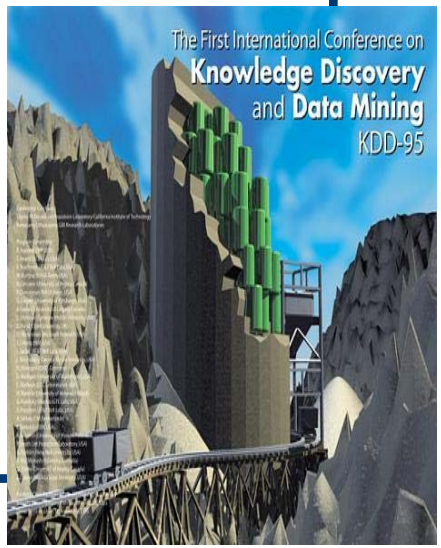
列挙学校：第3コマ
パターンマイニングにおける列挙アルゴリズム

有村 博紀
北海道大学大学院 情報科学研究科
コンピュータサイエンス専攻
http://www-ikn.ist.hokudai.ac.jp/~arim

教材最新版PPTのダウンロード：http://www-ikn.ist.hokudai.ac.jp/~arim/jtalks.html

3コマ目の目的:
データマイニングを題材に

- 列挙はどんなことに使えそうか？
- 応用分野では，列挙はどのように使われているか？
- データマイニングと機械学習における列挙に関連した話題を提供



http://www.sigkdd.org/kdd/1995/poster/kdd1995-poster-thumb.jpg

目次

パート1: データマイニングと頻出集合発見

- データマイニング
- 頻出集合マイニング

パート2: 系列とグラフのマイニング

- 系列とグラフのマイニング概観
- 列挙によるパターンのマイニング
- 列挙とマイニングに関連した話題

列挙学校：第3コマ

パート1: データマイニングと頻出集合発見

有村 博紀
北海道大学大学院 情報科学研究科 コンピュータサイエンス専攻
email: {arim,kida}@ist.hokudai.ac.jp
http://www-ikn.ist.hokudai.ac.jp/~arim

.

Backgrounds

データマイニング

- 大量のデータから人間にとって有用なパターンや規則を半自動的にとりだす方法の研究
- 1990年代半ばから研究が盛んになった
  - Apriori algorithm [Agrawal, Srikant, VLDB1994]
- 潜在的には古くからある研究の集大成.
  - ただし，大量データに対する効率的計算に重点
- 機械学習・数理統計学・データベース技術の境界分野

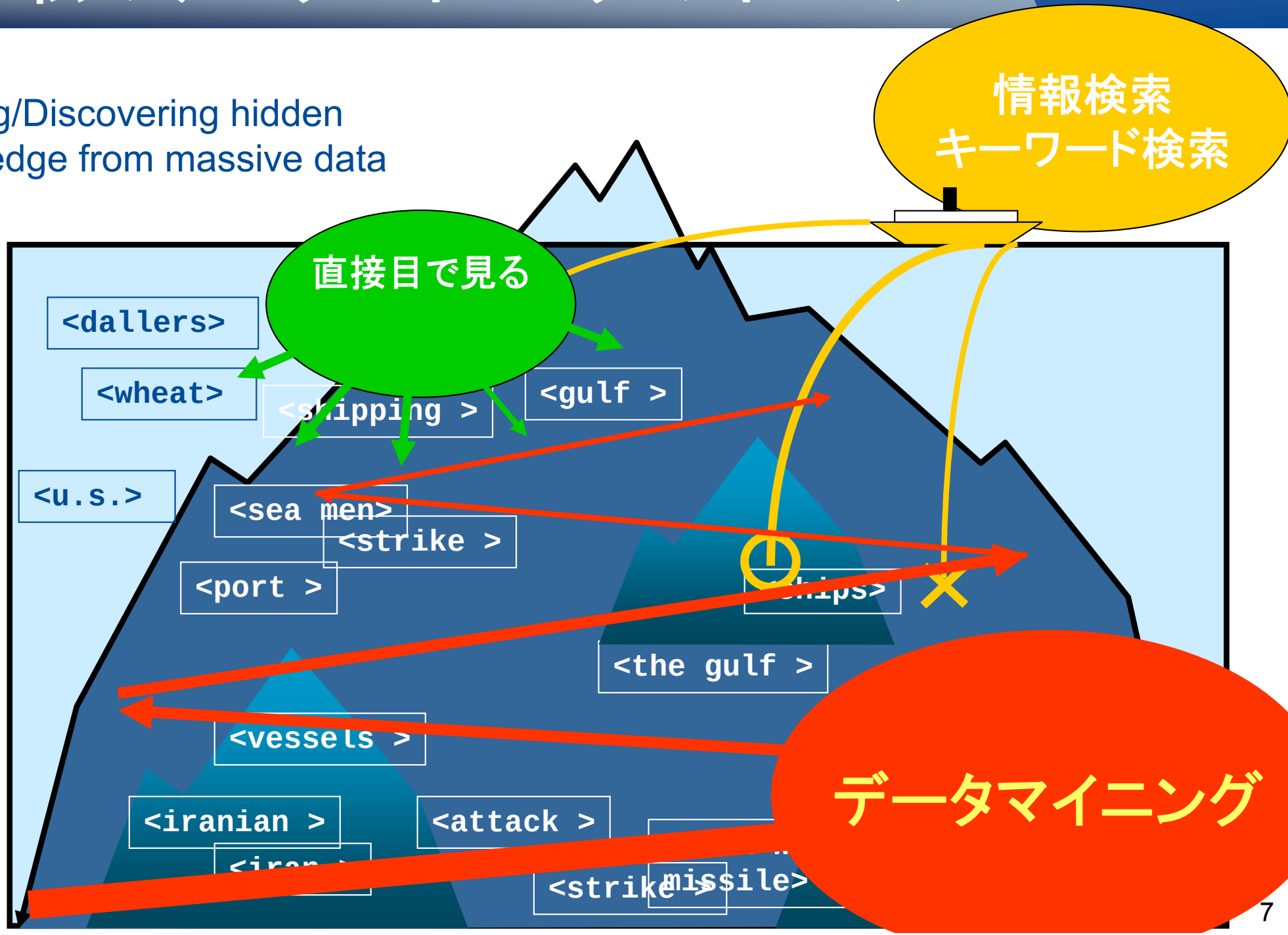
Backgrounds

データマイニングのプロセスの全体

- 1.対象領域の理解
- 2.データ集合の前処理
- 3.パターンの発見（狭義のデータマイニング）
- 4.得られたパターンの解析
- 5.解析結果の利用

私のデータマイニングのイメージ

Finding/Discovering hidden
knowledge from massive data



Backgrounds：4テラバイトのデータ

4テラバイト ＝4,000,000,000,000バイト

- 400字詰めA4原稿用紙で309km^2
  ＝ 山手線の内側の約3倍の面積
- 1秒間に10文字入力すると12,683年
- CD 6153枚 ＝ 音楽として聴くと316日間
- 記憶装置からの読み込みに 15.2時間
- ちょっとした計算(N^2時間）
  2.2 GFLOPS × 32PE のスーパーコンピュータで
  800万年以上かかる。

•高速なアルゴリズムが必要！！！

データマイニングの動向

パターン発見

- トランザクションデータから共通して出現する規則性を発見する
- 頻出パターン発見 [Agrawal et a. ‘94]
- 最適化マイニング [森下 ’96, ’98, ‘00]

予測学習・自動分類

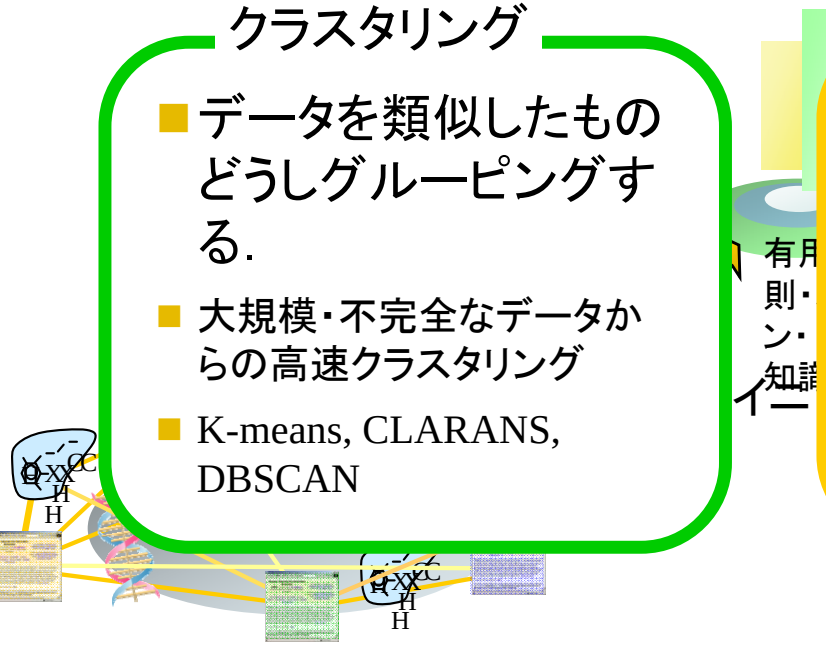
- 不完全なデータから，未知の規則を学習する
- SVM [Vapnik ‘96],
- Boosting [Shapire & Kearns ‘96]
- C4.5 [Quinlan ‘96]

構造マイニング

- 非定型構造データから特徴的な部分構造を規則性を発見する
- グラフマイニング[Washio & Motoda ‘00], [Zaki ’02], [Uno, Asai, Arimura, ’02, ‘03]

クラスタリング

- データを類似したものどうしグルーピングする.
- 大規模・不完全なデータからの高速クラスタリング
- K-means, CLARANS, DBSCAN



確率モデリング

- 高次元大規模データから不確実な現象を予測・モデル化する
- ベイジアンネットワーク [Pearl ’90s]
- HMM [Asai], MCMC, ベイズ推定・MDL・AIC

新しいタイプのデータマイニング

- テキストマイニング
  - 自然言語テキスト
  - 情報抽出
  - 意味マイニング
- ストリームマイニング
  - センサー監視

Frequent Itemset Mining Definitions

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

動機：結合ルールマイニング

- Association Rule Mining [Agrawal 1993/1994]
  - Finding combination of “items” frequently appearing in a given database

- トランザクションデータ
- バスケットデータ
- 二値データベース

- レコード/タプル
- バスケット

ID	Chips	Mustard	Sausage	Softdrink	Beer
001	1	0	0	0	1
002	1	1	1	1	1
003	1	0	1	0	0
004	0	0	1	0	1
005	0	1	1	1	1
006	1	1	1	0	1
007	1	0	1	1	1
008	1	1	1	0	0
009	1	0	0	1	0
010	0	1	1	0	1

- カラム/属性
- アイテム

トランザクション／レコードの意味
「レコード003の顧客は，ポテトチップとソーセージを一緒に買った」

動機：結合ルールマイニング

- Frequent Itemset **X** = { Mustard, Sausage, Beer }
  - with support/frequency 40%

ID	Chips	Mustard	Sausage	Softdrink	Beer
001	1	0	0	0	1
002	1	1	1	1	1
003	1	0	1	0	0
004	0	0	1	0	1
005	0	1	1	1	1
006	1	1	1	0	1
007	1	0	1	1	1
008	1	1	1	0	0
009	1	0	0	1	0
010	0	1	1	0	1

- アイテム集合 X
- Xの出現リスト
  Occ(X) = {002, 005, 006, 010}
- Xの頻度（サポート）
  freq(X) = |Occ(X)| = 4/10 = 40%

アイテム集合 X の意味
X =「マスタードとソーセージ，ビールを一緒に買う人」が全体の40%いた

頻出アイテム集合マイニング

Frequent Itemset Mining Problem

- Given: A database DB over a set Σ of items, and a number $\sigma \geqq 0$ called “minsup” (minimum support)
- Problem: Find all itemsets $X \subseteq \Sigma$ appearing in no less than σ records of DB

Definitions: Database

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

Alphabet of items

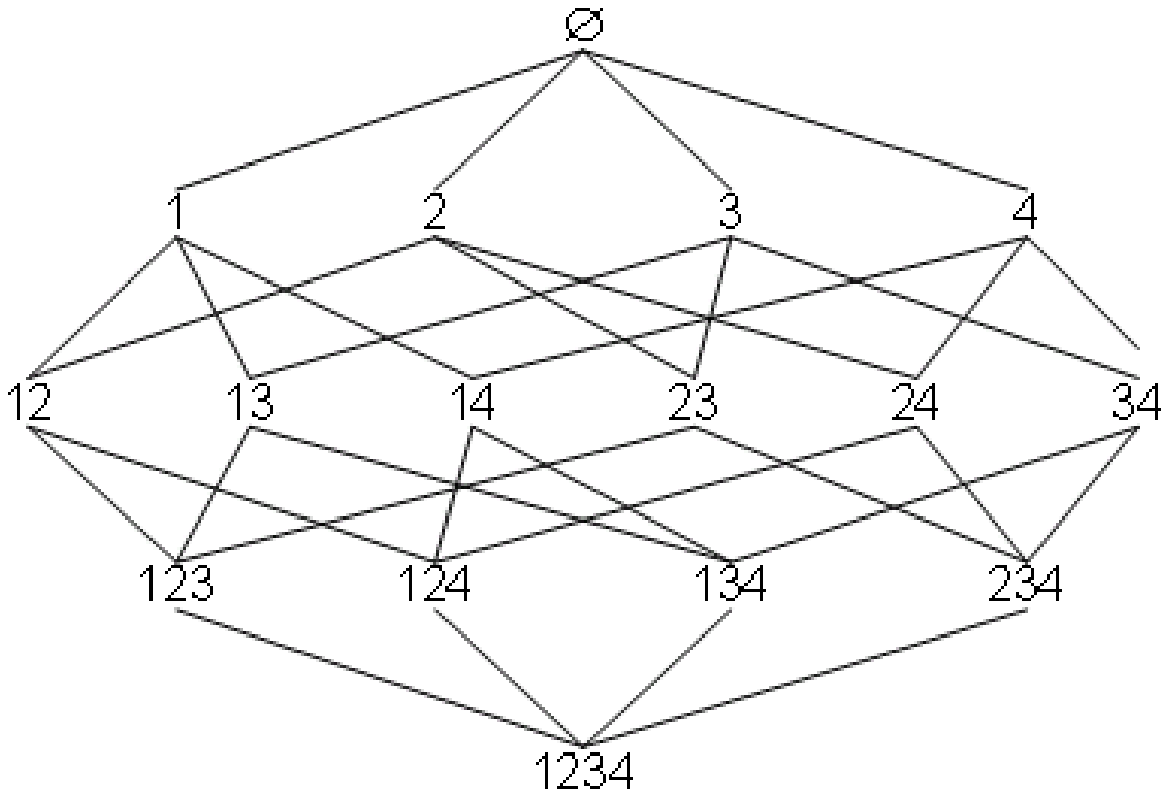
$I = \{1,2,3,4\}$

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



Example:
The set lattice
for Σ = {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

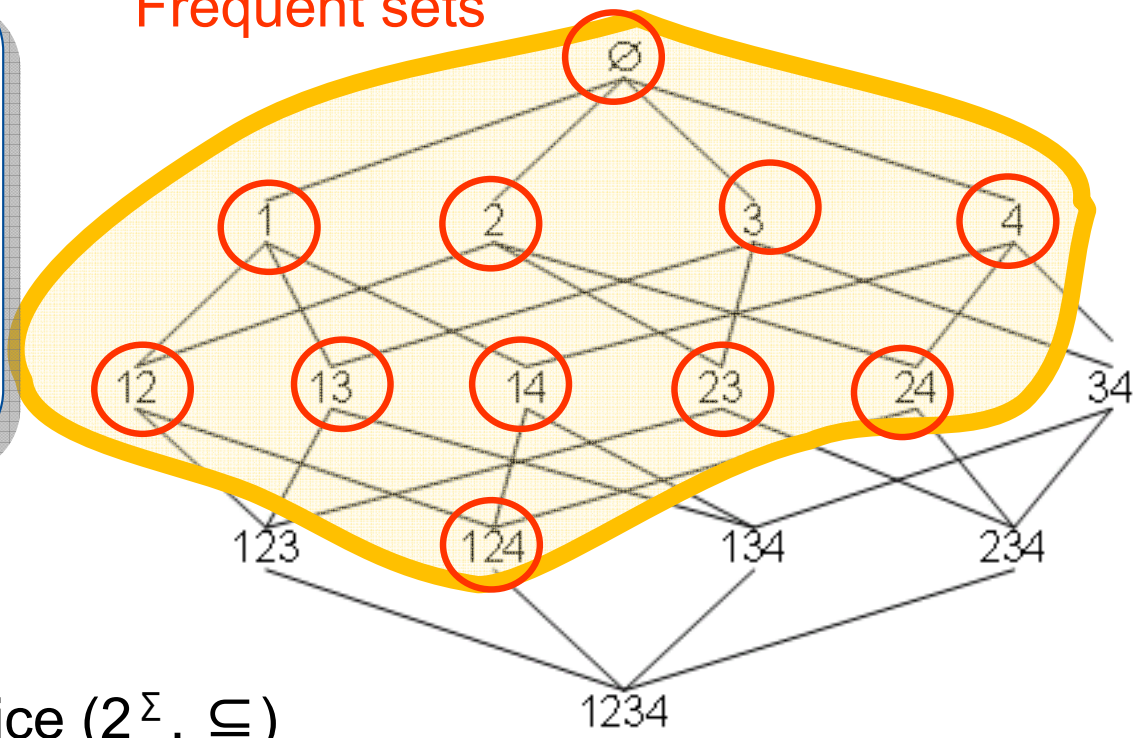
Definitions: Frequent sets

- 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 $Fr(X, T) = |Occ(X, T)| \geqq \sigma$.

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

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

頻出アイテム集合マイニング

Frequent Itemset Mining Problem

- Given: A database DB over a set Σ of items, and a number $\sigma \geqq 0$ called “minsup” (minimum support)
- Problem: Find all itemsets $X \subseteq \Sigma$ appearing in no more than σ records of DB

Applications to more sophisticated data mining problems

- Association rule [Agrawal, Srikant ‘94]
  - {Mustard, Sausage, Beer => PotateChips } **with frequency** 40%
- Optimized classification rule [Sese & Morishita PODS’90]
  - **If** gene0001 **&** gene0012 **then** diabetes **with classification error** 8.5%
- Itemset boosting/SVM [Saigo, Uno, Tsuda Bioinformatics **23(18)** 2007]
  - Learning a linear classifier over itemsets as composite features
- Weighted substructure mining [Nowozin, Tsuda, Uno, Kudo, Bakir, CVPR’07]
  - Application to image processing

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]

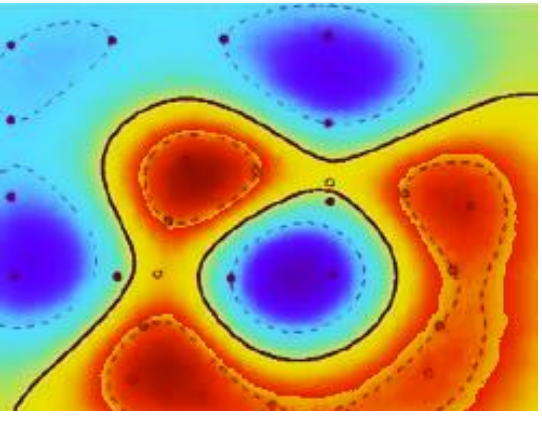
- 多数の機械学習アルゴリズムを統合して高精度予測
- オンライン予測の理論と深い関連

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)

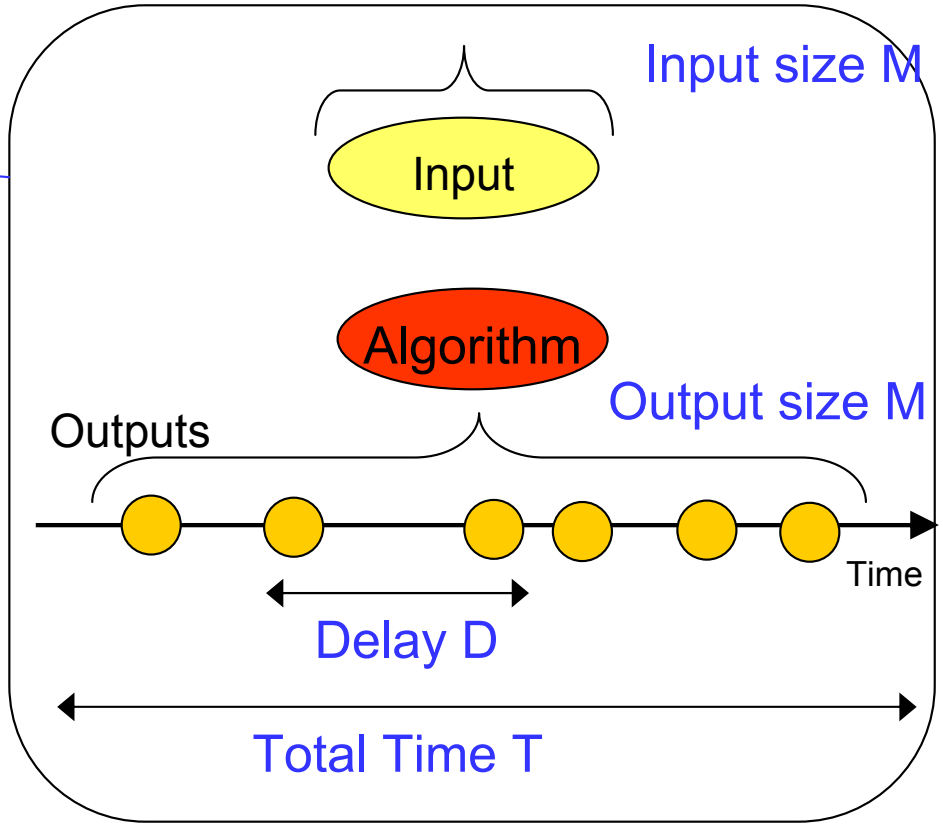
How to model efficient data mining algorithms?

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

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)

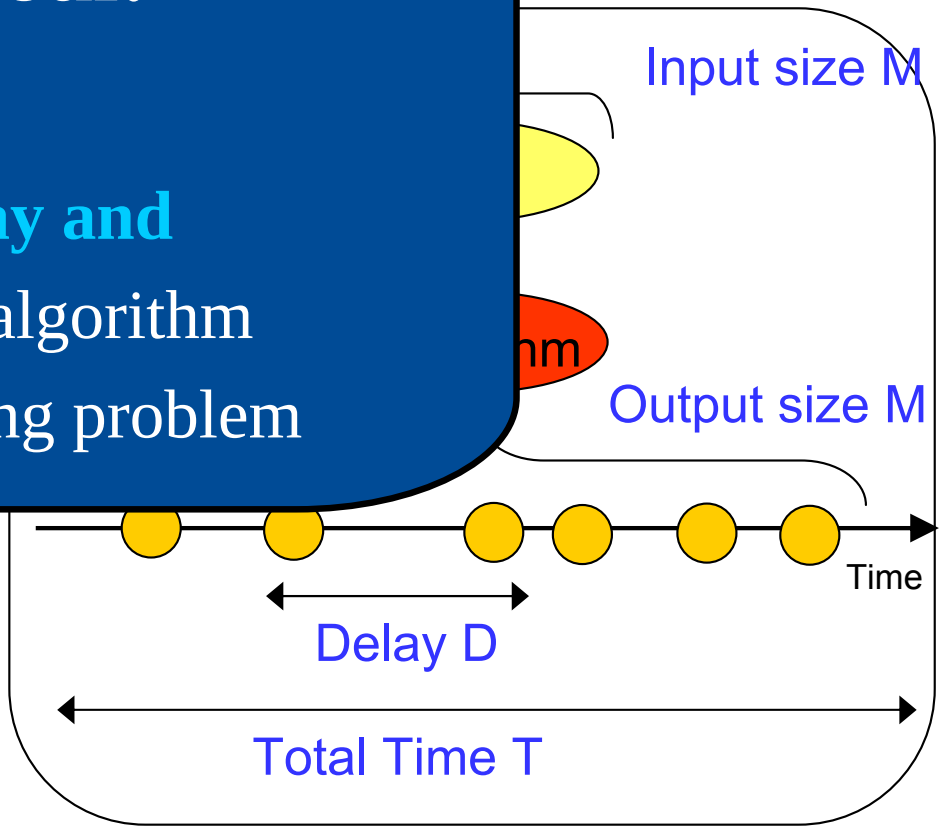
Modeling data mining as enumeration

- **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
- **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

Problems

- 1. How to enumerate all frequent subsets without duplicates?

→列挙の問題！

- 2. How to compute the frequencies quickly for each subset?

“Anti-monotonicity” of frequent sets

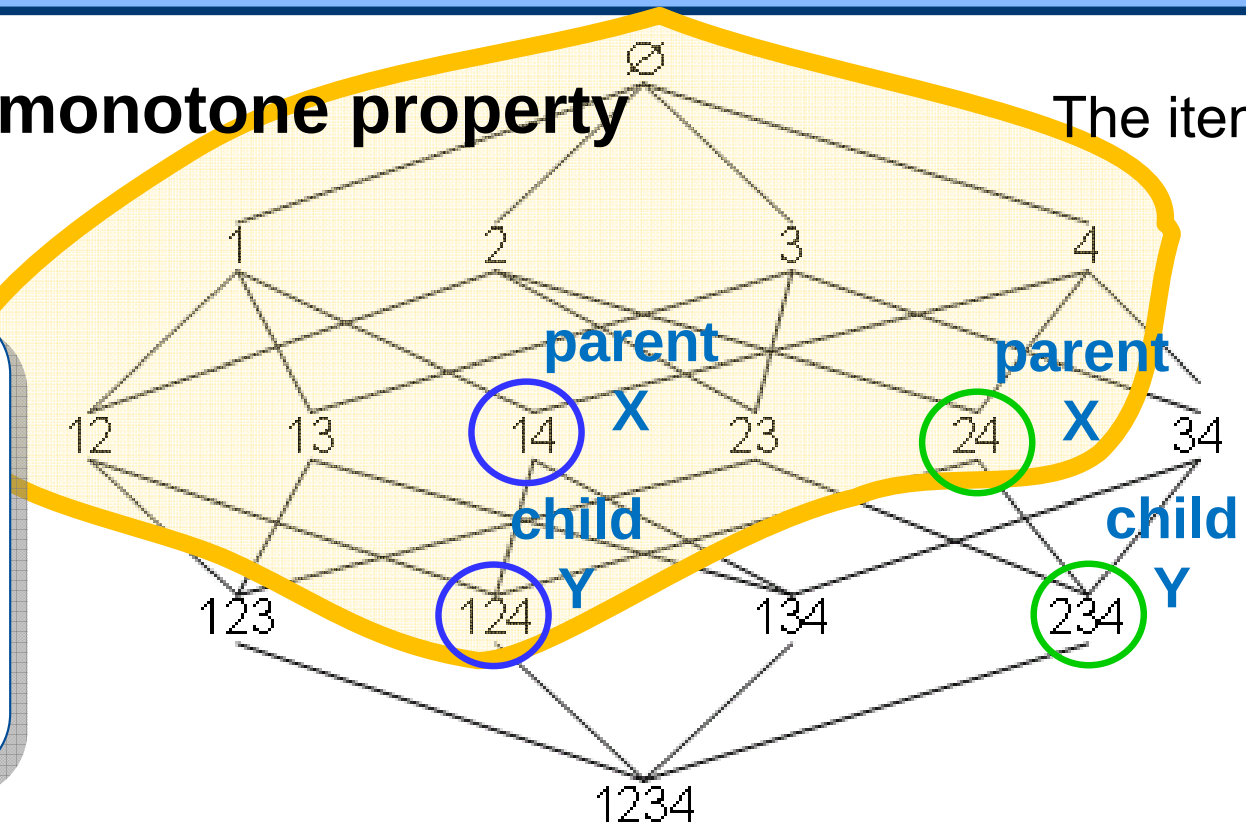
- **Lemma**: For every minsup σ, if **Y** is σ-frequent then any subset **X** of X is also σ-frequent in DB ***T.***
- **Corollary:** Every non-empty σ-frequent set **Y** is obtianed from some σ-frequent set **X** by adding some new element.
- The class F_σ of all frequent sets is monotone subclass of $(2^\Sigma, \subseteq)$

Anti-monotone property

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

minsup σ = **2**

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



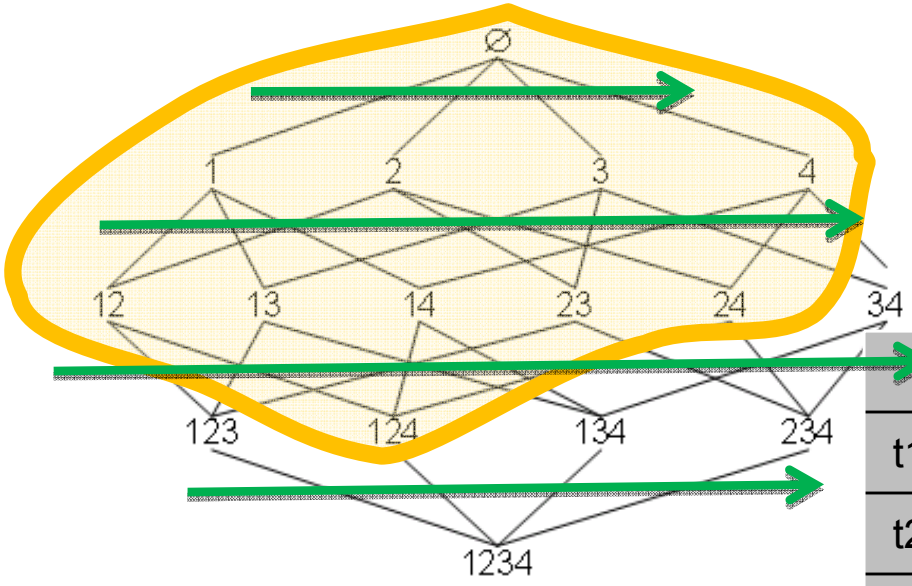
	1	2	3	4	5
t1	○		○		
t2		○		○	
t3	○	○	○	○	
t4		○	○		○
t5	○	○		○	

database

Two approaches to Frequent sets mining

Apriori algorithm [1994]

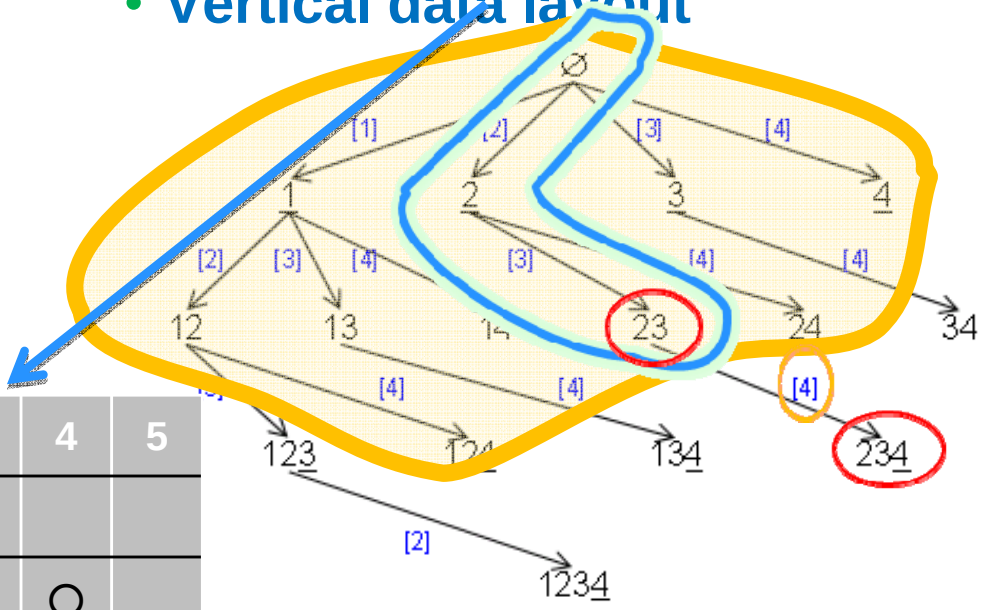
- Breadth-first search (BFS)
- Horizontal data layout



- 1st generation
- External memory algorithm

Backtrack algorithm [1997-1998]

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

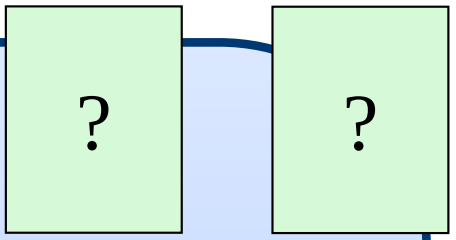


- 2nd generation
- In-core algorithm
- Space efficient

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

database

BFS (Breadth-first search) algorithm



Dr. Agrawal
IBM Almaden Lab.

Prof. Mannila
Helsinki Inst. Tech
Univ. Helsinki

APRIORI Algorithm

- Most popular, the 1st generation frequent itemset miner
- [Agrawal & Srikant, VLDB'94; Mannila, Toivonen, Verkamo, KDD'94]

Cadidate Generation: **BFS (Breadth-first search)**

- Starting from the smallest element, search the itemset lattice from smaller to larger
- Generate itemsets **level by level** (level-wise algorithm)

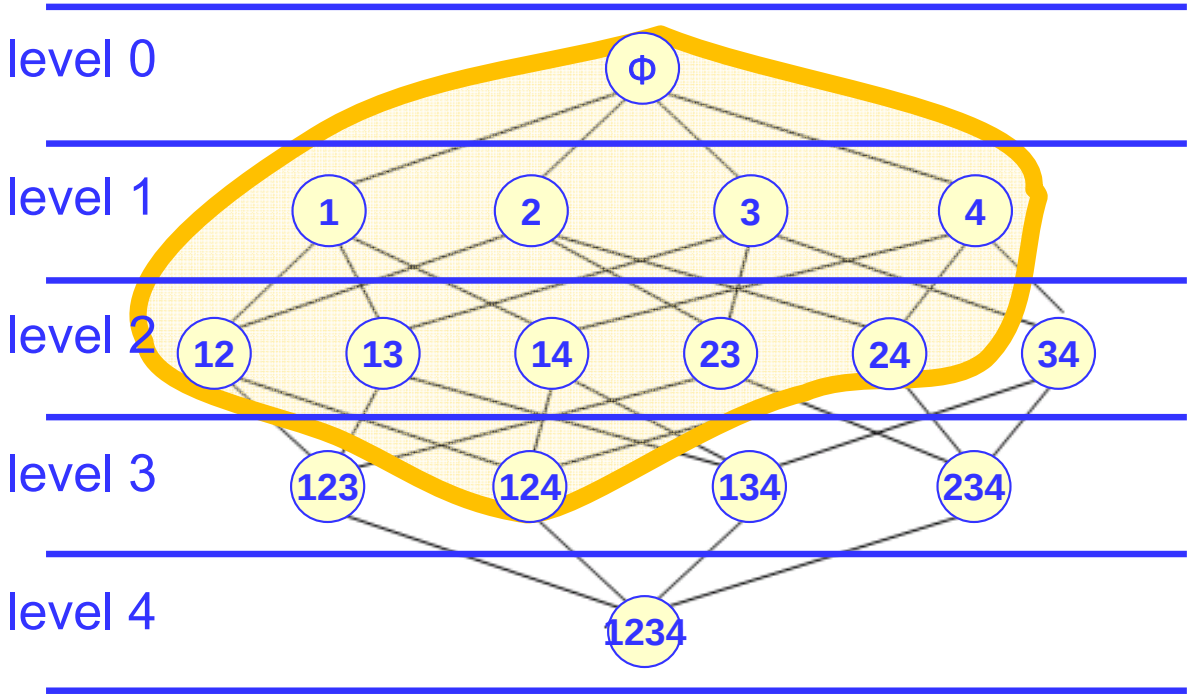
Frequency Counting: **Horizontal Data Layout**

- Sequentially scanning a large database placed on a hard disk from left to right to compute the frequencies
- All the candidate itemsets is stored in a dictionary (a *hash tree*) in main memory.

Basic Idea: Cadidate Generation by BFS

Apriori algorithm [Agrawal, Srikant, 1994] (Also called Level-wise [Mannila et al., 1994])

- Slice the item set-lattice 2^{Σ} into levels $L_0, L_1, \ldots, L_n$ ()
- Compute $\boldsymbol{F}_k = \{ X \in L_k : X \text{ is a frequent set of size } k \}$ for $k = 0,1,2, \ldots$ in a bottom-up manner
- Generation of the k-th candidate set $\boldsymbol{C}_{k+1}$ *from* $\boldsymbol{F}_k$:
  - For each $a_1 \ldots a_k$ and $b_1 \ldots b_k \in \boldsymbol{F}_i$ with $a_1 = b_1 \ldots a_{k-1} = b_{k-1}$ and $a_k < b_k$, add $a_1 \ldots a_k b_k$ into $\boldsymbol{C}_i$



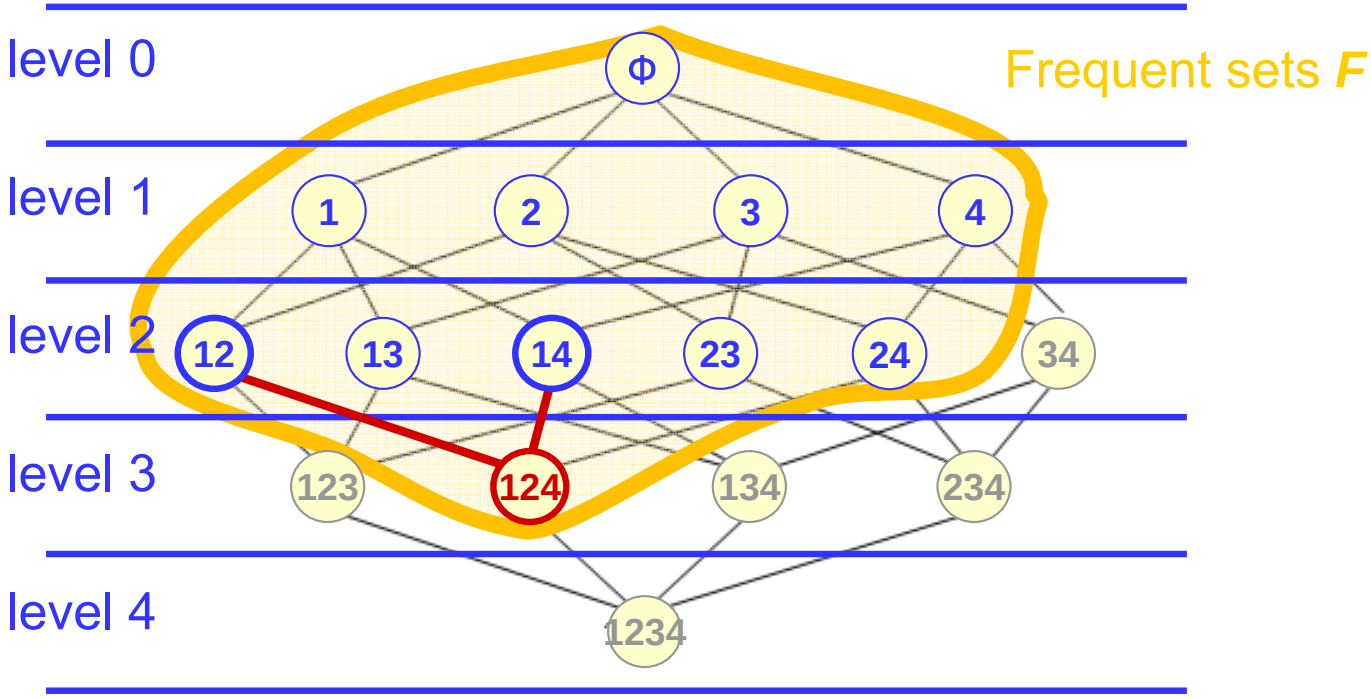
	1	2	3	4	5
t1	○		○		
t2		○		○	
t3	○	○	○	○	
t4		○	○		○
t5	○	○		○	

database

Basic Idea: Cadidate Generation by BFS

Apriori algorithm [Agrawal, Srikant, 1994] (Also called Level-wise [Mannila et al., 1994])

- Slice the item set-lattice 2^{Σ} into levels $L_0, L_1, \ldots, L_n$ ()
- Compute $\boldsymbol{F}_k = \{ X \in L_k : X \text{ is a frequent set of size } k \}$ for $k = 0,1,2, \ldots$ in a bottom-up manner
- Generation of the k-th candidate set $\boldsymbol{C}_{k+1}$ *from* $\boldsymbol{F}_k$:
  - For each $a_1 \ldots a_k$ and $b_1 \ldots b_k \in \boldsymbol{F}_i$ with $a_1 = b_1 \ldots a_{k-1} = b_{k-1}$ and $a_k < b_k$, add $a_1 \ldots a_k b_k$ into $\boldsymbol{C}_i$



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

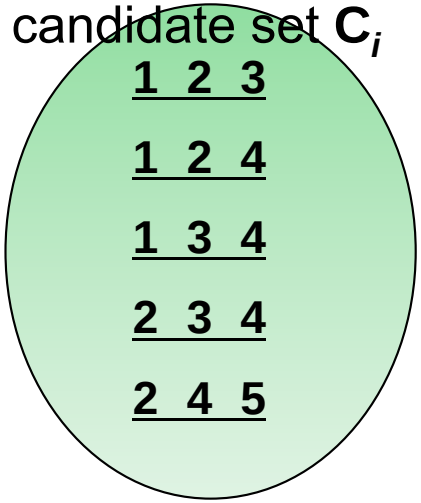
database

Basic Idea: Frequency Counting

Horizontal layout

- Scanning the database tuple by tuple
- For each tuple ***t***, increment the count of all candidate itemset that are subsets of ***t***.

0) Given a candidate set C_i



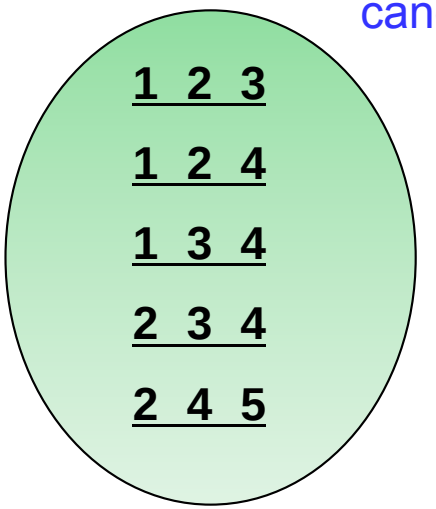
1) Scan the database sequentially tuple by tuple

Basic Idea: Frequency Counting

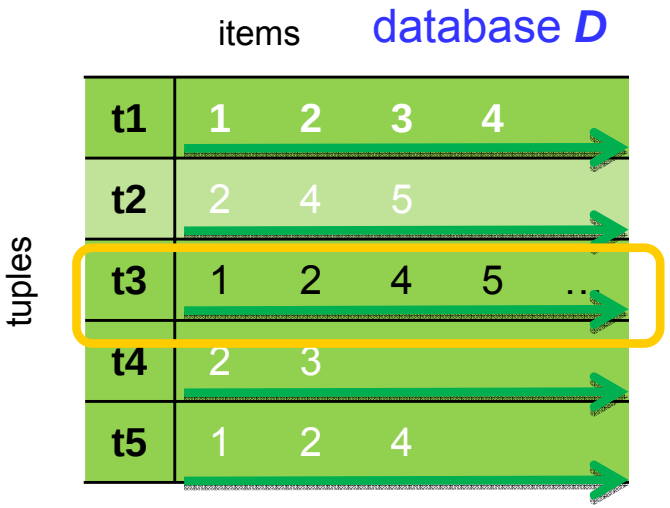
Horizontal layout

- Compute the frequencies of all candidate sets in C_i by scanning the database tuple by tuple
- For each tuple ***t***, increment the count of all candidate itemset that are subsets of ***t***.

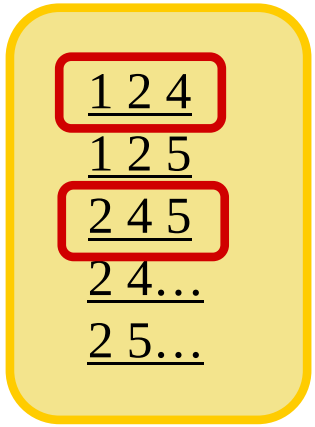
candidate set C_i



1) Scan the database sequentially tuple by tuple

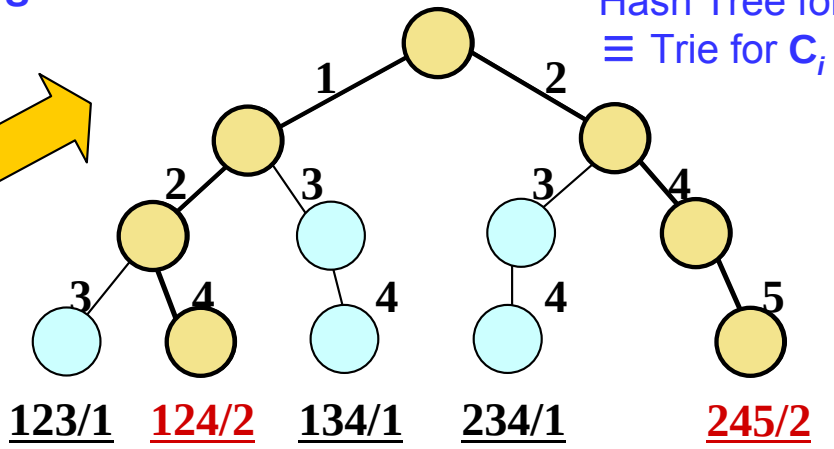


2) For each tuple ***t***, enumerate all subsets ***s*** of ***t*** with size *k*



3) Lookup the subset ***s*** in the hash tree and then increment its count

Hash Tree for C_i ≡ Trie for C_i



Breadth-first search algorithm

アルゴリズム Apriori(T:データベース, $0 \leq \sigma \leq |T|$: minsup):

出力：全頻出集合の集合F.

サイズ1の頻出集合の全体F_1を計算する． i = 2.

while (F_{i-1} /= ∅) **do** //**ステージi:**

- STEP1: F_{i-1}に含まれるサイズ(i-1)の頻出集合同士を組み合わせて，サイズiの候補集合の集まりC_iを計算する.
- STEP2: データベースを一度だけ走査し，各候補集合$X \in C_i$の頻度Freq(X)を一度に計算する． F_i = ∅.
- STEP3: C_iから頻度σ以上のすべての候補集合を取り出し，F_iに加える.
- STEP4: i = i + 1.

return $F = F_1 \cup \ldots \cup F_i$ を出力する.

Apriori アルゴリズム [Agrawal 1994, Mannila 1995]

The state-of-the-art algorithm for association rules

Clever use of the present computer technology

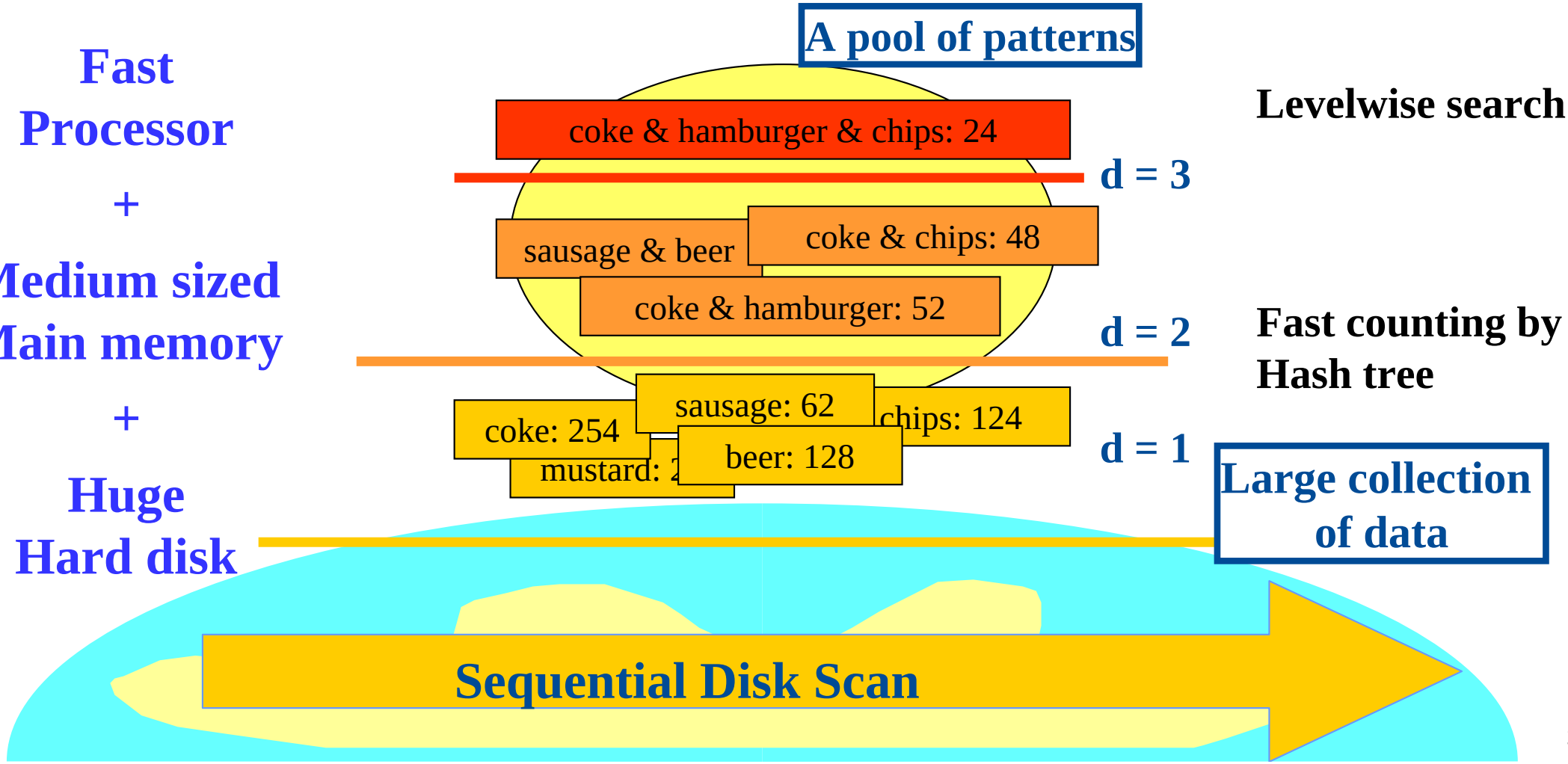
Fast Processor

+

Medium sized Main memory

+

Huge Hard disk



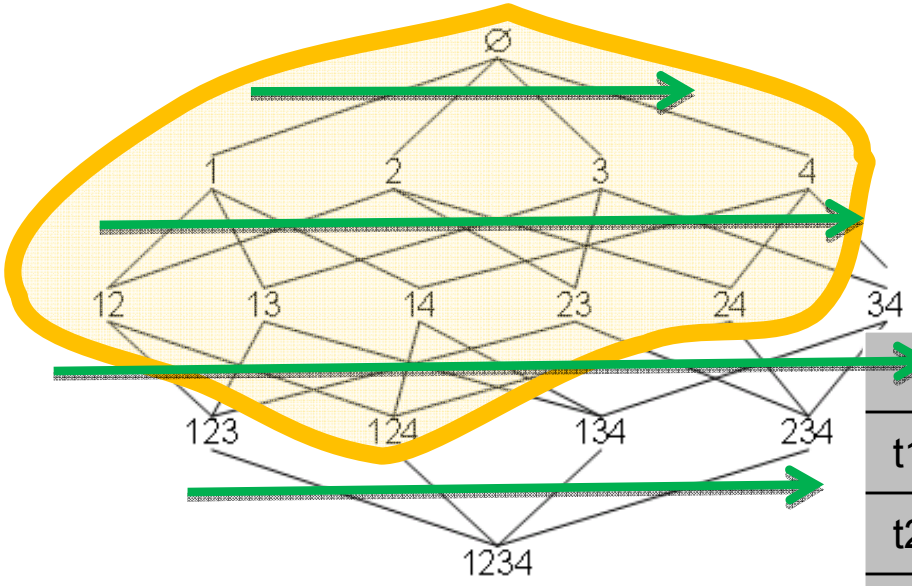
Levelwise search

Fast counting by Hash tree

Two approaches to Frequent sets mining

Apriori algorithm [1994]

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



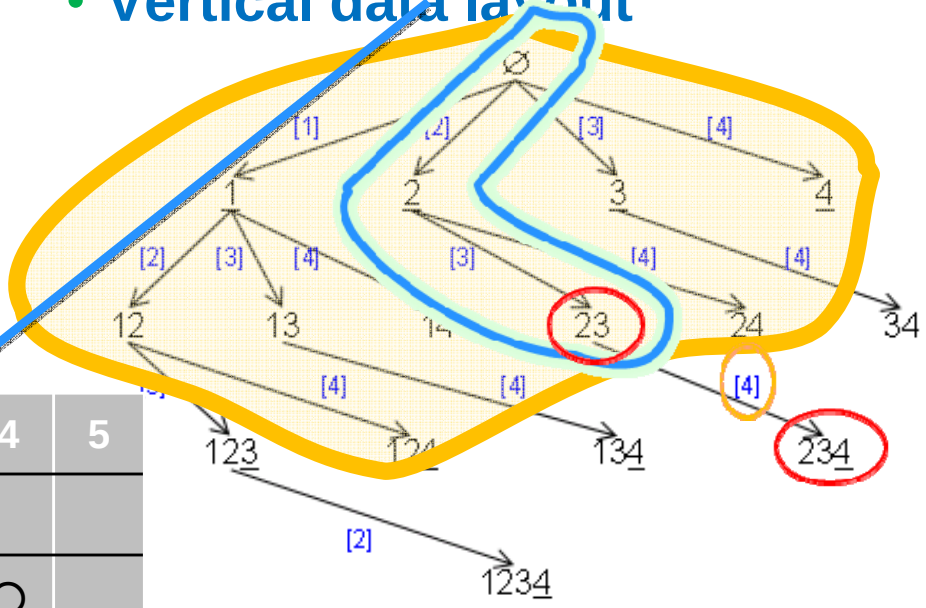
- 1st generation
- External memory algorithm

	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 T = (F, E_{pa}, ϕ)** for the class of frequent sets
  - Starting from ϕ, search **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

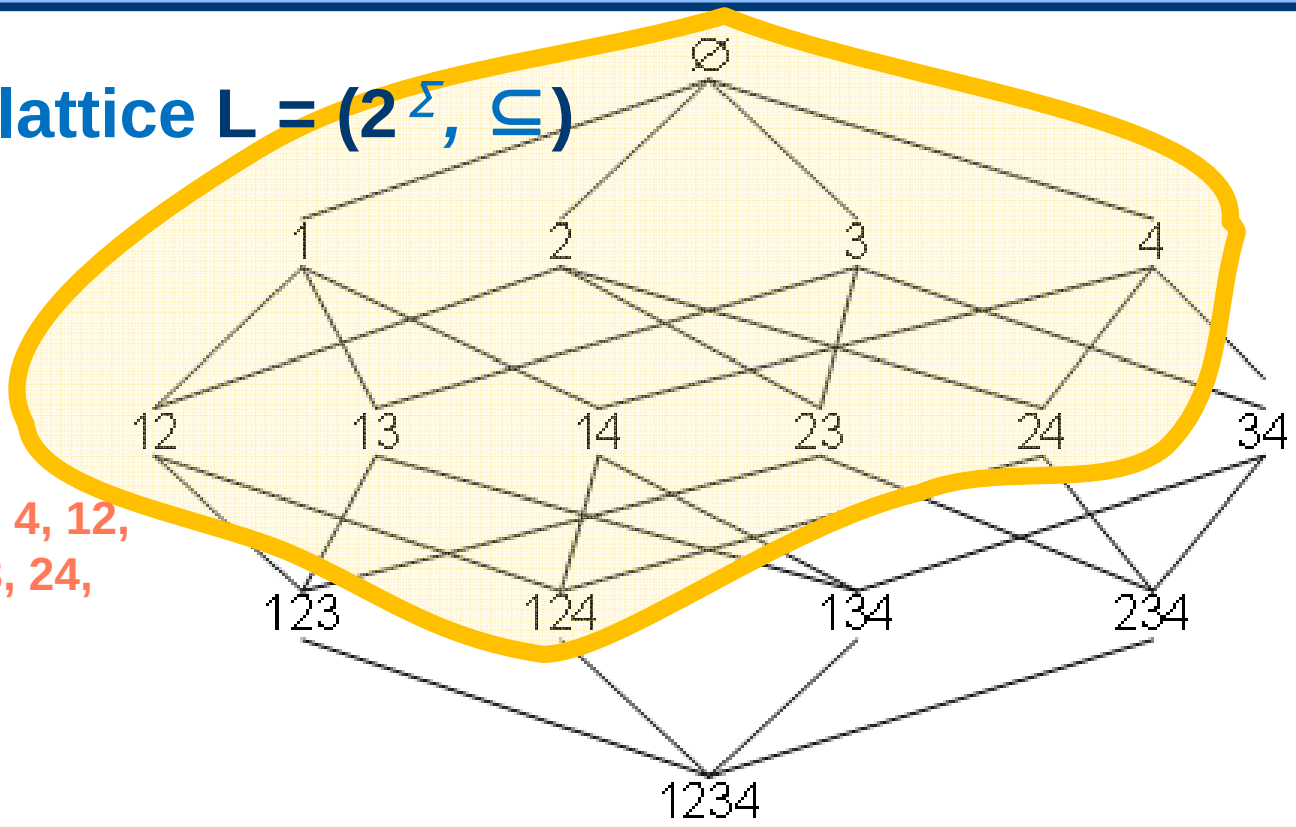
- For each enumerated itemset X, maintain its occurrence list Occ(X) to compute the frequencies.
- Incremental update of Occ(X) is possible: "*Downward closure*"

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 $L = (2^{\Sigma}, \subseteq)$



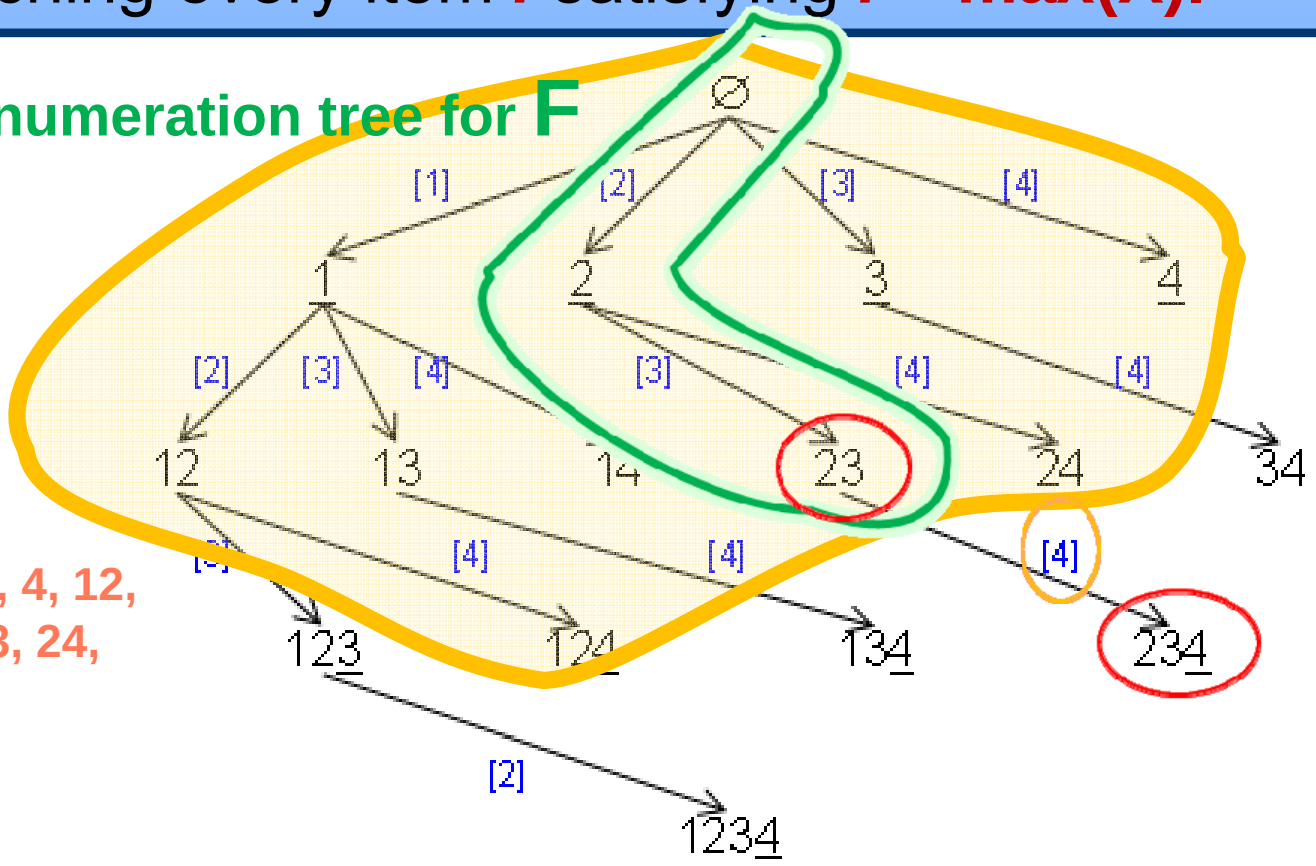
The set lattice for
$\Sigma = \{1,2,3,4\}$

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

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 F



The set lattice for
Σ = {1,2,3,4}

σ = 2
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, $Occ(X \cup \{a\}) = \{ t \in Occ(X) : a \in t \}$

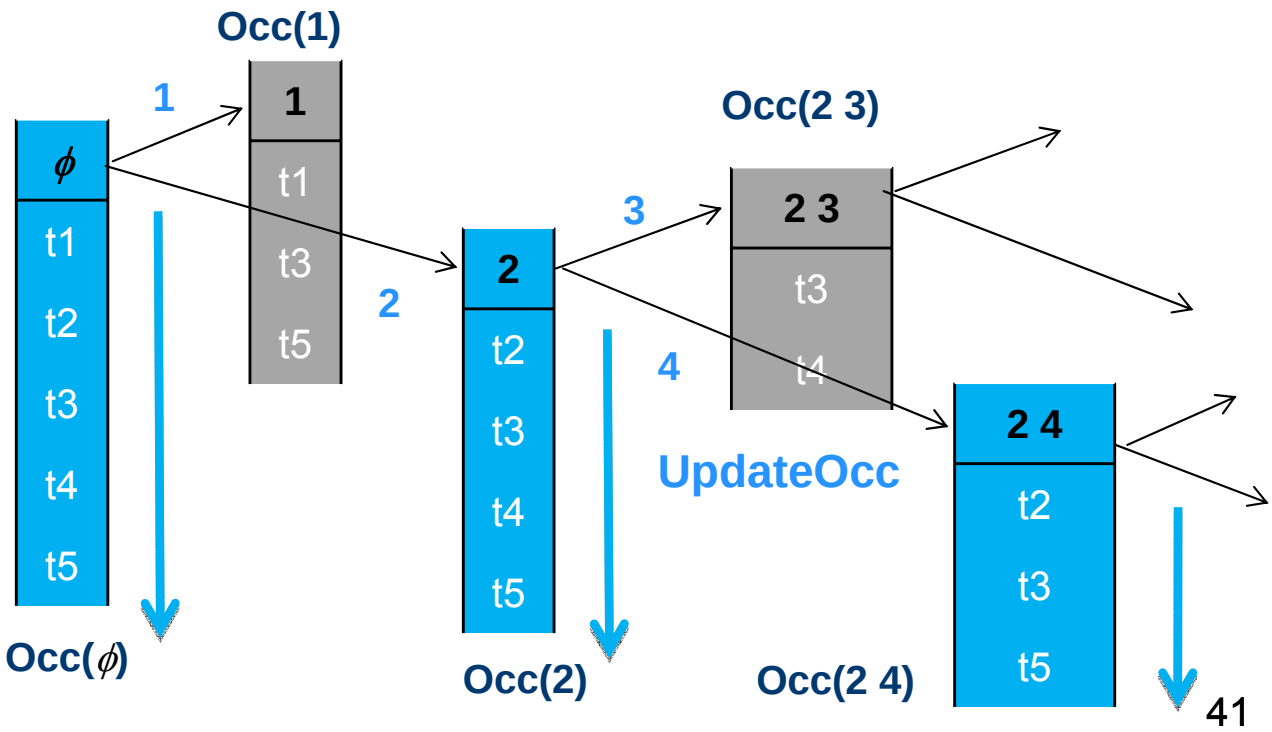
database ***D***

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

Vertical layout: The occurrence list representation

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

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



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

- $Y = X \cup \{a\}$; $Occ(Y) = \varnothing$;
- **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

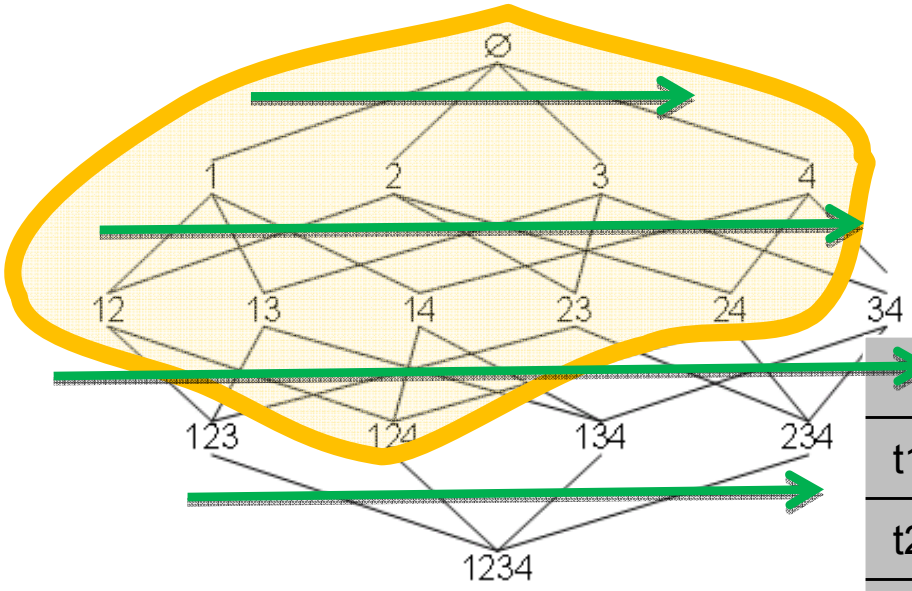
- The algorithm **Backtrack** can be implemented to enumerates all σ-frequent itemsets **X** in a given transaction database ***T***
  - in $O(l \cdot |Occ(X)|)$ time per frequent itemset
  - using $O(l \cdot |Occ(X)|)$ space
- where l is the maximum length of tuples in ***T*** and $|Occ(X)|$ is the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

Summary: BFS vs. DFS

Apriori algorithm [1994]

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



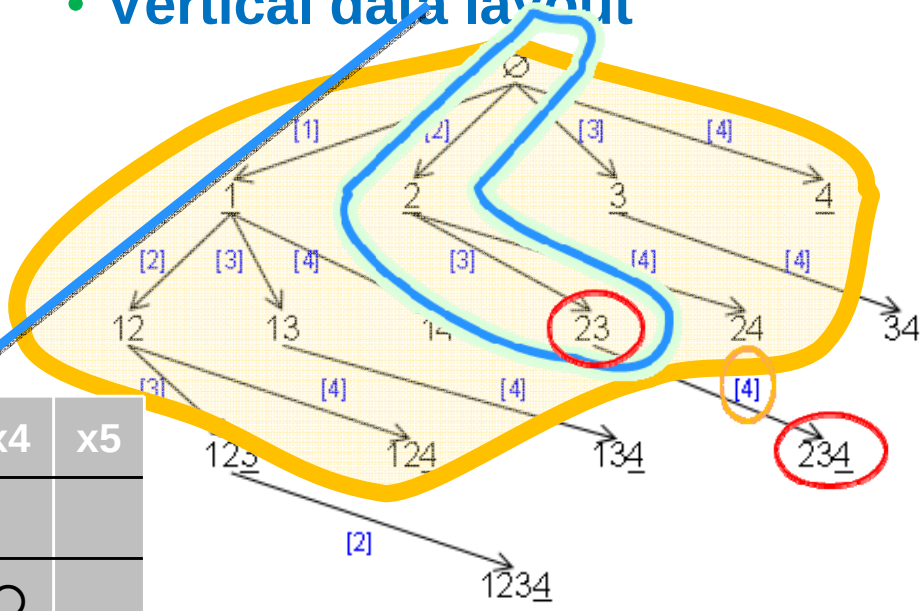
- 1st generation
- External memory algorithm

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

database

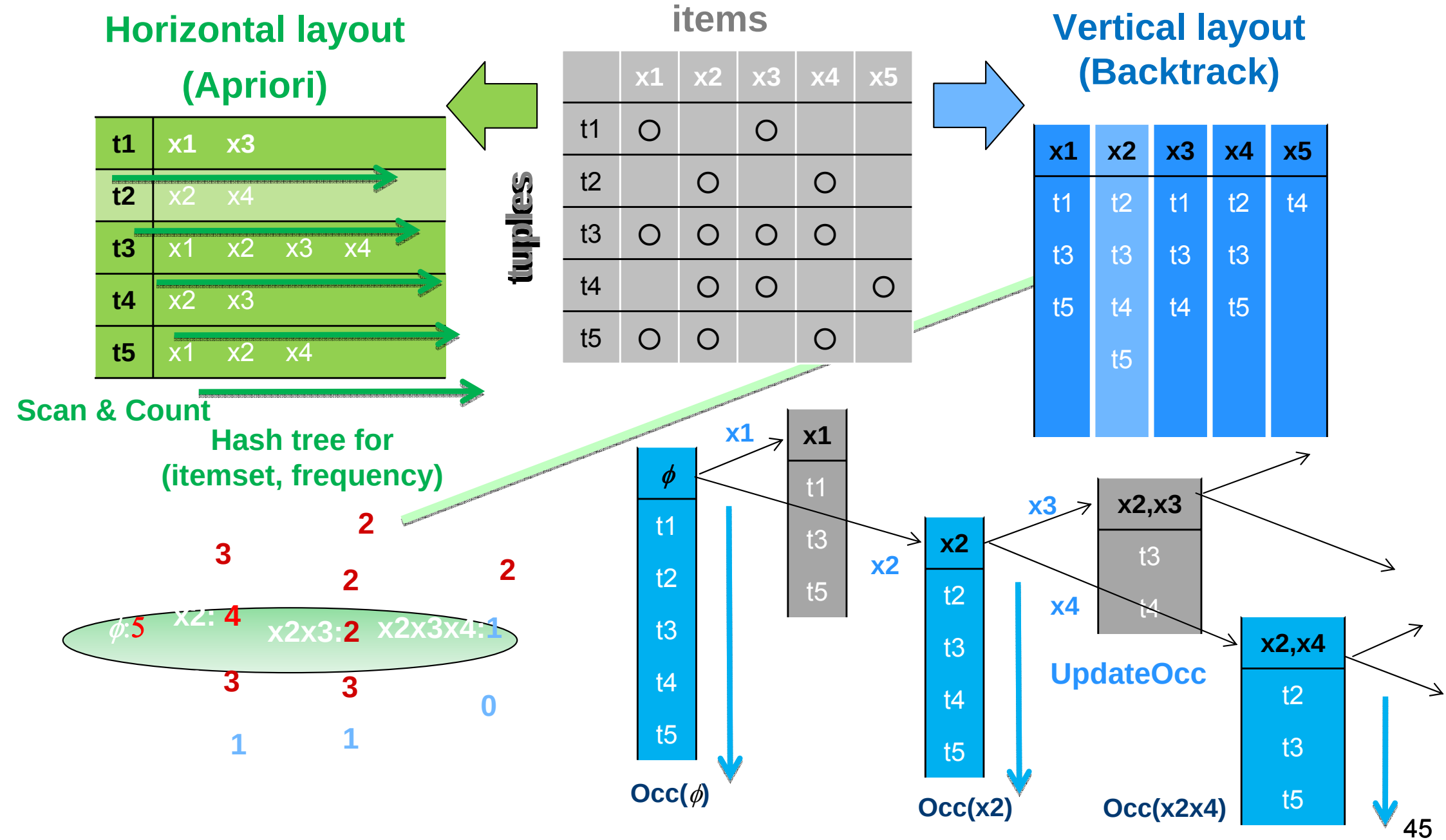
Backtrack algorithm [1997-1998]

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



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

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

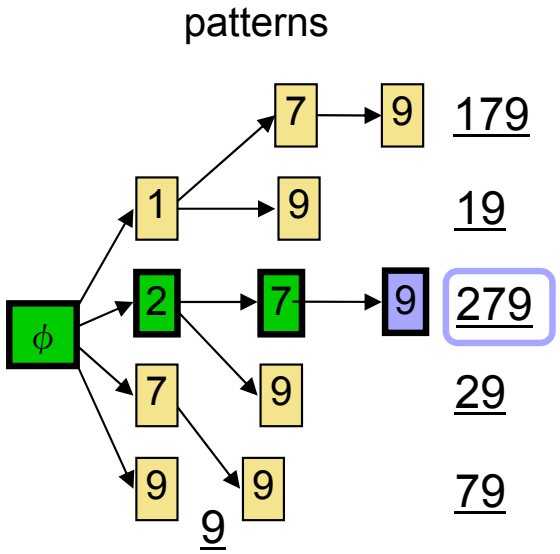
Φ

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

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

{2,9} {7,9}

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



データベース

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

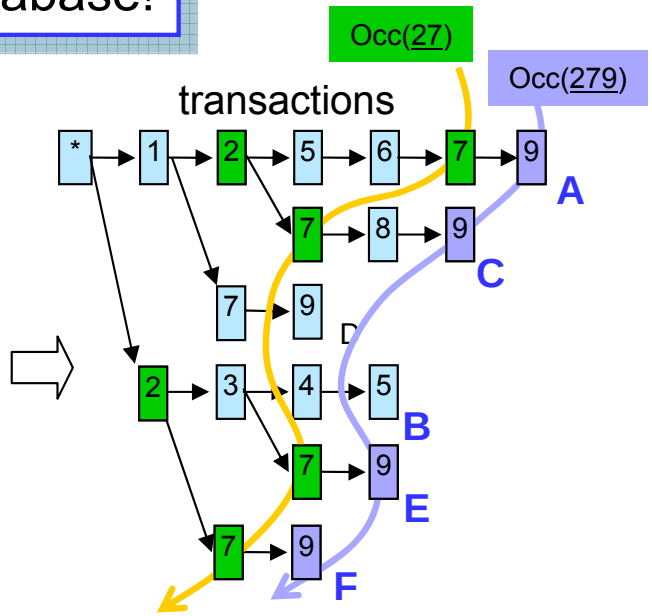
B: 2,3,4,5

C: 1,2,7,8,9

D: 1,7,9

E: 2,3,7,9

F: 2,7,9



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

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)

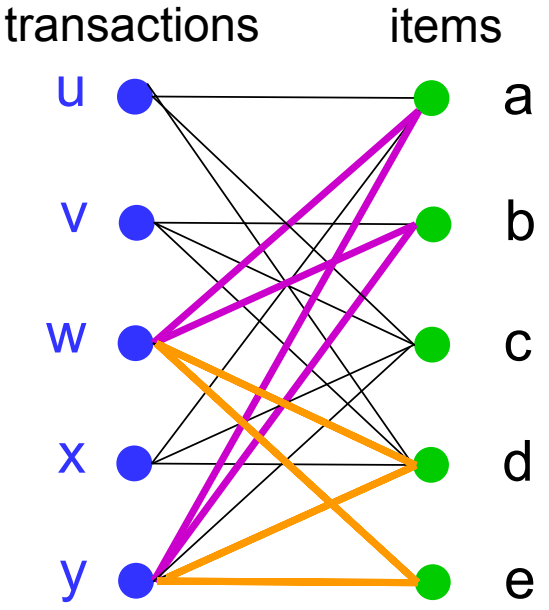
第2回 FIMI Workshop で宇野・清見・有村組(LCM ver.2) が優勝！
FIMI'04 Best Implementation Award

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/

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

パート1：データマイニングと頻出集合発見

- [AIS 93] R. Agrawal, T. Imielinski, A. N. Swami: Mining Association Rules between Sets of Items in Large Databases, Proc. SIGMOD Conference 1993, pp. 207-216, 1993.
- [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.
- [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)
- [Bayardo, SIGMOD'98] R. J. Bayardo Jr.: Efficiently Mining Long Patterns from Databases, Proc. SIGMOD Conference 1998: pp. 85-93, 1998.
- [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.
- [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 ― 頻出アイテム集合発見アルゴリズム ―」,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)

列挙学校：第3コマ

パート2: 系列とグラフのマイニング

有村 博紀
北海道大学大学院 情報科学研究科 コンピュータサイエンス専攻
email: {arim,kida}@ist.hokudai.ac.jp
http://www-ikn.ist.hokudai.ac.jp/~arim

.

まとめ

パート1: データマイニングと頻出集合発見

- データマイニング
- 頻出集合マイニング

パート2: 系列とグラフのマイニング

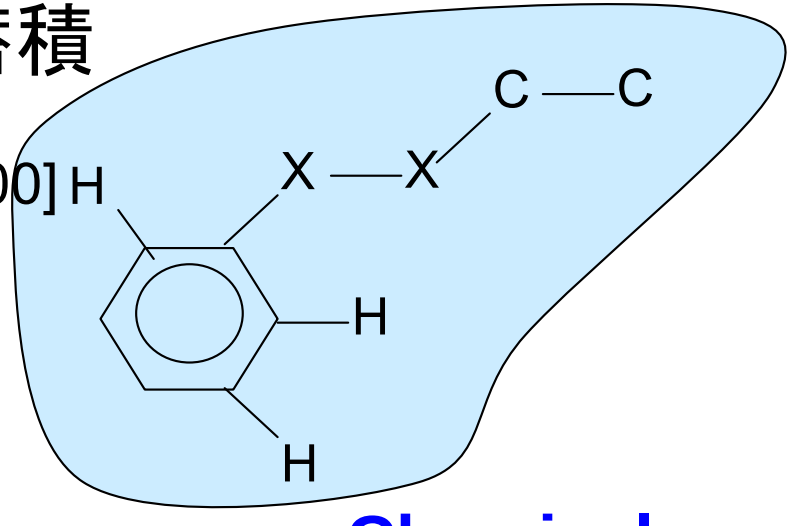
- 系列とグラフのマイニング概観
- 列挙によるパターンのマイニング
- 列挙とマイニングに関連した話題

Mining Semi-Structured Data

- Sequences
- Trees
- Graphs

半構造データ

- 大量のウェブページやXML文書が蓄積
- 半構造データ [Abiteboul, Buneman, Suciu 00]
  - ウェブページとXML [2nd ed, W3C 00],
  - ゲノムデータベース [Entrez, NCBI]
  - 化合物データや，CADデータ
  - 自然言語データも



Chemical compounds

Web access patterns

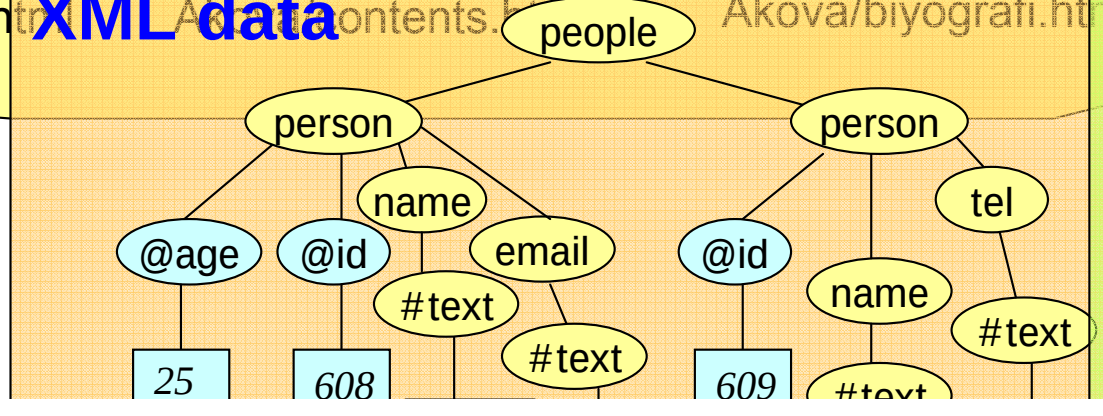
Path/sair_listesi.html

Path/poems/akgun_akova/index.html

Akova/picture.html

Akova/biyografi.html

XML data




```
<people>
  <person age="25" id="608">
    <name>John</name>
    <email>john@abc.com</email>
  </person>
  <person id="609">
    <name>Mary</name>
    <tel>555-4567</tel>
  </person>
</people>
```

Web pages

History of Sequence Mining

~1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008

HMM (Hidden Markov Model)[*1]
[Haussler, Krogh et al., 26th HICSS 1993]
[Durvin, Eddy, Krogh, Mitchison, Cambridge, 1998]

Mining frequent episodes in an event sequence (BFS)
[Mannila, Toivonen, Verkamo, DMKD, 1(3), 1997]

MOTIF program
[Smith, Annau *et al*., PNAS, **87**,1990]

PRATT program
[Jonassen et al., Protein Sci. **4**, 1995]

TEIRESIAS program
[Rigoutsos, Floratos, Bioinformatics **14**, 1998]

Mining frequent itemset sequence (DFS)
Spade [Zaki, 1998; Mach. Learn. 2000]

eMOTIF program
[Nevill-Manning et al., PNAS **95**, 1998]

Mining frequent sequences (DFS + Projected DB)
PrefixSpan [Pei, Han, et al., ICDE 2001]

Mining frequent closed frequent sequences (DFS)
CloSpan [Yan, Han, et al., SDM 2003]

PROJECTION (Random projection)
[Buhler, Tompa, J. Comp. Bio. **9(2)**, 2002]

Mining frequent closed sequences (DFS)
BIDE [Wang & Han, ICDE 2004]

Mining frequent closed sequeces with wildcards (DFS)
AU [Arimura, Uno, LNCS 3827, ISAAC2005]

a PTAS algorithm for concensus motif problem in Hamming distance
[Li, Ma, Wang, STOC 1999]

Mining frequent closed sequeces with gaps (DFS)
AU [Arimura, Uno, LNAI 4914, LLLL2007]

54

History of Tree & Graph Mining

~1995 Algorithm for finding subgraphs by MDL principle
Subdue [Holder *et al*. (KDD'94)]

1996

1997 Finding frequent paths [Wang and Liu (KDD'97)]

1998 Finding Semi-structured Schema
[Nestrov, Abiteboul *et al*. (SIGMOD'98)]

1999

2000 Finding frequent subgraphs
AGM [Inokuchi, Wahio, Motoda (PKDD'00, MLJ. 2003)]

FSG [Kuramochi *et al*. (ICDM'01)

2001

Finding frequent / optimal ordered trees
FREQT [Asai et al. (SDM'02)], **Treeminer** [Zaki (KDD'02)]

2002 Finding frequent subgraphs
gSpan [Yan and Han (ICDM'02)], **VG** [Venetik, *et al*.(ICDM'02)]

2003

Finding frequent unordered trees
UNOT [Asai, Uno, Nakano, Arimura (SDM'03)], **NK** [Nijssen, Kok (MGTS'03)]

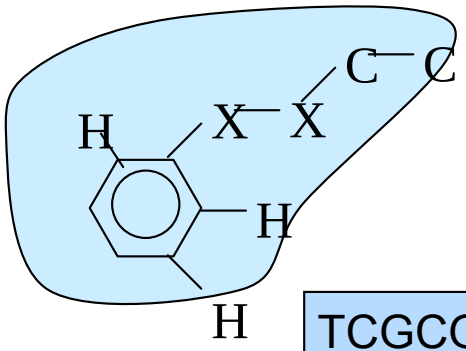
Frequent Maximal/Closed Trees & Graphs mining
[Yan&Han '03; Termier et al.'04] and many algorithms in 2000s

プラン

- ここでは，時間が許す限り，系列，木，グラフのパターンに関する具体的なデータマイニングアルゴリズムと最近の話題を紹介する予定
- 機械学習で用いられるカーネル法と列挙に関する話題にも触れたい

Learning Graphs from examples

- unknown function f : **Graphs** → {+1, –1}



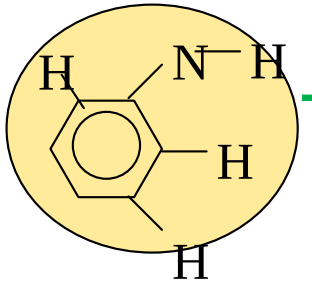
TCGCGAGGT –1

TCGCGAGGCTAGCT –1

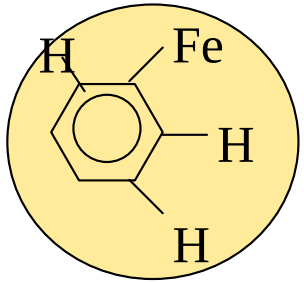
TCGCGAGGCTAT –1

negative examples

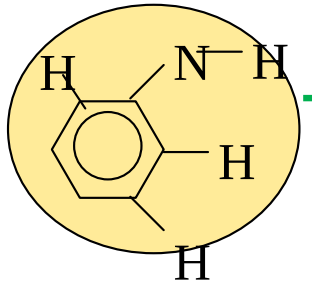
Classify



+1



+1



+1

GCAGAGTAT +1

TCGCGAGGCTAT +1

positive examples

Kernel methods and Enumeration

- Kernel methods for learning structured data
- #P-hardness and enumeration
- Hardness of learning and prediction

まとめ

パート1: データマイニングと頻出集合発見

- データマイニング
- 頻出集合マイニング

パート2: 系列とグラフのマイニング

- 系列とグラフのマイニング概観
- 列挙によるパターンのマイニング
- 列挙とマイニングに関連した話題

最新版PPTは次からダウンロードしてください: http://www-ikn.ist.hokudai.ac.jp/~arim/jtalks.html