大数据计算基础

BIG DATA COMPUTING

刘显敏 海量数据
liuxianmin@hit.edu.cn

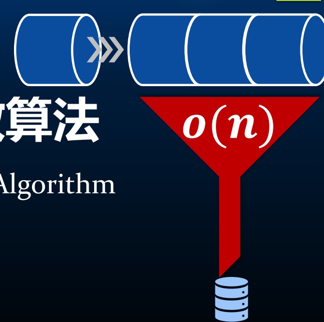
大数据算法

2025年秋

2

空间高效算法

Space Efficient Algorithm



流模型/One-Pass计算模型

3

- 输入是数据流的方式，可由$\langle a_1, a_2, . . . , a_n\rangle$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

路由器监测
数据库查询

multi-pass
semi-streaming

问题1：近似计数

4

流模型（Streaming Model）

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

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

定义 [The Counting Problem]

给定数据流σ以及a_i，计算f_i。

问题1：近似计数

5

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 [The Approximate Counting Problem]

给定数据流σ以及a_i，计算$\hat{f}_i$，满足

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

$\epsilon = 1/3$
$\delta = 1\%$

问题1：近似计数

6

Morris算法

/** 维护一个计数器X **/

1. $X \leftarrow 0$;
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** $\hat{f}_i = 2^X - 1$

离散型随机变量X

期望

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

方差

$$\mathbf{var}[X] = \mathbf{E}[(X - \mathbf{E}[X])^2] = \mathbf{E}[X^2] - (\mathbf{E}[X])^2$$

Morris算法近似性能分析

- 期望 $\mathbf{E}[2^X - 1] = f_i$
- 方差 $\mathbf{var}[2^X - 1] = f_i(f_i - 1)/2 \le f_i^2/2 = O(f_i^2)$
- 利用切比雪夫不等式

问题1：近似计数 7

- 两个有用的概率不等式

马尔可夫(Markov)不等式

对任意非负随机变量X以及常数$a>0$，有

$$\mathbf{Pr}(X \geq a) \leq \frac{\mathbf{E}[X]}{a}$$

切比雪夫(Chebyshev)不等式

$$\mathbf{Pr}(|X-\mathbf{E}[X]| \geq a) \leq \frac{\mathbf{var}[X]}{a^2}$$

问题1：近似计数 8

- 切比雪夫：$\mathbf{Pr}(|Z-\mathbf{E}[Z]| \geq a) \leq \frac{\mathbf{var}[Z]}{a^2}$
  - 令$a=\sqrt{\frac{\mathbf{var}[Z]}{c}}$，其中$c$是任意常数
  - 则有，$\mathbf{Pr}\left(|Z-\mathbf{E}[Z]| \geq \sqrt{\frac{\mathbf{var}[Z]}{c}}\right) \leq c$
  - 即，$Z=\mathbb{E}[Z] \pm \sqrt{\mathbf{var}[Z]/c}$ 至少以1-c概率成立
- $Z=2^X-1$，至少以$1-c$概率，Morris算法

$$\hat{f}_i = \mathbf{2^X-1} = \mathbb{E}[\mathbf{2^X-1}] \pm \sqrt{\frac{\mathbf{var}[2^X-1]}{\boldsymbol{c}}}$$

问题1：近似计数 9

- 至少以$1-c$概率，Morris算法

$$\hat{f}_i = \mathbf{2^X-1} = \mathbb{E}[\mathbf{2^X-1}] \pm \sqrt{\frac{\mathbf{var}[2^X-1]}{\boldsymbol{c}}}$$

$\mathbf{E[2^X-1]} = f_i$

$\mathbf{var}[2^X-1] \leq f_i^2/2 = \mathrm{O}(f_i^2)$

$$\hat{f}_i = f_i \pm \sqrt{\frac{1}{2c}} f_i = \left(1 \pm \sqrt{\frac{1}{2c}}\right) f_i$$

即 $\hat{f}_i = f_i \pm O(f_i)$

问题1：近似计数 10

Morris算法的**空间代价分析**

定理 [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$f(y)=2^y$，有$2^{\mathbb{E}[X]} \leq \mathbb{E}[2^X]=f_i+1$

$$\mathbb{E}[X] \leq \log(f_i+1)$$

$$\Rightarrow \mathbb{E}[\mathrm{space}(X)] \approx \mathrm{loglog} f_i = O(\mathrm{loglog}\, n)$$

问题1：近似计数 12

如何获得一个$1+\epsilon$近似算法

Morris算法：$\hat{f}_i = f_i \pm \sqrt{\frac{1}{2c}} f_i = \left(1 \pm \sqrt{\frac{1}{2c}}\right) f_i$

2. **return** $\frac{1}{k}\sum_{i=1}^{k}(2^{X_i}-1)$;

Morris+算法的估计结果分析

- 期望$\mathbf{E}[Y]=\mathbf{E}\left[\frac{1}{k}\sum_{i=1}^{k}(2^{X_i}-1)\right]=f_i$
- 方差$\mathbf{var}[Y]=\frac{f_i(f_i-1)}{2k} \leq \frac{f_i^2}{2k} = O\left(\frac{f_i^2}{k}\right)$
- 利用切比雪夫不等式

问题1：近似计数 13

定理 [Morris+]至少$1-c$概率, Morris+算法满足

$$Y=f_i \pm \frac{1}{\sqrt{2kc}} f_i$$

- 令$k=\frac{1}{2c\epsilon^2}=O\left(\frac{1}{\epsilon^2}\right)$, Morris+是$(1\pm\epsilon)$近似算法
- 令$k=\frac{1}{2\delta\epsilon^2}=O\left(\frac{1}{\epsilon^2}\frac{1}{\delta}\right)$, 可将概率改进为$1-\delta$
  - 空间代价为$O\left(\frac{\mathrm{loglog}\, n}{\epsilon^2\delta}\right)$

问题1：近似计数 14

- 以$1-\delta$概率$(1\pm\epsilon)$-近似, Morris+需$k = O(\frac{1}{\epsilon^2}\frac{1}{\delta})$
- 利用Morris+算法$\mathcal{M}$：以概率$1-c \geq 0.9$得到$1\pm\epsilon$近似，设计一个$1-\delta$概率$(1\pm\epsilon)$-近似算法
- 空间代价为$O\left(\frac{1}{\epsilon^2}\log\frac{1}{\delta}\mathrm{loglo}\right.$

算法分析
(1)计算期望
(2)利用切尔诺夫不等式

Morris++算

/** *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;

问题1：近似计数 15

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

令$X_1, X_2, \ldots, X_m$是独立、取值{0, 1}的随机变量，变量和的期望为$\mu = \mathbb{E}[\sum_i X_i]$，令$\epsilon \in [0,1]$，则有

$$\Pr\left[\left|\sum_i X_i - \mu\right| > \epsilon\mu\right] \leq 2e^{-\epsilon^2\mu/3}$$

问题1：近似计数 16

算法分析

- $X_i = 1 \Leftrightarrow$第i个Morris+算法$\mathcal{M}$输出结果符合要求

若要$\Pr[\sum_i X_i < 0.5t] \leq e^{-c't} < \delta$
只需$t = O\left(\log\left(\frac{1}{\delta}\right)\right)$

9t

- 易知 法输出结果将符合要求，那么错误概率如下

$$t = O\left(\log\left(\frac{1}{\delta}\right)\right)$$

$$\Rightarrow 空间代价:\ O\left(\frac{1}{\epsilon^2}\log\frac{1}{\delta}\log\log n\right)$$

问题3：固定大小采样 17

- 随机采样是一个非常重要的工具
- 随机采样：从大小为n的集合中均匀选择s个样本
  - 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 ← 0;
2. **for** 每一个元素 x **do**
3.   n ← n + 1;
4.   以 $\frac{1}{n}$ 概率令S ← x;
5. **return** S;

问题3：固定大小采样 18

定理[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$i \in [1, n]$，$\Pr[S = x_i] = \frac{1}{n}$。

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

- 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

问题3：固定大小采样 19

- 获取大小为s的均匀采样（无放回情况）

水库抽样算法

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

- 证明：A是从数据流中以无放回形式均匀抽取的样本

问题3：固定大小采样 20

- 扩展到weighted的情况
- 当s = 1时，是以$\mathbf{Pr}[S = x_i] = \frac{w_i}{\sum w_i}$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 ← 0;
2. **for** 每一个元素 x **do**
3. W ← W + w_x;
4. 以 $\frac{w_x}{W}$ 概率令S ← 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

问题3：固定大小采样 21

- 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$\mathbf{Pr}[x_i] = \frac{w_i}{W}$ 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 xi **do**
2. r_i ← (0, 1)间的均匀随机数;
3. $w_i' = r_i^{1/w_i}$;
4. S[1, . . . ,s]为最大的s个w'对应的s个数据;
5. **return** S;

问题4：Bloom Filter 22

- Bloom Filter（布隆过滤器）由Bloom于1970年提出
  - Burton H. Bloom. **Space/time trade-offs in hash coding with allowable errors**. *Commun. ACM*, 13(7), 1970, 422–426.
- 例1：垃圾邮件检查
  - 手头有一个垃圾邮件来源地址列表
- 例2：广告推荐、订阅-发布系统
  - 关键词列表
- 简单做法：遍历一遍名单或者列表
- 大数据场景下，名单太长怎么办

问题4：Bloom Filter 23

定义[Membership Problem]令 U 是整数集合 {1, . . . , |U|}，S 是 U 的一个大小为 n 的子集合，预处理 S，给定整数 q ∈ U，判断是否 q ∈ 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 Ω(nw)
  - 是否可以在 o(nw) 代价解决?

问题4：Bloom Filter 24

- 放松要求
  - √ 错误判断q ∈ S（false positives）
  - × 错误判断q ∉ S（false negatives）

定义[Approximate Membership Problem]给定整数 q ∈ U，设计算法 ① q ∈ S ⟹输出 ‘yes’，② q ∉ S⟹以至少 1 − δ 概率输出 ‘no’。

问题4：Bloom Filter 25

定义[Ideal Hash Function] 哈希函数 h : U → [m]是理想的，若对每个 k ∈ 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| → [m]，m = $\frac{n}{\delta}$;
2. 随机选取 h ∈ $\mathcal{H}$，并维护数组A[m];
3. **for** i ∈ S **do** A[h(i)] = 1;
4. 给定查询 q，**return** ‘yes’ 当且仅当A[h(i)] = 1;

- $q \in S$ ⟹输出 ‘yes’
- $q \notin S \Longrightarrow \sum_{j\in S} \mathbf{Pr}[h(q) = h(j)] \le \frac{n}{m} = \delta$

问题4：Bloom Filter

26

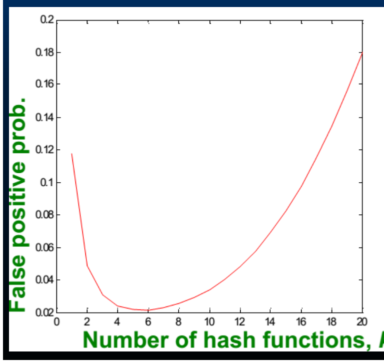
Bloom Filter方法

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

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

问题4：Bloom Filter

27



性质总结

- 需要被过滤掉的一定被过滤
- 适合做预处理
- 适合硬件实现
- 适合并行

问题2：不重复元素数

28

流模型（Streaming Model）

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

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

定义 [The Distinct Elements Problem]

给定数据流σ，计算$\sum_{1 \le i \le m} \mathbf{I}[f_i > 0]$。

$$\mathbf{I}[f_i > 0] = 1 \Leftrightarrow f_i > 0$$

问题2：不重复元素数

29

精确计算需要多少空间代价？

- 方法1： $O(m)$ 位.
  - 为$[m]$中每个元素维护1个bit, 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.

问题2：不重复元素数

30

- 一个理想化的方案：假设可以存储实数，真实值为D
- 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，随机选取哈希函数h : [m] ↦ [0, 1];
2. for 每一个元素i do
3.     z = min{z, h(i)};
4. return 1/z − 1.
```

问题2：不重复元素数

31

- FM算法的分析：令算法结果$X = \frac{1}{z} - 1$
- $\mathbb{E}[z] = \frac{1}{D+1}$，$\mathbf{var}[z] \le \frac{2}{(D+1)(D+2)}$

$$\Pr\left[\left|z - \frac{1}{D+1}\right| > c\frac{1}{D+1}\right] < \frac{2(D+1)^2}{c^2 \cdot (D+1)(D+2)} < \frac{2}{c^2}$$

$$\Pr[|X - D| > \epsilon D] \le \Pr\left[\left|z - \frac{1}{D+1}\right| > \frac{\epsilon D}{(D+1+\epsilon D)}\frac{1}{(D+1)}\right]$$

令$c = \frac{\epsilon D}{D+1+\epsilon D}$

$$< \frac{2(D+1+\epsilon D)^2}{\epsilon^2 D^2} = 2\left(\frac{D+1}{\epsilon D} + 1\right)^2 < 2\left(\frac{2}{\epsilon} + 1\right)^2$$

问题2：不重复元素数

32

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] ↦ [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.

问题2：不重复元素数

33

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

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

令$c = \frac{\epsilon D}{D+1+\epsilon D} \Longrightarrow$

$$\mathbf{Pr}[|X - D| > \epsilon D] < \frac{2(D+1+\epsilon D)^2}{k\epsilon^2 D^2} < \frac{2}{k}\left(\frac{2}{\epsilon} + 1\right)^2$$

- 这里，ϵ为精度要求，假设概率要求为1 − δ，只需

$$\frac{2}{k}(\frac{2}{\epsilon} + 1)^2 < \delta$$

$$\Rightarrow k > \frac{2}{\delta}\left(\frac{2}{\epsilon} + 1\right)^2 = O(\frac{1}{\epsilon^2\delta})$$

问题2：不重复元素数

34

利用Median技术

FM++算法

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

- FM++是以1 − δ概率保证(1 ± ϵ)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

问题2：不重复元素数

35

然而，我们无法存储实数，需要更实际的算法

- 设计一个$O(\log m)$位空间代价的$O(1)$近似算法

$$\frac{1}{C} \times D \le \hat{D} \le C \times D$$

Practical FM算法

1. 随机选取**2-相对独立哈希函数**$h: [m] \mapsto [m]$;
2. $z = 0$;
3. **for** 每一个元素j **do**
   $z = \max\{z, \mathrm{zeros}(h(j))\}$;
4. **return** $\hat{D} = 2^{z+1/2}$;

$\mathbf{zeros}(p) = \mathbf{max}\{i | p\%2^i = 0\}$

0 1 0 1 0 0

$\mathbf{zeros}(\cdot) = 2$

问题2：不重复元素数

36

定义[2-相对独立哈希函数]

一个从[X]到[Y]的哈希函数族$\mathcal{H}$是**2-相对独立的**，若对$\forall h \in \mathcal{H}, a, b \in [Y]$及$i, j \in [X]$ (满足$i \ne j$), 有

$$\mathbf{Pr}(h(i) = a \wedge h(j) = b) = \frac{1}{Y^2}$$

定义[k-相对独立哈希函数]

哈希函数族$\mathcal{H}$是**k-相对独立的**，若对$\forall h \in \mathcal{H}$，$j_1, \dots, j_k \in [Y]$及$i_1, \dots, i_k \in [X]$ (满足$i_x \ne i_y$), 有

$$\mathbf{Pr}(h(i_1) = j_1 \wedge \dots \wedge h(i_k) = j_k) = 1/Y^k$$

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

问题2：不重复元素数

37

2-相对独立哈希函数族的例子

假设Y是一个素数(若不是，可以取比Y大的第一个素数), 随机选$p, q \in \{0,1,2,\dots, Y\text{-}1\}$, 函数族如下

$$\mathcal{H} = \{h_{p,q}(x) = (px + q)\%Y\}$$

- 显然，存储一个h只需$O(\log Y)$位的空间代价

k-相对独立哈希函数的例子

- $\mathcal{H}_{poly}$：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$\mathcal{H}_{poly}$某个函数,需$O(2 \log n)$位

问题2：不重复元素数 38

Practical FM算法

1. 随机选取**2-相对独立哈希函数**$h: [m] \mapsto [m]$;
2. $z = 0$;
3. **for** 每一个元素j **do**
   $z = \max\{z, \mathrm{zeros}(h(j))\}$;
4. **return** $\hat{D} = 2^{z+1/2}$;

存储哈希函数需$O(\log m)$位；存储z需$\log\log m$位

定理 Practical FM算法以$1 - \frac{2\sqrt{2}}{C}$概率满足$\frac{D}{C} \le \hat{D} \le C \times D$。

问题2：不重复元素数 39

定理 Practical FM算法以$1 - \frac{2\sqrt{2}}{C}$概率满足$\frac{D}{C} \le \hat{D} \le C \times D$。

- 令$X_{r,j} = 1 \Leftrightarrow \mathrm{zeros(h(j))} \ge r$，其中$j \in [m]$且$r \ge 0$
- 令$Y_r = \sum_{j \in \sigma} X_{r,j} = \sum_{j: f_j > 0} X_{r,j}$，$Y_0$即是不重复元素数目
  - Y_r：哈希值二进制尾部**0的数目** $\ge r$的元素数
- 假设算法结束时，$z = t$
  - $t \ge r \Leftrightarrow \exists j$有$\mathrm{zeros(h(j))} \ge r \Leftrightarrow X_{r,j} = 1 \Leftrightarrow Y_r > 0$
  - $t < r \Leftrightarrow \forall j$有$\mathrm{zeros(h(j))} < r \Leftrightarrow X_{r,j} = 0 \Leftrightarrow Y_r = 0$

Median技术⇒1−δ概率保证$O(1)$近似，空间$O(\log\frac{1}{\delta}\log m)$

Practical FM: ❶$z = \max\{z, \mathrm{zeros}(h(j))\}$; **❷** $\hat{D} = 2^{z+1/2}$;

问题2：不重复元素数 40

- 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] → [m];
2. 随机选2-相对独立哈希函数g : [m] → $[b\epsilon^{-4}\log^2 m]$;
3. $z = 0$, B = ∅;
4. **if** zeros(h(j)) ≥ z **then**
5. $B = B \cup \{(g(j), zeros(h(j)))\}$;
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** $\hat{D} = |B|2^z$;

问题2：不重复元素数 41

定理[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，空间代价为$O(\log m + \frac{1}{\epsilon^2}(\log\frac{1}{\epsilon} + \log\log m))$
- Ziv Bar-Yossef et al. Counting Distinct Elements in a Data Stream. In: *RANDOM 2002*