COMMONWEALTH OF AUSTRALIA

Copyright Regulations 1969

WARNING

This material has been reproduced and communicated to you by or on behalf of the University of Sydney pursuant to Part VB of the Copyright Act 1968 (**the Act**). The material in this communication may be subject to copyright under the Act. Any further copying or communication of this material by you may be the subject of copyright protection under the Act.

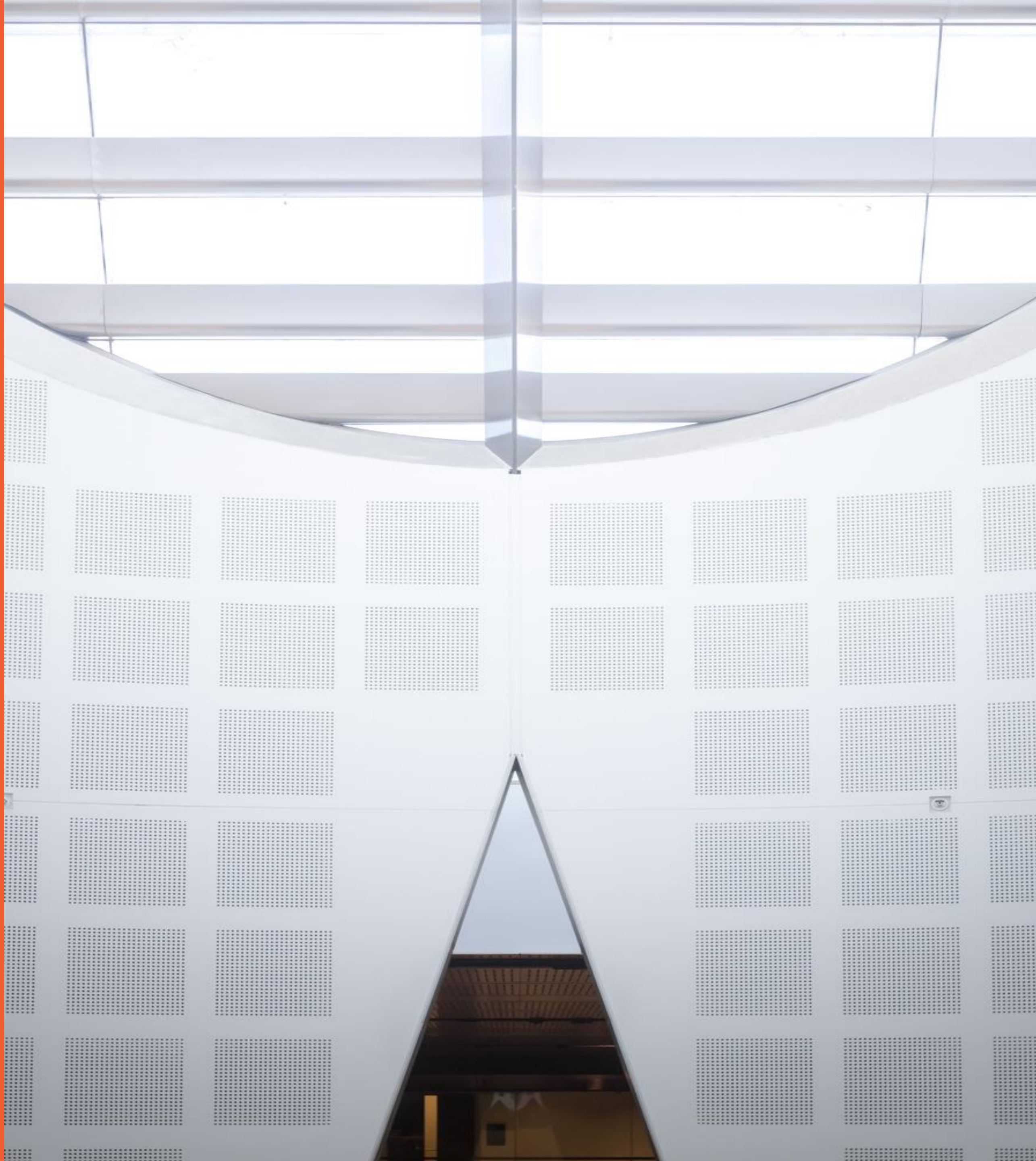
Do not remove this notice.

COMPx270: Randomised and Advanced Algorithms
Lecture 9: Streaming and Sketching II

Clément Canonne

School of Computer Science

THE UNIVERSITY OF SYDNEY



Some housekeeping

- Mid*-semester break!
- A3 is out (Oct 11 +5), A2 (after Simple Extension) due tomorrow
- Don't forget the "participation" assignment (Oct 18)
- I will be releasing a sample exam over the break (Friday w.h.p.)
- Feedback welcome

Back from last week: Distinct Elements

Distinct Elements, the Tidemark (AMS) algorithm (1/4)

```
1: Pick $h: [n] \to [n]$ from a strongly universal hashing family
2: $z \leftarrow 0$
3: **for all** $1 \leq i \leq m$ **do**
4:     Get item $a_i \in [n]$
5:     **if** $\text{zeros}(h(a_i)) \geq z$ **then**
6:         $z \leftarrow \text{zeros}(h(a_i))$
7: **return** $\sqrt{2} \cdot 2^z$
```

Distinct Elements, the Tidemark (AMS) algorithm (2/4)

```
1: Pick $h\colon [n] \to [n]$ from a strongly universal hashing family
2: $z \leftarrow 0$
3: for all $1 \leq i \leq m$ do
4:     Get item $a_i \in [n]$
5:     if $\text{zeros}(h(a_i)) \geq z$ then
6:         $z \leftarrow \text{zeros}(h(a_i))$
7: return $\sqrt{2} \cdot 2^z$
```

Distinct Elements, the Tidemark (AMS) algorithm (3/4)

```
1: Pick $h\colon [n] \to [n]$ from a strongly universal hashing family
2: $z \leftarrow 0$
3: for all $1 \le i \le m$ do
4:     Get item $a_i \in [n]$
5:     if $\mathrm{zeros}(h(a_i)) \ge z$ then
6:         $z \leftarrow \mathrm{zeros}(h(a_i))$
7: return $\sqrt{2} \cdot 2^z$
```

Distinct Elements, the Tidemark (AMS) algorithm (4/4)

Theorem 42. *The (median trick version of the)* Tidemark *(AMS) algorithm is a* randomised *one-pass algorithm which, for any given parameter* $\delta \in (0,1]$, *provides an estimate* $\widehat{d}$ *of the number* d *of distinct elements of the stream such that, for some absolute constant* $C > 0$,

$$\Pr\left[\frac{1}{C} \cdot d \leq \widehat{d} \leq C \cdot d\right] \geq 1 - \delta$$

with space complexity

$$s = O\left(\log n \cdot \log \frac{1}{\delta}\right).$$

Can we do better?

Distinct Elements, the BJKST algorithm (1/4)

```
Input: Parameter $\varepsilon \in (0,1]$
1:  Set $k \leftarrow O(\log^2 n/\varepsilon^4)$, $T \leftarrow \Theta(1/\varepsilon^2)$
2:  Pick $h\colon [n] \to [n]$ from a strongly universal hashing family
3:  Pick $g\colon [n] \to [k]$ from a strongly universal hashing family

4:  $z \leftarrow 0$, $B \leftarrow \emptyset$
5:  for all $1 \le i \le m$ do
6:      Get item $a_i \in [n]$
7:      if $\mathrm{zeros}(h(a_i)) \ge z$ then
8:          $B \leftarrow B \cup \{(g(a_i), \mathrm{zeros}(h(a_i)))\}$
9:          while $|B| \ge T$ do
10:             $z \leftarrow z + 1$
11:             Remove every $(a, b)$ with $b < z$ from $B$
12: return $|B| \cdot 2^z$
```

Distinct Elements, the BJKST algorithm (2/4)

```
Input: Parameter $\varepsilon \in (0,1]$
1:  Set $k \leftarrow O(\log^2 n/\varepsilon^4)$, $T \leftarrow \Theta(1/\varepsilon^2)$
2:  Pick $h\colon [n] \to [n]$ from a strongly universal hashing family
3:  Pick $g\colon [n] \to [k]$ from a strongly universal hashing family

4:  $z \leftarrow 0$, $B \leftarrow \emptyset$
5:  for all $1 \le i \le m$ do
6:      Get item $a_i \in [n]$
7:      if $\text{zeros}(h(a_i)) \ge z$ then
8:          $B \leftarrow B \cup \{(g(a_i), \text{zeros}(h(a_i)))\}$
9:          while $|B| \ge T$ do
10:             $z \leftarrow z + 1$
11:             Remove every $(a,b)$ with $b < z$ from $B$
12: return $|B| \cdot 2^z$
```

Distinct Elements, the BJKST algorithm (2/4)

```
Input: Parameter $\varepsilon \in (0,1]$
1: Set $k \leftarrow O(\log^2 n/\varepsilon^4)$, $T \leftarrow \Theta(1/\varepsilon^2)$
2: Pick $h\colon [n] \to [n]$ from a strongly universal hashing family
3: Pick $g\colon [n] \to [k]$ from a strongly universal hashing family

4: $z \leftarrow 0$, $B \leftarrow \emptyset$
5: for all $1 \le i \le m$ do
6:     Get item $a_i \in [n]$
7:     if $\mathrm{zeros}(h(a_i)) \ge z$ then
8:         $B \leftarrow B \cup \{(g(a_i), \mathrm{zeros}(h(a_i)))\}$
9:         while $|B| \ge T$ do
10:            $z \leftarrow z + 1$
11:            Remove every $(a,b)$ with $b < z$ from $B$
12: return $|B| \cdot 2^z$
```

Distinct Elements, the BJKST algorithm (4/4)

Theorem 43. *The (median trick version of the)* BJKST *algorithm is a* randomised *one-pass algorithm which, for any given parameters* $\varepsilon, \delta \in (0,1]$*, provides an estimate* $\widehat{d}$ *of the number* d *of distinct elements of the stream such that, for some absolute constant* $C > 0$,

$$\Pr\left[(1-\varepsilon)\cdot d \le \widehat{d} \le (1+\varepsilon)d \right] \ge 1-\delta$$

with space complexity

$$s = O\left(\left(\log n + \frac{\log(1/\varepsilon) + \log\log n}{\varepsilon^2} \right) \cdot \log\frac{1}{\delta} \right).$$

… Can we do better?

Back to Week 9!

A question 🪡

You design a streaming algorithm A to solve some problem. The stream of data arrives.

A question 🪡

You design a streaming algorithm A to solve some problem. The stream of data arrives.

628, 516, 163, 509, 15, 499, 772, 588, 737, 439, 79, 866, 186, 18, 854, 459, 146, 518, 748, 737, 685, 188, 939, 724, 27, 719, 263, 795, 120, 573, 853, 132, 522, 3, 298, 123, 932, 993, 180, 674, 1, 619, 989, 142, 496, 178, 191, 524, 716, 501, 677, 712, 452, 768, 591, 551, 439, 397, 229, 214, 43, 639, 353, 610, 737, 203, 933, 279, 877, 30, 513, 518, 616, 714, 633, 804, 422, 731, 867, 184, 124, 881, 595, 193, 254, 240, 4, 260, 303, 319, 757, 723, 309, 365, 278, 512, 658, 233, 393, 875

A question

You design a streaming algorithm A to solve some problem. The stream of data arrives. A outputs its answer:

21.5

A question 🪡

Oh, no! That wasn't the end. More data arrives.

A question 🪡

Oh, no! That wasn't the end. More data arrives.

834, 992, 528, 12, 181, 274, 159, 150, 716, 71, 755, 4, 324, 398, 802, 176, 302, 941, 678, 934, 546, 753, 812, 47, 755, 721, 893, 53, 410

A question 🪡

Oh, no! That wasn't the end. More data arrives. A outputs its answer on this:

18.1

A question 🪡

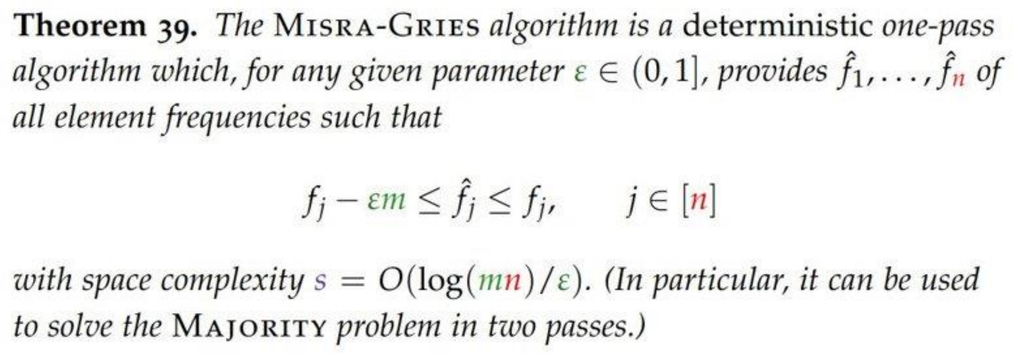
How do you combine 21.5 and 18.1 to get the answer on the whole data stream?

Sketching

Even better: linear sketching

Frequent Elements (Heavy Hitters)

Remember Misra-Gries?



Theorem 39. *The* MISRA-GRIES *algorithm is a* deterministic *one-pass algorithm which, for any given parameter $\varepsilon \in (0,1]$, provides $\hat{f}_1, \dots, \hat{f}_n$ of all element frequencies such that*

$$f_j - \varepsilon m \le \hat{f}_j \le f_j, \qquad j \in [n]$$

with space complexity $s = O(\log(mn)/\varepsilon)$. (In particular, it can be used to solve the MAJORITY *problem in two passes.)*

Frequent Elements (Heavy Hitters): ℓ_1, ℓ_2, etc.

Frequent Elements (Heavy Hitters): CountSketch (1/4)

```
Input: Parameters $\varepsilon, \delta \in (0,1]$
1: Set $k \leftarrow O(1/\varepsilon^2)$, and initialize an array $C$ of size $k$ to zero
2: Pick $h\colon [n] \to [k]$ from a strongly universal hashing family
3: Pick $g\colon [n] \to \{-1,1\}$ from a strongly universal hashing family
4: for all $1 \leq i \leq m$ do
5:     Get item $a_i = (j,c) \in [n] \times \{-B,\dots,B\}$    ▷ Assume $B = O(1)$
6:     $C[h(j)] \leftarrow C[h(j)] + c \cdot g(j)$
Output: On query $j \in [n]$, return $\widehat{f}_j \leftarrow g(j) \cdot C[h(j)]$
```

Frequent Elements (Heavy Hitters): CountSketch (2/4)

```
Input: Parameters $\varepsilon, \delta \in (0,1]$
1: Set $k \leftarrow O(1/\varepsilon^2)$, and initialize an array $C$ of size $k$ to zero
2: Pick $h\colon [n] \to [k]$ from a strongly universal hashing family
3: Pick $g\colon [n] \to \{-1,1\}$ from a strongly universal hashing family
4: for all $1 \leq i \leq m$ do
5:     Get item $a_i = (j,c) \in [n] \times \{-B,\dots,B\}$    ▷ Assume $B = O(1)$
6:     $C[h(j)] \leftarrow C[h(j)] + c \cdot g(j)$
Output: On query $j \in [n]$, return $\widehat{f}_j \leftarrow g(j) \cdot C[h(j)]$
```

Frequent Elements (Heavy Hitters): CountSketch (3/4)

```
Input: Parameters $\varepsilon, \delta \in (0,1]$
1: Set $k \leftarrow O(1/\varepsilon^2)$, and initialize an array $C$ of size $k$ to zero
2: Pick $h\colon [n] \to [k]$ from a strongly universal hashing family
3: Pick $g\colon [n] \to \{-1,1\}$ from a strongly universal hashing family
4: for all $1 \le i \le m$ do
5:     Get item $a_i = (j,c) \in [n] \times \{-B,\dots,B\}$    ▷ Assume $B = O(1)$
6:     $C[h(j)] \leftarrow C[h(j)] + c \cdot g(j)$
Output: On query $j \in [n]$, return $\widehat{f}_j \leftarrow g(j) \cdot C[h(j)]$
```

Frequent Elements (Heavy Hitters): CountSketch (4/4)

Theorem 44. *The (median trick version of the)* CountSketch *algorithm is a* randomised *one-pass sketching algorithm which, for any given parameters* $\varepsilon, \delta \in (0,1]$, *provides a (succinctly represented) estimate* $\widehat{f}$ *of frequency vector* f *of the stream such that, for every* $j \in [n]$

$$\Pr\left[\left| \widehat{f}_j - f_j \right| \leq \varepsilon \|f_{-j}\|_2 \right] \geq 1 - \delta$$

with space complexity

$$s = O\left(\frac{\log(nm)}{\varepsilon^2} \log \frac{1}{\delta} \right).$$

Frequent Elements (Heavy Hitters): CountMinSketch (1/4)

```
Input: Parameters $\varepsilon, \delta \in (0,1]$
1: Set $k \leftarrow O(1/\varepsilon)$ and $T \leftarrow O(\log(1/\delta))$, and initialize a two-
   dimensional array $C$ of size $T \times k$ to zero
2: Pick $h_1, \ldots, h_T : [n] \rightarrow [k]$ independently from a strongly uni-
   versal hashing family
3: for all $1 \leq i \leq m$ do
4:     Get item $a_i = (j, c) \in [n] \times \{0, \ldots, B\}$    ▷ Assume $B = O(1)$
5:     for all $1 \leq t \leq T$ do
6:         $C[t][h_t(j)] \leftarrow C[t][h_t(j)] + c$
Output: On query $j \in [n]$, return $\widehat{f}_j \leftarrow \min_{1 \leq t \leq T} C[t][h_t(j)]$
```

Frequent Elements (Heavy Hitters): CountMinSketch (2/4)

```
Input: Parameters $\varepsilon, \delta \in (0,1]$
1: Set $k \leftarrow O(1/\varepsilon)$ and $T \leftarrow O(\log(1/\delta))$, and initialize a two-dimensional array $C$ of size $T \times k$ to zero
2: Pick $h_1, \ldots, h_T : [n] \to [k]$ independently from a strongly universal hashing family
3: for all $1 \leq i \leq m$ do
4:     Get item $a_i = (j, c) \in [n] \times \{0, \ldots, B\}$    ▷ Assume $B = O(1)$
5:     for all $1 \leq t \leq T$ do
6:         $C[t][h_t(j)] \leftarrow C[t][h_t(j)] + c$
Output: On query $j \in [n]$, return $\widehat{f}_j \leftarrow \min_{1 \leq t \leq T} C[t][h_t(j)]$
```

Frequent Elements (Heavy Hitters): CountMinSketch (3/4)

```
Input: Parameters $\varepsilon, \delta \in (0,1]$
1: Set $k \leftarrow O(1/\varepsilon)$ and $T \leftarrow O(\log(1/\delta))$, and initialize a two-dimensional array $C$ of size $T \times k$ to zero
2: Pick $h_1, \ldots, h_T : [n] \rightarrow [k]$ independently from a strongly universal hashing family
3: for all $1 \le i \le m$ do
4:     Get item $a_i = (j, c) \in [n] \times \{0, \ldots, B\}$    ▷ Assume $B = O(1)$
5:     for all $1 \le t \le T$ do
6:         $C[t][h_t(j)] \leftarrow C[t][h_t(j)] + c$
Output: On query $j \in [n]$, return $\widehat{f}_j \leftarrow \min_{1 \le t \le T} C[t][h_t(j)]$
```

Frequent Elements (Heavy Hitters): CountMinSketch (4/4)

Theorem 45. *The* CountMinSketch *algorithm is a* randomised *one-pass sketching algorithm which, for any given parameters* $\varepsilon, \delta \in (0,1]$, *provides a (succinctly represented) estimate* $\widehat{f}$ *of frequency vector* f *of the stream such that, for every* $j \in [n]$

$$\Pr\left[\left|\widehat{f}_j - f_j\right| \le \varepsilon \|f_{-j}\|_1\right] \ge 1 - \delta$$

with space complexity

$$s = O\left(\frac{\log(nm)}{\varepsilon} \log \frac{1}{\delta}\right).$$

(Moreover, $\widehat{f}_j$ *is always an overestimate:* $\widehat{f}_j \ge f_j$ *for all* $j \in [n]$.*)*

Wait a minute...

This seems strictly worse than Misra-Gries!

- Randomised instead of deterministic!
- Uses more space!
- Also in the cash register model!
- Also an ℓ_1 guarantee!

Wait a minute…

This seems strictly worse than Misra-Gries!

- Randomised instead of deterministic!
- Uses more space!
- Also in the cash register model!
- Also an ℓ_1 guarantee!

Yes, but:

- Linear sketch!
- Much faster per time step!
- Can be extended to the strict turnstile model!

Recap