IKN Special Lecture /情報知識ネットワーク特論 Data Mining 2: Frequent Itemset Mining: Breadth-first search algorithms (APRIORI)

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

Today's talk

Lecture 2: Frequent Itemset Mining: BFS algorithms

- Frequent Itemset Mining
- Breadth-first search algorithms:
  Apriori algorithm (Agrawal et al., VLDB 1994)

Points

- Set lattice
- Properties of frequent item sets
- Breadth-first search (BFS) of a graph
- External memory algorithms

今日の内容

2回: 頻出集合発見:深さ優先探索

- データマイニング
- 頻出集合マイニング
- 幅優先探索アルゴリズム
  （Agrawal の Apriori アルゴリズム）

ポイント

- 集合束
- 頻出集合発見問題
- 幅優先探索アルゴリズム
- 高速化の工夫（外部記憶アルゴリズム）

Backgrounds

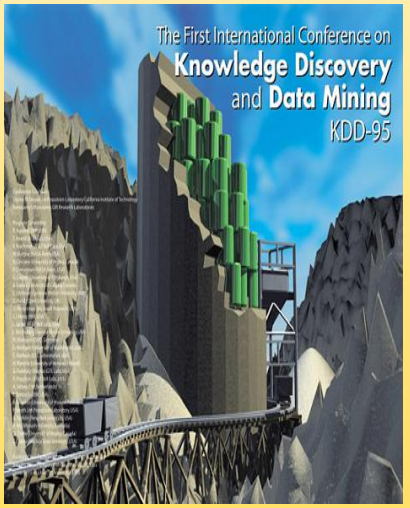
Data Mining

- A formal study of **efficient methods** for extracting **interesting rules and patterns** from massive data
- Emerged in the mid 90's, and growing in 2000
  - Apriori algorithm [Agrawal, Srikant, VLDB1994]
- Interdesciplinary area of
  - *Machine Learning, Statistics, and Data Engineering*
- Emphasis on efficient processing of massive data at first, and then quality of discovery/mining

Backgrounds

The whole process of Data Mining

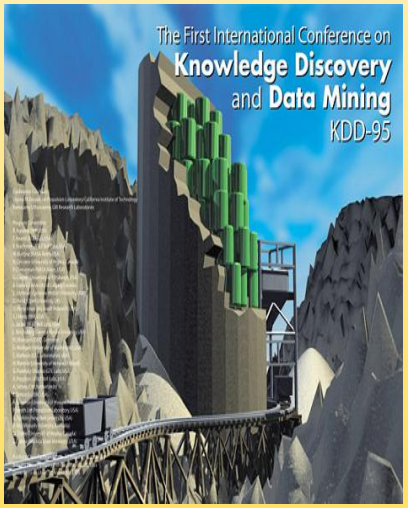
- 1. Understanding expert domain
- 2. Preprocessing of data sets
- 3. Pattern Mining/Discovery
  （Data mining in narrow sense）
- 4. Analysis of Patterns
- 5. Use of Analysis/Results



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



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

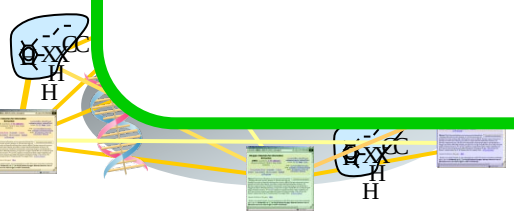
- 不完全なデータから，未知の規則を学習する
- SVM [Vapnik ‘96],
- Boosting [Shapire & Kearns ‘96]
- C4.5 [Quinlan ‘96]

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
  - 近似統計処理

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

今日の練習問題

問題（集合束）:

- S = {1, 2, 3, 4} を集合とする．半順序集合$(2^S, \subseteq)$のハッセ図を書け．
  ここに，2^SはSの部分集合すべての族である．
  ただし，簡便のため，集合{1, 2, 3}を 123 と書いてよい．
- Write the "Hasse diagram" of the partially ordered set $(2^S, \subseteq)$ defined by the family $\mathbf{2^S}$ of all subsets of a given set S = {1, 2, 3, 4}.

用語

- 半順序集合 (Partially ordered set):
  A pair (**B**, ≦) of a family **B** of objects and a partial order ≦ over **B**.
- 集合束（Set lattice）:
  集合Sのすべての部分集合からなる半順序集合$(2^S, \subseteq)$をいう．ここで⊆は集合の包含関係．厳密には束$(2^S, \cap, \cup)$のこと．
- ハッセ図 (Hasse diagram):
  Sの部分集合すべてを，包含関係⊆で上方に小さいもの（下方に大きなもの）が下に来るように書き，⊆に関して直接の親子関係（包含関係）にあるものを線で結んだもの．

Today's Exercise

Write your student ID, full name, year (e.g. MC1), Labo name, today's date (e.g. YYMMDD)

Let $S_4 = \{1, 2, 3, 4\}$ be a set of n = 4 elements. Answer the following questions.

1.Is the empty set ∅ a subset of the set S_4? (Yes or No)

2.Write all subsets of S_4. How many subsets are there?

3.(Option) Write a program that receives a number $N \geq 0$, and then prints all subsets of $S_N = \{1, 2, \ldots, N\}$ without repetition.

- You can describe your program in any natural language (Japanese, English, …), or any of your favorite programming language such as Java, C, C++, Python, Ruby, Haskel, etc. (as long as I can understand it :-)

4.(Option) How many subsets of the N elements set $S_N = \{1, 2, \ldots, N\}$ in general?

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

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

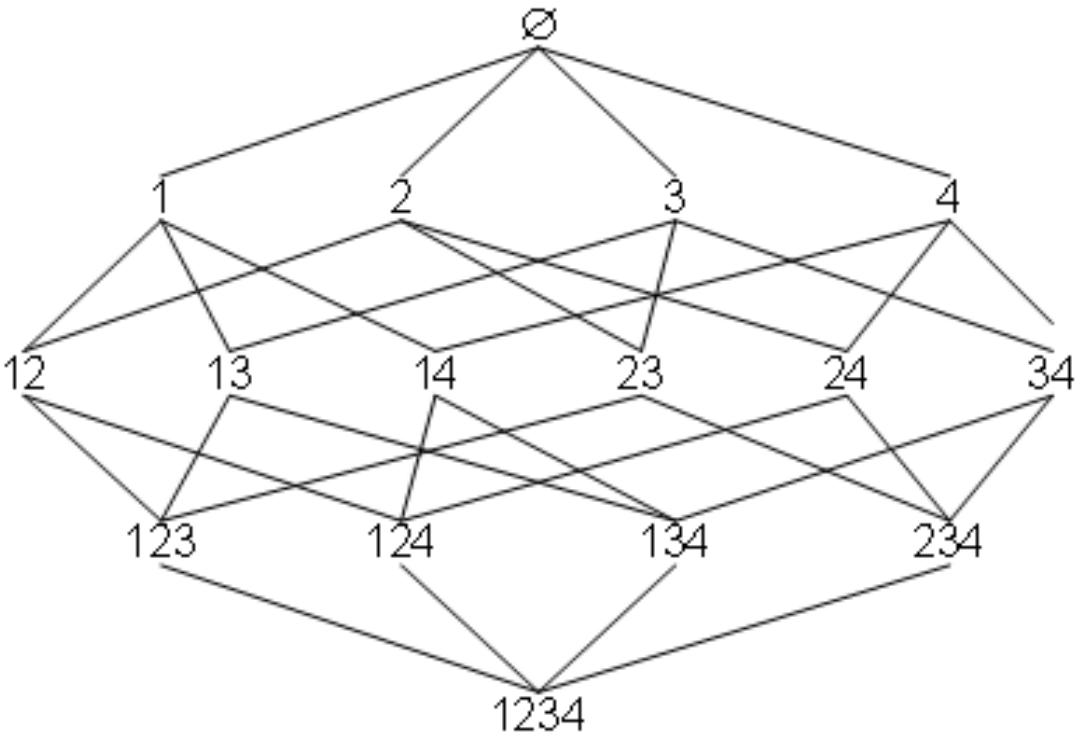
$I = \{1,2,3,4\}$

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

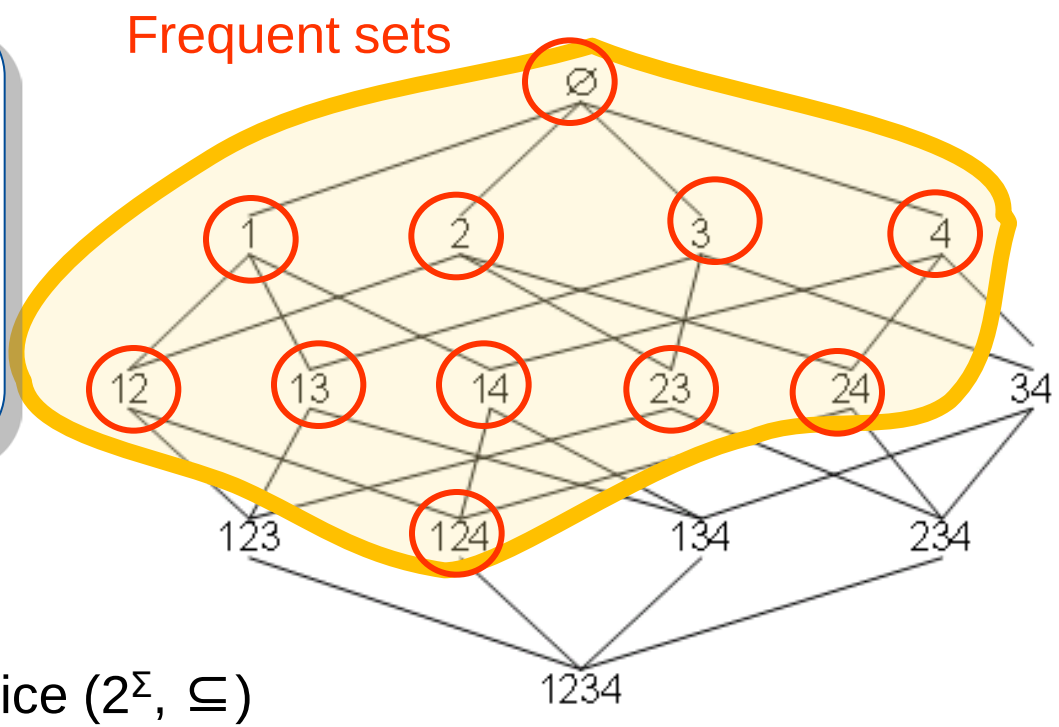
17

Definitions: Frequent sets

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

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

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.

Frequent Itemset Mining Algorithms

Apriori Algorithm (BFS algorithm)

Problems

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

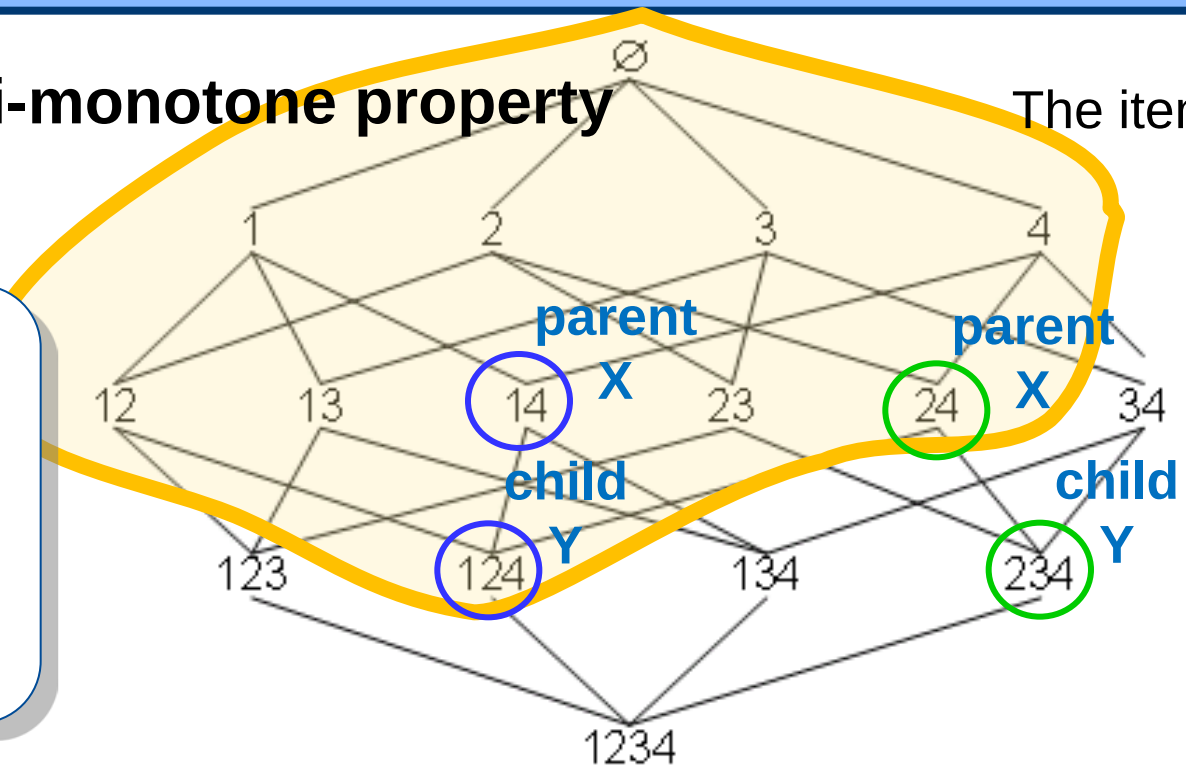
- **Lemma**: For every minsup σ, if **Y** is σ-frequent then any subset **X** of X is also σ-frequent in DB ***T.***
- **Corollary:** Every non-empty σ-frequent set **Y** is obtianed from some σ-frequent set **X** by adding some new element.
- The class $\mathcal{F}_\sigma$ of all frequent sets is monotone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

Anti-monotone property

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

minsupσ= **2**

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



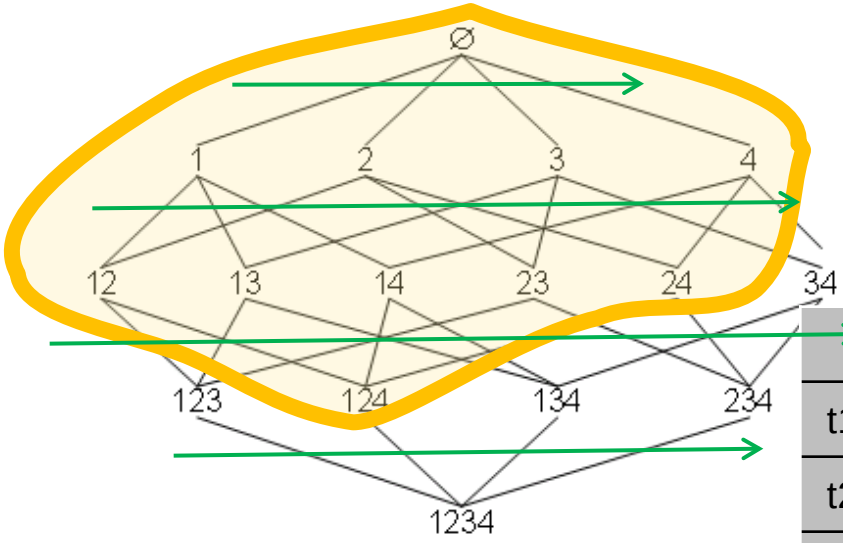
	1	2	3	4	5
t1	○		○		
t2		○		○	
t3	○	○	○	○	
t4		○	○		○
t5	○	○		○	

database

Two approaches to Frequent sets mining

Apriori algorithm [1994]

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



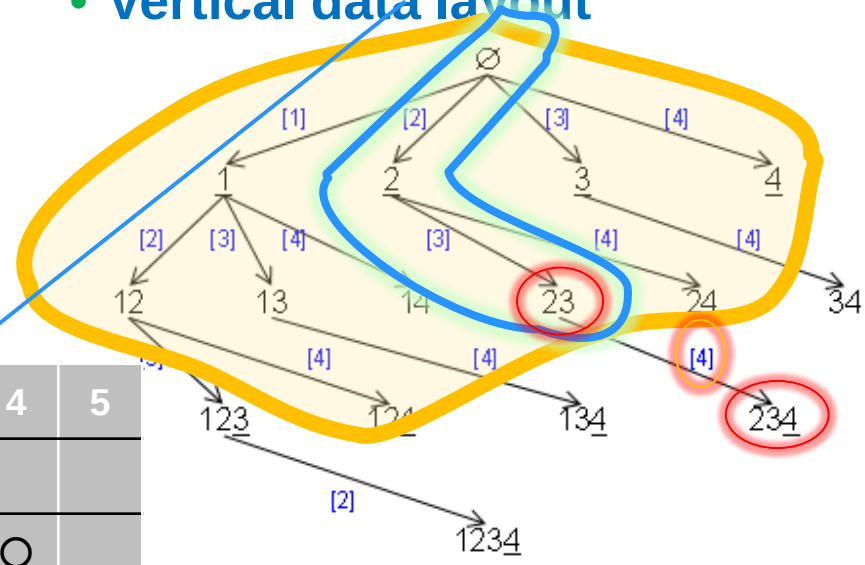
- 1st generation
- External memory algorithm

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

database

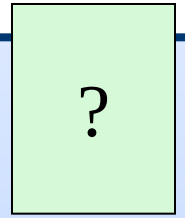
Backtrack algorithm [1997-1998]

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

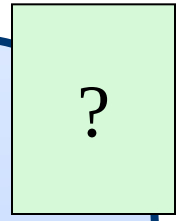


- 2nd generation
- In-core algorithm
- Space efficient

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)

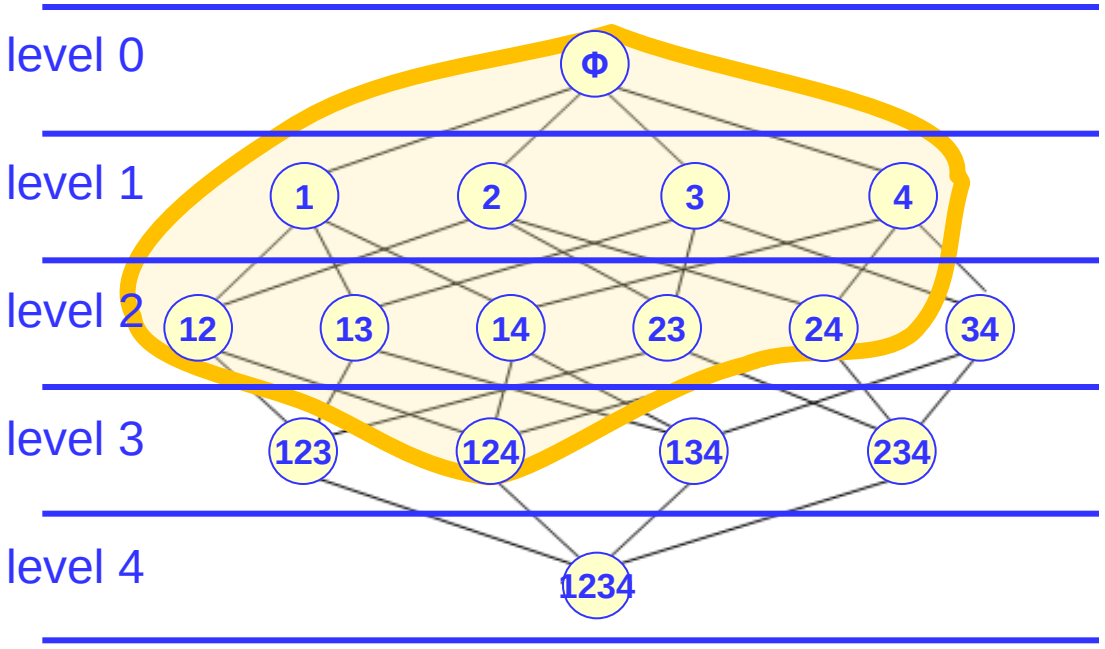
Frequency Counting: **Horizontal Data Layout**

- 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 $\boldsymbol{C}_i$



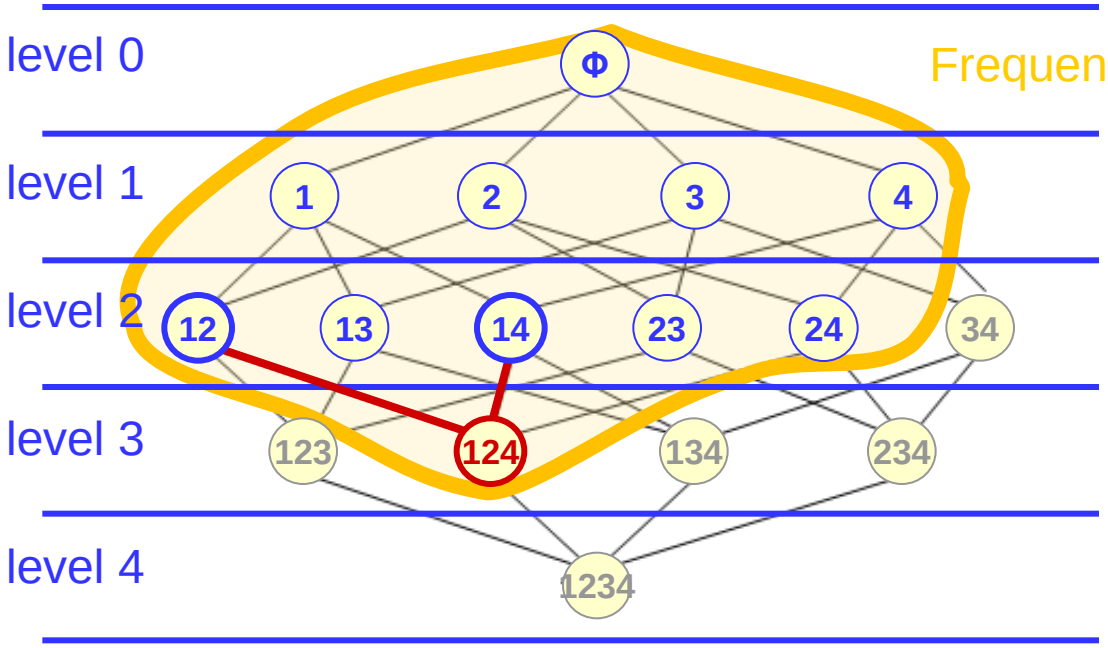
	1	2	3	4	5
t1	○		○		
t2		○		○	
t3	○	○	○	○	
t4		○	○		○
t5	○	○		○	

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	○	○		○	

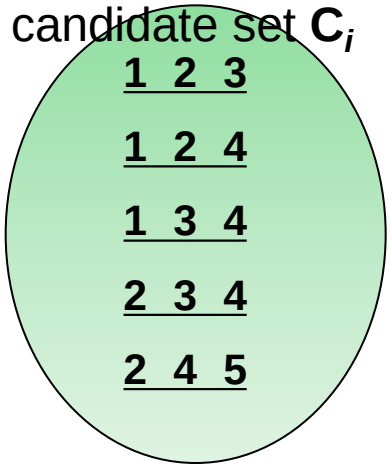
database

Basic Idea: Frequency Counting

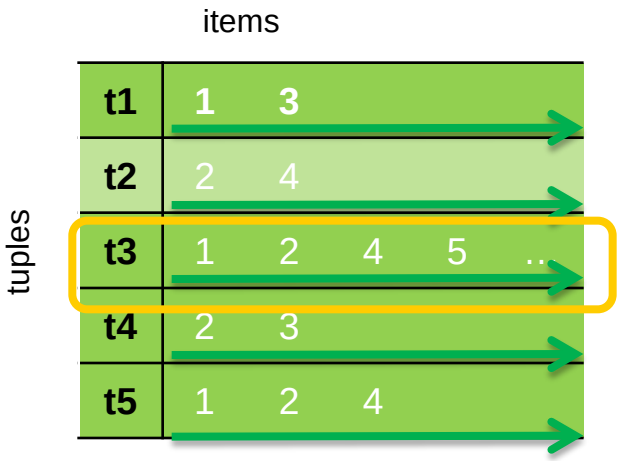
Horizontal layout

- 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

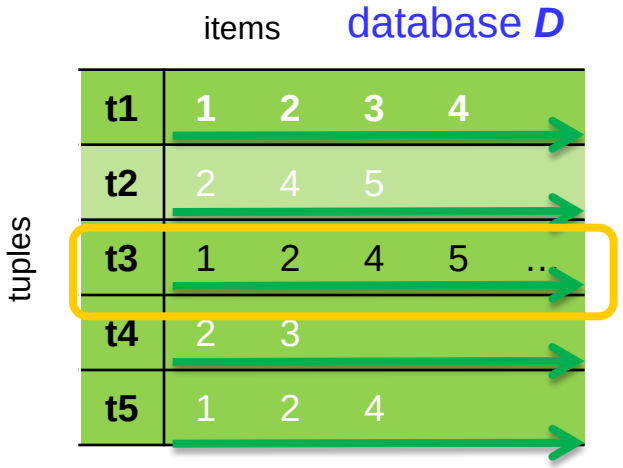
Horizontal layout

- 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*.

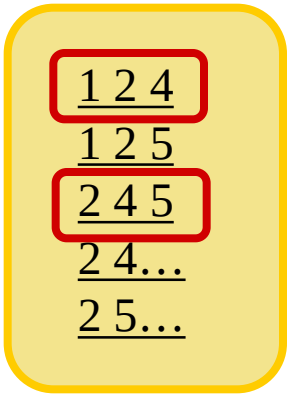
candidate set C_i

1 2 3
1 2 4
1 3 4
2 3 4
2 4 5

1) Scan the database sequentially tuple by tuple

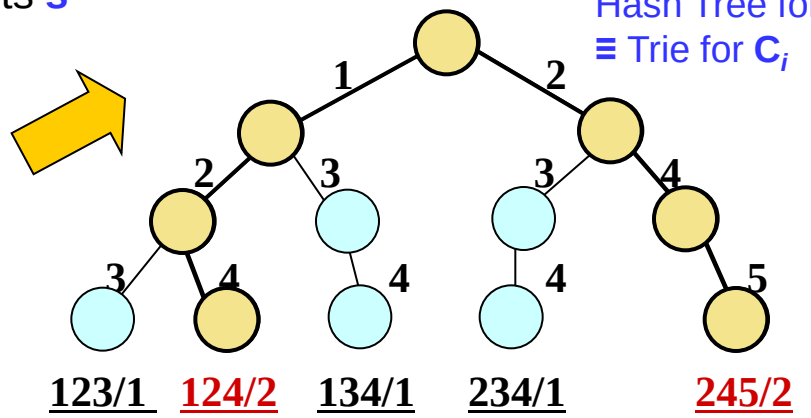


2) For each tuple *t*, enumerate all subsets *s* of *t* with size *k*



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

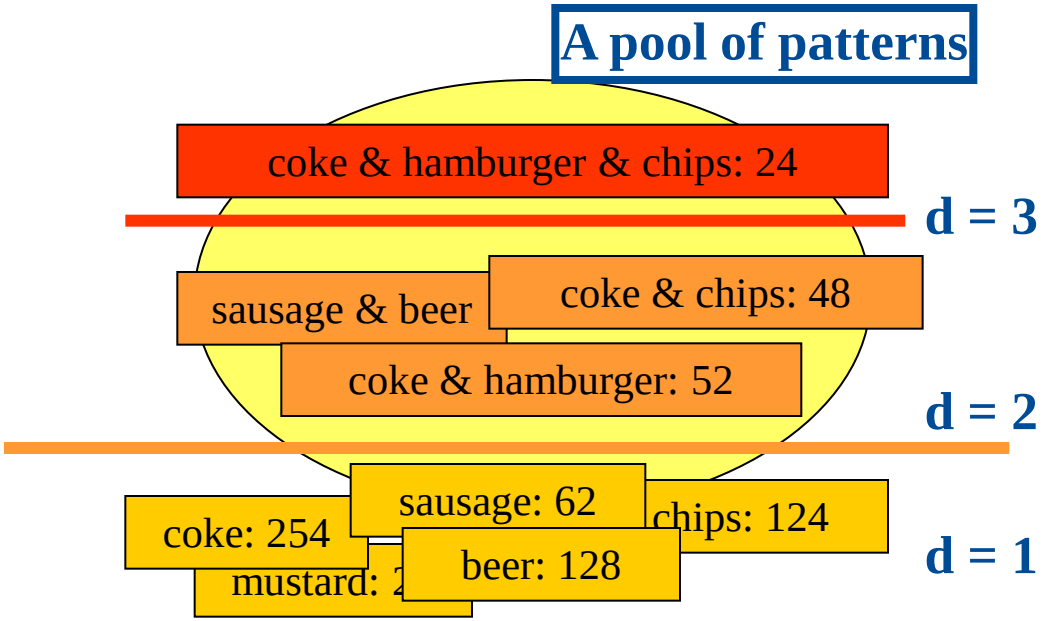
Fast Processor

+

Medium sized Main memory

+

Huge Hard disk



Levelwise search

Fast counting by Hash tree

Large collection of data

Sequential Disk Scan

まとめ

2回: 頻出集合発見:深さ優先探索

- 頻出集合マイニング
- 幅優先探索アルゴリズム
  - Aprioriアルゴリズム

ポイント

- 頻出集合の単調性
- 幅優先探索(レベル探索アルゴリズム)
- 頻度計算の高速化(外部記憶アルゴリズム)