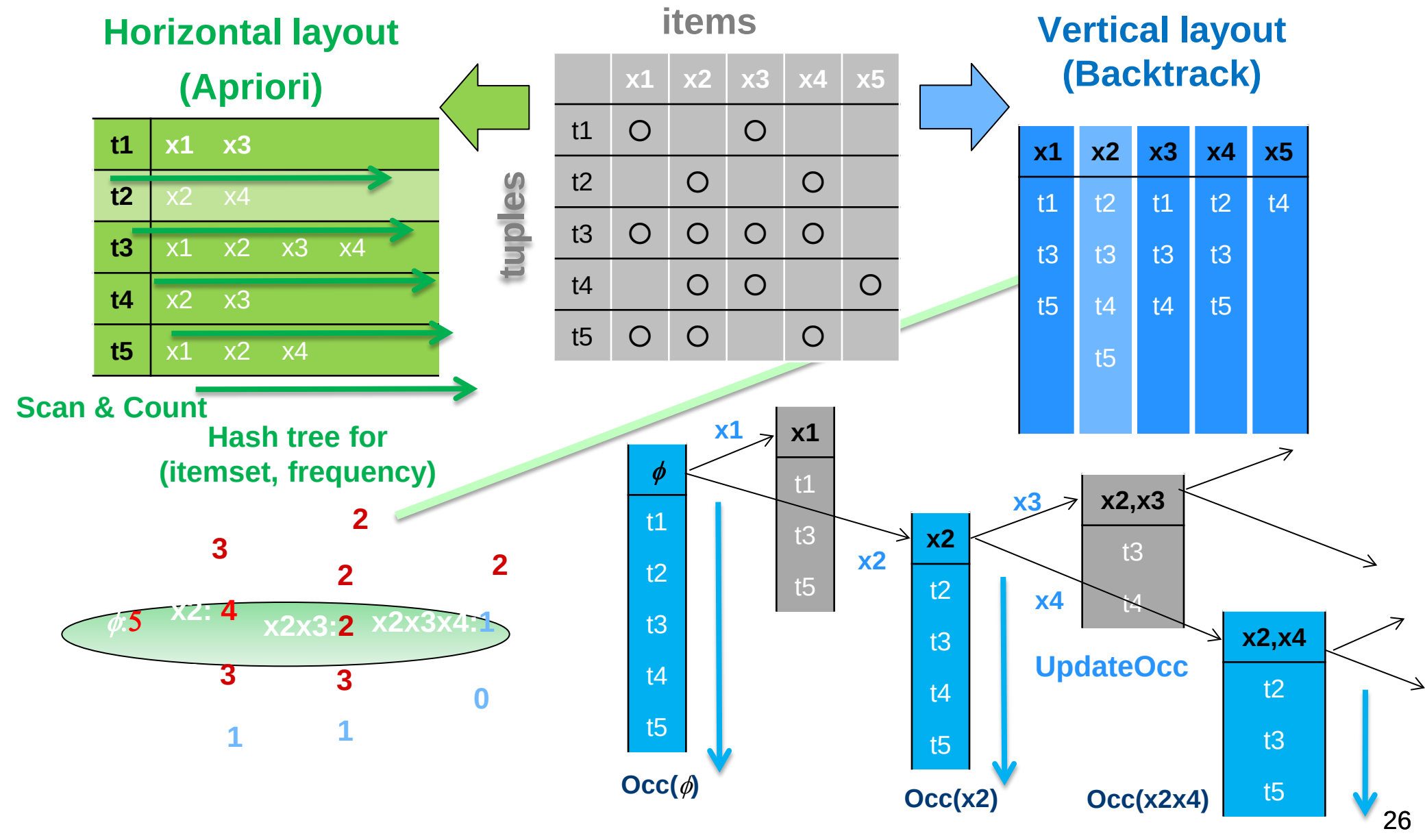


- 2nd generation
- Space efficient
- In-core algorithm

	x1	x2	x3	x4	x5
t1	○		○		
t2		○		○	
t3	○	○	○	○	
t4		○	○		○
t5	○	○		○	

database

Summary: Horizontal & Vertical Layout



FP-growthアルゴリズム

[Han, Pei, Yin, SIGMOD'00]

- パターンとデータをトライ(Trie/Prefix tree/FP-Tree)に圧縮して格納
- DFS＋出現リスト方式の変種
- 実際のデータに対して高速といわれている

Example) Expand pattern 27 into 279 and then update its occurrence list

Pattern 279 appears 4 times in the database!

頻出集合

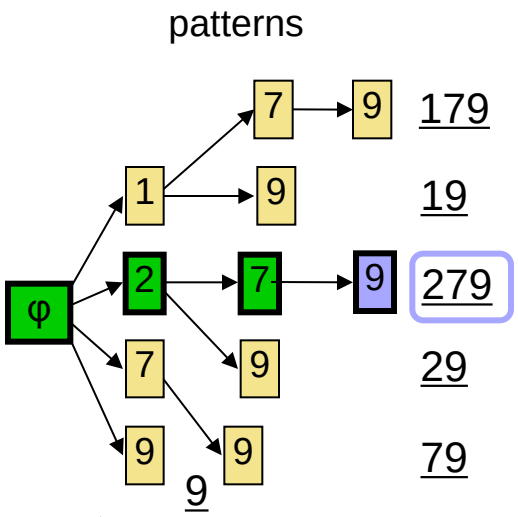
Φ

{1} {2} {7} {9}

{1,7} {1,9} {2,7}

{2,9} {7,9} {1,7,9}

{2,7,9}



頻出パターン木（FP-Tree）
＝トライを用いて圧縮されたパターン集合

データベース

A: 1,2,5,6,7,9

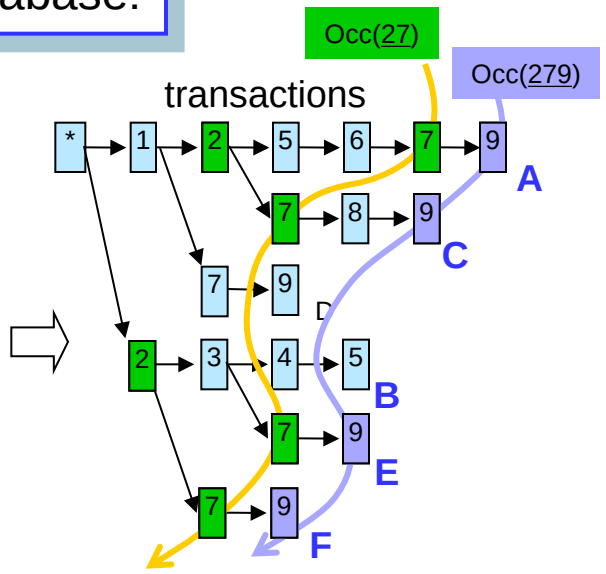
B: 2,3,4,5

C: 1,2,7,8,9

D: 1,7,9

E: 2,3,7,9

F: 2,7,9



トライを用いて圧縮されたデータベース

J. Han, J. Pei, Y. Yin, Mining Frequent Patterns without Candidate Generation, *Proc. SIGMOD Conference 2000*, pp. 1-12 (2000).

飽和集合／極大集合マイニング

- **LCM** (Linear-time Closed Itemset Miner)
  - 超高速代表元集合マイニングアルゴリズム
  - 理論的性能保証：出力線形時間アルゴリズム
  - 公開 & 応用

くわしくは宇野毅明の講義で．．．

高速な代表元パターンマイニング
(宇野・有村, DS'04, FIMI'04)

ニュース速報 (Nov. 1, 2004, Brighton, UK)

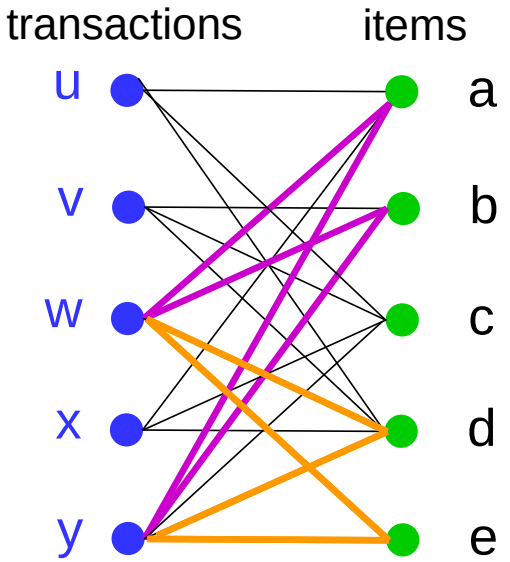
第2回 FIMI Workshop で宇野・清見・有村組(LCM ver.2) が優勝！
FIMI'04 Best Implementation Award

宇野毅明先生 (NII, A01班)

- **FIMI Workshop**:
「頻出アイテム集合マイニング実装（FIMI)」に関するデータマイニング分野のプログラミングコンテスト
今年は，8+5 実装が最終エントリ (問合せ80件超)
- **LCMアルゴリズム**: PPC拡張による出力多項式時間の閉アイテム集合列挙アルゴリズム [宇野・有村 DS'04]

極大2部クリーク c

({w, y}, {a, b, d})



- LCM, 宇野先生HP, program codes：http://research.nii.ac.jp/~uno/codes.htm
- FIMI Frequent Itemset Mining Implementations Repository: http://fimi.cs.helsinki.fi/

まとめ

3回: 頻出集合発見:深さ優先アルゴリズム

- 頻出集合マイニング
- 深さ優先探索アルゴリズム
  - Backtrackアルゴリズム
  - アルゴリズム:Eclat, FP-growth, LCM

ポイント

- 頻出集合の単調性
- 集合列挙木上
- バックトラックによる深さ優先探索

Summary

Frequent Itemset Mining

- Finding all "frequent" sets of elements appearing in a given transaction data base.
- Introduced by Agrawal and Srikant [VLDB'94]
- One of the most basic data mining problem

Algorithms

- BFS algorithm — Apriori [Agrawal et al, '94]
- DFS algorithm — Backtrack [Zaki et al. '97; Morishita'98]

Applications

- Feature extraction for SVM & Boosting
- Closed and Maximal Frequent Itemset Mining
- Sequences and Graphs

Bibliography

Frequent Itemset Mining: DFS algorithms

- [Agrawal & Srikant, VLDB'94] R. Agrawal, R. Srikant: Fast Algorithms for Mining Association Rules in Large Databases. Proc. VLDB 1994, pp. 487-499, 1994. (Apriori)
- [AMST 96] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, A. I. Verkamo: Fast Discovery of Association Rules, Advances in Knowledge Discovery and Data Mining, pp. 307-328, 1996.
- [Bayardo, SIGMOD'98] R. J. Bayardo Jr.: Efficiently Mining Long Patterns from Databases, Proc. SIGMOD Conference 1998: pp. 85-93, 1998. (set-enumeration tree)
- [Han et al. SIGMOD'00] J. Han, J. Pei, Y. Yin, Mining Frequent Patterns without Candidate Generation, Proc. SIGMOD Conference 2000, pp. 1-12, 2000. (FP-growth)
- [Morishita & Sese PODS'00] S. Morishita, J. Sese: Traversing Itemset Lattice with Statistical Metric Pruning, Proc. PODS 2000, pp. 226-236, 2000. (set-enumeration tree)
- [Zaki et al., KDD'97] M. J. Zaki, S. Parthasarathy, M. Ogihara, W. Li, New algorithms for fast discovery of association rules, Proc. KDD 1997, pp. 283-286, 1997. (Eclat)
- [宇野, 有村] 宇野毅明, 有村博紀,「データインテンシブコンピューティング その2 ー 頻出アイテム集合発見アルゴリズム ー」,人工知能学会誌, レクチャーシリーズ「知能コンピューティングとその周辺」, (編)西田豊明, 第2回, Vol.22, No.3, 2007年5月. (PDF: http://www-ikn.ist.hokudai.ac.jp/~arim/jtalks.html)