海量数据流处理技术

童咏昕

计算机学院 软件开发环境国家重点实验室

yxtong@buaa.edu.cn

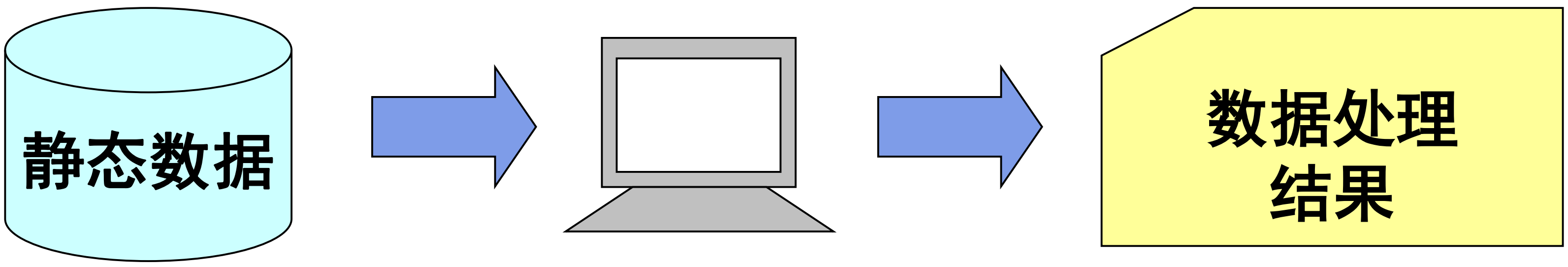
课程提纲

- 数据流简介
- 基础性问题
- 确定性算法
  - Majority算法
  - Misra-Gries(MG)算法
  - Space-Saving算法
  - Lossy Counting算法
- 随机性算法
  - Sticky Sampling算法
  - Count-Min(CM)Sketch算法

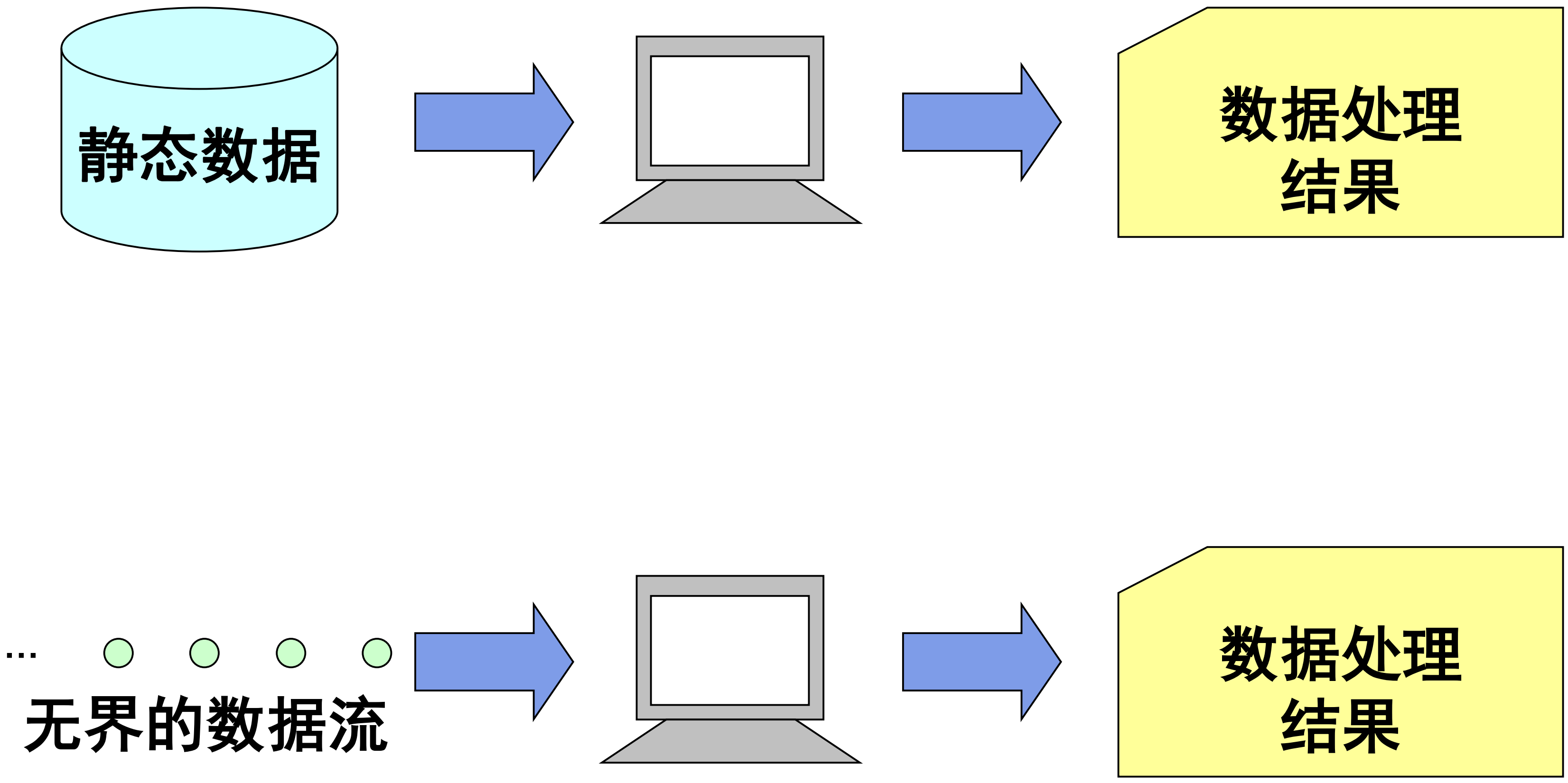
课程提纲

- 数据流简介
- 基础性问题
- 确定性算法
  - Majority算法
  - Misra-Gries(MG)算法
  - Space-Saving算法
  - Lossy Counting算法
- 随机性算法
  - Sticky Sampling算法
  - Count-Min(CM)Sketch算法

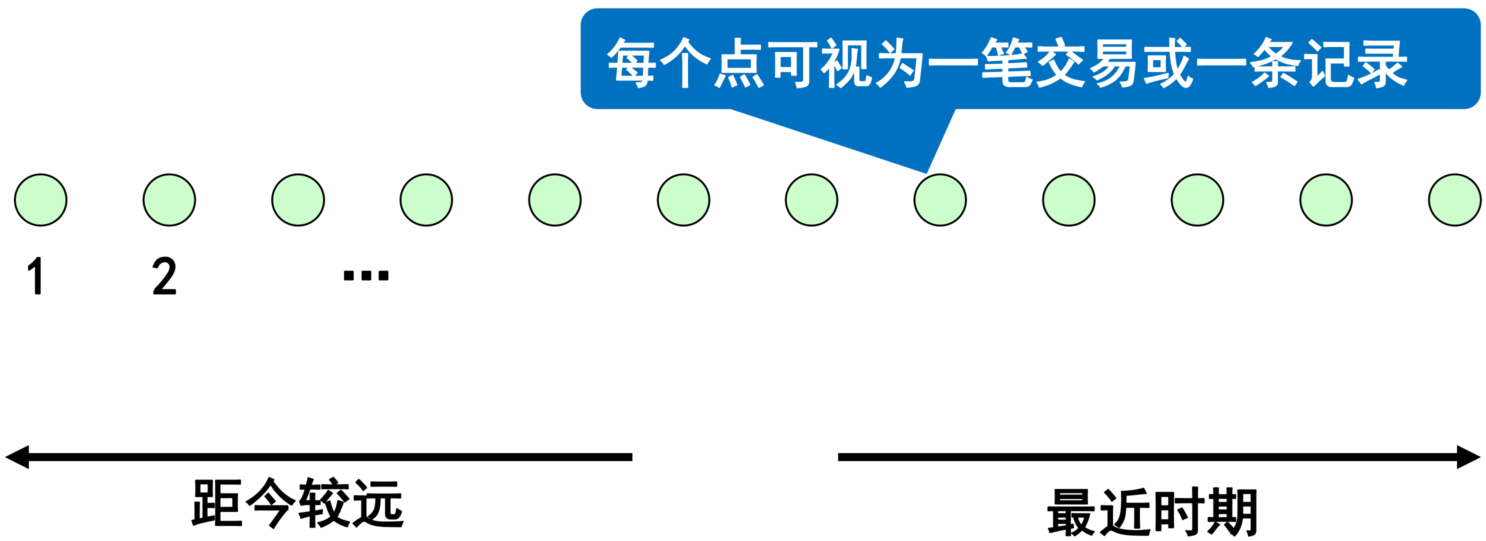
静态数据 vs. 流状数据



静态数据 vs. 流状数据

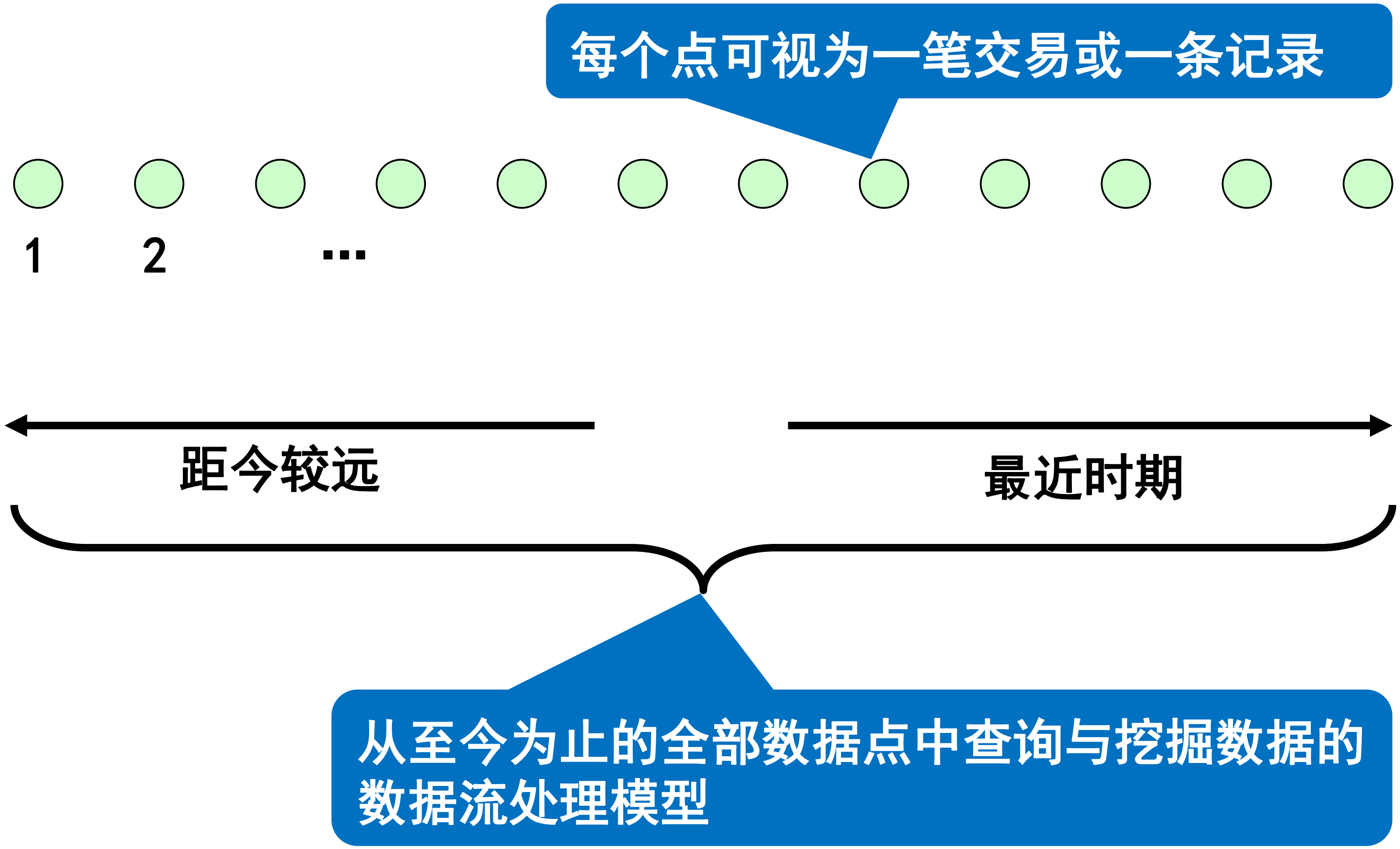


数据流概念



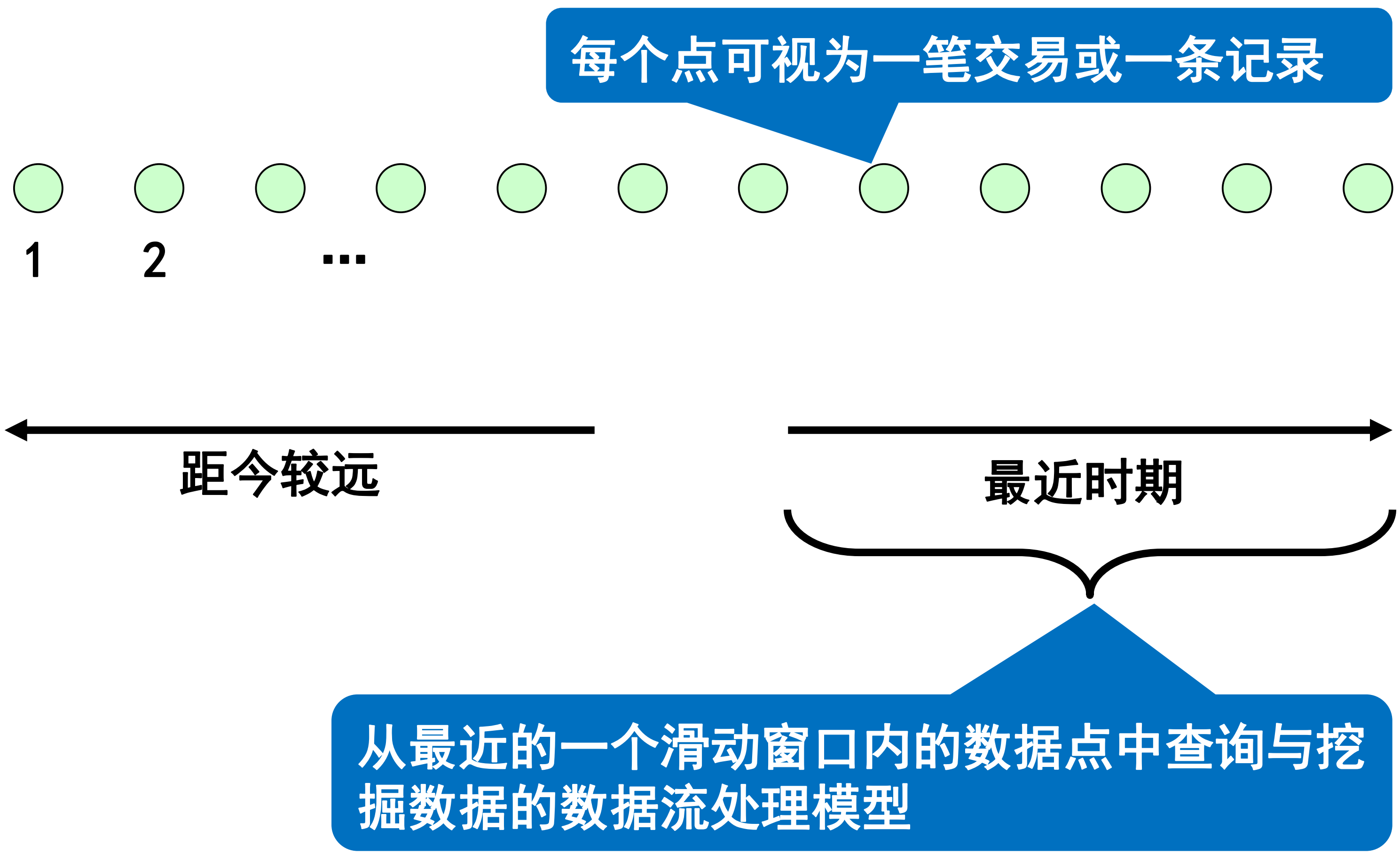
数据流模型

- 完整数据流模型(Entire Data Stream Model)



数据流模型

- 滑动窗口数据流模型(Sliding-Window Model)



课程提纲

- 数据流简介
- 基础性问题
- 确定性算法
  - Majority算法
  - Misra-Gries(MG)算法
  - Space-Saving算法
  - Lossy Counting算法
- 随机性算法
  - Sticky Sampling算法
  - Count-Min(CM)Sketch算法

数据流处理的基础性问题

差异项种类查询问题：给定一个长度为N的数据流，该问题旨在完整数据流模型下统计差异项（即不同项）种类的个数！

数据流处理的基础性问题

差异项种类查询问题：给定一个长度为N的数据流，该问题旨在完整数据流模型下统计差异项（即不同项）种类的个数！

差异项频率查询问题：给定一个长度为N的数据流，该问题完整数据流模型下统计每个差异项（即不同项）的出现频率！

数据流处理的基础性问题

差异项种类查询问题：给定一个长度为N的数据流，该问题旨在完整数据流模型下统计差异项（即不同项）种类的个数！

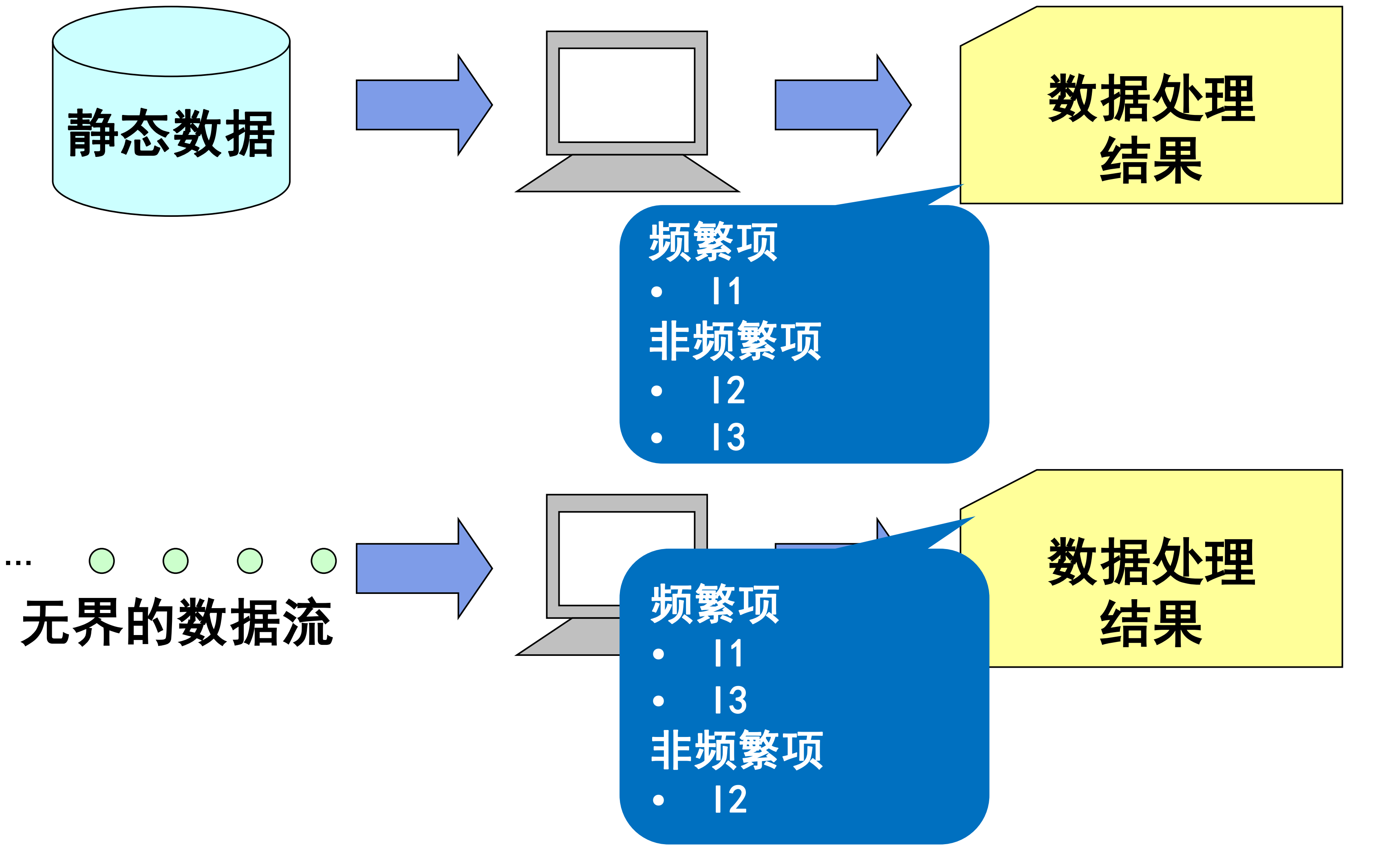
频繁项查询问题：给定一个长度为N的数据流和频繁阈值s，该问题旨在完整数据流模型下发现全部频繁项，即每个项出现频率大于sN！

差异项频率查询问题：给定一个长度为N的数据流，该问题完整数据流模型下统计每个差异项（即不同项）的出现频率！

数据流处理算法特性

	传统静态数据处理	动态数据流处理
数据类型	有限规模的静态数据	无限规模、动态且高速抵达的流状态数据
存储方式	硬盘/存储有限	存储空间不足
响应效率	非实时查询	实时查询
查询结果	精准结果	近似（或精准）结果

静态数据查询 vs. 数据流查询



假阳性(False Positive)

- E.g.
  - 期望输出
    - 频繁项
      - I_1
    - 非频繁项
      - I_2
      - I_3
  - 算法输出
    - 频繁项
      - I_1
      - I_3
    - 非频繁项
      - I_2

假阳性
-某项被分类为频繁项
-但该项事实上是非频繁项

哪个项是假阳性? I_3

还有么? **没了**

假阳性数量 = 1

如果称算法不存在假阳性

所有非频繁项在算法输出中都被分类为非频繁项

假阴性(False Negative)

- E. g.
  - 期望输出
    - 频繁项
      - I_1
    - 非频繁项
      - I_2
      - I_3
  - 算法输出
    - 频繁项
      - I_1
      - I_3
    - 非频繁项
      - I_2

假阴性
-某项被分类为非频繁项
-但该项事实上是频繁项

哪个项是假阴性? I_3

还有么? 没了

假阴性数量 = 1

假阳性数量 = 0

如果称算法不存在假阴性

所有频繁项在算法输出中都被分类为频繁项

数据流查询的近似误差

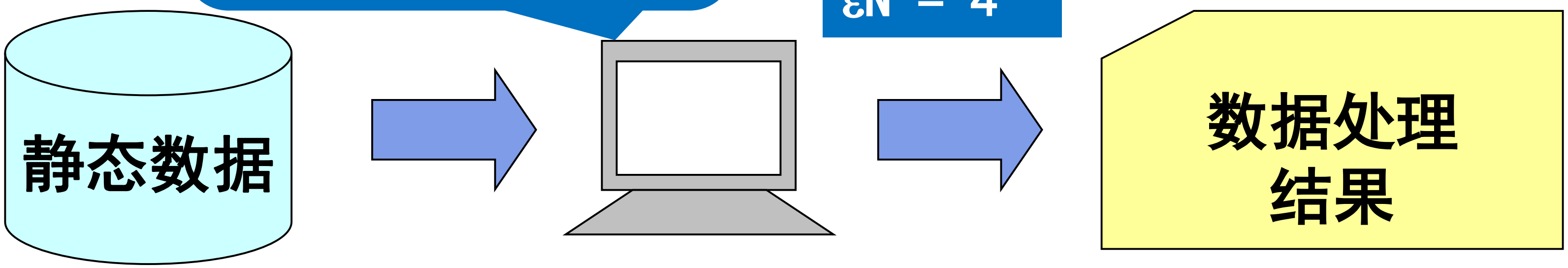
N:流中项总数

N = 20

ε = 0.2

εN = 4

所有项真实统计信息
- I1: 10
- I2: 8
- I3: 12



项	真实出现次数	估计出现次数	Diff. D	D <= εN ?
I_1	10	10	0	Yes
I_2	8	4	4	Yes
I_3	12	10	2	Yes

所有项估计统计信息
- I1: 10
- I2: 4
- I3: 10

容错概要(ε-deficient synopsis)

- 数据流长度N：为流的当前长度(或项出现的总次数)
- 近似误差ε：为输入参数(0和1之间的实数)

- 一个算法如果满足以下条件，则符合ε-deficient synopsis
  - 条件 1：不存在假阴性.

    在算法的输出中，所有真正频繁的项都被输出为频繁项
  - 条件 2：估计频率和真实频率的差值不超过εN.
  - 条件 3：所有真实频率少于(s-ε)N的项都会被分类为不频繁项.

近似频繁项查询问题：给定一个长度为N的数据流、频繁阈值s和近似误差ε，该问题旨在完整数据流模型下发现全部的近似频繁项，即每个项出现频率大于(s-ε)N！

课程提纲

- 数据流简介
- 基础性问题
- 确定性算法
  - Majority算法
  - Misra-Gries(MG)算法
  - Space-Saving算法
  - Lossy Counting算法
- 随机性算法
  - Sticky Sampling算法
  - Count-Min(CM)Sketch算法

课程提纲

- 数据流简介
- 基础性问题
- 确定性算法
  - Majority算法
  - Misra-Gries(MG)算法
  - Space-Saving算法
  - Lossy Counting算法
- 随机性算法
  - Sticky Sampling算法
  - Count-Min(CM)Sketch算法

热身:过半数Majority算法(>1/2)

- 问题：如何找出数据流S中出现次数超过1/2的元素
  - |S|很大,无法记录每个元素的出现次数

热身:过半数Majority算法(>1/2)

- 输入：
  - 数据流$S=x_1, x_2, \cdots, x_n, x_i \in U$
- 输出：流S中出现频率超过s=1/2的元素

- 算法：MAJORITY
  - 如果存在多数元素,则MAJORITY一定返回该元素

Majority算法

- MAJORITY: Initialize(x)
  - key ← ∅, value ← 0
- MAJORITY: Update(x)
  - If value = 0 then
    - key ← x, value = 1
  - Else if key = x then
    - value = value + 1
  - Else
    - value = value − 1
- MAJORITY: Return()
  - If value > 0 then return key
  - Else return ∅

Majority算法

- MAJORITY: Initialize(x)
  - key ← ∅, value ← 0
- MAJORITY: Update(x)
  - If value = 0 then
    - key ← x, value
  - Else if key = x then
    - value = value + 1
  - Else
    - value = value − 1
- MAJORITY: Return()
  - If value > 0 then return key
  - Else return ∅

初始化key为空，计数器为0

Majority算法

- MAJORITY: Initialize(x)
  - key ← ∅, value ← 0
- MAJORITY: Update(x)
  - If value = 0 then
    - key ← x, value = 1
  - Else if key = x then
    - value = value + 1
  - Else
    - value = value - 1
- MAJORITY: Return()
  - If value > 0 then return key
  - Else return ∅

如果计数器为0，记录当前元素
并设计数器为1

Majority算法

- MAJORITY: Initialize(x)
  - key ← ∅, value ← 0
- MAJORITY: Update(x)
  - If value = 0 then
    - key ← x, value = 1
  - Else if key = x then
    - value = value + 1
  - Else
    - value = value
- MAJORITY: Return
  - If value > 0 then return key
  - Else return ∅

如果计数器不为0，且x与key记录元素相同，增加计数器

Majority算法

- MAJORITY: Initialize(x)
  - key ← ∅, value ← 0
- MAJORITY: Update(x)
  - If value = 0 the
    - key ← x, value
  - Else if key = x then
    - value = value + 1
  - Else
    - value = value − 1
- MAJORITY: Return()
  - If value > 0 then return key
  - Else return ∅

如果计数器不为0,且x与key记录元素不同,减小计数器

Majority算法

- MAJORITY: Initialize(x)
  - key ← ∅, value ← 0
- MAJORITY: Update(x)
  - If value = 0 then
    - key ← x, value = 1
  - Else if key = x then
    - value = value
  - Else
    - value = value - 1
- MAJORITY: Return()
  - If value > 0 then return key
  - Else return ∅

如果计数器不为0，则key中记录元素为出现次数超过1/2元素

Majority算法实例

- a, b, a, b, a, b, a, b, a, a, b
- key = ∅, value = 0

两种元素

- If value = 0 then
  - key ← x, value = 1
- Else if key = x then
  - value = value + 1
- Else
  - value = value − 1

Majority算法实例

- a, b, a, b, a, b, a, b, a, a, b
- key = a, value = 1

- If value = 0 then
  - key ← x, value = 1
- Else if key = x then
  - value = value + 1
- Else
  - value = value − 1

Majority算法实例

- a, b, a, b, a, b, a, b, a, a, b
- key = a, value = 0

- If value = 0 then
  - key ← x, value = 1
- Else if key = x then
  - value = value + 1
- Else
  - value = value − 1

Majority算法实例

- a, b, a, b, a, b, a, b, a, a, b
- key = a, value = 1

- If value = 0 then
  - key ← x, value = 1
- Else if key = x then
  - value = value + 1
- Else
  - value = value - 1

Majority算法实例

- a, b, a, b, a, b, a, b, a, a, b
- key = a, value = 0

- If value = 0 then
  - key ← x, value = 1
- Else if key = x then
  - value = value + 1
- Else
  - value = value - 1

Majority算法实例

- a, b, a, b, a, b, a, b, a, a, b
- key = a, value = 1

- If value = 0 then
  - key ← x, value = 1
- Else if key = x then
  - value = value + 1
- Else
  - value = value − 1

Majority算法实例

- a, b, a, b, a, b, a, b, a, a, b
- key = a, value = 0

- If value = 0 then
  - key ← x, value = 1
- Else if key = x then
  - value = value + 1
- Else
  - value = value - 1

Majority算法实例

- a, b, a, b, a, b, a, b, a, a, b
- key = a, value = 1

- If value = 0 then
  - key ← x, value = 1
- Else if key = x then
  - value = value + 1
- Else
  - value = value - 1

Majority算法实例

- a, b, a, b, a, b, a, b, a, a, b
- key = a, value = 0

- If value = 0 then
  - key ← x, value = 1
- Else if key = x then
  - value = value + 1
- Else
  - value = value - 1

Majority算法实例

- a, b, a, b, a, b, a, b, a, a, b
- key = a, value = 1

- If value = 0 then
  - key ← x, value = 1
- Else if key = x then
  - value = value + 1
- Else
  - value = value - 1

Majority算法实例

- a, b, a, b, a, b, a, b, a, a, b
- key = a, value = 2

- If value = 0 then
  - key ← x, value = 1
- Else if key = x then
  - value = value + 1
- Else
  - value = value - 1

Majority算法实例

- a, b, a, b, a, b, a, b, a, a, b
- key = a, value = 1

- If value = 0 then
  - key ← x, value = 1
- Else if key = x then
  - value = value + 1
- Else
  - value = value - 1

Majority算法实例

- a, b, a, b, a, b, a, b, a, a, b
- key = a, value = 1
- 存在多数元素a
- If value = 0 then
  - key ← x, value = 1
- Else if key = x then
  - value = value + 1
- Else
  - value = value - 1

Majority算法实例

- a, b, a, c, a, e, a, d, a, f, d
- key = $\varnothing$, value = 0

- If value = 0 then
  - key ← x, value = 1
- Else if key = x then
  - value = value + 1
- Else
  - value = value - 1

多种元素且无任意元素频率过半

MAJORITY不适用

Majority算法实例

- a, b, a, c, a, e, a, d, a, f, d
- key = a, value = 1

- If value = 0 then
  - key ← x, value = 1
- Else if key = x then
  - value = value + 1
- Else
  - value = value - 1

Majority算法实例

- a, b, a, c, a, e, a, d, a, f, d
- key = a, value = 0

- If value = 0 then
  - key ← x, value = 1
- Else if key = x then
  - value = value + 1
- Else
  - value = value - 1

Majority算法实例

- a, b, a, c, a, e, a, d, a, f, d
- key = a, value = 1

- If value = 0 then
  - key ← x, value = 1
- Else if key = x then
  - value = value + 1
- Else
  - value = value - 1

Majority算法实例

- a, b, a, c, a, e, a, d, a, f, d
- key = a, value = 0

- If value = 0 then
  - key ← x, value = 1
- Else if key = x then
  - value = value + 1
- Else
  - value = value - 1

Majority算法实例

- a, b, a, c, a, e, a, d, a, f, d
- key = a, value = 1

- If value = 0 then
  - key ← x, value = 1
- Else if key = x then
  - value = value + 1
- Else
  - value = value - 1

Majority算法实例

- a, b, a, c, a, e, a, d, a, f, d
- key = a, value = 0

- If value = 0 then
  - key ← x, value = 1
- Else if key = x then
  - value = value + 1
- Else
  - value = value - 1

Majority算法实例

- a, b, a, c, a, e, a, d, a, f, d
- key = a, value = 1

- If value = 0 then
  - key ← x, value = 1
- Else if key = x then
  - value = value + 1
- Else
  - value = value - 1

Majority算法实例

- a, b, a, c, a, e, a, d, a, f, d
- key = a, value = 0

- If value = 0 then
  - key ← x, value = 1
- Else if key = x then
  - value = value + 1
- Else
  - value = value - 1

Majority算法实例

- a, b, a, c, a, e, a, d, a, f, d
- key = a, value = 1

- If value = 0 then
  - key ← x, value = 1
- Else if key = x then
  - value = value + 1
- Else
  - value = value - 1

Majority算法实例

- a, b, a, c, a, e, a, d, a, f, d
- key = a, value = 0

- If value = 0 then
  - key ← x, value = 1
- Else if key = x then
  - value = value + 1
- Else
  - value = value - 1

Majority算法实例

- a, b, a, c, a, e, a, d, a, f, d
- key = d, value = 1

- If value = 0 then
  - key ← x, value = 1
- Else if key = x then
  - value = value + 1
- Else
  - value = value - 1

Majority算法实例

- a, b, a, c, a, e, a, d, a, a, d
- key = d, value = 1
- d并非多数元素
- 需要检查. 但对于流数据, 无法检查

- If value = 0 then
  - key ← x, value = 1
- Else if key = x then
  - value = value + 1
- Else
  - value = value - 1

过半数Majority算法扩展(>1/k)

- 问题：如何找出数据流S中出现频率超过s=1/k的元素
  - |S|很大，无法记录每个元素的出现次数

过半数Majority算法扩展(>1/k)

- 问题：如何找出数据流S中出现频率超过s=1/k的元素
  - |S|很大,无法记录每个元素的出现次数

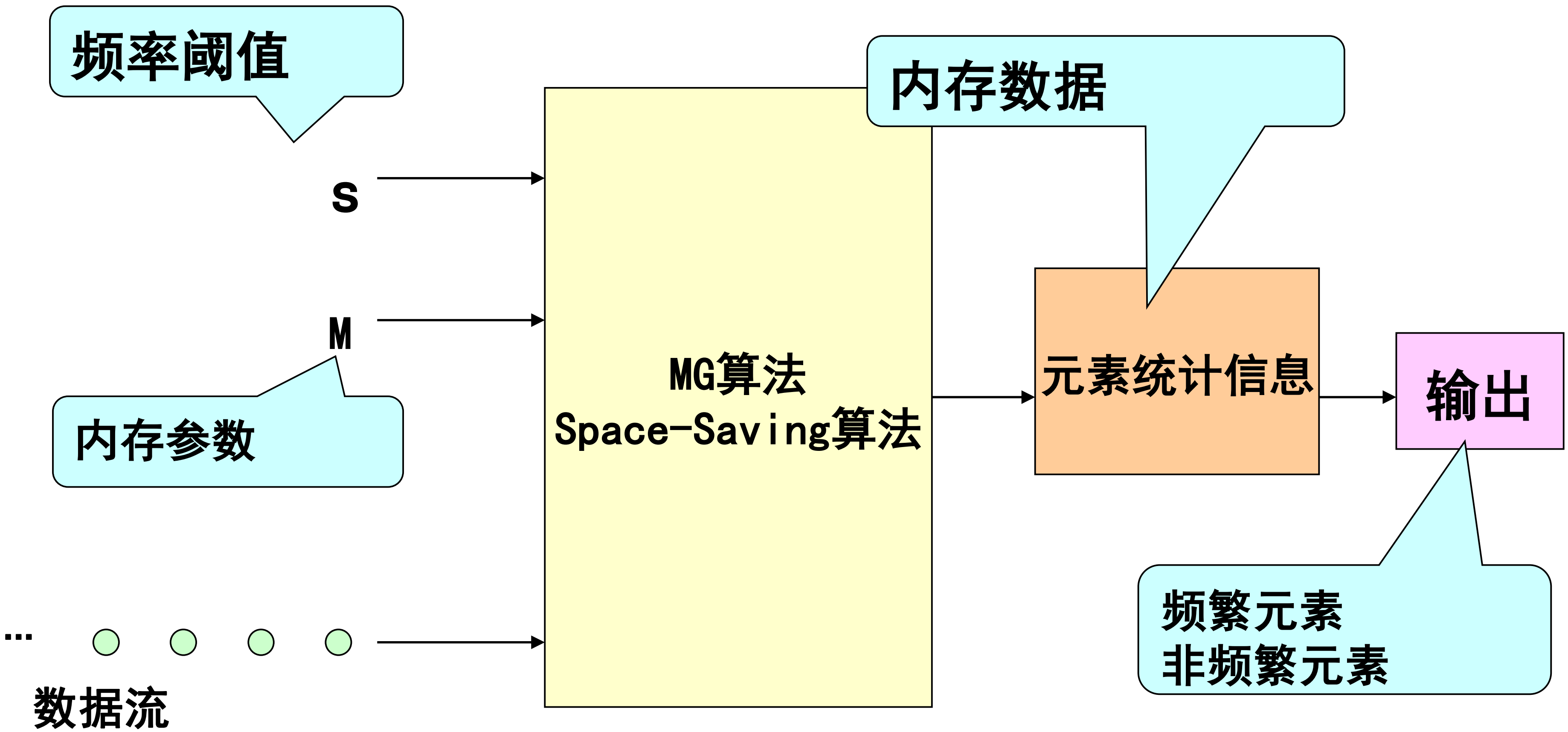
- 输入：
  - 数据流$S=x_1, x_2, \cdots, x_n$, $x_i \in U$
- 输出：S中所有出现频率超过s=1/k的元素

- 算法：
  - MG算法
  - Space-Saving算法

课程提纲

- 数据流简介
- 基础性问题
- 确定性算法
  - Majority算法
  - Misra-Gries(MG)算法
  - Space-Saving算法
  - Lossy Counting算法
- 随机性算法
  - Sticky Sampling算法
  - Count-Min(CM)Sketch算法

MG算法与Space-Saving算法框架



Misra-Gries(MG)算法

- MG:Update(x,w)
  - If $x \in T$
    - $w_x \leftarrow w_x + w$
  - Else
    - $T \leftarrow T \cup \{x\}$
    - $w_x \leftarrow w$
    - If $|T| > k$ then
      - $m = \min(\{w_x : x \in T\})$
      - Forall $j \in T$ do
        - $w_j \leftarrow w_j - m$
        - If $w_j = 0$ then $T \leftarrow T \setminus \{j\}$

Misra-Gries(MG)算法

- MG:Update(x, w)
  - If $x \in T$
    - $w_x \leftarrow w_x + w$
  - Else
    - $T \leftarrow T \cup \{x\}$
    - $w_x \leftarrow w$
    - If $|T| > k$ then
      - $m = \min(\{w_x : x \in T\})$
      - Forall $j \in T$ do
        - $w_j \leftarrow w_j - m$
        - If $w_j = 0$ then $T \leftarrow T \setminus \{j\}$

对于已经在T中的元素，直接更新其权重

Misra-Gries (MG) 算法

对于不在T中的元素，将其加入T并初始化其权重

- MG:Update(x, w)
  - If $x \in T$
    - $w_x \leftarrow w_x + w$
  - Else
    - $T \leftarrow T \cup \{x\}$
    - $w_x \leftarrow w$
    - If $|T| > k$ then
      - $m = \min(\{w_x : x \in T\})$
      - Forall $j \in T$ do
        - $w_j \leftarrow w_j - m$
        - If $w_j = 0$ then $T \leftarrow T \setminus \{j\}$

Misra-Gries(MG)算法

如果|T|>k，将T中每个元素的权重减去T中元素的最小权重，并删除权重为0的元素

- MG:Update(x, w)
  - If $x \in T$
    - $w_x \leftarrow w_x + w$
  - Else
    - $T \leftarrow T \cup \{x\}$
    - $w_x \leftarrow w$
    - If $|T|>k$ then
      - $m=\min(\{w_x : x \in T\})$
      - Forall $j \in T$ do
        - $w_j \leftarrow w_j - m$
        - If $w_j=0$ then $T \leftarrow T \setminus \{j\}$

MG算法实例

- a, b, a, c, d, e, a, d, f, a, d
- T={(−, 0), (−, 0), (−, 0)}

内存参数|T|=3

- If $x \in T$
  - $w_x \leftarrow w_x + w$
- Else
  - $T \leftarrow T \cup \{x\}$
  - $w_x \leftarrow w$
  - If $|T| > k$ then
    - $m = \min(\{w_x : x \in T\})$
    - Forall $j \in T$ do
      - $w_j \leftarrow w_j - m$
      - If $w_j = 0$ then $T \leftarrow T \setminus \{j\}$

MG算法实例

- a, b, a, c, d, e, a, d, f, a, d
- T={(a, 1), (–, 0), (–, 0)}

内存参数|T|=3

- If $x \in T$
  - $w_x \leftarrow w_x + w$
- Else
  - $T \leftarrow T \cup \{x\}$
  - $w_x \leftarrow w$
  - If $|T|>k$ then
    - $m=\min(\{w_x : x \in T\})$
    - Forall $j \in T$ do
      - $w_j \leftarrow w_j - m$
      - If $w_j=0$ then $T \leftarrow T \setminus \{j\}$

MG算法实例

- a, b, a, c, d, e, a, d, f, a, d
- T={(a, 1), (b, 1), (-, 0)}

内存参数|T|=3

- If $x \in T$
  - $w_x \leftarrow w_x + w$
- Else
  - $T \leftarrow T \cup \{x\}$
  - $w_x \leftarrow w$
  - If $|T|>k$ then
    - $m=\min(\{w_x : x \in T\})$
    - Forall $j \in T$ do
      - $w_j \leftarrow w_j - m$
      - If $w_j=0$ then $T \leftarrow T \setminus \{j\}$

MG算法实例

- a, b, a, c, d, e, a, d, f, a, d
- T={(a, 2), (b, 1), (-, 0)}

内存参数|T|=3

- If $x \in T$
  - $w_x \leftarrow w_x + w$
- Else
  - $T \leftarrow T \cup \{x\}$
  - $w_x \leftarrow w$
  - If $|T| > k$ then
    - $m = \min(\{w_x : x \in T\})$
    - Forall $j \in T$ do
      - $w_j \leftarrow w_j - m$
      - If $w_j = 0$ then $T \leftarrow T \setminus \{j\}$

MG算法实例

- a, b, a, c, d, e, a, d, f, a, d
- T={(a, 2), (b, 1), (c, 1)}

内存参数|T|=3

- If $x \in T$
  - $w_x \leftarrow w_x + w$
- Else
  - $T \leftarrow T \cup \{x\}$
  - $w_x \leftarrow w$
  - If $|T| > k$ then
    - $m = \min(\{w_x : x \in T\})$
    - Forall $j \in T$ do
      - $w_j \leftarrow w_j - m$
      - If $w_j = 0$ then $T \leftarrow T \setminus \{j\}$

MG算法实例

- a， b， a， c， d， e， a， d， f， a， d
- T={(a, 2), (b, 1), (c, 1), (d, 1)}

内存参数|T|=3

- If $x \in T$
  - $w_x \leftarrow w_x + w$
- Else
  - $T \leftarrow T \cup \{x\}$
  - $w_x \leftarrow w$
  - If $|T|>k$ then
    - $m=\min(\{w_x : x \in T\})$
    - Forall $j \in T$ do
      - $w_j \leftarrow w_j - m$
      - If $w_j=0$ then $T \leftarrow T \setminus \{j\}$

MG算法实例

- a, b, a, c, d, e, a, d, f, a, d
- T={(a, 1), ~~(b, 0), (c, 0), (d, 0)~~}

内存参数|T|=3

- If $x \in T$
  - $w_x \leftarrow w_x + w$
- Else
  - $T \leftarrow T \cup \{x\}$
  - $w_x \leftarrow w$
  - If $|T| > k$ then
    - $m = \min(\{w_x : x \in T\})$
    - Forall $j \in T$ do
      - $w_j \leftarrow w_j - m$
      - If $w_j = 0$ then $T \leftarrow T \setminus \{j\}$

MG算法实例

- a, b, a, c, d, e, a, d, f, a, d
- T={(a, 1), (–, 0), (–, 0)}

内存参数|T|=3

- If $x \in T$
  - $w_x \leftarrow w_x + w$
- Else
  - $T \leftarrow T \cup \{x\}$
  - $w_x \leftarrow w$
  - If $|T| > k$ then
    - $m = \min(\{w_x : x \in T\})$
    - Forall $j \in T$ do
      - $w_j \leftarrow w_j - m$
      - If $w_j = 0$ then $T \leftarrow T \setminus \{j\}$

MG算法实例

- a, b, a, c, d, e, a, d, f, a, d
- T={(a,1),(e,1),(-,0)}

内存参数|T|=3

- If $x \in T$
  - $w_x \leftarrow w_x + w$
- Else
  - $T \leftarrow T \cup \{x\}$
  - $w_x \leftarrow w$
  - If $|T|>k$ then
    - $m=\min(\{w_x : x \in T\})$
    - Forall $j \in T$ do
      - $w_j \leftarrow w_j - m$
      - If $w_j=0$ then $T \leftarrow T \setminus \{j\}$

MG算法实例

- a, b, a, c, d, e, a, d, f, a, d
- T={(a, 2), (e, 1), (–, 0)}

内存参数|T|=3

- If $x \in T$
  - $w_x \leftarrow w_x + w$
- Else
  - $T \leftarrow T \cup \{x\}$
  - $w_x \leftarrow w$
  - If $|T| > k$ then
    - $m = \min(\{w_x : x \in T\})$
    - Forall $j \in T$ do
      - $w_j \leftarrow w_j - m$
      - If $w_j = 0$ then $T \leftarrow T \setminus \{j\}$

MG算法实例

- a, b, a, c, d, e, a, d, f, a, d
- T={(a, 2), (e, 1), (d, 1)}

内存参数|T|=3

- If $x \in T$
  - $w_x \leftarrow w_x + w$
- Else
  - $T \leftarrow T \cup \{x\}$
  - $w_x \leftarrow w$
  - If $|T|>k$ then
    - $m=\min(\{w_x : x \in T\})$
    - Forall $j \in T$ do
      - $w_j \leftarrow w_j - m$
      - If $w_j=0$ then $T \leftarrow T \setminus \{j\}$

MG算法实例

- a, b, a, c, d, e, a, d, f, a, d
- T={(a,2),(e,1),(d,1),(f,1)}

内存参数|T|=3

- If $x \in T$
  - $w_x \leftarrow w_x + w$
- Else
  - $T \leftarrow T \cup \{x\}$
  - $w_x \leftarrow w$
  - If $|T|>k$ then
    - $m=\min(\{w_x : x \in T\})$
    - Forall $j \in T$ do
      - $w_j \leftarrow w_j - m$
      - If $w_j=0$ then $T \leftarrow T \setminus \{j\}$

MG算法实例

- a, b, a, c, d, e, a, d, f, a, d
- T={(a, 1), ~~(e, 0), (d, 0), (f, 0)~~}

内存参数|T|=3

- If $x \in T$
  - $w_x \leftarrow w_x + w$
- Else
  - $T \leftarrow T \cup \{x\}$
  - $w_x \leftarrow w$
  - If $|T|>k$ then
    - $m=\min(\{w_x : x \in T\})$
    - Forall $j \in T$ do
      - $w_j \leftarrow w_j - m$
      - If $w_j=0$ then $T \leftarrow T \backslash \{j\}$

MG算法实例

- a, b, a, c, d, e, a, d, f, a, d
- T={(a, 1), (–, 0), (–, 0)}

内存参数|T|=3

- If $x \in T$
  - $w_x \leftarrow w_x + w$
- Else
  - $T \leftarrow T \cup \{x\}$
  - $w_x \leftarrow w$
  - If $|T|>k$ then
    - $m=\min(\{w_x : x \in T\})$
    - Forall $j \in T$ do
      - $w_j \leftarrow w_j - m$
      - If $w_j=0$ then $T \leftarrow T \setminus \{j\}$

MG算法实例

- a, b, a, c, d, e, a, d, f, a, d
- T={(a, 2), (−, 0), (−, 0)}

内存参数|T|=3

- If $x \in T$
  - $w_x \leftarrow w_x + w$
- Else
  - $T \leftarrow T \cup \{x\}$
  - $w_x \leftarrow w$
  - If $|T|>k$ then
    - $m=\min(\{w_x : x \in T\})$
    - Forall $j \in T$ do
      - $w_j \leftarrow w_j - m$
      - If $w_j=0$ then $T \leftarrow T \setminus \{j\}$

MG算法实例

- a, b, a, c, d, e, a, d, f, a, d
- T={(a, 2), (d, 1), (−, 0)}

内存参数|T|=3

- If x∈T
  - $w_x \leftarrow w_x + w$
- Else
  - $T \leftarrow T \cup \{x\}$
  - $w_x \leftarrow w$
  - If |T|>k then
    - $m = \min(\{w_x : x \in T\})$
    - Forall j∈T do
      - $w_j \leftarrow w_j - m$
      - If $w_j = 0$ then $T \leftarrow T \setminus \{j\}$

MG算法实例

- a, b, a, c, d, e, a, d, f, a, d
- T={(a,2),(d,1),(-,0)}

- T中每个元素的估计误差不超过W/|T|=11/3
  - W为流中所有元素权重加和

- 使用T找出流中出现频率超过1/2的元素
- 流中出现频率超过1/2的元素为a
  - a: 2 + 11/3 > 11/2

Space-Saving算法

- Space-Saving:Update(x, w)
  - If $x \in T$
    - $w_x \leftarrow w_x + w$
  - Else
    - $y \leftarrow \mathrm{argmin}_{y \in T} w_j$
    - $w_x \leftarrow w_y + w$
    - $T \leftarrow T \cup \{x\} \setminus \{y\}$

Space-Saving算法

- Space-Saving:Update(x, w)
  - If $x \in T$
    - $w_x \leftarrow w_x + w$
  - Else
    - $y \leftarrow \mathrm{argmin}_{y \in T} w_j$
    - $w_x \leftarrow w_y + w$
    - $T \leftarrow T \cup \{x\} \setminus \{y\}$

对于已经在T中的元素，直接更新其权重

Space-Saving算法

- Space-Saving:Update(x,w)
  - If $x \in T$
    - $w_x \leftarrow w_x + w$
  - Else
    - $y \leftarrow \mathrm{argmin}_{y \in T} w_j$
    - $w_x \leftarrow w_y + w$
    - $T \leftarrow T \cup \{x\} \setminus \{y\}$

对于不在T中的元素，将其权重加上T中元素的最小权重，并取代T中权重最小的元素

Space-Saving算法实例

- a, b, a, c, d, e, a, d, f, a, d
- T={(-, 0), (-, 0), (-, 0), (-, 0)}

内存参数|T|=4

- If $x \in T$
  - $w_x \leftarrow w_x + w$
- Else
  - $y \leftarrow \operatorname{argmin}_{y \in T} w_j$
  - $w_x \leftarrow w_y + w$
  - $T \leftarrow T \cup \{x\} \setminus \{y\}$

Space-Saving算法实例

- a, b, a, c, d, e, a, d, f, a, d
- T={(a, 1), (-, 0), (-, 0), (-, 0)}

内存参数|T|=4

- If $x \in T$
  - $w_x \leftarrow w_x + w$
- Else
  - $y \leftarrow \mathrm{argmin}_{y \in T} w_j$ (y = -)
  - $w_x \leftarrow w_y + w$ (w_y = 0)
  - $T \leftarrow T \cup \{x\} \setminus \{y\}$

Space-Saving算法实例

- a, b, a, c, d, e, a, d, f, a, d
- T={(a, 1), (b, 1), (-, 0), (-, 0)}

内存参数|T|=4

- If $x \in T$
  - $w_x \leftarrow w_x + w$
- Else
  - $y \leftarrow \arg\min_{y \in T} w_j$ (y = -)
  - $w_x \leftarrow w_y + w$ (w_y = 0)
  - $T \leftarrow T \cup \{x\} \setminus \{y\}$

Space-Saving算法实例

- a, b, a, c, d, e, a, d, f, a, d
- T={(a, 2), (b, 1), (-, 0), (-, 0)}

内存参数|T|=4

- If $x \in T$
  - $w_x \leftarrow w_x + w$
- Else
  - $y \leftarrow \text{argmin}_{y \in T} w_j$
  - $w_x \leftarrow w_y + w$
  - $T \leftarrow T \cup \{x\} \setminus \{y\}$

Space-Saving算法实例

- a, b, a, c, d, e, a, d, f, a, d
- T={(a, 2), (b, 1), (c, 1), (-, 0)}

内存参数|T|=4

- If $x \in T$
  - $w_x \leftarrow w_x + w$
- Else
  - $y \leftarrow \mathrm{argmin}_{y \in T} w_j$ (y = -)
  - $w_x \leftarrow w_y + w$ (w_y = 0)
  - $T \leftarrow T \cup \{x\} \setminus \{y\}$

Space-Saving算法实例

- a, b, a, c, d, e, a, d, f, a, d
- T={(a, 2), (b, 1), (c, 1), (d, 1)}

内存参数|T|=4

- If $x \in T$
  - $w_x \leftarrow w_x + w$
- Else
  - $y \leftarrow \mathrm{argmin}_{y \in T} w_j$ (y = -)
  - $w_x \leftarrow w_y + w$ (w_y = 0)
  - $T \leftarrow T \cup \{x\} \setminus \{y\}$

Space-Saving算法实例

- a, b, a, c, d, e, a, d, f, a, d
- T={(a, 2), (e, 2), (c, 1), (d, 1)}

内存参数|T|=4

- If $x \in T$
  - $w_x \leftarrow w_x + w$
- Else
  - $y \leftarrow \mathrm{argmin}_{y \in T} w_j$ (y = b)
  - $w_x \leftarrow w_y + w$ (w_y = 1)
  - $T \leftarrow T \cup \{x\} \setminus \{y\}$

Space-Saving算法实例

- a, b, a, c, d, e, a, d, f, a, d
- T={(a, 3), (e, 2), (c, 1), (d, 1)}

内存参数|T|=4

- If $x \in T$
  - $w_x \leftarrow w_x + w$
- Else
  - $y \leftarrow \mathrm{argmin}_{y \in T} w_j$
  - $w_x \leftarrow w_y + w$
  - $T \leftarrow T \cup \{x\} \setminus \{y\}$

Space-Saving算法实例

- a, b, a, c, d, e, a, d, f, a, d
- T={(a, 3), (e, 2), (c, 1), (d, 2)}

内存参数|T|=4

- If $x \in T$
  - $w_x \leftarrow w_x + w$
- Else
  - $y \leftarrow \mathrm{argmin}_{y \in T} w_j$
  - $w_x \leftarrow w_y + w$
  - $T \leftarrow T \cup \{x\} \setminus \{y\}$

Space-Saving算法实例

- a, b, a, c, d, e, a, d, f, a, d
- T={(a, 3), (e, 2), ~~(c, 1)~~, (d, 2)}
- T={(a, 3), (e, 2), (f, 2), (d, 2)}
- If $x \in T$
  - $w_x \leftarrow w_x + w$
- Else
  - $y \leftarrow \arg\min_{y \in T} w_j$ (y = c)
  - $w_x \leftarrow w_y + w$ (w_y = 1)
  - $T \leftarrow T \cup \{x\} \setminus \{y\}$

内存参数|T|=4

Space-Saving算法实例

- a, b, a, c, d, e, a, d, f, a, d
- T={(a, 4), (e, 2), (f, 2), (d, 2)}

内存参数|T|=4

- If $x \in T$
  - $w_x \leftarrow w_x + w$
- Else
  - $y \leftarrow \mathrm{argmin}_{y \in T} w_j$
  - $w_x \leftarrow w_y + w$
  - $T \leftarrow T \cup \{x\} \setminus \{y\}$

Space-Saving算法实例

- a, b, a, c, d, e, a, d, f, a, d
- T={(a, 4), (e, 2), (f, 2), (d, 3)}

内存参数|T|=4

- If $x \in T$
  - $w_x \leftarrow w_x + w$
- Else
  - $y \leftarrow \mathrm{argmin}_{y \in T} w_j$
  - $w_x \leftarrow w_y + w$
  - $T \leftarrow T \cup \{x\} \setminus \{y\}$

Space-Saving算法实例

- a, b, a, c, d, e, a, d, f, a, d
- T={(a,4),(e,2),(f,2),(d,3)}

- T中每个元素出现次数估计误差不超过W/|T|=11/4
  - W为流中所有元素权重加和

- 使用T找出流中出现频率超过1/3的元素
- 流中出现频率超过1/3的元素为a,e,f,d
  - a: 4+11/4 > 11/3
  - e: 2+11/4 > 11/3,
  - f: 2+11/3 > 11/3
  - d: 3+11/4 > 11/3

MG算法与Space-Saving算法关系

- $T_{MG}=\{(a,2),(d,1),(-,0)\}$
- $T_{Space\text{-}Saving}=\{(a,4),(e,2),(f,2),(d,3)\}$

- $T_{MG}=\{(a,2),(d,1),(-,0)\}$
- $T_{Space\text{-}Saving}=\{(a,2),(e,0),(f,0),(d,1)\}$

- $T_{MG}=\{(a,2),(d,1)\}$
- $T_{Space\text{-}Saving}=\{(a,2),(d,1)\}$

- MG算法$|T|=k$与Space-Saving算法$|T|=k+1$等价

MG算法与Space-Saving算法分析

- 内存参数为M,即|T|=M
  - 对每个元素频率的估计误差最大为1/M

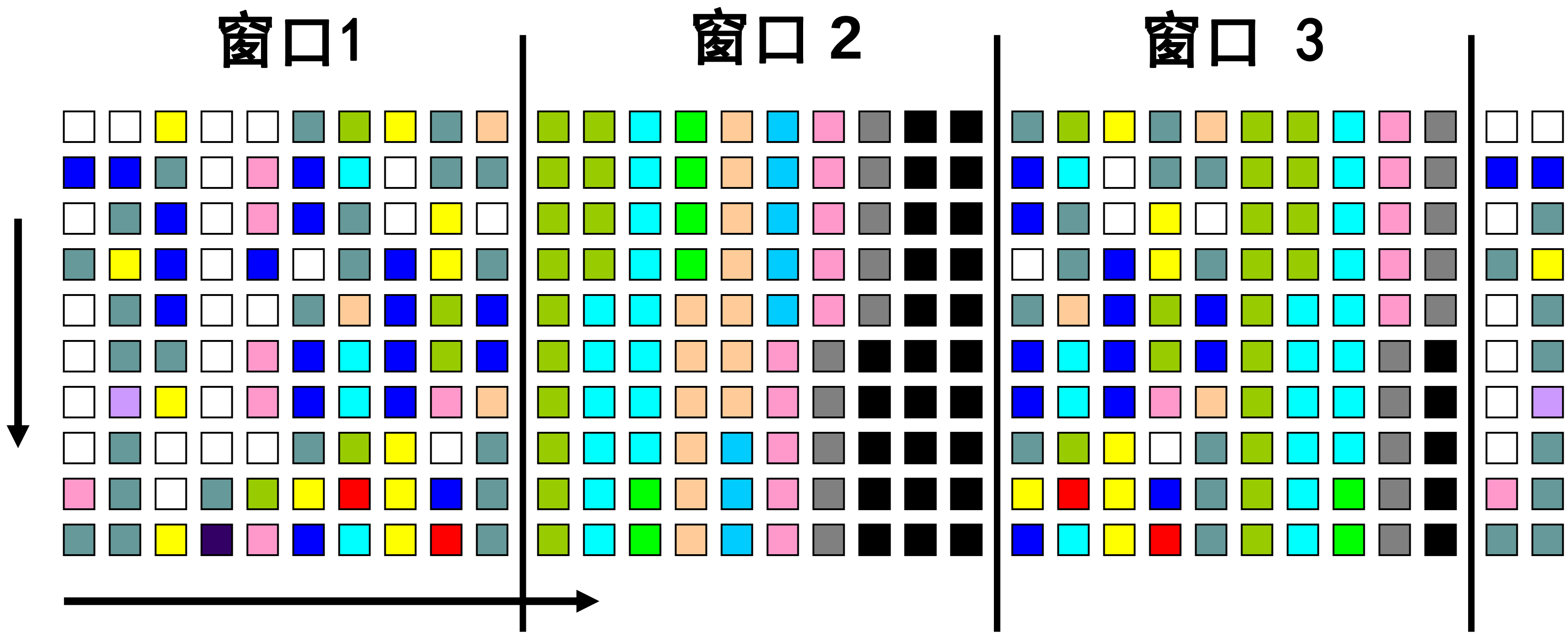
- 使用T寻找流中出现频率超过s的元素
  - 当满足E≤ 1/M ≤ ε 时
  - 流中所有出现频率超过s的元素都不会漏掉
    - 即不存在假阴性(No false negative)

课程提纲

- 数据流简介
- 基础性问题
- 确定性算法
  - Majority算法
  - Misra-Gries(MG)算法
  - Space-Saving算法
  - Lossy Counting算法
- 随机性算法
  - Sticky Sampling算法
  - Count-Min(CM)Sketch算法

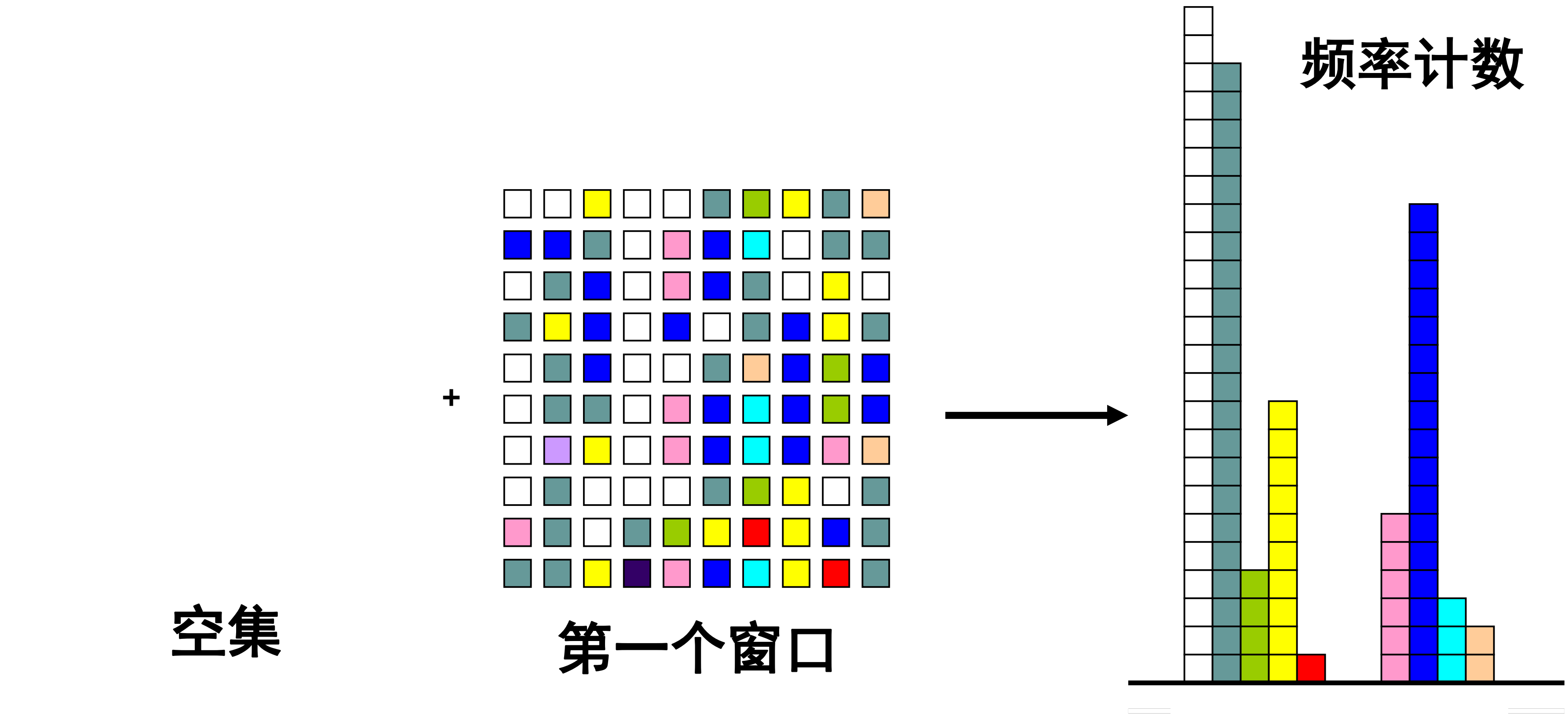
Lossy Counting算法核心思想

- 第一步：基于窗口的数据流切分



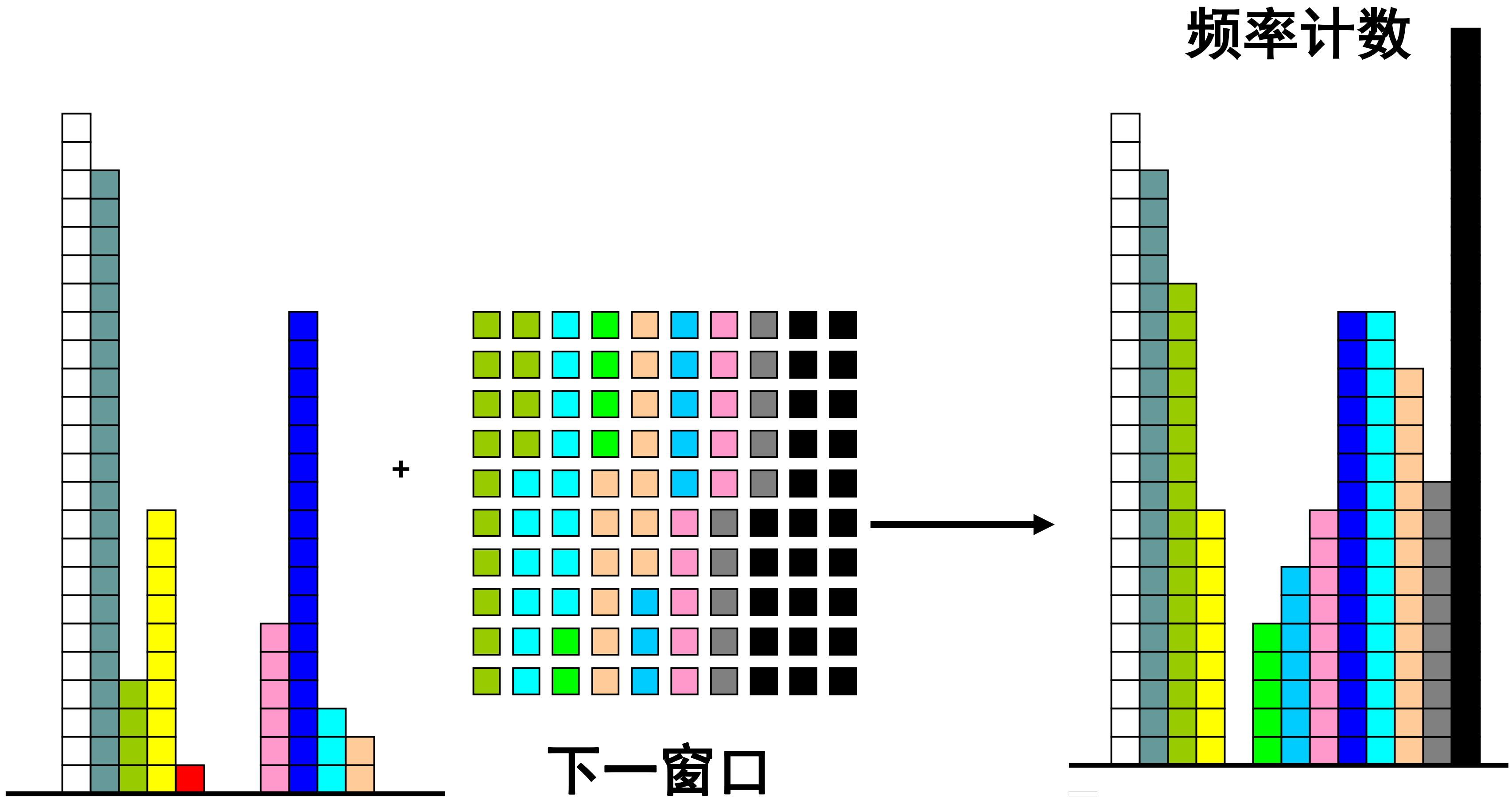
Lossy Counting算法核心思想

- 第二步：每到窗口边界，每项计数减一

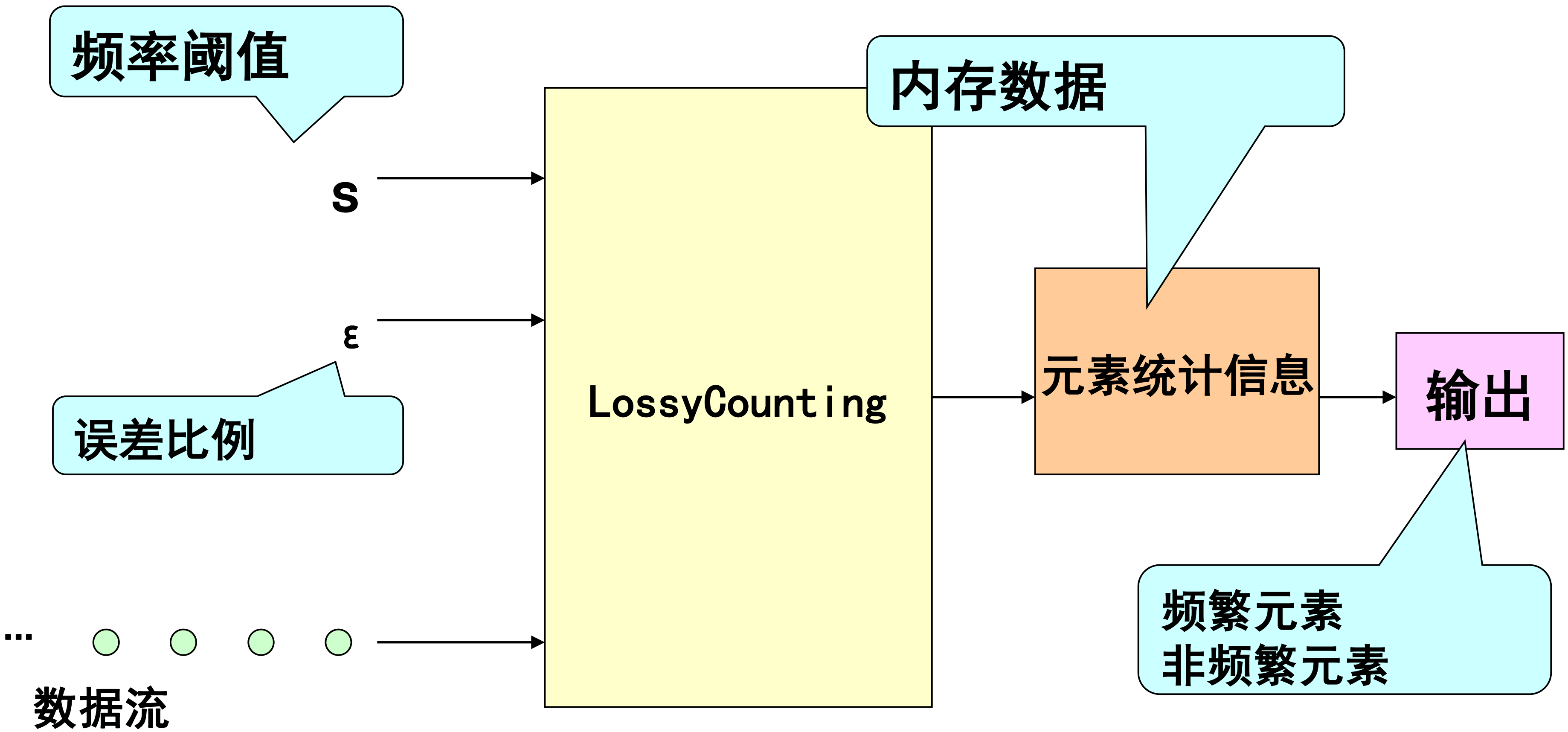


Lossy Counting算法核心思想

- 第一步：每到窗口边界，每项计数减一



Lossy Counting算法框架



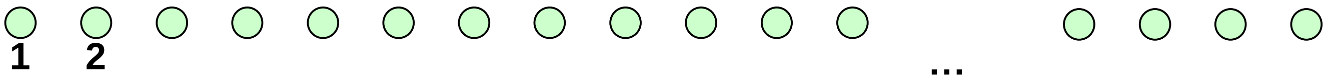
Lossy Counting算法

问题定义：统计流数据中出现频率大于给定值s的元素。

- **算法框架**
  - 根据参数设置流数据窗口大小。
  - 对于顺序到来的一个窗口内的数据更新内存中数据结构。
  - 每执行完一个窗口内的数据，更新内存中的数据结构（以减少空间），更新参数。

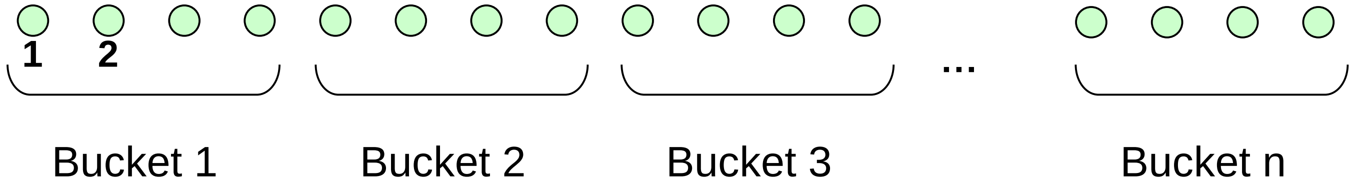
Lossy Counting算法

问题定义：统计流数据中出现频率大于给定值s的元素。



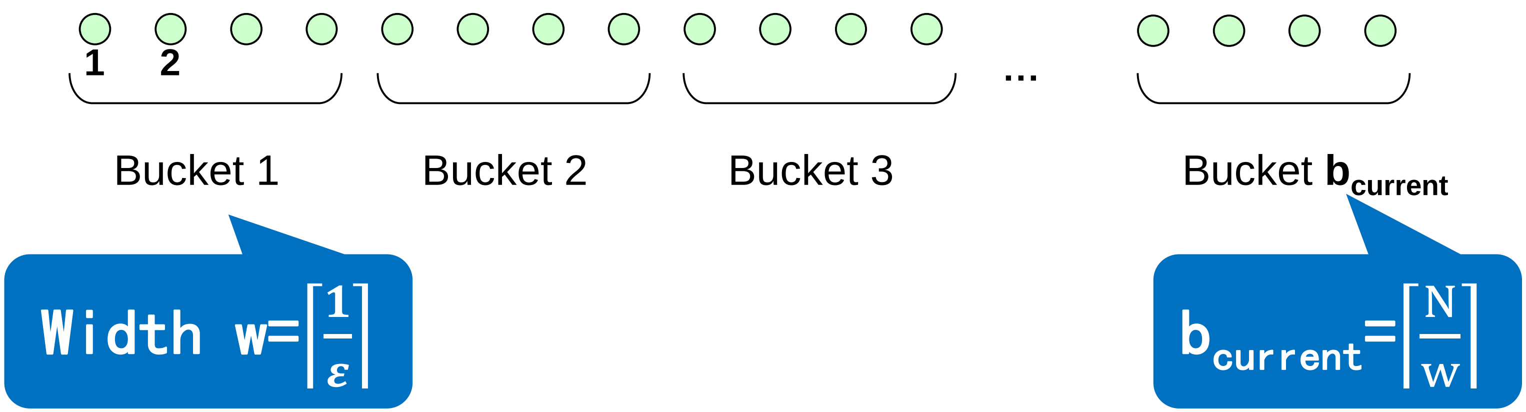
Lossy Counting算法

问题定义：统计流数据中出现频率大于给定值s的元素。



Lossy Counting算法

问题定义：统计流数据中出现频率大于给定值s的元素。



Lossy Counting算法

问题定义：统计流数据中出现频率大于给定值s的元素。

- **算法介绍**
  - 内存中保存部分元素、其出现次数以及该元素的最大误差(e, f, Δ)，对于数据流顺序到来的元素，若数据出现的元素已存在，将次数加一；若不存在，则将(e, 1, Δ)放入内存。
    - 窗口大小不变，动态更新内存存储信息。
    - 数据结构：字典
    - 返回数据结构中频度大于sN-εN的元素。
    - 确定性算法
    - 至多使用$\frac{1}{\varepsilon}\log(\varepsilon N)$个key-value对
- **相关参数**
    - 窗口大小参数$w = \lceil 1/\varepsilon \rceil$

Lossy Counting算法

问题定义：统计流数据中出现频率大于给定值s的元素。

元素

元素自被添加之后出现的次数

1. D: Empty set
   - Will contain (e, f, Δ)

f的最大可能误差

2. When data e arrives,
   - If e exists in D,
     - Increment f in (e, f, Δ)
   - If e does not exist in D,
3. s, Remove some entries in D whenever
   $N \equiv 0 \bmod w$
   (i.e., whenever it reaches the bucket boundary)
   The rule of deletion is:
   (e, f, Δ) is deleted if
   $f + \Delta <= b_{current}$
4. [Output] Get a list of items where
   $f + \varepsilon N >= sN$

Lossy Counting算法

ε=0.2, 窗口大小w=[1/ε]=5,窗口个数b=1

a

s

元素	频度	误差

1. D: Empty set
   - Will contain (e, f, Δ)
2. When data e arrives,
   - If e exists in D,
     - Increment f in (e, f, Δ)
   - If e does not exist in D,
3. s, Remove some entries in D whenever $N \equiv 0 \mod w$
   The rule of deletion is:
   (e, f, Δ) is deleted if
   $f + \Delta <= b_{current}$
4. [Output] Get a list of items where
   $f + \varepsilon N >= sN$

Lossy Counting算法

ε=0.2, 窗口大小w=[1/ε]=5,窗口个数b=1

a b

s

元素	频度	误差
a	1	0

1. D: Empty set
   - Will contain (e, f, Δ)
2. When data e arrives,
   - If e exists in D,
     - Increment f in (e, f, Δ)
   - If e does not exist in D,
3. s, Remove some entries in D whenever
   N $\equiv$ 0 mod w
   The rule of deletion is:
   (e, f, Δ) is deleted if
   $f + \Delta <= b_{current}$
4. [Output] Get a list of items where
   $f + \varepsilon N >= sN$

Lossy Counting算法

ε=0.2, 窗口大小w=[1/ε]=5,窗口个数b=1

a b a

s

元素	频度	误差
a	2	0
b	1	0

1. D: Empty set
   - Will contain (e, f, Δ)
2. When data e arrives,
   - If e exists in D,
     - Increment f in (e, f, Δ)
   - If e does not exist in D,
3. s, Remove some entries in D whenever $N \equiv 0 \bmod w$
   The rule of deletion is:
   (e, f, Δ) is deleted if
   $f + \Delta <= b_{current}$
4. [Output] Get a list of items where
   $f + \varepsilon N >= sN$

Lossy Counting算法

ε=0.2, 窗口大小w=[1/ε]=5,窗口个数b=1

a b a c

s

元素	频度	误差
a	2	0
b	1	0

1. D: Empty set
   - Will contain (e, f, Δ)
2. When data e arrives,
   - If e exists in D,
     - Increment f in (e, f, Δ)
   - If e does not exist in D,
3. s, Remove some entries in D whenever $N \equiv 0 \bmod w$
   The rule of deletion is:
   (e, f, Δ) is deleted if
   $f + \Delta <= b_{current}$
4. [Output] Get a list of items where
   $f + \varepsilon N >= sN$

Lossy Counting算法

ε=0.2, 窗口大小w=[1/ε]=5,窗口个数b=1

a b a c d

s

元素	频度	误差
a	2	0
b	1	0
c	1	0

1. D: Empty set
   - Will contain (e, f, Δ)
2. When data e arrives,
   - If e exists in D,
     - Increment f in (e, f, Δ)
   - If e does not exist in D,
3. s, Remove some entries in D whenever $N \equiv 0 \bmod w$
   The rule of deletion is:
   (e, f, Δ) is deleted if
   $f + \Delta <= b_{current}$
4. [Output] Get a list of items where
   $f + \varepsilon N >= sN$

Lossy Counting算法

ε=0.2, 窗口大小w=[1/ε]=5,窗口个数b=1

a b a c d

s

元素	频度	误差
a	2	0
b	1	0
c	1	0
d	1	0

1. D: Empty set
   - Will contain (e, f, Δ)
2. When data e arrives,
   - If e exists in D,
     - Increment f in (e, f, Δ)
   - If e does not exist in D,
3. s, Remove some entries in D whenever $N \equiv 0 \bmod w$
   The rule of deletion is:
   (e, f, Δ) is deleted if
   $f + \Delta <= b_{current}$
4. [Output] Get a list of items where
   $f + \varepsilon N >= sN$

Lossy Counting算法

ε=0.2, 窗口大小w=[1/ε]=5,窗口个数b=1

a b a c d

s

元素	频度	误差	
a	2	0	2+0>1
b	1	0	1+0<=1, drop
c	1	0	1+0<=1, drop
d	1	0	1+0<=1, drop

1. D: Empty set
   - Will contain (e, f, Δ)
2. When data e arrives,
   - If e exists in D,
     - Increment f in (e, f, Δ)
   - If e does not exist in D,
3. s, Remove some entries in D whenever $N \equiv 0 \bmod w$
   The rule of deletion is:
   (e, f, Δ) is deleted if
   $f + \Delta <= b_{current}$
4. [Output] Get a list of items where
   $f + \varepsilon N >= sN$

Lossy Counting算法

ε=0.2, 窗口大小w=[1/ε]=5,窗口个数b=2

a b a c d | e

s

元素	频度	误差
a	2	0

1. D: Empty set
   - Will contain (e, f, Δ)
2. When data e arrives,
   - If e exists in D,
     - Increment f in (e, f, Δ)
   - If e does not exist in D,
3. s, Remove some entries in D whenever $N \equiv 0 \bmod w$
   The rule of deletion is:
   (e, f, Δ) is deleted if
   $f + \Delta <= b_{current}$
4. [Output] Get a list of items where
   $f + \varepsilon N >= sN$

Lossy Counting算法

ε=0.2, 窗口大小w=[1/ε]=5,窗口个数b=2

a b a c d | e a

s

元素	频度	误差
a	2	0
e	1	1

1. D: Empty set
   - Will contain (e, f, Δ)
2. When data e arrives,
   - If e exists in D,
     - Increment f in (e, f, Δ)
   - If e does not exist in D,
3. s, Remove some entries in D whenever $N \equiv 0 \bmod w$
   The rule of deletion is:
   (e, f, Δ) is deleted if
   $f + \Delta <= b_{current}$
4. [Output] Get a list of items where
   $f + \varepsilon N >= sN$

Lossy Counting算法

ε=0.2, 窗口大小w=[1/ε]=5,窗口个数b=2

a b a c d | e a d

s

元素	频度	误差
a	3	0
e	1	1

1. D: Empty set
   - Will contain (e, f, Δ)
2. When data e arrives,
   - If e exists in D,
     - Increment f in (e, f, Δ)
   - If e does not exist in D,
3. s, Remove some entries in D whenever $N \equiv 0 \bmod w$
   The rule of deletion is:
   (e, f, Δ) is deleted if
   $f + \Delta <= b_{current}$
4. [Output] Get a list of items where
   $f + \varepsilon N >= sN$

Lossy Counting算法

ε=0.2, 窗口大小w=[1/ε]=5,窗口个数b=2

a b a c d | e a d f

s

元素	频度	误差
a	3	0
e	1	1
d	1	1

1. D: Empty set
   - Will contain (e, f, Δ)
2. When data e arrives,
   - If e exists in D,
     - Increment f in (e, f, Δ)
   - If e does not exist in D,
3. s, Remove some entries in D whenever $N \equiv 0 \bmod w$
   The rule of deletion is:
   (e, f, Δ) is deleted if
   $f + \Delta <= b_{current}$
4. [Output] Get a list of items where
   $f + \varepsilon N >= sN$

Lossy Counting算法

ε=0.2, 窗口大小w=[1/ε]=5,窗口个数b=2

a b a c d | e a d f a

s

元素	频度	误差
a	3	0
e	1	1
d	1	1
f	1	1

1. D: Empty set
   - Will contain (e, f, Δ)
2. When data e arrives,
   - If e exists in D,
     - Increment f in (e, f, Δ)
   - If e does not exist in D,
3. s, Remove some entries in D whenever N $\equiv$ 0 mod w
   The rule of deletion is:
   (e, f, Δ) is deleted if
   $f + \Delta <= b_{current}$
4. [Output] Get a list of items where
   $f + \varepsilon N >= sN$

Lossy Counting算法

ε=0.2, 窗口大小w=[1/ε]=5,窗口个数b=2

a b a c d | e a d f a

s

元素	频度	误差
a	4	0
e	1	1
d	1	1
f	1	1

3+0>2

1+1<=2,drop

1+1>=2,drop

1+1>=2,drop

1. D: Empty set
   - Will contain (e, f, Δ)
2. When data e arrives,
   - If e exists in D,
     - Increment f in (e, f, Δ)
   - If e does not exist in D,
3. s, Remove some entries in D whenever $N \equiv 0 \bmod w$
   The rule of deletion is:
   (e, f, Δ) is deleted if
   $f + \Delta <= b_{current}$
4. [Output] Get a list of items where
   $f + \varepsilon N >= sN$

Lossy Counting算法

ε=0.2, 窗口大小w=[1/ε]=5,窗口个数b=2

a b a c d | e a d f a | d

s

元素	频度	误差
a	4	0

1. D: Empty set
   - Will contain (e, f, Δ)
2. When data e arrives,
   - If e exists in D,
     - Increment f in (e, f, Δ)
   - If e does not exist in D,
3. s, Remove some entries in D whenever $N \equiv 0 \bmod w$
   The rule of deletion is:
   (e, f, Δ) is deleted if
   $f + \Delta <= b_{current}$
4. [Output] Get a list of items where
   $f + \varepsilon N >= sN$

Lossy Counting算法

ε=0.2, 窗口大小w=[1/ε]=5,窗口个数b=3

a b a c d | e a d f a | d

s

元素	频度	误差
a	4	0
d	1	2

1. D: Empty set
   - Will contain (e, f, Δ)
2. When data e arrives,
   - If e exists in D,
     - Increment f in (e, f, Δ)
   - If e does not exist in D,
3. s, Remove some entries in D whenever $N \equiv 0 \bmod w$
   The rule of deletion is:
   (e, f, Δ) is deleted if
   $f + \Delta <= b_{current}$
4. [Output] Get a list of items where
   $f + \varepsilon N >= sN$

Lossy Counting算法

ε=0.2, 窗口大小w=[1/ε]=5,窗口个数b=3

a b a c d | e a d f a | d

s

元素	频度	误差
a	4	0
d	1	2

[Output] Get a list of items where
f + εN >= sN

s=0.3, 取使得f>=N(s- ε)的元素，为**a**

课程提纲

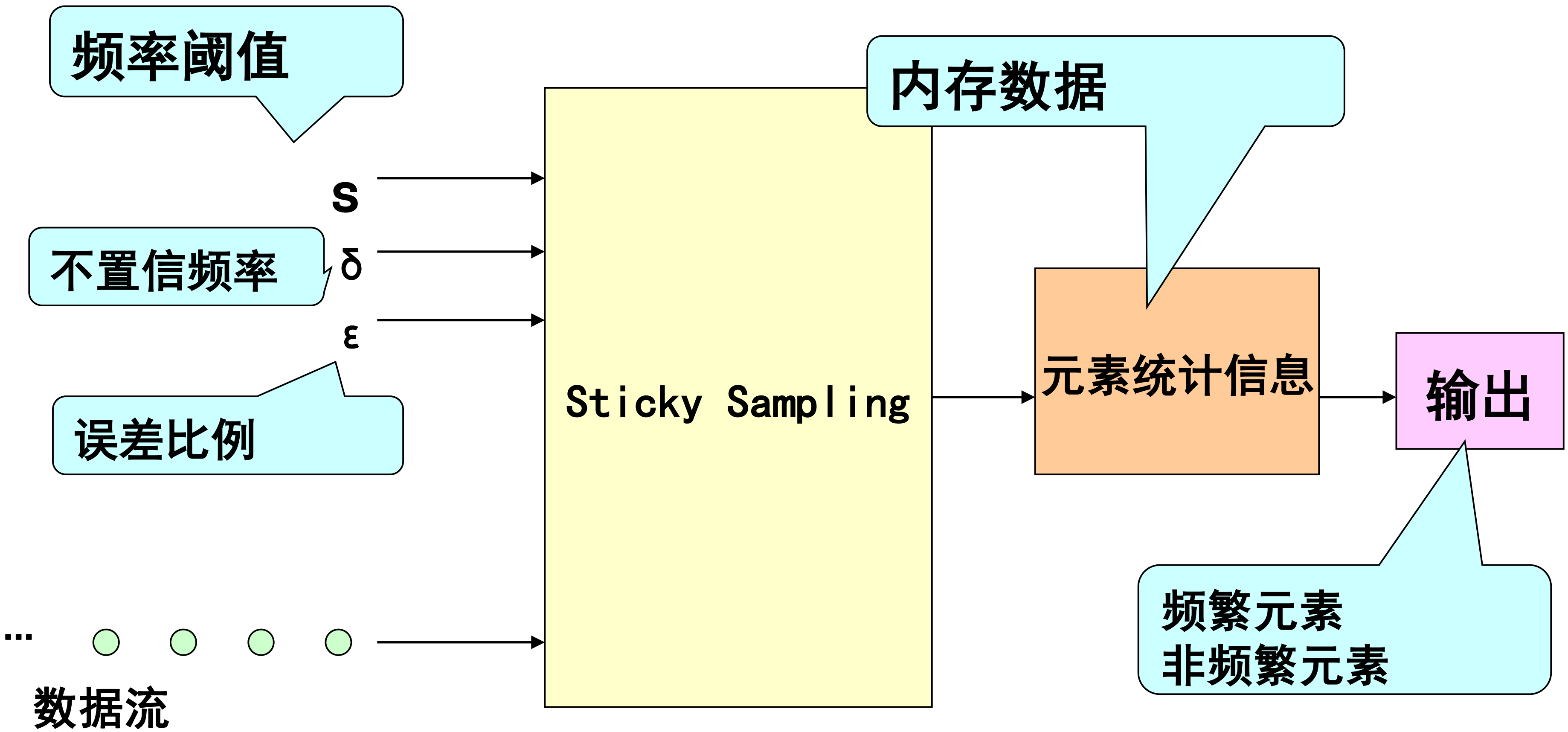
- 数据流简介
- 基础性问题
- 确定性算法
  - Majority算法
  - Misra-Gries(MG)算法
  - Space-Saving算法
  - Lossy Counting算法
- 随机性算法
  - Sticky Sampling算法
  - Count-Min(CM)Sketch算法

Sticky Sampling算法

问题定义：统计流数据中出现频率大于给定值s的元素。

- **算法框架**
  - 基本数据流模型
  - 根据参数设置流数据窗口大小。
  - 对于顺序到来的一个窗口内的数据更新内存中数据结构。
  - 每执行完一个窗口内的数据，更新内存中的数据结构（以减少空间），更新参数。

Sticky Sampling算法框架



Sticky Sampling算法

问题定义：统计流数据中出现频率大于给定值s的元素。

- **算法介绍**
  - 内存中保存部分元素及其出现次数(e, f)，对于数据流顺序到来的元素，数据出现的元素已存在，将次数加一；若不存在，做一次抽样决定是否将其放入内存。
    - 每处理完一个窗口的数据，抽样频率减半，窗口大小加倍，更新数据结构。
    - 返回数据结构中频度大于sN-εN的元素。
    - 至多使用$\frac{1}{\varepsilon}\log(s^{-1}\delta^{-1})$个key-value对

Sticky Sampling算法

问题定义：统计流数据中出现频率大于给定值s的元素。

- **相关参数**
  - 抽样率频率参数r，值随数据流的增加而变化。
  - 置信频率参数δ
  - 样窗口参数 $t = \frac{1}{\varepsilon}\log(s^{-1}\delta^{-1})$，窗口大小依次为2t, 2t, 4t, 8t, …

2t	**1**
2t	**2**
4t	**4**
…	**…**

Sticky Sampling算法

问题定义：统计流数据中出现频率大于给定值s的元素。

- **相关参数**
  - 抽样率频率参数r，值随数 … 而变化。
  - 置信频率参数δ
  - 样窗口参数 $t = \frac{1}{\varepsilon}\log(s^{-1}\delta^{-1})$，窗口大小依次为 2t, 2t, 4t, 8t, …

e.g.
s = 0.02
ε = 0.01
δ= 0.1

t = 622

1~1244	**2t**	**1**
1245~2488	**2t**	**2**
2489~4976	**4t**	**4**
	…	**…**

Sticky Sampling算法

问题定义：统计流数据中出现频率大于给定值s的元素。

元素 估计频度

1. S: empty list
   → will contain (e, f)
2. When data e arrives,
   - if e exists in S, increment f in (e, f)
   - if e does not exist in S, add entry (e, 1) with prob. 1/r (where r: sampling rate)
3. When r changes,
   - For each entry (e, f),
     - Repeatedly toss a coin with P(head) = 1/r until the outcome of the coin toss is head
     - If the outcome of the toss is tail,
       - Decrement f in (e, f)

4. [Output] Get a list of items where
   $f + \varepsilon N >= sN$

Sticky Sampling算法

抽样窗口参数t=1, ε=0.5
第1个窗口r=1, 窗口大小2, 抽样频率1/r=1

a

0.3<1/r, insert

S

元素	频度

1. S: empty list
   → will contain (e, f)
2. When data e arrives,
   - if e exists in S, increment f in (e, f)
   - if e does not exist in S, add entry (e, 1) with prob. 1/r
3. When r changes,
   - For each entry (e, f),
     - Repeatedly toss a coin with P(head) = 1/r until the outcome of the coin toss is head
     - If the outcome of the toss is tail,
       - Decrement f in (e, f)

4. [Output] Get a list of items where
   $f + \varepsilon N >= sN$

Sticky Sampling算法

抽样窗口参数t=1, ε=0.5
第1个窗口r=1, 窗口大小2, 抽样频率1/r=1

a b

S

元素	频度
a	1

1. S: empty list
   → will contain (e, f)
2. When data e arrives,
   - if e exists in S, increment f in (e, f)
   - if e does not exist in S, add entry (e, 1) with prob. 1/r
3. When r changes,
   - For each entry (e, f),
     - Repeatedly toss a coin with P(head) = 1/r until the outcome of the coin toss is head
     - If the outcome of the toss is tail,
       - Decrement f in (e, f)
4. [Output] Get a list of items where
   $f + \varepsilon N >= sN$

Sticky Sampling算法

抽样窗口参数t=1, ε=0.5
第1个窗口r=1, 窗口大小2, 抽样频率1/r=1

a b

s

元素	频度
a	1
b	1

1. S: empty list
   → will contain (e, f)
2. When data e arrives,
   - if e exists in S, increment f in (e, f)
   - if e does not exist in S, add entry (e, 1) with prob. 1/r
3. When r changes,
   - For each entry (e, f),
     - Repeatedly toss a coin with P(head) = 1/r until the outcome of the coin toss is head
     - If the outcome of the toss is tail,
       - Decrement f in (e, f)
4. [Output] Get a list of items where $f + \varepsilon N >= sN$

Sticky Sampling算法

抽样窗口参数t=1, ε=0.5
第1个窗口r=1, 窗口大小2, 抽样频率1/r=1

a b

s

元素	频度
a	1
b	1

0<1

0<1

1. S: empty list
   → will contain (e, f)
2. When data e arrives,
   - if e exists in S, increment f in (e, f)
   - if e does not exist in S, add entry (e, 1) with prob. 1/r
3. When r changes,
   - For each entry (e, f),
     - Repeatedly toss a coin with P(head) = 1/r until the outcome of the coin toss is head
     - If the outcome of the toss is tail,
       - Decrement f in (e, f)
4. [Output] Get a list of items where
   $f + \varepsilon N >= sN$

Sticky Sampling算法

抽样窗口参数t=1, ε=0.5
第2个窗口r=2, 窗口大小2, 抽样频率1/r=1/2

a b | a

不需要sample

s

元素	频度
a	1
b	1

1. S: empty list
   → will contain (e, f)
2. When data e arrives,
   - if e exists in S, increment f in (e, f)
   - if e does not exist in S, add entry (e, 1) with prob. 1/r
3. When r changes,
   - For each entry (e, f),
     - Repeatedly toss a coin with P(head) = 1/r until the outcome of the coin toss is head
     - If the outcome of the toss is tail,
       - Decrement f in (e, f)
4. [Output] Get a list of items where $f + \varepsilon N >= sN$

Sticky Sampling算法

抽样窗口参数t=1, ε=0.5
第2个窗口r=2, 窗口大小2, 抽样频率1/r=1/2

a b | a c

0.4<1/r, insert

s

元素	频度
a	2
b	1

1. S: empty list
   → will contain (e, f)
2. When data e arrives,
   - if e exists in S, increment f in (e, f)
   - if e does not exist in S, add entry (e, 1) with prob. 1/r
3. When r changes,
   - For each entry (e, f),
     - Repeatedly toss a coin with P(head) = 1/r until the outcome of the coin toss is head
     - If the outcome of the toss is tail,
       - Decrement f in (e, f)
4. [Output] Get a list of items where
   f + εN >= sN

Sticky Sampling算法

抽样窗口参数t=1, ε=0.5
第2个窗口r=2, 窗口大小2, 抽样频率1/r=1/2

a b | a c

s

元素	频度
a	2
b	1
c	1

1<2

1>=1,drop

0<1

1. S: empty list
   → will contain (e, f)
2. When data e arrives,
   - if e exists in S, increment f in (e, f)
   - if e does not exist in S, add entry (e, 1) with prob. 1/r
3. When r changes,
   - For each entry (e, f),
     - Repeatedly toss a coin with P(head) = 1/r until the outcome of the coin toss is head
     - If the outcome of the toss is tail,
       - Decrement f in (e, f)
4. [Output] Get a list of items where
   $f + \varepsilon N >= sN$

Sticky Sampling算法

抽样窗口参数t=1, ε=0.5
第3个窗口r=4, 窗口大小4, 抽样频率1/r=1/4

a b | a c | d

0.1<1/r, insert

s

元素	频度
a	1
c	1

1. S: empty list
   → will contain (e, f)
2. When data e arrives,
   - if e exists in S, increment f in (e, f)
   - if e does not exist in S, add entry (e, 1) with prob. 1/r
3. When r changes,
   - For each entry (e, f),
     - Repeatedly toss a coin with P(head) = 1/r until the outcome of the coin toss is head
     - If the outcome of the toss is tail,
       - Decrement f in (e, f)
4. [Output] Get a list of items where $f + \varepsilon N >= sN$

Sticky Sampling算法

抽样窗口参数t=1, ε=0.5
第3个窗口r=4, 窗口大小4, 抽样频率1/r=1/4

a b | a c | d e

0.4>1/r, drop

S

元素	频度
a	1
c	1
d	1

1. S: empty list
   → will contain (e, f)
2. When data e arrives,
   - if e exists in S, increment f in (e, f)
   - if e does not exist in S, add entry (e, 1) with prob. 1/r
3. When r changes,
   - For each entry (e, f),
     - Repeatedly toss a coin with P(head) = 1/r until the outcome of the coin toss is head
     - If the outcome of the toss is tail,
       - Decrement f in (e, f)
4. [Output] Get a list of items where $f + \varepsilon N >= sN$

Sticky Sampling算法

抽样窗口参数t=1, ε=0.5
第3个窗口r=4, 窗口大小4, 抽样频率1/r=1/4

a b | a c | d e a

不需要sample

s

元素	频度
a	1
c	1
d	1

1. S: empty list
   → will contain (e, f)
2. When data e arrives,
   - if e exists in S, increment f in (e, f)
   - if e does not exist in S, add entry (e, 1) with prob. 1/r
3. When r changes,
   - For each entry (e, f),
     - Repeatedly toss a coin with P(head) = 1/r until the outcome of the coin toss is head
     - If the outcome of the toss is tail,
       - Decrement f in (e, f)
4. [Output] Get a list of items where
   f + εN >= sN

Sticky Sampling算法

抽样窗口参数t=1, ε=0.5
第3个窗口r=4, 窗口大小4, 抽样频率1/r=1/4

a b | a c | d e a d

不需要sample

s

元素	频度
a	2
c	1
d	1

1. S: empty list
   → will contain (e, f)
2. When data e arrives,
   - if e exists in S, increment f in (e, f)
   - if e does not exist in S, add entry (e, 1) with prob. 1/r
3. When r changes,
   - For each entry (e, f),
     - Repeatedly toss a coin with P(head) = 1/r until the outcome of the coin toss is head
     - If the outcome of the toss is tail,
       - Decrement f in (e, f)
4. [Output] Get a list of items where $f + \varepsilon N >= sN$

Sticky Sampling算法

抽样窗口参数t=1, ε=0.5
第3个窗口r=4, 窗口大小4, 抽样频率1/r=1/4

a b | a c | d e a d

S

元素	频度	
a	2	**0**<2
c	1	**1>=**1, drop
d	2	**0**<2

1. S: empty list
   → will contain (e, f)
2. When data e arrives,
   - if e exists in S, increment f in (e, f)
   - if e does not exist in S, add entry (e, 1) with prob. 1/r
3. When r changes,
   - For each entry (e, f),
     - Repeatedly toss a coin with P(head) = 1/r until the outcome of the coin toss is head
     - If the outcome of the toss is tail,
       - Decrement f in (e, f)
4. [Output] Get a list of items where $f + \varepsilon N >= sN$

Sticky Sampling算法

抽样窗口参数t=1, ε=0.5
第4个窗口r=8, 窗口大小8, 抽样频率1/r=1/8

a b | a c | d e a d | f

0.2>1/r, drop

s

元素	频度
a	2
c	1
d	2

1. S: empty list
   → will contain (e, f)
2. When data e arrives,
   - if e exists in S, increment f in (e, f)
   - if e does not exist in S, add entry (e, 1) with prob. 1/r
3. When r changes,
   - For each entry (e, f),
     - Repeatedly toss a coin with P(head) = 1/r until the outcome of the coin toss is head
     - If the outcome of the toss is tail,
       - Decrement f in (e, f)
4. [Output] Get a list of items where
   $f + \varepsilon N >= sN$

Sticky Sampling算法

抽样窗口参数t=1, ε=0.5
第4个窗口r=8, 窗口大小8, 抽样频率1/r=1/8

a b | a c | d e a d | f a

不需要sample

s

元素	频度
a	2
c	1
d	2

1. S: empty list
   → will contain (e, f)
2. When data e arrives,
   - if e exists in S, increment f in (e, f)
   - if e does not exist in S, add entry (e, 1) with prob. 1/r
3. When r changes,
   - For each entry (e, f),
     - Repeatedly toss a coin with P(head) = 1/r until the outcome of the coin toss is head
     - If the outcome of the toss is tail,
       - Decrement f in (e, f)
4. [Output] Get a list of items where $f + \varepsilon N >= sN$

Sticky Sampling算法

抽样窗口参数t=1, ε=0.5
第4个窗口r=8, 窗口大小8, 抽样频率1/r=1/8

a b | a c | d e a d | f a d

不需要sample

S

元素	频度
a	3
c	1
d	2

1. S: empty list
   → will contain (e, f)
2. When data e arrives,
   - if e exists in S, increment f in (e, f)
   - if e does not exist in S, add entry (e, 1) with prob. 1/r
3. When r changes,
   - For each entry (e, f),
     - Repeatedly toss a coin with P(head) = 1/r until the outcome of the coin toss is head
     - If the outcome of the toss is tail,
       - Decrement f in (e, f)
4. [Output] Get a list of items where
   f + εN >= sN

Sticky Sampling算法

抽样窗口参数t=1, ε=0.5
第4个窗口r=8, 窗口大小8, 抽样频率1/r=1/8

a b | a c | d e a d | f a d

1. S: empty list
   → will contain (e, f)
2. When data e arrives,
   - if e exists in S, increment f in (e, f)
   - if e does not exist in S, add entry (e, 1) with prob. 1/r
3. When r changes,
   - For each entry (e, f),
     - Repeatedly toss a coin with P(head) = 1/r until the outcome of the coin toss is head
     - If the outcome of the toss is tail,
       - Decrement f in (e, f)
4. [Output] Get a list of items where
   $f + \varepsilon N >= sN$

S

元素	频度
a	3
c	1
d	3

Sticky Sampling算法

抽样窗口参数t=1, ε=0.5
第4个窗口r=8, 窗口大小8, 抽样频率1/r=1/8

a b | a c | d e a d | f a d

[Output] Get a list of items where
f + εN >= sN

s=0.6, 取使得f>=(s- ε)N的元素，结果为a,d

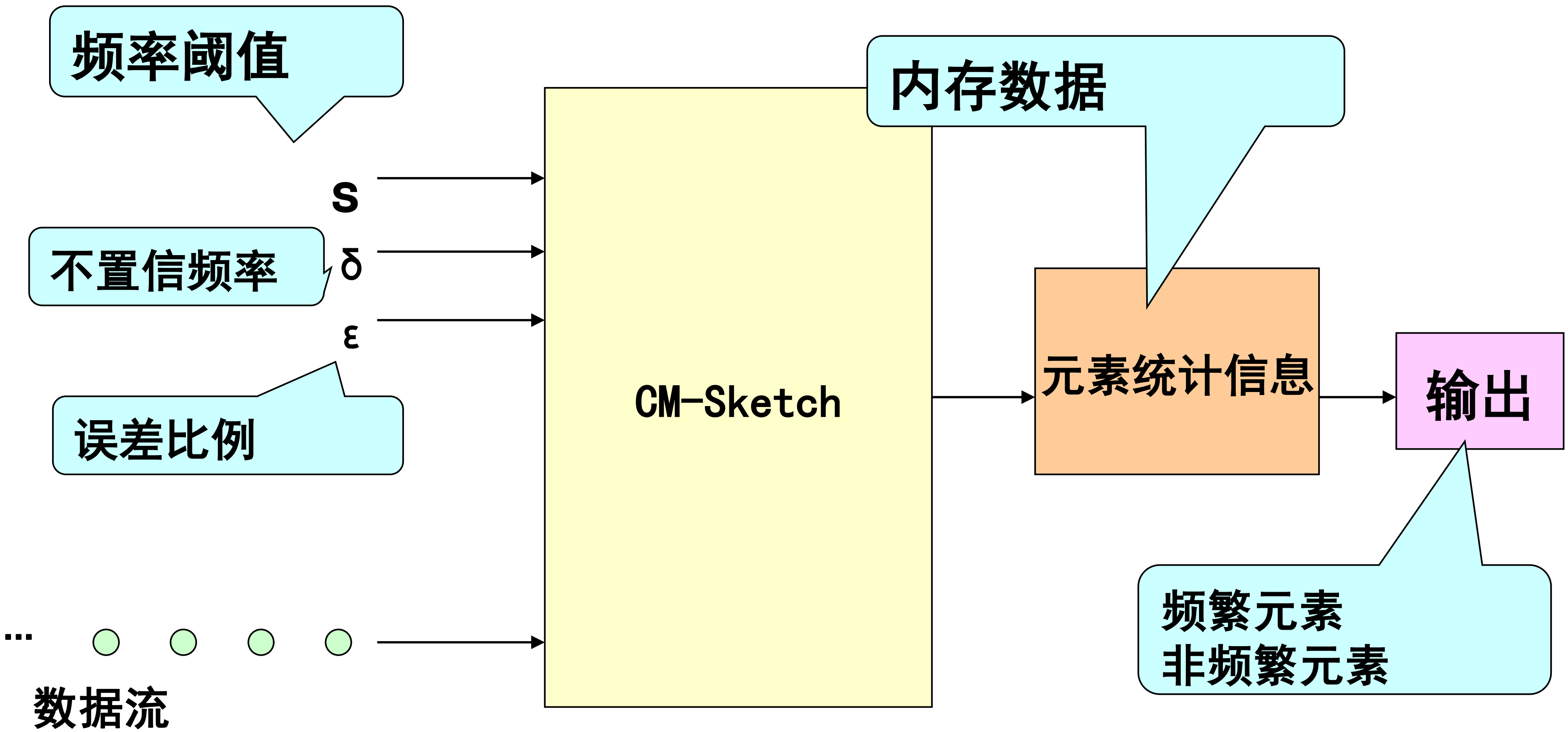
S

元素	频度
a	3
c	1
d	3

课程提纲

- 数据流简介
- 基础性问题
- 确定性算法
  - Majority算法
  - Misra-Gries(MG)算法
  - Space-Saving算法
  - Lossy Counting算法
- 随机性算法
  - Sticky Sampling算法
  - Count-Min(CM)Sketch算法

CM Sketch算法框架



CM Sketch

问题定义：估计流数据中元素出现的总次数。

- 算法设计
  - 使用Hash函数将元素映射到长度更小的数组中。
  - 使用多个独立的Hash函数减小Hash碰撞导致的误差。
  - 多个映射结果中最小的一个作为估计结果(Count Min).
  - 要求哈希函数h:{1,2,⋯,n}→{1,2,⋯,w}碰撞频率不超过1/m.

$$\mathrm{Pr}\{x_i, x_j \epsilon \{1,2,\ldots,n\} \wedge x_i \neq x_j \wedge P(x_i) = \mathrm{P}(x_j)\} < 1/\mathrm{m}$$

  - 数据结构：二维数组
  - 元素的估计值$\mathrm{E}st(e) = \min_{1 \le i \le d}\{CM[i, h_i(e)]\}$

m为元素种类

- 相关参数
  - 数组深度$w = \lceil e/\varepsilon \rceil$为数组列数
  - 哈希函数个数$d = \lceil ln(1/\delta) \rceil$为数组行数

CM Sketch

问题定义：统计流数据中出现频率大于给定值s的元素。

- CM伪代码

```
CM_Sketch():
h1,h2,…,hd //Hash Function
CM←array[d,w]
For every element (e,f):
    for i from 1 to d:
        CM[i,hi(e)]←CM[i,hi(e)]+f
```

CM Sketch

ε=1,δ=0.1

w=$\lceil e/\varepsilon \rceil$=4, d=$\lceil ln(1/\delta) \rceil$=3

CM

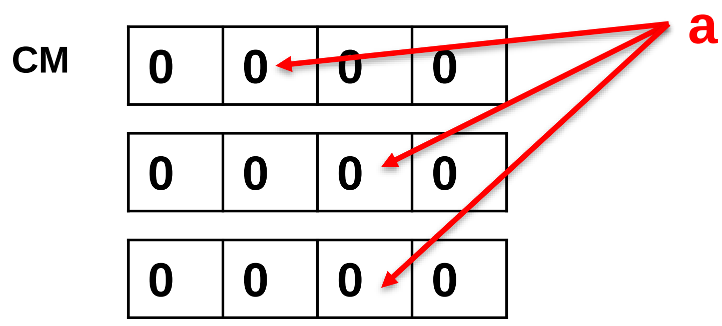
0	0	0	0
0	0	0	0
0	0	0	0

a

$$CM[i, h_i(e)] = CM[i, h_i(e)] + 1$$

CM Sketch

ε=1,δ=0.1
w=⌈e/ε⌉=4, d=⌈$ln(1/\delta)$⌉=3

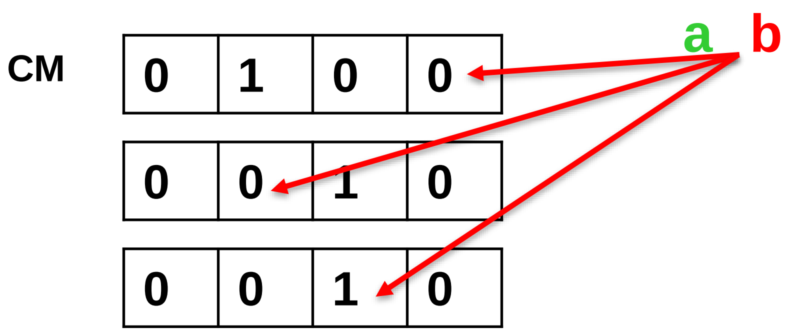


$$CM[i, h_i(e)] = CM[i, h_i(e)] + 1$$

CM Sketch

$\varepsilon=1, \delta=0.1$

$w=\lceil e/\varepsilon \rceil=4$, $d=\lceil ln(1/\delta) \rceil=3$

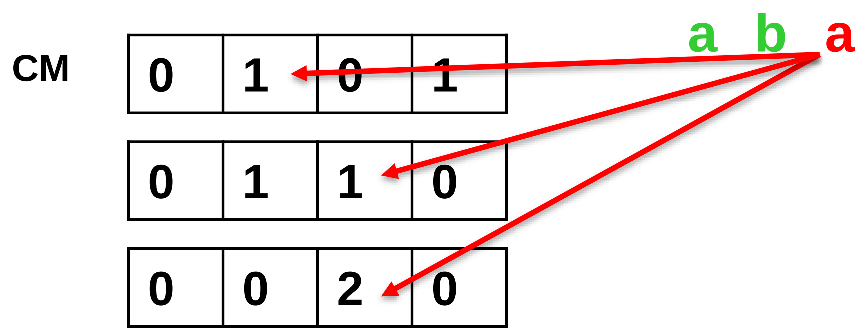


$$CM[i, h_i(e)] = CM[i, h_i(e)]+1$$

CM Sketch

ε=1,δ=0.1

w=$\lceil e/\varepsilon \rceil$=4, d=$\lceil ln(1/\delta) \rceil$=3



$CM[i, h_i(e)] = CM[i, h_i(e)]$+1

CM Sketch

ε=1,δ=0.1

$w = \lceil e/\varepsilon \rceil$=4, $d = \lceil \ln(1/\delta) \rceil$=3

a b a c

CM

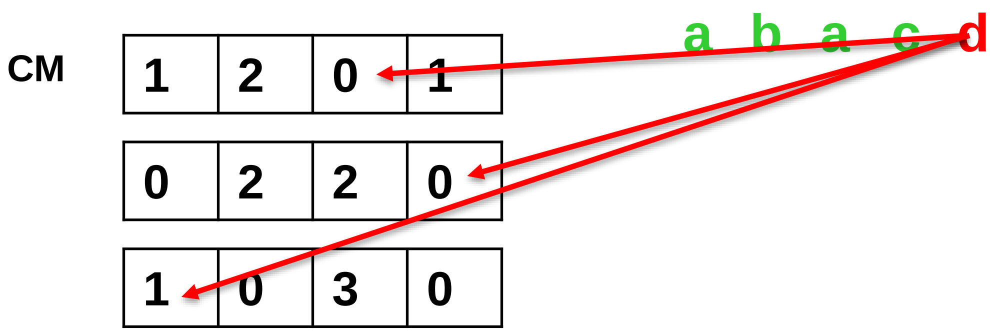
0	2	0	1
0	1	2	0
0	0	3	0

$$CM[i, h_i(e)] = CM[i, h_i(e)]+1$$

CM Sketch

ε=1,δ=0.1

$w=\lceil e/\varepsilon \rceil$=4, $d=\lceil ln(1/\delta) \rceil$=3

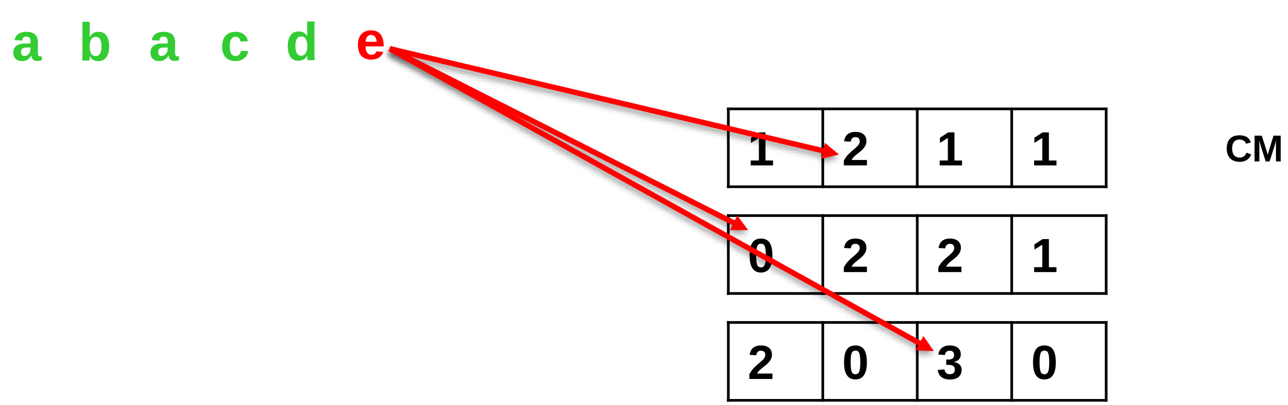


CM[i,hi(e)]←CM[i,hi(e)]+f

CM Sketch

$\varepsilon=1, \delta=0.1$

$w=\lceil e/\varepsilon \rceil=4,\ d=\lceil \ln(1/\delta) \rceil=3$

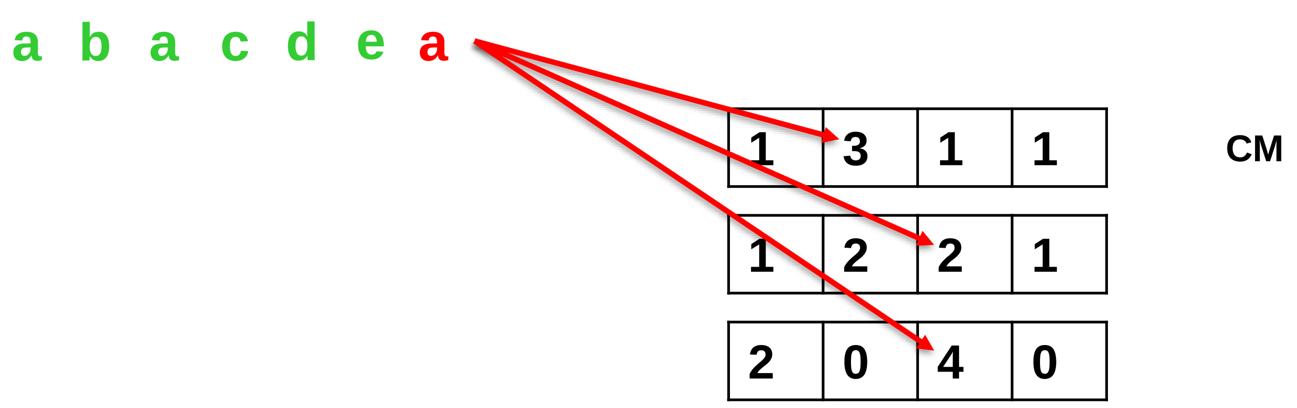


$$CM[i, h_i(e)] = CM[i, h_i(e)]+1$$

CM Sketch

$\varepsilon=1, \delta=0.1$

$w=\lceil e/\varepsilon \rceil=4$, $d=\lceil \ln(1/\delta) \rceil=3$



$$CM[i, h_i(e)] = CM[i, h_i(e)]+1$$

CM Sketch

ε=1,δ=0.1

$w=\lceil e/\varepsilon \rceil$=4, $d=\lceil ln(1/\delta) \rceil$=3

a b a c d e a d

1	4	1	1	CM
1	2	3	1	
2	0	5	0	

$$CM[i, h_i(e)] = CM[i, h_i(e)]+1$$

CM Sketch

ε=1,δ=0.1

$w=\lceil e/\varepsilon \rceil$=4, $d=\lceil ln(1/\delta) \rceil$=3

a b a c d e a d f

CM

1	4	2	1
1	2	3	4
1	0	5	0

$CM[i,h_i(e)] = CM[i,h_i(e)]$+1

CM Sketch

ε=1,δ=0.1
w=$\lceil e/\varepsilon \rceil$=4, d=$\lceil ln(1/\delta) \rceil$=3

a b a c d e a d f a

1	4	2	2
1	3	3	4
2	0	5	0

CM

$$CM[i, h_i(e)] = CM[i, h_i(e)]+1$$

CM Sketch

ε=1,δ=0.1

$w=\lceil e/\varepsilon \rceil$=4, $d=\lceil ln(1/\delta) \rceil$=3

a b a c d e a d f a d

1	5	2	2
1	3	4	4
2	0	6	0

CM

$$CM[i, h_i(e)] = CM[i, h_i(e)]+1$$

CM Sketch

ε=1,δ=0.1
w=$\lceil e/\varepsilon \rceil$=4, d=$\lceil ln(1/\delta) \rceil$=3

a b a c d e a d f a d

1	5	3	2	CM
1	3	4	5	
3	0	6	0	

$$CM[i, h_i(e)] = CM[i, h_i(e)]+1$$
$$Est(e) = \min_{1 \le i \le d}\{CM[i, h_i(e)]\}$$

CM Sketch

ε=1,δ=0.1

w=$\lceil e/\varepsilon \rceil$=4, d=$\lceil ln(1/\delta) \rceil$=3

a b a c d e a d f a d

1	5	3	2
1	3	4	5
3	0	6	0

CM

Est(a) = 4

$$CM[i, h_i(e)] = CM[i, h_i(e)]+1$$

$$Est(e) = \min_{1 \le i \le d}\{CM[i, h_i(e)]\}$$

CM Sketch

ε=1,δ=0.1
$w=\lceil e/\varepsilon \rceil$=4, $d=\lceil ln(1/\delta) \rceil$=3

a b a c d e a d f a d

1	5	3	2	CM
1	3	4	5	
3	0	6	0	

Est(d) = 3

$$CM[i, h_i(e)] = CM[i, h_i(e)]+1$$
$$Est(e) = \min_{1 \le i \le d} \{CM[i, h_i(e)]\}$$

数据流算法算法比较

	算法	ε-Deficient Synopsis	内存消耗
确定性算法	Majority [1]	100%	1
	Misra-Gries [5]	100%(E≤ε)	M
	Space-Saving [4]	100%(E≤ε)	M
	Lossy Counting [3]	100%	$\lceil 1/\varepsilon \log(\varepsilon N) \rceil$
随机性算法	Sticky Sampling [3]	1-δ	$\lceil 2/\varepsilon \ln(s^{-1}\delta^{-1}) \rceil$
	CM-Sketch [2]	1-δ	$\lceil e/\varepsilon \rceil \times \lceil \ln(1/\delta) \rceil$

参考文献

1. G. Cormode and M. Hadjieleftheriou. Finding frequent items in data Streams. In VLDB 2008.
2. G. Cormode and S. Muthukrishnan. An improved data stream summary: The count-min sketch and its applications. Journal of Algorithms, 2005.
3. G. Manku and R. Motwani. Approximate frequency counts over data streams. In VLDB 2002.
4. A. Metwally, D. Agrawal, and A. E. Abbadi. An integrated efficient solution for computing frequent and top-k elements in data streams. ACM Transactions on Database Systems, 2006.
5. J. Misra and D. Gries. Finding repeated elements. Science of Computer Programming, 1982.

谢谢