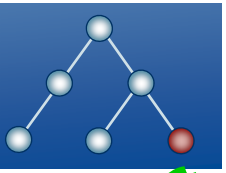
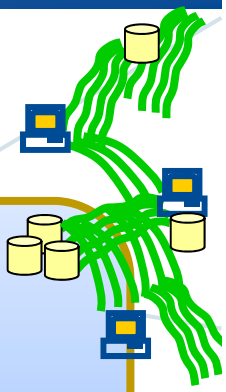




情報知識ネットワーク特論
Data Mining 6:
Stream Mining algorithms

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/ikn-tokuron/
http://www-ikn.ist.hokudai.ac.jp/~arim

今日の内容

6回:ストリームマイニング

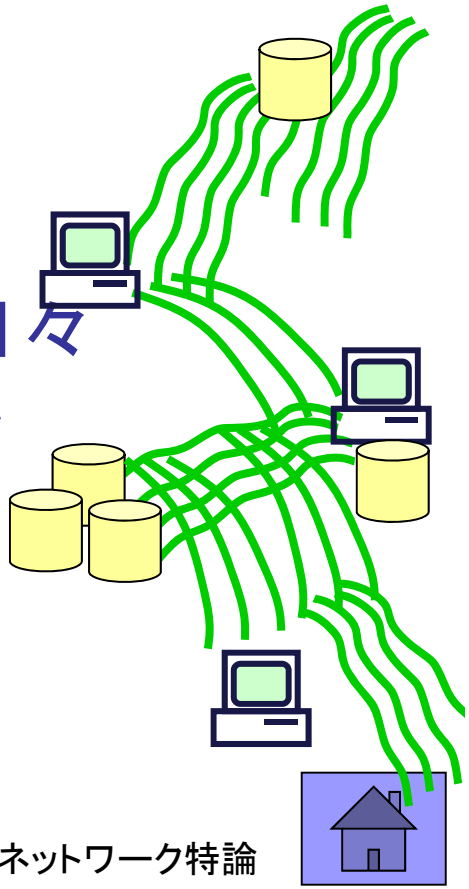
- ストリームマイニングとは
- ストリームに対する近似統計手法
- 近似カウンティング
- ストリームからのアイテム集合発見
- 半構造データストリーム

ポイント

- 超低メモリアルゴリズム

データストリームとは

- 背景
  - インターネットに代表される高速なネットワークと大規模センシング技術の発展
- データストリーム
  - 新しい大規模データ
  - 時間的に変化する大量のデータが日々刻々と生成・集積・消費される様子をデータの流れとしてとらえたもの



ストリームデータの例

- ネットワークの解析データ
- DARPA IDS Evaluation DataSet
  (http://www.ll.mit.edu/IST/ideval/)

(service, host_ip, dist_ip)

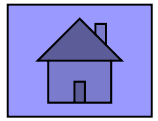
(telnet, 192.168.1.30, 192.168.0.20)
(ftp, 192.168.1.30, 192.168.0.20)
(smtp, 192.168.0.40, 192.168.1.30)
(auth,
(smtp,
(shell,
(sunrpc,
(ftp-data
(ftp-data
(telnet,
(ftp-data
(finger, 192.168.1.30, 192.168.0.20)
(smtp, 192.168.1.30, 192.168.0.20)
(smtp, 192.168.1.30, 192.168.0.20)
(http, 192.168.1.30, 192.168.0.40)

大きなデータ

- ドメインサイズ U
  数$10 \times 2^{32} \times 2^{32}$（個）
- データサイズ N
  300万個（100MB）

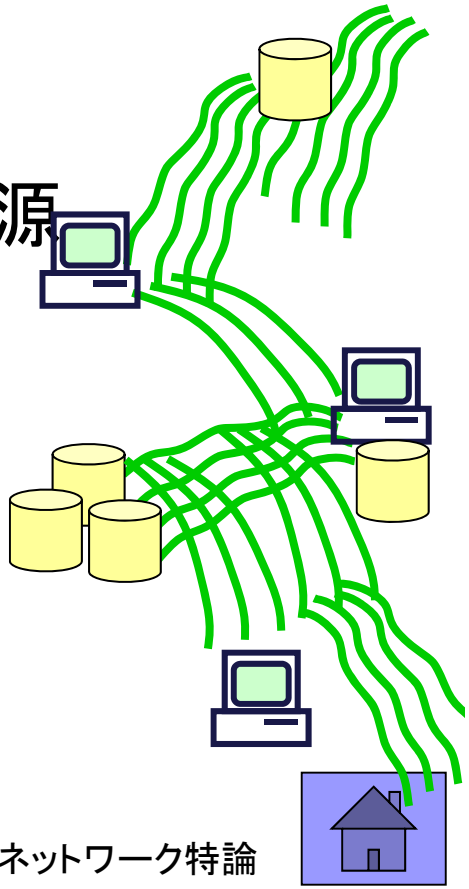
小さな答え

- 異なりアイテム数 D
  64,636個
- 頻出アイテム（0.1%）D
  24個



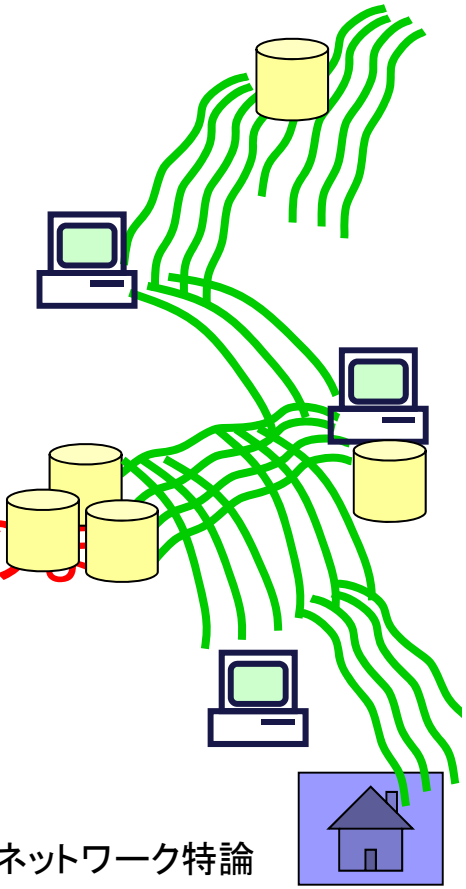
データストリームの例

- 金融や流通分野における取引記録（トランザクション・ログ）
- 電話会社・Webサービスプロバイダの通信記録（call records, access log）
- センサー・ネットワークや，オンラインニュース，経済情報など多数の情報源から時系列的に生成されるデータ



データストリームマイニング

- 通信・流通分野での要求
  - 大規模データストリームから，いつでも望むときに必要な情報を取り出したい
- データストリームの特性
  - 膨大な量のデータが，(massive)
  - 高速なストリームを通じて，(high-speed)
  - 時間的に変化しながら (transient)
  - 終わりなく到着し続ける (unlimited)
- 従来のデータマイニング技術は，そのまま適用できない!

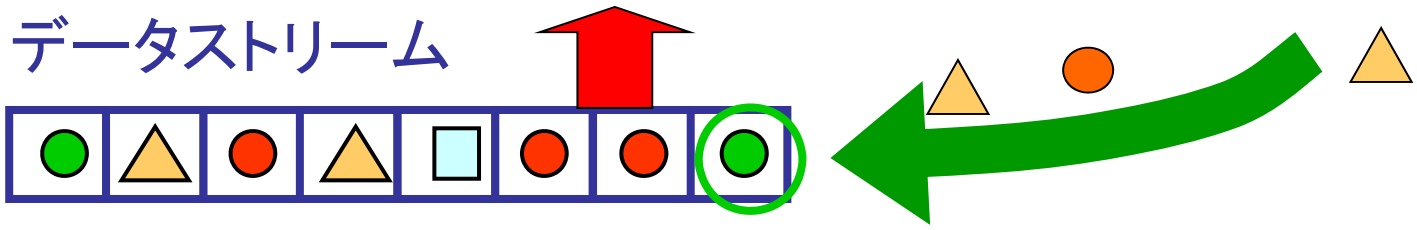


データストリーム処理研究の歴史

- 1980年代以前
  - 統計量の外部記憶計算や低メモリデータ構造の研究
- 1990年代
  - 通信分野でのストリームを対象とした統計処理システム
  - 計算量分野でのストリームアルゴリズムの実証的研究
  - データベース分野でのストリームに対する連続問い合わせや，集約計算の研究
  - 次第に注目される
- 2000年代：ストリームデータマイニング
  - データストリームを対象としたデータマイニング・機械学習の研究が盛んに．情報検索や，時系列モデリング分野．
- 古くて新しい技術
  - 本来，データマイニングは，基幹業務系システムで，日常的なトランザクション処理の履歴データを，データ倉庫に集約・格納し，意思決定支援に有効活用することから始まった

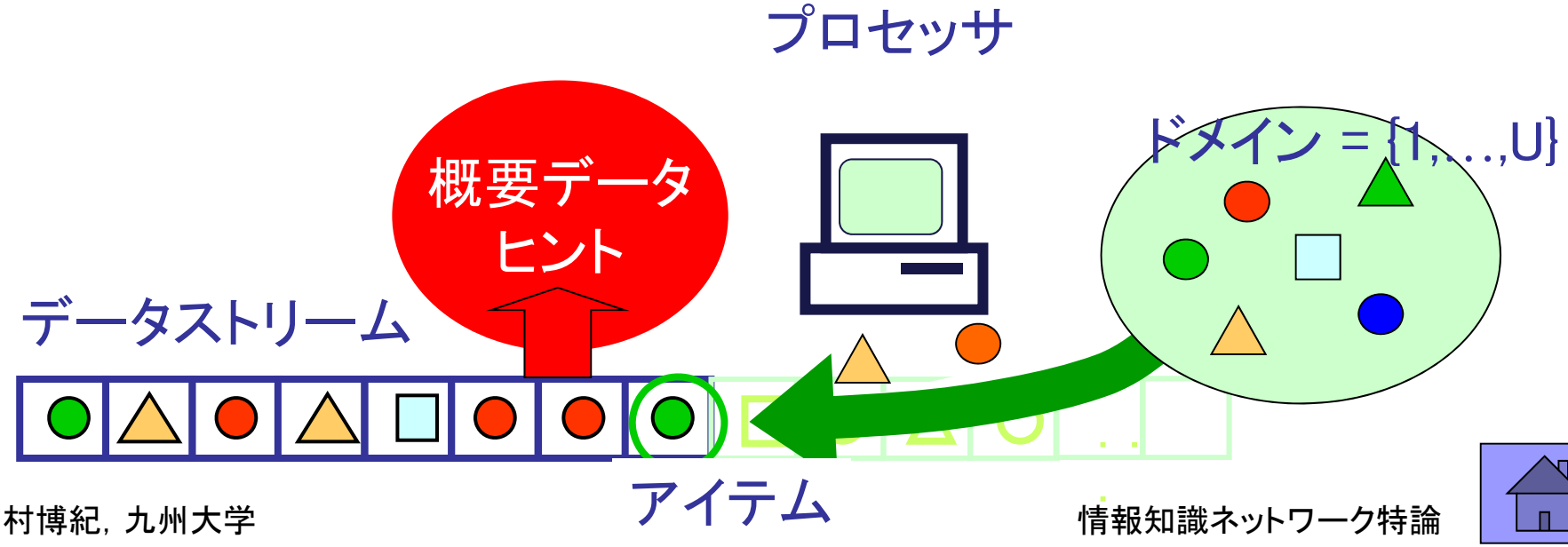
ストリームマイニングの仕事

- データストリームに対して
  - アイテム／属性値に対する統計をとる
  - 特徴的なパターンを発見する
  - 分類ルールを構築する／予測する
  - 複数のストリームの相関を求める
  - トレンドを検出する



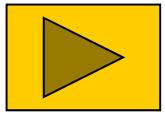
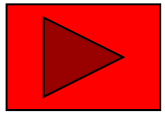
ストリームデータ処理

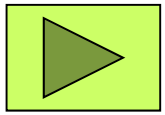
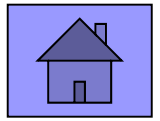
- 膨大で高速なデータストリームから，
- 時間とともに変化するパターンや規則を発見・抽出し，（マイニングの仕事のとき）
- 限定された計算資源を用いて働き続ける
- ただし，近似的な解でよい



アウトライン

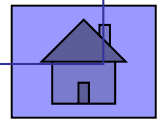
- ストリームデータ
- 近似カウンティング
  - Manku & Motwani (VLDB'02)
- ストリームからのパターン発見
  - Hidber (SIGMOD'99)
  - 浅井, 有村, 有川 (ICDM'02)
- ストリームに対する近似統計手法
  - ストリーム統計のいろいろ（Alon, Matias, Szegedy, STOC'96）
  - 頻度モーメントの計算手法
- まとめ

関連研究: SIGKDD 2003他から

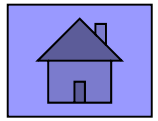
- **Association Rules:** (Chang & Lee, SIGKDD'03)
  Finding recent frequent itemsets adaptively over online data streams
- **Decision Trees:** (Wang, Fan, P. S. Yu, J. Han, SIGKDD'03)
  Mining concept-drifting data streams using ensemble classifiers
- **Multiple streams:** (Zhu & Shasha, SIGKDD'03)
  Efficient elastic burst detection in data streams
- **Multiple streams:** (Babcock & C. Olston, SIGMOD'03)
  Distributed Top-K Monitoring
- **Multiple streams:** (Guha, Gunopulos, Koudas, SIGKDD'03)
  Correlating synchronous and asynchronous data streams
- **IDS:** (Yamanishi, Takeuchi, Williams, SIGKDD'02)
  Online Unsupervised outlier detection using finite mixtures with discounting learning algorithms
- **IDS:** (W. Lee & S. J. Stolfo, Usenix, Security, 1998)
  Data Mining Approaches for Intrusion Detection



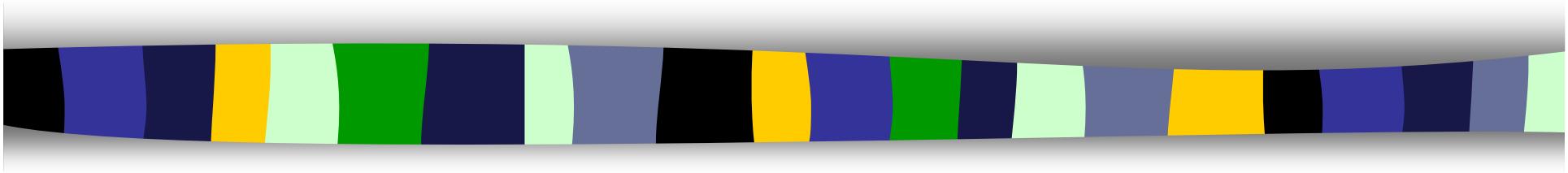
まとめ

- ストリームデータ
  - 高速なデータストリームから，時間変化しながら，連続して供給される，大量のデータ
- ストリームアルゴリズム
  - 膨大で高速なデータストリーム
  - 限定された計算資源を用いて働き続ける
  - 近似的な解でよい

- ストリームデータ
- ストリームに対する近似統計手法
  - ストリーム統計のいろいろ
  - 頻度モーメントの計算手法
- 近似カウンティング
  - Manku & Motwani (VLDB'02)
- ストリームからのアイテム集合発見
- 半構造データストリーム
  - 浅井，有村，有川 (ICDM'02)



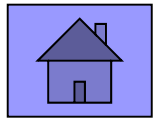
近似カウンティング



Approximate Frequency Counts over Data Streams,
(Manku, Motwani, Proc. VLDB'02, 2002)

A Simple Algorithm for Finding Frequent Elements in
(Streams and Bags, Karp, Papadimitriou, Shenker,
Manuscript, Feb. 2003)

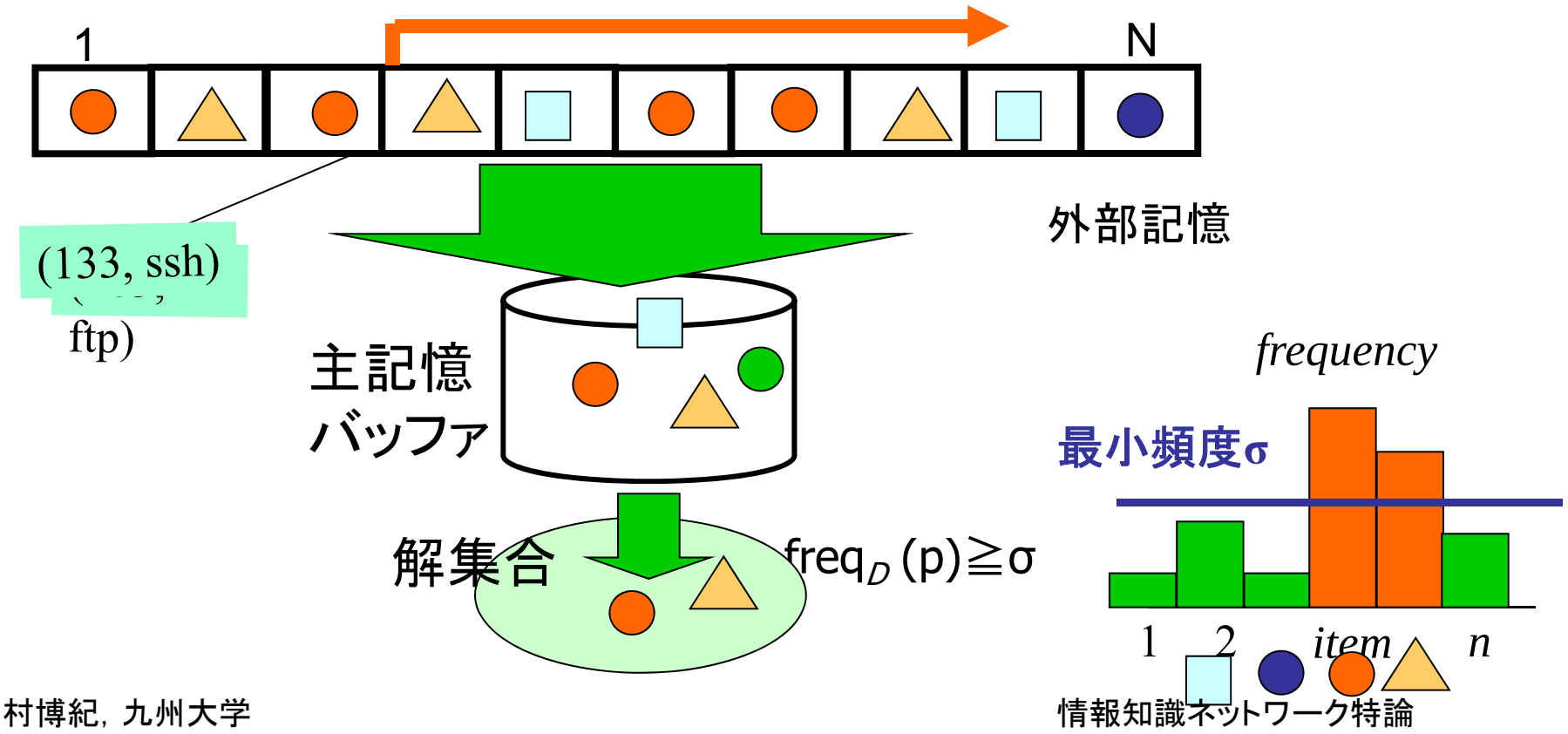
領域効率の良い頻出データアイテム発見アルゴリズム,
(川副, 浅井, 有村, DEWS'03, 2003)



頻出データアイテム発見問題

入力：長さNのアイテム配列 A，最小頻度値0≦σ≦1

問題：Aでの出現頻度がσ以上のすべての要素を見つけよ.



ストリームデータの例

- DARPA IDS Evaluation DataSet
  （http://www.ll.mit.edu/IST/ideval/）

(service, host_ip, dist_ip)

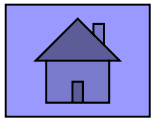
(telnet, 192.168.1.30, 192.168.0.20)
(ftp, 192.168.1.30, 192.168.0.20)
(smtp,
(auth,
(smtp,
(shell,
(sunrpc,
(ftp-data
(ftp-data
(telnet,
(ftp-data
(finger,
(smtp, 192.168.1.30, 192.168.0.20)
(smtp, 192.168.1.30, 192.168.0.20)
(http, 192.168.1.30, 192.168.0.40)

大きなデータ

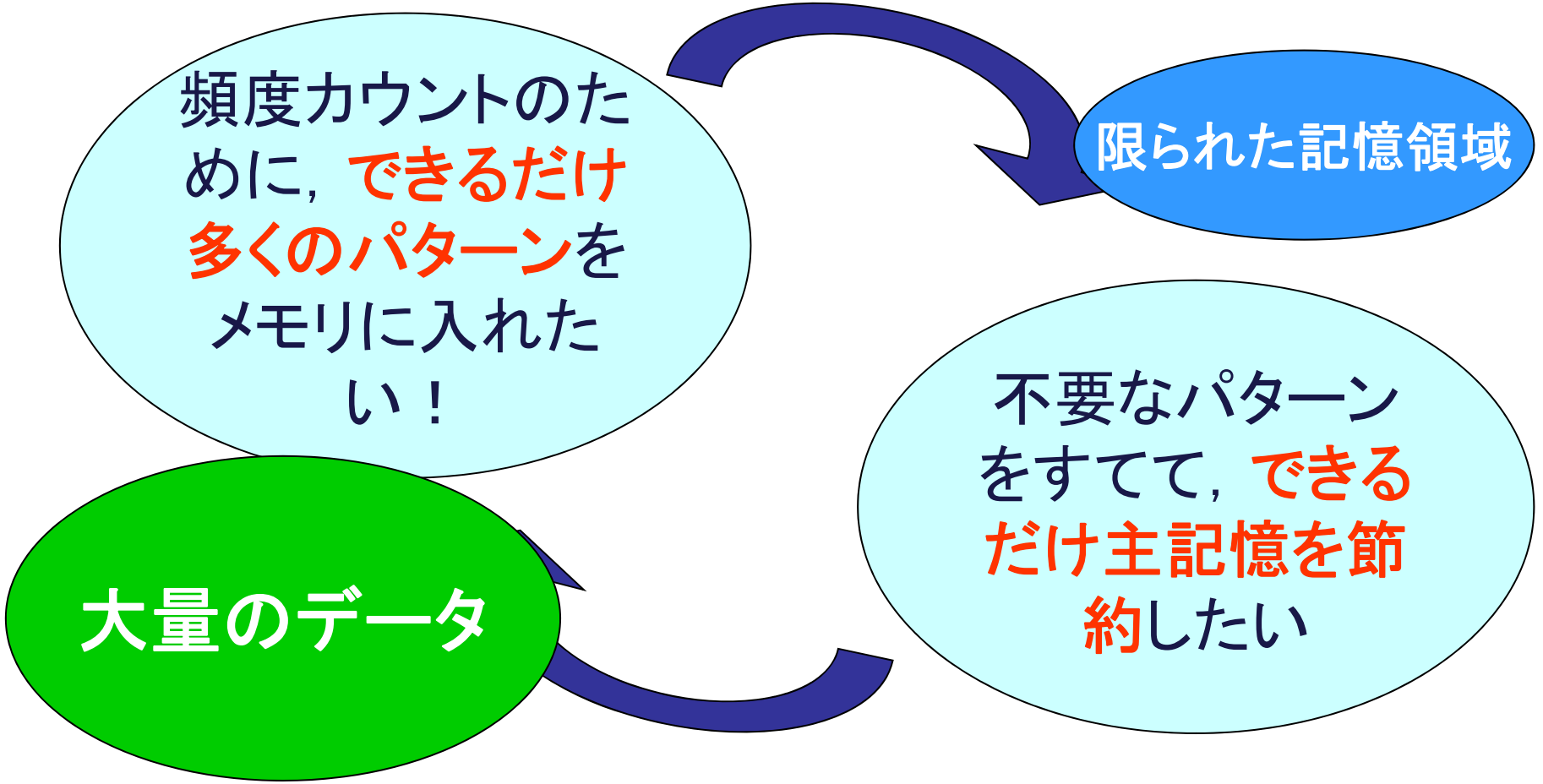
- ドメインサイズ U
  数$10 \times 2^{32} \times 2^{32}$（個）
- データサイズ N
  300万個（100MB）
- メモリサイズ M
  数MB

小さな答え

- 異なりアイテム数 D
  64,636個
- 頻出アイテム（0.1%）D
  24個



オンライン＆大規模データマイニングのジレンマ



オンラインアルゴリズム LossyCounting

(Manku & Motwani, VOLDB'02)

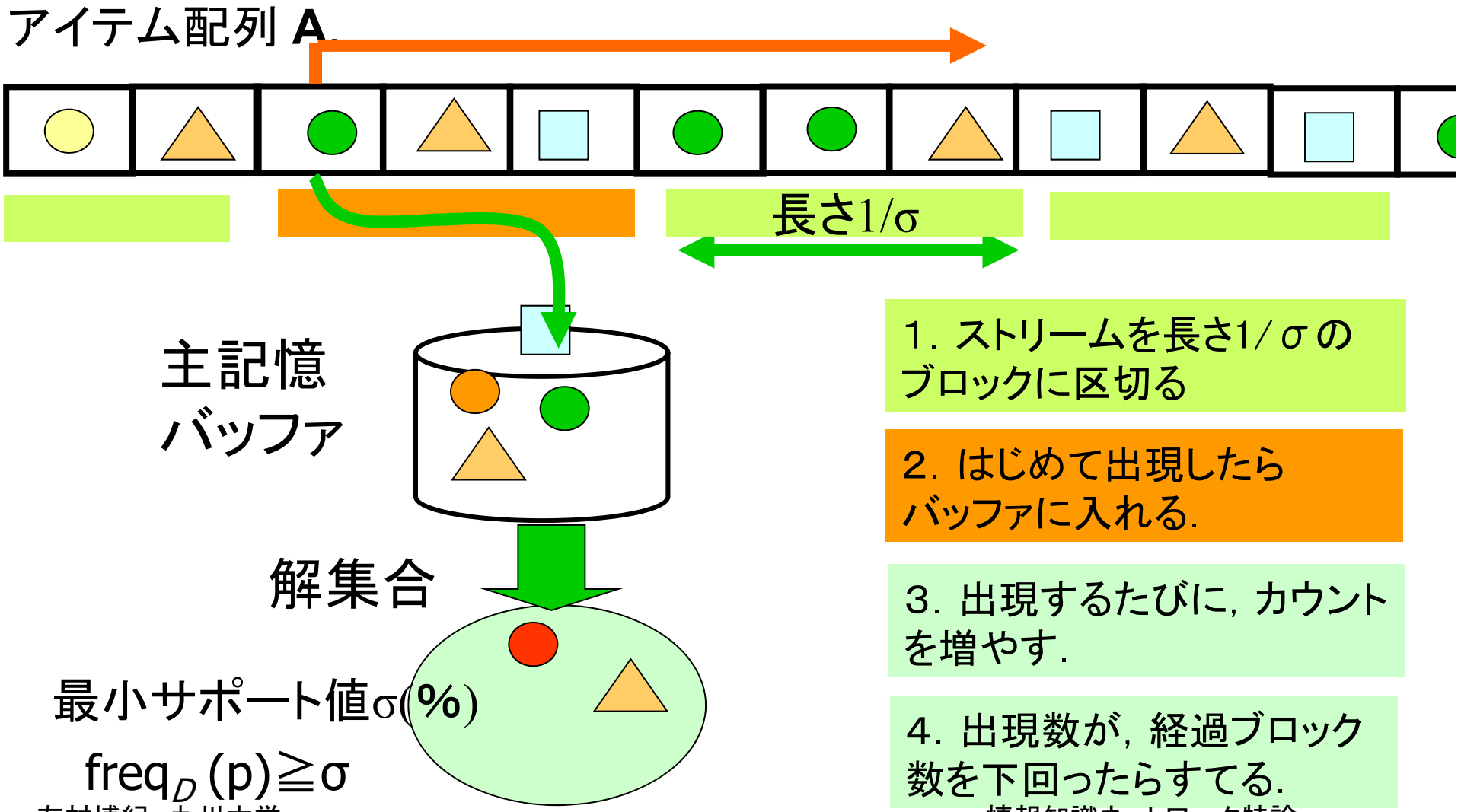
- はじめての1パス近似アルゴリズム
- 最小頻度σとサイズNのデータに対して，すべての頻出アイテムを，O(1/σlog N)領域で出力する.
- 近似:いくつかの非頻出アイテムも出力する

- 基本的アイディア:「ノルマ方式」
  - 最初は，無条件にメモリ上のバッファへ.
  - 出現するたびにカウントを増やす.
  - 一定期間ごとのノルマを達成しなければ，バッファから脱落する.

オンラインアルゴリズム LossyCounting

(Manku & Motwani, VOLDB'02)

外部記憶



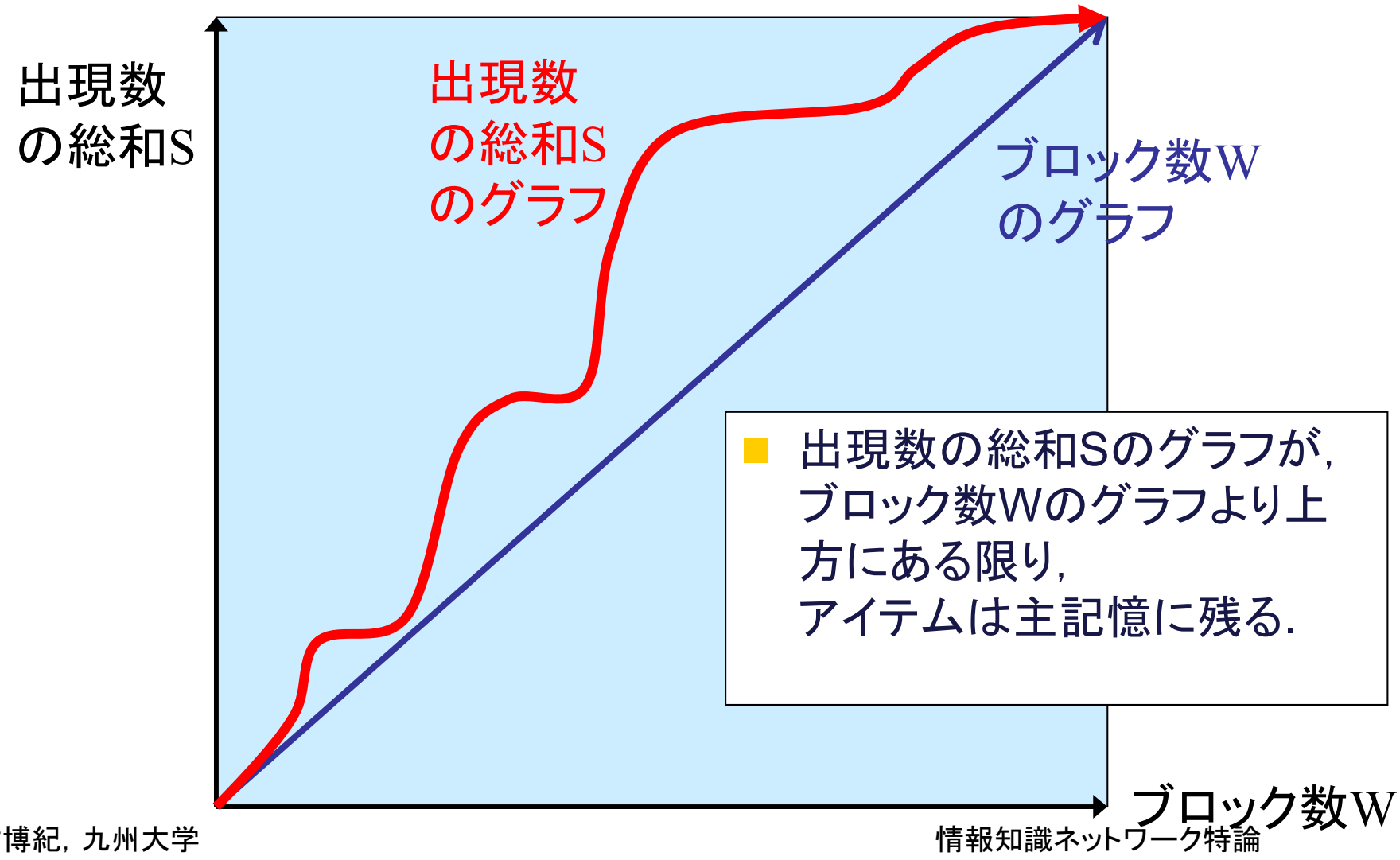
1. ストリームを長さ1/σのブロックに区切る

2. はじめて出現したらバッファに入れる.

3. 出現するたびに, カウントを増やす.

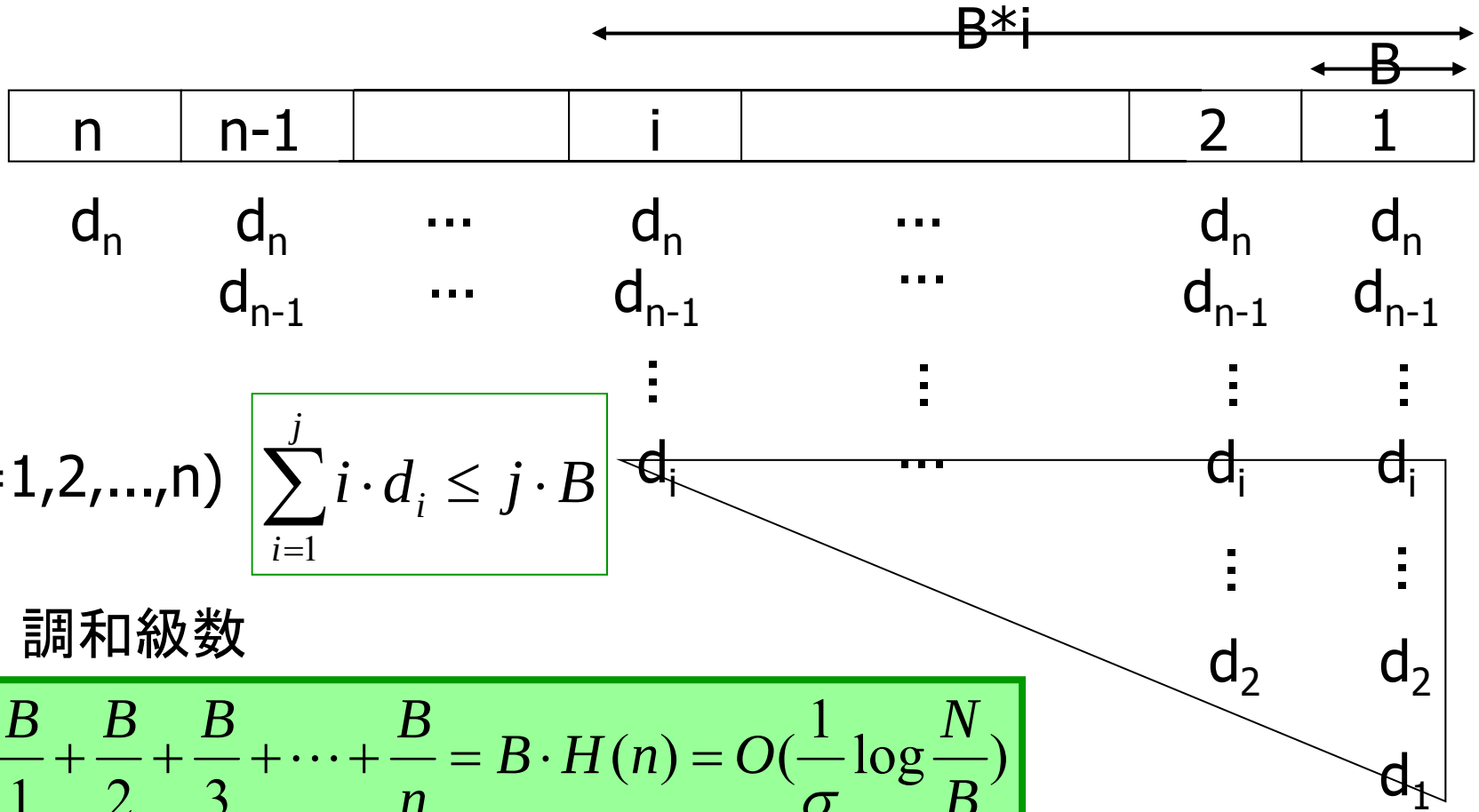
4. 出現数が, 経過ブロック数を下回ったらすてる.

Lossy Counting (M&M '02)



領域計算量の解析(Manku&Motwani '02)

生存期間ごとに分けて調べる



(j=1,2,...,n) $\sum_{i=1}^{j} i \cdot d_i \le j \cdot B$

調和級数

$$\frac{B}{1}+\frac{B}{2}+\frac{B}{3}+\cdots+\frac{B}{n}=B\cdot H(n)=O(\frac{1}{\sigma}\log\frac{N}{B})$$

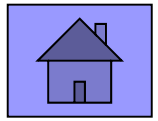
n = N/B, B = N/(N*σ)=1/σ

Lossy Countingアルゴリズム

定理(Manku & Motwani, VLDB'02):

アルゴリズム Lossy Counting は
1回の走査で，頻出データアイテム発見
問題を近似的に解く.

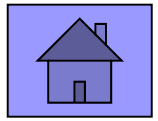
アルゴリズム Lossy Counting の
領域計算量は**O(1/σ log N)**である.



KPSアルゴリズム

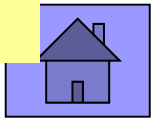
（Karp, Papadimitriou, Shenker, Manuscript, 2003）

- 1パス近似アルゴリズム
- 2パスで厳密解（最適方式）
- 初めて，領域計算量O(1/σ)を達成した！
- 基本的アイディア
  - 「連帯責任」方式



KPSの基本的アイディア
「連帯責任」方式

1. バッファサイズを $M = \lfloor 1/\sigma \rfloor + 1$ に設定する (σ頻出要素は最多でも$\lfloor 1/\sigma \rfloor$個しかないので，それにプラス1する．）
2. 初回の出現は，無条件にバッファに追加．以降，出現毎にカウントを1ずつ増やす．
3. バッファがあふれたら，全員からカウントを１点ずつ引くことを，カウント0の者がでるまで繰り返す．
4. 0点になったら退出！

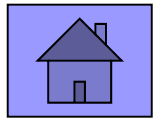


KPSの理論的解析

定理3：任意のストリームS = S[1..n]と相対頻度σ in [0,1]に対して，アルゴリズムKPSは，Sの走査終了時にすべての頻出アイテムをバッファ（メモリ）に含む．

系4：アルゴリズムKPSは，すべての頻出アイテムを，最適な領域計算量O(1/σ)で見つける（ただし非頻出なものも見つける）．

※ O(1/σ)より小さなメモリで頻出アイテムすべてを見つけることはできなことが知られている



ネットワークログマイニング

$_i$(service, host_ip, dist_ip)

DARPA IDS Evaluation DataSet *
|D|=3,013,862個（100MB）
異なりアイテム数=64,636
σ=0.1% (3,014個)　#Answer=24個

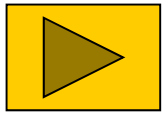
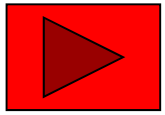
Algorithm	Time(sec)	Space(#item)
Naive	60.84	64,636
DoubleScan	**100.67**	**2,893**

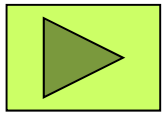
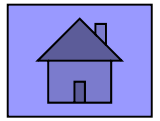
*http://www.ll.mit.edu/IST/ideval/

(ftp-data,192.168.0.20,192.168.1.30)
(ftp-data,192.168.0.20,192.168.1.30)
(ftp-data,192.168.0.20,192.168.1.30)
(telnet,192.168.1.30,192.168.0.20)
(ftp-data,192.168.0.20,192.168.1.30)
(finger,192.168.1.30,192.168.0.20)
(http,192.168.1.30,192.168.0.20)
(http,192.168.1.30,192.168.0.20)
(ftp,192.168.1.30,192.168.0.40)
(ftp,192.168.0.40,192.168.1.30)
(ftp-data,192.168.1.30,192.168.0.40)
(ftp-data,192.168.1.30,192.168.0.40)
(ftp-data,192.168.1.30,192.168.0.40)
(ftp-data,192.168.1.30,192.168.0.40)
(ftp-data,192.168.1.30,192.168.0.40)
(telnet,192.168.0.40,192.168.1.30)

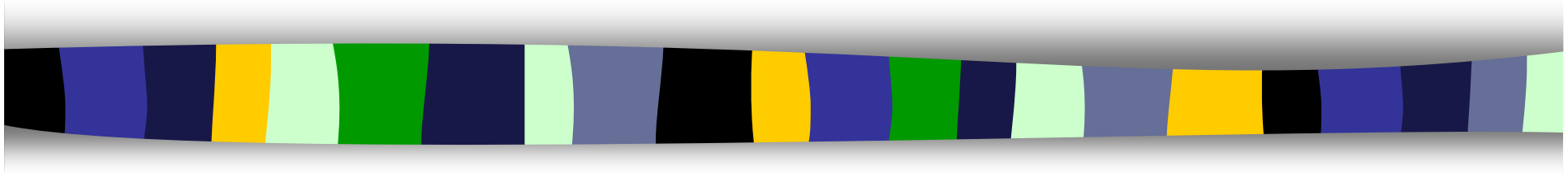
アウトライン

- ストリームデータ
- 近似カウンティング
  - Manku & Motwani (VLDB'02)
- ストリームからのパターン発見
  - Hidber (SIGMOD'99)
  - 浅井, 有村, 有川 (ICDM'02)
- ストリームに対する近似統計手法
  - ストリーム統計のいろいろ（Alon, Matias, Szegedy, STOC'96）
  - 頻度モーメントの計算手法
- まとめ

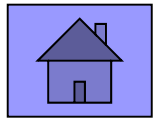



ストリームからのパターン発見



半構造データストリームから頻出木パターンを発見するための効率よいオンラインアルゴリズム

Online Algorithms for Mining Semi-structured Data Stream, T. Asai, H. Arimura, K. Abe, S. Kawasoe, S. Arikawa, Proc. IEEE ICDM'02, 2002

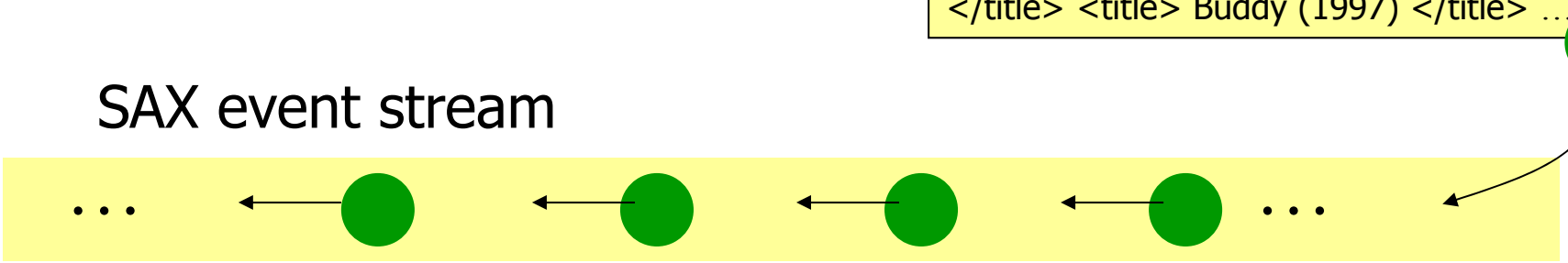


Background

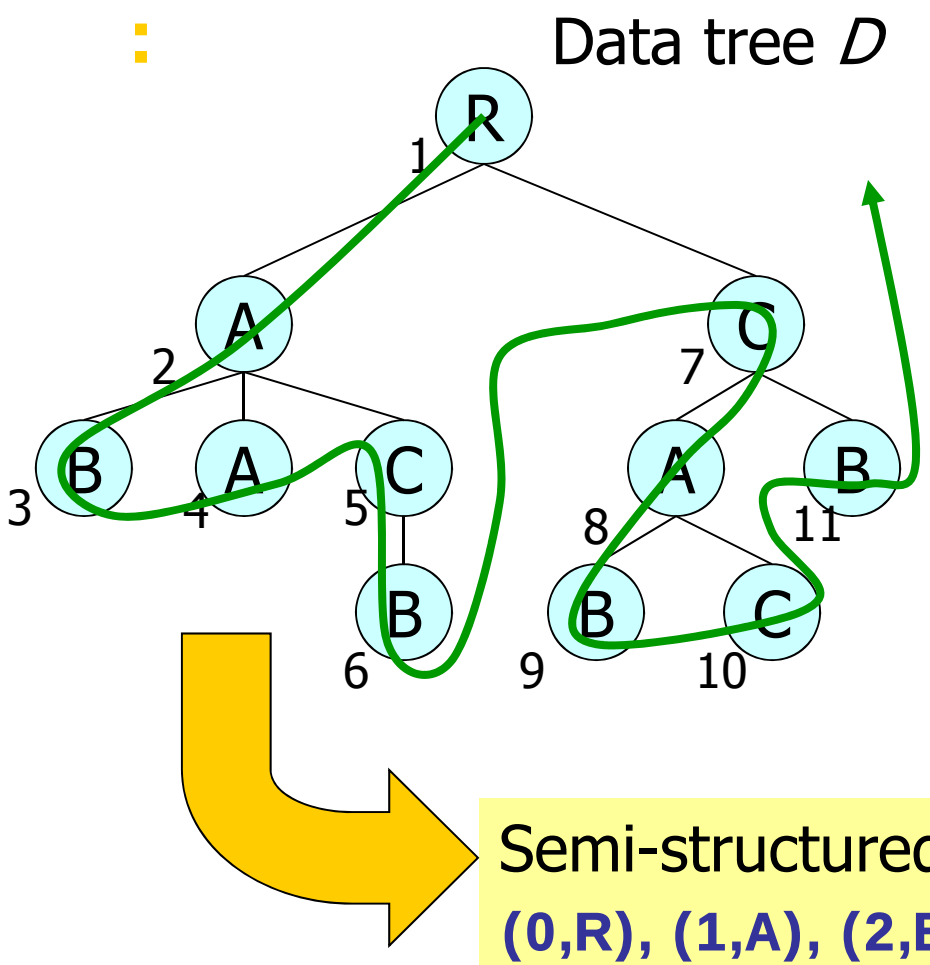
<moviedb><movie><title>Godfather</title><year>1972</year><directed_by><person><name>Francis Ford Coppola </name> <birth_name> Francis Ford Coppola </birth_name> <date_of_birth> <day> 7 April </day> <year> 1939 </year> <locate> Detroit, Michigan, USA </locate> </date_of_birth> <mini_biography> He was born in 1939 in Detroit, USA, but he grew

- **Emerging applications on Internets**
  - Eg. Network monitoring, web management, e-commerce
  - Not a static collection but a transient **data stream**
- **Unbound, Rapid, Continuous, Time varying**
- Traditional data mining methods cannot be directly applied.

(1998) (TV) </title> <title> Outrage (1998) (TV) </title> <title> Buddy (1997) </title> …



Our Definition of Semi-structured Data Stream: (depth, label)-pair representation

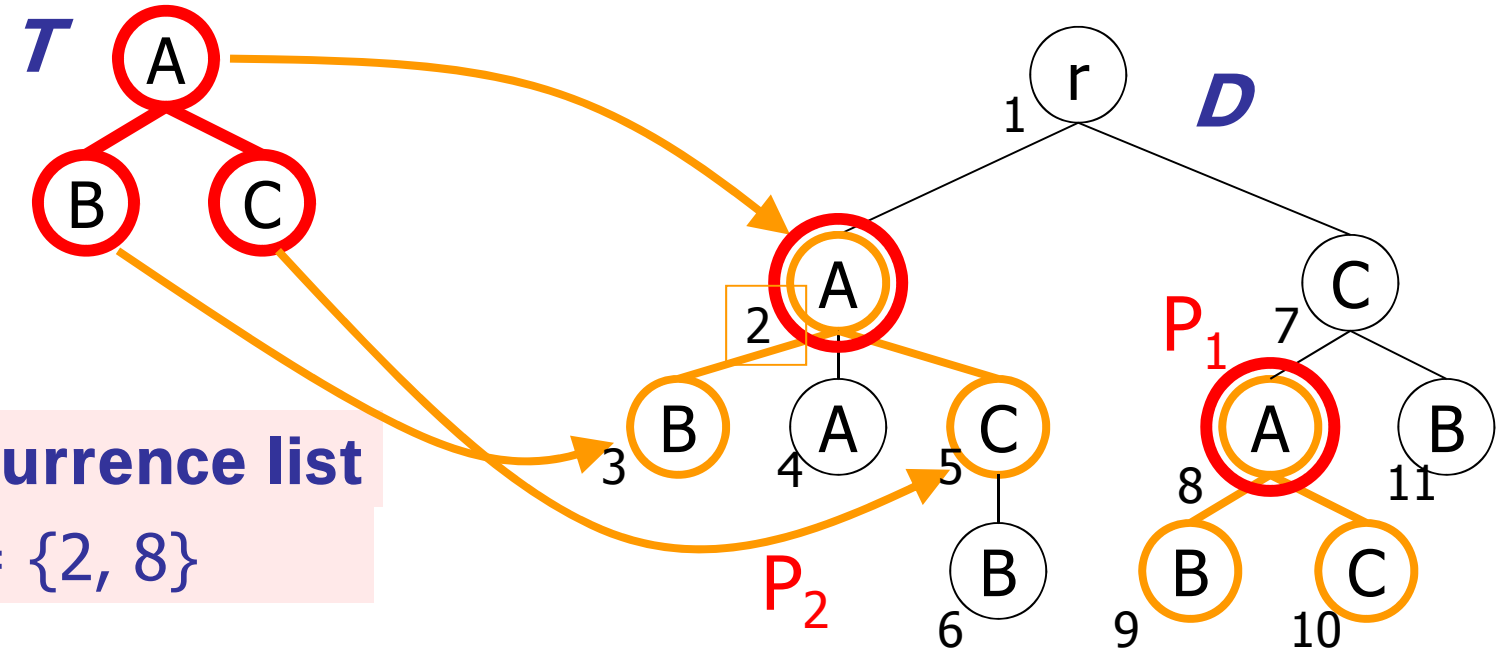


Semi-structured data stream w.r.t. *D*

(0,R), (1,A), (2,B), (2,A), (2,C), (3,B), (1,C), (2,A), (3,B), (3,C), (2,B)

The Occurrences of a Pattern

- A root occurrence of T:
  - The node to which the root of T maps by a matching function
- The root count of T:
  - The number of distinct root occurrences of T in D.



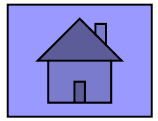
Root occurrence list

$Occ_D(T) = \{2, 8\}$

木構造データからの頻出パターン発見手法

頻出順序木パターンを発見する
効率のよいアルゴリズム

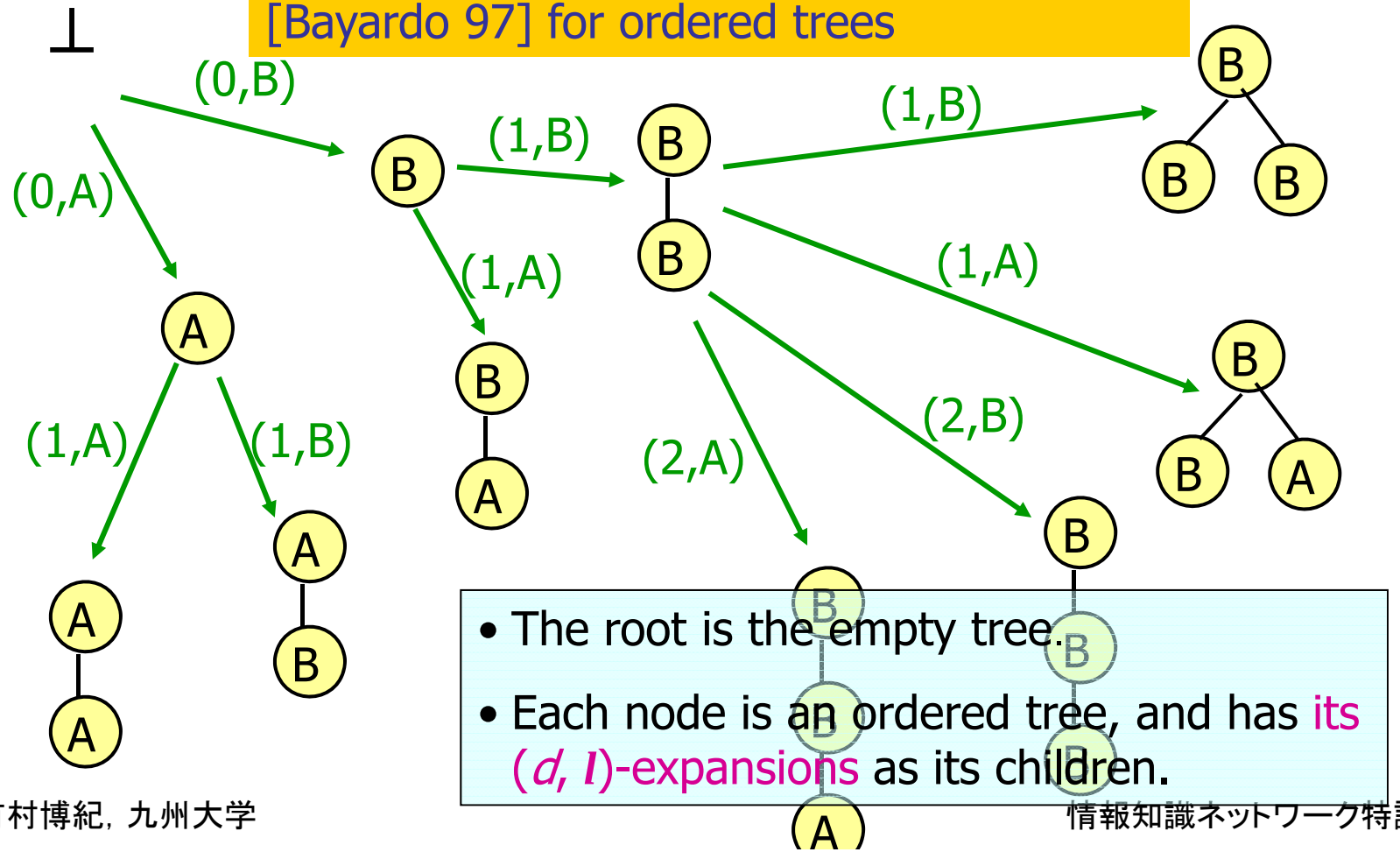
- **FREQT** [Asai *et al.* (SIAM DM'02, PKDD'02)]
  - **最右拡張**を用いた効率のよい順序木の枚挙
  - **最右葉出現**の漸増的な更新
- **Treeminer** [Zaki (SIGKDD'02)]
  - 効率のよい順序木の枚挙
  - 我々の研究とは独立



Ordered tree enumeration tree

[Asai *et al.*, SDM'02; Zaki, SIGKDD'02]

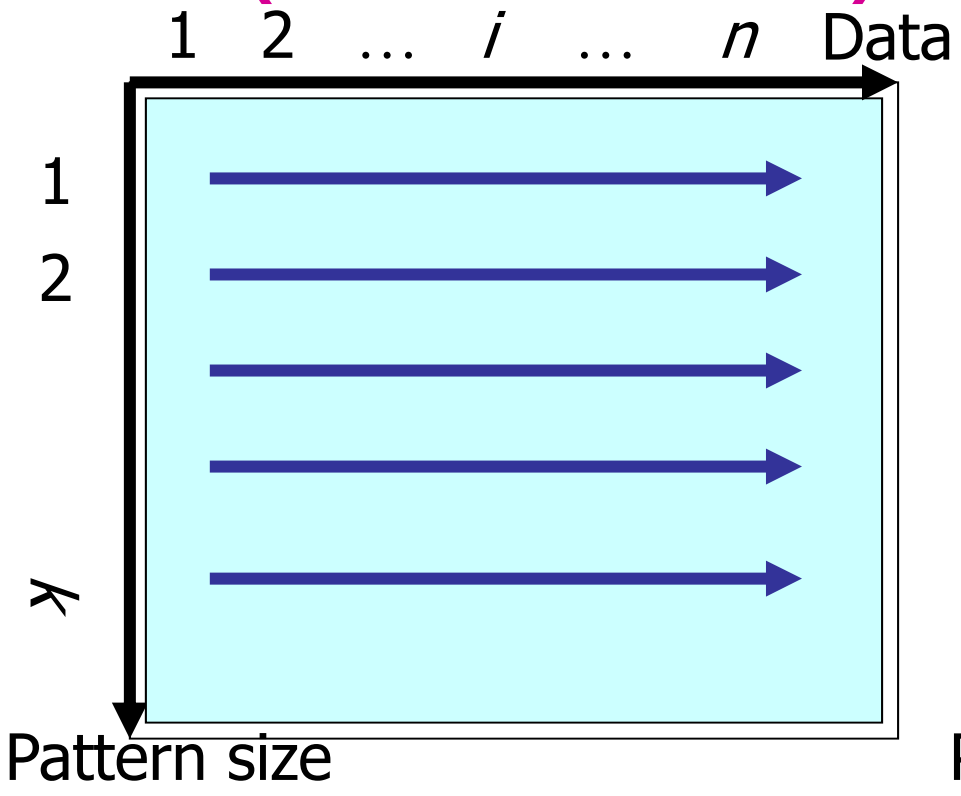
A generalization of **set enumeration tree** [Bayardo 97] for ordered trees



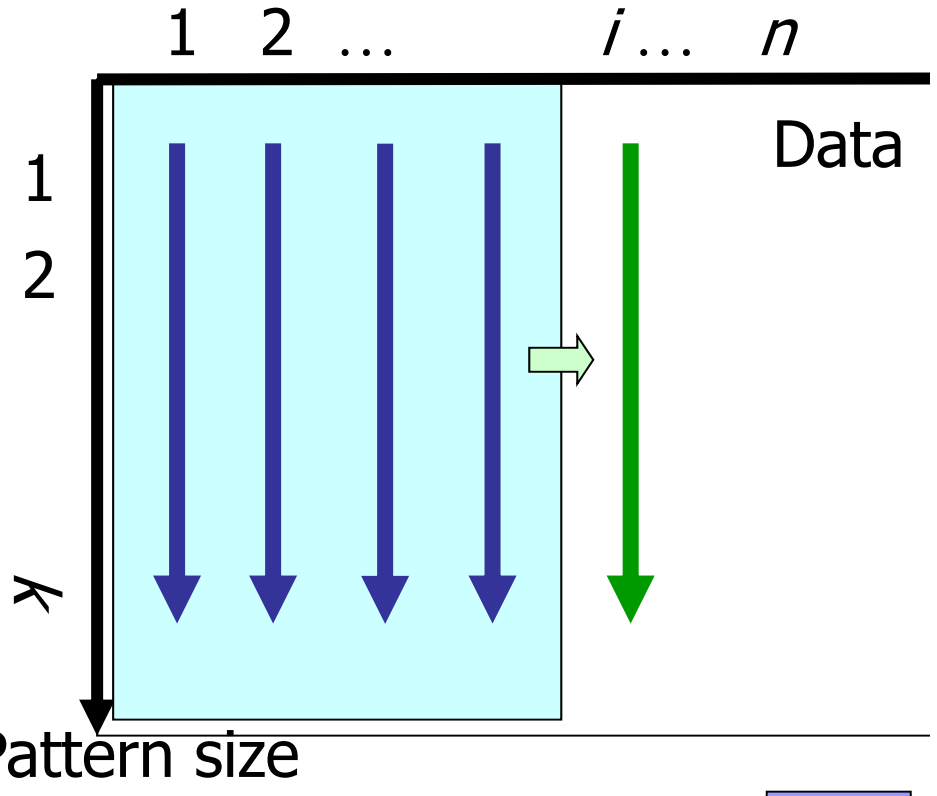
- The root is the empty tree.
- Each node is an ordered tree, and has its (d, l)-expansions as its children.

Offline vs. Online

FREQT (Offline)
Horizontal Scan
(Level-wise search)

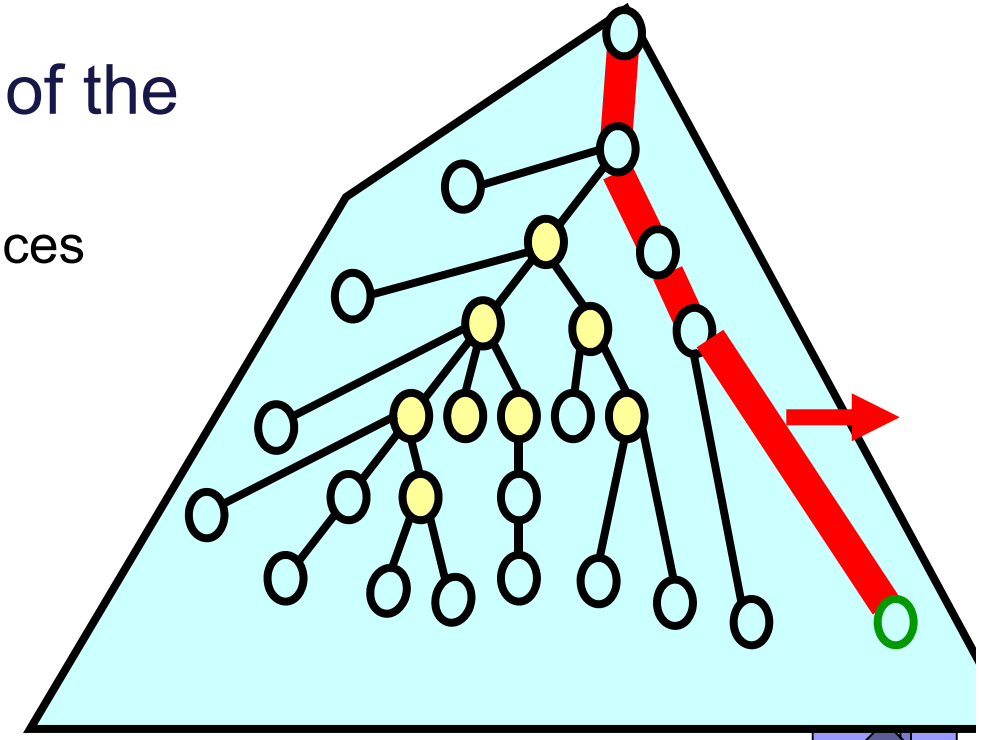


StreamT (Online)
Vertical Scan

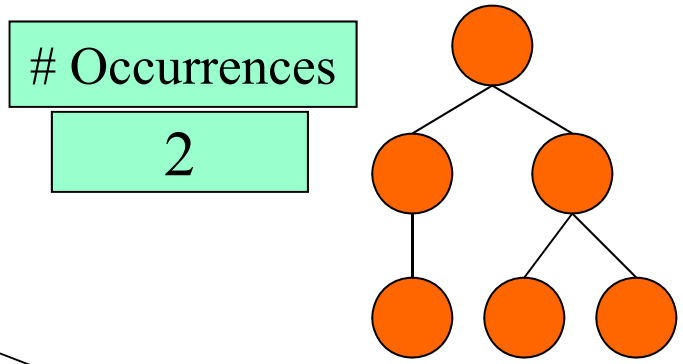


Sweep branch

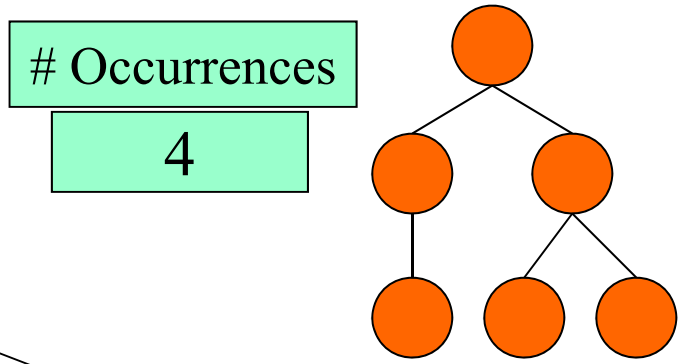
- The unique path from the root to the current node ***vi***
- The algorithm sweeps the sweep branch ***SB*** rightwards
- Records the occurrences of the candidate patterns on ***SB***
  - ➔ Use root and bottom occurrences



Tree sweeping technique

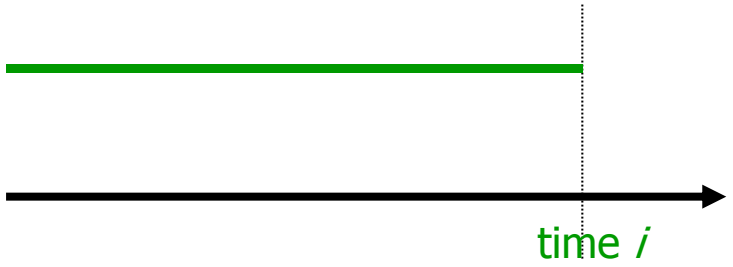


Tree sweeping technique



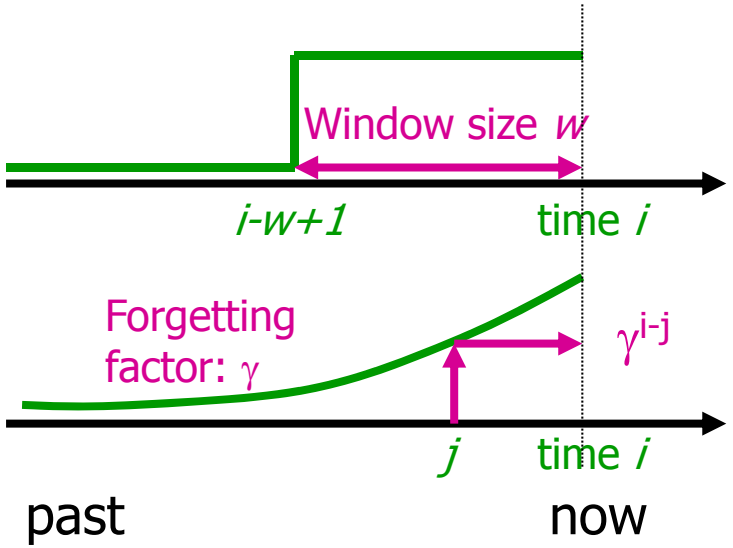
Various online models

- Basic model [Hidber 99]



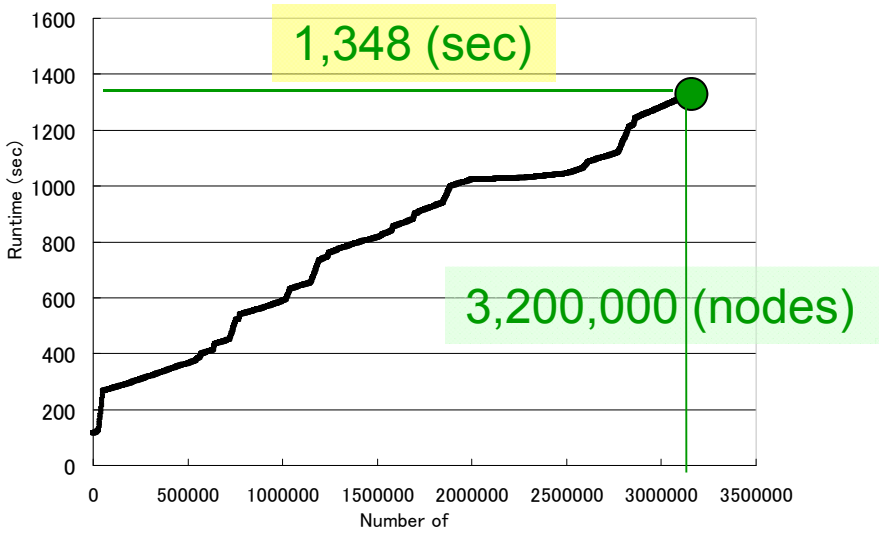
Unsuitable to tracking rapid trend changes

- Sliding window model [Manilla *et al.* 95]
- Forgetting model [Yamanishi *et al.* 00]



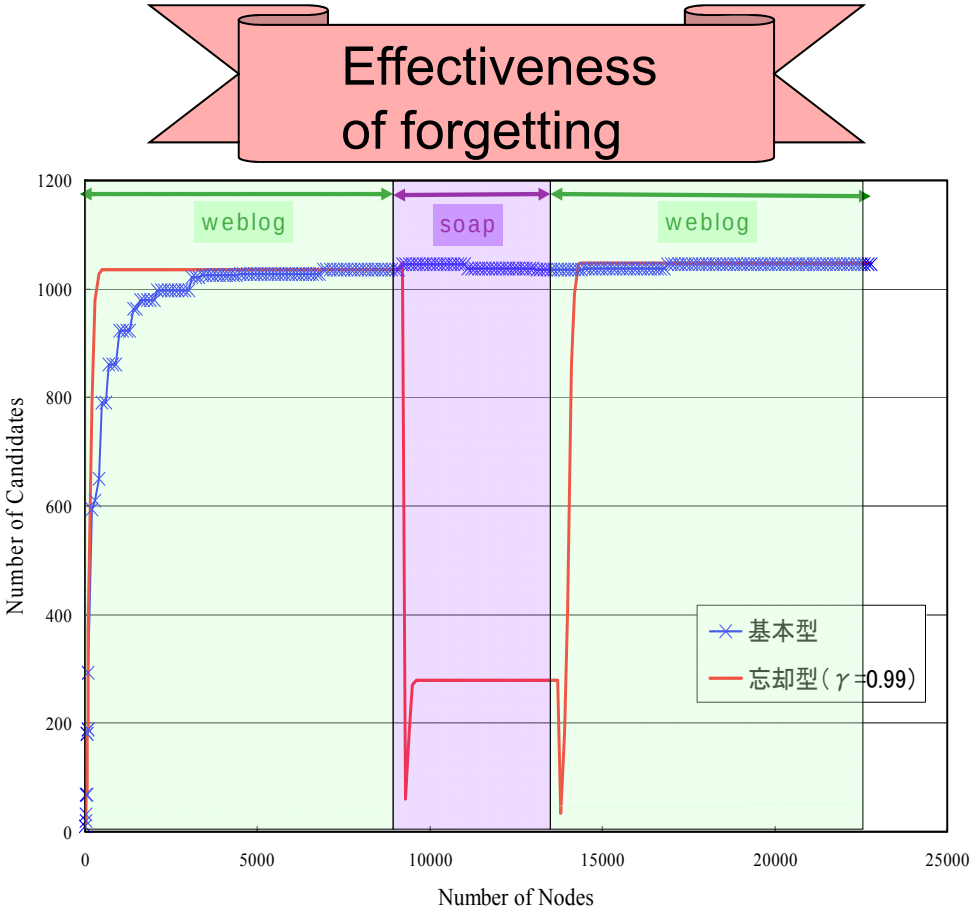
Experiments:

Scalability and effectiveness of forgetting



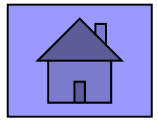
scalability

Data size: 130MB
# of nodes: 3,185,138
# of labels: 72

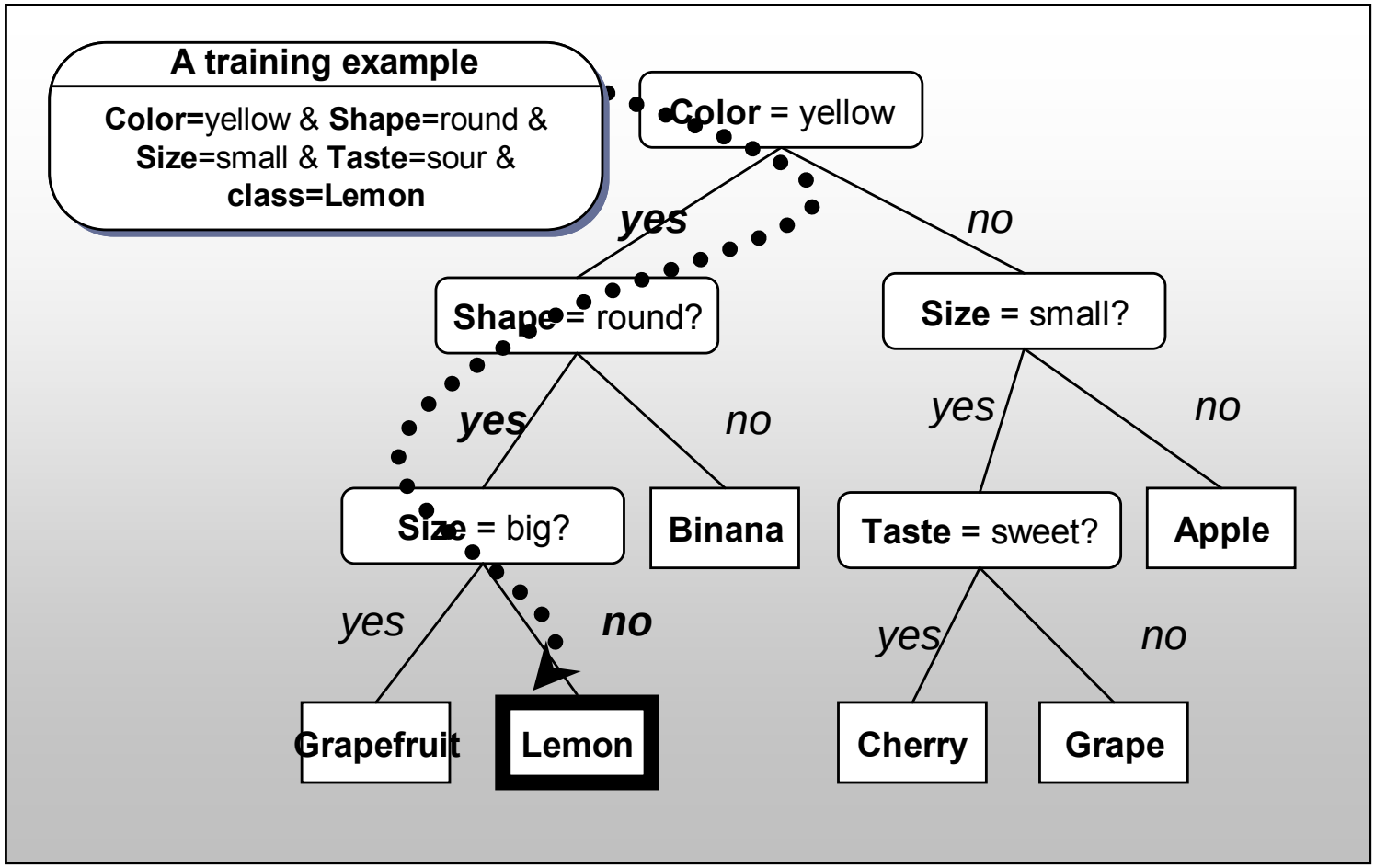


決定木のオンライン構築

- VFDT (vary fast Decision Tree Learner) [Domingos, Hulten, KDD'00]
  - 決定木を根から初めて，次第に成長（C4.5と同じ）
  - 全データの到着を待たずに，適応的に木を生長させる.

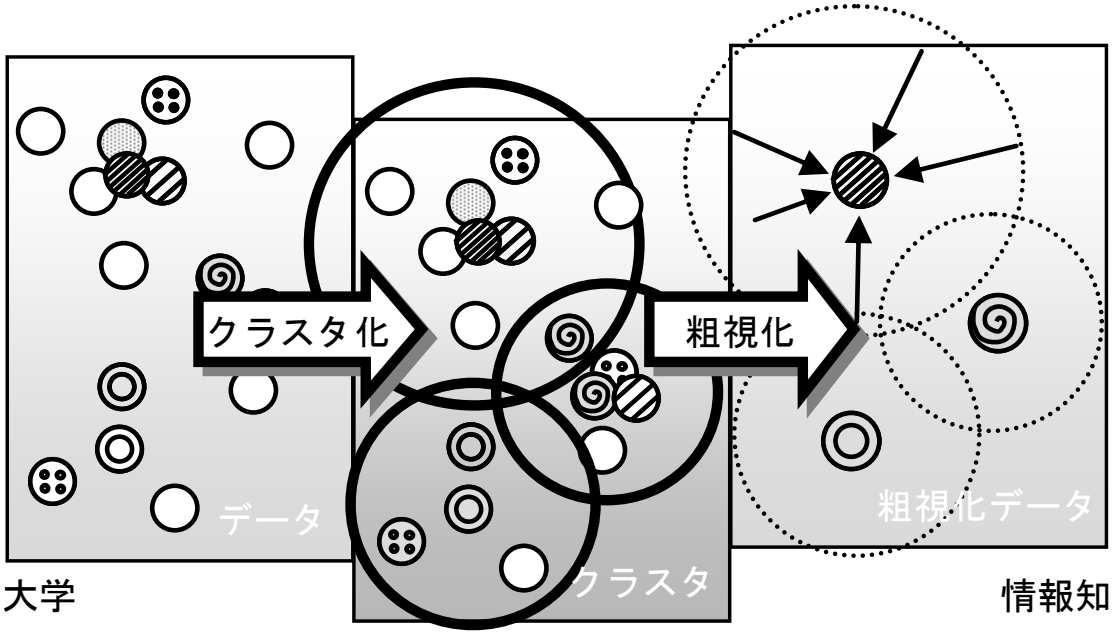


VFDT (vary fast Decision Tree Learner) [Domingos, Hulten, KDD'00]



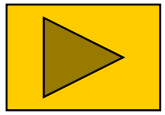
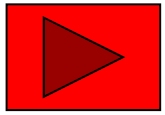
ストリーム志向クラスタリング

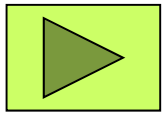
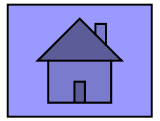
- BIRCH [Zhang, Ramakrishnan, Livny, DMKD, 1997]
  - データを粗視化してクラスタリングを大規模化
  - B木に似た構造で，適応的にクラスタを成長・分割
  - プロトタイプを用いる



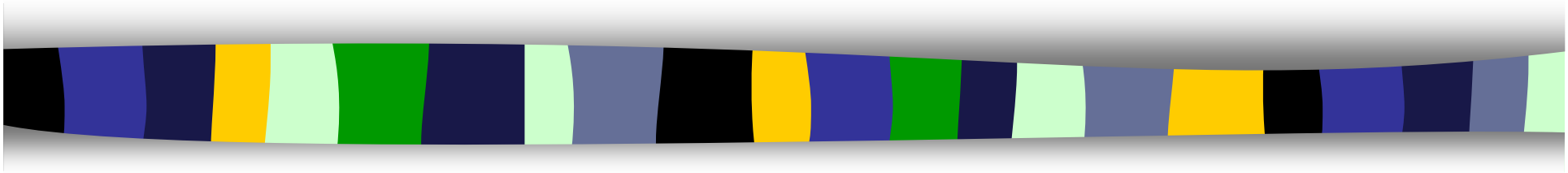
アウトライン

- ストリームデータ
- 近似カウンティング
  - Manku & Motwani (VLDB'02)
- ストリームからのパターン発見
  - Hidber (SIGMOD'99)
  - 浅井, 有村, 有川 (ICDM'02)
- ストリームに対する近似統計手法
  - ストリーム統計のいろいろ（Alon, Matias, Szegedy, STOC'96）
  - 頻度モーメントの計算手法
- まとめ

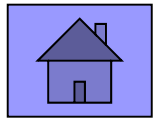



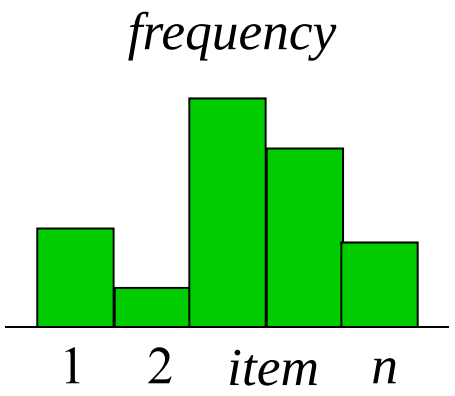
ストリームに対する近似統計手法



- Probabilistic counting **(Flajolet, Martin, FOCS'83)**
- The space complexity of approximating the frequency moments （**Alon, Matias, Szegedy, STOC'96**）
- Tracking Join and Self-Join Sizes in Limited Storages **(Alon, Gibbons, Matias, Szegedy, PODS '99)**
- Synopsis Data Structures for Massive Data Sets **(Gibbons, Matias, SODA '99)**

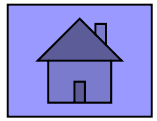


ストリームに対する統計

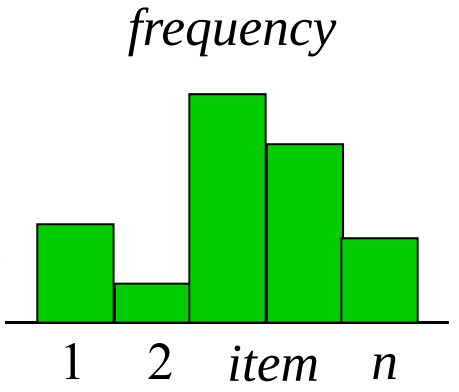


- アイテムの種類
  - 何種類のアイテムが出現しているか？
- Skewness
  - アイテムの分布がどれくらい偏っているか？
- 最頻アイテム
  - 与えられた頻度より多く出現している
    アイテムを見つけよ
- ホットリスト
  - 頻度が高い方から上位 K 個のアイテムを
    知りたい

(telr
(ftp,
(smt
(aut
(smt
(she
(sun
(ftp-data, 192.168.0.20, 192.168.1.30)
(ftp-data, 192.168.0.20, 192.168.1.30)



頻度モーメント (Frequency moment)

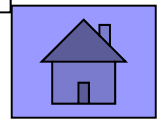


- **N** = {1, 2, …, *n*}: 要素のドメイン
- **A** = (a_1, a_2, …, a_m): データストリーム
- m_i : ストリーム中の要素 i の出現数

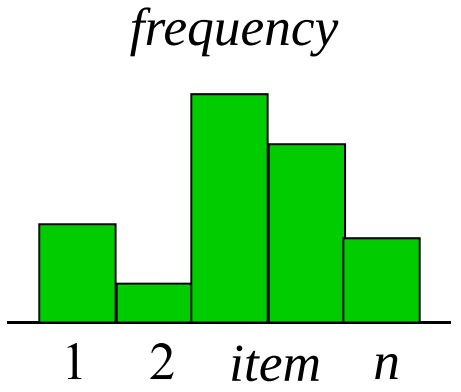
- 頻度モーメント F_k :

$$F_k = \sum_{1 \le i \le n} (m_i)^k = (m_1)^k + \ldots + (m_n)^k$$

$$F_\infty = \max_{1 \le i \le n} (m_i)^k$$

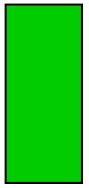


頻度モーメント (Frequency moment)



- F_0：異なるアイテムの種類
- F_1：たんなるデータの総数（カウンタ）
- F_2：片より度／Gini指数（頻度の分散）
  - Estimated Size of Self-Join
  - Skew Handling in Parallel Join（*DeWitt et al. VLDB'92*）
  - Query Result Size Estimation (*Ioannidis & Poosala SIGMOD'95*)
- F_∞: 頻度の最大値（最頻アイテム）

- ヒストグラム：どの統計量も，**O(*n*)**個の出現数カウンタをもてば簡単に計算可能!
- たった1個の出現数カウンタ だけで計算できないか？



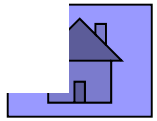
ランダム射影を用いた F_2 の計算

(Alon, Matias, Szegedy, STOC'96)

- ランダム射影(Random Projection)
  同じアイテム同士は同じ符号に，違うアイテム同士は高い確率で違う符号をわりあてるハッシュ関数

  $$h : \mathbf{N} \rightarrow \{+1, -1\}$$

  - 条件0：アイテム **x** に対して正負の符号をわりあてる.
  - 条件1：$h(x)$ の期待値は **0** .
  - 条件2：異なるアイテム **x, y** に対して，対 $h(x)$ と $h(y)$ は独立
  - 条件3：h は4独立．つまり，4つの異なるアイテム **x, y, z, w** に対して，$h(x)$, $h(y)$, $h(z)$, $h(w)$ が独立.
- 上の条件をみたすハッシュ関数は
  O(1) (int) = **O(log n)** (bits)で実現可能

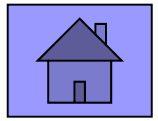


ランダム射影を用いた F_2 の計算

(Alon, Matias, Szegedy, STOC'96)

- アルゴリズム
  - データストリーム $\mathbf{A} = (a_1, a_2, \ldots, a_m)$ に対して,

    次のアイテム $\mathbf{a_i}$ を受け取って，値 $\boldsymbol{h(a_i)} \in \{+1, -1\}$ を変数 $\mathbf{Z}$ に足し込む.
  - $\boldsymbol{F_2}$ の推定値として $\mathbf{Z^2}$ を返す
- 結果
  - 上のアルゴリズムは高い確率で$\boldsymbol{F_2}$ を推定
  - 対の独立性で平均の一致を保障し，4つ組の独立性で分散（誤差）の小ささを保障する

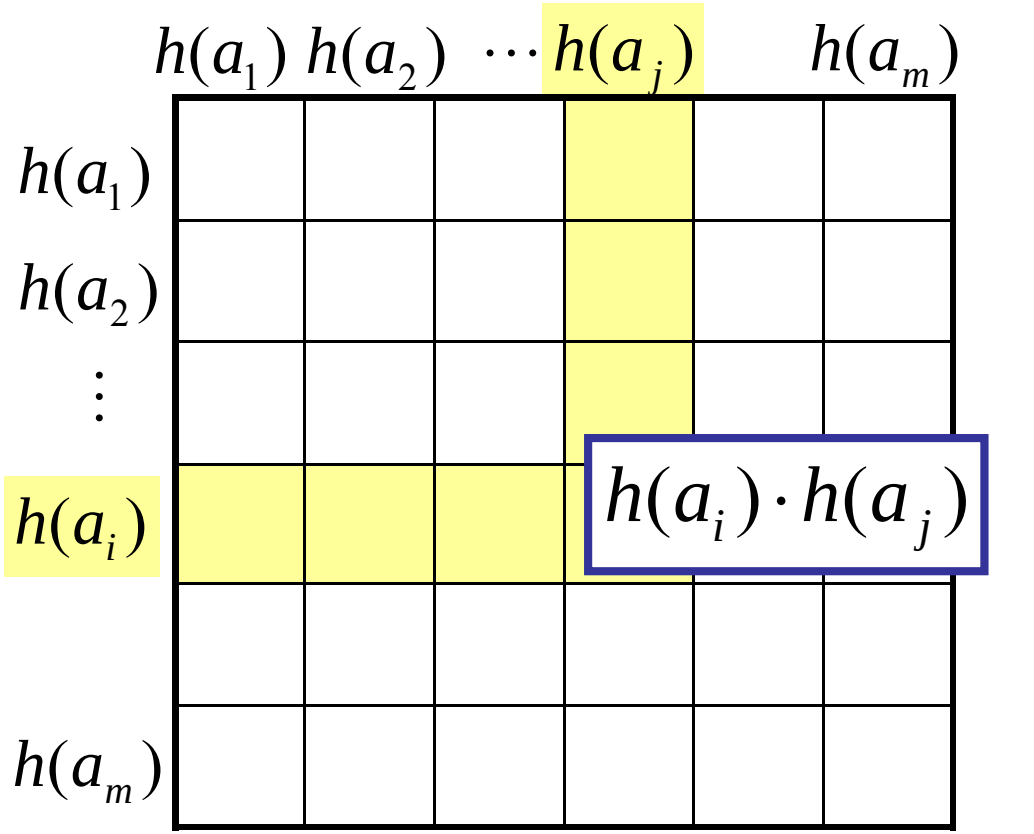


なぜこれで F_2 が計算できるか？

$$Z^2 = (h(a_1) + \ldots + h(a_m))^2 = \sum_{1 \le i, j \le m} h(a_i) \cdot h(a_j)$$

Z^2 は，全アイテムの組の符号の積和に等しい

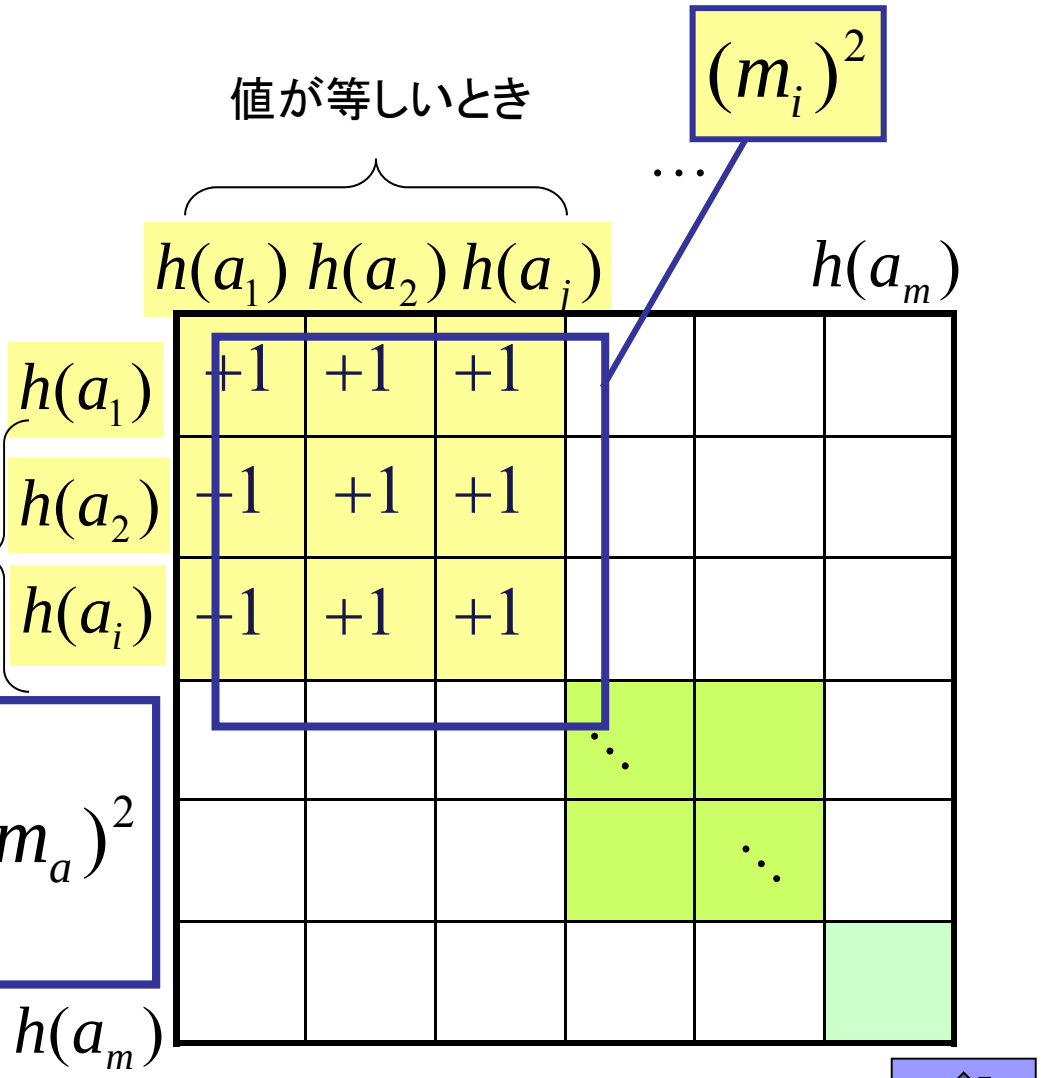
$$\sum_{1 \le i \ne j \le m} h(a_i) \cdot h(a_j) = 0$$



なぜこれで F_2 が計算できるか？

- 2つの同じアイテムに対しては，積はつねに＋１．
- 積和の期待値は$(m_a)^2$

$$E\left[\sum_{1\le i,j\le m, a_i=a_j=a} h(a_i)\cdot h(a_j)\right] = (m_a)^2$$



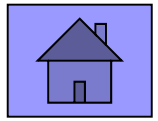
なぜこれで F_2 が計算できるか？

$$Z^2 = (h(a_1) + \ldots + h(a_m))^2 = \sum_{1 \le i, j \le m} h(a_i) \cdot h(a_j)$$

- 独立性から，２つの異なるアイテムに対しては，次の４つが等確率で生起
- **(+1, -1), (-1, +1)**
  (+1, +1), (- 1, -1)
- 積和の期待値は **0**

	$h(a_1)$	$h(a_i)$		$\cdots$ $h(a_j)$		$h(a_m)$
$h(a_1)$				−1	+1	+1
$h(a_i)$				+1	−1	−1
				−1	−1	+1
$h(a_j)$	−1	+1	−1			+1
	+1	−1	−1			−1
$h(a_m)$	+1	−1	+1	+1	−1	

$$E\left[\sum_{1 \le i \ne j \le m, a_i \ne a_j} h(a_i) \cdot h(a_j)\right] = 0$$



なぜこれで F_2 が計算できるか？

	$h(a_1)$	$h(a_i)$		$h(a_j)$		$h(a_m)$
$h(a_1)$	+1	+1	+1	−1	+1	+1
$h(a_i)$	+1	+1	+1	+1	−1	−1
	+1	+1	+1	−1	−1	+1
$h(a_j)$	−1	+1	−1	+1	+1	+1
	+1	−1	−1	+1	+1	−1
$h(a_m)$	+1	−1	+1	+1	−1	+1

$$E[Z^2] = E\left[\left(\sum_{1\le i\le m} h(a_i)\right)^2\right]$$

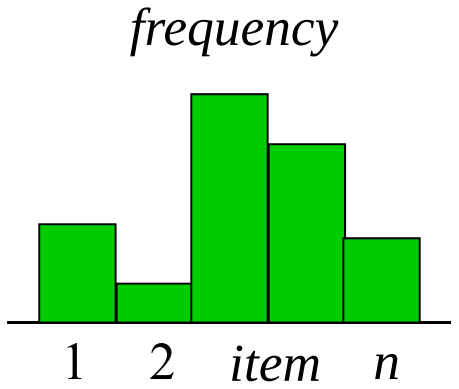
$$= E\left[\sum_{1\le i,j\le m} (h(a_i)\cdot h(a_j))\right]$$

$$= E\left[\sum_{1\le i,j\le m, 1\le a\le n, a_i=a_j=a} h(a_i)\cdot h(a_j)\right]$$

$$+ E\left[\sum_{1\le i,j\le m, a_i\neq a_j} h(a_i)\cdot h(a_j)\right]$$

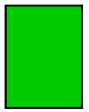
$$= \sum_{1\le a\le n} m_a + 0 = F_2$$

頻度モーメント (Frequency moment)



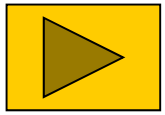
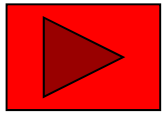
- F_0：異なるアイテムの種類
  確率的近似 **O(log n) (bit) = O(1) (int)**
- F_1：たんなるデータの総数（カウンタ）
  確率的近似 **O(loglog n) (bit)**
- F_2：片より度／Gini指数（頻度の分散）
  確率的 **O(log n) (bit) = O(1) (int)**

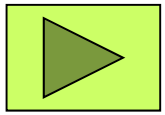
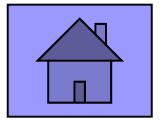
- F_2 は，たった1個の出現数カウンタだけで計算できる！



アウトライン

- ストリームデータ
- 近似カウンティング
  - Manku & Motwani (VLDB'02)
- ストリームからのパターン発見
  - Hidber (SIGMOD'99)
  - 浅井, 有村, 有川 (ICDM'02)
- ストリームに対する近似統計手法
  - ストリーム統計のいろいろ（Alon, Matias, Szegedy, STOC'96）
  - 頻度モーメントの計算手法
- まとめ

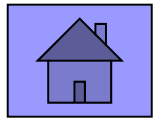



まとめ

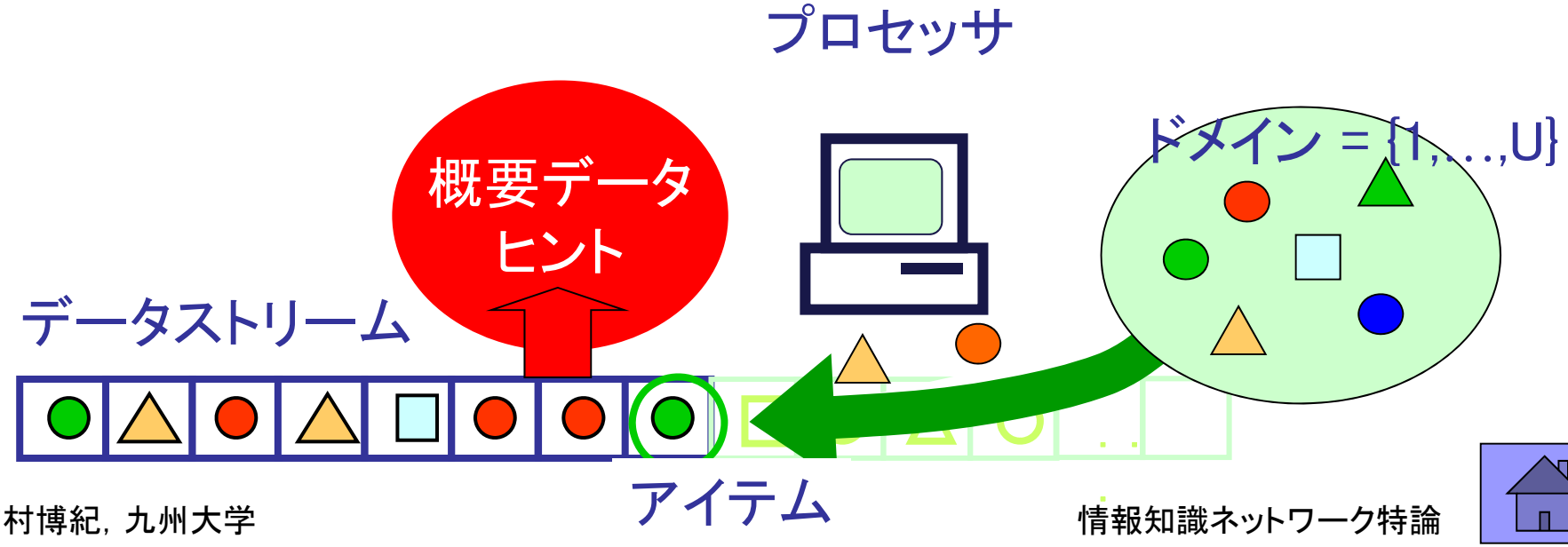
- ストリームデータ
  - 高速なデータストリームから，時間変化しながら，連続して供給される，大量のデータ
- ストリームアルゴリズム
  - 膨大で高速なデータストリーム
  - 限定された計算資源を用いて働き続ける
  - 近似的な解でよい

- ストリームデータ
- ストリームに対する近似統計手法
  - ストリーム統計のいろいろ
  - 頻度モーメントの計算手法
- 近似カウンティング
  - Manku & Motwani (VLDB'02)
- ストリームからのアイテム集合発見
- 半構造データストリーム
  - 浅井，有村，有川 (ICDM'02)



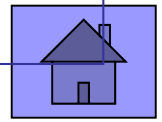
ストリームデータ処理

- 膨大で高速なデータストリームから，
- 時間とともに変化するパターンや規則を発見・抽出し，（マイニングの仕事のとき）
- 限定された計算資源を用いて働き続ける
- ただし，近似的な解でよい



関連研究: SIGKDD 2003他から

- **Association Rules:** (Chang & Lee, SIGKDD'03)
  Finding recent frequent itemsets adaptively over online data streams
- **Decision Trees:** (Wang, Fan, P. S. Yu, J. Han, SIGKDD'03)
  Mining concept-drifting data streams using ensemble classifiers
- **Multiple streams:** (Zhu & Shasha, SIGKDD'03)
  Efficient elastic burst detection in data streams
- **Multiple streams:** (Babcock & C. Olston, SIGMOD'03)
  Distributed Top-K Monitoring
- **Multiple streams:** (Guha, Gunopulos, Koudas, SIGKDD'03)
  Correlating synchronous and asynchronous data streams
- **IDS:** (Yamanishi, Takeuchi, Williams, SIGKDD'02)
  Online Unsupervised outlier detection using finite mixtures with discounting learning algorithms
- **IDS:** (W. Lee & S. J. Stolfo, Usenix, Security, 1998)
  Data Mining Approaches for Intrusion Detection



今日の内容

6回:ストリームマイニング

- ストリームマイニングとは
- ストリームに対する近似統計手法
- 近似カウンティング
- ストリームからのアイテム集合発見
- 半構造データストリーム

ポイント

- 超低メモリアルゴリズム