GEMS OF TCS

STREAMING ALGORITHMS

Sasha Golovnev
September 13, 2021
Fruit Game

Credit: Jelani Nelson

(https://www.youtube.com/watch?v=CorP4I23w0o&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:

\[ S = x_1 + \ldots x_{n-1} \]

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

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

• count ← 0; m ← ⊥

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

- count ← 0; m ← ⊥

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

- count ← 0; m ← ⊥

- 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 $\text{count} = 0$, then $m ← x_i$ and $\text{count} ← 1$
  - Elself $x_i = m$, then $\text{count} ++$
  - Else $\text{count} --$

- Return $m$
EXAMPLE
PROOF
ANOTHER VIEW
**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 \( \text{count}_j \) ++
  - Else
    - Let \( \text{count}_j \) 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 \( \text{count}_1 \) --, \( \ldots, \text{count}_k \) --

- Return \( \text{m}_1, \ldots, \text{m}_k \)
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”
• 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?
MORRIS ALGORITHM
MORRIS ALGORITHM

• $c \leftarrow 0$
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
PROBABILITY OF SUCCESS

• \( \mathbb{E}[\text{output}] = n \)
Probability of Success

- $\mathbb{E}[\text{output}] = n$
- By Markov’s, $\Pr[\text{output} \geq 2n] \leq 1/2$
**Probability of Success**

- $E[\text{output}] = n$

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

- 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 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