

大数据算法

Big Data Algorithms

刘显敏
liuxianmin@hit.edu.cn

▣ 数学基础

▣ 外存模型算法

▣ 亚线性空间算法

3 / 200

概率基础

定义1.1[概率空间, probability space]

一个概率空间包含三要素:

- ▷ 样本空间(sample space) Ω : 概率空间描述的随机行为所有可能结果
- ▷ 事件(events) $\mathcal{F}$: 每个事件是 Ω 的一个子集合
- ▷ 概率函数 $\Pr: \mathcal{F} \rightarrow [0, 1]$: 满足概率测度要求

满足概率测度需要如下条件

- ▷ 事件集合 $\mathcal{F}$ 包含 Ω , 对集合补操作和集合并操作封闭
 - ▷ 概率函数满足
 - 非负性($\Pr$ 取值非负)
 - 规范性($\Pr(\emptyset) = 0, \Pr(\Omega) = 1$)
 - 完全可加性(对任意两两不相交的事件 $A_1, \dots, A_n$ 有 $\Pr(\cup A_i) = \sum_i \Pr(A_i)$), 事件序列是无穷可数的也成立
- 在通常的离散概率空间(discrete probability space), 每一个 Ω 中的元素都是一个基本事件, $\mathcal{F} = 2^\Omega$

4 / 200

概率基础

定义1.2[条件概率, conditional probability]

给定事件 E_2 , E_1 发生的条件概率为

$$\Pr(E_1|E_2) = \frac{\Pr(E_1 \cap E_2)}{\Pr(E_2)}$$

条件概率只有当 $\Pr(E_2) > 0$ 的时候才有意义

定义1.3[全概率公式, law of total probability]

令 $E_1, E_2, \dots, E_n$ 为概率空间的两两不相交事件且 $\cup E_i = \Omega$, 我们有

$$\Pr(B) = \sum_{i=1}^n \Pr(B \cap E_i) = \sum_{i=1}^n \Pr(B|E_i) \Pr(E_i)$$

5 / 200

概率基础

定义1.6[随机变量, random variable]

一个样本空间 Ω 上的随机变量 X 是一个实值函数 $X: \Omega \mapsto \mathcal{R}$, 一个离散随机变量是取值为有穷或者可数无穷多个值的变量。

- ▷ 给定离散随机变量 X 和实数 a , 事件 $X = a$ 代表集合 $\{s \in \Omega: X(s) = a\}$

$$\Pr(X = a) = \sum_{s \in \Omega: X(s)=a} \Pr(s)$$

定义1.7[离散随机变量期望]

$$\mathbb{E}[X] = \sum_i i \cdot \Pr(X = i)$$

定理1.2[期望的另一种计算方式]

X 是一个取值为非负整数的离散随机变量, 有 $\mathbb{E}[X] = \sum_{i=1}^{\infty} \Pr(X \geq i)$

12 / 200

概率基础

定义1.8[连续随机变量, continuous random variable]

一个随机变量 X 的**概率分布**由它的**分布函数** $F(x)$ 给出, 对任意 $x \in \mathbb{R}$, 定义 $F(x) = \Pr(X \leq x)$; X 是连续随机变量, 如果 $F(x)$ 是 x 的连续函数。

- ▷ 连续随机变量 X , 必有 $\Pr[X = x] = 0$, 因此有 $\Pr(X \leq x) = \Pr(X < x)$

定义1.9[概率密度函数]

如果存在函数 $f(x)$, 有

$$F(a) = \int_{-\infty}^a f(t) dt$$

, 则称 $f(x)$ 是 $F(x)$ 的密度函数, 且 $f(x) = F'(x)$

定理1.3[连续随机变量的期望]

X 一个连续随机变量, 有 $\mathbb{E}[X] = \int_{-\infty}^{+\infty} xf(x) dx$ 。

13 / 200

概率基础

定理1.5[马尔可夫不等式]

对任意非负随机变量 X 和 $a > 0$, 有 $\Pr(X \geq a) \leq \frac{\mathbb{E}[X]}{a}$

定义1.12[方差]

$$\text{Var}[X] = \mathbb{E}[(X - \mathbb{E}[X])^2] = \mathbb{E}[X^2] - \mathbb{E}[X]^2$$

定理1.6[切比雪夫不等式]

对任意随机变量 X 和 $a > 0$, 有 $\Pr(|X - \mathbb{E}[X]| \geq a) \leq \frac{\text{Var}[X]}{a^2}$

定理1.7[切尔诺夫不等式, Chernoff/Hoeffding Bound]

令 $X_1, X_2, \dots, X_m$ 是独立的、取值 $\in \{0, 1\}$ 的随机变量, 变量和的期望为 $\mu = \mathbb{E}[\sum_i X_i]$, 令 $\epsilon \in [0, 1]$, 有 $\Pr[|\sum_i X_i - \mu| > \epsilon \mu] \leq 2e^{-\epsilon^2 \mu / 3}$ 。

18 / 200

▣ 数学基础

▣ 外存模型算法

▣ 亚线性空间算法

39 / 200

外存计算模型

- ▷ RAM模型：常用算法设计模型
- ▷ External Memory模型，或称I/O模型：用于大数据场景

I/O模型：CPU+内存+外存

- ▷ 内存：大小为 M ，页面大小为 B
- ▷ 外存：大小无限，页面大小为 B
- ▷ CPU可以直接访问内存，速度非常快
- ▷ CPU访问外存数据需先读入内存，速度非常慢

以块（block）或者页面为读写单位的原因

- ▷ 外存的连续访问比随机访问代价小

I/O模型上算法设计的目标

- ▷ 最小化数据传输量，即内存和外存之间交换的数据块数目，或者读/写外存的块数

例如，假设有 N 个数据，扫描数据代价是 $O(\frac{N}{B})$ 或 $O(\lceil \frac{N}{B} \rceil)$ ，不是 $O(N)$

43 / 200

问题1：外存栈

栈（Stack）

- ▷ 内存维护大小为 $2B$ 的数组，实现内存栈结构
- ▷ 内存中存储 k 个栈顶数据， $k \leq 2B$
- ▷ 外存中存储其余数据

如何处理栈操作？

- ▷ 压栈（push）
 - ① 如果内存中数据少于 $2B$ ，直接内存压栈
 - ② 否则，向外存写入 B 个数据，然后，内存压栈
- ▷ 弹栈（pop）
 - ① 如果内存不为空，直接内存弹栈
 - ② 否则，从外存读入最新写入的 B 个数据，然后，内存弹栈

I/O代价分析：

- ▷ 最坏情况代价： $O(1)$ 次I/O
- ▷ 均摊代价（amortized analysis）： $O(1/B)$ ，最优

43 / 200

问题2：外存队列

队列（Queue）

- ▷ 内存维护2个大小为 B 的数组 A 和 B ，一个用于出队，一个用于入队
- ▷ A 和 B 中分别存储 k 个队头数据和 k' 个队尾数据， $k, k' \leq 2B$
- ▷ 外存中存储其余数据

如何处理队列操作？

- ▷ 入队（insert）
 - ① 如果 B 中数据少于 B ，直接内存入队
 - ② 否则，向外存写入 B 个数据，然后，内存入队
- ▷ 出队（remove）
 - ① 如果 A 不为空，直接内存出队
 - ② 否则，从外存读入最早写入的 B 个数据，然后，内存出队

I/O代价分析：

- ▷ 最坏情况代价： $O(1)$ 次I/O
- ▷ 均摊代价（amortized analysis）： $O(1/B)$ ，最优

45 / 200

问题3：外存链表

邻接链表（Linked List）

- ▷ 三种操作：insert(x, p), remove(p), traverse(p, k)

想法一：将List中连续元素存放到一个大小为 B 的块中 如何更新？

想法二：块“半满” $\Rightarrow$ 数据至少 $B/2$ ；内存维护大小为 B 缓冲区

- ▷ remove: 删除后若小于 $B/2$ ，则与邻居块合并，合并后大于 B 则均分
- ▷ insert: 插入后如果大于 B ，平均划分
- ▷ traverse: $O(2k/B)$ ，insert和remove最坏情况代价为 $O(1)$
- ▷ 均摊代价： N 次连续插入 $O(2N/B)$ ，连续删除 $O(\log_2 B \cdot N/B)$

想法三：连续的两个块至少包含 $2B/3$ 个数据；内存维护大小为 B 缓冲区

- ▷ remove: 删除并检查是否有相邻块使得数据量 $\leq 2B/3$ ，有则合并
- ▷ insert: 若当前块满，向邻居插入；邻居均满，均分当前块
- ▷ traverse: $O(3k/B)$
- ▷ 均摊代价： N 次连续插入 $O(2N/B)$ ，连续删除 $O(3N/B)$
- ▷ 均摊代价： N 次连续更新 $O(12N/B)$

顺序扫描（存储空间）代价： $O(N/B) \Rightarrow O(2N/B) \Rightarrow O(3N/B)$

46 / 200

问题4：搜索问题—B+树

搜索问题：设计数据结构支持操作insert(x), remove(x), query(x)

①二分查找树

②(a, b)-tree: $2 \leq a \leq (b+1)/2$

- ▷ 类似二分查找树 $\Rightarrow (p_0, k_1, p_1, k_2, p_2, \dots, k_c, p_c)$
- ▷ 根节点有0或 ≥ 2 个孩子，其它非叶子节点的孩子数目 $\in [a, b]$
- ▷ remove: 找到对应叶子节点，删除后若小于 a ，与邻接块合并，合并后若大于 b ，平均划分，进而在上一层递归删除节点或者调整键值
- ▷ insert: 找到对应叶子节点，插入后若大于 b ，均分，递归上一层插入
- ▷ query: $O(\log_a(N/a))$

③B-tree: $a = B/2, b = B, O(\log_B N)$

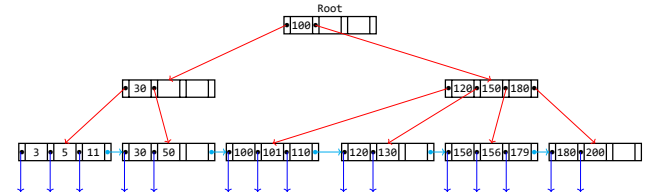
④B+tree: 自动平衡、数据按键值有序存储、 $O(\log n)$ 代价操作

- ▷ 二分查找树的扩展，允许节点有多个孩子节点
- ▷ 为以块（block）为基本单位读写数据而设计

48 / 200

问题4：搜索问题—B+树

B+Tree: $n = 3$



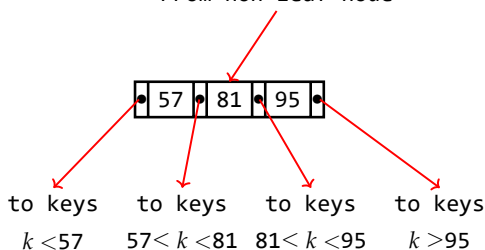
49 / 200

问题4：搜索问题—B+树

B+Tree: $n = 3$

非叶子节点（non-leaf）

From non-leaf node



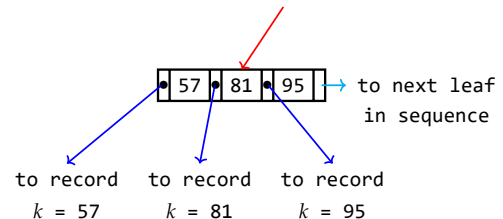
50 / 200

问题4：搜索问题—B+树

B+Tree: $n = 3$

叶子节点（leaf）

From non-leaf node



51 / 200

问题4：搜索问题—B+树

B+Tree中节点的大小

- ▷ $n + 1$ 个指针
- ▷ n 个键值 (keys)

主要思想：避免节点闲置空间过多

节点至少使用

- ▷ $\lceil (n + 1)/2 \rceil$ pointers (Non-leaf Nodes)
- ▷ $\lfloor (n + 1)/2 \rfloor$ pointers (Leaf Nodes)

52 / 200

问题4：搜索问题—B+树

B+Tree Rules of order n

叶子节点都在同一层（最底层），balanced tree
除了指向下一个叶子节点的指针（sequence pointer），叶子节点的指针指向数据（record）

Number of pointers/keys for B+Tree

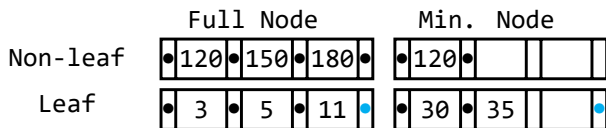
	Max Pointers	Max Keys	Min Ptrs (To Data)	Min Keys
Non-leaf (non-root)	$n+1$	n	$\lceil (n+1)/2 \rceil$	$\lceil (n+1)/2 \rceil - 1$
Leaf (non-root)	$n+1$	n	$\lfloor (n+1)/2 \rfloor$	$\lfloor (n+1)/2 \rfloor$
Root	$n+1$	n	1	1

53 / 200

问题4：搜索问题—B+树

	Max Pointers	Max Keys	Min Ptrs (To Data)	Min Keys
Non-leaf (non-root)	$n+1$	n	$\lceil (n+1)/2 \rceil$	$\lceil (n+1)/2 \rceil - 1$
Leaf (non-root)	$n+1$	n	$\lfloor (n+1)/2 \rfloor$	$\lfloor (n+1)/2 \rfloor$
Root	$n+1$	n	1	1

$n = 3$



54 / 200

问题4：搜索问题—B+树

B+Tree中节点的性质

- ▷ 除了平凡情况，根结点中至少有两个指针被使用，否则，可删除该根结点；
- ▷ 叶结点的键值是数据文件中键值的副本，键值从左向右有序地分布在底层叶结点中；
- ▷ 叶结点中，最后一个指针指向右侧下一个叶结点，其它指针中，至少有 $\lfloor (n+1)/2 \rfloor$ 个指针指向数据记录，指针可以不使用，第 i 个指针如果被使用则指向第 i 个键值对应的记录；
- ▷ 非叶结点中，所有 $n+1$ 个指针如果被使用都指向下一层结点，最少 $\lceil (n+1)/2 \rceil$ 个指针被使用。如果有 j 个指针被使用，那么有 $j - 1$ 个键值。

55 / 200

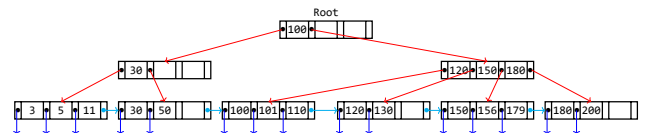
问题4：搜索问题—B+树

B+Tree中节点的性质 (con.)

- ▷ 假设非叶结点中键值依次为 $K_1, K_2, \dots, K_{j-1}$ ，那么第一个指针指向存储小于 K_1 的一些键值的下一层结点；
- ▷ 第二个指针指向存储大于等于 K_1 且小于 K_2 的键值的下一层结点；
- ▷ 依次类推，第 j 个指针指向存储一些大于等于 K_{j-1} 的键值的结点；
- ▷ 小于 K_1 以及大于等于 K_{j-1} 的键值有可能还有一部分存储在兄弟结点中。

56 / 200

问题4：搜索问题—B+树



B+Tree中的查找操作（假设查找的键值为 K ）

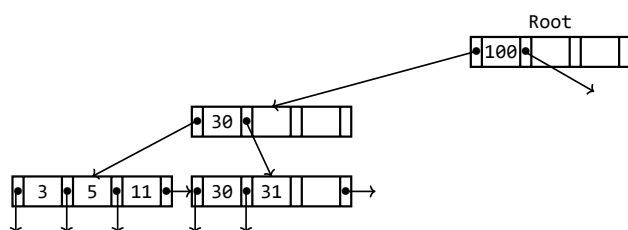
- ▷ 如果处于叶子结点中，那么就在该结点的键值中查找，若有 K ，则可找到对应记录；
- ▷ 如果处于非叶子结点中，依次考虑键值 $K_1, \dots, K_n$ ：
 - $K_i \leq K < K_{i+1}$ ，在第 $i + 1$ 个指针指向结点中递归查找；
 - $K < K_1$ ，在第 1 个指针指向的结点中查找；
 - $K_n \leq K$ ，在第 $n+1$ 个指针指向的结点中查找。

57 / 200

问题4：搜索问题—B+树—插入操作

(a) Simple Case (b) Leaf Overflow
(c) Non-leaf Overflow (d) New Root

Insert key = 32

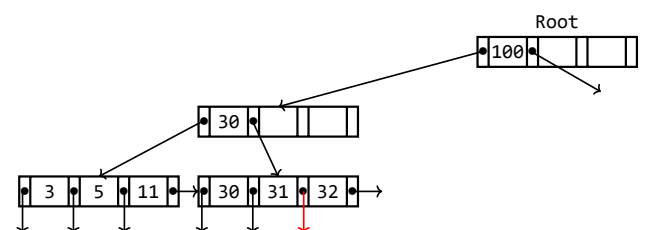


58 / 200

问题4：搜索问题—B+树—插入操作

(a) Simple Case (b) Leaf Overflow
(c) Non-leaf Overflow (d) New Root

Insert key = 32

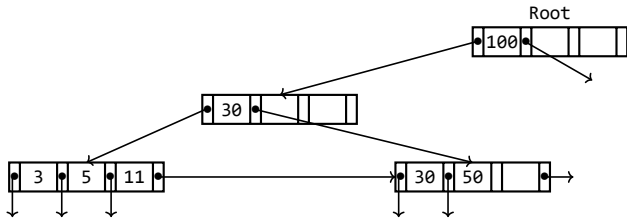


59 / 200

问题4：搜索问题—B+树—插入操作

- (a) Simple Case (b) Leaf Overflow
(c) Non-leaf Overflow (d) New Root

Insert key = 7

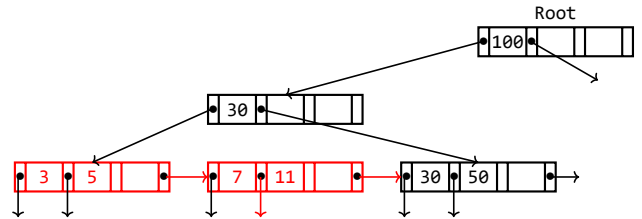


59 / 200

问题4：搜索问题—B+树—插入操作

- (a) Simple Case (b) Leaf Overflow
(c) Non-leaf Overflow (d) New Root

Insert key = 7

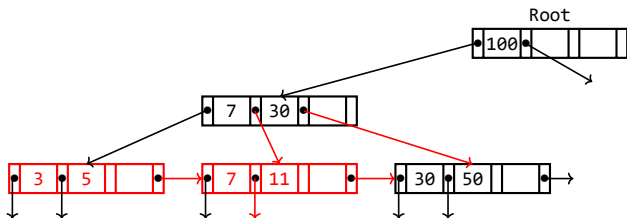


59 / 200

问题4：搜索问题—B+树—插入操作

- (a) Simple Case (b) Leaf Overflow
(c) Non-leaf Overflow (d) New Root

Insert key = 7

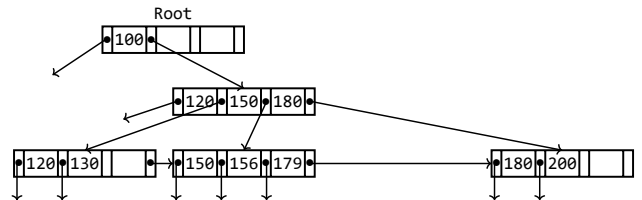


59 / 200

问题4：搜索问题—B+树—插入操作

- (a) Simple Case (b) Leaf Overflow
(c) Non-leaf Overflow (d) New Root

Insert key = 160

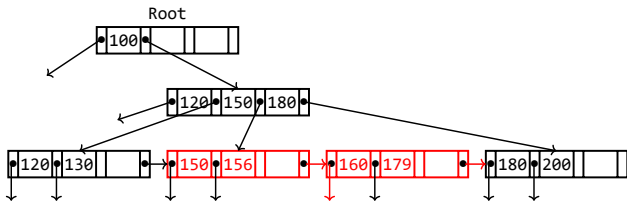


60 / 200

问题4：搜索问题—B+树—插入操作

- (a) Simple Case (b) Leaf Overflow
(c) Non-leaf Overflow (d) New Root

Insert key = 160

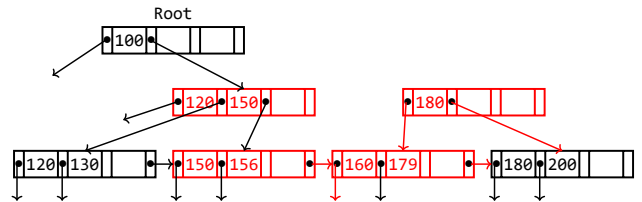


60 / 200

问题4：搜索问题—B+树—插入操作

- (a) Simple Case (b) Leaf Overflow
(c) Non-leaf Overflow (d) New Root

Insert key = 160

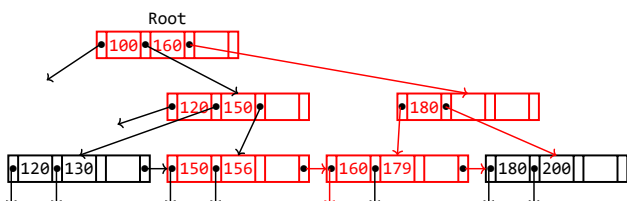


60 / 200

问题4：搜索问题—B+树—插入操作

- (a) Simple Case (b) Leaf Overflow
(c) Non-leaf Overflow (d) New Root

Insert key = 160

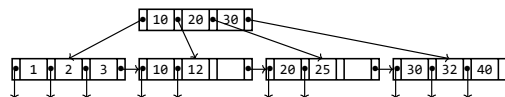


60 / 200

问题4：搜索问题—B+树—插入操作

- (a) Simple Case (b) Leaf Overflow
(c) Non-leaf Overflow (d) New Root

Insert key = 45

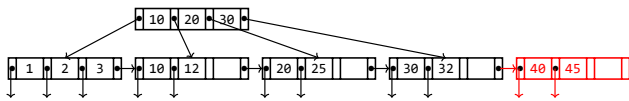


61 / 200

问题4：搜索问题—B+树—插入操作

- (a) Simple Case (b) Leaf Overflow
(c) Non-leaf Overflow (d) New Root

Insert key = 45

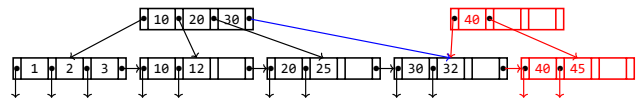


61 / 200

问题4：搜索问题—B+树—插入操作

- (a) Simple Case (b) Leaf Overflow
(c) Non-leaf Overflow (d) New Root

Insert key = 45

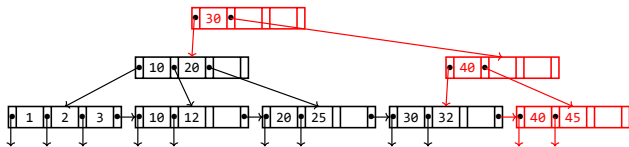


61 / 200

问题4：搜索问题—B+树—插入操作

- (a) Simple Case (b) Leaf Overflow
(c) Non-leaf Overflow (d) New Root

Insert key = 45



61 / 200

问题4：搜索问题—B+树—插入操作

B+Tree的插入操作

- ▷ 假设插入的键值为K，首先找到对应的叶结点L
- ▷ 如果L中有空闲空间，那么直接插入，结束；否则将叶结点分裂为两个结点，并且把其中的键分到这两个新结点中，使得键个数满足最小要求；
- ▷ 某一层的结点分裂在上一层看来相当于在上层插入新的键-指针，因此，在上层可以继续递归插入，如有结点分裂，则向上一层推进；
- ▷ 如果要在根结点中插入键-指针，并且导致根结点分裂，那么就在上一层新建根结点，该根结点指向刚分裂的两个新结点，键由这两个结点确定；

62 / 200

问题4：搜索问题—B+树—插入操作

B+Tree的插入操作，当分裂叶结点N时

- ▷ 假设N是一个容量为n个键的叶结点，那么我们正要插入第n+1个键-指针。
- ▷ 创建一个新结点M，让M为N的右侧兄弟，将键排序，前 $\lceil (n+1)/2 \rceil$ 个留在N中，其他键-指针放入M中。
- ▷ 注意N和M中指针-键的数目都足够，即至少有 $\lfloor (n+1)/2 \rfloor$ 个键-指针对。
- ▷ 上一层新加入的键是M中最小的键值。

63 / 200

问题4：搜索问题—B+树—插入操作

B+Tree的插入操作，当分裂非叶结点N时

- ▷ 假设N是一个容量为n个键、n+1个指针的非叶结点，那么我们正要插入第n+2个指针、第n+1个键。
- ▷ 创建一个新结点M，让M为N的右侧兄弟。
- ▷ 将键-指针排序，前 $\lceil (n+2)/2 \rceil$ 个指针留在N中，剩下的 $\lfloor (n+2)/2 \rfloor$ 个指针放入M中。
- ▷ 前 $\lceil n/2 \rceil$ 个键留在N中，后 $\lfloor n/2 \rfloor$ 个键放入M中，中间那个键留出来，插入到上一层结点中，该键指针指向M。
- ▷ 注意N和M中指针-键的数目都足够。

64 / 200

问题4：搜索问题—B+树—删除操作

- (a) Simple Case (b) Coalesce with neighbor
(c) Re-distribute (d) Cases b or c non-leaf

65 / 200

问题4：搜索问题—B+树—删除操作

- (a) Simple Case (b) Coalesce with neighbor
(c) Re-distribute (d) Cases b or c non-leaf

66 / 200

问题4：搜索问题—B+树-删除操作

- (a) Simple Case (b) Coalesce with neighbor
(c) Re-distribute (d) Cases b or c non-leaf

67 / 200

问题4：搜索问题—B+树-删除操作

- (a) Simple Case (b) Coalesce with neighbor
(c) Re-distribute (d) Cases b or c non-leaf

68 / 200

问题4：搜索问题—B+树-删除操作

B+Tree的删除操作

- ▷ 假设删除的键值为K，首先找到对应的叶结点L。
- ▷ 如果L中删除K后仍然具有满足最小要求的键个数，停止，否则需要做如下处理：
- ▷ 尝试与L的相邻兄弟节点之一合并（合并后仍能放入同一节点），合并后，相当于在上层结点删除了一个键值，那么递归处理；
- ▷ 否则，考虑L的相邻兄弟，假设其中一个能够提供L一个键-指针，并且去除该键-指针后该兄弟结点仍然满足键数的最小要求，那么L从该兄弟处借得一对键-指针，并更新父节点的对应键值；
 - 如果两个兄弟都无法提供一个键-指针，那么必然是如下情况：L的键数少于最小数，L兄弟M的键数恰好为最小数，那么两个结点可以合并。

69 / 200

问题4：搜索问题—B+树-性能

结点中指针数目的最大值：n+1，记录条数：N

B+Tree的插入操作： $O(\log_{\lceil (n+1)/2 \rceil}(N))$

B+Tree的删除操作： $O(\log_{\lceil (n+1)/2 \rceil}(N))$

B+Tree的实际性能

- ▷ Typical Fill-Factor: 67%
 - 假设最大扇出 (fanout) 为320
 - Average Fanout = $320 * 0.67 = 214$
- ▷ Typical Capacities:
 - Height 4: $214^4 = 2,097,273,616$ entries
 - Height 3: $214^3 = 9,800,344$ entries
- ▷ Pages per level:
 - Level 1 = 1 page = 32 KB
 - Level 2 = 214 pages = 6.7 MB
 - Level 3 = 45,796 pages = 1431 MB
 - Level 4 ≈ 300 GB

70 / 200

问题5：矩阵乘法

计算矩阵乘法？

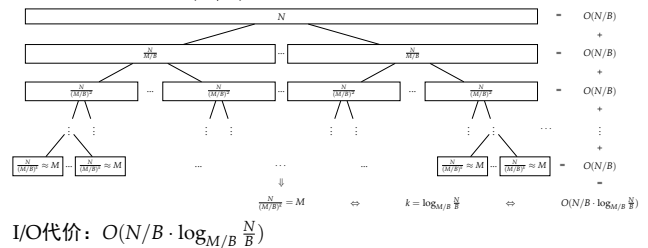
- ▷ 输入两个大小为 $N \times N$ 的矩阵X和Y
- ▷ 将矩阵分为大小为 $\sqrt{M}/2 \times \sqrt{M}/2$ 的块
- ▷ 考虑 $X \times Y$ 矩阵中的每个块，显然共有 $(\frac{N}{\sqrt{M}})^2$ 个块需要输出
- ▷ 每个块需要扫描 $(\frac{N}{\sqrt{M}})$ 对输入块
- ▷ 每次内存计算需要 $O(M/B)$ 次I/O
- ▷ 共计 $O((\frac{N}{\sqrt{M}})^3 \cdot M/B) = O(\frac{N^3}{B\sqrt{M}})$ 次I/O

71 / 200

问题6：外存排序问题

外存排序问题 (External Sort)

- ▷ 给定N个数据
- ▷ 分成大小为 $O(M)$ 的组，每组可在内存排序，需要 $O(M/B)$ 次I/O
- ▷ 排好序的分组，进行归并 (Merge)
- ▷ 每次可以归并 $O(M/B)$ 个分组



72 / 200

- 数学基础
- 外存模型算法
- 亚线性空间算法

74 / 200

流模型/One-Pass计算模型

- ▷ 输入是数据流的方式，可由 $\langle a_1, a_2, \dots, a_n \rangle$ 表示
 - 例1：每个数据是 $[1, m]$ 之间的整数
 - 例2：每个数据是一条边 (u, v)
- ▷ 允许空间代价：sublinear $o(n)$ 、polylog $O(\log^c n)$
- ▷ 流算法的分析维度
 - 空间代价
 - 时间代价 worst-case time for each data
 - 全体时间代价 相当于 amortized-case time
 - 输出结果质量
 - 算法成功的概率 如果是随机算法
- ▷ 典型应用：路由器监测、数据库查询等
- ▷ 扩展模型：multi-pass、semi-streaming

75 / 200

问题1：近似计数

流模型 (Streaming Model)

- ① 给定数据流 $\sigma = \langle a_1, a_2, \dots, a_n \rangle$, 其中 $a_i \in [m]$
- ② 对应的频次向量为

$$\mathbf{f} = (f_1, \dots, f_m), \text{ s.t. } \sum_{1 \leq i \leq m} f_i = n.$$

定义1.14[The Counting Problem]

给定数据流 σ 以及 a_i , 计算 f_i 。

76 / 200

问题1：近似计数

为了计算 f_i , 假设 a_i 共出现 n 次

- ▷ 需要 $O(\log n)$ 位的空间代价存储 n
- ▷ 如果 n 非常大, $O(\log n)$ 也许不可接受
- ▷ 是否有可能占用更少的空间代价?
 - 当然不可能, 除非我们放宽要求
- ▷ 我们将设计一个占用 $O(\log \log n)$ 位空间的近似算法

定义1.15[The Approximate Counting Problem]

给定数据流 σ 以及 a_i , 计算 $\hat{f}_i$, 满足

$$\Pr[|\hat{f}_i - f_i| > \epsilon f_i] < \delta$$

实际应用中, 可令 $\epsilon = 1/3$, $\delta = 1\%$

77 / 200

问题1：近似计数

Morris算法

```

/** 维护一个计数器X */
1. X ← 0;
2. for 每一个元素  $a_i$  do
    以概率  $\frac{1}{2^X}$  更新  $X \leftarrow X + 1$ ;
3. return  $\hat{f}_i = 2^X - 1$ 
    
```

Morris算法的估计结果分析

- ▷ 计算期望 $\mathbb{E}[2^X - 1] = n$
- ▷ 计算方差 $\text{Var}[2^X - 1] \leq n(n-1)/2 = O(n^2)$
- ▷ 利用切比雪夫不等式

79 / 200

问题1：近似计数

- ▷ 根据切比雪夫不等式, $Z = \mathbb{E}[Z] \pm \sqrt{\text{Var}[Z]/c}$ 成立的概率至少为 $1 - c$

- ▷ 至少以 $1 - c$ 概率, Morris算法满足 $2^{X_n} - 1 = \mathbb{E}[2^{X_n} - 1] \pm \sqrt{\text{Var}[2^{X_n} - 1]/c} = n \pm O(n)$

Morris算法的空间代价分析

- ▷ f 是凸函数, 如果 $f(\frac{a+b}{2}) \leq \frac{f(a)+f(b)}{2}$; 等价的, $\forall \lambda \in [0, 1]$, 有 $f(\lambda a + (1-\lambda)b) \leq \lambda f(a) + (1-\lambda)f(b)$

定理1.16[Jensen's inequality]

令 Z 为随机变量, $\mathbb{E}[Z] < \infty$, 如果 f 是凸函数, 那么有 $f(\mathbb{E}[Z]) \leq \mathbb{E}[f(Z)]$ 。

- ▷ 令 $Y_n = 2^{X_n}$, 有 $2^{\mathbb{E}[X_n]} \leq \mathbb{E}[Y_n] = n + 1$
- ▷ $\mathbb{E}[X_n] \leq \log(n+1) \Rightarrow \mathbb{E}[\text{space}(X_n)] = O(\log \log n)$

82 / 200

问题1：近似计数

如何获得一个 $1 + \epsilon$ 近似算法

Morris+算法

```

/** 重复 Morris 算法 k 次 */
1. 同时独立执行 k 个 Morris 算法;
2. 令各算法结果为  $(X_1, \dots, X_k)$ ;
3. return  $\frac{1}{k} \sum_{i=1}^k (2^{X_i} - 1)$ ;
    
```

Morris+算法的估计结果分析

- ▷ 计算期望 $\mathbb{E}[Y] = \mathbb{E}[\frac{1}{k} \sum_{i=1}^k (2^{X_i} - 1)] = n$
- ▷ 计算方差 $\text{Var}[Y] = O(\frac{n^2}{k})$
- ▷ 利用切比雪夫不等式

84 / 200

问题1：近似计数

定理1.18[Analysis of Morris+ Algorithm]

至少以 $1 - c$ 的概率, Morris+算法满足

$$Y = n \pm \frac{1}{\sqrt{k}} O(n)$$

- ▷ 令 $k = O(\frac{1}{\epsilon^2})$, 上述算法是一个 $(1 \pm \epsilon)$ 近似算法, 算法以常数概率如 $1 - c = 0.9$ 满足近似标准
- ▷ 进一步, 令 $k = O(\frac{1}{\epsilon^2 \delta})$, 可以将概率改进为 $1 - \delta$
 - $\Pr[|Y - n| > \epsilon n] < \frac{\text{Var}[Y]}{\epsilon^2 n^2} = O(\frac{1}{k \epsilon^2})$
 - 只需令 $O(\frac{1}{k \epsilon^2}) < \delta = O(\delta)$, 即 $k = O(\frac{1}{\epsilon^2 \delta})$
 - 空间代价为 $O(\frac{\log \log n}{\epsilon^2 \delta})$

86 / 200

问题1：近似计数

- ▷ 为了以 $1 - \delta$ 概率得到 $(1 \pm \epsilon)$ -近似, Morris+需要 $k = O(\frac{1}{\epsilon^2 \delta})$ 次运行
- ▷ 假设, 有 Morris+算法 $\mathcal{M}$ 以常数概率 $1 - c \geq 0.9$ 得到 $1 \pm \epsilon$ 近似
- ▷ 设计一个 $1 - \delta$ 概率 $(1 \pm \epsilon)$ -近似算法, 代价为 $O(\frac{1}{\epsilon^2} \log \frac{1}{\delta} \log \log n)$

Morris++ 算法

```

/** Median Trick */
1. 同时独立执行  $t = O(\log(1/\delta))$  个  $\mathcal{M}$  算法;
2. 取  $t$  次估计结果的中间值 (median) 返回;
    
```

Morris++算法的近似结果分析

- ▷ 计算期望
- ▷ 利用切尔诺夫不等式

87 / 200

问题1：近似计数

定理1.19[切尔诺夫不等式, Chernoff/Hoeffding Bound]

令 $X_1, X_2, \dots, X_m$ 是独立的、取值在 $\{0, 1\}$ 的随机变量, 变量的和的期望为 $\mu = \mathbb{E}[\sum_i X_i]$, 令 $\epsilon \in [0, 1]$, 有 $\Pr[|\sum_i X_i - \mu| > \epsilon \mu] \leq 2e^{-\epsilon^2 \mu/3}$.

- ▷ 令 $X_i = 1 \Leftrightarrow$ 第 i 个 Morris+算法 $\mathcal{M}$ 输出的结果满足条件 $\mathbb{E}[X_i] \geq 0.9$ 且 $\mu = \mathbb{E}[\sum_{i=1}^t X_i] \geq 0.9t$
- ▷ 易知, 如果 $\sum_i X_i > 0.5t$, 那么 Morris++ 算法输出结果将满足条件
- ▷ 那么错误概率如下

$$\begin{aligned} \Pr[\sum_i X_i < 0.5t] &= \Pr[\sum_i X_i - 0.9t < -0.4t] \\ &\leq \Pr[|\sum_i X_i - \mu| > 0.4t] \\ &= \Pr[|\sum_i X_i - \mu| > \mu 0.4/0.9] \end{aligned}$$

$$\begin{aligned} \text{切尔诺夫不等式} \Rightarrow &\leq 2e^{-\frac{16}{81 \times 3} \cdot 0.9t} \\ &\leq e^{-c't} \end{aligned}$$

- ▷ 若想要 $\Pr[\sum_i X_i < 0.5t] \leq e^{-c't} < \delta$, 只需要 $t = O(\log(1/\delta))$

88 / 200

问题2：不重复元素数

流模型 (Streaming Model)

- ① 给定数据流 $\sigma = \langle a_1, a_2, \dots, a_n \rangle$, 其中 $a_i \in [m]$
- ② 对应的频次向量为

$$\mathbf{f} = (f_1, \dots, f_m), \text{ s.t. } \sum_{1 \leq i \leq m} f_i = n.$$

定义1.16[The Distinct Elements Problem]

给定数据流 σ , 计算 $\sum_{1 \leq i \leq m} \mathbb{I}[f_i > 0]$, $\mathbb{I}[f_i > 0] = 1$ 当且仅当 $f_i > 0$.

89 / 200

问题2：不重复元素数

精确计算需要多少空间代价?

- ▷ 方法1: $O(m)$ 位.
为每一个 $[m]$ 中的元素维护一个位, 长度为 m 的向量.
- ▷ 方法2: $O(n \log m)$ 位.
维护 n 个数, 每一个使用 $\log m$ 位.

90 / 200

问题2：不重复元素数

- ▷ 一个理想化的解决方案: 假设可以存储实数
- ▷ 利用哈希函数 (hash function)
 - $h: [m] \mapsto [0, 1]$
 - $h(i)$ 的函数值是 $[0, 1]$ 实数, 均匀分布

FM 算法 [Flajolet-Martin 1985]

- ```
/** 维护变量z */
1. 令 $z = 1$, 并随机选取哈希函数 $h: [m] \mapsto [0, 1]$;
2. for 每一个元素 i do
3. $z = \min\{z, h(i)\}$;
4. return $1/z - 1$.
```

100 / 200

## 问题2：不重复元素数

利用多次运行

### FM+ Algorithm

- ```
/** Maintain a variable z */
1. for  $j$  from 1 to  $k$ 
2.   随机选取一个哈希函数  $h_j: [m] \mapsto [0, 1]$ 
3.    $z_j = 1$ .
4.   每次遇到  $i$ , 更新:  $z_j = \min(z_j, h_j(i))$ 
5.  $Z = \frac{1}{k} \sum_{j=1}^k z_j$ 
6. return  $1/Z - 1$ .
```

107 / 200

问题2：不重复元素数

FM+算法分析: $X = \frac{1}{Z} - 1$, $Z = \frac{1}{k} \sum_{j=1}^k z_j$, $\mathbf{E}[Z] = \mathbf{E}[z]$, $\mathbf{Var}[Z] = \frac{\mathbf{Var}[z]}{k}$

$$\Pr[|Z - \frac{1}{D+1}| > c \frac{1}{D+1}] < \frac{(D+1)^2}{c^2} \mathbf{Var}[Z]$$

$$c = \frac{\epsilon D}{D+1 + \epsilon D} \Rightarrow \Pr[|X - D| > \epsilon D] < \frac{2(D+1 + \epsilon D)^2}{k \epsilon^2 D^2} < \frac{2}{k} (\frac{2}{\epsilon} + 1)^2$$

这里, ϵ 为精度要求, 假设概率要求为 $1 - \delta$, 只需

$$\begin{aligned} \frac{2}{k} (\frac{2}{\epsilon} + 1)^2 &< \delta \\ \Rightarrow k &> \frac{2}{\delta} (\frac{2}{\epsilon} + 1)^2 = O(\frac{1}{\epsilon^2 \delta}) \end{aligned}$$

108 / 200

问题2：不重复元素数

利用Median技术

FM++ 算法

- 运行具有常数概率的FM+算法 $t = \Theta(\log \frac{1}{\delta})$ 次;
- $\hat{D}$ 为所有 t 个结果的中间值;
- return** $\hat{D}$;

- ▷ FM++ 是一个以 $1 - \delta$ 概率保证 $(1 \pm \epsilon)$ 近似的算法
- ▷ 整体代价为 $O(\frac{1}{\epsilon^2} \log \frac{1}{\delta})$ 个实数的存储空间

110 / 200

问题2：不重复元素数

然而, 我们无法存储实数

- ▷ 设计一个 $O(\log m)$ 位空间代价的 $O(1)$ 近似算法
- ▷ $\frac{1}{C} \times D \leq \hat{D} \leq C \times D$

Practical FM 算法

- 随机选取2-相对独立哈希函数 $h: [m] \mapsto [m]$;
- $z = 0$;
- if** $\text{zeros}(h(j)) > z$ **then** $z = \text{zeros}(h(j))$;
 $\text{zeros}(p) = \max\{i | p \% 2^i = 0\}$ $\begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 \end{bmatrix} \Rightarrow \text{zeros}(\cdot) = 2$
- return** $\hat{D} = 2^{z+1/2}$;

120 / 200

问题2：不重复元素数

定义1.17[2-相对独立哈希函数]

一个从 $[X]$ 到 $[Y]$ 的哈希函数族 $\mathcal{H}$ 被称作是2-相对独立的, 如果 $\forall a, b \in [Y]$ 和 $\forall i, j \in [X]$, 其中 $i \neq j$, 有

$$\Pr_{h \in \mathcal{H}} (h(i) = a \wedge h(j) = b) = \frac{1}{Y^2}$$

例1.2[构建2-相对独立哈希函数]

令 Y 是一个素数 (如果不是, 可以取比 Y 大的第一个素数), 随机选取整数 $p, q \in \{0, 1, 2, \dots, Y-1\}$, 令 $h(x) = (px + q) \% Y$

- ▷ 显然, 存储一个这样的函数只需要 $O(\log Y)$ 位的空间代价
- ▷ $h(i) = a \wedge h(j) = b \Rightarrow pi + q \equiv a, pj + q \equiv b \Rightarrow p(i-j) \equiv a-b, q \equiv a-pi$
 $\Rightarrow$ 因为 Y 是素数, 只有一对 p, q 满足该等式 $\Rightarrow p, q$ 被选中的概率是 $\frac{1}{Y^2}$

121 / 200

问题2：不重复元素数

定义1.18[k-相对独立哈希函数]

从 $[a]$ 到 $[b]$ 的哈希函数族 $\mathcal{H}$ 是 k -相对独立的, 如果 $\forall j_1, \dots, j_k \in [b]$ 和 $\forall i_1, \dots, i_k \in [a]$, 其中 $i_x \neq i_y$, 有

$$\Pr_{h \in \mathcal{H}} (h(i_1) = j_1 \wedge \dots \wedge h(i_k) = j_k) = 1/b^k$$

▷ 理论上, 可以利用 $\log_2 |\mathcal{H}|$ 位存储某个 $h \in \mathcal{H}$

例1.3[k-相对独立哈希函数的例子]

- ▷ 所有形如 $[a] \rightarrow [b]$ 的函数构成的 $\mathcal{H}$, $|\mathcal{H}| = b^a$;
- ▷ $\mathcal{H}_{poly}$: 至多为 $k-1$ 阶多项式, 系数在 $[n]$ 中, n 为素数, $|\mathcal{H}_{poly}| = n^k$, 存储需 $O(k \log n)$ 位
- ▷ 存储2-相对独立的 $\mathcal{H}_{poly}$ 某个函数, 需要 $O(2 \log n)$ 位

122 / 200

问题2：不重复元素数

存储哈希函数需 $O(\log m)$ 位; 存储 z 需 $\log \log m$ 位

定理1.27

Practical FM算法以 $1 - \frac{2\sqrt{2}}{c}$ 的概率满足 $D/C \leq \hat{D} \leq C \times D$ 。

- ▷ 令 $X_{r,j} = 1 \Leftrightarrow \text{zeros}(h(j)) \geq r$, 其中 $j \in [m]$ 且 $r \geq 0$
 - ▷ 令 $Y_r = \sum_{j \in [m]} X_{r,j} = \sum_{j: f_j > 0} X_{r,j}$
 - Y_r : 哈希值二进制尾部0数目 $\geq r$ 的元素数
 - ▷ 假设算法结束时, $z = t$
 - $t \geq r \Leftrightarrow \exists j \text{ 有 } \text{zeros}(h(j)) \geq r \Leftrightarrow X_{r,j} = 1 \Leftrightarrow Y_r > 0$
 - $t < r \Leftrightarrow \forall j \text{ 有 } \text{zeros}(h(j)) < r \Leftrightarrow X_{r,j} = 0 \Leftrightarrow Y_r = 0$
 - ▷ $E[X_{r,j}] = \Pr(\text{zeros}(h(j)) \geq r) = \frac{1}{2^r}$; $E[Y_r] = \frac{D}{2^r}$
- [Median] $1 - \delta$ 概率保证 $O(1)$ 近似, 空间 $O(\log \frac{1}{\delta} \log m)$

124 / 200

问题2：不重复元素数

▷ 两层hash, $O(\log m + \frac{1}{\epsilon^2}(\log \frac{1}{\epsilon} + \log \log m))$ 空间算法

BJKST 算法

1. 随机选2-相对独立哈希函数 $h: [m] \rightarrow [m]$;
2. 随机选2-相对独立哈希函数 $g: [m] \rightarrow [b\epsilon^{-4} \log^2 m]$;
3. $z = 0, B = \emptyset$;
4. **if** $\text{zeros}(h(j)) \geq z$ **then**
5. $B = B \cup \{(g(j), \text{zeros}(h(j)))\}$;
6. **if** $|B| > \frac{c}{\epsilon^2}$ **then**
7. $z = z + 1$, 从 B 中删除 (α, β) , 其中 $\beta < z$;
8. **return** $\hat{D} = |B|^{2z}$;

127 / 200

问题2：不重复元素数

定理1.28[BJKST算法性能]

BJKST算法使用 $O(\log m + \frac{1}{\epsilon^2}(\log \frac{1}{\epsilon} + \log \log m))$ 空间, 可以实现至少 $2/3$ 概率保证 $(1 \pm \epsilon)$ 近似。

- ▷ 存储 h 需 $O(\log m)$
- ▷ 存储 g 需 $O(\log(b\epsilon^{-4} \log^2 m))$
- ▷ 存储 $|B|$ 个 g 取值需 $O(\epsilon^{-2} \log(b\epsilon^{-4} \log^2 m))$
- ▷ 存储 $|B|$ 个 zeros 取值需 $O(\epsilon^{-2} \log \log m)$
- ▷ 因此, 空间代价 $O(\log m + \frac{1}{\epsilon^2}(\log \frac{1}{\epsilon} + \log \log m))$

📖 Ziv Bar-Yossef et al. "Counting Distinct Elements in a Data Stream". In: *RANDOM 2002*

132 / 200

问题3：点查询

流模型 (The Streaming Model)

- ① 给定数据流 $\sigma = \langle a_1, a_2, \dots, a_n \rangle$, 其中 $a_i \in [m]$
- ② 对应的频次向量为

$$\mathbf{f} = (f_1, \dots, f_m), \text{ s.t. } \sum_{1 \leq i \leq m} f_i = n.$$

定义1.19[点查询 (point query) 问题]

给定数据流 σ 和事先未知查询 a_i , 输出 f_i 。

134 / 200

问题3：点查询

ℓ_p 范数: $\|\mathbf{x}\|_p = (\sum_i |x_i|^p)^{1/p}$

定义1.20[ℓ_p -近似点查询 (频度估计)]

给定数据流 σ 以及事先未知查询 a_i 输出 $\hat{f}_i$ 满足 $\hat{f}_i \in f_i \pm \epsilon \|\mathbf{f}\|_p$ 。

- ▷ $\|\mathbf{x}\|_1 \geq \|\mathbf{x}\|_2 \geq \dots \geq \|\mathbf{x}\|_\infty$, p 越大, 估计越准确
- ▷ $\|\mathbf{x}\|_0 = m$, 是不同元素的数目
- ▷ $\|\mathbf{x}\|_1 = n$, 是数据流的长度
- ▷ $\|\mathbf{x}\|_\infty = \max\{f_i\}$, 是最大频度

135 / 200

问题3：点查询- ℓ_1 近似

Misra-Gries 算法[1982]

- ```
/** Maintain a set A consists of pairs (i, f_i) **/
```
1.  $A \leftarrow \emptyset$ ;
  2. 对每一个数据流中的元素  $e$ ,
    - (a) 如果  $e \in A$ , 令  $(e, \hat{f}_e) \leftarrow (e, \hat{f}_e + 1)$
    - (b) 否则, 如果  $|A| < 1/\epsilon$ , 将  $(e, 1)$  插入  $A$
    - (c) 否则, 将所有  $A$  中计数减1  
 $(j, \hat{f}_j) \leftarrow (j, \hat{f}_j - 1)$ ,  
如果  $\hat{f}_j = 0$ , 从  $A$  中删除  $(j, 0)$
  3. 对于查询  $i$ , 如果  $i \in A$ , **return**  $\hat{f}_i$ , 否则 **return** 0;

137 / 200

## 问题3：点查询- $\ell_1$ 近似

定理1.29

Misra-Gries算法空间代价为  $O(\epsilon^{-1} \log mn)$ , 对任意查询  $i$ , 返回  $\hat{f}_i$  满足

$$f_i - \epsilon n \leq \hat{f}_i \leq f_i$$

- ▷ 如果不发生减1的情况, 那么  $\hat{f}_i = f_i$
- ▷ 如果发生了减1的情况, 有  $\hat{f}_i < f_i$
- ▷ 假设共发生了  $c$  次减1的情况, 总数减少  $\epsilon \leq n$
- ▷ 每个计数至多减少  $c$ ,  $\hat{f}_i \geq f_i - c \geq f_i - \epsilon n$

138 / 200

### 问题3：点查询- $\ell_1$ 近似

#### Metwally 算法[2005]

- /\*\* Maintain a set A consists of pairs (i,  $\hat{f}_i$ ) \*/*
1.  $A \leftarrow \emptyset$ ;
  2. 对每一个数据流中的元素  $e$ ,
    - (a) 如果  $e \in A$ , 令  $(e, \hat{f}_e) \leftarrow (e, \hat{f}_e + 1)$
    - (b) 否则, 如果  $|A| < 1/\epsilon$ , 将  $(e, 1)$  插入  $A$
    - 否则, 将  $(e, MIN + 1)$  插入  $A$ , 并删除一个满足  $\hat{f}_{e'} = MIN = \min\{\hat{f}_i\}$  的元素
  3. 查询  $i$ , 如果  $i \in A$ , **return**  $\hat{f}_i$ , 否则 **return**  $MIN$ ;

139 / 200

### 问题3：点查询- $\ell_1$ 近似

定理1.30

Metwally算法空间代价为  $O(\epsilon^{-1} \log mn)$ , 对任意查询  $i$ , 返回  $\hat{f}_i$  满足

$$f_i \leq \hat{f}_i \leq f_i + \epsilon n$$

- ▷ 如果不发生删除的情况, 那么  $\hat{f}_i = f_i$
- ▷ 如果删除, 计数一定不大于删除后的MIN, 有  $\hat{f}_i \geq f_i$
- ▷  $A$ 中元素和总是  $n$ ,  $MIN \frac{1}{\epsilon} \leq n \Rightarrow MIN \leq \epsilon n$
- ▷ 每个元素至多超出真实值MIN,  $\hat{f}_i \leq f_i + \epsilon n$

140 / 200

### 问题3：点查询- $\ell_1$ 近似

#### 支持删除的流模型 (turnstile)

- ① 给定数据流  $\sigma = \langle \pm a_1, \pm a_2, \dots, \pm a_n \rangle$ ,  $a_i \in [m]$
- ② 对应的频次向量为

$$\mathbf{f} = (f_1, \dots, f_m)$$

- ▷ 初始化向量  $\mathbf{f} \in \mathbb{R}^m$  为0
- ▷ 每次更新  $(i, c)$ , 使得  $f_i \leftarrow f_i + c$ ,  $f_i$ 可以比0小
- ▷  $c = 1$  即是之前所用的普通流模型 (Vanilla)
- ▷  $c > 0$  被称作Cash Register流模型

142 / 200

### 问题3：点查询- $\ell_1$ 近似

#### Turnstile模型

#### Count 算法[2005]

1.  $k = \frac{b}{\epsilon}$ ,  $C[1 \dots k] \leftarrow 0$ ;
2. 随机选择一个2-相对独立哈希函数  $h: [m] \rightarrow [k]$ ;
3. **for** 每一次更新  $(j, c)$  **do**
4.  $C[h(j)] = C[h(j)] + c$ ;
5. **return**  $\hat{f}_a = C[h(a)]$  对查询元素  $a$ ;

- ▷ 空间代价  $O(\frac{b}{\epsilon} \cdot \log n)$
- ▷  $\Pr[|\hat{f}_a - f_a| \geq \epsilon \|\mathbf{f}_{-a}\|_1] \leq \frac{1}{b}$ , 其中  $\|\mathbf{f}_{-a}\|_1 = \|\mathbf{f}\|_1 - |f_a|$

144 / 200

### 问题3：点查询- $\ell_1$ 近似

Cash Register模型:  $c > 0$

#### Count-Min 算法[2005]

1.  $C[1 \dots t][1 \dots k] \leftarrow 0$ ,  $k = \frac{2}{\epsilon}$  且  $t = \lceil \log \frac{1}{\delta} \rceil$ ;
2. 随机选择  $t$  个2-相对独立哈希函数  $h_i: [m] \rightarrow [k]$ ;
3. **for** 每一次更新  $(j, c)$  **do**
4. **for**  $i = 1$  to  $t$  **do**
5.  $C[i][h_i(j)] = C[i][h_i(j)] + c$ ;
6. **return**  $\hat{f}_a = \min_{1 \leq i \leq t} C[i][h_i(a)]$  对查询元素  $a$ ;

- ▷ 空间代价  $O(\frac{1}{\epsilon} \log \frac{1}{\delta} \cdot \log n)$

147 / 200

### 问题3：点查询- $\ell_1$ 近似

定理1.31[Count-Min]

Count-Min 算法以  $1 - \delta$  概率给出  $\ell_1$  近似-点查询的  $(1 + \epsilon)$  近似, 具体地

$$f_a \leq \hat{f}_a \leq f_a + \epsilon \|\mathbf{f}_{-a}\|_1$$

这里,  $\|\mathbf{f}_{-a}\|_1 = \|\mathbf{f}\|_1 - |f_a|$ .

148 / 200

### 问题3：点查询- $\ell_1$ 近似

#### Turnstile模型

#### Count-Median 算法[2005]

1.  $C[1 \dots t][1 \dots k] \leftarrow 0$ ,  $k = \frac{2}{\epsilon}$  且  $t = \lceil \log \frac{1}{\delta} \rceil$ ;
2. 随机选择  $t$  个2-相对独立哈希函数  $h_i: [m] \rightarrow [k]$ ;
3. **for** 每一次更新  $(j, c)$  **do**
4. **for**  $i = 1$  to  $t$  **do**
5.  $C[i][h_i(j)] = C[i][h_i(j)] + c$ ;
6. **return**  $\hat{f}_a = \text{median}_{1 \leq i \leq t} C[i][h_i(a)]$  对查询  $a$ ;

- ▷ 空间代价  $O(\epsilon^{-1} \log \delta^{-1} \cdot \log n)$

151 / 200

### 问题3：点查询- $\ell_1$ 近似

定理1.32[Count-Median]

Count-Median 算法以  $1 - \delta$  概率给出  $\ell_1$  近似-点查询的  $(1 \pm \epsilon)$  近似, 具体

$$|\hat{f}_a - f_a| \leq \epsilon \|\mathbf{f}_{-a}\|_1$$

153 / 200

### 问题3：点查询- $\ell_2$ 近似

Turnstile模型，质量更好的估计结果

#### CountSketch 算法[2004]

1.  $C[1 \dots k] \leftarrow \mathbf{0}, k = \frac{3}{\epsilon^2}$ ;
2. 随机选择1个2-相对独立哈希函数  $h: [m] \rightarrow [k]$ ;
3. 随机选择1个2-相对独立哈希函数  $g: [m] \rightarrow \{-1, 1\}$ ;
4. **for** 每一个更新  $(j, c)$  **do**
5.      $C[h(j)] = C[h(j)] + c \cdot g(j)$ ;
6. **return**  $\hat{f}_a = g(a) \cdot C[h(a)]$  对查询  $a$ ;

- ▷ 空间代价  $O(\frac{1}{\epsilon^2} \cdot \log n)$
- ▷ 针对Turnstile模型，以常数2/3概率，有  $(1 \pm \epsilon)$  近似比

156 / 200

### 问题3：点查询- $\ell_2$ 近似

利用Median技术改进估计质量

#### CountSketch+ 算法

1.  $C[1 \dots t][1 \dots k] \leftarrow \mathbf{0}, k = \frac{3}{\epsilon^2}, t = O(\log \frac{1}{\delta})$ ;
2. 随机选择  $t$  个2-相对独立哈希函数  $h_i: [m] \rightarrow [k]$ ;
3. 随机选择  $t$  个2-相对独立哈希函数  $g_i: [m] \rightarrow \{-1, 1\}$ ;
4. **for** 每一个更新  $(j, c)$  **do**
5.     **for**  $i = 1$  to  $t$  **do**
6.          $C[i][h_i(j)] = C[i][h_i(j)] + c \cdot g_i(j)$ ;
7. **return**  $\hat{f}_a = \text{median}_{1 \leq i \leq t} g_i(a) C[i][h_i(a)]$  对查询  $a$ ;

- ▷ 空间代价  $O(\frac{1}{\epsilon^2} \log \frac{1}{\delta} \cdot \log n)$

160 / 200

### 问题3：点查询- $\ell_2$ 近似

$t = \log \frac{1}{\delta}$  使得切尔诺夫不等式

定理1.33[CountSketch+]

CountSketch+ 算法至少以  $1 - \delta$  概率给出  $\ell_2$  近似-点查询的  $(1 \pm \epsilon)$  近似估计，具体地

$$|\hat{f}_a - f_a| \leq \epsilon \|\mathbf{f}_{-a}\|_2$$

161 / 200

### 问题3：点查询

点查询（频度估计）问题的算法

| Method       | $\hat{f}_a - f_a \in$                  | Space                                                     | Pr       | Mod.  |
|--------------|----------------------------------------|-----------------------------------------------------------|----------|-------|
| Misra-Gries  | $[-\epsilon \ \mathbf{f}_{-a}\ _1, 0]$ | $O(\frac{\log m}{\epsilon}) = O(\frac{\log n}{\epsilon})$ | 0        | +     |
| Metwally     | $[0, \epsilon \ \mathbf{f}_{-a}\ _1]$  | $O(\frac{\log m}{\epsilon}) = O(\frac{\log n}{\epsilon})$ | 0        | +     |
| CountSketch+ | $\pm \epsilon \ \mathbf{f}_{-a}\ _2$   | $O(\frac{1}{\epsilon^2} \log \frac{1}{\delta} \log n)$    | $\delta$ | $\pm$ |
| Count-Min    | $[0, \epsilon \ \mathbf{f}_{-a}\ _1]$  | $O(\frac{1}{\epsilon} \log \frac{1}{\delta} \log n)$      | $\delta$ | +     |
| Count-Median | $\pm \epsilon \ \mathbf{f}_{-a}\ _1$   | $O(\frac{1}{\epsilon} \log \frac{1}{\delta} \log n)$      | $\delta$ | $\pm$ |

- ▷  $m < n$
- ▷ + 表示Cash Register Model， $\pm$  表示Turnstile Model

162 / 200

### 问题4：频度矩估计

假设我们现在处理Vanilla Model

定义1.23[频度矩--Frequency Moments]

给定数据流  $\sigma = \langle a_1, \dots, a_n \rangle$ ，假设  $a_i \in [m]$ ，令  $\mathbf{f} = \mathbf{f}(\sigma) = (f_1, f_2, \dots, f_m)$ ，显然， $\sum f_i = n$ 。数据流  $\sigma$  的第  $k$  阶频度矩（ $k$ th frequency moment）表示为  $F_k(\sigma)$  或者  $F_k$ ，

$$F_k = \sum_{i=1}^m |f_i|^k = \|\mathbf{f}\|_k^k$$

- ▷  $F_0$  是不重复元素的数目
- ▷  $F_1$  是计数问题
- ▷  $F_2$  可表示关系数据自连接（self join）操作结果大小
- ▷ 这部分为  $F_k (k \geq 2)$  估计问题提供亚线性空间的算法

163 / 200

### 问题4：频度矩估计

#### Basic AMS 算法

1.  $(L, r, a) \leftarrow (0, 0, 0)$ ;
2. **for** 第  $j$  个元素  $a_j$  **do**
3.      $L \leftarrow L + 1$ ;
4.      $\beta \leftarrow$  random bit with  $\Pr[\beta = 1] = \frac{1}{L}$ ;
5.     **if**  $\beta = 1$  **then**
6.          $a \leftarrow a_j, r \leftarrow 0$ ;
7.     **if**  $a_j \neq a$  **then**
8.          $r \leftarrow r + 1$ ;
9. **return**  $X = L(r^k - (r - 1)^k)$ ;

164 / 200

### 问题4：频度矩估计

空间代价

- ▷ 存储  $L$  和  $r$  需要  $\log n$  位
- ▷ 存储  $a$  需要  $\log m$  位
- ▷ 共计  $O(\log m + \log n)$  位代价

直观思想

- ▷ 随机选取一个位置  $y \in [n]$
- ▷  $a = a_y$
- ▷  $r = |\{j | j \geq y \text{ 并且 } a_j = a_y\}|$

$$\Pr[|X - \mathbf{E}[X]| \geq \epsilon \mathbf{E}[X]] \leq \frac{kn^{1-\frac{1}{k}} F_k^2}{\epsilon^2 F_k^2} = \frac{kn^{1-\frac{1}{k}}}{\epsilon^2}$$

165 / 200

### 问题4：频度矩估计

#### Final AMS 算法

1. 利用Median-of-Mean技术调用Basic AMS算法;
2. 计算  $t = c \log \frac{1}{\delta}$  个平均值;
3. 每个平均值是  $r = \frac{3k}{2} \cdot n^{1-\frac{1}{k}}$  次调用的平均值;
4. **return**  $t$  个数值的中间值;

- ▷  $\Pr[|X - \mathbf{E}[X]| \geq \epsilon \mathbf{E}[X]] \leq \frac{kn^{1-\frac{1}{k}}}{\epsilon^2} = \frac{1}{3} \Rightarrow \delta$
- ▷ 空间代价：  
 $tr \cdot O(\log m + \log n) = O(\frac{1}{\epsilon^2} \log \frac{1}{\delta} \cdot kn^{1-\frac{1}{k}} (\log m + \log n))$
- ▷ 该算法在  $k = 2$  时，代价为  $\tilde{O}(\epsilon^{-2} n^{0.5})$ ，过大

172 / 200

## 问题4：频度矩估计

### Basic $F_2$ AMS 算法(Tug-of-War Sketch)

1. 随机选择4-相对独立哈希函数  $h: [m] \rightarrow \{-1, 1\}$ ;
2.  $x \leftarrow 0$ ;
3. **for** 每一次更新  $(j, c)$  **do**
4.      $x \leftarrow x + c \cdot h(j)$ ;
5. **return**  $x^2$ ;

- ▷  $E[X^2] = F_2$ ;  $\text{Var}[X^2] \leq 2F_2^2$
- ▷ 令  $t = c \log \frac{1}{\delta}$  且  $r = \frac{3\text{Var}[X^2]}{c^2 E[X^2]^2} \leq \frac{6}{c^2} \Rightarrow (\epsilon, \delta)$ -近似算法
- ▷ 空间代价  $O(\frac{1}{\epsilon^2} \log \frac{1}{\delta} \log n)$

173 / 200

## 问题5：频繁元素

定义1.24[频繁元素, Frequent Items, heavy hitters]

给定数据流  $\sigma$  和参数  $k$ , 输出频繁元素集合  $\{j \mid f_j > n/k\}$ .

利用点查询的算法求解

- ▷ 令  $\epsilon = 1/k$ , 执行Misra-Gries 算法, 输出  $A$  中元素

### Misra-Gries 算法[1982]

1.  $A \leftarrow \emptyset$ ;
2. 对每一个数据流中的元素  $e$ ,
  - (a) 如果  $e \in A$ , 令  $(e, f_e) \leftarrow (e, f_e + 1)$
  - (b) 否则, 如果  $|A| < 1/\epsilon$ , 将  $(e, 1)$  插入  $A$
  - (c) 否则, 将所有  $A$  中计数减1, 即  $(j, \hat{f}_j) \leftarrow (j, \hat{f}_j - 1)$ , 如果  $\hat{f}_j = 0$ , 从  $A$  中删除  $(j, 0)$
3. 对于查询  $i$ , 如果  $i \in A$ , **return**  $\hat{f}_i$ , 否则 **return** 0;

- ▷  $f_j > n/k \Rightarrow \hat{f}_j > 0 \Rightarrow j \in A$

177 / 200

## 问题5：频繁元素

- ▷ 想估计最大频度, 是否可以在亚线性代价内实现?
  - 考虑一个hard情况, 每个元素频度都差不多, 最后一个元素决定最大频度是多少
- ▷ 放松要求: 最频繁的元素是heavy的, 设计算法

定义1.25[( $\alpha, p$ )-Heavy Hitters]

给定参数  $\alpha \in (0, 1)$  和  $p \in [1, \infty)$ , 对于数据流  $\sigma$ , 假设对应的频度向量为  $\mathbf{f}$ , ( $\alpha, p$ )-Heavy Hitters定义如下

$$HH_\alpha^p(\mathbf{f}) = \{i \in [m] : f_i \geq \alpha \|\mathbf{f}\|_p\}$$

定义1.26[ $\alpha$ -Heavy Hitters]

元素  $i$  是  $\alpha$ -heavy的, 如果  $f_i \geq \alpha \|\mathbf{f}\|_1 = \alpha \sum f_j$ . 上述元素  $i$  构成的集合表示为  $HH_\alpha$ .

178 / 200

## 问题5：频繁元素

定义1.27[Approximate  $\alpha$ -Heavy Hitters]

给定参数  $\alpha, \epsilon \in (0, 1)$ , 对于数据流  $\sigma$ , 假设对应的频度向量为  $\mathbf{f}$ , 计算一个集合  $S \subseteq [m]$  使得

$$HH_\alpha \subseteq S \subseteq HH_{(1-\epsilon)\alpha}$$

利用  $\ell_1$  点查询算法求解, 设计  $(\epsilon, \delta)$  近似算法

- ① 令  $\epsilon' = \epsilon\alpha/2$  且  $\delta' = \delta/m$ , 用Count-Median实现  $(\epsilon', \delta')$  近似点查询算法, 为每个元素  $i \in [m]$  估计  $\hat{f}_i$
- ② 返回集合  $S = \{i \in [m] : \hat{f}_i \geq (\alpha - \epsilon\alpha/2) \|\mathbf{f}\|_1\}$
- ▷ 算法空间代价:  $O(\frac{1}{\alpha\epsilon} (\log \frac{1}{\delta} + \log m) \log n)$

179 / 200

## 问题5：频繁元素

- ① 给定元素  $i \in HH_\alpha$ , 那么  $f_i \geq \alpha \|\mathbf{f}\|_1$ 

$$\hat{f}_i \geq f_i - \epsilon\alpha/2 \|\mathbf{f}\|_1 \geq (\alpha - \epsilon\alpha/2) \|\mathbf{f}\|_1 \Rightarrow i \in S$$
- ② 给定元素  $i \in S$ , 那么  $\hat{f}_i \geq (\alpha - \epsilon\alpha/2) \|\mathbf{f}\|_1$ 

$$\hat{f}_i \leq f_i + \epsilon\alpha/2 \|\mathbf{f}\|_1$$

$$\Rightarrow f_i \geq \hat{f}_i - \epsilon\alpha/2 \|\mathbf{f}\|_1 \geq (1 - \epsilon)\alpha \|\mathbf{f}\|_1 \Rightarrow i \in HH_{(1-\epsilon)\alpha}$$

- ▷ 至少以  $1 - \delta$  的概率,  $\forall i \in [m]$  上述①②都满足
  - $\forall i \in [m]$  ① 成立, 那么有  $HH_\alpha \subseteq S$
  - $\forall i \in [m]$  ② 成立, 那么有  $S \subseteq HH_{(1-\epsilon)\alpha}$

180 / 200

## 问题6：固定大小采样

- ▷ 随机采样是一个非常重要的工具
- ▷ 随机采样: 从大小为  $n$  的数据集中均匀的选择一个大小为  $s$  的采样
  - 一个想法: 遍历数据, 以  $\frac{s}{n}$  的概率选择每个数据
  - 缺点: 事先需要知道  $n$ , 无法保证样本集合大小

### 水库抽样算法-样本大小为1

1.  $n \leftarrow 0$ ;
2. **for** 每一个元素  $x$  **do**
3.      $n \leftarrow n + 1$ ;
4.     以  $\frac{1}{n}$  概率令  $S \leftarrow x$ ;
5. **return**  $S$ ;

182 / 200

## 问题6：固定大小采样

定理1.35[水库抽样算法的正确性]

令  $n$  为当前数据流的长度, 水库抽样算法的输出是均匀的, 即, 对任意的  $i \in [1, n]$ ,  $\Pr[S = x_i] = \frac{1}{n}$ .

- ▷ 可以用数学归纳法证明
- ▷ 也可以从如下公式得到

$$\Pr[S = x_i] = \frac{1}{i} \prod_{n \geq j > i} \left( \frac{j-1}{j} \right) = \frac{1}{n}$$

- ▷ 获取大小为  $s$  的均匀采样 (有放回情况)
  - 将样本大小为1的算法同时独立执行  $s$  份

183 / 200

## 问题6：固定大小采样

- ▷ 获取大小为  $s$  的均匀采样 (无放回情况)

### 水库抽样算法

1.  $n \leftarrow 0$ ;
2. 用流数据的前  $s$  个元素初始化  $A[1, \dots, s]$ ,  $n \leftarrow s$ ;
3. **for** 每一个元素  $x$  **do**
4.      $n \leftarrow n + 1$ ;
5.     令  $r$  为  $[1, n]$  内的随机整数;
6.     **if**  $r \leq s$  **then**
7.          $A[r] \leftarrow x$ ;

- ▷ 证明:  $A$  是从数据流中以无放回形式均匀抽取的样本

184 / 200

## 问题6：固定大小采样

- ▷ 扩展到weighted的情况
- ▷ 当 $s = 1$ 时，是以 $\Pr[S = x_i] = \frac{w_i}{\sum w_i}$ 概率选取数据
  - 水库抽样算法可以很容易扩展到这个情况

### 带权重水库抽样算法-样本大小为1

1.  $W \leftarrow 0$ ;
2. **for** 每一个元素  $x$  **do**
3.      $W \leftarrow W + w_x$ ;
4.     以  $\frac{w_x}{W}$  概率令  $S \leftarrow x$ ;
5. **return**  $S$ ;

- ▷ 样本大小为 $s$ 的有放回采样：同时独立运行算法 $s$ 次

186 / 200

## 问题6：固定大小采样

- ▷ 有权重、无放回、样本大小为 $s$
- ▷ 定义可以通过观察如下概念式算法来理解
  - 当前数据流总权重 $W$ ，以 $\Pr[x_i] = \frac{w_i}{W}$ 选一个数据
  - 重复上述步骤 $s$ 次，注意每次 $W$ 不同

### 带权重水库抽样算法-无放回抽取 $s$ 个样本

1. **for** 每一个元素  $x_i$  **do**
2.      $r_i \leftarrow (0, 1)$ 间的均匀随机数;
3.      $w'_i = r_i^{1/w_i}$ ;
4.      $w[1, \dots, s]$ 维护上述过程中得到的最大的 $s$ 个 $w'$ ;
5.      $S[1, \dots, s]$ 为对应的 $s$ 个数据;
6. **return**  $S$ ;

187 / 200

## 问题7：Bloom Filter

### 定义1.28[Membership Problem]

令 $U$ 是整数集合 $\{1, \dots, |U|\}$ ， $S$ 是 $U$ 的一个大小为 $n$ 的子集，预处理 $S$ ，给定整数 $q \in U$ ，判断是否 $q \in S$ 。

- ▷  $U$ 中整数可以用 $w = \log |U|$ 位表示
- ▷ 哈希表，查询期望时间是 $O(1)$ 
  - 如果要求精确答案，那么至少用 $\log \binom{|U|}{n}$ 位，当 $|U| \gg n$ ，即 $\Omega(nw)$
  - 是否可以在 $o(nw)$ 代价解决？
- ▷ 放松要求
  - ☑ 错误判断 $q \in S$  (false positives)
  - ☒ 错误判断 $q \notin S$  (false negatives)

### 定义1.29[Approximate Membership Problem]

给定整数 $q \in U$ ，设计算法① $q \in S \Rightarrow$ 输出‘yes’，② $q \notin S \Rightarrow$ 以至少 $1 - \delta$ 概率输出‘no’。

192 / 200

## 问题7：Bloom Filter

### 定义1.30[Ideal Hash Function]

哈希函数 $h: U \rightarrow [m]$ 是理想的 (ideal)，如果对每一个 $k \in U$ ， $h(k)$ 均匀独立地从 $[1, m]$ 间取值。

### 近似哈希的方法

1. 令 $\mathcal{H}$ 是独立理想哈希函数族： $|U| \rightarrow [m]$ ， $m = \frac{n}{\delta}$ ;
2. 随机选取 $h \in \mathcal{H}$ ，并维护数组 $A[m]$ ;
3. **for**  $i \in S$  **do**  $A[h(i)] = 1$ ;
4. 给定查询 $q$ ，**return** ‘yes’ 当且仅当 $A[h(i)] = 1$ ;

- ▷  $q \in S \Rightarrow$ 输出‘yes’
- ▷  $q \notin S \Rightarrow \sum_{j \in S} \Pr[h(q) = h(j)] \leq \frac{n}{m} = \delta$

193 / 200

## 问题7：Bloom Filter

### Bloom Filter方法

1. 令 $\mathcal{H}$ 是独立理想哈希函数族： $|U| \rightarrow [m]$ ;
2. 随机选取 $h_1 \dots h_k \in \mathcal{H}$ ，并维护数组 $A[m]$ ;
3. **for**  $i \in S$  **do**
4.     **for**  $j \in [1, k]$  **do**
5.          $A[h_j(i)] = 1$ ;
6. 对查询 $q$ ，**return** ‘yes’  $\Leftrightarrow \forall j \in [k], A[h_j(q)] = 1$ ;

- ▷ 代价： $m = O(n \log \frac{1}{\delta}) \Rightarrow O(n \log \frac{1}{\delta})$

194 / 200

## 内容总结

外存算法

亚线性空间算法    大数据算法    并行算法

亚线性时间算法

- ▷ 李建中老师课题组（海量数据计算研究中心）
- ▷ 联系方式：liuxianmin@hit.edu.cn、lijzh@hit.edu.cn