The Communication Complexity of Equality

OUR PROBLEM- $\mathbb{E}Q$

OUR PROBLEM- EQ

1. Alice has x, Bob has y, both of length n.

OUR PROBLEM- EQ

1. Alice has x, Bob has y, both of length n.
2. They want to see if $x = y$ communicating as few bits as possible.

OUR PROBLEM- EQ

1. Alice has x, Bob has y, both of length n.
2. They want to see if $x = y$ communicating as few bits as possible.
3. We call this problem EQ.

OBVIOUS PROTOCOL

OBVIOUS PROTOCOL

1. Alice has $a_1 \cdots a_n$. Bob has $b_1 \cdots b_n$.

OBVIOUS PROTOCOL

1. Alice has $a_1 \cdots a_n$. Bob has $b_1 \cdots b_n$.
2. Alice sends $a_1 \cdots a_n$ to Bob (n bits).

OBVIOUS PROTOCOL

1. Alice has $a_1 \cdots a_n$. Bob has $b_1 \cdots b_n$.
2. Alice sends $a_1 \cdots a_n$ to Bob $\quad\quad (n$ bits).
3. Bob compares $a_1 \cdots a_n$ to $b_1 \cdots b_n$.
   If equal send 1, else send 0. (1 bit.)

OBVIOUS PROTOCOL

1. Alice has $a_1 \cdots a_n$. Bob has $b_1 \cdots b_n$.
2. Alice sends $a_1 \cdots a_n$ to Bob (n bits).
3. Bob compares $a_1 \cdots a_n$ to $b_1 \cdots b_n$.
   If equal send 1, else send 0. (1 bit.)

So EQ can be solved with $n + 1$ bits.

VOTE!

VOTE!

1. EQ requires $\sim n$ bits.

VOTE!

1. EQ requires $\sim \boldsymbol{n}$ bits.
2. Can do EQ with $\sim \sqrt{\boldsymbol{n}}$ bits, but no better.

VOTE!

1. EQ requires $\sim \boldsymbol{n}$ bits.
2. Can do EQ with $\sim \sqrt{\boldsymbol{n}}$ bits, but no better.
3. Can do EQ with $\sim \mathbf{\log n}$ bits, but no better.

VOTE!

1. EQ requires $\sim n$ bits.
2. Can do EQ with $\sim \sqrt{n}$ bits, but no better.
3. Can do EQ with $\sim \log n$ bits, but no better.
4. **UNKNOWN TO BILL!**

BAD NEWS

BAD NEWS

EQ requires $\mathbf{n+1}$ bits.

BAD NEWS

EQ requires $\mathbf{n+1}$ bits.

So, for Alice and Bob to determine if two n-bit strings are equal requires $\mathbf{n+1}$ bits.

BAD NEWS

EQ requires $\mathbf{n+1}$ bits.

So, for Alice and Bob to determine if two n-bit strings are equal requires $\mathbf{n+1}$ bits.

(Proven by Andrew Yao in 1979.)

ALLOW ERROR

What if we

ALLOW ERROR

What if we

1. Allow Alice and Bob to flip coins, and

ALLOW ERROR

What if we

1. Allow Alice and Bob to flip coins, and
2. allow a probability of error $\leq \frac{1}{n}$.

NAIVE IDEA

NAIVE IDEA

1. Alice has $a_1 \cdots a_n$. Bob has $b_1 \cdots b_n$.

NAIVE IDEA

1. Alice has $a_1 \cdots a_n$. Bob has $b_1 \cdots b_n$.
2. Alice picks random $S \subseteq \{1, \ldots, n\}$, $|S| = 10$.

NAIVE IDEA

1. Alice has $a_1 \cdots a_n$. Bob has $b_1 \cdots b_n$.
2. Alice picks random $S \subseteq \{1, \ldots, n\}$, $|S| = 10$.
3. For $i \in S$ Alice sends (i, a_i). $10 \log n$ bits.

NAIVE IDEA

1. Alice has $a_1 \cdots a_n$. Bob has $b_1 \cdots b_n$.
2. Alice picks random $S \subseteq \{1, \ldots, n\}$, $|S| = 10$.
3. For $i \in S$ Alice sends (i, a_i). $10 \log n$ bits.
4. For each (i, a_i) that Bob checks "$a_i = b_i$?".

NAIVE IDEA

1. Alice has $a_1 \cdots a_n$. Bob has $b_1 \cdots b_n$.
2. Alice picks random $S \subseteq \{1, \ldots, n\}$, $|S| = 10$.
3. For $i \in S$ Alice sends (i, a_i). $10 \log n$ bits.
4. For each (i, a_i) that Bob checks "$a_i = b_i$?".
5. If always YES, Bob sends 1, else sends 0.

GOOD AND BAD

GOOD AND BAD

1. Protocol is $\sim \log n$ bits. **GOOD!**

GOOD AND BAD

1. Protocol is $\sim \log n$ bits. **GOOD!**
2. Prob of error $\to 1$ as $n \to \infty$. **BAD!**

GOOD AND BAD

1. Protocol is $\sim \log n$ bits. **GOOD!**
2. Prob of error $\to 1$ as $n \to \infty$. **BAD!**
3. Does well if input is unif chosen. **GOOD!**

GOOD AND BAD

1. Protocol is $\sim \log n$ bits. **GOOD!**
2. Prob of error $\to 1$ as $n \to \infty$. **BAD!**
3. Does well if input is unif chosen. **GOOD!**
4. Not really what we want. **BAD!**

GOOD AND BAD

1. Protocol is $\sim \log n$ bits. **GOOD!**
2. Prob of error $\to 1$ as $n \to \infty$. **BAD!**
3. Does well if input is unif chosen. **GOOD!**
4. Not really what we want. **BAD!**
5. **KEY PROBLEM** Protocol too local.

LESS NAIVE IDEA

LESS NAIVE IDEA

1. Alice has $a_1 \cdots a_n$. Bob has $b_1 \cdots b_n$.

LESS NAIVE IDEA

1. Alice has $a_1 \cdots a_n$. Bob has $b_1 \cdots b_n$.
2. Alice computes $a_1 + \cdots + a_n$.

LESS NAIVE IDEA

1. Alice has $a_1 \cdots a_n$. Bob has $b_1 \cdots b_n$.
2. Alice computes $a_1 + \cdots + a_n$.
   Sends $PAR(a_1 + \cdots + a_n) = 1$ if sum is **Odd**

LESS NAIVE IDEA

1. Alice has $a_1 \cdots a_n$. Bob has $b_1 \cdots b_n$.
2. Alice computes $a_1 + \cdots + a_n$.
   Sends $PAR(a_1 + \cdots + a_n) = 1$ if sum is **Odd**
   Sends $PAR(a_1 + \cdots + a_n) = 0$ if sum is **Even**.

LESS NAIVE IDEA

1. Alice has $a_1 \cdots a_n$. Bob has $b_1 \cdots b_n$.
2. Alice computes $a_1 + \cdots + a_n$.
   Sends $PAR(a_1 + \cdots + a_n) = 1$ if sum is **Odd**
   Sends $PAR(a_1 + \cdots + a_n) = 0$ if sum is **Even**.
3. Bob computes $PAR(b_1 + \cdots + b_n)$.

LESS NAIVE IDEA

1. Alice has $a_1 \cdots a_n$. Bob has $b_1 \cdots b_n$.
2. Alice computes $a_1 + \cdots + a_n$.
   Sends $PAR(a_1 + \cdots + a_n) = 1$ if sum is **Odd**
   Sends $PAR(a_1 + \cdots + a_n) = 0$ if sum is **Even**.
3. Bob computes $PAR(b_1 + \cdots + b_n)$.
   $PAR(a_1 + \cdots + a_n) = PAR(b_1 + \cdots + b_n)$ then 1 $(x = y)$

LESS NAIVE IDEA

1. Alice has $a_1 \cdots a_n$. Bob has $b_1 \cdots b_n$.
2. Alice computes $a_1 + \cdots + a_n$.
   Sends $PAR(a_1 + \cdots + a_n) = 1$ if sum is **Odd**
   Sends $PAR(a_1 + \cdots + a_n) = 0$ if sum is **Even**.
3. Bob computes $PAR(b_1 + \cdots + b_n)$.
   $PAR(a_1 + \cdots + a_n) = PAR(b_1 + \cdots + b_n)$ then 1 $(x = y)$
   $PAR(a_1 + \cdots + a_n) \neq PAR(b_1 + \cdots + b_n)$ then 0 $(x \neq y)$

GOOD AND BAD

GOOD AND BAD

1. Only send ~ 1 bit. **GOOD.**

GOOD AND BAD

1. Only send ~ 1 bit. **GOOD.**
2. Bit used ALL of $a_1 \cdots a_n$. **GOOD.**

GOOD AND BAD

1. Only send ~ 1 bit. **GOOD.**
2. Bit used ALL of $a_1 \cdots a_n$. **GOOD.**
3. Protocol will be wrong alot. **BAD.**

GOOD AND BAD

1. Only send ~ 1 bit. **GOOD.**
2. Bit used ALL of $a_1 \cdots a_n$. **GOOD.**
3. Protocol will be wrong alot. **BAD.**
4. Speculation: Use

GOOD AND BAD

1. Only send ~ 1 bit. **GOOD.**
2. Bit used ALL of $a_1 \cdots a_n$. **GOOD.**
3. Protocol will be wrong alot. **BAD.**
4. Speculation: Use
   $a_n + \cdots + a_1 \pmod 3$

GOOD AND BAD

1. Only send ~ 1 bit. **GOOD.**
2. Bit used ALL of $a_1 \cdots a_n$. **GOOD.**
3. Protocol will be wrong alot. **BAD.**
4. Speculation: Use
   $a_n + \cdots + a_1 \pmod 3$
   $a_n + \cdots + a_1 \pmod p$

GOOD AND BAD

1. Only send ~ 1 bit. **GOOD.**
2. Bit used ALL of $a_1 \cdots a_n$. **GOOD.**
3. Protocol will be wrong alot. **BAD.**
4. Speculation: Use
   $a_n + \cdots + a_1 \pmod 3$
   $a_n + \cdots + a_1 \pmod p$
   more?

GOOD AND BAD

1. Only send ~ 1 bit. **GOOD.**
2. Bit used ALL of $a_1 \cdots a_n$. **GOOD.**
3. Protocol will be wrong alot. **BAD.**
4. Speculation: Use
   $a_n + \cdots + a_1 \pmod 3$
   $a_n + \cdots + a_1 \pmod p$
   more?
   We won't be doing that but we will be using mods.

NEED A THEOREM

NEED A THEOREM

1. If f is a polynomial **over the reals** of degree d then

NEED A THEOREM

1. If f is a polynomial **over the reals** of degree d then f has at most d roots.

NEED A THEOREM

1. If f is a polynomial **over the reals** of degree d then f has at most d roots.
2. If f is a polynomial **over the complex numbers** of degree d

NEED A THEOREM

1. If f is a polynomial **over the reals** of degree d then f has at most d roots.
2. If f is a polynomial **over the complex numbers** of degree d then f has at most d roots.

NEED A THEOREM

1. If f is a polynomial **over the reals** of degree d then f has at most d roots.
2. If f is a polynomial **over the complex numbers** of degree d then f has at most d roots.
3. Let p be a prime. If f is a polynomial **over Z_p** of degree d

NEED A THEOREM

1. If f is a polynomial **over the reals** of degree d then f has at most d roots.
2. If f is a polynomial **over the complex numbers** of degree d then f has at most d roots.
3. Let p be a prime. If f is a polynomial **over Z_p** of degree d then f has at most d roots.

RANDOMIZED PROTOCOL

RANDOMIZED PROTOCOL

1. Alice has $a_0 a_1 \cdots a_{n-1}$. Bob has $b_0 b_1 \cdots b_{n-1}$.

RANDOMIZED PROTOCOL

1. Alice has $a_0 a_1 \cdots a_{n-1}$. Bob has $b_0 b_1 \cdots b_{n-1}$.
2. Alice sends Bob a prime p, $n^2 \leq p \leq 2n^2$.

RANDOMIZED PROTOCOL

1. Alice has $a_0 a_1 \cdots a_{n-1}$. Bob has $b_0 b_1 \cdots b_{n-1}$.
2. Alice sends Bob a prime p, $n^2 \le p \le 2n^2$.
3. Alice picks $z \in \{1, \ldots, p-1\}$ **Randomly.**
   Alice computes, mod p,

RANDOMIZED PROTOCOL

1. Alice has $a_0 a_1 \cdots a_{n-1}$. Bob has $b_0 b_1 \cdots b_{n-1}$.
2. Alice sends Bob a prime p, $n^2 \le p \le 2n^2$.
3. Alice picks $z \in \{1, \ldots, p-1\}$ **Randomly.**
   Alice computes, mod p,
   $y = a_0 + a_1 z + a_2 z^2 + \cdots + a_{n-1} z^{n-1}$

RANDOMIZED PROTOCOL

1. Alice has $a_0 a_1 \cdots a_{n-1}$. Bob has $b_0 b_1 \cdots b_{n-1}$.
2. Alice sends Bob a prime p, $n^2 \le p \le 2n^2$.
3. Alice picks $z \in \{1, \ldots, p-1\}$ **Randomly.**
   Alice computes, mod p,
   $y = a_0 + a_1 z + a_2 z^2 + \cdots + a_{n-1} z^{n-1}$
   Alice sends (z, y) to Bob.

RANDOMIZED PROTOCOL

1. Alice has $a_0 a_1 \cdots a_{n-1}$. Bob has $b_0 b_1 \cdots b_{n-1}$.
2. Alice sends Bob a prime p, $n^2 \le p \le 2n^2$.
3. Alice picks $z \in \{1, \ldots, p-1\}$ **Randomly.**
   Alice computes, mod p,
   $y = a_0 + a_1 z + a_2 z^2 + \cdots + a_{n-1} z^{n-1}$
   Alice sends (z, y) to Bob.
4. Bob computes, mod p,

RANDOMIZED PROTOCOL

1. Alice has $a_0a_1\cdots a_{n-1}$. Bob has $b_0b_1\cdots b_{n-1}$.
2. Alice sends Bob a prime p, $n^2 \le p \le 2n^2$.
3. Alice picks $z \in \{1,\ldots,p-1\}$ **Randomly.**
   Alice computes, mod p,
   $y = a_0 + a_1z + a_2z^2 + \cdots + a_{n-1}z^{n-1}$
   Alice sends (z, y) to Bob.
4. Bob computes, mod p,
   $y' = b_0 + b_1z + b_2z^2 + \cdots + b_{n-1}z^{n-1}$

RANDOMIZED PROTOCOL

1. Alice has $a_0 a_1 \cdots a_{n-1}$. Bob has $b_0 b_1 \cdots b_{n-1}$.
2. Alice sends Bob a prime p, $n^2 \le p \le 2n^2$.
3. Alice picks $z \in \{1, \ldots, p-1\}$ **Randomly.**
   Alice computes, mod p,
   $y = a_0 + a_1 z + a_2 z^2 + \cdots + a_{n-1} z^{n-1}$
   Alice sends (z, y) to Bob.
4. Bob computes, mod p,
   $y' = b_0 + b_1 z + b_2 z^2 + \cdots + b_{n-1} z^{n-1}$
   If $y = y'$ then send 1, else send 0.

GOOD!

GOOD!

1. Protocol exchanges $\sim \log n$ bits.

GOOD!

1. Protocol exchanges $\sim \log n$ bits.
2. Prob of error is $\leq \frac{1}{n}$.

GOOD!

1. Protocol exchanges $\sim \log n$ bits.
2. Prob of error is $\leq \frac{1}{n}$.
   WHY

GOOD!

1. Protocol exchanges $\sim \log n$ bits.
2. Prob of error is $\leq \frac{1}{n}$.
   WHY
   If there is an error then z is a root of the poly $a(x) - b(x)$

GOOD!

1. Protocol exchanges $\sim \log n$ bits.
2. Prob of error is $\leq \frac{1}{n}$.
   WHY
   If there is an error then z is a root of the poly $a(x) - b(x)$
   There are only n such roots so the probability of this is very low:

GOOD!

1. Protocol exchanges $\sim \log n$ bits.
2. Prob of error is $\leq \frac{1}{n}$.
   WHY
   If there is an error then z is a root of the poly $a(x) - b(x)$
   There are only n such roots so the probability of this is very low:
   There are n roots and there are $p \geq n^2$ elements to pick from.

GOOD!

1. Protocol exchanges $\sim \log n$ bits.
2. Prob of error is $\leq \frac{1}{n}$.
   WHY
   If there is an error then z is a root of the poly $a(x) - b(x)$
   There are only n such roots so the probability of this is very low:
   There are n roots and there are $p \geq n^2$ elements to pick from.
   Prob of getting a root is $\leq \frac{n}{n^2} = \frac{1}{n}$.

GOOD!

1. Protocol exchanges $\sim \log n$ bits.
2. Prob of error is $\leq \frac{1}{n}$.
   WHY
   If there is an error then z is a root of the poly $a(x) - b(x)$
   There are only n such roots so the probability of this is very low:
   There are n roots and there are $p \geq n^2$ elements to pick from.
   Prob of getting a root is $\leq \frac{n}{n^2} = \frac{1}{n}$.
3. This protocol is due to Melhorn and Schmidt, 1982.

FOR MORE INFORMATION

COMMUNICATION COMPLEXITY
by Kushilevitz and Nisan.

FOR MORE INFORMATION

COMMUNICATION COMPLEXITY
by Kushilevitz and Nisan.
COMMUNICATION COMPLEXITY AND APPLICATIONS
by Rao and Yehudayoff.