Sublinear Algorithms

Lecture 6

Sofya Raskhodnikova

Penn State University

Thanks to Madhav Jha (Penn State) for help with creating these slides.

Communication Complexity

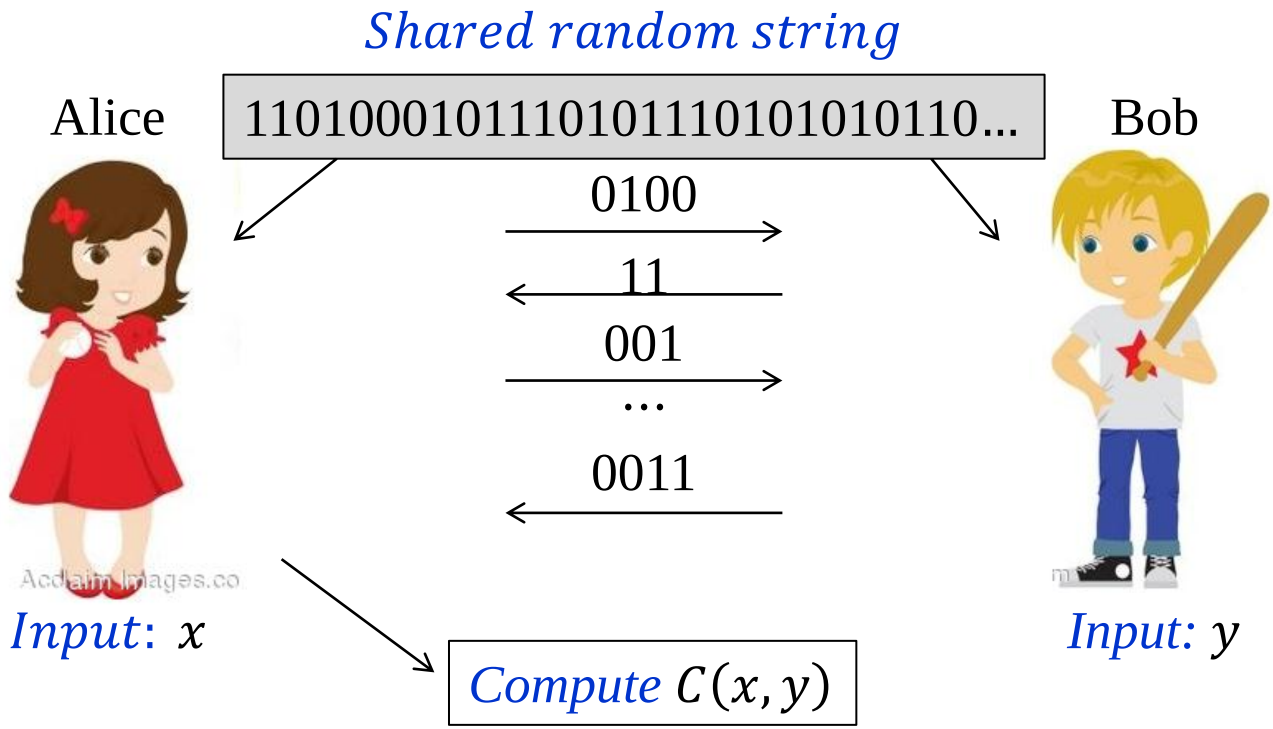
A Method for Proving Lower Bounds

[Blais Brody Matulef 11]

Use known lower bounds
for other models of computation

Partially based on slides by Eric Blais

(Randomized) Communication Complexity



Goal: minimize the number of bits exchanged.

- Communication complexity of a protocol is the maximum number of bits exchanged by the protocol.
- Communication complexity of a function C, denoted $R(C)$, is the communication complexity of the best protocol for computing C.

Example: Set Disjointness $DISJ_k$

Alice | 11010001011101011101010110... | Bob

Input: $S \subseteq [n]$, $|S| = k$.

Input: $T \subseteq [n]$, $|T| = k$

Compute $DISJ_k(S, T)$

$$= \begin{cases} \boldsymbol{accept} & \text{if } S \cap T = \emptyset \\ \boldsymbol{reject} & \text{otherwise} \end{cases}$$

Theorem [Hastad Wigderson 07]

$$R(\mathrm{DISJ}_k) \geq \Omega(k) \text{ for all } k \leq \frac{n}{2}.$$

A lower bound using CC method

Testing if a Boolean function is a k-parity

Linear Functions Over Finite Field $\mathbb{F}_2$

A Boolean function $f: \{0,1\}^n \to \{0,1\}$ is *linear* (also called *parity*) if
$f(x_1, \dots, x_n) = a_1x_1 + \cdots + a_nx_n$ for some $a_1, \dots, a_n \in \{0,1\}$

no free term

- Work in finite field $\mathbb{F}_2$
  - Other accepted notation for $\mathbb{F}_2$: GF_2 and $\mathbb{Z}_2$
  - Addition and multiplication is mod 2
  - $\boldsymbol{x}=(x_1, \dots, x_n)$, $\boldsymbol{y}=(y_1, \dots, y_n)$, that is, $\boldsymbol{x}, \boldsymbol{y} \in \{0,1\}^n$
    $\boldsymbol{x}+\boldsymbol{y}=(x_1+y_1, \dots, x_n+y_n)$

example

$$\begin{array}{r} 001001 \\ +\ 011001 \\ \hline 010000 \end{array}$$

Linear Functions Over Finite Field $\mathbb{F}_2$

A Boolean function $f: \{0,1\}^n \to \{0,1\}$ is *linear* (also called *parity*) if

$$f(x_1, \dots, x_n) = a_1 x_1 + \cdots + a_n x_n \text{ for some } a_1, \dots, a_n \in \{0,1\}$$

$$\Updownarrow$$

$[n]$ is a shorthand for $\{1, \dots n\}$

$$f(x_1, \dots, x_n) = \sum_{i \in S} x_i \textit{ for some } S \subseteq [n].$$

Notation: $\chi_S(x) = \sum_{i \in S} x_i$.

Testing if a Boolean function is Linear

Input: Boolean function $f: \{0,1\}^n \to \{0,1\}$

Question:

Is the function linear or ε-far from linear

($\geq \varepsilon 2^n$ values need to be changed to make it linear)?

Later in the course:

Famous BLR (Blum Lubi Rubinfeld 90) test runs in $O\left(\frac{1}{\varepsilon}\right)$ time

k-Parity Functions

k-Parity Functions

A function $f : \{0,1\}^n \to \{0,1\}$ is a k-parity if

$$f(x) = \chi_S(x) = \sum_{i \in S} x_i$$

for some set $S \subseteq [n]$ of size $|S| = k$.

Testing if a Boolean Function is a k-Parity

Input: Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ and an integer k

Question: Is the function a k-parity or ε-far from a k-parity

($\geq \varepsilon 2^n$ values need to be changed to make it a k-parity)?

Time:

$\mathrm{O}(k \log k)$ [Chakraborty Garcia−Soriano Matsliah]

$\Omega(\min(k, n-k))$ [Blais Brody Matulef 11]

- Today: $\Omega(k)$ for $k \leq n/2$

Today's bound implies $\Omega(\min(k, n-k))$

Important Fact About Linear Functions

Fact. Two different linear functions disagree on half of the values.

- Consider functions χ_S and χ_T where $S \neq T$.
  - Let i be an element on which S and T differ (w.l.o.g. $i \in S \setminus T$)
  - Pair up all n-bit strings: $(\boldsymbol{x}, \boldsymbol{x}^{(i)})$ where $\boldsymbol{x}^{(i)}$ is $\boldsymbol{x}$ with the i^{th} bit flipped.
  - For each such pair, $\chi_S(\boldsymbol{x}) \neq \chi_S(\boldsymbol{x}^{(i)})$

    but $\chi_T(\boldsymbol{x}) = \chi_T(\boldsymbol{x}^{(i)})$

    So, χ_S and χ_T differ on exactly one of $\boldsymbol{x}, \boldsymbol{x}^{(i)}$.
  - Since all $\boldsymbol{x}$'s are paired up,

    χ_S and χ_T differ on half of the values.

$$
\begin{array}{c|c|c|}
 & 0 & 0 \\
 & 1 & 1 \\
 & 1 & 0 \\
\boldsymbol{x} & a & b \\
 & 0 & 1 \\
 & \cdot & \cdot \\
 & \cdot & \cdot \\
 & \cdot & \cdot \\
\boldsymbol{x}^{(i)} & 1-a & b \\
 & 0 & 0 \\
 & 1 & 0 \\
 & 0 & 1 \\
\hline
 & \chi_S(x) & \chi_T(x)
\end{array}
$$

Corollary. A k'-parity function, where $k' \neq k$, is ½-far from any k-parity.

Reduction from $DISJ_{\boldsymbol{k/2}}$ to Testing k-Parity

- Let T be the best tester for the k-parity property for $\varepsilon = 1/2$
  - query complexity of T is $q(\text{testing } k\text{−parity})$.
- We will construct a communication protocol for $DISJ_{\boldsymbol{k/2}}$ that runs T and has communication complexity $2 \cdot q(\text{testing } k\text{−parity})$.

holds for CC of every protocol for $DISJ_{\boldsymbol{k}}$

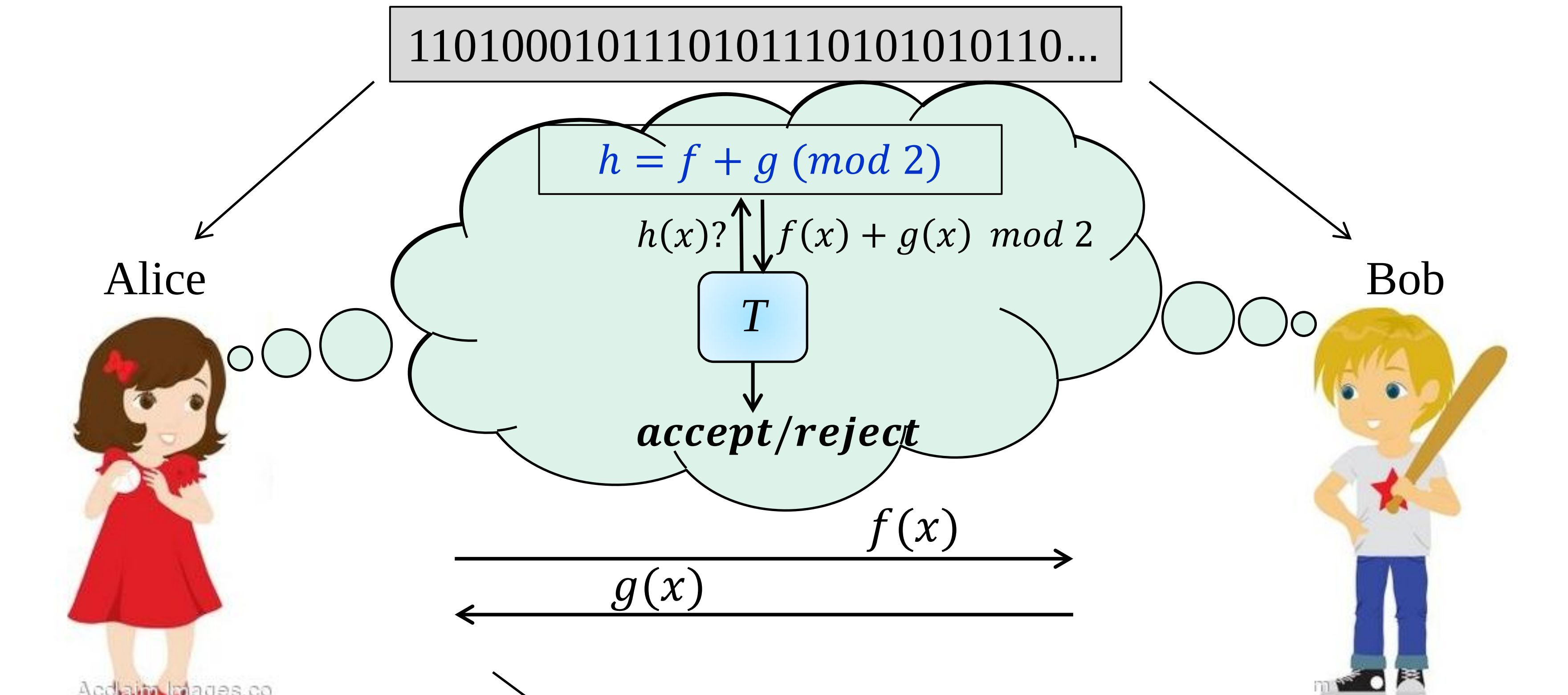
[Hastad Wigderson 07]

- Then $2 \cdot q(\text{testing } k\text{−parity}) \geq R\left(\mathrm{DISJ}_{k/2}\right) \geq \Omega(k/2)$ for $k \leq n/2$

$$\Downarrow$$

$$q(\text{testing } k\text{-parity}) \geq \Omega(k) \quad \text{for } k \leq n/2$$

Reduction from $DISJ_{k/2}$ to Testing k-Parity



Input: $S \subseteq [n]$, $|S| = k/2$.
Compute: $f = \chi_S$

Output T's answer

Input: $T \subseteq [n]$, $|T| = k/2$
Compute: $g = \chi_T$

- T receives its random bits from the shared random string.

Analysis of the Reduction

Queries: Alice and Bob exchange 2 bits for every bit queried by T

Correctness:

- $h = f + g \ (mod\ 2) = \chi_S + \chi_T \ (mod\ 2) = \chi_{S\Delta T}$
- $|S\Delta T| = |S| + |T| - 2|S \cap T|$
- $|S\Delta T| = \begin{cases} k & \text{if } S\cap T = \emptyset \\ \leq k-2 & \text{if } S\cap T \neq \emptyset \end{cases}$

$$h \text{ is} \begin{cases} k\text{−parity} & \text{if } S\cap T = \emptyset \\ k'\text{−parity where } k' \neq k & \text{if } S\cap T \neq \emptyset \end{cases}$$

1/2-far from every k-parity

Summary: q(testing k-parity) $\geq \Omega(k)$ for $k \leq n/2$

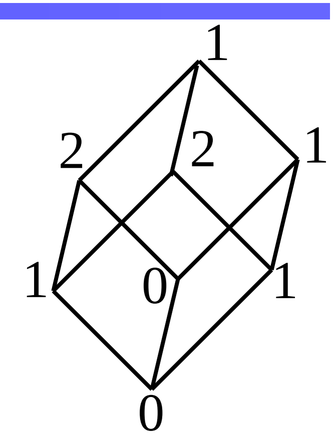
Testing Lipschitz Property on Hypercube

Lower Bound

Lipschitz Property of Functions f: $\{0,1\}^n \to$ ***R***

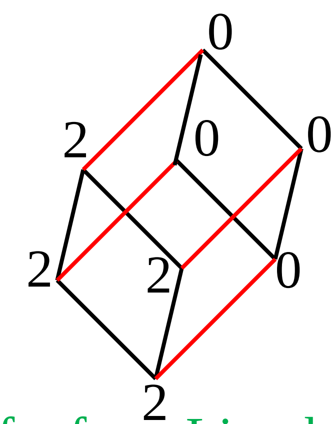
[Jha Raskhodnikova]

- A function $f : \{0,1\}^n \to \mathrm{R}$ is Lipschitz if changing a bit of x changes $f(x)$ by at most 1.



Lipschitz

- Is f Lipschitz or ε-far from Lipschitz (f has to change on many points to become Lipschitz)?
  - Edge $x - y$ is violated by f if $|f(x) - f(y)| > 1$.



$\frac{1}{2}$-far from Lipschitz

Time:

- $O(n^2/\varepsilon)$, logarithmic in the size of the input, 2^n
- $\Omega(n)$

Testing Lipschitz Property

Theorem

Testing Lipschitz property of functions f: $\{0,1\}^n \to \{0,1,2\}$ requires $\Omega(n)$ queries.

Prove it.

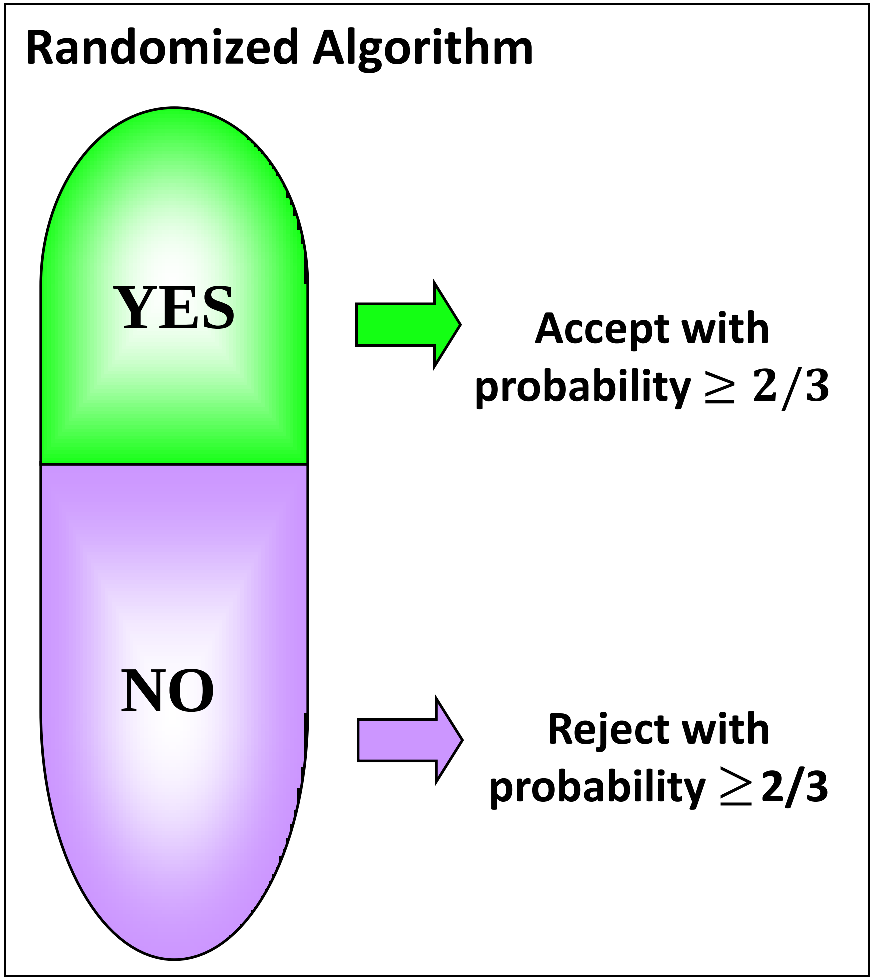
Summary of Lower Bound Methods

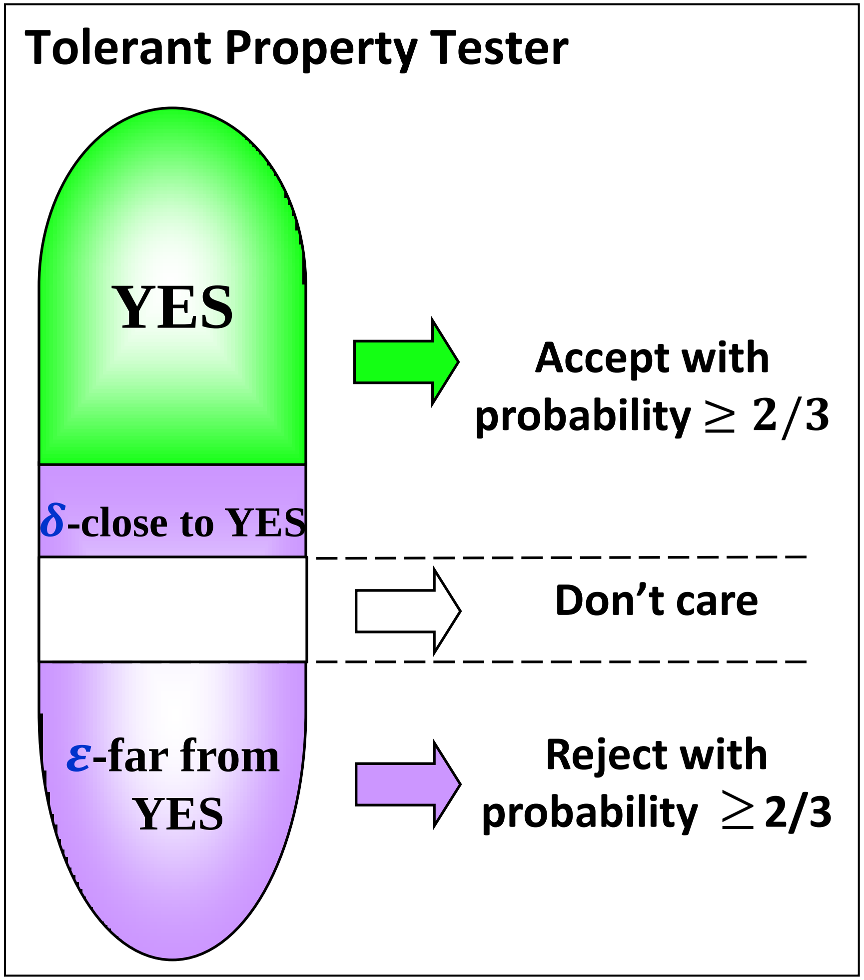
- Yao's Principle
  - testing membership in 1*, sortedness of a list and monotonicity of Boolean functions

- Reductions from communication complexity problems
  - testing if a Boolean function is a k-parity

Other Models of Sublinear Computation

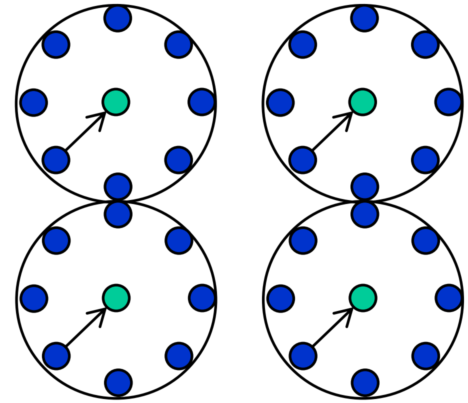
Tolerant Property Tester [Rubinfeld Parnas Ron]





Sublinear-Time "Restoration" Models

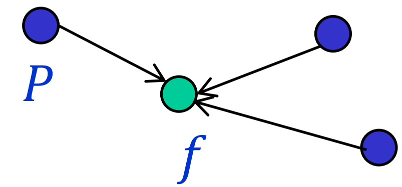
Local Decoding



Input: A slightly corrupted codeword

Requirement: Recover individual bits of the closest codeword with a constant number of queries per recovered bit.

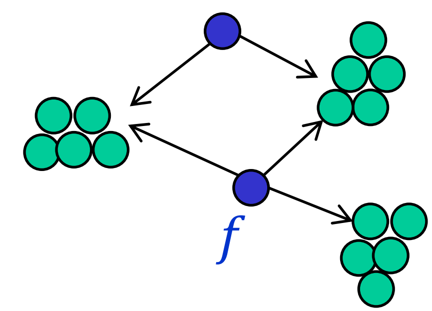
Program Checking



Input: A program P computing f correctly on most inputs.

Requirement: Self-correct program P: for a given input x, compute $f(x)$ by making a few calls to P.

Local Reconstruction



Input: Function f nearly satisfying some property P

Requirement: Reconstruct function f to ensure that the reconstructed function g satisfies P, changing f only when necessary. For each input x, compute $g(x)$ with a few queries to f.

Generalization: Local Computation

[Rubinfeld Tamir Vardi Xie 2011]

- Compute the i-th character y_i of a legal output y.
- If there are several legal outputs for a given input, be consistent with one.
- Example: maximal independent set in a graph.

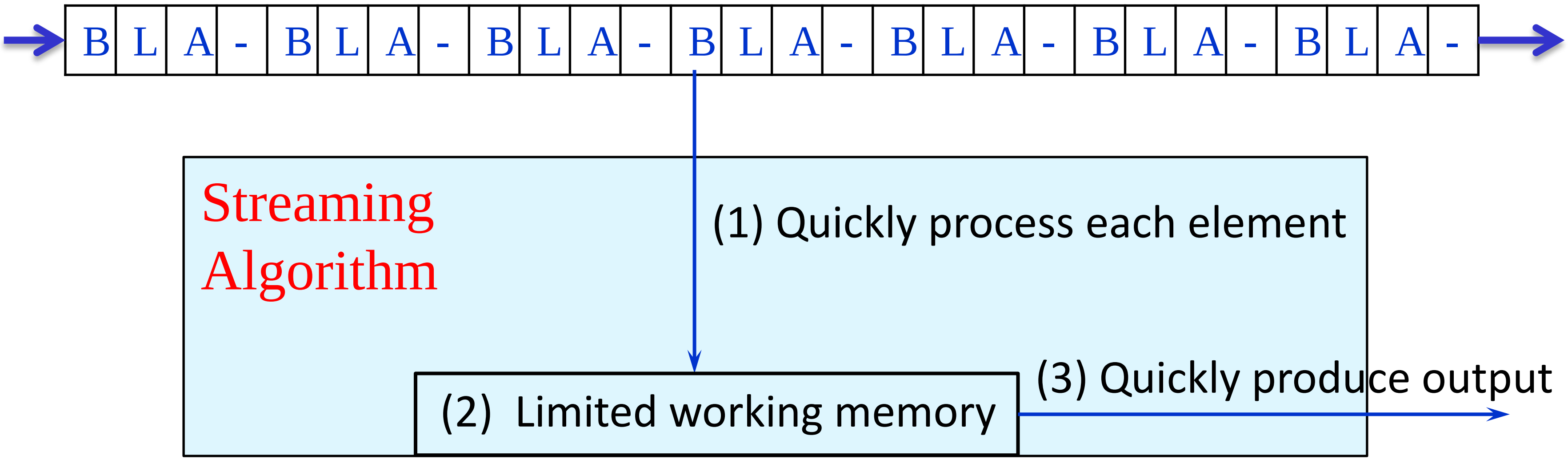
Sublinear-Space Algorithms

What if we cannot get a sublinear-time algorithm?

Can we at least get sublinear space?

Note: sublinear space is broader (for any algorithm, space complexity ≤ time complexity)

Data Stream Model



Motivation: internet traffic analysis

Model the stream as m elements from $[n]$, e.g.,

$$\langle x_1, x_2, \dots, x_m \rangle = 3, 5, 3, 7, 5, 4, \dots$$

Goal: Compute a function of the stream, e.g., median, number of distinct elements, longest increasing sequence.

Streaming Puzzle

A stream contains $n-1$ **distinct** elements from $[n]$ in arbitrary order.

Problem: Find the missing element, using $O(\log n)$ space.

Sampling from a Stream of Unknown Length

Problem: Find a uniform sample s from a stream $\langle x_1, x_2, \ldots, x_m \rangle$ of unknown length m

Algorithm

1. Initially, $s \leftarrow x_1$
2. On seeing the t^{th} element, $s \leftarrow x_t$ with probability $1/t$

Analysis:

What is the probability that $s = x_i$ at some time $t \geq i$?

$$\Pr[s = x_i] = \frac{1}{i} \cdot \left(1 - \frac{1}{i+1}\right) \cdot \ldots \cdot \left(1 - \frac{1}{t}\right)$$
$$= \frac{1}{i} \cdot \frac{i}{i+1} \cdot \ldots \cdot \frac{t-1}{t} = \frac{1}{t}$$

Space: $O(k \log n)$ bits to get k samples.

Conclusion

Sublinear algorithms are possible in many settings

- simple algorithms, more involved analysis
- nice combinatorial problems
- unexpected connections to other areas
- many open questions

In the remainder of the course, we will cover research papers in the area.