GEMS OF TCS

STREAMING ALGORITHMS

Sasha Golovnev
September 7, 2022
Fruit Game

Credit: Jelani Nelson

(https://www.youtube.com/watch?v=CorP4I23wOo&t=2434s)
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 storing 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 storing 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 storing 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 storing 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 \( \{0, \ldots, n\} \).
Missing Number

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

- Return the only missing number
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:
  \[ S = x_1 + \ldots + x_n \]

- Sum of all numbers in range \{0, \ldots, n\} is
  \[ S = \frac{n(n+1)}{2} \]
**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 \( S - s = \frac{n(n+1)}{2} - s \)
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 $\{0, \ldots, n\}$
- Return both missing numbers
- Efficient algorithm?
Streaming Algorithm

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

\[
S = x_1 + \ldots x_{n-1} \\
t = x_1^2 + \ldots x_{n-1}^2
\]
• 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

\[ T = \frac{n(n+1)(2n+1)}{6} \]
STREAMING ALGORITHM

• If missing numbers are $a$ and $b$, then

\[ a + b = S - s \]
\[ a^2 + b^2 = T - t \]
Streaming Algorithm

• If missing numbers are $a$ and $b$, then

\[
a + b = S - s
\]
\[
a^2 + b^2 = T - t
\]

• One pass through stream, efficient processing, $O(\log n)$ space
Majority Element
MAJORITY ELEMENT

• Stream has element occurring > \( n/2 \) times
MAJORITY ELEMENT

- Stream has element occurring > $n/2$ times
- Find it!
STREAMING ALGORITHM

- count ← 0; m ← ⊥
Streaming Algorithm

- count ← 0; m ←⊥

- For each element $x_i$ of Stream:
STREAMING ALGORITHM

• \( \text{count} \leftarrow 0; \ m \leftarrow \perp \)

• For each element \( x_i \) of Stream:
  • If \( \text{count} = 0 \), then \( m \leftarrow x_i \) and \( \text{count} \leftarrow 1 \)
Streaming Algorithm

1. $\text{count} \leftarrow 0$; $\text{m} \leftarrow \bot$

2. For each element $x_i$ of Stream:
   - If $\text{count} = 0$, then $\text{m} \leftarrow x_i$ and $\text{count} \leftarrow 1$
   - Else if $x_i = \text{m}$, then $\text{count} \leftarrow \text{count} + 1$
   - Else $\text{count} \leftarrow \text{count} - 1$

3. Return $\text{m}$
STREAMING ALGORITHM

- count ← 0; m ← \text{⊥}

- For each element $x_i$ of Stream:
  - If count = 0, then m ← $x_i$ and count ← 1
  - Elseif $x_i = m$, then count ++
  - Else count --
STREAMING ALGORITHM

- count ← 0; m ← ⊥

- For each element $x_i$ of Stream:
  - If count = 0, then $m ← x_i$ and count ← 1
  - Elself $x_i = m$, then count ++
  - Else count --

- Return m
Example
PROOF
ANOTHER VIEW
MISRA-GRIES ALGORITHM

1. \( \text{count} \rightarrow 0; \ m \rightarrow \perp \)

2. For each element \( x_i \) of Stream:
   - If \( x_i = m_j \), then \( \text{count}_j++ \)
   - Else
     - Let \( \text{count}_j \) be min in \( \text{count}_1, \ldots, \text{count}_k \)
       - If \( \text{count}_j = 0 \), then \( m_j = x_i; \text{count}_j = 1 \)
       - Else \( \text{count}_1--; \ldots; \text{count}_k--; \)

3. Return \( m_1, \ldots, m_k \)
Misra-Gries Algorithm

- \( \text{count}_1, \ldots, \text{count}_k \leftarrow 0; \ m_1, \ldots, m_k \leftarrow \bot \)

- For each element \( x_i \) of Stream:
  - If \( x_i = m_j \), then \( \text{count}_j \) ++
  - Else
    - Let \( \text{count}_j \) be min in \( \text{count}_1, \ldots, \text{count}_k \)
    - If \( \text{count}_j = 0 \), then \( m_j = x_i; \ \text{count}_j = 1 \)
    - Else \( \text{count}_1 --, \ldots, \text{count}_k -- \)

- Return \( m_1, \ldots, m_k \)
Approximate Counting
• Router receives stream of network packages

• Want to count number of packages from IP "1.2.3.4"

• Efficient algorithm?

• Efficient approximate algorithm?
• Router receives stream of network packages

• Want to count number of packages from IP "1.2.3.4"
• Router receives stream of network packages

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

• Efficient algorithm?
• Router receives stream of network packages

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

• Efficient algorithm?

• Efficient approximate algorithm?
OVERVIEW
MORRIS ALGORITHM

• \( c \leftarrow 0 \)
• When see next element:
  • with probability \( \frac{1}{2} \) increment \( c \)
  • with probability \( \frac{1}{2} \) do nothing
• Return \( 2^{c-1} \)
MORRIS ALGORITHM

1. $c \leftarrow 0$
MORRIS ALGORITHM

• $c \leftarrow 0$

• When see next element:
  • with probability $\frac{1}{2^c}$ increment $c$
  • with probability $1 - \frac{1}{2^c}$ do nothing
Morris Algorithm

- $c \leftarrow 0$
- When see next element:
  - with probability $\frac{1}{2^c}$ increment $c$
  - with probability $1 - \frac{1}{2^c}$ do nothing
- Return $2^c - 1$
ANALYSIS
• $\mathbb{E}[\text{output}] = n$
POBABILITY OF SUCCESS

• $\mathbb{E}[\text{output}] = n$

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

• By Markov’s, $\Pr[\text{output} \geq 2n] \leq 1/2$

• Similar inequalities show that $\Pr[\text{output} \in [n - O(n), n + O(n)] \geq 0.9$
• $\mathbb{E}[\text{output}] = n$

• 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 inequality: from Expectation to Probability
SUMMARY

• One pass through stream may be sufficient

• Use Randomness and Approximation

• Markov’s inequality: from Expectation to Probability

• Amplify probability by Repetitions