Sublinear Algorithms
Lecture 4

Sofya Raskhodnikova
Penn State University

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

Tentative Plan

Lecture 1. Background. Testing properties of images and lists.

Lecture 2. Testing properties of lists. Sublinear-time approximation for graph problems.

Lecture 3. Testing properties of functions. Linearity testing.

Lecture 4. Techniques for proving hardness. Other models for sublinear computation.

Query Complexity

- Query complexity of an algorithm is the maximum number of queries the algorithm makes.
  - Usually expressed as a function of input length (and other parameters)
  - Example: the test for sortedness (from Lecture 2) had query complexity O(log n) for constant ε.
  - running time $\geq$ query complexity
- Query complexity of a problem P, denoted $q(P)$, is the query complexity of the best algorithm for the problem.
  - What is $q(\text{testing sortedness})$? How do we know that there is no better algorithm?

Today: Two techniques for proving lower bounds on $q(P)$.

Yao's Principle

A Method for Proving Lower Bounds

A Lower Bound Game

Players: Evil algorithms designer Al and poor lower bound prover Lola.

Game1

Move 1. Al selects a **randomized** algorithm for the problem.

Move 2. Lola selects an input on which the algorithm is as slow as possible.

Game2

Move 1. Lola selects a distribution on inputs.

Move 2. Al selects a **deterministic** algorithm which works on Lola's distribution as fast as possible.

Yao's Minimax Principle (easy direction): Lola can perform in Game1 at least as well as she can perform in Game2.

A Lower Bound for Testing Sortedness

Input: a list of n numbers $x_1, x_2, \ldots, x_n$

Question: Is the list sorted or ε-far from sorted?

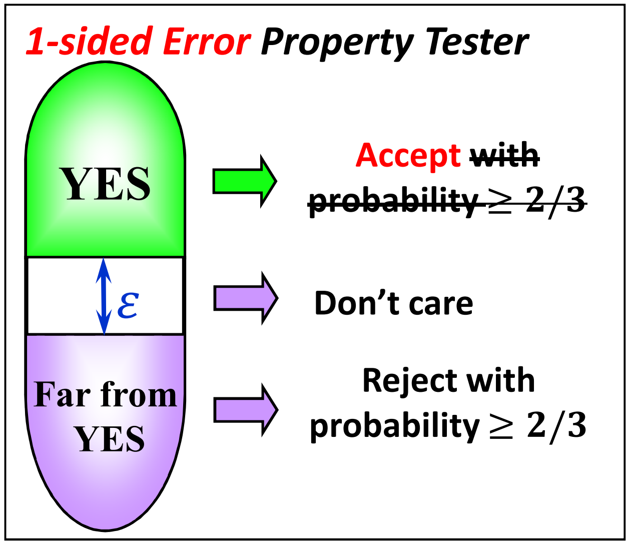
Already saw: two different $O((\log n)/\varepsilon)$ time testers.

Known [Ergün Kannan Kumar Rubinfeld Viswanathan 98, Fischer 01]:

$\Omega(\log n)$ queries are required for all constant $\varepsilon \leq 1/2$

Today: $\Omega(\log n)$ queries are required for all constant $\varepsilon \leq 1/2$
for every 1-sided error nonadaptive test.

- A test has 1-sided error if it always accepts all YES instances.
- A test is nonadaptive if its queries that do not depend on answers to previous queries.



1-Sided Error Tests Must Catch "Mistakes"

- A pair (x_i, x_j) is **violated** if $x_i < x_j$

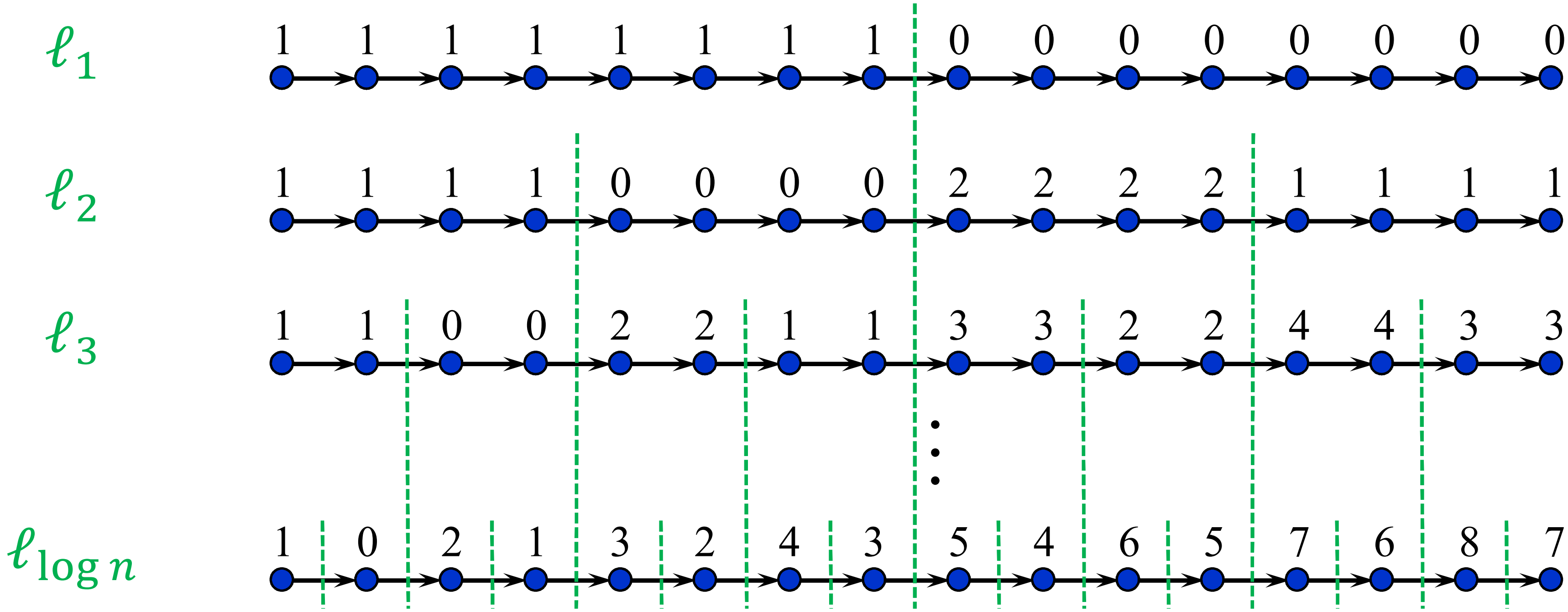
Claim. A 1-sided error test can reject only if it finds a violated pair.

Proof: Every sorted partial list can be extended to a sorted list.

1	?	?	4	…	7	?	?	9

Yao's Principle Game [Jha]

Lola's distribution is uniform over the following $\log n$ lists:

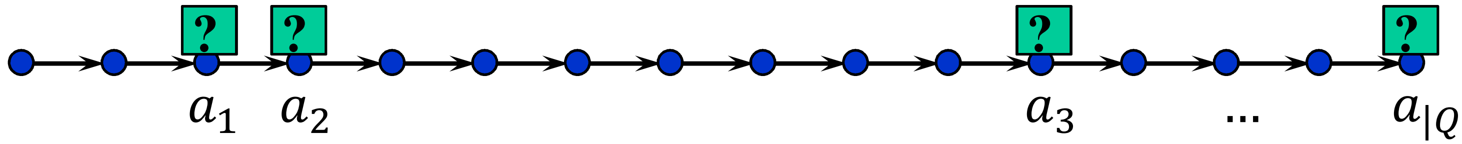


Claim 1. All lists above are 1/2-far from sorted.

Claim 2. Every pair (x_i, x_j) is violated in exactly one list above.

Yao's Principle Game: Al's Move

Al picks a set $Q = \{a_1, a_2, \ldots, a_{|Q|}\}$ of positions to query.



- His test must be correct, i.e., must find a violated pair with probability $\geq 2/3$ when input is picked according to Lola's distribution.
- Q contains a violated pair $\Leftrightarrow$ (a_i, a_{i+1}) is violated for some i

$$\Pr_{\ell \leftarrow \text{Lola's distribution}}[(a_i, a_{i+1}) \text{ for some } i \text{ is vilolated in list } \ell] \leq \frac{|Q|-1}{\log n}$$

By the Union Bound

- If $|Q| \leq \frac{2}{3}\log n$ then this probability is $< \frac{2}{3}$
- So, $|Q| = \Omega(\log n)$
- By Yao's Minimax Principle, every randomized 1-sided error nonadaptive test for sortedness must make $\Omega(\log n)$ queries.

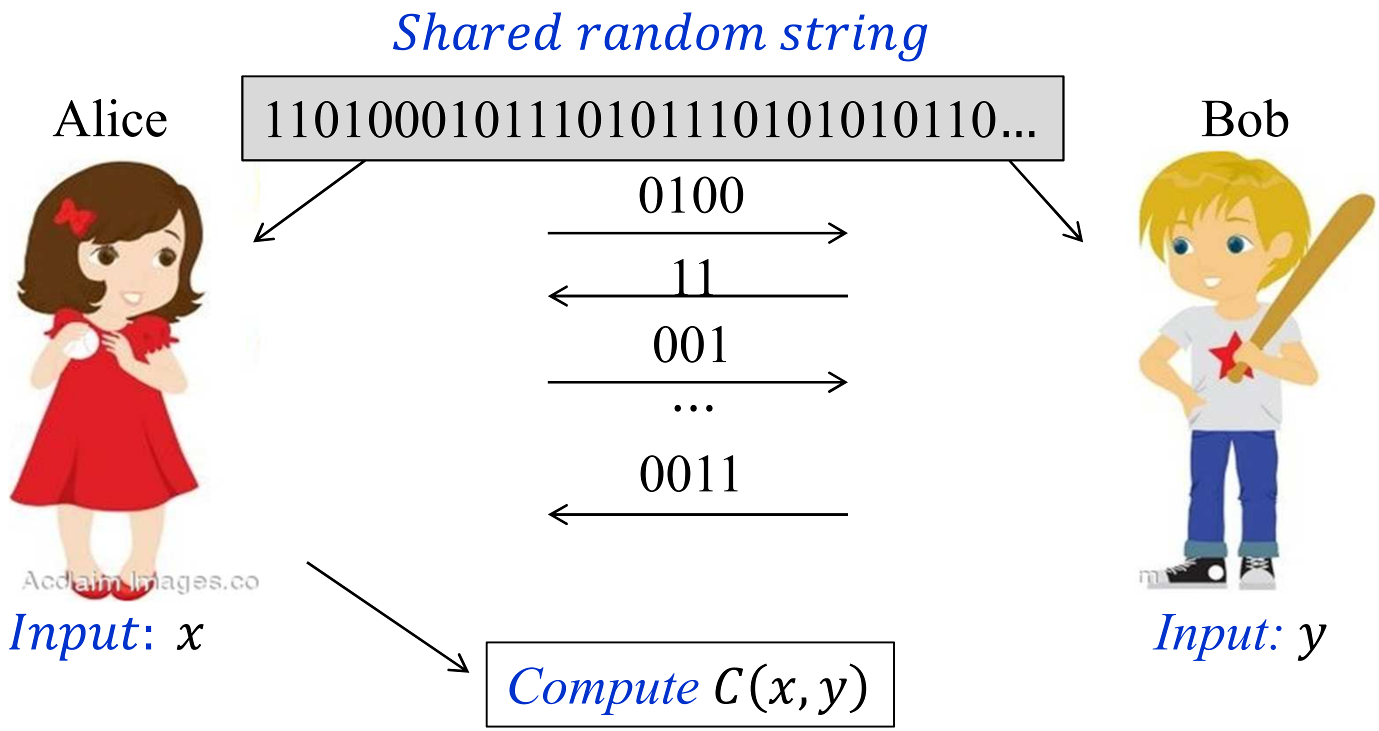
Communication Complexity

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 $DISJ_{\boldsymbol{k}}$

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]

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

k-Parity Functions

Recall: $f : \{0,1\}^n \to \{0,1\}$ is *linear* if $f(x_1, \dots, x_n) = \sum_{i \in S} x_i$ for some $S \subseteq [n]$.

Last time: linearity is testable in $O(1/\varepsilon)$ time.

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

$O(\min(k \log k, (n-k)\log(n-k), n))$ [Chakraborty Garcia−Soriano Matsliah]

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

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

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

Reduction from $DISJ_{k/2}$ 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 $DISJ_{k/2}$ that runs T and has communication complexity $2 \cdot q(\text{testing } k\text{−parity})$.

holds for CC of every protocol for $DISJ_k$

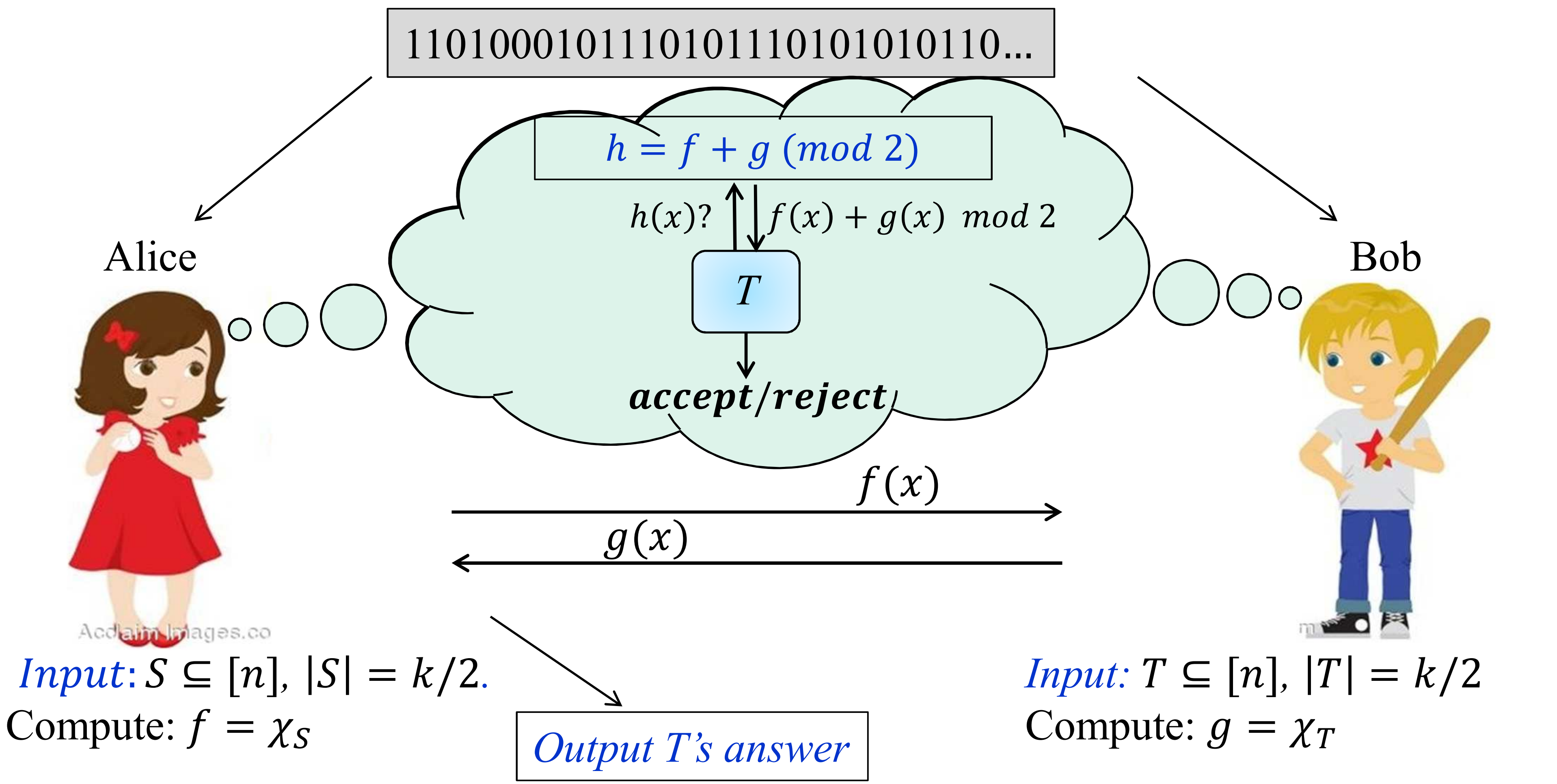
[Hastad Wigderson 07]

- 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



- 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

- Recall that two different linear functions disagree on half of the values:

$\langle \chi_S, \chi_T \rangle = 1 - 2 \cdot$ (fraction of *disagreements* between χ_S and χ_T) $= 0$ for $S \neq T$

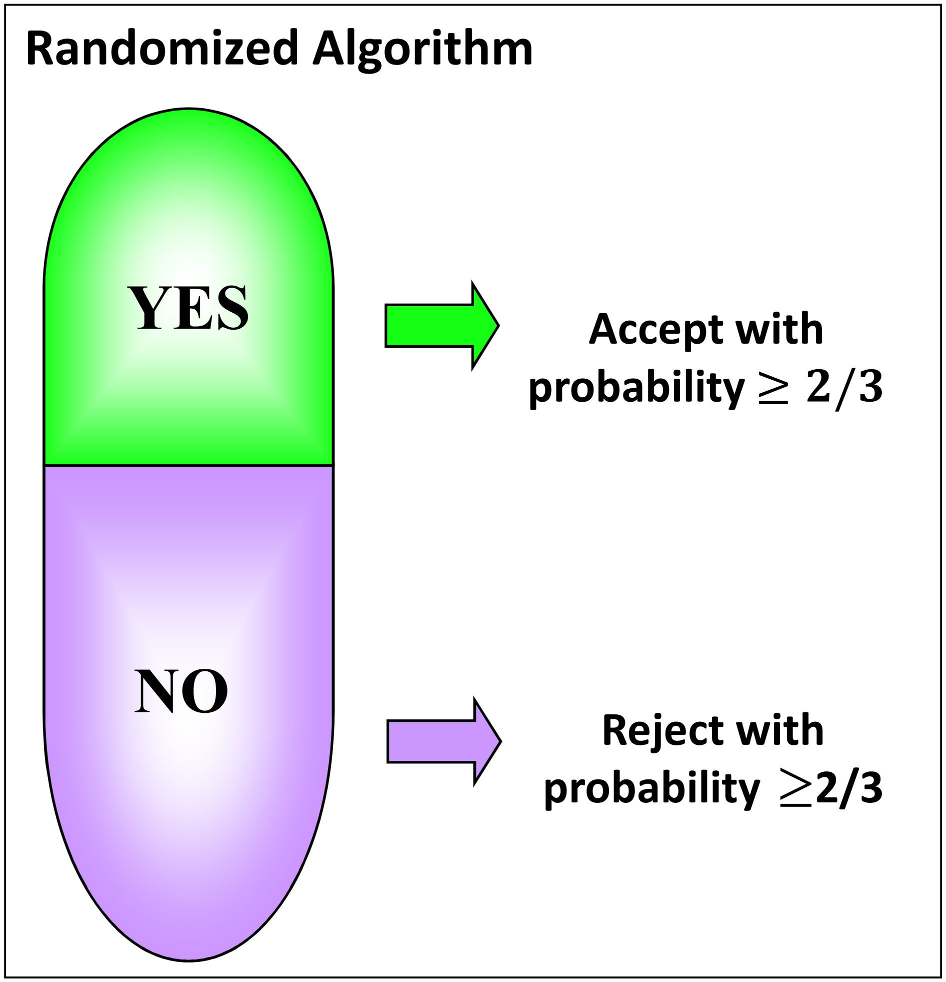
Summary: q(testing k-parity) $\geq \Omega(k)$ for $k \leq n/2$

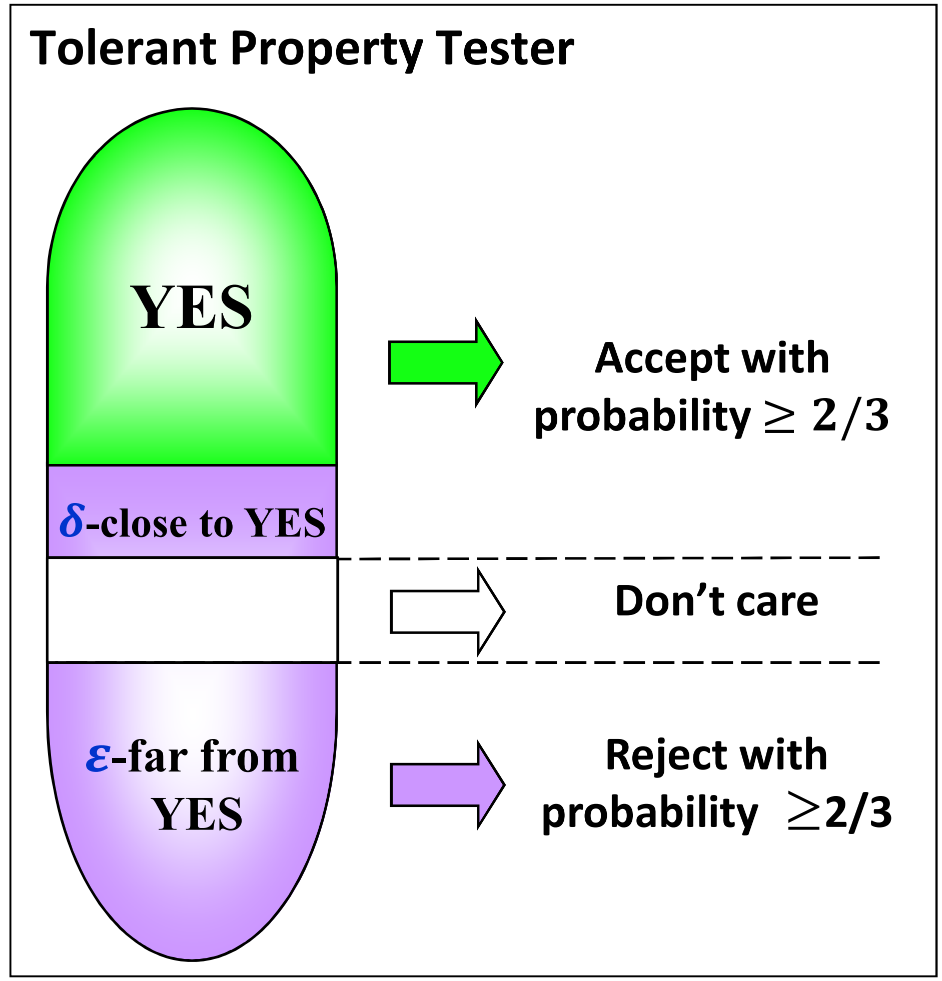
Summary of Lower Bounds

- Yao's Principle
  - testing sortedness
- Reductions from communication complexity problems
  - testing if a function is a k-parity

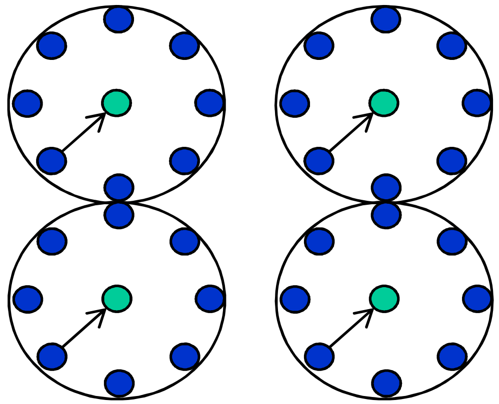
Other Models of Sublinear Computation

Tolerant Property Tester [Rubinfeld Parnas Ron]





Sublinear-Time "Restoration" Models

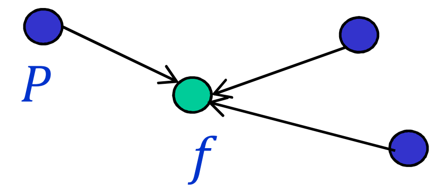


Local Decoding

Input: a slightly corrupted codeword

Requirement: recover a given bit of the closest codeword with a constant number of queries.

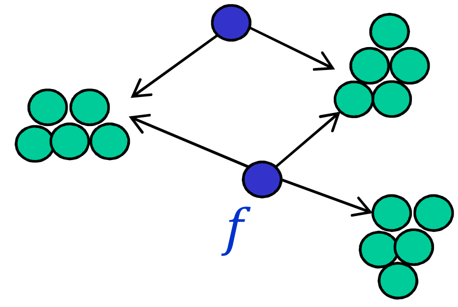
Program Checking



Input: a program P computing f with a small error probability.

Requirement: self-correct program P – for a given argument 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 a given argument x, compute $g(x)$ with a few queries to f.

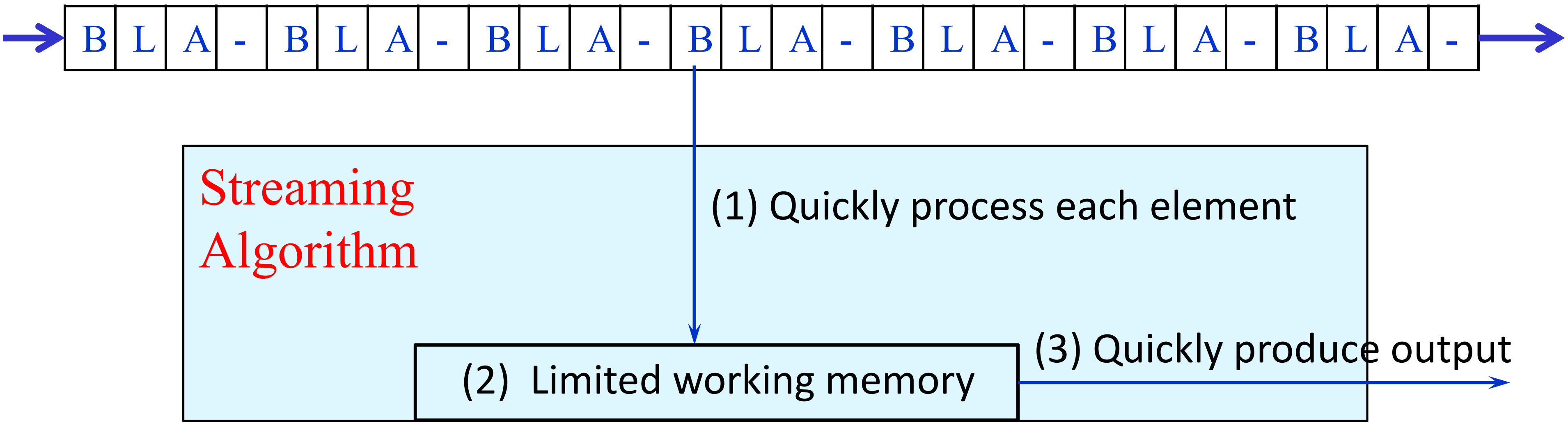
Sublinear-Space Algorithms

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 $\leq$ time complexity)

Data Stream Model



- Motivation: network traffic, database transactions, sensor networks, satellite data feed

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.

Based on Andrew McGregor's slides: http://www.cs.umass.edu/~mcgregor/slides/10-jhu1.pdf

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 $\langle x_1, x_2, \dots, x_m \rangle$ 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 $t \geq i$?

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

$$= \frac{1}{i} \cdot \frac{i}{i+1} \cdot \dots \cdot \frac{t-1}{t} = \frac{1}{t}$$

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