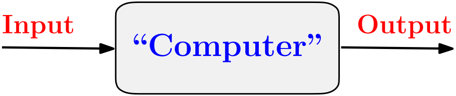
Streaming Algorithms

- Streaming Model of Computation
- Streaming Algorithms and DFA
- Stream: Motivation and Applications
- Synopsis: Sliding Window, Histogram, Wavelets
- Sampling from Stream: Reservoir Sampling
- Linear Sketch
- Count-Min Sketch
- AMS Sketch

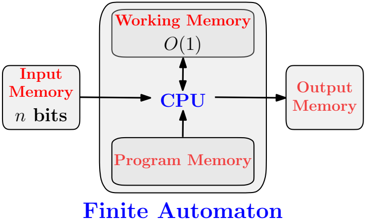
Imdad ullah Khan

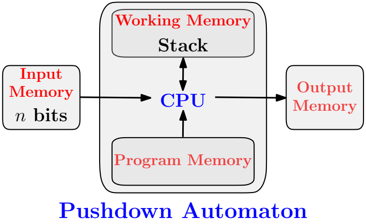
The Streaming Model of Computation

Models of Computation

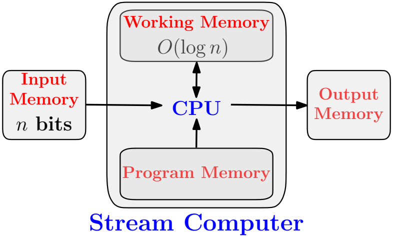


More detailed view of model of the "computers" we studied:





Stream Computation Model: Streaming Algorithm



A streaming algorithm has three components

Algorithm Streaming algorithm: Input $\mathcal{S} = \sigma_1\sigma_2\sigma_3\cdots \in \Sigma^*$

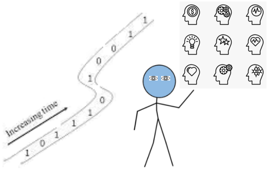
INITIALIZE(*vars*) ▷ $O\left((\log n)^c\right)$-bits

while σ : (next symbol in $\mathcal{S}$) is not end of stream **do**

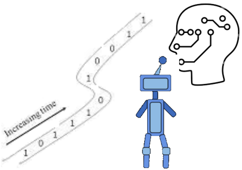
  Pseudocode using σ and *vars* ▷ $O(1)$-time

Pseudocode for **Accept/Reject** or **Output** based on *vars*

Differences in DFA and Streaming Algorithms



Finite Automaton



Stream Computer

Streaming algorithms have poly-logarithmic working memory

▷ memory increases with the size of inputs (though very slowly)

Streaming algorithms can recognize (some) non-regular language

Can output more than a bit e.g. can flush-out its working memory

▷ it can be used for non-decision problems

In some versions streaming algorithms can do multiple passes on input

Can be randomized - output is correct up to error parameters $0 < \epsilon, \delta < 1$

▷ working memory is polynomial in $1/\epsilon, 1/\delta, \log n$

Differences in DFA and Streaming Algorithms

BALANCED $= \{w | w$ has equal number of 0s and 1s$\}$ is not regular

▷ Proved it using the pumping lemma

No DFA recognizes BALANCED

Streaming Algorithm for BALANCED problem

Algorithm Streaming Algorithm for BALANCED($w = w_1 w_2 \dots w_n$)

$C_0 \leftarrow 0$
$C_1 \leftarrow 0$
for $i = 1 \rightarrow n$ **do**
  if $w_i = 1$ **then**
    $C_1 \leftarrow C_1 + 1$
  else
    $C_0 \leftarrow C_0 + 1$
if $C_0 = C_1$ **then**
  Accept
else
  Reject

This streaming algorithm recognizes BALANCED with $2 \log n$ bits

DFA vs Streaming Algorithms

$L_1 = \text{1-DOMINANT} = \{w | w \text{ has more 1s than 0s}\}$

Let $L_b = \text{BALANCED}$

Knowing L_b is not regular, doesn't imply L_1 is not regular $\triangleright$ $L_1 \neq \overline{L_b}$

$L_0 = \text{0-DOMINANT} = \{w | w \text{ has more 0s than 1s}\}$

$$L_b = \overline{L_1} \cap \overline{L_0} = \overline{L_1 \cup L_2}$$

$\triangleright$ L_1 is regular $\implies$ L_0 is regular (flip 0's and 1's in the supposed DFA)

$\rightarrow L_1 \cup L_0$ is regular $\rightarrow \overline{L_1 \cup L_2}$ is regular $\rightarrow \overline{L_1}$ is regular $\rightarrow L_1$ is regular

Thus, no DFA recognizes 1-DOMINANT, while the following streaming algorithm recognizes it with $2 \log n$ bits

Can we do it in fewer bits?

Streaming Algorithm for 1-DOMINANT problem

Algorithm Streaming Algorithm for 1-DOMINANT($w = w_1 w_2 \dots w_n$)

```
C ← 0
x ← 0                                    ▷ C is an integer, x is a bit
for i = 1 → n do
    if C = 0 then
        C ← 1
        x ← w_i
    else if C ≠ 0 AND x = w_i then
        C ← C + 1
    else
        C ← C − 1
if C > 0 AND x = 1 then
    Accept
else
    Reject
```

x records the bit currently in majority

C records the excess frequency of x over $\overline{x}$

PDA vs Streaming Algorithms

$\text{BALANCED-}3 = \{a^m b^m c^m | m \geq 0\}$ cannot be recognized by a PDA

▷ PDA cannot match number of a's, b's, and c's using one stack

Algorithm Streaming Algorithm for $\text{BALANCED-}3(w = w_1 w_2 \dots w_n)$

```
C_a ← 0   C_b ← 0   C_c ← 0   phase ← 0      ▷ phase: 0 for a's, 1 for b's, 2 for c's
for i = 1 → n do
    if phase = 0 then
        if w_i = a then   C_a ← C_a + 1
        else if w_i = b then   C_b ← C_b + 1   phase ← 2
        else   Reject
    else if phase = 1 then
        if w_i = b then   C_b ← C_b + 1
        else if w_i = c then   C_c ← C_c + 1   phase ← 3
        else   Reject
    else
        if w_i = c then   C_c ← C_c + 1
        else   Reject
if C_a = C_b = C_c then   Accept
else   Reject
```