

# Finding duplicates in a data stream

Jaikumar Radhakrishnan

School of Technology and Computer Science  
Tata Institute of Fundamental Research  
Mumbai

joint work with

**Parikshit Gopalan**  
University of Washington, Seattle  
& Microsoft Research, SVC

16 September 2008, Dagstuhl

# The problem: FindDuplicate

Alphabet:  $[m]$ .

The input: A stream of symbols  $X = \langle x_1, x_2, \dots, x_n \rangle$ , where  $n \geq m + 1$ .

The goal: Find some  $a \in [m]$  that occurs at least twice in the stream.

# The problem: FindDuplicate

Alphabet:  $[m]$ .

The input: A stream of symbols  $X = \langle x_1, x_2, \dots, x_n \rangle$ , where  $n \geq m + 1$ .

The goal: Find some  $a \in [m]$  that occurs at least twice in the stream.

# The problem: FindDuplicate

**Alphabet:**  $[m]$ .

**The input:** A stream of symbols  $X = \langle x_1, x_2, \dots, x_n \rangle$ , where  $n \geq m + 1$ .

**The goal:** Find some  $a \in [m]$  that occurs at least twice in the stream.

**Muthukrishnan '05:** The problem illustrates the tradeoff between random and sequential computation. It can be solved with random access in  $O(\log m)$  space.

*Can we find the duplicate after a few passes over the input?*

**Tarui '07:** A  $k$ -pass deterministic streaming algorithm requires space  $m^{\Omega(\frac{1}{2k-1})}$ .

In particular, every deterministic single pass algorithm requires space  $\Omega(m)$ .

**Muthukrishnan '05:** The problem illustrates the tradeoff between random and sequential computation. It can be solved with random access in  $O(\log m)$  space.

*Can we find the duplicate after a few passes over the input?*

**Tarui '07:** A  $k$ -pass deterministic streaming algorithm requires space  $m^{\Omega(\frac{1}{2k-1})}$ .

In particular, every deterministic single pass algorithm requires space  $\Omega(m)$ .

## Theorem

*There is a randomized algorithm that solves FindDuplicate using  $O((\log m)^4)$  space with high probability.*

## Stream of increments

Alphabet:  $[m]$ .

Stream:  $I = \langle (a_1, c_1), (a_2, c_2), \dots, (a_n, c_n) \rangle$ , where each  $a_j \in [m]$  and  $c_j$  is an integer, positive or negative.

Frequency vector:  $\langle f[1], f[2], \dots, f[m] \rangle$  where  $f[a] = \sum_{j:a_j=a} c_j$ .

# From FindDuplicate to FindPositive

Given the data stream

$$X = \langle x_1, x_2, \dots, x_n \rangle,$$

consider the stream of increments

$$I = \langle (1, -1), (2, -1), \dots, (m, -1), (x_1, 1), (x_2, 1), \dots, (x_n, 1) \rangle.$$

Note that if  $n > m$ , then for the frequency vector  $f$  associated with this stream, we have

$$\sum_{a \in [m]} f[a] > 0. \quad (1)$$

## FindPositive

Given a stream of increments whose frequency vector  $f$  satisfies (1), determine an  $a$  such that  $f[a] > 0$ .

# From FindDuplicate to FindPositive

Given the data stream

$$X = \langle x_1, x_2, \dots, x_n \rangle,$$

consider the stream of increments

$$I = \langle (1, -1), (2, -1), \dots, (m, -1), (x_1, 1), (x_2, 1), \dots, (x_n, 1) \rangle.$$

Note that if  $n > m$ , then for the frequency vector  $f$  associated with this stream, we have

$$\sum_{a \in [m]} f[a] > 0. \quad (1)$$

## FindPositive

Given a stream of increments whose frequency vector  $f$  satisfies (1), determine an  $a$  such that  $f[a] > 0$ .

# FindPositive in three easy steps

**Isolate:** Pick a subset  $S \subseteq [m]$  with a dictator: we say that  $a \in S$  is a dictator if

$$f[a] > \sum_{b \in S \setminus \{a\}} |f[b]|.$$

**Find:** Guess who the dictator is.

**Verify:** Verify that the guess is correct.

# FindPositive in three easy steps

**Isolate:** Pick a subset  $S \subseteq [m]$  with a dictator: we say that  $a \in S$  is a dictator if

$$f[a] > \sum_{b \in S \setminus \{a\}} |f[b]|.$$

**Find:** Guess who the dictator is.

**Verify:** Verify that the guess is correct.

# FindPositive in three easy steps

**Isolate:** Pick a subset  $S \subseteq [m]$  with a dictator: we say that  $a \in S$  is a dictator if

$$f[a] > \sum_{b \in S \setminus \{a\}} |f[b]|.$$

**Find:** Guess who the dictator is.

**Verify:** Verify that the guess is correct.

# What can one do with a frequency vector?

- A frequency vector defines a 'halfspace' function on  $h : \{+1, -1\}^m \rightarrow \{+1, -1\}^m$  defined by

$$w \mapsto \text{sgn}(w \cdot f).$$

- For a subset  $S \subseteq [m]$ , we have  $h_S : \{+1, -1\}^S \rightarrow \{+1, -1\}$  defined by

$$w \mapsto \text{sgn}(w \cdot f|_S).$$

## The connection

If  $a \in S$  is a dictator, then  $h_S(w) = w[a]$ ; if  $f[a] \leq 0$ , then  $\Pr_w[h_S(w) \neq w[a]] \geq \frac{1}{2}$ .

## Computing dot products is easy

If the entries of  $w$  and membership in  $S$  can be tested efficiently, then  $h_S(w)$  can be computed efficiently.

# What can one do with a frequency vector?

- A frequency vector defines a ‘halfspace’ function on  $h : \{+1, -1\}^m \rightarrow \{+1, -1\}^m$  defined by

$$w \mapsto \text{sgn}(w \cdot f).$$

- For a subset  $S \subseteq [m]$ , we have  $h_S : \{+1, -1\}^S \rightarrow \{+1, -1\}$  defined by

$$w \mapsto \text{sgn}(w \cdot f|_S).$$

## The connection

If  $a \in S$  is a dictator, then  $h_S(w) = w[a]$ ; if  $f[a] \leq 0$ , then  $\Pr_w[h_S(w) \neq w[a]] \geq \frac{1}{2}$ .

## Computing dot products is easy

If the entries of  $w$  and membership in  $S$  can be tested efficiently, then  $h_S(w)$  can be computed efficiently.

# What can one do with a frequency vector?

- A frequency vector defines a ‘halfspace’ function on  $h : \{+1, -1\}^m \rightarrow \{+1, -1\}^m$  defined by

$$w \mapsto \text{sgn}(w \cdot f).$$

- For a subset  $S \subseteq [m]$ , we have  $h_S : \{+1, -1\}^S \rightarrow \{+1, -1\}$  defined by

$$w \mapsto \text{sgn}(w \cdot f|_S).$$

## The connection

If  $a \in S$  is a dictator, then  $h_S(w) = w[a]$ ; if  $f[a] \leq 0$ , then  $\Pr_w[h_S(w) \neq w[a]] \geq \frac{1}{2}$ .

## Computing dot products is easy

If the entries of  $w$  and membership in  $S$  can be tested efficiently, then  $h_S(w)$  can be computed efficiently.

# Isolation: easy cases

## Example

Suppose the frequency vector has only one positive component. Then,  $[m]$  itself is a dictatorship.

## Example

Suppose  $\ell^+(f) = n > \ell^-(f)$ . Suppose  $f[a] = s$  for  $\frac{n}{s}$  values of  $a$ , and zero or negative for the rest.

A random set of size  $s/2$  is a dictatorship with constant probability.

# Isolation: easy cases

## Example

Suppose the frequency vector has only one positive component. Then,  $[m]$  itself is a dictatorship.

$$\begin{aligned}\ell^+(f) &= \sum_{a:f[a]>0} f[a]; \\ \ell^-(f) &= \sum_{a:f[a]<0} |f[a]|.\end{aligned}$$

Note that  $\ell^+(f) > \ell^-(f)$ .

## Example

Suppose  $\ell^+(f) = n > \ell^-(f)$ . Suppose  $f[a] = s$  for  $\frac{n}{s}$  values of  $a$ , and zero or negative for the rest.

A random set of size  $s/2$  is a dictatorship with constant probability.

# Isolation: easy cases

## Example

Suppose the frequency vector has only one positive component. Then,  $[m]$  itself is a dictatorship.

$$\begin{aligned}\ell^+(f) &= \sum_{a:f[a]>0} f[a]; \\ \ell^-(f) &= \sum_{a:f[a]<0} |f[a]|.\end{aligned}$$

Note that  $\ell^+(f) > \ell^-(f)$ .

## Example

Suppose  $\ell^+(f) = n > \ell^-(f)$ . Suppose  $f[a] = s$  for  $\frac{n}{s}$  values of  $a$ , and zero or negative for the rest.

A random set of size  $s/2$  is a dictatorship with constant probability.

# Isolation: generating a random dictatorship

## Fact

There is always an  $s$  (a power of 2) such that  $f[a] \geq s$  for  $\sim \ell^+(f)/(s \log m)$  values  $a \in [m]$ .

## Randomly generating the dictatorship

- Let  $s$  be random power of two in the range  $\{1, 2, \dots, \lceil \log \ell^+(f) \rceil\}$ .
- Let  $S = \{a_1, a_2, \dots, a_s\}$ , where each  $a_i$  is uniformly distributed in  $[m]$  and the  $a_i$ 's are pairwise independent.

## Lemma

*With probability  $\Omega((\log m)^{-2})$  the set  $S$  is a dictatorship.*

# Isolation: generating a random dictatorship

## Fact

There is always an  $s$  (a power of 2) such that  $f[a] \geq s$  for  $\sim \ell^+(f)/(s \log m)$  values  $a \in [m]$ .

## Randomly generating the dictatorship

- Let  $s$  be random power of two in the range  $\{1, 2, \dots, \lceil \log \ell^+(f) \rceil\}$ .
- Let  $S = \{a_1, a_2, \dots, a_s\}$ , where each  $a_i$  is uniformly distributed in  $[m]$  and the  $a_i$ 's are pairwise independent.

## Lemma

*With probability  $\Omega((\log m)^{-2})$  the set  $S$  is a dictatorship.*

# Isolation: generating a random dictatorship

## Fact

There is always an  $s$  (a power of 2) such that  $f[a] \geq s$  for  $\sim \ell^+(f)/(s \log m)$  values  $a \in [m]$ .

## Randomly generating the dictatorship

- Let  $s$  be random power of two in the range  $\{1, 2, \dots, \lceil \log \ell^+(f) \rceil\}$ .
- Let  $S = \{a_1, a_2, \dots, a_s\}$ , where each  $a_i$  is uniformly distributed in  $[m]$  and the  $a_i$ 's are pairwise independent.

## Lemma

*With probability  $\Omega((\log m)^{-2})$  the set  $S$  is a dictatorship.*

# Isolation: generating a random dictatorship

## Fact

There is always an  $s$  (a power of 2) such that  $f[a] \geq s$  for  $\sim \ell^+(f)/(s \log m)$  values  $a \in [m]$ .

## Randomly generating the dictatorship

- Let  $s$  be random power of two in the range  $\{1, 2, \dots, \lceil \log \ell^+(f) \rceil\}$ .
- Let  $S = \{a_1, a_2, \dots, a_s\}$ , where each  $a_i$  is uniformly distributed in  $[m]$  and the  $a_i$ 's are pairwise independent.

## Lemma

*With probability  $\Omega((\log m)^{-2})$  the set  $S$  is a dictatorship.*

**Remark:** It takes only  $O(\log m)$  bits to represent such a set.

# Locating the dictator

**Assumption:** The set  $S$  has a dictator  $d$ .

**Binary search:** Compute the dot product of  $f|_S$  with  $\lceil \log m \rceil$  coordinate vectors  $w_1, w_2, \dots$ , defined by

$$w_j[a] = a[j],$$

where we think of  $a$  as a  $\{+1, -1\}$ -vector of length  $\lceil \log m \rceil$ . The sign of  $w_j \cdot f$  gives the  $j$  bit of the dictator.

## Lemma

*If  $d$  is the dictator in  $S$ , then the above procedure determines  $d$  correctly. But if  $S$  has no dictator, . . .*

# Locating the dictator

**Assumption:** The set  $S$  has a dictator  $d$ .

**Binary search:** Compute the dot product of  $f|_S$  with  $\lceil \log m \rceil$  coordinate vectors  $w_1, w_2, \dots$ , defined by

$$w_j[a] = a[j],$$

where we think of  $a$  as a  $\{+1, -1\}$ -vector of length  $\lceil \log m \rceil$ . The sign of  $w_j \cdot f$  gives the  $j$  bit of the dictator.

## Lemma

*If  $d$  is the dictator in  $S$ , then the above procedure determines  $d$  correctly. But if  $S$  has no dictator, ...*

# Locating the dictator

**Assumption:** The set  $S$  has a dictator  $d$ .

**Binary search:** Compute the dot product of  $f|_S$  with  $\lceil \log m \rceil$  coordinate vectors  $w_1, w_2, \dots$ , defined by

$$w_j[a] = a[j],$$

where we think of  $a$  as a  $\{+1, -1\}$ -vector of length  $\lceil \log m \rceil$ . The sign of  $w_j \cdot f$  gives the  $j$  bit of the dictator.

## Lemma

*If  $d$  is the dictator in  $S$ , then the above procedure determines  $d$  correctly. But if  $S$  has no dictator, ...*

**Given:** A set  $S$  and  $a \in S$ .

**Goal:** If  $f[a] \leq 0$ , then don't accept it as a candidate positive element.

**Idea:** Pick a random sequence  $w \in \{+1, -1\}^S$  and verify that  $w[a] = \text{sgn}(w \cdot f|_S)$ .

## Lemma

*If  $f[a]$  is not positive, the above procedure rejects  $a$  with probability at least  $\frac{1}{2}$ .*

# Space requirement

**Isolation:** We pick  $\ell = \theta((\log m)^2)$  sets to ensure that with constant probability at least one of them is a dictatorship.

Space:  $O((\log m)^3)$ .

**Find:** Need to maintain  $O(\log m)$  bits per set for each running sum for each coordinate vector.

Space:  $\ell \times O(\log m) \times O(\log m) = O((\log m)^4)$ .

**Verify:** Using ideas from [Indyk 06], we can use Nisan's generator [Nisan 92] to generate all the randomness required in this step in  $O((\log m)^2)$  space. For each set we maintain  $O(\log m)$  bits to compute the dot product with each of the  $O(\log m)$  vectors  $w$ .

Space:  $\ell \times O(\log m) \times O(\log m) = O((\log m)^4)$ .

# Space requirement

**Isolation:** We pick  $\ell = \theta((\log m)^2)$  sets to ensure that with constant probability at least one of them is a dictatorship.

Space:  $O((\log m)^3)$ .

**Find:** Need to maintain  $O(\log m)$  bits per set for each running sum for each coordinate vector.

Space:  $\ell \times O(\log m) \times O(\log m) = O((\log m)^4)$ .

**Verify:** Using ideas from [Indyk 06], we can use Nisan's generator [Nisan 92] to generate all the randomness required in this step in  $O((\log m)^2)$  space. For each set we maintain  $O(\log m)$  bits to compute the dot product with each of the  $O(\log m)$  vectors  $w$ .

Space:  $\ell \times O(\log m) \times O(\log m) = O((\log m)^4)$ .

# Space requirement

**Isolation:** We pick  $\ell = \theta((\log m)^2)$  sets to ensure that with constant probability at least one of them is a dictatorship.

Space:  $O((\log m)^3)$ .

**Find:** Need to maintain  $O(\log m)$  bits per set for each running sum for each coordinate vector.

Space:  $\ell \times O(\log m) \times O(\log m) = O((\log m)^4)$ .

**Verify:** Using ideas from [Indyk 06], we can use Nisan's generator [Nisan 92] to generate all the randomness required in this step in  $O((\log m)^2)$  space. For each set we maintain  $O(\log m)$  bits to compute the dot product with each of the  $O(\log m)$  vectors  $w$ .

Space:  $\ell \times O(\log m) \times O(\log m) = O((\log m)^4)$ .

# Space requirement

**Isolation:** We pick  $\ell = \theta((\log m)^2)$  sets to ensure that with constant probability at least one of them is a dictatorship.

Space:  $O((\log m)^3)$ .

**Find:** Need to maintain  $O(\log m)$  bits per set for each running sum for each coordinate vector.

Space:  $\ell \times O(\log m) \times O(\log m) = O((\log m)^4)$ .

**Verify:** Using ideas from [Indyk 06], we can use Nisan's generator [Nisan 92] to generate all the randomness required in this step in  $O((\log m)^2)$  space. For each set we maintain  $O(\log m)$  bits to compute the dot product with each of the  $O(\log m)$  vectors  $w$ .

Space:  $\ell \times O(\log m) \times O(\log m) = O((\log m)^4)$ .

- Can also find positive elements when  $\ell_2^+(f) > \ell_2^-(f)$ ? Slightly more work.
- For  $p > 2$ , no efficient algorithm exists for the assumption  $\ell_p^+(f) > \ell_p^-(f)$ ? (Reduction from lower bounds for multi-party set disjointness problem [Chakrabarti, Khot, Sun '03].)

# Open questions

- 1 We conjecture that FindDuplicate can be solved in a single pass using  $O(\log m)$  space? (It is possible to improve our the isolation lemma slightly and obtain an algorithm that uses  $(O(\log m))^3$  space.)
- 2 In the Verify step, we eliminate those candidates coordinates whose influence on the halfspace function is negative. This can be done without using Nisan's generator, for our version of the problem. However, for the general problem of estimating the influence of each variable, no pseudorandom generator is known.
- 3 Jun Tarui asks if the deterministic problem is hard (for multiple passes) even if the input stream has length  $n = 2m$ .

Thank you!