Gems of TCS

Streaming Algorithms

Sasha Golovnev

February 4, 2021

Streaming Algorithms

- Massively long stream of data

Streaming Algorithms

- Massively long stream of data
  Instagram, search queries, network packets

Streaming Algorithms

- Massively long stream of data
  Instagram, search queries, network packets

  $x_1, x_2, x_3, \ldots, x_n$

Streaming Algorithms

- Massively long stream of data
  Instagram, search queries, network packets
  $x_1, x_2, x_3, \ldots, x_n$
- Data has grown: we can't afford even stoing it

Streaming Algorithms

- Massively long stream of data
  Instagram, search queries, network packets
  $x_1, x_2, x_3, \ldots, x_n$
- Data has grown: we can't afford even stoing it
- n inputs, space $\sqrt{n}$; $\log^{10} n$; $\log n$

Streaming Algorithms

- Massively long stream of data
  Instagram, search queries, network packets
  $x_1, x_2, x_3, \ldots, x_n$
- Data has grown: we can't afford even stoing it
- n inputs, space $\sqrt{n}$; $\log^{10} n$; $\log n$
- Efficient processing of stream

Streaming Algorithms

- Massively long stream of data
  Instagram, search queries, network packets
  $x_1, x_2, x_3, \ldots, x_n$
- Data has grown: we can't afford even stoing it
- n inputs, space $\sqrt{n}$; $\log^{10} n$; $\log n$
- Efficient processing of stream
- Mostly randomized algorithms

Missing Number

Missing Number

- Stream contains n distinct numbers in range $\{0, \ldots, n\}$ all but one

Missing Number

- Stream contains n distinct numbers in range $\{0, \ldots, n\}$
- Return the only missing number

Could sort – linear space,
go through stream
many times

Missing Number

- Stream contains n distinct numbers in range $\{0, \ldots, n\}$
- Return the only missing number
- Efficient algorithm?

Streaming Algorithm

- Compute sum of all elements in stream:

$$S = x_1 + \ldots x_n$$

Streaming Algorithm

- Compute sum of all elements in stream:

$$s = x_1 + \dots x_n$$

- Sum of all numbers in range $\{0, \dots, n\}$ is $S = \frac{n(n+1)}{2}$

Streaming Algorithm

- Compute sum of all elements in stream:

0 2 1 7 4 5 6

$$s = x_1 + \ldots x_n$$

- Sum of all numbers in range $\{0, \ldots, n\}$ is $S = \frac{n(n+1)}{2}$

0 + 1 + 2 + ... + 6 − (0 + 2 + 1 + 7 + 4 + 5 + 6)

- Missing number is $S - s = \frac{n(n+1)}{2} - s$

See element → process quickly – one addition

$O(\log n)$

Streaming Algorithm

- Compute sum of all elements in stream:

$$s = x_1 + \ldots x_n$$

- Sum of all numbers in range $\{0, \ldots, n\}$ is $S = \frac{n(n+1)}{2}$
- Missing number is $S - s = \frac{n(n+1)}{2} - s$
- One pass through stream, efficient processing, $O(\log n)$ space

Two Missing Elements

- Stream contains $n - 1$ distinct numbers in range $\{0, \ldots, n\}$

Two Missing Elements

- Stream contains $n - 1$ distinct numbers in range $\{0, \ldots, n\}$

- Return both missing numbers

Two Missing Elements

- Stream contains $n - 1$ distinct numbers in range $\{0, \ldots, n\}$

$s = x_1 + \ldots + x_{n-1}$

$S = 0 + 1 + \ldots + n$

- Return both missing numbers

$S - s = a + b$

- Efficient algorithm?

Don't want to sort

Streaming Algorithm

- Compute sum and sum of squares of all elements in stream:

sum of inputs

$$s = x_1 + \ldots x_{n-1}$$

$$t = x_1^2 + \ldots x_{n-1}^2$$

sum of their squares

Streaming Algorithm

- Compute sum and sum of squares of all elements in stream:

$$s = x_1 + \ldots x_{n-1}$$
$$t = x_1^2 + \ldots x_{n-1}^2$$

- Sum of all numbers in range $\{0, \ldots, n\}$ is $S = \frac{n(n+1)}{2}$
  Sum of squares of all numbers in range $\{0, \ldots, n\}$ is $T = \boxed{\frac{n(n+1)(2n+1)}{6}}$ $T = \sum_{i=0}^{n} i^2$

Streaming Algorithm

- If missing numbers are a and b, then

I know

we know

$$u = \underline{a + b} = \underline{S - s}$$

we know

$$v = \underline{a^2 + b^2} = \boxed{T - t}$$

want to find a, b

$$w = \frac{u^2 - v}{2} = \frac{(a+b)^2 - a^2 - b^2}{2} = \frac{2ab}{2} = ab$$

$u = a + b$ I know u, w

$w = a \cdot b$ want to find a, b

$a = u - b$

$w = (u-b)b$

$b^2 - \underline{u}b + \underline{w} = 0$

I know

Find b

$D = u^2 - 4w$

$$b = \frac{u \pm \sqrt{u^2 - 4w}}{2}$$

Two solutions are the two missing els

Streaming Algorithm

- If missing numbers are a and b, then

$S = x_1 + \dots + x_n$

$T = x_1^2 + \dots + x_n^2$

$$a + b = S - s$$

$$a^2 + b^2 = T - t$$

This can be generalized

k els are missing

$S_1 = x_1 + \dots + x_n$

$S_2 = x_1^2 + \dots + x_n^2$

$\dots$

$S_k = x_1^k + \dots + x_n^k$

- One pass through stream, efficient processing, $O(\log n)$ space

Majority Element

Majority Element

n- length of stream

- Stream has element occuring $> n/2$ times

Majority Element

- Stream has element occuring $> n/2$ times
- Find it!

Sort , Median

We can't afford storing input

Streaming Algorithm

candidate for Maj

- count $\leftarrow$ 0; m $\leftarrow \perp$ Null

Streaming Algorithm

- $\mathsf{count} \leftarrow 0; \quad \mathsf{m} \leftarrow \perp$
- For each element x_i of Stream:

Streaming Algorithm

- $\mathbf{count} \leftarrow 0; \quad \mathbf{m} \leftarrow \perp$

new candidate

- For each element x_i of Stream:
  - If $\mathbf{count} = 0$, then $\mathbf{m} \leftarrow x_i$ and count ← 1

Streaming Algorithm

- $\mathbf{count} \leftarrow 0$; $\mathbf{m} \leftarrow \perp$
- For each element x_i of Stream:
  - If $\mathbf{count} = 0$, then $\mathbf{m} \leftarrow x_i$ and count ← 1
  - ElseIf $x_i = \mathbf{m}$, then $\mathbf{count}\,\mathbf{++}$

Streaming Algorithm

- $\textbf{count} \leftarrow 0; \quad \textbf{m} \leftarrow \perp$

- For each element x_i of Stream:
  - If $\textbf{count} = 0$, then $\textbf{m} \leftarrow x_i$ and count $\leftarrow$ 1
  - ElseIf $x_i = \textbf{m}$, then $\textbf{count}$ ++
  - Else $\textbf{count}$ -- $x_i \neq m$

Streaming Algorithm

two variables
1 pass
processes input els:
efficiently

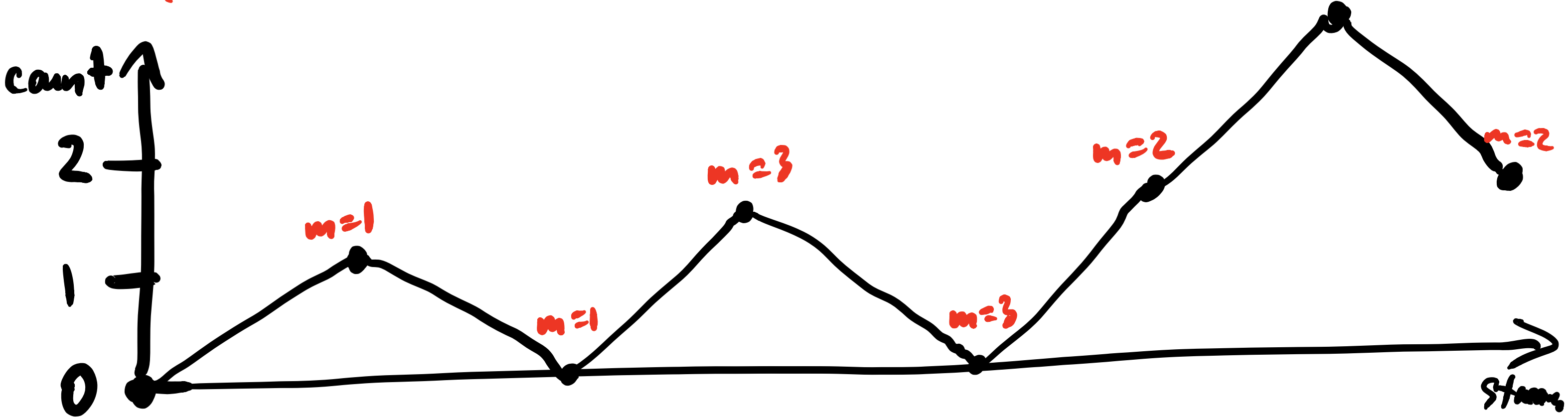
- count $\leftarrow 0$; m $\leftarrow \perp$
- For each element x_i of Stream:
  - If count $= 0$, then m $\leftarrow x_i$ and count $\leftarrow 1$
  - ElseIf $x_i =$ m, then count ++
  - Else count -- $x_i \neq m$
- Return m

EXAMPLE

$n = 7$ $\quad$ $Maj = 2$

x_i		1	2	3	2	2	2	1
m		1	1	3	3	2	2	$\boxed{2}$
count	0	1	0	1	0	1	2	1

Return 2

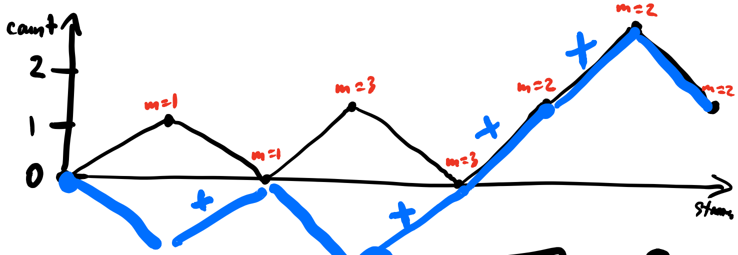


count ≥ 0

PROOF

$Maj = 2$

x_i: 1 2 3 2 2 2 1



variable count'
this is only for analysis
not for algorithm

$$\text{count}' = \begin{cases} \boxed{\text{count}} & \text{if } m = Maj \\ -\text{count} & \text{if } m \neq Maj \end{cases}$$

— true Maj elt

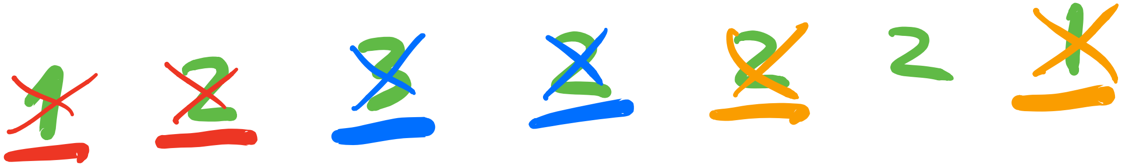
— When I see Maj, increment count'

Proof: if $m = Maj \Rightarrow$ count++ $\Rightarrow$ count'++
if $m \neq Maj \Rightarrow$ count-- $\Rightarrow$ count'++ □

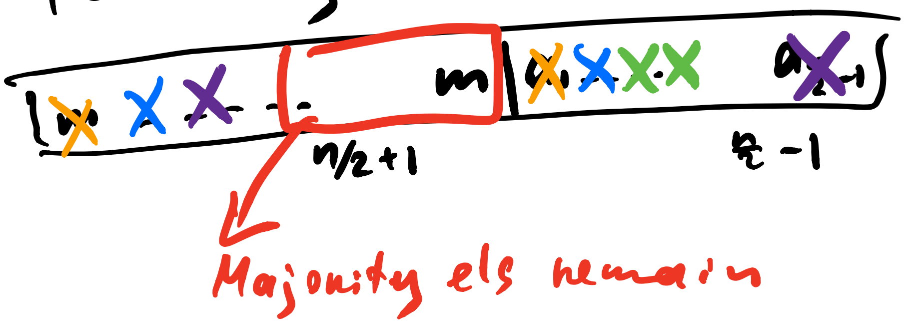
— See Maj $> \frac{n}{2}$ times $\Rightarrow$ count' is incremented $> \frac{n}{2}$
$\Rightarrow$ count' is decremented $< \frac{n}{2}$

In the end,
— count' $> 0 \Rightarrow$ count' = count $\Rightarrow m = Maj \Rightarrow$ output Maj

Another View



- Pairs up distinct els
- Kills all these pairs
- Remaining els are Maj



Misra-Gries Algorithm

Assume there is Maj in stream
($> \frac{n}{2}$ occurrences).

Find it

without this assumption, we'll make two passes through input str.

— candidate $\underline{\underline{m}}$
— count how many times m appears in stream

For $k \in \mathbb{N}$, k-Heavy Hitters: Find all els that appear $\geq \frac{n}{k}$ in the stream
($\leq k$ such els)

Maj = case $k=2$

Misra-Gries Algorithm

- $\text{count}_1, \ldots, \text{count}_k \leftarrow 0$; $\text{m}_1, \ldots, \text{m}_k \leftarrow \perp$
- For each element x_i of Stream:
  - If $x_i = \text{m}_j$, then count_j ++
  - Else
    - Let count_j be min in $\text{count}_1, \ldots \text{count}_k$
    - If $\text{count}_j = 0$, then $\text{m}_j = x_i$; $\text{count}_j = 1$
    - Else count_1 --, ..., count_k --
- Return $\text{m}_1, \ldots, \text{m}_k$ ← contain all els of stream that appear $\geq n/k$ times

Approximate Counting

- Router receives stream of network packages

- Router receives stream of network packages

- Want to count number of packages from IP “1.2.3.4”

~~1.1.1.1.~~ 1.2.3.4. ~~2.2.2.2.~~ 1.2.3.4

EQ: 1.2.3.4 1.2.3.4.

$\leq n$ of them in the stream

output length of the stream

- Router receives stream of network packages
- Want to count number of packages from IP "1.2.3.4"
- Efficient algorithm?

count = 0

See input: count++

In the end, count = stream length

$\log_2(n+1)$ bits

Can we use fewer than log n bits?

Input length $\leq n$

outputs $\in \{0, 1, 2, \ldots, n-1, n\}$

2 bits $\boxed{0 \mid 1}$

00
01
10
11

≤ 4 distinct answers

If $< \log_2 n$ bits $\Rightarrow$ different answers $< 2^{\log_2 n} = n$

$\lceil \log_2(n+1) \rceil$ bits is optimal

- Router receives stream of network packages
- Want to count number of packages from IP "1.2.3.4"
- Efficient algorithm?
- Efficient approximate algorithm?

$\log\log n$ ← exponentially better than previous sol

Overview

n Trivial alg stores n

n=147 | 1 0 0 1 0 0 1 1 | ← log n

what if instead of storing n in binary
I'm storing the length of "n in binary"

Instead of storing 147, I'd store number 8

| 1 0 0 0 |

- Instead of log n bits to write {1, ..., n}
  store loglog n bits to write {1, ..., log n}

If length = 4 Then

$8 \leq n \leq 15$

8 | 1 | 0 | 0 | 0 |

-

15 | 1 | 1 | 1 | 1 |

$2^{\text{length}-1} \leq n < 2^{\text{length}}$

Morris Algorithm

c — $\approx$ length of n in binary

$n \approx 2^c$

when should increment c?

I want to increment c after seeing $\boxed{2^c}$ new els

Now see new el

w.p. $\frac{1}{2^c}$ $c{+}{+}$

w.p. $\left(1-\frac{1}{2^c}\right)$ don't update c

After seeing 2^c els, I expect to increment once

Morris Algorithm

$n = 0$

$\boxed{2^c - 1 \approx n}$

- $c \leftarrow 0$

Morris Algorithm

- $c \leftarrow 0$
- When see next element:
  - with probability $\frac{1}{2^c}$ increment ~~x~~ c
  - with probability $1 - \frac{1}{2^c}$ do nothing

Morris Algorithm

$n \approx 2^c - 1$

- $c \leftarrow 0$
- When see next element:
  - with probability $\frac{1}{2^c}$ increment $\cancel{X}$ c
  - with probability $1 - \frac{1}{2^c}$ do nothing
- Return $2^c - 1$

Probability of Success

- Thm
  - $\boxed{\mathbb{E}[\text{output}] = n}$
- Markov's:
  $E[\text{output} > 5n] < 1/5$

1. Markov's ineq: $\Pr[\text{output} \notin [n-O(n), n+O(n)]] < 0.9$

2. Amplify prob. of success by repeating this alg several times:
   $\Pr[\text{output} \notin [\frac{n}{2}, 2n]] < 0.01$

Probability of Success

- $\mathbb{E}[\text{output}] = n$
- By Markov's, $\Pr[\text{output} \geq 2n] \leq 1/2$

Probability of Success

- $\mathbb{E}[\text{output}] = n$
- By Markov's, $\Pr[\text{output} \geq 2n] \leq 1/2$
- Similar inequalities show that $\Pr[\text{output} \in [n - O(n), n + O(n)] \geq 0.9$

Probability of Success

- $\mathbb{E}[\text{output}] = n$
- By Markov's, $\Pr[\text{output} \geq 2n] \leq 1/2$
- Similar inequalities show that $\Pr[\text{output} \in [n - O(n), n + O(n)]] \geq 0.9$
- Again, repeating Algorithm several times significantly amplifies probability of success

Summary

- One pass through stream may be sufficient

Summary

- One pass through stream may be sufficient
- Use Randomness and Approximation

Summary

- One pass through stream may be sufficient
- Use Randomness and Approximation
- Markov's ineqaulity: from Expectation to Probability

Summary

- One pass through stream may be sufficient
- Use Randomness and Approximation
- Markov's ineqaulity: from Expectation to Probability

Max-Cut

- Amplify probability by Repetitions