# Algorithms for Big Data (IX)

Chihao Zhang

Shanghai Jiao Tong University

Nov. 15, 2019

# Review

# Review

Last week, we saw the communication model.

# Review

Last week, we saw the communication model.

In this model, Alice and Bob collaborate to compute some function $f(x, y)$.

# Review

Last week, we saw the communication model.

In this model, Alice and Bob collaborate to compute some function $f(x, y)$.

Alice has $x$ and Bob $y$, the complexity is measured by bits communicated between them.

# Review

Last week, we saw the communication model.

In this model, Alice and Bob collaborate to compute some function $f(x, y)$.

Alice has $x$ and Bob $y$, the complexity is measured by bits communicated between them.

We focus on the special one-way communication model.

# Randomness

# Randomness

We allow randomness in our communication protocol.

# Randomness

We allow randomness in our communication protocol.

In this course, we consider protocols using public coins.

# Randomness

We allow randomness in our communication protocol.

In this course, we consider protocols using public coins.

In this model, we assume there exits some random source in the environment so that both Alice and Bob can see it.

# Randomness

We allow randomness in our communication protocol.

In this course, we consider protocols using public coins.

In this model, we assume there exits some random source in the environment so that both Alice and Bob can see it.

Our lower bound applies to protocols using public coins, and hence also applies to ones using private coins.

# Problems

# Problems

For $\mathbf{x} \in \{0,1\}^n$, sometimes we view it as the indicator vector of some subset of $[n]$, $S(x) = \{i \in [n] \, : \, x_i = 1\}$.

# Problems

For $\mathbf{x} \in \{0,1\}^n$, sometimes we view it as the indicator vector of some subset of $[n]$,
$S(x) = \{i \in [n] : x_i = 1\}$.

$\mathrm{EQ}(\mathbf{x}, \mathbf{y}) \triangleq \mathbf{1}[\mathbf{x} = \mathbf{y}]$

# Problems

For $\mathbf{x} \in \{0,1\}^n$, sometimes we view it as the indicator vector of some subset of $[n]$, $S(x) = \{i \in [n] \,:\, x_i = 1\}$.

$\mathrm{EQ}(\mathbf{x}, \mathbf{y}) \triangleq \mathbf{1}[\mathbf{x} = \mathbf{y}]$

- We showed by a counting argument that any deterministic protocol to compute EQ requires at needs $n$ bits of one-way communication.

# Problems

For $\mathbf{x} \in \{0,1\}^n$, sometimes we view it as the indicator vector of some subset of $[n]$, $S(x) = \{i \in [n] : x_i = 1\}$.

$\mathrm{EQ}(\mathbf{x}, \mathbf{y}) \triangleq \mathbf{1}[\mathbf{x} = \mathbf{y}]$

- We showed by a counting argument that any deterministic protocol to compute EQ requires at needs $n$ bits of one-way communication.
- There exits a random protocol using $O(\log n)$ bits of communication.

# Problems

For $\mathbf{x} \in \{0,1\}^n$, sometimes we view it as the indicator vector of some subset of $[n]$, $S(x) = \{i \in [n] : x_i = 1\}$.

$\mathrm{EQ}(\mathbf{x}, \mathbf{y}) \triangleq \mathbf{1}[\mathbf{x} = \mathbf{y}]$

- We showed by a counting argument that any deterministic protocol to compute EQ requires at needs $n$ bits of one-way communication.
- There exits a random protocol using $O(\log n)$ bits of communication.

$\mathrm{DISJ}(\mathbf{x}, \mathbf{y}) \triangleq \mathbf{1}[S(x) \cap S(y) = \varnothing]$.

# Problems

For $\mathbf{x} \in \{0,1\}^n$, sometimes we view it as the indicator vector of some subset of $[n]$, $S(x) = \{i \in [n] : x_i = 1\}$.

$\mathrm{EQ}(\mathbf{x}, \mathbf{y}) \triangleq \mathbf{1}[\mathbf{x} = \mathbf{y}]$

- We showed by a counting argument that any deterministic protocol to compute EQ requires at needs $n$ bits of one-way communication.
- There exits a random protocol using $O(\log n)$ bits of communication.

$\mathrm{DISJ}(\mathbf{x}, \mathbf{y}) \triangleq \mathbf{1}[S(x) \cap S(y) = \varnothing]$.

- We showed that any randomized protocol requires $\Omega(\log n)$ bits of communication.

# Yao's Lemma

# Yao's Lemma

The main tool to prove lower bounds for randomized protocol is Yao's lemma.

**Lemma**

If there exists some distribution $\mathcal{D}$ over $\{0,1\}^a \times \{0,1\}^b$ such that any deterministic one-way communication protocol $P$ with

$$\mathbf{Pr}_{(x,y)\sim\mathcal{D}}\left[P \text{ is wrong on } (x,y)\right] \le \varepsilon$$

costs at least $k$ bits, then any randomized one-way protocol with error at most $\varepsilon$ on any input also costs at least $k$ bits one-way communication.

# Yao's Lemma

The main tool to prove lower bounds for randomized protocol is Yao's lemma.

**Lemma**

If there exists some distribution $\mathcal{D}$ over $\{0,1\}^a \times \{0,1\}^b$ such that any deterministic one-way communication protocol $P$ with

$$\mathbf{Pr}_{(x,y)\sim\mathcal{D}}\left[P \text{ is wrong on } (x,y)\right] \le \varepsilon$$

costs at least $k$ bits, then any randomized one-way protocol with error at most $\varepsilon$ on any input also costs at least $k$ bits one-way communication.

We remark that "costs at least $k$ bits" of a randomized protocol applies to the worst input with worst random bits.

# Proof of Yao's Lemma

# Proof of Yao's Lemma

Let $Q$ be a randomized protocol costs less than $k$ bits in the worst case.

# Proof of Yao's Lemma

Let $Q$ be a randomized protocol costs less than $k$ bits in the worst case.

We can first toss all coins and then run the protocol based on the results. Therefore, we can view $Q$ as a distribution over $t$ deterministic protocols $P_1, \ldots, P_t$.

# Proof of Yao's Lemma

Let $Q$ be a randomized protocol costs less than $k$ bits in the worst case.

We can first toss all coins and then run the protocol based on the results. Therefore, we can view $Q$ as a distribution over $t$ deterministic protocols $P_1, \ldots, P_t$.

By the definition of the costs of a random protocol, each $P_i$ costs less than $k$ bits.

# Proof of Yao's Lemma

Let $Q$ be a randomized protocol costs less than $k$ bits in the worst case.

We can first toss all coins and then run the protocol based on the results. Therefore, we can view $Q$ as a distribution over $t$ deterministic protocols $P_1, \dots, P_t$.

By the definition of the costs of a random protocol, each $P_i$ costs less than $k$ bits.

Thus, by the assumption, there exists a distribution $\mathcal{D}$ such that

$$\forall i \in [t], \quad \mathbf{Pr}_{(x,y)\sim\mathcal{D}}\left[P_i \text{ is wrong on } (x,y)\right] > \varepsilon.$$

# Proof of Yao's Lemma

Let $Q$ be a randomized protocol costs less than $k$ bits in the worst case.

We can first toss all coins and then run the protocol based on the results. Therefore, we can view $Q$ as a distribution over $t$ deterministic protocols $P_1, \ldots, P_t$.

By the definition of the costs of a random protocol, each $P_i$ costs less than $k$ bits.

Thus, by the assumption, there exists a distribution $\mathcal{D}$ such that

$$\forall i \in [t], \quad \mathbf{Pr}_{(x,y)\sim\mathcal{D}}\left[P_i \text{ is wrong on } (x,y)\right] > \varepsilon.$$

This is sufficient to imply that for some $(x, y)$,

$$\mathbf{Pr}\left[Q \text{ is wrong on} (x,y)\right] > \varepsilon.$$

## Proof of Yao's Lemma

Let $Q$ be a randomized protocol costs less than $k$ bits in the worst case.

We can first toss all coins and then run the protocol based on the results. Therefore, we can view $Q$ as a distribution over $t$ deterministic protocols $P_1, \ldots, P_t$.

By the definition of the costs of a random protocol, each $P_i$ costs less than $k$ bits.

Thus, by the assumption, there exists a distribution $\mathcal{D}$ such that

$$\forall i \in [t], \quad \mathbf{Pr}_{(x,y)\sim\mathcal{D}}\left[P_i \text{ is wrong on } (x,y)\right] > \varepsilon.$$

This is sufficient to imply that for some $(x, y)$,

$$\mathbf{Pr}\left[Q \text{ is wrong on} (x,y)\right] > \varepsilon.$$

We remark that the converse of Yao's lemma is also correct.

# Reduction

# Reduction

We prove lower bounds for streaming problem from communication complexity via reductions.

# Reduction

We prove lower bounds for streaming problem from communication complexity via reductions.

The argument looks like: suppose we have an efficient streaming algorithm for problem $A$, then we can design an efficient protocol to solve communication problem $B$.

# Reduction

We prove lower bounds for streaming problem from communication complexity via reductions.

The argument looks like: suppose we have an efficient streaming algorithm for problem $A$, then we can design an efficient protocol to solve communication problem $B$.

For example, we can

- View the input $\mathbf{x}$ (and $\mathbf{y}$) as a stream of elements in $S(x)$ (and $S(y)$).
- Alice solves the streaming problem on $S(x)$ and send the snapshot of the current memory to Bob.
- Bob continues to solve the streaming problem on $S(x) \oplus S(y)$.

# Reduction

We prove lower bounds for streaming problem from communication complexity via reductions.

The argument looks like: suppose we have an efficient streaming algorithm for problem $A$, then we can design an efficient protocol to solve communication problem $B$.

For example, we can

- View the input $\mathbf{x}$ (and $\mathbf{y}$) as a stream of elements in $S(x)$ (and $S(y)$).
- Alice solves the streaming problem on $S(x)$ and send the snapshot of the current memory to Bob.
- Bob continues to solve the streaming problem on $S(x) \oplus S(y)$.

We successfully prove a lower bound for computing $F_\infty$ using above strategy, via the complexity of DISJ.

# Lower Bounds for $st$-connectivity

# Lower Bounds for $st$-connectivity

We are given a graph $G = (V, E)$ where $V = \{v_1, \ldots, v_n\}$.

# Lower Bounds for $st$-connectivity

We are given a graph $G = (V, E)$ where $V = \{v_1, \ldots, v_n\}$.

Determine whether $s = v_1$ and $t = v_2$ are connected.

# Lower Bounds for $st$-connectivity

We are given a graph $G = (V, E)$ where $V = \{v_1, \dots, v_n\}$.

Determine whether $s = v_1$ and $t = v_2$ are connected.

We now prove that and deterministic algorithm requires $\Omega(n)$ bits of communication.

# Lower Bounds for $st$-connectivity

We are given a graph $G = (V, E)$ where $V = \{v_1, \ldots, v_n\}$.

Determine whether $s = v_1$ and $t = v_2$ are connected.

We now prove that and deterministic algorithm requires $\Omega(n)$ bits of communication.

Given an instance $(x, y)$ of DISJ, Alice forms her edge stream $\{\{s, v_i\} : x_i = 1 \wedge i > 2\}$ and Bob forms his edge stream $\{\{t, v_j\} : y_j = 1 \wedge j > 2\}$.

# Lower Bounds for $st$-connectivity

We are given a graph $G = (V, E)$ where $V = \{v_1, \ldots, v_n\}$.

Determine whether $s = v_1$ and $t = v_2$ are connected.

We now prove that and deterministic algorithm requires $\Omega(n)$ bits of communication.

Given an instance $(x, y)$ of DISJ, Alice forms her edge stream $\{\{s, v_i\} : x_i = 1 \wedge i > 2\}$ and Bob forms his edge stream $\{\{t, v_j\} : y_j = 1 \wedge j > 2\}$.

Alice first sends $x_1$ and $x_2$ to Bob. If these two bits already determines $\mathrm{DISJ}(x, y)$, then output so.

# Lower Bounds for $st$-connectivity

We are given a graph $G = (V, E)$ where $V = \{v_1, \dots, v_n\}$.

Determine whether $s = v_1$ and $t = v_2$ are connected.

We now prove that and deterministic algorithm requires $\Omega(n)$ bits of communication.

Given an instance $(x, y)$ of DISJ, Alice forms her edge stream $\{\{s, v_i\} : x_i = 1 \wedge i > 2\}$ and Bob forms his edge stream $\{\{t, v_j\} : y_j = 1 \wedge j > 2\}$.

Alice first sends $x_1$ and $x_2$ to Bob. If these two bits already determines $\mathrm{DISJ}(x, y)$, then output so.

Otherwise $\mathrm{DISJ}(\mathbf{x}, \mathbf{y}) = 1$ iff $s$ and $t$ are not connected.

# Lower Bounds for $F_0$

# Lower Bounds for $F_0$

Recall that in BJKST algorithm, we can estimate $F_0$ within $(1 \pm \varepsilon)$ using $O(\frac{1}{\varepsilon^2} \log n)$ bits of memory.

# Lower Bounds for $F_0$

Recall that in BJKST algorithm, we can estimate $F_0$ within $(1 \pm \varepsilon)$ using $O(\frac{1}{\varepsilon^2} \log n)$ bits of memory.

We now show that the dependency of $\varepsilon$, $\frac{1}{\varepsilon^2}$, is tight.

# Lower Bounds for $F_0$

Recall that in BJKST algorithm, we can estimate $F_0$ within $(1 \pm \varepsilon)$ using $O(\frac{1}{\varepsilon^2} \log n)$ bits of memory.

We now show that the dependency of $\varepsilon$, $\frac{1}{\varepsilon^2}$, is tight.

We will show in class that when $\varepsilon = \frac{1}{\sqrt{n}}$, the protocol to approximate $F_0$ within $(1 \pm \varepsilon)$ requires at least $\Omega(n)$ bits.

# Lower Bounds for $F_0$

Recall that in BJKST algorithm, we can estimate $F_0$ within $(1 \pm \varepsilon)$ using $O(\frac{1}{\varepsilon^2} \log n)$ bits of memory.

We now show that the dependency of $\varepsilon$, $\frac{1}{\varepsilon^2}$, is tight.

We will show in class that when $\varepsilon = \frac{1}{\sqrt{n}}$, the protocol to approximate $F_0$ within $(1 \pm \varepsilon)$ requires at least $\Omega(n)$ bits.

Can we prove this using the strategy we demonstrated in the previous example via a reduction from DISJ?

We shall reduce from the problem of computing the Hamming distance between two strings.

We shall reduce from the problem of computing the Hamming distance between two strings.

Alice and Bob want to compute $d_H(\mathbf{x}, \mathbf{y})$, which is exactly $|S(\mathbf{x}) \setminus S(\mathbf{y})| + |S(\mathbf{y}) \setminus S(\mathbf{x})|$.

We shall reduce from the problem of computing the Hamming distance between two strings.

Alice and Bob want to compute $d_H(\mathbf{x}, \mathbf{y})$, which is exactly $|S(\mathbf{x}) \setminus S(\mathbf{y})| + |S(\mathbf{y}) \setminus S(\mathbf{x})|$.

Since $F_0$ of $S(\mathbf{x}) \oplus S(\mathbf{y})$ is $|S(\mathbf{x}) \cup S(\mathbf{y})|$, we have

$$|S(\mathbf{x}) \setminus S(\mathbf{y})| = F_0 - |S(\mathbf{y})|, \quad |S(\mathbf{y}) \setminus S(\mathbf{x})| = F_0 - |S(\mathbf{y})|.$$

We shall reduce from the problem of computing the Hamming distance between two strings.

Alice and Bob want to compute $d_H(\mathbf{x}, \mathbf{y})$, which is exactly $|S(\mathbf{x}) \setminus S(\mathbf{y})| + |S(\mathbf{y}) \setminus S(\mathbf{x})|$.

Since $F_0$ of $S(\mathbf{x}) \oplus S(\mathbf{y})$ is $|S(\mathbf{x}) \cup S(\mathbf{y})|$, we have

$$|S(\mathbf{x}) \setminus S(\mathbf{y})| = F_0 - |S(\mathbf{y})|, \quad |S(\mathbf{y}) \setminus S(\mathbf{x})| = F_0 - |S(\mathbf{y})|.$$

This implies that $d_H(\mathbf{x}, \mathbf{y}) = 2F_0 - |S(\mathbf{x})| - |S(\mathbf{y})|$.

We shall reduce from the problem of computing the Hamming distance between two strings.

Alice and Bob want to compute $d_H(\mathbf{x}, \mathbf{y})$, which is exactly $|S(\mathbf{x}) \setminus S(\mathbf{y})| + |S(\mathbf{y}) \setminus S(\mathbf{x})|$.

Since $F_0$ of $S(\mathbf{x}) \oplus S(\mathbf{y})$ is $|S(\mathbf{x}) \cup S(\mathbf{y})|$, we have

$$|S(\mathbf{x}) \setminus S(\mathbf{y})| = F_0 - |S(\mathbf{y})|, \quad |S(\mathbf{y}) \setminus S(\mathbf{x})| = F_0 - |S(\mathbf{y})|.$$

This implies that $d_H(\mathbf{x}, \mathbf{y}) = 2F_0 - |S(\mathbf{x})| - |S(\mathbf{y})|$.

A multiplicative error of $(1 \pm \frac{1}{\sqrt{n}})$ to $F_0$ provides at most $O(\sqrt{n})$ additive error of $d_H(\mathbf{x}, \mathbf{y})$.

We shall reduce from the problem of computing the Hamming distance between two strings.

Alice and Bob want to compute $d_H(\mathbf{x},\mathbf{y})$, which is exactly $|S(\mathbf{x}) \setminus S(\mathbf{y})| + |S(\mathbf{y}) \setminus S(\mathbf{x})|$.

Since $F_0$ of $S(\mathbf{x}) \oplus S(\mathbf{y})$ is $|S(\mathbf{x}) \cup S(\mathbf{y})|$, we have

$$|S(\mathbf{x}) \setminus S(\mathbf{y})| = F_0 - |S(\mathbf{y})|, \quad |S(\mathbf{y}) \setminus S(\mathbf{x})| = F_0 - |S(\mathbf{y})|.$$

This implies that $d_H(\mathbf{x},\mathbf{y}) = 2F_0 - |S(\mathbf{x})| - |S(\mathbf{y})|$.

A multiplicative error of $(1 \pm \frac{1}{\sqrt{n}})$ to $F_0$ provides at most $O(\sqrt{n})$ additive error of $d_H(\mathbf{x},\mathbf{y})$.

We only need to prove that $O(\sqrt{n})$ additive error of $d_H(\mathbf{x},\mathbf{y})$ is hard.

# Gap-Hamming

# Gap-Hamming

We call the problem Gap-Hamming.

# Gap-Hamming

We call the problem Gap-Hamming.

$$\text{Gap-Ham}_c(x, y) = \begin{cases} 1 & \text{if } d_H(x, y) \le \frac{n}{2} - c\sqrt{n} \\ 0 & \text{if } d_H(x, y) \ge \frac{n}{2} + c\sqrt{n} \\ \text{undefined} & \text{otherwise.} \end{cases}$$

# Gap-Hamming

We call the problem Gap-Hamming.

$$\text{Gap-Ham}_c(x, y) = \begin{cases} 1 & \text{if } d_H(x, y) \leq \frac{n}{2} - c\sqrt{n} \\ 0 & \text{if } d_H(x, y) \geq \frac{n}{2} + c\sqrt{n} \\ \text{undefined} & \text{otherwise.} \end{cases}$$

We will show that solving Gap-Hamming needs $\Omega(n)$ bits of one-way communication even if randomness is permitted.

We will reduce from INDEX. Recall that we assume Alice holds a string $\mathbf{x} \in \{0, 1\}^n$ and Bob holds an index $i \in [n]$.

We will reduce from INDEX. Recall that we assume Alice holds a string $\mathbf{x} \in \{0, 1\}^n$ and Bob holds an index $i \in [n]$.

$$\mathrm{INDEX}(\mathbf{x}, i) = x_i.$$

We will reduce from INDEX. Recall that we assume Alice holds a string $\mathbf{x} \in \{0, 1\}^n$ and Bob holds an index $i \in [n]$.

$$\mathrm{INDEX}(\mathbf{x}, i) = x_i.$$

Alice and Bob generate an instance $(\mathbf{x}', \mathbf{y}')$ of Gap-Hamming without any communication, and then try to deduce the value of $x_i$ from it...

We will reduce from INDEX. Recall that we assume Alice holds a string $\mathbf{x} \in \{0, 1\}^n$ and Bob holds an index $i \in [n]$.

$$\mathrm{INDEX}(\mathbf{x}, i) = x_i.$$

Alice and Bob generate an instance $(\mathbf{x}', \mathbf{y}')$ of Gap-Hamming without any communication, and then try to deduce the value of $x_i$ from it...

This seems to be impossible...

We will reduce from INDEX. Recall that we assume Alice holds a string $\mathbf{x} \in \{0,1\}^n$ and Bob holds an index $i \in [n]$.

$$\mathrm{INDEX}(\mathbf{x}, i) = x_i.$$

Alice and Bob generate an instance $(\mathbf{x}', \mathbf{y}')$ of Gap-Hamming without any communication, and then try to deduce the value of $x_i$ from it...

This seems to be impossible... Let us see the power of randomness.

We will reduce from INDEX. Recall that we assume Alice holds a string $\mathbf{x} \in \{0,1\}^n$ and Bob holds an index $i \in [n]$.

$$\mathrm{INDEX}(\mathbf{x}, i) = x_i.$$

Alice and Bob generate an instance $(\mathbf{x}', \mathbf{y}')$ of Gap-Hamming without any communication, and then try to deduce the value of $x_i$ from it...

This seems to be impossible... Let us see the power of randomness.

We generate $(\mathbf{x}', \mathbf{y}')$ bit by bit.

The idea is that, for each $j$, we want to generate $x'_j, y'_j$ in a way such that the event $x'_j = y'_j$ is correlated to the value of $x_i$.

The idea is that, for each $j$, we want to generate $x'_j, y'_j$ in a way such that the event $x'_j = y'_j$ is correlated to the value of $x_i$.

Assume the public random string is $\mathbf{r}$ (of length $n$), which can be seen by both Alice and Bob.

The idea is that, for each $j$, we want to generate $x'_j, y'_j$ in a way such that the event $x'_j = y'_j$ is correlated to the value of $x_i$.

Assume the public random string is $\mathbf{r}$ (of length $n$), which can be seen by both Alice and Bob.

Recall that $i$ is the index that Bob holds. He always generate $r_i$ as the current bit of $\mathbf{y}'$ (denoted by $b$).

The idea is that, for each $j$, we want to generate $x'_j, y'_j$ in a way such that the event $x'_j = y'_j$ is correlated to the value of $x_i$.

Assume the public random string is $\mathbf{r}$ (of length $n$), which can be seen by both Alice and Bob.

Recall that $i$ is the index that Bob holds. He always generate $r_i$ as the current bit of $\mathbf{y}'$ (denoted by $b$).

Alice generates $1$ if $d_H(\mathbf{x}, \mathbf{r}) < \frac{n}{2}$, generates $0$ if $d_H(\mathbf{x}, \mathbf{r}) > \frac{n}{2}$ (denoted by $a$).

The idea is that, for each $j$, we want to generate $x'_j, y'_j$ in a way such that the event $x'_j = y'_j$ is correlated to the value of $x_i$.

Assume the public random string is $\mathbf{r}$ (of length $n$), which can be seen by both Alice and Bob.

Recall that $i$ is the index that Bob holds. He always generate $r_i$ as the current bit of $\mathbf{y}'$ (denoted by $b$).

Alice generates 1 if $d_H(\mathbf{x}, \mathbf{r}) < \frac{n}{2}$, generates 0 if $d_H(\mathbf{x}, \mathbf{r}) > \frac{n}{2}$ (denoted by $a$).

The key observation is that:

- if $x_i = 1$, then $a = b$ is more likely to happen;
- if $x_i = 0$, then $a \neq b$ is more likely to happen.

We assume $n$ is odd.

We assume $n$ is odd.

Assuming Alice generates $a$ and Bob generates $b$.

We assume $n$ is odd.

Assuming Alice generates $a$ and Bob generates $b$.

Conditional on the random bits $\mathbf{r}_i$, we define two events $\mathcal{E}_1$ and $\mathcal{E}_2$.

We assume $n$ is odd.

Assuming Alice generates $a$ and Bob generates $b$.

Conditional on the random bits $\mathbf{r}_i$, we define two events $\mathcal{E}_1$ and $\mathcal{E}_2$.

Event $\mathcal{E}_1$: the value of $a$ has already been determined by $\mathbf{r}_{-i}$, then $\mathbf{Pr}\left[a = b \mid \mathcal{E}_1\right] = \frac{1}{2}$.

We assume $n$ is odd.

Assuming Alice generates $a$ and Bob generates $b$.

Conditional on the random bits $\mathbf{r}_i$, we define two events $\mathcal{E}_1$ and $\mathcal{E}_2$.

Event $\mathcal{E}_1$: the value of $a$ has already been determined by $\mathbf{r}_{-i}$, then $\mathbf{Pr}\left[a = b \mid \mathcal{E}_1\right] = \frac{1}{2}$.

Event $\mathcal{E}_2$: $d_H(\mathbf{x}_{-i}, \mathbf{r}_{-i}) = \frac{n}{2}$, so the value of $a$ is determined by whether $r_i = x_i$.

We assume $n$ is odd.

Assuming Alice generates $a$ and Bob generates $b$.

Conditional on the random bits $\mathbf{r}_i$, we define two events $\mathcal{E}_1$ and $\mathcal{E}_2$.

Event $\mathcal{E}_1$: the value of $a$ has already been determined by $\mathbf{r}_{-i}$, then $\mathbf{Pr}\left[a = b \mid \mathcal{E}_1\right] = \frac{1}{2}$.

Event $\mathcal{E}_2$: $d_H(\mathbf{x}_{-i}, \mathbf{r}_{-i}) = \frac{n}{2}$, so the value of $a$ is determined by whether $r_i = x_i$.

Conditional on $\mathcal{E}_2$, if $x_i = 1$, then $a = b$; if $x_i = 0$, then $a \neq b$.

We assume $n$ is odd.

Assuming Alice generates $a$ and Bob generates $b$.

Conditional on the random bits $\mathbf{r}_i$, we define two events $\mathcal{E}_1$ and $\mathcal{E}_2$.

Event $\mathcal{E}_1$: the value of $a$ has already been determined by $\mathbf{r}_{-i}$, then $\mathbf{Pr}\left[a = b \mid \mathcal{E}_1\right] = \frac{1}{2}$.

Event $\mathcal{E}_2$: $d_H(\mathbf{x}_{-i}, \mathbf{r}_{-i}) = \frac{n}{2}$, so the value of $a$ is determined by whether $r_i = x_i$.

Conditional on $\mathcal{E}_2$, if $x_i = 1$, then $a = b$; if $x_i = 0$, then $a \neq b$.

$$\mathbf{Pr}\left[a = b\right] = \mathbf{Pr}\left[\mathcal{E}_1\right] \cdot \mathbf{Pr}\left[a = b \mid \mathcal{E}_1\right] + \mathbf{Pr}\left[\mathcal{E}_2\right] \cdot \mathbf{Pr}\left[a = b \mid \mathcal{E}_2\right] = \begin{cases} \frac{1}{2} + \mathbf{Pr}\left[\mathcal{E}_2\right] & \text{if } x_i = 1 \\ \frac{1}{2} - \mathbf{Pr}\left[\mathcal{E}_2\right] & \text{if } x_i = 0 \end{cases}$$

The probability of $\mathcal{E}_2$ is $\binom{n-1}{(n-1)/2}2^{1-n} = \frac{c}{\sqrt{n}}$ for some constant $c$ by the Stirling formula ($n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$).

The probability of $\mathcal{E}_2$ is $\binom{n-1}{(n-1)/2}2^{1-n} = \frac{c}{\sqrt{n}}$ for some constant $c$ by the Stirling formula ($n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$).

Therefore, we have

$$\mathbf{Pr}\left[a = b\right] = \begin{cases} \frac{1}{2} + \frac{c}{\sqrt{n}} & \text{if } x_i = 1 \\ \frac{1}{2} - \frac{c}{\sqrt{n}} & \text{if } x_i = 0 \end{cases}$$

The probability of $\mathcal{E}_2$ is $\binom{n-1}{(n-1)/2}2^{1-n} = \frac{c}{\sqrt{n}}$ for some constant $c$ by the Stirling formula ($n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$).

Therefore, we have

$$\mathbf{Pr}\left[a = b\right] = \begin{cases} \frac{1}{2} + \frac{c}{\sqrt{n}} & \text{if } x_i = 1 \\ \frac{1}{2} - \frac{c}{\sqrt{n}} & \text{if } x_i = 0 \end{cases}$$

Using the Chernoff bound, we can generate $\Theta(n)$ bits and use a protocol for Gap-Hamming to solve INDEX.